

Spiral 2-5

Sequential Logic Constructs

Learning Outcomes

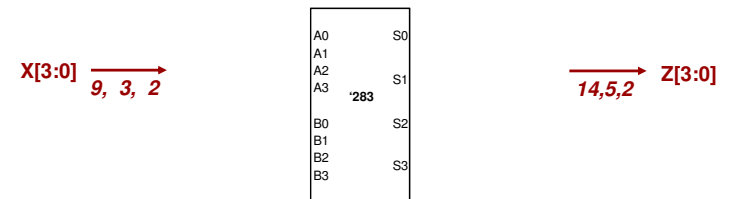
- I understand how a bistable works
 - I understand how a bistable holds, sets, and resets
- I understand the issues that glitches pose to bistables and the need for latches
- I understand the difference between level-sensitive and edge-sensitive
- I understand how to create an edge-triggered FF from 2 latches

How sequential building blocks work

BISTABLES, LATCHES, AND FLIP-FLOPS

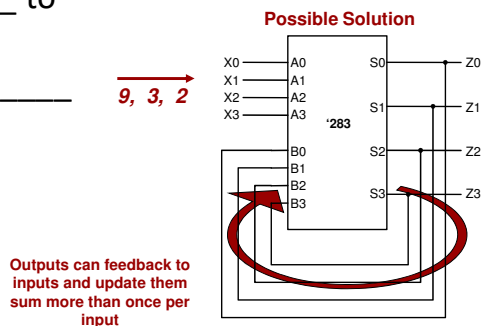
Sequential Logic

- Suppose we have a sequence of input numbers on X[3:0] that are entered over time that we want to sum up
- Possible solution: Route the outputs _____ so we can add the current sum to the input X



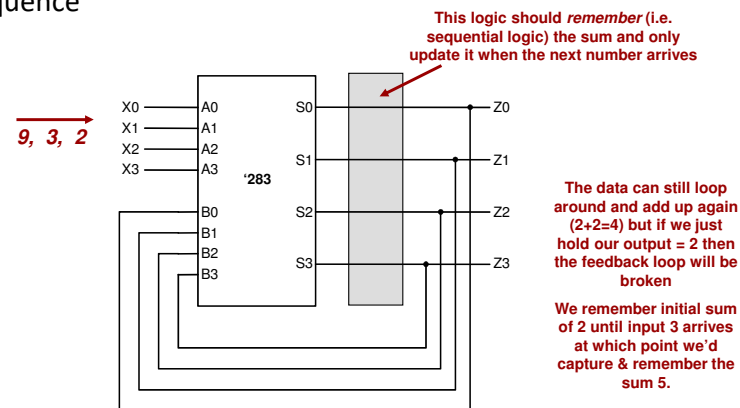
Sequential Logic

- Problem 1: No way to _____
- Problem 2: Outputs can _____ to inputs and be added _____ per input number



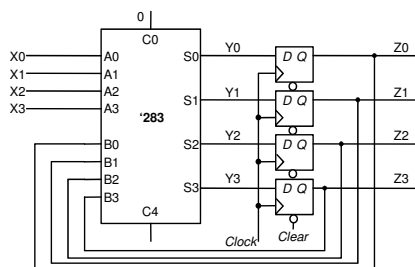
Sequential Logic

- Add logic at outputs to just capture and remember the new sum until we're ready to input the next number in the sequence



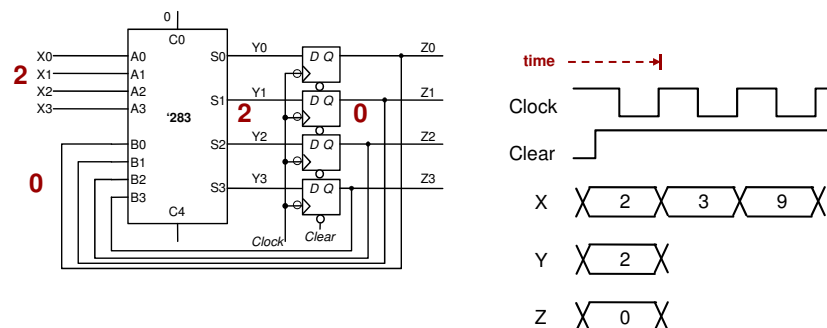
Sequence Adder

- If X changes _____ then Z should also change _____ per cycle
- That is why we will use a register (flip-flops) to ensure the outputs can only update _____ per cycle



