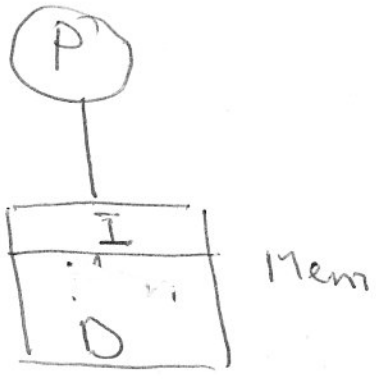
Main Memory

- Introduction (cpu and Mem)
- SRAM (organization, Timing, Expansion)
- DRAM (organization, Timing, Access modes, Interleaving, EXPANSION)

Introduction

Processor and Memory model

- Instructions stored in Memory. Processor uses Fetch
- Data stored in Mem Processor uses load/store



Each single instruction might access memory twice (Fetch, and load/Store)

Example lw R1, 100(R2)
R1 ← M[100+R2]

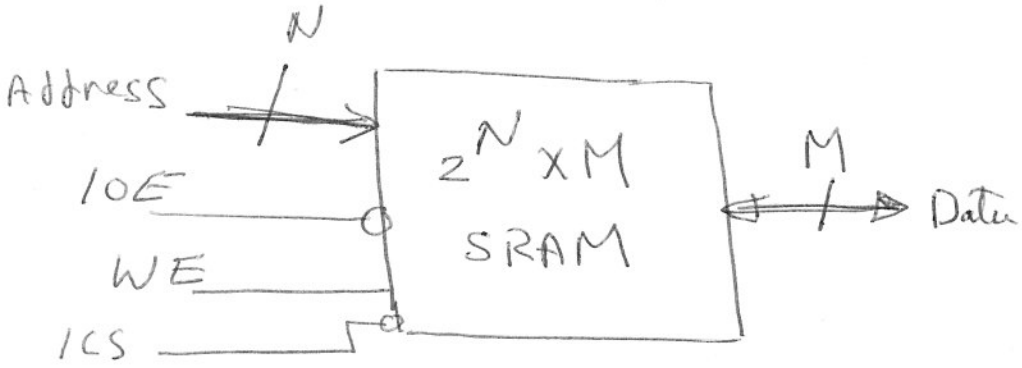
- Using RAM = Radom Access memory. All accesses have same access time
- Computer system performance depends on memory speed (or cpu??)

Main Memory

SRAM "Static Random Access"

- Fast, small in size, high power consumption, high cost / storage bit

Block Diagram



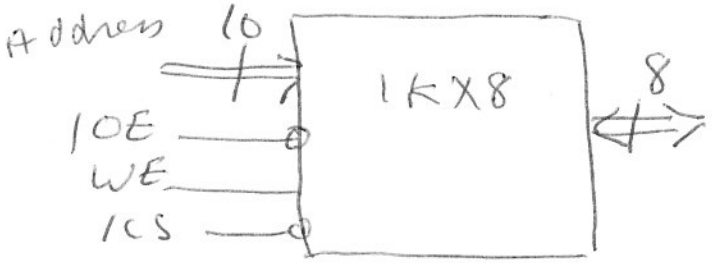
Memory size = number of locations x size of each loc.
$= 2^N \times M$ in bits

N = number of address lines
M = number of storage bits in each location

Example:
- Draw BD for a 1K X 8 SRAM

- Address = $\log_2 1024$
= 10

Data Bus = 8 bits



Example

Find the value of Address, data and control signals to write FE. To Loc 1C2 in a 1K x 8

Address = 1C2 = 01 1100 0010

A9 - - A0

Data = FE = 1111 1110

D7 - - - D0

WE = pulse (Timing)

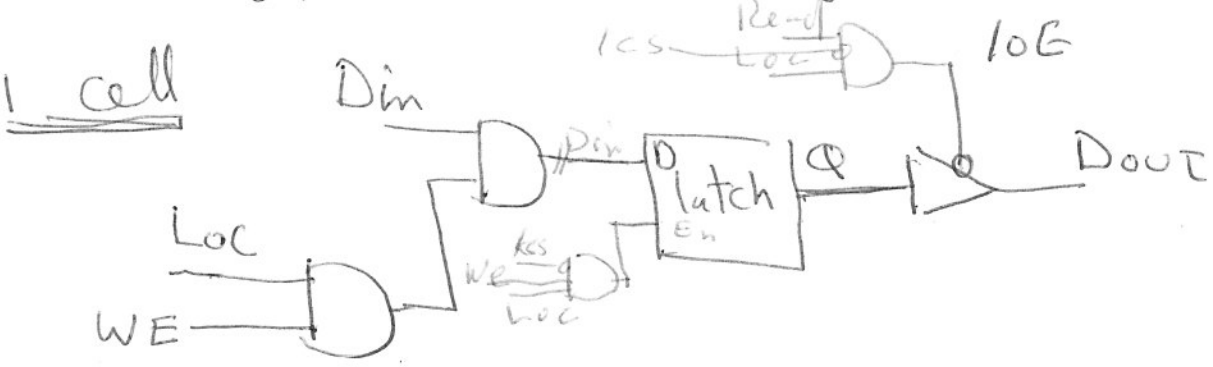
/CS = 0 Active

/OE = 1 False

SRAM organization and Implementation

-using latches (Transistors) organized as 1024 location by 8 latches in each location

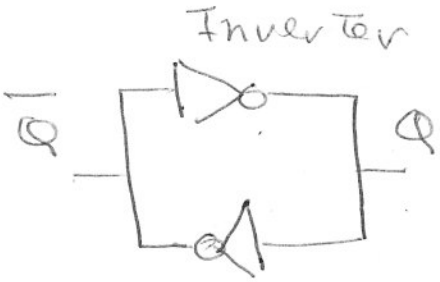
number of latches = 8K bit



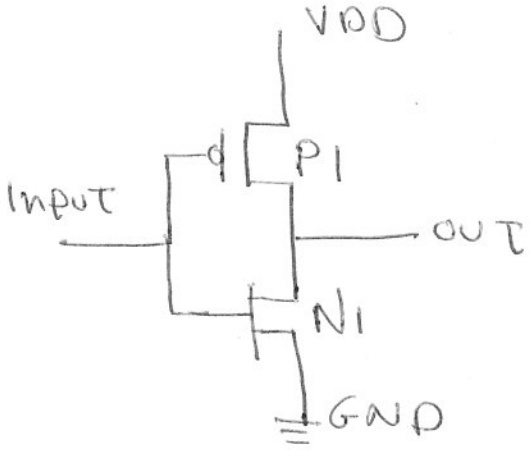
circuit complexity: cost, speed....

