Fundamental of Static Random Access Memory

Hanwool Jeong

hwjeong@kw.ac.kr

Contents

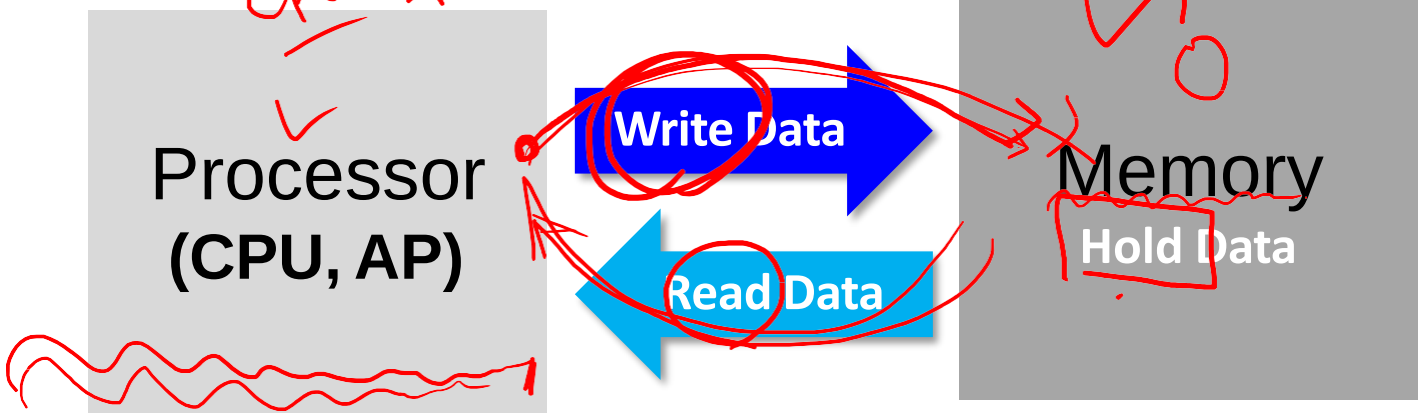
- Backgrounds & Motivation
- Structure of SRAM
- Operation of SRAM
- State-of-art SRAM Application; Computation in Memory
- Conclusion

Contents

- **Backgrounds & Motivation**
- Structure of SRAM
- Operation of SRAM
- State-of-art SRAM Application; Computation in Memory
- Conclusion

Memory Operation

- Processor calculates
- Memory can **hold/read/write** data



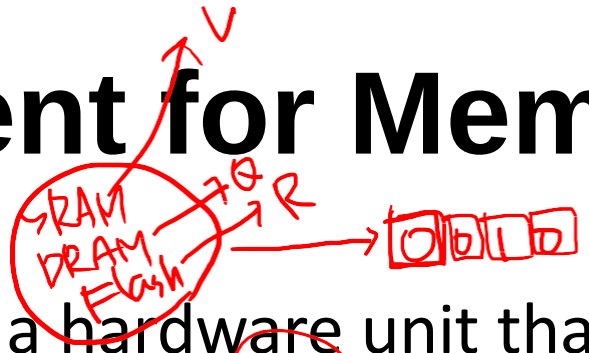
Data in Memory

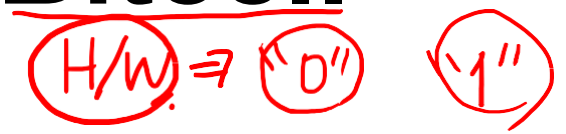
- Memory bitcell stores data in binary form : **“0” and “1”**
- Memory should be able to be in **two electrically separable states**.





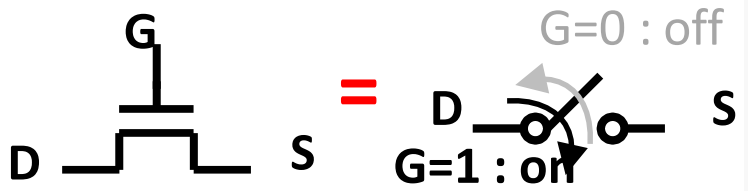
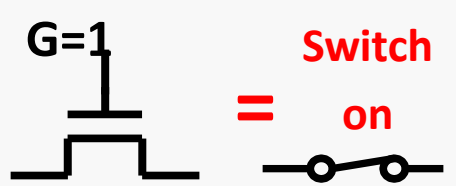
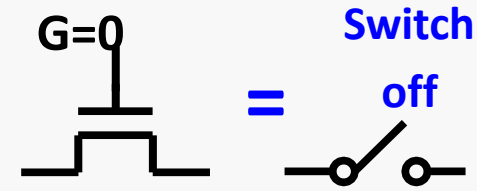
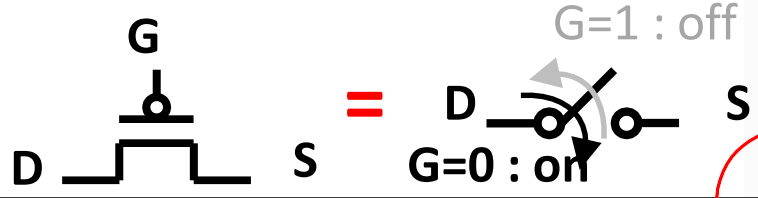
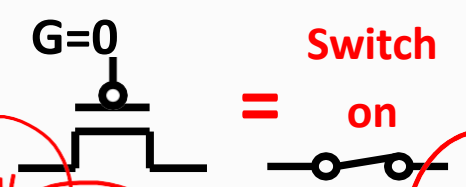
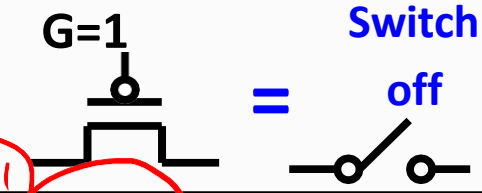
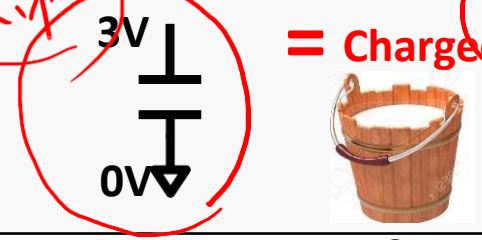
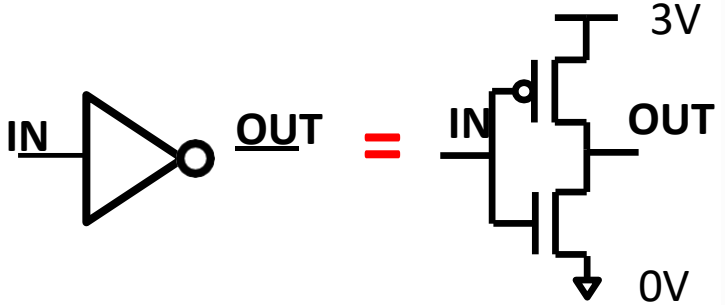
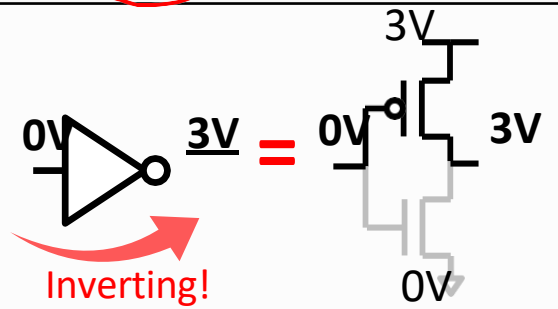
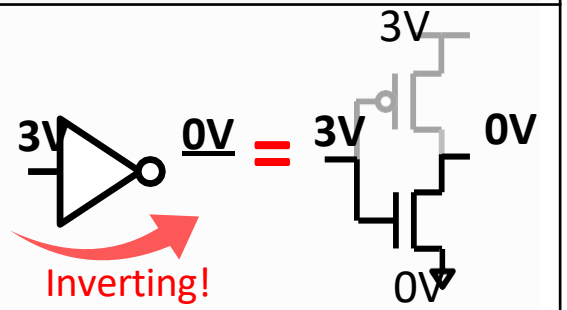
Requirement for Memory Bitcell



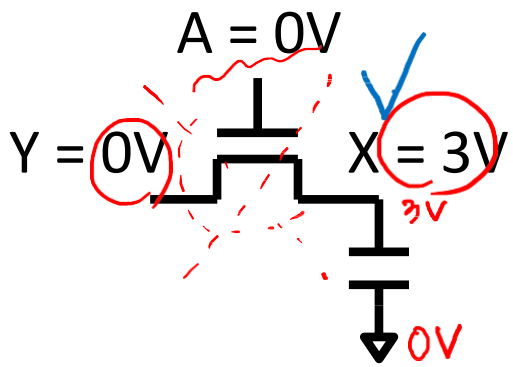
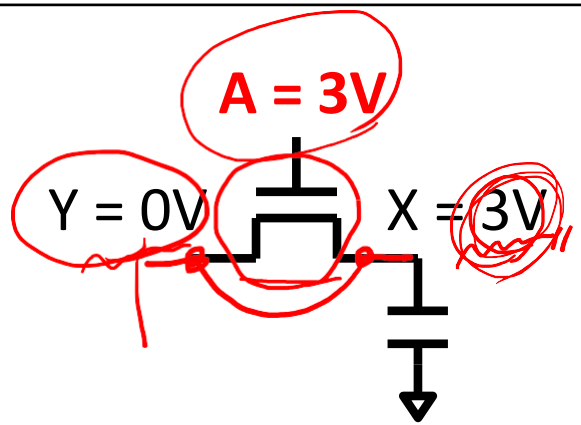
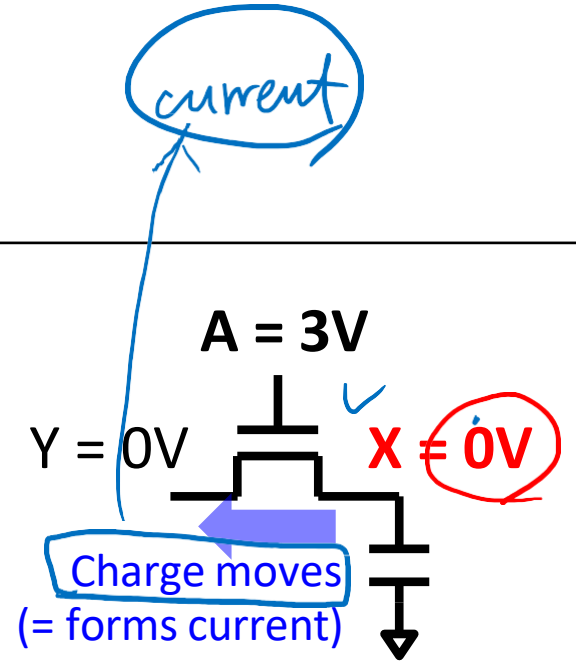


- **Memory bitcell** is a hardware unit that stores 1 bit data “0" or "1"
- Memory bitcell should **1)hold/write/read 2)two possible states**
  1) Hold : to keep the current state of the bitcell from the noise
  2) Write : to change the memory bitcell into wanted state
  3) Read : to figure out what the memory bitcell state is
- Can you make memory bitcell?

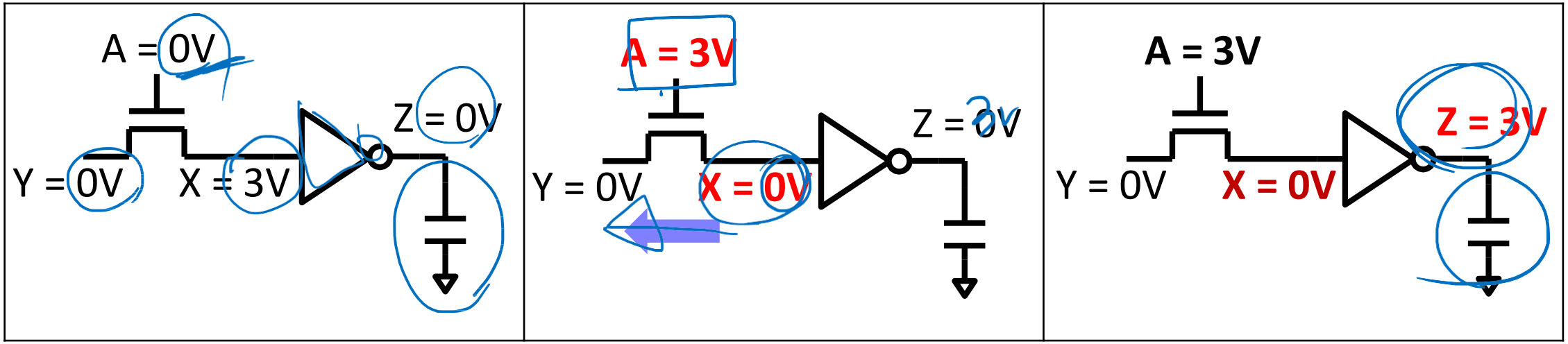
What Do We Have?

Name		Symbol	Operation	
MOSFET	nMOSFET			
	pMOSFET			
Capacitor				
Inverter				

Example (1)

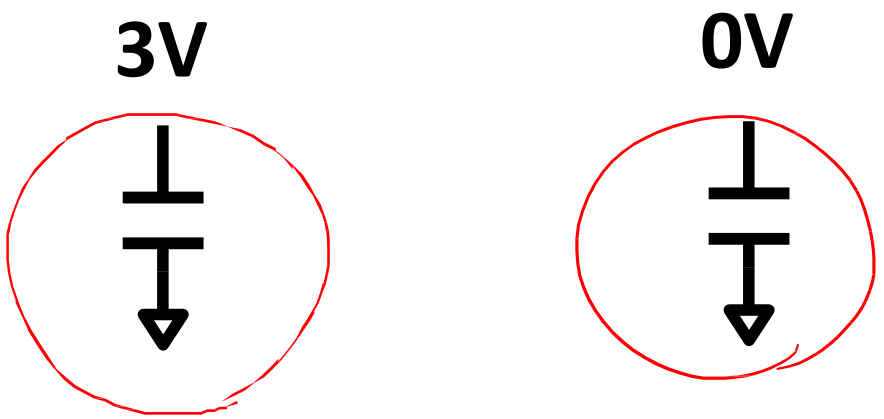
		
✓ nMOSFET is off by A = 0V. ✓ Capacitor is charged.	✓ nMOSFET is turned on **by A = 0V** and connects X to Y.	✓ X is equalized to Y as X = 0V ✓ **Capacitor is discharged**

Example (2)



Let's Make A Memory Bitcell
: Hold Two Possible States

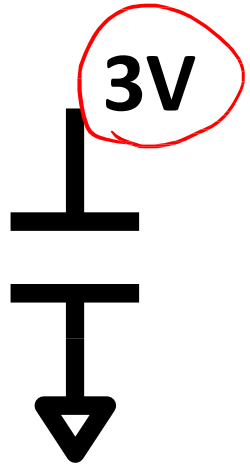
- Remember: Memory bitcell should **hold**/write/read **two possible states**
- Capacitor can represent two electrically separable states: We can exploit this



➔ **Can we use a capacitor as a memory bitcell? How about writing data?**

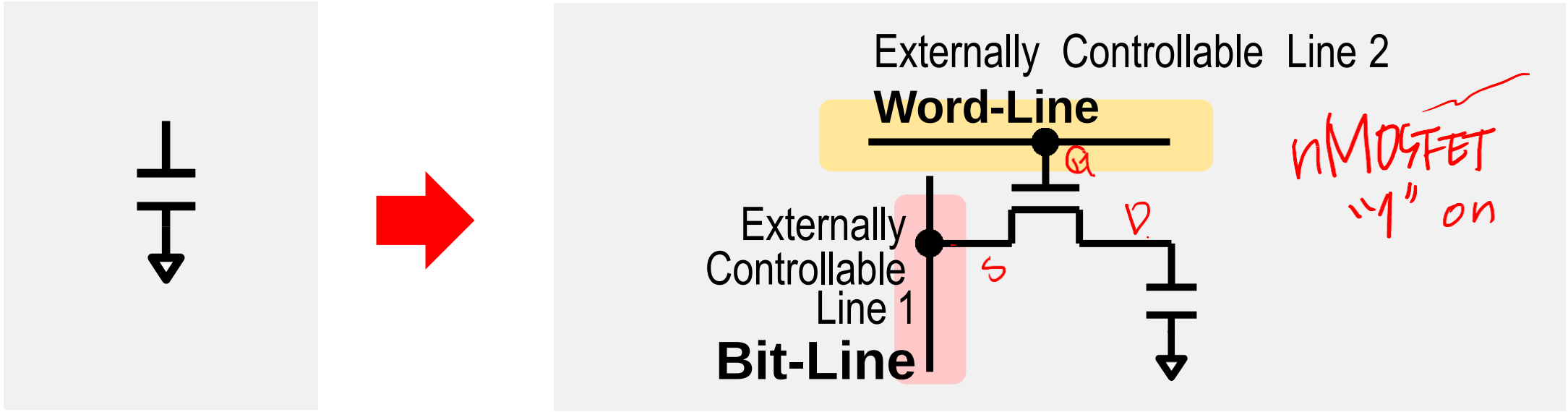
Problem of Capacitor Memory Bitcell

- Can you change the following state into data "0"? ➔ **NO**



Data “1”

Modification of Memory Bitcell

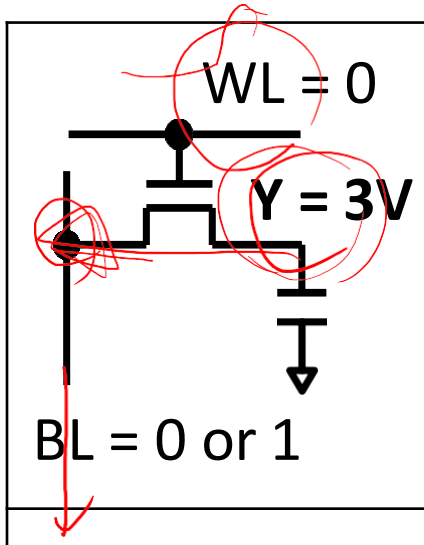
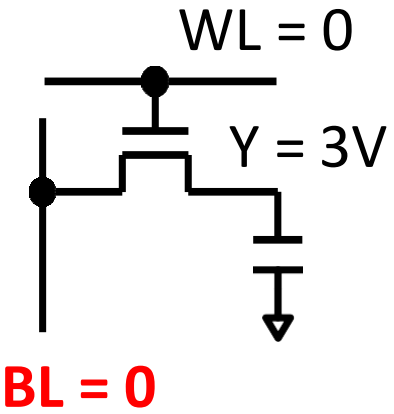
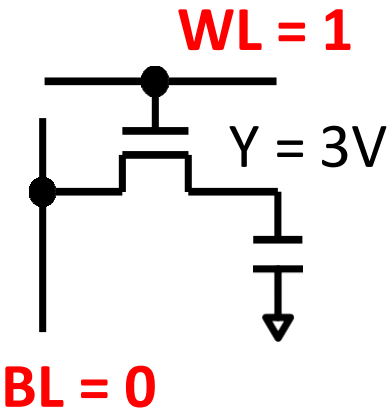
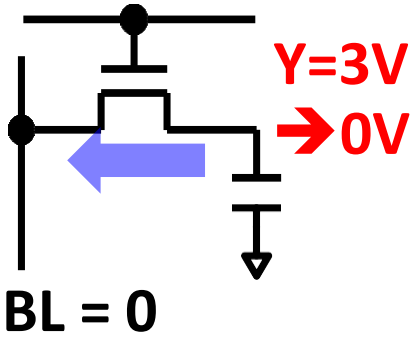
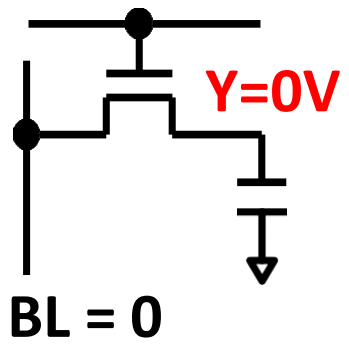


Capacitor

Capacitor
+ nMOSFET w/
Two externally controllable lines

Let's Make A Memory Bitcell
: Write Two Possible States

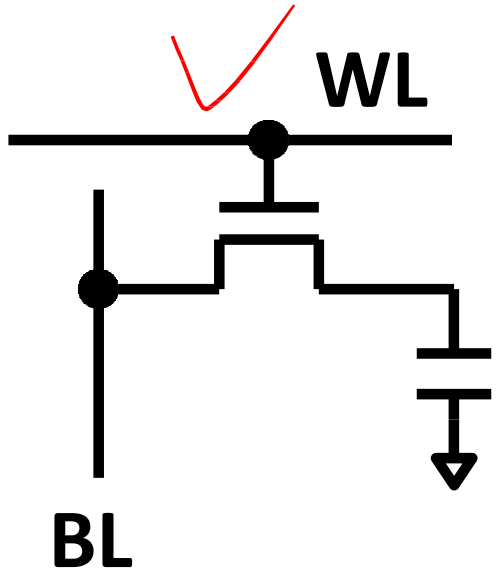
- Let's write data "0" to memory bitcell initially storing "1"

				
Hold data "1"	Driving BL to "0"	Driving WL to "1"	Y is discharged	**Hold data "0"**

1T1C DRAM Bitcell

- Dynamic Random Access Memory (DRAM) Bitcell

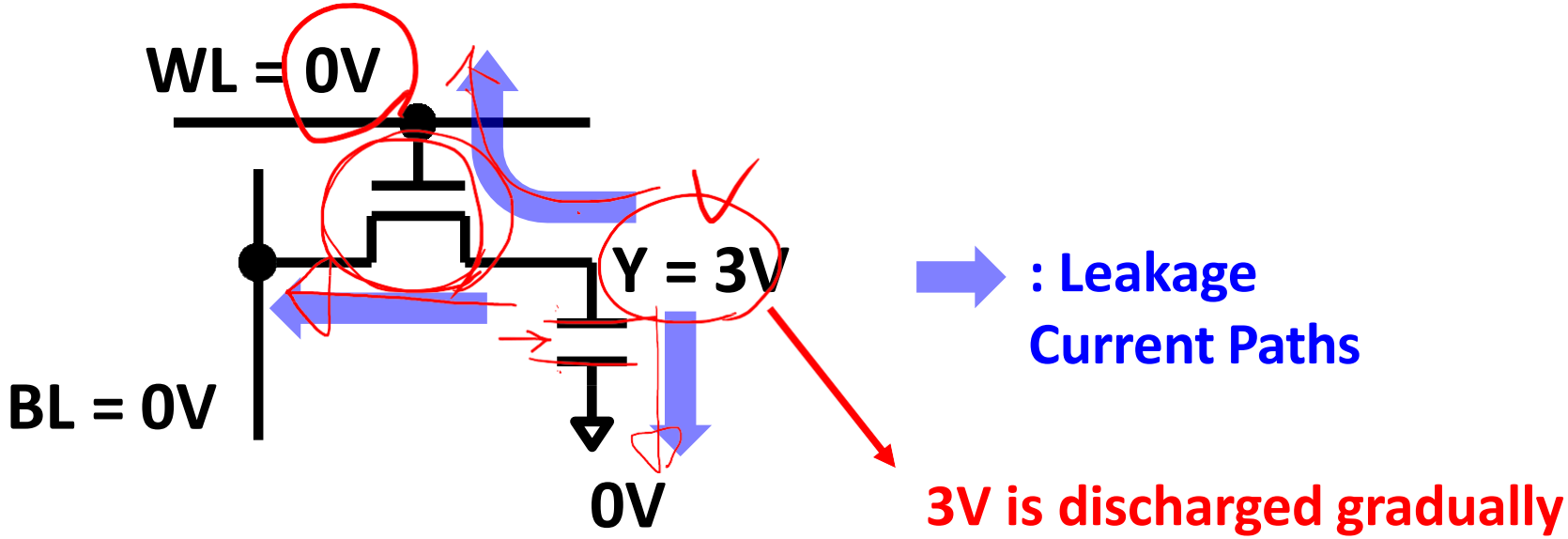




Limitation of DRAM

- Suppose that data “1” is stored in the DRAM bitcell
- What if there is a leakage current or Noise?

