
EECE 481

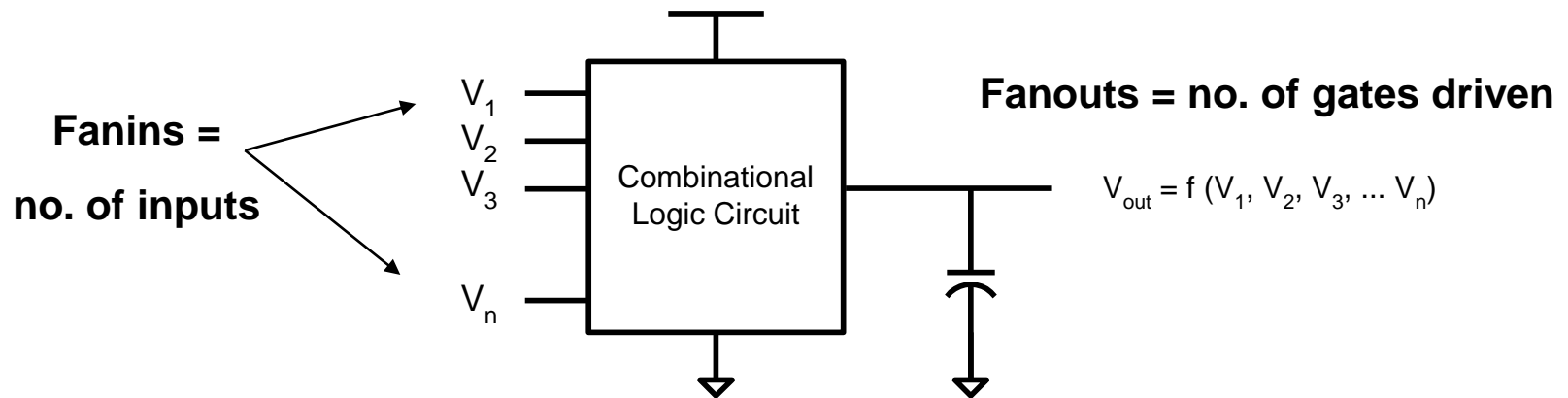
Static MOS Gate and Flip-Flop Circuits Lecture 8

Reza Molavi
Dept. of ECE
University of British Columbia
reza@ece.ubc.ca

Slides Courtesy : Dr. Res Saleh (UBC), Dr. D. Sengupta (AMD), Dr. B. Razavi (UCLA)

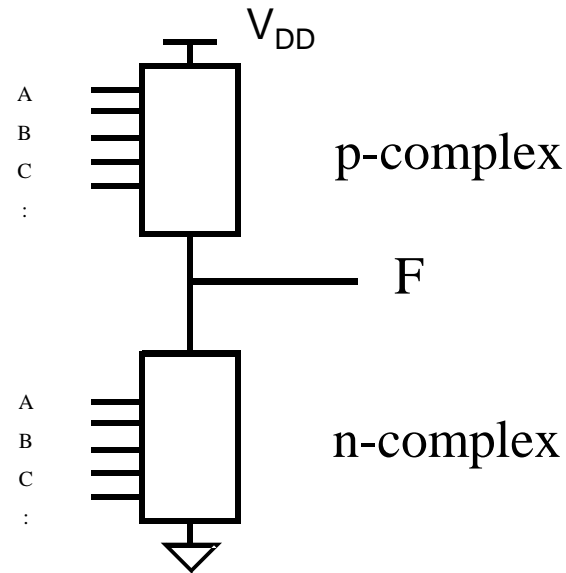
Combinational MOS Logic

- Now that we understand the logic abstraction and the properties of valid logic gates, we can consider the issues of design basic building blocks of digital systems
- Typical combinational gate is a multiple input – single output system
- Performs Boolean operations on multiple input variables, drives one or more gates
- Design parameters and considerations:
 - Propagation delay
 - Static and dynamic power
 - Area
 - Noise margins (VTC)



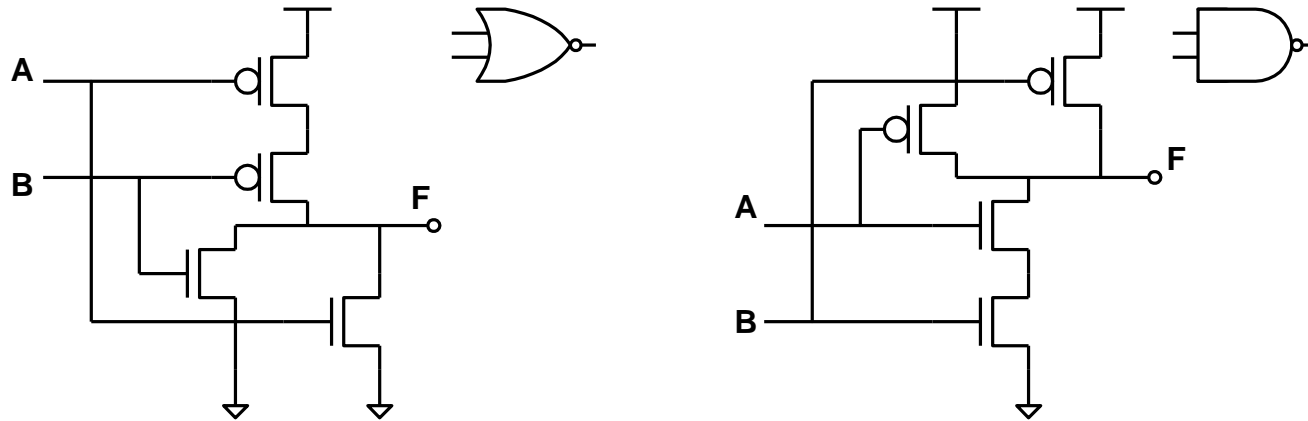
Pull-up and Pull-down Networks

Basic Structure
Of all CMOS logic gates



- PMOS pull-up and NMOS pull-down networks are duals of each other
- Configuration of pull-up and pull-down networks create a current connection from the output to *either* V_{DD} or GND , based on the inputs
- PMOS devices have lower drive capability and thus require wider devices to achieve the same on-resistance as its pull-down counterpart

Static CMOS Logic Gates



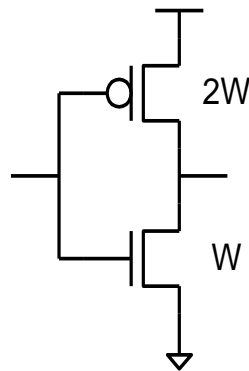
- These are the most common type of static gates
- Can implement any Boolean expression with these two gates
- Why is static CMOS so popular?
 - It's very robust!
 - it will eventually produce the right answer
 - Power, shrinking V_{DD} , more circuit noise, process variations, etc. limit use of other design styles

