More on Sequential Logic

CS 64: Computer Organization and Design Logic
Lecture #15
Winter 2019

Ziad Matni, Ph.D.
Dept. of Computer Science, UCSB

I'm a CS major
What's a CS major?
It means he's afraid of hardware
No it doesn't
CIRCUITS, CPUs, EMBEDDED SYSTEMS
Stop it Patrick you're scaring him!

Administrative

- Lab #7
  - Due next week on Wednesday 3/6
  - Paper copy to submit in HFH 2nd Floor (CS64 box)

Administrative

- The Last 3 Weeks of CS 64:

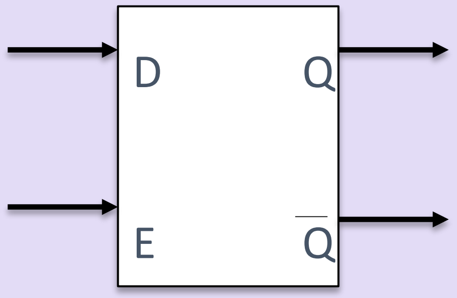
Date	L #	Topic	Lab	Lab Due
2/26	14	Combinatorial Logic, Sequential Logic 1	7 (CL+SL)	Wed. 3/6
2/28	15	Sequential Logic 2		
3/5	16	FSM 1	8 (FSM)	Wed. 3/13
3/7	17	FSM 2		
3/12	18	Digital Logic Review	9 (Ethics)	Fri. 3/15
3/14	19	CS Ethics & Impact Final Exam Review		

Lecture Outline

- More on Sequential Logic
- Class exercises

The Gated D Latch

- The gated D-Latch is very commonly used in electronic circuits in computer hardware, especially as a register because it's a circuit that holds memory!



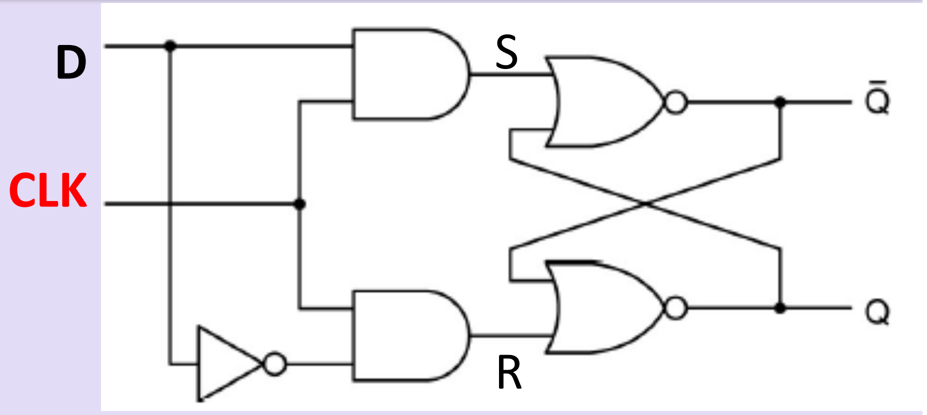
Whatever data you present to the input D,

the D-Latch will **hold** that value (*as long as input E is 0*)

You can **present** this value to output Q *as soon as input E is 1*.

Enabling the Latch Synchronously: **The Clocked D Latch**

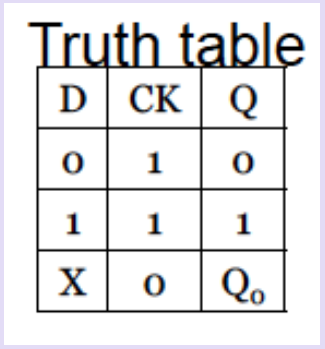
- If you apply a **synchronous clock** on input E, you get a **clocked D latch.**



- A clock is an input that cycles from 1 to 0, then back to 1 again in a set time period
  - e.g.: if a clock input cycles this in a period of 1 ms, we call it a 1 MHz clock (1 Hz = 1 / 1 second)

- ***Note 1***: When CLK is 0, both S and R inputs to the latch are 0 too, so the Q output holds its value whatever it is ($Q = Q_0$)

