TOPICS IN COMPUTING WITH EMERGING TECHNOLOGIES

BROWN

FALL 2020

PROF. IRIS BAHAR

SEPTEMBER 14, 2020

LECTURE 2: MEMORY DESIGN

1

INSTRUCTOR

- Iris Bahar
  - Prof. of Engineering, Prof. of CS
  - Office: CIT449
  - Research interests: energy-efficient computing, computer architecture, robotics, emerging computing technologies
  - Teaching interests: digital design, robotics, emerging technologies, VLSI, low power computing, design

2

ABOUT THIS COURSE

- This course will consider how emerging device technologies will affect our past assumptions about computing from both a hardware and software perspective.
- Class will include a mix of lectures and discussion on assigned reading of recent publications. Students will be responsible for leading and participating in these discussions.
- A final project of your choosing will allow you to delve deeper into a topic discussed in class

3

COURSE FORMAT

- This course is officially being offered in a hybrid format
  - TBD: Percentage in-person vs. remote
  - This semester will be one big experiment, so we all need to be flexible…
- Class time will be primarily discussion based, so it is important that you participate during class (in person or zooming)
- Lecture will be *synchronous only*
- In addition, supplementary online discussion will be required via canvas (in teams or individually).

4

DISCUSSION #0: INTRODUCTION

1. Log on to Canvas and select this course: CSCI2952J
   - Go to Modules → Course Orientation and read through the course Welcome & Overview
   - Set up your Profile and Notification preferences, then click Next at the bottom of the screen to view Assignment #0
2. **Self introduction**: share with the class something about yourself
   - Where is your hometown? What year are you in your studies?
   - What research topics are you working on?
   - Why are you interested in taking this course?
   - What are your hobbies?
   - Add a fun photo or video clip
3. Click **Reply** to post your response to the questions posed above.
4. After a few of your peers have posted their introductions, click **Reply** and respond to 2-3 posts.
5. Please ***complete by Sept. 18*** (the sooner the better)

5

SUBMIT TOPICS OF INTEREST

- Find under Modules→Week #1
  - Click on list of topics
  - Also find under Assignments→Assignment #1
- Think of topics you are most interested in learning about this semester. They may or may not relate to your own research.
- Submit as a text entry with a list of 2-4 topics you would like to cover this semester.
- This will help me plan paper topics for the semester and pair up people with mutual interests.
- Due by Sept 16

6

OVERVIEW OF EMERGING TECHNOLOGIES

Read and comment on 2 survey papers on emerging technologies

- Find under Modules→Week #1
  - Read the following 2 papers:
    - *Computing's Energy Problem (and what we can do about it)*
    - *The era of hyper-scaling in electronics*
  - Click on online discussion
- For the first reading assignment, please complete evaluation alone. Post 1-2 comments that related to the following (or something similar that sparks your interest):
  - What big new idea did you learn?
  - How does this relate to your own research interests?
  - What topics were you most familiar with? *DO NOT FEEL LIKE YOU NEED TO BE FAMILIAR WITH MOST OF THE TOPICS DISCUSSED IN THE PAPERS*
- Post your comments by the end of Monday, Sept. 14.

7

COMING UP FOR WEEK #2

- Many emerging technologies focus on replacements for silicon-based memory design
- Before we jump into research papers, I will spend a week reviewing computing memory hierarchy design
- Recommended textbook:
  - Hennessy, Patterson, *Computer Organization and Design: The Hardware/Software Interface*, Morgan Kaufmann

8

COMPUTER MEMORY DESIGN

EXPLOITING MEMORY HIERARCHY

9

MEMORY DEFINITIONS

- Function – functionality, nature of the storage mechanism
  - static and dynamic; volatile and nonvolatile
- Access pattern – random, serial, content addressable
- Input-output architecture – number of data input and output ports (multi-ported memories)
- Application – embedded, secondary, tertiary

10

MEMORY DEFINITIONS

RWM		NVRWM	ROM
Random Access	Non-Random Access	EPROM	Mask-programmed
SRAM (cache, register file) DRAM (main memory)	FIFO, LIFO Shift Register CAM	EEPROM FLASH	Electrically-programmed (PROM)

