3827 OH

Eumin Hong (eh2890)

Columbia University

February 15, 2022

Overview

Announcements

Announcements: Upcoming Assessments

- For HW~~2~~ 3, do not hand in "Warmup Problems" – answers are on CourseWorks
- HW3 is due on Friday 2/18
- HW4 is due on Friday 2/25
- Will have HW1 and HW2 graded soon

Announcements: Homework 1 and Homework Submission

- Common errors from Homework 1:
  - Detecting when overflow happens (last two carries don't match)
  - Missing bits in question 5 (floating point number) – group bits in 4
  - Misinterpretation of signed magnitude/1's complement/2's complement
- Submitting homework on Gradescope:
  - Be sure you submit to the right TA and right homework number
  - There are a lot of assignments, so just search for text

$$\underbrace{\qquad}_{1}\qquad\underbrace{\qquad\qquad\qquad}_{8}\qquad\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{23}$$

Announcements: Feedback

- Form: `https://forms.gle/cnUmKVNYN7WvRbHA6`
- From feedback since last time:
  - "...it would be nice if you asked for HW problems after 15 mins of going through the important concepts rather than spending 40+ mins on the important concepts and then remaining time on the hw."
    - Will start taking HW problems earlier and cover concepts as we go
    - Will denote notable concepts with "NC"
  - "It also would be nice if when going through the concepts you do some example problems that are similar to the homework."
    - Coming up with questions is always hard, so I will probably default to using the homework questions to show concepts I am referring to

Homework 3 Warmup

Homework 3 Warmup: Problem 1

Build a MUX from a Decoder and some AND and OR gates.

- NC: Symmetry of inputs and circuit logic

Homework 3 Warmup: Problem 2

A MUX-with-Enable is a MUX with one additional selector input, E. When $E = 1$, the MUX-with-Enable behaves like a traditional MUX. When $E = 0$, the MUX-with-Enable is disabled and outputs 0. Build a MUX-with-Enable from a MUX (without enable) and some AND gates.

Homework 3 Warmup: Problem 3

A 1-to-2^k DEMUX takes one data input I and a k-bit selector S as input and outputs 0 on each of $2^k - 1$ outputs, and outputs I on the jth output where j is the unsigned binary value represented by S. Construct a DEMUX from a decoder and a bunch of AND gates.

Homework 3 Warmup: Problem 4

Use five 2-to-1 MUXs and as many NOT gates as needed, buid a circuit that take a 5-bit value and a selector input S and returns

- when $S = 0$, the original number is returned
- when $S = 1$, the 1's complement of the number is returned.
- NC: MUXes as if/then/else statements

Homework 3 Warmup: Problem 5

Build a circuit using two 2-to-1 MUXs that takes a 2 bit-input B_1B_0 and a 1-bit selector S and returns B_1B_0 when $S = 0$ and returns B_0B_1 (i.e., switches the bit-order) when $S = 1$.

- NC: MUXes as if/then/else statements

Homework 3 Warmup: Problem 6

Solve problem 3 of the "Harder Problems" using four 16-to-1 MUXs, one MUX for each output NSH, NSL, EWH, EWL. Now solve using four 8-to-1 MUXs.

- NC: MUXes as brute-force solution

Homework 3 Warmup: Problem 7

Design a circuit that receives a k-bit string $A = A_{k-1}A_{k-2}\dots A_1A_0$ and, using $k-2$ 4-to-1 MUXs, outputs k-bit string $B = B_{k-1}B_{k-2}\dots B_1B_0$ with the following properties

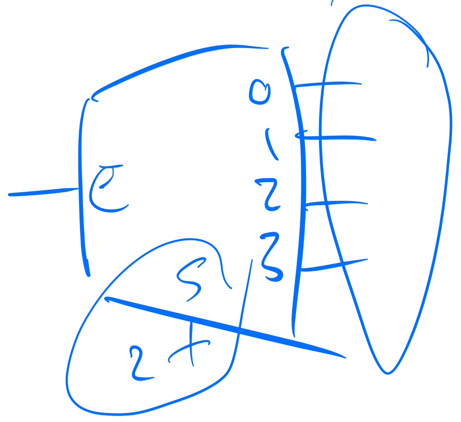
- $B_0 = A_0$, $B_{k-1} = A_{k-1}$
- $B_i = A_i$, whenever $A_{i+1} \neq A_{i-1}, 0 < i < k-1$
- $B_i = A_{i-1}$, whenever $A_{i+1} = A_{i-1}, 0 < i < k-1$
- NC: MUXes as if/then/else statements

Homework 3 Harder Problems

Homework 3 Harder Problems: Problem 1

Construct a 4-to-16 line decoder with an enable input using five 2-to-4 line decoders with enable inputs (Hint: Start at the outputs: If all that is being used is decoders, then how many decoders are connected directly to outputs?)

