EE671: VLSI Design
Spring 2024/25

Laxmeesha Somappa

Department of Electrical Engineering

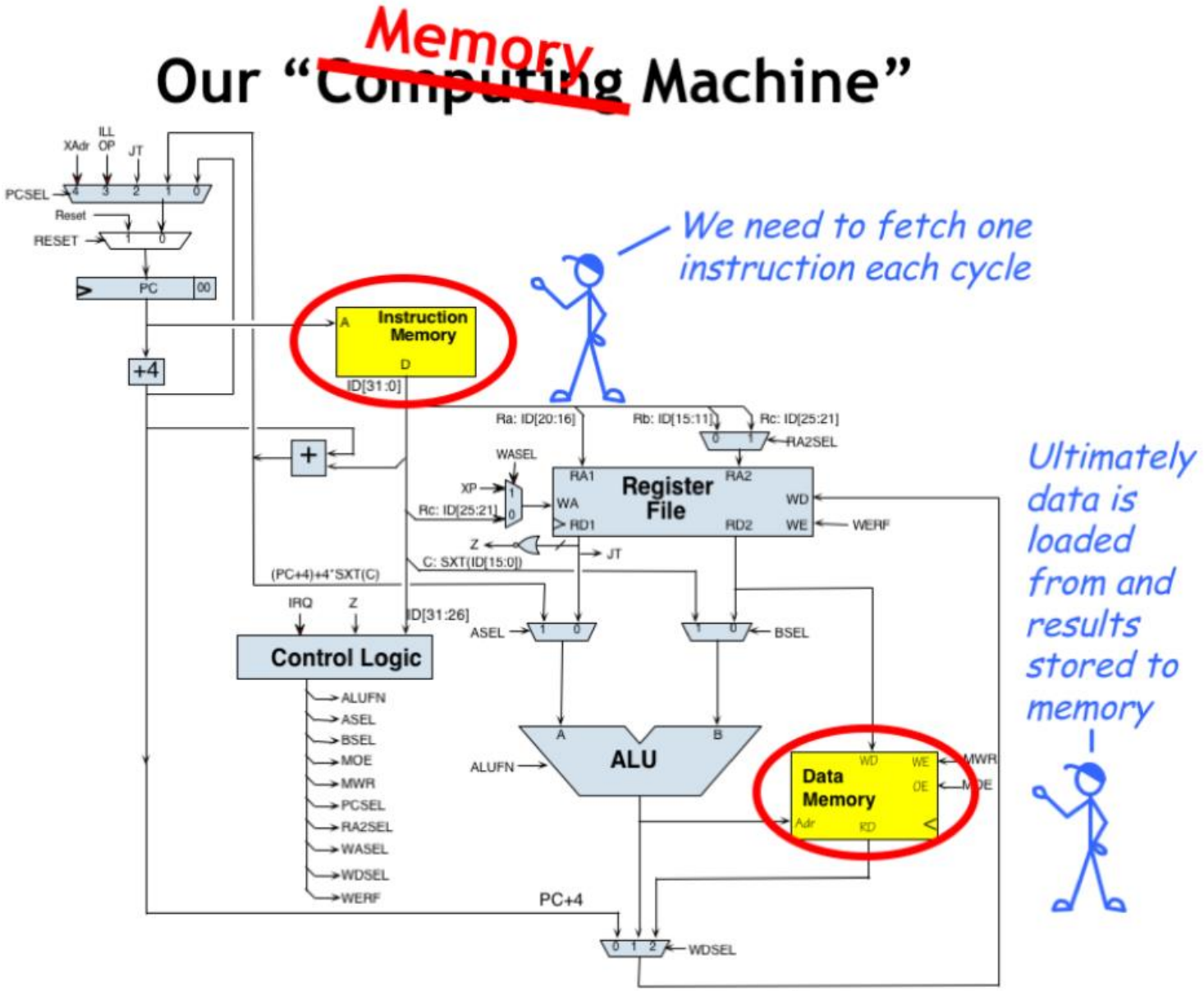
IIT Bombay

laxmeesha@ee.iitb.ac.in

LECTURE – 31
MEMORY – HIERARCHY & IMPLEMENTATION

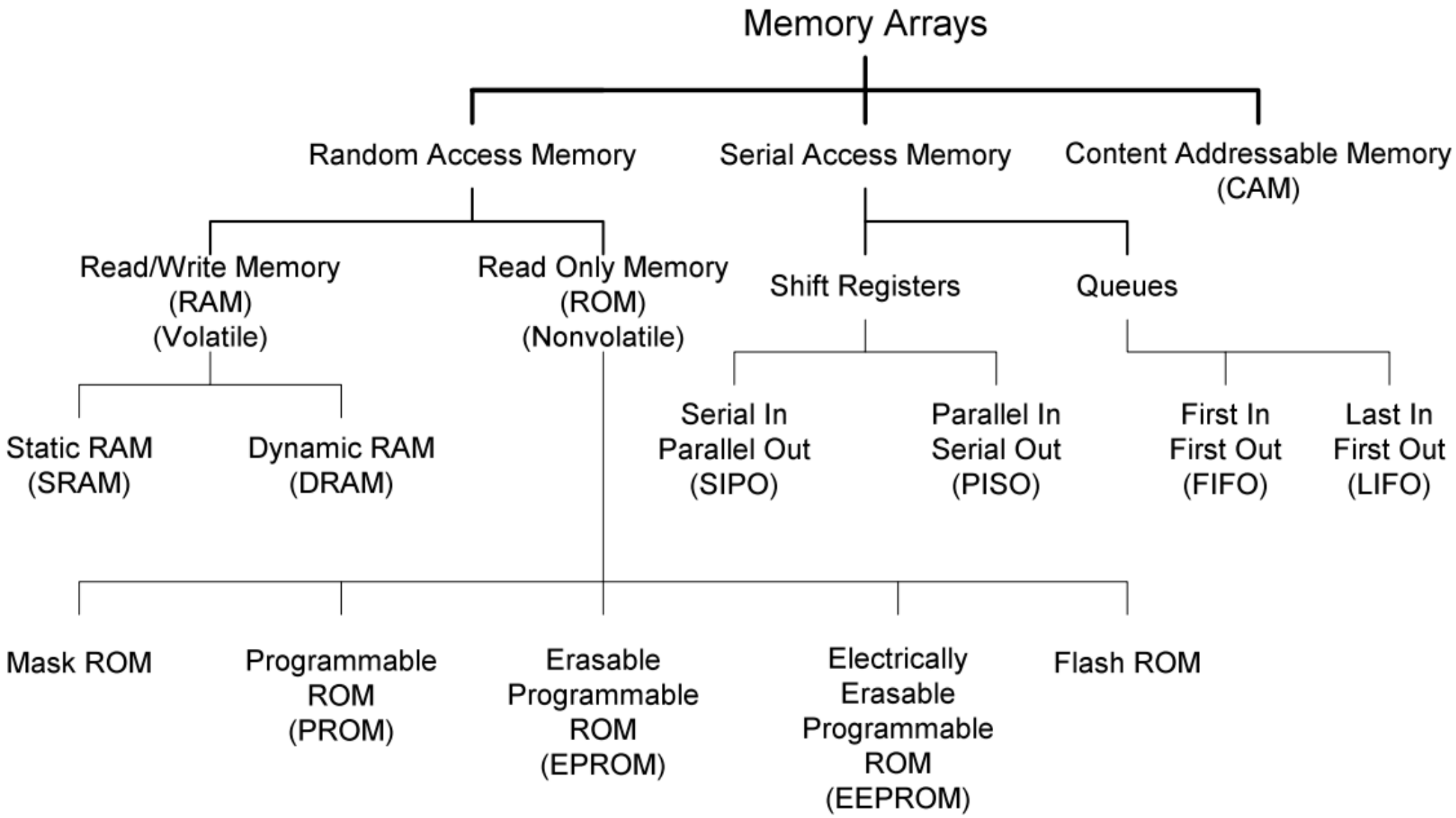
laxmeesha@ee.iitb.ac.in

A GENERAL PROCESSOR OVERVIEW

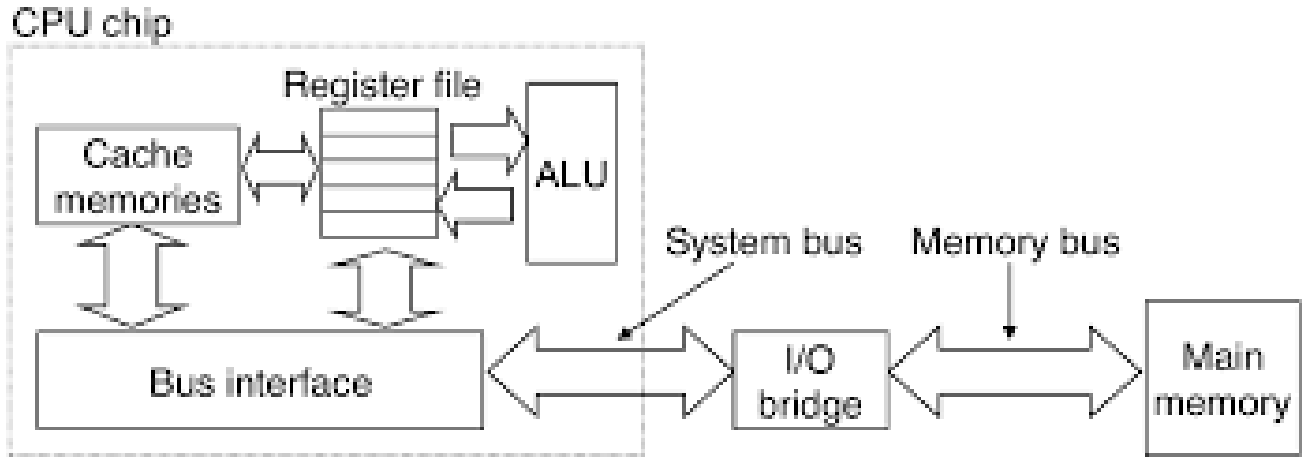


laxmeesha@ee.iitb.ac.in
