CSE 390B, Spring 2022

Building Academic Success Through Bottom-Up Computing

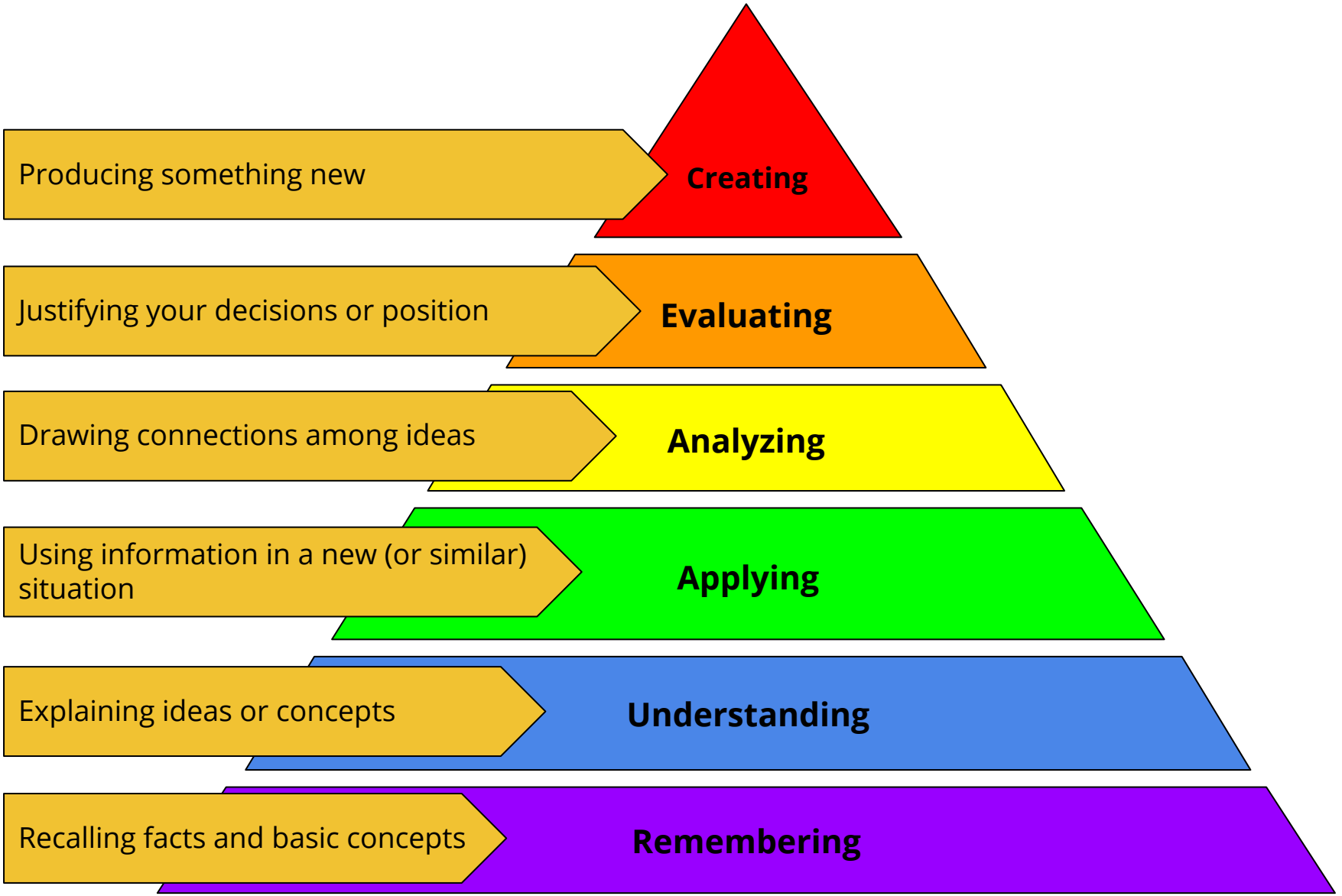
Bloom's Taxonomy & Sequential Logic

Bloom's Taxonomy, Cornell Note Taking Method,
Representing Time in Hardware, Sequential Logic

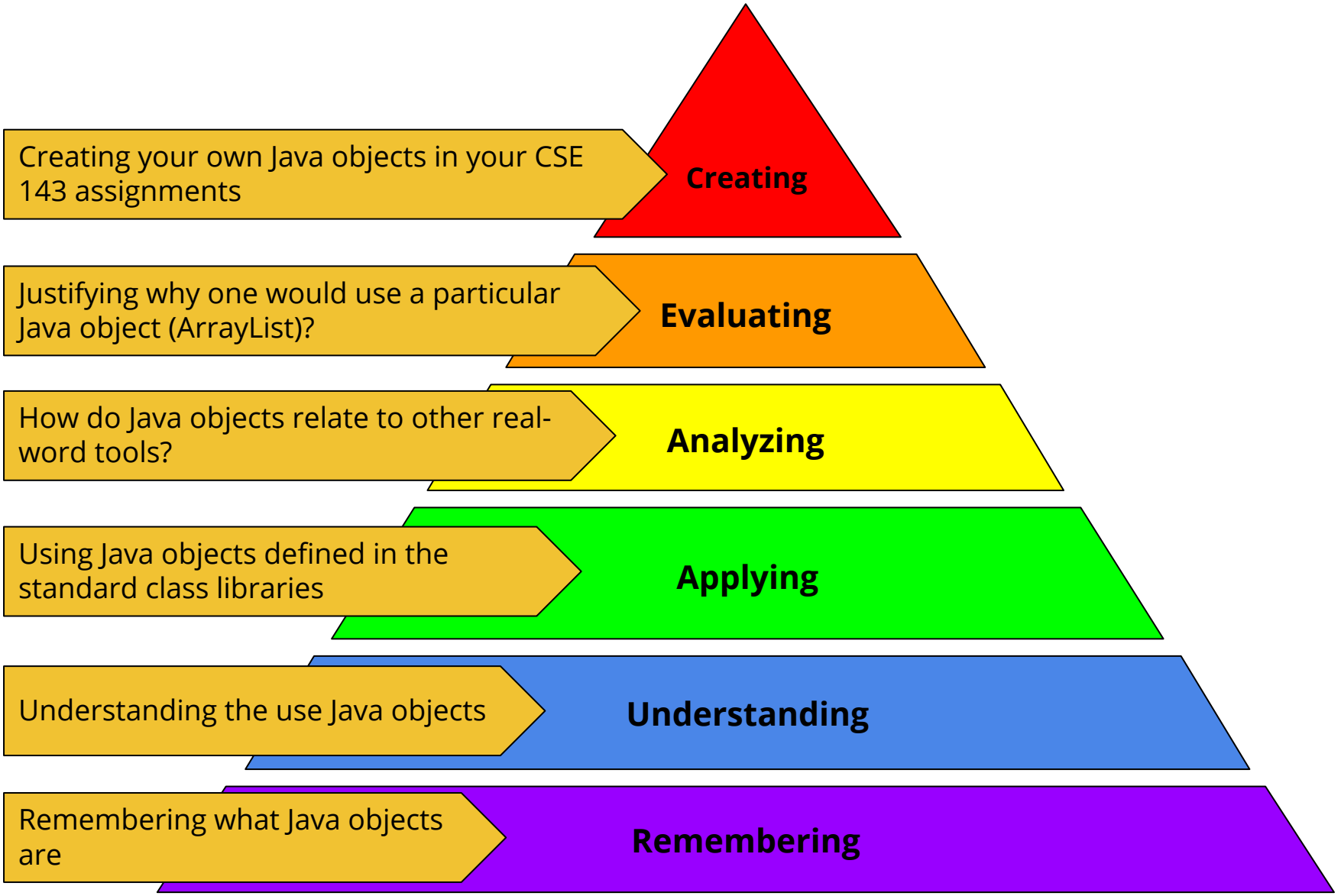
Lecture Outline

- **Bloom's Taxonomy**
  - **Applying Higher Levels of Cognition to Learning**
- Cornell Note Taking Method
  - System for Taking, Organizing, and Reviewing Notes
- Representing Time in Hardware
  - Sequential Logic vs. Combinational Logic
  - Adding a Clock
- Sequential Logic
  - Data Flip-Flop (DFF) Implementation and Examples

Bloom's Taxonomy



Bloom's Taxonomy in Action



Lecture Outline

- Bloom's Taxonomy
  - Applying Higher Levels of Cognition to Learning
- **Cornell Note Taking Method**
  - **System for Taking, Organizing, and Reviewing Notes**
- Representing Time in Hardware
  - Sequential Logic vs. Combinational Logic
  - Adding a Clock
- Sequential Logic
  - Data Flip-Flop (DFF) Implementation and Examples

Applying the Cornell Note-Taking Method

- You are going to try it… TODAY!
- Thursday you will come & contrast Cornell notes with your classmates
- You are also going to try it with one of your other classes as part of Project 4 (more on Thursday)

Lecture Outline

- Bloom's Taxonomy
  - Applying Higher Levels of Cognition to Learning
- Cornell Note Taking Method
  - System for Taking, Organizing, and Reviewing Notes
- **Representing Time in Hardware**
  - **Sequential Logic vs. Combinational Logic**
  - **Adding a Clock**
- Sequential Logic
  - Data Flip-Flop (DFF) Implementation and Examples

Poll Everywhere

Vote at https://pollev.com/cse390b

Which of the following statements about sequential logic is FALSE?

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

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

C. The outputs for a DFF at time t=2 is determined by the inputs at time t=3

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

E. We're lost…

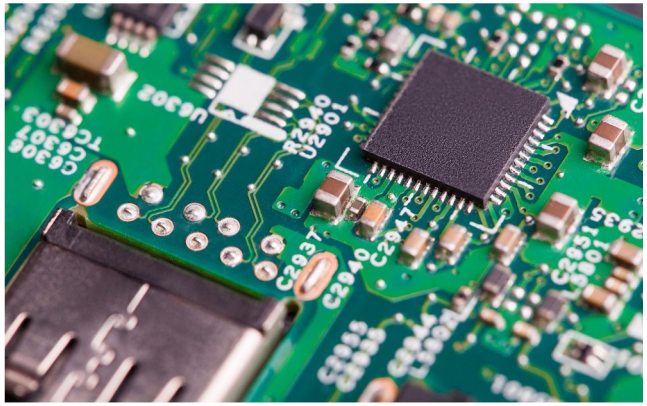
Combinational vs. Sequential Logic

- So far, we have ignored "time" in our circuits
- Our chips use **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**

Why Consider Time Now?

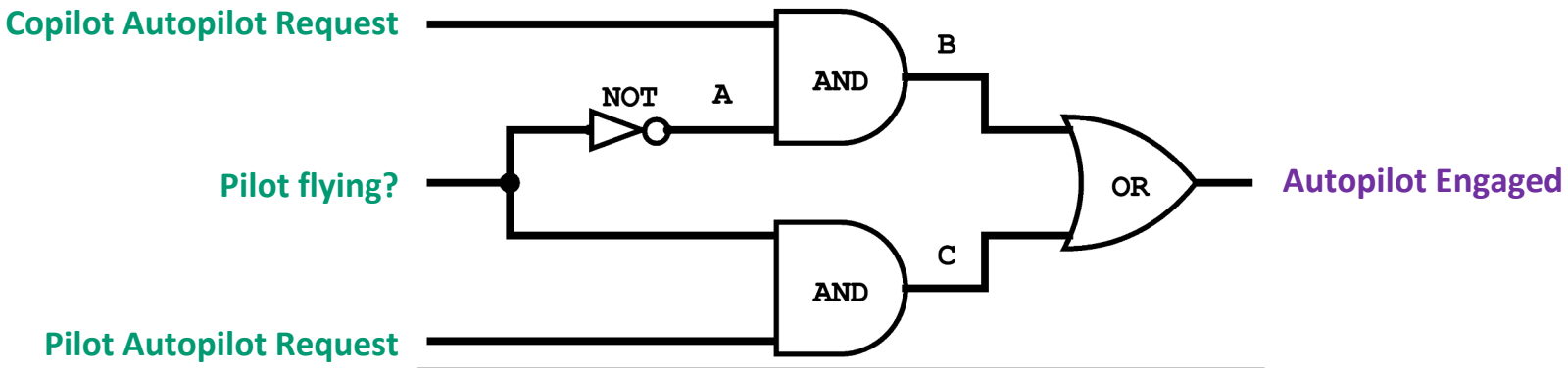
- Needed for our **abstraction**
  - We need to talk about hardware maintaining state for memory
  - We need vocabulary to talk about time

- Needed for our **implementation**
  - Physical implementations of chips cannot be instantaneous
  - We need to account for physical delays in signal propagation



