Georgia Tech College of Engineering
School of Electrical
and Computer Engineering

ECE 2020:

Flip Flops and State Machines

Instructor: Samuel Talkington

October 24, 2024

Tues 11:59

Logistics

- **Exam 2 grades now available**
  - Mean: 86.87
  - Median: 88.5
  - Standard Deviation: 9.13
- **Old notes for this module** from a previous semester are now available on Canvas; our focus may differ, but I want you to have as many resources as possible.
- **We'll discuss the mid-semester survey** this class

Coming soon

- **Problem set 4:** Released tomorrow. hopefully
- **Prelab 2:** Released, officially due October 31st before lab, but you can turn it in whenever.
- **Lab 2:** Released, will occur in-class on October 31st, 2024.
- I will try to bring candy

Redemption

I believe engineering is an iterative process[1], so we will have these redemption opportunities on the following big-ticket assignments:

- Exam 2 ✓
- Lab reports—final lab portfolio due by the end of the semester.

[1] and I'm passionate enough about this job to spend hours regrading for you all

Exam 2 revisions

- You will have the opportunity to make a revised submission with up to **one** teammate—containing the problems you **both** got wrong—for partial credit towards Exam 2.
- Be prepared to indicate your teammate during today's participation puzzle.
- Collaboration is **not allowed** outside of you and your teammate.
- If resubmitting, you must include a $\approx$ 1 page reflection statement.

Reflection statement guidelines

- Prepare a 1-2 paragraph summary of what we learned in Exam 2. I do not want just a bulleted list of topics, I want you to use complete sentences and establish context (Why is what we have learned relevant? How does it connect with other classes?). *The more insight you give, the better.*
- Prepare a 1 paragraph statement that **explicitly** describes the contributions, improvements, and learning achieved by each teammate.
- Each teammate must explicitly describe their contributions and learning to the new submission in the reflection to receive credit.
- **Submissions with unclear disclosures of both teammates contributions will receive zero partial credit.**

Exam 2 Hints

1.) Building trading

$p_1, \ldots, p_k \in \{0, 1\}$ — ML

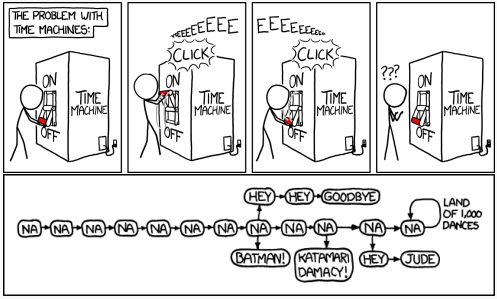
1c.)

	P_3	P_2	P_1	P_0
	0	0	0	0
	0	0	0	1
	0	0	1	0
	0	0	1	1
→	0	0	1	1

Next up: the core of computers

Agenda: next 2 weeks

- Sequential logic
- Latches
- Flip flops
- State machines



Source: xkcd

Motivation: How does memory work

Todays agenda:

- Wrapping up flip-flops
- Fixing up the example from last time
- More examples with flip flops
- Finite state machines
- Example state machine problem

Recap: Flip Flops

Data flip flops

Edge detector

Example: AND gate with multiple inverters, fed into D-Latch

The data flip-flop

What is it

A data flip-flop is the combination of an edge detector + data latch. It is edge-triggered memory.

Symbol + equation

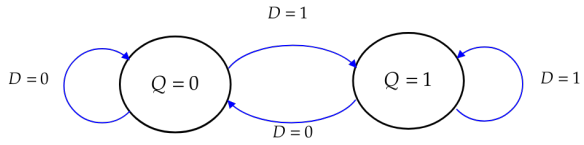
Characteristic table

in_k	$state_k$	$state_{k+1}$	function
D_k	Q_k	Q^+	$\overline{Q}^+$
0	0	0	reset
0	1	0	reset
1	0	1	set
1	1	1	set

State transition

For flip-flops, we can now **truly make a transition table!**

D Flip Flop state transition diagram



Transition table

D	clk	Q
0	↑	0
1	↑	1

Types of activation characteristics for the D flip flops

D flip-flops are like Pokemon—there are multiple types. Each type has different activation, or "trigger" characteristics.

