Physics 364, 2010-11-01 Bill Ashmanskas

Lecture 8

Why digital?

- Many quantities inherently discontinuous (discrete), e.g. your account balance at bank, the number of long-sleeved black jersey shirts you own, your birthday, ...
- Discretization permits perfect reproduction. ⇒ noise immunity
- Often can state required precision and discretize to keep that precision
- Once in digital form, data can be readily manipulated in useful ways, e.g. digital signal processing. ~~in MATLAB like~~

Why binary?

- It's a sensible alternative to decimal!
- Easy: ON/OFF
  TRUE/FALSE
  OPEN/CLOSED
  maximal/minimal

Binary numbers

0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
⋮	⋮

addition
subtraction

representing negative values:

(seldom) × sign, magnitude
(often) → two's complement

multiplication
long division

fixed point
floating point

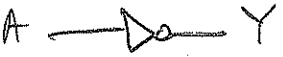
"bit" = "binary digit"

a string of N bits sometimes represents an integer $0 \le i < 2^N$, sometimes a signed, fixed-point, or floating-point number, or sometimes represents N logical truth values.

bit = 0, 1
bit = FALSE, TRUE
bit = GROUND, VCC (LOW, HIGH)

logic gates

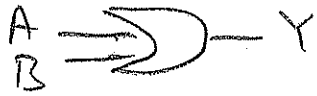
A	Y
1	0
0	1

Inverter & 

$Y = \overline{A}$
(NOT A)
!A , ¬A

OR (disjunction)

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



Y = A + B
A || B
A OR B
A ∨ B

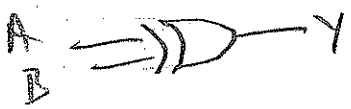
AND (conjunction)

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Y = AB , A · B
A && B
A AND B
A ∧ B

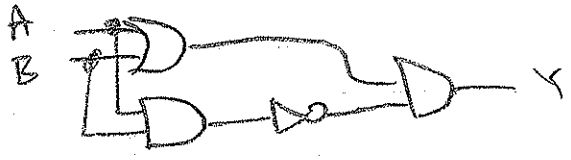
XOR (exclusive OR)

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0



Y = A ⊕ B
A ⊻ B
A XOR B
~~A ≠ B~~

(A XOR B) = (A OR B) AND NOT (A AND B)

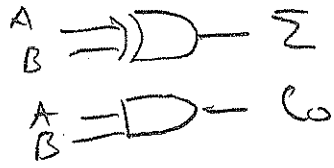


1-bit adder

A	B	Σ	carry out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$\Sigma = A \text{ XOR } B$

$C_{OUT} = A \text{ AND } B$

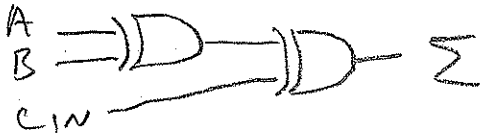


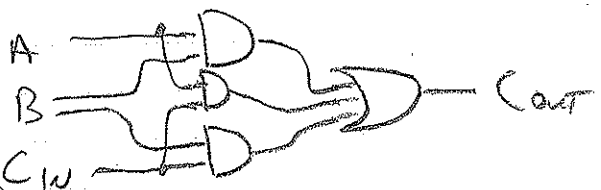
1-bit full adder

C_{IN}	A	B	C_{OUT}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

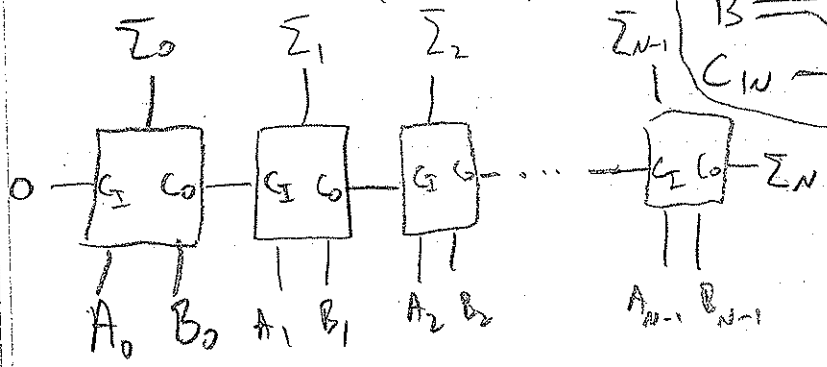
$\Sigma = (A \text{ XOR } B) \text{ XOR } C_{IN}$

$C_{OUT} = AB \text{ OR } AC_{IN} \text{ OR } BC_{IN}$





add two N-bit numbers → (N+1)-bit sum



multiplexer ("mux")

A_1	A_0	D_3	D_2	D_1	D_0	OUT
0	0					D_0
0	1					D_1
1	0					D_2
1	1					D_3

(choose one of N inputs to copy to output)

demultiplexer ("demux")

A_1	A_0	D	$OUT_{3,2,1,0}$
0	0	D	0 0 0 D
0	1	D	0 0 D 0
1	0	D	0 D 0 0
1	1	D	D 0 0 0

(choose one of N outputs to which to copy input)

(Forgot to include priority encoder.)

other sorts of logic output:

- "Open collector": logic 0 ⇒ ground; logic 1 ⇒ floating (high impedance)
  (also "open drain")

- "tri-state" (or "three-state")
  possible output states are { 0, 1, high impedance }
  like "yes," "no," and (not speaking)

  ⇒ can create a sort of "party line" telephone — called a "bus"

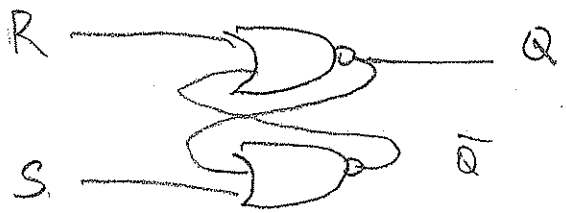
  many senders can take turns driving data onto the bus

All of the above was "combinational" (sometimes called "combinatorial") logic — the present output reflects some combination of the present inputs.