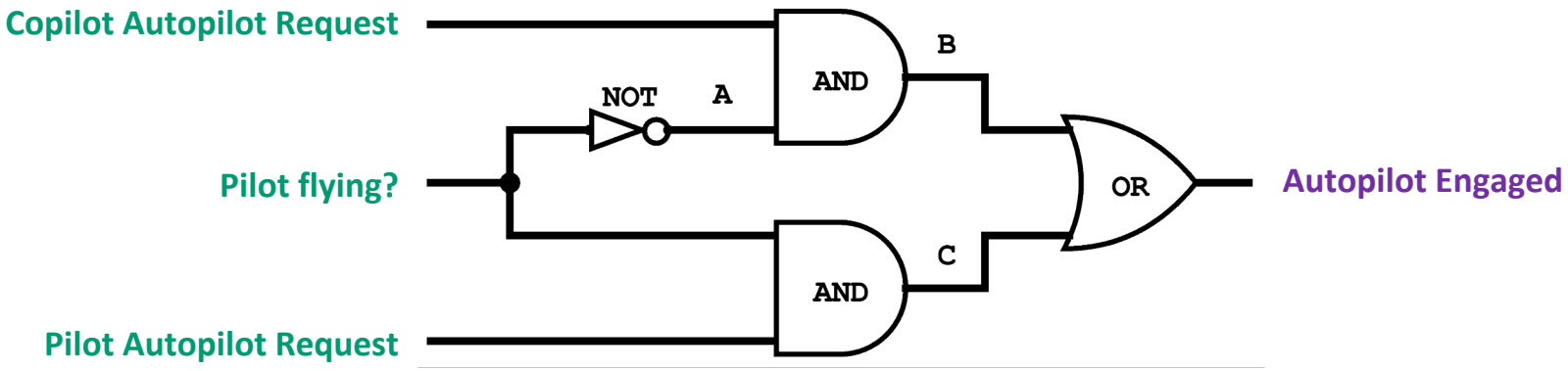
Autopilot Control Circuit Example

- 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



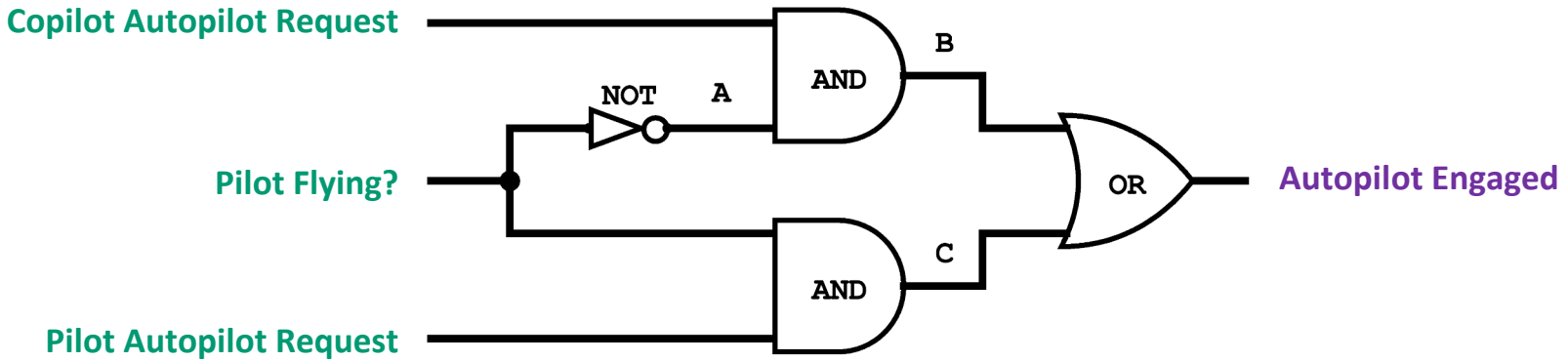
Autopilot Control Circuit Example

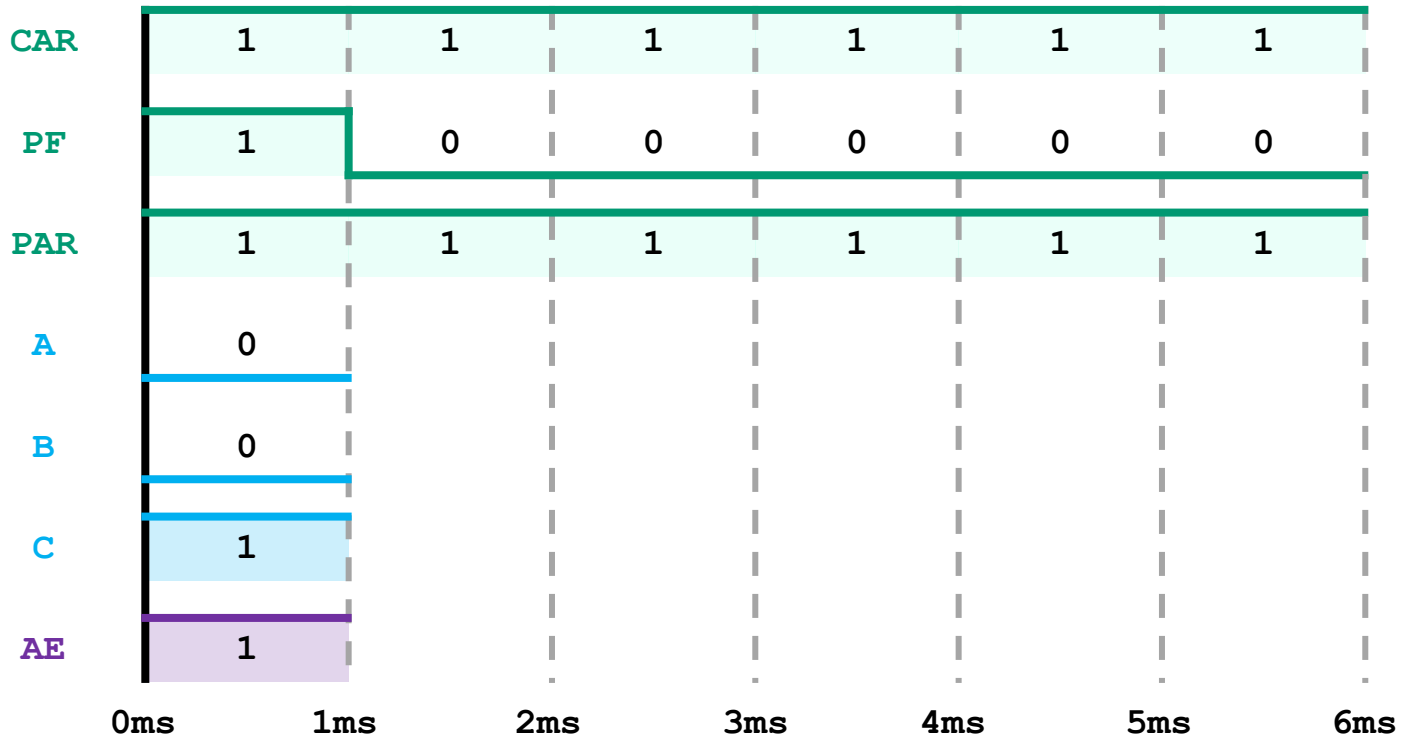
- Consider this autopilot control circuit:
  - Either the pilot or copilot is flying at any time
  - The pilot and copilot can separately request autopilot
  - Only the person flying can request autopilot



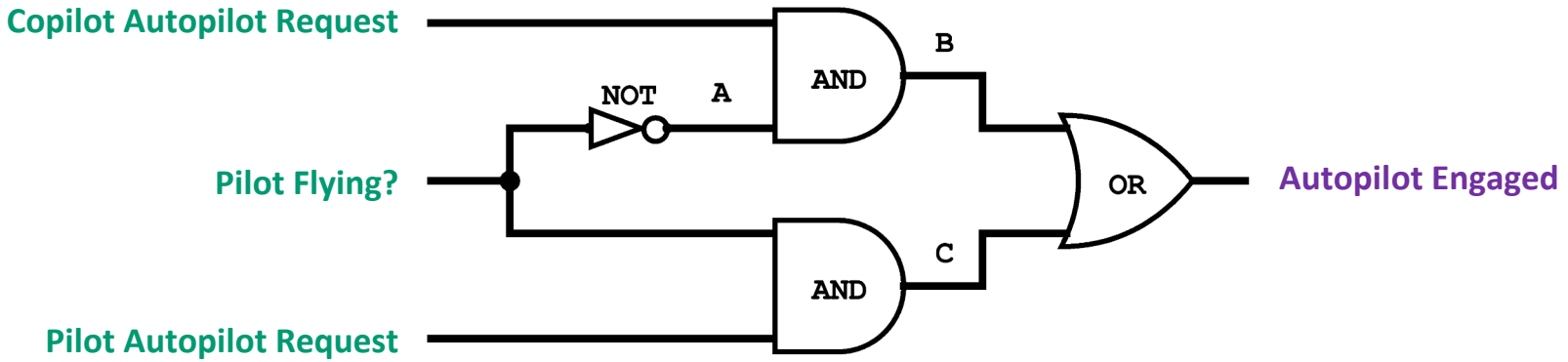
- Let's assume every logic gate takes `1ms` to compute
  - For example, if an input changes at `t=4ms`, the gate will only output the new result at `t=5ms`

