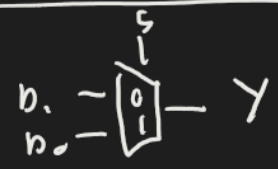


5 Combinational Building Blocks

Let's talk about multiplexers and decoders.

Suppose we want to choose between two things. We use an input to make a choice on what to let through.

Multiplexers



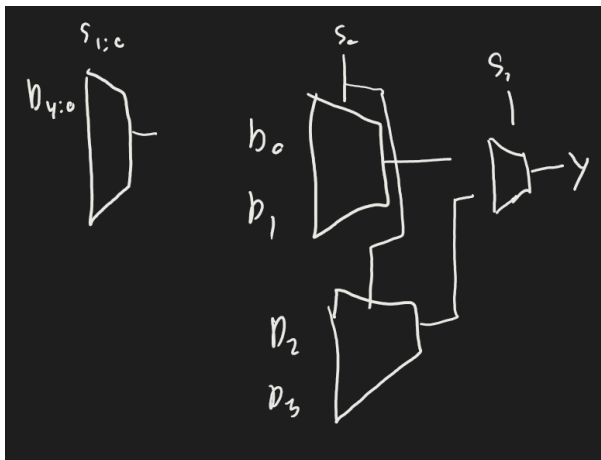
s	b ₁	b ₀	Y
0	x	0	0
0	x	1	1
1	0	x	0
1	1	x	1

	b ₁ b ₀ 00	01	11	10
s 00	0	1	1	0
01	0	0	1	1

$$Y = \bar{s} b_0 + s b_1$$

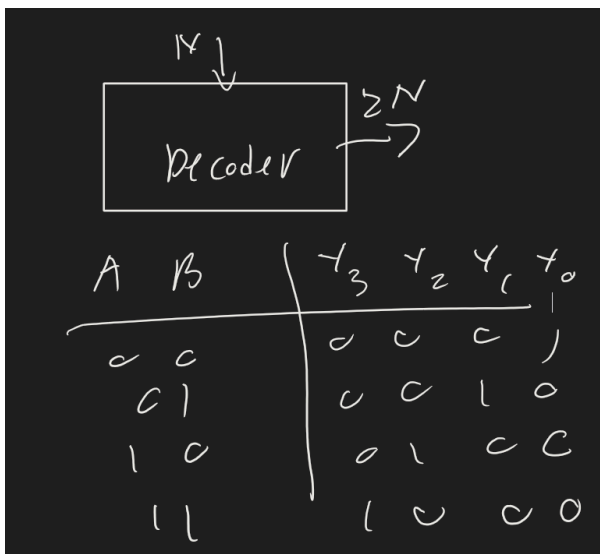
We can write out the whole circuit, but we can abstract it

We can make a 4-1 mux from 3 2-1 mux. (mux = multiplexer)

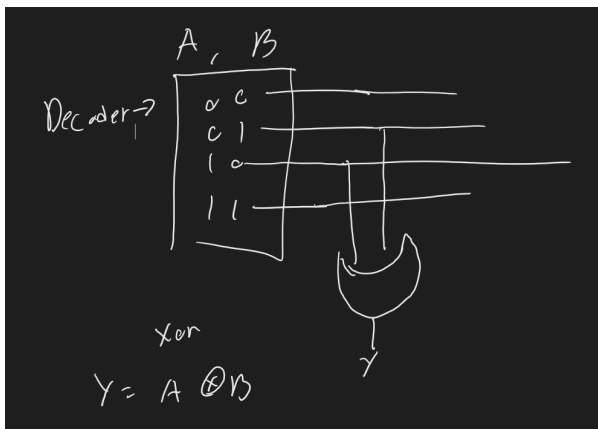


The fun thing is we can choose any working of a gate and set the d0 d1 d2 d3 to be the output we want for that combination of input s1 s2, so you can make any logic gate out of multiplexers. This is how field programmable gate arrays work.

Decoders:



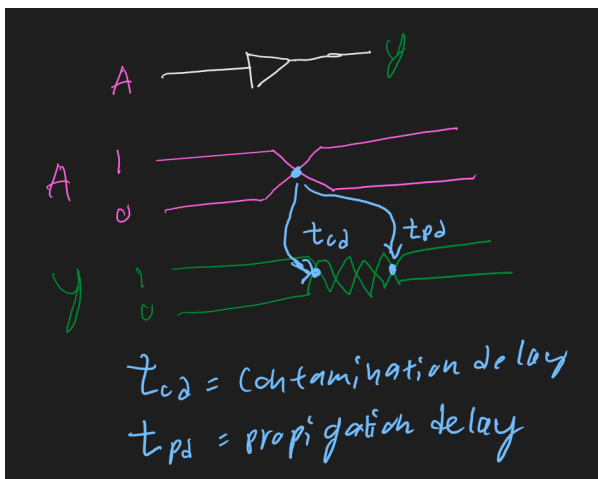
We can use decoders to make logic gates, the decoder can become a truth table:



Speed

In combinational logic, we have a delay from the change in input until the output changes.