  ➔ **Data can be “dynamically” changed**

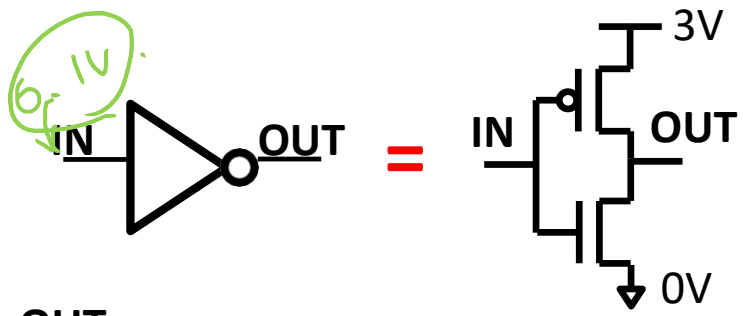


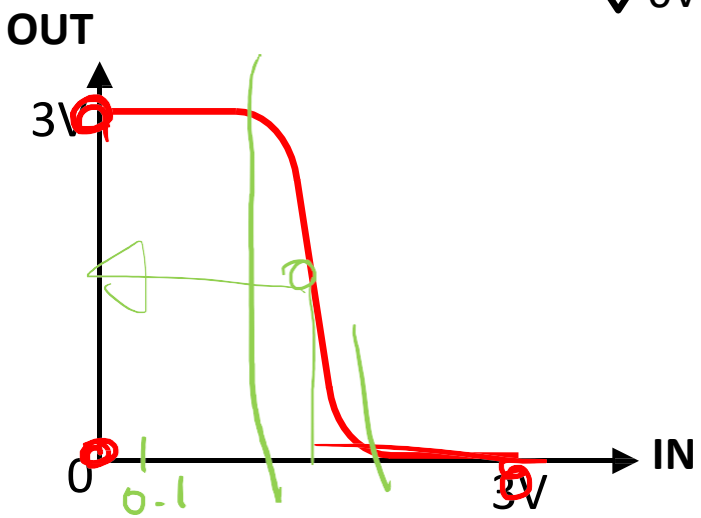
Contents

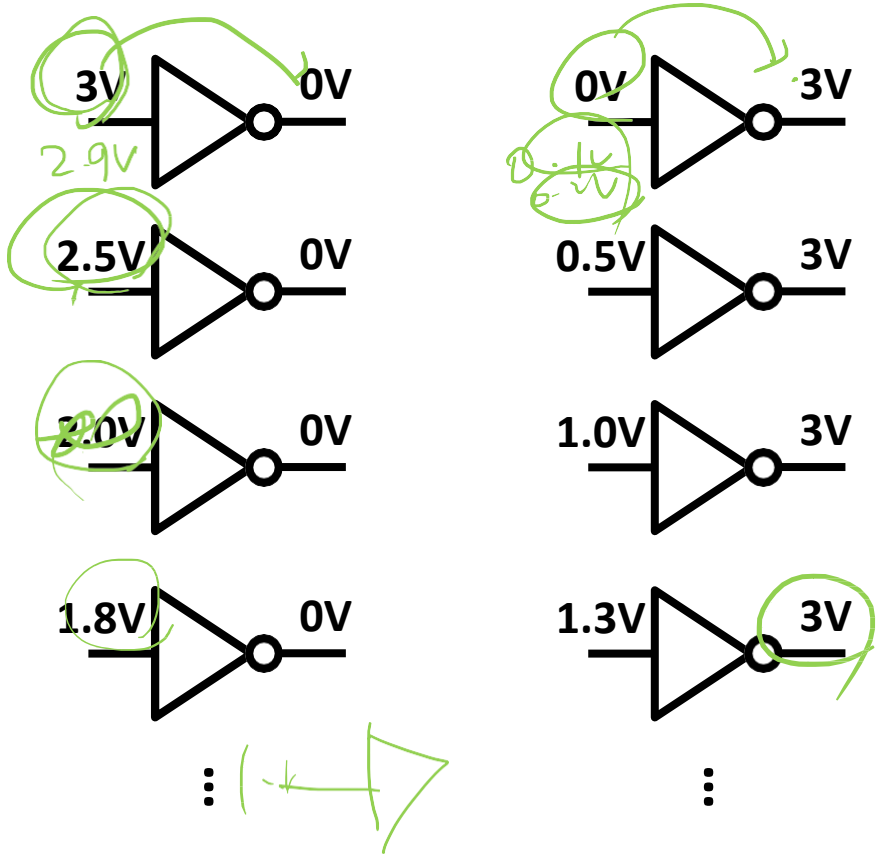
- Backgrounds & Motivation
- **Structure of SRAM**
- Operation of SRAM
- State-of-art SRAM Application; Computation in Memory
- Conclusion

Regeneration Feature of Inverter

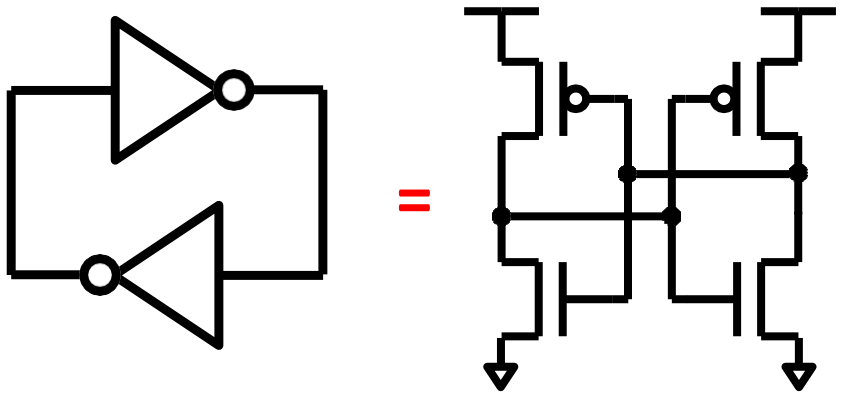
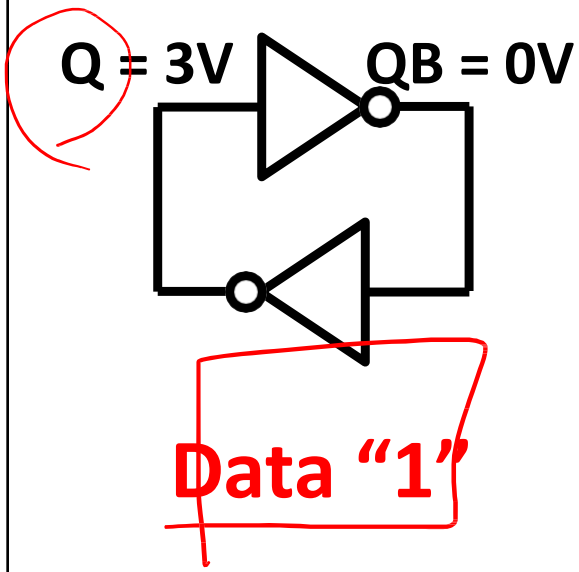
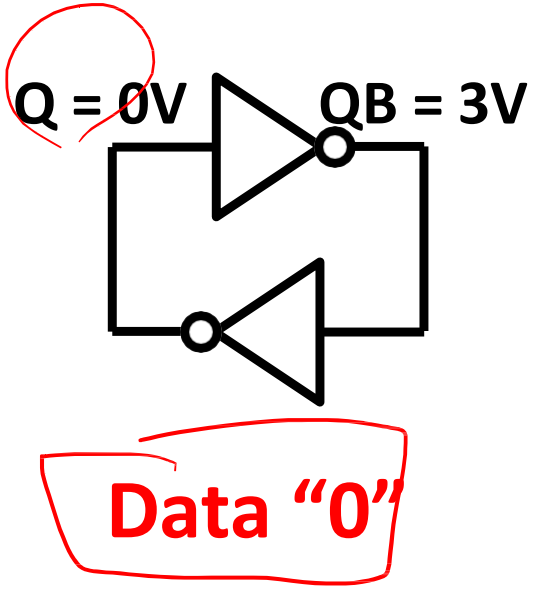
- While the input can have arbitrary values, output can become 3V or 0V



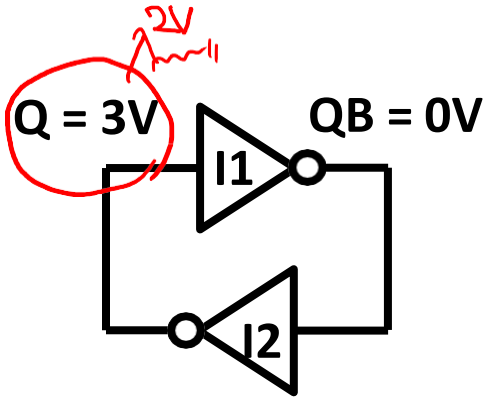
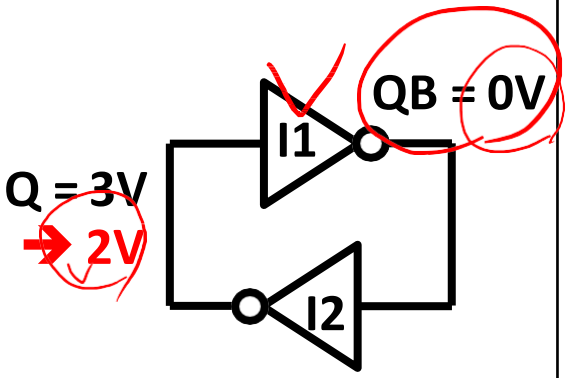
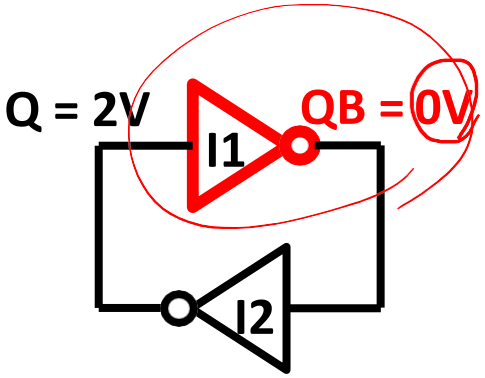
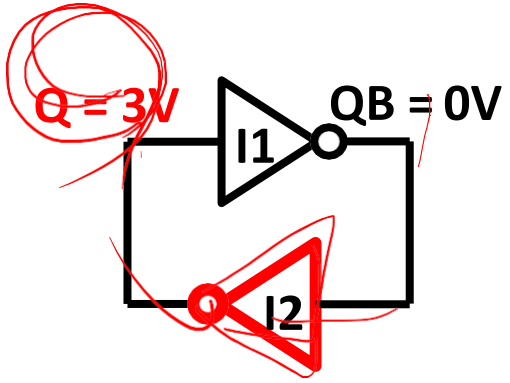




Cross-coupled Inverter as A Memory Bitcell

Structure	Data States	
		

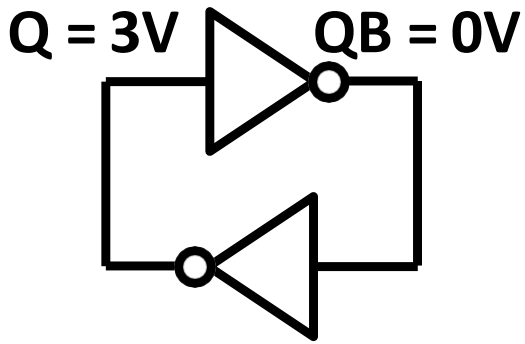
Robustness of Cross-Coupled Inverter

			
Hold data “1”	Q is accidentally lowered to 2V	QB is kept 0V because of regeneration of I1	Q1 is back to 3V because of regeneration of I2

➔ Cross-coupled inverter is free from the problem which DRAM suffers from

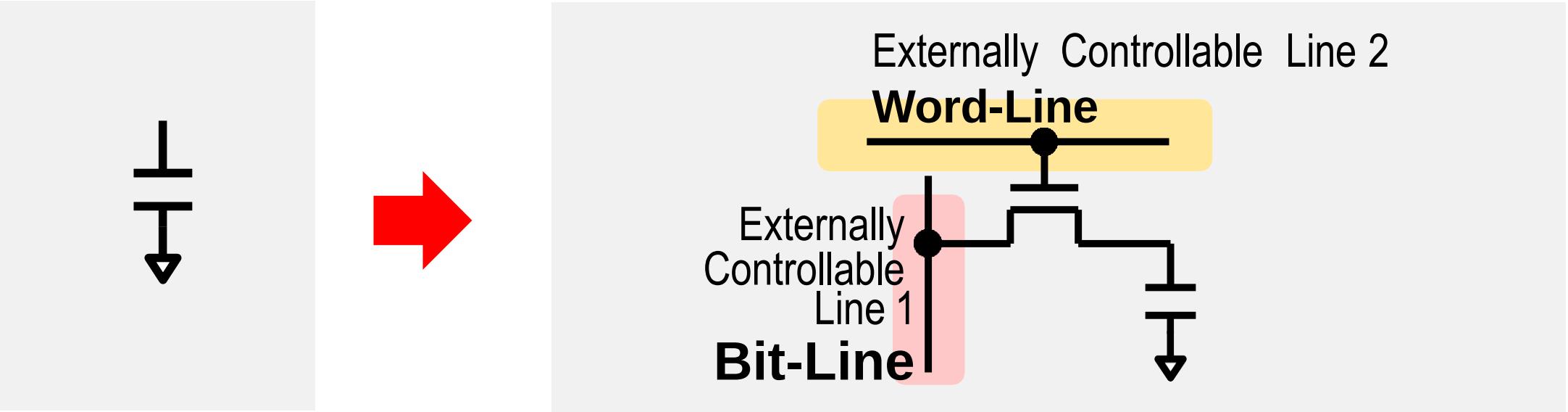
Cross-coupled Inverters as Memory Bitcell?

- Can you change the following state into data "0"? ➔ **NO**



Data “1”

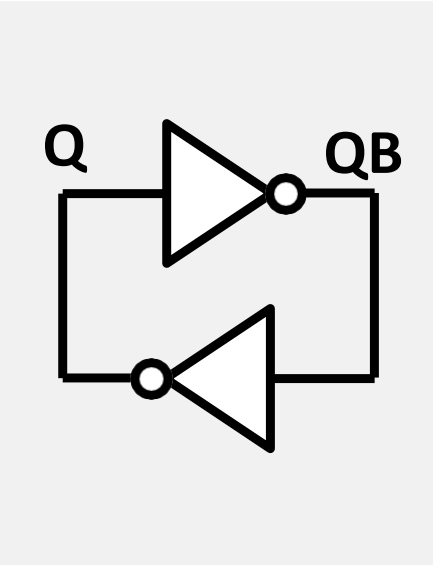
Revisit: What We Did for DRAM Bitcell

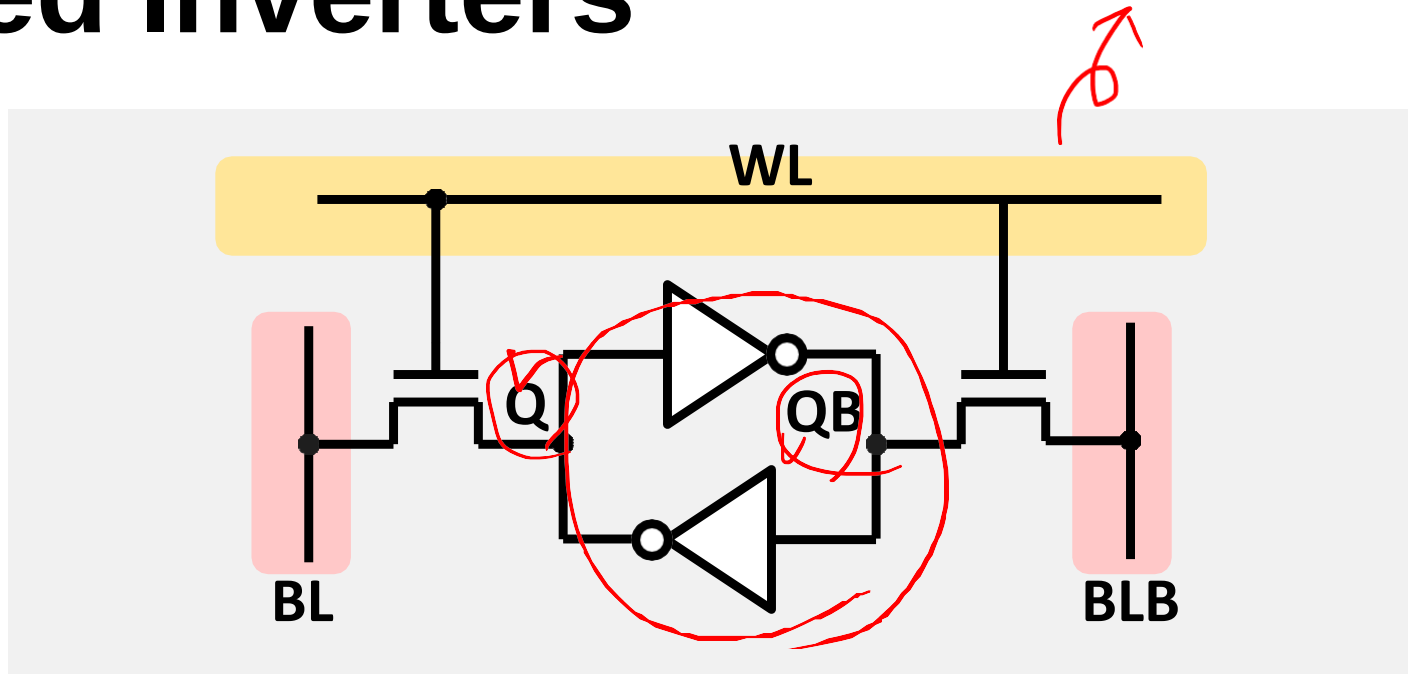


Capacitor

Capacitor
+ nMOSFET w/
Two externally controllable lines

Enabling Read/Write Operation to Cross-Coupled Inverters



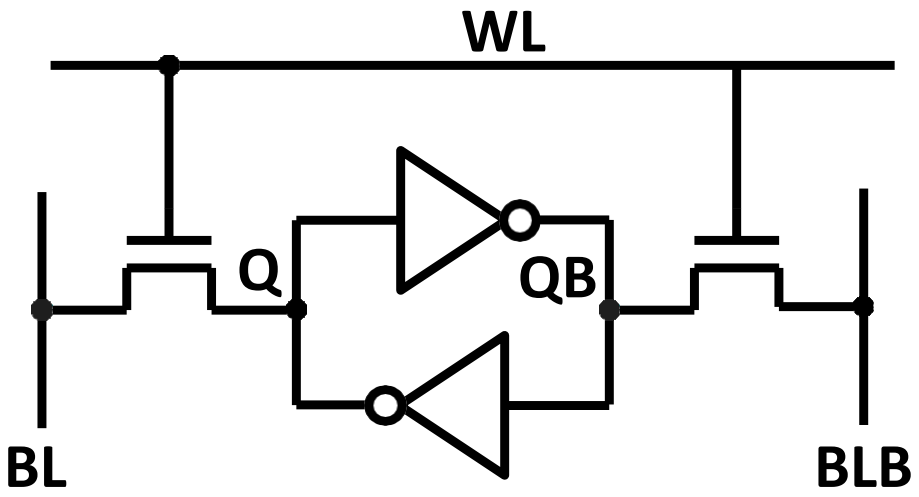


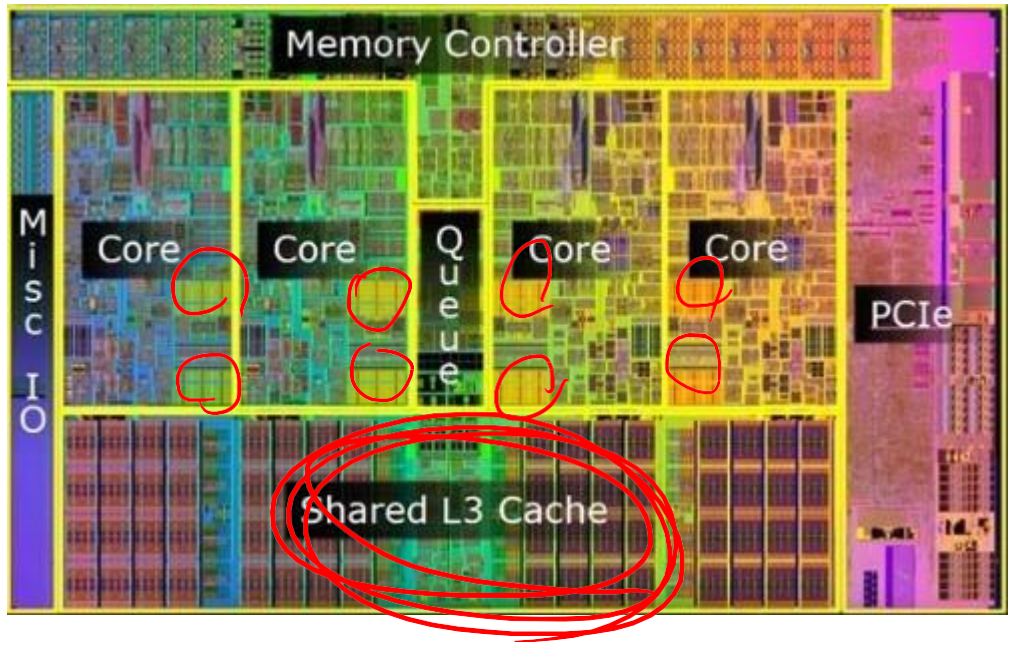
Cross-Coupled Inverter

Cross-Coupled Inverter
+ Two nMOSFET w/
2+1 externally controllable lines

6T SRAM Bitcell

- Static Random Access Memory (SRAM) Bitcell
- “Static” means invariant memory storage operation over time





Contents

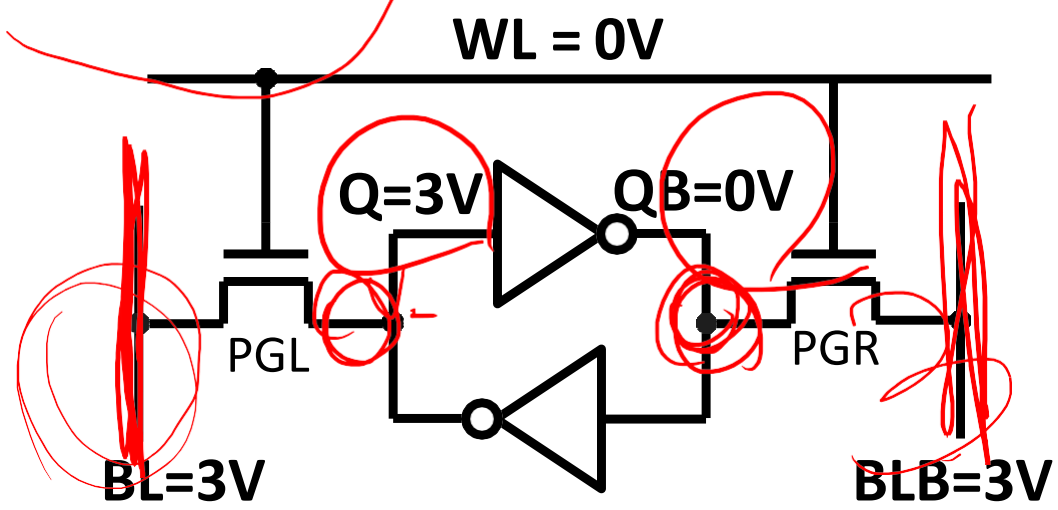
- Backgrounds & Motivation
- Structure of SRAM
- **Operation of SRAM**
- State-of-art SRAM Application; Computation in Memory
- Conclusion

Revisit: Requirement of Memory Bitcell

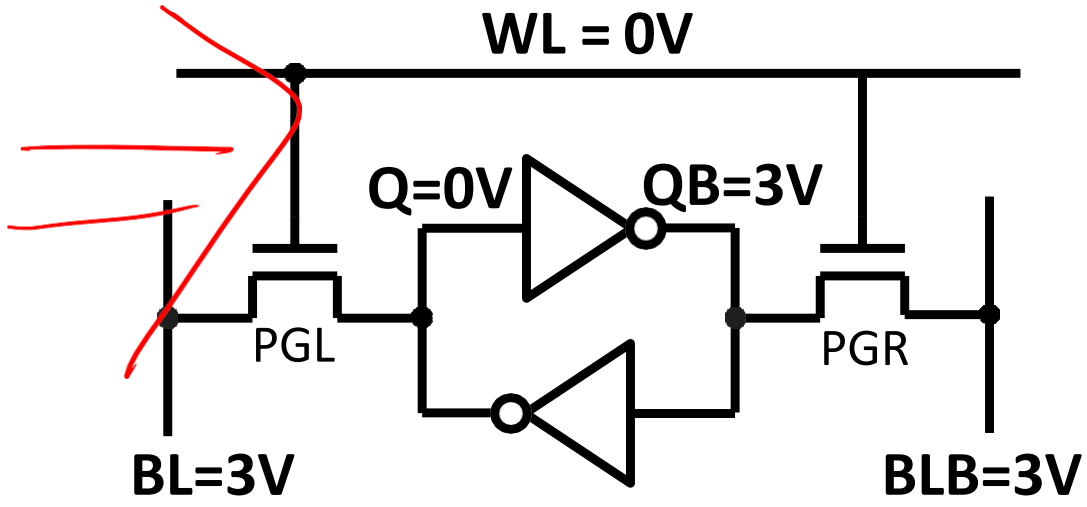
- **Memory bitcell** is a hardware unit that stores 1 bit data “0" or "1"
- Memory bitcell should **1)hold/write/read 2)two possible states**
  1) Hold : to keep the current state of the bitcell from the noise
  2) Write : to change the memory bitcell into wanted state
  3) Read : to figure out what the memory bitcell state is

Hold Operation

- Q and QB are isolated BL & BLB because of WL = 0V making PGL & PGR turned off.