Image courtesy: MIT-OCW

Memory Array Options



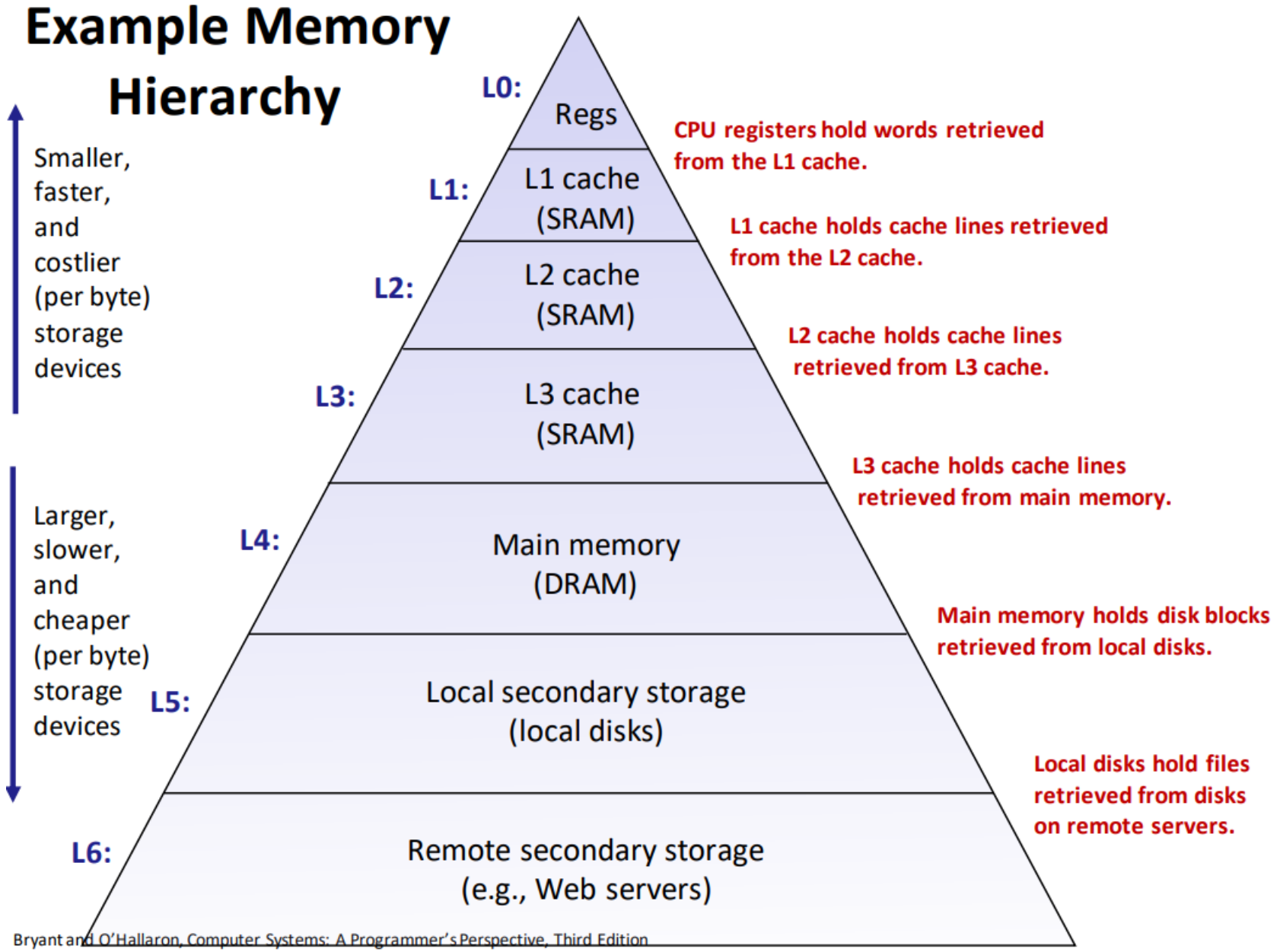
Memory Hierarchy in Processors



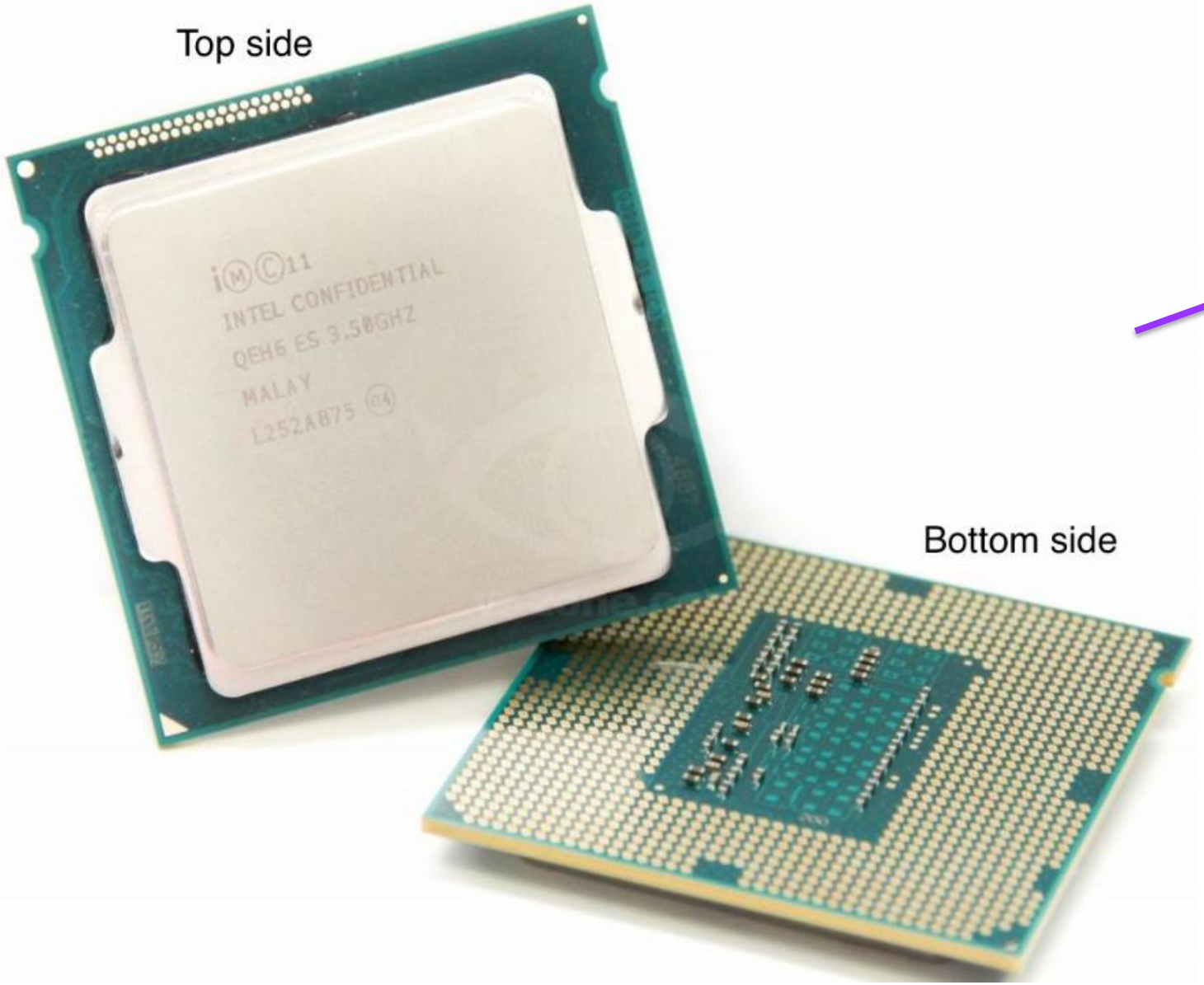
	Capacity	Latency	Cost/GB	
Register	1000s of bits	20 ps	$$$$	Processor Datapath
SRAM	~10 KB-10 MB	1-10 ns	~$1000	Memory Hierarchy
DRAM	~10 GB	80 ns	~$10	
Flash*	~100 GB	100 us	~$1	I/O subsystem
Hard disk*	~1 TB	10 ms	~$0.10	

