ECE260B Winter 22

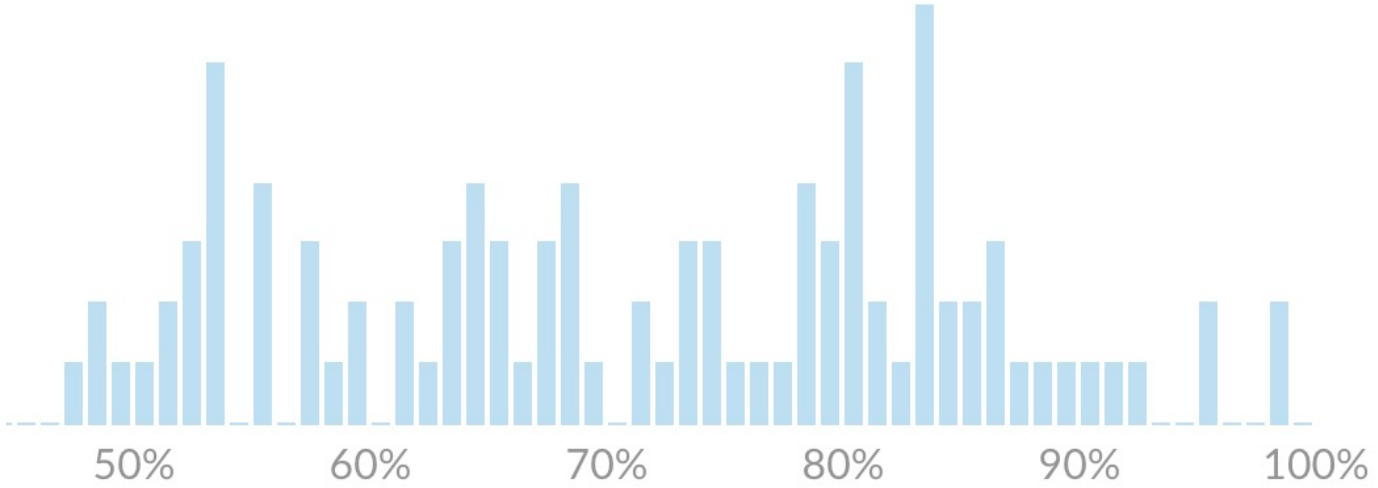
Memory

Prof. Mingu Kang

Midterm Statistics

Average Score	High Score	Low Score	Standard Deviation	Average Time
71%	99%	[redacted]	10.25	01:09:57

• • •



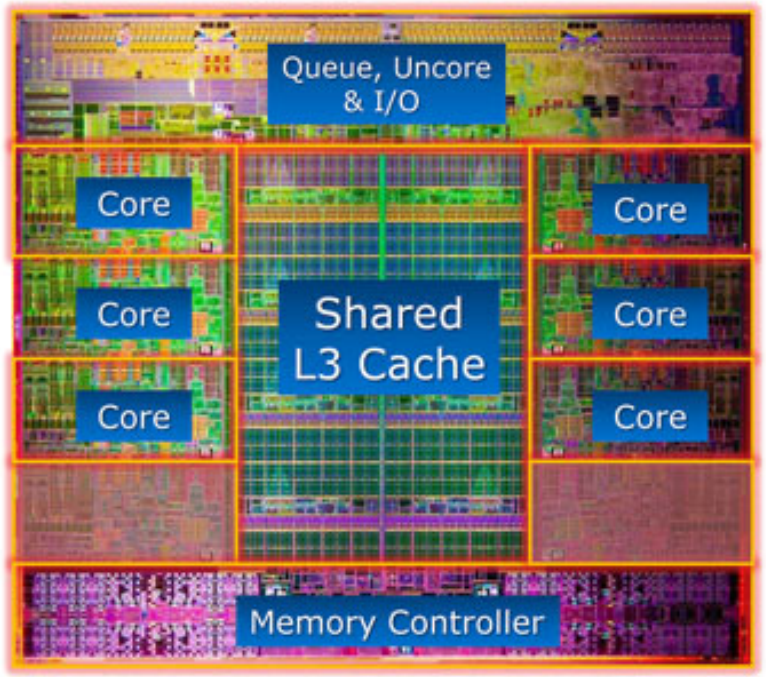
Great job !

Rebuttal Policy

- Score will be given by the end of tomorrow (Feb 25)
- Answer will be given, and simple solution will be given for calculation questions
- Submit your rebuttal through the comment in Canvas or email by 1PM Feb 28 (Mon), **No exception whether you see this message or not. No rebuttal accepted after deadline**
- No question allowed related to midterm problem before Feb 28th 1 PM
- Only one rebuttal opportunity is given, thus describe your logic as clear as possible
- Simply describing your assumption in Q1 does not guarantee a credit
- No in-person verbal rebuttal allowed to be synchronous

Processor Area Becoming Memory Dominated

Intel® Core™ i7-3960X Processor Die Detail



- **On chip SRAM contains 50-90% of total transistor count**
  - Xeon: 48M/110M
  - Itanium 2: 144M/220M
- **SRAM is a major source of chip static power dissipation**
  - Dominant in ultra-low power applications
  - Substantial fraction in others

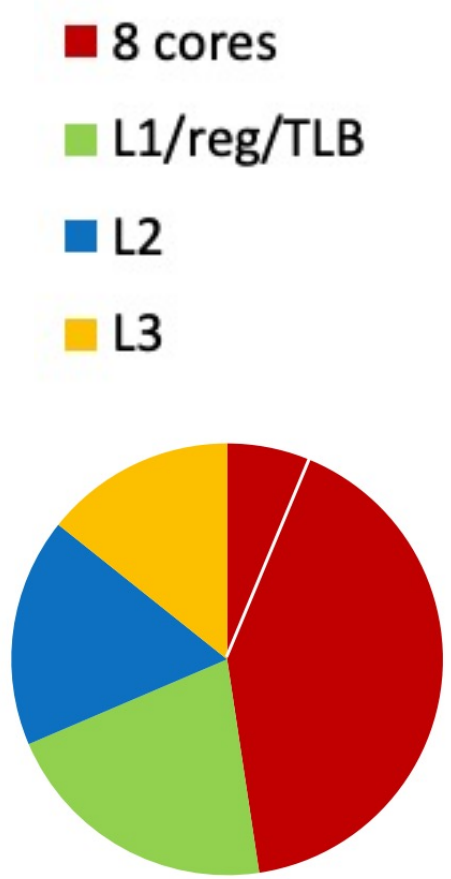
https://www.computerhope.com/issues/ch001371.htm

Energy in Computation vs. Memory

Integer	
Add	
8 bit	0.03pJ
32 bit	0.1pJ
Mult	
8 bit	0.2pJ
32 bit	3 pJ

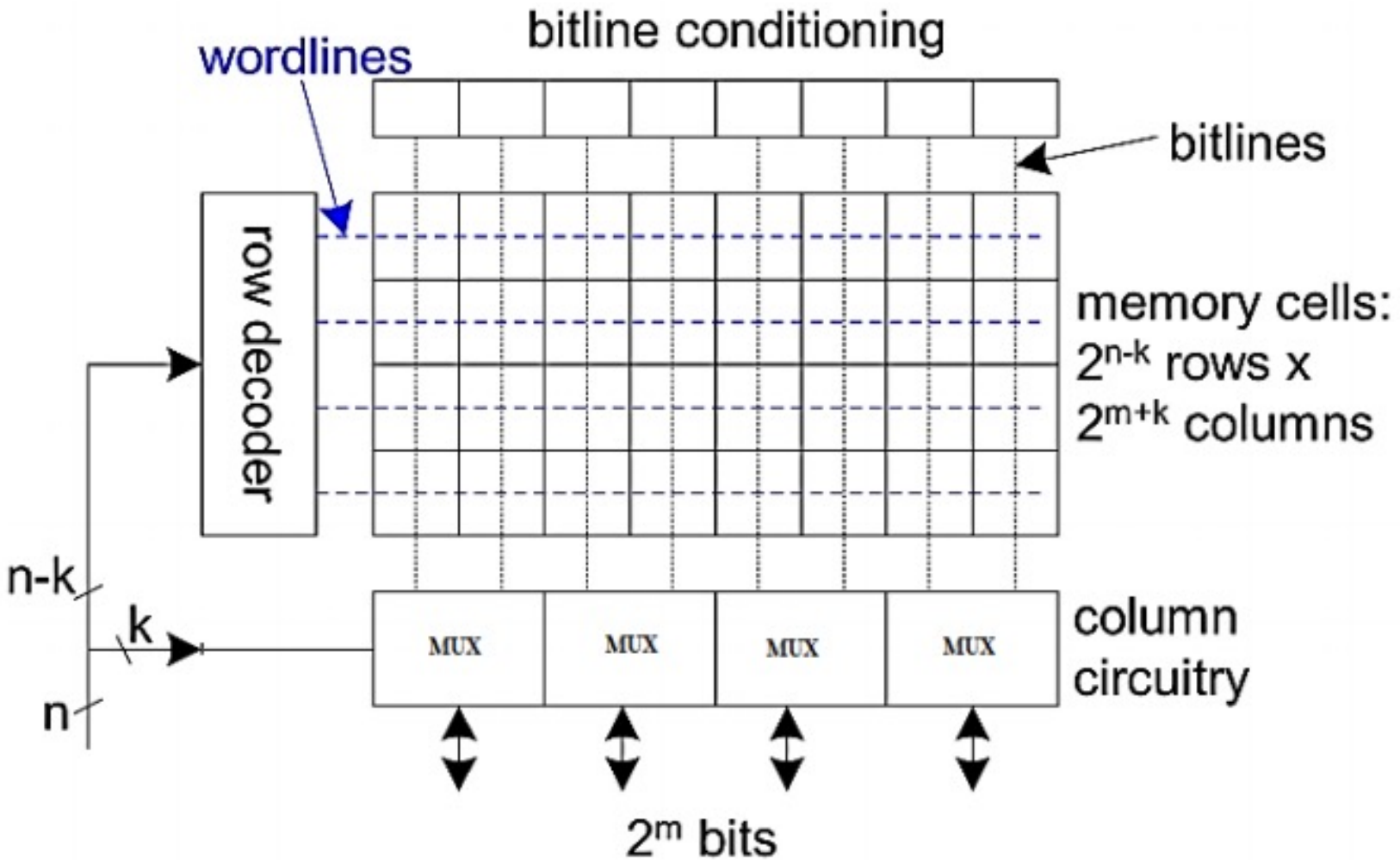
FP	
FAdd	
16 bit	0.4pJ
32 bit	0.9pJ
FMult	
16 bit	1pJ
32 bit	4pJ

Memory	
Cache	(64bit)
8KB	10pJ
32KB	20pJ
1MB	100pJ
DRAM	1.3-2.6nJ



ISSCC 2014 M. Horowitz

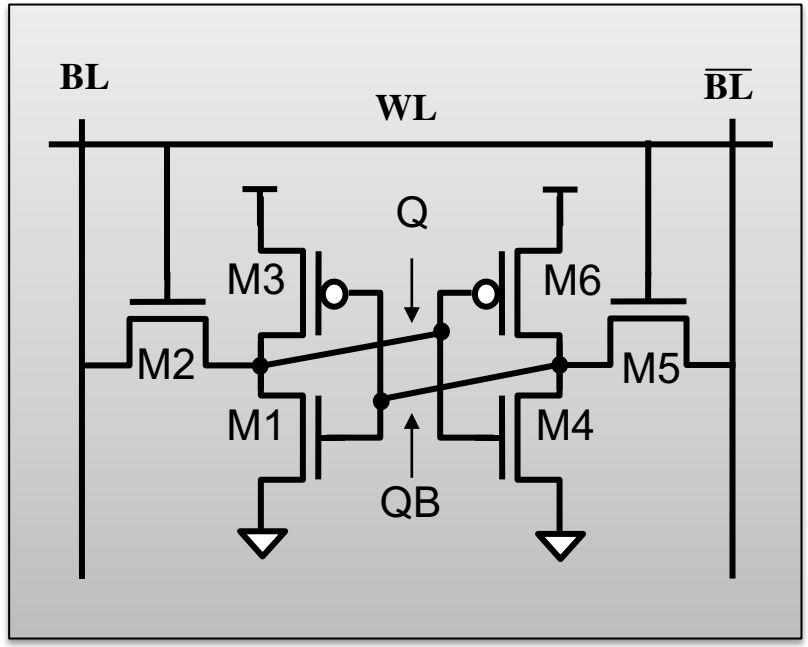
SRAM Architecture



- Total $2^{(m+n)}$ bits
- Word size = 2^m bits
- Column mux ratio = 2^k:1
- Sense amp / Write driver shared every 2^k columns
- 2^k-1 columns are discharged unnecessarily

https://www.chegg.com/homework-help/questions-and-answers/base-following-structure-memory-complete-following-table-unknown-characteristics-memories--q7995547

SRAM Operations

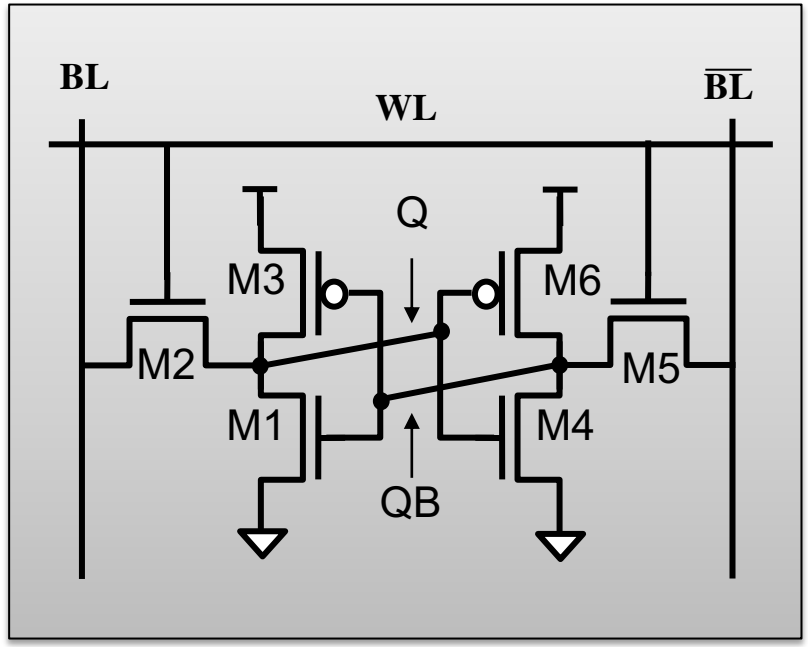


Traditional 6-Transistor (6T) SRAM cell

Three tasks of a cell

- Hold (data retention)
  - WL=0; BLs=X
- Write
  - WL=1; BLs driven with new data
- Read
  - WL=1; BLs precharged and left floating