- Read-Only Memories (ROM)
  - Truly read-only
    - Written in the factory, and never written after installation
  - Mostly read and rarely written
    - Much faster to read than write
  - Non-volatile, e.g., flash memory
- Random Access Memory (RAM)
  - Read and write any location at similar speeds
  - Volatile: loses contents when powered off
  - SRAM, DRAM

11

PRINCIPLE OF LOCALITY

- Programs access a small proportion of their address space at any time
- Temporal locality
  - Items accessed recently are likely to be accessed again soon
  - e.g., instructions in a loop, induction variables
- Spatial locality
  - Items near those accessed recently are likely to be accessed soon
  - E.g., sequential instruction access, array data

12

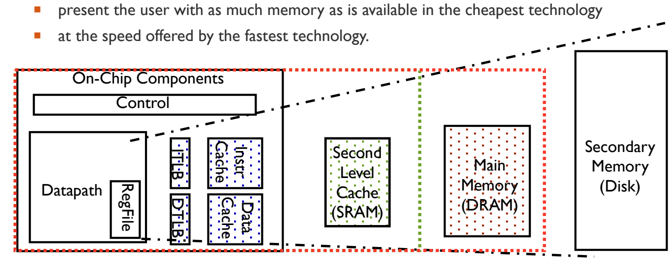
TAKING ADVANTAGE OF LOCALITY

- Memory hierarchy
- Store everything on disk (non-volatile memory)
- Copy recently accessed (and nearby) items from disk to smaller DRAM memory
  - Main memory (generally volatile)
- Copy more recently accessed (and nearby) items from DRAM to smaller SRAM memory
  - Cache memory attached to CPU

13

A TYPICAL MEMORY HIERARCHY

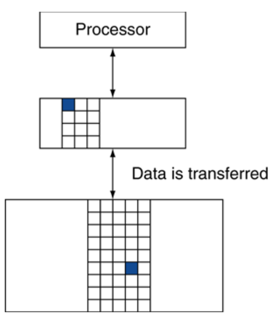
- By taking advantage of the principle of locality, we can:
  - present the user with as much memory as is available in the cheapest technology
  - at the speed offered by the fastest technology.



Speed (ns):	.1's	1's	10's	100's	1,000's
Size (bytes):	100's	KB's	10KB's	MB's	TB's
Cost:	highest				lowest

14

MEMORY HIERARCHY LEVELS



- Block (aka line): unit of copying
  - May be multiple words
- If accessed data is present in upper level
  - Hit: access satisfied by upper level
    - Hit ratio: hits/accesses
- If accessed data is absent
  - Miss: block copied from lower level
    - Time taken: miss penalty
    - Miss ratio: misses/accesses = 1 – hit ratio
  - Then accessed data supplied from upper level

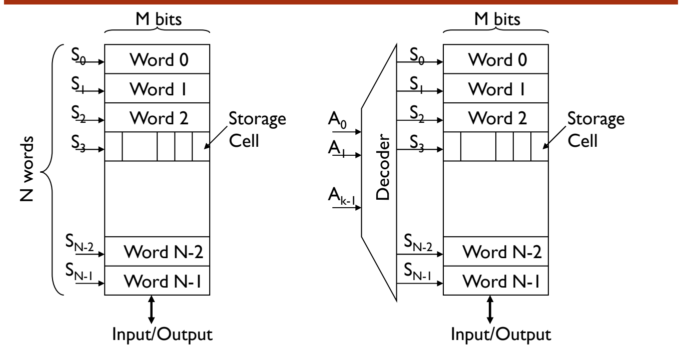
15

READ/WRITE MEMORIES (RAMS)

- Static – SRAM
  - data is stored as long as voltage supply is enabled
  - large cells (6 FETs/cell) → fewer bits/chip
  - fast → used where speed is important (e.g., caches)
- Dynamic – DRAM
  - periodic refresh required (every 1 to 4 ms)
  - small cells (1 to 3 FETs/cell) → more bits/chip
  - slower → used for main memories