It remembers nothing. Past inputs are irrelevant.

Suppose you want to store information for later use. Need a device with internal state, i.e. present inputs alone insufficient to determine output.

Consider this weird arrangement of NOR gates:



S	R	Q
0	1	0
1	0	1
0	0	$Q_{previous}$ (!)
1	1	0 (usually considered illegal input)

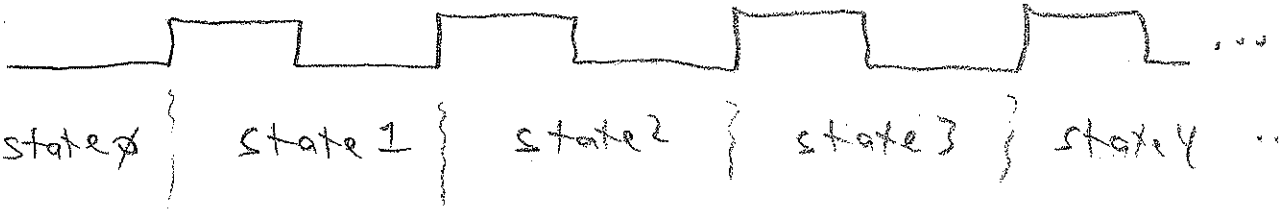
called a flip-flop (or a "bistable multivibrator" if you're feeling encyclopedic)

It can remember stored input!
A one-bit memory.

Once your components can have an internal state, it's helpful to have a mechanism to get many of them to change state in concert — like the conductor of an orchestra, or a drum beat for marching troops, or the high school bell to signal the change of classrooms.

In sequential logic, this drumbeat role is played by the clock.

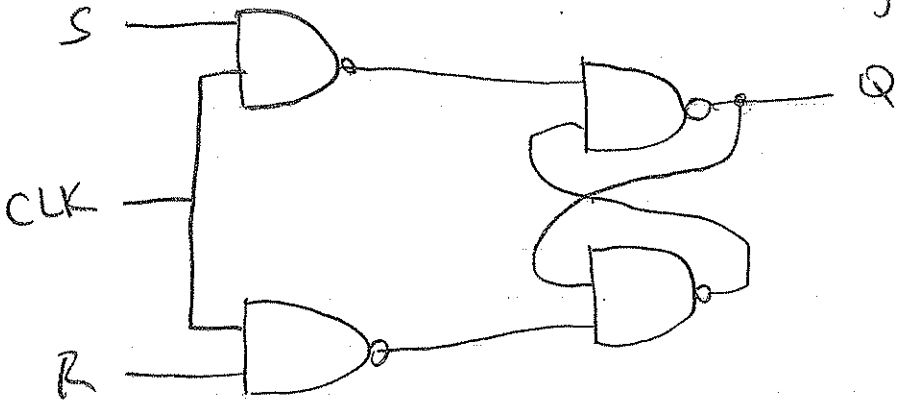
Typically, everybody agrees to change state on the rising edge of the clock.



How do you design a flip-flop to do that?

Here's a start (not quite satisfactory):

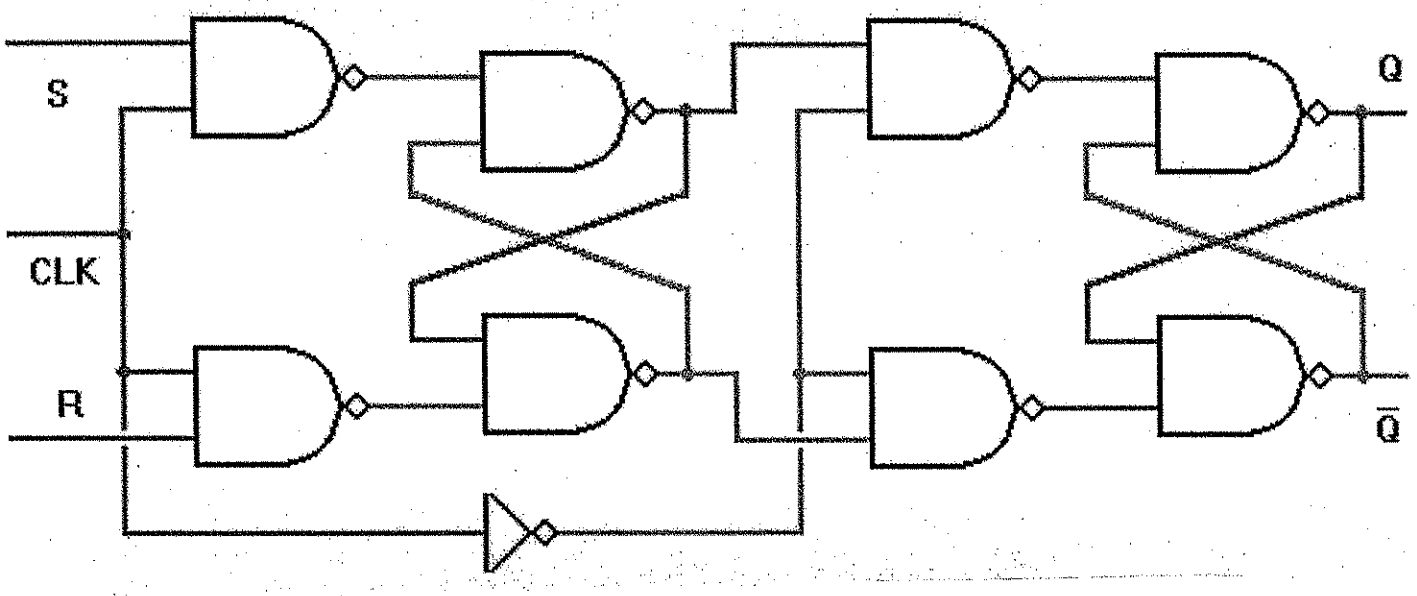
[Problem is that CLK is level-sensitive rather than edge-sensitive.]



