

Sequential Logic Design

CS 64: Computer Organization and Design Logic

Lecture #15

Fall 2019

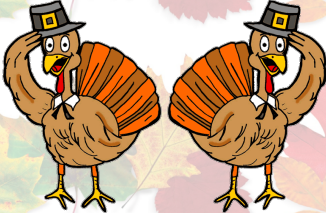

Ziad Matni, Ph.D.

Dept. of Computer Science, UCSB

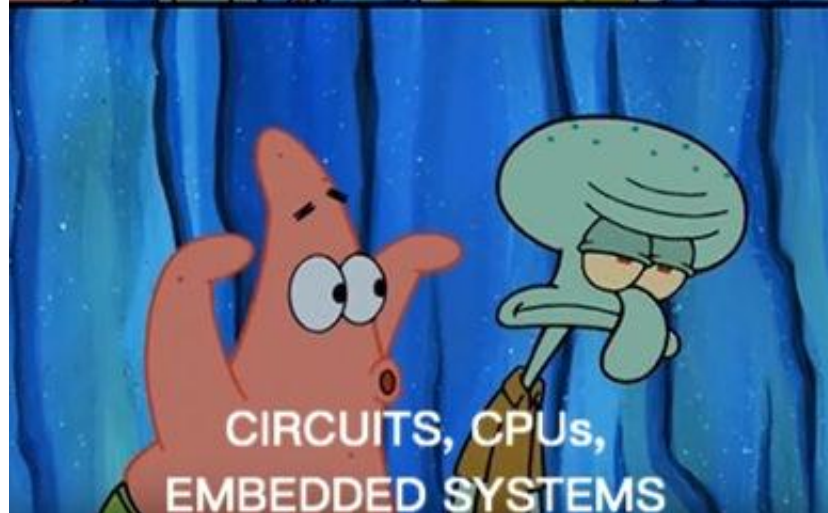
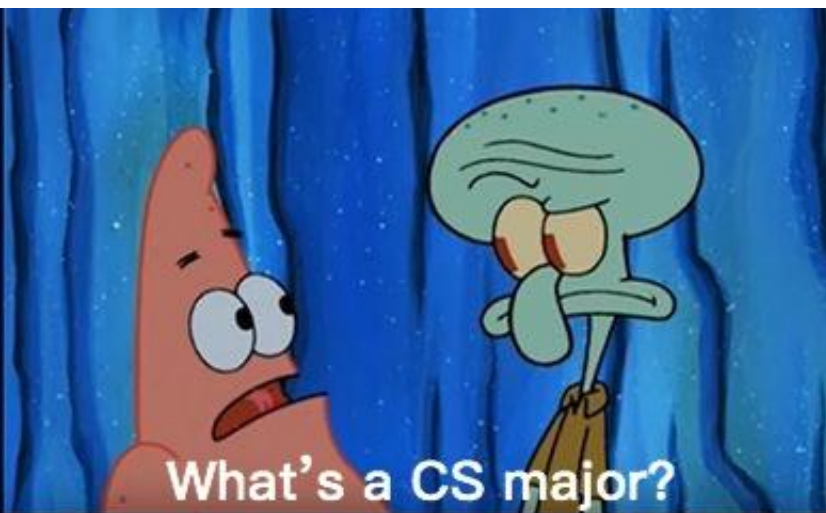
Administrative

- **NO CLASS ON WEDNESDAY! NO LAB ON FRIDAY!
LIFE IS GOOD!**
- Lab 7 is due on Wednesday
- New Lab 8 will be posted on Wednesday
(Don't Forget!!)

Schedule for the Rest of the Quarter

Date	Topic	Lab Due
W 11/20	Combinatorial Logic / Sequential Logic	Lab 6
M 11/25	Sequential Logic	
W 11/27	NO CLASS! 	 Lab 7
M 12/2	Finite State Machines (FSM)	
W 12/4	FSM / Ethics & Social Impact of CS	Lab 8
F 12/6	(no class)	Lab 9

Final Exam is on Tuesday, Dec. 10th at 12:00 PM in this classroom – details will follow