17

1-DIMENSIONAL MEMORY ARCHITECTURE

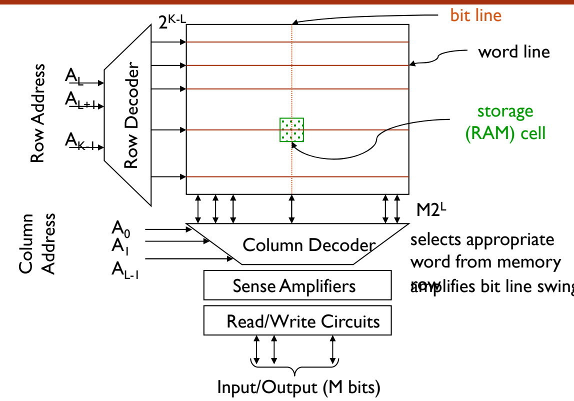


N words → N select signals

Decoder reduces # of inputs
K = $\log_2$ N

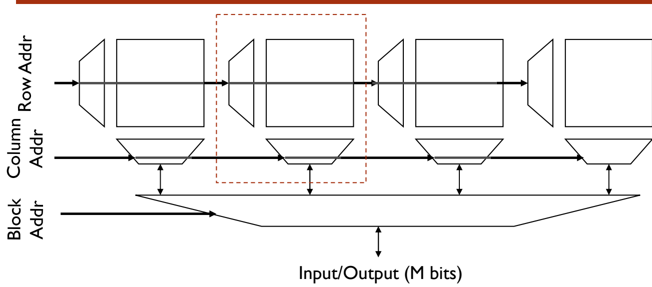
18

2D MEMORY ARCHITECTURE



19

3D (I.E., BANKED) MEMORY ARCHITECTURE



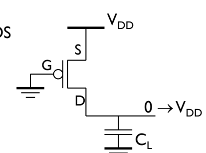
Advantages:
1. Shorter word and bit lines so faster access
2. Block addr activates only 1 block saving power

20

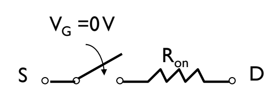
BUILDING A MEMORY CELL

- Everything starts with transistors (i.e., switches)

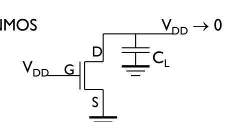
PMOS



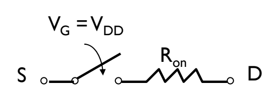
If G=0V, switch closes and current flows from S→D, charging C_L to V_{DD}



NMOS

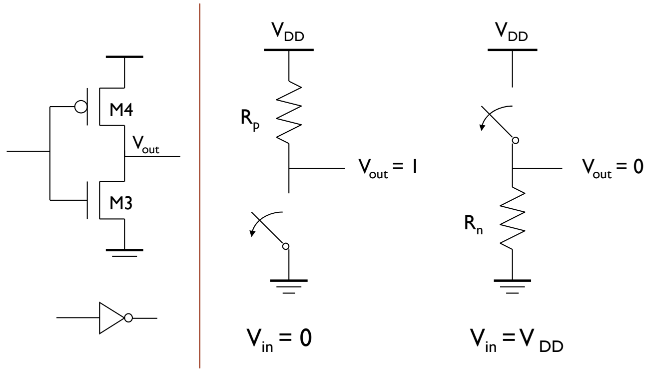


If G= V_{DD}, switch closes and current flows from S→D, discharging C_L to 0V



21

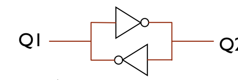
CMOS INVERTER: STEADY STATE RESPONSE



22

CROSS-COUPLED INVERTERS

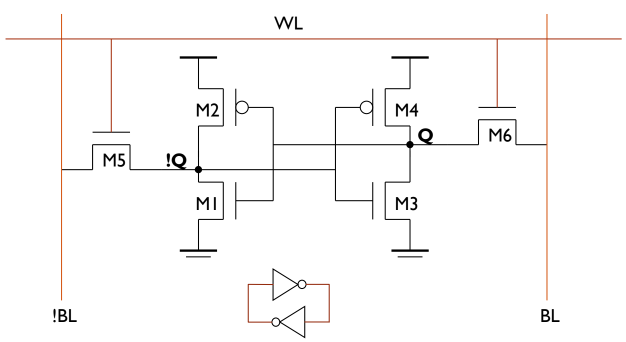
- Build a memory cell out of inverters



- Bi-stable element
  - If Q1=V_{DD}, Q2=0V
  - If Q2=V_{DD}, Q1=0V
  - Q1 and Q2 will hold their values indefinitely (as long as power is supplied)
- *So what if you want to change the values stored in the cell?*