- ***Note 2***:When CLK is 1: if D = 1, then Q =1, if D = 0, then Q = 0

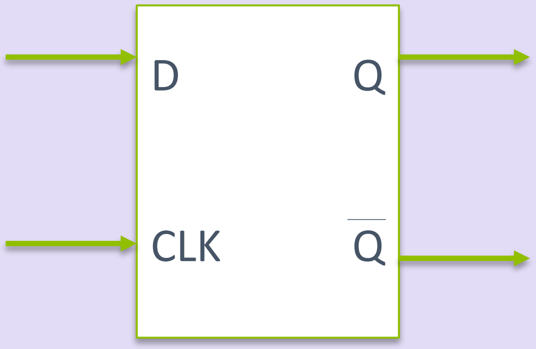


Truth table

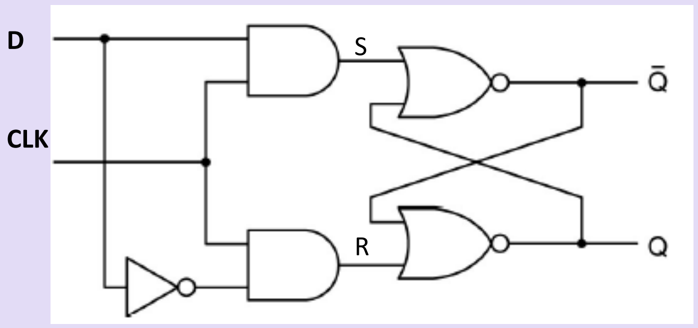
D	CK	Q
0	1	0
1	1	1
X	0	Q_0

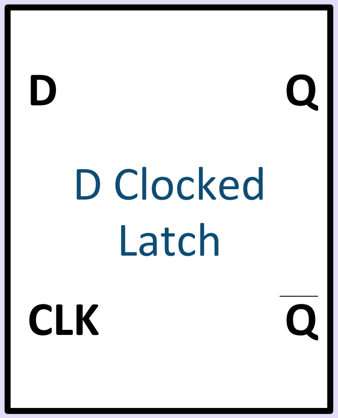
Clocked D Latch as Digital Sampler

- This clocked latch can be used as a "programmable" memory device that "samples" an input on a regular basis