More Properties of Static CMOS Logic

- Fully complementary
- Low static power dissipation!
- Outputs swing full rail – Vdd (V_{OH}) to Gnd (V_{OL})
- Works fine at low Vdd voltages
 - But lower Vdd = less current = slower speed
- Combinational operation
 - Feed it some inputs, wait some delay, result comes out
 - No clocks required for normal operation
- Moderately good performance
 - Drive strength is proportional to transistor size
 - Large loads require large W
- Dual logic networks for N- and P-Channel devices

“Beta Ratio” for Static Gates

- The ratio between the NMOS and PMOS device is called the beta ratio
- We need to size CMOS static gates to deliver a target speed. But how?
- Start by sizing the inverter to deliver the target speed, then map size to gate
- Suppose:
 $R_{eqp} \text{ (PMOS)} \approx 2.5 R_{eqn} \text{ (NMOS)}$ under identical conditions
 (actualy $R_{eqp}=30k\Omega$ and $R_{eqn}=12.5k\Omega$)
- Then ratio between PMOS:NMOS should be 2.5:1
- Beta ratio sets:
 - Switching point of the gate output drive
 - Input capacitance
 - L-to-H vs. H-to-L transition times
 - Usually find a 2:1 ratio in CMOS inverter



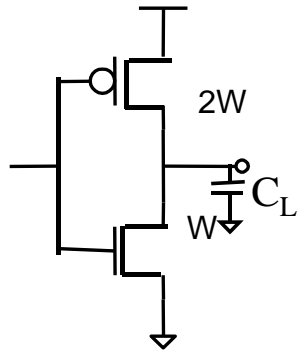
Is 2:1 the right ratio?

For equal rise and fall times. 2.5:1

How about for minimum delay? 1.7:1

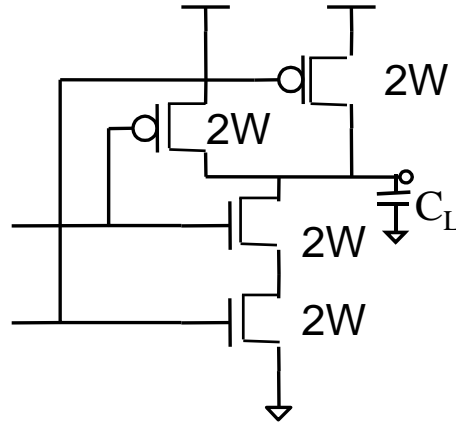
NAND and NOR Sizing

First Design Inverter

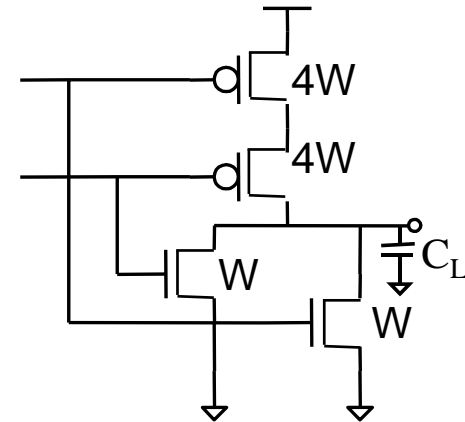


(a) Inverter

Then Map Results to Gates



(b) NAND

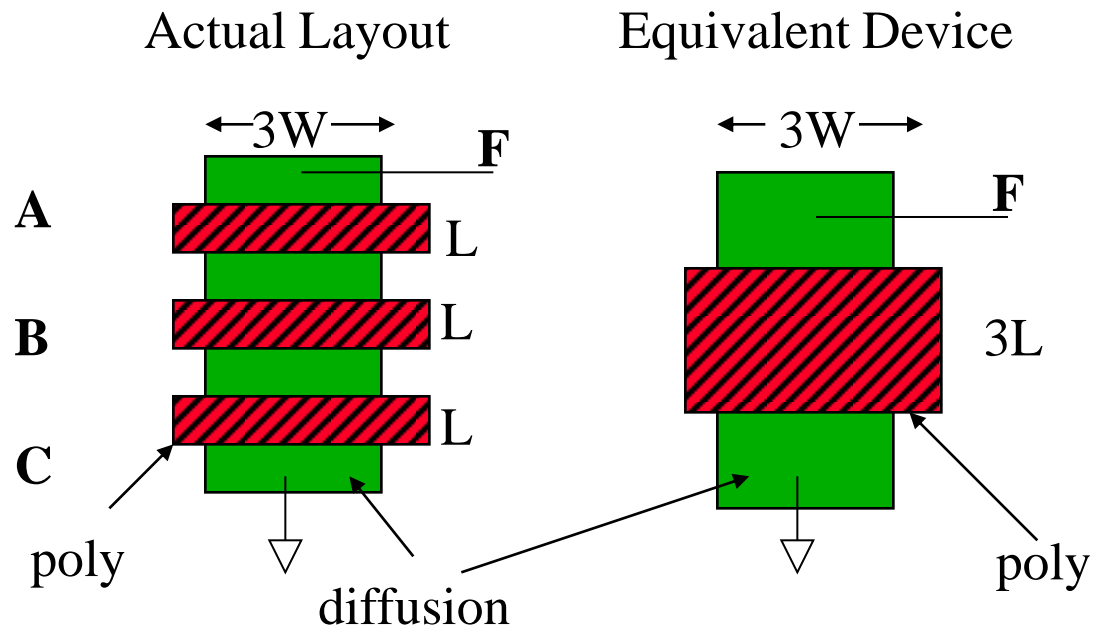


(c) NOR

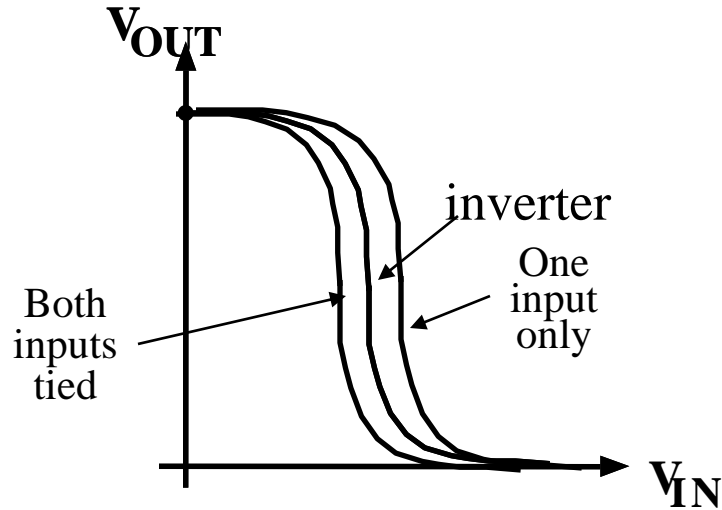
- Drive strength determined by device widths - W (assume L is minimum size)
- For the moment, consider only C_L (we are ignoring the device self-capacitance)
- Pick the right sizes for the basic inverter and then assign values to gates
- What does that mean for parallel and series combinations?
 - For parallel transistors, direct mapping from inverter
 - For series transistors, need to compute equivalent sizes