See www.play-hookey.com/digital/clocked_rs_latch.html

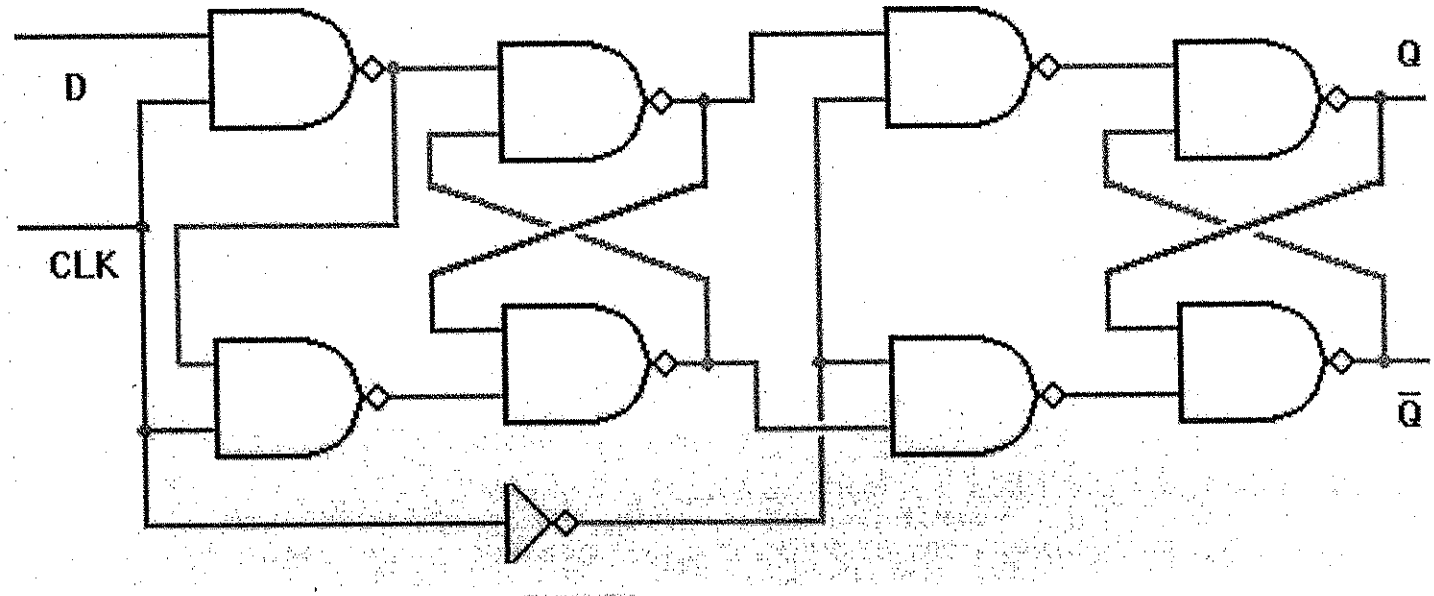
[See animations at www.play-hookey.com/digital]

Somewhat better — records SR values at clock edge.



Called "master - slave" flip flop.

The most common FF is D-flop
(vs. less common SR-flop, JK-Flop, T-flop, ...)

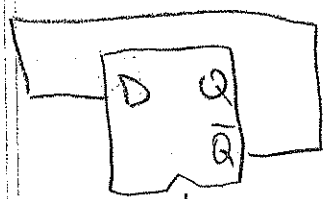


each clock cycle, Q reflects value of D on previous clock cycle.

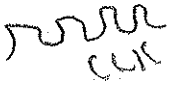
D-type

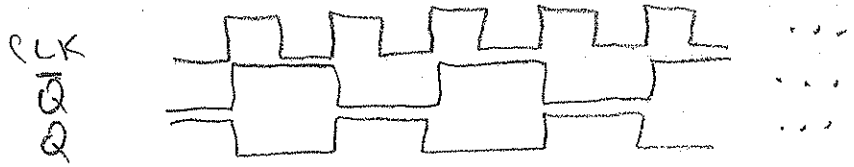
In a real flip-flop, the change of Q is synchronous to the leading edge of CLK.

Q changes immediately after ↑ and reflects state D had immediately before ↑

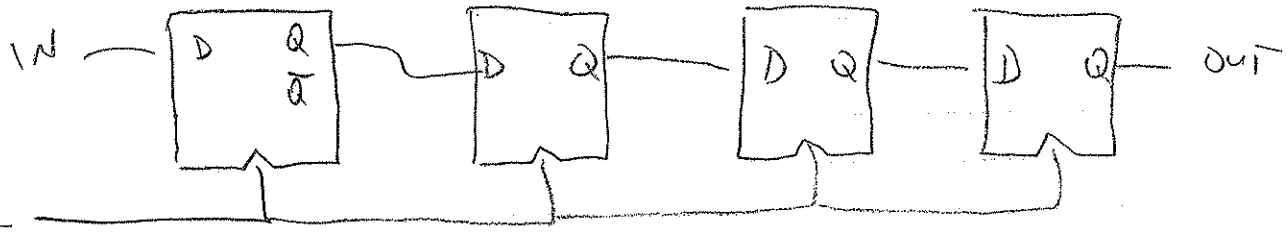


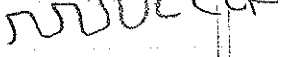
what does this do?





sometimes called "divide by 2"