23

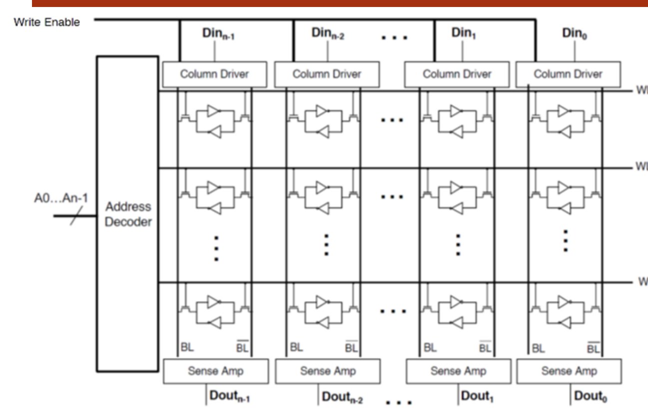
6-TRANSISTOR SRAM CELL



- Switches M5/M6 allow you to probe and modify the cell value

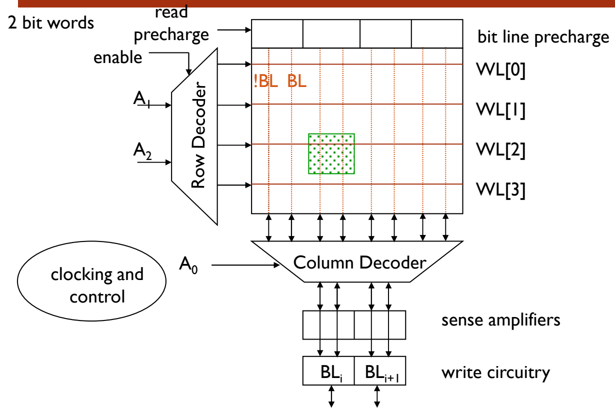
24

SRAM ARCHITECTURE



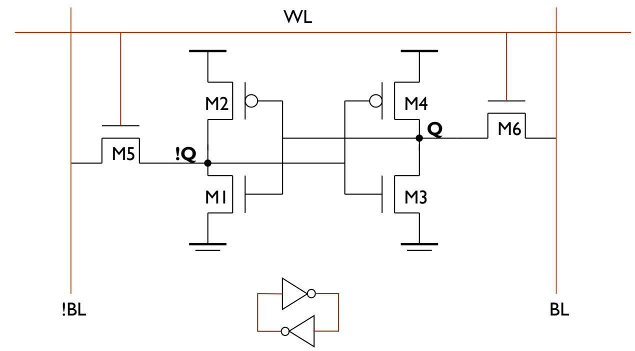
25

4X4 SRAM MEMORY



26

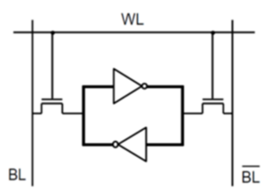
6-TRANSISTOR SRAM CELL



27

SRAM WRITE

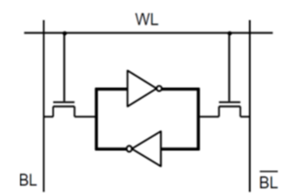
- Drive on bit line high, the other low (depending on the desired write value)
- Then enable the word line (WL)
- Bit lines overpower cell with new value



28

SRAM READ

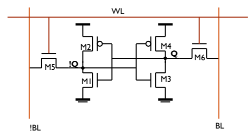
- Precharge both bit lines high
- Then enable word line (WL)
- One of the two bit lines will be pulled down by the cell
  - Change detected by sensor amplifier



How does the SRAM distinguish between reading or writing a cell?
How do you make sure you don't accidentally overwrite a cell value when you are trying to read?
How do you make sure successful overwrite the cell value when you are trying to write?