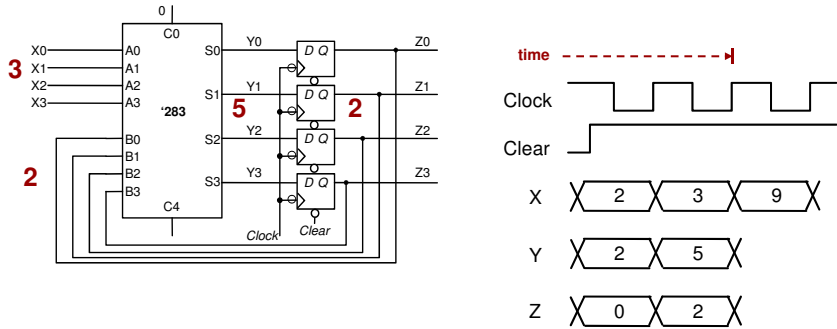
Sequence Adder

- The 0 on Clear will cause Z to be initialized to 0, but then Z can't change until the next positive edge
- That means we will just keep adding 0 + 2 = 2



Sequence Adder

- At the edge the flip-flops will sample the D inputs and then remember 2 until the next positive edge
- That means we will just keep adding $3 + 2 = 5$



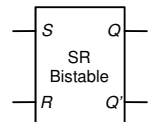
Sequential Logic

- But how do flip-flops work?
- Our first goal will be to design a circuit that can remember one bit of information
- Easiest approach...
- But how do you change the input?
 - A signal should only have one driver

SET/RESET BISTABLES

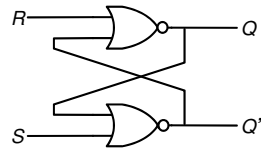
RS (or SR) Bistable

- Terminology
 - _____ = Force output to 1
 - _____ = Force output to 0
- Set/Reset Bistable Circuit
 - A circuit that can set or reset its output...
 - ...but then can remember its current output value once the inputs are removed



RS (SR) Bistable

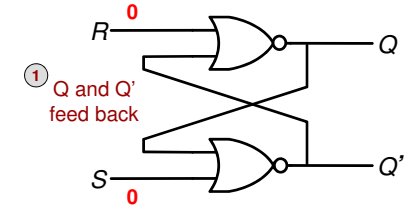
- Cross-Connected NOR gates (outputs feed back to inputs)
- When Set = 1, Q should be forced to ___
- When Reset = 1, Q should be forced to ___
- When neither are 1, Q should _____ at its present value



RS (SR) Bistable

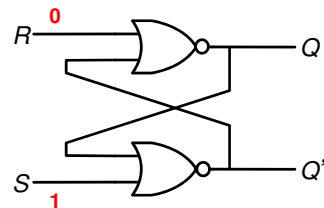
Always start your analysis from the output Q and cycle it around the loop

| S | R | Q | Q' |
|---|---|---|----|
| 0 | 0 | | |
| 1 | 0 | | |
| 0 | 1 | | |
| 1 | 1 | | |



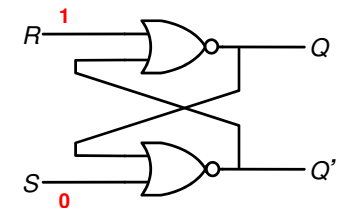
RS (SR) Bistable

| S | R | Q | Q' |
|---|---|----------------|------------------|
| 0 | 0 | Q ₀ | Q ₀ ' |
| 1 | 0 | | |
| 0 | 1 | | |
| 1 | 1 | | |



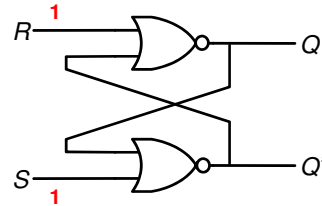
RS (SR) Bistable

| S | R | Q | Q' |
|---|---|----------------|------------------|
| 0 | 0 | Q ₀ | Q ₀ ' |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | | |



RS (SR) Bistable

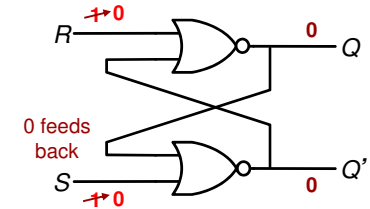
| S | R | Q | Q' |
|---|---|----------------|------------------|
| 0 | 0 | Q ₀ | Q ₀ ' |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | | |



• 1,1 combination violates the _____

RS (SR) Bistable

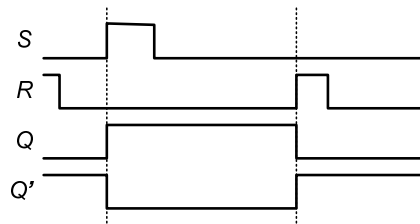
| S | R | Q | Q' |
|---|---|----------------|------------------|
| 0 | 0 | Q ₀ | Q ₀ ' |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 |
| | | (illegal) | (illegal) |



• It cannot be "remembered"...meaning as soon as R or S goes to 0 then it will set and reset; if R and S go to 0 at the same instant, then we will have unpredictable behavior

Another Waveform

- Waveform for an SR bistable with active-hi inputs (cross-connected NOR gates)