This is a "pipeline"

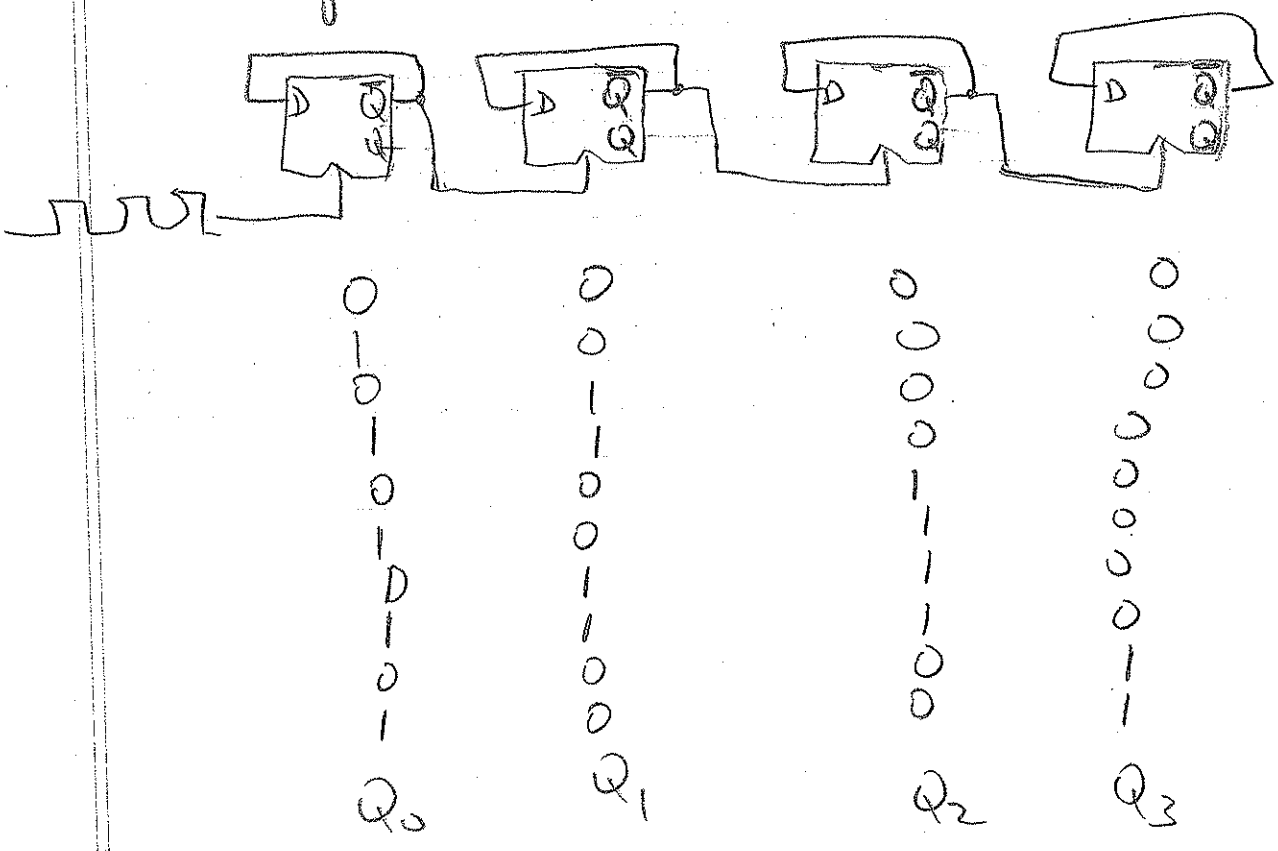
Like a firefighter's bucket hand-off

or a factory assembly line.

This particular pipeline is a "shift register"

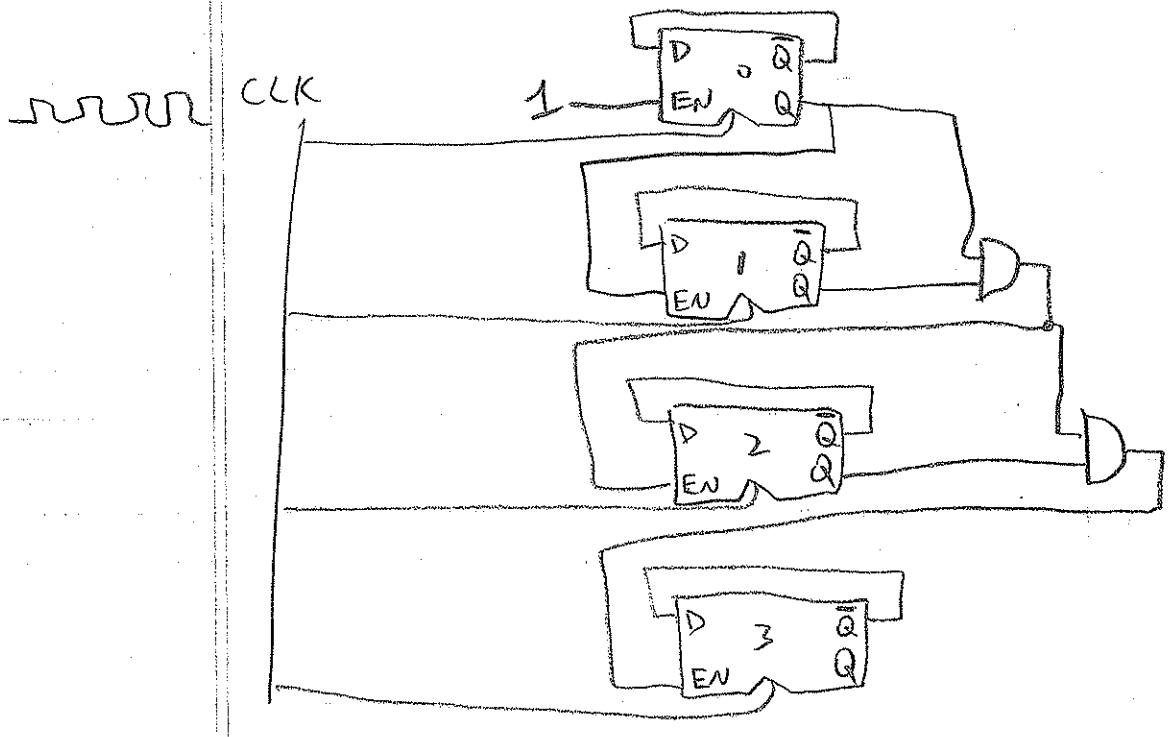
(illustrate)

