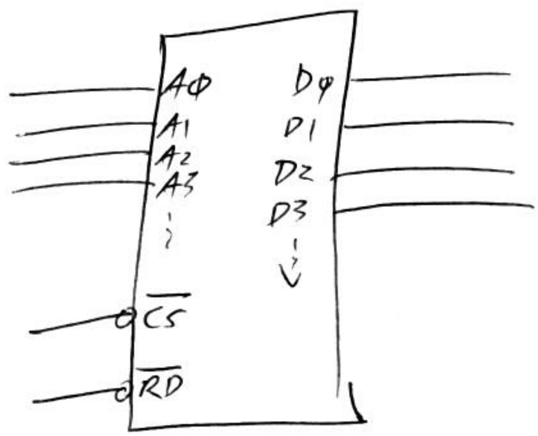
Memory:

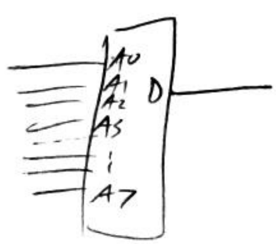
Basic idea -- large set of S-R latches, in simplest form. In fact, for PROM we have only fixed codes!

Empirically, we have, for ROM,

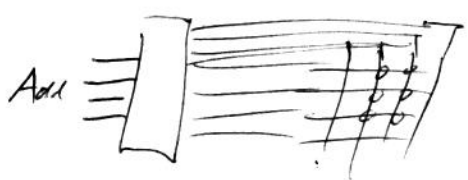


Address lines select internal locations, each of which stores n bits of data.

Easiest --- n x 1;



(here 256 bits)

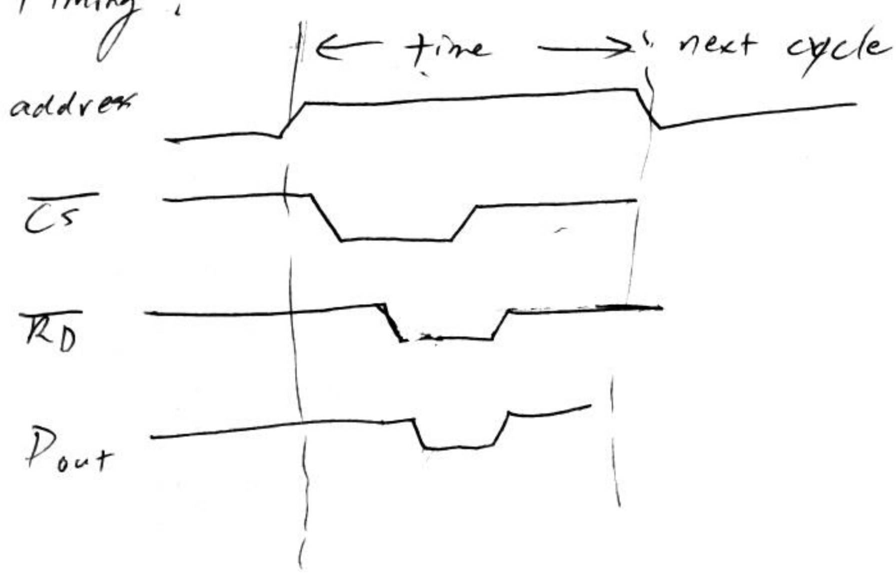


Erasure -- expose to UV light for several minutes.

2764	8K × 8
27256	32K × 8

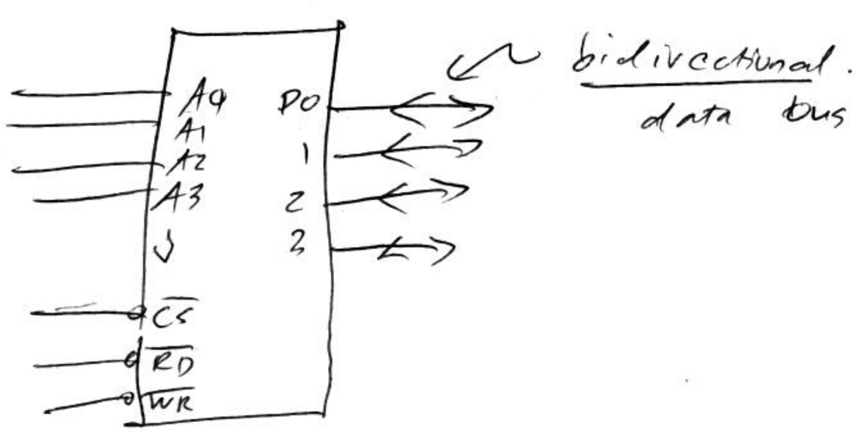
Programming -- set address & data, apply a high programming voltage.