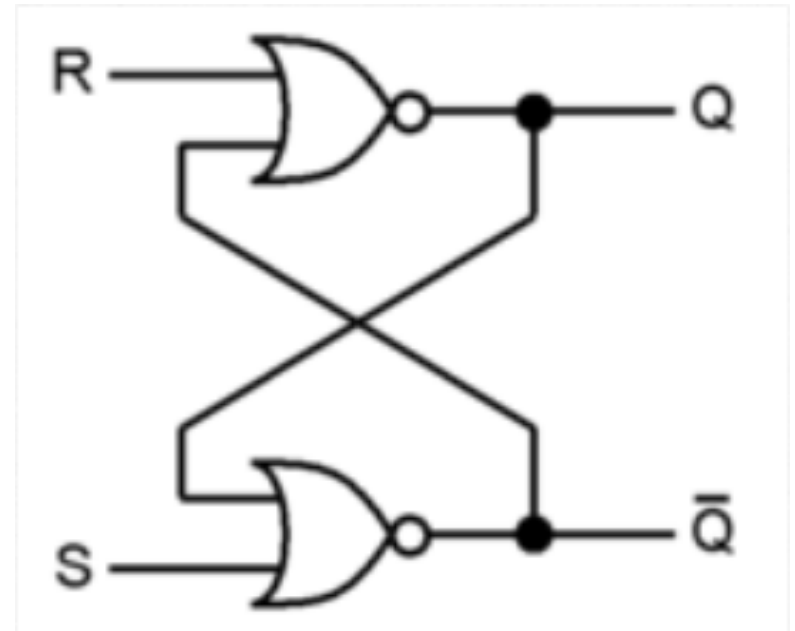


Lecture Outline

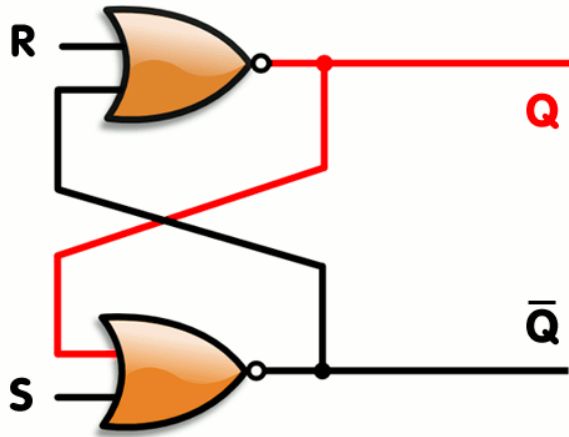
- **S-R Latch**
- Gated S-R Latch
- Gated D-Latch
- The D-Flip Flop (D-FF)

The S-R Latch

- Only involves 2 NORs
- The outputs are ***fed-back*** to the inputs
- The result is that the output state (either a 1 or a 0) is ***maintained*** *even if the input changes!*



How a S-R Latch Works



- Note that if one NOR input is **0**, the output becomes the inverse of the other input
- So, if output Q already exists and if **S = 0, R = 0**, then Q will remain at whatever it was before! (**hold output state**)
- If **S = 0, R = 1**, then Q becomes 0 (**reset output**)
- If **S = 1, R = 0**, then Q becomes 1 (**set output**)
- Making S = 1, R = 1 is not allowed (**gives an undetermined output**)

S	R	Q ₀	Comment
0	0	Q*	Hold output
0	1	0	Reset output
1	0	1	Set output
1	1	X	Undetermined

Consequences?

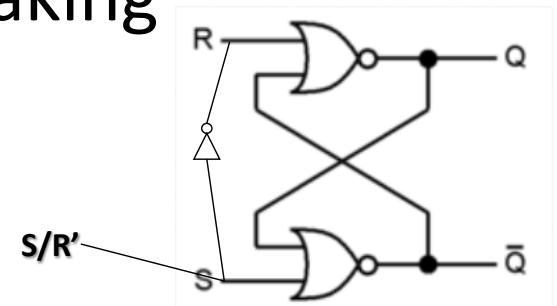
- As long as **S = 0 and R = 0**, the circuit output holds memory of its prior value (state)
- To change the output, just make
S = 1 (but also R = 0) to make the output 1 (set) **OR**
S = 0 (but also R = 1) to make the output 0 (reset)
- Just avoid S = 1, R = 1...

S	R	Q ₀	Comment
0	0	Q*	Hold output
0	1	0	Reset output
1	0	1	Set output
1	1	X	Undetermined

About that $S = 1, R = 1$ Case...

S	R	Q_0	Comment
0	0	Q^*	Hold output
0	1	0	Reset output
1	0	1	Set output
1	1	X	Undetermined

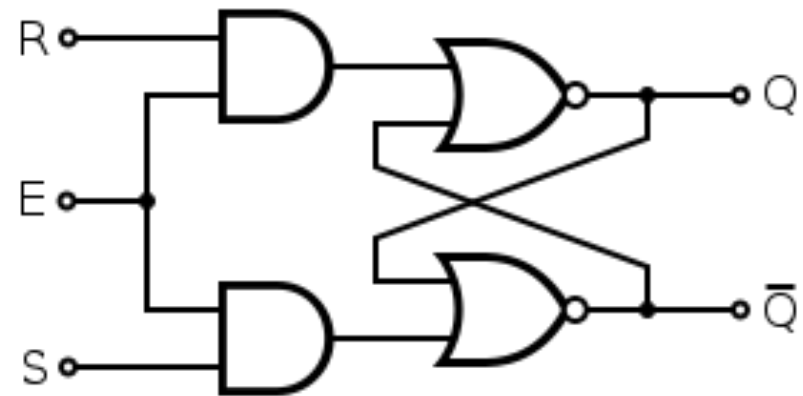
- What if we avoided it on purpose by making $R = \text{NOT}(S)$?
 - Where's the problem?