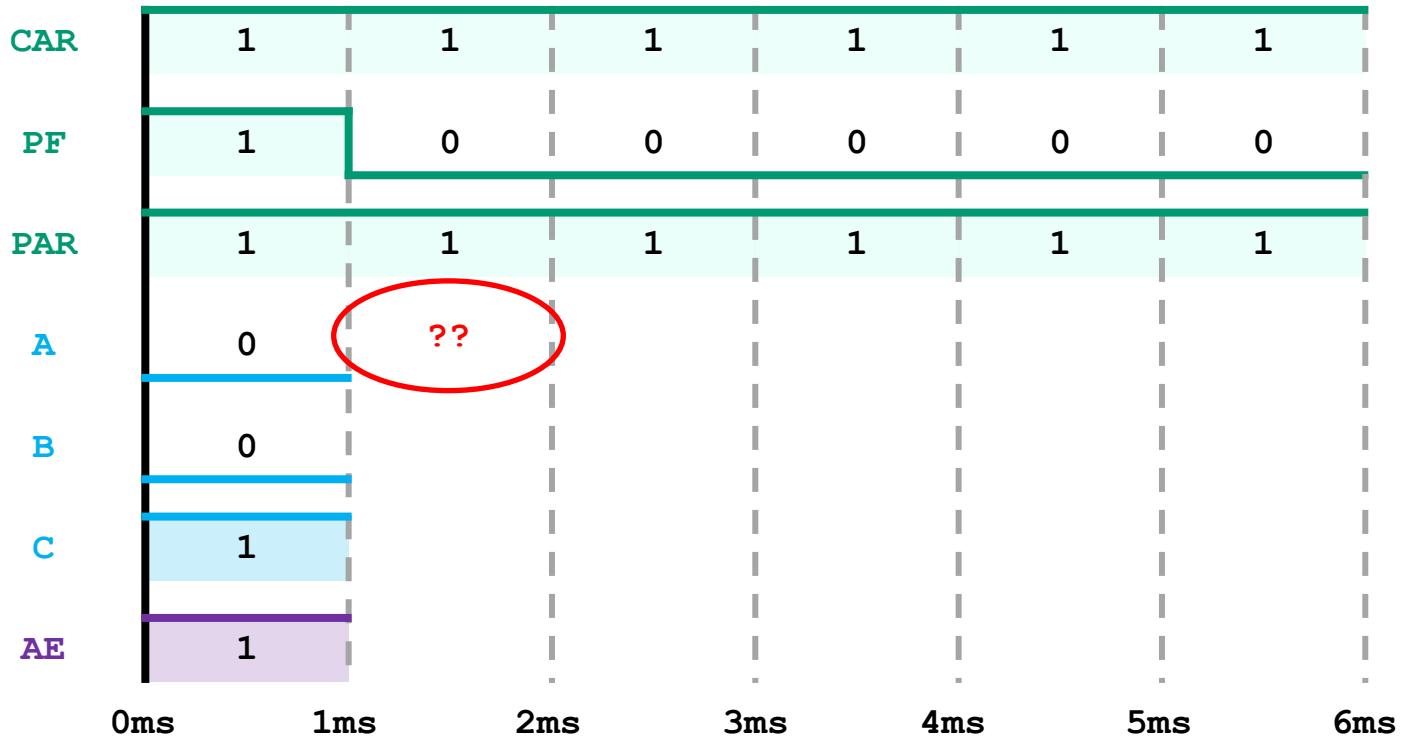
Autopilot Control Circuit Example



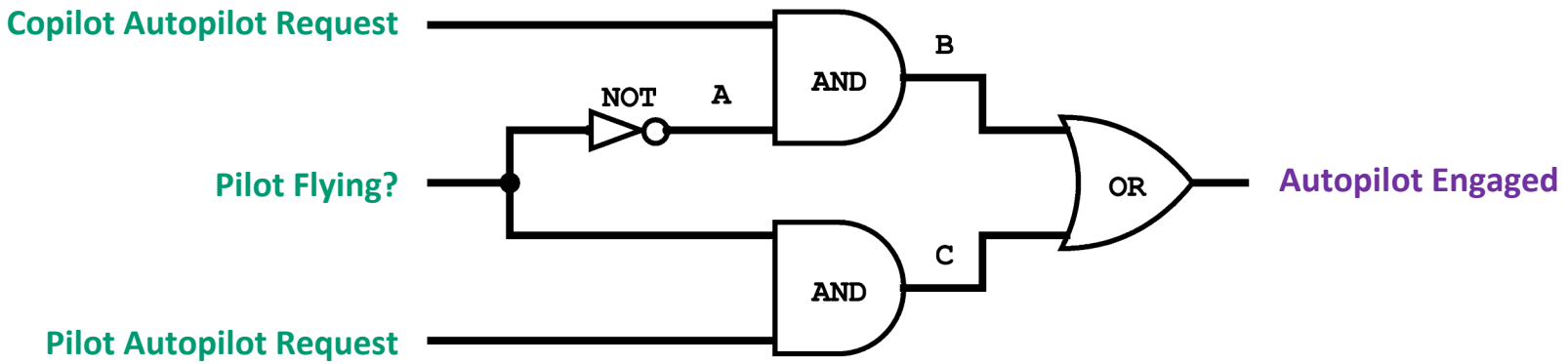


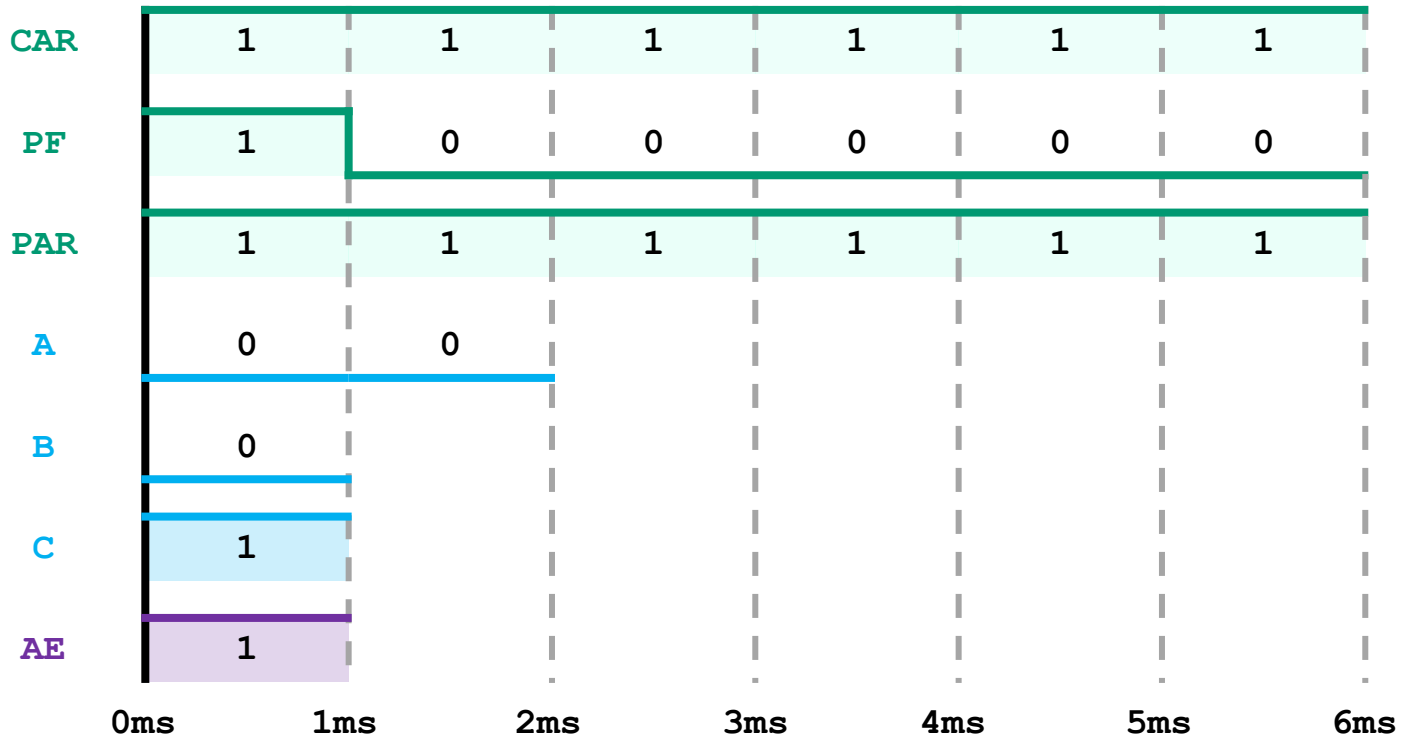
Autopilot Control Circuit Example



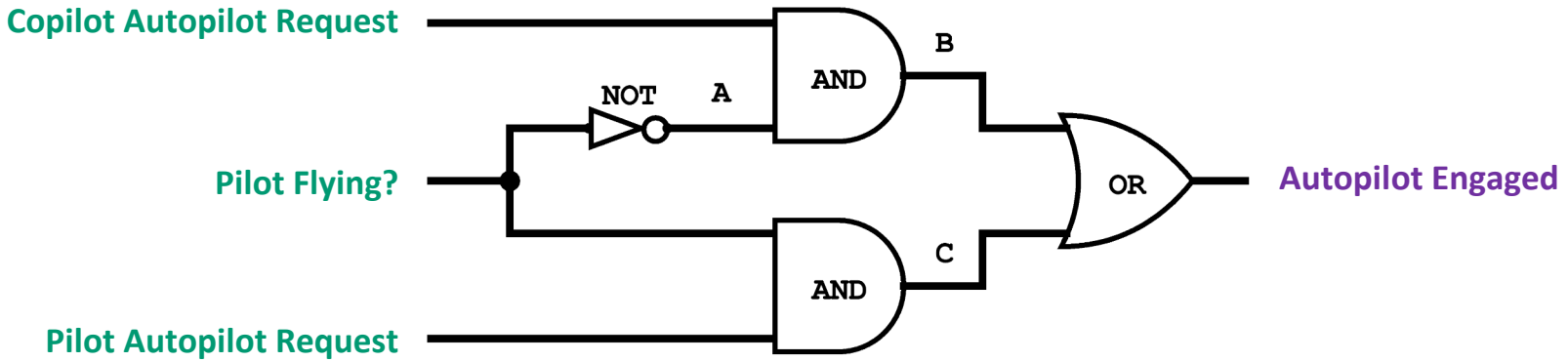


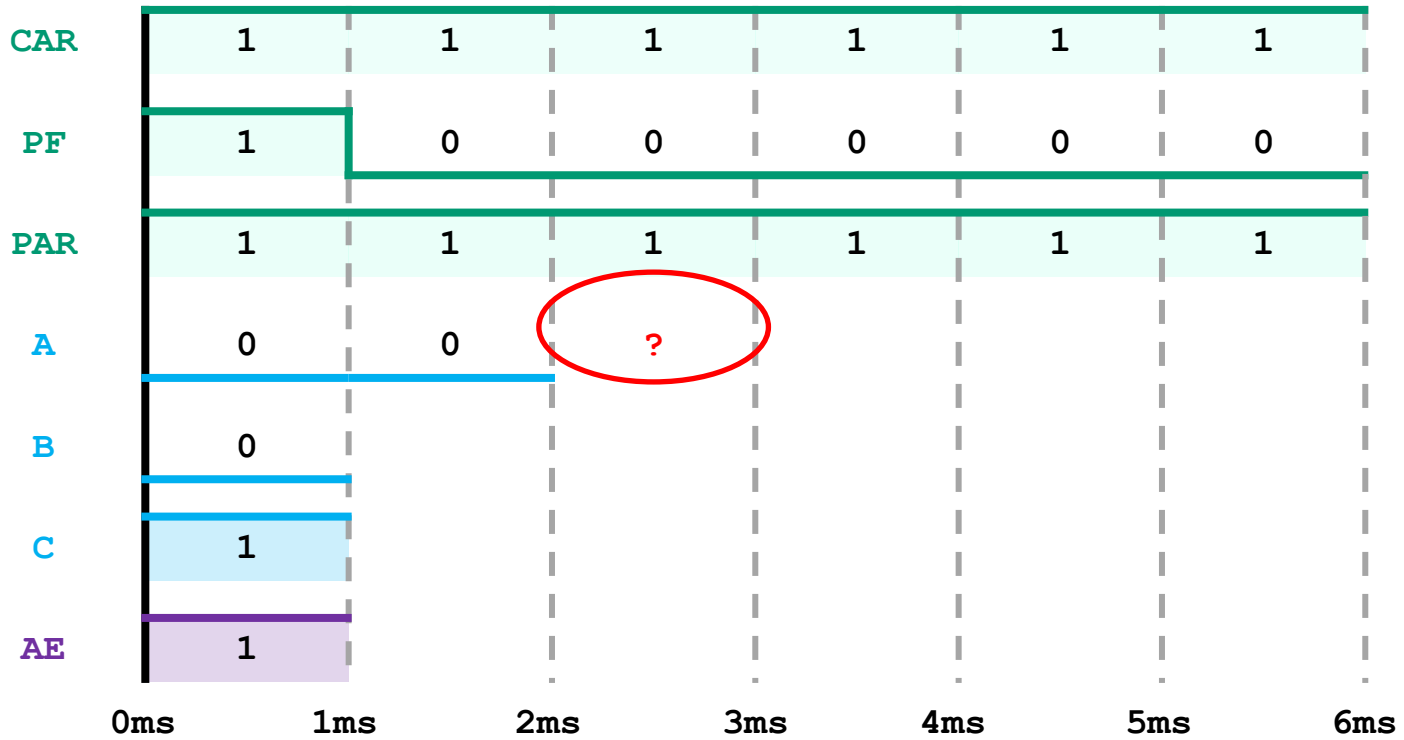
Autopilot Control Circuit Example



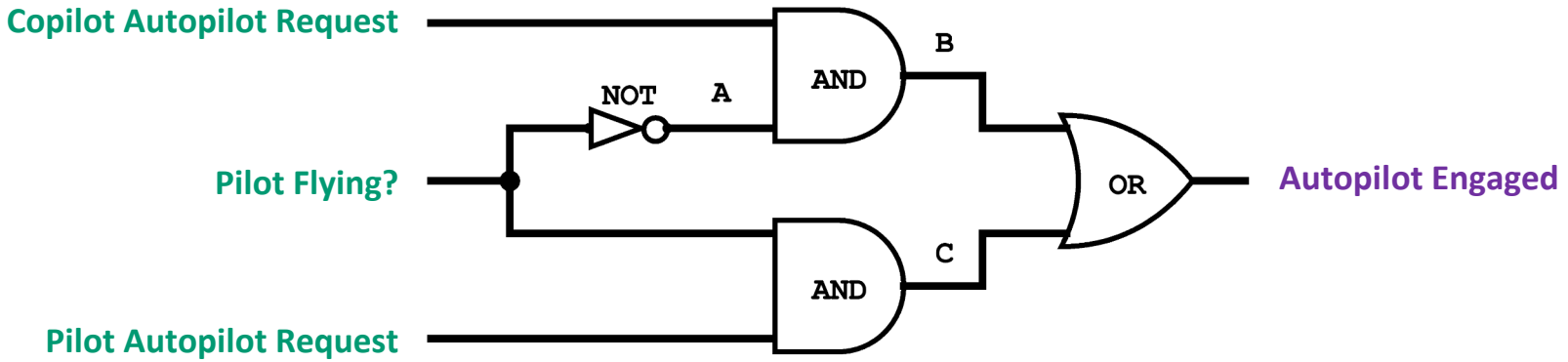