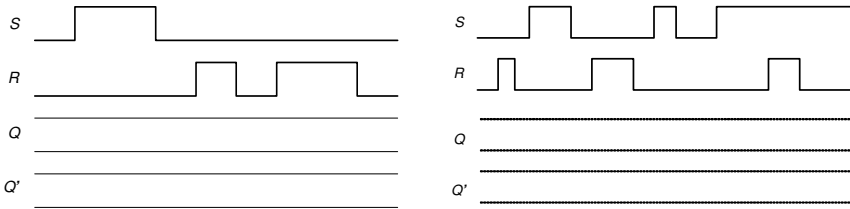


Criteria for a Bistable

1. Able to independently _____ => Force Q=1
2. Able to independently _____ => Force Q=0
3. Able to _____ => Q = Q₀

Exercises

- Complete the waveforms below for an RS bistable with active hi inputs

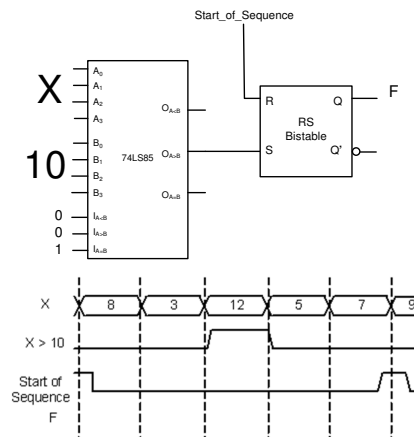


A problem with bistables

MOTIVATION FOR LATCHES

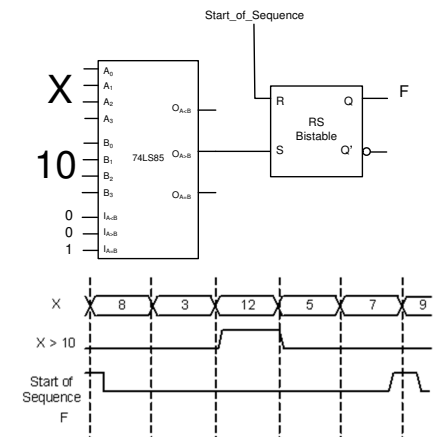
Problem w/ Bistables

- Bistables will remember input values whether we _____
- Imagine we connect the Set input to the output of a comparator to check if any number in a sequence is > 10 and then remember that



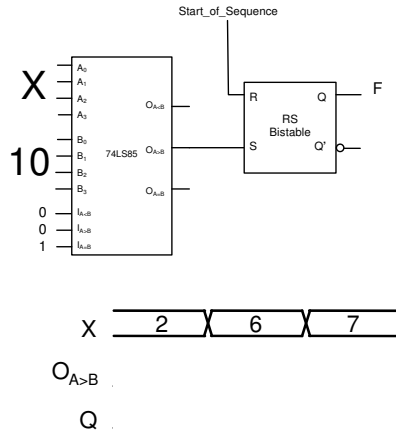
Problem w/ Bistables

- When inputs change in a combinational circuit, the outputs may transition back and forth between 1 and 0
- This is called a "_____ " and is caused due to the propagation delay of the _____ logic



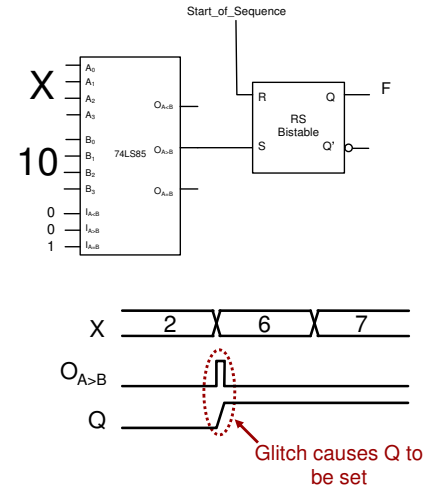
Problem w/ Bistables

- Suppose we get a sequence: 2,6,7
- At the end Q should still = 0 since no numbers > 10
- However, if when the inputs change a small glitch occurs on $A > B$, the bistable will



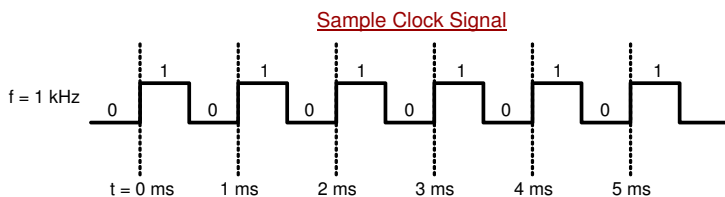
Problem w/ Bistables

- Output should have been 0 at end of sequence
- Problem: Glitch was remembered
- Need some way to _____ inputs until they are _____ and _____