SRAM Cell Metrics

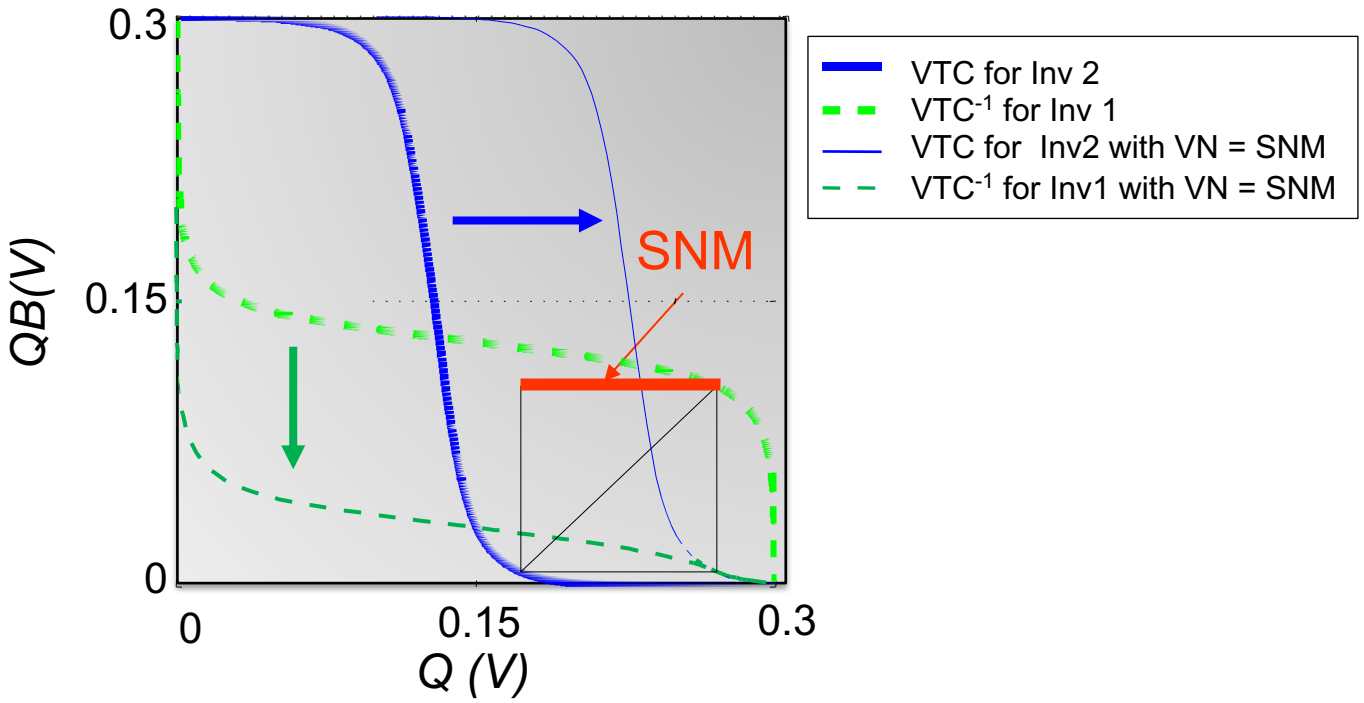


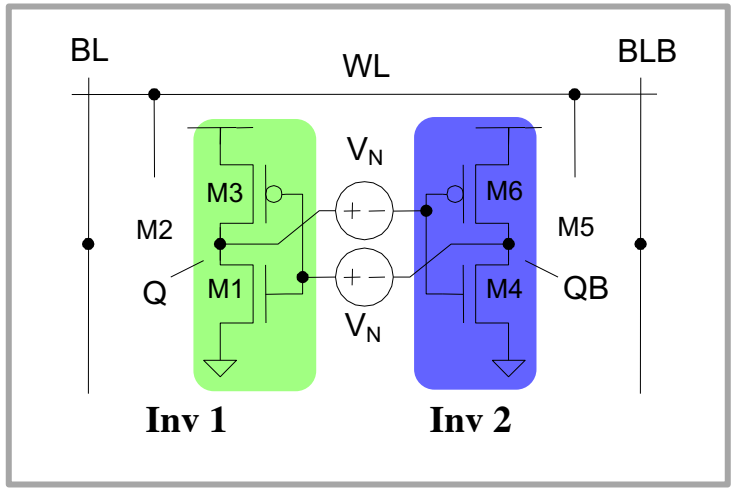
Traditional 6-Transistor (6T) SRAM cell

Key functionality metrics

- Hold (Data retention)
  - Static Noise Margin (SNM)
  - Data retention voltage (DRV)
- Read
  - Static Noise Margin (SNM)
- Write
  - Write Margin

Static Noise Margin (SNM)

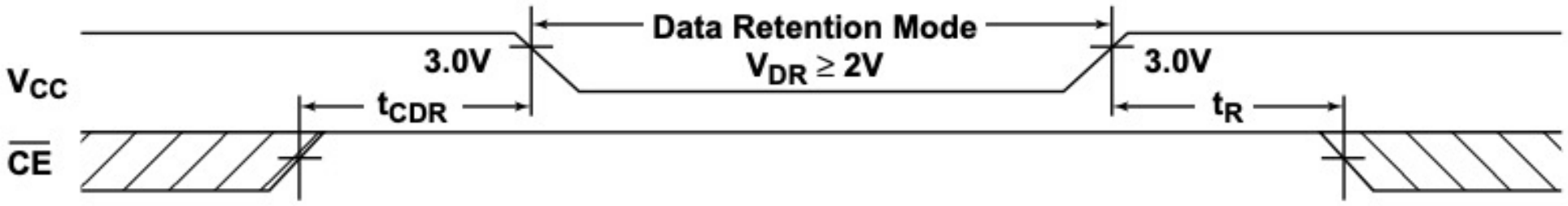




- SNM: a measure of the cell's stability by quantifying the DC noise required to flip the cell
- SNM is length of side of the largest embedded square on the butterfly curve

[Ref: E. Seevinck, JSSC'87]

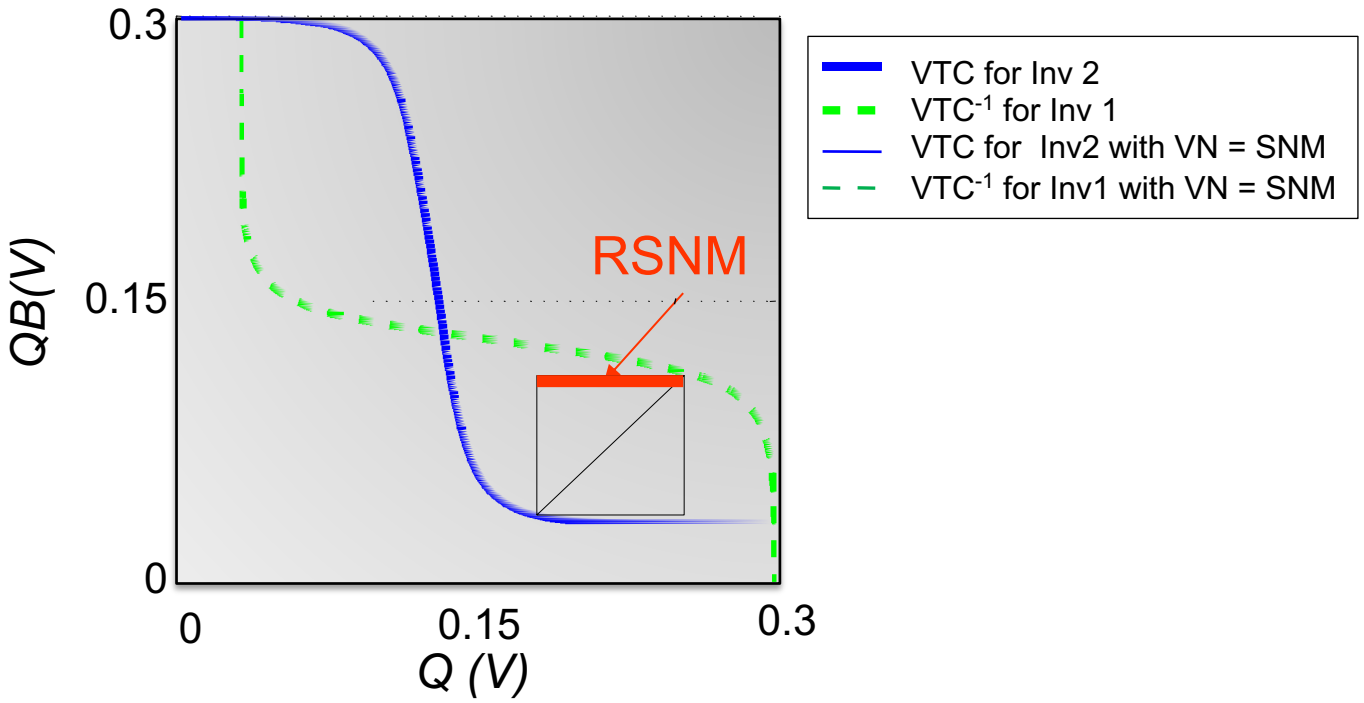
Retention Mode

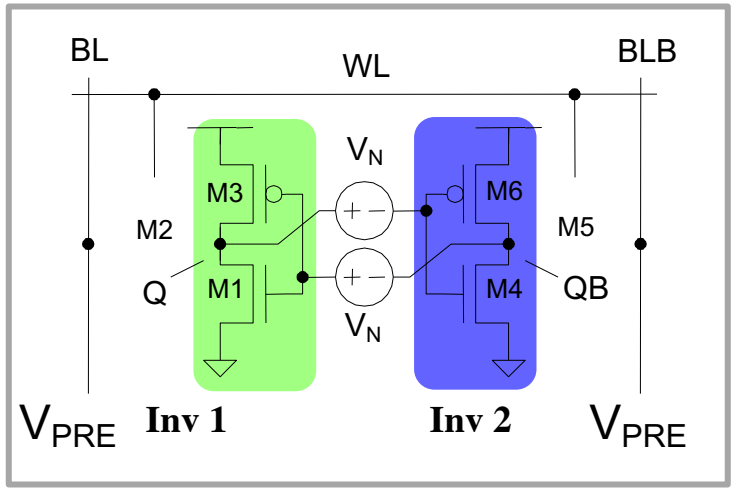


Source: Cypress/ICE, "Memory 1997" 22460

- Supply voltage is lowered during retention mode to minimize leakage
- SNM is further reduced
- t_{CDR}: time to switch to retention mode
- t_R: time to recover from retention mode
- Note above diagram assumes the retention mode begins with CE enabling, but sometimes there is extra pin (e.g., RET pin)

Read static Noise Margin (RSNM)



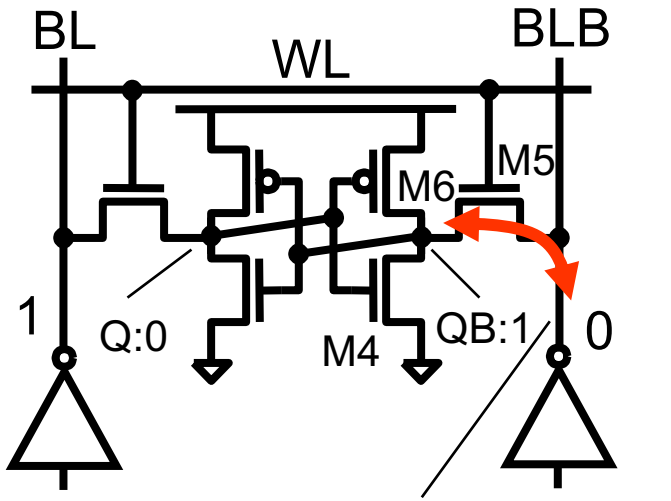


- V_{PRE} is floating voltage, but assume that it is forced for the worst-case consideration

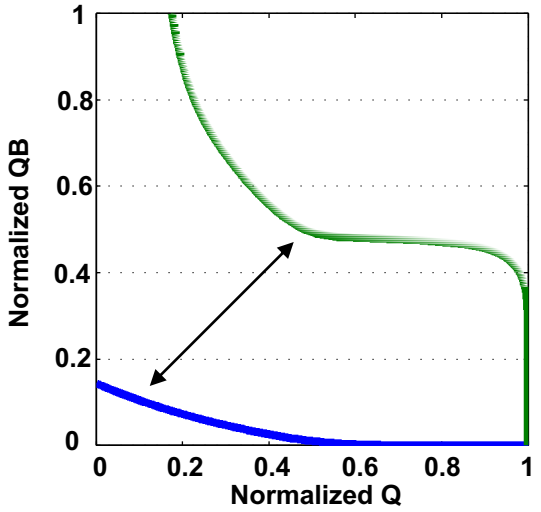
- RSNM: similar to SNR, but set the BL & BLB voltages to be V_{PRE} to emulate the read operation
- $I_{M4} > I_{M5}$, otherwise the data will be flipped (lost)

[Ref: E. Seevinck, JSSC'87]

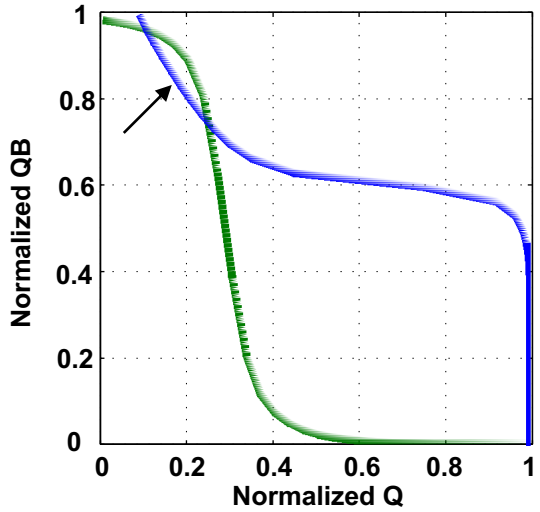
Write Margin



Dominant fight (ratioed)