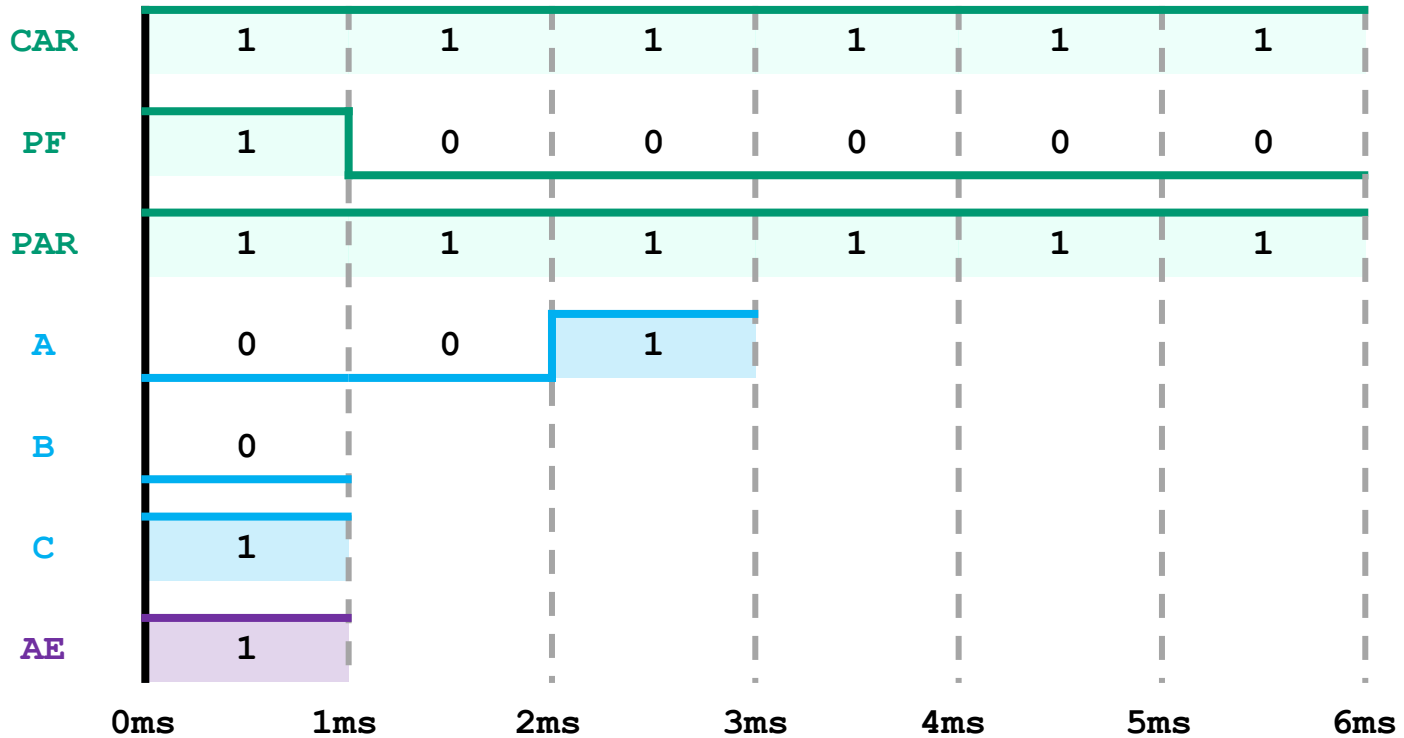
Autopilot Control Circuit Example





Autopilot Control Circuit Example

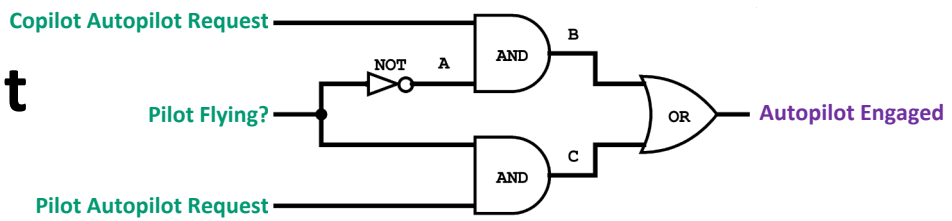


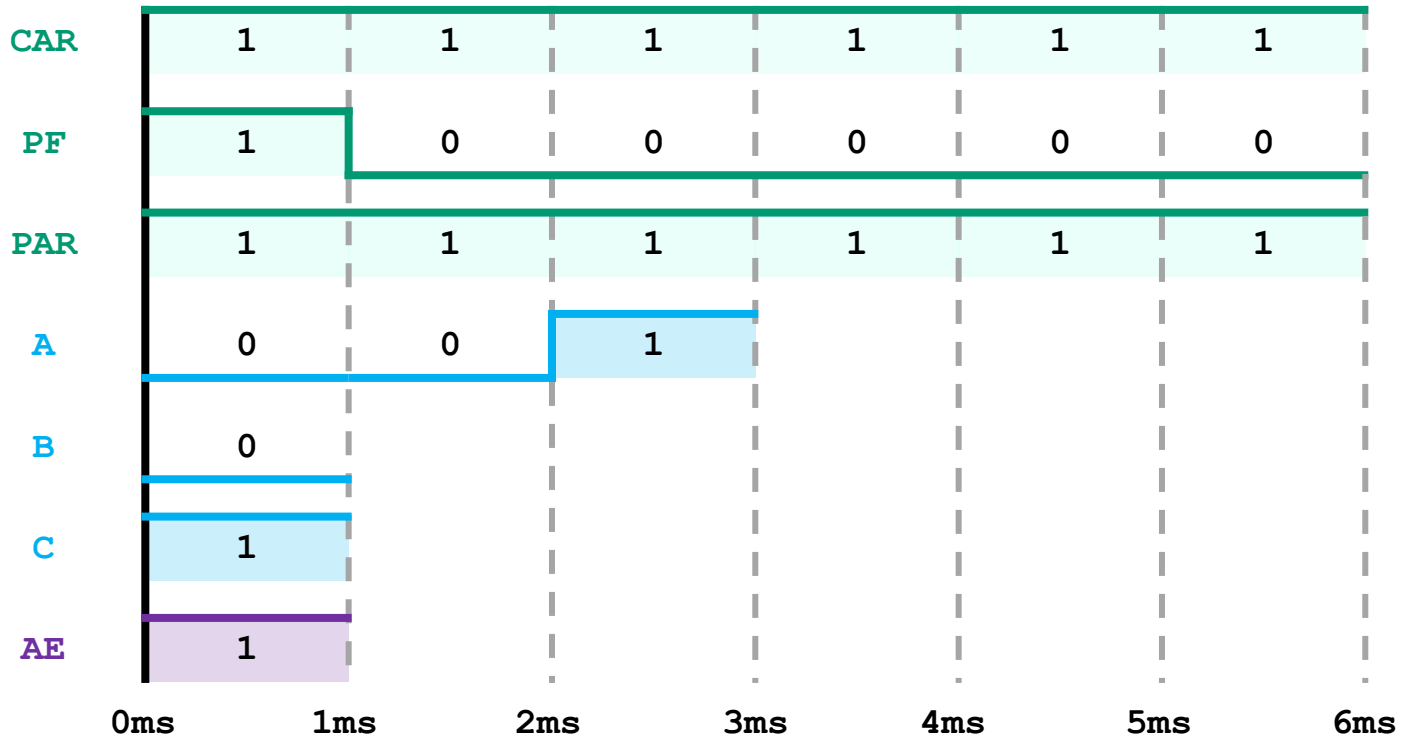


Poll Everywhere

Vote at https://pollev.com/cse390b

Describe the behavior of the Autopilot Engaged (AE) output between 1ms to 6ms.

