

CSE 390B, Spring 2023

Building Academic Success Through Bottom-Up Computing

# Growth Mindset & Sequential Logic

Growth vs. Fixed Mindset, Introduction to Sequential Logic,  
Representing Time in Hardware, The Data Flip-Flop (DFF)

# Lecture Outline

- ❖ **Growth vs. Fixed Mindset**
  - **Setting SMART Goals**
  
- ❖ Introduction to Sequential Logic
  - The Problem of Combinational Logic
  - Autopilot Control Circuit Example
  
- ❖ Representing Time in Hardware
  - Clock Signals and Units of Time in Hardware
  
- ❖ The Data Flip-Flop (DFF)
  - Implementation and Examples

# Growth vs. Fixed Mindset



FIXED



GROWTH

MINDSETS

# Setting SMART Goals

- ❖ **S** – Be specific, simple and significant.
- ❖ **M** – Make sure your goals are measurable. How many times within a week, month, the quarter do you want to do x goal?
- ❖ **A** – Make sure your goals are achievable. Is your goal within your scope of control?
- ❖ **R** – Be realistic and reasonable.
- ❖ **T** – Be time-bound. When will you accomplish your goal?

# SMART Goals Group Discussion

SPRING QUARTER  
GOALS

What are skills, practices  
or habits that are not  
strengths YET?

SPHERE OF CONTROL

Getting a 4.0 in a course

vs.

Attending course  
office hours

SMART GOAL  
FRAMEWORK

**S** — Specific

**M** — Measurable

**A** — Achievable

**R** — Realistic

**T** — Timebound

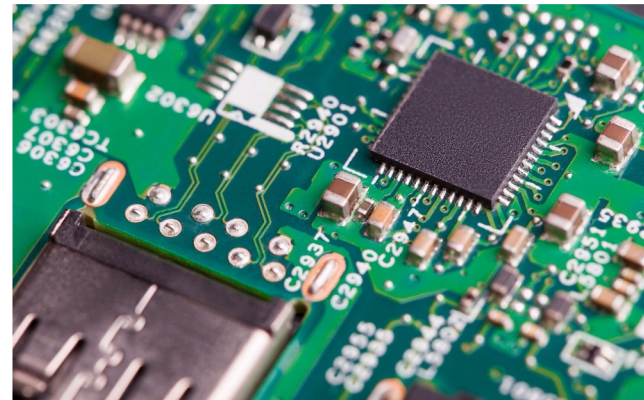
Attending CSE 390B  
office hours at least  
5x this quarter  
(or once every other week)

# Lecture Outline

- ❖ Growth vs. Fixed Mindset
  - Setting SMART Goals
- ❖ **Introduction to Sequential Logic**
  - **The Problem of Combinational Logic**
  - **Autopilot Control Circuit Example**
- ❖ Representing Time in Hardware
  - Clock Signals and Units of Time in Hardware
- ❖ The Data Flip-Flop (DFF)
  - Implementation and Examples

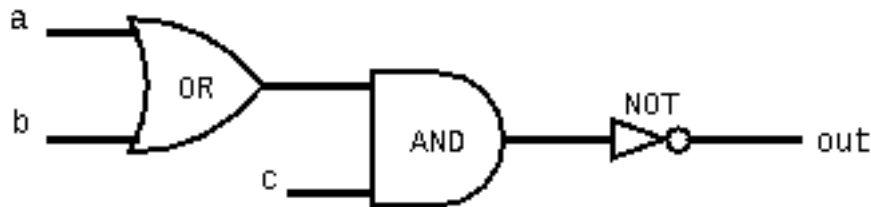
# Why Consider Time Now?

- ❖ Needed for our **abstraction**
  - We need to talk about hardware maintaining state for memory
  - We need vocabulary to talk about time
  
- ❖ Needed for our **implementation**
  - Physical implementations of chips cannot be instantaneous
  - We need to account for physical delays in signal propagation



# The Problem with Combinational Logic

- ❖ Consider the following circuit:

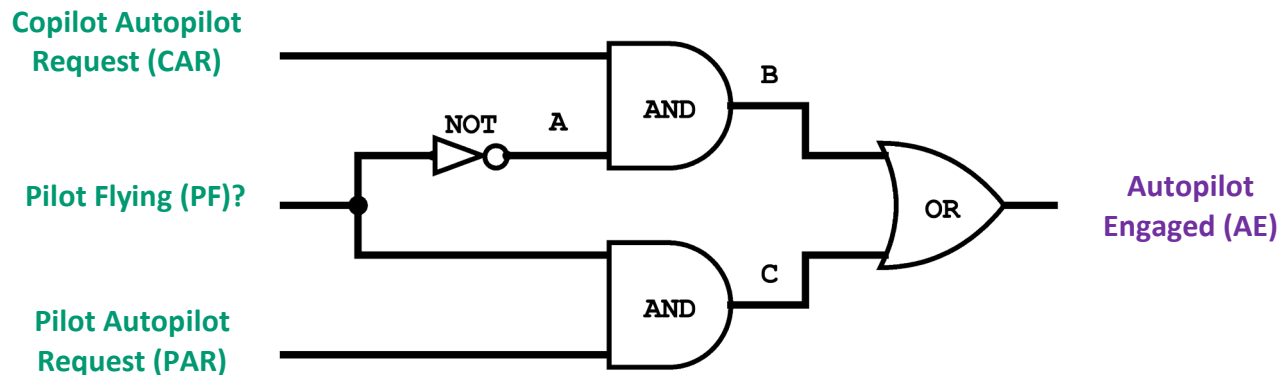


- ❖ Let's assume that  $a=0$ ,  $b=1$ , and  $c=0$ 
  - The output should be 1
- ❖ What's the result if we change  $b=0$  and  $c=1$ ?
  - The result should still be 1
  - However, `out` is briefly 0 if we change `c` first



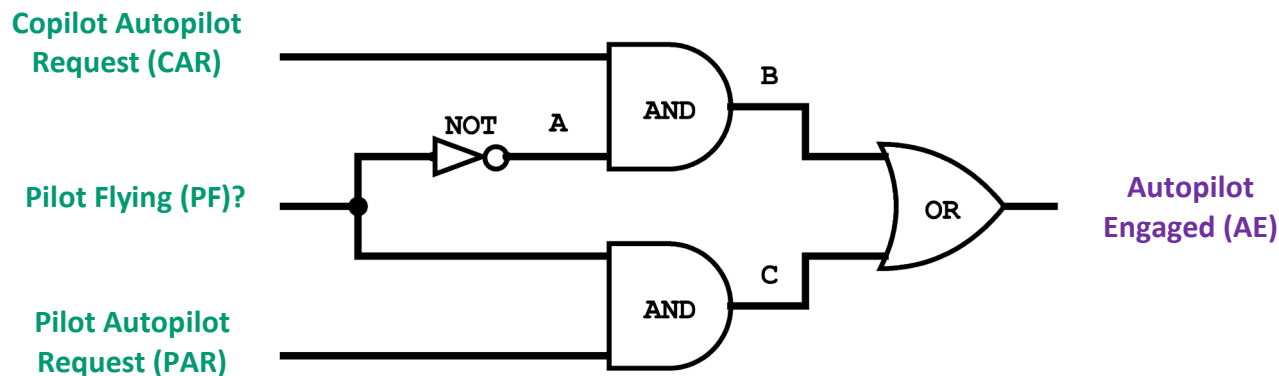
# Autopilot Control Circuit Example

- ❖ Consider this autopilot control circuit:
  - Either the pilot or copilot is flying at any time
  - The pilot and copilot can separately request autopilot
  - Only the person flying can request autopilot



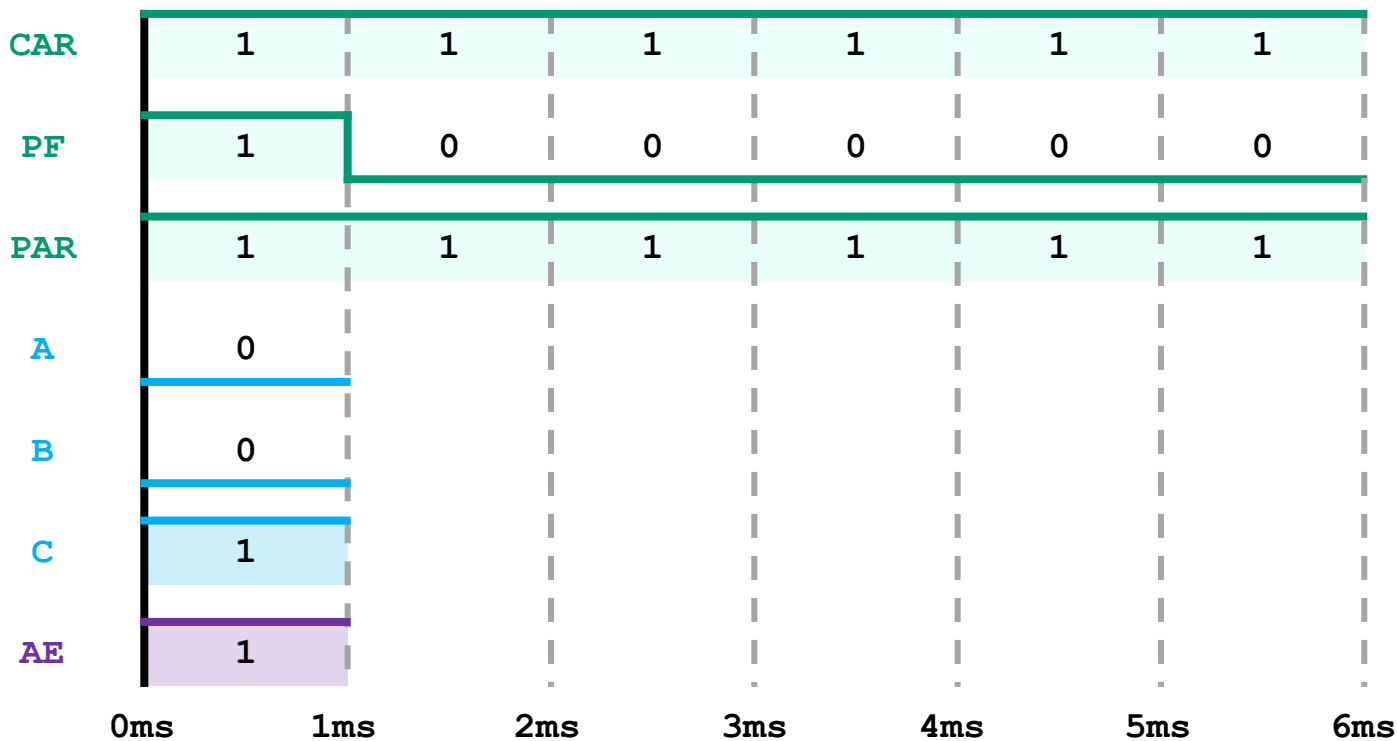
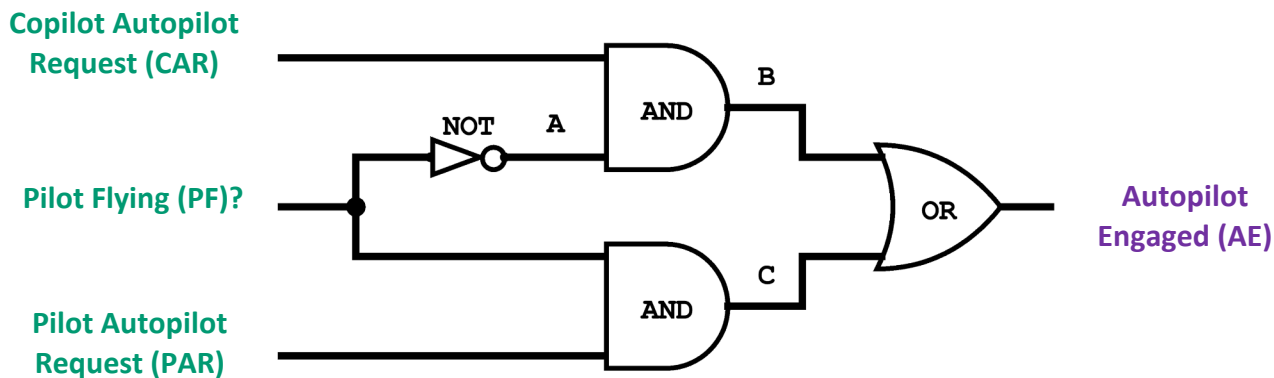
# Autopilot Control Circuit Example