Successful write:
Negative “SNM”

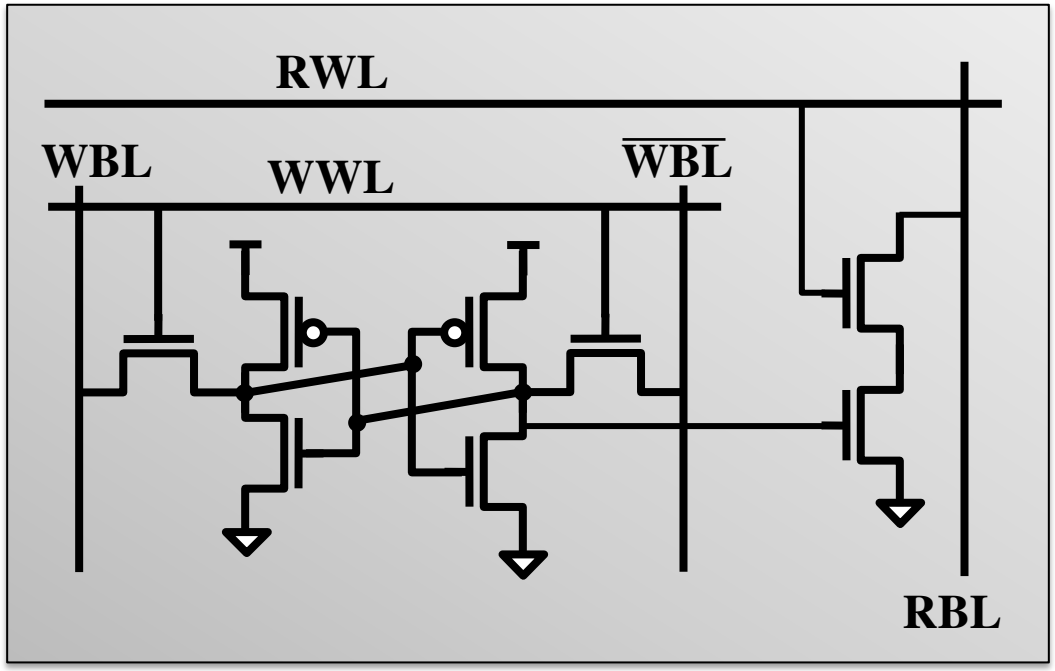


Write failure:
Positive SNM

- Data 0 should be written to QB by flipping the current data 1
- Note, data 1 cannot be written as $I_{M4} > I_{M5}$
- $I_{M5} > I_{M6}$, otherwise the data 0 cannot be written
- Thus, $I_{M4} > I_{M5} > I_{M6}$

[Ref: E. Seevinck, JSSC’87]

8-T SRAM Bitcell

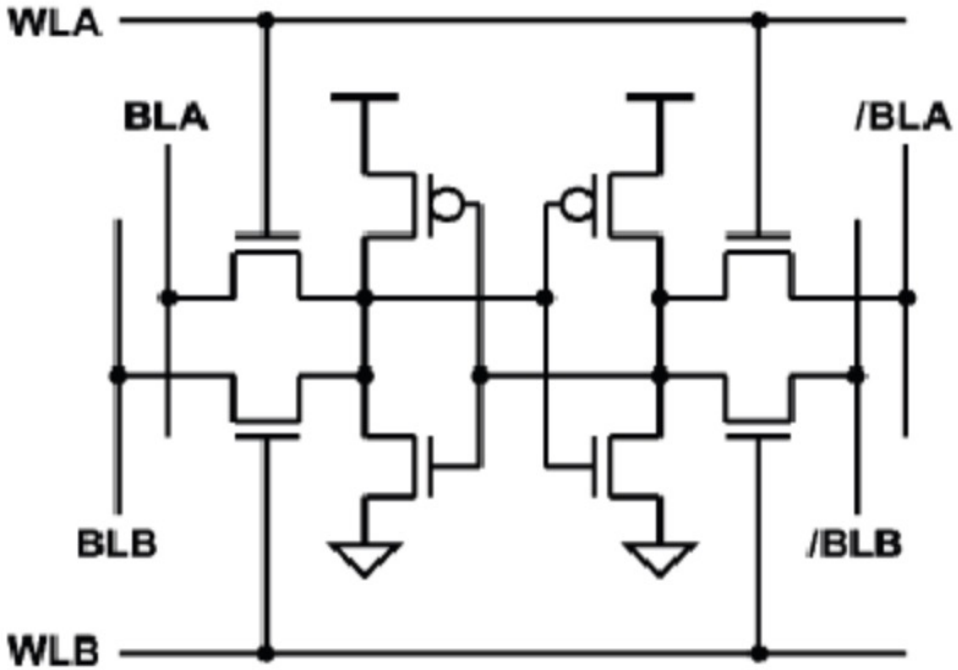


8T SRAM cell

- Key tradeoff is with functional robustness (e.g. Remove read SNM) vs. 30% area overhead
- Dual port support (1R / 1W port) – suitable for register file
- Allows lower V_{DD}

[Ref: L. Chang, VLSI'05]

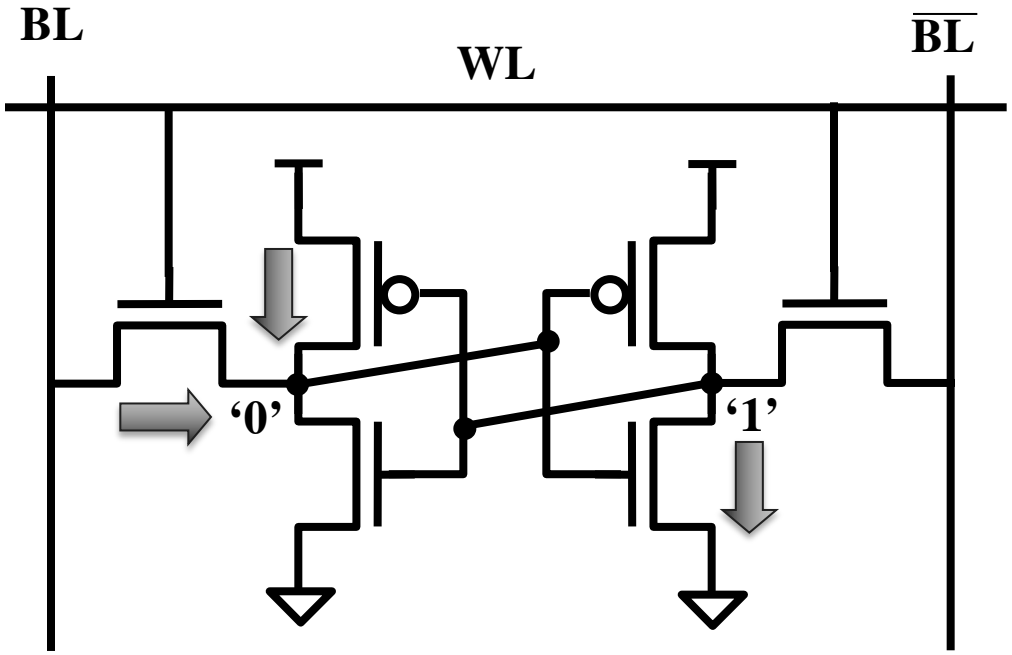
Dual-port SRAM (Register File)



- Another type of dual-port SRAM
- Both ports (A and B) can be used either read and write functionality

K.Zhang, A research of dual-port SRAM cell using 8T, ICSICT 2010

Leakage Currents

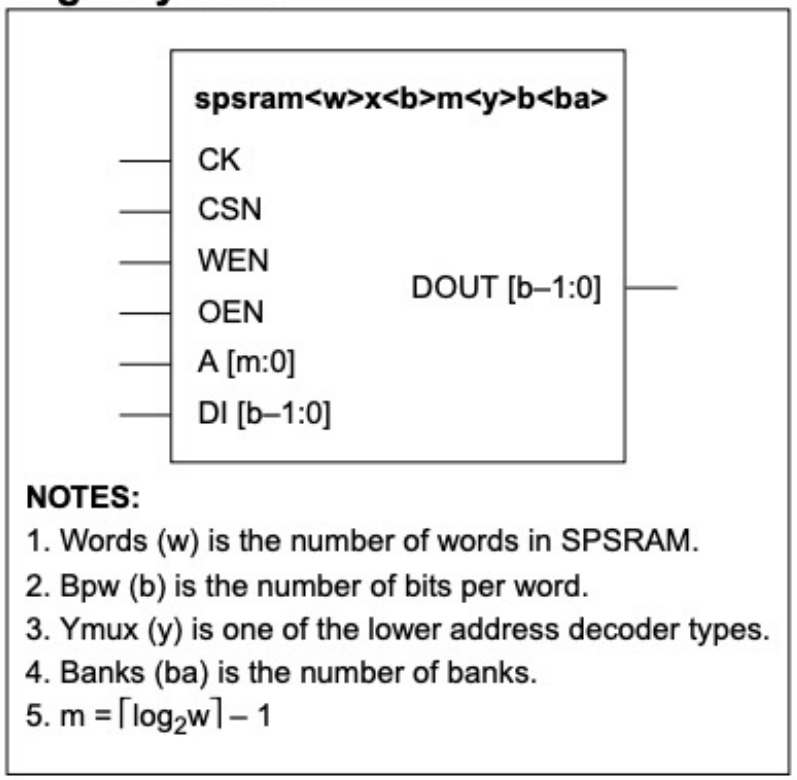


⇨ Sub-threshold leakage

- Leakage Power dominates while the memory holds data (Stand-by mode)
- Supply voltage is reduced to minimize the leakage current (so called, sleep mode or retention mode or stand-by mode) -> but, causes lower SNR

Memory Compiler-based Single-port SRAM (SPSRAM)

Logic Symbol



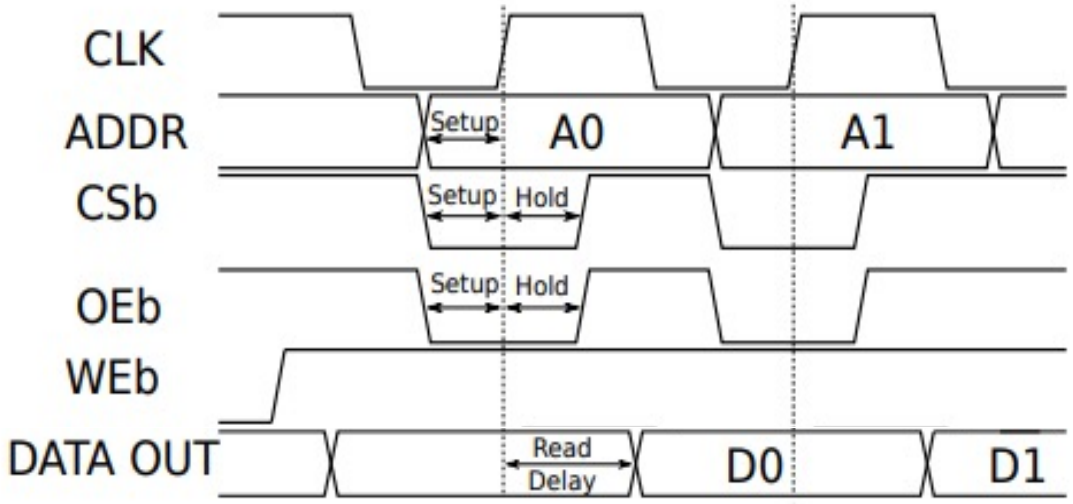
NOTES:

1. Words (w) is the number of words in SPSRAM.
2. Bpw (b) is the number of bits per word.
3. Ymux (y) is one of the lower address decoder types.
4. Banks (ba) is the number of banks.
5. $m = \lceil \log_2 w \rceil - 1$

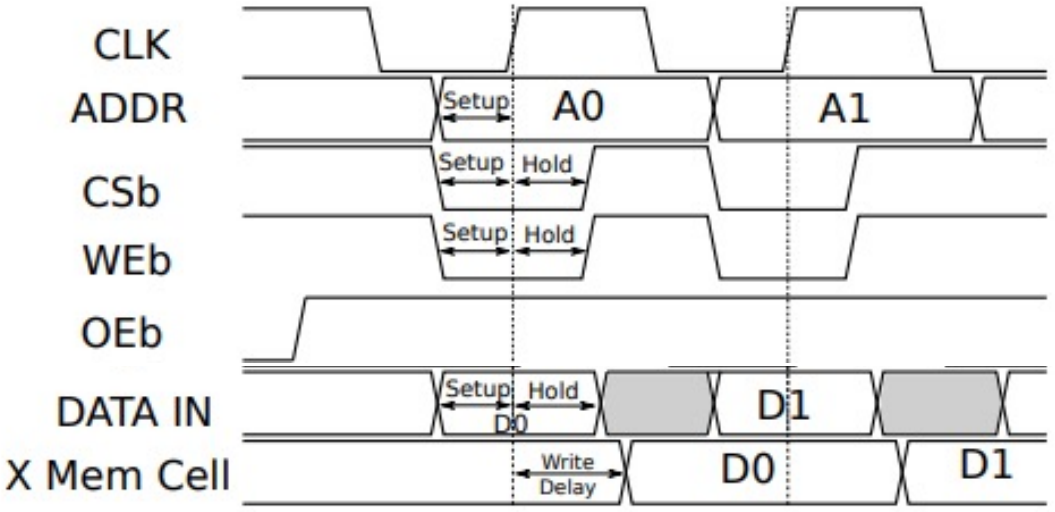
- CSN (or CEN): chip select negative
- OEN: output enable negative
  @OEN = 1, DOUT is tri-state
- A: address
- DI (or D): input data
- DOUT (or Q): output data