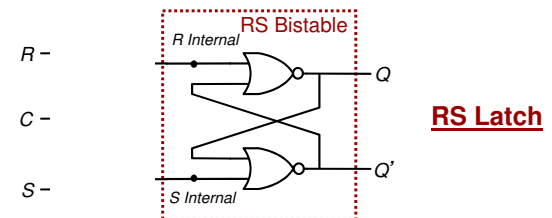
Clock Signals

- A clock signal is an alternating sequence of 1's and 0's
- It can be used to help ignore the inputs of a bistable when there might be glitches or other invalid values
- Idea:
 - When clock is 0, ignore inputs
 - When clock is 1, respond to inputs



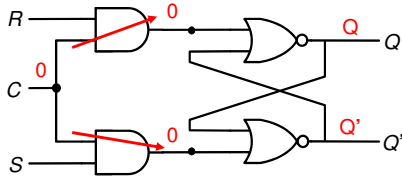
Latches

- Latches are bistables that include a new _____
- The clock input will tell the latch when to _____ the inputs (when C=0) and when to _____ to them (when C=1)



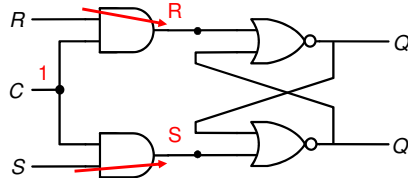
Latches

RS Latch (C=0)



C=0 causes S=R=0 and thus Q and Q' remain unchanged

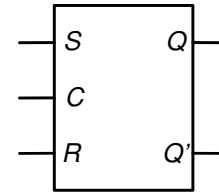
RS Latch (C=1)



C=1 allows S,R to pass and thus Q and Q' are set, reset or remain unchanged based on those inputs

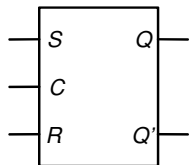
SR-Latch

- When C = 0, Q holds (remembers) its value
- When C = 1, Q responds as a normal SR-bistable

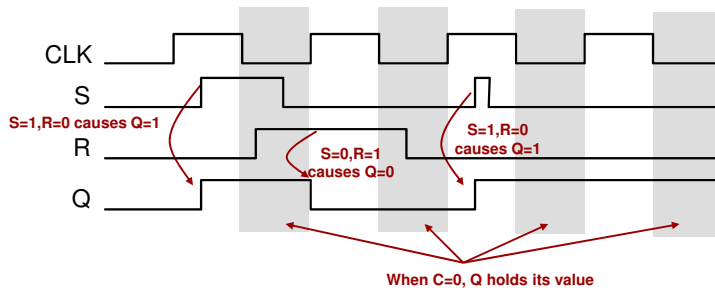


| CLK | S | R | Q | Q' |
|-----|---|---|----------------|------------------|
| 0 | x | x | Q ₀ | Q ₀ ' |
| 1 | 0 | 0 | Q ₀ | Q ₀ ' |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | illegal | illegal |

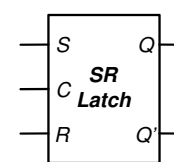
SR-Latch



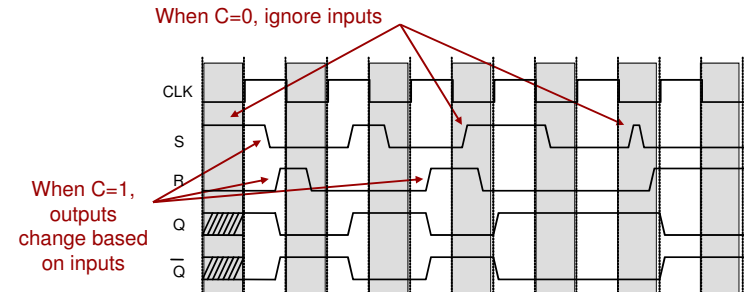
| CLK | S | R | Q | Q' |
|-----|---|---|----------------|------------------|
| 0 | x | x | Q ₀ | Q ₀ ' |
| 1 | 0 | 0 | Q ₀ | Q ₀ ' |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | illegal | illegal |



RS (SR) Latches

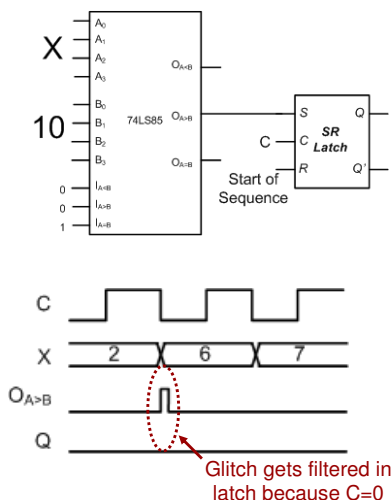


| C | S | R | Q | Q' |
|---|---|---|----------------|------------------|
| 0 | x | x | Q ₀ | Q ₀ ' |
| 1 | 0 | 0 | Q ₀ | Q ₀ ' |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | illegal | illegal |