- This, by itself, precludes a case when $R = S = 0$
 - You'd need that if you want to preserve the previous output state!
- Solution: the ***clocked latch*** and ***the flip-flop***

Adding an “Enable” Input: The Gated S-R Latch

- Create a way to “gate” the inputs
 - R/S inputs go through *only if* an “enable input” (E) is 1
 - If E is 0, then the S-R latch gets $SR = 00$ and it holds the state of previous outputs



- So, the truth table would change from a “normal” S-R Latch:

S	R	Q_0	Comment
0	0	Q^*	Hold output
0	1	0	Reset output
1	0	1	Set output
1	1	X	Undetermined

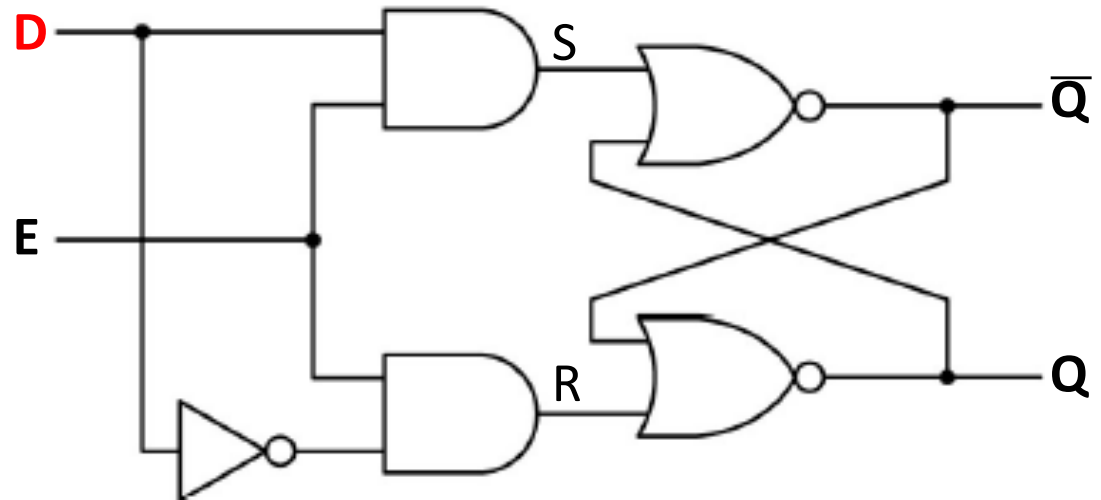


S	R	E	Q_0	Comment
X	X	0	Q^*	Hold output
0	1	1	0	Reset output
1	0	1	1	Set output

We got rid of the “undetermined” state!!! 😊😊😊

Combining R and S inputs into One: The Gated D Latch

- Force S and R inputs to *always be opposite of each other*
 - Make them the same as an input D, where $D = S$ and $\neg D = R$.



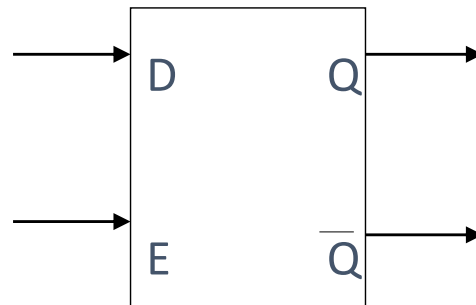
- Create a way to “gate” the D input
 - D input goes through only if an enable input (E) is 1
 - If E is 0, then hold the state of the previous outputs

D	E	Q_0	Comment
X	0	Q^*	Hold output
0	1	0	Reset output
1	1	1	Set output

We got rid of an extra input!!! 😊😊😊

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!



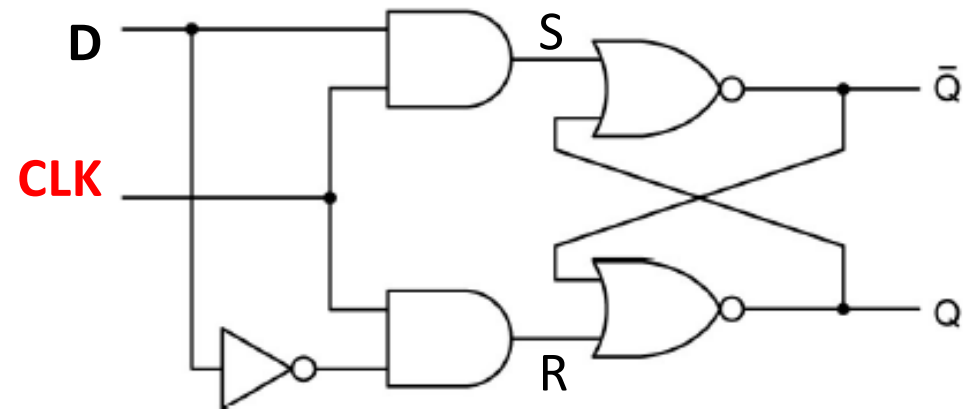
Whatever data you present to the input D,

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 ($1 \text{ Hz} = 1 / 1 \text{ second}$)
- **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 ($Q = Q_0$)



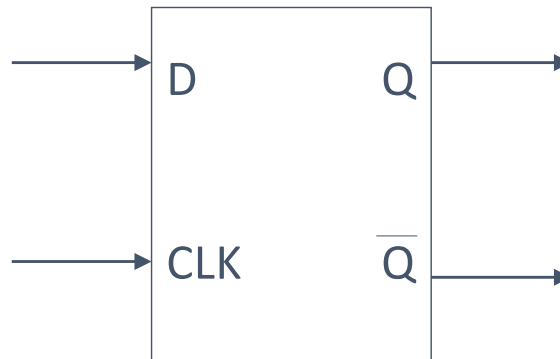
- **Note 2:** When CLK is 1:
if $D = 1$, then $Q = 1$,
if $D = 0$, then $Q = 0$