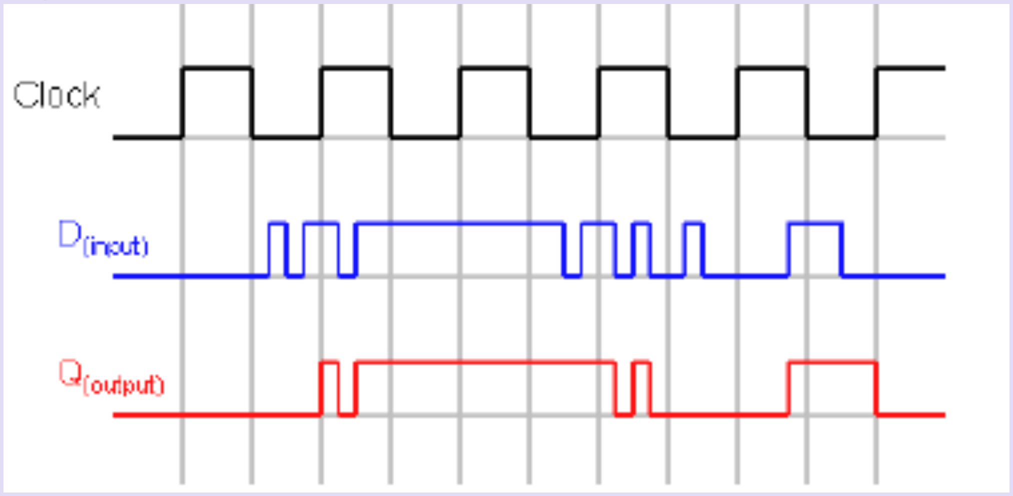


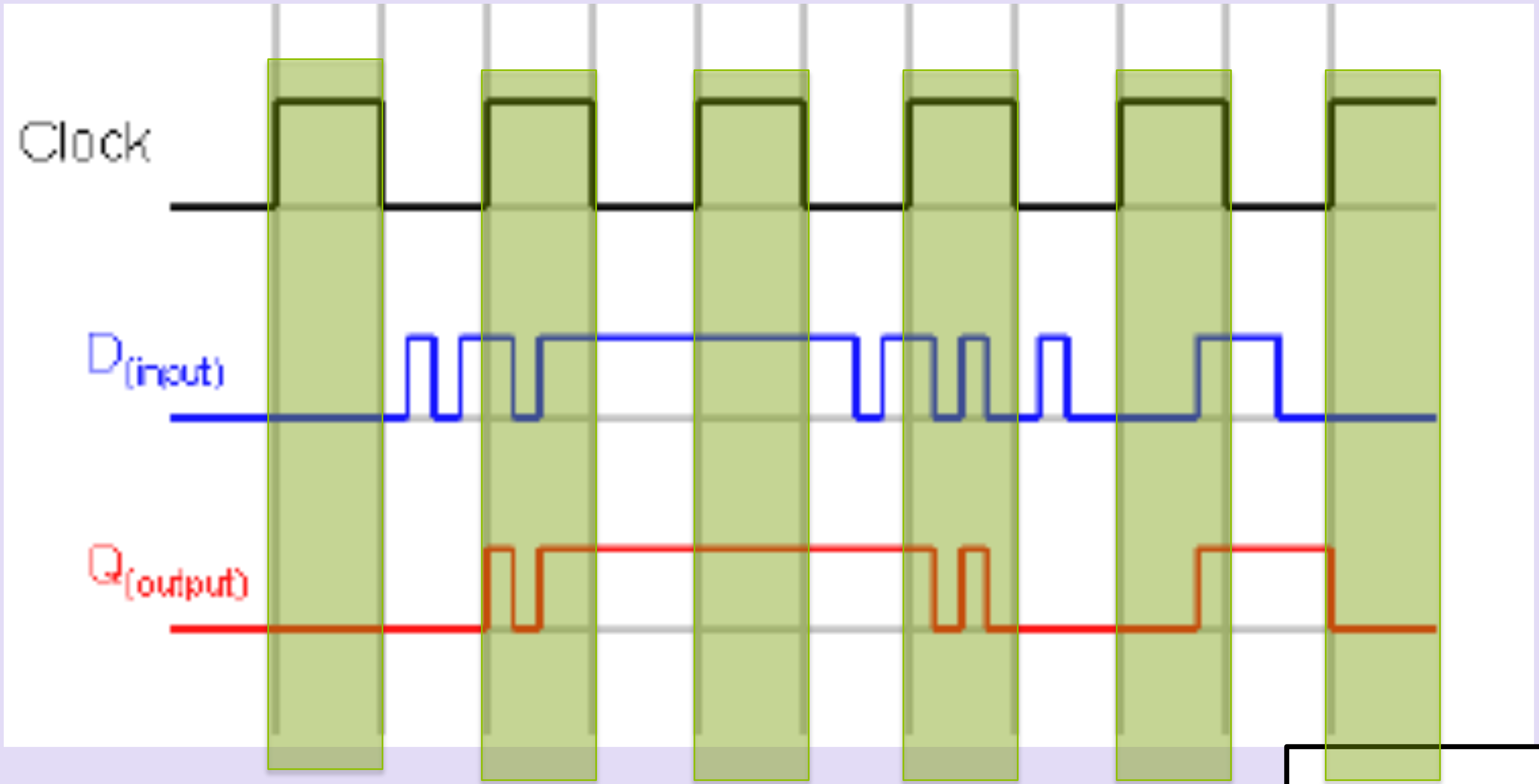
The Clocked D Latch
By Any Other Name…

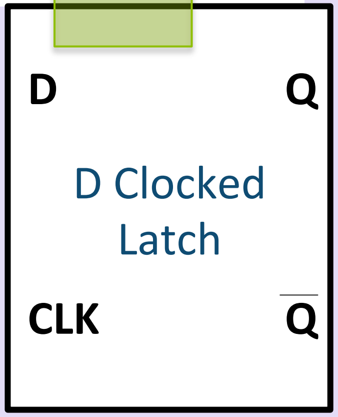




- Observing input and output "waveforms"



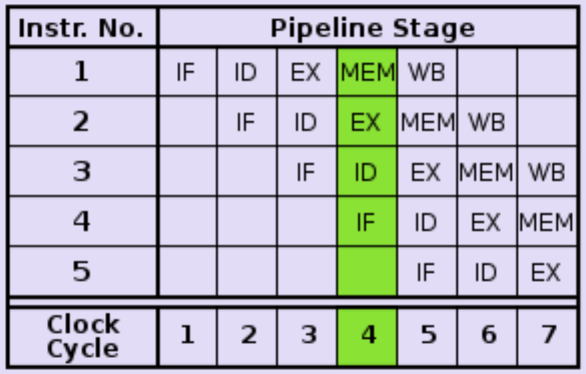

Clock
D(input)
Q(output)


D
Q
D Clocked Latch
CLK
$\overline{Q}$

The Joys of Sampling…

- Sampling data in a periodic way is advantageous
  - I can start designing more complex circuits that can help me do *synchronous* logical functions
    - *Synchronous*: in-time

- Very useful in *pipelining* designs used in CPUs
  - Pipelining: a technique that allows CPUs to execute instructions more efficiently – in parallel