Equivalent Sizes

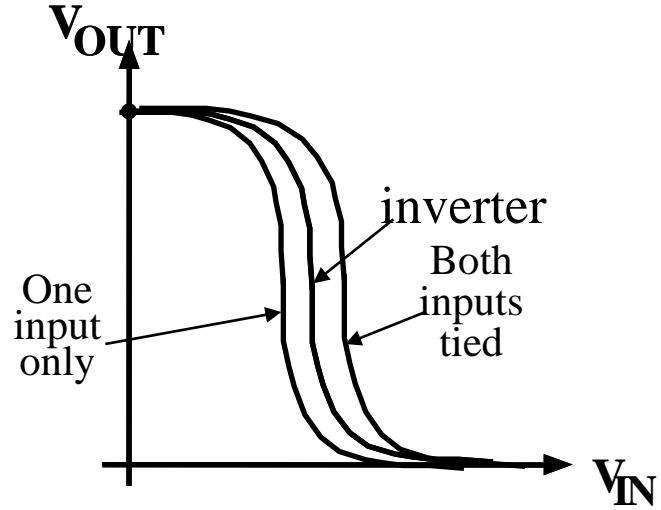
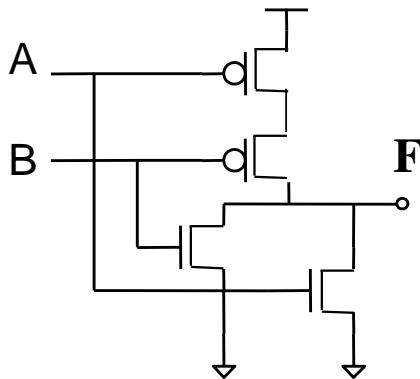
Consider a three-input NAND gate (NMOS portion only):



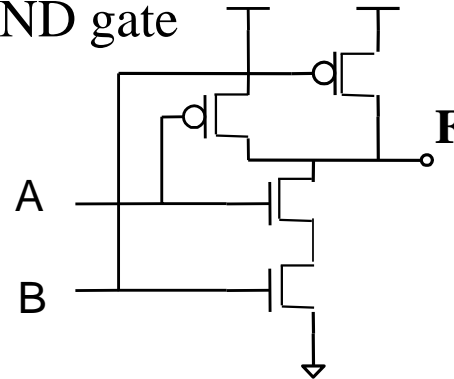
VTC and Noise Margins



NOR gate



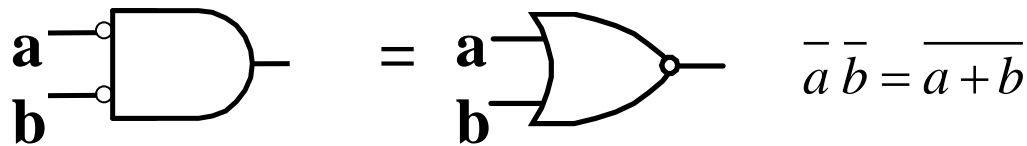
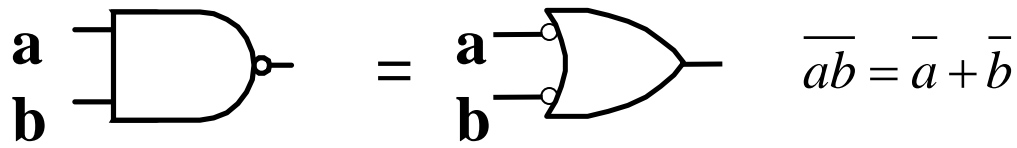
NAND gate



Complex Logic Circuits

- The ability to easily build complex logic gates is one of the most attractive features of MOS logic circuits
- Design principle of the pull-down network:
 - OR operations are performed by parallel connected drivers
 - AND operations are performed by series connected drivers
 - Inversion is provided by the nature of MOS circuit operation
- Don't get too carried away... Use this knowledge wisely
 - Remember that complex functions don't have to be implemented with a single gate
 - Can break up very complicated Boolean expressions into a cascade of gate stages
 - Limit series stacks to 3~4
- We will use De Morgan's Law to build the dual networks

Review of DeMorgan's Law



- De Morgan's theorem:

The complement of any logic function is found by complementing all input variables and replacing all AND operations with OR and all OR operations with AND

- Use De Morgan's law to find the complement of a function for the pull-down network (if needed)
- Use Duality to find the pull-up network

Complex CMOS Gate Design Example

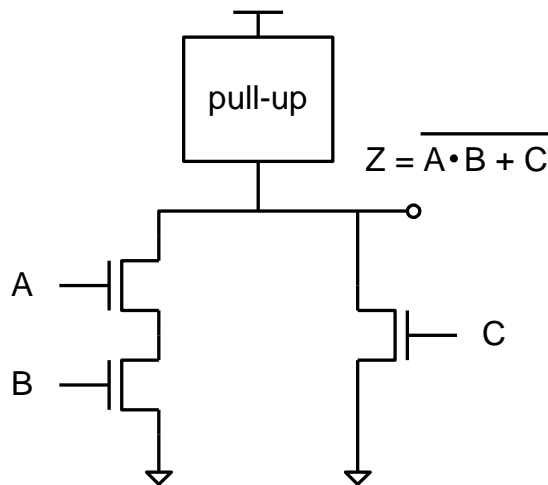
- Implement an AND-OR-INVERT (AOI) function

$$Z = \overline{A \cdot B + C}$$

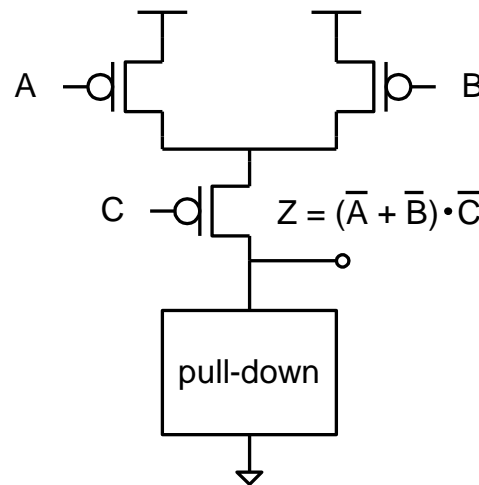
- Get the expression into forms that enable easy implementation of pull-up and pull-down networks

$$\overline{Z} = A \cdot B + C$$

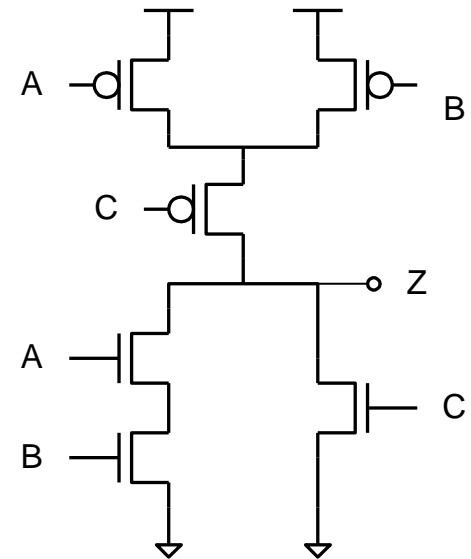
$$Z = (\overline{A} + \overline{B}) \cdot \overline{C}$$