- NC: Symmetry of circuit
- NC: matching combinational circuits to number of inputs/outputs
- NC: partitioning of input (i.e. $I_3 I_2 I_1 I_0$ into $I_3 I_2$ and $I_1 I_0$)



Homework 3 Harder Problems: Problem 2

A combinatorial circuit is specified by the following three Boolean functions:

$$F = X + \overline{Y} + \overline{X}YZ$$

Design the circuit with a decoder and external OR gates.

- NC: decoder as "cases" or minterms

Lec 3, 16

Homework 3 Harder Problems: Problem 3

ABCD 80s = 5 × 16 values

A traffic light controller receives a 4-bit input that changes every 5 seconds. The input sequence is a simple counter that counts from binary 0 to binary 15 and then starts again from binary 0. These signals go to 4 outputs, NSH, NSL, EWH, EWL. The first two outputs NSH and NSL respectively represent the high and low bits that are fed to the lamps that face in the North-south direction. The other two outputs EWH and EWL respectively represent the high and low bits that are fed to the lamps that face in the East-West direction. The following table indicates how setting the high and low bits determines the color of the lamp:

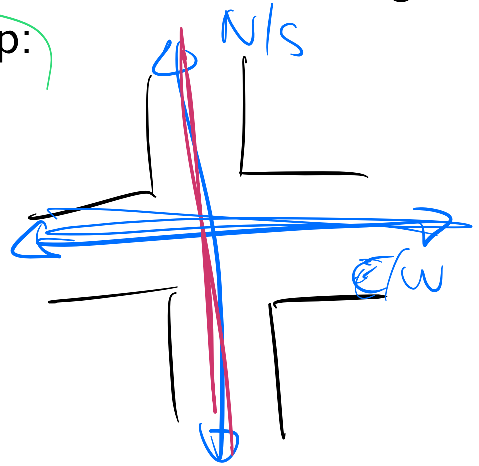
0000 → 0001 →

... →

1111 → 0000

High	Low	Lamp Color
0	0	green
0	1	Yellow
1	0	Red
1	1	Red

→ X



Homework 3 Harder Problems: Problem 3 (cont.)

9 vals   6vals   1vals

Each light should be green for 30 seconds, yellow for 5 seconds, and then red for 45 seconds. There should be two 5 second intervals when both lights are red. Assume that for the interval where the input is 0000, the North-South light has just turned green, and the East-West light has been red for 5 seconds already. → 1 val

Design the circuitry that feeds from *ABCD* to the ~~two~~ four outputs. You may represent your answer as algebraic equations (make sure to simplify).

- NC: at least one similar word problem (probably with FSMs) will be on exam

Q1: bits
Q2: comb. circuitry
Q3: FSM
finite state machine

NS   EW

		color		NS		EW	
time	A B C D	NS	EW	H	L	H	L
0	0 0 0 0	G	R	0	0		
1	0 1	G	R	0	0		
2	1 0	G	R	0	0		
3	1 1	G	R	0	0		
4	0 1 0 0	G	R	0	0		
5	0 1	G	R	0	0		
6	1 0	Y	R	0	1		
7	1 1	R	R	1	X		
8	1 0 0 0	R	G	X			
9	0 1	R	G				
10	1 0	R	G				
11	1 1	R	G				
12	1 1 0 0	R	G				
13	0 1	R	G				
14	1 0	R	Y				
15	1 1	R	R				

1) truth table

2) kmap

3) boolean expr.