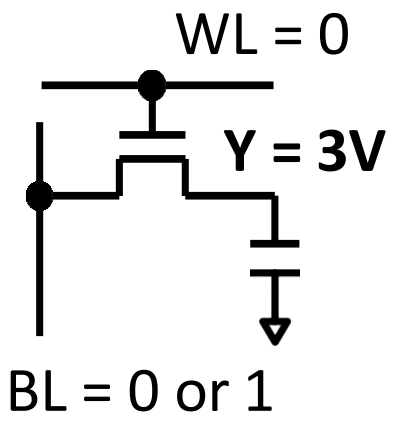
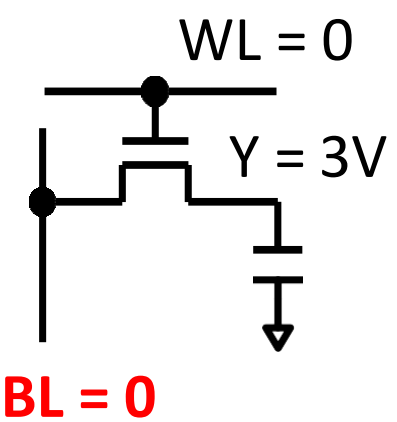
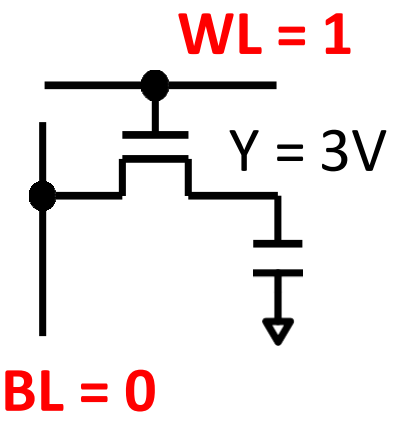
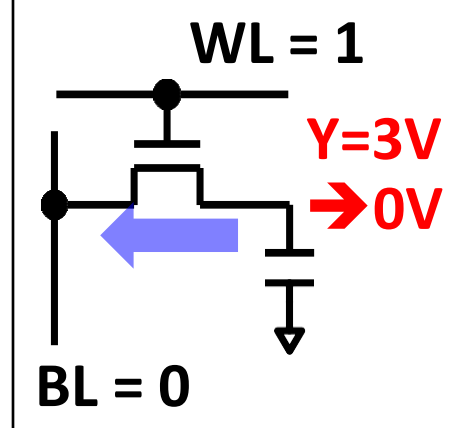
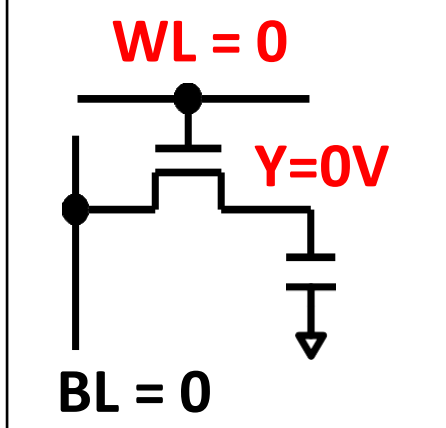
Hold Data “1”



Hold Data “0”

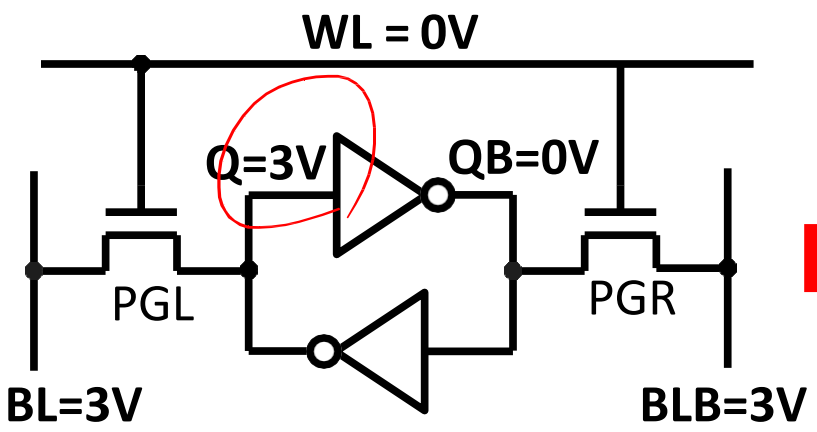
Revisiting DRAM Write Operation

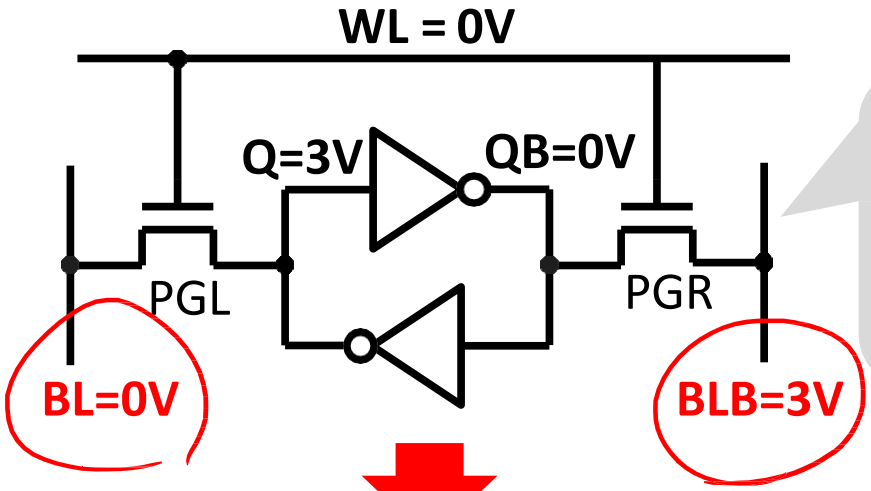
- Let's write data "0" to memory bitcell initially storing "1"

				
Hold data "1"	Driving BL to "0"	Driving WL to "1"	Y is discharged	**Hold data "0"**

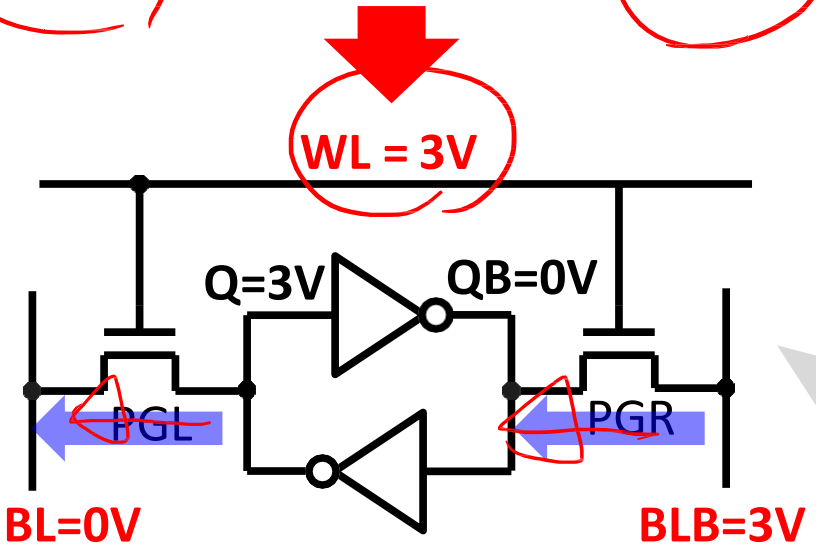
Write Operation

1) Hold Data “1”

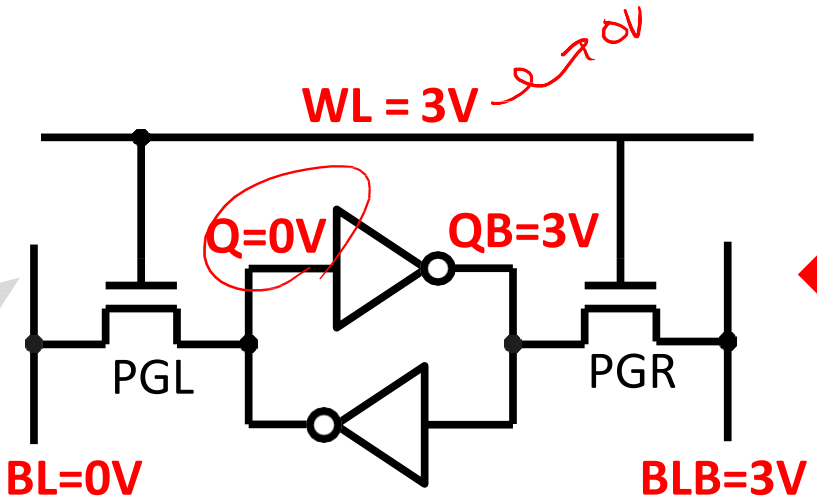




2) Driving BL/BLB as wanted



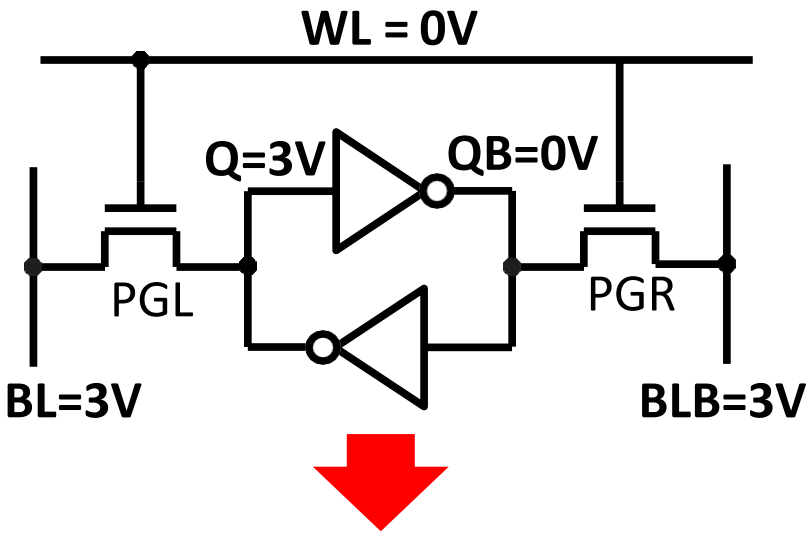
3) Driving WL high tuning on PGL/PGR



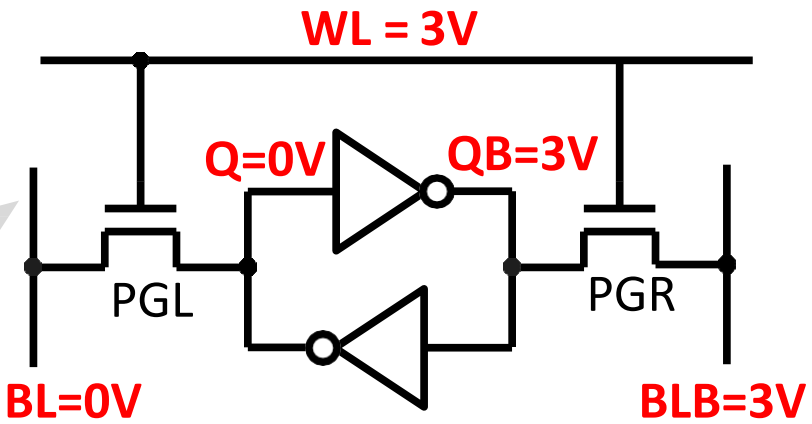
4) Q/QB change reflecting BL/BLB

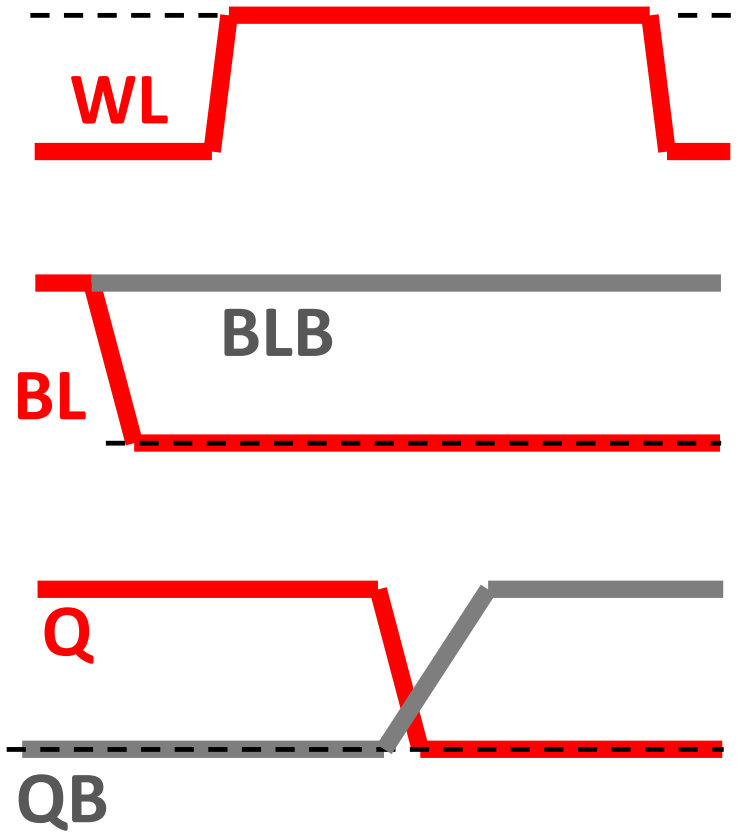
Write Operation Waveform

Initially Hold Da ta “1”



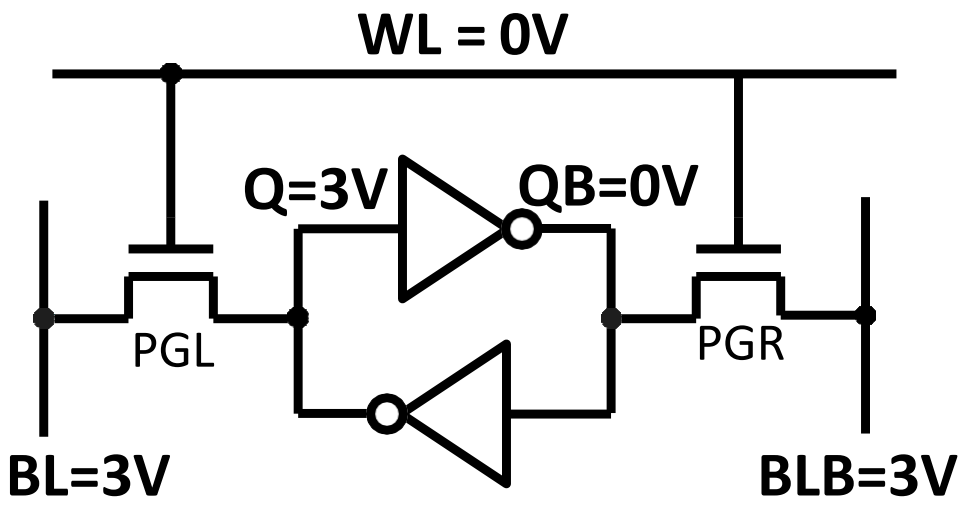
Data “0” Written



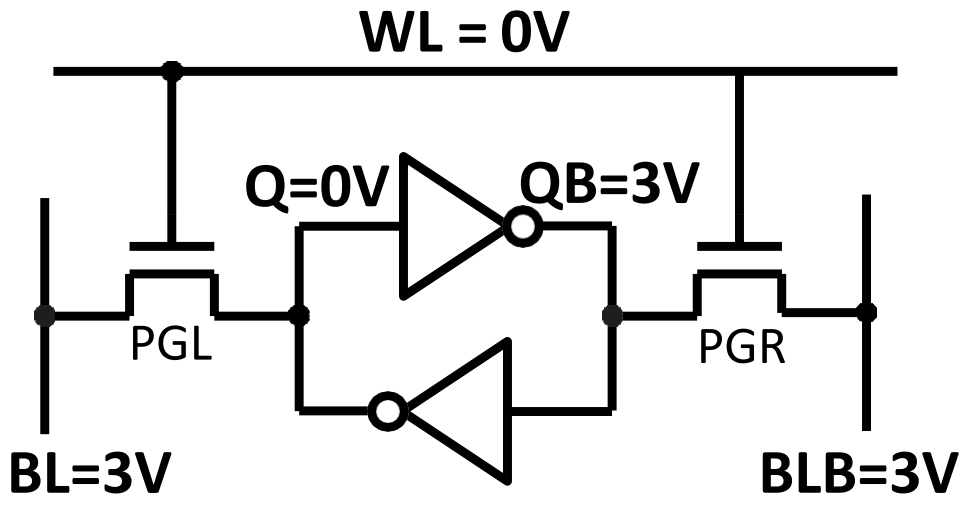


Goal of Read Operation

- Figuring out what the memory bitcell state is between “0” or “1”
- BL/BLB are externally connected, so we can utilize it.



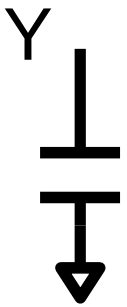
Data “1”



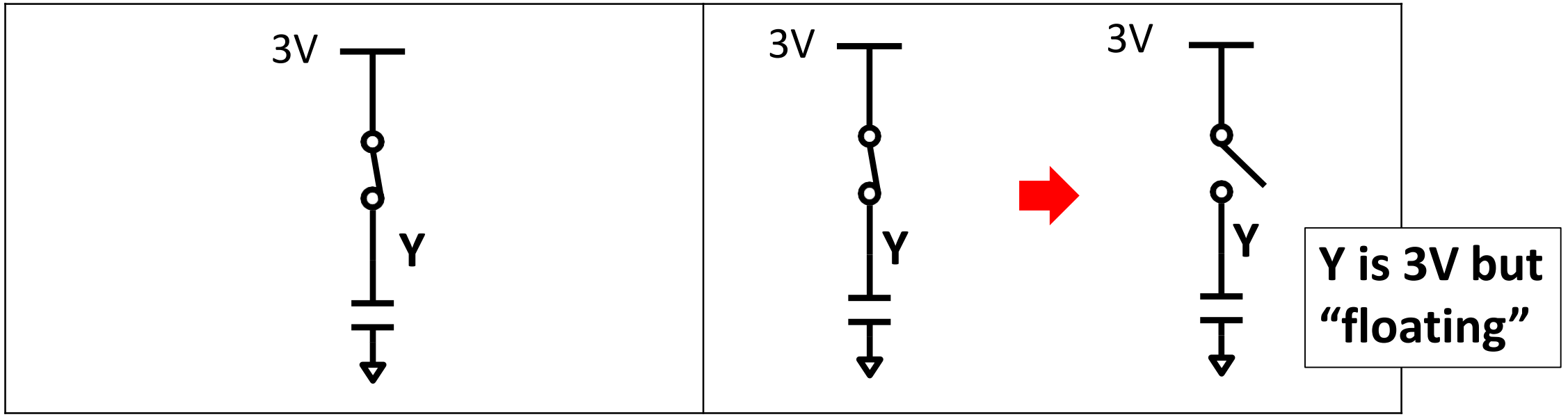
Data “0”

Precharge Operation

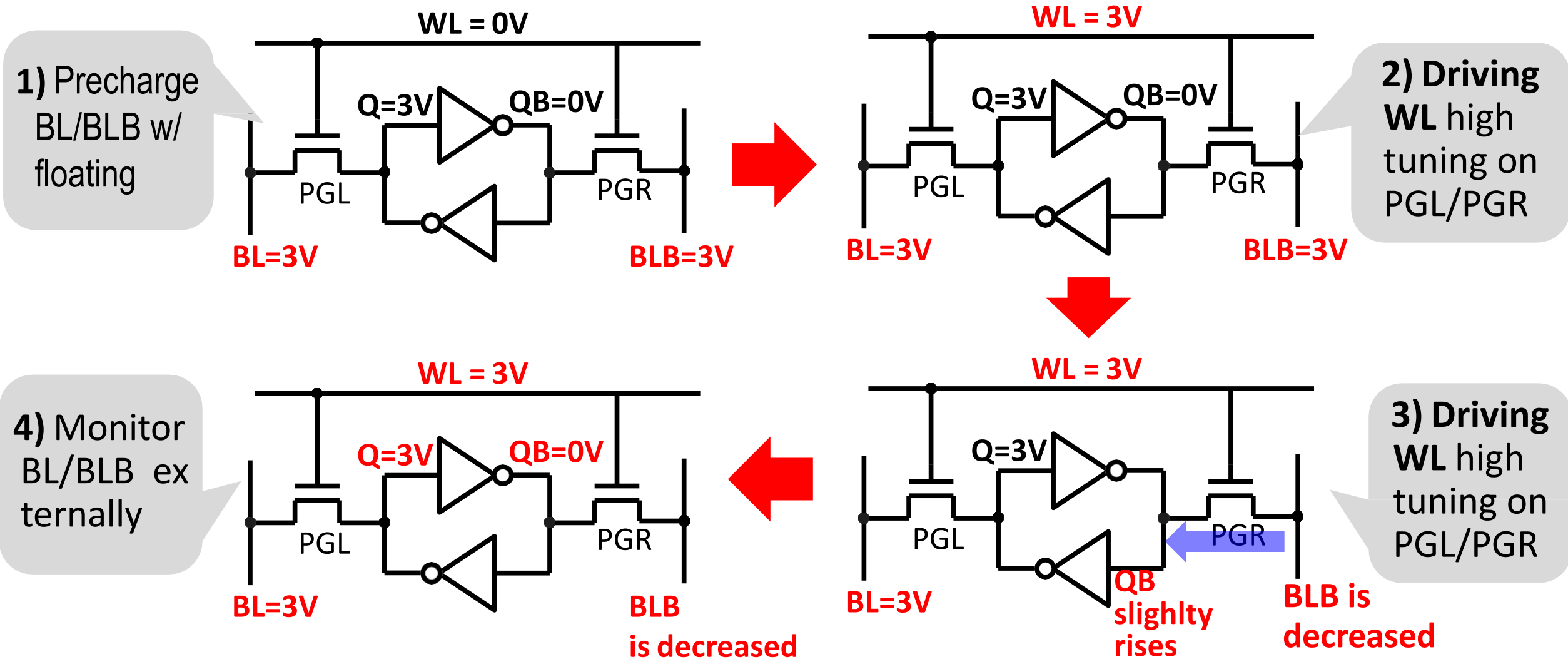
- Suppose that there is node Y with a capacitance as：



- Y can be charged to 3V in following two ways. What is difference?

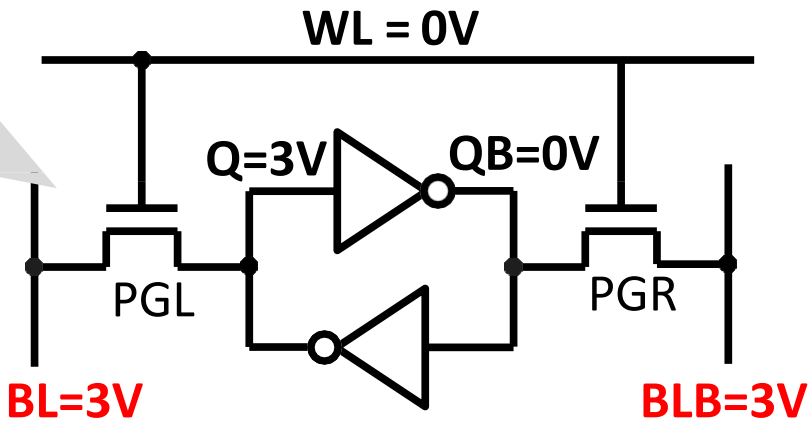


Read Operation

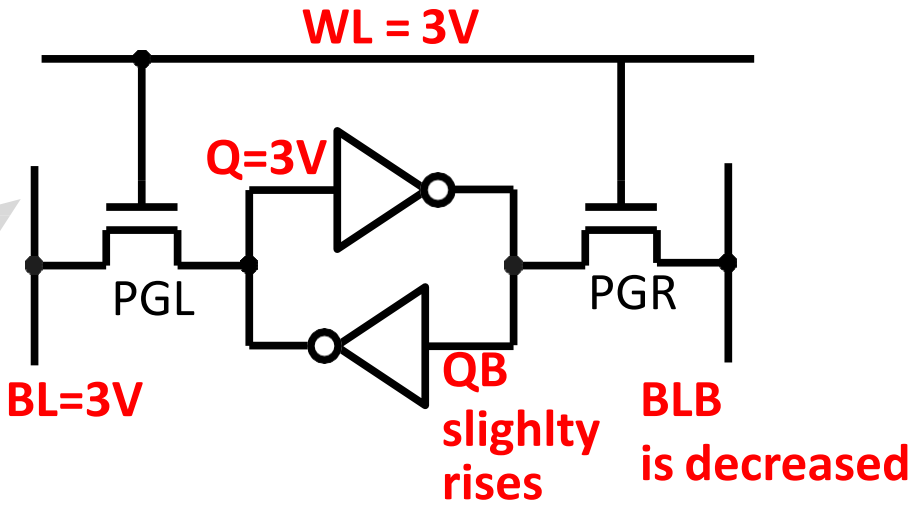


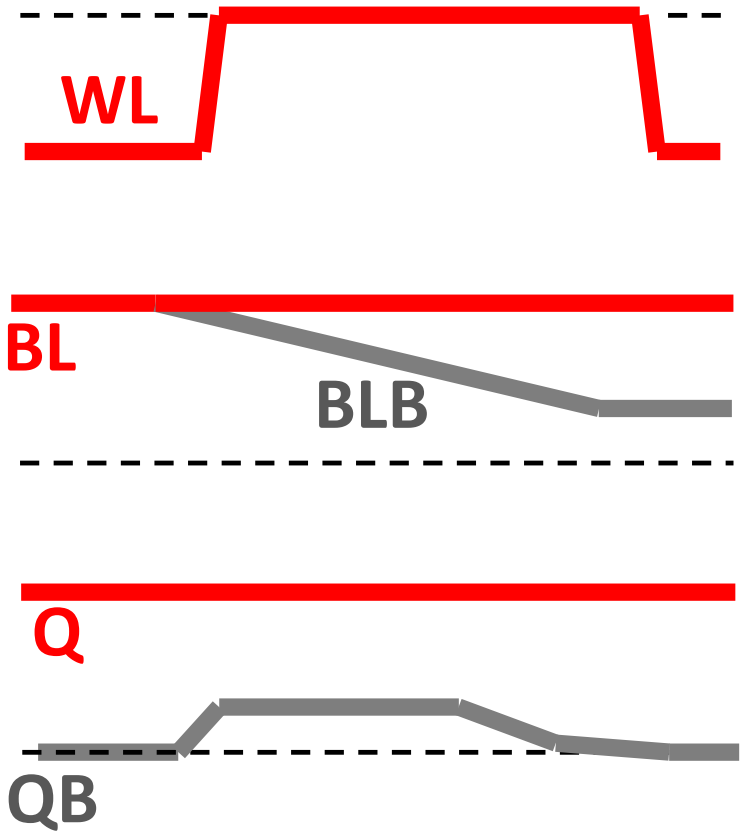
Read Operation Waveform

Hold
Data “1”



Monitor
BL/BLB
externally

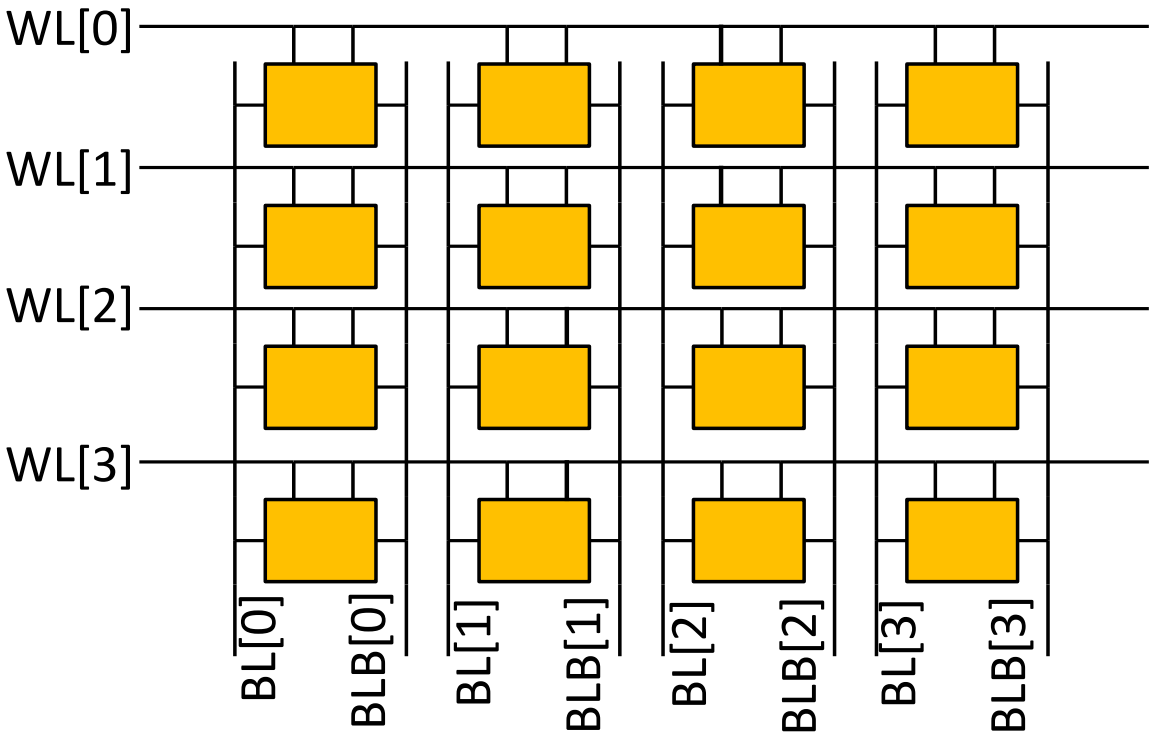


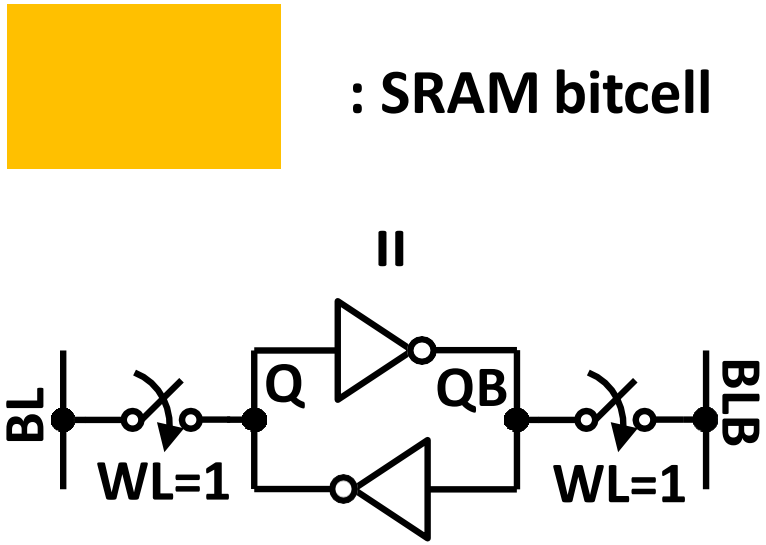


SRAM Array

- WLs & BLs are shared in the same row and same column, respectively.