- ❖ Consider this autopilot control circuit:
  - Either the pilot or copilot is flying at any time
  - The pilot and copilot can separately request autopilot
  - Only the person flying can request autopilot



- ❖ Let's assume every logic gate takes  $1\text{ms}$  to compute
  - For example, if an input changes at  $t=4\text{ms}$ , the gate will only output the new result at  $t=5\text{ms}$

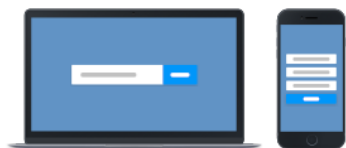
# Autopilot Control Circuit Example



< Lecture 5: Growth Mindset & Sequential Logic



**In just a few words, what would you expect the AE output to be look like?**



**1** Go to **PollEv.com**

**2** Enter **CSE390B**

**1** Text **CSE390B** to **22333**

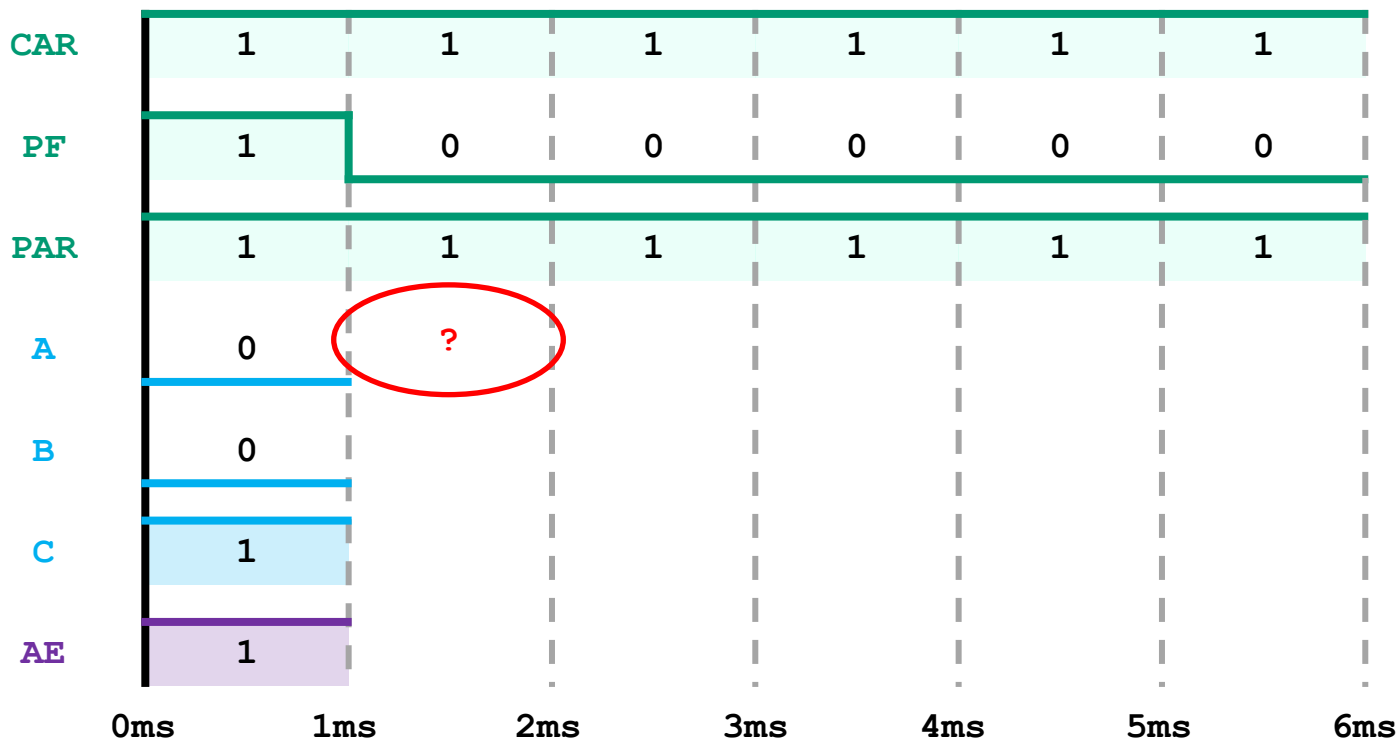
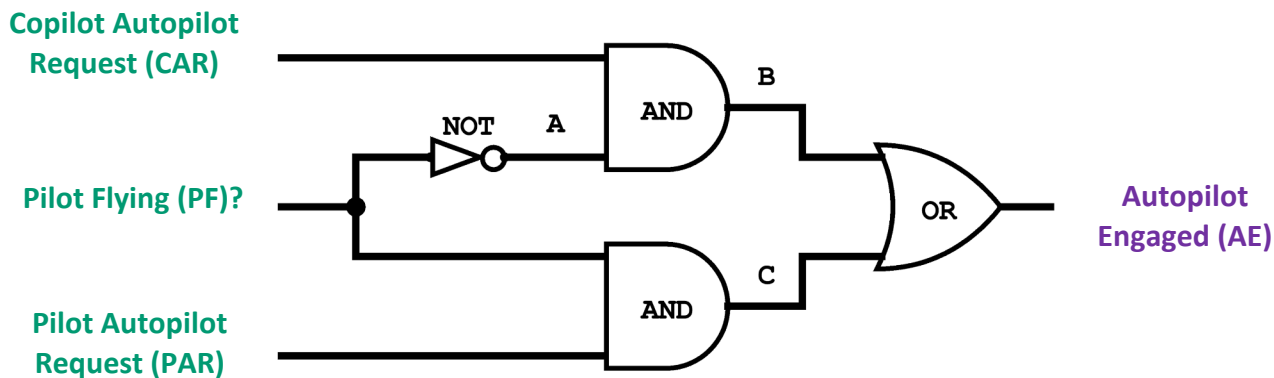
**2** Text in your message