Truth table

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

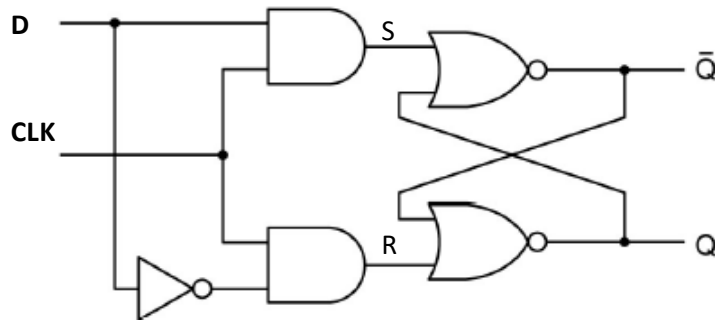
Clocked D Latch as Digital Sampler

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

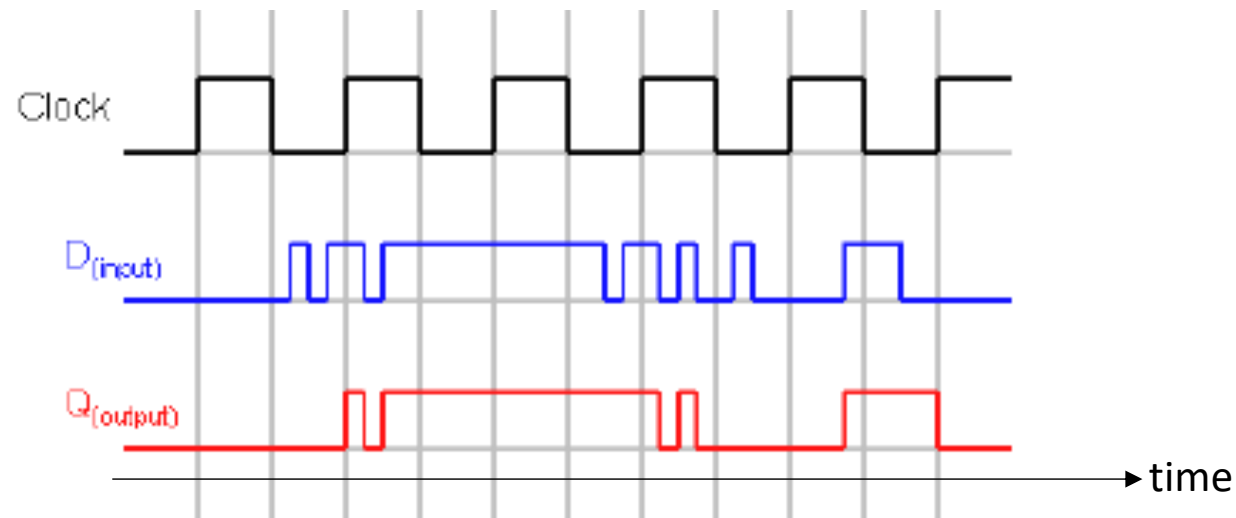


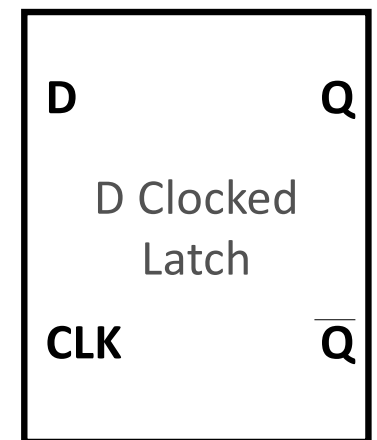
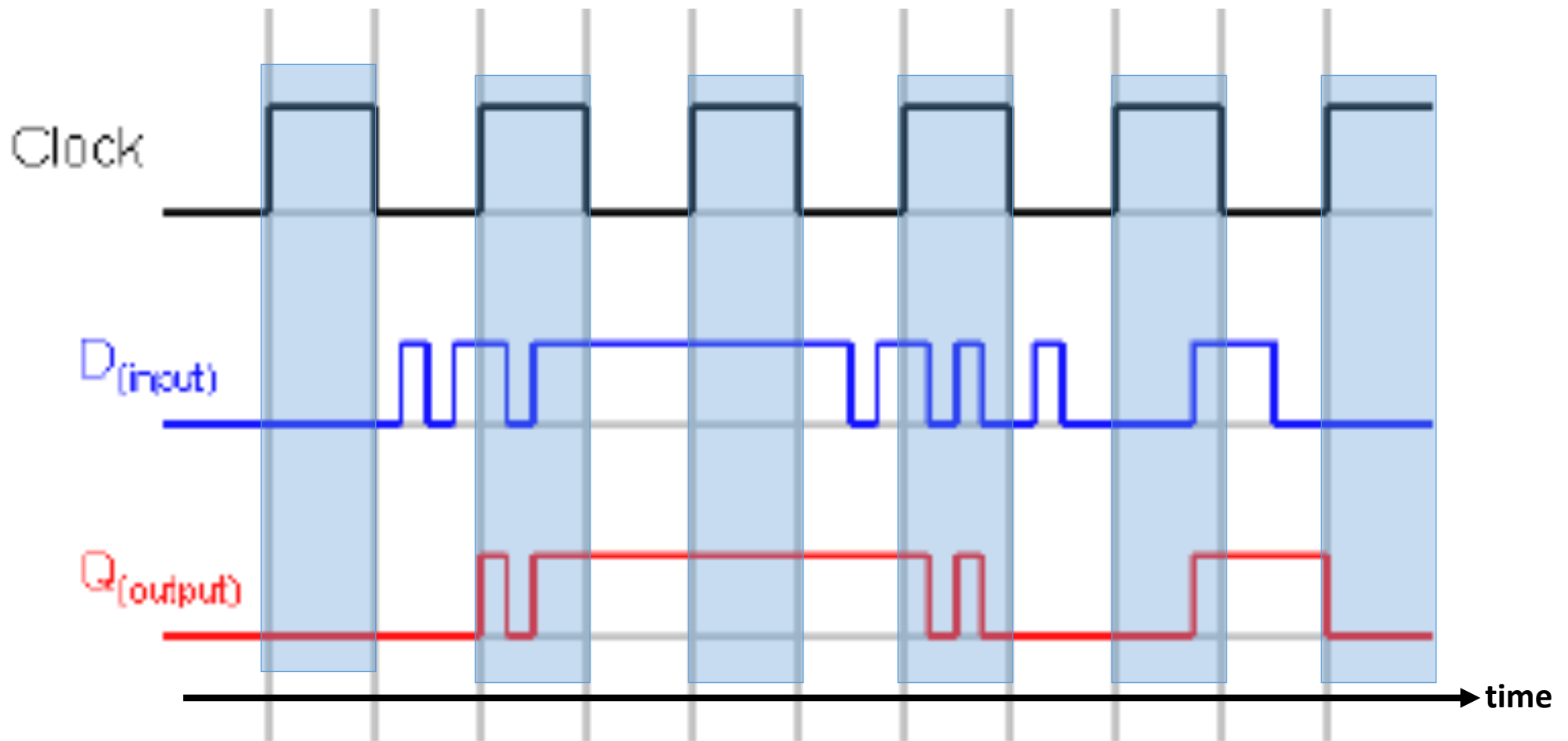
The Clocked D Latch

By Any Other Name...



- Observing input and output “waveforms”





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

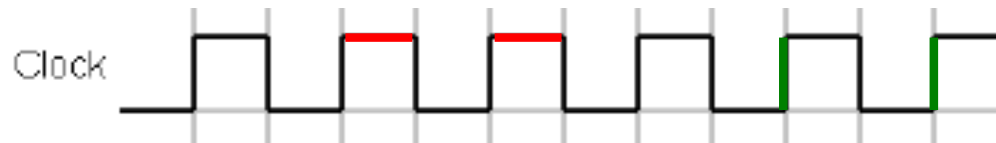
Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

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

The Most Efficient Way to Sample Inputs

Instead of sampling the latch input using a *level* of the clock...

- That is, when the clock is “1” (or “0”)