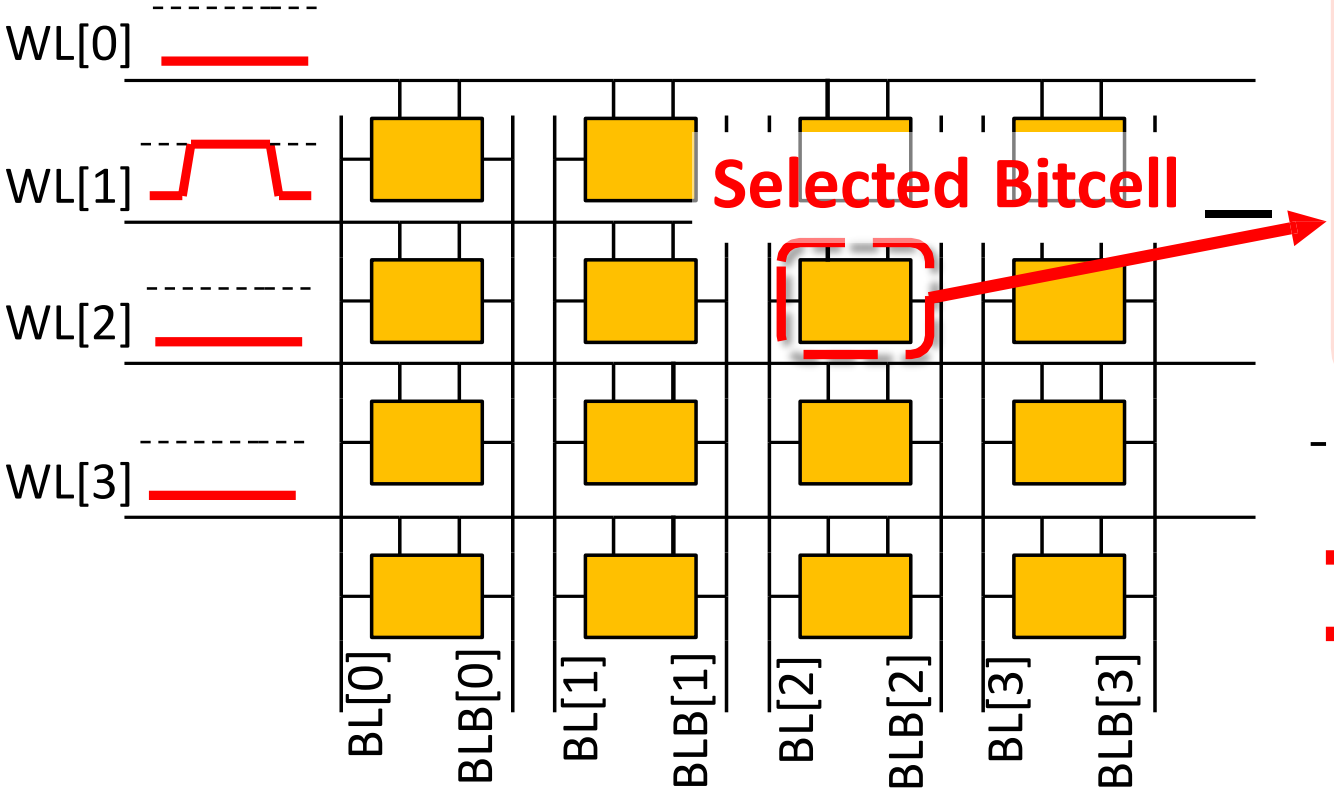
➔ How can you figure out memory state?

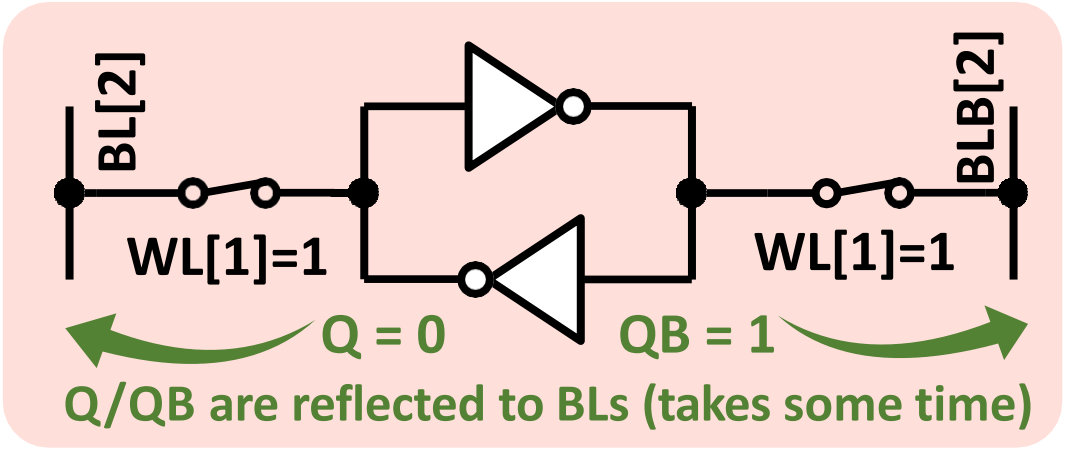


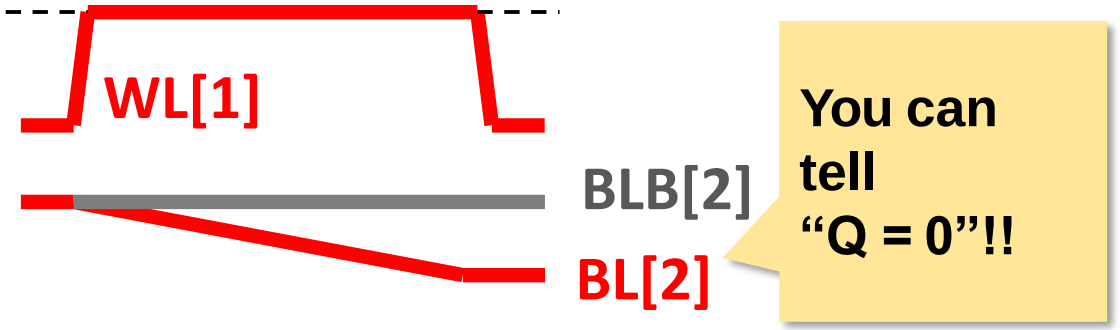


SRAM Read Access in Array

- BLs are precharged to “1” ➔ Driving **selected** WL = 1 ➔ Observing **selected** BL

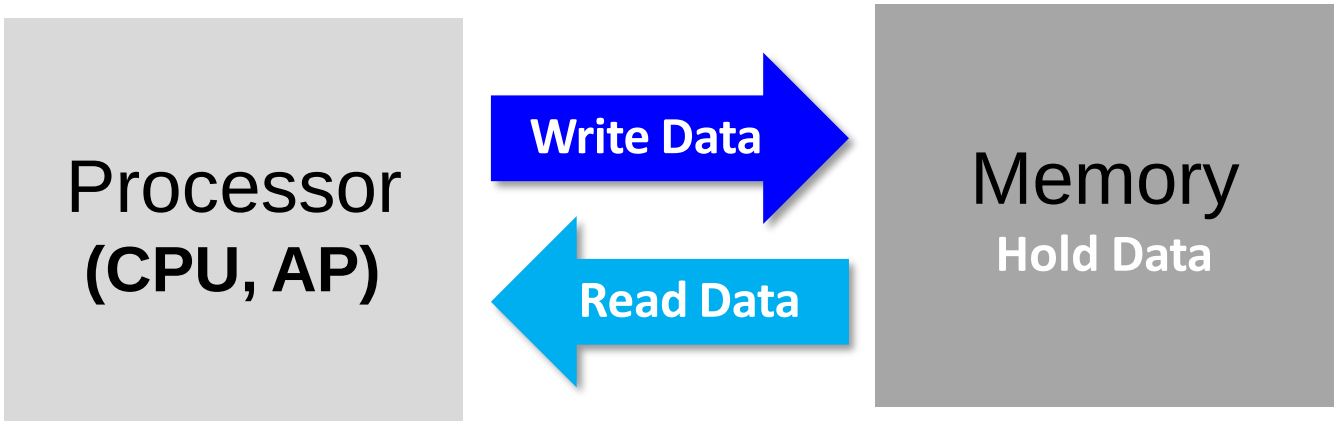






Revisit: Role of Memory

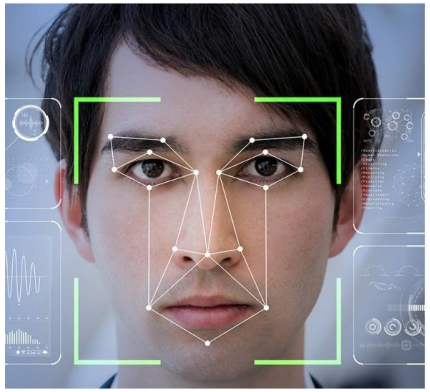
- Read : Processor sends address info. to memory and gets data from memory
- Write : Processor sends address info. & data to memory



Contents

- Backgrounds & Motivation
- Structure of SRAM
- Operation of SRAM
- **State-of-art SRAM Application; Computation in Memory**
- Conclusion

4IR with
Artificial Intelligence & Machine Learning



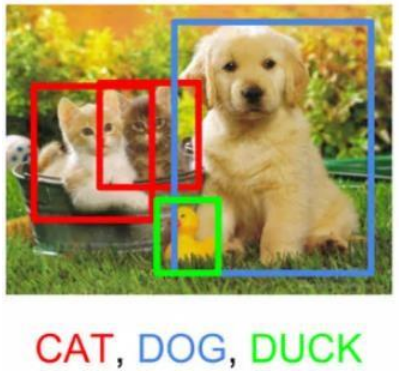
Face Recognition



Speech Recognition



Medical Diagnosis



Object Detection



Strategic Planning



Problem Detection

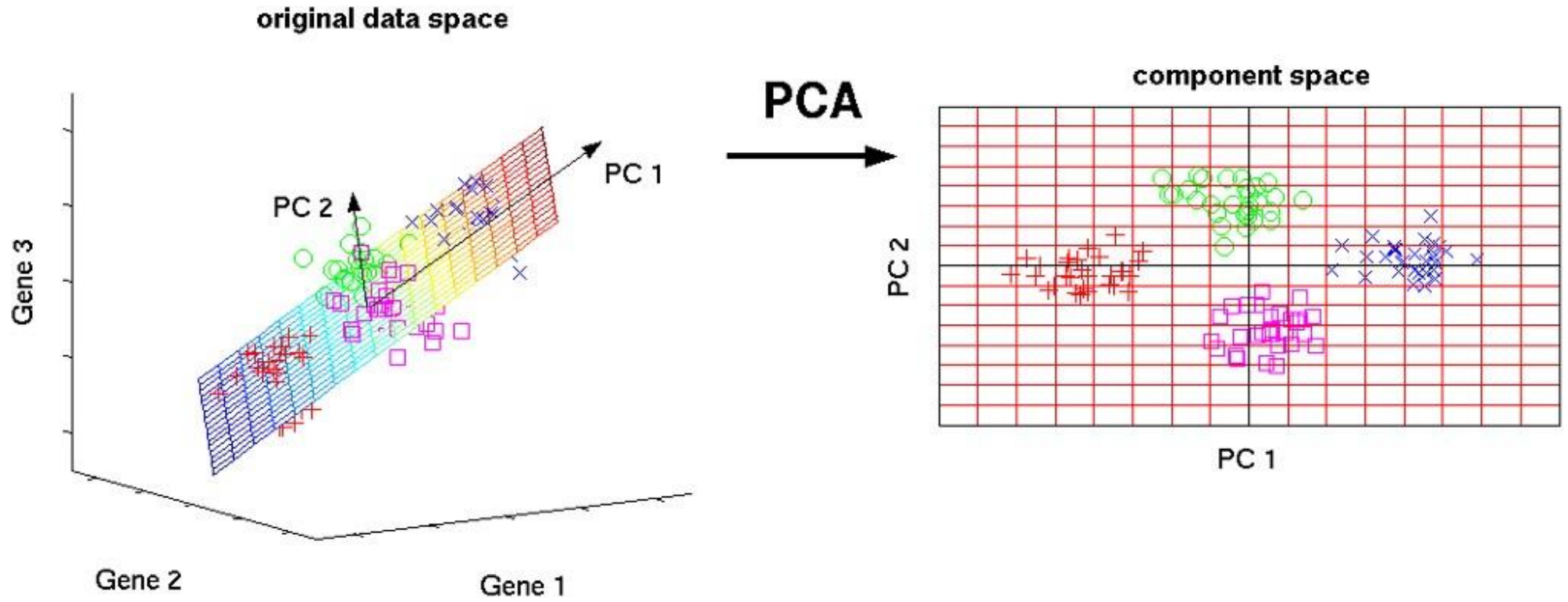
10 Common Machine Learning Algorithms

1. Principal Component Analysis
2. Linear Regression
3. K-Means Clustering
4. Logistic Regression
5. Support Vector Machine
6. Feed-Forward Neural Networks
7. Convolutional Neural Networks
8. Recurrent Neural Networks
9. Conditional Random Fields
10. Decision Trees

Source : https://towardsdatascience.com/top-10-machine-learning-algorithms-for-data-science-cdb0400a25f9

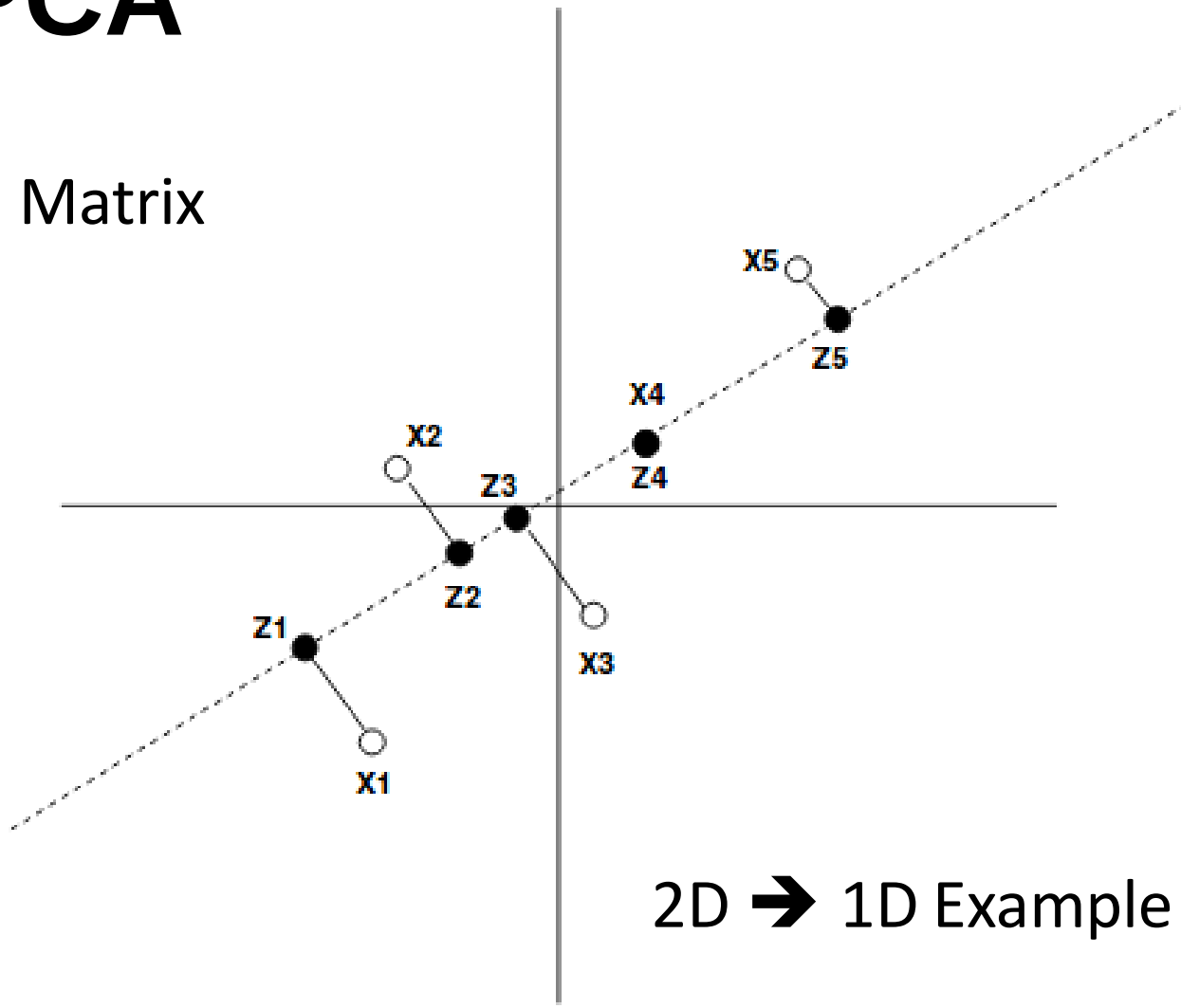
Principal Component Analysis (PCA)

- For dimension reduction ➔ Why do we use?



Key Operation of PCA

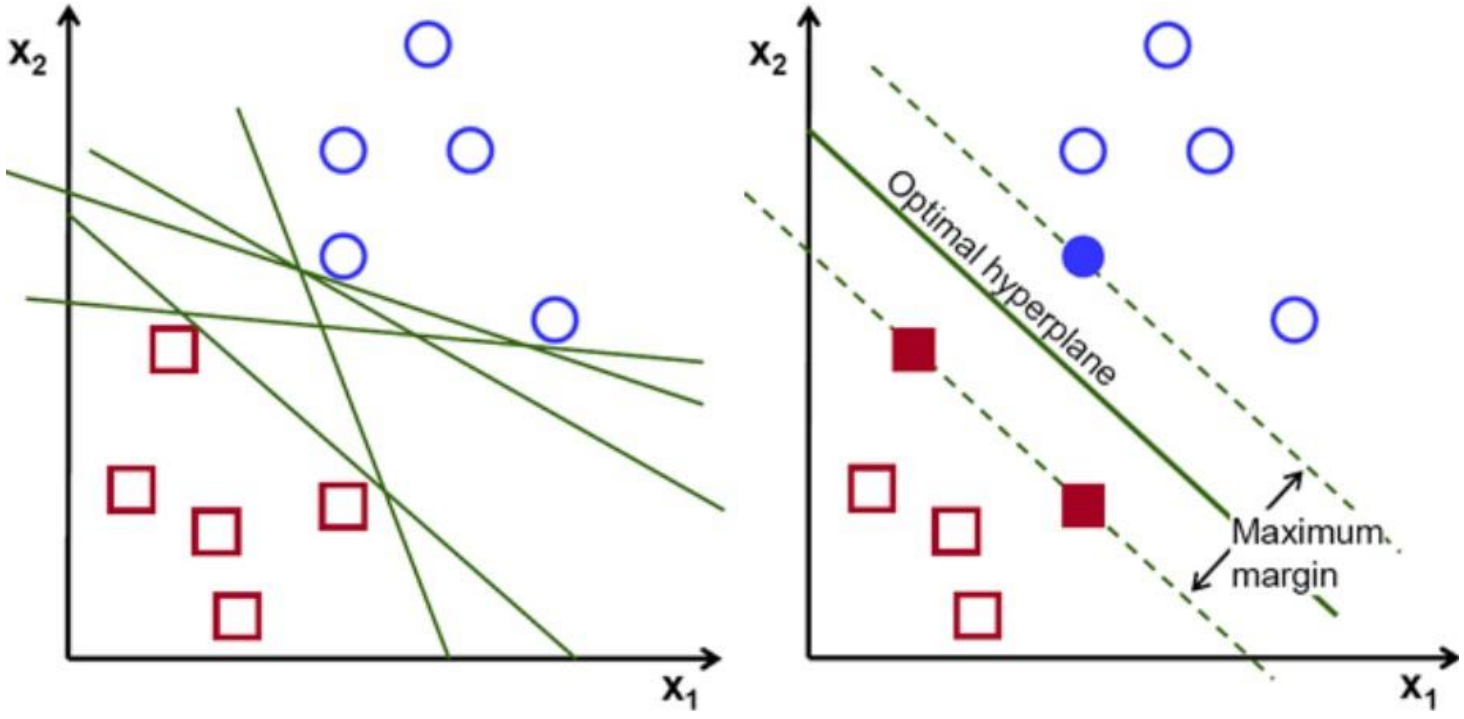
- Derivation of Eigen-value of Data Matrix
- Projection by matrix product
  - ➔ **Sum of product**



2D ➔ 1D Example

Support Vector Machine (SVM)

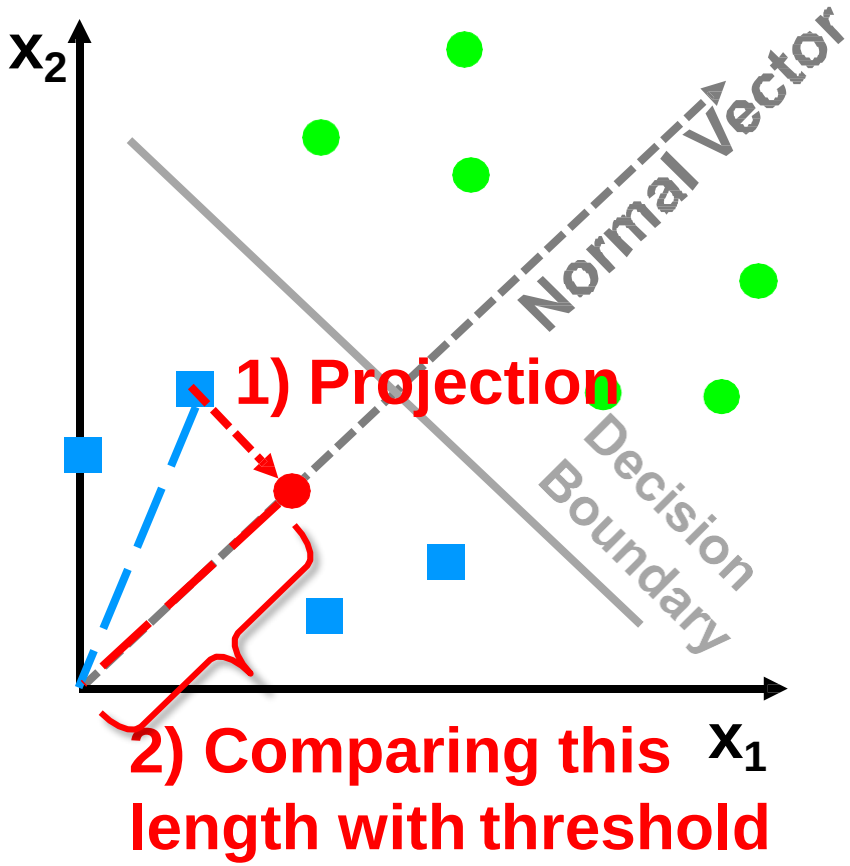
- Classification with best boundary (Optimal hyperplane)
- Face detection, Hand-writing recognition, Classification of image, etc.