Total Results: 0

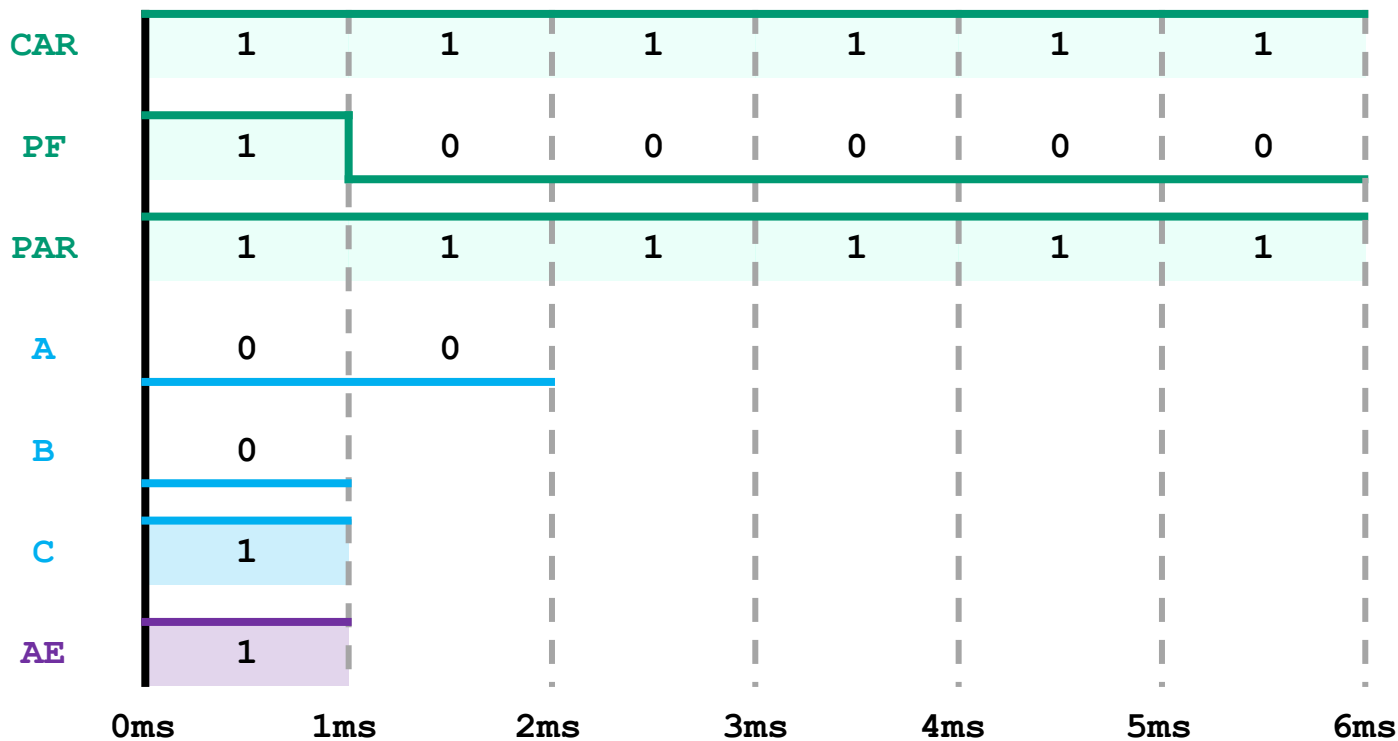
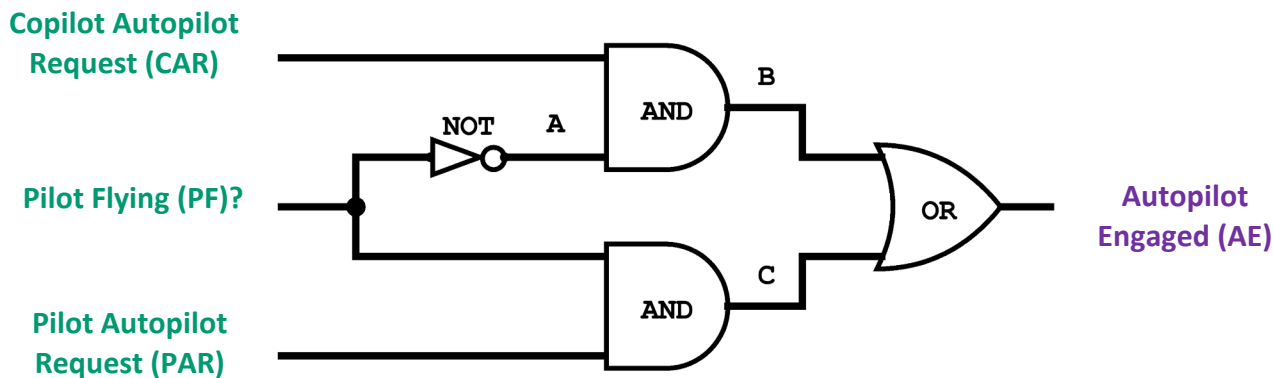
Powered by  **Poll Everywhere**



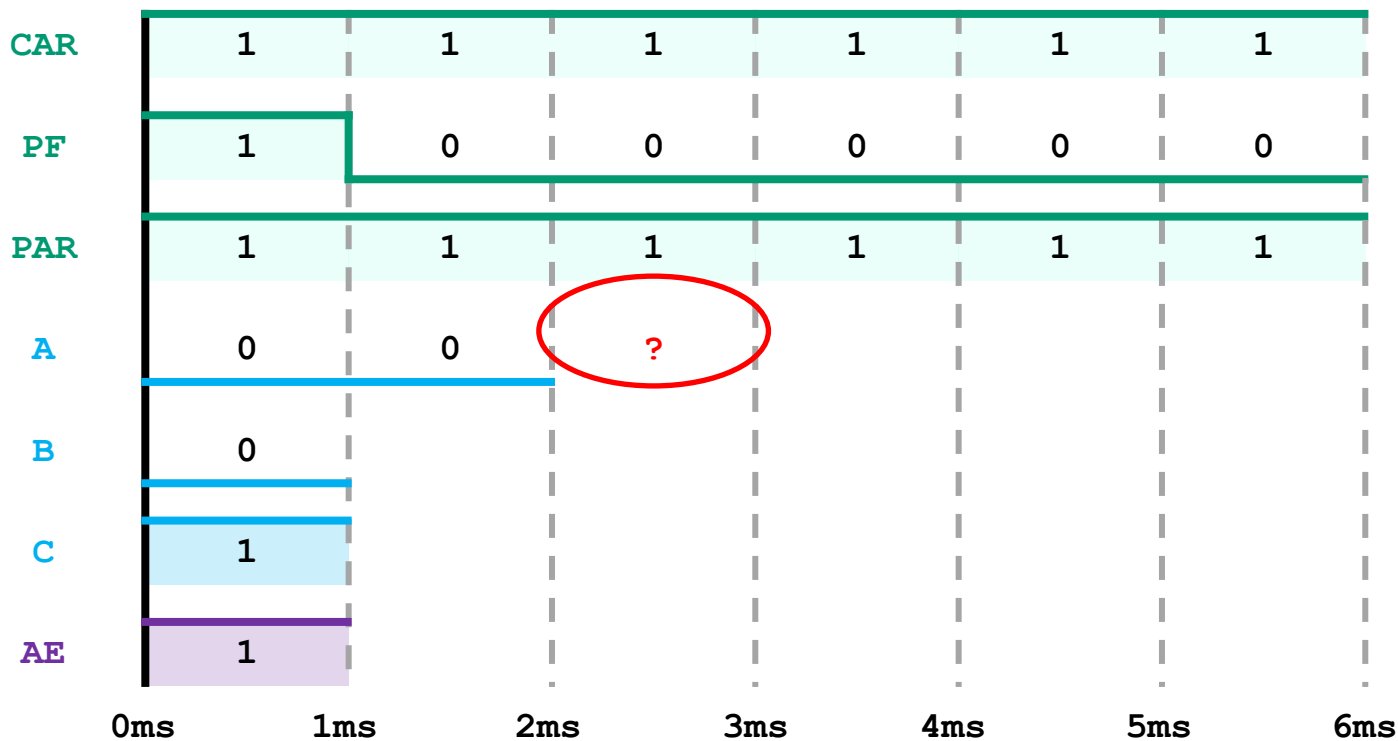
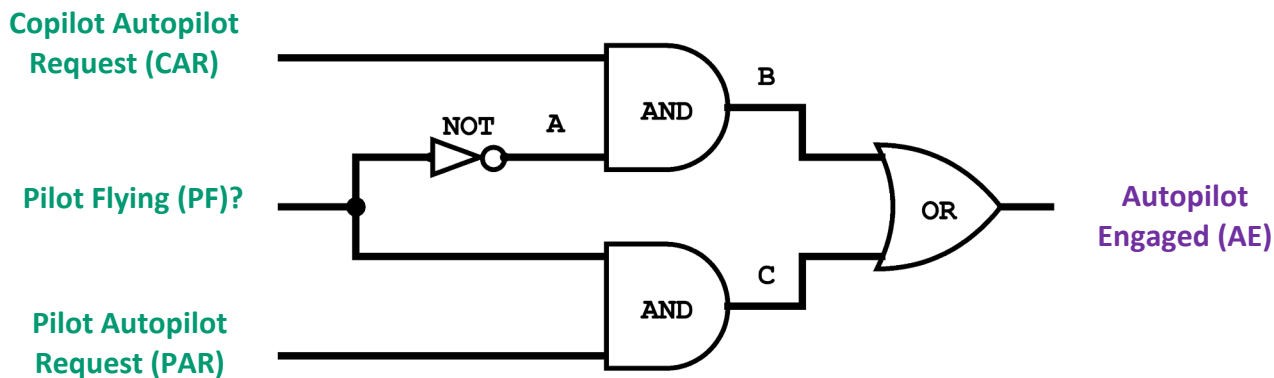
# Autopilot Control Circuit Example



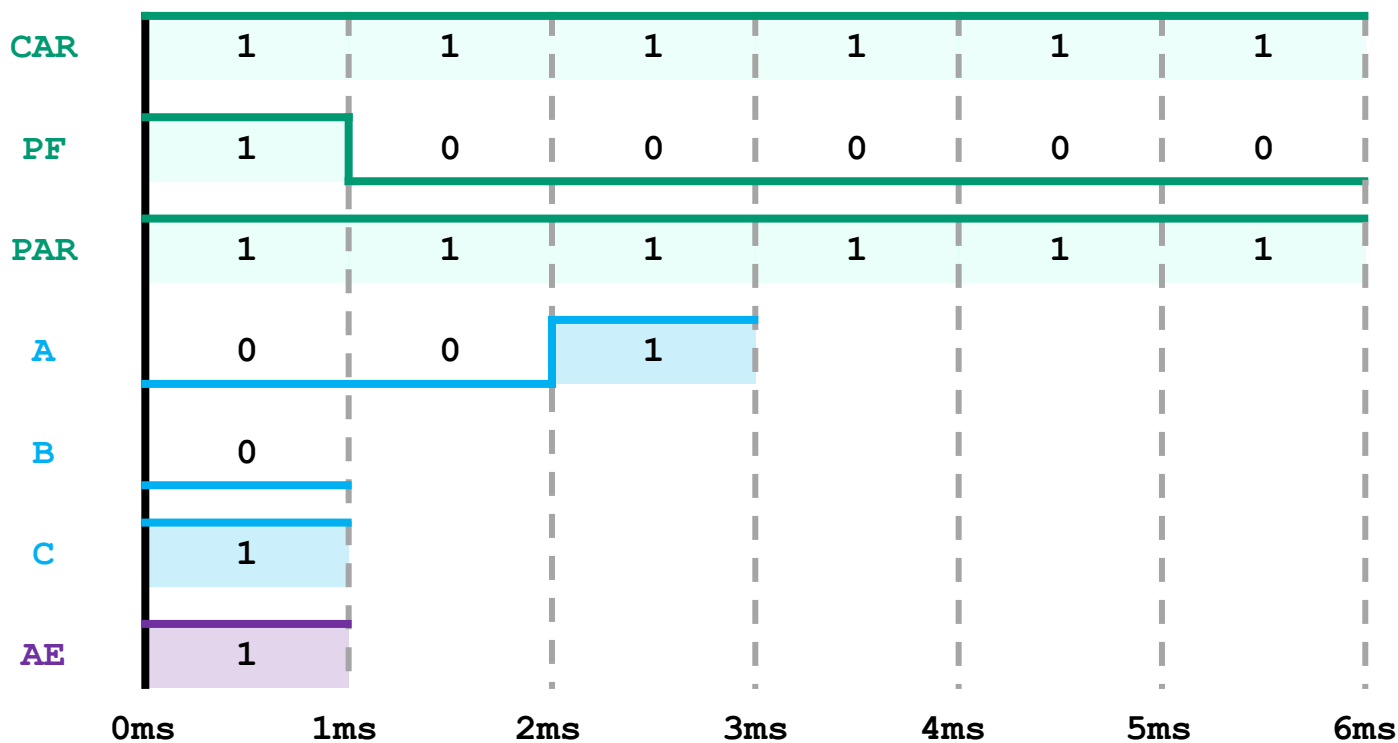
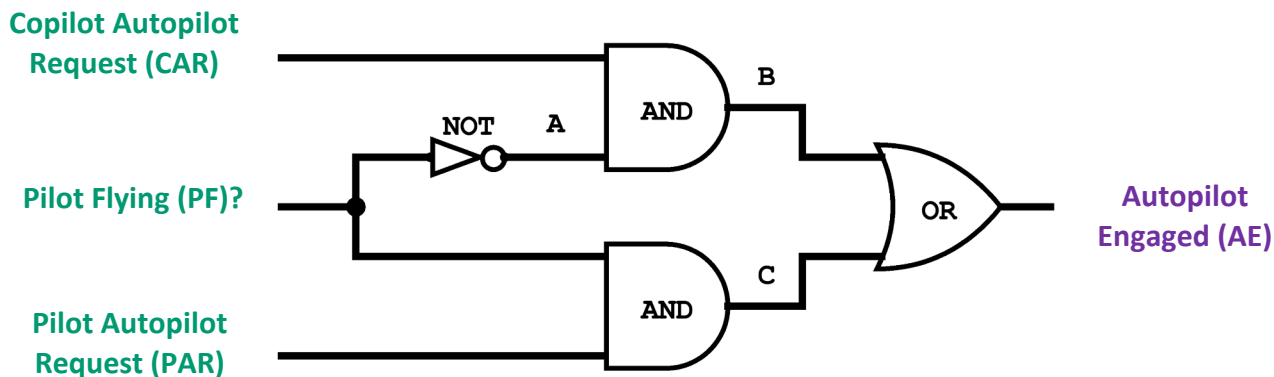
# Autopilot Control Circuit Example



