CSE 390B, Autumn 2022

Building Academic Success Through Bottom-Up Computing

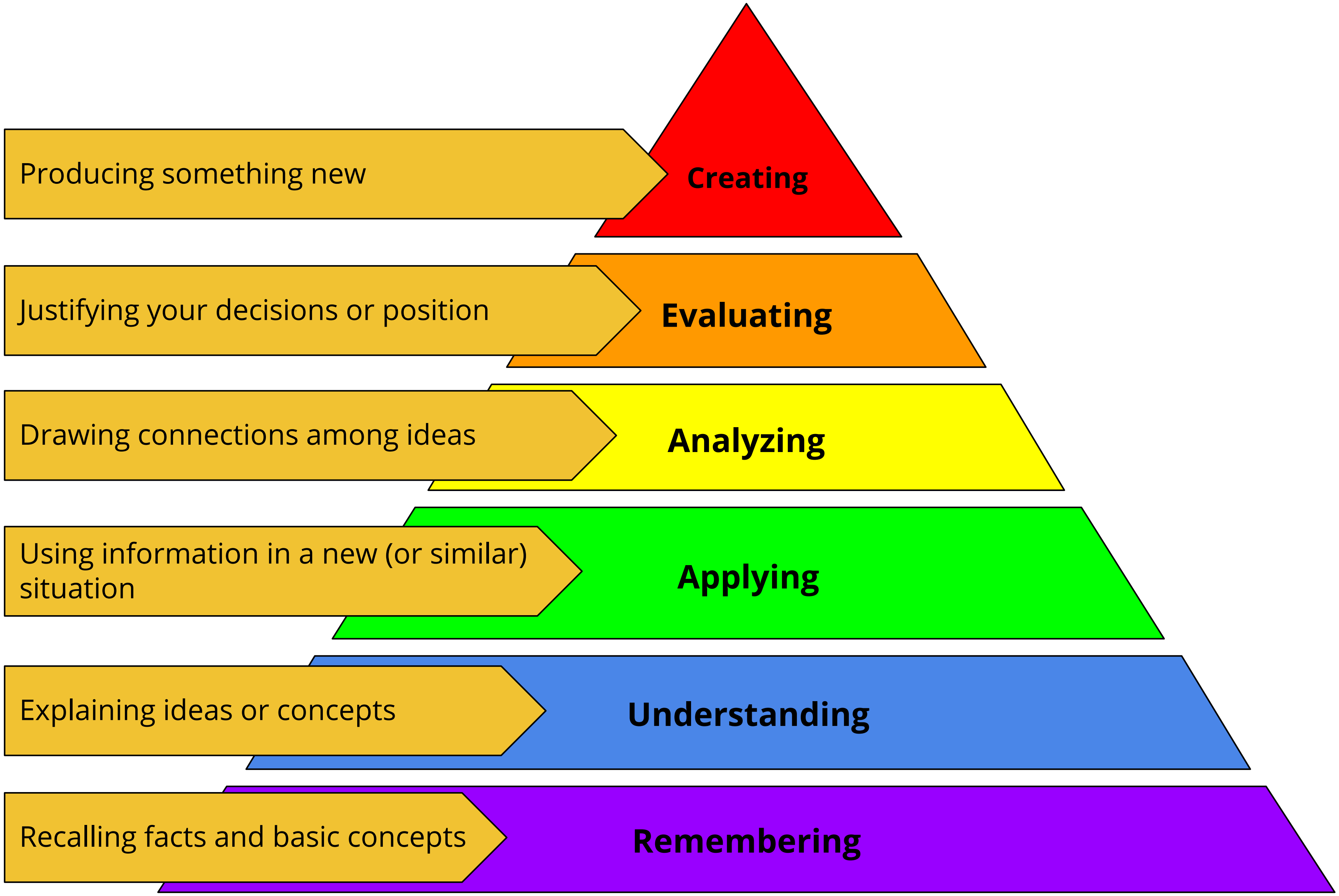
Bloom’s Taxonomy & Sequential Logic

Bloom’s Taxonomy, Introduction to Sequential Logic, Representing Time in Hardware, The Data Flip-Flop (DFF)

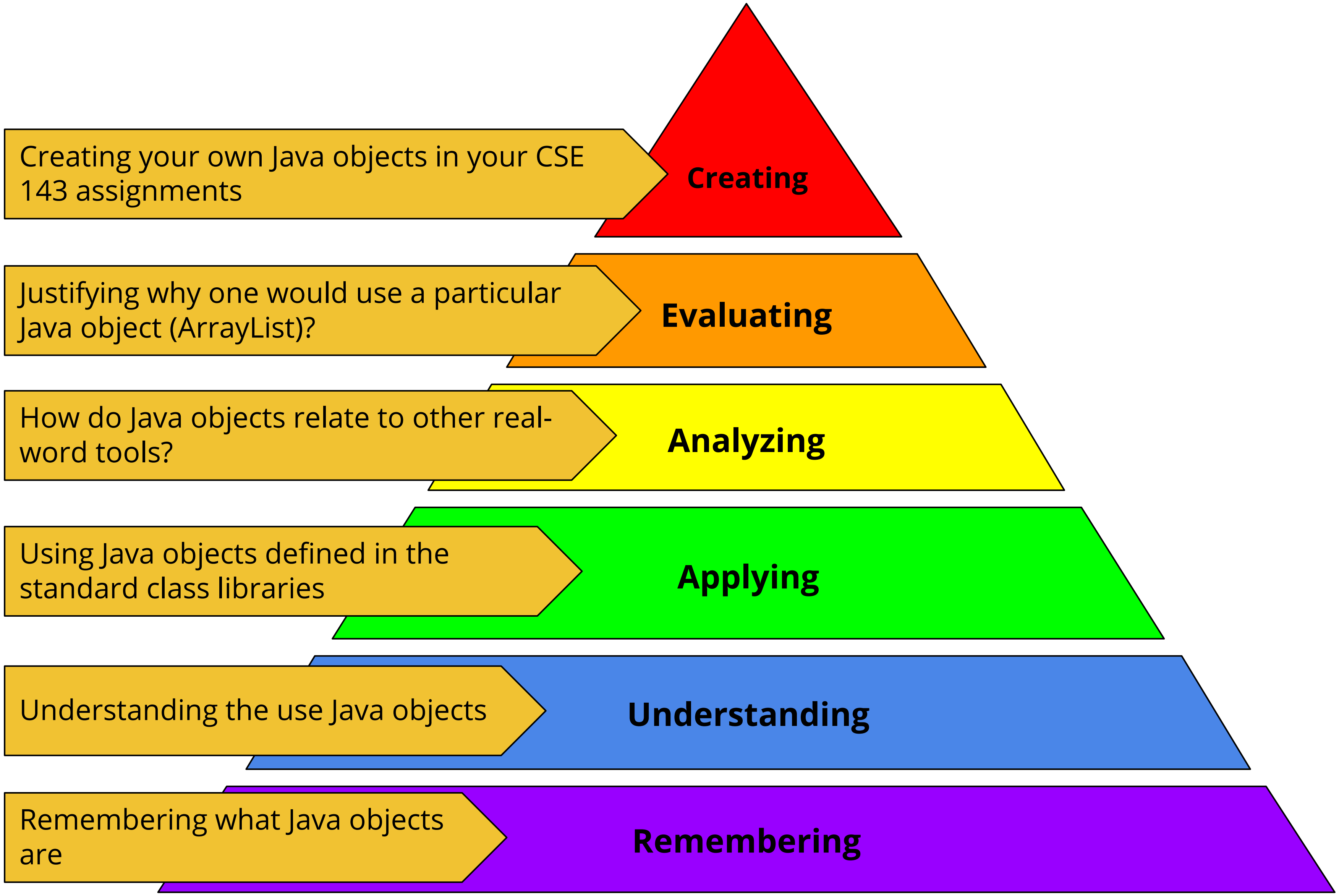
Lecture Outline

- **Bloom's Taxonomy**
  - **Applying Higher Levels of Cognition to Learning**
- Introduction to Sequential Logic
  - The Problem with Combinational Logic
- Representing Time in Hardware
  - Clock Signals and Units of Time in Hardware
- The Data Flip-Flop (DFF)
  - Implementation and Examples

Bloom's Taxonomy


Producing something new
Creating
Justifying your decisions or position
Evaluating
Drawing connections among ideas
Analyzing
Using information in a new (or similar) situation
Applying
Explaining ideas or concepts
Understanding
Recalling facts and basic concepts
Remembering

Bloom's Taxonomy in Action

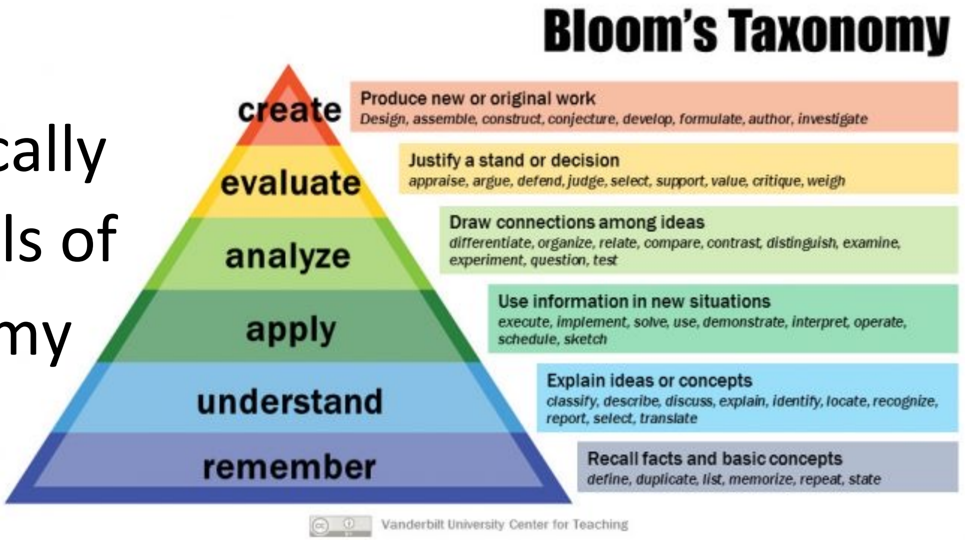

Creating your own Java objects in your CSE 143 assignments
Creating
Justifying why one would use a particular Java object (ArrayList)?
Evaluating
How do Java objects relate to other real-word tools?
Analyzing
Using Java objects defined in the standard class libraries
Applying
Understanding the use Java objects
Understanding
Remembering what Java objects are
Remembering

Bloom's Taxonomy Discussion

In groups of 3-4, discuss the following points:

- Identify various aspects of your academic responsibilities as a UW student (e.g., attending lecture) and categorize them in one of the levels on Bloom's Taxonomy

- Discuss how you can practically engage in higher-order levels of thinking in Bloom's Taxonomy

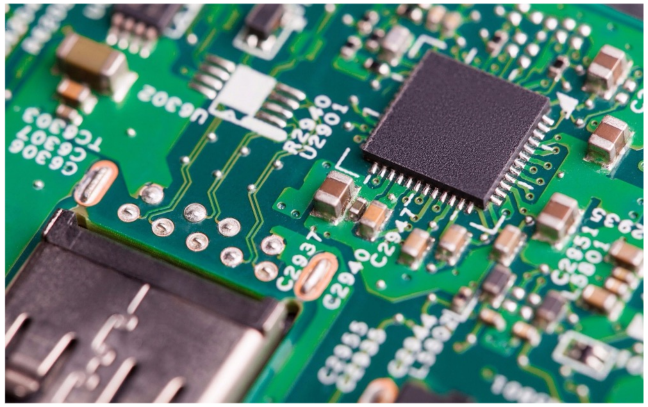

Bloom's Taxonomy
create
Produce new or original work
Design, assemble, construct, conjecture, develop, formulate, author, investigate
evaluate
Justify a stand or decision
appraise, argue, defend, judge, select, support, value, critique, weigh
analyze
Draw connections among ideas
differentiate, organize, relate, compare, contrast, distinguish, examine, experiment, question, test
apply
Use information in new situations
execute, implement, solve, use, demonstrate, interpret, operate, schedule, sketch
understand
Explain ideas or concepts
classify, describe, discuss, explain, identify, locate, recognize, report, select, translate
remember
Recall facts and basic concepts
define, duplicate, list, memorize, repeat, state
Vanderbilt University Center for Teaching

Lecture Outline

- Bloom’s Taxonomy
  - Applying Higher Levels of Cognition to Learning
- **Introduction to Sequential Logic**
  - **The Problem with Combinational Logic**
- Representing Time in Hardware
  - Clock Signals and Units of Time in Hardware
- The Data Flip-Flop (DFF)
  - Implementation and Examples

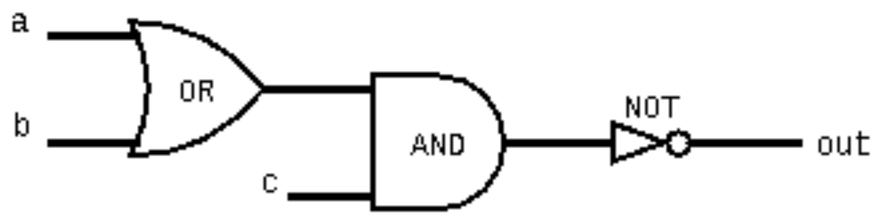
Why Consider Time Now?