Timing:



typ. time ~ 100 nsec or so.

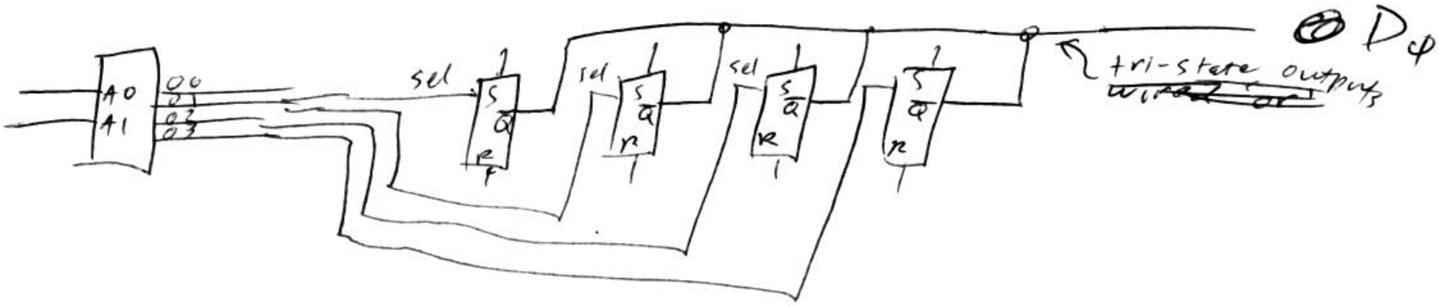
<u>Static RAM</u>: Very easy to use; just an array of S-R latches.



Here both read & write are random & fast,

1 - 100 nsec typ. < 1 nsec or so for cache..

Only difference from ff's is scale --

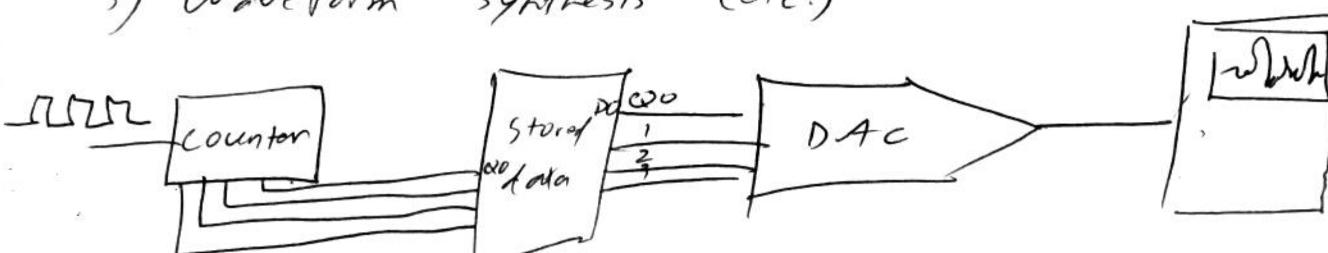


Uses:

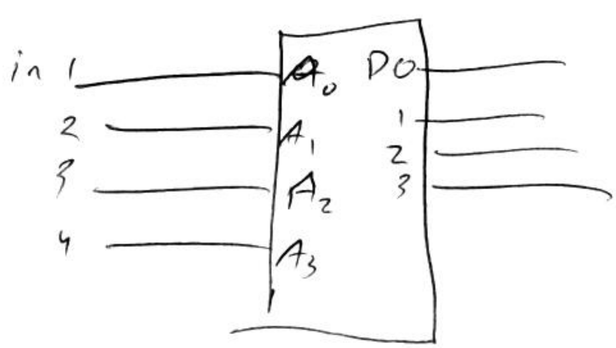
1) Information storage (of course)

2) Program storage " "

3) Waveform synthesis (etc.)