4) circuit

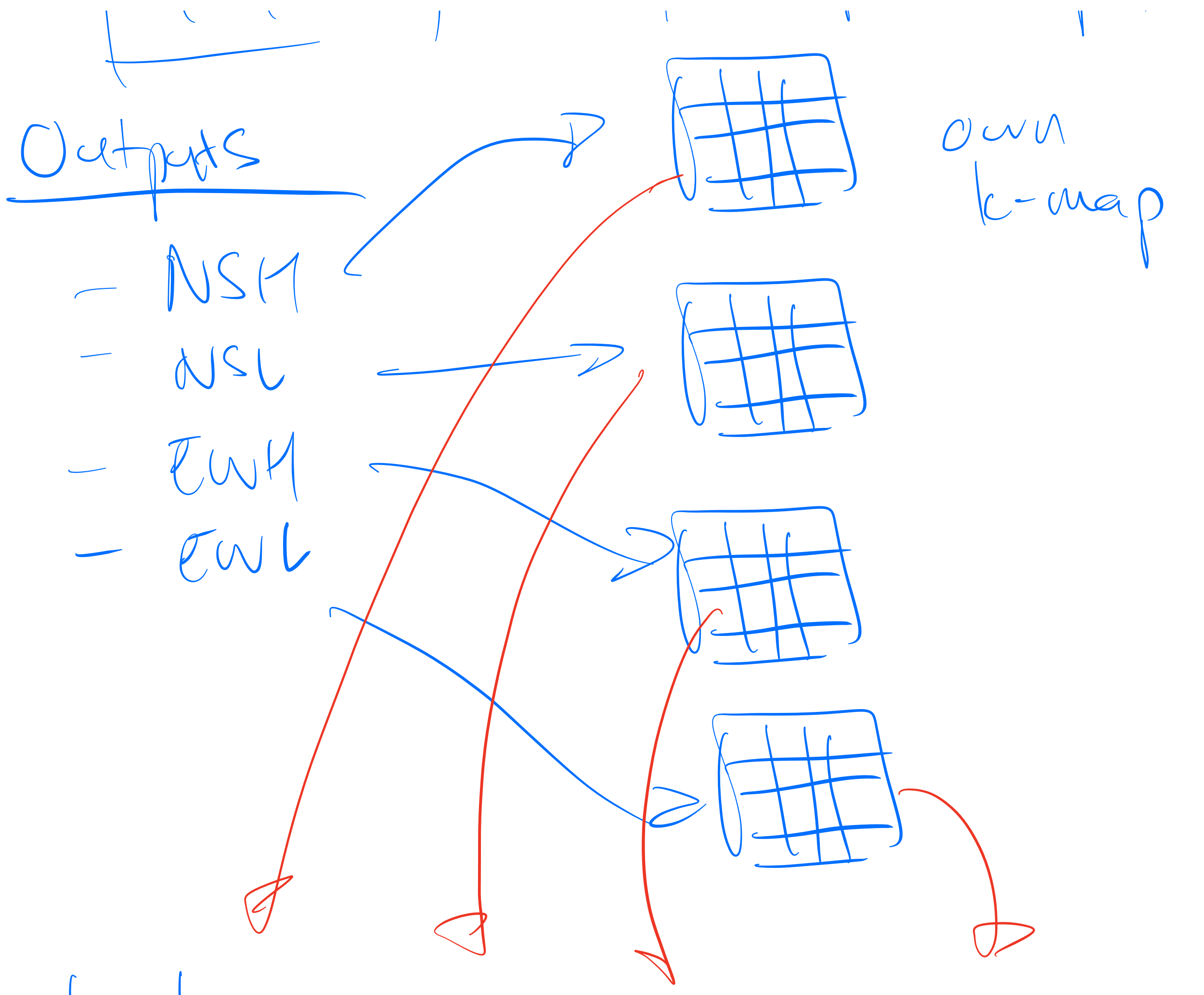
CD → less significant for NSH

kmap

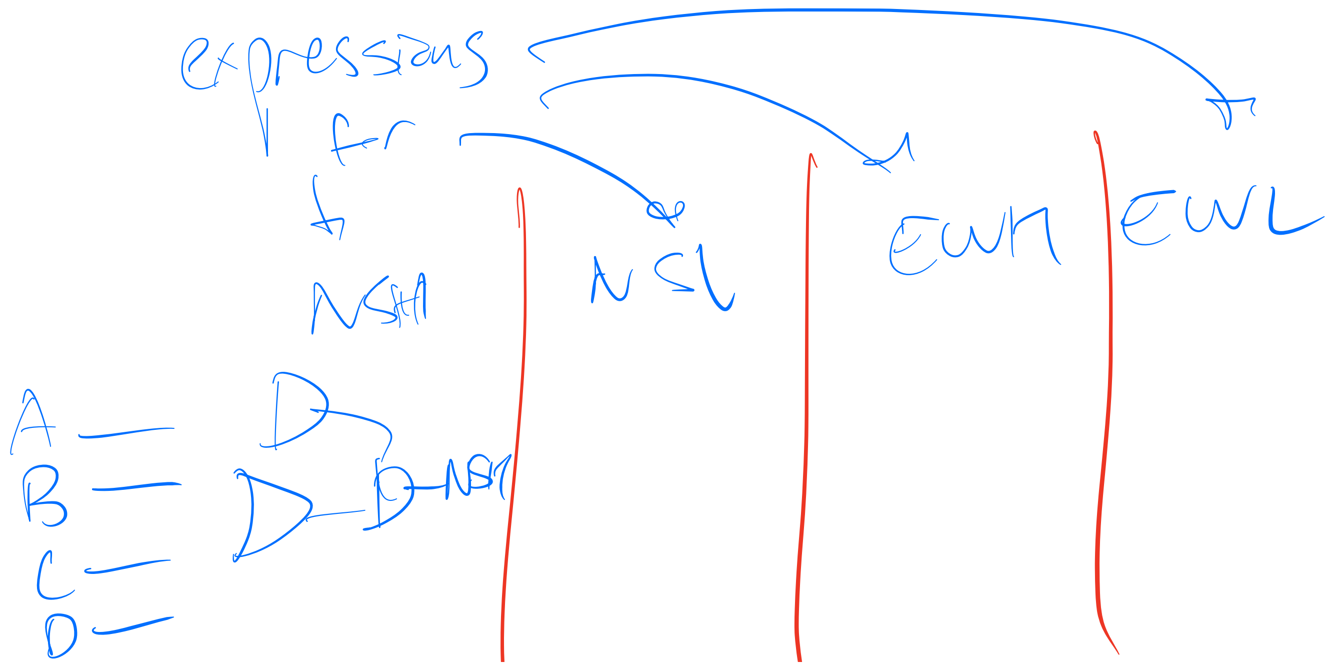
AB more sig.

0	1	3	2
4	5	7	6
12	13	15	14
8	9	11	10

1 kmap for each output



boolean expressions for NSH



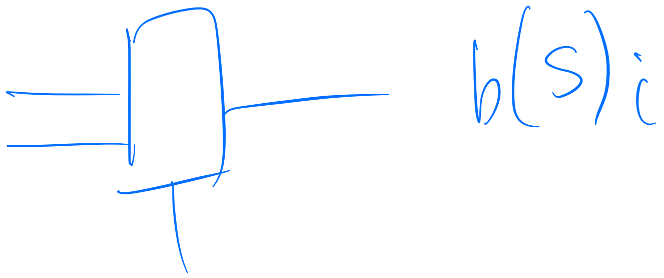
Homework 3 Harder Problems: Problem 4

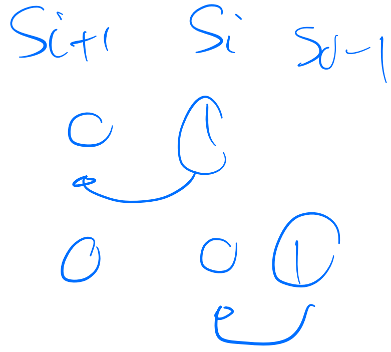
b(S)

The 01-swap operation, b, on a binary string S, permutes all occurences of 01 within the original string to 10 (the process is not recursive). For instance:

- $b(0) = 0, b(1) = 1, b(00) = 00, b(01) = 10, b(11) = 11$
- $b(000) = 000, b(001) = 010, b(010) = 100, b(100) = 100$
- $b(0\ 01\ 01\ 1\ 01\ 0) = 0\ 10\ 10\ 1\ 10\ 0$ (spacing added for clarity)

Design a circuit using 2:1 multiplexers that can be used to perform the 01-swap on a k-bit string $S = S_{k-1}S_{k-2}\dots S_0$. where each S_i is a bit.





Homework 3 Harder Problems: Problem 4 (cont.)

(a) First, show the circuit, built using the 2:1 MUX, whose output is the ith bit of $b(S)$ where $0 < i < k - 1$. YOU DO NOT NEED ANY AND, OR, ~~OR~~ NOT GATES, only a single 2:1 MUX. This is somewhat challenging, so think what input information you need.

(b) Use contraction to solve the edge cases when $i = 0, k - 1$. You do not have to simplify the internals of the MUX, just explain why you "contracted" as you did.