- 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:

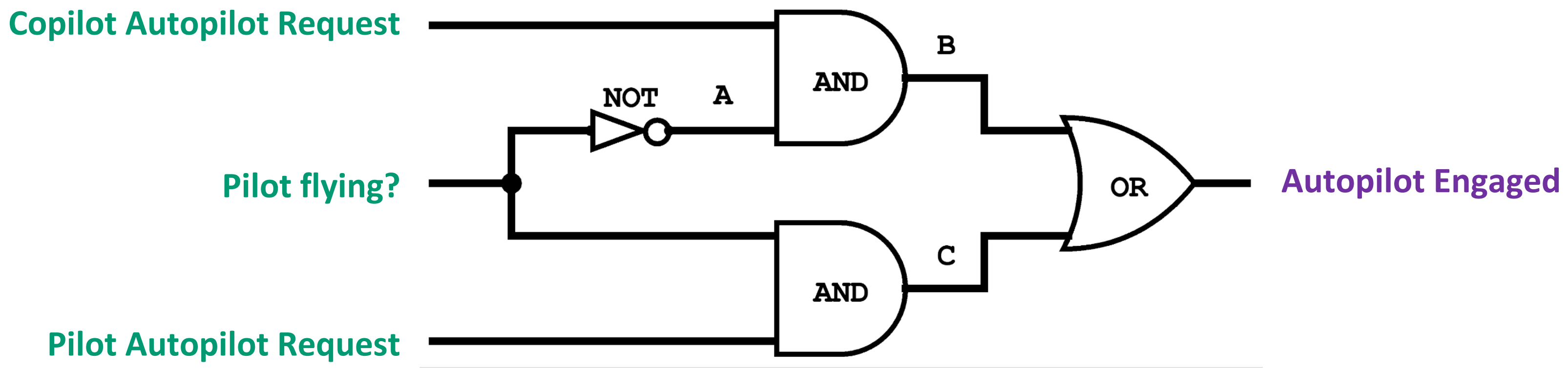


- Let's assume that `a=0`, `b=1`, and `c=0`
  - The output should be 1

- 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

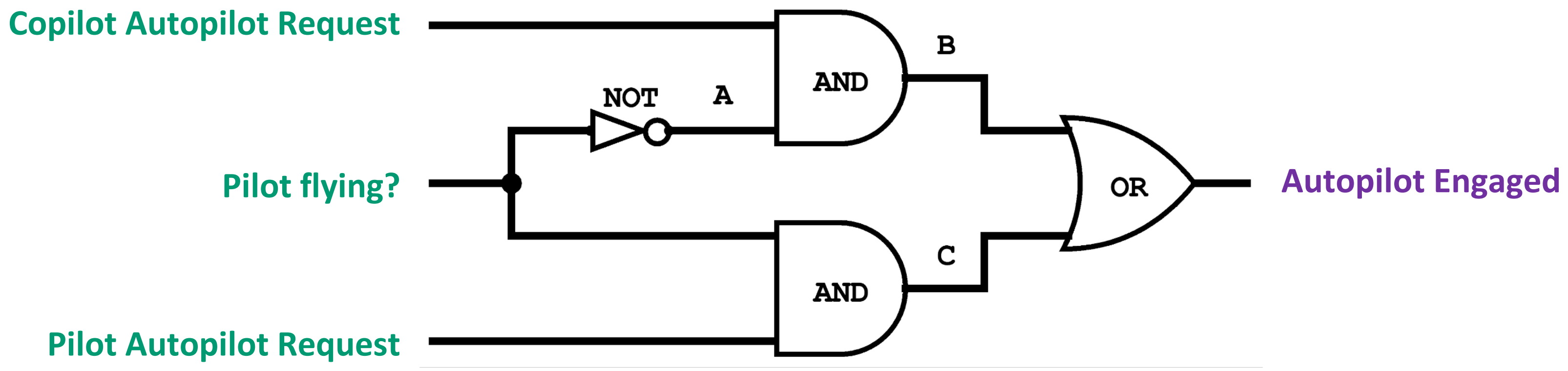
Autopilot Control Circuit Example

- 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



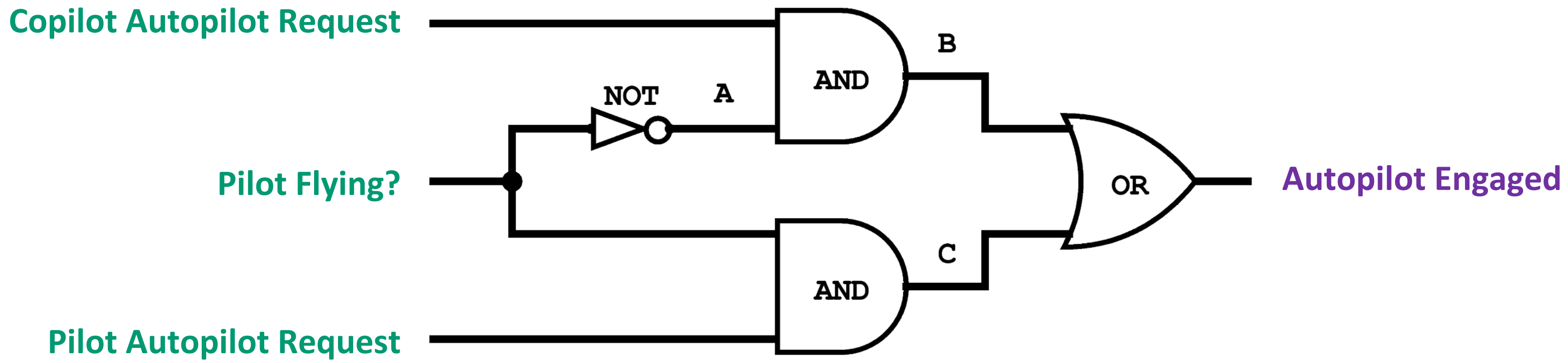
Autopilot Control Circuit Example

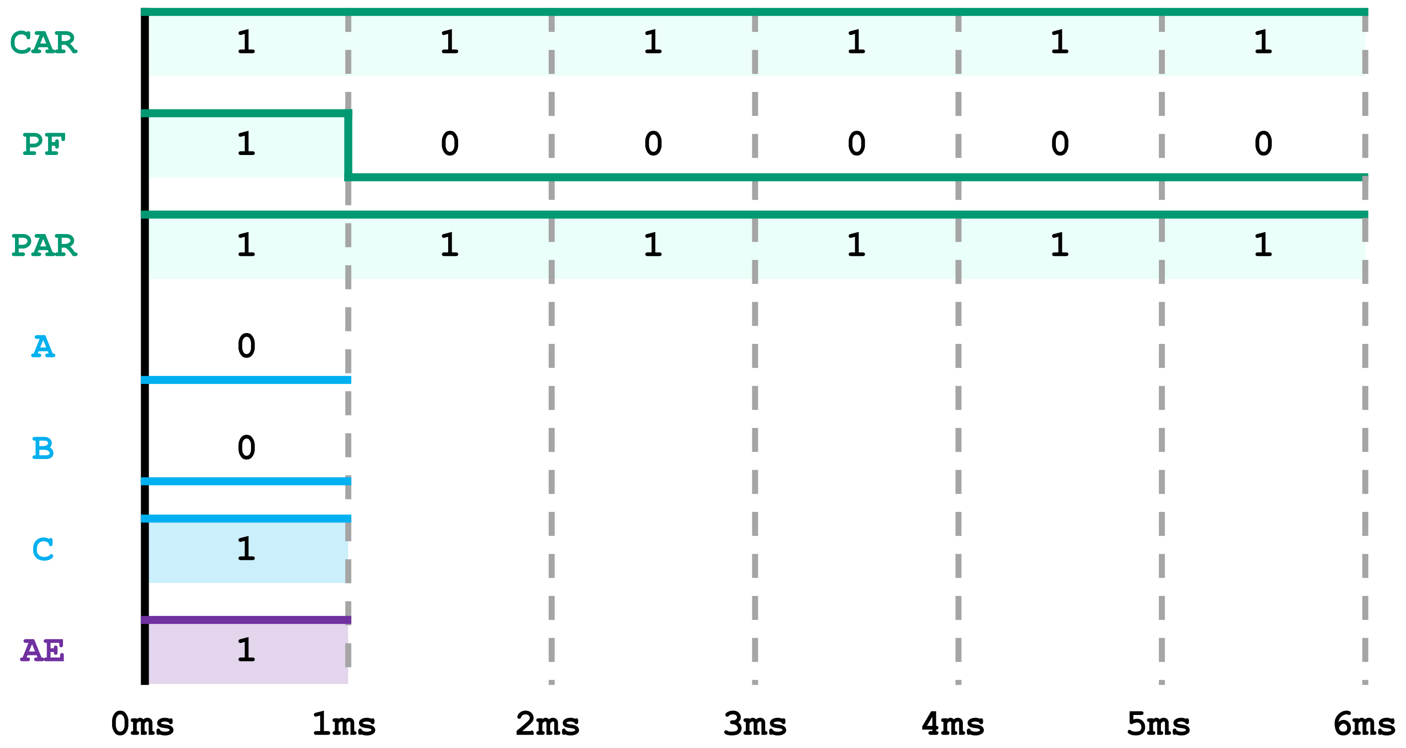
- Consider this autopilot control circuit:
  - Either the pilot or copilot is flying at any time
  - The pilot and copilot can separately request autopilot
  - Only the person flying can request autopilot