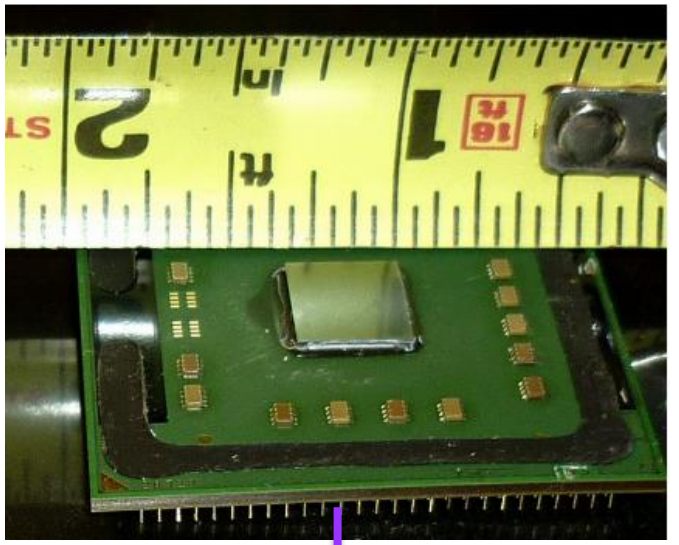
* non-volatile (retains contents when powered off)

Memory Hierarchy Full Picture



PROCESSOR PACKAGED





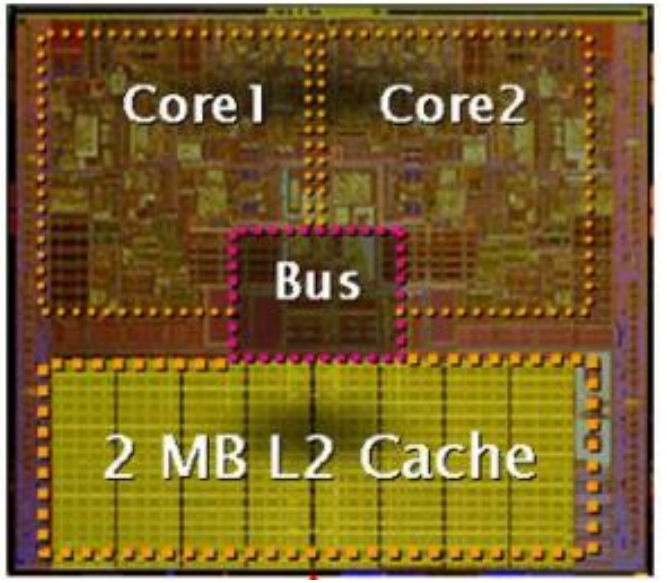
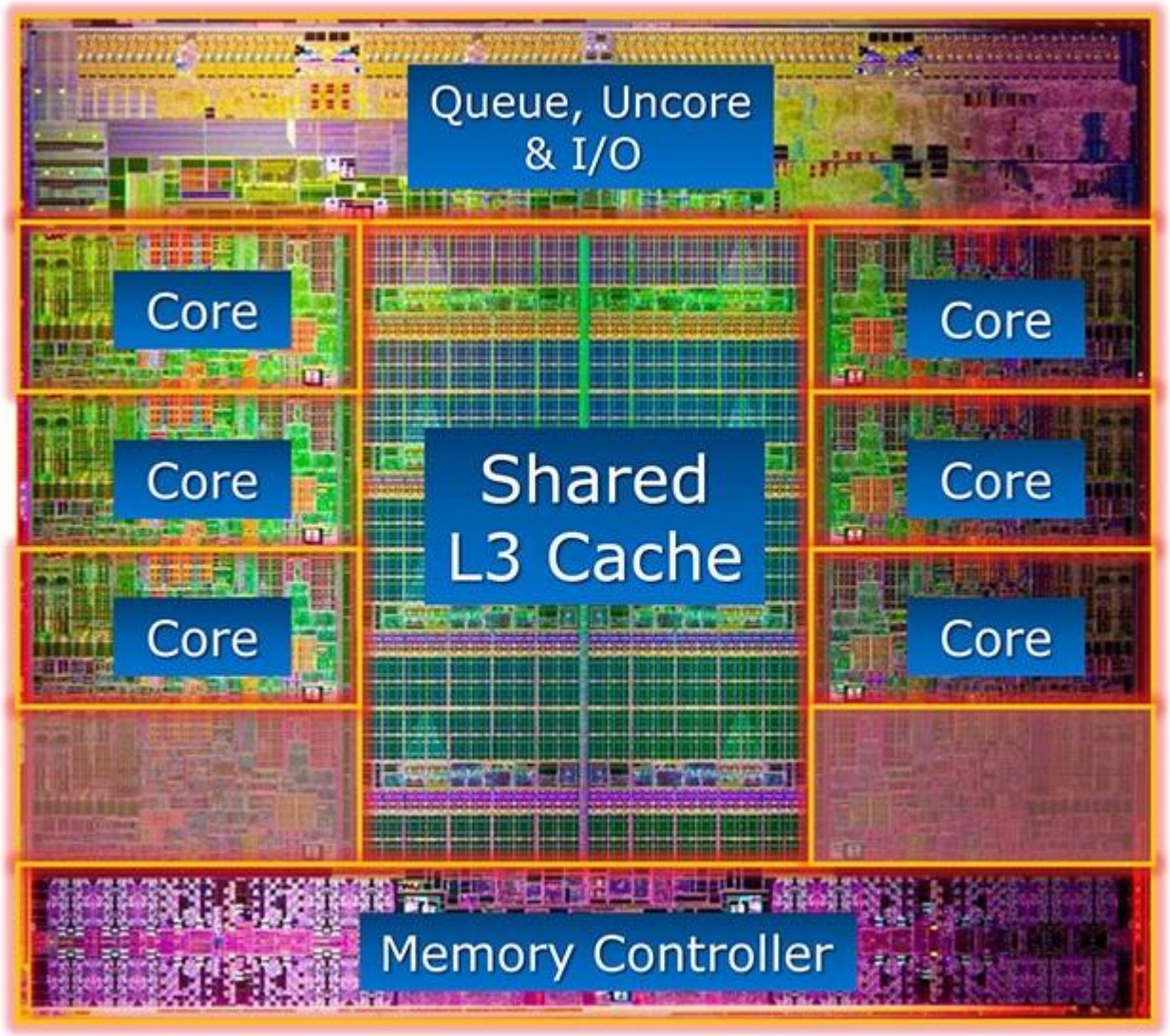