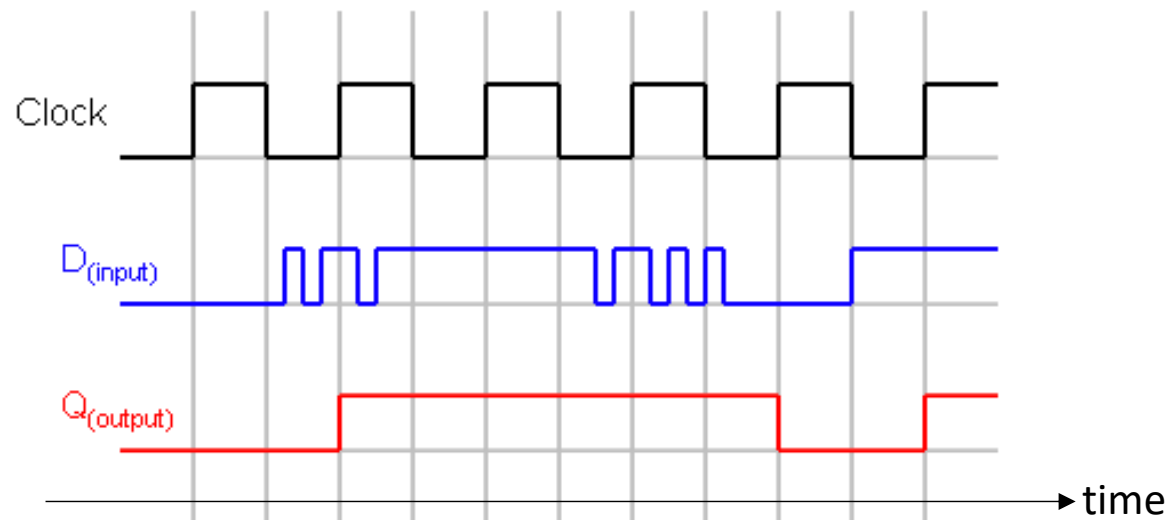


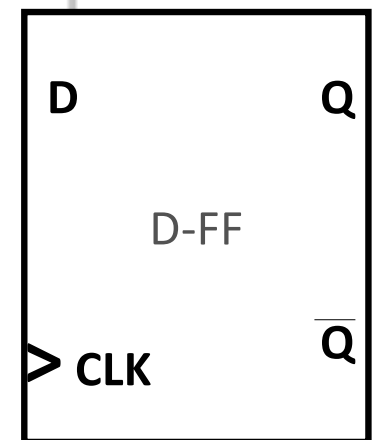
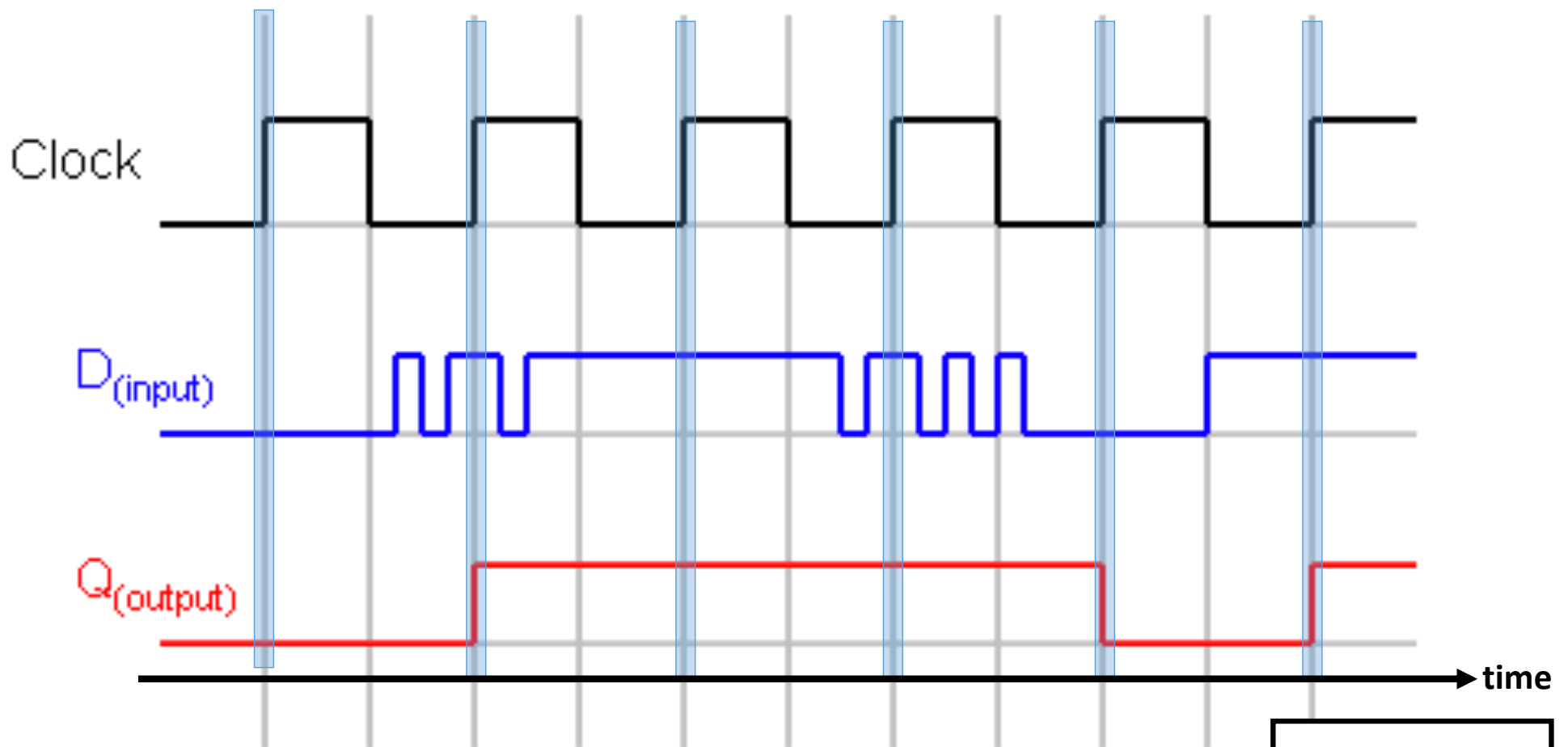
... sample the input at the *edge* of the clock

- That is,
when the clock is transitioning from $0 \rightarrow 1$, called a *rising* or *positive* edge
(or it could be done from $1 \rightarrow 0$, the *falling* edge a.k.a *negative* edge)
- Why is this more efficient??