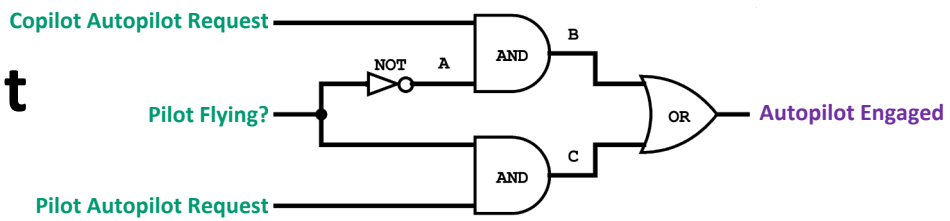


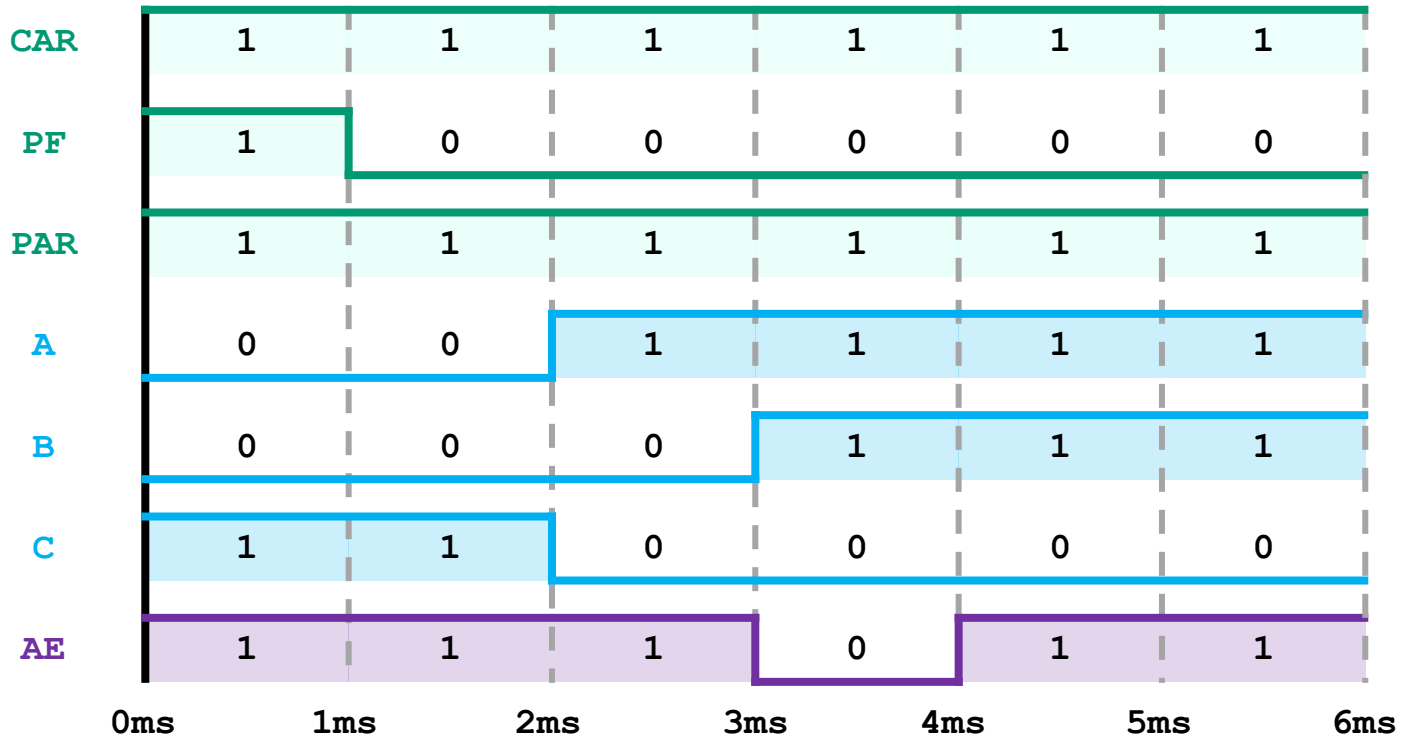


Poll Everywhere

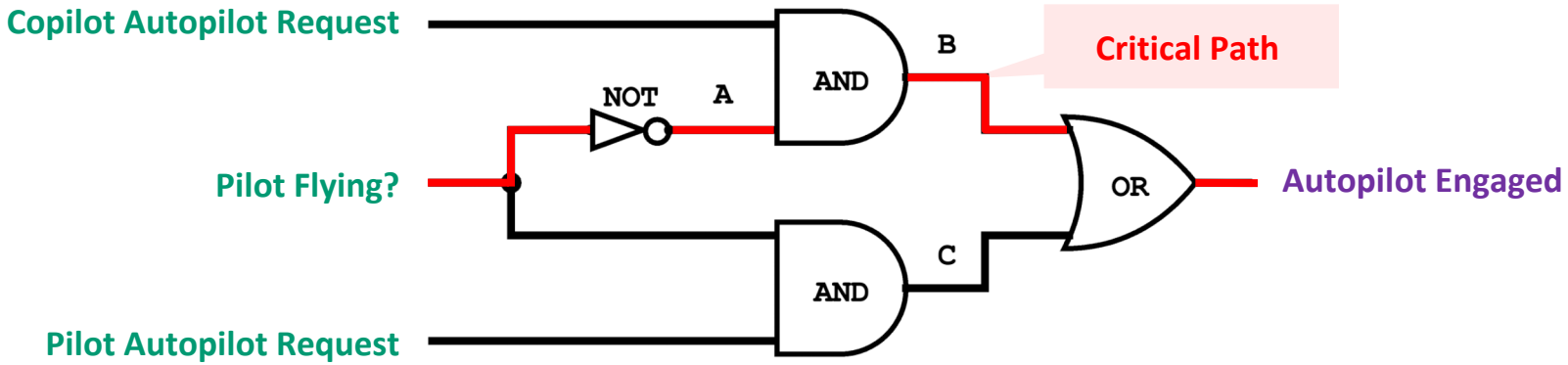
Vote at https://pollev.com/cse390b

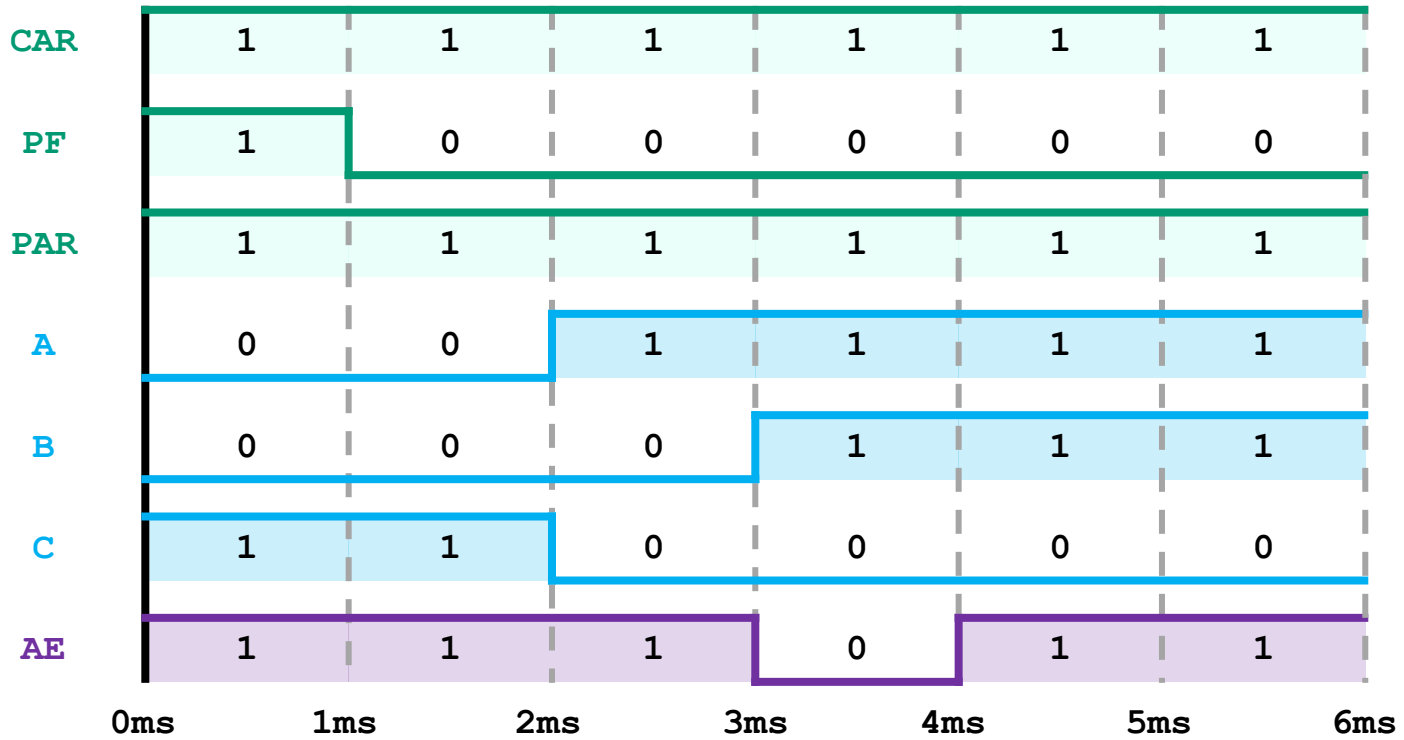
Describe the behavior of the Autopilot Engaged (AE) output between 1ms to 6ms.





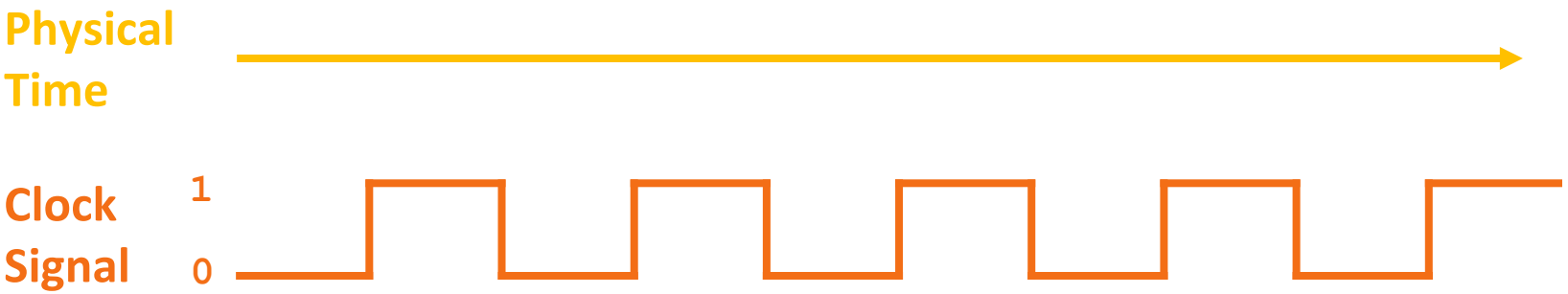
Autopilot Control Circuit Example





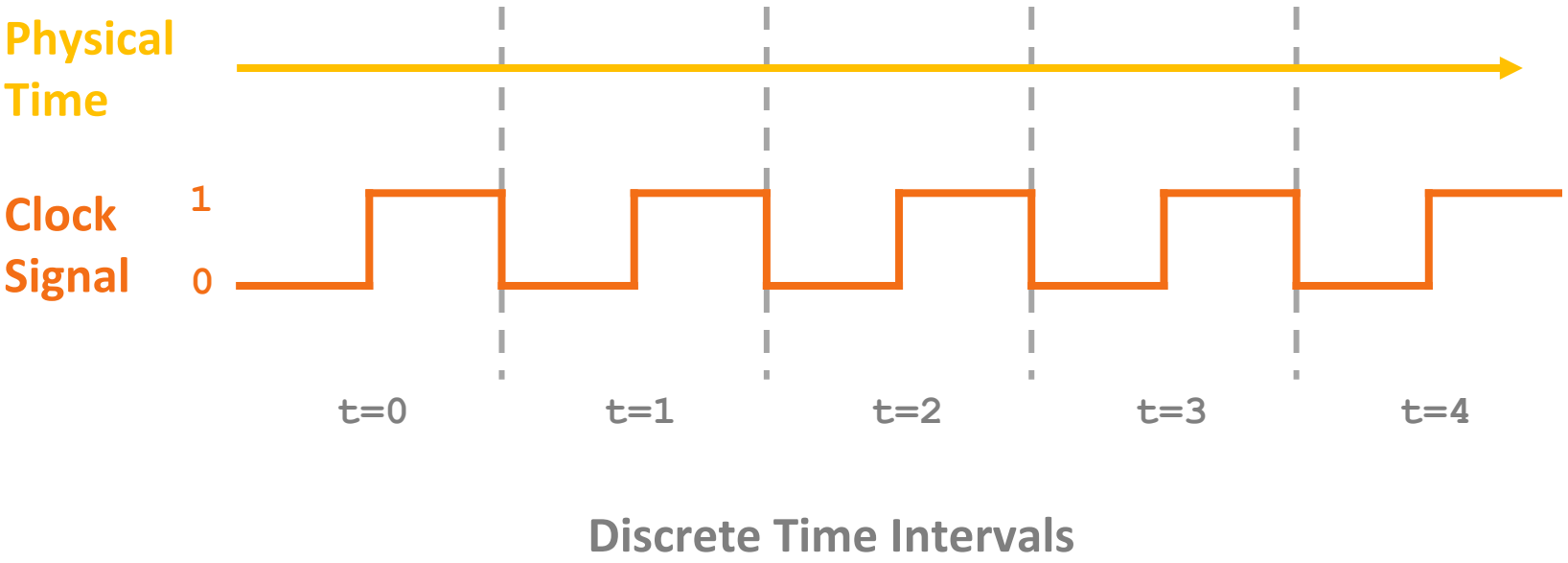
Physical Timekeeping

- 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

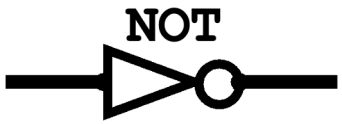


Physical Timekeeping

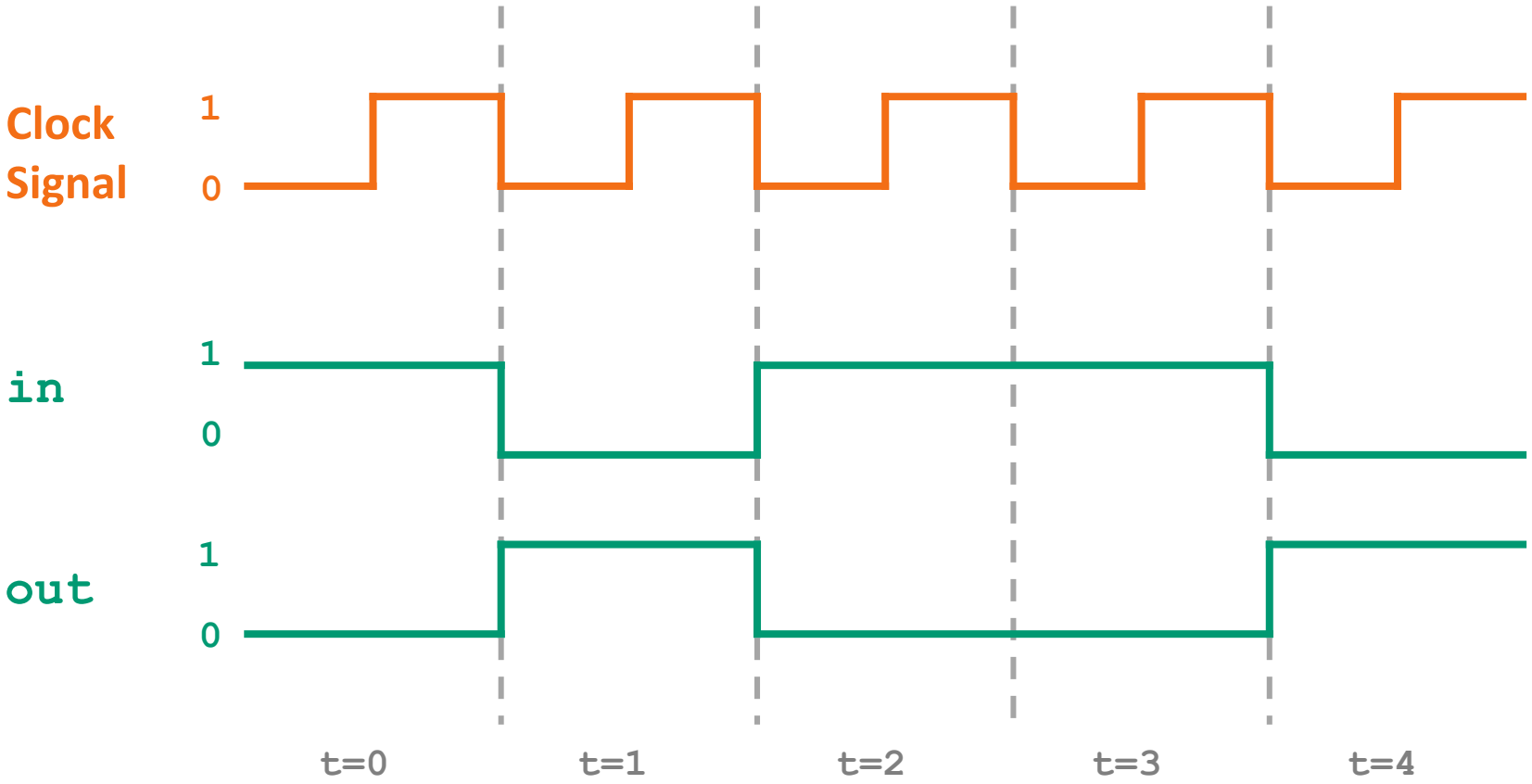
- Hardware keeps track of time using an alternating signal
  - Creates the idea of **discrete time:** state changes only occur in discrete intervals, right when signal alternates