SRAM organization

Latch



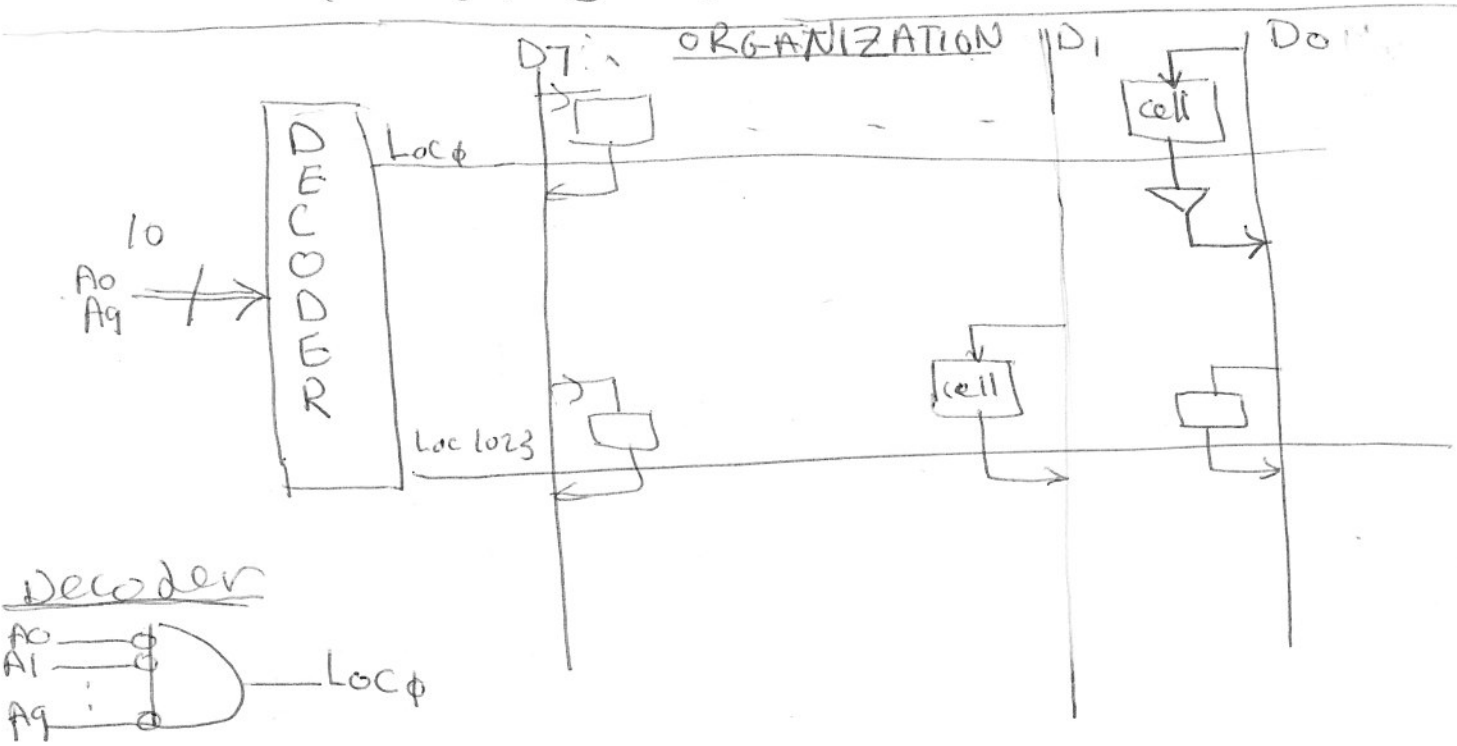
Inverter



Input = 0
P1 conducts, N1 off
OUT = VDD = 1

Input = 1
N1 conducts, P1 off
OUT = GND = 0

ORGANIZATION

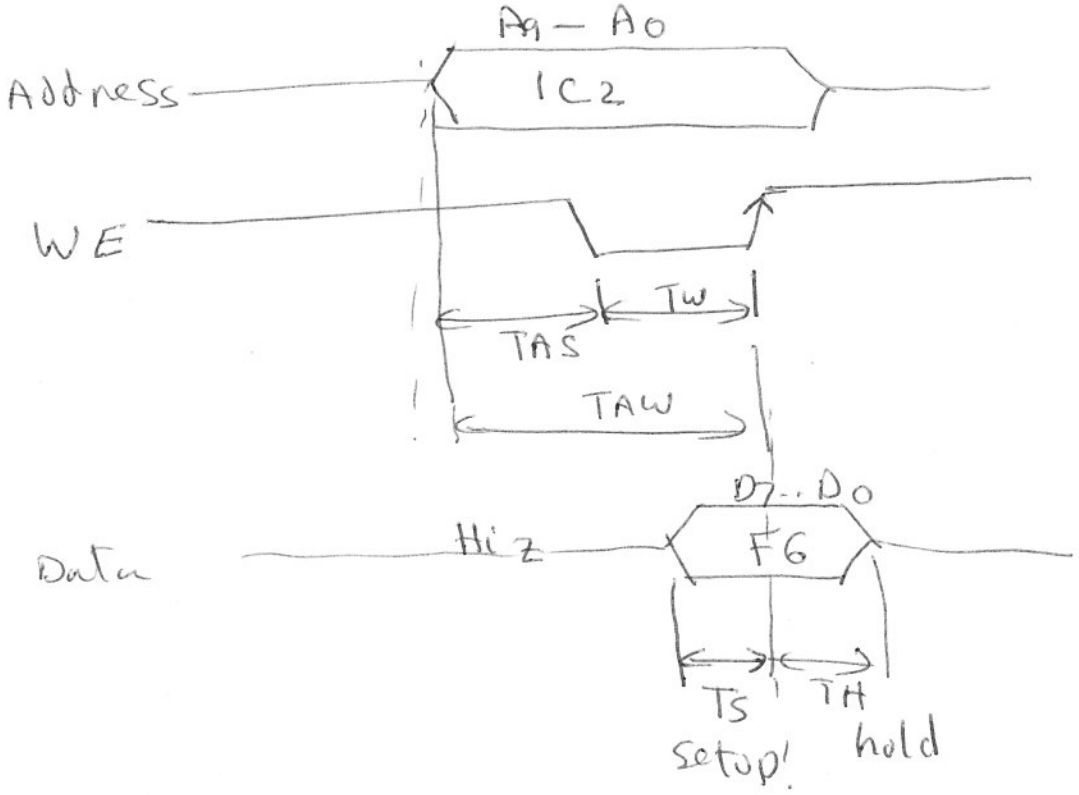


Decoder

SRAM operations and Timing

- Most important design consideration in high speed computer circuits
  - need to consider - min — max delay
    - wire delay
    - fall & rise time