The D-FF

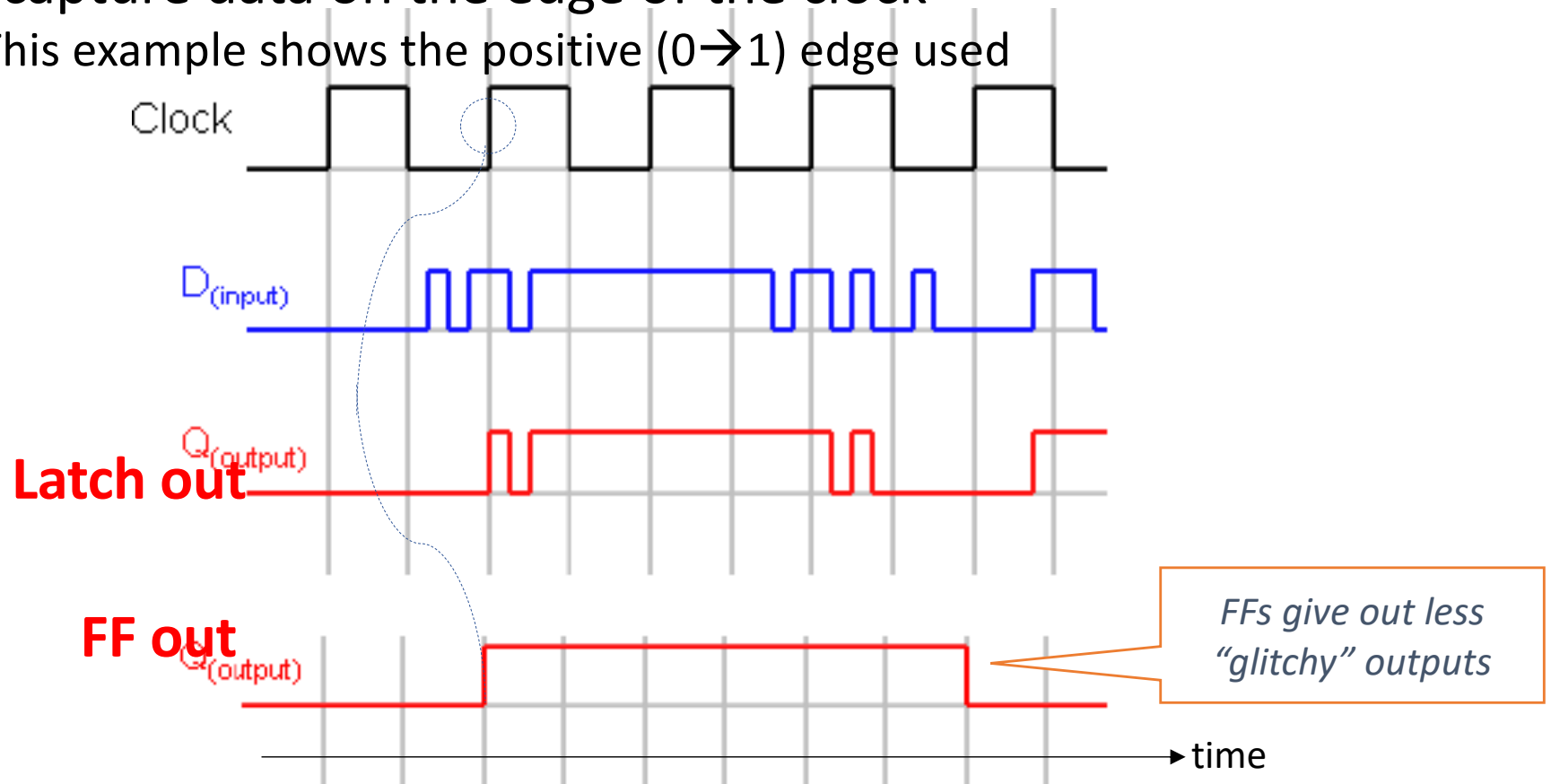
- When the input clock edge is *rising*, the input (D) is *captured* and placed on the output (Q)
 - Rising edge a.k.a positive edge FF
 - Some FF are negative edge FF (capture on the falling edge)





Latches vs. FFs

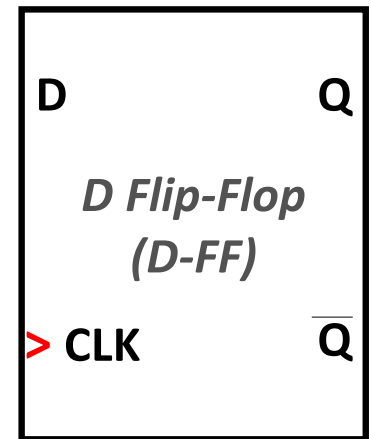
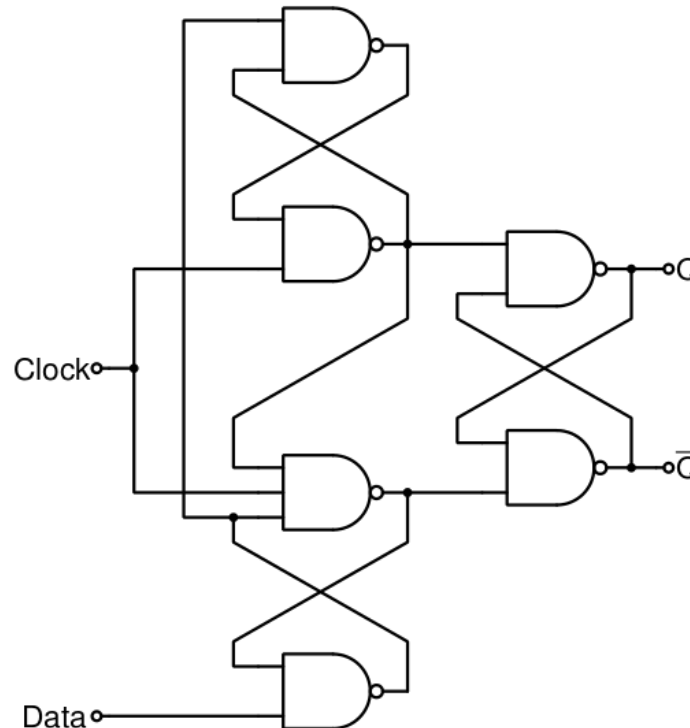
- Latches capture data on an entire 1 or 0 of the clock
- FFs capture data on the edge of the clock
 - This example shows the positive ($0 \rightarrow 1$) edge used