A "counter" is an often-seen circuit in sequential logic:



This "ripple clock" is not a good clock as the drum beat is not heard simultaneously by everyone.

www.play-hookey.com/digital/ripple_counter.html



Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	1	1

etc.

Synchronous counter

every bit update at same clock edge.

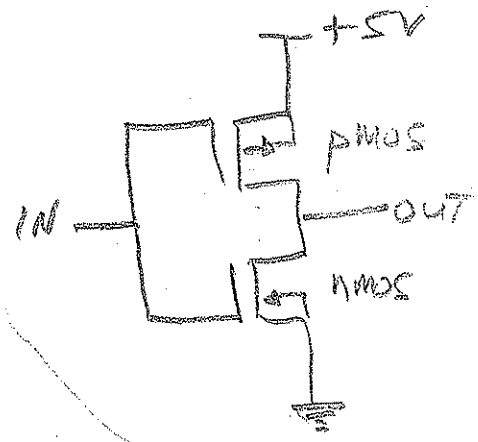
LAB 8 (early draft!)

~~CMOS~~ ~~NAND~~

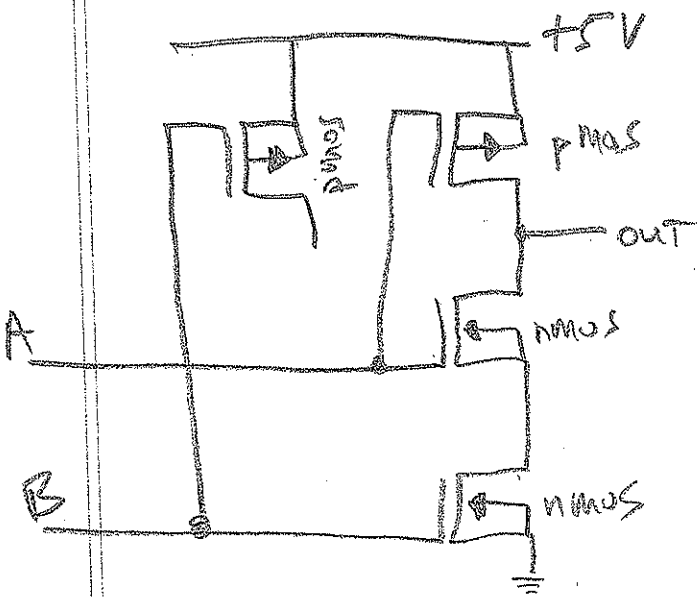
Recall the CMOS inverter and NAND gate you built or simulated last week. Nearly all real-world logic nowadays is CMOS-based. But it turns out, to my surprise, that all discrete logic components currently available in the lab are of the older TTL style (using BJTs), so we will spend this week working with TTL gates.

Partly this will give you a chance to learn a bit about the innards of logic gates; partly it will give you time to get well to the concepts; and partly it will illustrate for you what a terrific invention programmable logic is (FPGAs, etc.) (i.e. Xilinx)