Positive edge

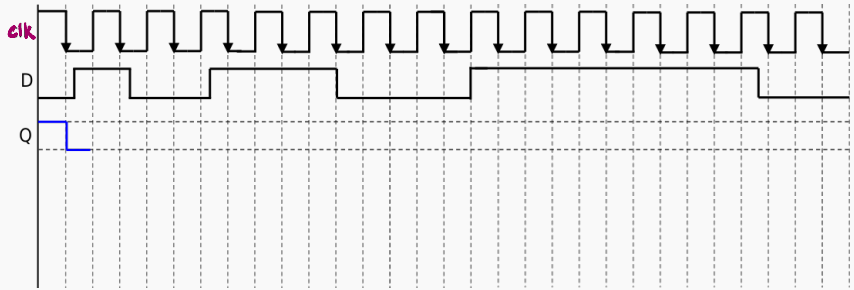
Negative edge

Dual edge

Puzzle

Puzzle 10/22

Fill in this timing diagram for a D flip-flop. Discuss with your neighbor and share. Can you repeat for rising edge-trigger? What about a dual-edge trigger?



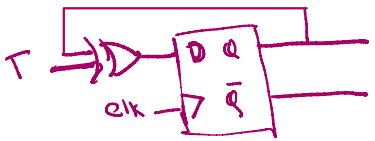
Flip Flop evolutions

The toggle flip-flop

What is it

A toggle (T) flip-flop is the combination of a **data** flip-flop and an XOR gate that *toggles* the state continuously.

Symbol + equation



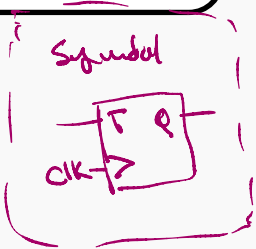
Characteristic table

T_t	Q_t	Q_{t+1}	$\overline{Q}_{t+1}$
0	0	0	hold state
0	1	1	hold state
1	0	1	toggle state
1	1	0	toggle state