We can consider a buffer, where $A \rightarrow Y$. It could start at 1 and go to 0 or the other way around.



If we turn off A , there will be some delay until that change happens in Y . For a brief moment there will be a confused signal, and then Y will settle. We define the contamination delay as time until the signal is messy, and the propagation delay as time to get to a consistent value.

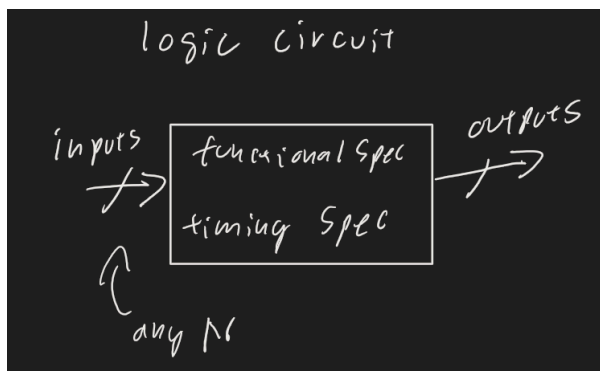
These delays are dependent on the temperature or physical characteristics of the circuit - maybe the oxide layers on the transistors are different thickness, maybe manufacturing variation etc.

These times add when we have multiple gates together, each lumped element has a minimum and maximum time to change.

Sequential elements

We've already developed combinational logic with truth tables, Boolean algebra, circuits, etc. Sequential elements depend on current and previous inputs.

We can record the previous state of a logic circuit and the current inputs to determine what the output is. Memory is called "state".

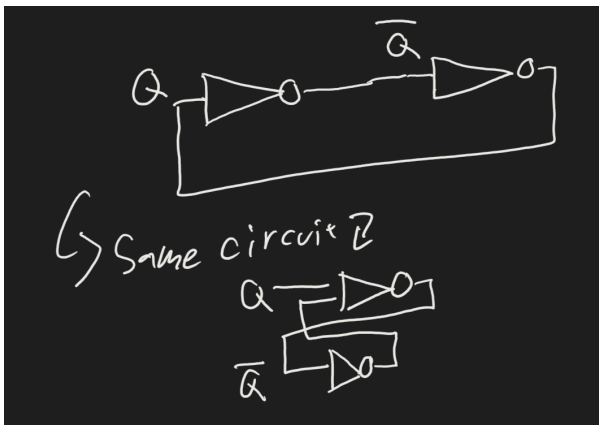


A circuit is made of nodes and elements.

Every element is combinational if every node is either an input or connects to exactly one output with no cyclic paths.

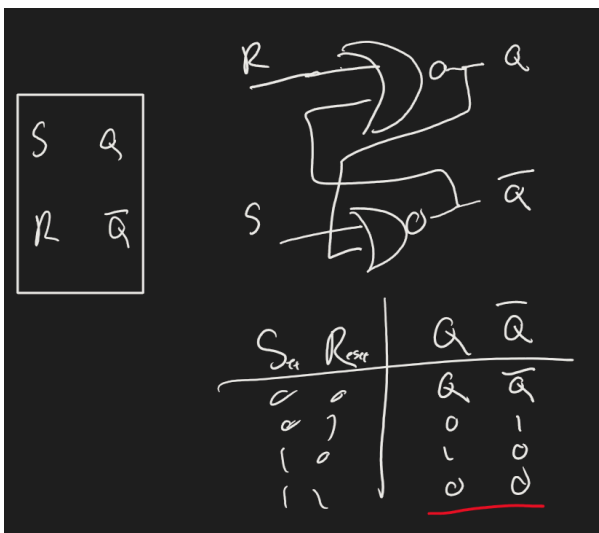
Sequential composition

when we have cyclic paths, then we are able to form memory.



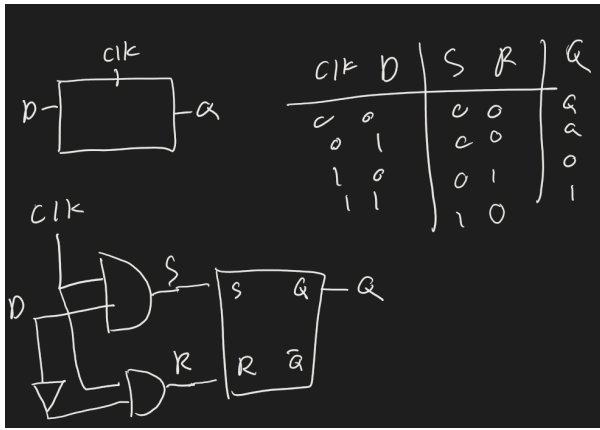
Here, if Q is high it will stay high, and low it will stay low. We can rewrite it to make symmetry.