Recall CMOS inverter



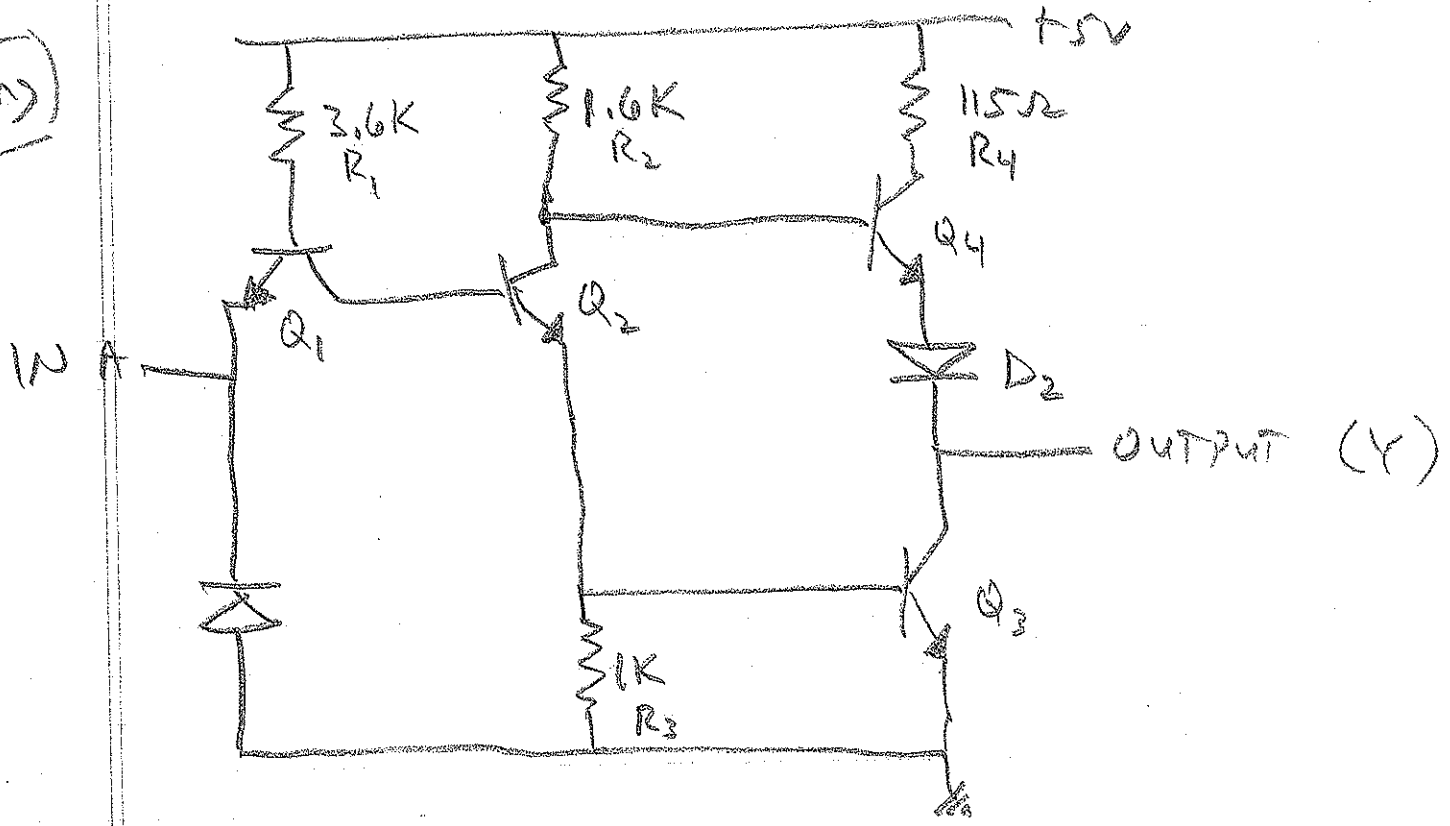
and CMOS NAND gate



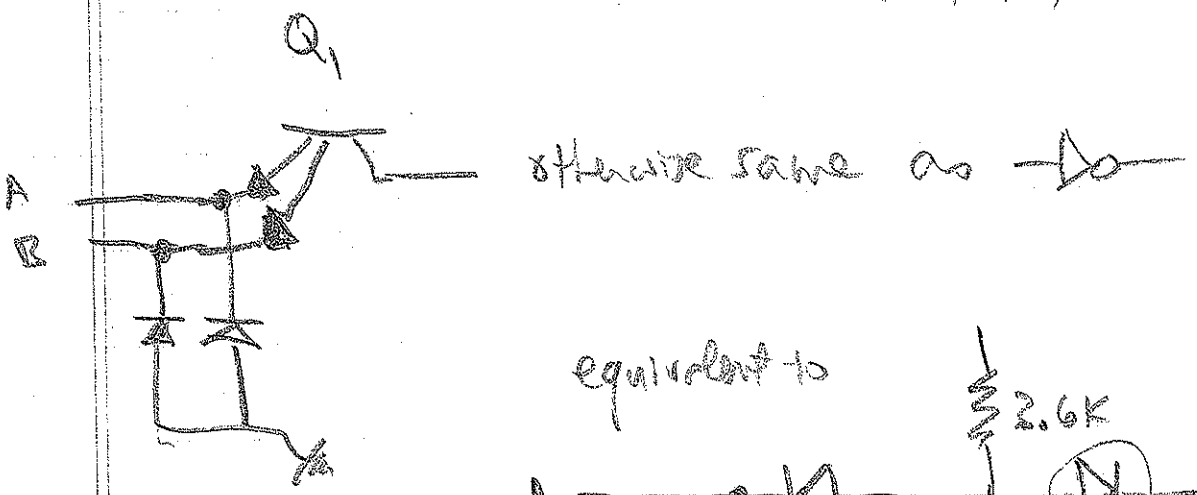
① check this out for sp) 2k

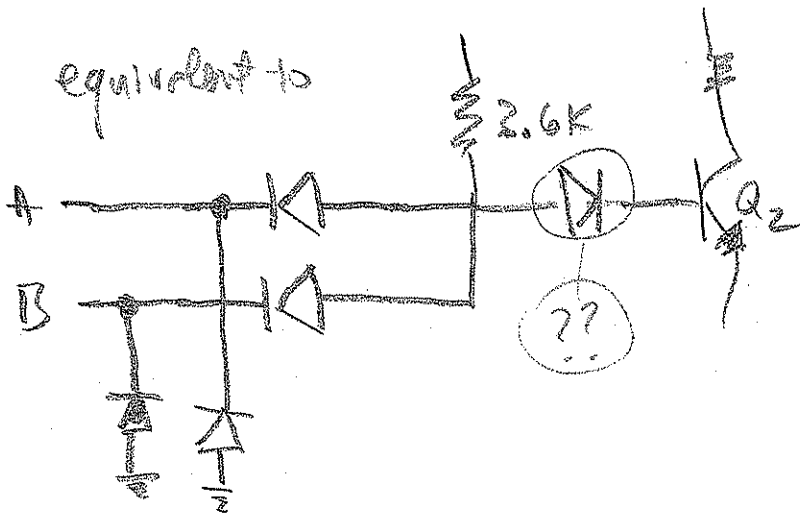
TTL inverter (7404 — somewhat different from 74LS04)

(12 pts)



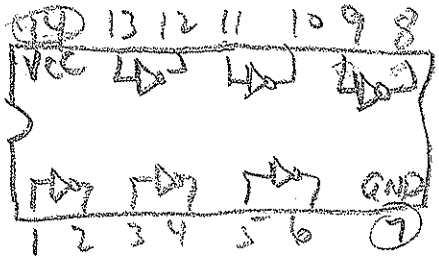
TTL NAND (7400)