Key Operation of SVM

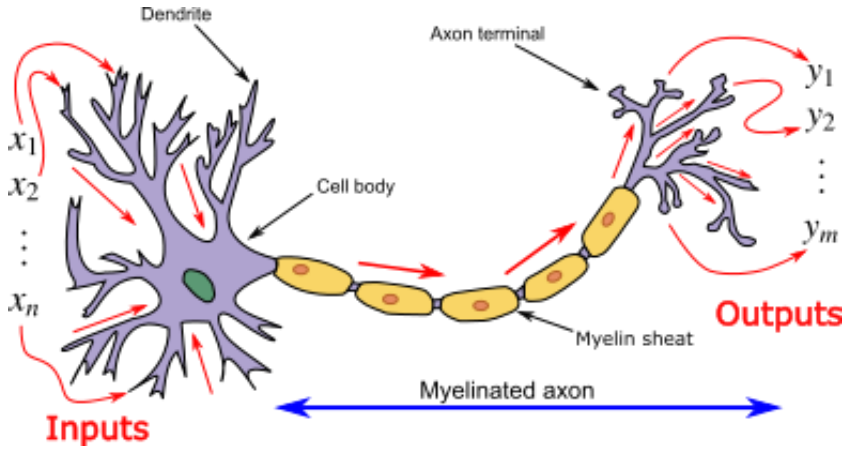
- Derivation of boundary hyperplane by optimization problem
- Classification Process:
  1) Project to normal vector (Matrix multiplication)
  2) Comparison the result with certain threshold

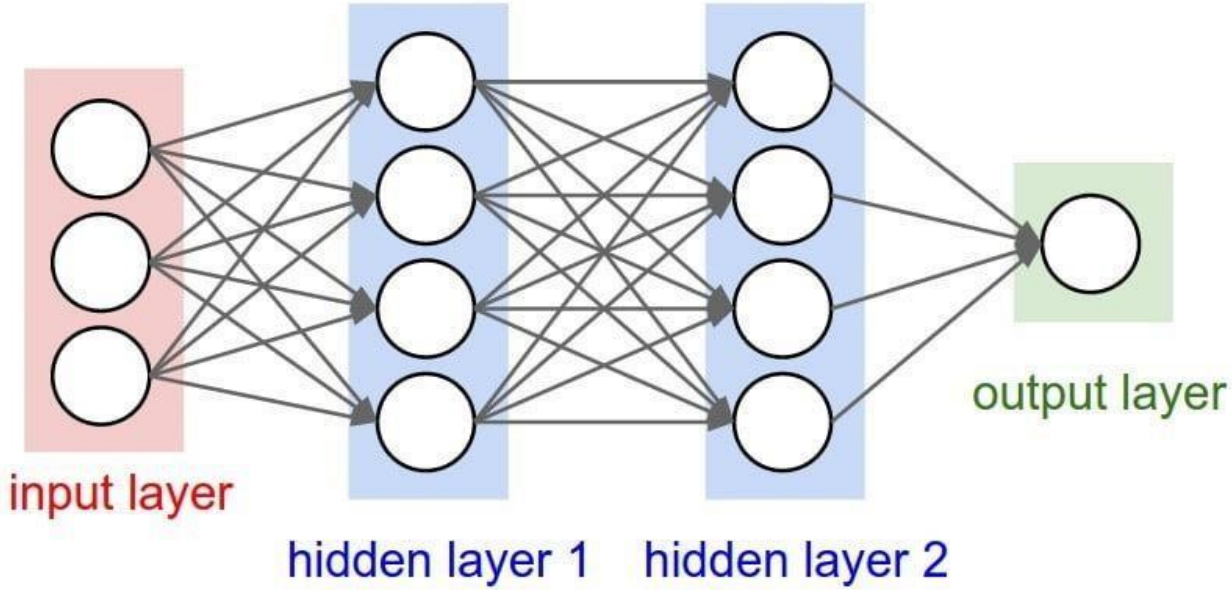
  ➔ **Sum of product** then **magnitude comparison**



Feed-Forward Neural Networks (FFNN)

- Mimicking the human brain structure

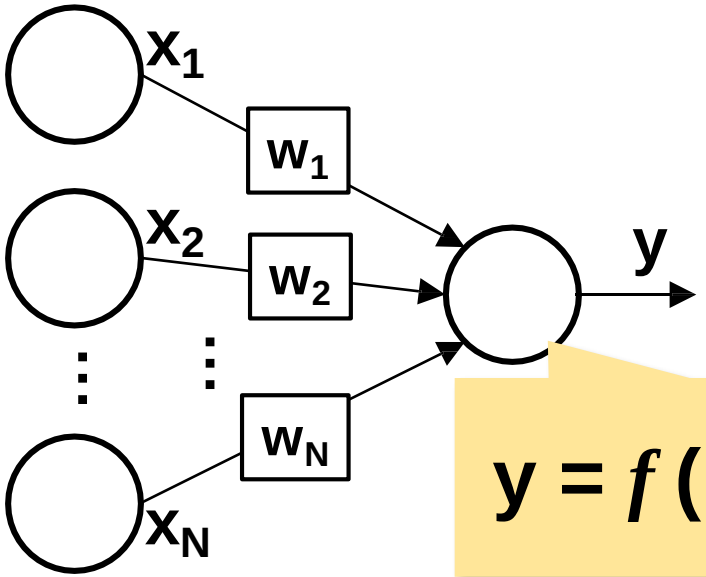




Key Operation of FFNN

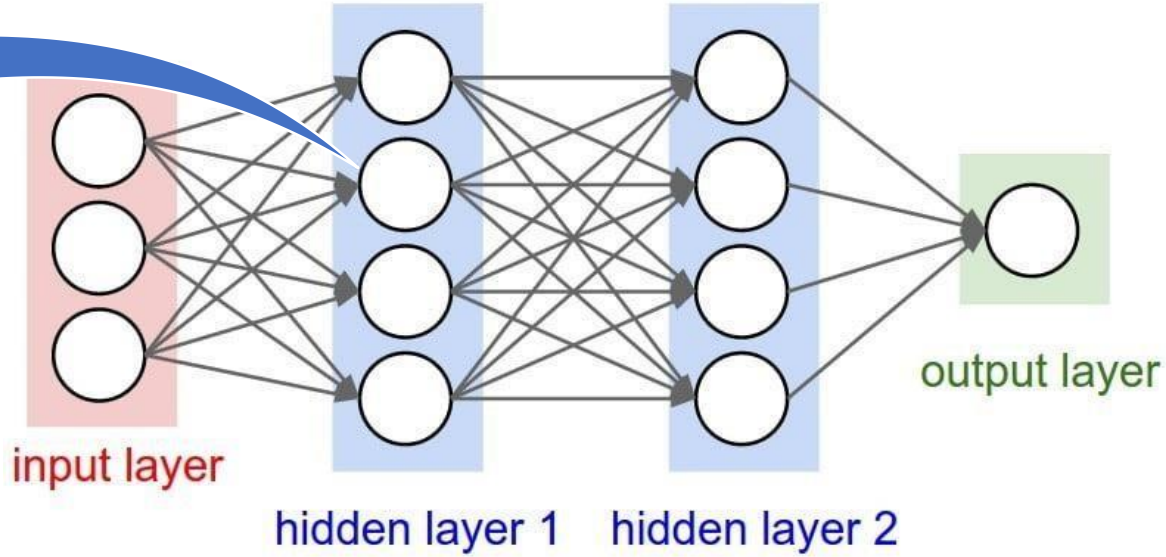
1) Derive the weighted sum of inputs (Matrix multiplication)
2) Applying a non-linear function *f* (sign or ReLU, usually)

➔ **Sum of product** then **magnitude comparison**



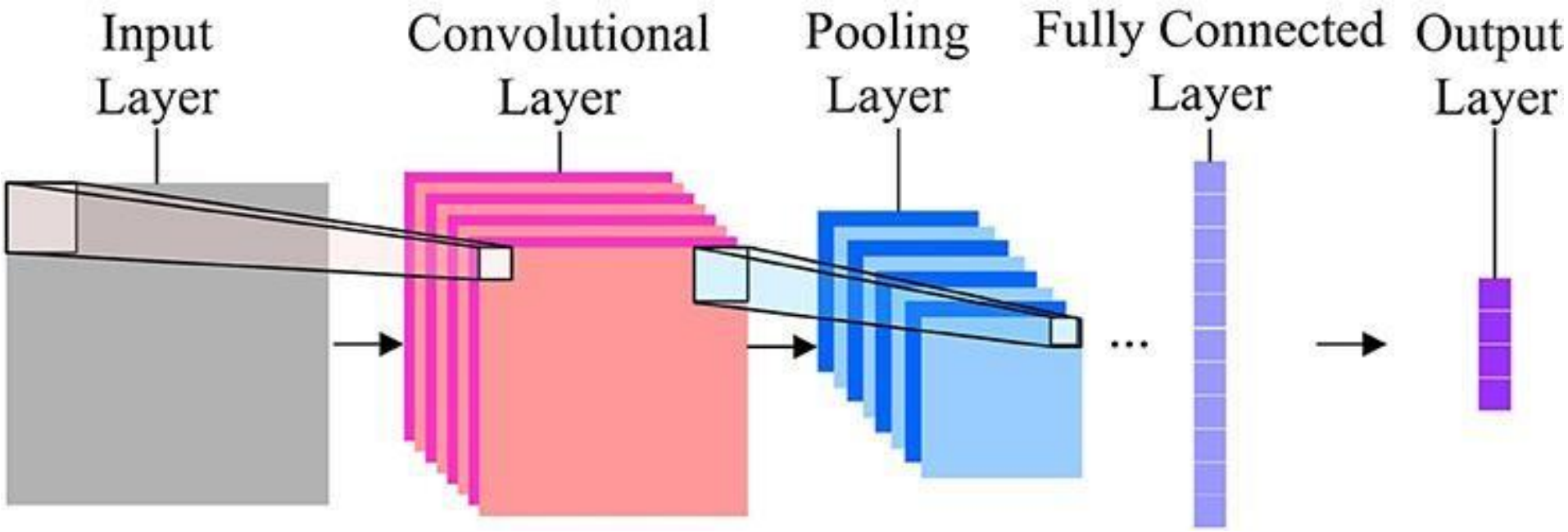
$$y = f\left(\Sigma w_k x_k\right)$$

f : **Non-linear function**



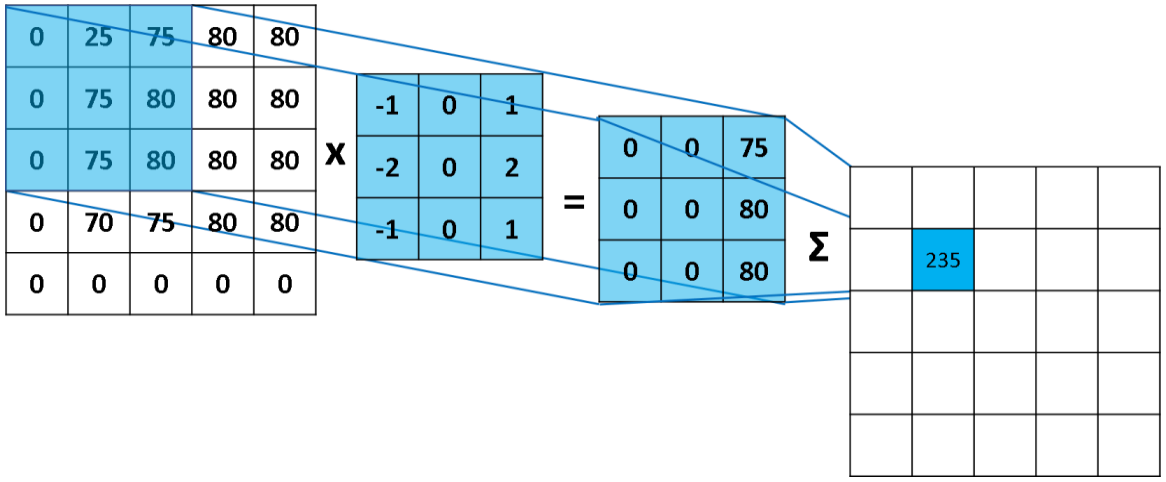
Convolutional Neural Networks (CNN)

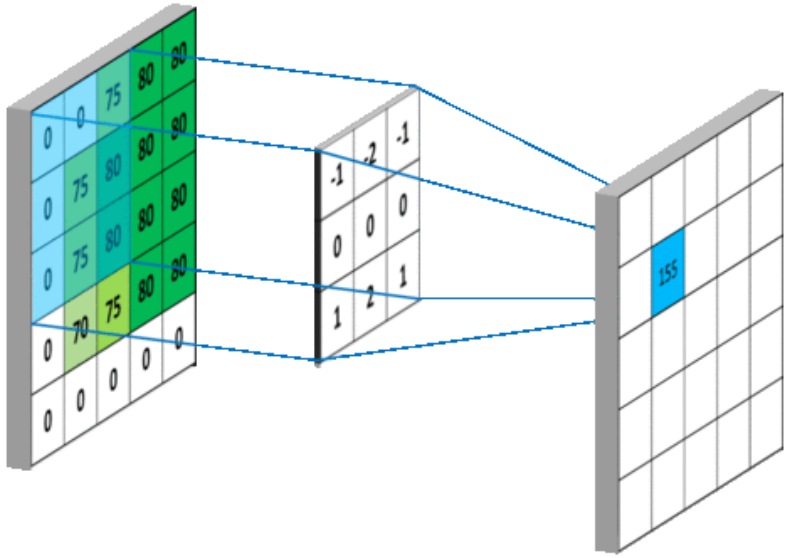
- Space information is conserved ➔ Useful for analyzing visual imagery



Convolutional Layer Operation in CNN

- Key operation: repetition of **Sum of products**
- Training: updating weights for maximizing accuracy





source: https://mlnotebook.github.io/post/CNN1

Machine Learning at Edge

Cloud

- More Complex and Deeper Algorithm
- Both Inference and Training



Edge

- Simpler and Shallower Algorithm
- Localized interference

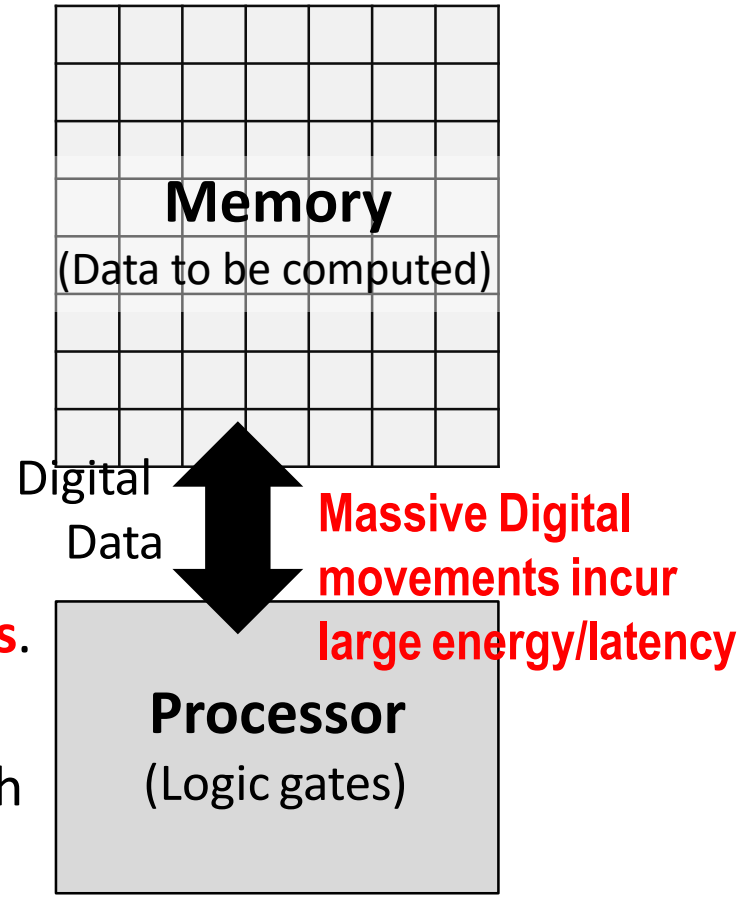


Desirable to achieve

- ✓ Low energy consumption
- ✓ Fast (real-time) processing
- ✓ Less communication with cloud

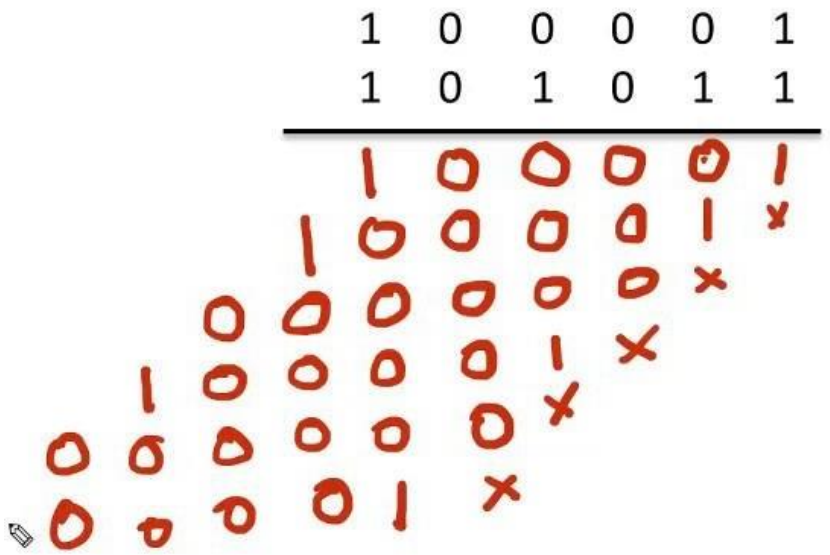
What Happens in Hardware for Calculating Sum of Product

- To calculate $\Sigma w_k x_k$, w & x should be expressed digitally.
  - Ex) if w = 13 ➔ w = 1101 for 4bit expression.
- Suppose that 4bits are allocated to express w & x:
  1) w & x are stored in memory (**4 bitcells** are required for each)
  2) w and x should be read from the memory and **moved to** processing unit.
  3) w and x should be applied to the 4bit multiplier in which **multiplication is performed by a number of logic gate stages**.
  4) 1)-3) are **repeated** to for various k values **for k cycles**
  5) N multiplication results are accumulated by 4bit adders which is also **addition is performed by a number of logic gates**.



Digital processing requires multiple logic stage operations

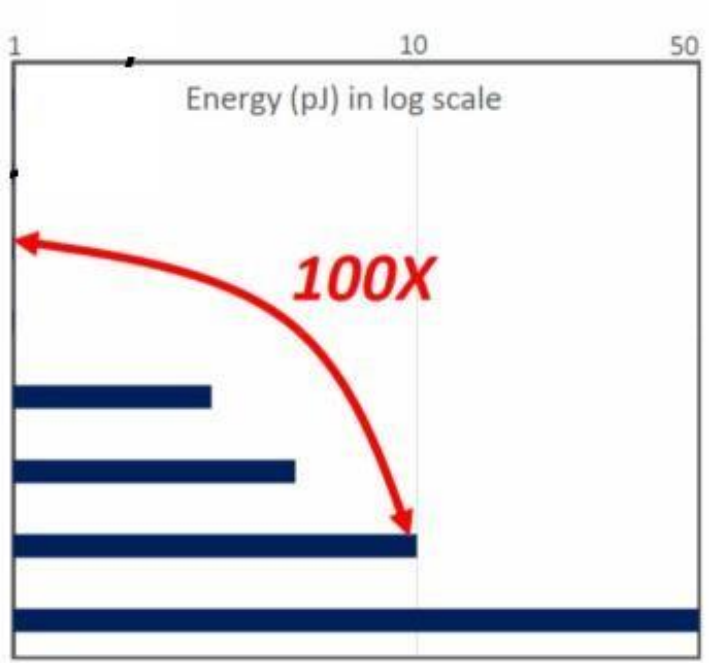
How Multiplication is Performed in Processor



Dominating Energy/Latency

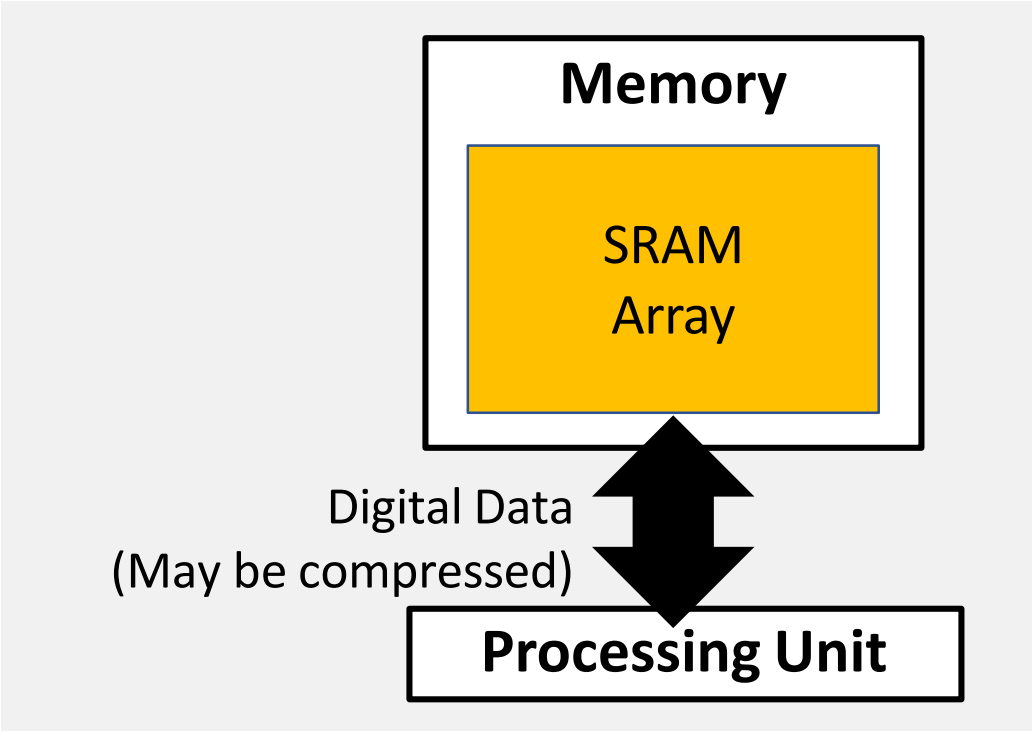
- Memory access dominates overall energy/latency, especially in ML applications

Operation	Energy (pJ)
Integer ADD (8b)	0.03
Integer ADD (16b)	0.05
Integer ADD (32b)	0.1
Integer MULT (8b)	0.2
Integer MULT (32b)	3.1
8KB SRAM Read (32b)	5
32KB SRAM Read (32b)	10
1MB SRAM Read (32b)	50



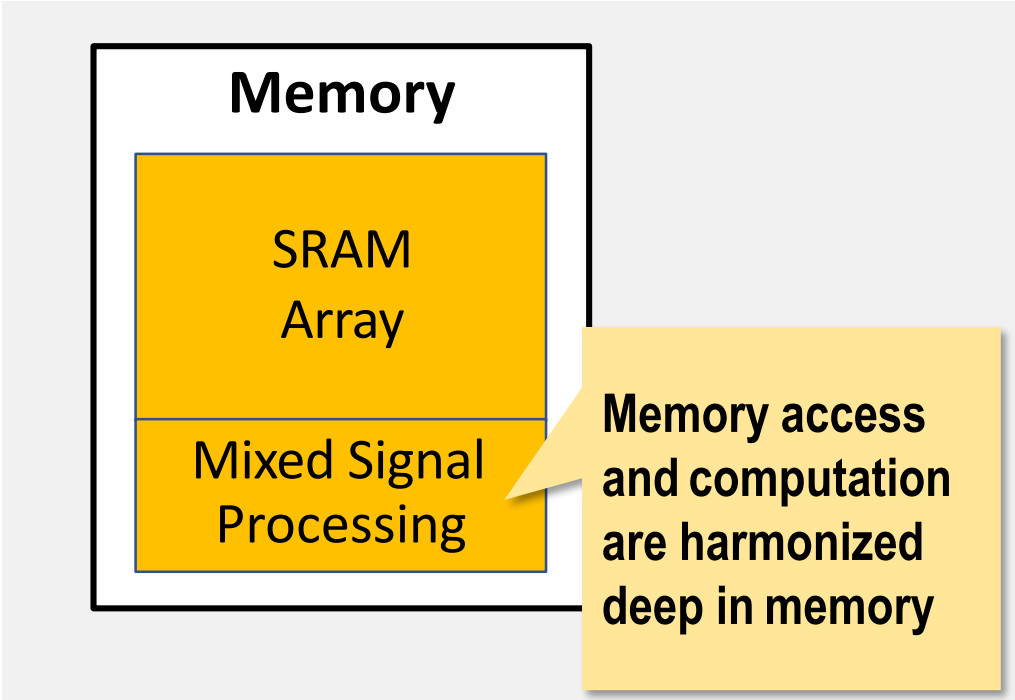
We can Say CiM as

Conventional H/W for ML



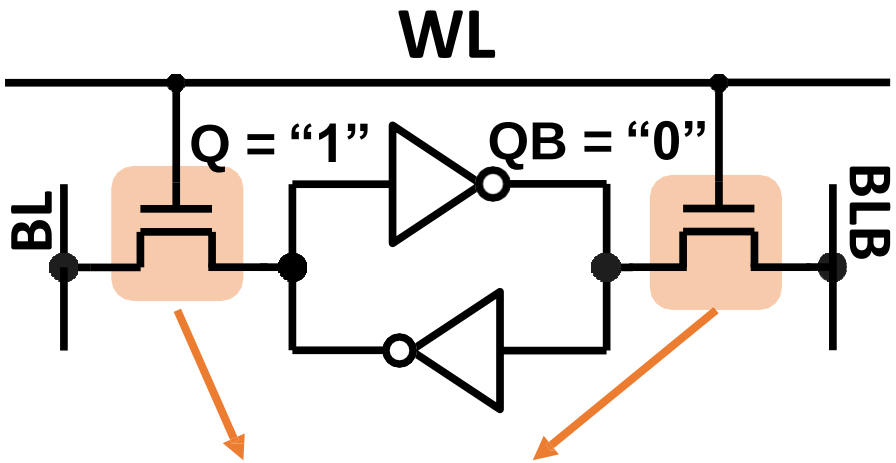
✓Energy/latency are do minated by data access