Solution with Latches

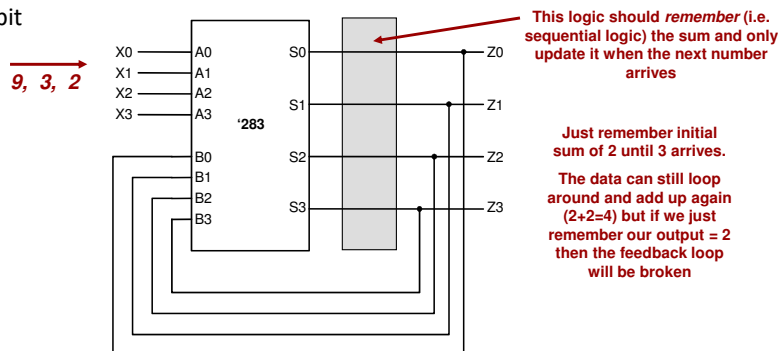
- $C = \underline{\quad}$ when _____ change
 - In fact, in a real digital system, it is C 's transition to 0 that triggers the inputs to change
 - Glitches occur during this time and are filtered
- When $C = 1$, inputs are stable and no glitches will occur



MOTIVATION FOR D-LATCHES

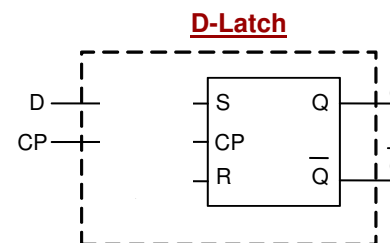
Adding a Sequence of Numbers

- Back to our example of adding a sequence of numbers
 - RS latches require 2 inputs (S,R) per output bit Q
 - In this scenario, we only have 1-bit of input per output
 - We'll modify an SR latch to become a latch that can remember 1 input bit



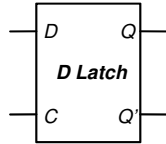
D-Latches

- D-Latches (Data latches) _____ data when the clock is _____ and _____ data when the clock is _____
- D-Latch is just an SR Latch with the D-input run into the S-input and _____ into the R-input



D-Latches

Hold Mode



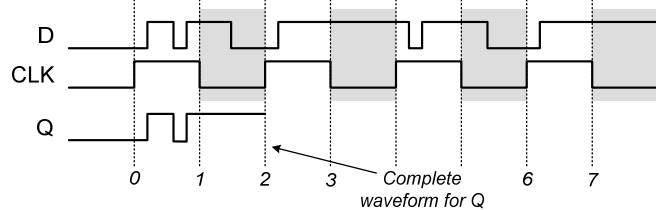
| C | D | Q | Q' |
|---|---|----------------|------------------|
| 0 | x | Q ₀ | Q ₀ ' |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Hold Mode
Transparent Mode

D-LATCH 7475

As clock is LOW, don't look at the D input

Triggering Rule: The Q output follows the D input (i.e. Q=D) when the clock or gate input is high (i.e. the latch is enabled). When the latch is disabled (Clock = LOW) the output remains put.



D-Latches

- When $C = 0$, $Q = Q_0$
 - Hold mode => Q stays the same
- When $C = 1$, $Q = D$
 - Transparent mode => Q follows D

Bistables vs. Latches

Bistables

- No clock input
 - outputs can change anytime the inputs change (including glitches)

Latches

- Clock/Gate/Enable input
 - outputs can only change during clock high/low time

Notation

- To show that Q remembers its value we can put it in the past tense:
 - $Q = Q_0$ (Current Value of Q = Old Value of Q)
- OR put it in the future tense
 - $Q^* = Q$ (Next Value of Q = Current Value of Q)

Indicates "next-value" of Q

| C | D | Q | Q' |
|---|---|----------------|------------------|
| 0 | x | Q ₀ | Q ₀ ' |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

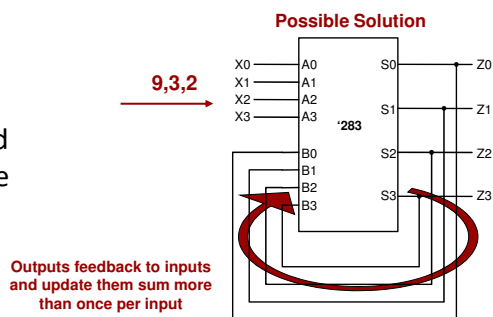
Current Value = Old Value

| C | D | Q* | Q'* |
|---|---|----|-----|
| 0 | x | Q | Q' |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Next Value = Current Value