Complement of Z

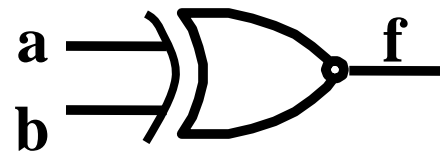
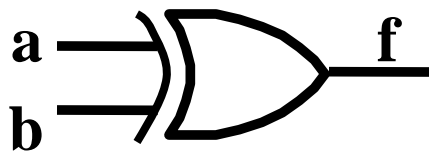
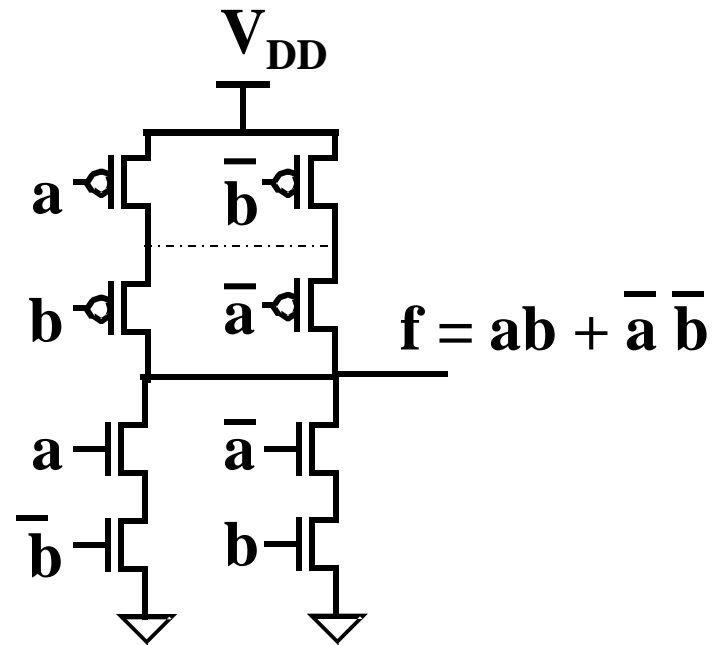
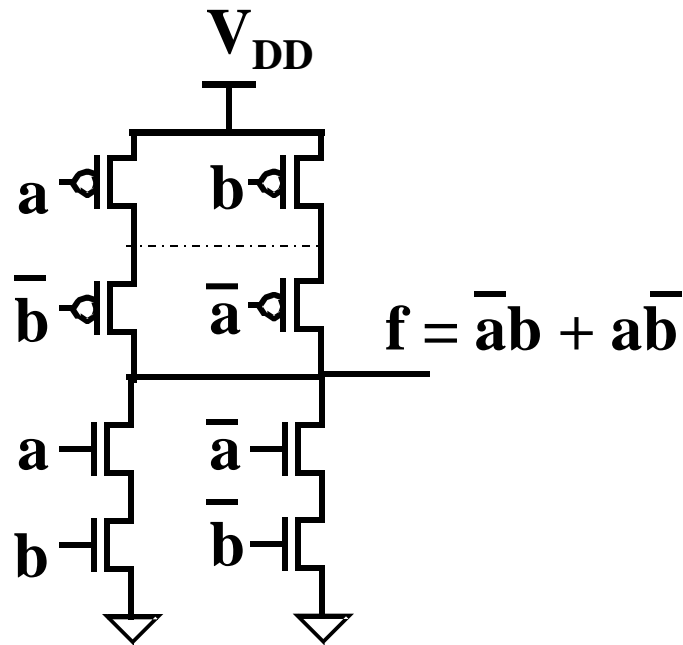