$$Q_{t+1} = T_t \oplus Q_t$$

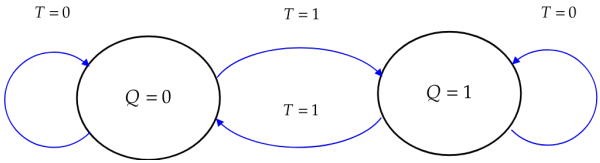
$T=0 \Rightarrow D=Q$

$T=1 \Rightarrow D=\overline{Q}$



Toggle flip flop state transitions

T flip flop state diagram



Rising edge

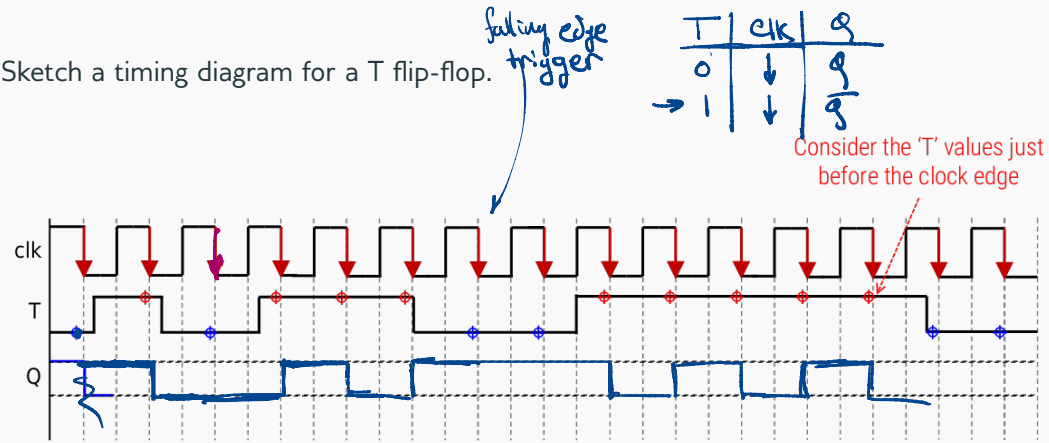
Transition table

T	clk	Q
0	↑	Q
1	↑	$\overline{Q}$

T	clk	Q
0	↓	Q
1	↓	$\overline{Q}$

Example: T Flip Flop timing diagram

Sketch a timing diagram for a T flip-flop.



State machines

How do computers actually work?

State machines

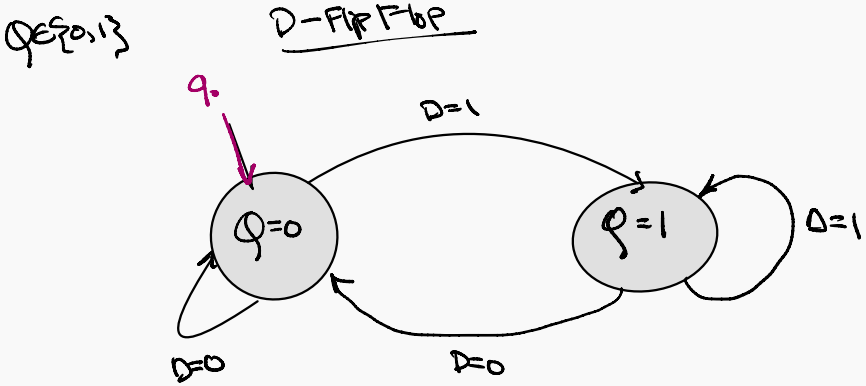
Combinational logic + memory

Latches and flip flops let us store information.

How do we perform actions based on what has happened in the past? (Our memory)

State machines... We've seen this this week! Let's remind ourselves:

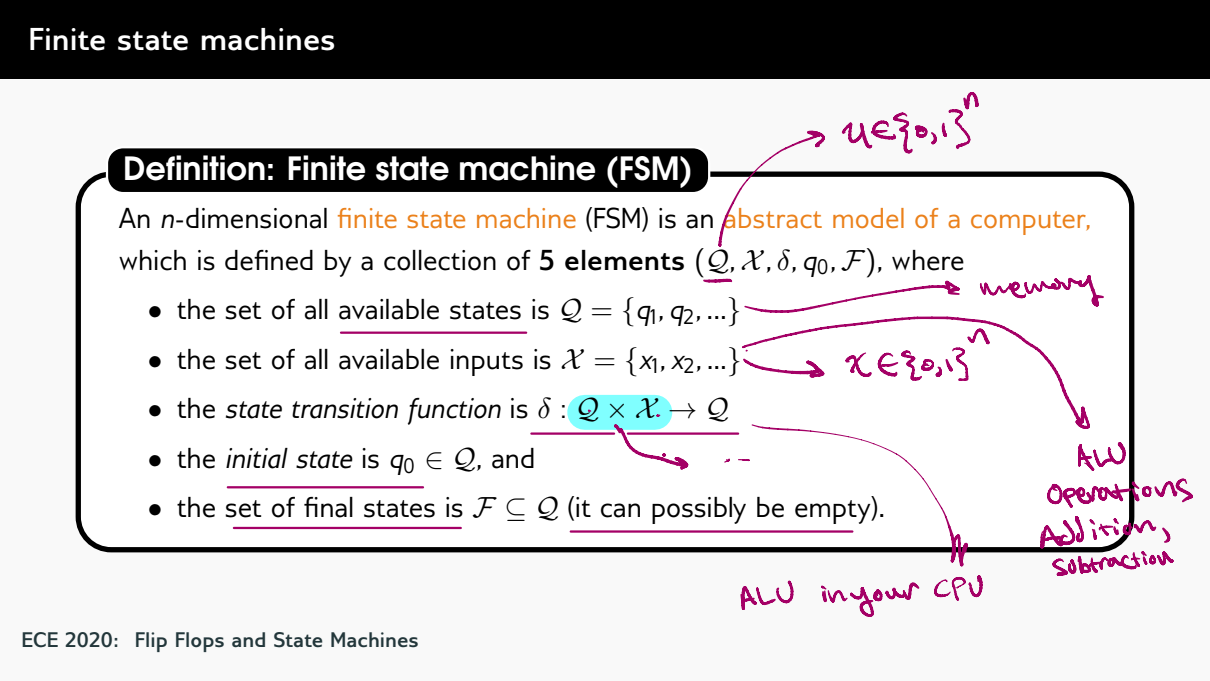
Defining state diagrams explicitly



Definition: Finite state machine (FSM)

An n-dimensional finite state machine (FSM) is an abstract model of a computer, which is defined by a collection of **5 elements** $(\mathcal{Q}, \mathcal{X}, \delta, q_0, \mathcal{F})$, where

- the set of all available states is $\mathcal{Q} = \{q_1, q_2, ...\}$
- the set of all available inputs is $\mathcal{X} = \{x_1, x_2, ...\}$
- the *state transition function* is $\delta : \mathcal{Q} \times \mathcal{X} \rightarrow \mathcal{Q}$
- the *initial state* is $q_0 \in \mathcal{Q}$, and
- the set of final states is $\mathcal{F} \subseteq \mathcal{Q}$ (it can possibly be empty).



Note: The above definition is technically a *synchronous sequential* FSM (i.e., controlled by a clock), which will be the focus of our course.

State transition

Markov Decision Process

State transition function

The state transition function $\delta : \mathcal{Q} \times \mathcal{X} \to \mathcal{Q}$ takes in a current state and the present input and returns a new element $q_{t+1} \in \mathcal{Q}$ from the set of all states $\mathcal{Q}$.

We can write:

$$q_{t+1} = \delta(q_t, x_t).$$

Current state

currently supplied input

synchronous FSM

- Depends on inputs and state at *discrete* instances of time
- e.g. clocked CPU chips, flip-flops, chip registers etc.
- this is what we care about

asynchronous FSM

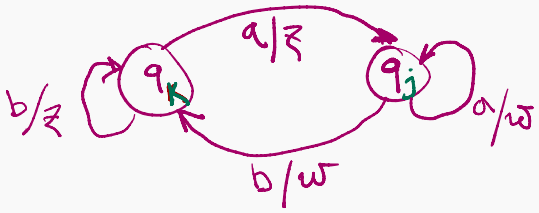
- Depends on inputs and states at any instance of time
- e.g. interrupt-driven computers, asynchronous communication systems, etc.

Example: Markov Decision Process

Example: Mealy and Moore State Machines

Mealy: Output depends on both $(q_t, x_t) \in Q \times X$

The input/out is labeled on each transition line



$Q = \{q_k, q_j\}$

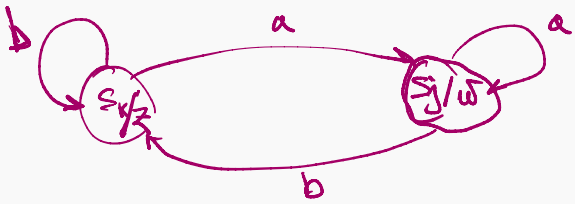
$X = \{a, b\}$

$Z = \{z, w\}$

$q(0) = q_1$

Example: Mealy and Moore State Machines

O/P depends on present only



Example:

Inputs $X = \{0,1\}$ $Z = \{0,1\}$ $Q = \{q_0, q_1\}$

$q(0) = q_0 \in Q$ $q_{t+1} = \delta(q_t) = \overline{q_t}$

Draw The FSM!