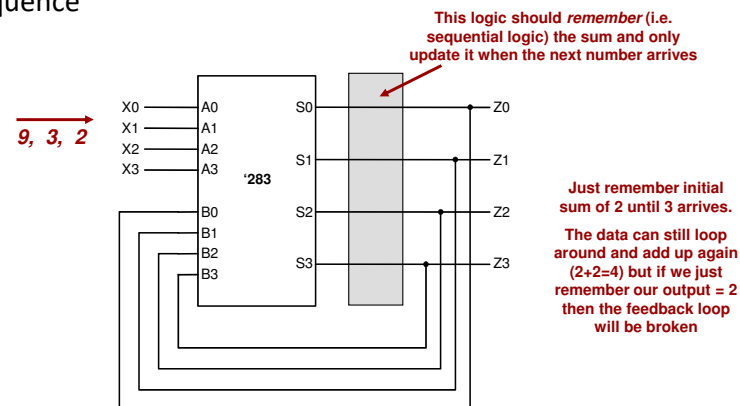
Adding a Sequence of Numbers

- Suppose we have a sequence of numbers that comes in over time that we want to sum up
- Possible solution: Route the outputs back to the inputs so we can add the current sum to the input X
- Problem 1: No way to initialize sum
- Problem 2: Outputs can race around to inputs and be added more than once per input number



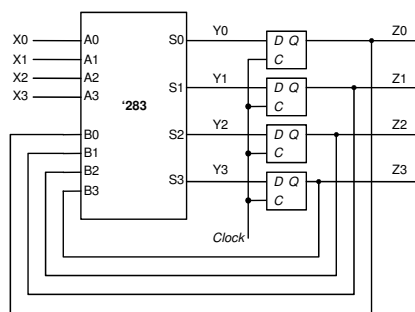
Adding a Sequence of Numbers

- Add logic at outputs to just capture and remember the new sum until we're ready to input the next number in the sequence



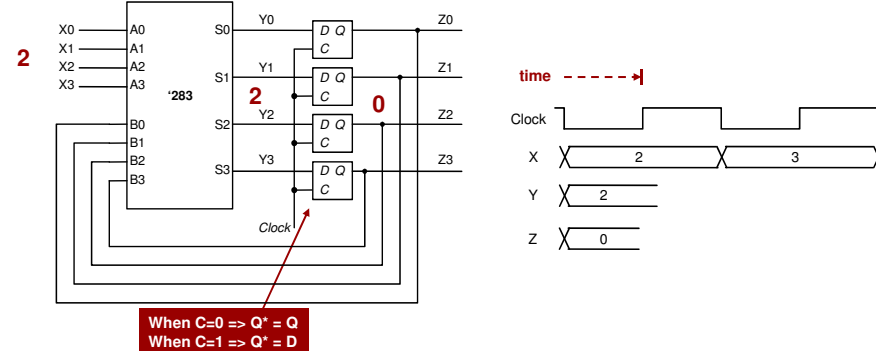
Adding a Sequence of Numbers

- What if we put D-Latches at the outputs



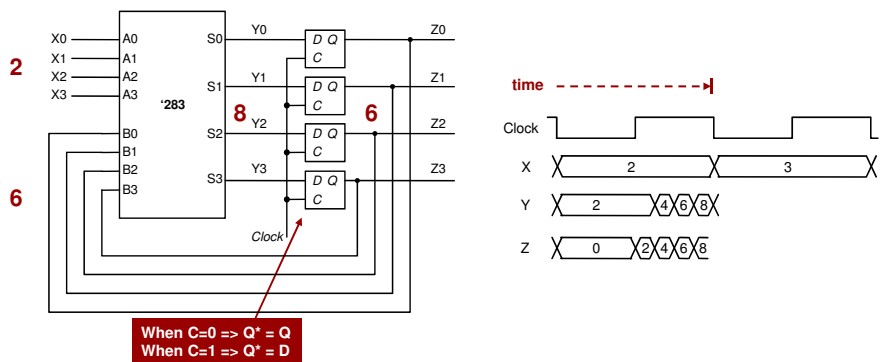
Adding a Sequence of Numbers

- Since the clock starts off low, the outputs of the latches can't change and just hold at 0