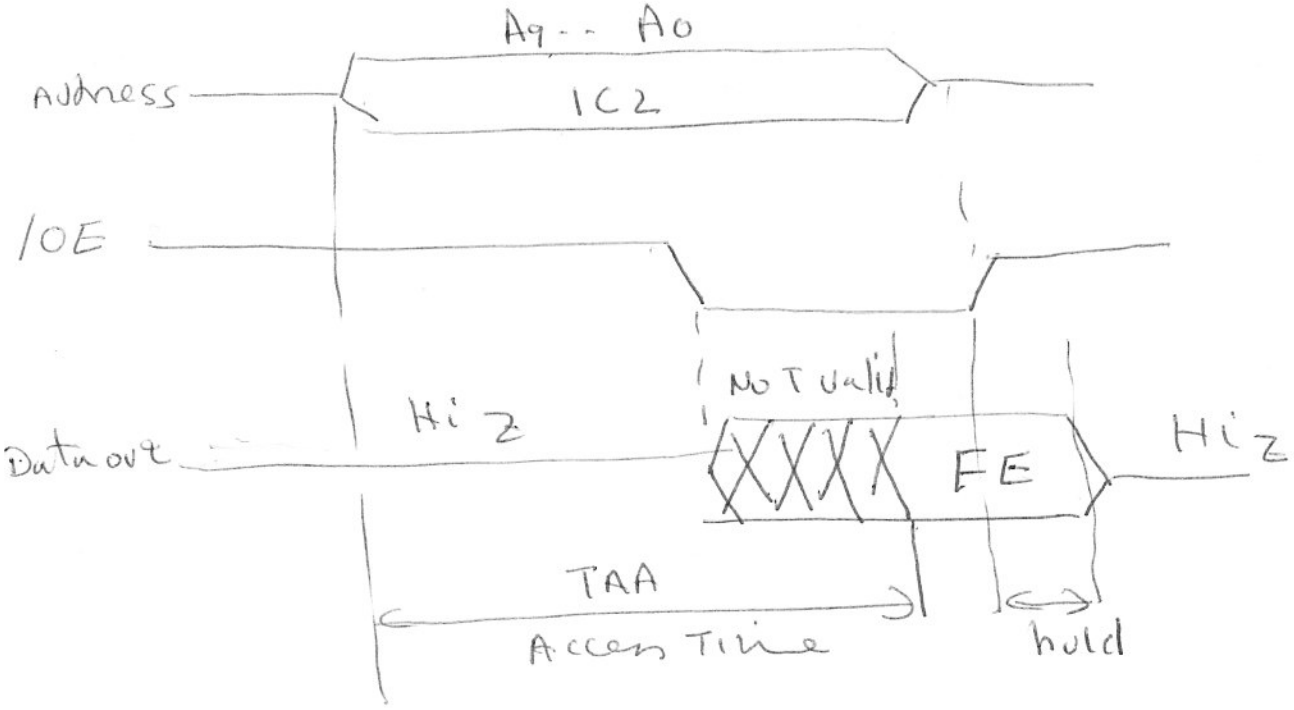
SRAM write operation



Address stable, WE pulse, data setup, WE edge hold data.

need to wait for decoder delay, latching of data.

SRAM Read Operation (Timing)



Memory Expansion

Construct 4K X 16 using 1K X 8

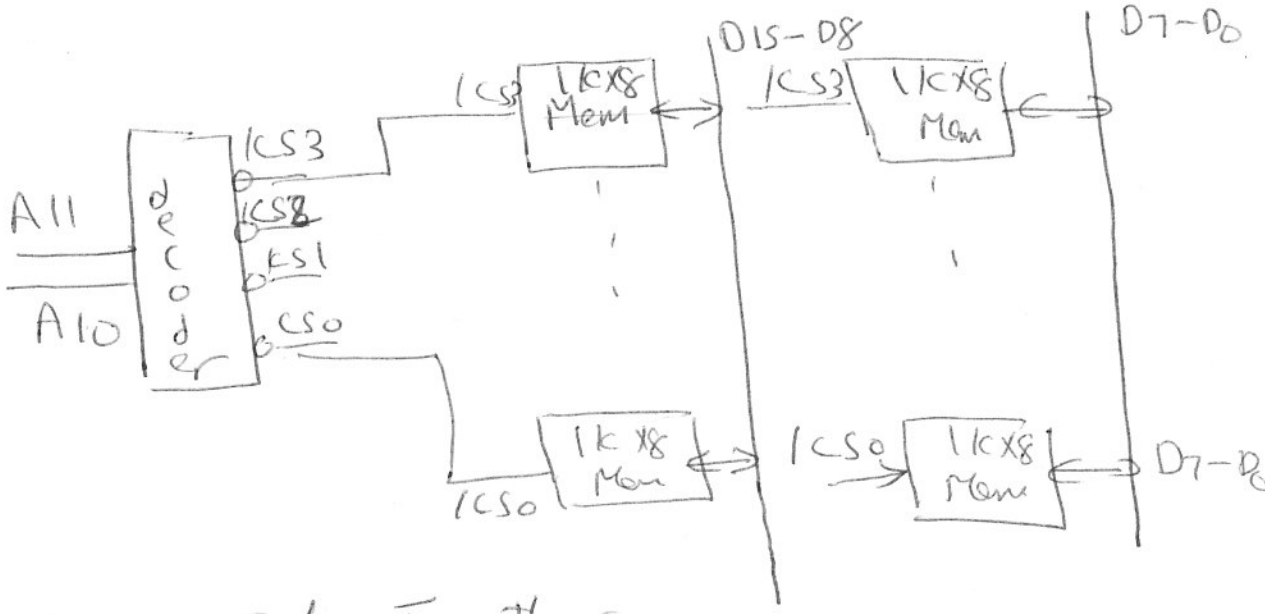
- need $\frac{4K \times 16}{1K \times 8} = 8$ chips

Address lines: $\log_2 4K = 12$

data lines = 16

Need external decoder to select 2 memory chips at a time.

organization = 4 X 2



WE = Connected Together

IOE = Connected Together

Main Memory

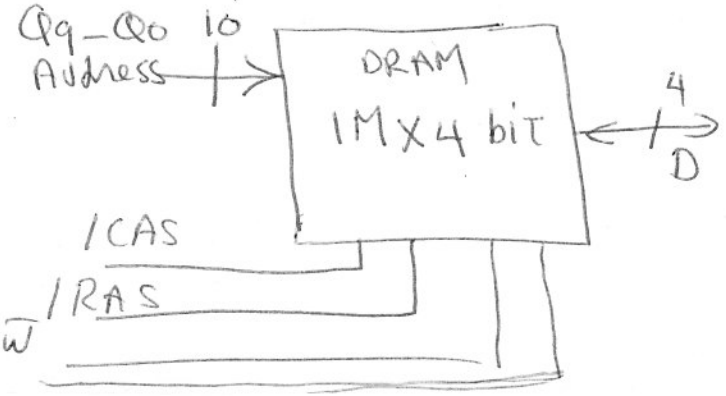
DRAM

"Dynamic Radom Access Memory"