# Autopilot Control Circuit Example

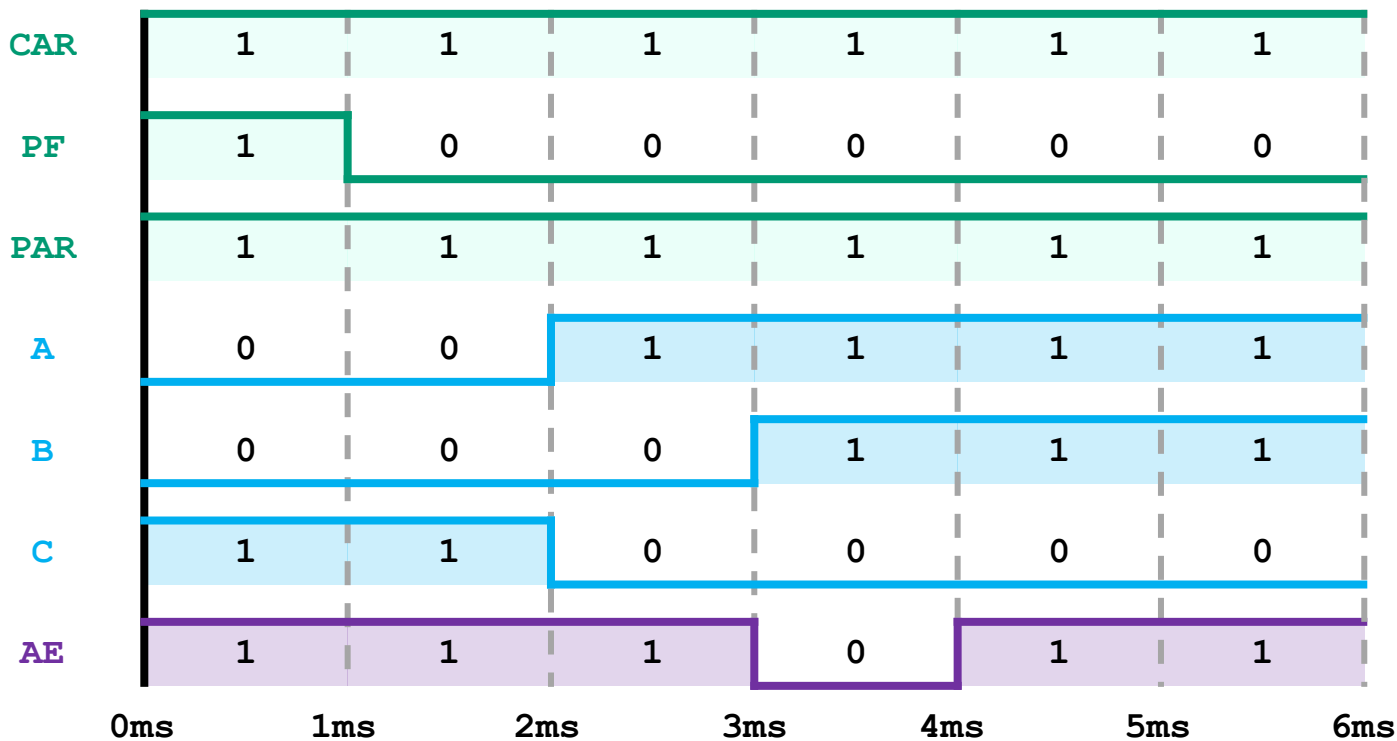
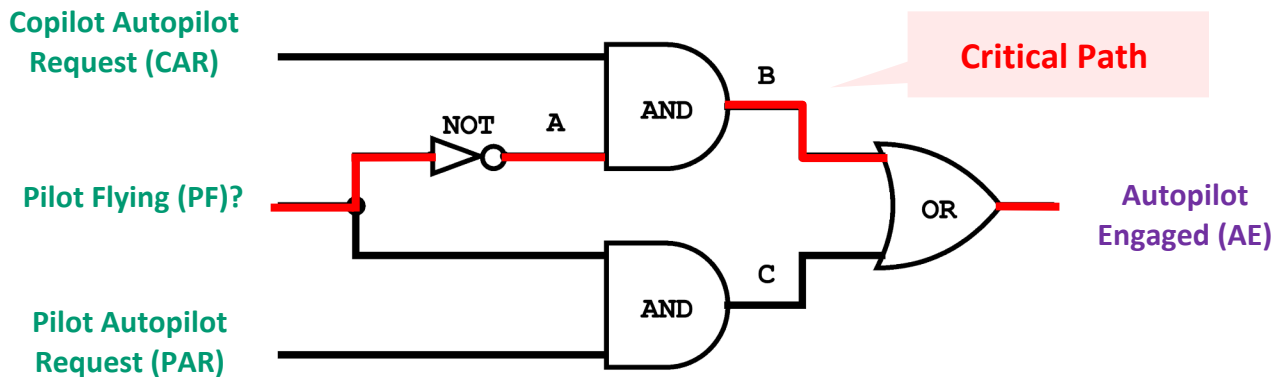


# Autopilot Control Circuit Example





# Autopilot Control Circuit Example



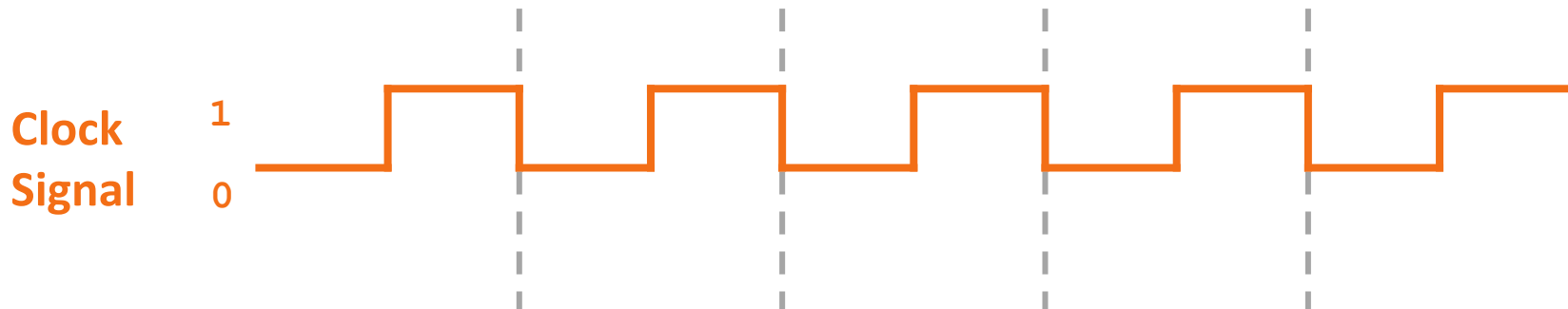
# Combinational vs. Sequential Logic

- ❖ So far, we have ignored “time” in our circuits
- ❖ Our chips used **combinational logic**
  - When given inputs, the chip computes its output instantaneously
  - The output is a function of the current inputs, with no memory of previous events
- ❖ Today, we’ll start exploring what happens when we consider time, ultimately building to **sequential logic**

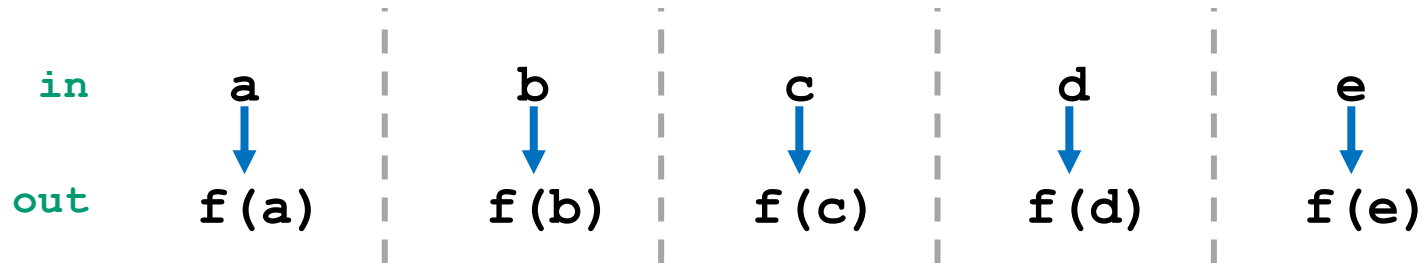
# What is Sequential Logic?

