BROWN

TOPICS IN COMPUTING WITH EMERGING TECHNOLOGIES

FALL 2020

PROF. IRIS BAHAR

SEPTEMBER 16, 2020

LECTURE 3: MEMORY DESIGN

1

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
- Thanks to those who have already submitted.
- Post your comments ASAP.

2

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

3

LECTURE SLIDES FOR WEEK #2

- Lecture slides are posted on Canvas (see Modules: week #2) and on the CS course webpages (cs.brown.edu/courses/csci2952j)
- Recommended textbooks:
  - Hennessy, Patterson, *Computer Organization and Design: The Hardware/Software Interface*, Morgan Kaufmann
  - *Neil H. E. Weste and David Harris, CMOS VLSI Design: A Circuit and Systems Perspective, 4th Edition, Addison Wesley Publishers, 2011*

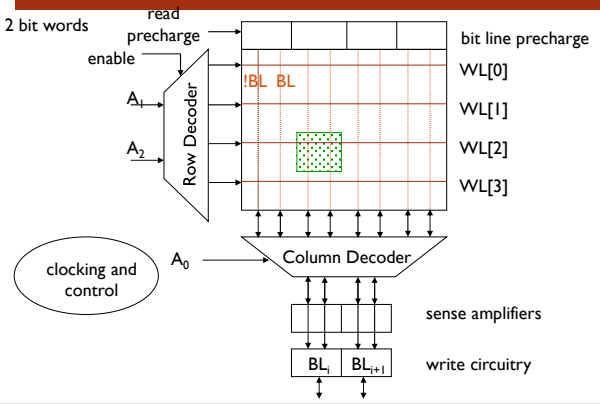
4

COMING UP FOR WEEK #3

- We will continue with our overview of memory design and conventional transistor design
- Paper discussion delayed by 1 week
- Papers for 4th week of class:
  - *Emerging NVM: A Survey on Architectural Integration and Research Challenges*
  - *Memory that never forgets: emerging nonvolatile memory and the implication for architecture design*
- I will post papers some time next week
  - I will assign teams for reviewing the papers
  - Expect different team assignments weekly
- I will also assign discussion leaders for the week
  - If you want to volunteer, let me know
  - We will rotate discussion leaders throughout the semester. Expect to lead 2-3 times
- Starting the 6th or 7th week of class, students will be able to choose papers to review

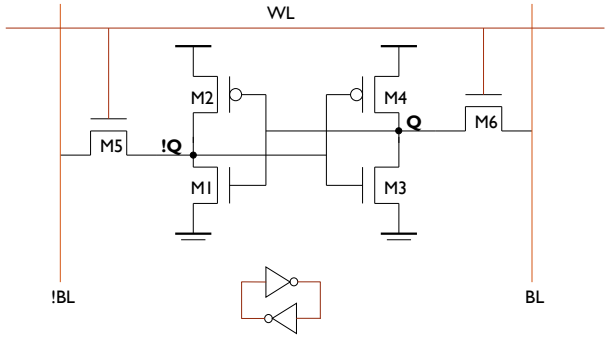
5

4X4 SRAM MEMORY



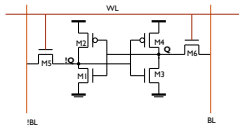
6

6-TRANSISTOR SRAM CELL



7

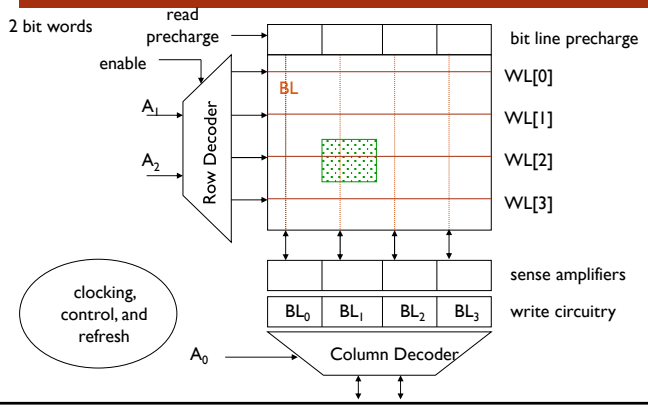
SRAM: SIZING IS EVERYTHING



- Key takeaways:
  - SRAM is very stable because of reinforcement of cross-coupled inverters
  - Retains value as long as connected to power
  - Correct sizing of transistors prevent read disturb and allow overwrite on write
  - Dual rail adds extra noise resilience
  - Use of sense amp to speed up transition of bit lines (!BL, BL) to "full rail" values

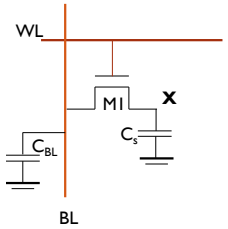
8

4X4 DRAM MEMORY



9

1-TRANSISTOR DRAM CELL



- Key takeaways:
  - Cell value is stored in capacitor C_s
  - No regenerative property: noise disturbance on cell can cause value to be lost.
  - Single rail, also making it more susceptible to noise disturbance
  - Read: lose value stored by reading cell, so write/refresh required
  - Read: Sense amp is required to reach full rail on BL
  - Write: single phase operation
  - Charge will leak from C_s after < 1 sec so periodically need to refresh by reading all rows and writing them back.