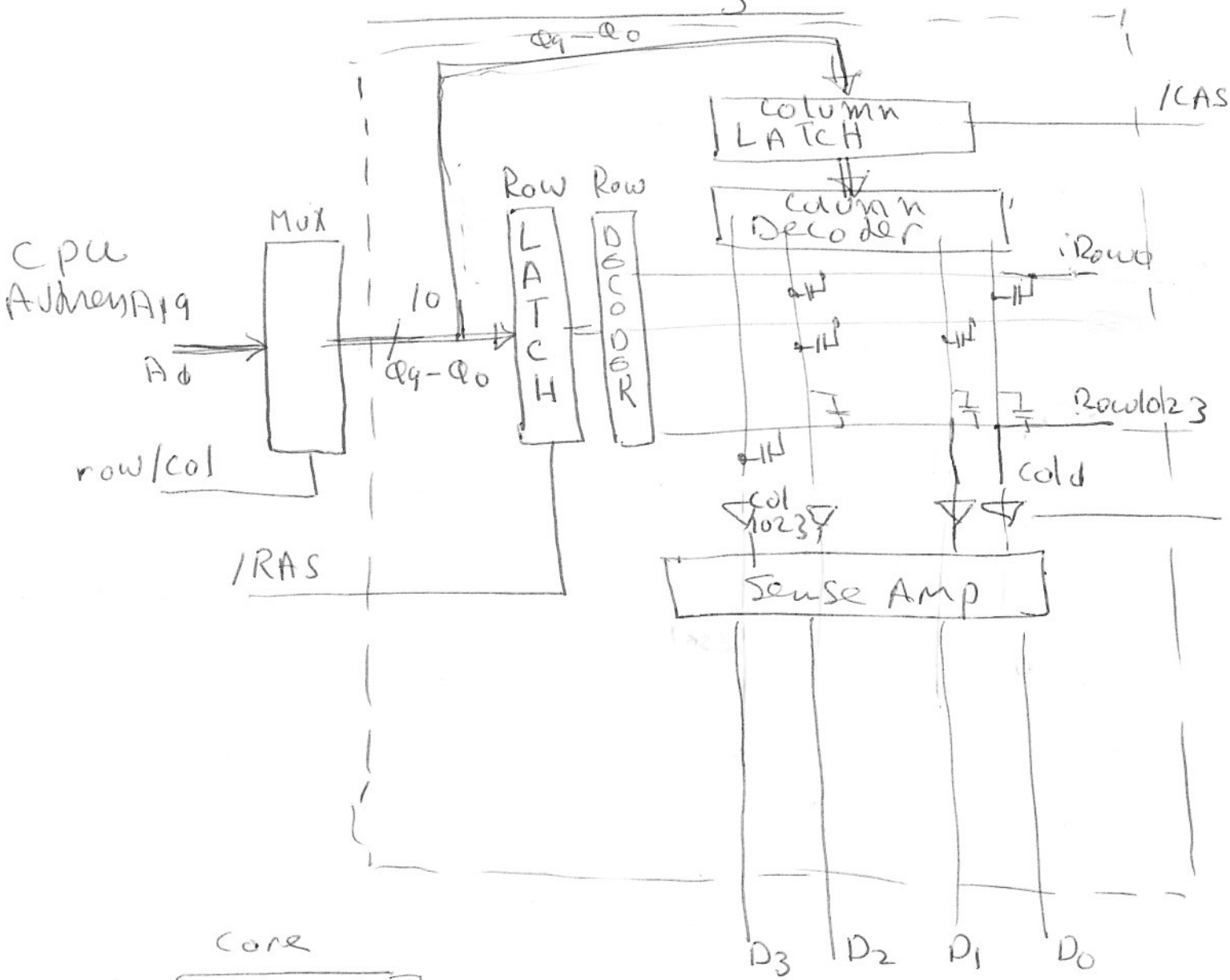
- Slow, large size, low power, less cost.
- Basic cell uses capacitor for storage
- Need Refresh every specific period (leakage of charges, ∴ dynamic)

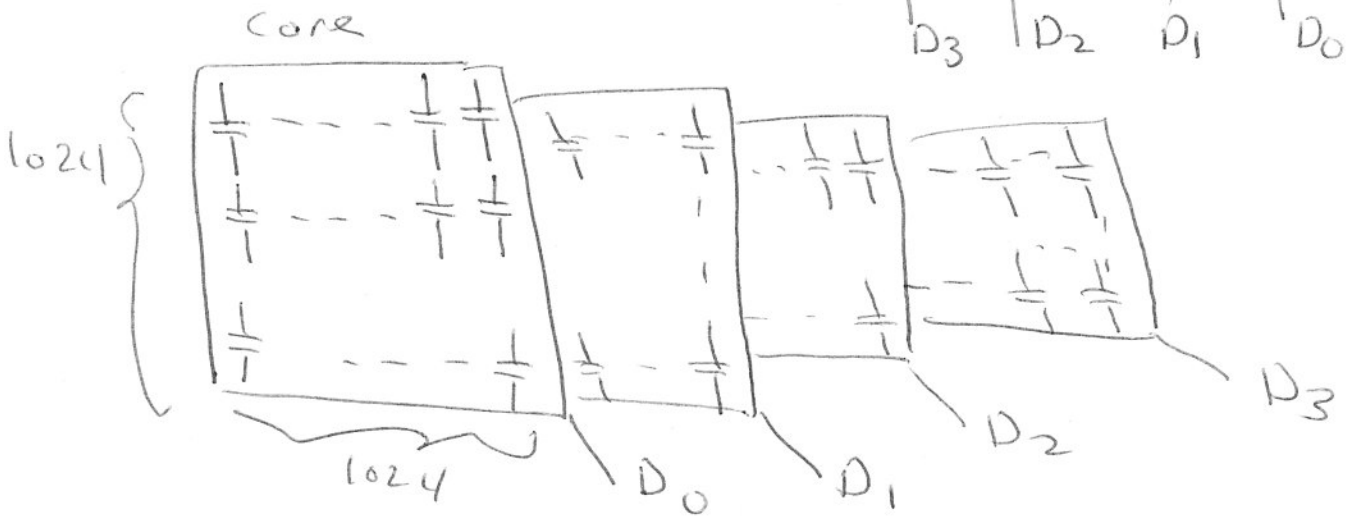
Block Diagram

Multiplexed Address (Rows, Columns)