Image courtesy: Intel

Processor Cache Area

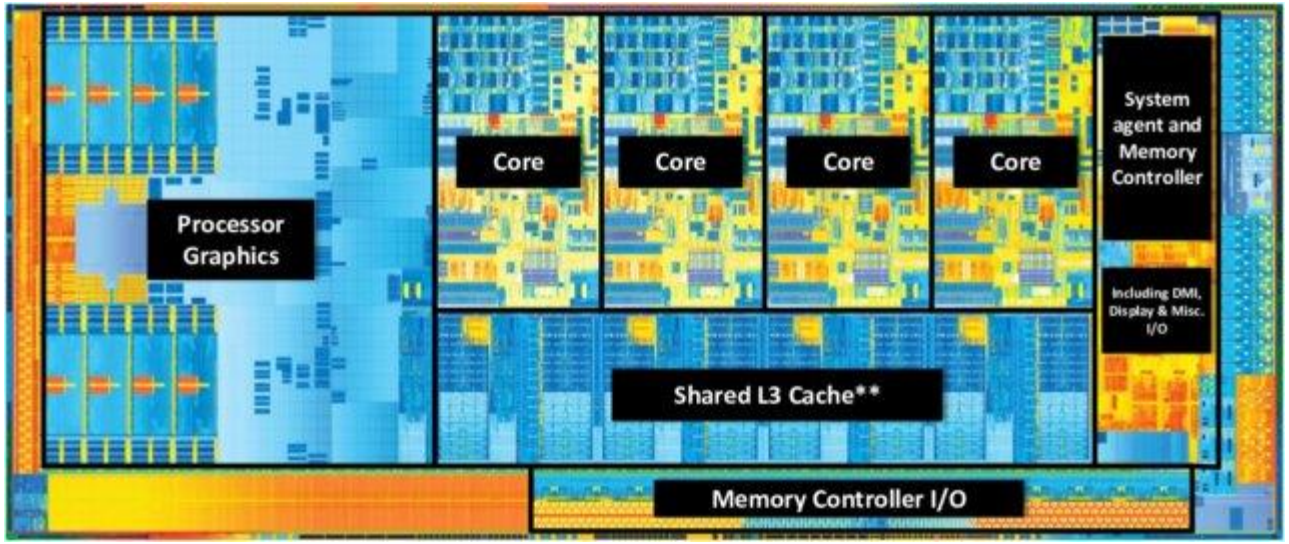


Intel® Core™ i7-3960X Processor Die Detail

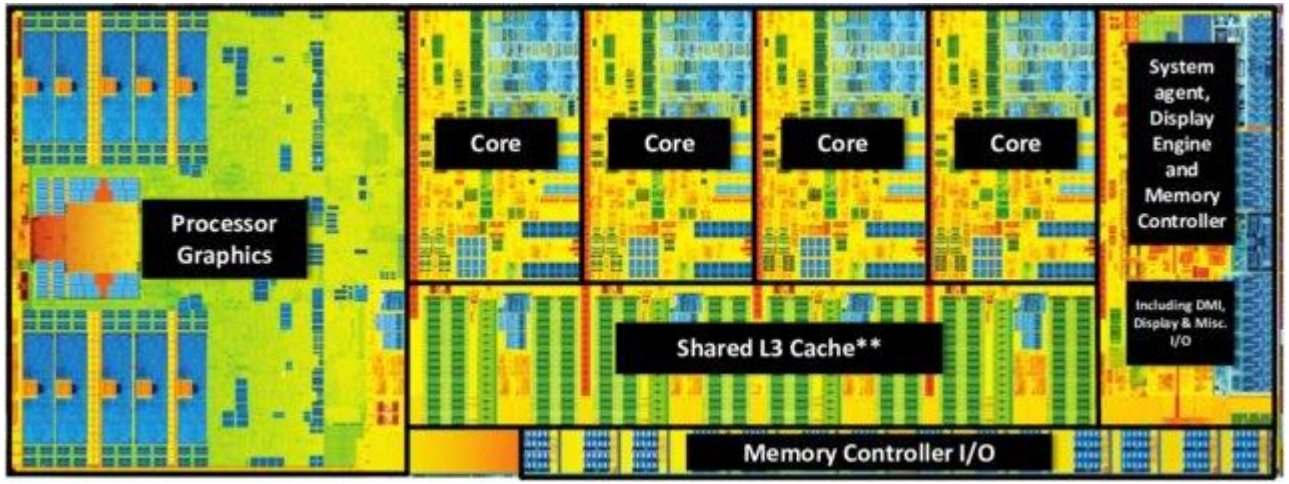
- Takeaway:
  - Cache (L3, L2 and L1) contributes > 60~70% of the chip area !!!
  - Cache memory: has to be very dense to pack more
  - Extremely compact layout !!!

laxmeesha@ee.iitb.ac.in
Image courtesy: Intel

Processor Cache Area



Quad-core Ivy Bridge



Quad-core Haswell

laxmeesha@ee.iitb.ac.in
Image courtesy: Intel

Processor Cache Area

Intel Raptor Lake (i9)

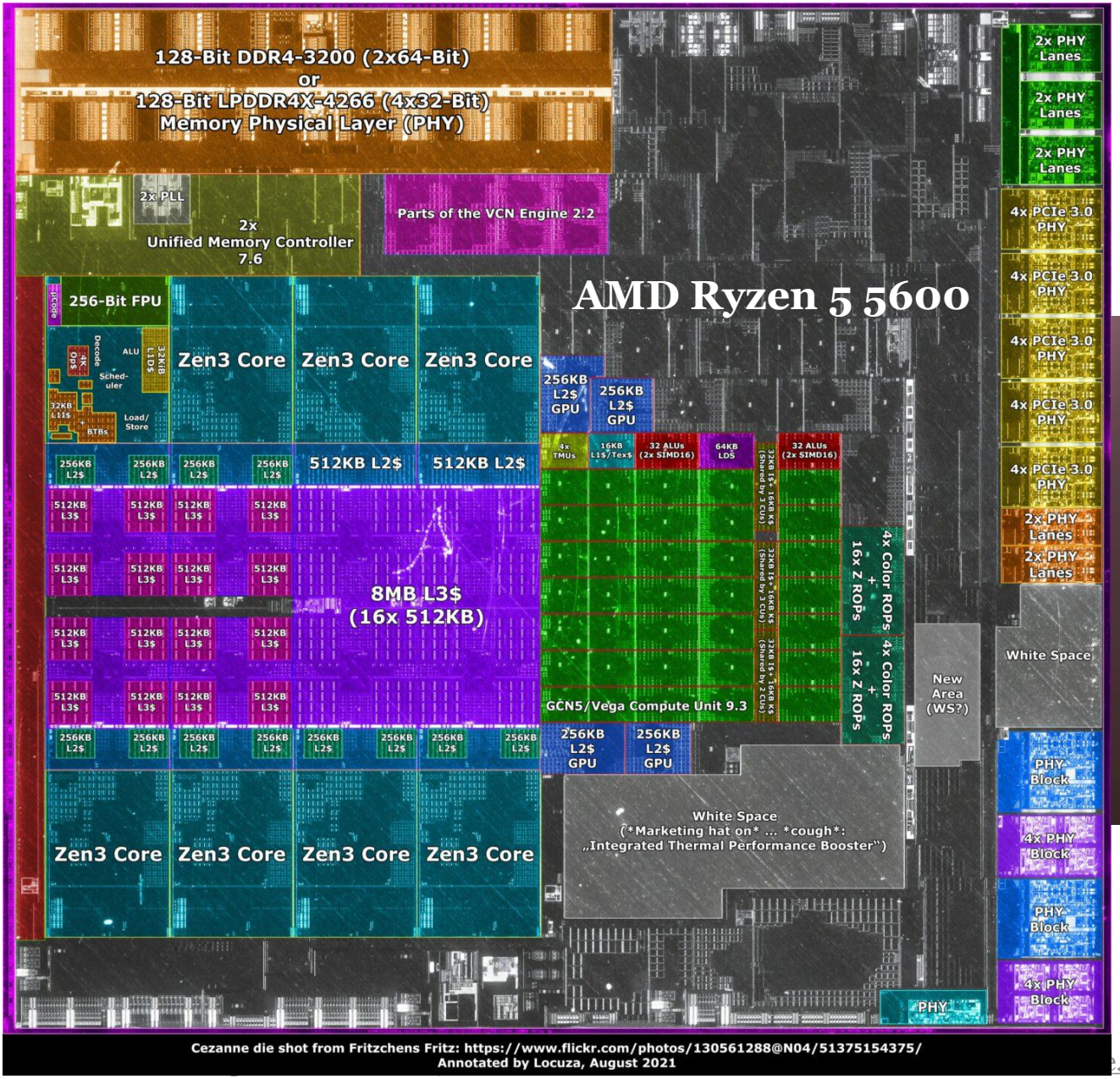
laxmeesha@ee.iitb.ac.in
Image courtesy: Intel

