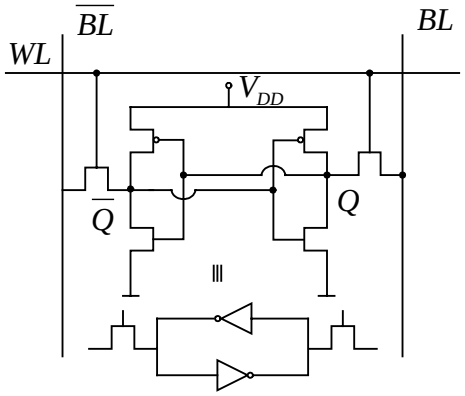
Random Access Memory



You can read/write
at comparable speed

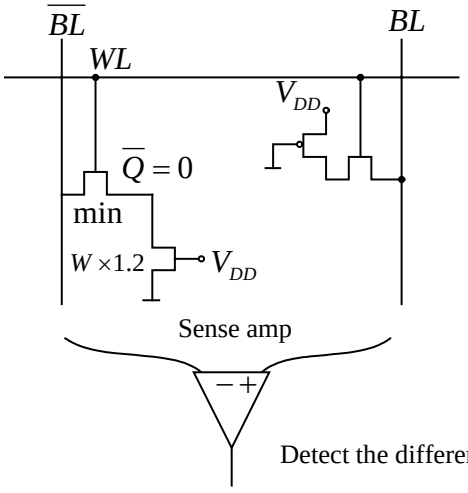
Six transistor CMOS RAM

Read Operation

active word line

BL , $\overline{BL}$ are precharged to V_{DD}

eq ckt : (Q = 1)



Size of pass transistor
has to be smaller
than M1 to avoid
read upset $\frac{NMOS \quad W/L}{Pass \quad W/L} > 1$

if the lines are precharged
to $V_{DD}/2$ better performance
can be achieved + no read upset

Detect the difference between the two lines

Write operation

* BL and $\overline{BL}$ are set to different values

* WL is activated

This works identical to SR latch

Cover powering the latch

$$\frac{PMOS\ \ W/L}{Pass\ \ W/L} > 1.8$$

$PMOS$ → $\min imum\ \ size$ $\times 2$

Write time → dominated by
t_p of cross-coupled inverters

Make it minimum

Resistive SRAM

→ Replace PMOS with resistive loads

* 4 trans instead of 6

* much smaller area since inter connection
routing is easier + no need for n-well
) $\frac{2}{3}$ $of\ \ area$

* higher power consumption (in stand by)

Resistors do not introduced slower cell as the precharge is done extrenally
so make R as big as possible limt → area

undoped poly as resistor $\left(T\,\Omega/\square\right)$

$10^{-15}\,A/cell$ should provide current at least 2 ordered magnitude its cell taken

Dynamic RAM

Resistor SRAM → resistor function is to provide current to compensate for leakage

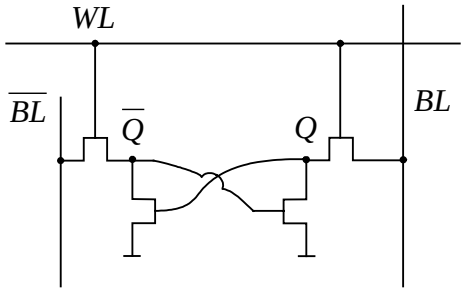
→ we can omit the resistors and refresh the memory periodically → dynamic RAM

refreshing → read the data then

write the same data

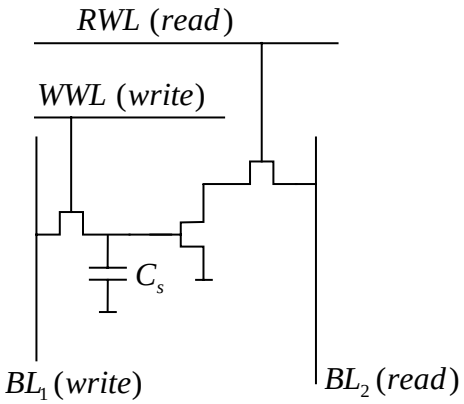
1 - 4 msec → refreshing period

deleting the resistors will result in



⇒ You do not need to store both Q and $\overline{Q}$ so you can further simplify it to 3 transistor version

3 transistor version



to write

* Put data on BL_1
* activate WWL

→ data will be store in the capacitor (C_s)

charge → 1

no charge in C_s → 0

to read

* Precharge BL_2 to V_{DD} or (V_{DD} - V_T)
* activate RWL

if BL_2 goes down (0) → you had stored 1

if BL_2 stays at V_{DD} → you had stored 0

→ you need to invert the data (inverting sense amplifier)