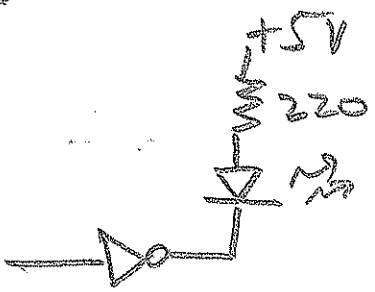
(2) Now get used to TTL inverter 74LS04

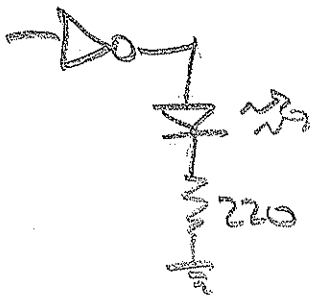
(a)



Connect Vcc & gnd. Measure pin #2 vout vs. pin #1 vin, not exceeding +5V !! Where is threshold?

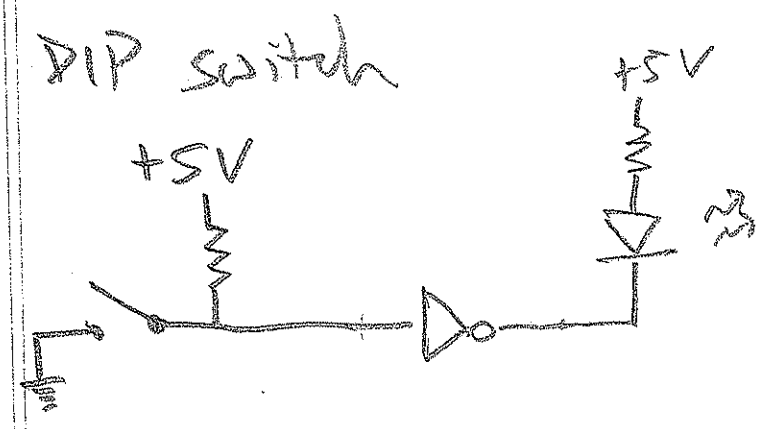
(b) Try LED 220Ω



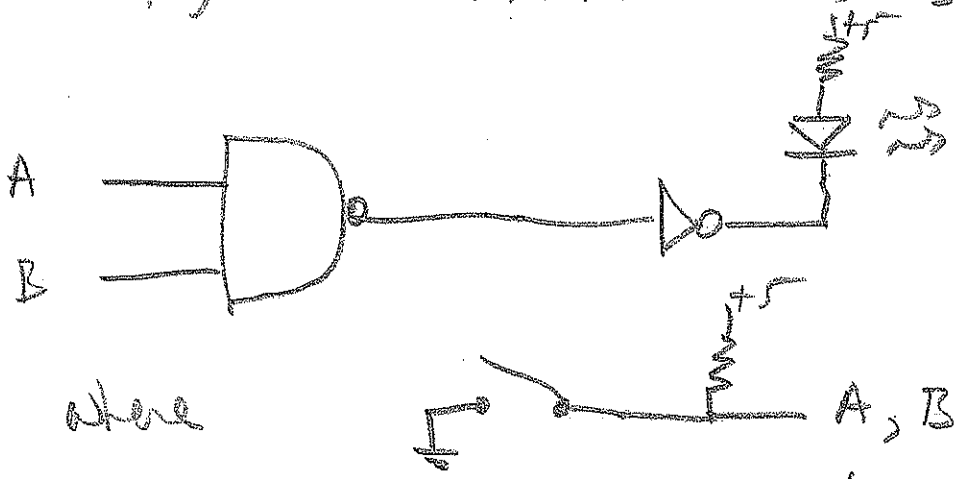


Which is current brighter? Measure (or infer) both ways.

(c) DIP switch

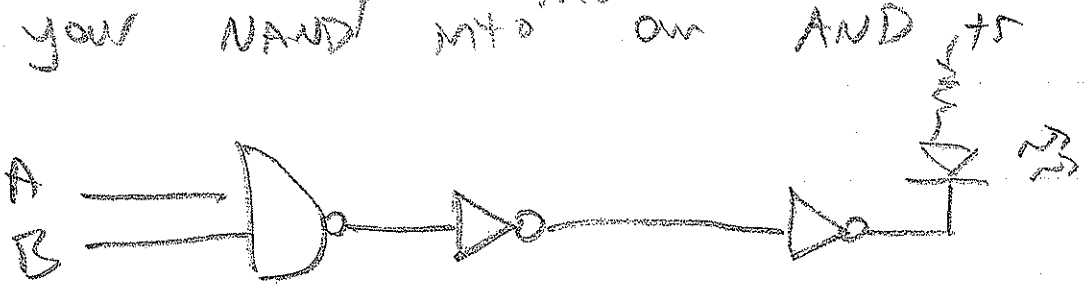


(3) (a) Now use your DIP switch (two inputs) and LED (with inverter) to try out 74LS00 NAND gate



Write out logic table.

(b) Use a spare inverter to make your NAND into an AND



(c) Now use a 74LS08 AND instead.

(d) Try 74LS32 OR now.

(e) Try 74LS86 XOR now

(4) Check out the 74LS83 4-bit binary full adder.

Prove to yourself that it correctly adds two 4-bit numbers together.

How would you build such an adder from discrete gates (AND, OR, XOR, inverters etc.)?

(Draw only — don't build it!)

↳ Don't forget Gnd, Vcc!

(5) Try a 74LS138 decoder!