Processor Cache Area

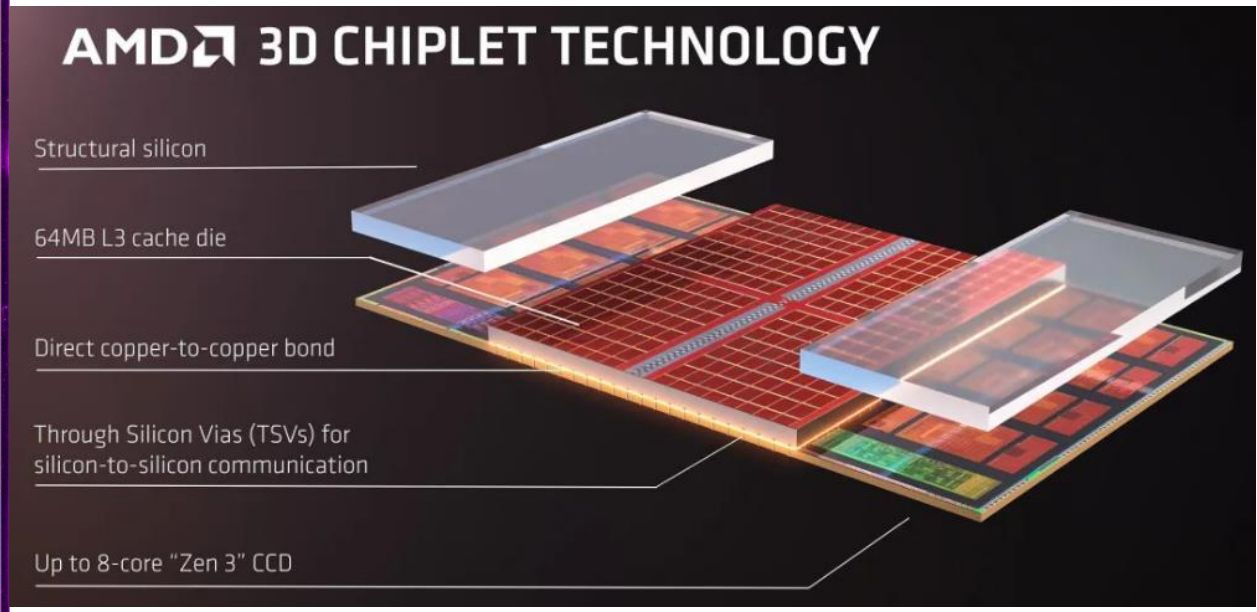
Intel Alder Lake

laxmeesha@ee.iitb.ac.in
Image courtesy: Intel

Processor Cache Area



Cezanne die shot from Fritzchens Fritz: https://www.flickr.com/photos/130561288@N04/51375154375/
Annotated by Locuza, August 2021

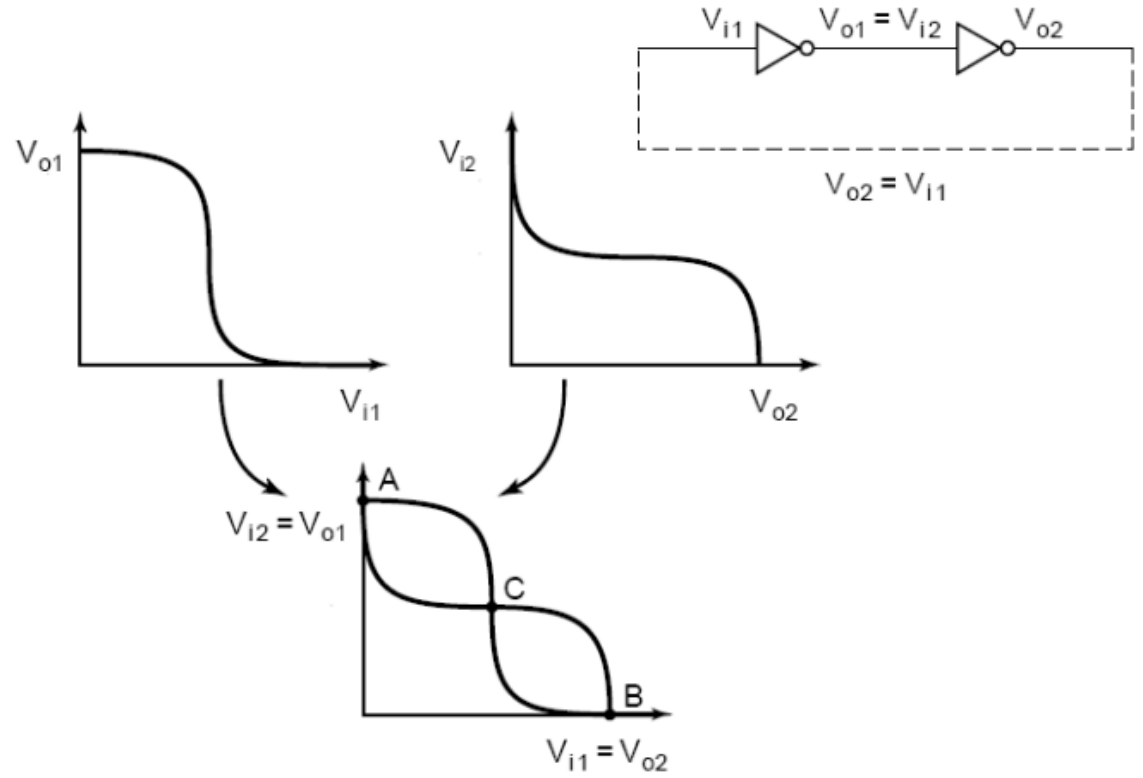


AMD Ryzen 7 Series 3D L3 Cache – Releasing some time soon!