DRAM ORGanization



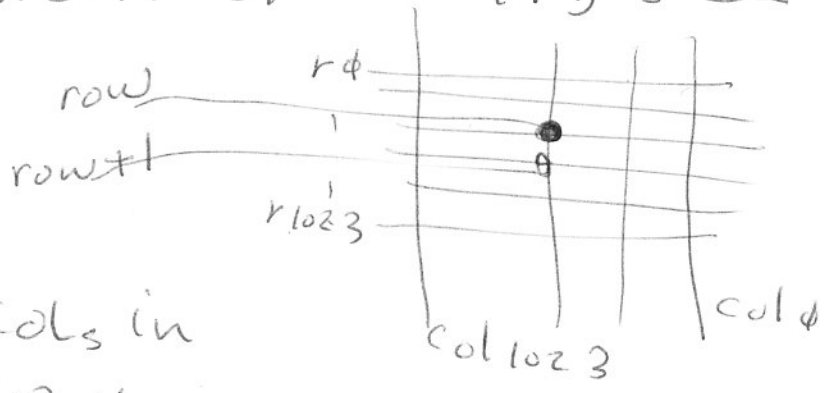


4M Capacitors

EXAMPLE

- USING 1MX8 DRAM, find row# & COL# for Address = 35E6A

- Find Address of row+1, same Col

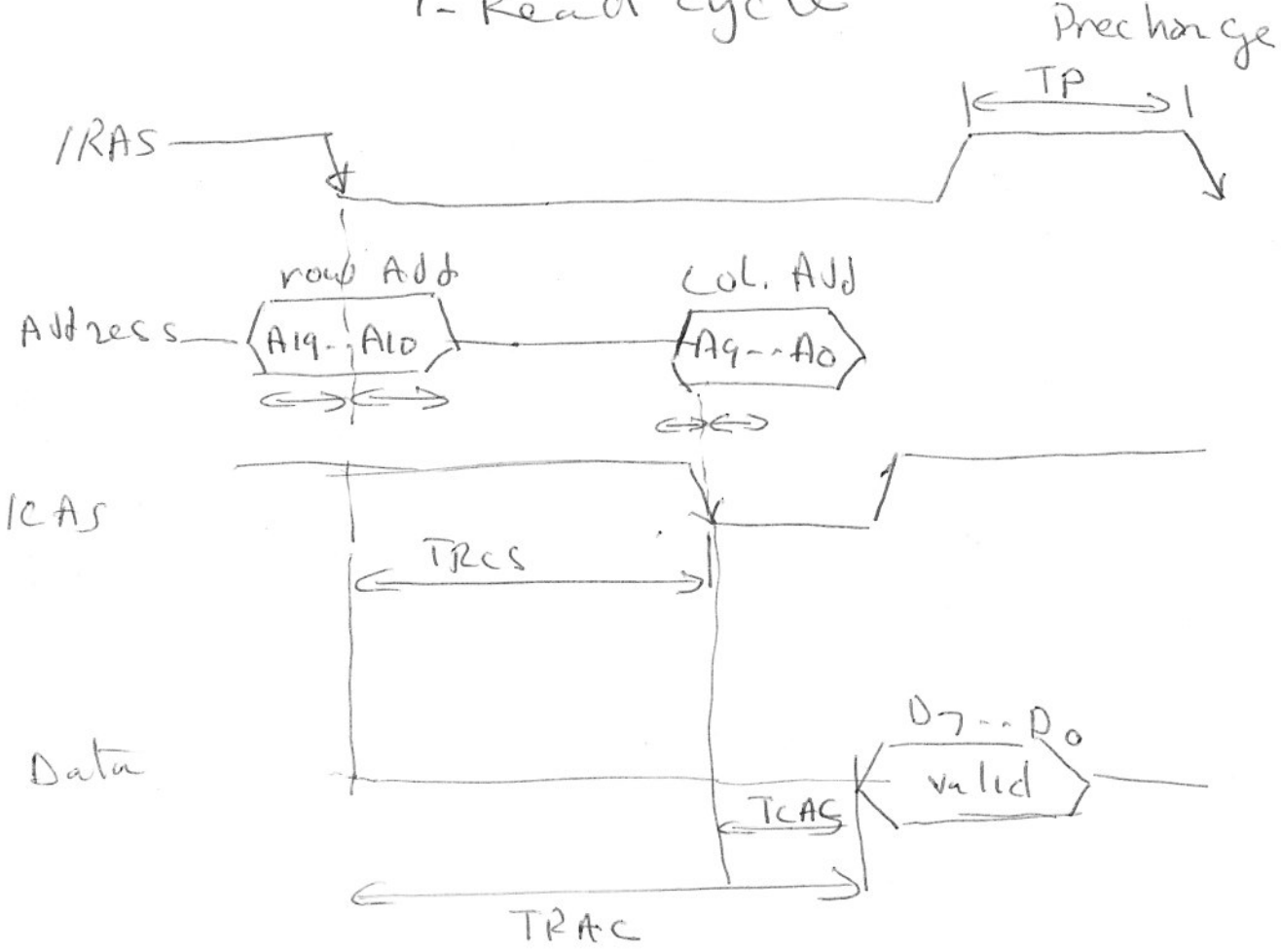


Number of Cols in a row = 1024

DRAM Operations and Timing

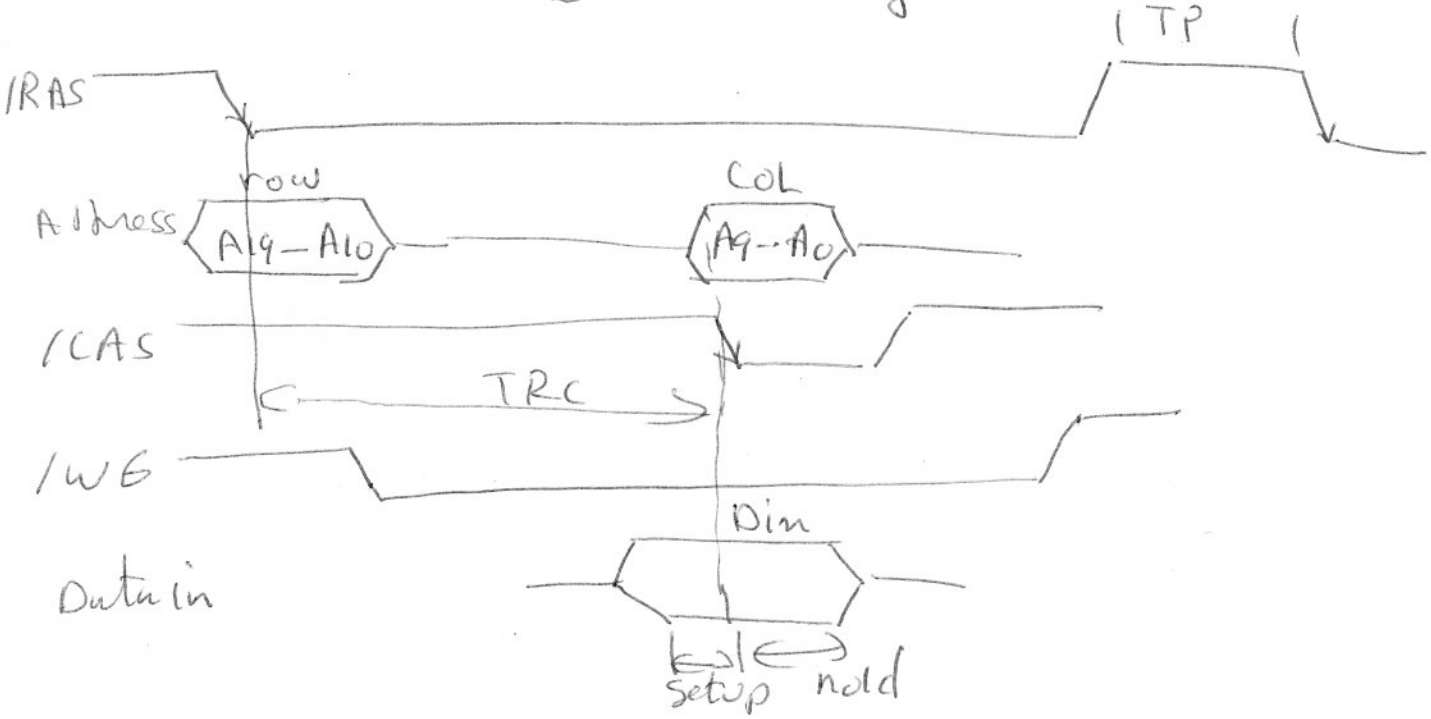
Complicated:- must latch row, latch column, using /RAS, /CAS
- support Referesh
- try to increase its speed (interleaving, FP mode, ...)