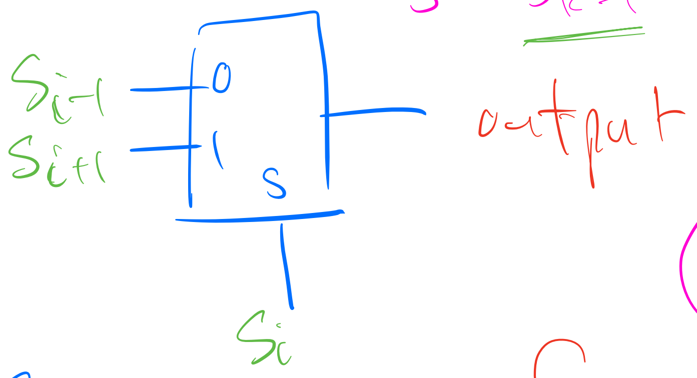
- NC: solving parts of larger question will appear on exam
- NC: contraction is setting certain inputs to constants and/or getting rid of unnecessary components

Lec 4, 11-7

$i \neq 0 \rightarrow$ not LSB

$i \neq k-1 \rightarrow$ not MSB

$S = S_{k-1} S_{k-2} \dots S_1 S_0$



$b(s)_i$

bit i of S after 01-swap

f(1)

S ... 0 0 1 0 1 0 0 ...

b(s) 0 1 0 1 0 0 ? ...

S_{i+1}	S_i	S_{i-1}	$b(s)_i$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Homework 3 Harder Problems: Problem 5

The isolated-1-shift-left operationg (i1sl for short) applied to a binary string S moves a 1 bit by one position to the left in the solution if the bit is not adjacent to any other 1's (i.e., 0's on both sides, or if is the least-significant bit, a 0 immediately above it).
For instance, the following show application of i1sl to various strings:

- $010 \to 100, 011 \to 011, 01010 \to 10100$
- $011010 \to 011100, 011100 \to 011100$

(a) Build a simplified circuit (using only AND, OR, NOT gates) whose output is the ith bit of the i1sl, where $1 < i < k - 1$ (there's a hint here about what inputs are needed). You may leave your answer in algebraic (SoP) form in terms of the S_j. Note that the ith bit can be determined by only looking at a few of the S_j.

$S_{i+1}\ S_i\ S_{i-1}\ S_{i-2}$

$0\ 1\ 0$

010

Homework 3 Harder Problems: Problem 5 (cont.)

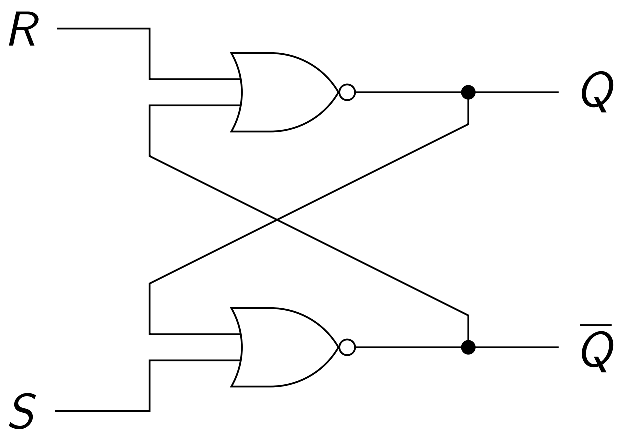
limited inputs $\rightarrow$ don't cares

- (b) Suppose it is known that the input string will never contain 3 consecutive 1's. Draw the simplified circuit.
- (c) Suppose the input string might contain 3 1's, but that no consecutive 4 bits contain 3 0's (i.e., 0000, 0001, 0010, 0100, or 1000 never appear as a substring), nor does 0101 ever appear (1010 might still apear though, e.g., 111011011010). Draw the simplified circuit.
- (d) Use contraction to design the circuits for the outputs of the $k-1$th, 1st, 0th bits.
  - NC: contraction is setting certain inputs to constants and/or getting rid of unnecessary components

Homework 4 Topics

Homework 4 Topics: SR Latch

- Latches just hold state – can be changed by changing the values of the inputs
- SR latch has inputs of “Set” (S) and “Reset” (R)
- SR latch has outputs of Q and $\overline{Q}$



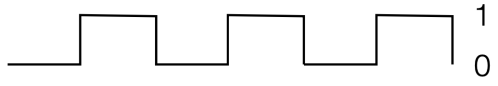
- For understanding behavior of circuit for given (R, S), think about when NOR operation gives 0
    - Or more generally, passing a constant to a gate can make its output constant as well

Homework 4 Topics: D Latch

- Basically SR latch with some more circuitry to avoid “illegal” combinations of inputs (e.g. when $R = 1, S = 1$ what does the SR latch do? How can it “set” $Q = 1$ and “reset” $Q = 0$ at the same time?)
- D latch has inputs D and C (control, which ensures $S \neq R$ in the underlying SR latch)
  - D is the value to store/write if $C = 1$
  - C can also be E for enable