Cell size 1/2 of SRAM

* you can further reduce the size of the cell
  for instance BL_1 and BL_2 can be merged into one line

* WWL and RWL can also be combined and
  you can read data just like when you refresh data

3-transistor cell dynamic RAM

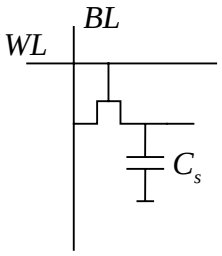
1. unlike SRAM → no constraint on device ratios
2. reading 3-t SRAM is non-destructive
3. standard CMOS technology → suitable for embedded memory
4. the voltage stored at X (C_S) is $V_{WWL} - T_{Th}$

Therefore during read time you have loss voltage

↘ less current ⟶ higher t_p

to avoid this situation you can bootstrap
V_{WWL} to higher than V_{DD}

1-transistor dynamic memory



⇒ not a gate capacitor ~ 30fF

So you need a technology that
gives you a reliable compact capacitancce

Write cycle

* activate WL
* put data on BL

  data will be stored (0 or 1) on the capacitor

$0 \rightarrow C_s = 0V$

$1 \rightarrow C_s = V_{DD} - V_T$ ⟶ You always use bootstrapping of WL

Read Cycle

* precharge bit line to V_{PRE} ($0 < V_{PRE} < V_{DD}$)
* activate WL
* V_{BL} starts to change by

$$\Delta V = \left(V_{Bit} - V_{PRE}\right) \cdot \frac{C_s}{C_s + C_{BL}} \qquad \sim 250mV$$

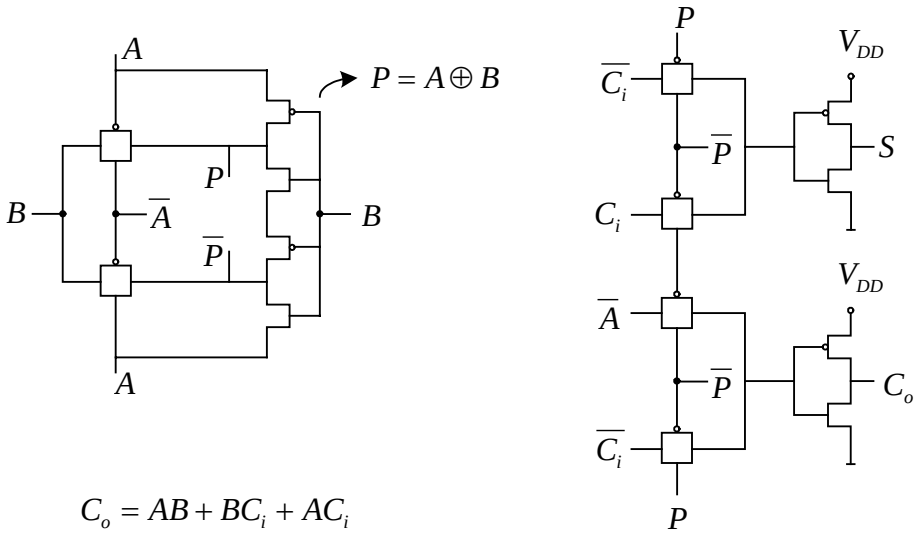
needs amplification

$$\frac{C_s}{C_s + C_{BL}} = charge\ transfer\ ratio\ \sim 1\%\ \ to\ \ 10\%$$

Direction of charge determines the data (0 or 1)

Adder

Consider transmission gate adder



$$C_o = AB + BC_i + AC_i$$

$$S = ABCC_i + \overline{C_o}\left(A + B + C_i\right) \quad \longleftarrow \quad S = A \oplus B \oplus C_i$$

Problem is the delay propagation in carry generation

that slows down the odd process $\longleftarrow$ $t_{adder} \approx (N-1)t_{carry} + t_{sum}$