Image courtesy: AMD

1- BIT MEMORY

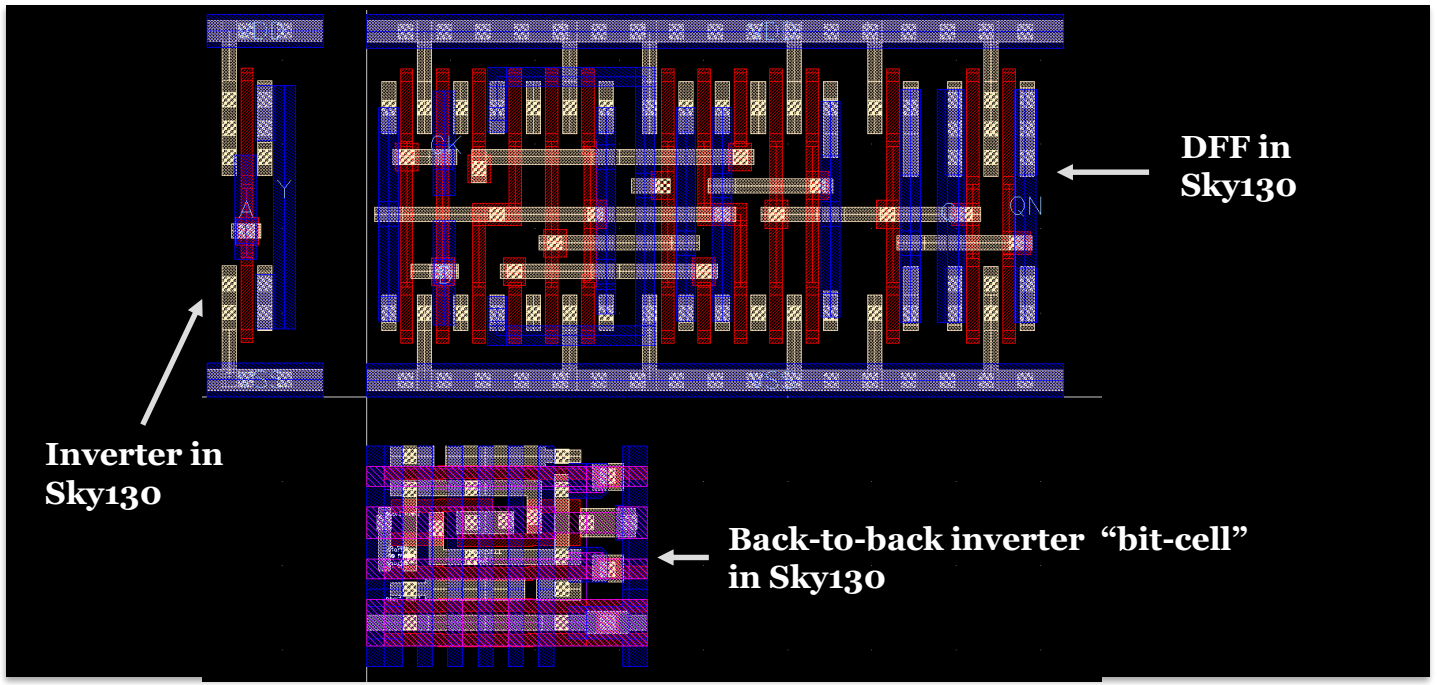
- Recall our discussion on 1-bit memory to implement DFF (master and a slave)
- Back to back inverter – two states → 1-bit memory



- Can we use a DFF to implement cache? – pros/cons?

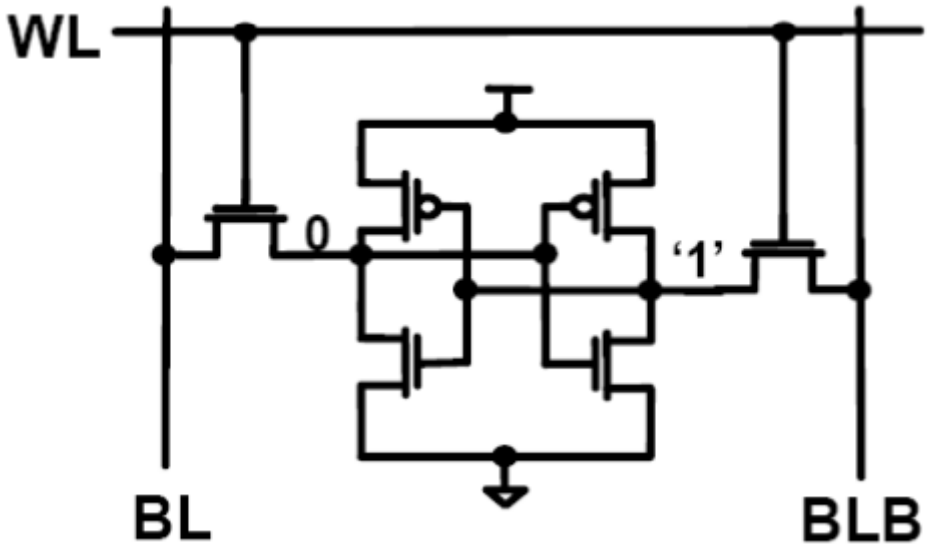
1- BIT MEMORY

- DFF: main usage was to implement 1-bit memory
- Cache: we will never access a single bit → access will be in bytes, half-word or word depending the micro-architecture of the processor
  - To implement a 1 byte memory → need 8 DFFs → is it worth it?
  - We have 2 back-to-back inverters per DFF and multiple inverters and TGs
  - DFF by default has a larger area to store a single bit memory !!!!!



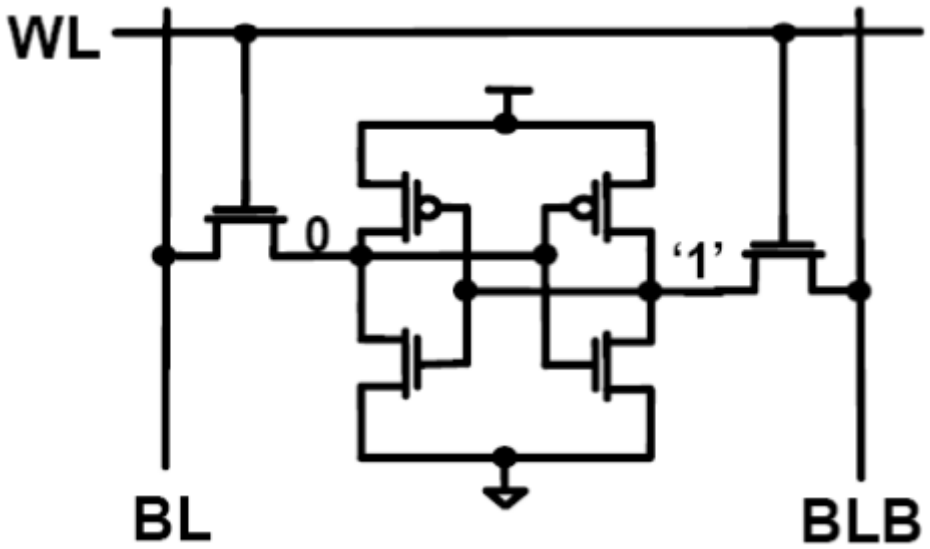
SRAM BIT-CELL

- Bit-cell: how to read/write/preserve a bit?
- Recall, in DFF we used TGs to perform read/write/preserve → can we reduce area further?
- Use NMOS only access transistors
- WL: word-line controls if read/write is to be performed. With WL = 1
  - If BL and BLB are driven to a value → we perform write
  - If BL and BLB are not driven → we perform read



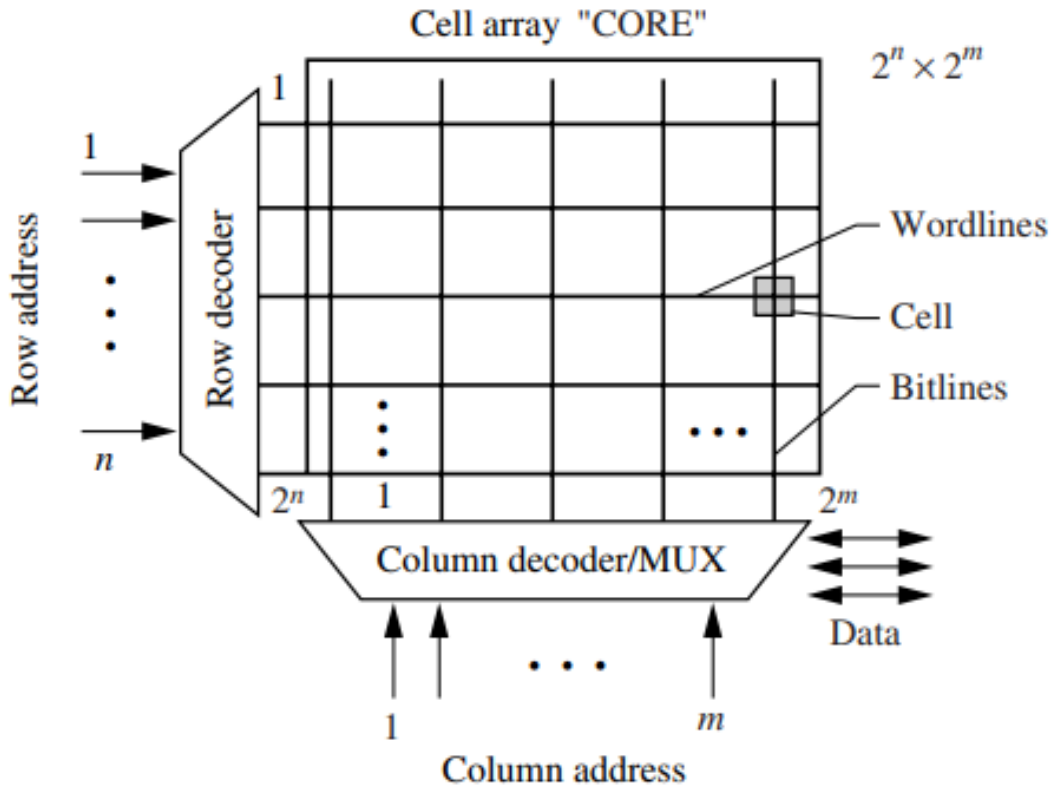
SRAM BIT-CELL

- SRAM bit-cell: Uses only 6 transistors to store 1-bit data → highly dense
- SRAM bit-cell: also called as 6T-cell
- Word-line controls if read/write is to performed
  - Word-line will have to be generated based on the memory address
- Bit-line: corresponds to data to be read or written (single-bit)