- RETN: retention mode enable

Read and Write Timing Diagram

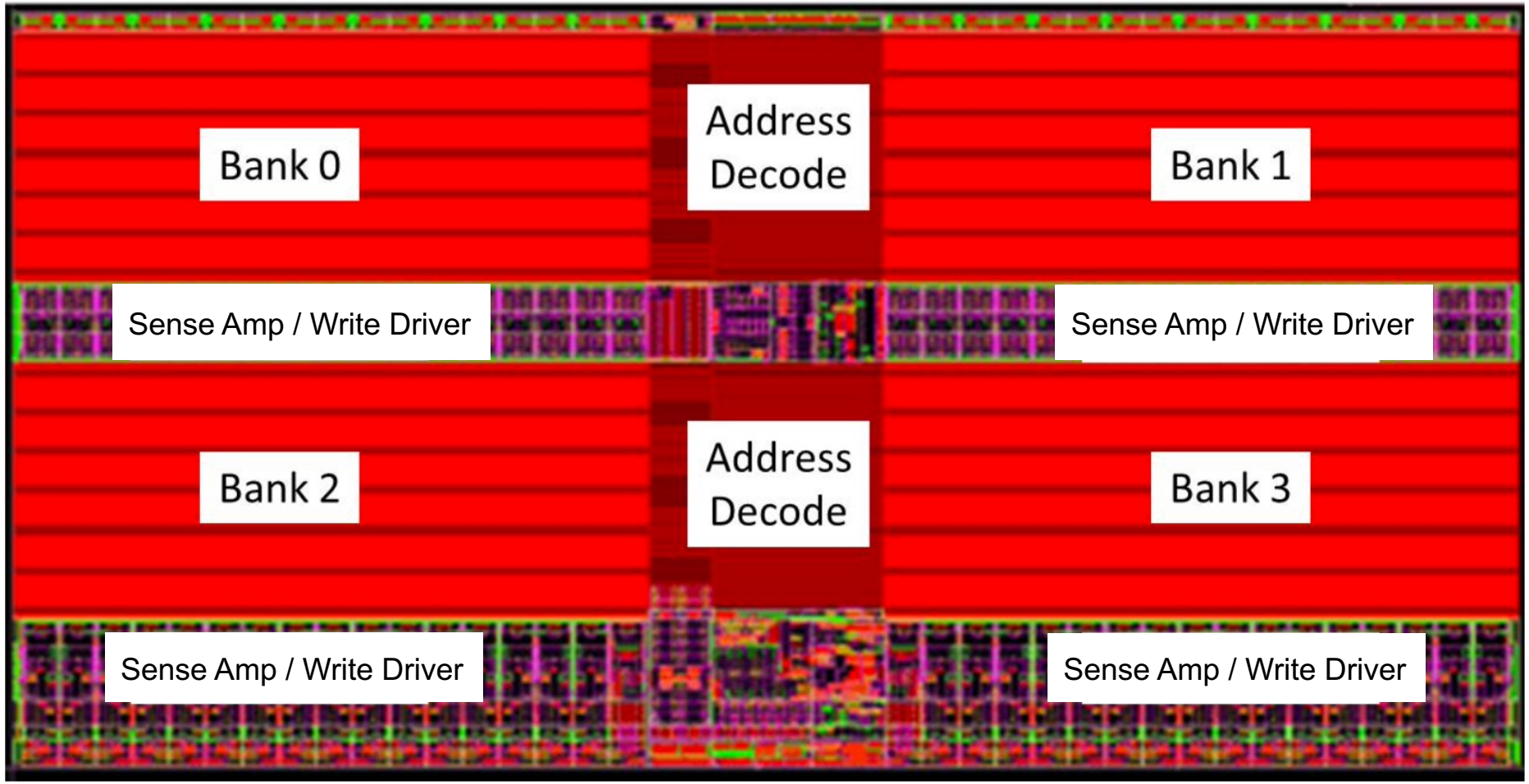


(a) Read operation timing



(b) Write operation timing

Multi-bank SRAM Layout



Example of Memory Compiler Options

Parameters			YMUX = 2	YMUX = 4	YMUX = 8	YMUX = 16	YMUX = 32
Words (w)		Min	4	8	16	32	64
		Max	512	1024	2048	4096	8192
		Step	2	4	8	16	32
Bpw (b)	ba = 1	Min	1	1	1	1	1
		Max	128	64	32	16	8
		Step	1	1	1	1	1
	ba = 2	Min	2	2	2	2	2
		Max	256	128	64	32	16
		Step	1	1	1	1	1

- Bpw: bit width per word
- YMUX = column mux
- ba: # of banks

http://users.ece.cmu.edu/~koopman/ece548/hw/hw5/meml80.pdf

Timing and Power Specs

1) Timing Characteristics [Unit: ns]

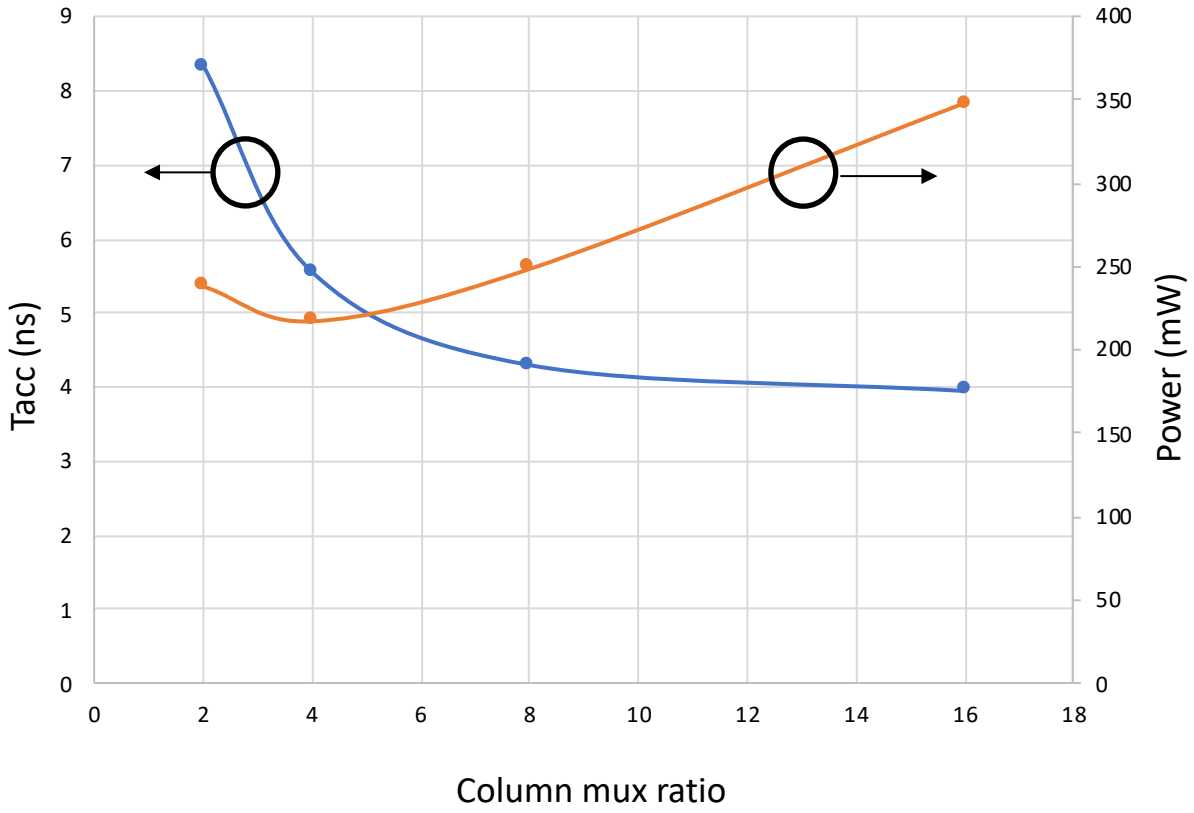
Timing Type	Timing Equation
	Y = 2
cycle_time	(5.8213e – 03 * W + 1.1171e – 02 * B / BA + 9.6628e – 01 * 0.02 * SL + 3.7318) * 1.08
minckh	(1.9096e – 03 * W + 2.4791e – 03 * B / BA + 1.2200 * 0.02 * SL + 2.1295)
minckl	(2.1603e – 03 * W + 3.5437e – 03 * B / BA + 1.7035 + 2.8208e – 07 * W * B / BA)
tacc	(2.8038e – 03 * W + 5.8650e – 03 * B / BA + 1.7115e – 01 * S + 4.9717e – 01 * 0.02 * SL + 2.3023)
	Y = 4
cycle_time	(2.9106e – 03 * W + 2.2342e – 02 * B / BA + 9.6628e – 01 * 0.02 * SL + 3.7318) * 1.08
minckh	(9.5482e – 04 * W + 4.9583e – 03 * B / BA + 1.2200 * 0.02 * SL + 2.1295)
minckl	(1.0801e – 03 * W + 7.0874e – 03 * B / BA + 1.7035 + 2.8207e – 07 * W * B / BA)
tacc	(1.4019e – 03 * W + 1.1730e – 02 * B / BA + 1.7115e – 01 * S + 4.9717e – 01 * 0.02 * SL + 2.3023)
	Y = 8
cycle_time	(1.4553e – 03 * W + 4.4684e – 02 * B / BA + 9.6628e – 01 * 0.02 * SL + 3.7318) * 1.08
minckh	(4.7741e – 04 * W + 9.9166e – 03 * B / BA + 1.2200 * 0.02 * SL + 2.1295)
minckl	(5.4009e – 04 * W + 1.4174e – 02 * B / BA + 1.7035 + 2.8208e – 07 * W * B / BA)
tacc	(7.0097e – 04 * W + 2.3460e – 02 * B / BA + 1.7115e – 01 * S + 4.9717e – 01 * 0.02 * SL + 2.3023)
	Y = 16
cycle_time	(7.2766e – 04 * W + 8.9369e – 02 * B / BA + 9.6628e – 01 * 0.02 * SL + 3.7318) * 1.08
minckh	(2.3870e – 04 * W + 1.9833e – 02 * B / BA + 1.2200 * 0.02 * SL + 2.1295)

2) Power Characteristics [Unit: μW]

Power Type	Power Equation
	Y = 2
power_ck	(1.7046e – 01 * W + 8.7926e – 01 * B + 1.2869 + 7.9009e – 04 * W * B) * VDD^2 * F
power_csn	(4.6346e – 03 * W + 1.3484e – 01 * B + 5.8215e – 01 – 9.0563e – 05 * W * B) * VDD^2 * F
	Y = 4
power_ck	(8.5232e – 02 * W + 1.7585 * B + 1.2869 + 7.9009e – 04 * W * B) * VDD^2 * F
power_csn	(2.3173e – 03 * W + 2.6969e – 01 * B + 5.8215e – 01 – 9.0563e – 05 * W * B) * VDD^2 * F
	Y = 8
power_ck	(4.1803e – 02 * W + 3.3360 * B + 3.3968e – 01 + 6.4193e – 04 * W * B) * VDD^2 * F
power_csn	(1.1586e – 03 * W + 5.3939e – 01 * B + 5.8215e – 01 – 9.0563e – 05 * W * B) * VDD^2 * F
	Y = 16
power_ck	(1.8749e – 02 * W + 5.2054 * B + 3.5471 + 8.7095e – 04 * W * B) * VDD^2 * F
power_csn	(5.7933e – 04 * W + 1.0787 * B + 5.8215e – 01 – 9.0563e – 05 * W * B) * VDD^2 * F
	Y = 32
power_ck	(9.0824e – 03 * W + 9.4829 * B + 4.6843 + 7.9026e – 04 * W * B) * VDD^2 * F
power_csn	(2.8966e – 04 * W + 2.1575 * B + 5.8215e – 01 – 9.0563e – 05 * W * B) * VDD^2 * F

http://users.ece.cmu.edu/~koopman/ece548/hw/hw5/meml80.pdf

Access Time and Power w.r.t. Column Mux Ratio

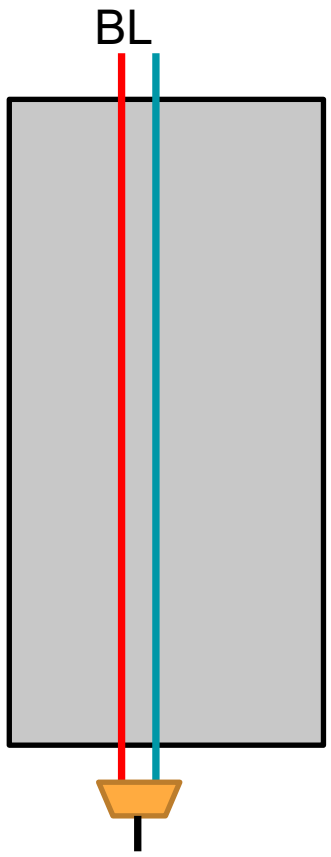


Setting

- number of words = 2048
- bit per word = 16 bits
- VDD = 1V
- Freq = 1 GHz

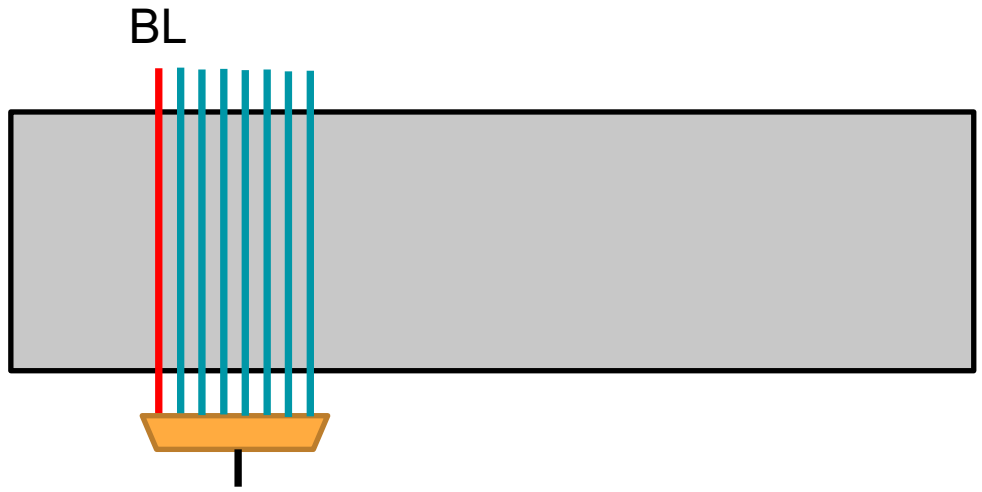
- Higher column mux leads to fast read operation, but higher power consumption