Consider the following signals

$G = AB$ $\longleftarrow$ generate carry

$D = \overline{A}\,\overline{B}$ $\longleftarrow$ delete carry

$P = A \oplus B$ $\longleftarrow$ propagate carry C_o and S

We can rewrite

$$C_o = G + PC_i$$

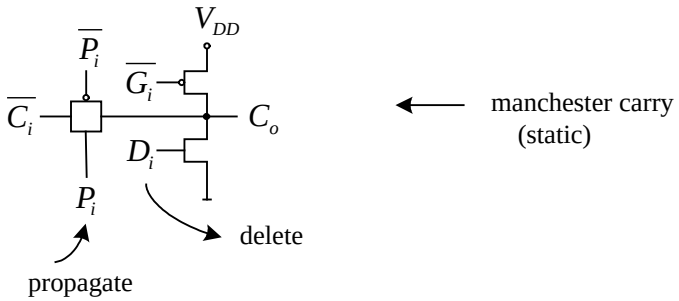
$$S = P \oplus C_i$$

Truth table for adder

generate delete propagate

A	B	Ci	G	D	P	S	Co	Carry State
0	0	0	0	1	1	0	0	Delete
0	0	1	0	1	1	1	0	Delete
0	1	0	0	0	0	1	0	Propagate
0	1	1	0	0	0	0	1	Propagate
1	0	0	0	0	0	1	0	Propagate
1	0	1	0	0	0	0	1	Propagate
1	1	0	1	0	1	0	1	Generate
1	1	1	1	0	1	1	1	Generate

So to make Co what you can do is to

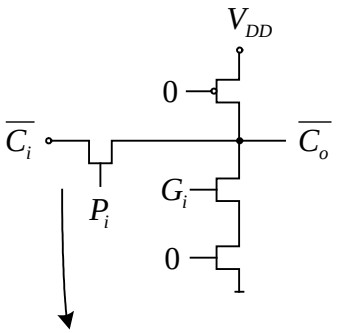


manchester carry (static)

has much shorter delay

reduces the delay of the ripple carry adder

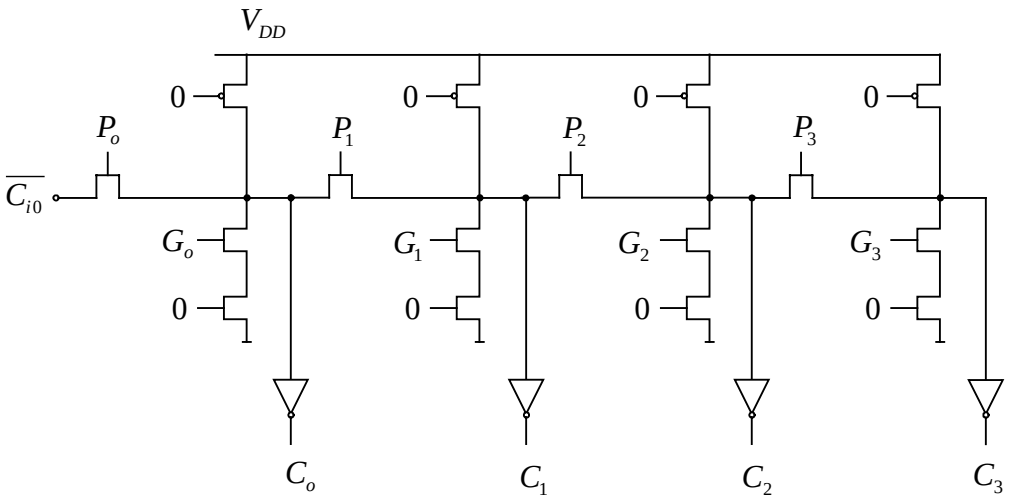
..Manchester Carry generation (dynamic)



no need to delete
since you precharge
the carry to V_{DD}

need only one pass transistor
instead of transmission gate

Carry Chain Adder



* G and P only depend on A and B so the delay in carry generation is only due to this ckt which is much faster than ripple carry adder

Advantages

* much easier/smaller layout
* much faster carry generation path

$$t_p = 0.69 \frac{N(N+1)}{2} RC \longrightarrow t_p = (N-1)\, t_{carry} + t_{sum}$$

campare with

R: pass transistor resistance

C: node capacitance (4 diffusion cap + 1 inverter cap)

distributed nature results in quadratic increase of delay

with N ⟶ if N = 128 (for supercomputers)

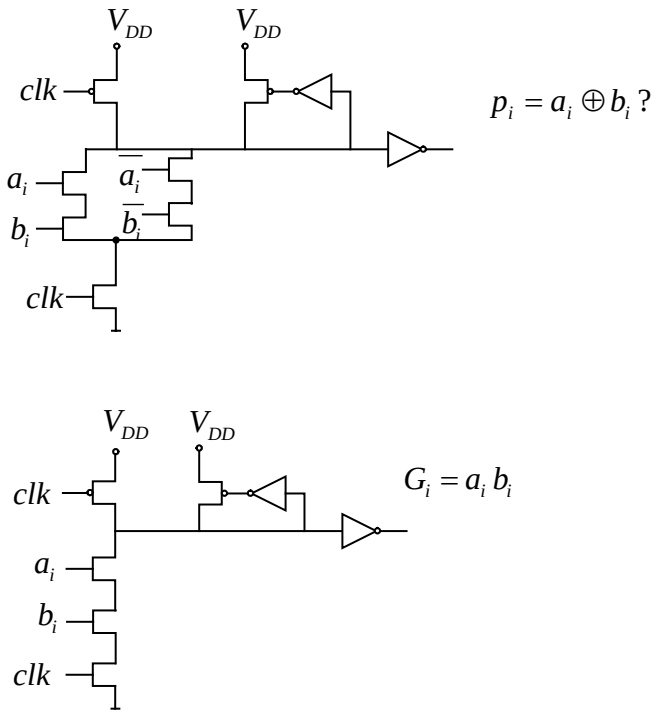
64 (for servers)