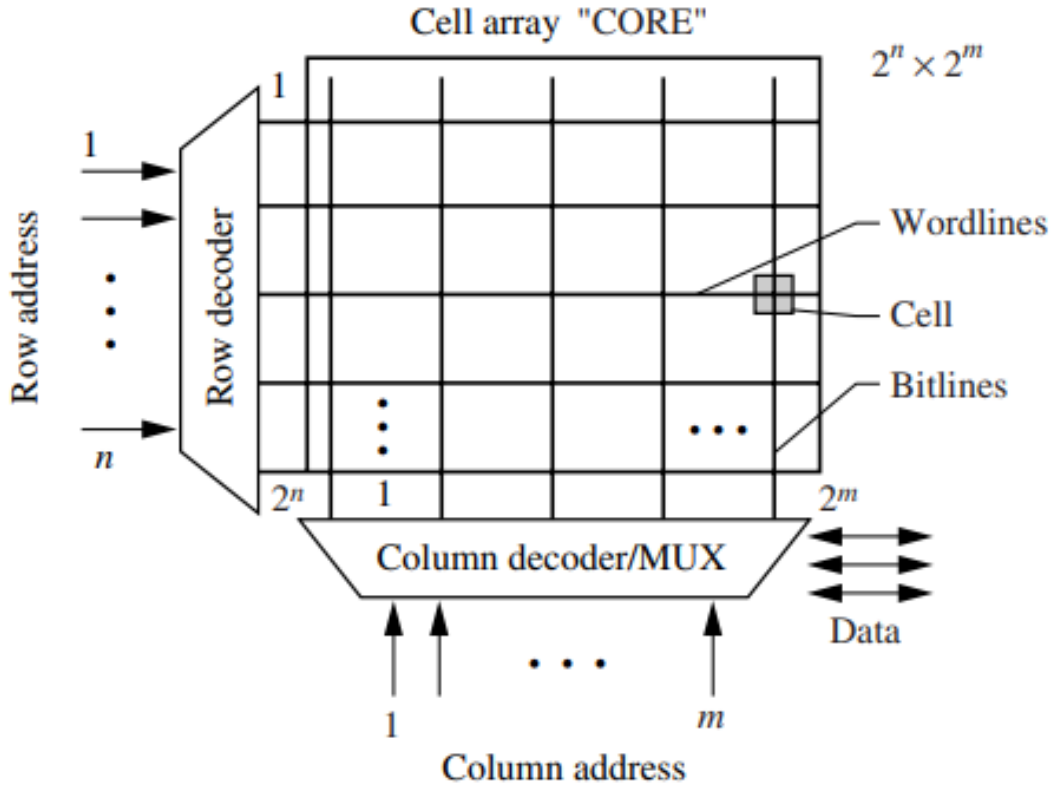
SRAM FULL PICTURE

- Address decoding: split into row and column decoding
  - Split between row and column → controls aspect ratio of memory
  - Say "n-bit" row address and "m-bit" column address
  - If n > m → more rows → tall memory
  - If m > n → more columns → wide memory

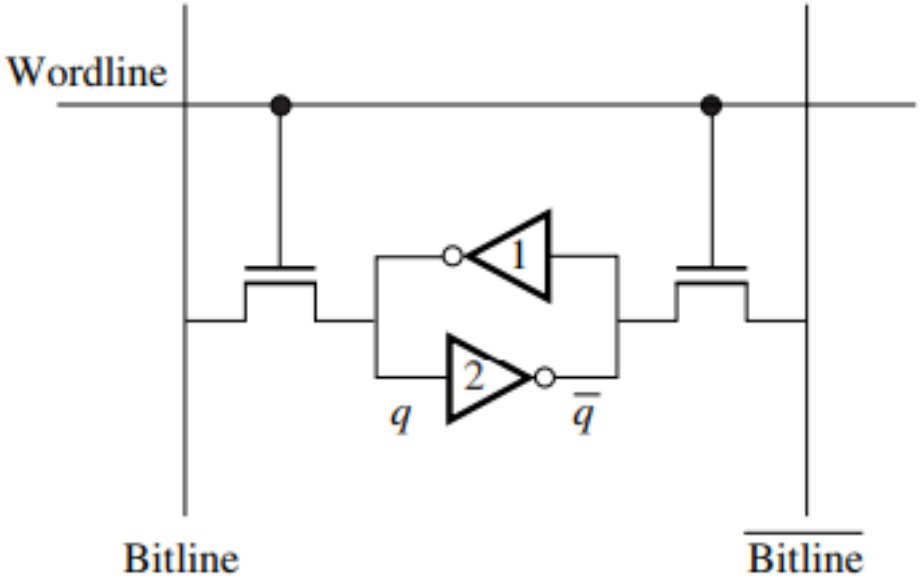


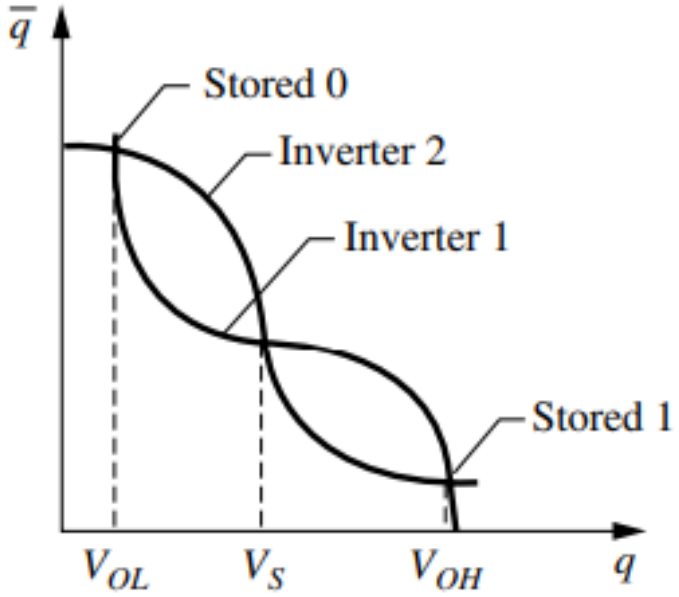
SRAM FULL PICTURE

- Implications:
  - If n is very large (tall memory) → the shared bitline per column is longer → this is a long metal in the layout → large bit-line capacitance → memory will be slower
  - If m is large (wide memory) → the wordline per row is longer → large wordline capacitance (because of routing and also the 2 access transistors gate cap per column) → Address decoder will drive larger load → delay limited by logical effort !!!!