29

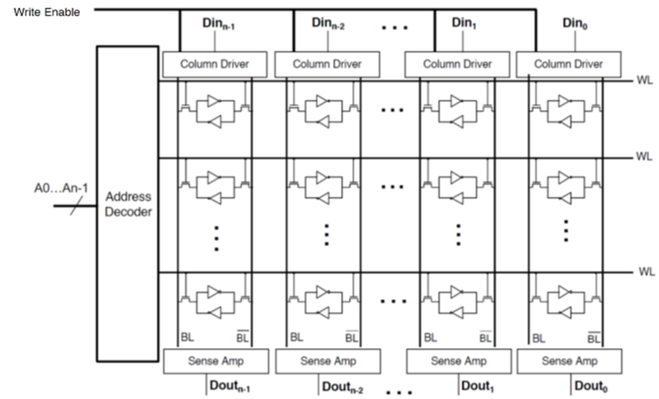
SRAM: SIZING IS EVERYTHING



- SRAM is very stable because of reinforcement of cross-coupled inverters
- Size (width) of transistor is directly related to its drive strength
- Read: Make stored value stronger than "reading" value
  - Make M1(M3) wider than M5(M6)
- Write: Make writing a "0" stronger than writing a "1"
  - Make PMOS devices (M2/M4) weaker than NMOS (M5/M6)

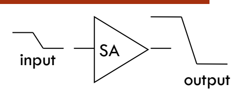
30

SRAM ARCHITECTURE



31

SENSE AMPLIFIERS



- Amplification: resolves data with small bit line swings
- Delay reduction: compensates for the limited drive capability of the memory cell

$$t_p = (C * \Delta V) / I_{av}$$

small

large

make Δ V as small as possible

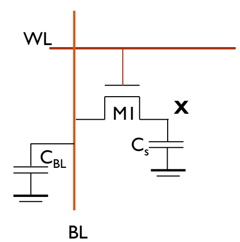
- Power reduction: eliminates a large part of the power dissipation

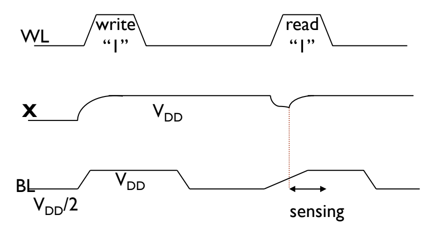
$$P = ½ C * V_{DD} * \Delta V * f$$

make Δ V as small as possible

32

1-TRANSISTOR DRAM CELL

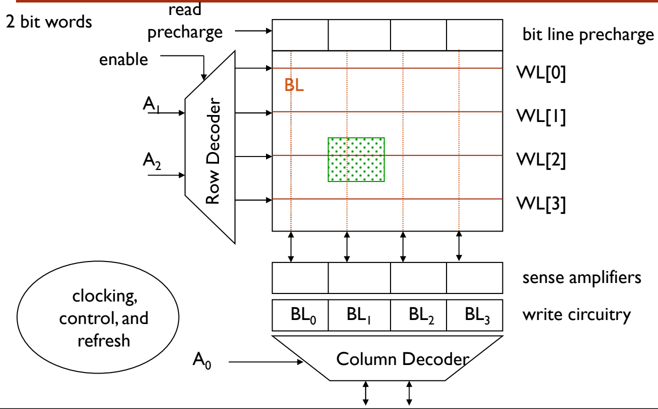




- Write: C_s is charged (or discharged) by asserting WL and BL
- Read: Charge redistribution occurs between C_{BL} and C_s. Sense amp is required to boost voltage to full rail (Vdd or Gnd).
- Read is destructive, so must refresh after read
- After some time (< 1 sec) charge will completely leak from C_s so periodic refresh needed even if nothing is read

33

4X4 DRAM MEMORY



34