Adding a Clock: Ideal

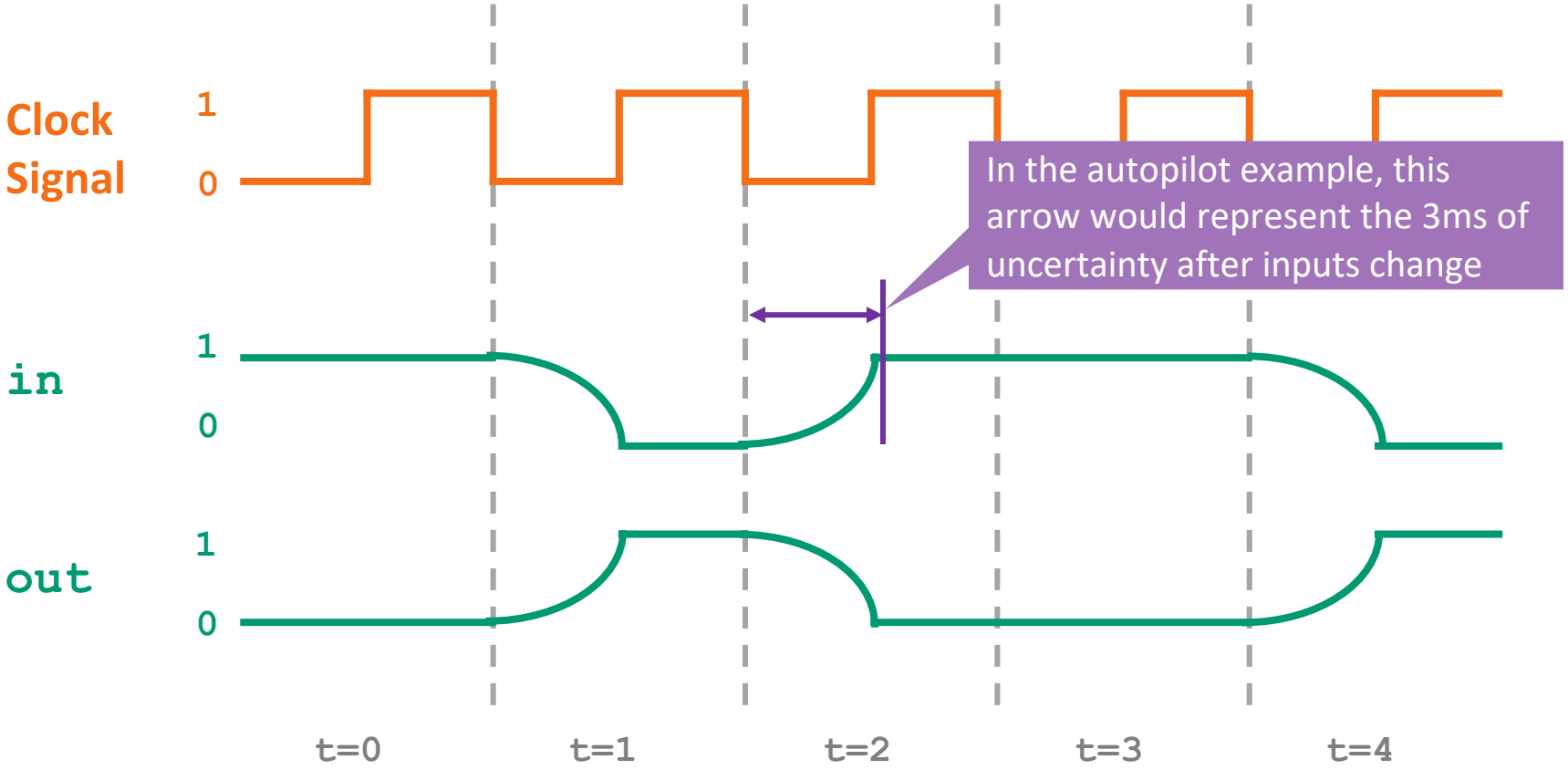


- We want this behavior from a simple, combinational Not gate:



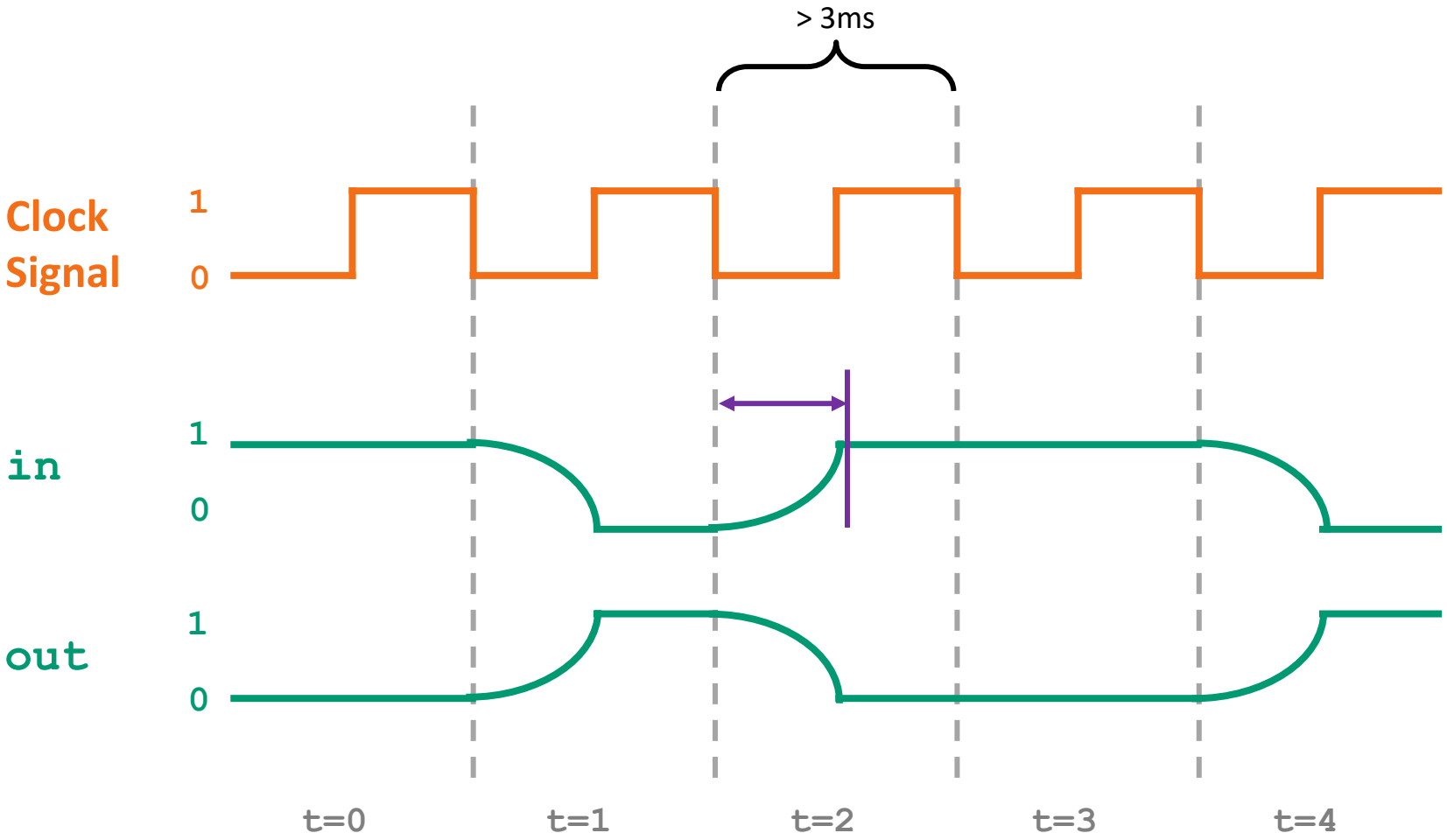
Adding a Clock: Reality

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



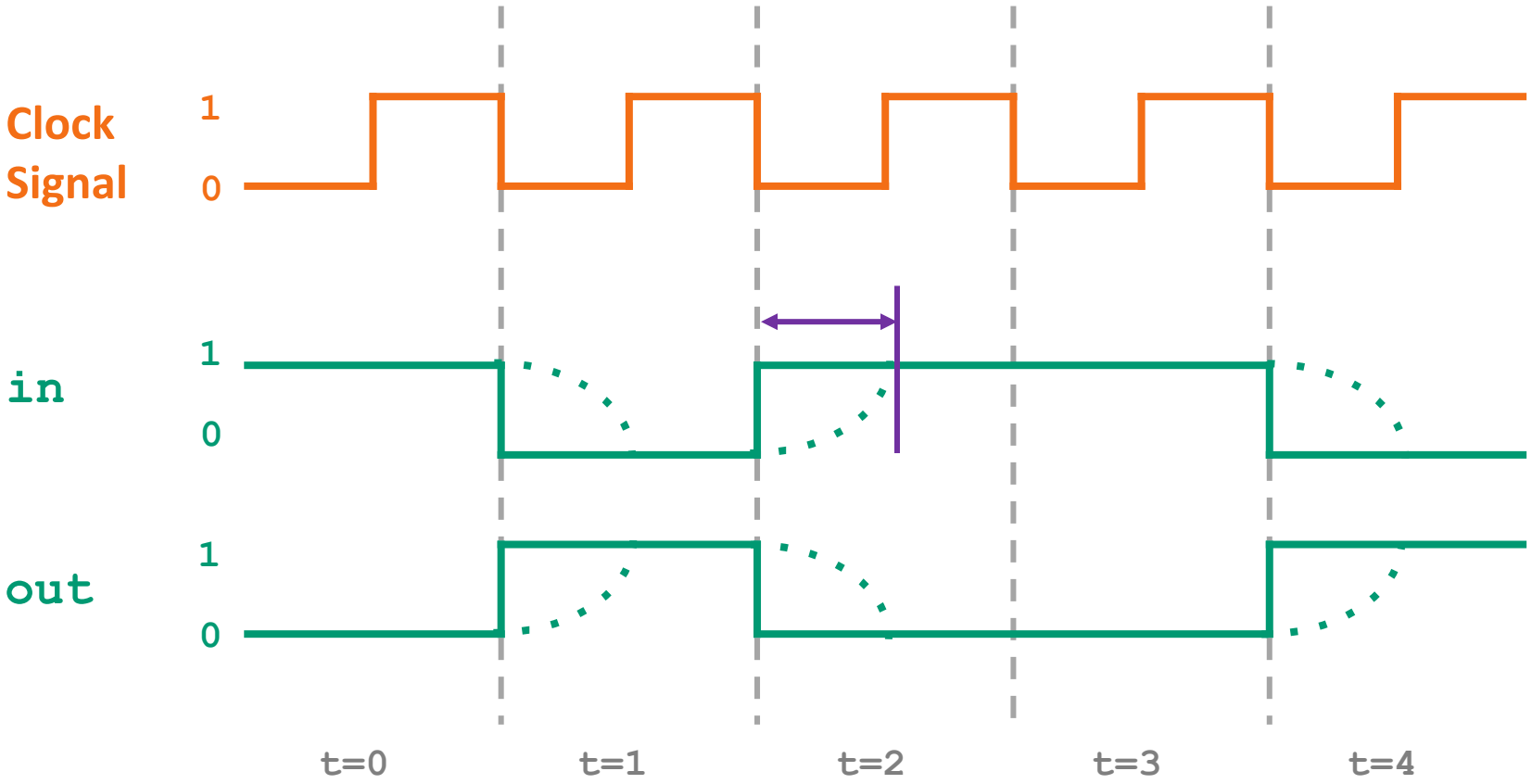
Adding a Clock: Clock Cycles

- 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



Five-minute Break!

- Feel free to stand up, stretch, use the restroom, drink some water, review your notes, or ask questions
- We'll be back at:
- Any song recommendations? Respond on Poll Everywhere at https://pollev.com/cse390b
- Research shows mid-lecture breaks reduce the decline of attention in the middle of lecture (Olmsted, 1999)

Olmsted III, John. "The Mid-Lecture Break: When Less Is More." *Journal of Chemical Education* (1999). https://pubs.acs.org/doi/abs/10.1021/ed076p525.

Poll Everywhere

Vote at https://pollev.com/cse390b

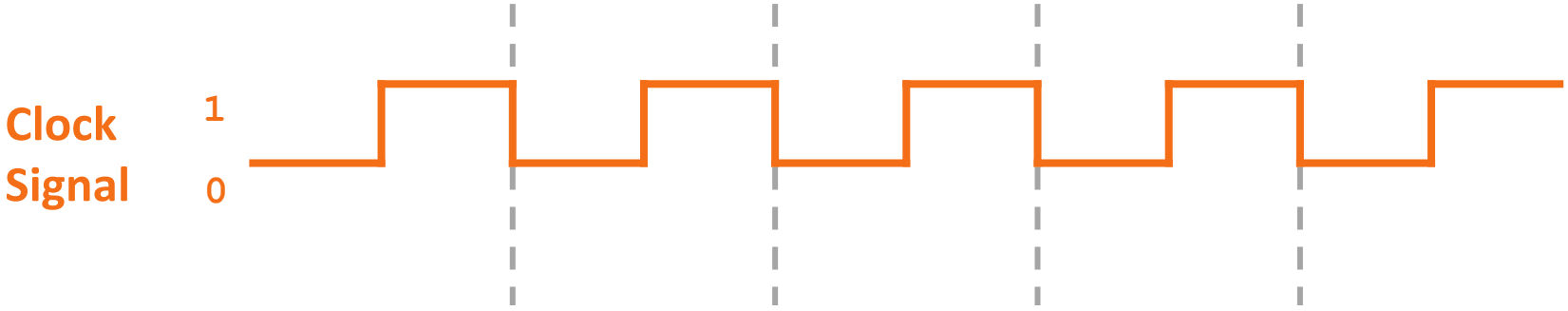
When is a (or are) better time(s) for Eric's Wednesday office hours (previously Wednesdays from 1:30-2:30pm)?