Dual of Complement of Z

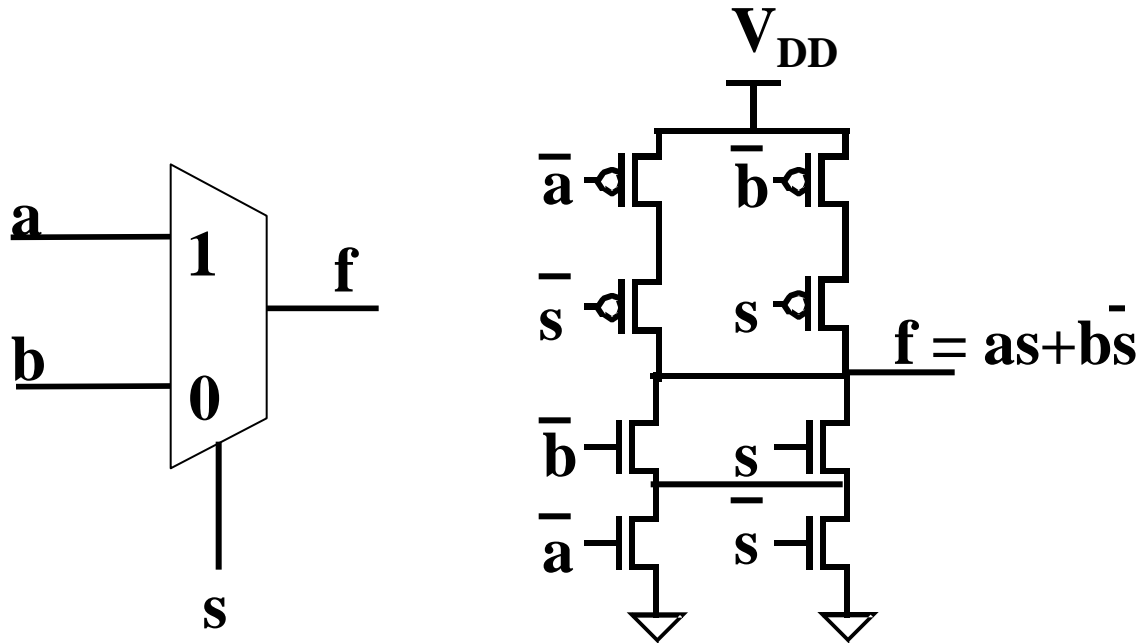


Combine Results

XOR and XNOR Gates



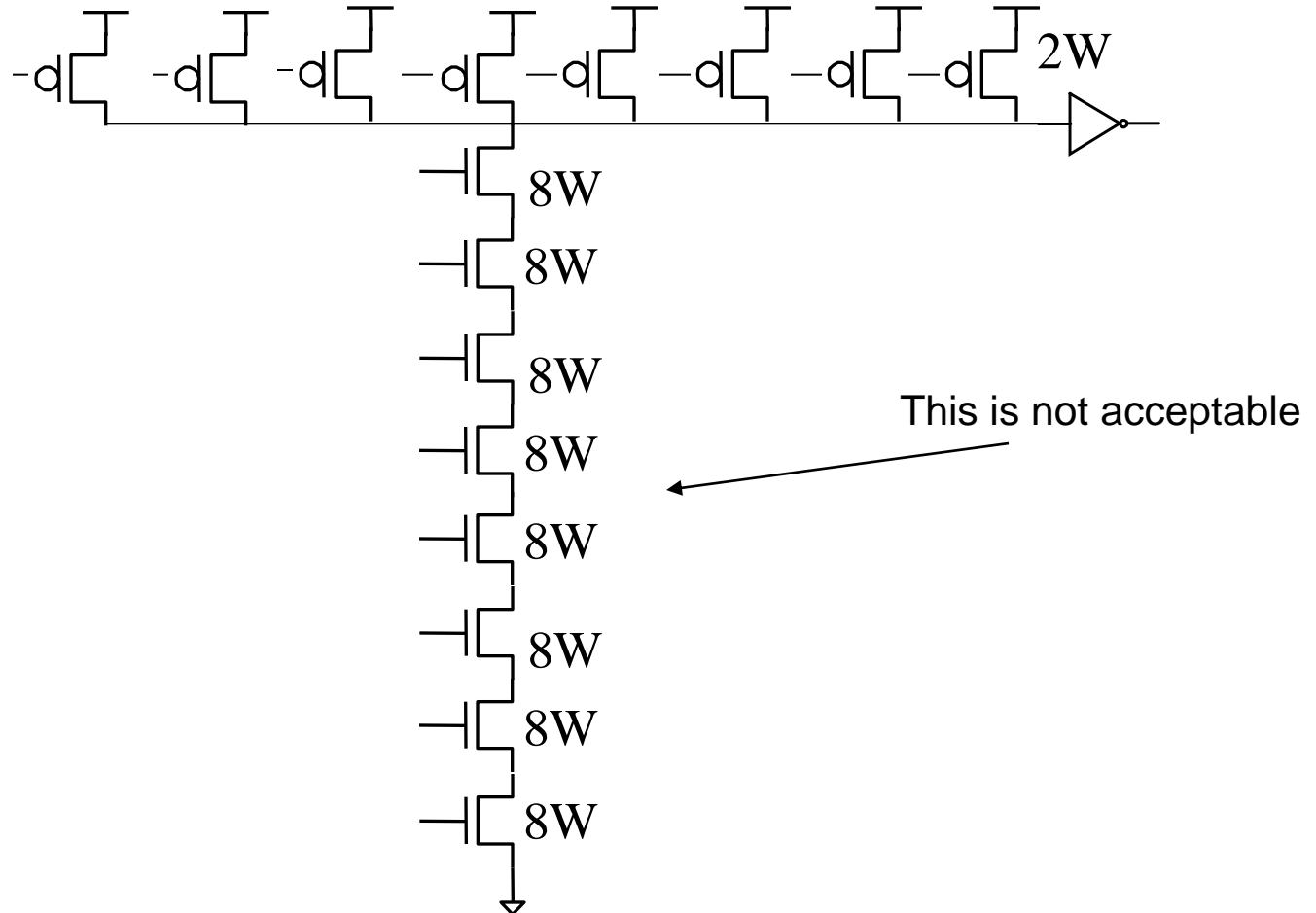
CMOS Multiplexer



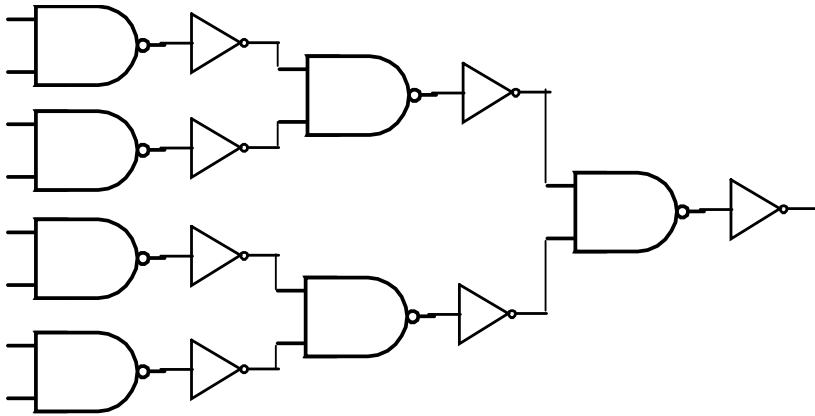
The Bad News

- Slows down dramatically for large fanins due to long series stack of transistors
 - fanin = number of inputs
 - PMOS series stacks worse than nMOS series stacks
- Large number of transistors
 - $2n$ devices for n -input NAND
 - At least 2 devices per input
- Bigger layout
 - $n+$ to $p+$ spacing rule and well spacing rule
 - Large device sizes required to counteract series stack
- Limit the fanin to 3 or 4...or delay and area will be too large

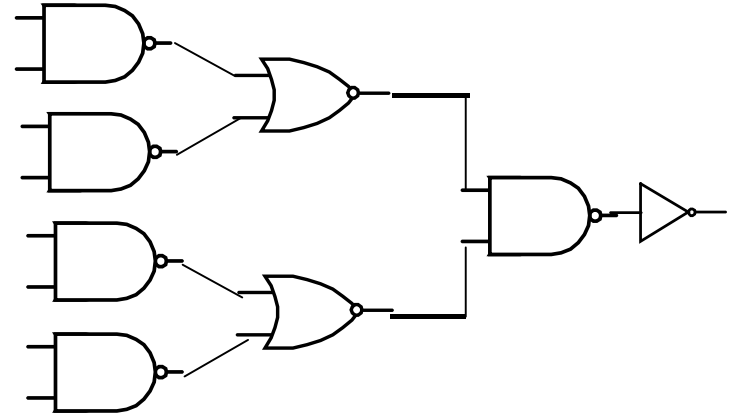
Eight-Input AND gate



Multi-level Logic Implementations



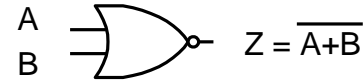
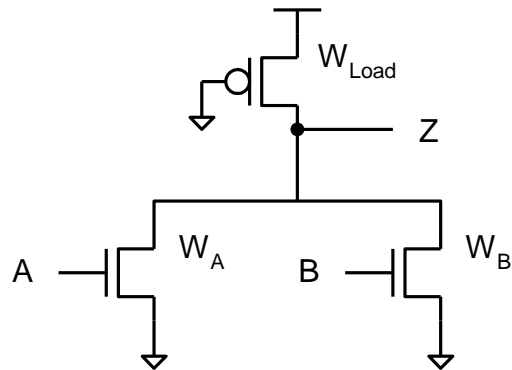
NAND2-INV-NAND2-INV



NAND2-NOR2-NAND2-INV

- There are many more options to try
- Which is the best? We need a quick way of answering this question

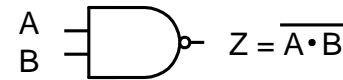
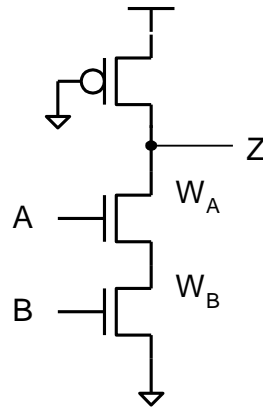
Pseudo-NMOS Logic – NOR gate