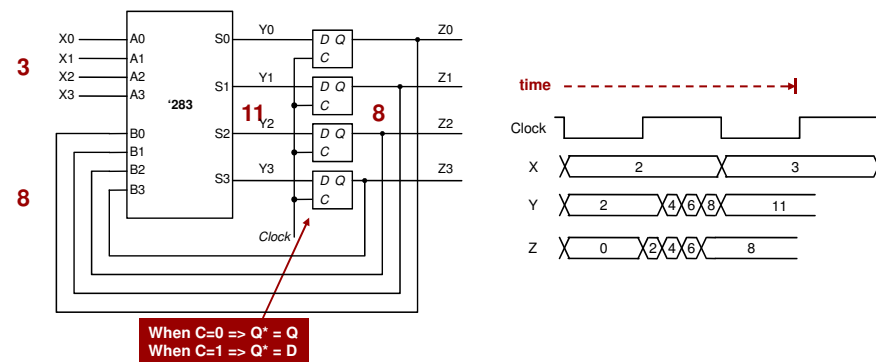
Adding a Sequence of Numbers

- This feedback loop continues until the clock goes low again



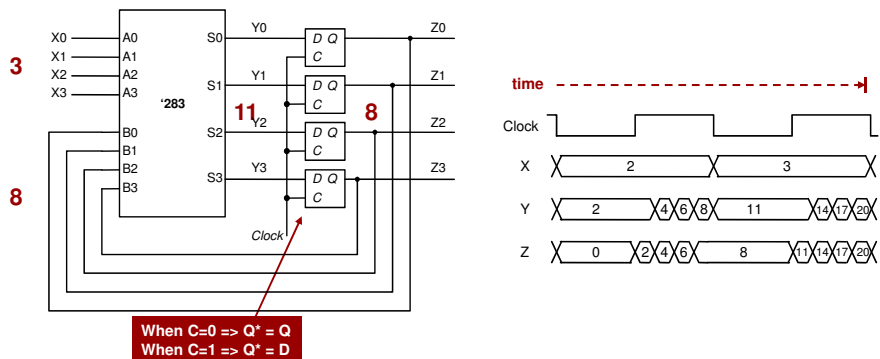
Adding a Sequence of Numbers

- When the clock goes low again, the outputs will hold at their current value 8 until the clock goes high



Adding a Sequence of Numbers

- Latches clearly don't work
- The goal should be to get **one change of the outputs per clock period**

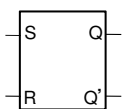


FLIP-FLOPS

Flip-Flops vs. Latches

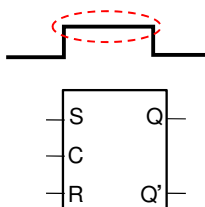
Bistables

- Asynchronous
- No clock input



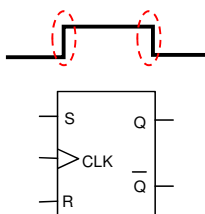
Latches

- Asynchronous
- Clock/Enable input
- Level Sensitive
 - Outputs can change anytime Clock = 1



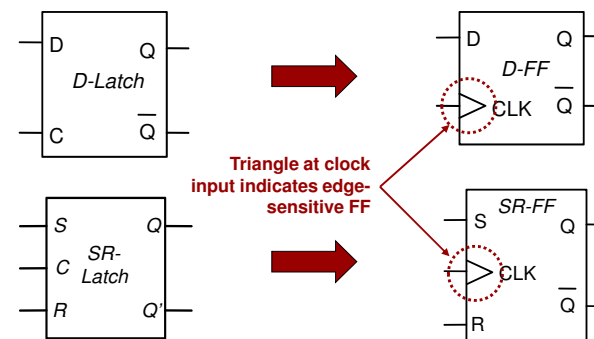
Flip-Flops

- Synchronous
- Clock Input
- Edge-Sensitive
 - Outputs change only on the positive (negative) edges



Flip-Flops

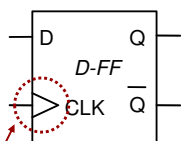
- Change D Latches to D Flip-Flops
- Change SR Latches to SR Flip-Flops



Flip-Flops

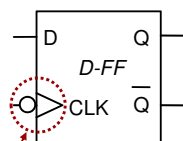
- To indicate negative-edge triggered use a bubble in front of the clock input

Positive-Edge Triggered D-FF



No bubble indicates positive-edge triggered

Negative-Edge Triggered D-FF

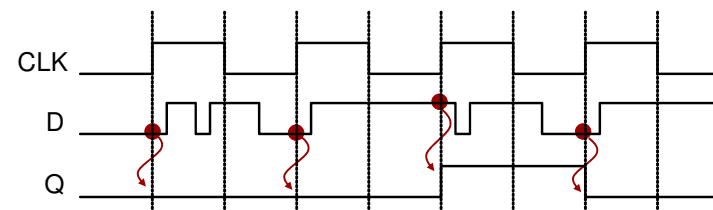


Bubble indicates negative-edge triggered

Positive-Edge Triggered D-FF

- Q looks at D only at the positive-edge

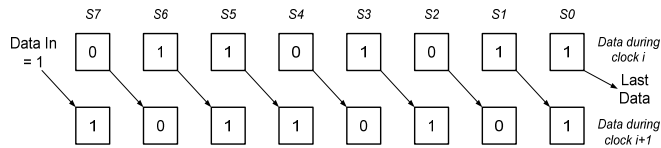
| CLK | D | Q* | Q** |
|-----|---|----|-----|
| 0 | x | Q | Q' |
| 1 | x | Q | Q' |
| ↑ | 0 | 0 | 1 |
| ↑ | 1 | 1 | 0 |