A. Wednesdays, 11:30-12:30pm
B. Wednesdays, 5:30-6:30pm
C. Thursdays, 11:30-12:30pm
D. Thursdays, 12:30-1:30pm
E. The time is good as is (Wednesdays, 1:30-2:30pm)

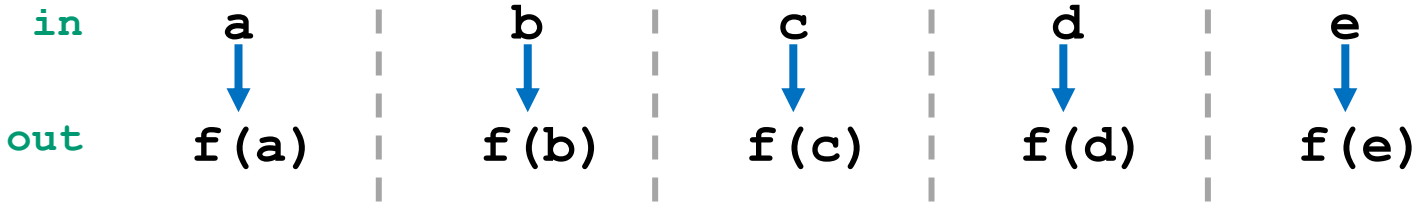
Lecture Outline

- Bloom's Taxonomy
  - Applying Higher Levels of Cognition to Learning
- Cornell Note Taking Method
  - System for Taking, Organizing, and Reviewing Notes
- Representing Time in Hardware
  - Sequential Logic vs. Combinational Logic
  - Adding a Clock
- **Sequential Logic**
  - **Data Flip-Flop (DFF) Implementation and Examples**

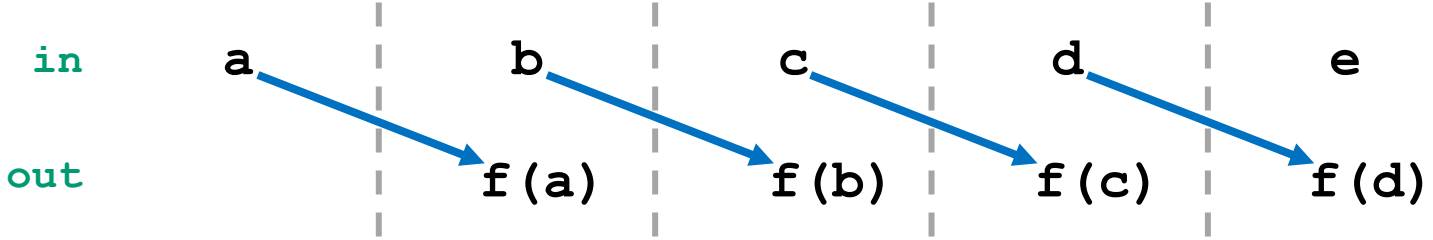
Combinational vs. Sequential Abstraction



Combinational: a function of the current inputs



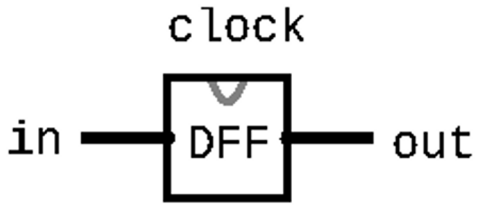
Sequential: a function of previous inputs (has "memory")



The Flip-Flop Gate

- Simplest state-keeping component
  - 1-bit input, 1-bit output
  - Wired to the clock signal
  - Always outputs its previous input: **`out(t) = 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)

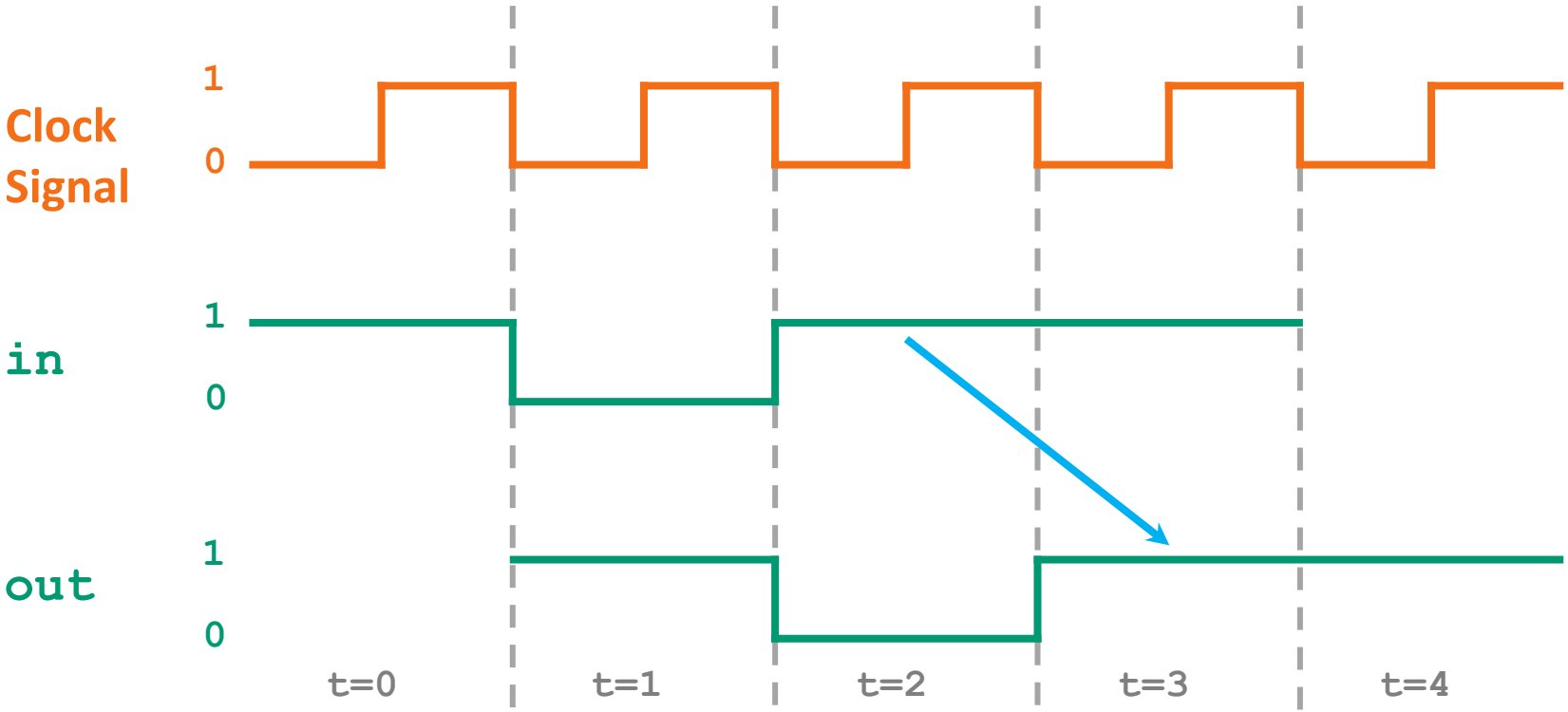


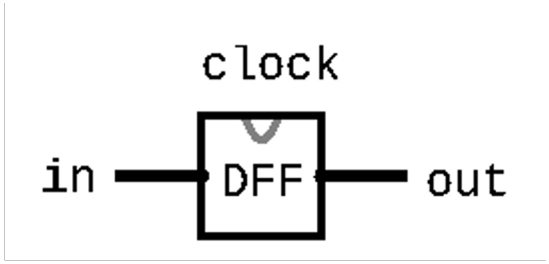
Aside: Treating Flip-Flop as Primitive

- Disclaimer: 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 Flip-Flop as primitive in the projects
  - Just like Nand, you can use the built-in implementation

Flip-Flop Behavior





Sequential Chips

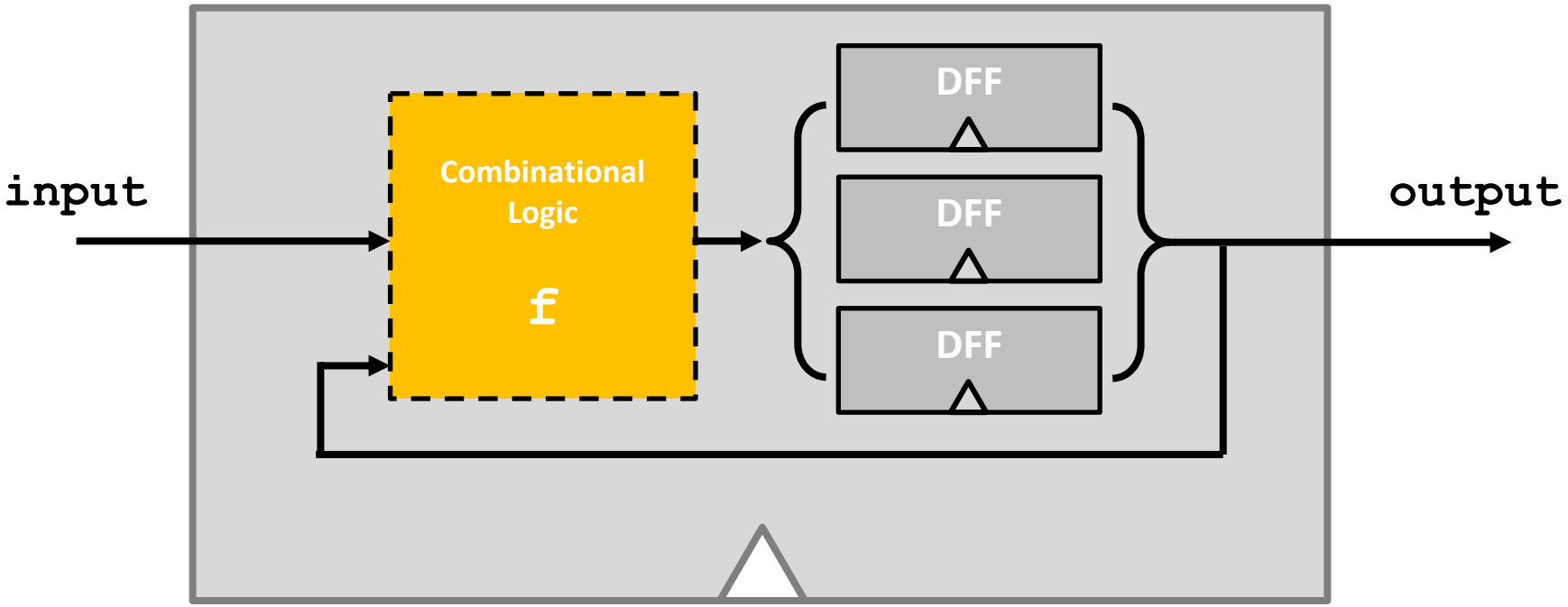
- A category of chips that utilize the clock signal, in addition to any combinational logic

- Capable of:
  - Maintaining state
  - Optionally, acting on that state and the current inputs
    - Can incorporate combinational logic as well

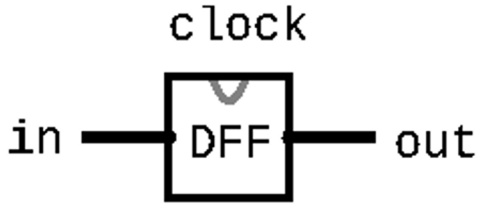
- Constructed from:
  - DFFs
  - Combinational logic (which is entirely constructed from Nand)

Sequential Chips

```
output(t) = f(state(t-1), input(t))
```



Flip-Flop: Time Series



- DFF Specification:

```
out(t) = in(t-1)
```

in	0	0	1	1	0	1	0	...
out	0	0	0	1	1	0	1	...
time	t=0	t=1	t=2	t=3	t=4	t=5	t=6	...