A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

- Design issues:
 - Sizing Ratio
 - Ratio pull-up to pull-down (V_{OL} & V_{OH})
 - Propagation delay
 - Subthreshold current can degrade V_{OH} slightly
 - V_{OL} decreases as more devices turning on

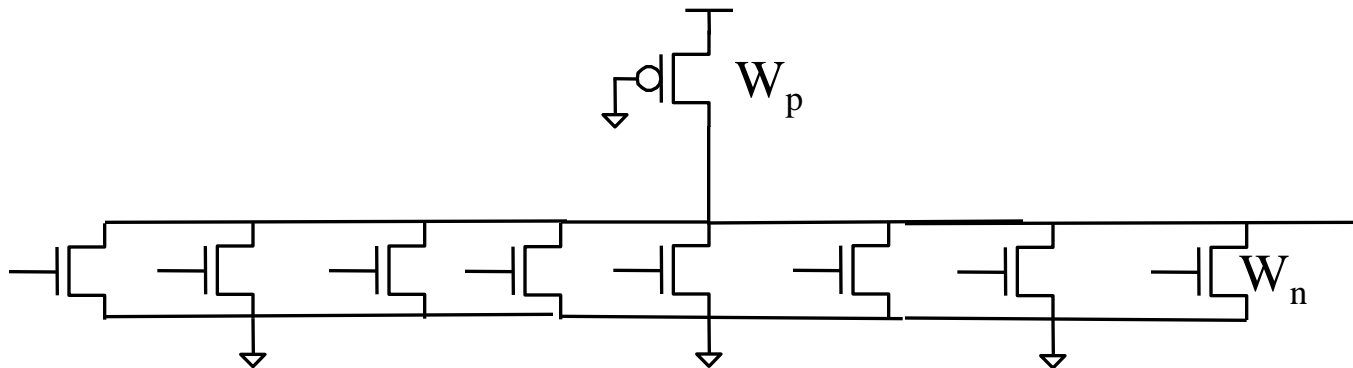
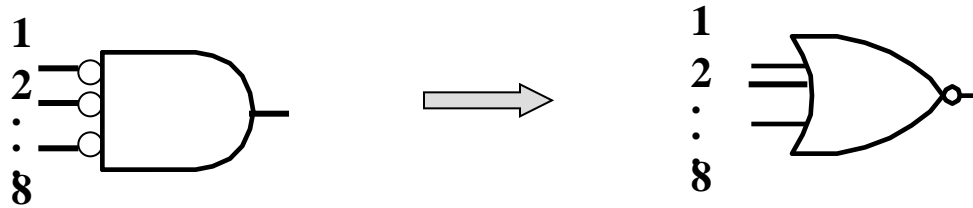
Pseudo-nMOS Logic – NAND Gate



A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

- Issues
 - Sizing Ratio
 - Need to make pull-down devices wider
 - Parasitic cap goes up with bigger devices
 - Lower devices in stack slower compared to upper ones because they see more capacitance

AND8 Option - Use Pseudo-NMOS



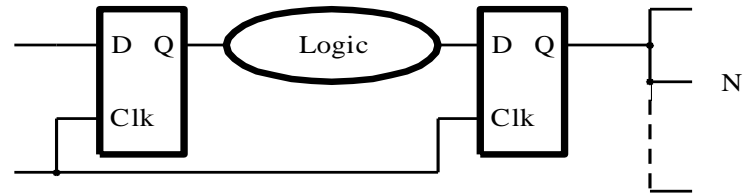
Properties of Static Pseudo-NMOS Gates

- DC power
 - always conducting current when output is low
- V_{OL} and V_{OH} depend on sizing ratio and input states
- **Poor low-to-high transition**
- Large fanin NAND gates tend to get big due to ratioing
- As transistor count increases, power consumption is too high
 - Cannot use this approach for all gates on the chip
- But what are its advantages?
 - Good for wide NOR structures
 - Memory decoder
 - Smaller number of transistors (area) / logic function

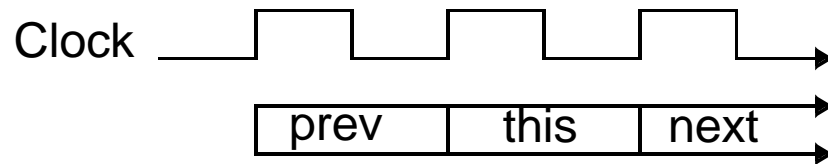
Flip-Flops and Latches

Flip-flops and latches are important logic elements used for storage

We typically build finite state machines from combinational logic (next state logic) and latches or flip-flops (storage elements) to store the state information.



We then control latches and flip-flops with a clock to create synchronous logic circuits. The clock ensures that we can tell the difference between previous, current and future states of the logic circuit



Latch vs. Flip-flop

Latch (level-sensitive, transparent)

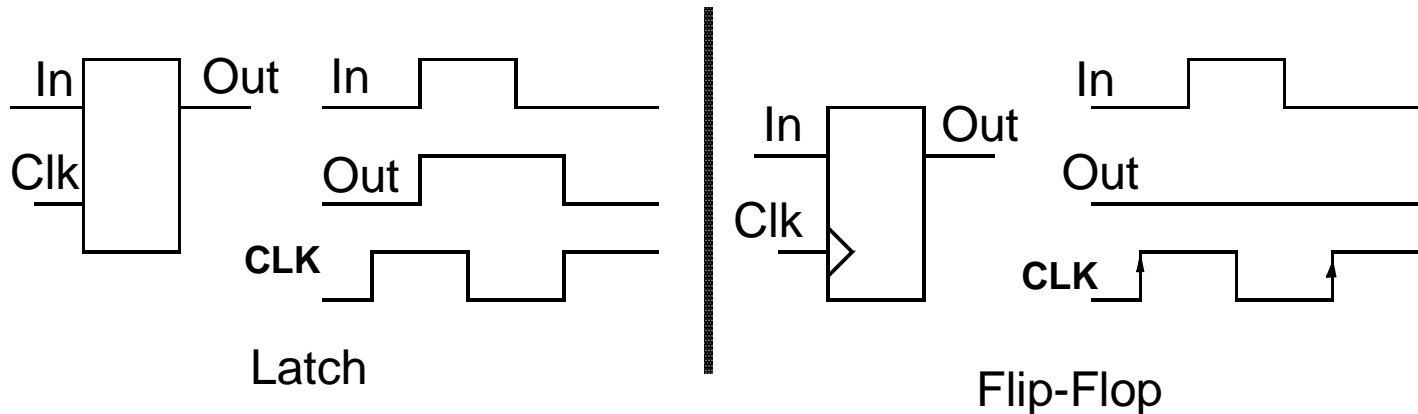
When the clock is high it passes **In** value to **Out**

When the clock is low, it holds value that **In** had when the clock fell

Flip-Flop (edge-triggered, non transparent)