10

ADVANCED DRAM ORGANIZATION

- Bits in a DRAM are organized as a rectangular array
  - DRAM accesses an entire row
  - Burst mode: supply successive words from a row with reduced latency
- Double data rate (DDR) DRAM
  - Transfer on rising and falling clock edges
- Quad data rate (QDR) DRAM
  - Separate DDR inputs and outputs

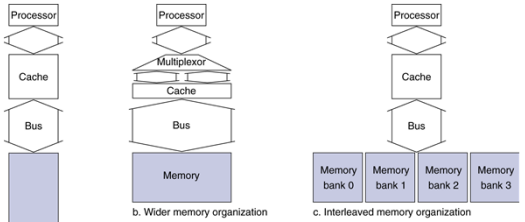
11

DRAM PERFORMANCE FACTORS

- Row buffer
  - Allows several words to be read and refreshed in parallel
- Synchronous DRAM
  - Allows for consecutive accesses in bursts without needing to send each address
  - Improves bandwidth
- DRAM banking
  - Allows simultaneous access to multiple DRAMs
  - Improves bandwidth

12

INCREASING MEMORY BANDWIDTH

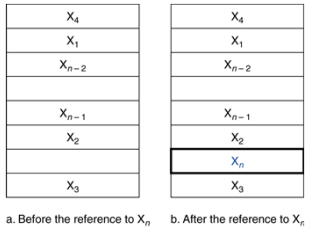


- 4-word wide memory
  - Miss penalty = 1 + 15 + 1 = 17 bus cycles
  - Bandwidth = 16 bytes / 17 cycles = 0.94 B/cycle
- 4-bank interleaved memory
  - Miss penalty = 1 + 15 + 4×1 = 20 bus cycles
  - Bandwidth = 16 bytes / 20 cycles = 0.8 B/cycle

13

CACHE MEMORY

- Cache memory
  - The level of the memory hierarchy closest to the CPU
- Given accesses $X_1, \ldots, X_{n-1}, X_n$



- *How do we know if the data is present?*
- *Where do we look?*

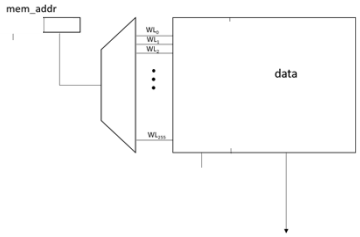
14

TAGS AND VALID BITS

- How do we know which particular block is stored in a cache location?
  - Store block address as well as the data
  - Actually, only need the high-order bits
  - Called the tag
- What if there is no data in a location?
  - Valid bit: 1 = present, 0 = not present
  - Initially 0

15

HOW DO YOU DESIGN A DATA CACHE FROM SRAM?

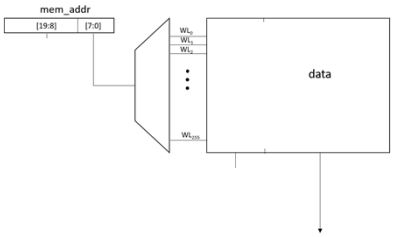


Say you want to store 8kbits (1kB) of data in your cache using SRAM

- *How many 32-bit words can you store in your SRAM?*
- *How many address bits do you need to access all words uniquely?*
- *If you have a 20-bit memory address, how many bit remain?*

16

WHAT IS THIS TAG FIELD?



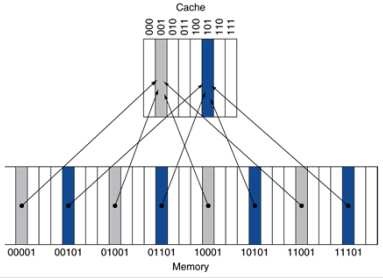
- *We have 8 bits of the address used to access the data*
- *How do you distinguish between address 1011100011 and address 0111100011 ?*
- *We need to store a piece of the memory address along with the data. This is the TAG field*

- *We have 20-8 = 12 bits remaining of the memory address that become the TAG field*
- *Read the tag along with the data to see if this the data you desire.*

17

DIRECT MAPPED CACHE

- Location determined by address
- Direct mapped: only one choice
  - (Block address) modulo (#Blocks in cache)