1- Read cycle



All Timing prameters must be satisfied For correct operation.

2- write cycle



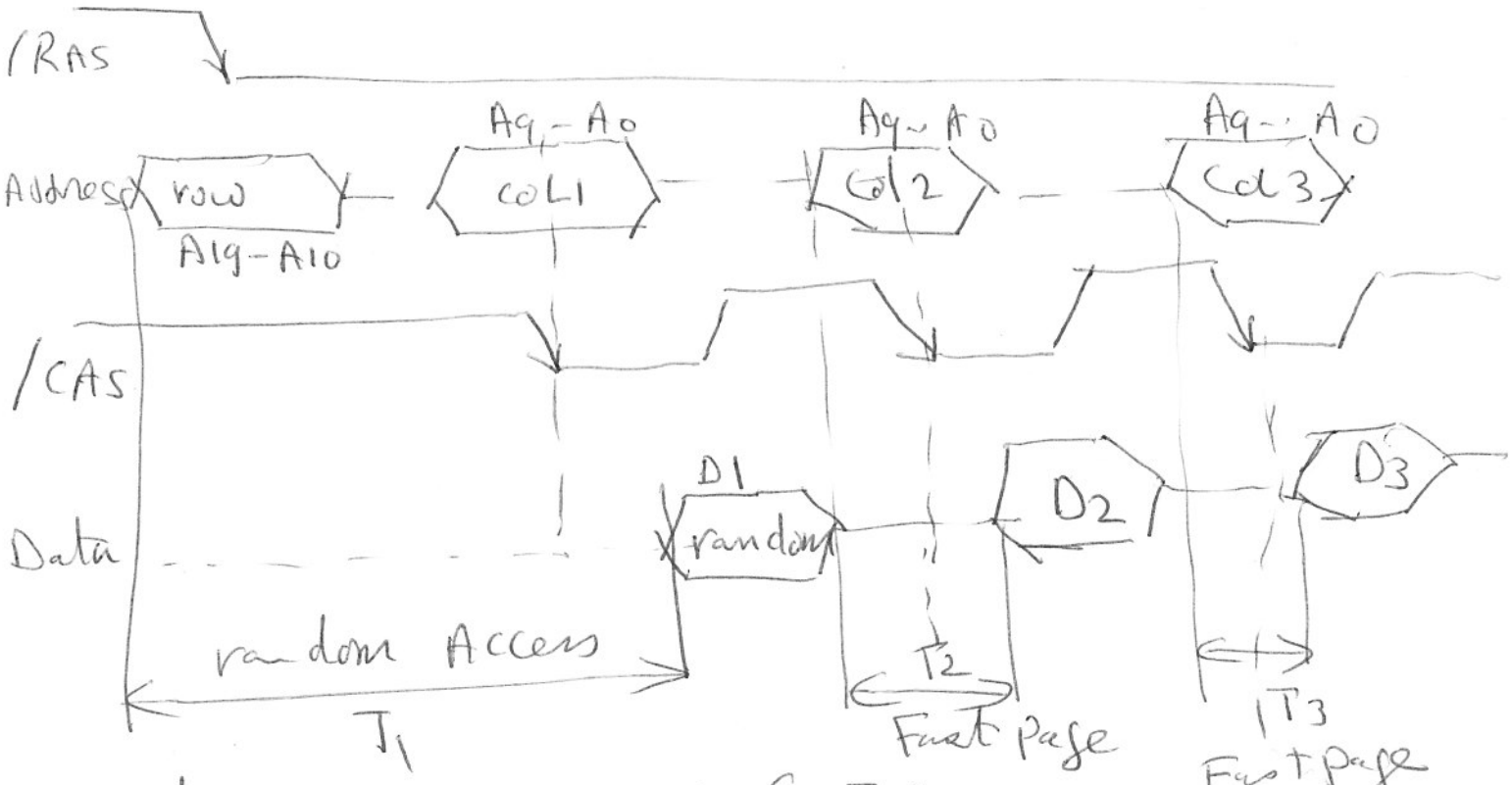
DRAM Access Modes

Fast Page Mode

Locations that are in the same row, need not to latch same row, could be accessed using faster FPM (3 Times faster)

The need is only to latch column address, while keeping /RAS active (=0) low to Latch same row.

Fast Page Mode Timing



random Access Time ~ 60-70ns

Fast page Time = 20-30ns

Cost of Supporting Fast Page mode:

1- Latch to accessed row
2- Comparator to compare latched row with incoming row address

IF current access is not on same row comparator out a miss.
DRAM must use random access mode (latch new row, column,)

Refresh

DRAM uses array of capacitors to store information, (due to leakage) they must be refreshed.

- All rows in DRAM may must be recharged at a fixed period that is = $\frac{\text{refresh Time}}{\text{number of rows}}$

Assume a 1M DRAM, and refresh Time = 8 ms,

Each row must be refreshed every $\frac{8\text{ ms}}{1024} \approx 8\,\mu s$

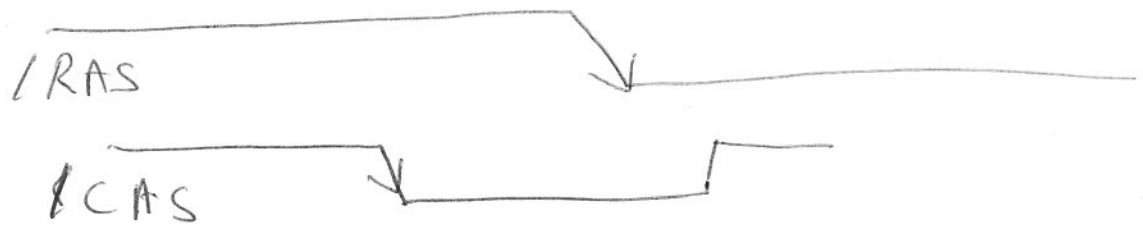
So in Fast Page mode, we cannot keep row active and /RAS = 0 more than 8 μs.

Example Find cost in Time of refresh, assume refresh operation Takes 120 ns.