Power & Speed Trade-off w.r.t. Column Mux Ratio



Column mux = 2:1

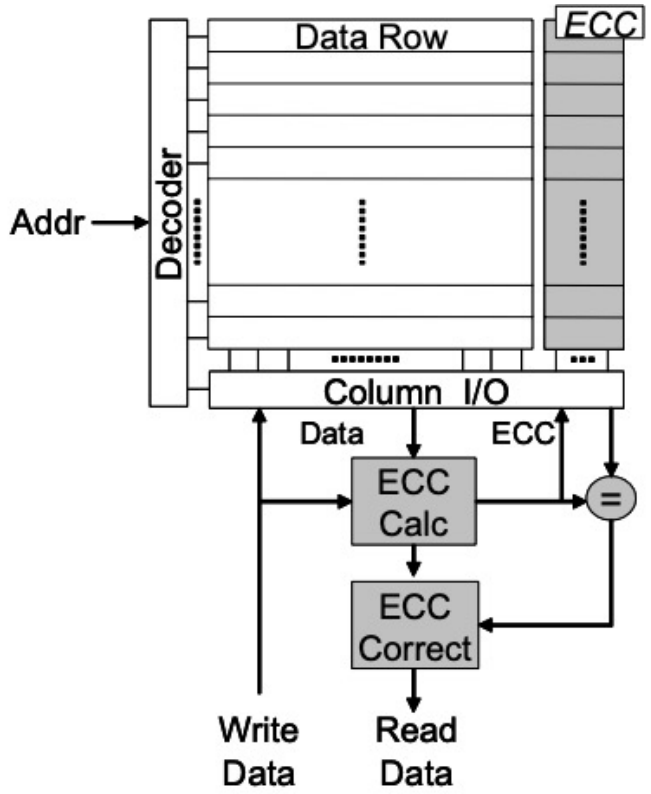
- long BLs: slow operation
- But, only 2 columns read at a time per 1 bit read

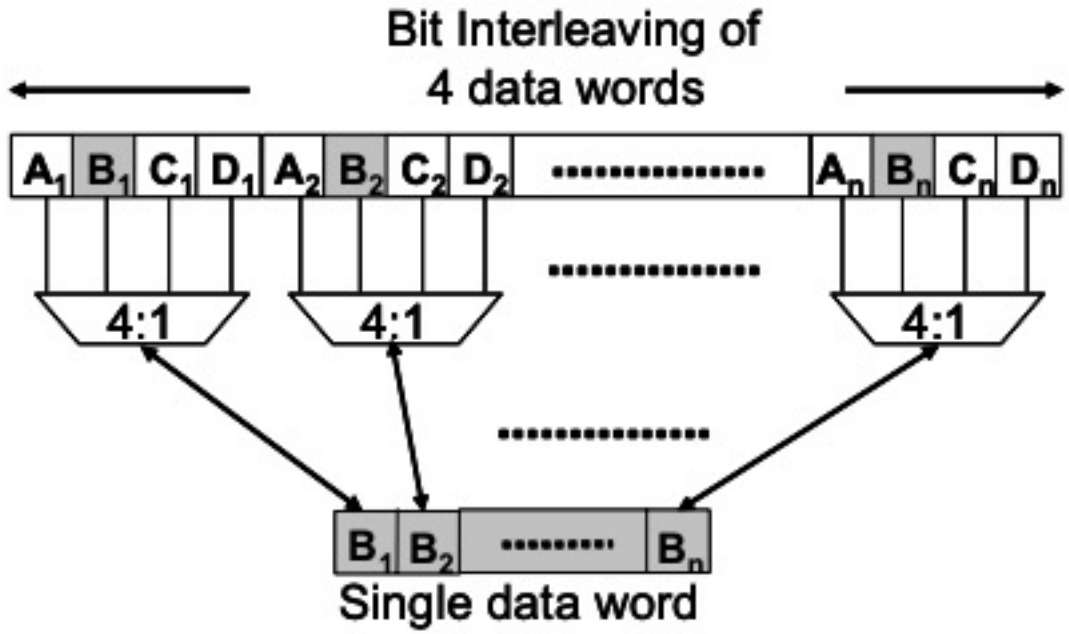


Column mux = 8:1

- Short BLs: fast operation
- But, 8 columns read at a time per 1 bit read
- Large energy consumption

Error Correcting Code Constraints w.r.t Column Mux

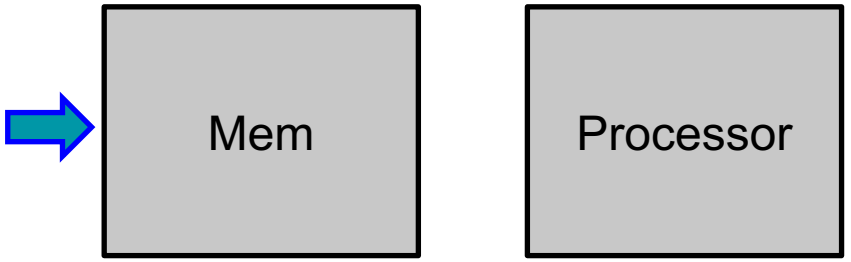




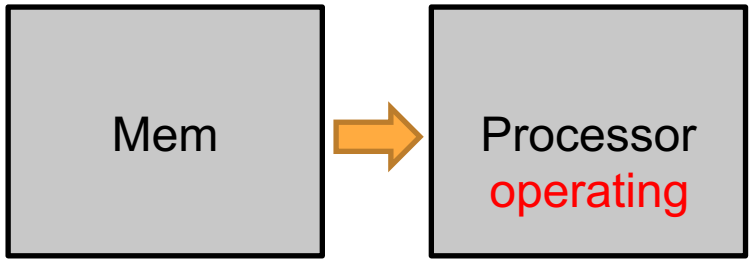
- Multi-bit failure per word is more difficult to correct by error correcting code
- Thus, bits in a word are interleaved to be spread (far each other spatially)
- The interleaving avoids multi-bit failure by alpha particles
- Higher tolerance with higher bit interleaving (column mux) ratio at the expense of power increase

J. Kim, Multi-bit Error Tolerant Caches Using Two-Dimensional Error Coding, MICRO 07

Memory Double Buffering



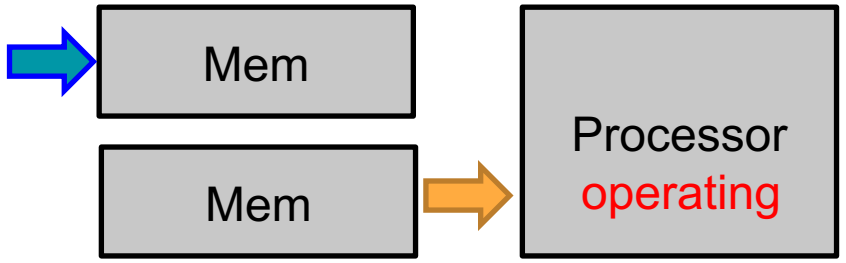
Writing



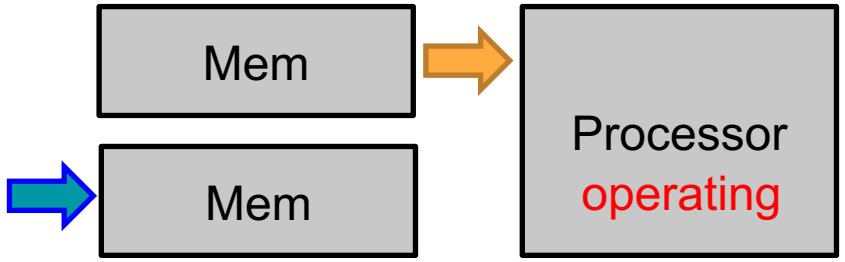
Reading

Without double buffering

- Processor needs to wait whiling writing



read & write during period1



read & write during period2

With double buffering

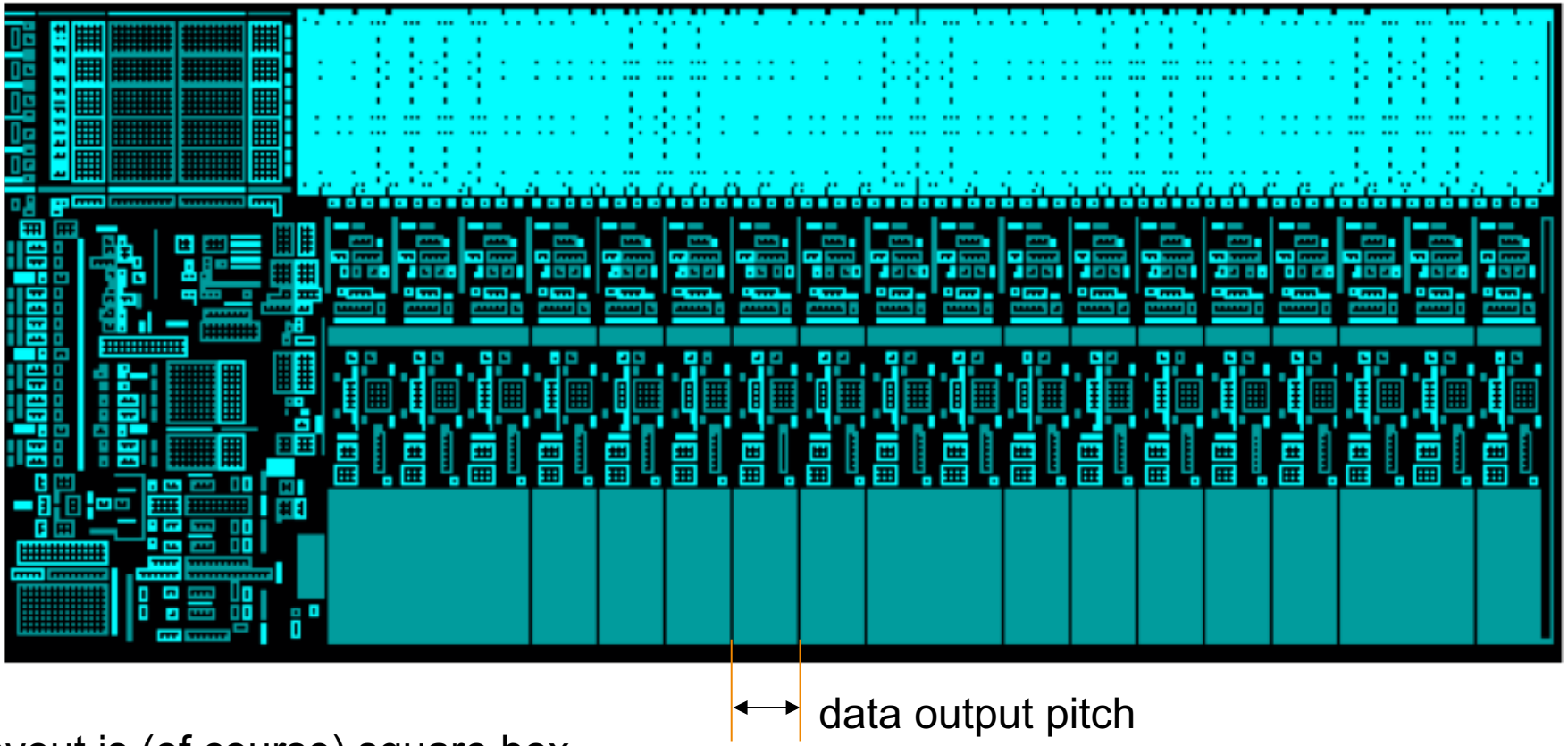
- Processor can be operated all the time
- Two independent memory banks are required

Synchronous vs. Asynchronous SRAM

- Synchronous SRAM
  - Operations are synched with clock,
  - i.e., controls, address, data are accepted at the rising (of falling) edge of clock

- Asynchronous SRAM
  - WEN = 1, whenever address changes, new data read
  - WEN = 0, whenever the WEN is disabled, new data is written

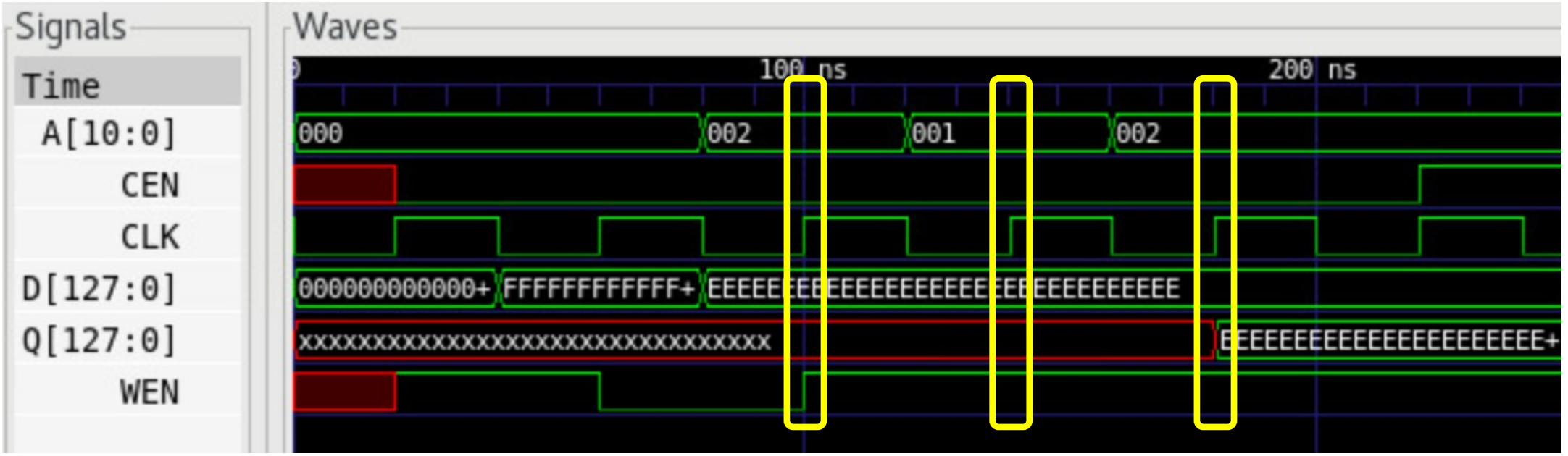
Memory Layout Considerations



- Memory layout is (of course) square box
- If the same capacity of memory is decomposed into 2 banks, the area becomes more than twice
- It is preferred the data pitch matches between memory vs. processor
- Memory compiler-generated layout uses low metals (e.g., up to M5), so you can route over memory

https://silvaco.com/foundation-ip/embedded-memory-compilers/

[Example] Memory Write and Read



1. “WEN=0 & ADD = 002 & D = EEEE..” are received by SRAM
2. “WEN=1 & ADD = 001” are received by SRAM
3. “WEN=1 & ADD = 002” are received by SRAM