Computation in Memory



✓ Energy/latency are significantly reduced

BL level vs. WL level when Q = 1 in Read Operation

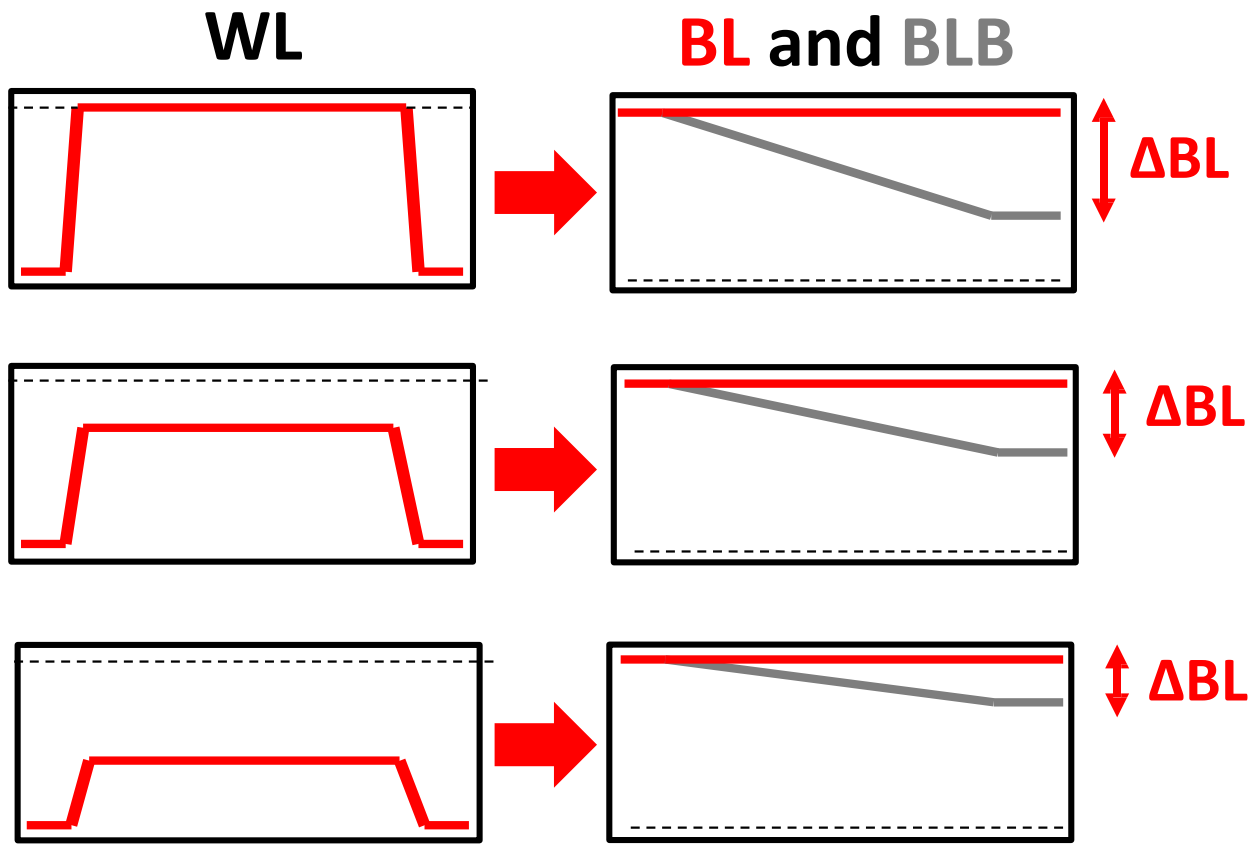


Current drivability is de termined by WL level

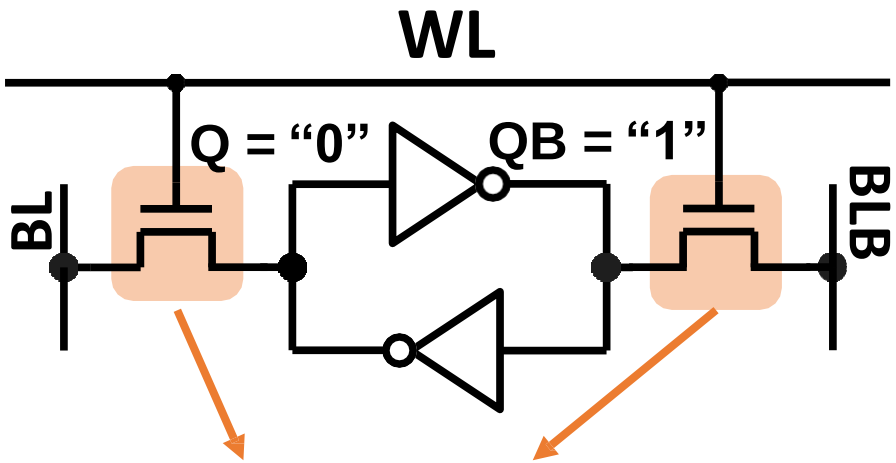
WL ↑ → Current ↑

→ BLs change faster

→ Final ΔBL = (BL-BLB)↑



BL level vs. WL level when Q = 0 in Read Operation

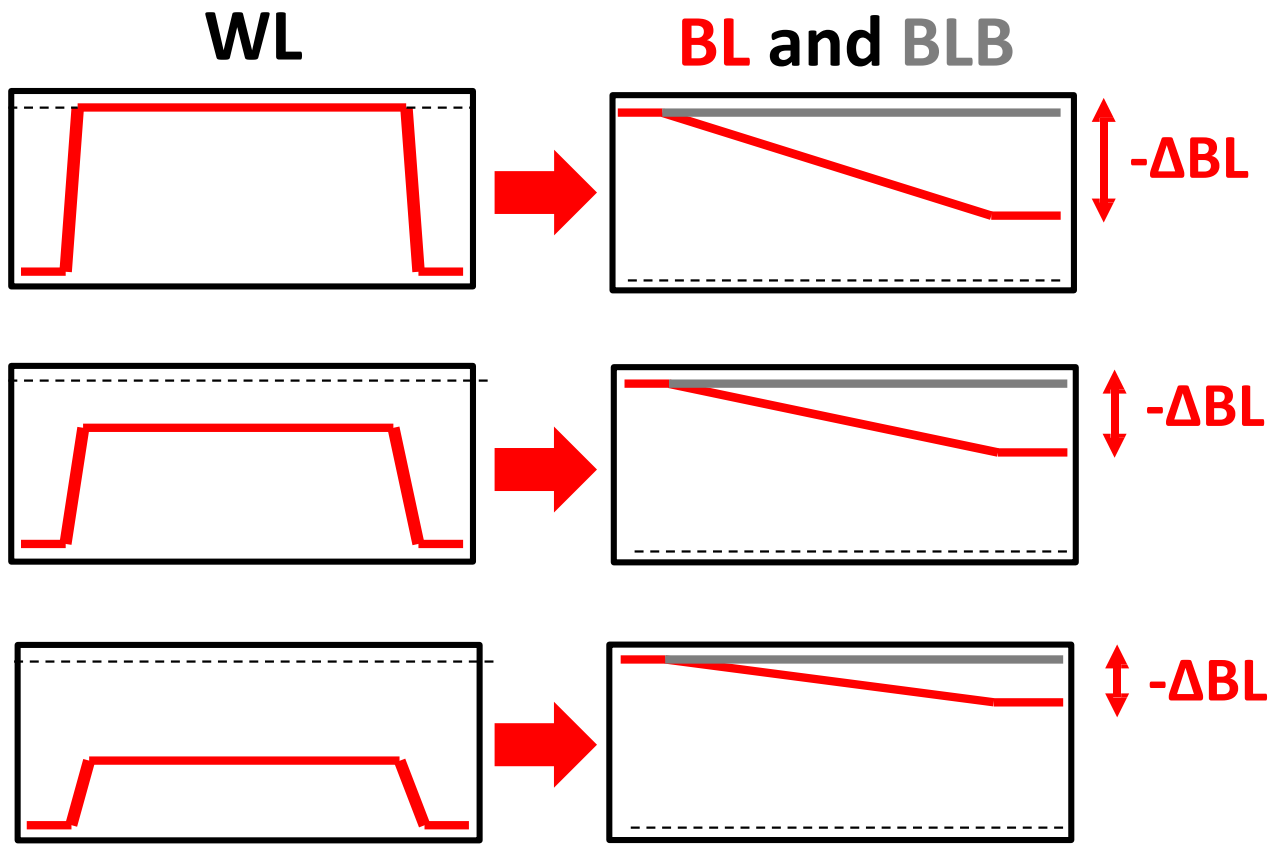


Current drivability is de termined by WL level

WL ↑ → Current ↑

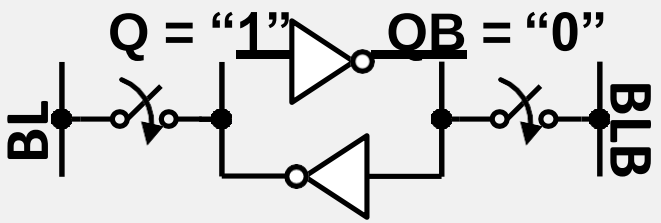
→ BLs change faster

→ Final -ΔBL = (BLB-BL)↑

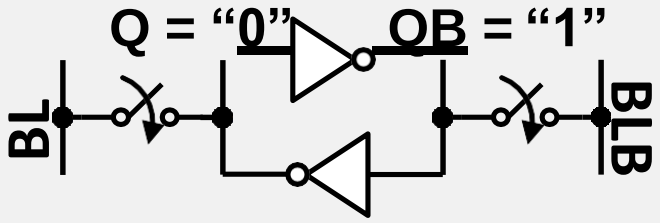


Implementing ΔBL as $p_k = w_k \times x_k$

Q is mapped to w_k



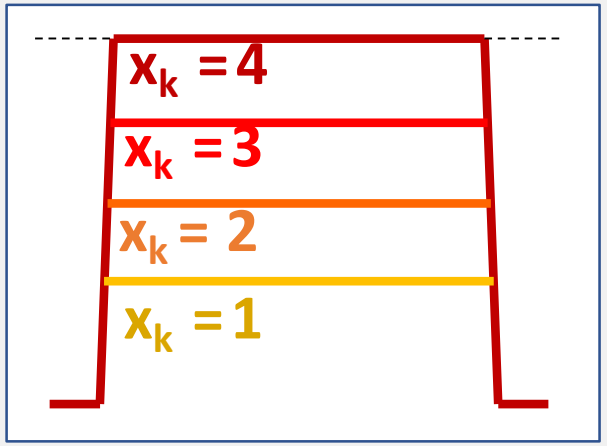
Mapped to $w_k = +1$



Mapped to $w_k = -1$

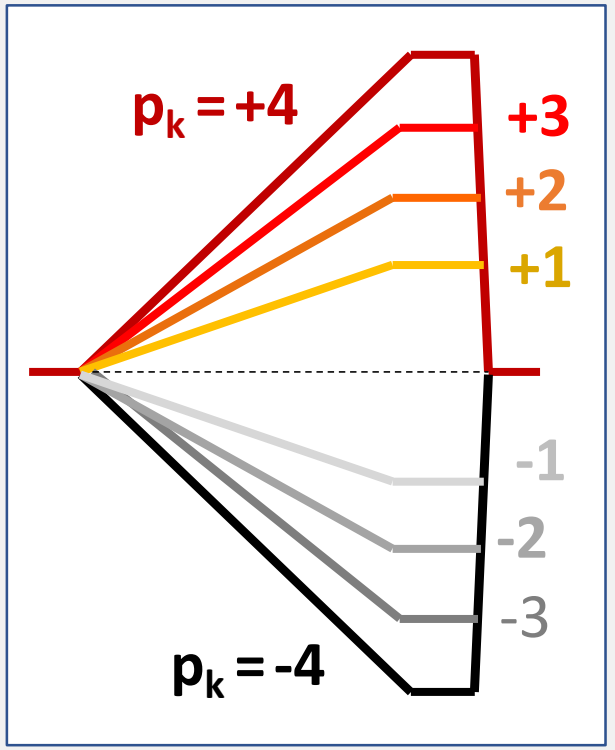
×

WL level is mapped to x_k



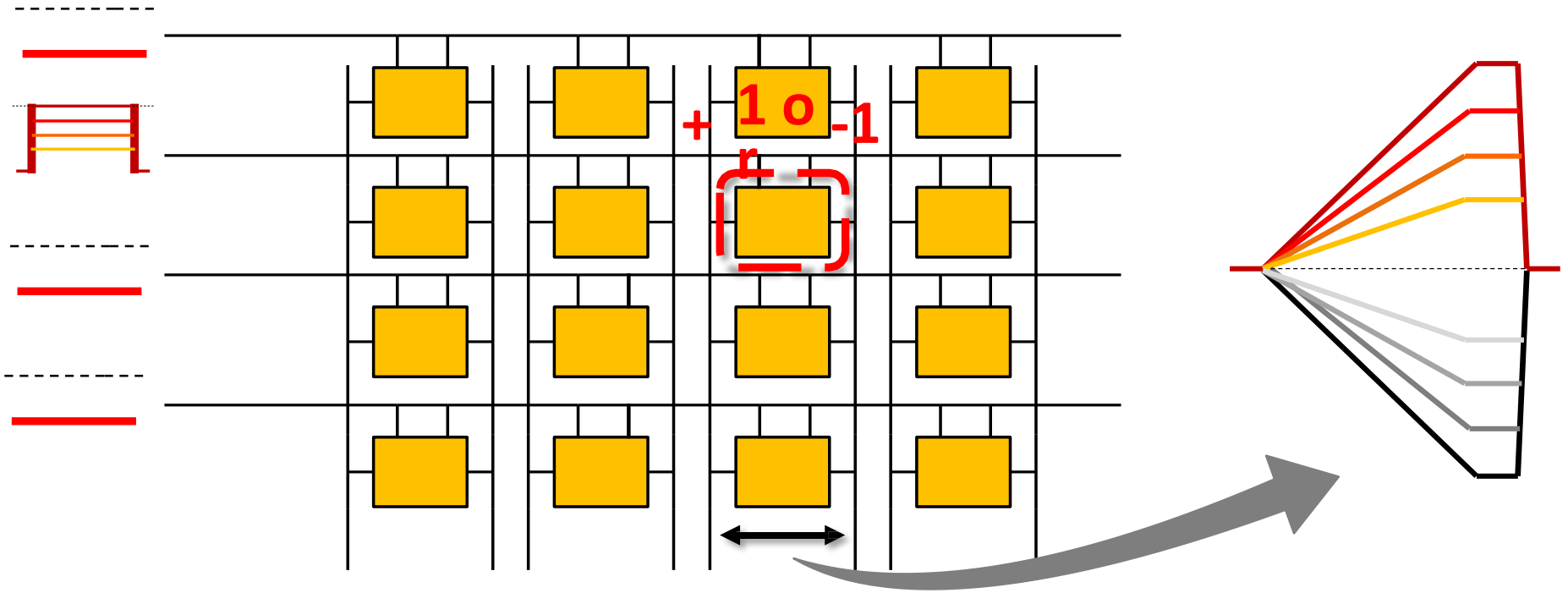
=

ΔBL is mapped to p_k



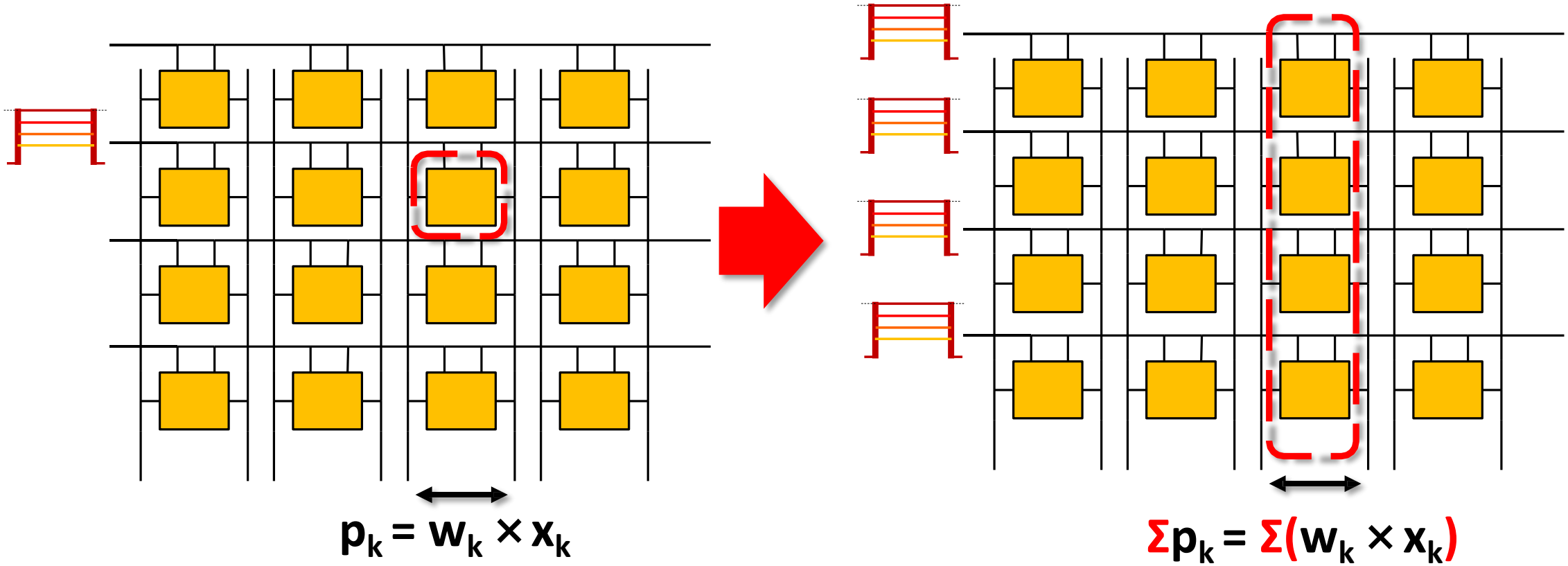
Revisiting the Array

- By observing ΔBL, we can obtain $p_k = w_k \times x_k$



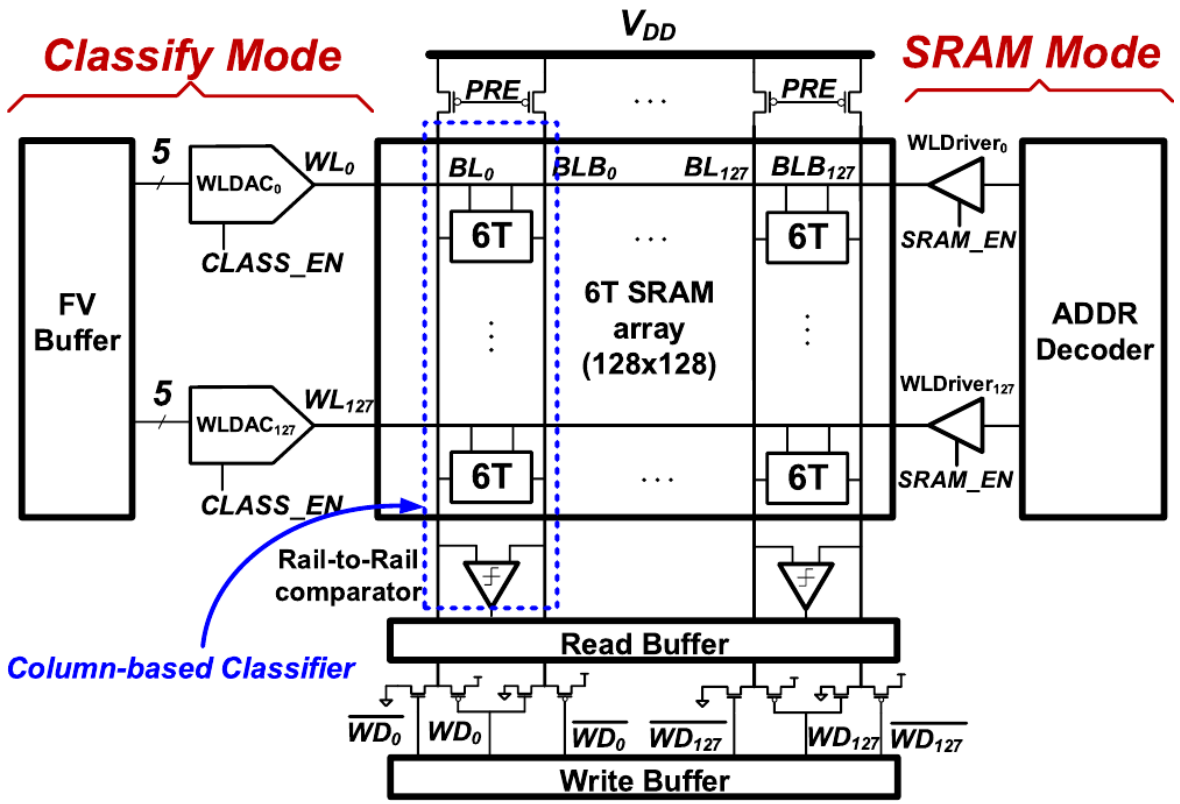
Extension!

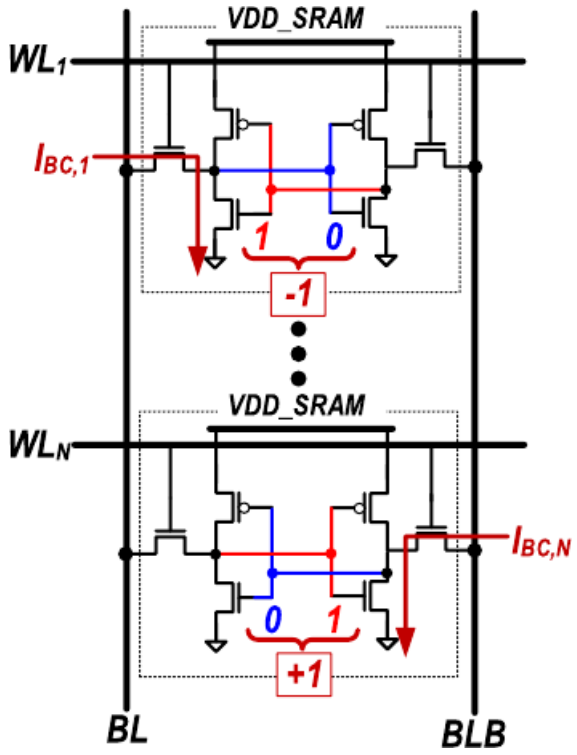
$p_k = w_k \times x_k$ ➔ $\Sigma p_k = \Sigma(w_k \times x_k)$



J. Zhang, JSSC 2017: Overall Structure

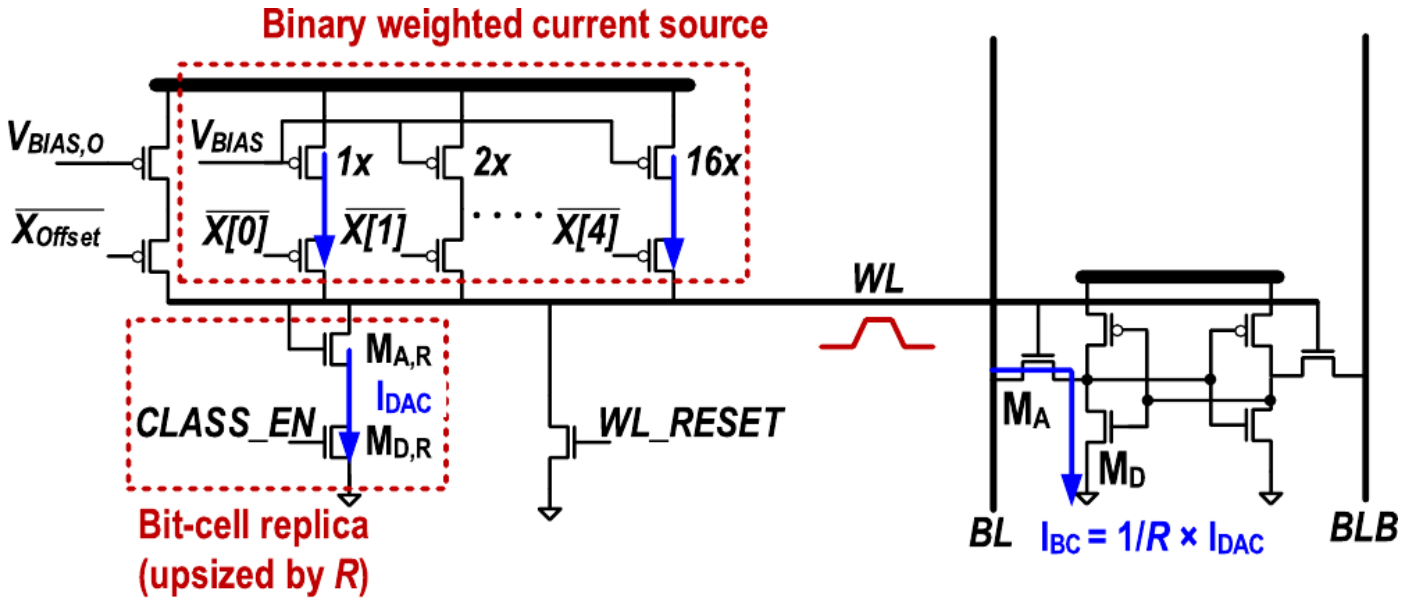
- Comparators at the BL end play a role of sign function for classification

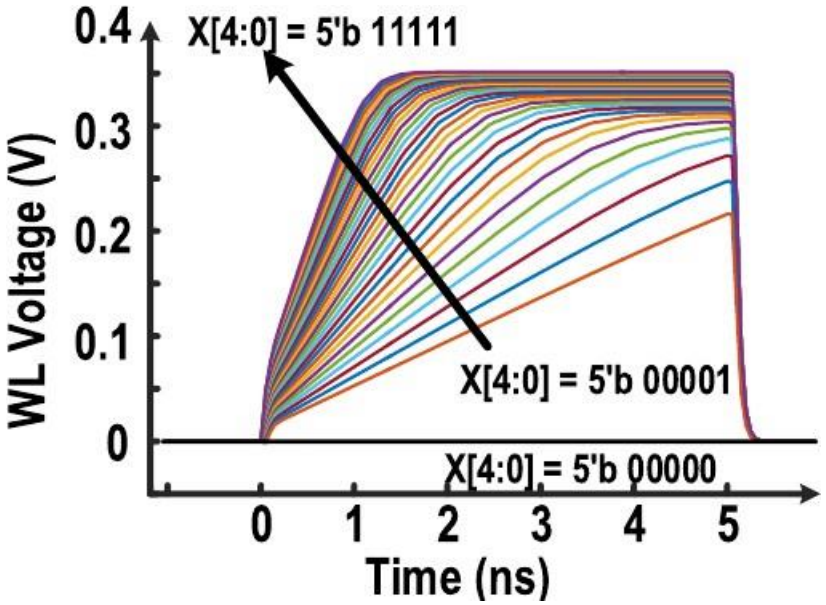




J. Zhang, JSSC 2017: WLDAC

- Various levels of WL are generated by WLDAC circuit shown below.