- ❖ Sequential logic incorporates time in hardware
- ❖ Output depends on present value of its input signals *and* a sequence of past inputs
- ❖ Sequential logic resolves the problem introduced with combinational logic
  - Does so by adding a **delay** for the entire circuit before evaluating the result
  - All the circuits will be evaluated between one **clock cycle** and the output will be evaluated at the end of the clock cycle

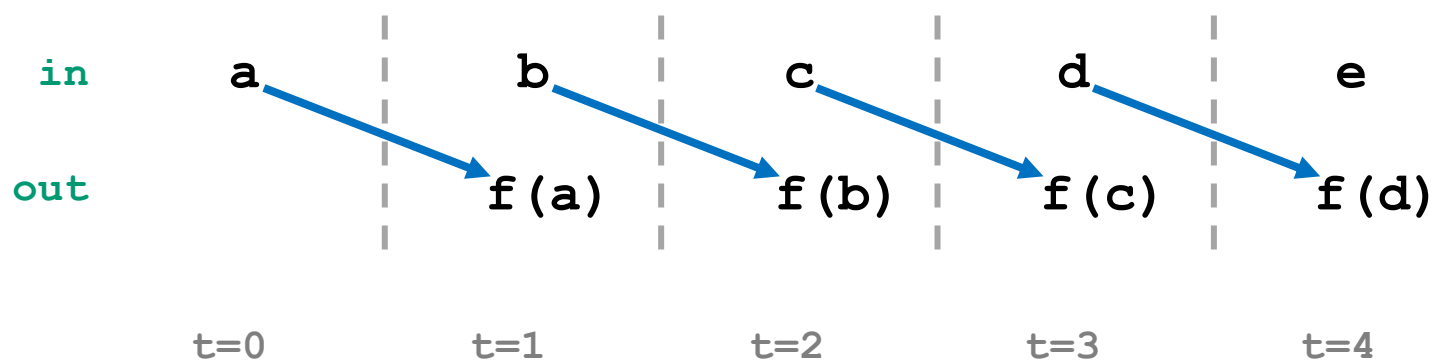
# Combinational vs. Sequential Abstraction



**Combinational:** a function of the current inputs



**Sequential:** a function of previous inputs (has “memory”)

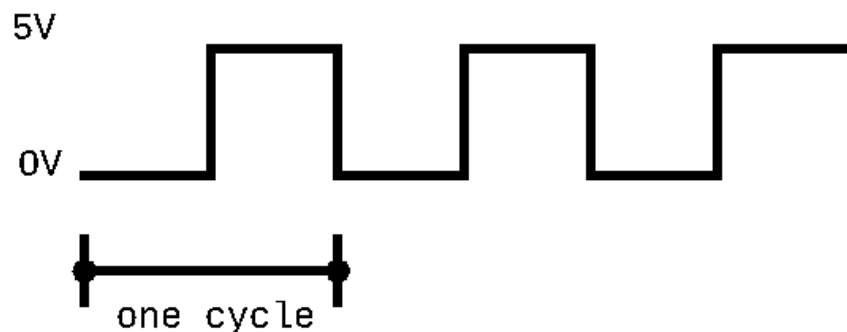


# Lecture Outline

- ❖ Growth vs. Fixed Mindset
  - Setting SMART Goals
  
- ❖ Introduction to Sequential Logic
  - The Problem of Combinational Logic
  - Autopilot Control Circuit Example
  
- ❖ **Representing Time in Hardware**
  - **Clock Signals and Units of Time in Hardware**
  
- ❖ The Data Flip-Flop (DFF)
  - Implementation and Examples

# Representing Time: Clock Signals

- ❖ We physically represent time in hardware with a **clock signal**
  - A clock signal changes its frequency at a set time rate
  - Alternates between a low signal and a high signal of equal length
- ❖ A **cycle** is a period of time between a low and high signal
  - Represents one unit of time in hardware
  - We can change how long a unit of time is by alternating the length of the low and high signals



# Physical Timekeeping

- ❖ Hardware keeps track of time using an alternating signal
  - Creates the idea of **discrete time**: state changes only occur in discrete intervals, right when signal alternates

Physical  
Time

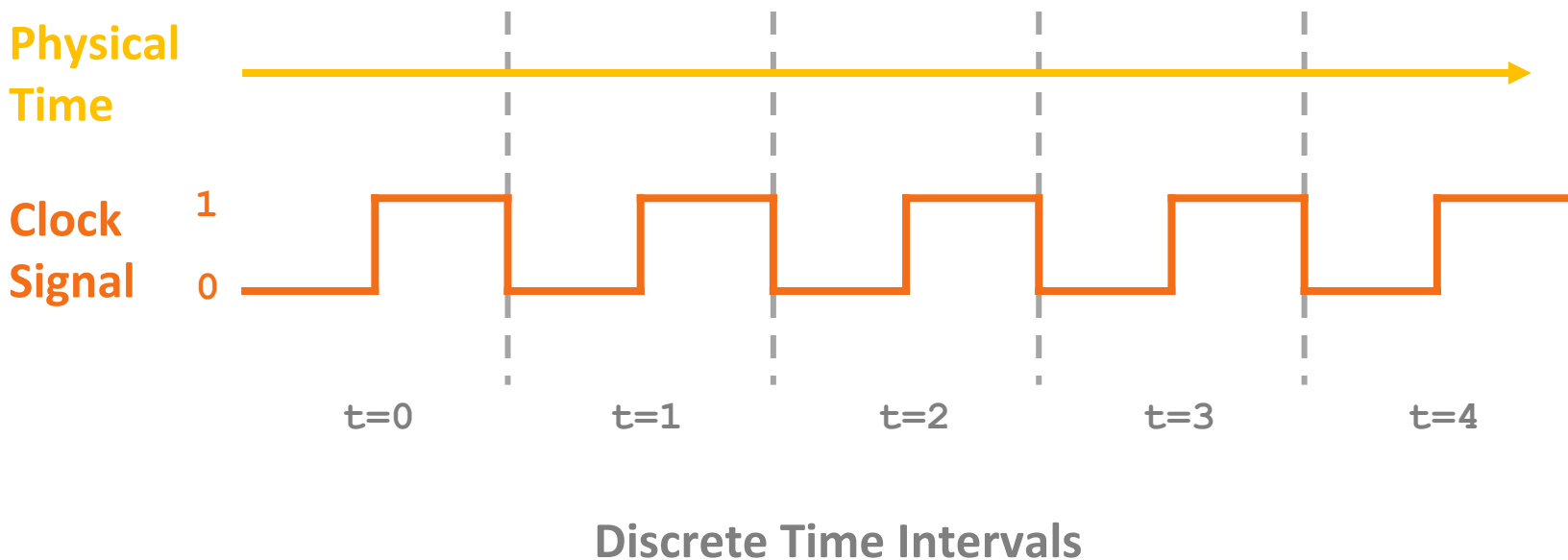


Clock  
Signal



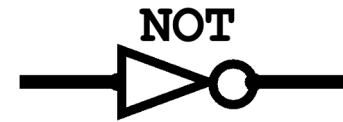
# Physical Timekeeping

- ❖ Hardware keeps track of time using an alternating signal
  - Creates the idea of **discrete time**: state changes only occur in discrete intervals, right when signal alternates