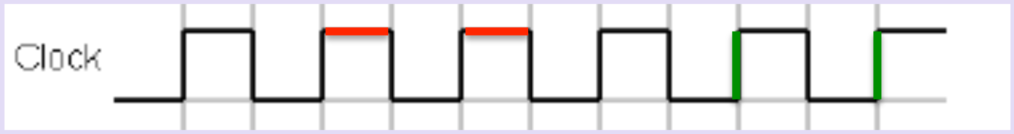


Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

Instruction fetch, decode, execute, memory access, register write

The Most Efficient Way to Sample Inputs

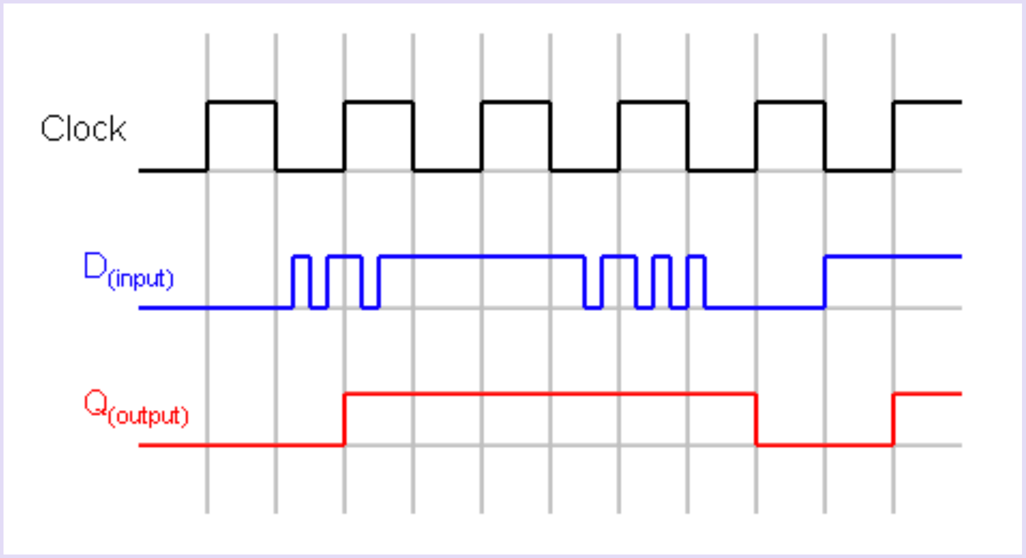
- Instead of sampling the input to the latch using a ***level*** of the clock…
  - That is, when the clock is “1” (or “0”)