32 (for PCs)

delay becomes extremely large ⟶ so every 3~4 bit do buffering

that would make delay linear function of N but adds up the buffer delay

Dynamic implementation of propagate and generate signals

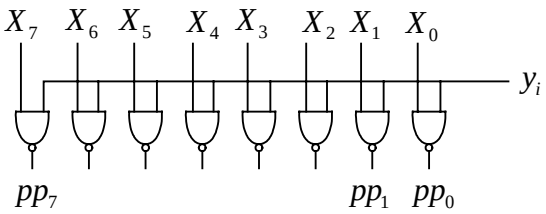


Multiplier

1st task to generate partial products

	1 0 1 0 1 0	← multiplicant
×	1 0 1 1	← multiplier
	1 0 1 0 1 0	partial products
	1 0 1 0 1 0	
	0 0 0 0 0 0	
	1 0 1 0 1 0	
	1 1 1 0 0 1 1 1 0	← result

Partial-product generation



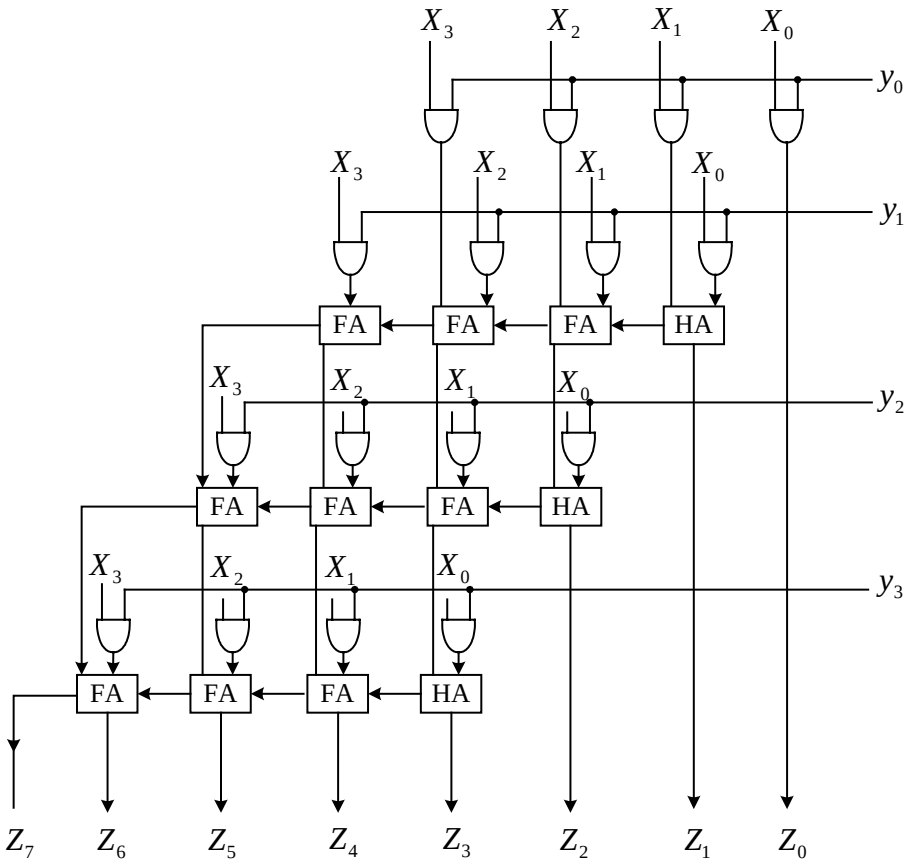
Then you have to sum the pp terms

you can use an array to do so

Array Multiplier (4x4)

* shifting is done by routing the signal
* pp generation is done by using AND gates

$$X \times Y = Z$$



Delay Depends on the critical path ⟶ what data are you multiplying

$$t_{mult} \approx \left[(M-1)+(N-2)\right]t_{carry} + (N+1)t_{sum} + t_{end}$$

$M \rightarrow X_0 \rightarrow X_M$

$N \rightarrow y_0 \rightarrow y_M$