% cost = $\frac{120\text{ ns}}{8\,\mu s} \approx 1.5\%$

Refresh Timing

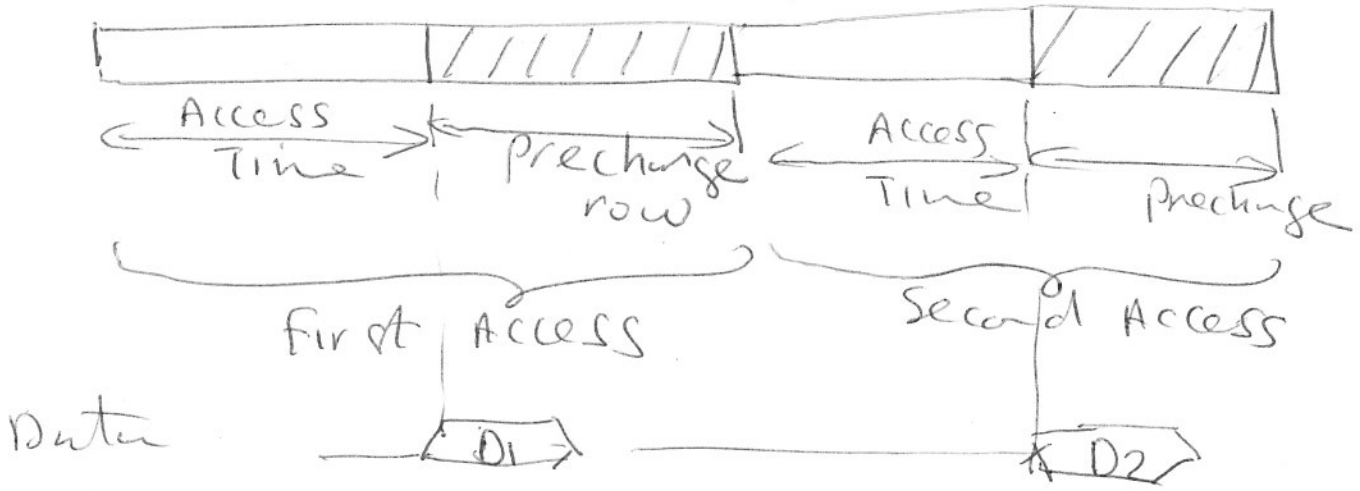
CAS before RAS refresh



This will allow an internal counter to output the row. (0 — 1024).

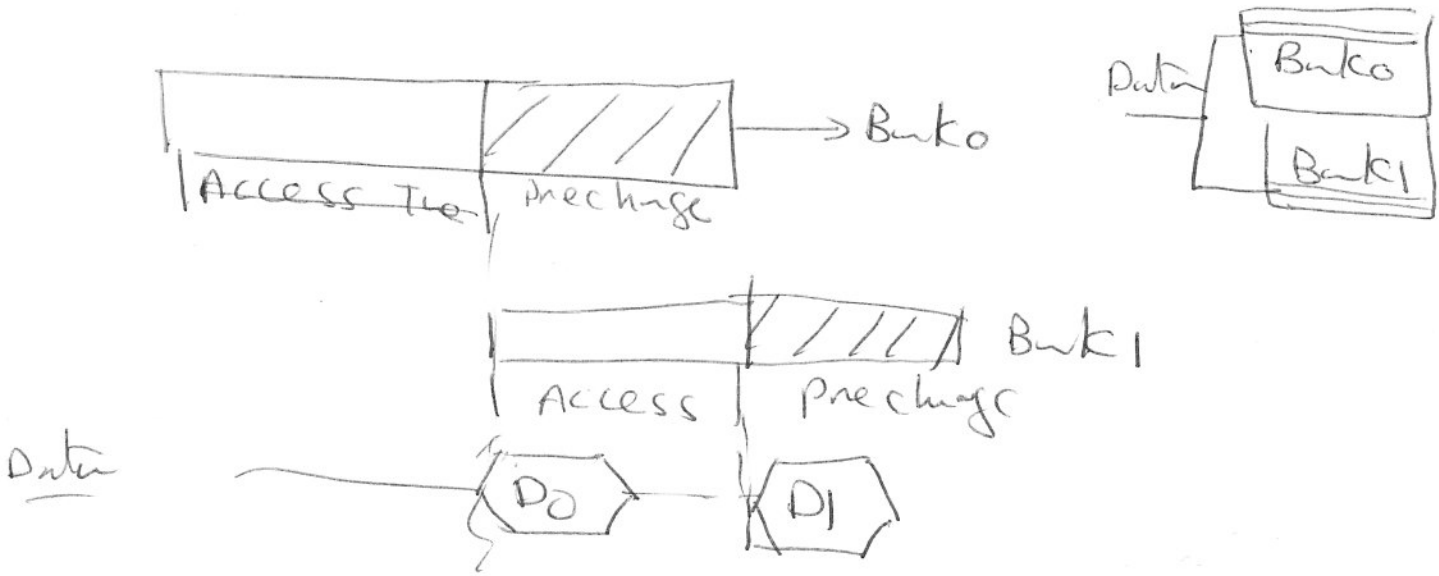
Increasing DRAM Performance
Interleaving

1- No interleaving



2- Interleaving

- Can hide precharge Time
- must use multiple Banks



Mapping Address To Banks

- If next address To Same bank, we cannot use ~~or~~ (hide Precharge) interleaving and must precharge row.
- must use Address mapping to fully utilize interleaving.

1- Low Address interleaving
- Select Banks around Low address bits

Consective Addresses map To different Bnks.

Address	
0	B0
1	B1
2	B2
3	B3
4	B0
5	B1
⋮	⋮

- good for vector processing
- most of Precharge latency could be hidden (for accesses To diff Bnks)

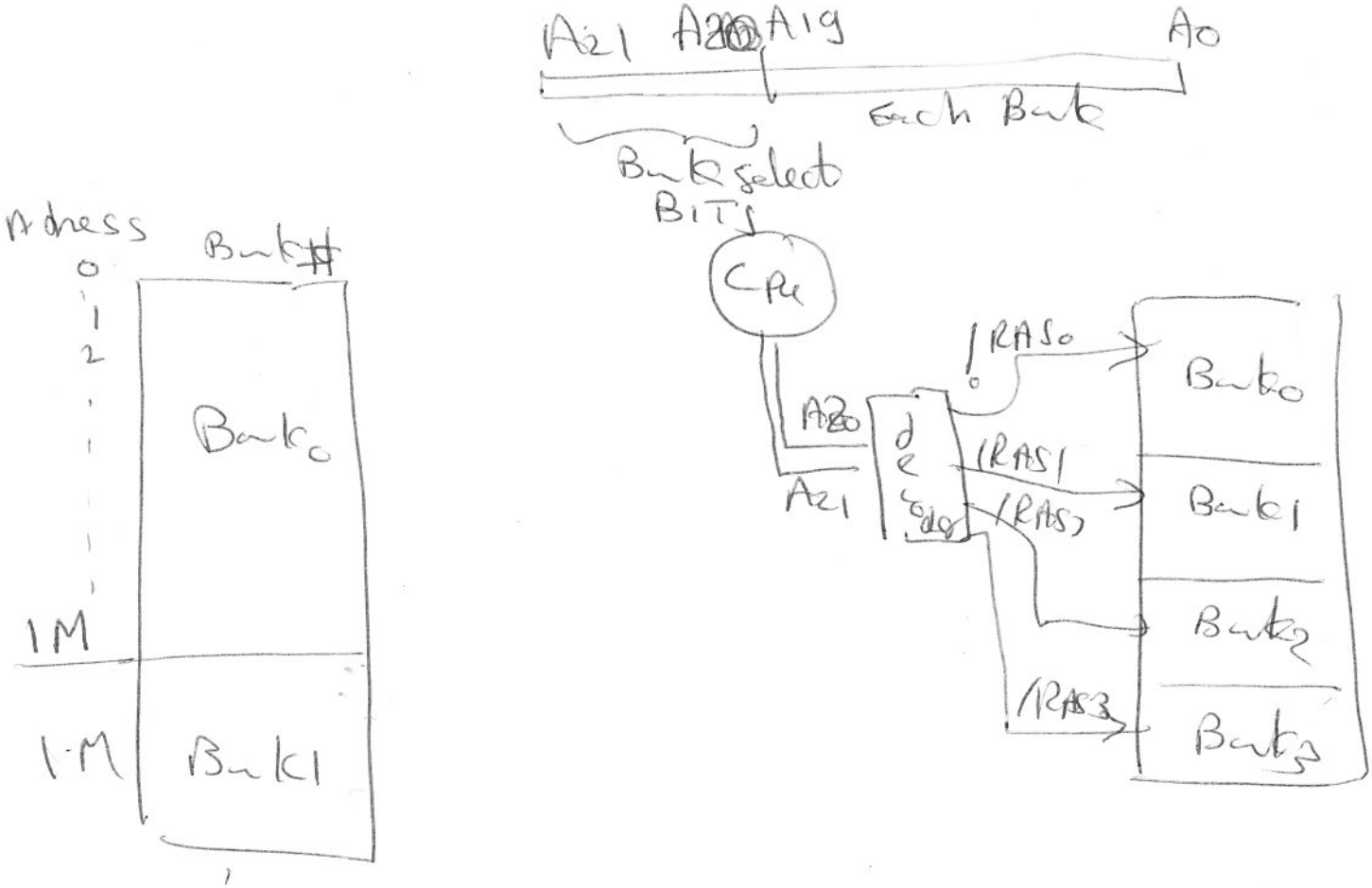
Example assume 4 bnks, loworder address interleaving, find the accessed bnk# for CPU Address = 3A5A2E