From tb, it is good to apply new input at CLK = 0, or pass through register

[Example] Memory Write and Read

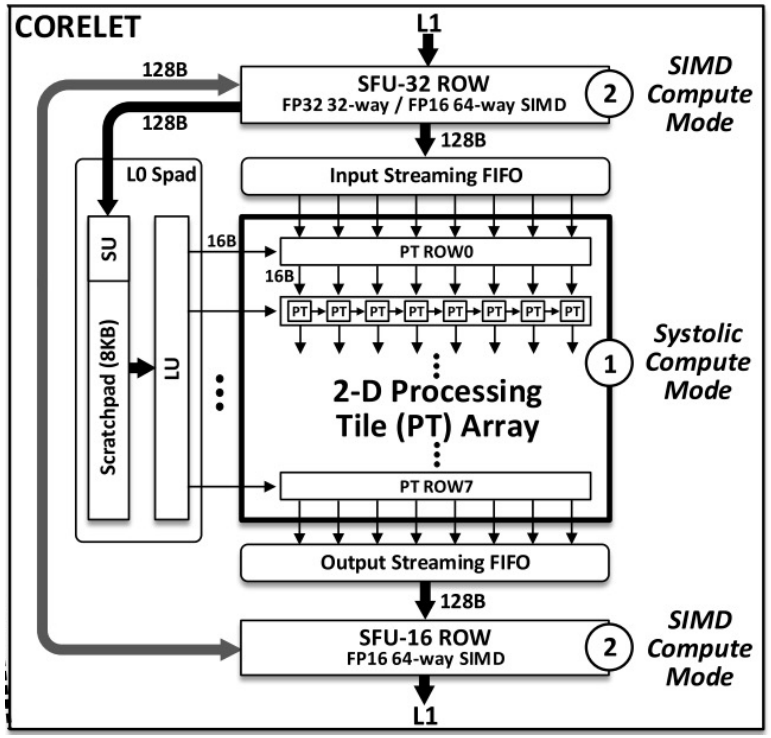
- Write ‘data.txt’ file contents into the memory at the address of 0 – 15.
- Then, read the memory from the address of A = 0 – 15.
- Compare the read data is the same as expected in the ‘data.txt’ file in the testbench

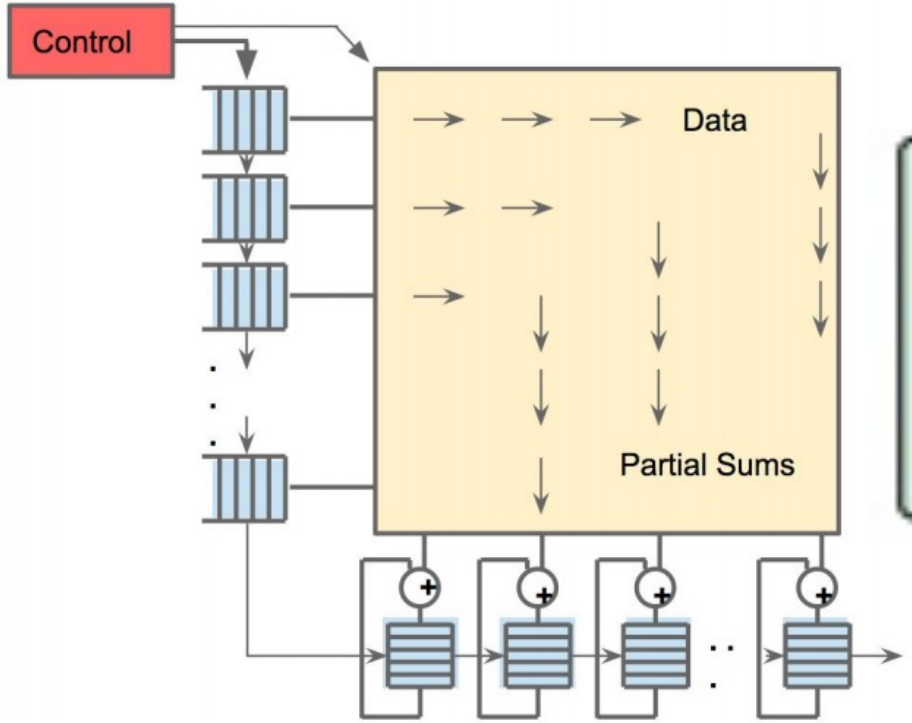
[HW2] Memory Write and Read with Double Buffering

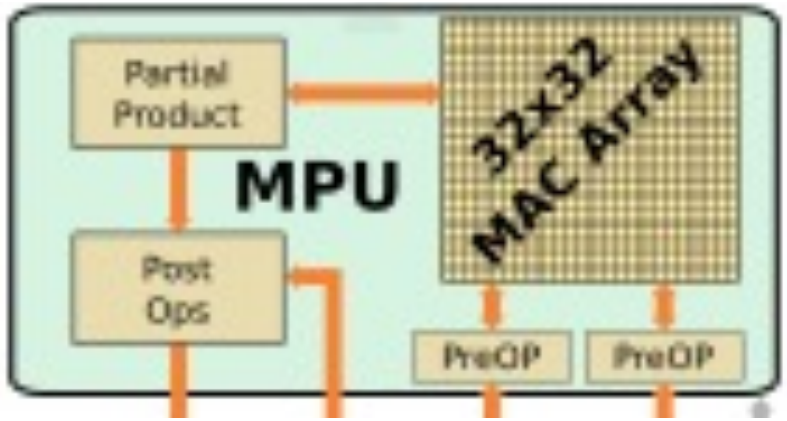
- You need to write 32 contents and read subsequently
- But, your memory has a limited space with 16 words.
- You have two SRAMs, EVEN and ODD memories.
- Write 0^{th} – 15^{th} contents into the EVEN memory at the address of 0 – 15.
- Then, read the memory from the address of A = 0 – 15 from EVEN memory while writing the 16^{th} – 31^{th} contents into ODD memory simulataneously.
- Finally, read the 16^{th} – 31^{th} contents from ODD memory.

Project Description

2D Systolic-based Architecture



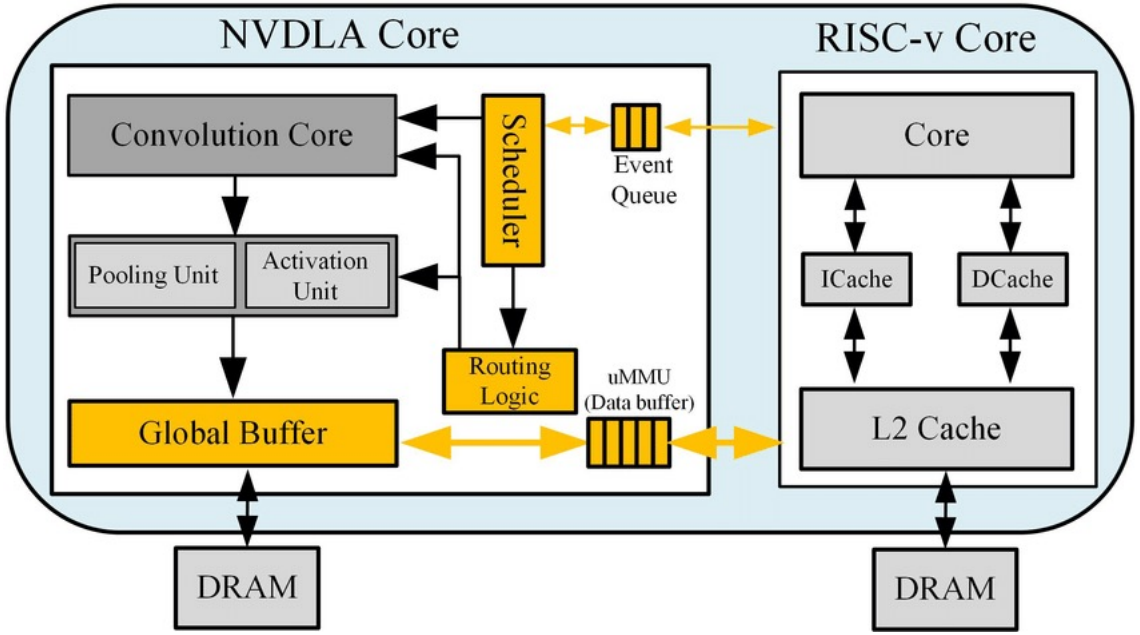


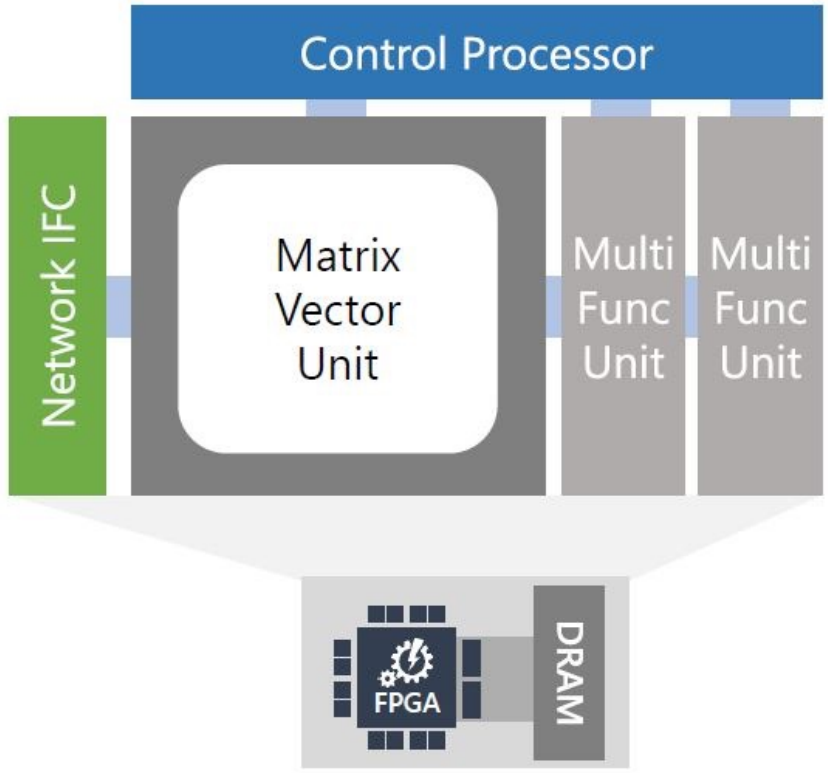


IBM Rapid **Google TPU** **Intel Nervana**

J.Oh, “A 3.0 TFLOPS 0.62V Scalable Processor Core for High Compute Utilization AI Training and Inference“
https://arxiv.org/pdf/1704.04760.pdf
https://fuse.wikichip.org/news/3270/intel-axes-nervana-just-two-months-after-launch/