Limitation by Binary w_k?

- w_k can have only two values : +1 or -1. Is it okay?
- According to M. Rastegari et al. in Arxiv '16, due to error resiliency,

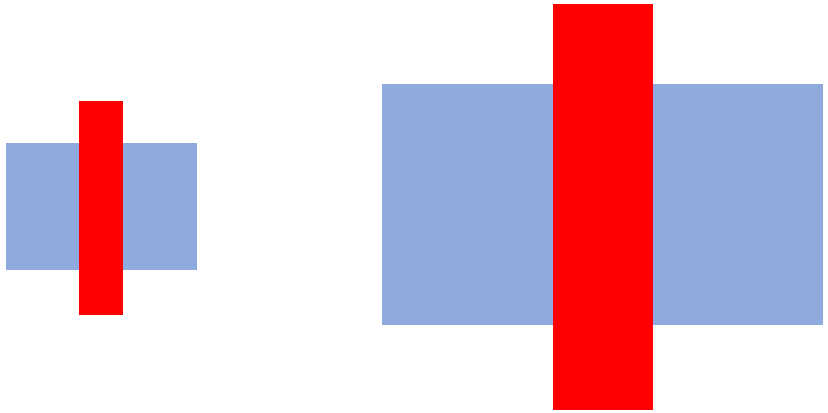
	Weight Array	Memory Savings	Classification Accuracy (AlexNet)
Standard Convolution	Real value	1x	80.2%
Binary Weight Convolution	Binary value (-1, +1)	**32x**	**79.4%**

$$Y_k = \sum_i W_{k,i} \times X_i = \alpha_k \sum_i \overline{w_{k,i}} \times X_i$$

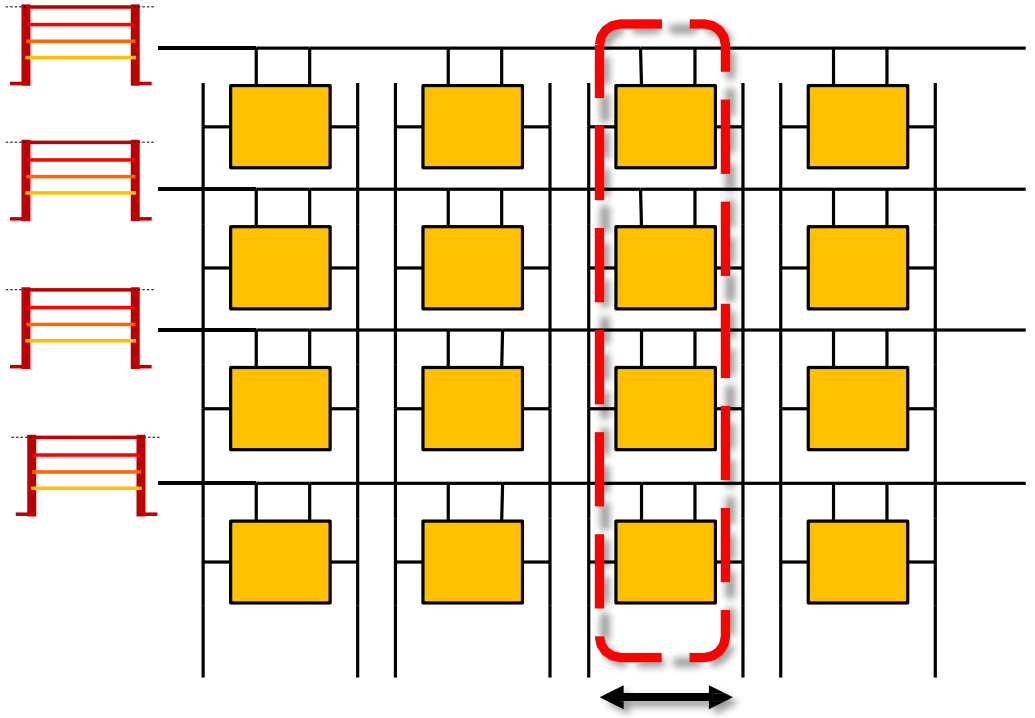
+1 or -1

Effect of Circuit Variation

- Consider the two transistors below. Which one is more susceptible to variation?



- SRAM bitcell are implemented using near minimum sized TRs and the design of WL driver is also constrained by the area.

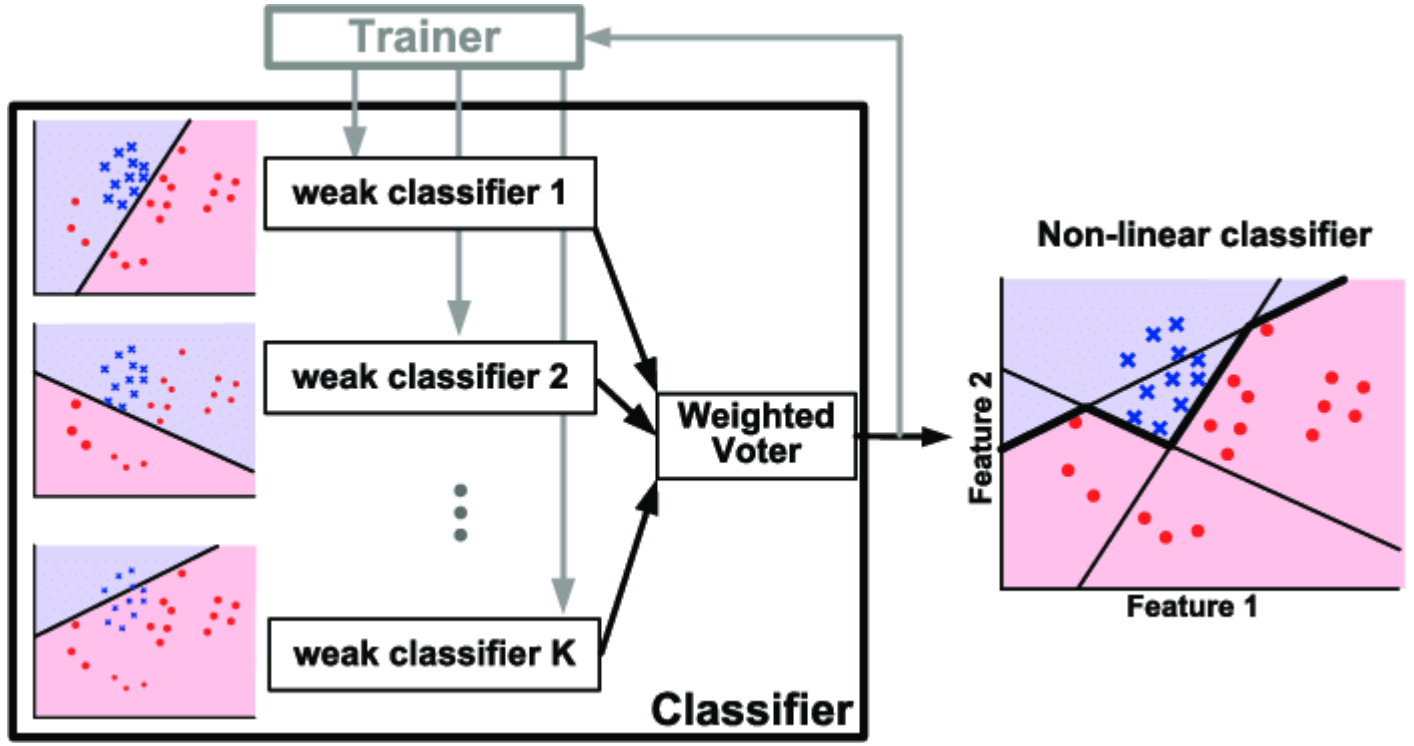


$$\Sigma p_k = \Sigma w_k \times x_k$$

➔ **WL and BL voltages are highly sensitive to variation, which limits classification accuracy.**

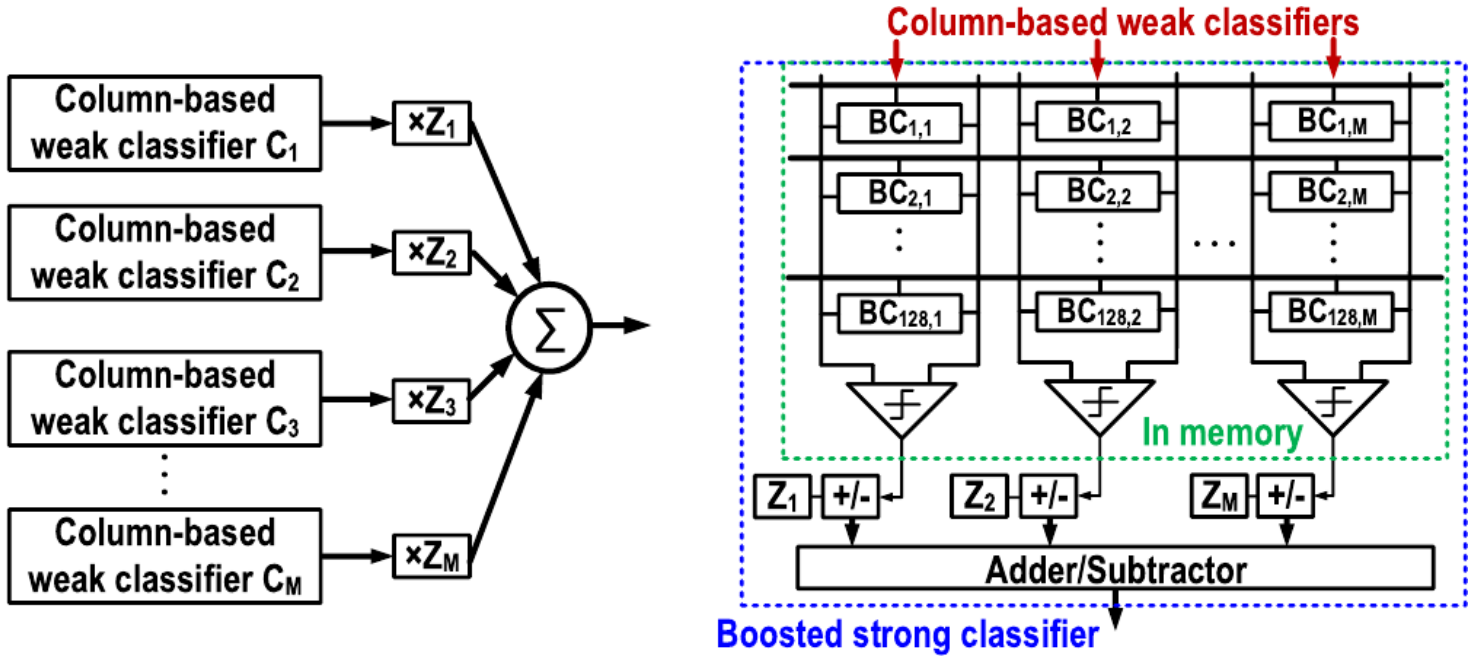
Applying Boosting

- By R. E. Schapire and Y. Freund, “Boosting : Foundations and Algorithms”, combining multiple weak classifiers (low accuracy) can establish a strong classifier :



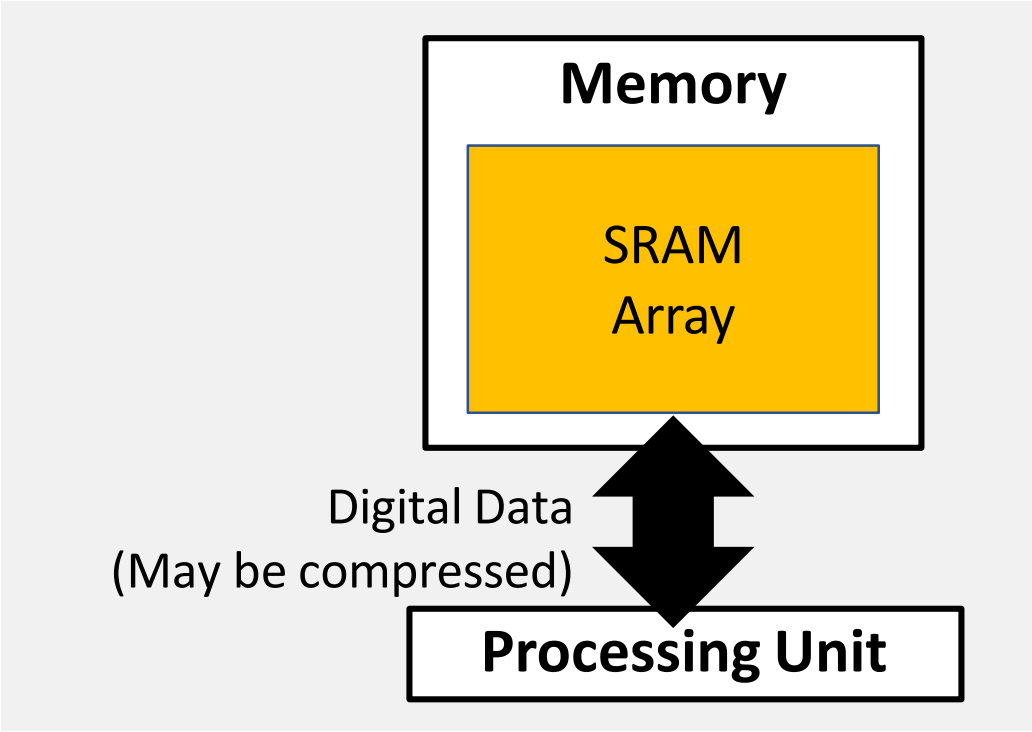
J. Zhang, JSSC 2017: Boosting

- 96% accuracy w/o circuit variation with real valued weight
  ➔ 91% accuracy with circuit variation with binary weight



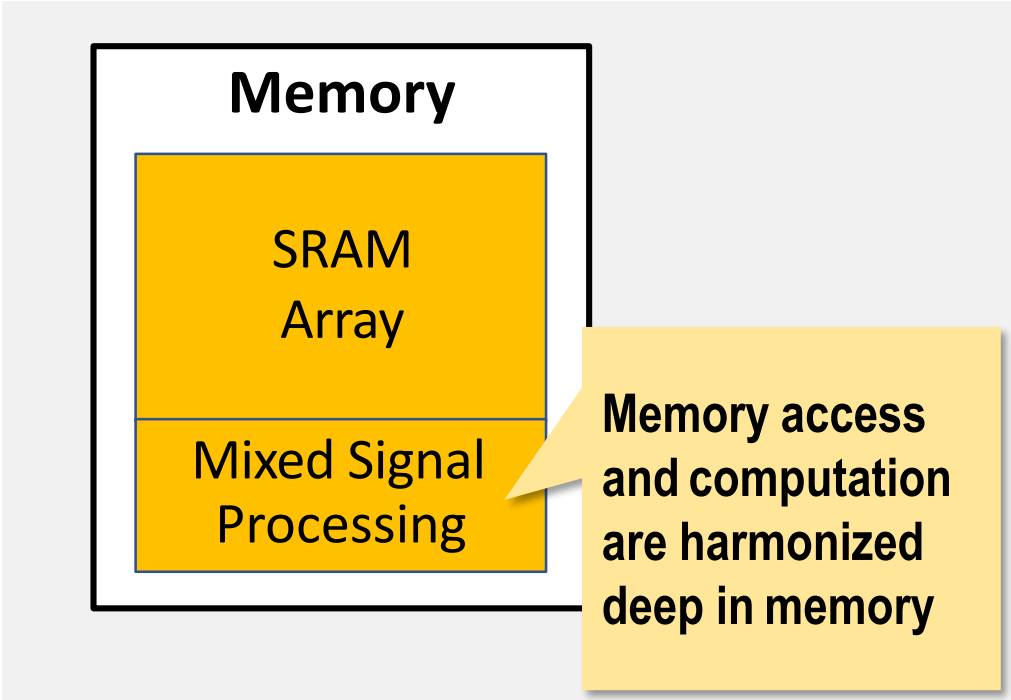
Revisiting the Concept of CiM

Conventional H/W for ML



✓Energy/latency are do minated by data access

Computation in Memory

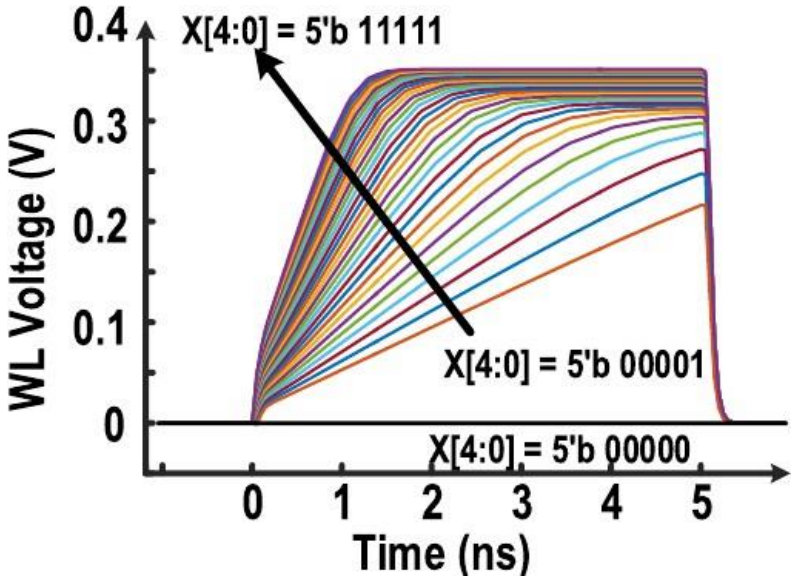


✓ Energy/latency are significantly reduced

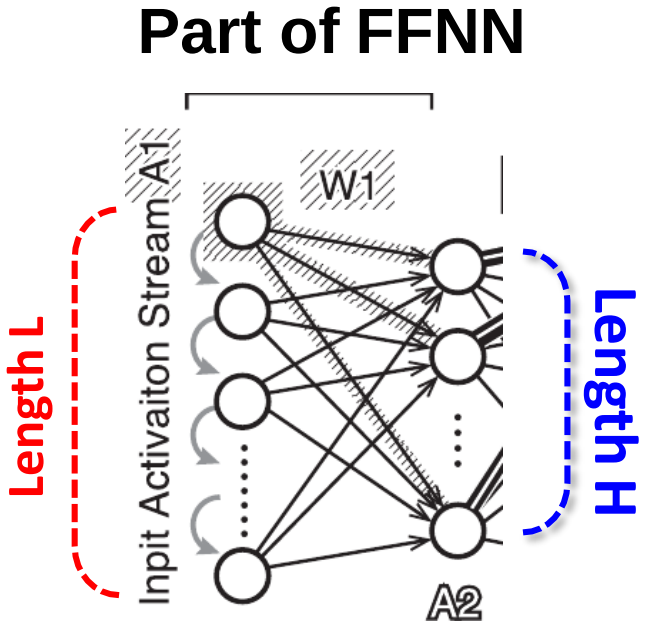
Issue of WL Variation

- WL = 400mV should express 32 different input levels. Even small change in WL by variation can result in an error.
- For some applications, inputs do not need to express wide bit width (e.g, handwriting).

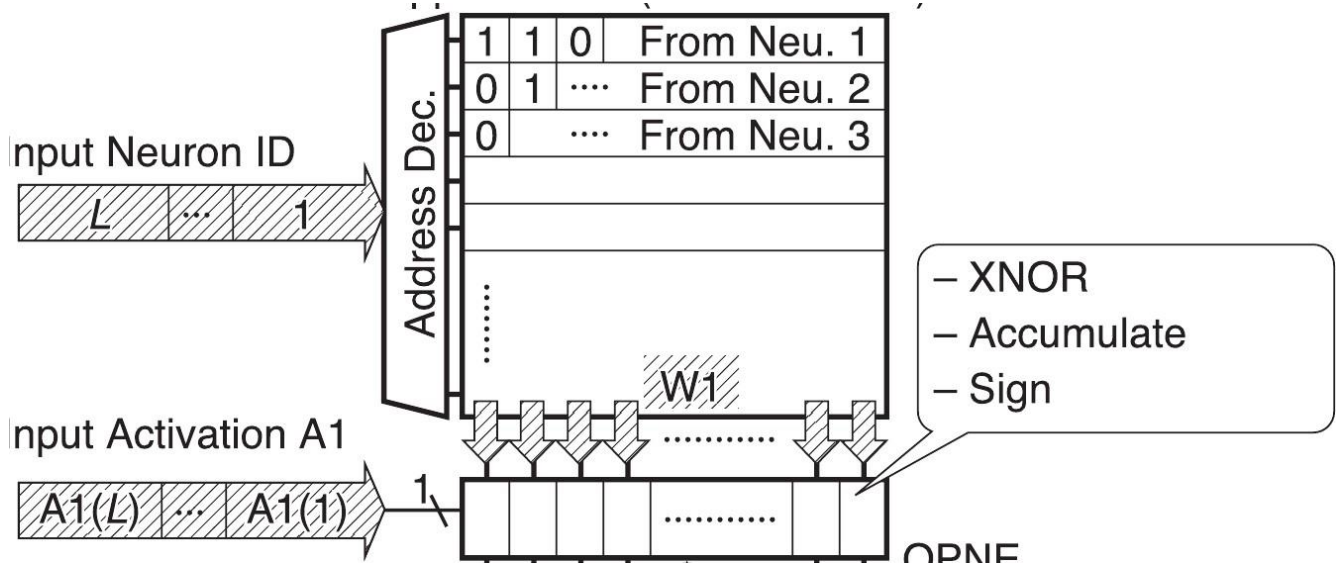
➔ How about using binary inputs?



K. Ando, JSSC 2018



$y = \text{sign}\left(\Sigma w_k x_k\right)$



1) Binary input activations are serially for L cycles
2) XNOR is used for multiplying $w_k \times x_k$.
3) Results are accumulated for L cycle and then activated.