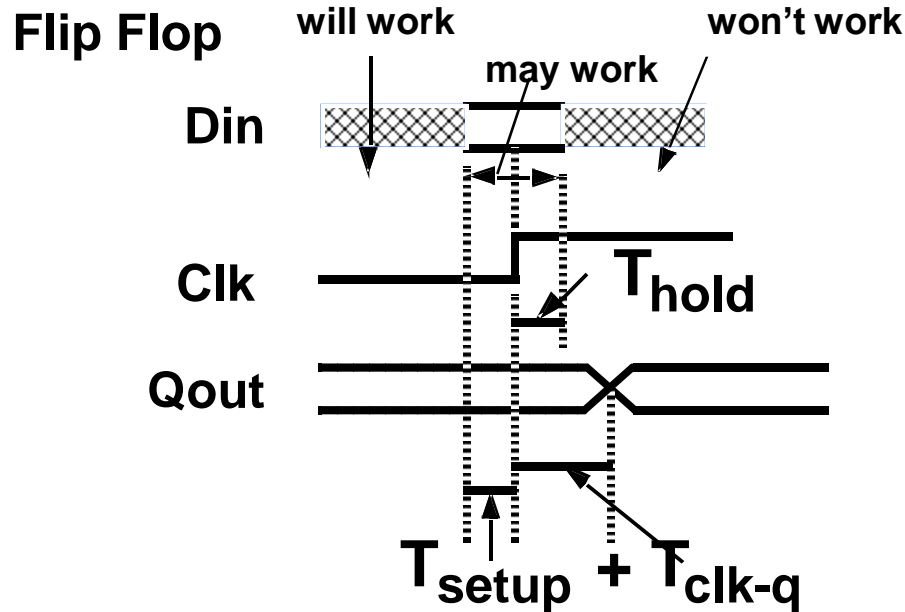
On the *rising* edge of clock (pos-edge trig), it transfers the value of **In** to **Out**

It holds the value at all other times.



FF Clocking Overhead

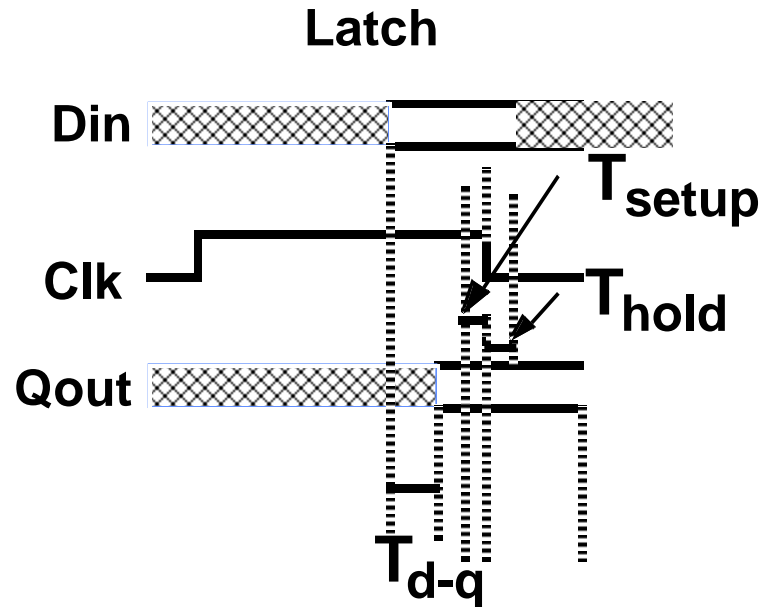
FF have setup and hold times that must be satisfied:



If Din arrives before setup time and is stable after the hold time, FF will work; if Din arrives after hold time, it will fail; in between, it may or may not work; FF delays the slowest signal by the setup + clk-q delay in the worst case

Latch Clocking Overhead

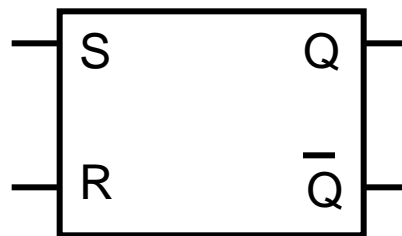
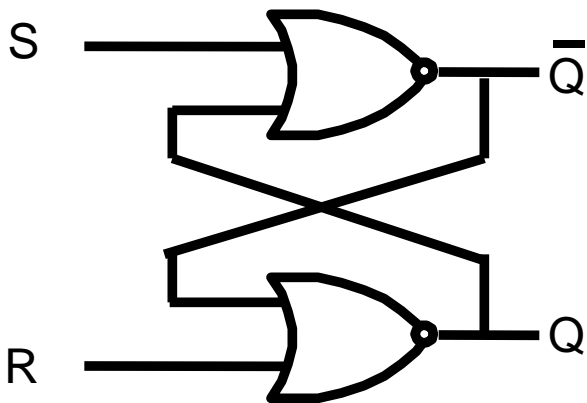
Latches also have setup and hold times that must be satisfied:



But latch has small setup and hold times; however, it delays the late arriving signals by T_{d-q} and this is more important than the setup and hold times.

SR Latch with NOR Gates

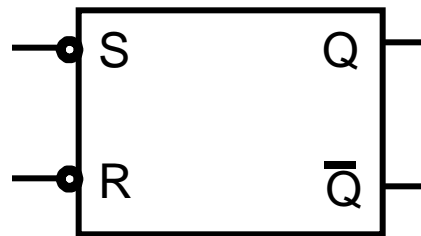
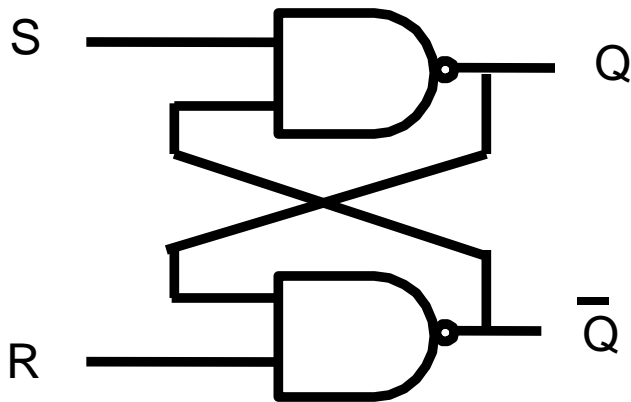
- Simplest FF is a cross-coupled pair of NOR gates
- When $S=1$, $Q=1$
- When $R=1$, $Q=0$
- By setting both $S=0$ and $R=0$, the previous state is held
- Illegal state occurs when $R=1$ and $S=1$ (actually, the final state is determined by which signal goes low last)