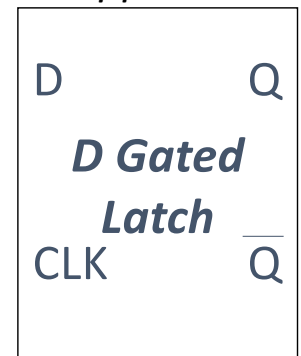
An Improvement on the Latch: The D Flip-Flop

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 $0 \rightarrow 1$ transition) of the clock



As opposed to:

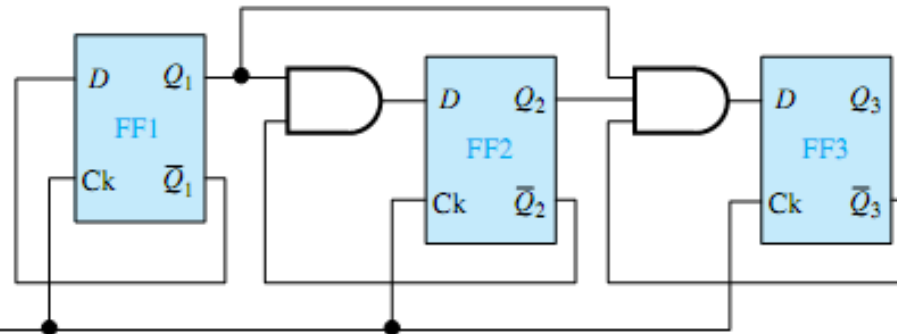


Note the (slight) difference in the 2 symbols...

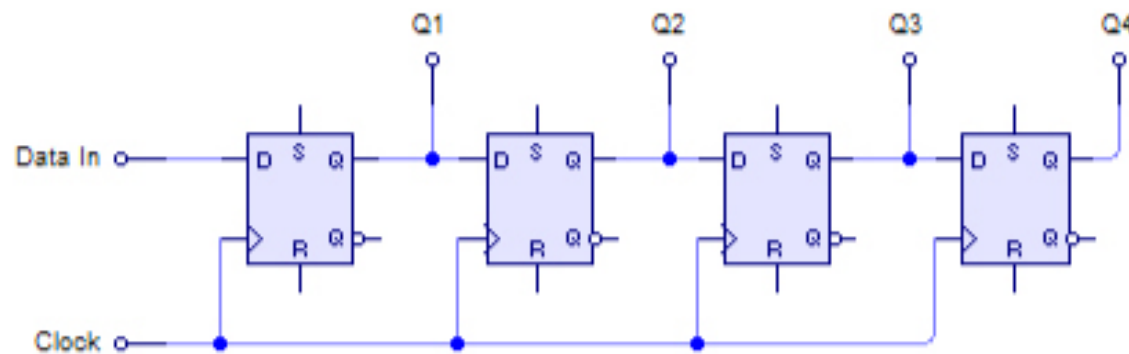
Again, don't worry about the circuit implementation details, but understand the uses!

Popular Uses for D-FFs

- Counter

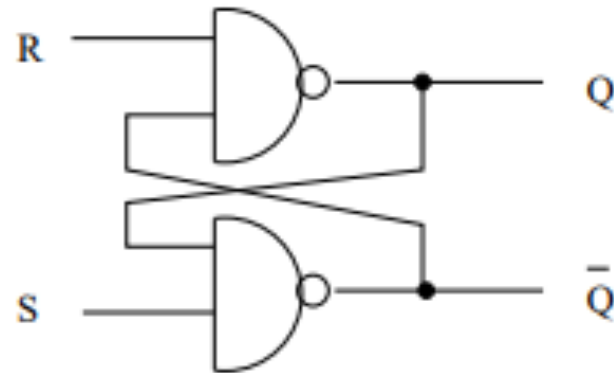


- Serial-to-Parallel



Class Exercise 1

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



Class Exercise 2

- Let's design a 3-bit counter using D-FFs and logic gates.
- What's needed:
 - This counts $000 \rightarrow 001 \rightarrow 010 \rightarrow \dots \rightarrow 111 \rightarrow 000$
 - 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

YOUR TO-DOs

- Lab 7 is due on Wednesday (11/27)!
- Look for Lab 8 posted online by Wed.
- Due NEXT Wednesday (12/4)



HAPPY THANKSGIVING!!!!



</LECTURE>