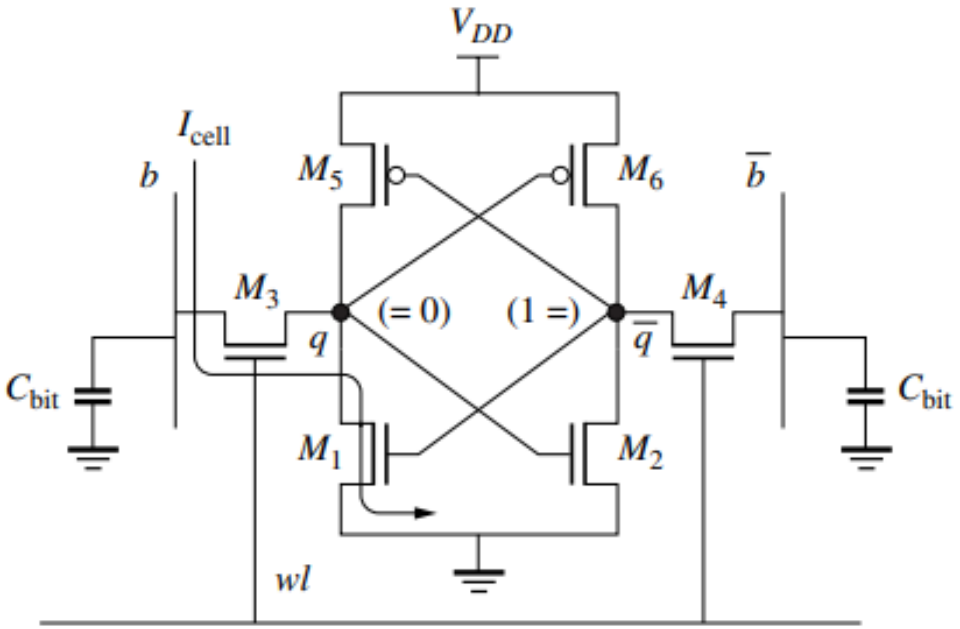
BACK TO THE BITCELL

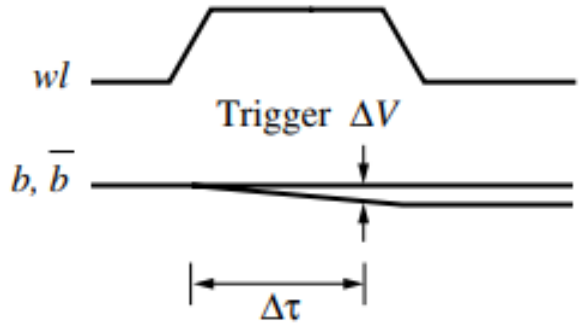




- Lets look at the read and write operations

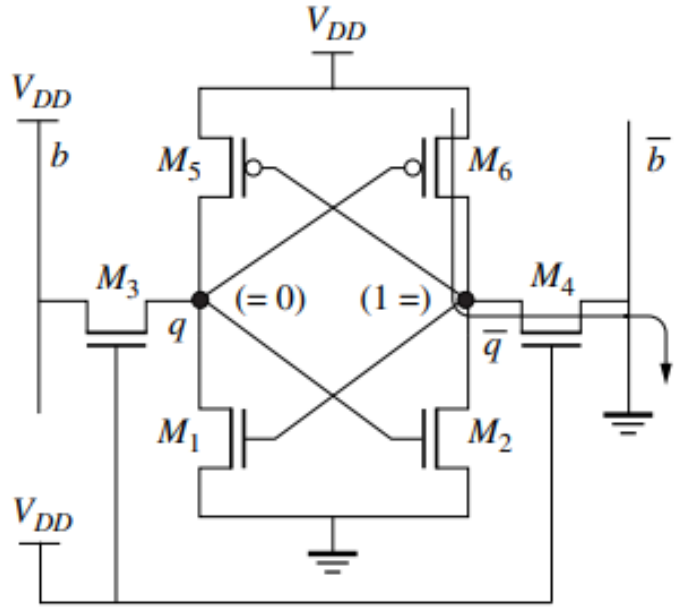
READ OPERATION

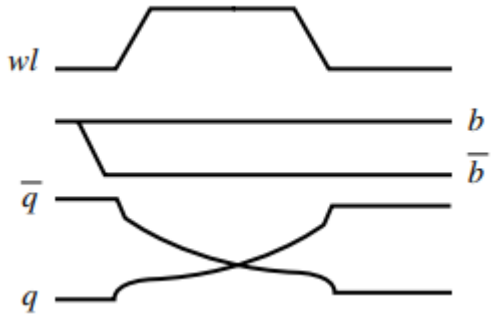




- Say bit "0" is already stored and we want to read
- Before performing any read → we pre-charge both BL and BLb to VDD → remove all prior read/write history on the lines → we cannot perform a read without pre-charging
- After pre-charge → WL is made high → capacitor discharge happens on one side!
- This small change in voltage will have to be amplified → will look at this later

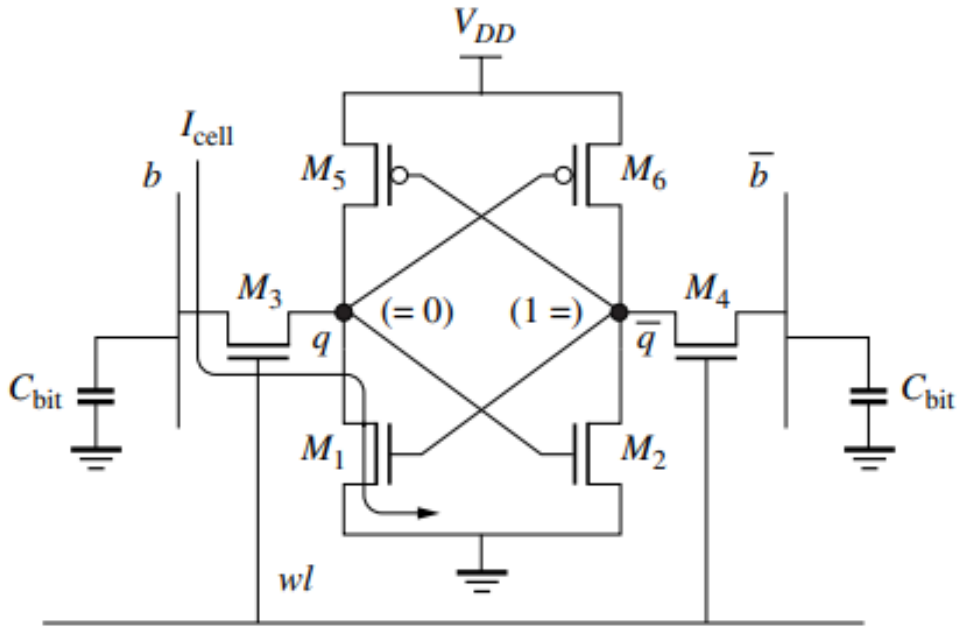
WRITE OPERATION

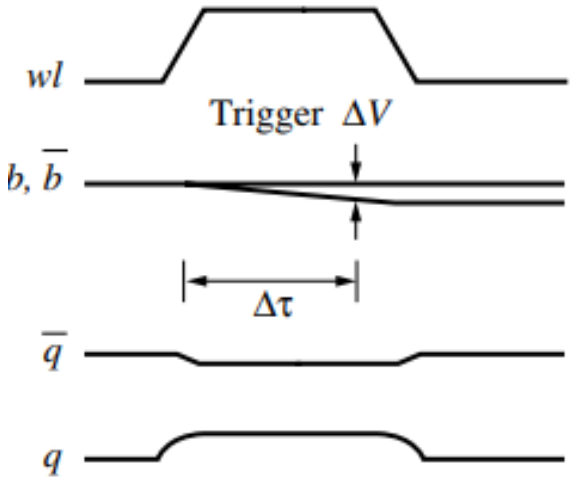




- To write new data → we need to overpower the existing cell data (if bits are different)
- During write → tie either BL or BLb to VDD/GND
- Say we want to write "1" into a cell that was earlier holding "0"
- In this case, M6 and M4 will fight to control node voltage at $\overline{q}$
- As a design criteria → M4 must be stronger than M6 !! (ie, M4 should have much smaller resistance compared to M6)

LOOK AT READ OPERATION AGAIN





- M3 and M1 are actually fighting for node "q" → if M3 is strong it can turn on M2 !
- During read, we do not want to disturb the state of node "q"
  - i.e, change in node voltage "q" should not flip the previously stored data !
- To ensure the this, M3 and M1 will have to be sized accordingly
  - i.e., M1 should be stronger than M3 !! (M1 will have lower resistor than M3)