S	R	Q	\bar{Q}
0	0	Q	\bar{Q}
0	1	0	1
1	0	1	0
1	1	0	0

SR Latch with NAND Gates

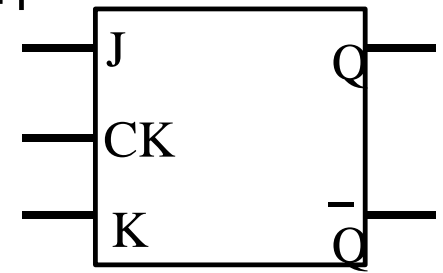
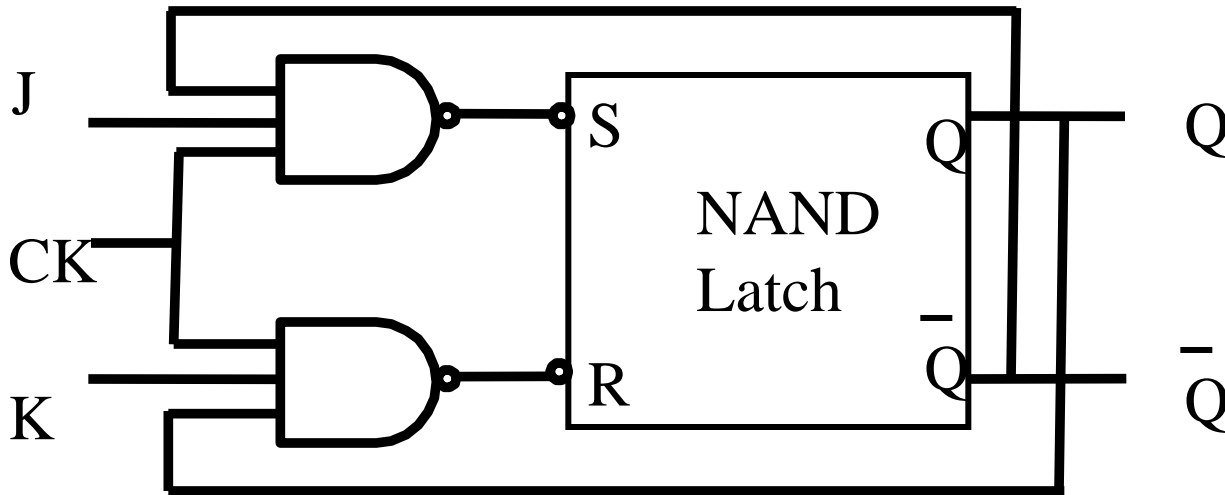
- Similar to NOR latch except that the signals are *active low*
- Illegal state is now $S=0$ and $R=0$
- Hold state is $S=1$ and $R=1$



S	R	Q	\bar{Q}
1	1	Q	\bar{Q}
0	1	1	0
1	0	0	1
0	0	1	1

JK Flip-flop

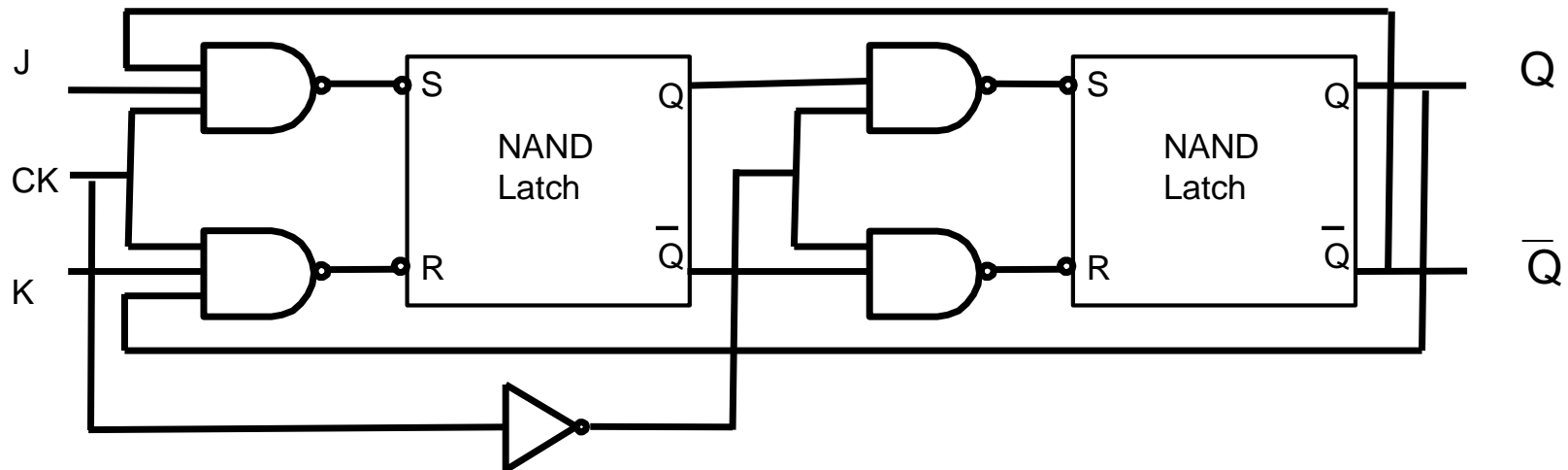
- To avoid illegal state, use JK flip-flop
- In NAND implementation, $J=K=1$ flips the state of the output
- Clock is used to enable the output
- Will oscillate if clock is high too long when $J=K=1$



J_n	K_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	\bar{Q}_n

Master-Slave JK Flip-flop

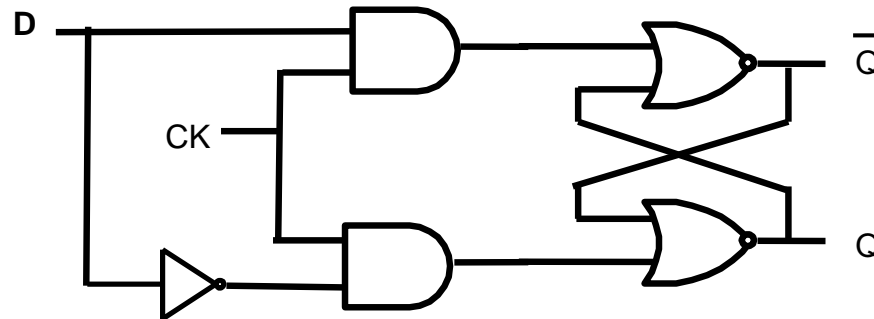
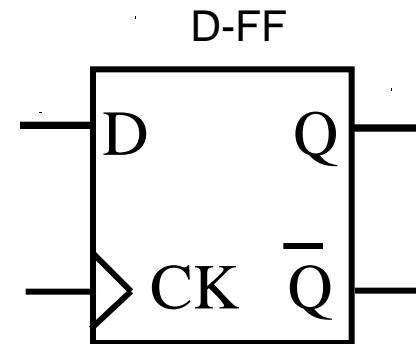
- Cascade of two JK Flip-flops
- Master activated by CK, Slave activated by $\overline{\text{CK}}$
- Master latches new data, slave launches old data



Clocked D Flip-flop

- Very useful FF
- Widely used in IC design for temporary storage of data
- May be *edge-triggered* (Flip-flop) or *level-sensitive* (transparent D-latch)

D	Q_{n+1}
0	0
1	1



Can implement
As AOI function