Example: `out(t=3) = in(t=2)`

DFF Example 1: Specification

- Example specification:

```
out(t) = Xor(a(t-1), b(t-1))
```

- Takes two inputs, `a` and `b`, and outputs the `Xor` of them
  - Note that out at time `t` is determined by `a` and `b` at time `t-1`
  - We will need to use a DFF

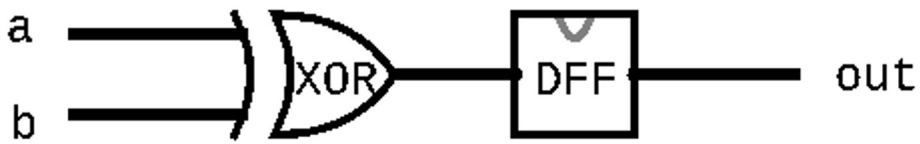
- Exercise: Draw out the corresponding circuit diagram

DFF Example 1: Circuit Diagram

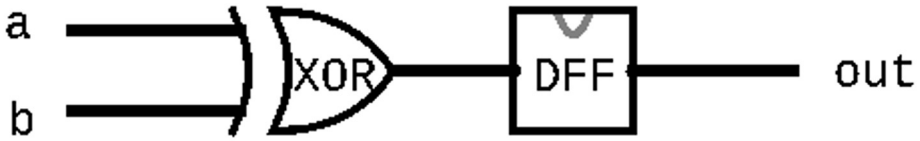
- Example specification:

```
out(t) = Xor(a(t-1), b(t-1))
```

- Circuit diagram:



DFF Example 1: HDL Implementation

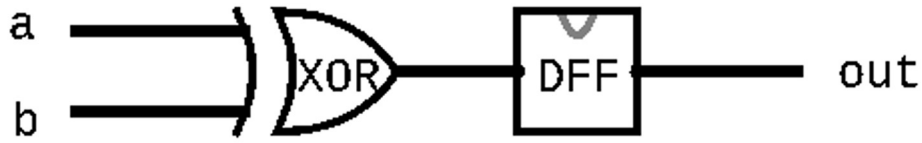



```
CHIP Example1 {
    IN a, b;
    OUT out;

    PARTS:
    Xor(a=a, b=b, out=xorout);
    DFF(in=xorout, out=out);
}
```

DFF Example 1: Time Series

- Example specification:




```
out(t) = Xor(a(t-1), b(t-1))
```

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

Example: `out(t=3) = Xor(a(t=2), b(t=2))`

DFF Example 2: Specification

- Example specification:

```
out(t) = Xor(out(t-1), in(t-1))
```

- Notice how the specification uses `out(t-1)` as an input for `out(t)`
  - Implies the necessity of circular wiring, separated by a DFF

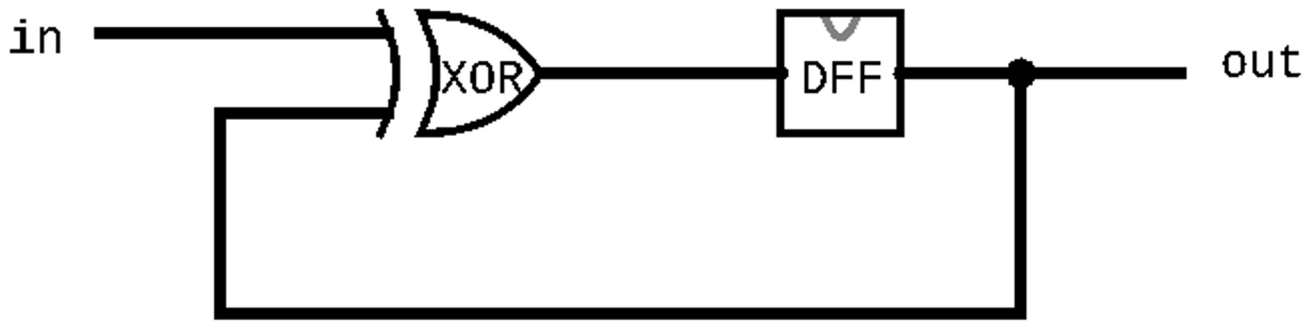
- Exercise: Draw out the corresponding circuit diagram

DFF Example 2: Circuit Diagram

- Example specification:

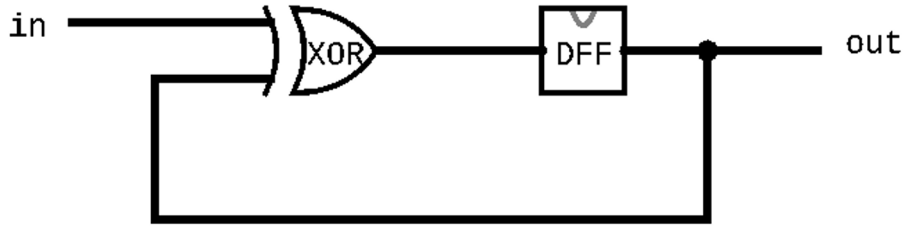
```
out(t) = Xor(out(t-1), in(t-1))
```

- Circuit diagram:



DFF Example 2: HDL Implementation

- Example specification:

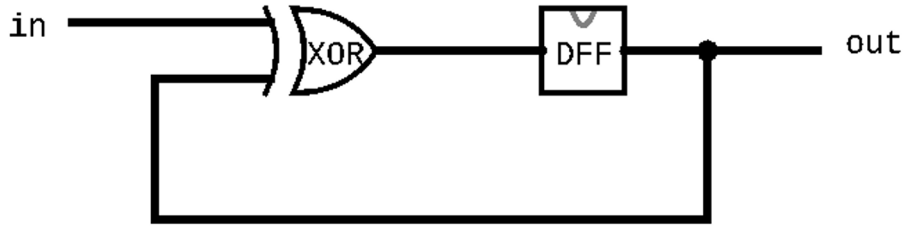



```
out(t) = Xor(out(t-1), in(t-1))
```

- Exercise: Write out the HDL Implementation

DFF Example 2: HDL Implementation

- Example specification:



out(t) = Xor(out(t-1), in(t-1))

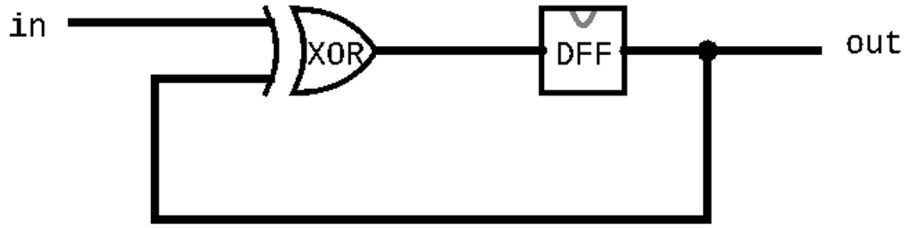
- Exercise: Write out the HDL Implementation

```
CHIP Example2 {
     IN in;
     OUT out;

     PARTS:
     Xor(a=in, b=prevout, out=xorout);
     DFF(in=xorout, out=prevout, out=out);
}
```

DFF Example 2: Time Series

- Example specification:




```
out(t) = Xor(out(t-1), in(t-1))
```

in	0	0	1	1	1	0	0	...
out	0	0	0	1	0	1	1	...
time	t=0	t=1	t=2	t=3	t=4	t=5	t=6	...

Example: `out(t=1) = Xor(in(t=0), out(t=0))`

DFF Example 3: Specification

- Example specification:

```
out(t) = And(Not(out(t-1)), in(t-1))
```

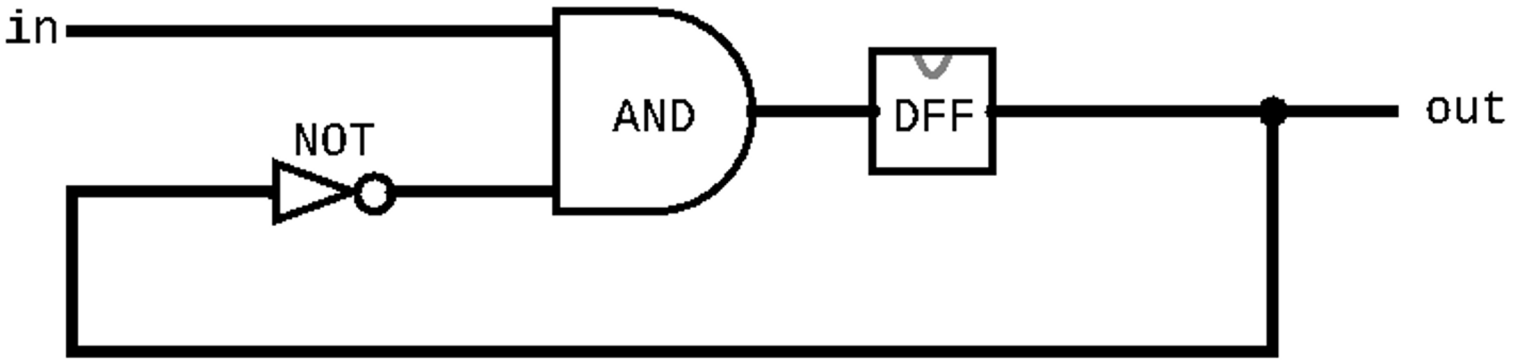
- Exercise: Draw out the corresponding circuit diagram

DFF Example 3: Circuit Diagram

- Example specification:

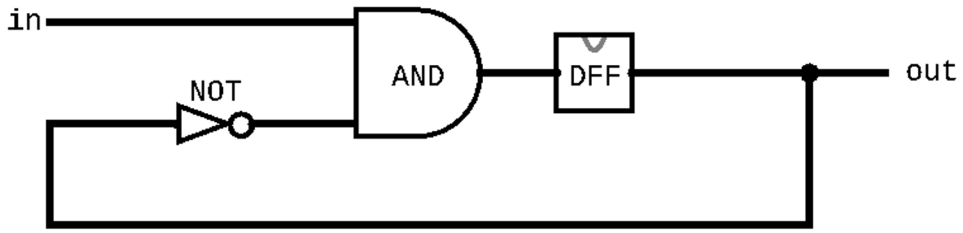
```
out(t) = And(Not(out(t-1)), in(t-1))
```

- Exercise: Draw out the corresponding circuit diagram



DFF Example 3: HDL Implementation

- Example specification:

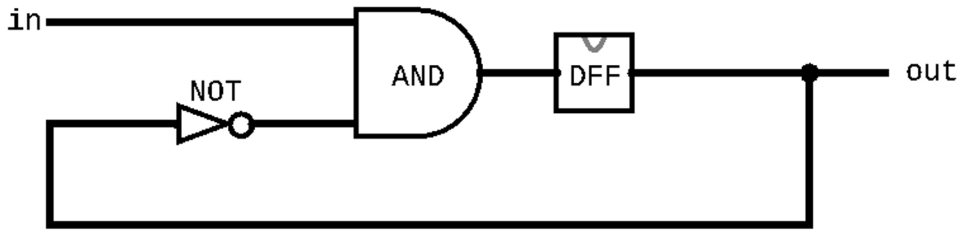



```
out(t) = And(Not(out(t-1)), in(t-1))
```

- Exercise: Write out the HDL Implementation

DFF Example 3: HDL Implementation

- Example specification:



out(t) = And(Not(out(t-1)), in(t-1))

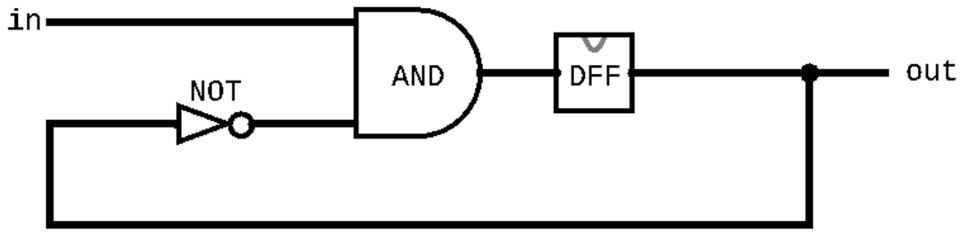
- Exercise: Write out the HDL Implementation

```
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);
}
```

DFF Example 3: Time Series

- Example specification:




```
out(t) = And(Not(out(t-1)), in(t-1))
```

in	1	1	0	1	1	0	0	...
out	0	1	0	0	1	0	0	...
time	t=0	t=1	t=2	t=3	t=4	t=5	t=6	...

Example: `out(t=1) = And(Not(out(t=0)), in(t=0))`

Lecture 5 Wrap-up

- Project Reminders
  - Project 2 grades released by Thursday (feedback can be viewed on Gradescope)
  - **Project 3 due Thursday (4/14) at 11:59pm PDT**

- Lecture 6 Reading: Storing Data Using Memory

- Course staff will create a guide for untagging and Eric will cover the Project 4 overview in greater depth on Thursday