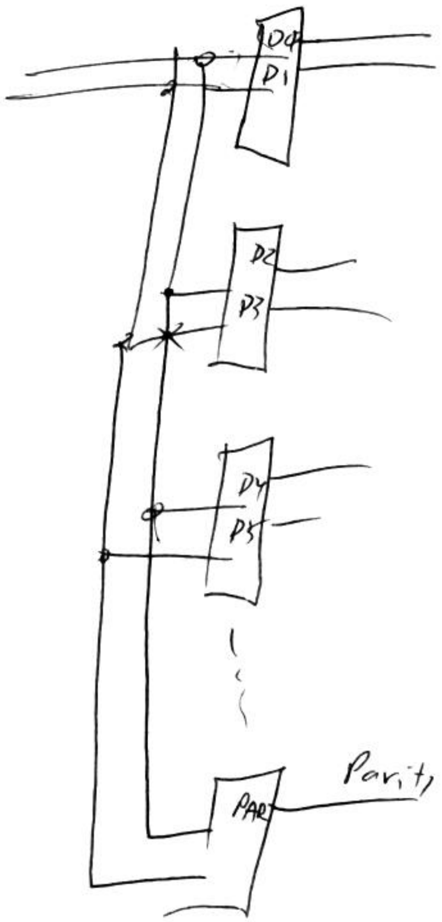


4) Arbitrary logic functions

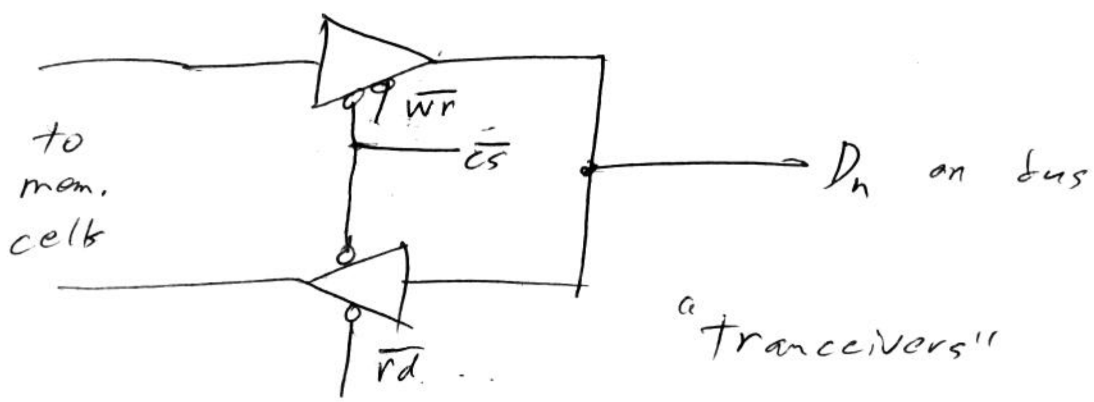


Just store proper output for every input.

lots of bits? Use chips in ∥:



But what about the data bus?? Let's look at a simple parallel bus:
We have a bidirectional situation ---
for each bit,



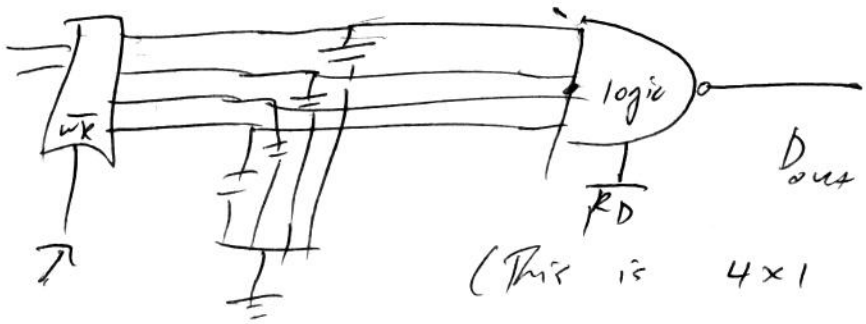
"Tranceivers"

Only one device can drive bus at a time.

⇒ be careful about control signals, transients

Dynamic RAM (main memory on PC's)

Just a bunch of capacitors, typically --



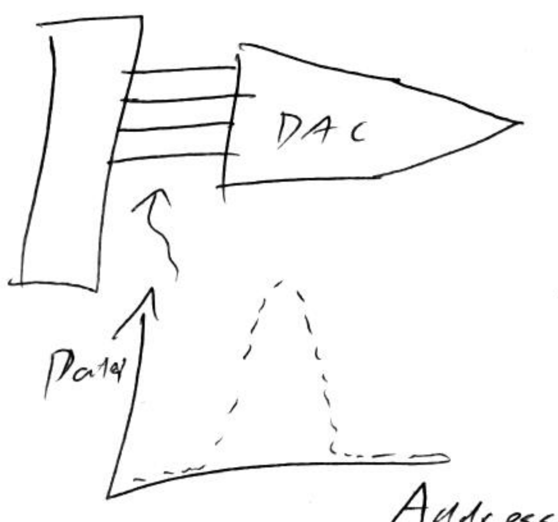
(This is 4×1 = 4 bit)

Writing -- store charge for required ≤ 1s.

Reading -- interrogate w/ logic gates.

Low thresh'd ⇒ poor protection typ.

Ex: Gaussian waveform generator



So store:

A_0	A_1	A_2	Q_ϕ (LSB)	Q_1	Q_2	---	Q_8 (MSB)
0	0	0	0	0	0	----	0
1	0	0	1	0	———		
0	1	0	0	1	0	———	
↓			01	1	0	———	etc.