A3 A2 A1 A0
1 1 1 0

Accessed bnk = 2

2. Method 2: High order Address Interleaving

High order Address bits select Bank



- Consecutive addresses map to same Bank.
- NOT good for precharge latency hiding.
- okay for using Fast Page Mode (same row, same bank)

Example find Bank, row numbers of

Address 3A5A2E, 3A5A2F

A21 A20 → Bank # = 3

row = 1010 0101 01

Bank # = 3

1010 0101 10

512 + 128 + 64 + 4 + 2

Example: for(i=0; i<1000; i++){
array[i] = 5*i;
}

assume array[0] is at address 500
and DRAM is 1M x 8, uses 4 banks

row access Time = 40ns, colum access Time = 40ns, Precharge Time = 30ns

find total Time if
A- use low order address interleaving
B- use high order address interleaving
(use precharge, fast page modes)

500 → hex / binary
→ Bank #1
→ row #1

for high order interleaving
0 — 1023 → row 0 → 500 random + 501–1023 F.P
1024–2047 → row 1 → 1024 random + 1024–1500 F.P

row 0
row 1

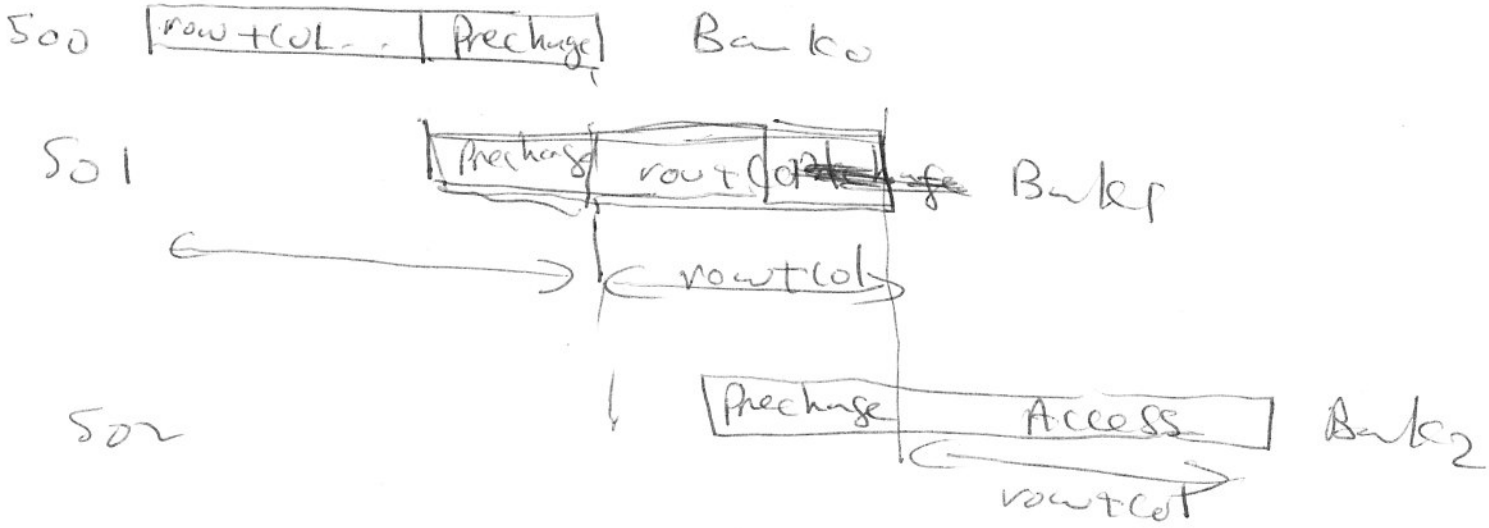
Time: 2 x (40+40+30) + 998 * (40)
random B.P

Low order Address Interleaving

array [0] = 500 ⟶ Bank0 ⟨row+col+Precharge⟩ (random)

[1] = 501 ⟶ Bank1 ⟨row+col⟩

[2] = 502 ⟶ Bank2 ⟨row+col⟩



Total Time = (40+40+30) + 999 * (40+40)

---

high order Address interleaving with precharge

array [0] = 500 ⟶ Bank0, row 0

array [999] = 1499 ⟶ Bank0, row 1

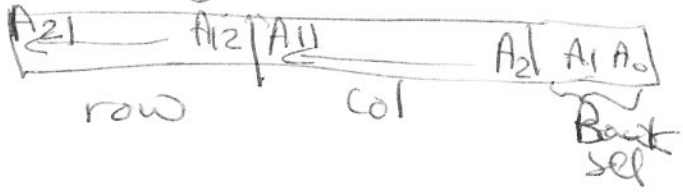
same Bank

Total Time = (40+40+30) X 1000

Performance of low order Address interleaving using DRAM's Fast Page Mode.

- Each bank latches a row
- if access in same row, use FPM

array[0] = 500 → Bank, row, col



500 = 111110100

col = 125, B_0

row = 0

501 = B1, r0

502 = B2, r0

503 = B3, r0

504 = B0, r0 hit

Total Time = 4 x (40 + 40 + 30) + 996 x (10)

random

First Access To bank = miss

random access