Homework 4 Topics: Clocking

- Clock signal oscillates between 0 and 1 with a fixed period
  - "Rising edge" of clock signal is the transition from "low" to "high" (i.e. $0 \rightarrow 1$)



  - Kind of like a metronome, gives the circuit a sense of time on which it can operate
  - Why should I care about clock? It determines the rate at which instructions are executed

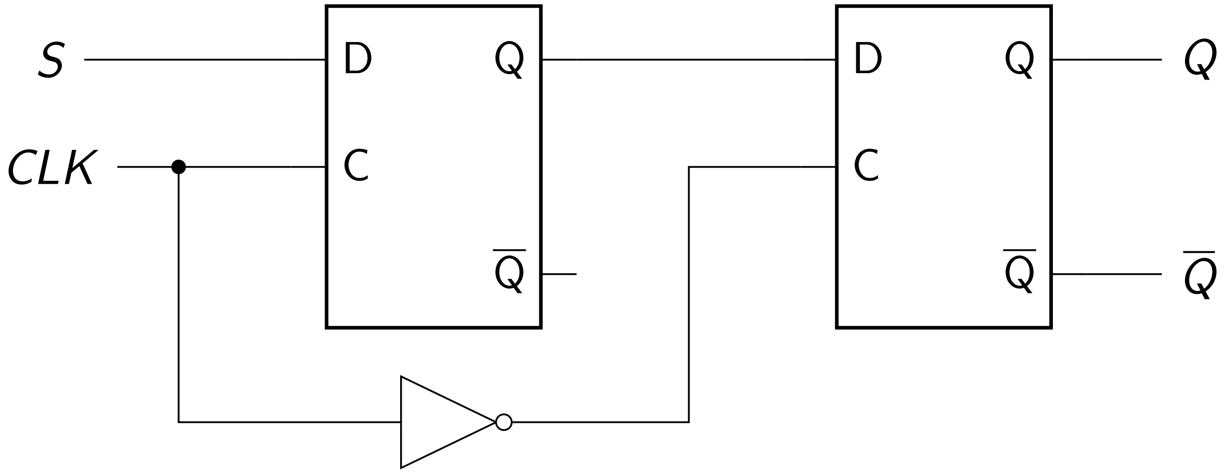
Apple M1

Apple M1 chip

General information	
Launched	November 10, 2020[1]
Designed by	Apple Inc.
Common manufacturer(s)	TSMC
Product code	APL1102[2]
Performance	
Max. CPU clock rate	3.2 GHz[1]

Homework 4 Topics: Flip-Flops

- Latches do not support clocking individually
- Connecting latches in series (and ensuring that the clock inputs are complemented or staggered) acts as a flip-flop
- Example of D Flip-Flop:



- Left D latch is updated when $CLK = 1$ (new inputs read to left D latch), right D latch is updated when $CLK = 0$ (load new inputs to right D latch)

Homework 4 Topics: Flip-Flop Behavior

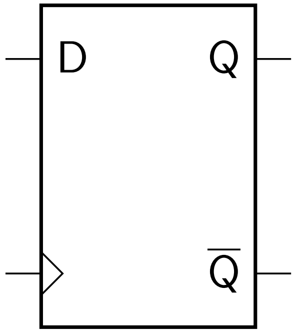
- For the D flip-flop:

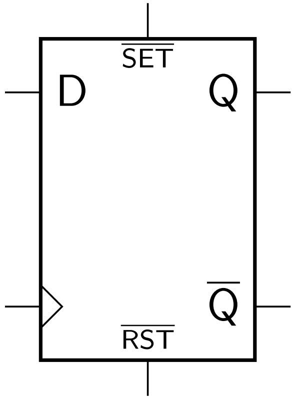
$D(t)$	$Q(t+1)$
0	0
1	1

- The input t is time, or more specifically, the tth clock cycle
    - A clock cycle is one period of the clock
- In other words, the input D at clock cycle t becomes the output Q at clock cycle $t+1$ (the next clock cycle)

Homework 4 Topics: Flip-Flop Abstraction

- The two latch circuit from earlier is abstracted and is a D flip-flop (D since it has the same inputs as a D latch) (left circuit)





- The triangular input (bottom left) is for the clock signal
- How do we initialize the values of a flip-flop?
  - Use set and reset inputs (right circuit), implementation is not necessary to know (we are not really worried about initialization in this course)