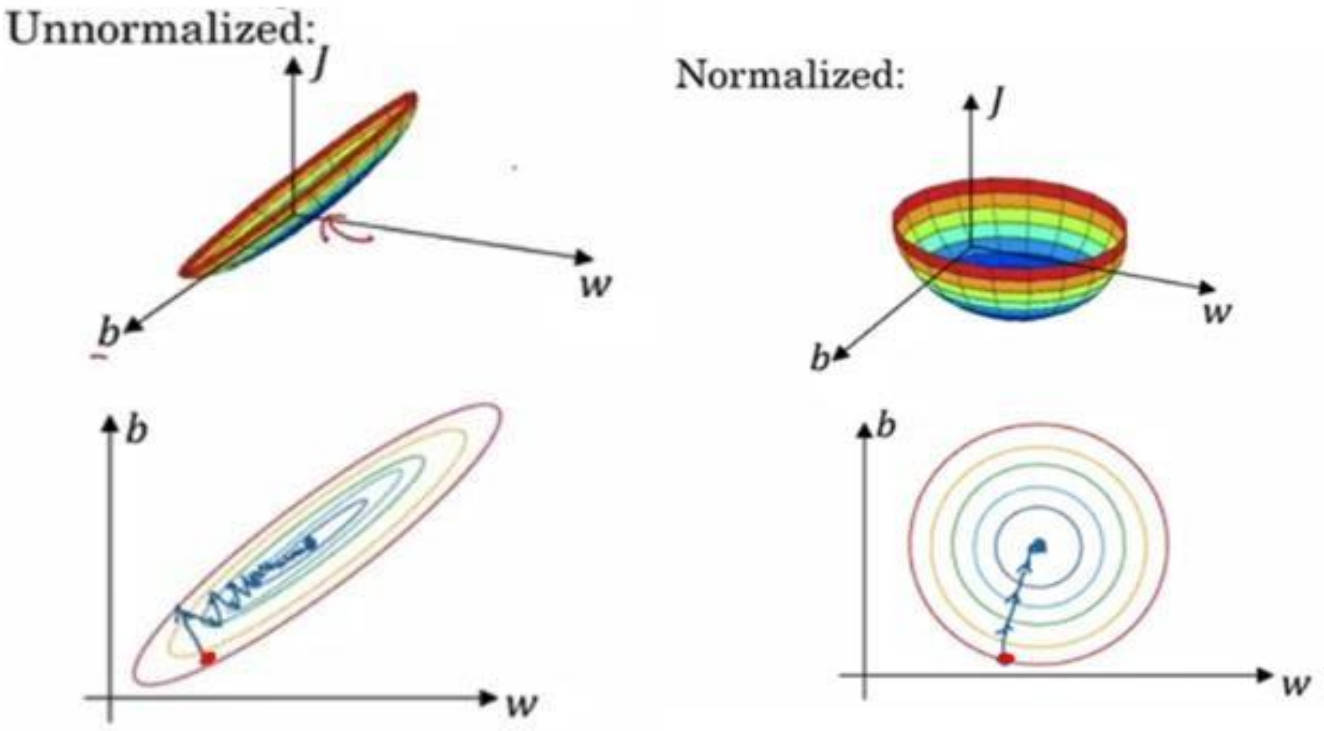
➔ Hard to achieve sufficiently high accuracy with binary w & binary x.

Batch Normalization

- For better learning efficiency, prevention of overfitting and improving the classification accuracy (resolving internal covariant shift issue).

$$\hat{x} = \gamma \left(\frac{x - \mu}{\sigma} \right) + \beta$$

$$\begin{aligned} a' &= \text{sign}(\hat{x}) \\ &= \text{sign} \left\{ \gamma \left(\frac{x - \mu}{\sigma} \right) + \beta \right\} \\ &= \text{sign} \left\{ x + \left(-\mu + \frac{\sigma}{\gamma} \beta \right) \right\} \end{aligned}$$



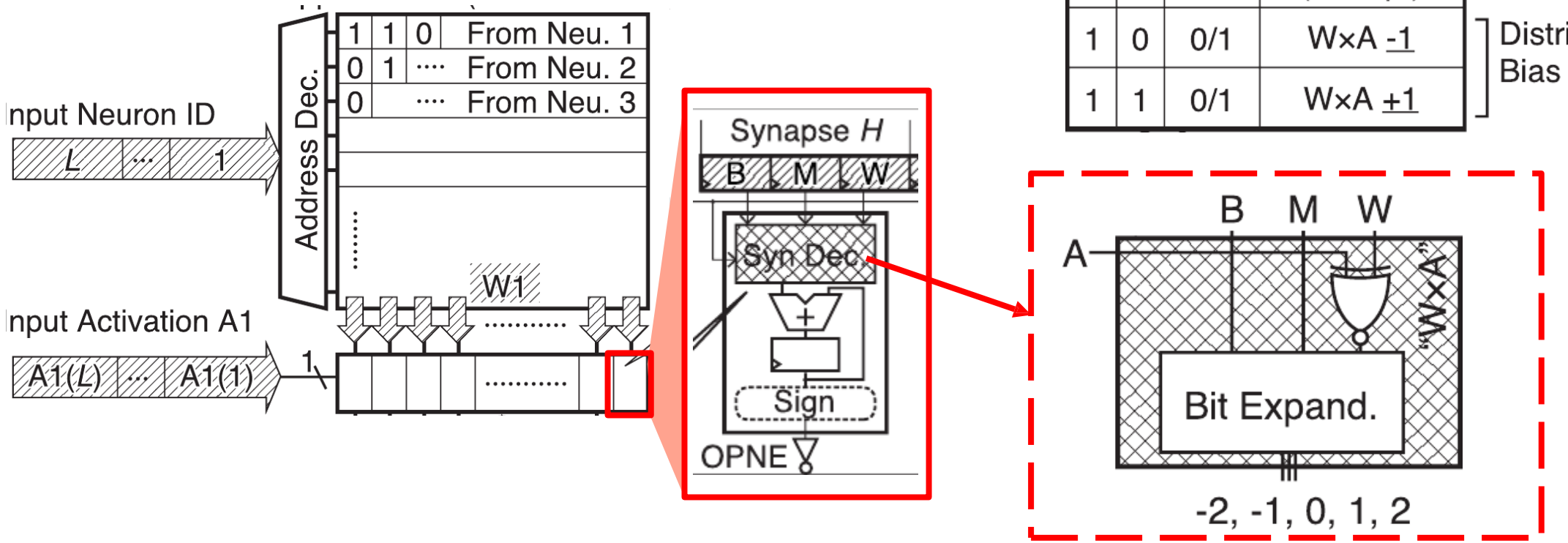
K. Ando, JSSC 2018: Ternary Weight & Bias to X

- $y = \text{sign}\left(\Sigma w_k x_k + b \right)$

+1, -1, 0

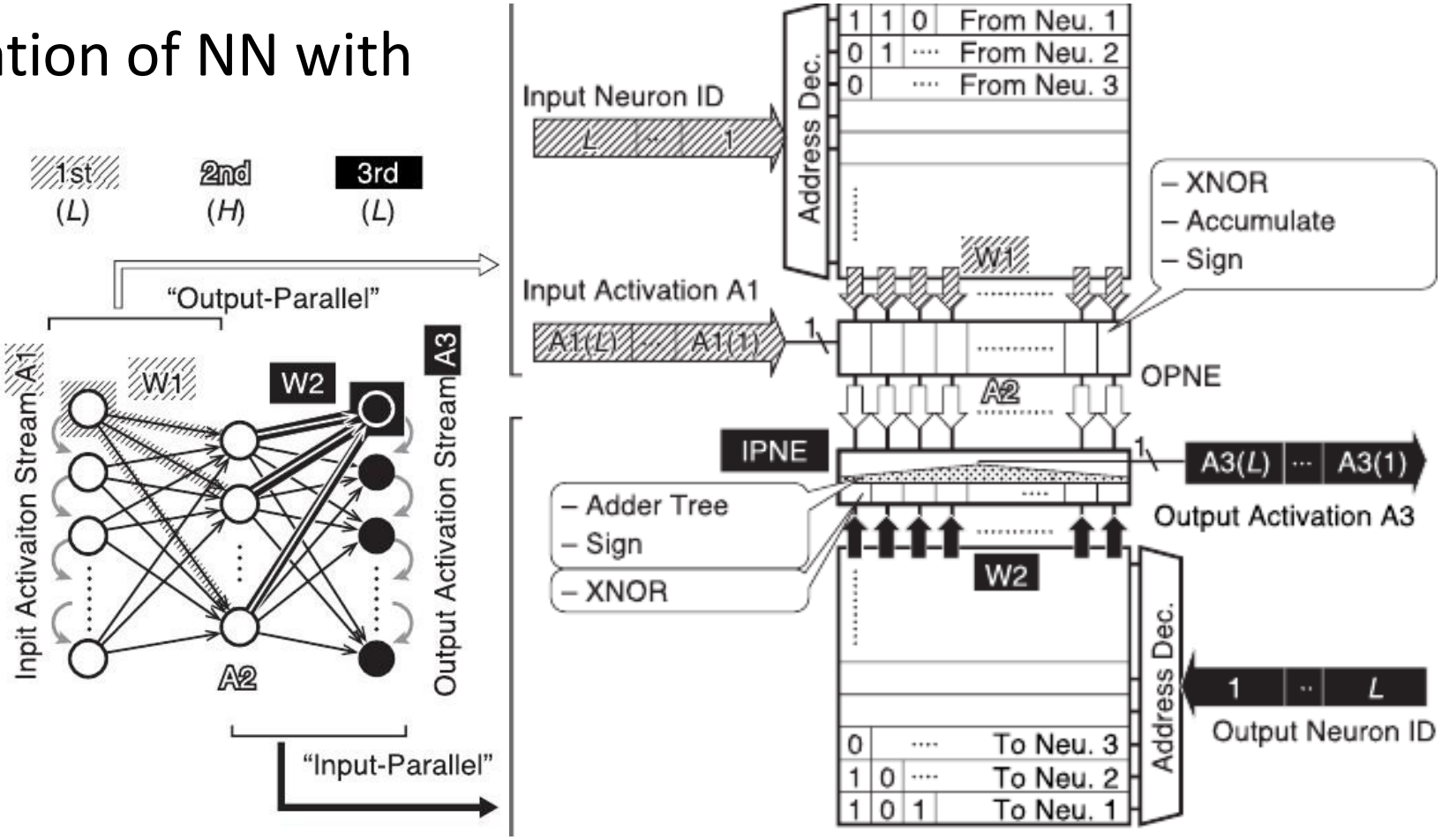
For considering batch normalization

B	M	W	Operation	
0	0	0/1	W×A	
0	1	X	0×A (No Op.)	"Ternary"
1	0	0/1	W×A -1	Distributed Bias
1	1	0/1	W×A +1	



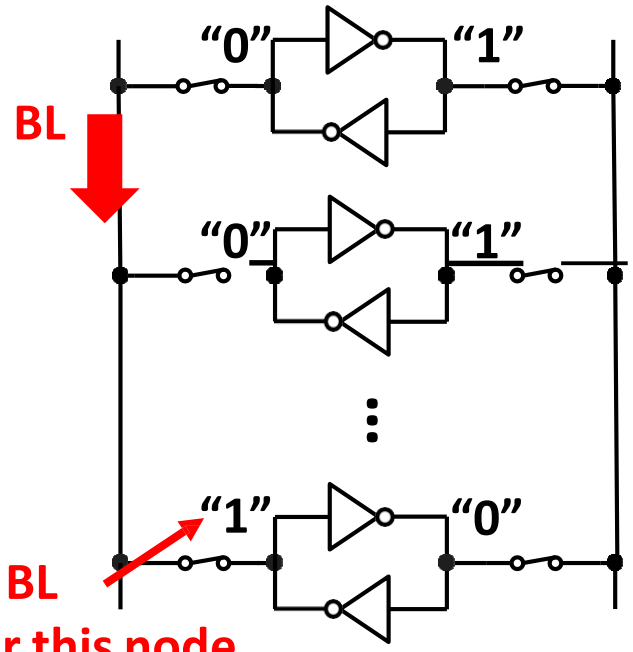
K. Ando, JSSC 2018: Overall Structure

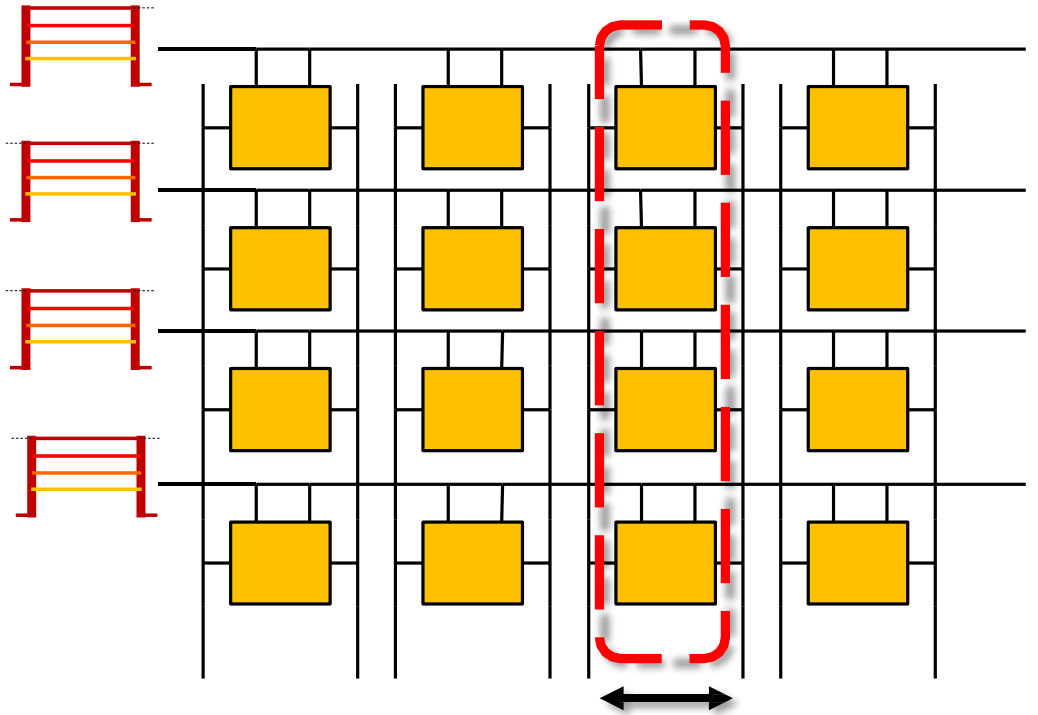
- Highly parallel operation of NN with hardware matching.



Disturbance Issue

- Enabling multiple bitcells at the same column
  ➔ Bitcell data can be flipped by large ΔBL

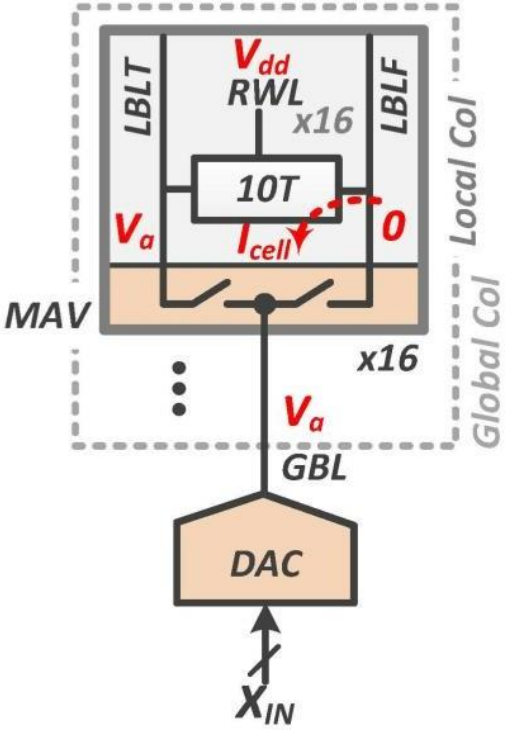


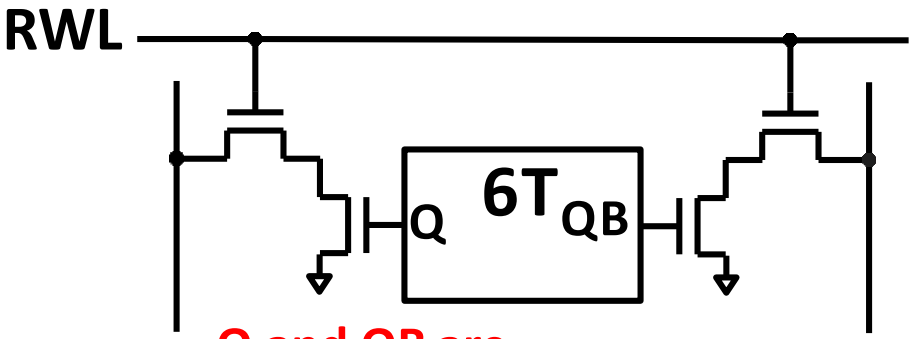


$\Sigma p_k = \Sigma w_k \times x_k$

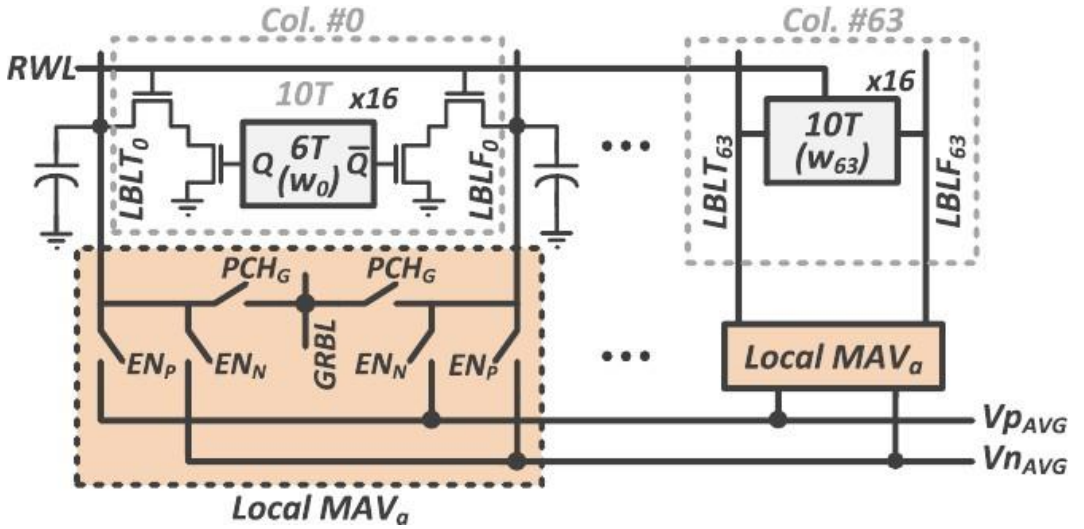
Avishek Biswas, ISSCC 2018: Conv-SRAM

- Utilizing 10T SRAM : free from disturbance issue
- 6bit input is generated on BL voltage by column DAC
  : BL variation is mitigated as DAC is less area constrained
  1) Binary weight (+1, -1) is stored in the bitcell
  2) Single row enable (single WL), 1 column ➔ $w_k \times x_k$
  3) Summation of ΔBL of multiple columns ➔ $\Sigma(w_k \times x_k)$





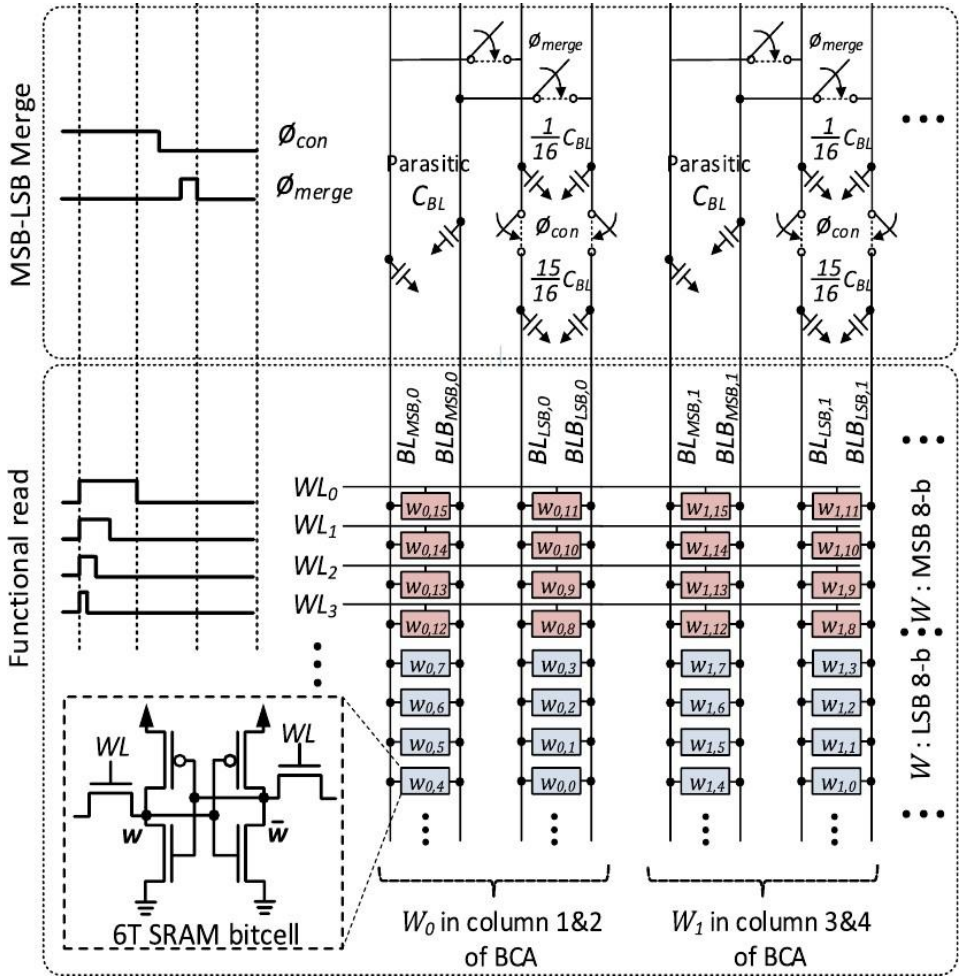
Q and QB are isolated from the read current path



S. K. Gonugondla, ISSCC 2018: Realizing Multibit w_k

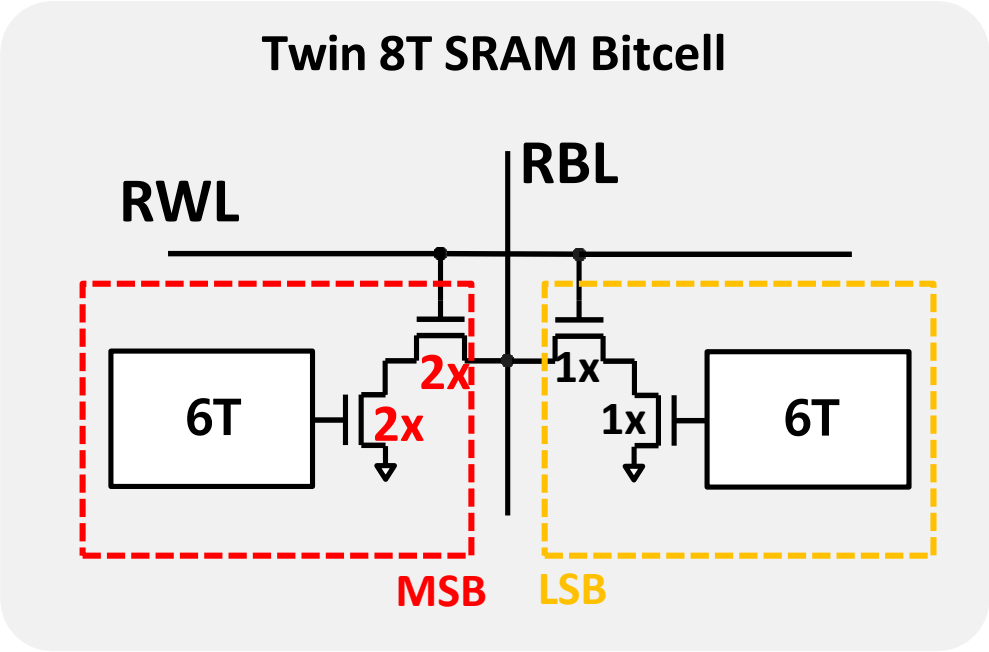
- Not WL itself, but pulse width modulation is used to obtain multibit weight.
- MSB-LSB can be merged by analog voltage multiplier using charge sharing (**analog domain voltage multiplication**)

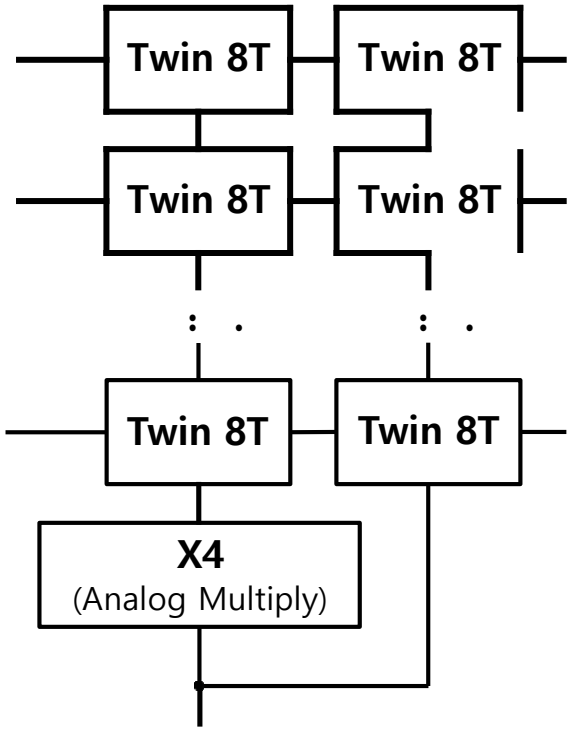
➔ Trade-off



Xin Si, ISSCC 2019: Twin 8T SRAM for Two-bit Weight

- Two-bit weight can achieve better accuracy, with MSB-LSB twin bitcell
- With analog voltage multiplication (not detailed shown), even larger length can be used for weight (4bit or larger, for example).





Contents

- Backgrounds & Motivation
- Structure of SRAM
- Operation of SRAM
- State-of-art SRAM Application; Computation in Memory
- **Conclusion**

Conclusion

- Memory bitcell, which stores 1bit data, should be able to hold/write/read two electrically distinguishable states.
- Backgrounds & Motivation
- Structure of SRAM
- Operation of SRAM

Conclusion

- Matrix multiplications are widely used in ML algorithms. The energy and latency are dominated by the memory access.
- To cut off overhead by memory access, the computations are combined with the memory access, implemented deep inside the memory.
- Due to the mixed signal operation, the noise margin (or SNR) is significantly reduced, but can be minimized with co-optimization with algorithms.
- Still, reduced noise margin should be carefully handled and resolved in the future to enable CiM in sub 10nm technology.
- In addition, scalability and flexibility of CiM should be improved to support various application in the future.