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

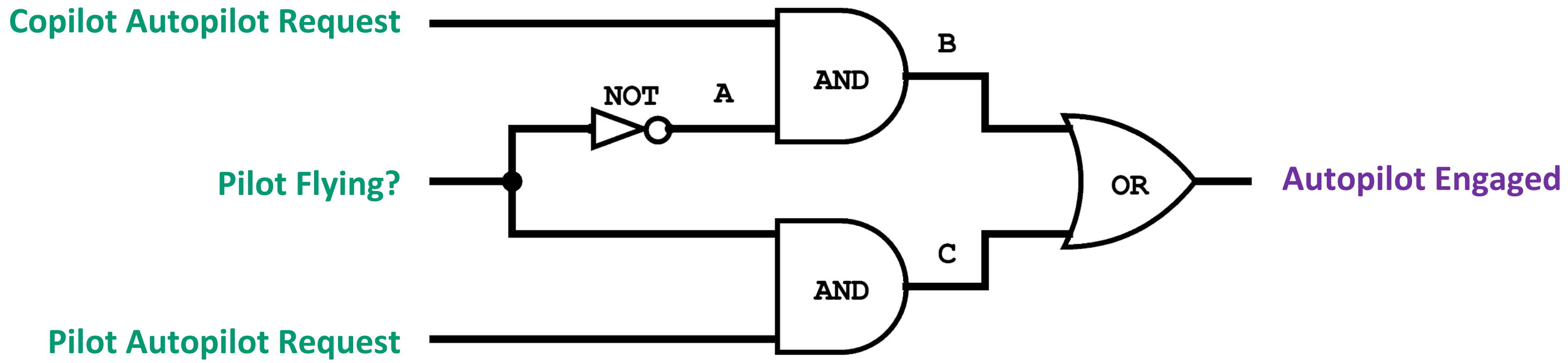
Autopilot Control Circuit Example

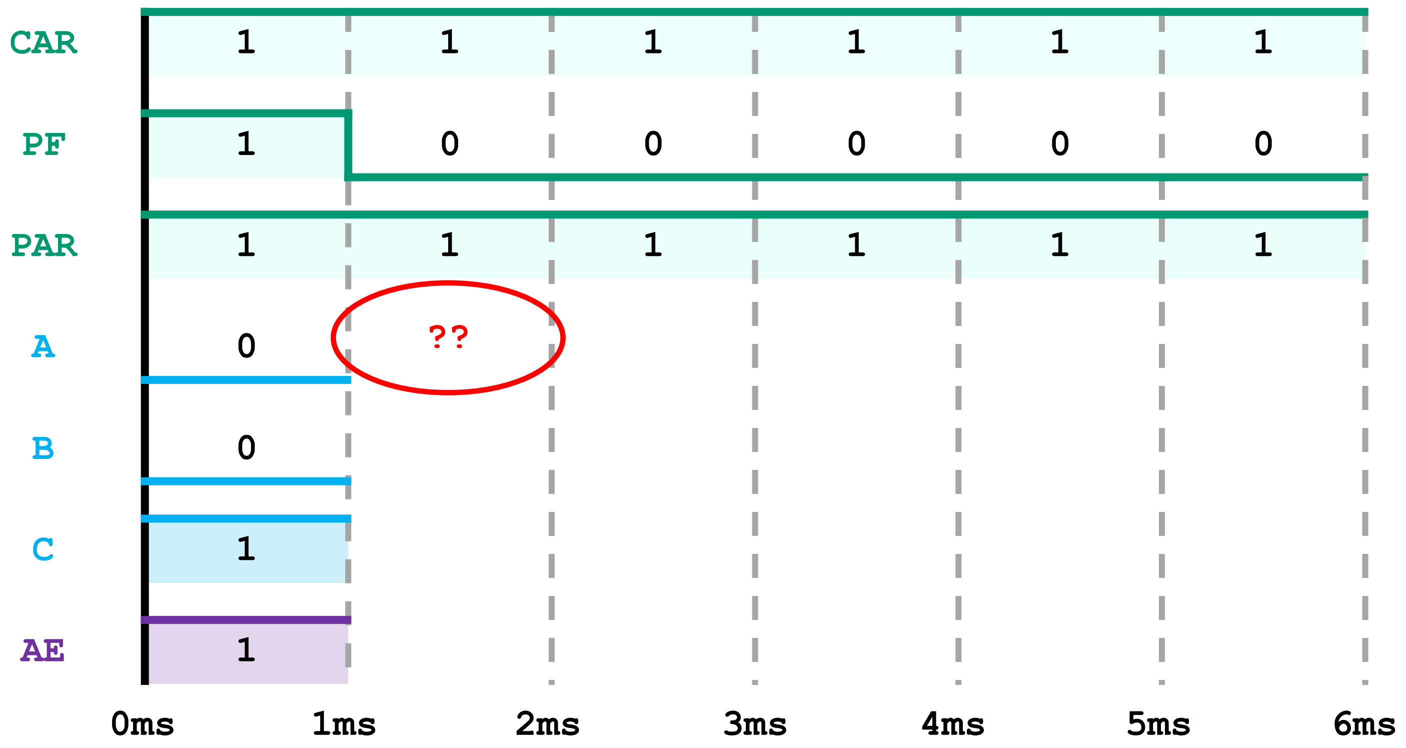




	0–1ms	1–2ms	2–3ms	3–4ms	4–5ms	5–6ms
CAR	1	1	1	1	1	1
PF	1	0	0	0	0	0
PAR	1	1	1	1	1	1
A	0					
B	0					
C	1					
AE	1					

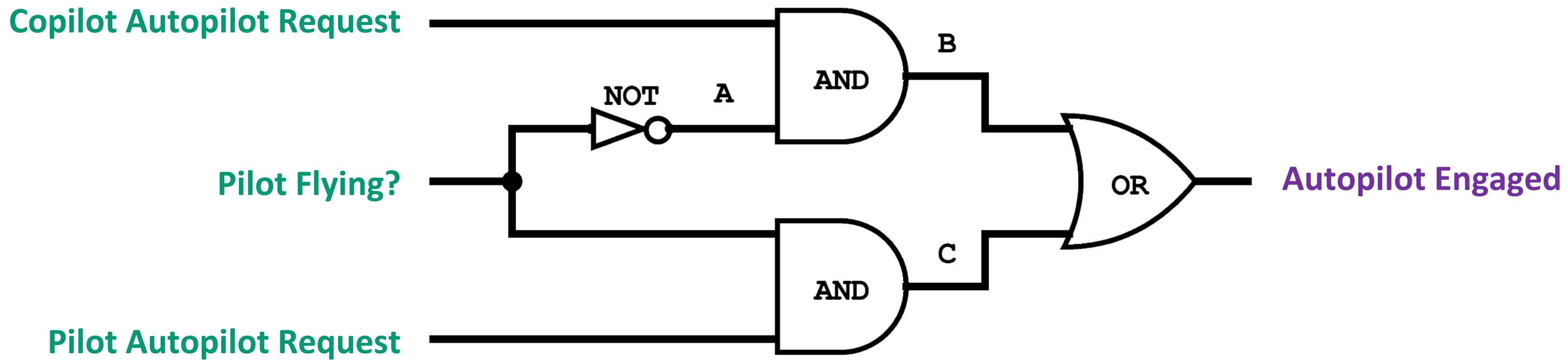
Autopilot Control Circuit Example

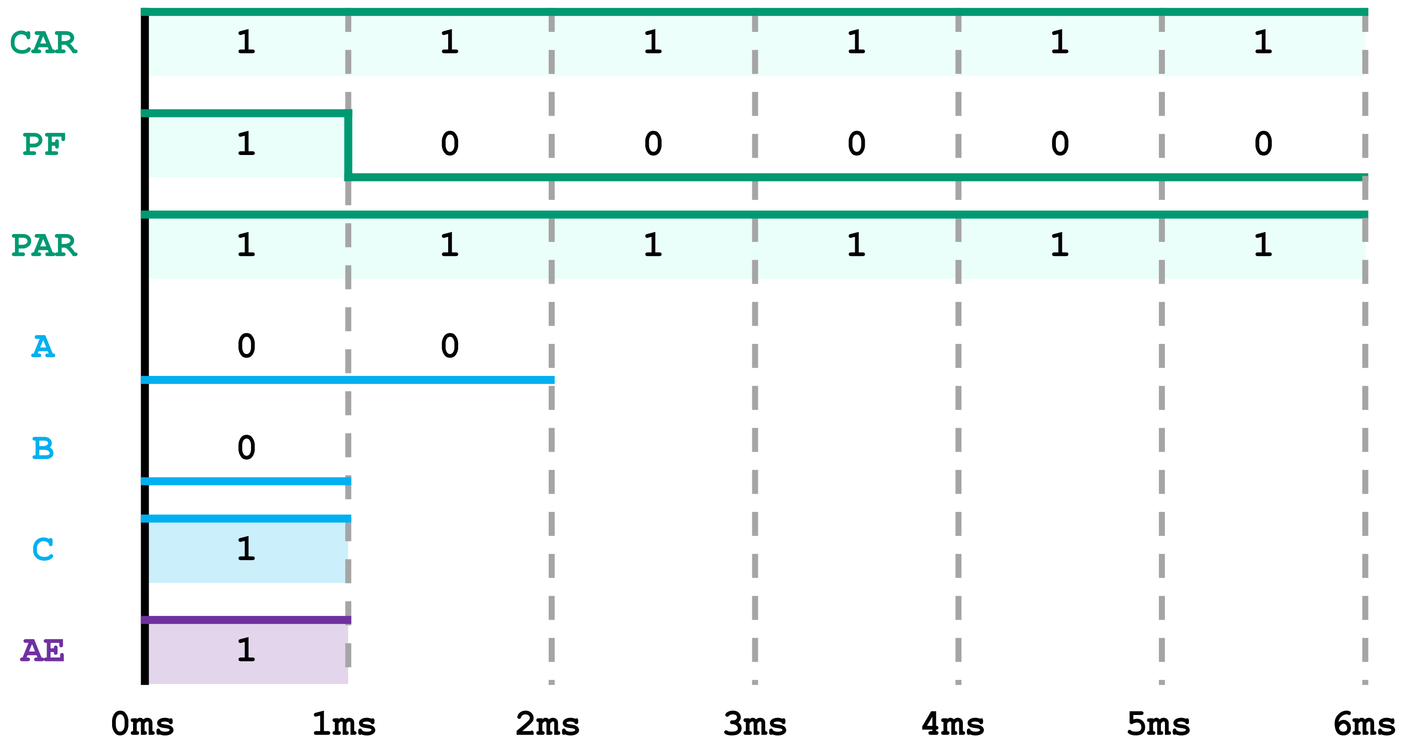




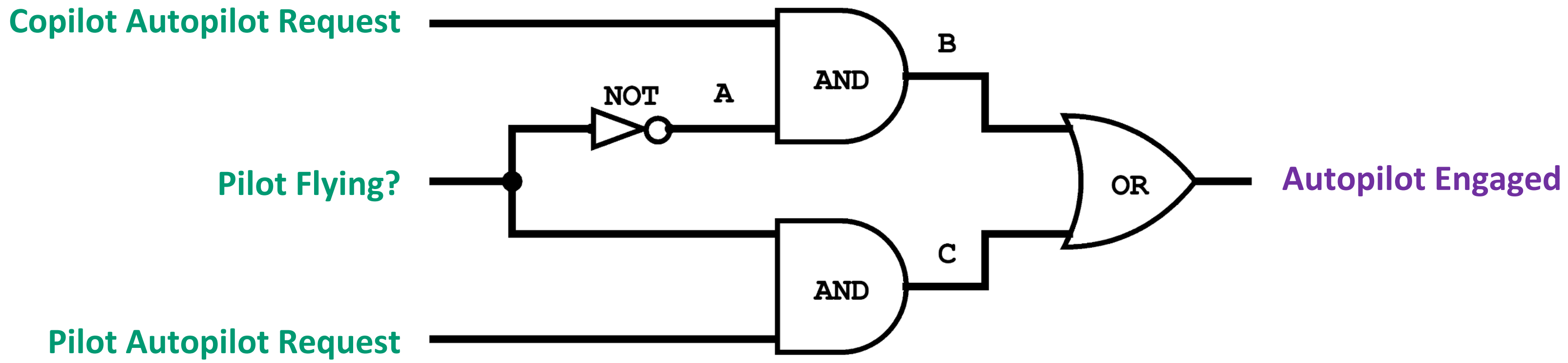
	0–1ms	1–2ms	2–3ms	3–4ms	4–5ms	5–6ms
CAR	1	1	1	1	1	1
PF	1	0	0	0	0	0
PAR	1	1	1	1	1	1
A	0	??				
B	0					
C	1					
AE	1					