1-D Vector Processor based Architecture



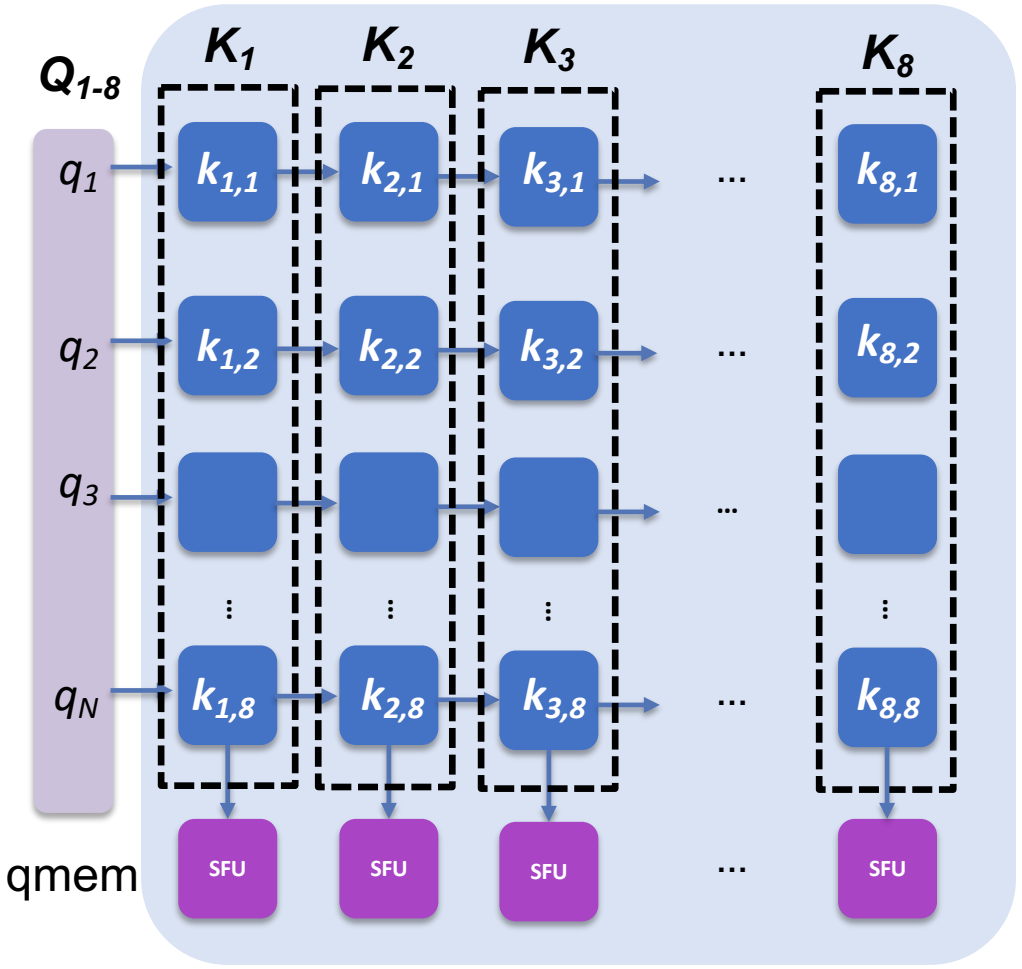


NVIDA NVDLA

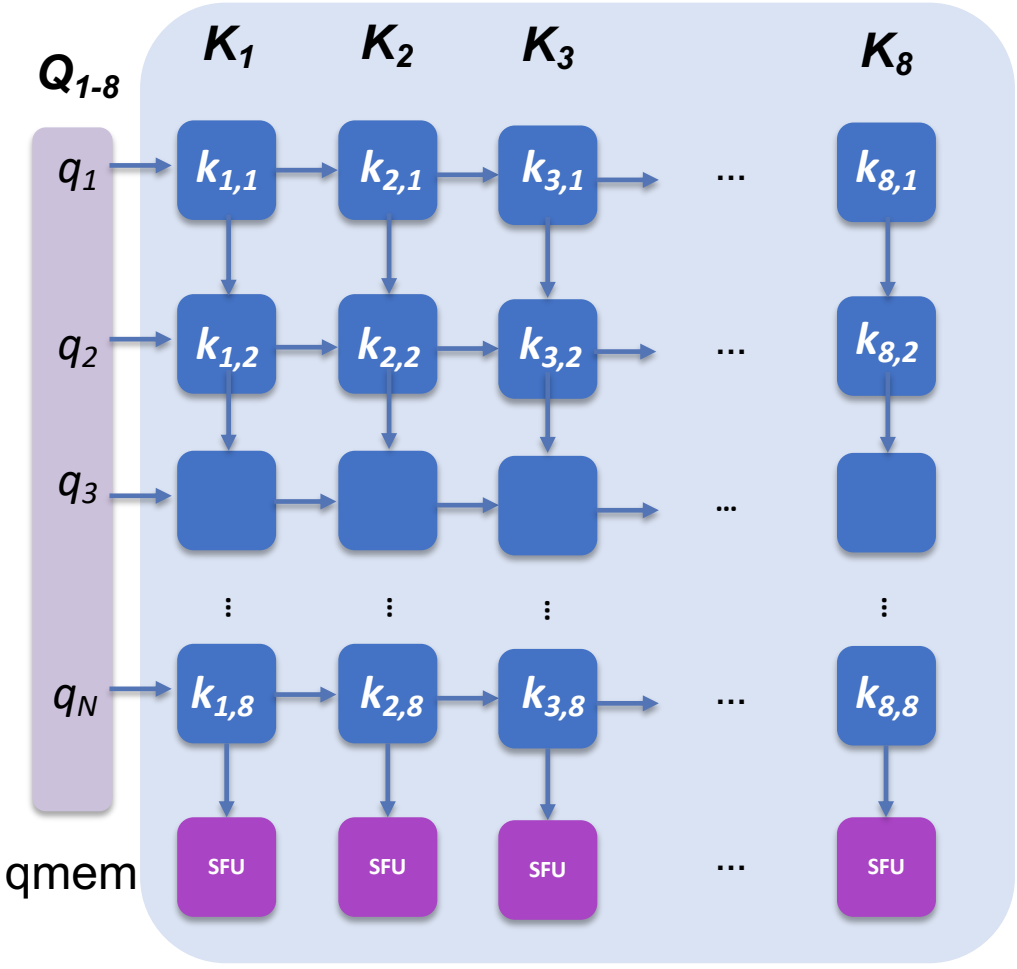
Microsoft Brainwave

https://www.researchgate.net/figure/Elastic-NVDLA-and-RISC-V-implement-DNNGuard-architecture_fig4_339920895
https://www.servethehome.com/microsoft-shows-off-project-brainwave-persistent-inferencing-from-fpga-cache/microsoft-brainwave-soft-dpu-architecture/