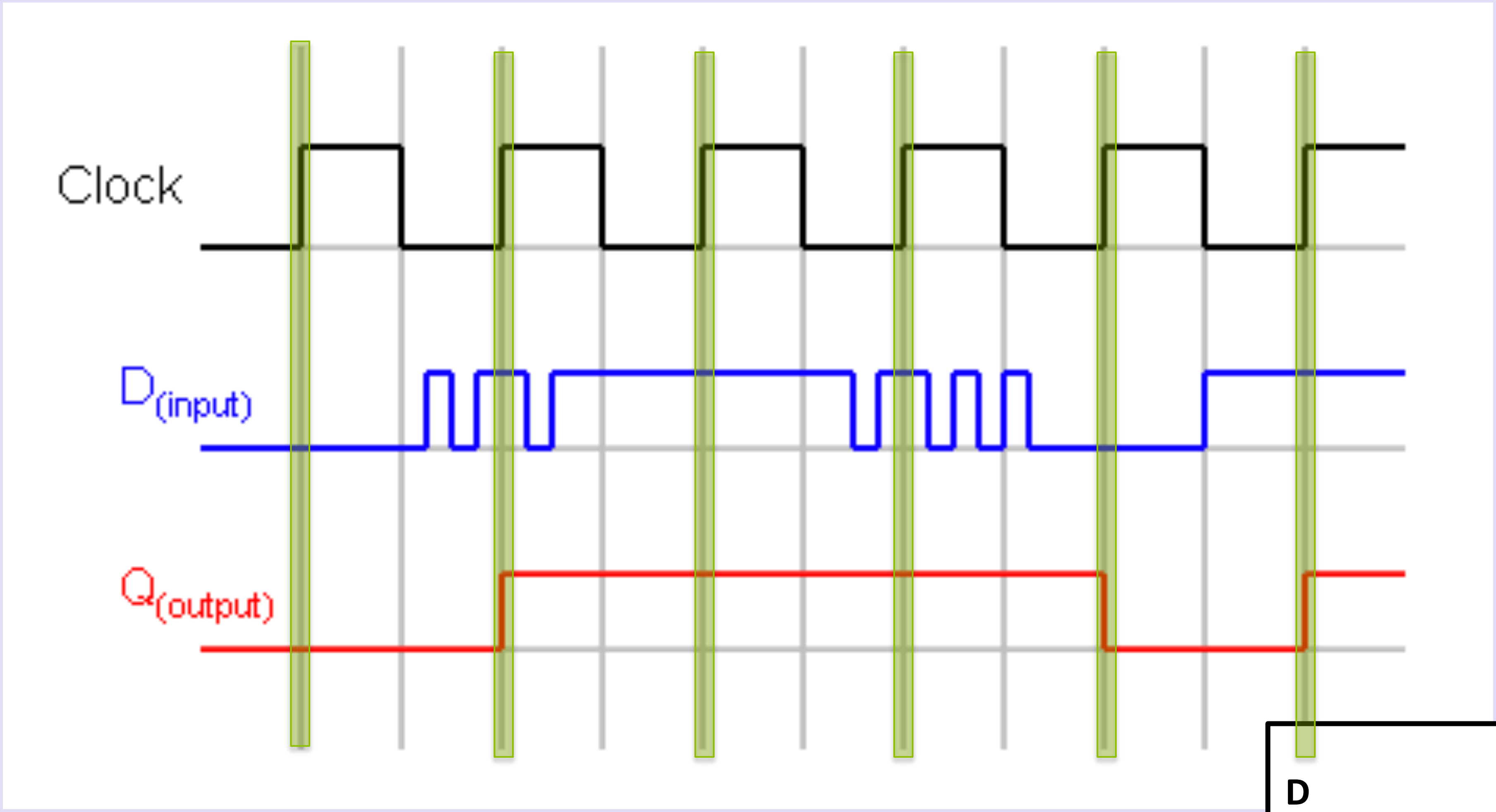


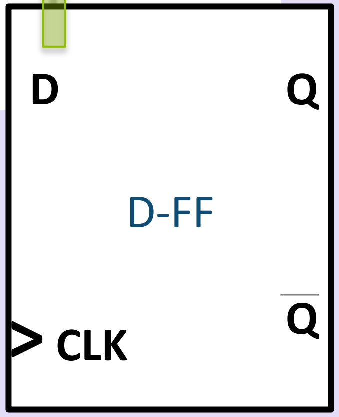
- … sample the input at the ***edge*** of the clock
  - That is, when the clock is transitioning from 0→1, called a *rising* or *positive* edge
    (or it could be done from 1→0,
    the *falling* edge a.k.a *negative* edge)
  - Why??

The D-FF

- When the input clock edge is *rising*, the input (D) is *captured* and placed on the output (Q)
  - Rising edge a.k.a positive edge FF
  - Some FF are negative edge FF (capture on the falling edge)