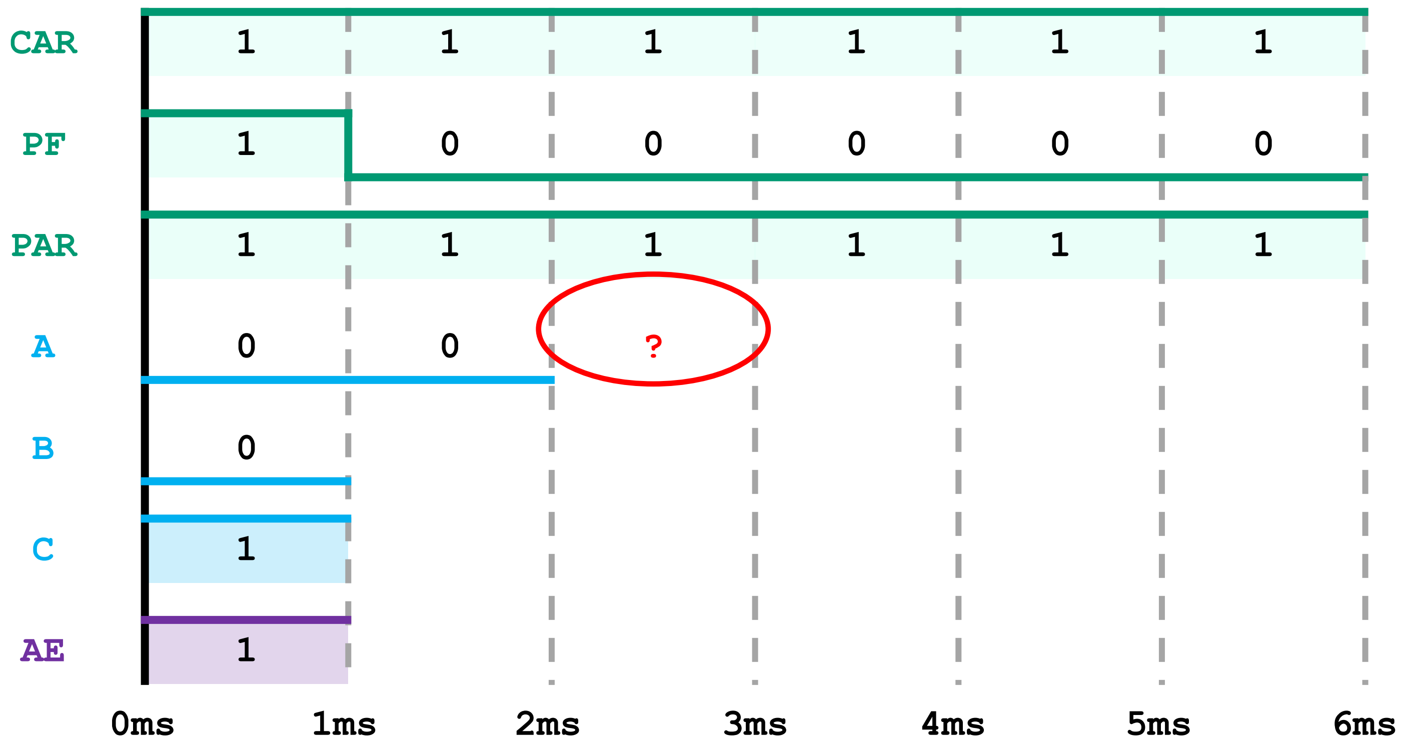
Autopilot Control Circuit Example



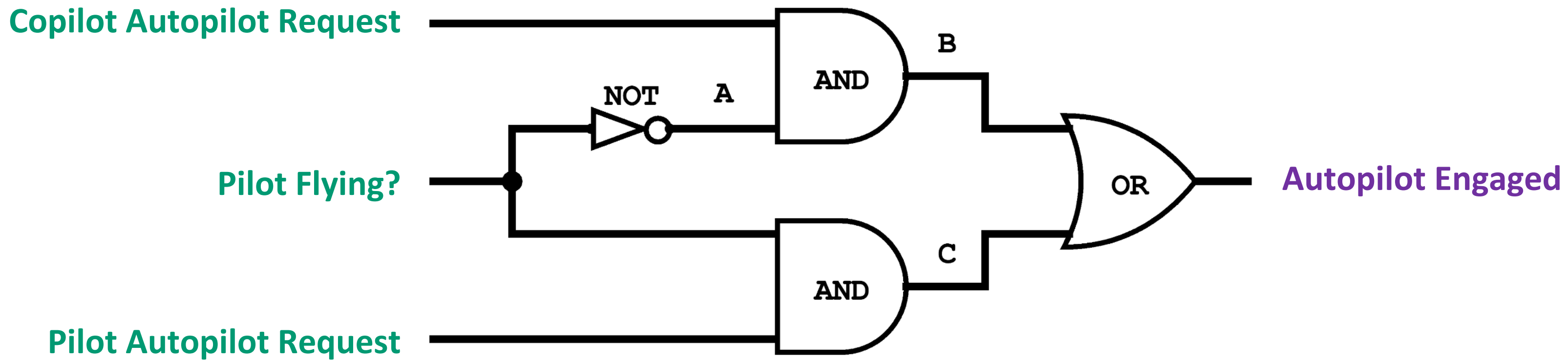


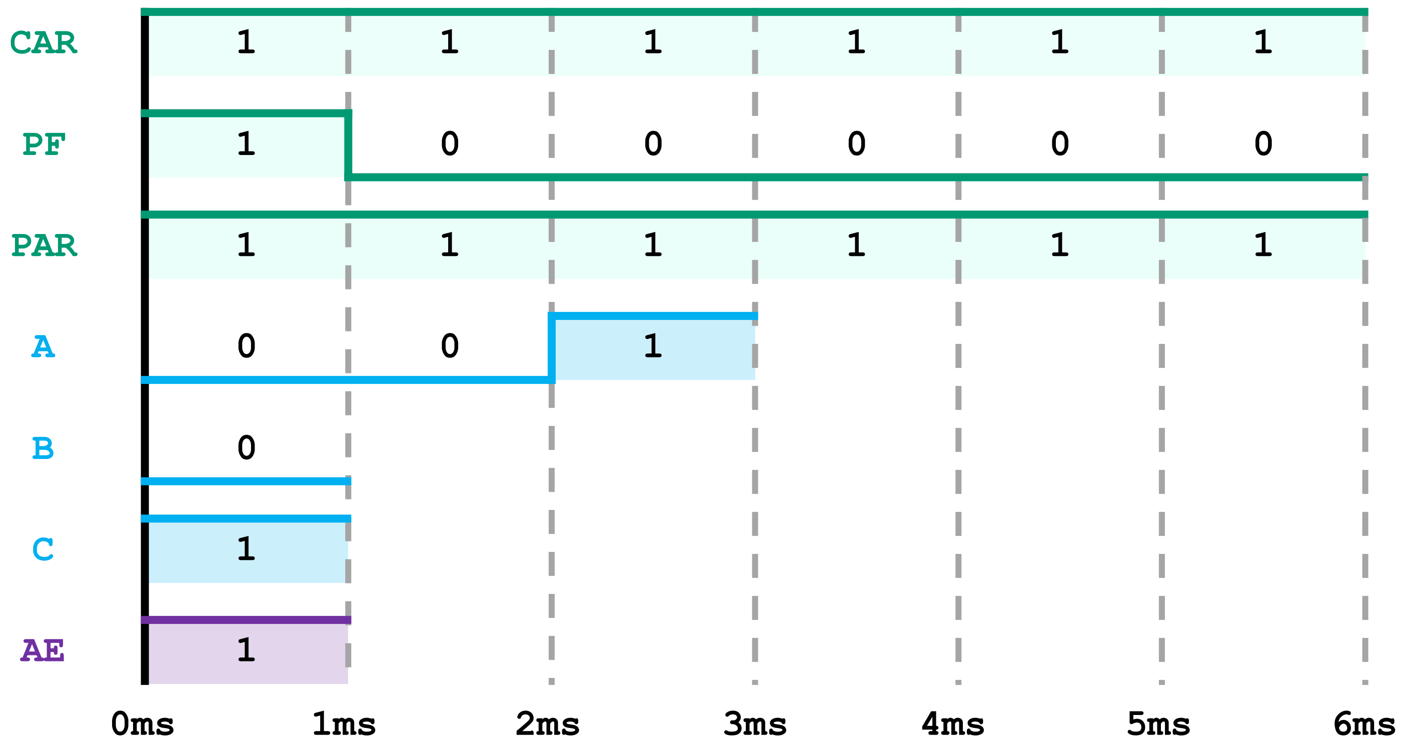
Autopilot Control Circuit Example





Autopilot Control Circuit Example

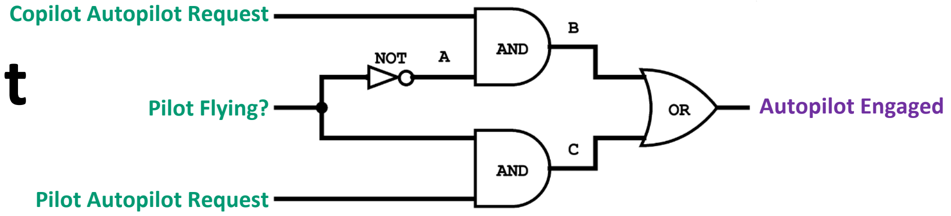


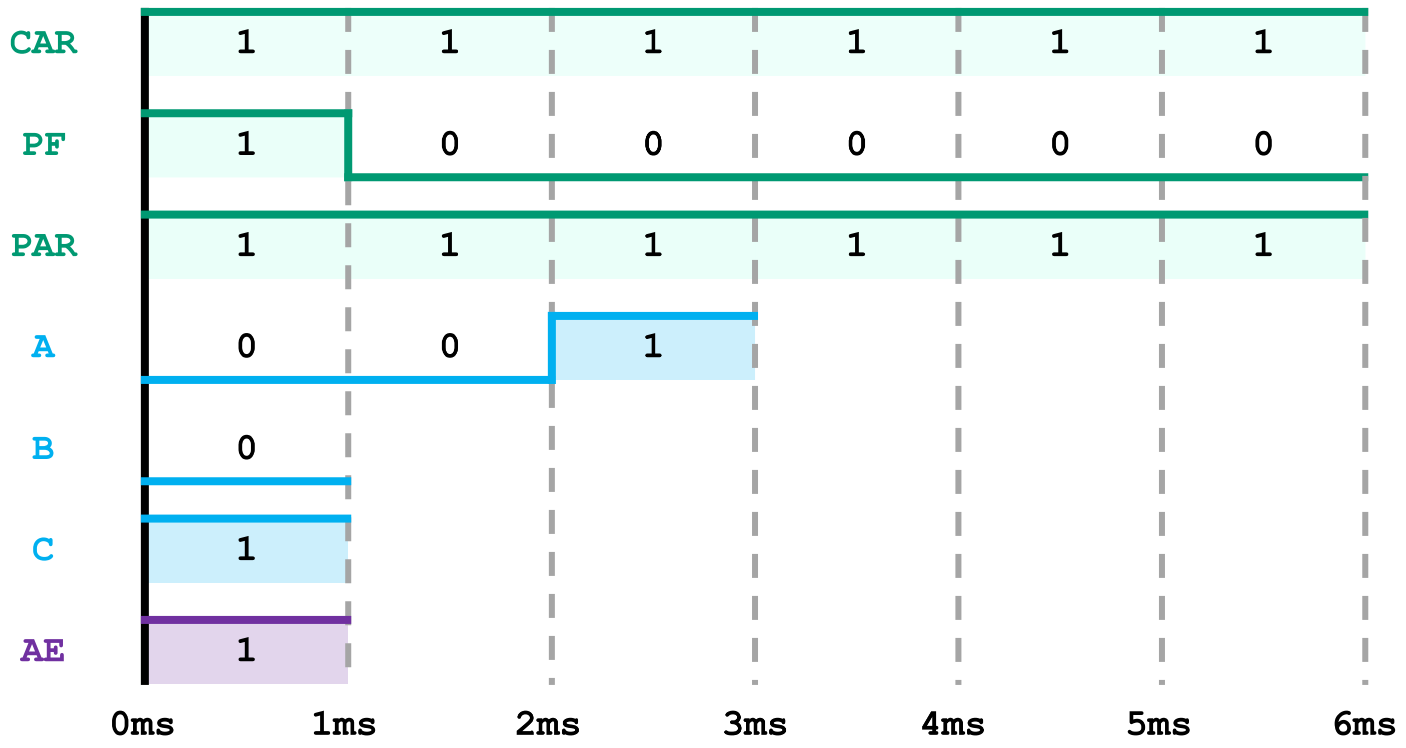


Poll Everywhere

Vote at https://pollev.com/cse390b

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



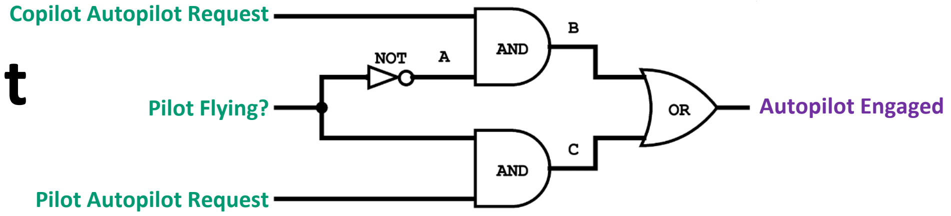


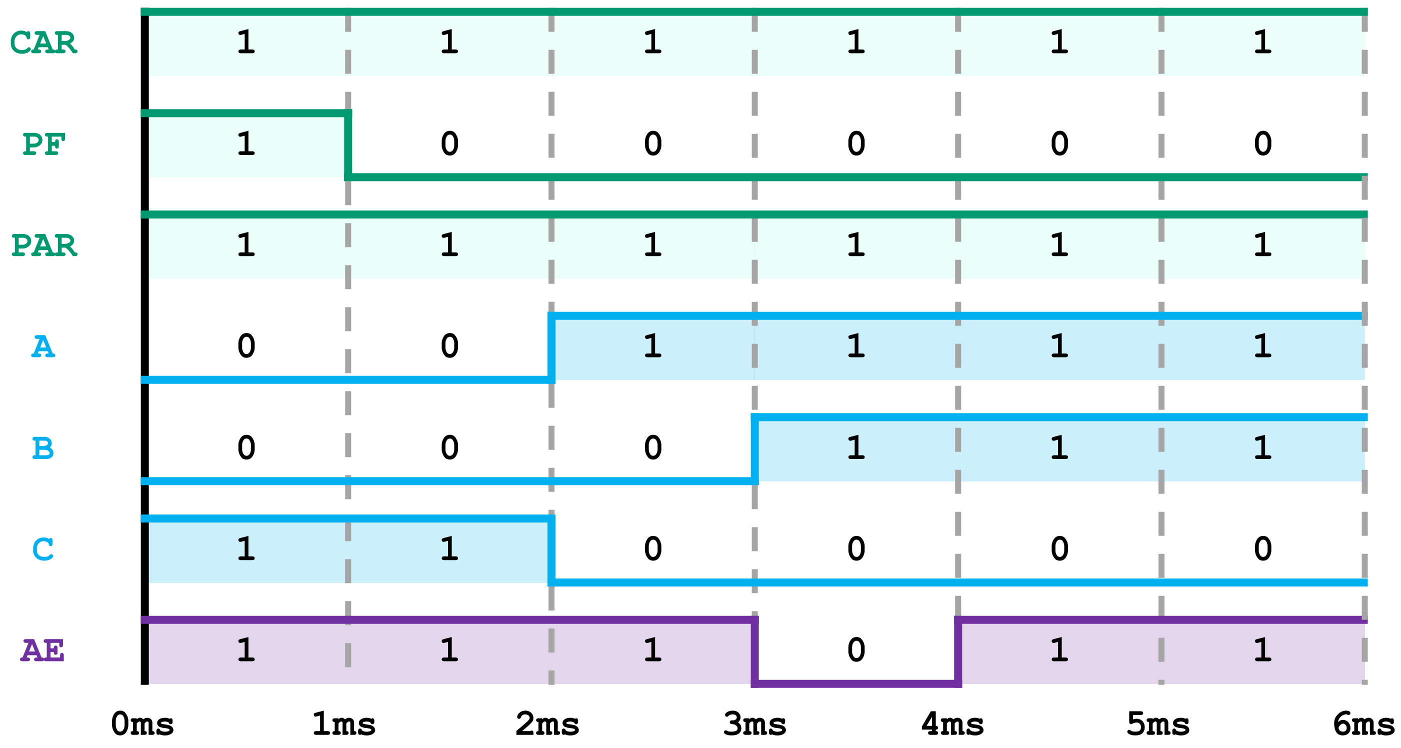
	0–1ms	1–2ms	2–3ms	3–4ms	4–5ms	5–6ms
CAR	1	1	1	1	1	1
PF	1	0	0	0	0	0
PAR	1	1	1	1	1	1
A	0	0	1			
B	0					
C	1					
AE	1					

Poll Everywhere

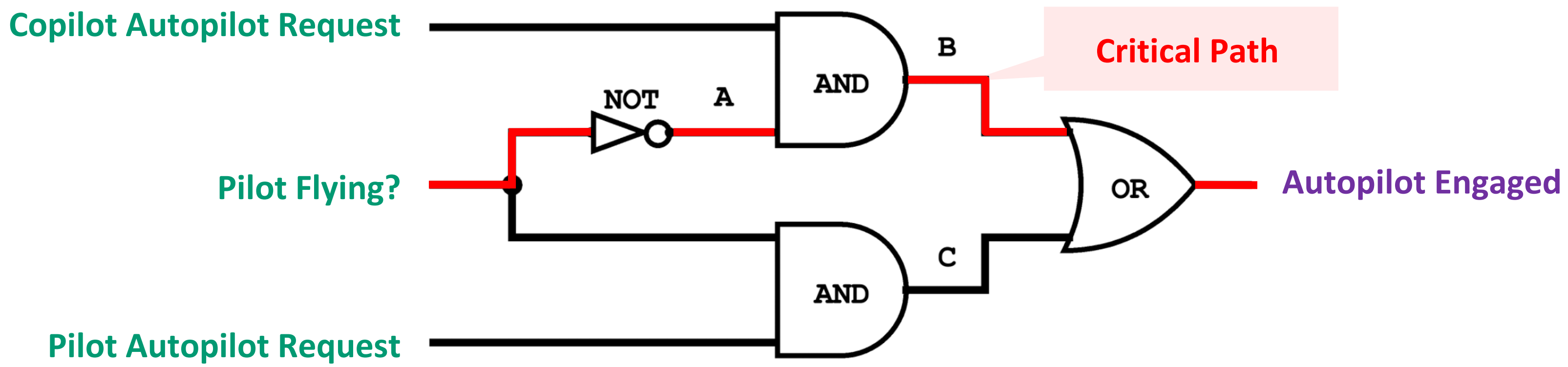
Vote at https://pollev.com/cse390b

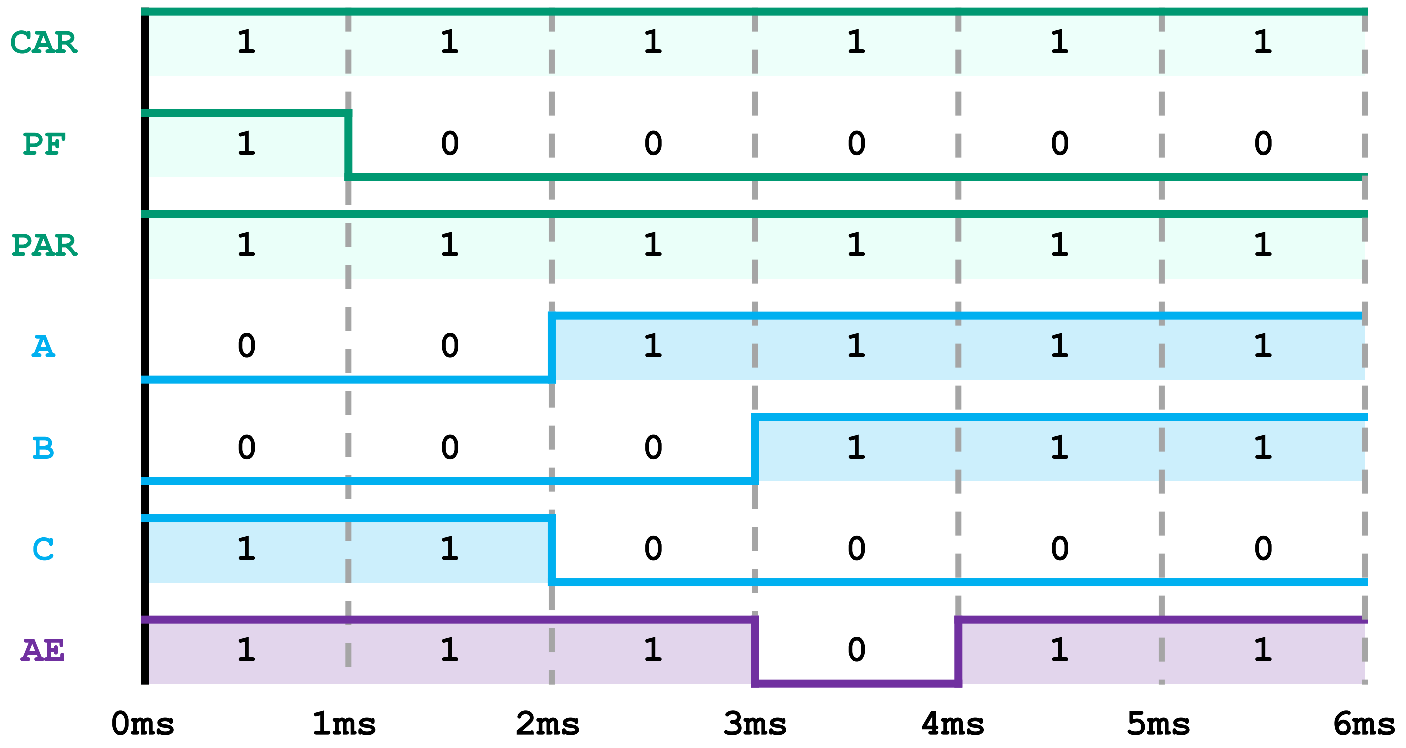