- *#Blocks is a power of 2*
- *Use low-order address bits*

18

CACHE EXAMPLE

- 8-blocks, 1 word/block, direct mapped
- Initial state

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

19

CACHE EXAMPLE

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Miss	110

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	**Y**	**10**	**Mem[10110]**
111	N		

20

CACHE EXAMPLE

Word addr	Binary addr	Hit/miss	Cache block
26	11 010	Miss	010

Index	V	Tag	Data
000	N		
001	N		
010	**Y**	**11**	**Mem[11010]**
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

21

CACHE EXAMPLE

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Hit	110
26	11 010	Hit	010

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

22

CACHE EXAMPLE

Word addr	Binary addr	Hit/miss	Cache block
16	10 000	Miss	000
3	00 011	Miss	011
16	10 000	Hit	000

Index	V	Tag	Data
000	**Y**	**10**	**Mem[10000]**
001	N		
010	Y	11	Mem[11010]
011	**Y**	**00**	**Mem[00011]**
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

23

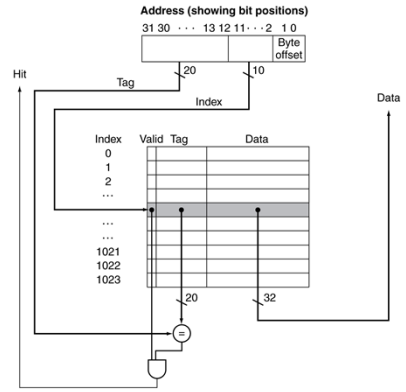
CACHE EXAMPLE

Word addr	Binary addr	Hit/miss	Cache block
18	10 010	Miss	010

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	**Y**	**10**	**Mem[10010]**
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

24

ADDRESS SUBDIVISION



- 32 bits = 4 bytes (2 bits for byte offset)
- 1024 unique blocks
- What is the data capacity of this cache?
  - $2^{10} * 2^2 = 2^{12} = 4kB$
- How many bits total are required?
  - Tag: $2^{10} * 20 = 20,480$
  - Total = 4kB + 20,480
  - Overhead: ~62%

25

CACHE MISSES

- On cache hit, CPU proceeds normally
- On cache miss
  - Stall the CPU pipeline (wait for data for CPU to proceed)
  - Fetch block from next level of hierarchy
  - Instruction cache miss
    - Restart instruction fetch
  - Data cache miss
    - Complete data access

27

BLOCK SIZE CONSIDERATIONS

- Larger blocks should reduce miss rate
  - Due to spatial locality
- But in a fixed-sized cache
  - Larger blocks → fewer unique blocks
    - More competition → increased miss rate
  - Larger blocks → pollution (if spatial locality is weak)
- Larger miss penalty
  - Takes longer to fill block with new data
  - Can override benefit of reduced miss rate
  - Early restart and critical-word-first can help

28

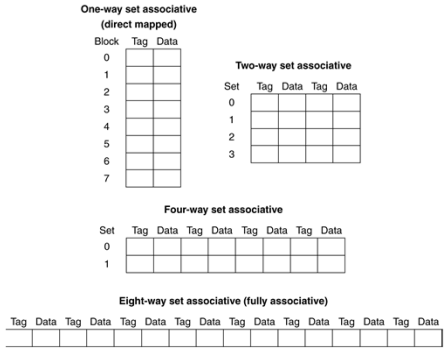
ASSOCIATIVE CACHES

- Fully associative
  - Allow a given block to go in any cache entry
  - Requires all entries to be searched at once
  - Comparator per entry (expensive)
- *n*-way set associative
  - Each set contains *n* entries
  - Block number determines which set
    - (Block number) modulo (#Sets in cache)
  - Search all entries in a given set at once
  - *n* comparators (less expensive)

34

SPECTRUM OF ASSOCIATIVITY

- For a cache with 8 entries



36

ASSOCIATIVITY EXAMPLE

- Compare 4-block caches
  - Direct mapped, 2-way set associative, fully associative
  - Block access sequence: 0, 8, 0, 6, 8
- Direct mapped

Block address	Cache index	Hit/miss	Cache content after access			
			0	1	2	3
0	0	miss	Mem[0]			
8	0	miss	Mem[8]			
0	0	miss	Mem[0]			
6	2	miss	Mem[0]		Mem[6]	
8	0	miss	Mem[8]		Mem[6]	

- Address 8, 0 conflict in the cache (data thrashing)