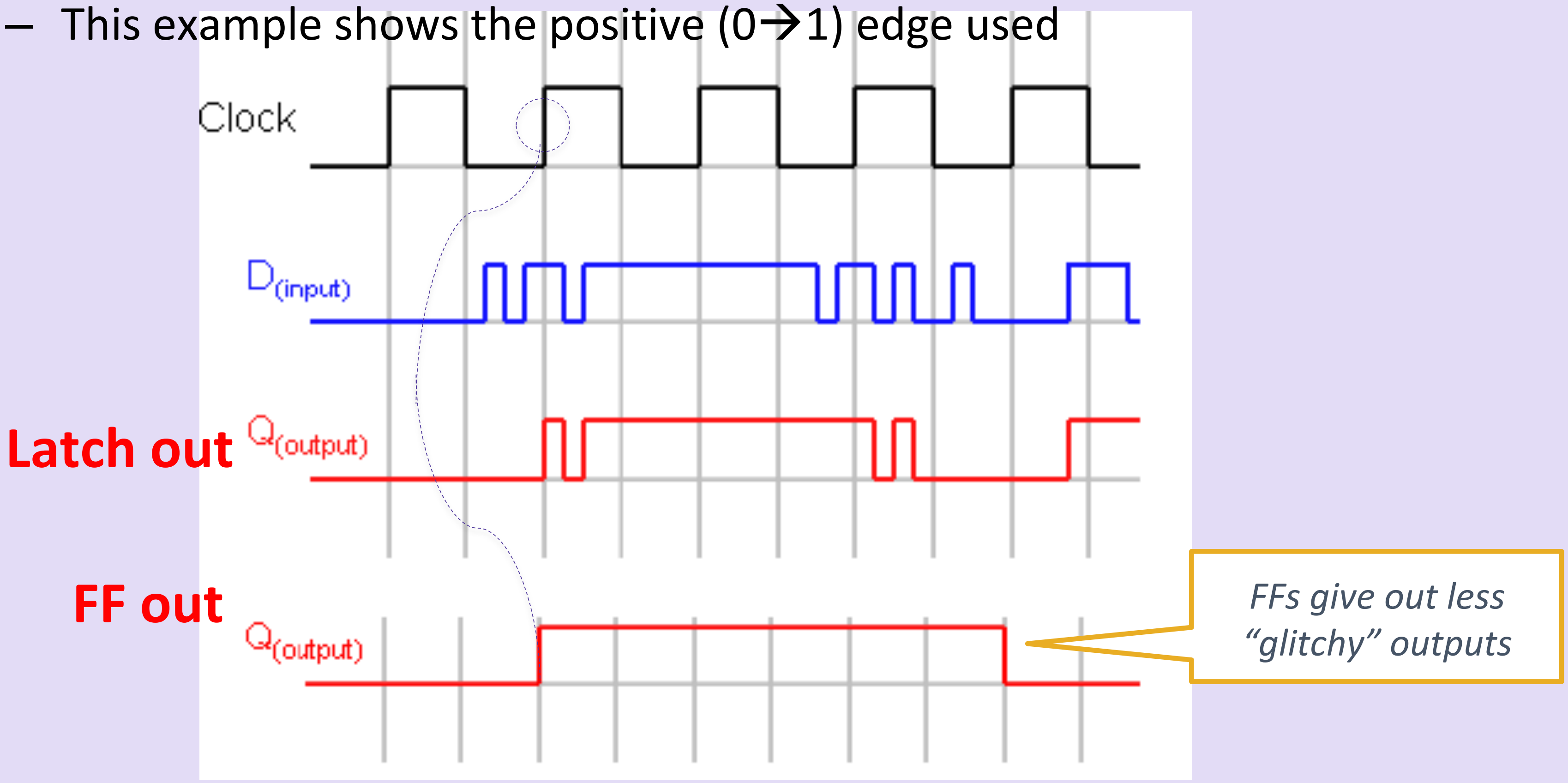



Clock
D(input)
Q(output)


D
Q
D-FF
CLK
$\overline{Q}$

Latches vs. FFs

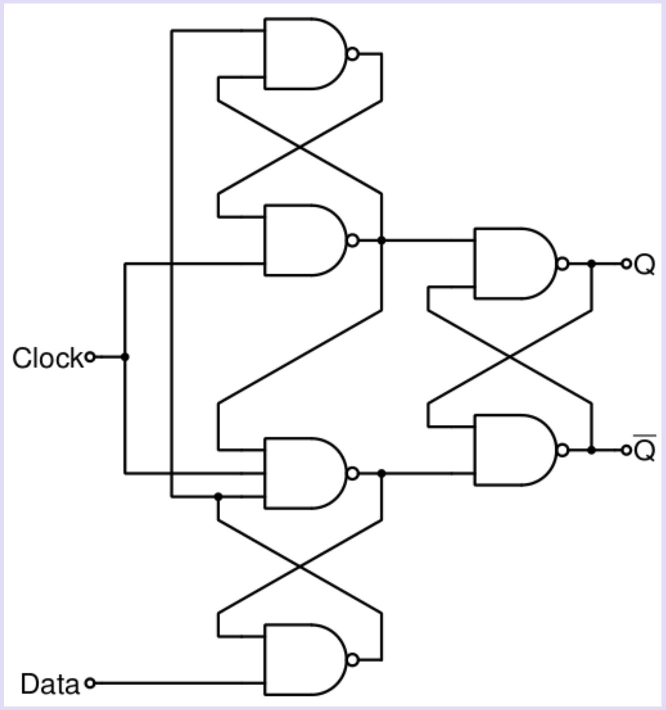
- Latches capture data on an entire 1 or 0 of the clock
- FFs capture data on the edge of the clock
  - This example shows the positive (0→1) edge used

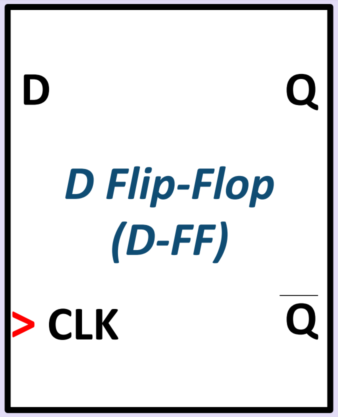


An Improvement on the Latch: **The D Flip-Flop**

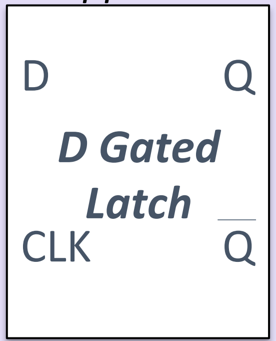
***Don't worry** about the circuit implementation details, but understand the use!*