Describe the behavior of the Autopilot Engaged (AE) output between 1ms to 6ms.





Autopilot Control Circuit Example





	0–1ms	1–2ms	2–3ms	3–4ms	4–5ms	5–6ms
CAR	1	1	1	1	1	1
PF	1	0	0	0	0	0
PAR	1	1	1	1	1	1
A	0	0	1	1	1	1
B	0	0	0	1	1	1
C	1	1	0	0	0	0
AE	1	1	1	0	1	1

Combinational vs. Sequential Logic

- So far, we have ignored "time" in our circuits

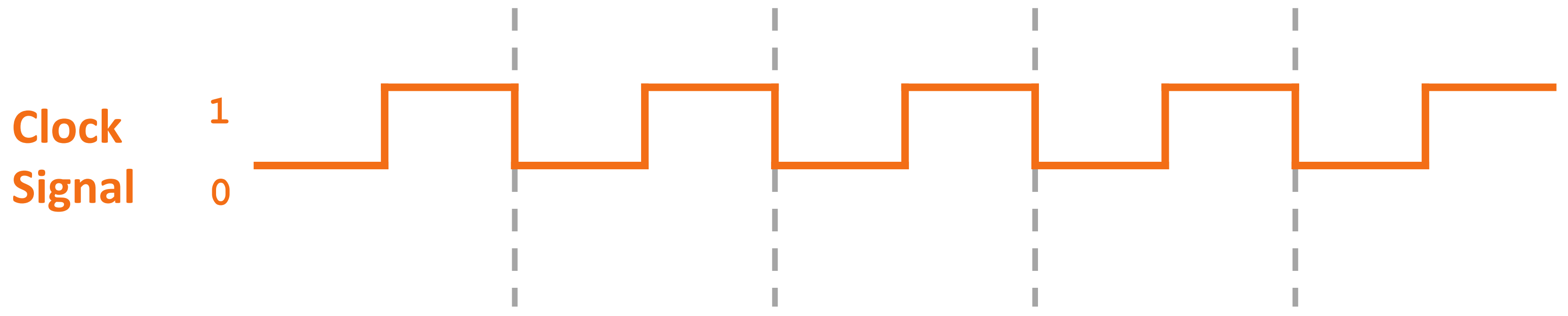
- Our chips used **combinational logic**
  - When given inputs, the chip computes its output instantaneously
  - The output is a function of the current inputs, with no memory of previous events

- Today, we'll start exploring what happens when we consider time, ultimately building to **sequential logic**

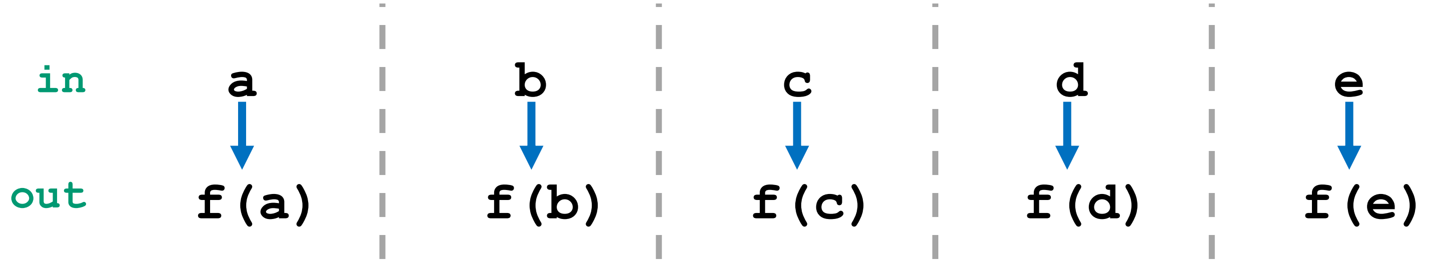
What is Sequential Logic?