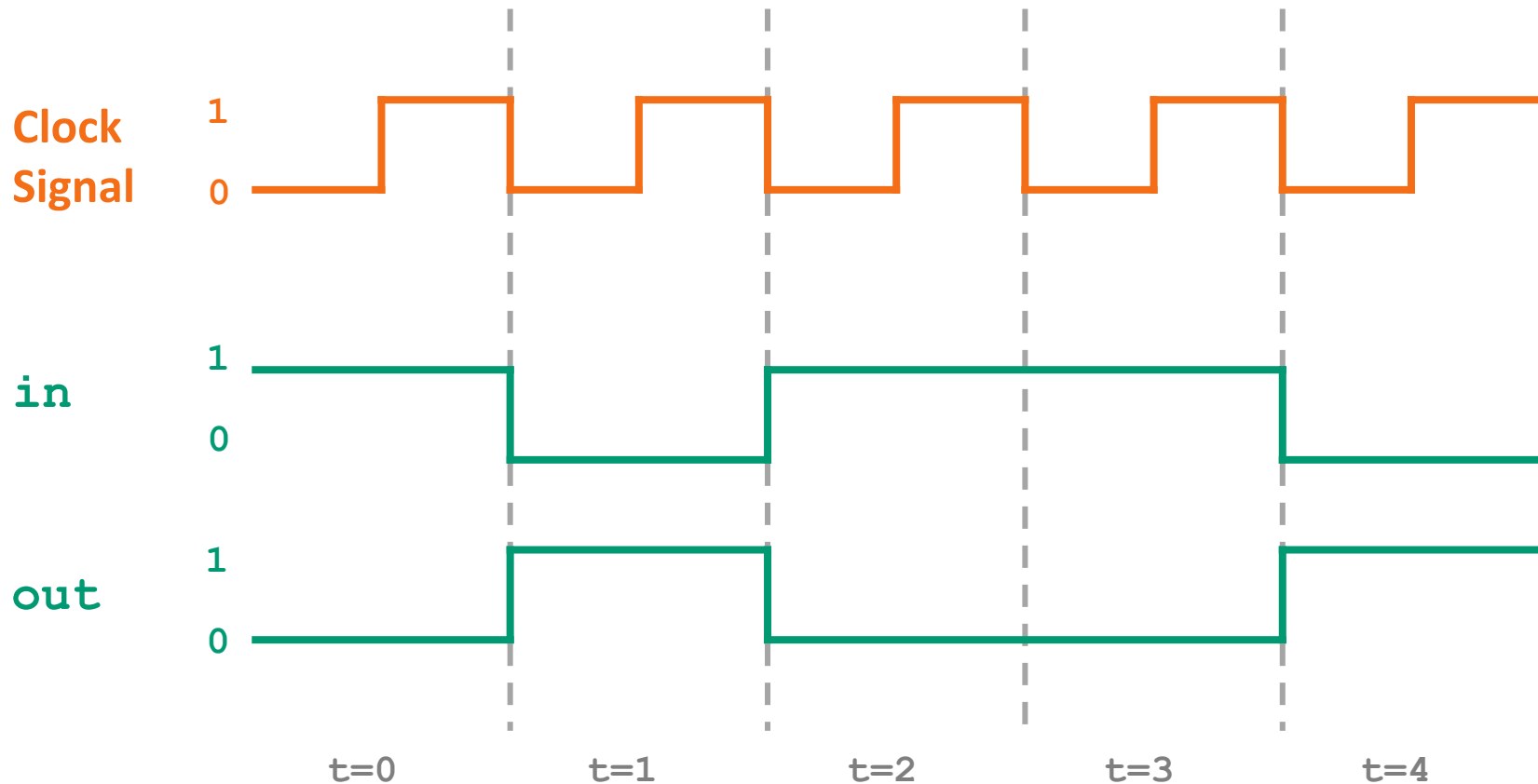




# Adding a Clock: Ideal

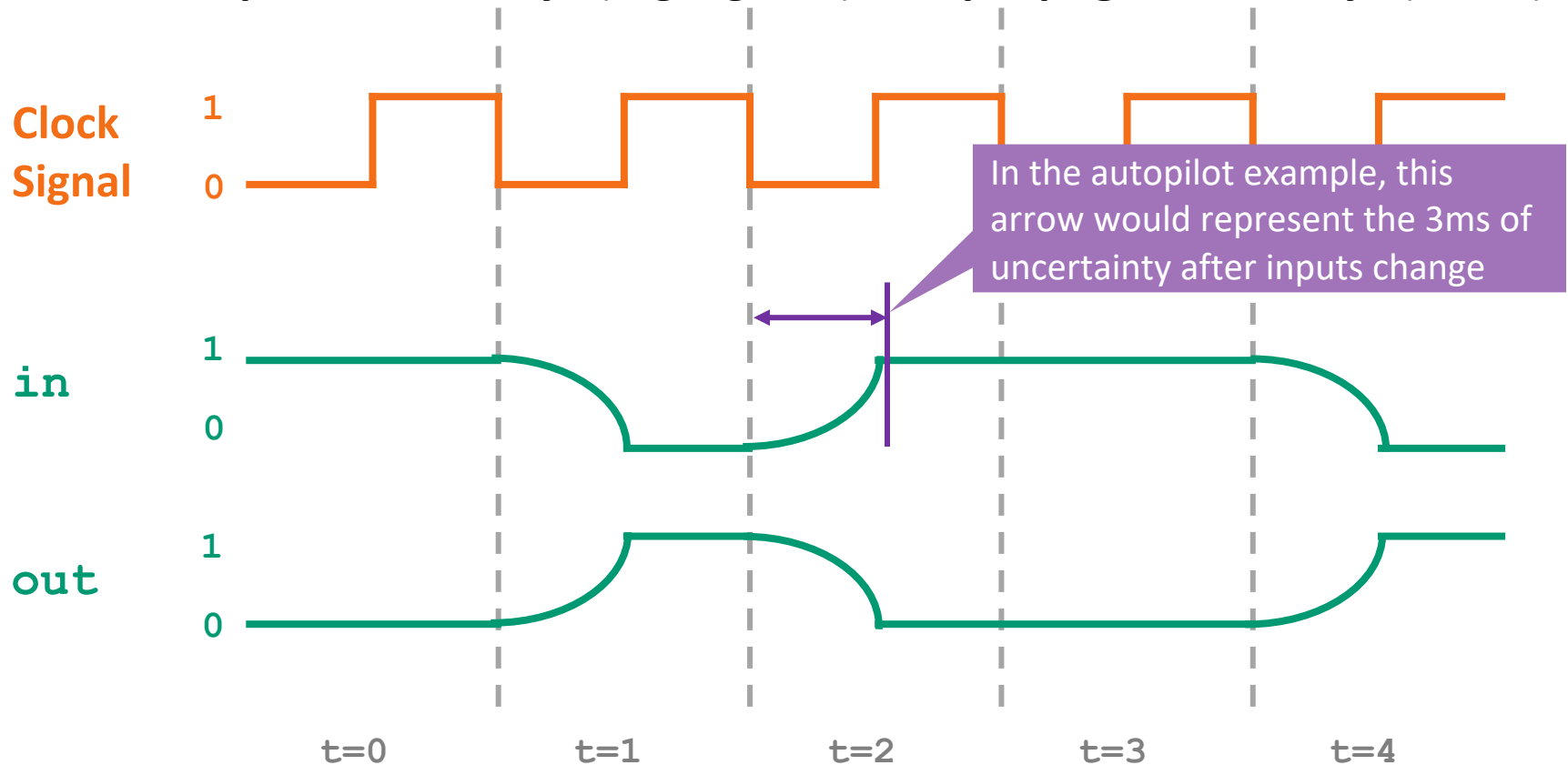


- ❖ We want this behavior from a simple, combinational Not gate:



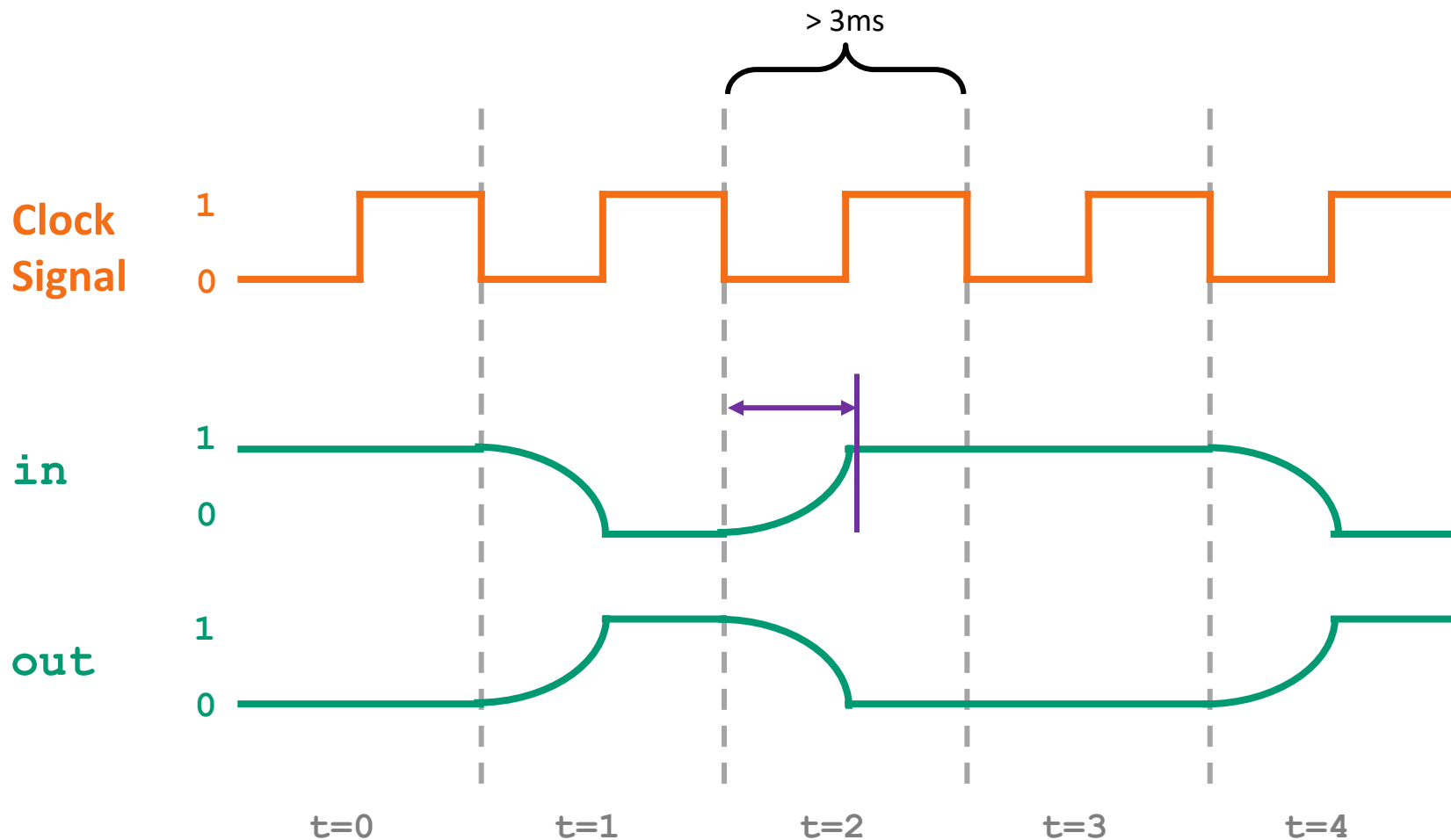
# Adding a Clock: Reality

- ❖ Combinational logic may be incorrect for a period immediately after inputs change
  - **Computation delays** (logic gates) and **propagation delays** (wires)