The **D Flip-Flop** only changes the output (Q) into the input (D) at the ***positive edge*** (the 0 → 1 transition) of the clock





As opposed to:

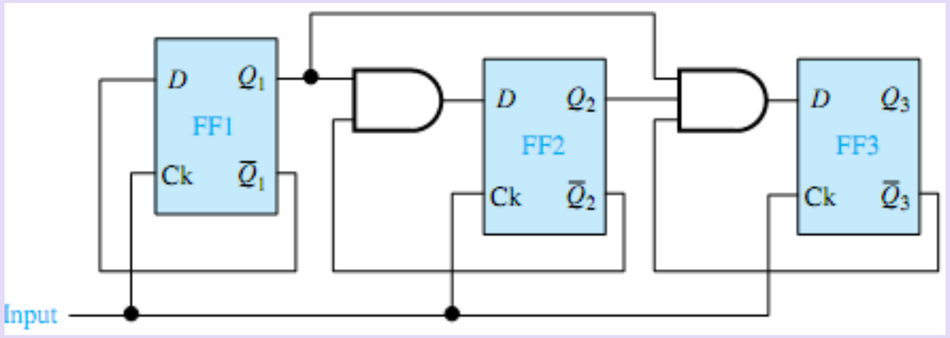


Note the (slight) difference in the 2 symbols…

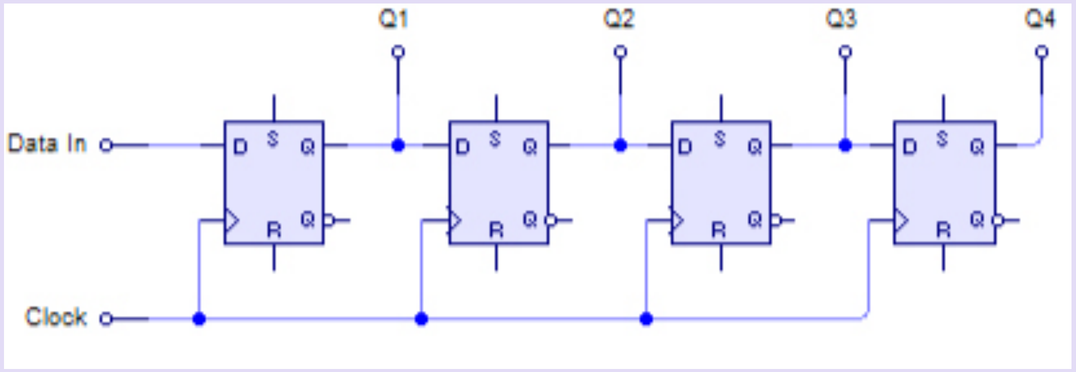
Again, don't worry about the circuit implementation details, but understand the uses!

Popular Uses for D-FFs

- Counter

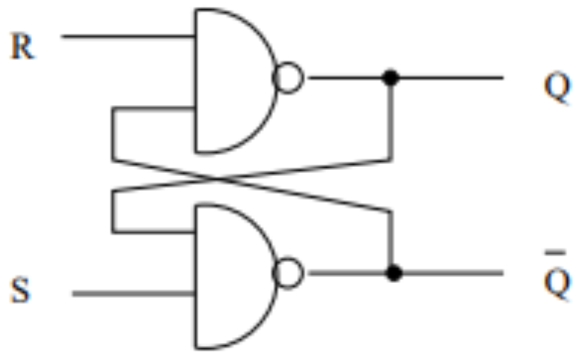


- Serial-to-Parallel converter



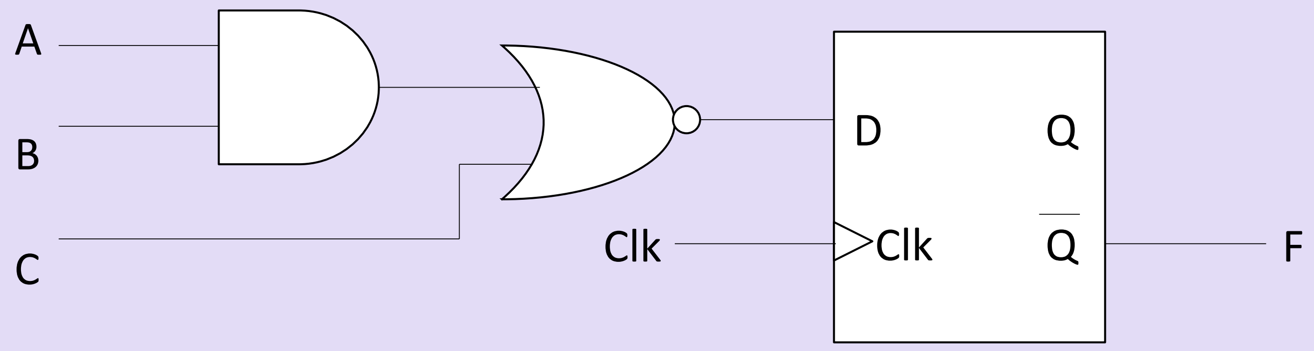
Class Exercise 1

The figure below shows an RS latch made out of NAND gates (rather than NOR gates). How do Q and $\overline{Q}$ depend on the RS inputs? i.e. verify that the circuit can indeed be used as a RS latch.