We can make this a little more complicated:



This allows you to write Q as either 1 or 0 and then hold on to that memory. However, if we put in 1 1 ... bad stuff. Let's make something slightly more complicated to help us with handling this memory piece.

Drum roll..... welcome the D latch!

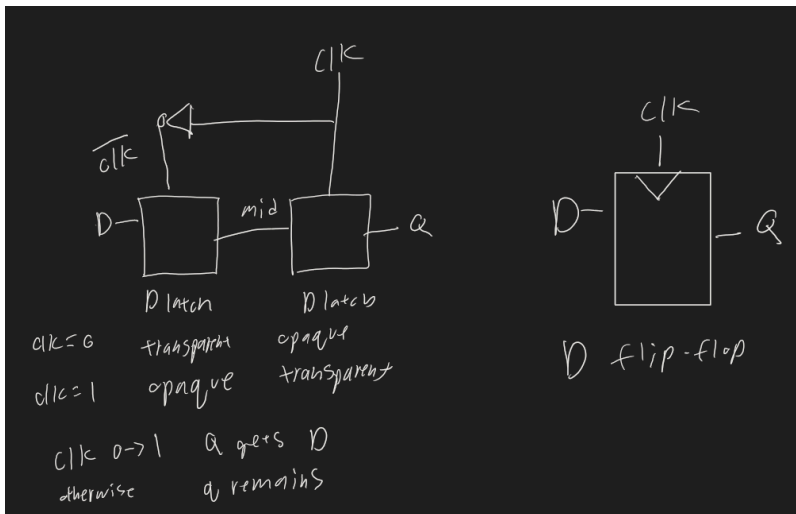


When $CLK = 1$, D flows to Q .

When $CLK = 0$, Q remembers.

We want to make this still a little bit easier to use, so we make a D Flip Flop.

We take two D latches and hook them up like so:



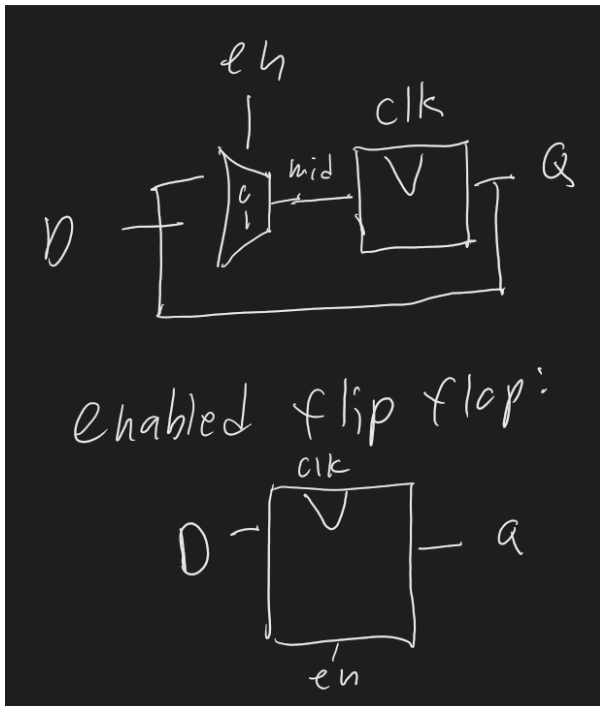
For the Q latch, whenever Q is high it will take on the value of D, but with the flip-flop we will only change values when the clock rises. We make life easy by using this D latch.

ON THE RISING EDGE OF THE CLOCK, Q GETS D

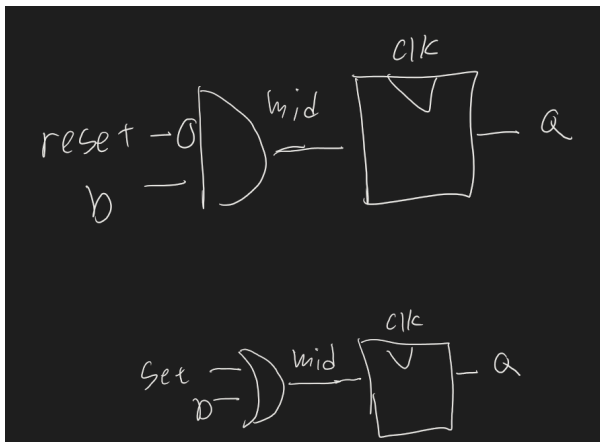
ON THE RISING EDGE OF THE CLOCK, Q GETS D

... Don't forget.

We can have an enabled flip flop so that we can choose when Q really listens to input D instead of on every clock signal.



We can now choose when we write information. We can put in some more element to make a resettable Flip flop, where we can either reset Q to 0 or pass D, or set Q to 1 or pass D.



Synchronous Sequential Discipline

Every element is either a register or combinational logic.

We have at least one register

All registers receive the same clock.

Every cyclic path has at least one register.