- 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

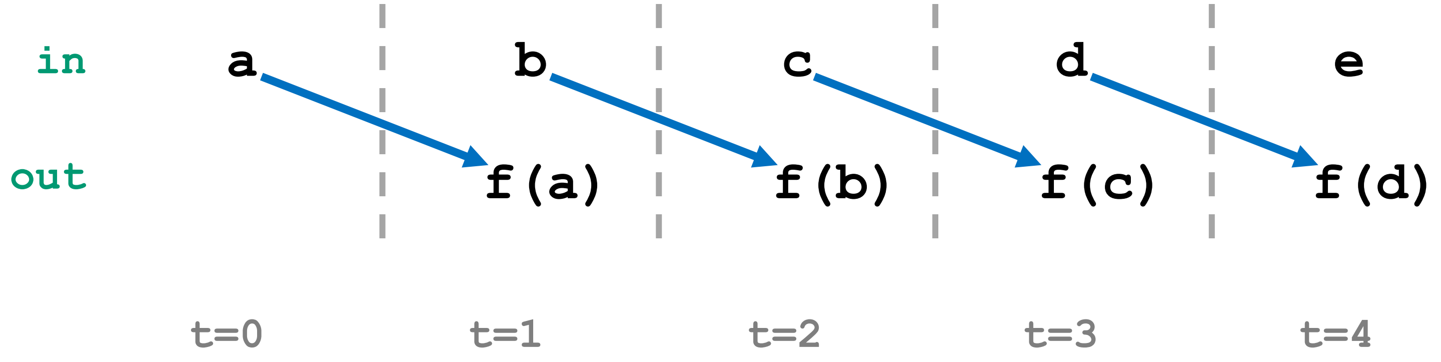
Combinational vs. Sequential Abstraction



Combinational: a function of the current inputs



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



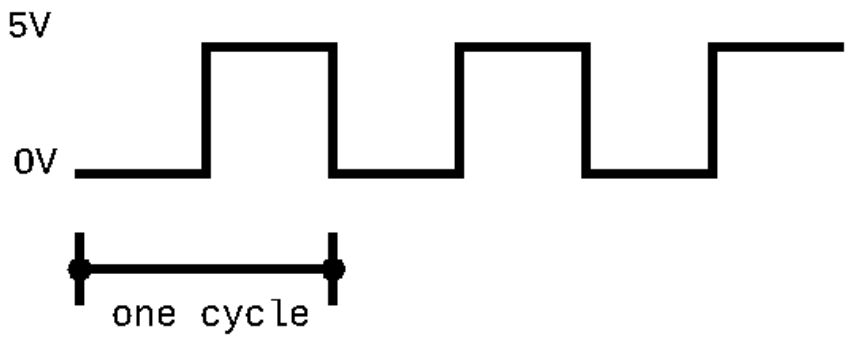
Lecture Outline

- Bloom's Taxonomy
  - Applying Higher Levels of Cognition to Learning
- Introduction to Sequential Logic
  - The Problem with Combinational Logic
- **Representing Time in Hardware**
  - **Clock Signals and Units of Time in Hardware**
- The Data Flip-Flop (DFF)
  - Implementation and Examples

Representing Time: Clock Signals

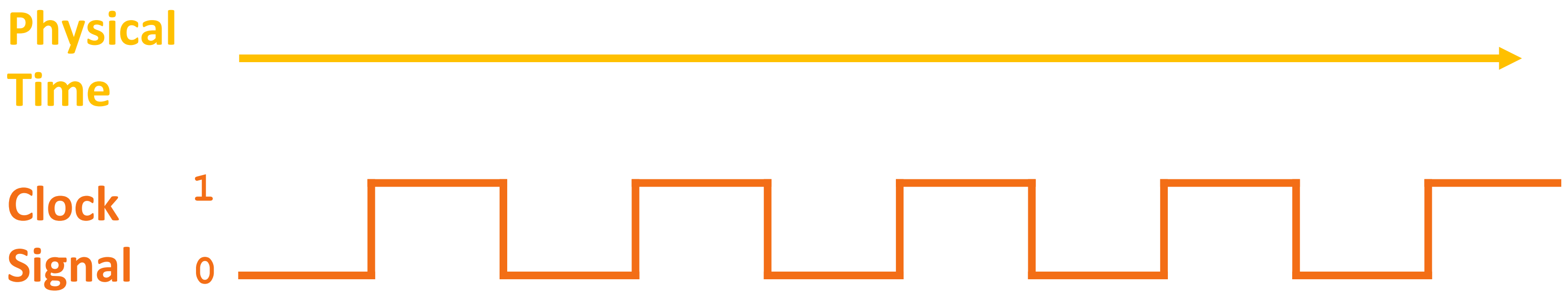
- We physically represent time in hardware with a **clock signal**
  - A clock signal changes its frequency at a set time rate
  - Alternates between a low signal and a high signal of equal length