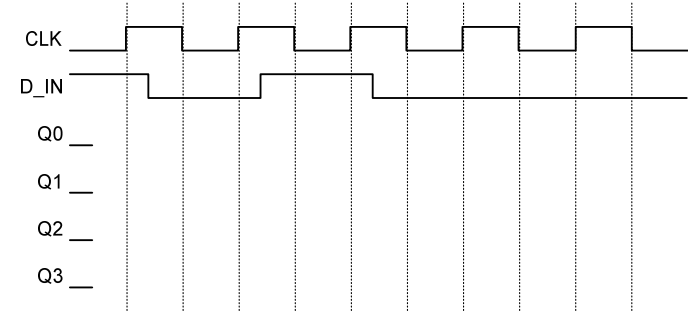
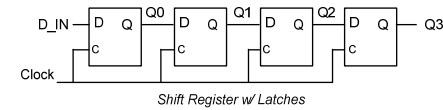
Q only samples D at the positive edges and then holds that value until the next edge

Shift Register

- A shift register is a device that acts as a 'queue' or 'FIFO' (First-in, First-Out).
- It can store n bits and each bit moves one step forward each clock cycle
 - One bit comes in the overall input per clock
 - One bit 'falls out' the output per clock



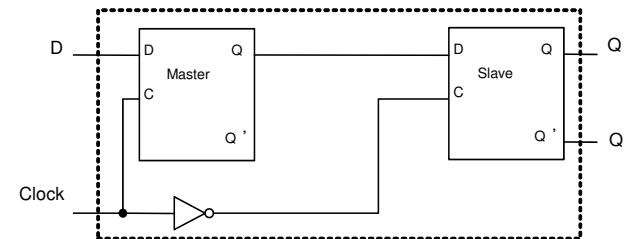
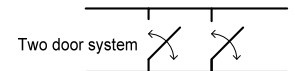
Shift Register



BUILDING A FLIP FLOP

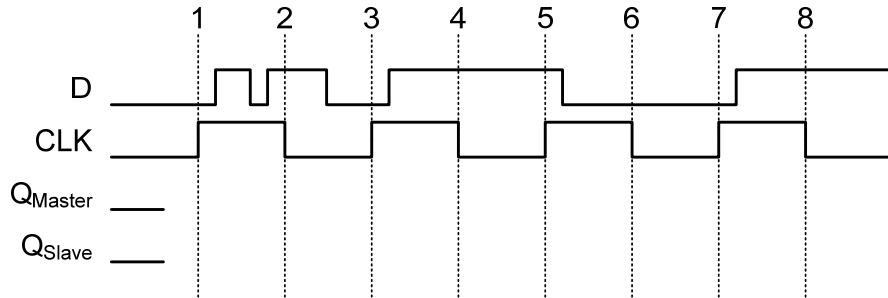
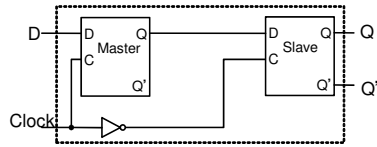
Master-Slave D-FF

- To build an **edge-triggered** D-FF we can use two D-Latches



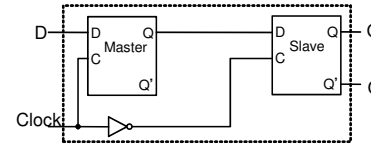
These 2 latches form a flip-flop

Complete the Waveform

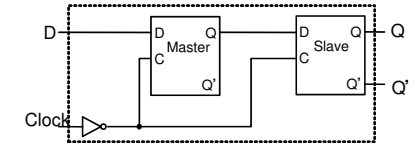


Master-Slave D-FF

- To implement a positive edge-triggered D-FF change the _____



-Edge Triggered



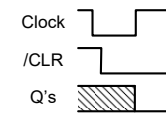
-Edge Triggered

ASYNCHRONOUS VS. SYNCHRONOUS PRESET & CLEAR

Synchronous vs. Asynchronous

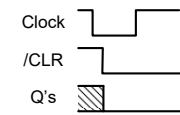
- The preset and clear inputs can be built to be **synchronous** or **asynchronous**
- These terms refer to when the initialization takes place
 - Asynchronous...initialize when _____
 - Synchronous...initialize at _____

Synchronous



Synchronous /PRE or /CLR means the signal must be active at a clock edge before Q will initialize

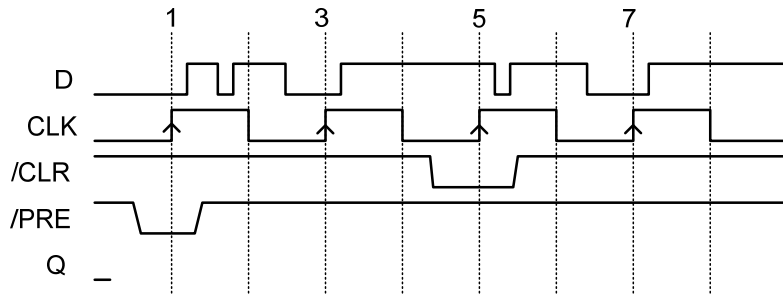
Asynchronous



Asynchronous /PRE or /CLR means Q will initialize as soon as the /PRE or /CLR signal is activated

Preset / Clear Example

- Assume an **asynchronous** Preset and Clear



Preset / Clear Example

- Assume an **synchronous** Preset and Clear