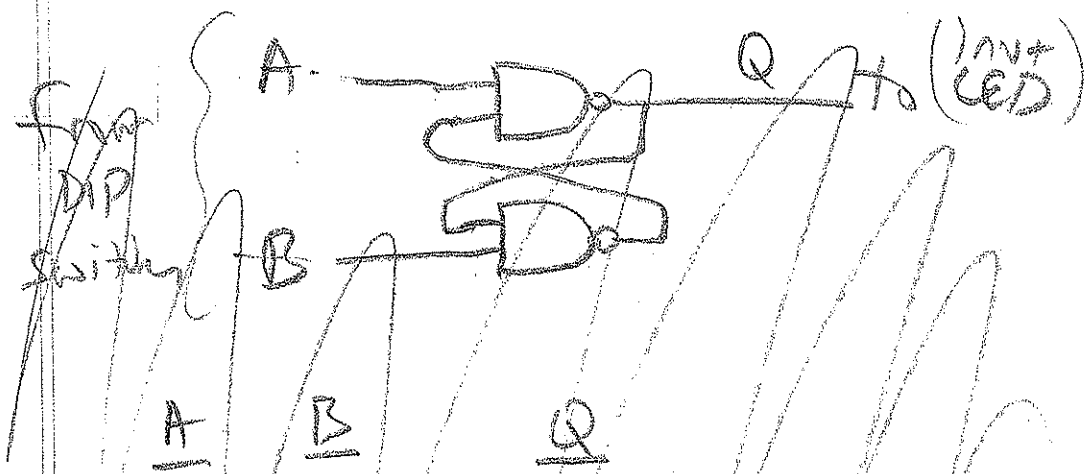
E_1, E_2 → GND ; E_3 → Vcc

use DIP to control A_0, A_1, A_2

display $\overline{O}_0 \rightarrow \overline{O}_7$ on LEDs.

For added fun, make E_3 oscillate as a 1 Hz square wave from function generator (careful! TTL pulse with voltage levels).

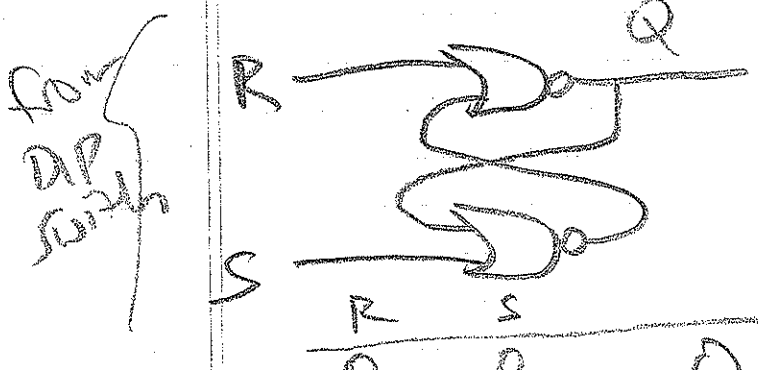
(6) primitive flip-flop



A	B	Q
0	0	1
0	1	1
1	0	0
1	1	Q

(sort of forbidden)

Notice that A acts as an active-low set*, while B acts as an active-low reset*. With both (active-low) inputs deasserted, Q remembers its old value!



7402 NOR

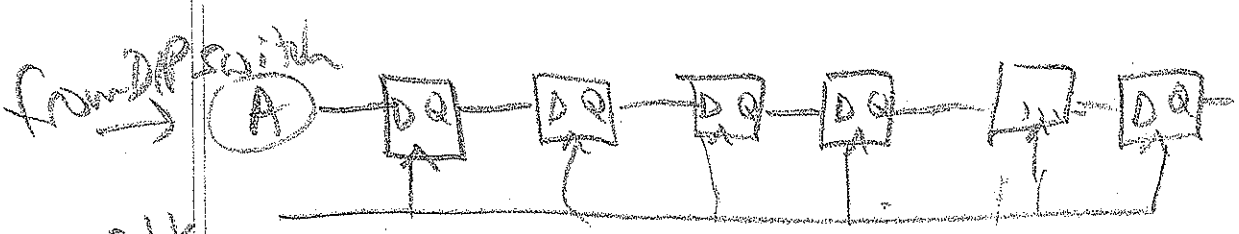
R	S	Q
0	0	1
0	1	1
1	0	0
1	1	(forbidden / 0)

See www.play-hookey.com/digital/d_(nand_nor)_flip-flop.html

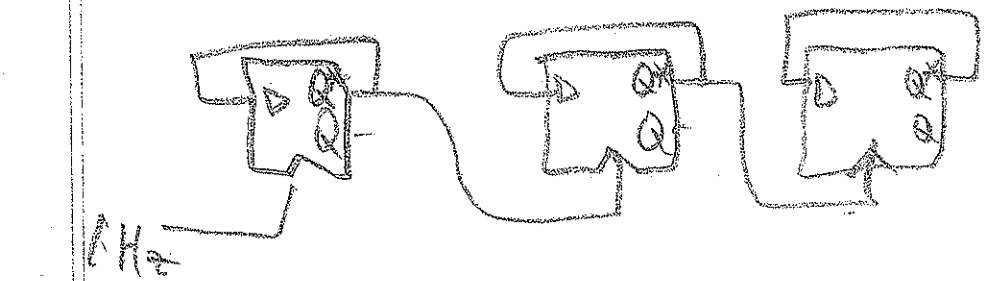
(5) Usefulness of clock: like the conductor of an orchestra, or the change-of class bell in a high school — tells everyone to change state at the same time.

Having a memory (hence statefulness) and having a clock (for coordinated changes of state) often go together.

74LS74 dual DFF (3 of them)



send all 6 Q's to your inverter/LED inputs.



notice that each successive Q counts at half the speed of its predecessor.

"ripple clock" (bad design, actually, as we no longer have a single drummer)

(8) Try 74LS164 8-bit shift register with same setup.

Try 74LS169 4-bit counter, with 1Hz clock, displaying to 4 LEDs

(9) One of the few metal discrete: a one-shot

74121 (74LS121 ?)