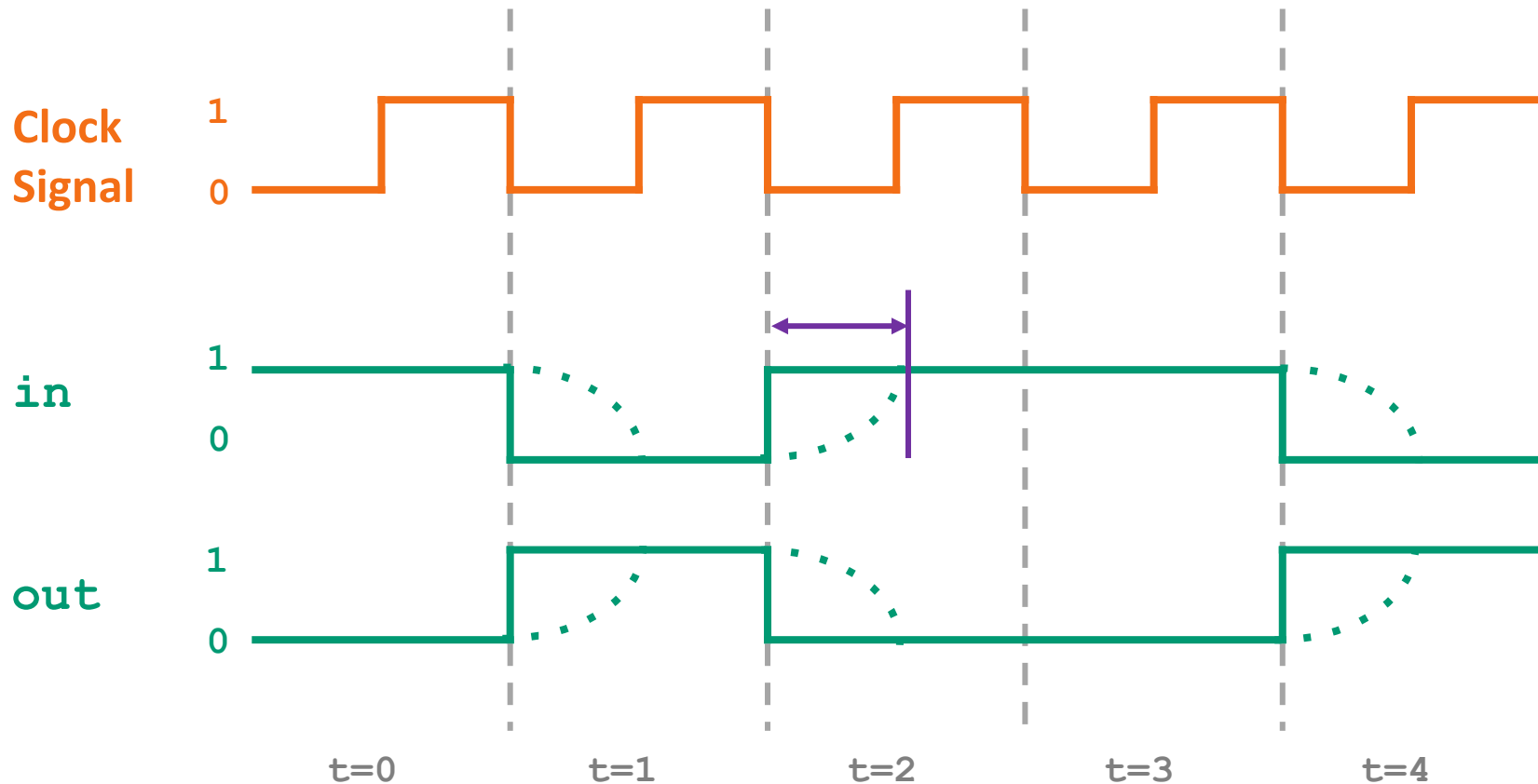
# Adding a Clock: Clock Cycles

- ❖ Choose a clock cycle length slightly longer than the delays



# Adding a Clock: Abstraction

- ❖ If we use a long enough clock cycle, we can *pretend* that combinational chips (like Not) work instantly



## < Lecture 5: Growth Mindset & Sequential Logic



🌐 When poll is active, respond at **PollEv.com/cse390b**

📧 Text **CSE390B** to **22333** once to join

# Which of the following statements about sequential logic is FALSE?

We represent time with a clock signal that alternates between high and low signals

The DFF allows us to reuse outputs as inputs in a circuit

The outputs for a DFF at the second clock cycle is determined by the inputs at the third clock cycle

Sequential logic is important because it addresses slight delays caused in circuit gates

We're lost...

Total Results: 0

Powered by  **Poll Everywhere**

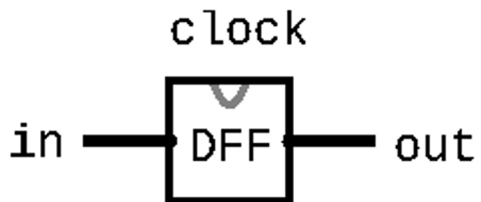


# Lecture Outline

- ❖ Growth vs. Fixed Mindset
  - Setting SMART Goals
- ❖ Introduction to Sequential Logic
  - The Problem of Combinational Logic
  - Autopilot Control Circuit Example
- ❖ Representing Time in Hardware
  - Clock Signals and Units of Time in Hardware
- ❖ **The Data Flip-Flop (DFF)**
  - **Implementation and Examples**

# The Data Flip-Flop Gate

- ❖ Simplest state-keeping component
  - 1-bit input, 1-bit output
  - Wired to the clock signal
  - Always outputs its previous input:  $\text{out}(t) = \text{in}(t-1)$
- ❖ Implementation: a gate that can flip between two stable states (remembering 0 vs. remembering 1)
  - Gates with this behavior are “Data Flip Flops” (DFFs)

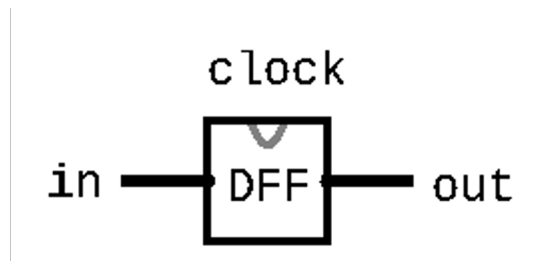
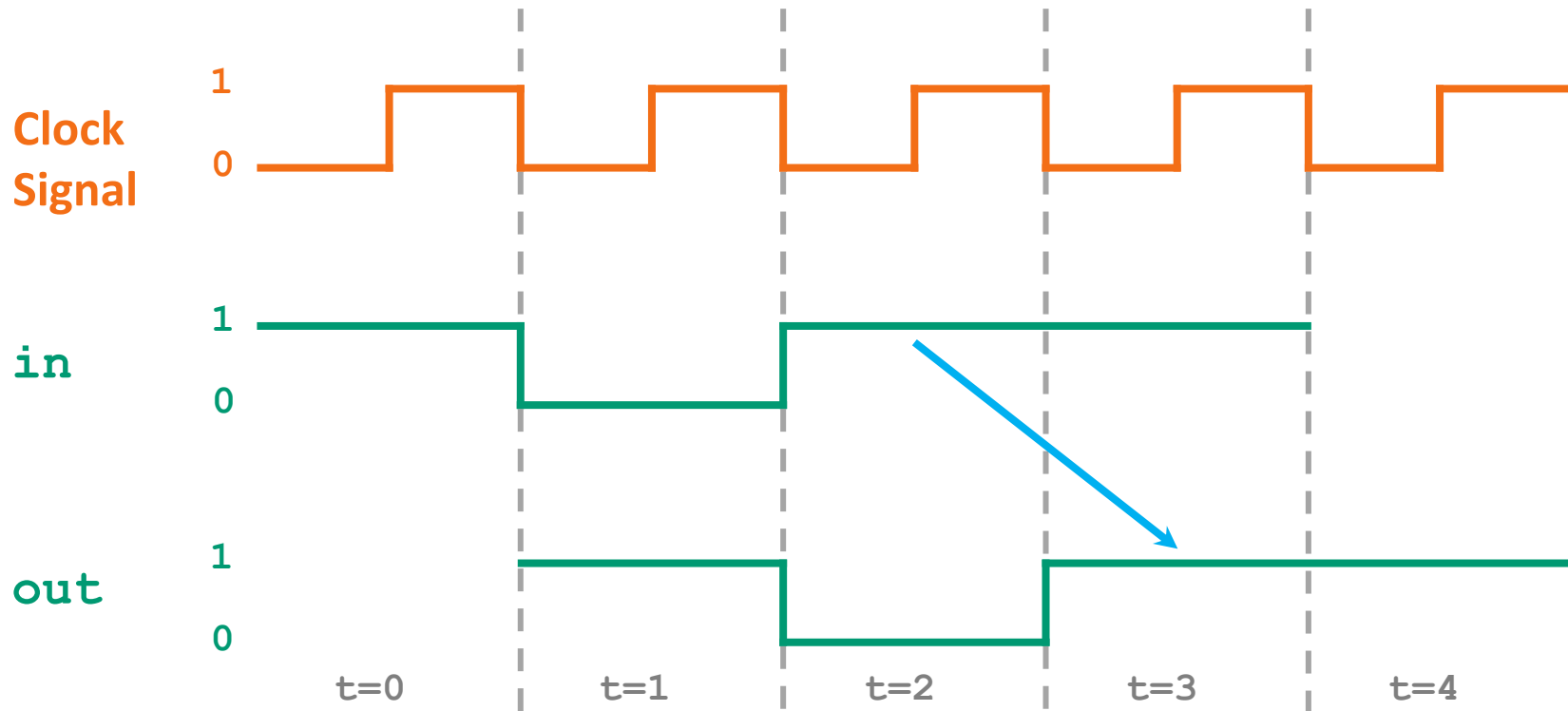


# Aside: Treating the DFF as a Primitive

- ❖ Disclaimer: DFFs can be made from Nand gates exclusively
  - But requires wiring them together in a “messy” loop that the hardware simulator can’t simulate and isn’t very educational
- ❖ For simplicity, we will treat the DFF as a primitive in the projects
  - Just like Nand, you can use the built-in implementation



# Data Flip-Flop (DFF) Behavior

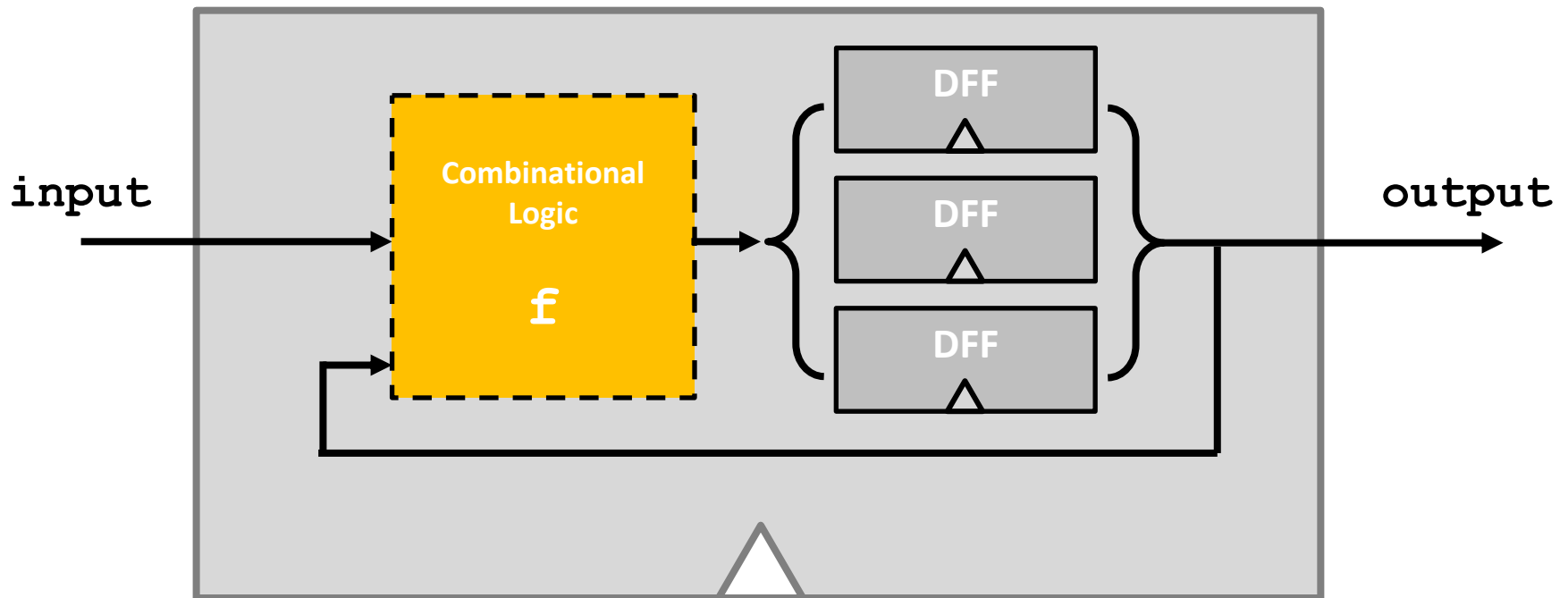


# Sequential Chips

- ❖ A category of chips that utilize the clock signal, in addition to any combinational logic
  