37

ASSOCIATIVITY EXAMPLE

- 2-way set associative

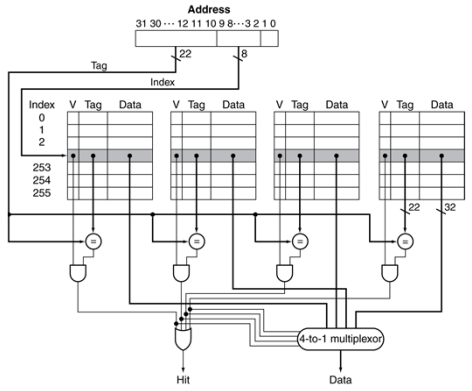
Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0	miss	Mem[0]			
8	0	miss	Mem[0]	Mem[8]		
0	0	hit	Mem[0]	Mem[8]		
6	0	miss	Mem[0]	Mem[6]		
8	0	miss	Mem[8]	Mem[6]		

- Fully associative

Block address		Hit/miss	Cache content after access			
0		miss	Mem[0]			
8		miss	Mem[0]	Mem[8]		
0		hit	Mem[0]	Mem[8]		
6		miss	Mem[0]	Mem[8]	Mem[6]	
8		hit	Mem[0]	Mem[8]	Mem[6]	

38

SET ASSOCIATIVE CACHE ORGANIZATION



40

MULTILEVEL CACHES

- Primary cache attached to CPU
  - Small, but fast
- Level-2 cache services misses from primary cache
  - Larger, slower, but still faster than main memory
- Main memory services L-2 cache misses
- Some high-end systems include L-3 cache

42

MULTILEVEL CACHE EXAMPLE

- Given
  - CPU base CPI = 1, clock rate = 4GHz → 0.25ns per clock cycle
  - Miss rate/instruction = 2%
  - Main memory access time = 100ns
- With just primary cache
  - Miss penalty = 100ns/0.25ns = 400 cycles
  - Effective CPI = 1 + 0.02 × 400 = 9

43

EXAMPLE (CONT.)

- Now add L-2 cache
  - Access time = 5ns
  - Global miss rate to main memory = 0.5%
- Primary miss with L-2 hit
  - Penalty = 5ns/0.25ns = 20 cycles
- Primary miss with L-2 miss
  - Extra penalty = 400 cycles
- CPI = 1 + 0.02 × 20 + 0.005 × 400 = 3.4
- Performance ratio = 9/3.4 = 2.6

44

MAIN MEMORY SUPPORTING CACHES

- Use DRAMs for main memory
  - Fixed width (e.g., 1 word)
  - Connected by fixed-width clocked bus
    - Bus clock is typically slower than CPU clock
- Example cache block read
  - 1 bus cycle for address transfer
  - 15 bus cycles per DRAM access
  - 1 bus cycle per data transfer
- For 4-word block, 1-word-wide DRAM
  - Miss penalty = 1 + 4×15 + 4×1 = 65 bus cycles
  - Bandwidth = 16 bytes / 65 cycles = 0.25 B/cycle

46

MEASURING CACHE PERFORMANCE

- Components of CPU time
  - Program execution cycles
    - Includes cache hit time
  - Memory stall cycles
    - Mainly from cache misses
- With simplifying assumptions:

$$\text{Memory stall cycles} = \frac{\text{Memory accesses}}{\text{Program}} \times \text{Miss rate} \times \text{Miss penalty}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Misses}}{\text{Instruction}} \times \text{Miss penalty}$$

47

CACHE PERFORMANCE EXAMPLE

- Given
  - I-cache miss rate = 2%
  - D-cache miss rate = 4%
  - Miss penalty = 100 cycles
  - Base CPI (ideal cache) = 2
  - Load & stores are 36% of instructions
- Miss cycles per instruction
  - I-cache: 0.02 × 100 = 2
  - D-cache: 0.36 × 0.04 × 100 = 1.44
- Actual CPI = 2 + 2 + 1.44 = 5.44
  - Ideal CPU is 5.44/2 =2.72 times faster

48

AVERAGE ACCESS TIME

- Hit time is also important for performance
- Average memory access time (AMAT)
  - AMAT = Hit time + Miss rate × Miss penalty
- Example
  - CPU with 1ns clock, hit time = 1 cycle, miss penalty = 20 cycles, I-cache miss rate = 5%
  - AMAT = 1 + 0.05 × 20 = 2ns
    - 2 cycles per instruction

49

PERFORMANCE SUMMARY

- When CPU performance increased
  - Miss penalty becomes more significant
- Decreasing base CPI
  - Greater proportion of time spent on memory stalls
- Increasing clock rate
  - Memory stalls account for more CPU cycles
- Can't neglect cache behavior when evaluating system performance

50

INTERACTIONS WITH ADVANCED CPUS

- Out-of-order CPUs can execute instructions during cache miss
  - Pending store stays in load/store unit
  - Dependent instructions wait in reservation stations
    - Independent instructions continue
- Effect of miss depends on program data flow
  - Much harder to analyse
  - Use system simulation

51