- 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



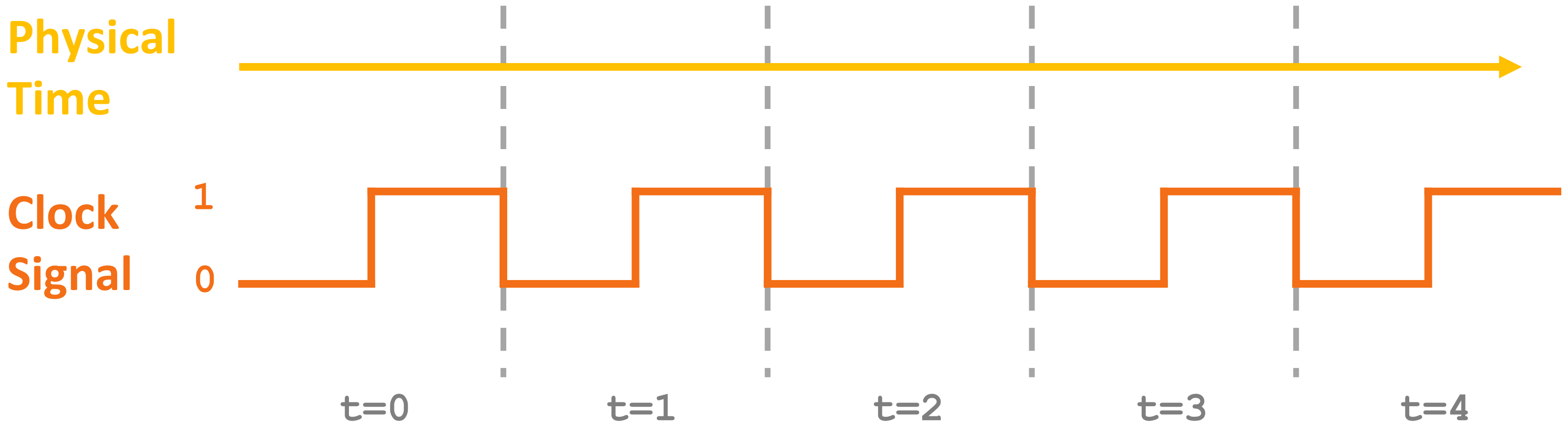
Physical Timekeeping

- 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



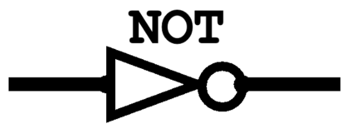
Physical Timekeeping

- 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

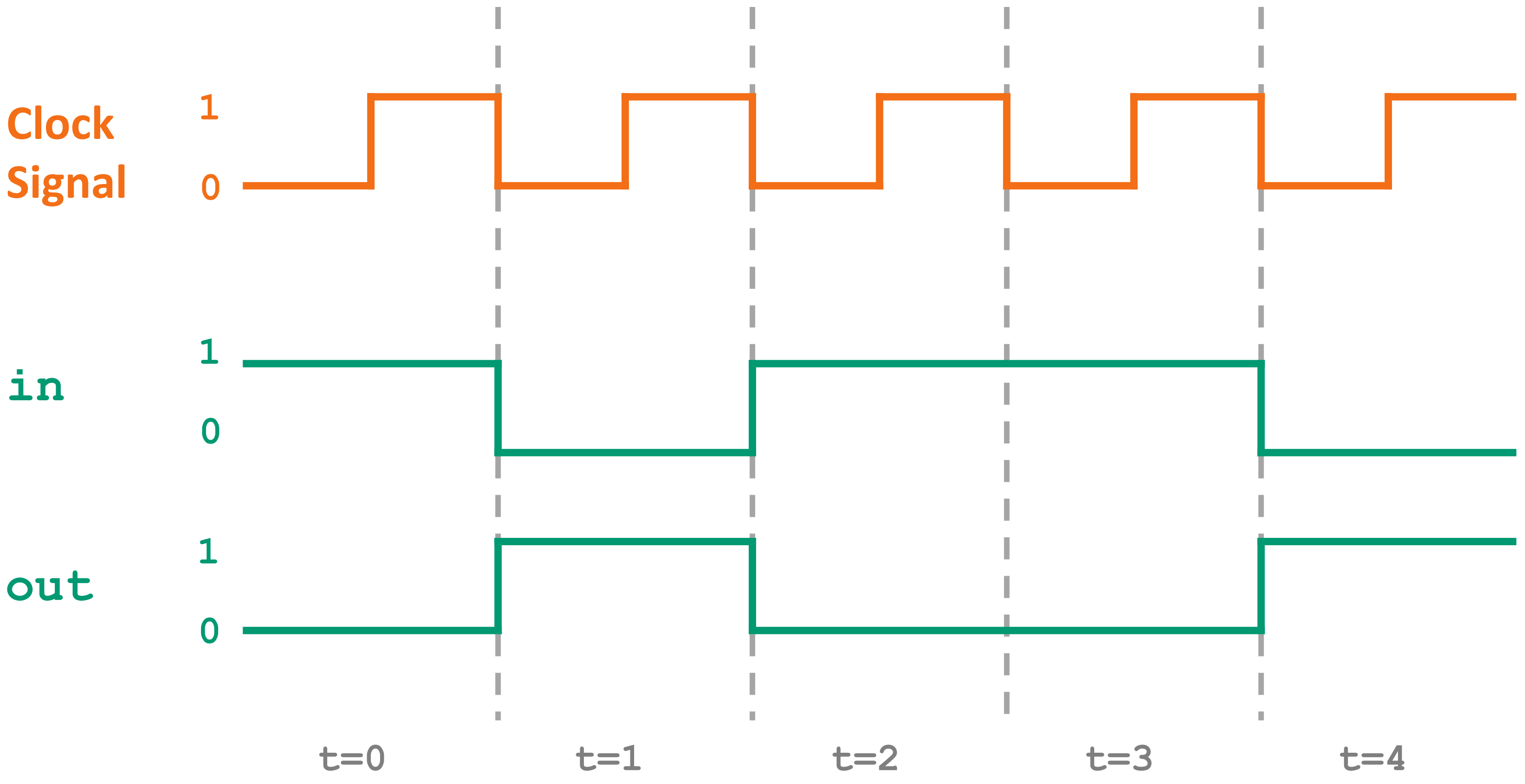


Discrete Time Intervals

Adding a Clock: Ideal

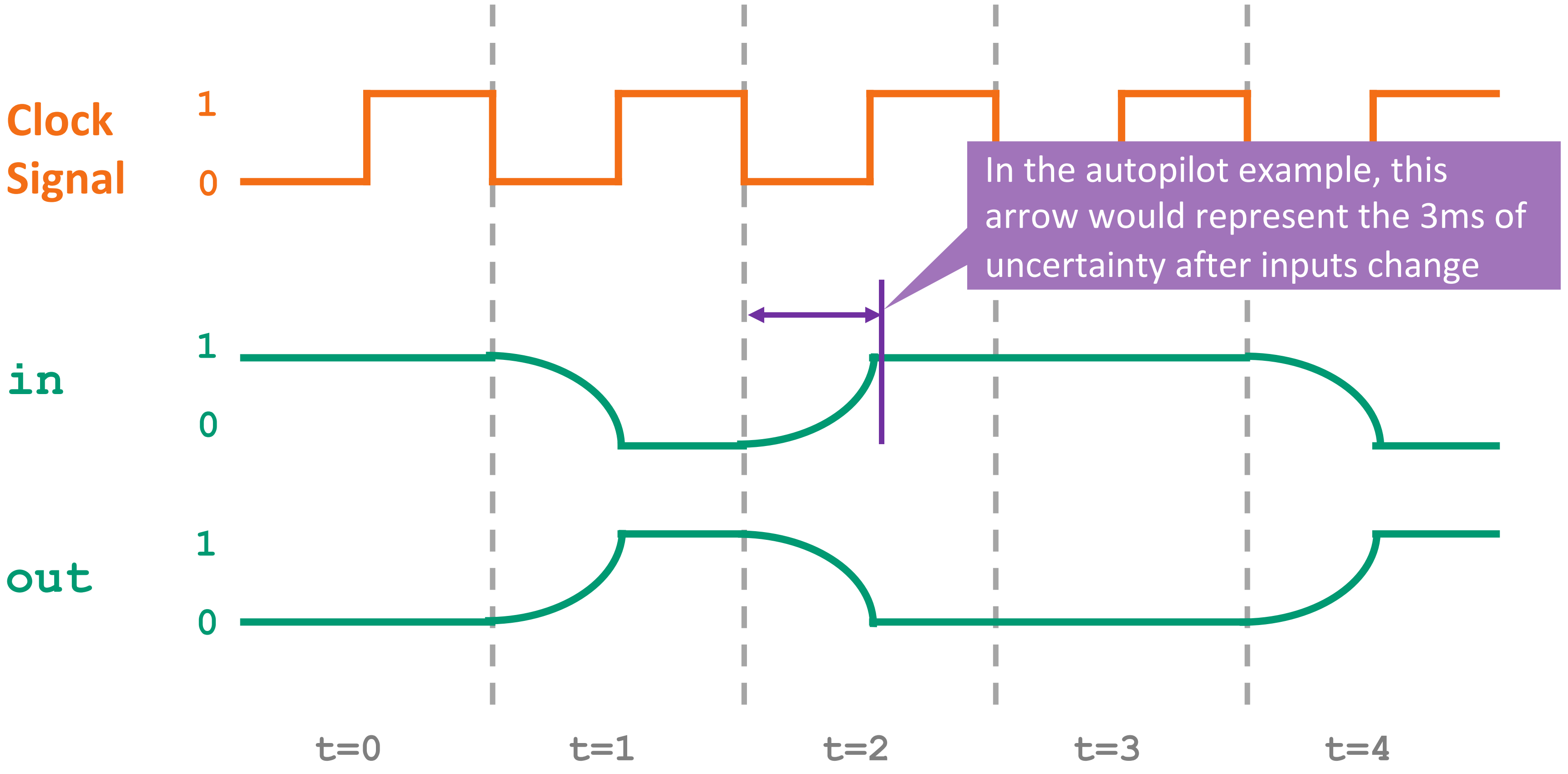


- We want this behavior from a simple, combinational Not gate:



Adding a Clock: Reality

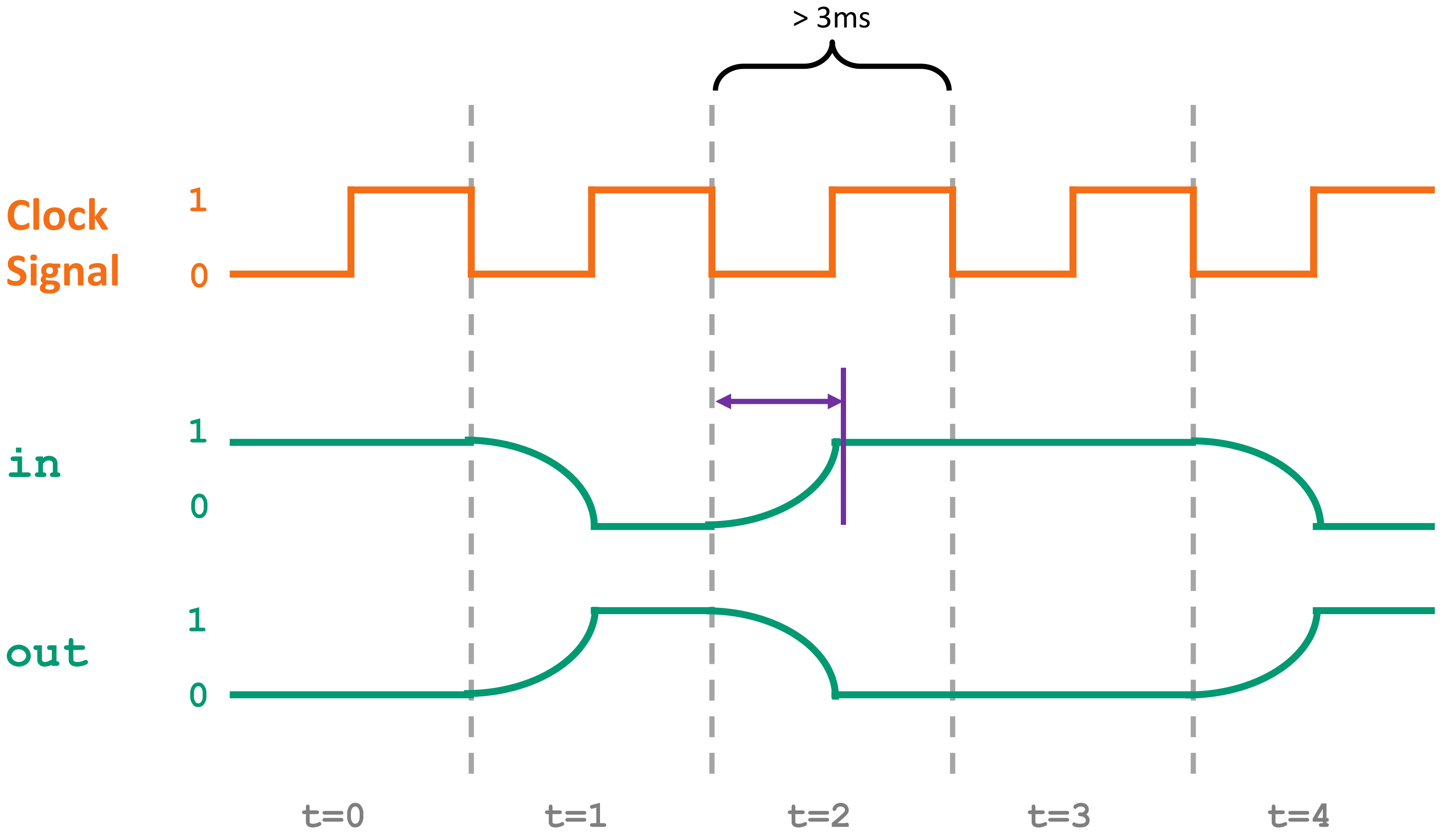
- Combinational logic may be incorrect for a period immediately after inputs change
  - **Computation delays** (logic gates) and **propagation delays** (wires)



In the autopilot example, this arrow would represent the 3ms of uncertainty after inputs change

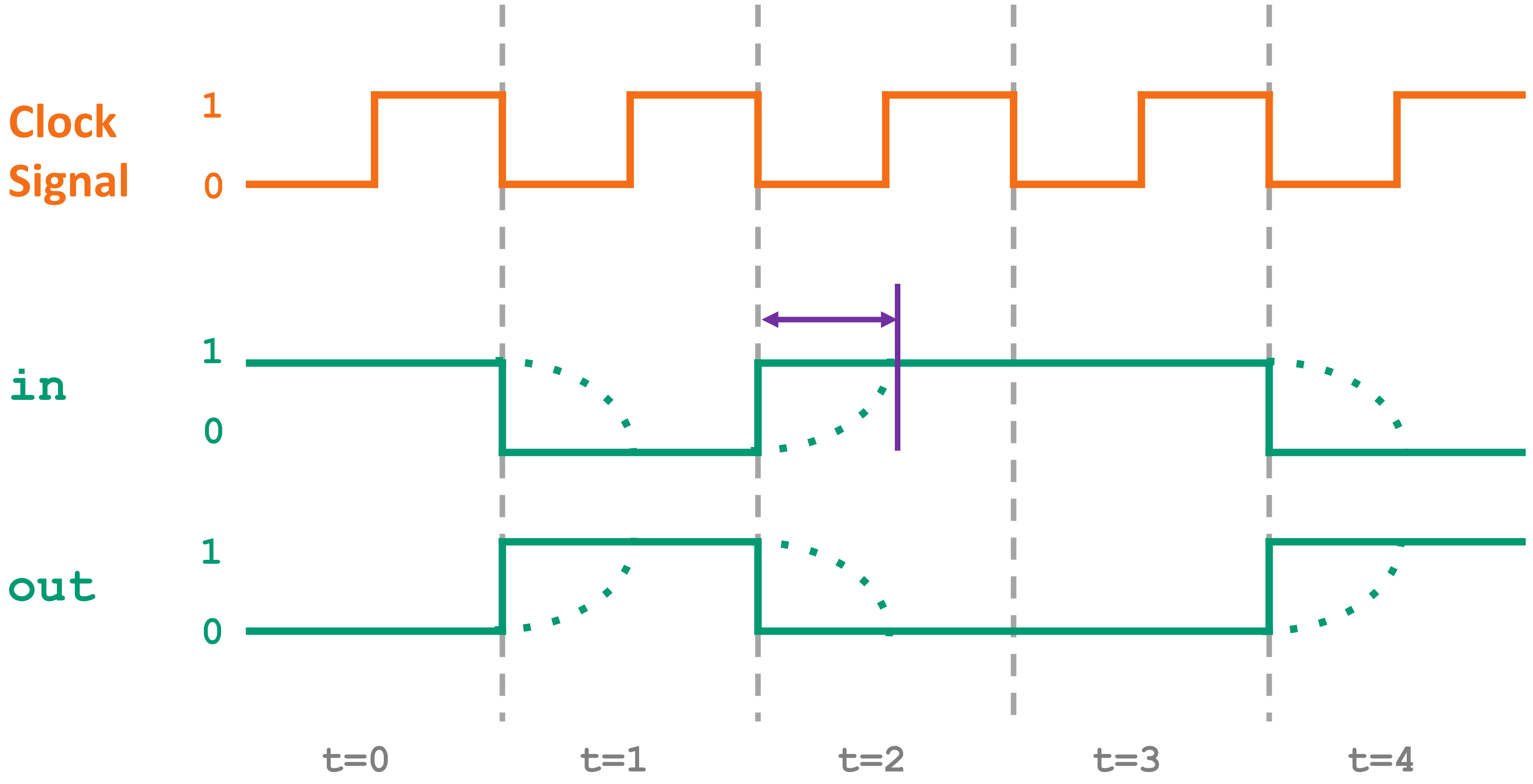
Adding a Clock: Clock Cycles

- 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:

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…

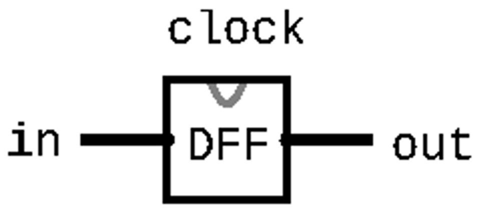
Lecture Outline

- Bloom’s Taxonomy
  - Applying Higher Levels of Cognition to Learning
- Introduction to Sequential Logic
  - The Problem with Combinational Logic
- 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: **`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)

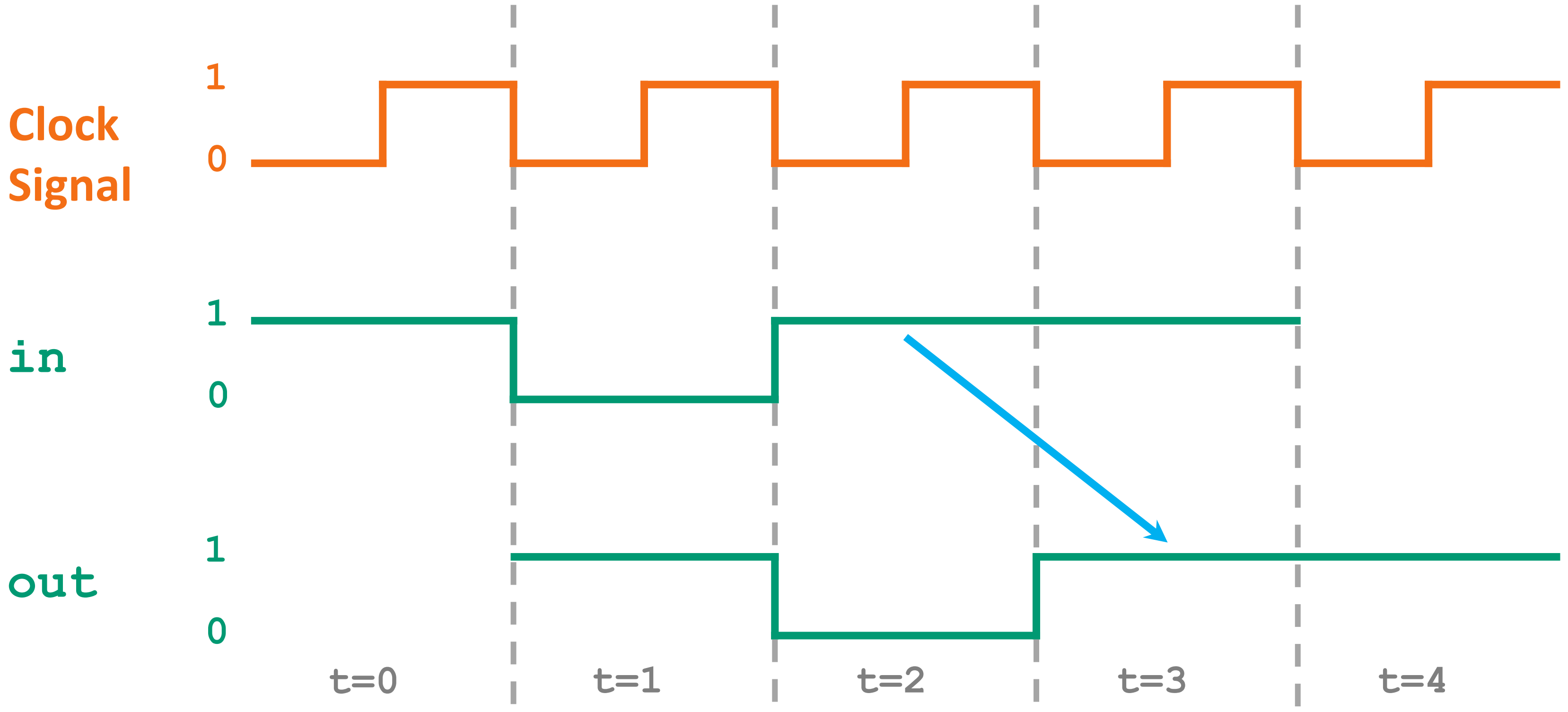


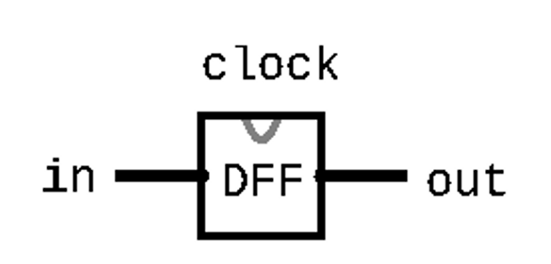
Aside: Treating the DFF as a Primitive

- 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

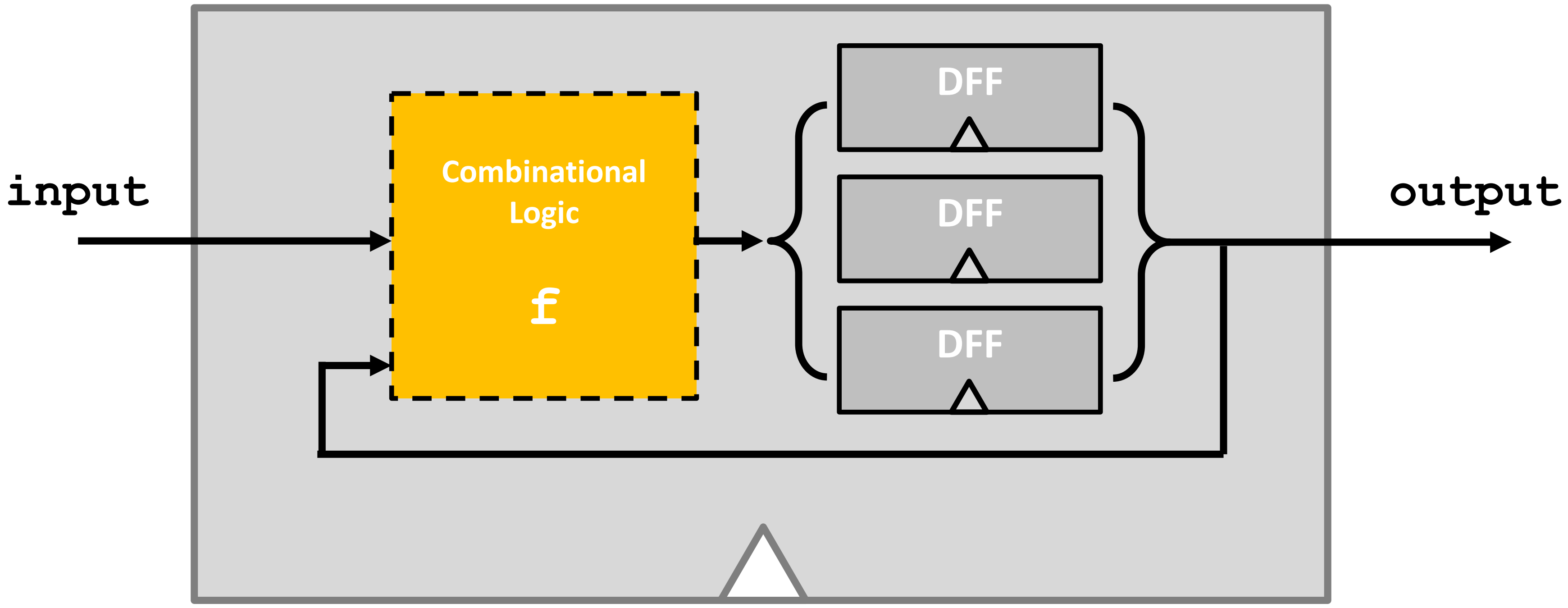
- A category of chips that utilize the clock signal, in addition to any combinational logic

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

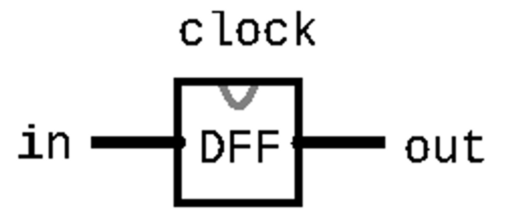
- Constructed from:
  - DFFs
  - Combinational logic (which is entirely constructed from Nand)

Sequential Chips

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



D 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 and HDL implementation

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 1: Circuit Diagram & HDL

- Example specification:

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

- Circuit diagram:

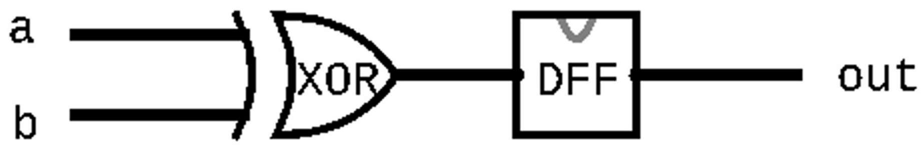
- HDL:

DFF Example 1: Circuit Diagram & HDL

- Example specification:

  `out(t) = 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:
  `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 and HDL implementation

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 2: Circuit Diagram & HDL

- Example specification:

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

- Circuit diagram:

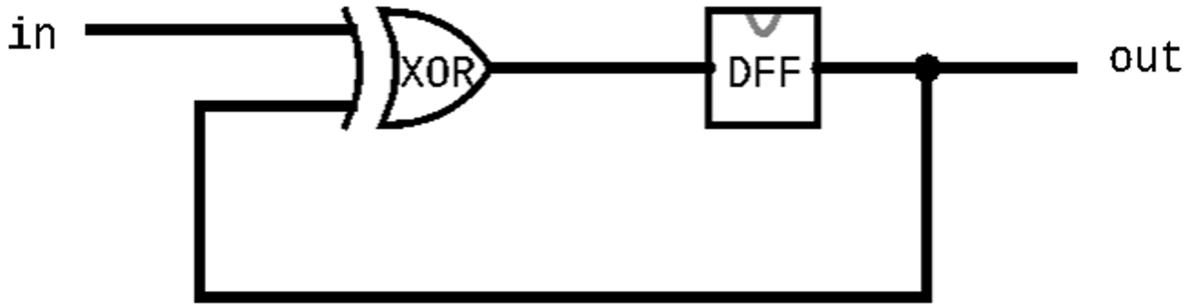
- HDL:

DFF Example 2: Circuit Diagram & HDL

- Example specification:

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

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

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

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))`

DFF Example 3: Circuit Diagram & HDL

- Example specification:

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

- Circuit diagram:

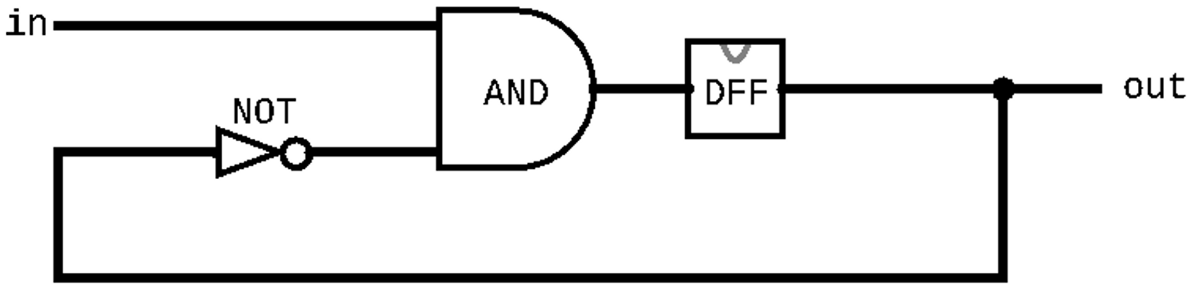
- HDL:

DFF Example 3: Circuit Diagram & HDL

- Example specification:

  `out(t) = And(Not(out(t-1)), 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 6 Reminders

- Exciting lecture topics this Thursday!
  - Metacognitive Subject: Cornell Note-taking Method
  - Technical Subject: Computer Memory

- **Project 3 due this Thursday (10/20) at 11:59pm**

- Preston has office hours after class in CSE2 153
  - Feel free to post your questions on the Ed board as well