- ❖ Capable of:
  - Maintaining state
  - Optionally, acting on that state and the current inputs
    - Can incorporate combinational logic as well
  
- ❖ Constructed from:
  - DFFs
  - Combinational logic (which is entirely constructed from Nand)

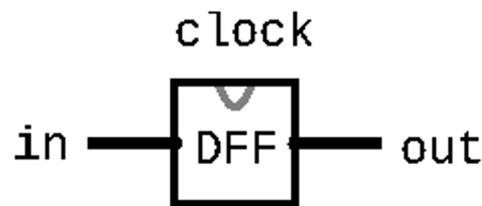
# Sequential Chips

$$\text{output}(t) = f(\text{state}(t-1), \text{input}(t))$$



# D Flip-Flop: Time Series

## ❖ DFF Specification:



$$\text{out}(t) = \text{in}(t-1)$$

<b>in</b>	0	0	1	1	0	1	0	...
<b>out</b>	0	0	0	1	1	0	1	...
<b>time</b>	t=0	t=1	t=2	t=3	t=4	t=5	t=6	...

Example:  $\text{out}(t=3) = \text{in}(t=2)$

# DFF Example 1: Specification

- ❖ Example specification:

$$\text{out}(t) = \text{Xor}(a(t-1), b(t-1))$$

- ❖ Takes two inputs,  $a$  and  $b$ , and outputs the `XOR` of them
  - Note that  $\text{out}$  at time  $t$  is determined by  $a$  and  $b$  at time  $t-1$
  - We will need to use a DFF
- ❖ Exercise: Draw out the corresponding circuit diagram and HDL implementation

# DFF Example 1: Time Series

- ❖ Example specification:

$$\text{out}(t) = \text{Xor}(a(t-1), b(t-1))$$

<b>a</b>	0	0	1	1	1	0	0	...
<b>b</b>	0	1	0	1	1	1	0	...
<b>out</b>	0	0	1	1	0	0	1	...
<b>time</b>	t=0	t=1	t=2	t=3	t=4	t=5	t=6	...

- ❖ Example:  $\text{out}(t=3) = \text{Xor}(a(t=2), b(t=2))$

# DFF Example 1: Circuit Diagram & HDL

- ❖ Example specification:

$$\text{out}(t) = \text{Xor}(a(t-1), b(t-1))$$

- ❖ Circuit diagram:

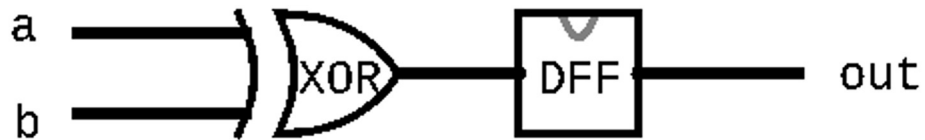
- ❖ HDL:

# DFF Example 1: Circuit Diagram & HDL

- ❖ Example specification:

$$\text{out}(t) = \text{Xor}(a(t-1), b(t-1))$$

- ❖ Circuit diagram:



- ❖ HDL:

```
CHIP Example1 {  
    IN a, b;  
    OUT out;  
  
    PARTS:  
    Xor(a=a, b=b, out=xorout);  
    DFF(in=xorout, out=out);  
}
```



# DFF Example 2: Specification

- ❖ Example specification:

$$\text{out}(t) = \text{Xor}(\text{out}(t-1), \text{in}(t-1))$$

- ❖ Notice how the specification uses  $\text{out}(t-1)$  as an input for  $\text{out}(t)$ 
  - Implies the necessity of circular wiring, separated by a DFF
- ❖ Exercise: Draw out the corresponding circuit diagram and HDL implementation

# DFF Example 2: Time Series

- ❖ Example specification:

$$\text{out}(t) = \text{Xor}(\text{out}(t-1), \text{in}(t-1))$$

<b>in</b>	0	0	1	1	1	0	0	...
<b>out</b>	0	0	0	1	0	1	1	...
<b>time</b>	t=0	t=1	t=2	t=3	t=4	t=5	t=6	...

- ❖ Example:  $\text{out}(t=1) = \text{Xor}(\text{in}(t=0), \text{out}(t=0))$

# DFF Example 2: Circuit Diagram & HDL

- ❖ Example specification:

$$\text{out}(t) = \text{Xor}(\text{out}(t-1), \text{in}(t-1))$$

- ❖ Circuit diagram:

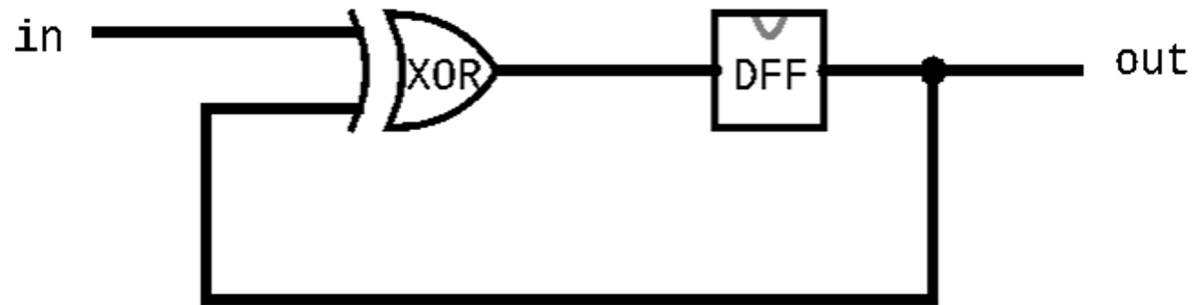
- ❖ HDL:

# DFF Example 2: Circuit Diagram & HDL

- ❖ Example specification:

$$\text{out}(t) = \text{Xor}(\text{out}(t-1), \text{in}(t-1))$$

- ❖ Circuit diagram:



- ❖ HDL:

```
CHIP Example2 {  
    IN in;  
    OUT out;  
  
    PARTS:  
    Xor(a=in, b=prevout, out=xorout);  
    DFF(in=xorout, out=prevout, out=out);  
}
```

# DFF Example 3: Specification

- ❖ Example specification:

$$\text{out}(t) = \text{And}(\text{Not}(\text{out}(t-1)), \text{in}(t-1))$$

- ❖ Exercise: Draw out the corresponding circuit diagram and HDL implementation

# DFF Example 3: Time Series

- ❖ Example specification:

$$\text{out}(t) = \text{And}(\text{Not}(\text{out}(t-1)), \text{in}(t-1))$$

<b>in</b>	1	1	0	1	1	0	0	...
<b>out</b>	0	1	0	0	1	0	0	...
<b>time</b>	t=0	t=1	t=2	t=3	t=4	t=5	t=6	...

- ❖ Example:

$$\text{out}(t=1) = \text{And}(\text{Not}(\text{out}(t=0)), \text{in}(t=0))$$

# DFF Example 3: Circuit Diagram & HDL

- ❖ Example specification:

$$\text{out}(t) = \text{And}(\text{Not}(\text{out}(t-1)), \text{in}(t-1))$$

- ❖ Circuit diagram:

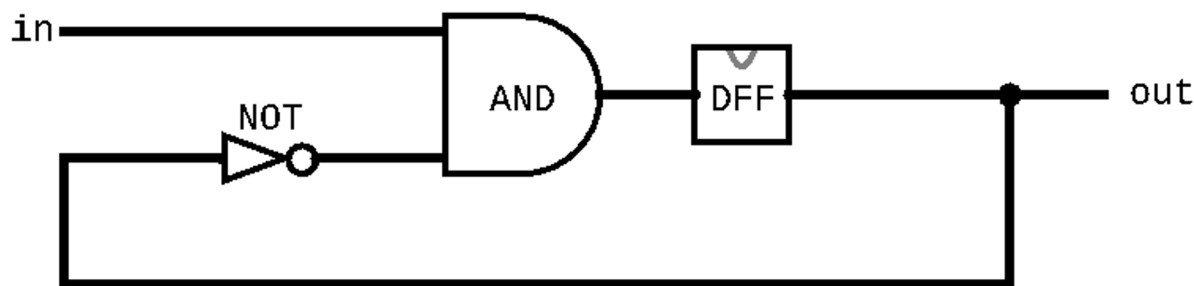
- ❖ HDL:

# DFF Example 3: Circuit Diagram & HDL

- ❖ Example specification:

$$\text{out}(t) = \text{And}(\text{Not}(\text{out}(t-1)), \text{in}(t-1))$$

- ❖ Circuit diagram:



- ❖ HDL:

```
CHIP Example3 {
```

```
  IN in;
  OUT out;
```

```
  PARTS:
```

```
    Not(in=prevout, out=notprevout);
    And(a=in, b=notprevout, out=andout);
    DFF(in=andout, out=prevout, out=out);
```

```
}
```



# Post-Lecture 5 Reminders

- ❖ Topics this Thursday:
  - Metacognitive Subject: Cornell Note-taking
  - Technical Subject: Building Memory
- ❖ **Project 3 due this Thursday (4/13) at 11:59pm**
- ❖ Preston has office hours after class in CSE2 152
  - Feel free to post your questions on the Ed board as well