Class Exercise 2

Given waveforms for A, B, C, and Clk (see blackboard), determine the output waveform for F



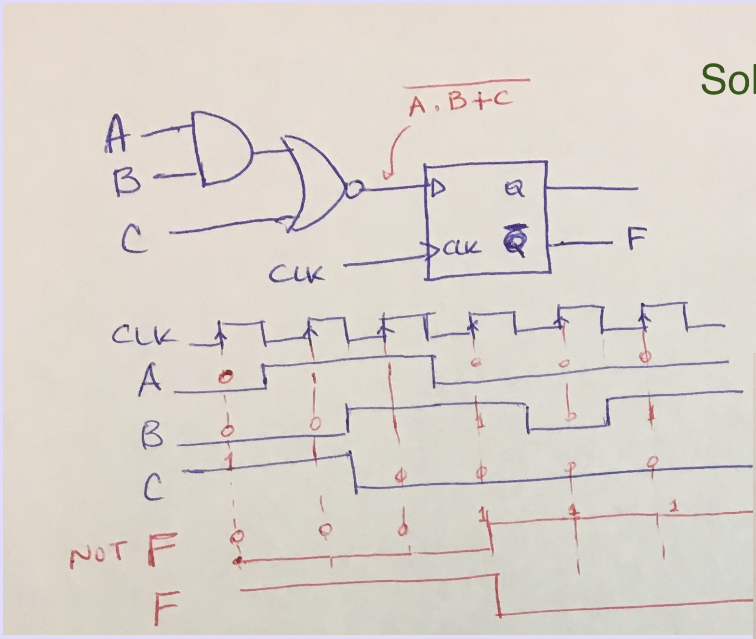
Class Exercise 3

- Let's design a 3-bit counter using D-FFs and logic gates.

- What's needed:
  - This counts 000 → 001 → 010 → … → 111 → 000
    - i.e. from 0 to 7 and then loops again to 0, etc…

- To describe this behavior, let's start with a T.T.
  - We'll utilize K-Maps, if needed to figure out what the "next states" look like based on "current states"
  - We'll translate that into a digital circuit design

Solutions to Class Exercises 2 and 3 From Lecture 14



	Current state			Next state		
	Q_2	Q_1	Q_0	Q_2^*	Q_1^*	Q_0^*
0	0	0	0	0	0	1
1	0	0	1	0	1	0
2	0	1	0	0	1	1
3	0	1	1	1	0	0
4	1	0	0	1	0	1
5	1	0	1	1	1	0
6	1	1	0	1	1	1
7	1	1	1	0	0	0

K-Map for Q_2^*

$Q_0 \backslash Q_2Q_1$	00	01	11	10
0			1	1
1		1		1

$$Q_2^* = Q_2.\overline{Q_0} + Q_2.\overline{Q_1} + \overline{Q_2}.Q_1.Q_0$$

for Q_1^*

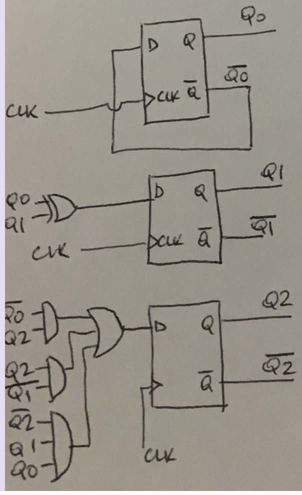
$Q_0 \backslash Q_2Q_1$	00	01	11	10
0		1	1	
1	1			1

$$Q_1^* = Q_1.\overline{Q_0} + \overline{Q_1}.Q_0 = Q_1 \oplus Q_0$$

for Q_0^*

$Q_0 \backslash Q_2Q_1$	00	01	11	10
0	1	1	1	1
1				

$$Q_0^* = \overline{Q_0}$$



YOUR TO-DOs

- Lab 7
  - Due back on Wednesday
  - Paper copy – not electronic
  - Drop off in the CS64 BOX in HFH 2nd Floor

</LECTURE>