Architecture

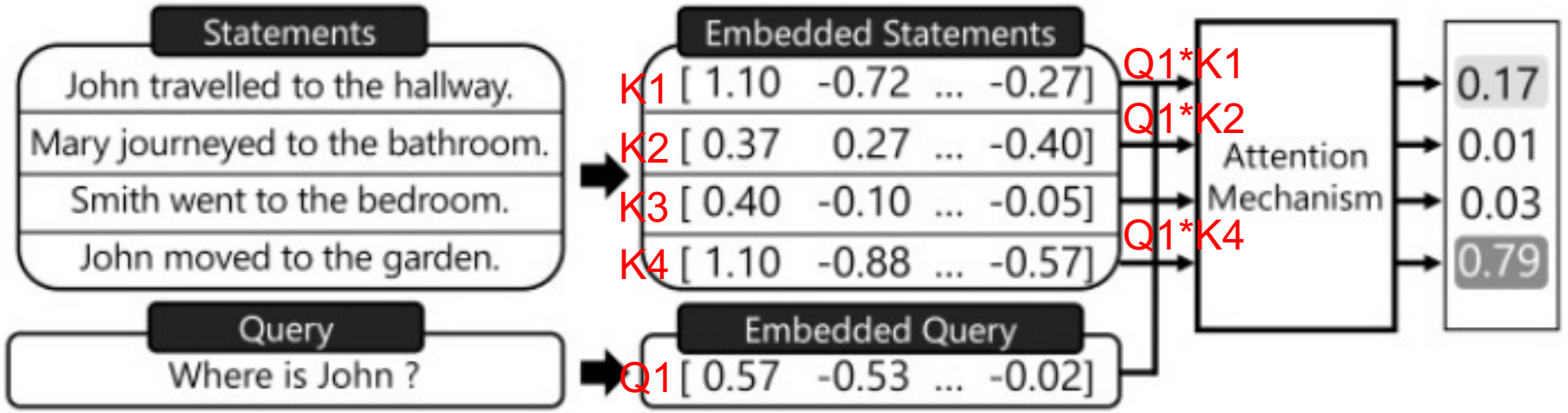


1-D vector processor architecture



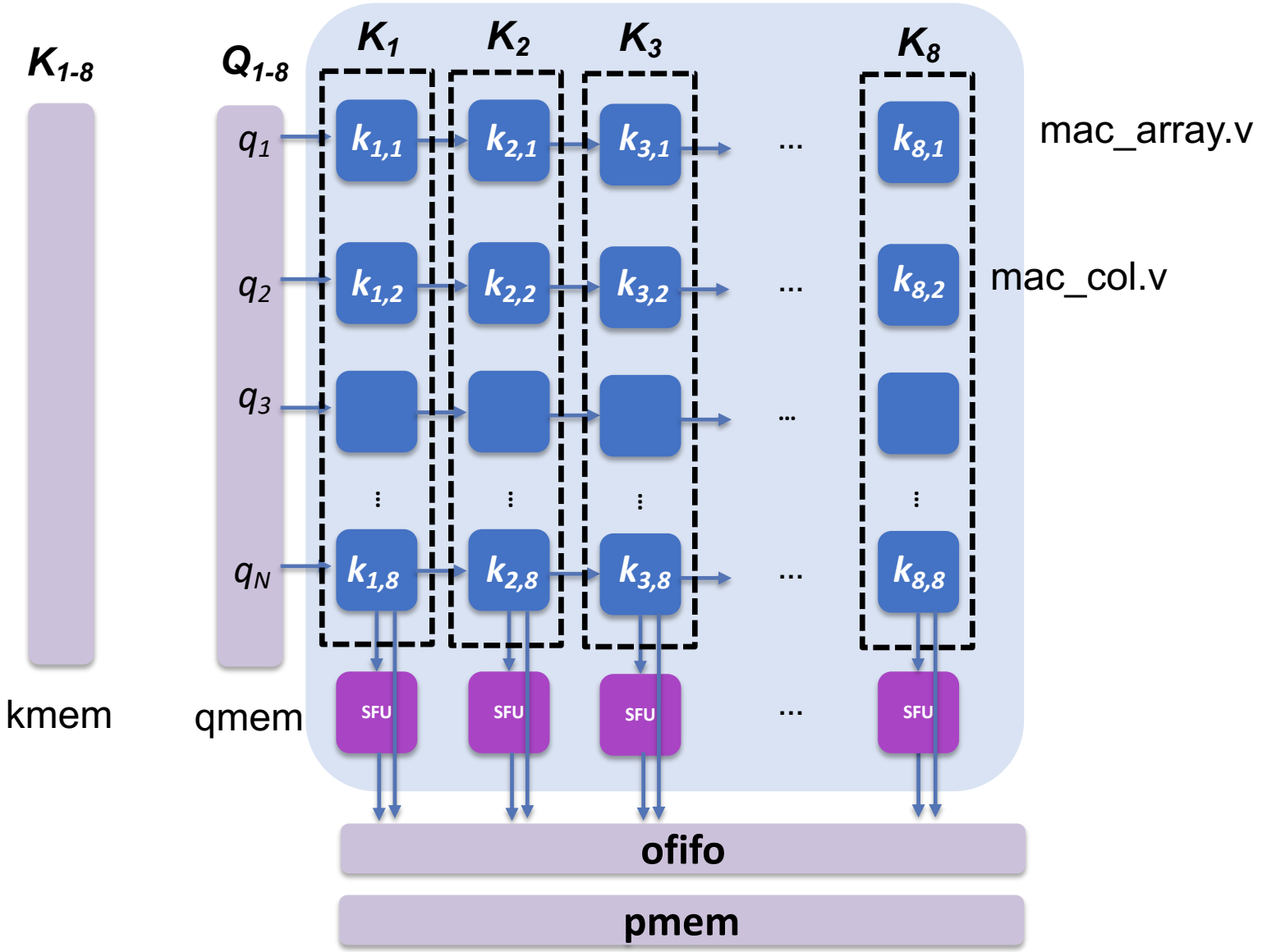
2D-systolic array architecture

Front-end Computation in Transformer Model for Natural Language Processing

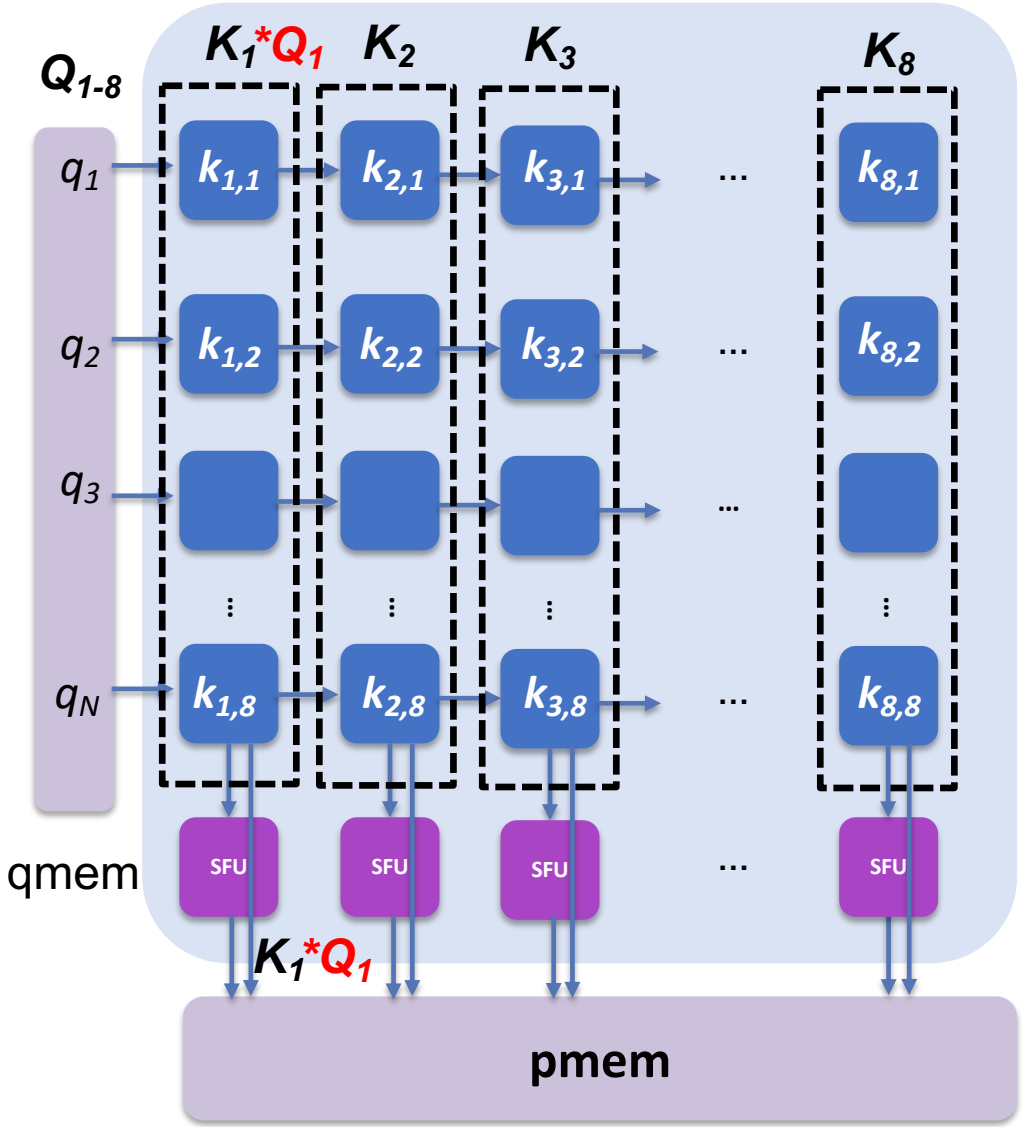


Facebook bAbi QA task processing

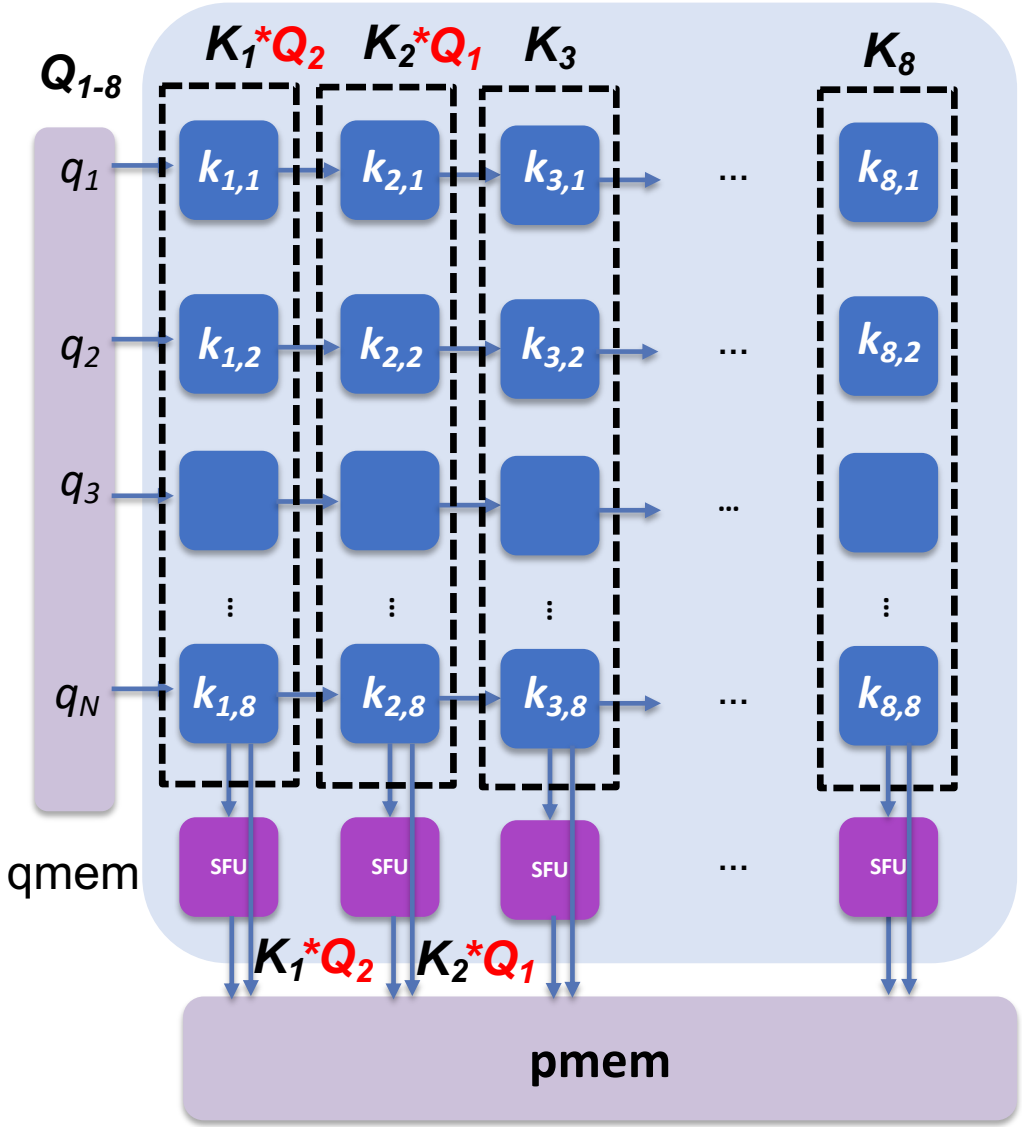
Architecture



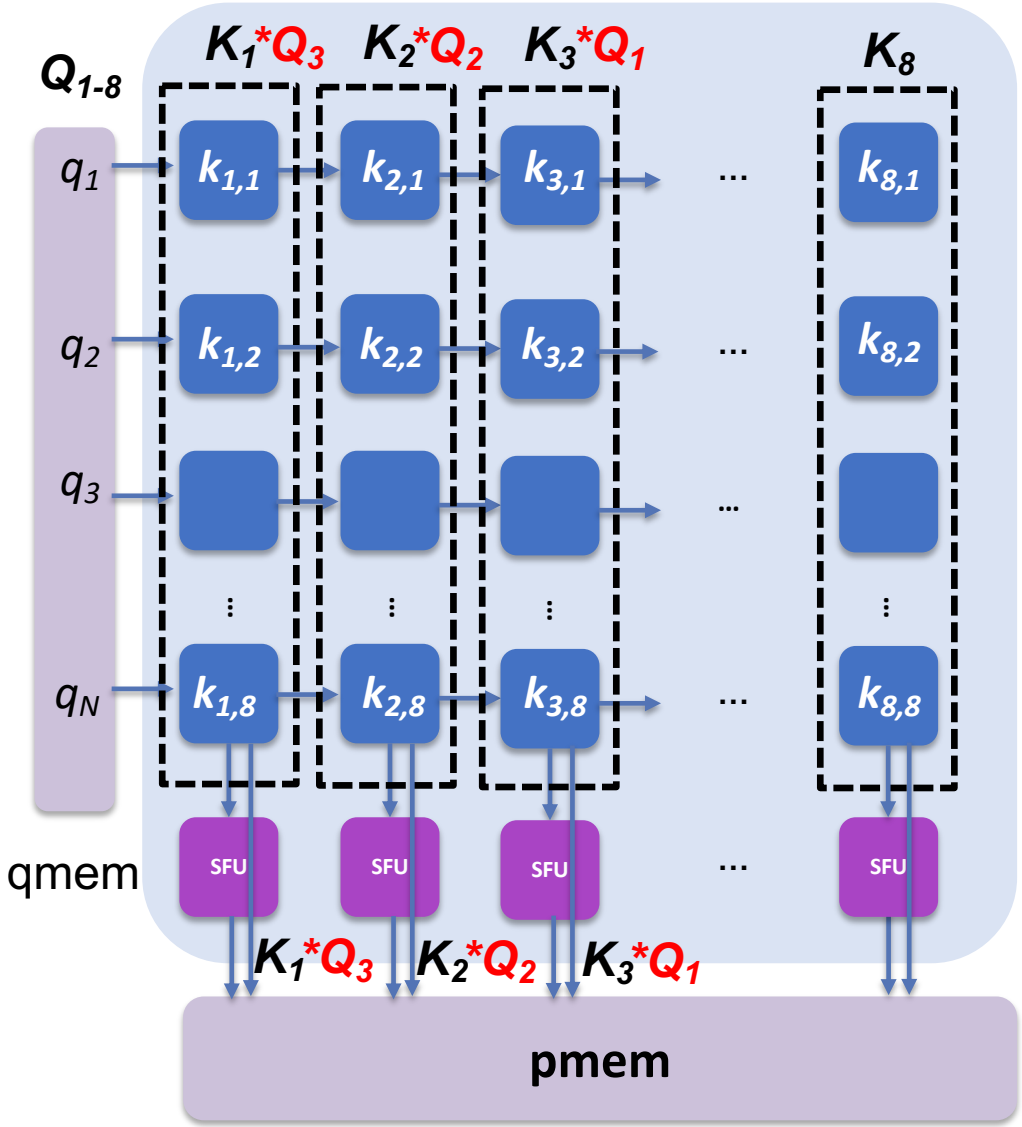
Computation



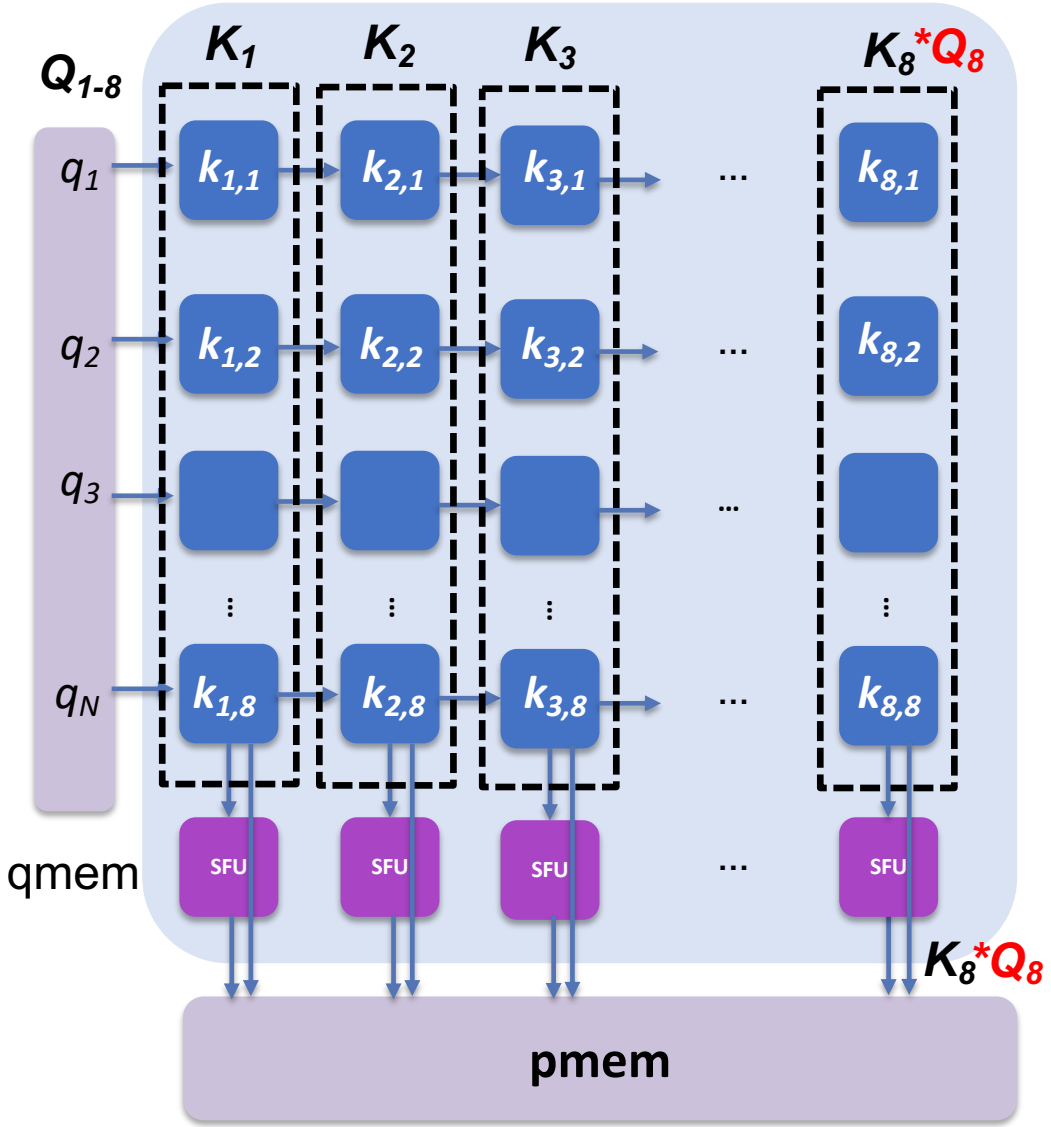
Computation



Computation



Computation



Processing Stage

- Stage1: memory write stage

  - qmem & kmem: fill both memory with “qdata.txt” and “kdata.txt”

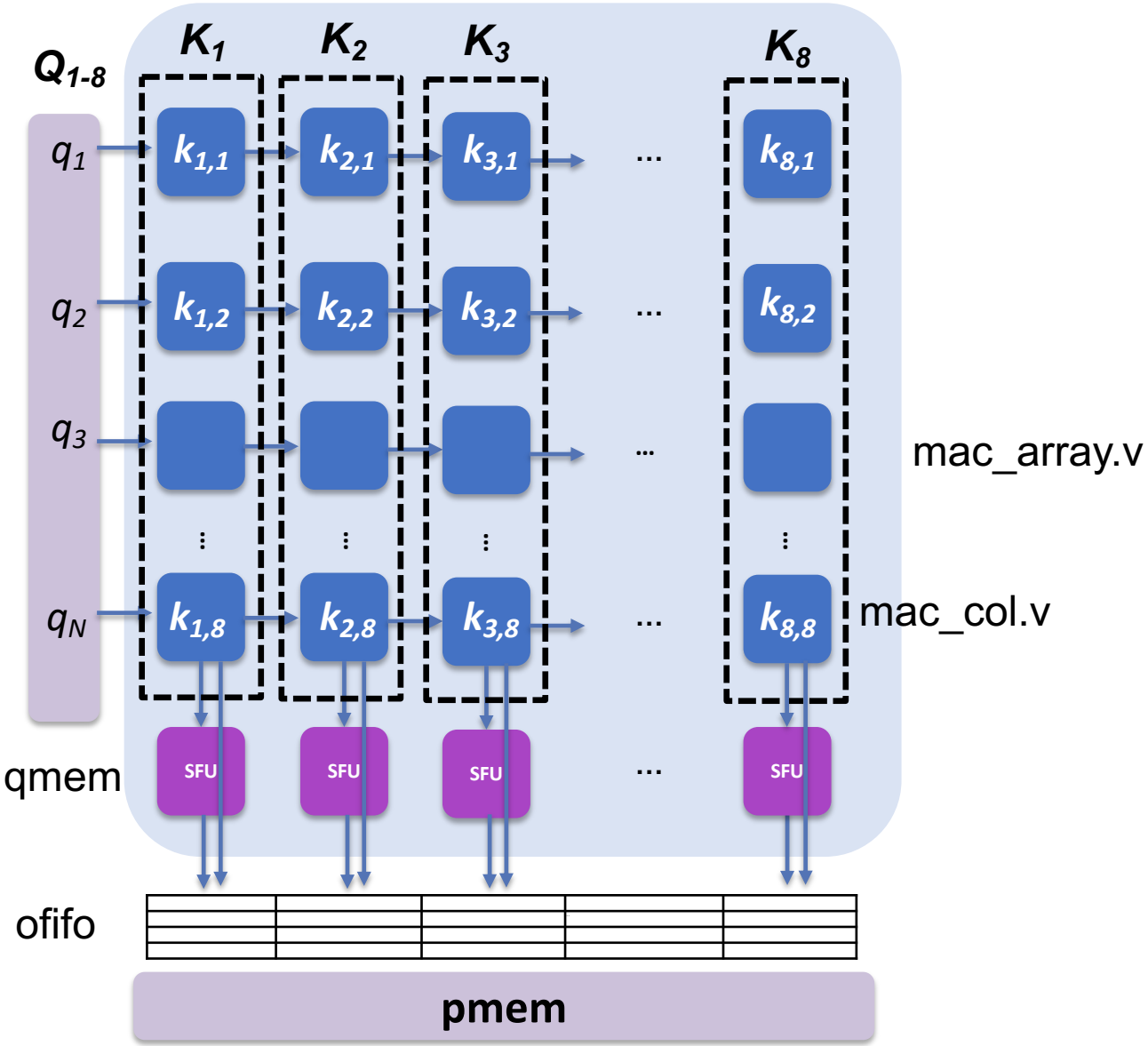
- Stage2: k-data loading stage

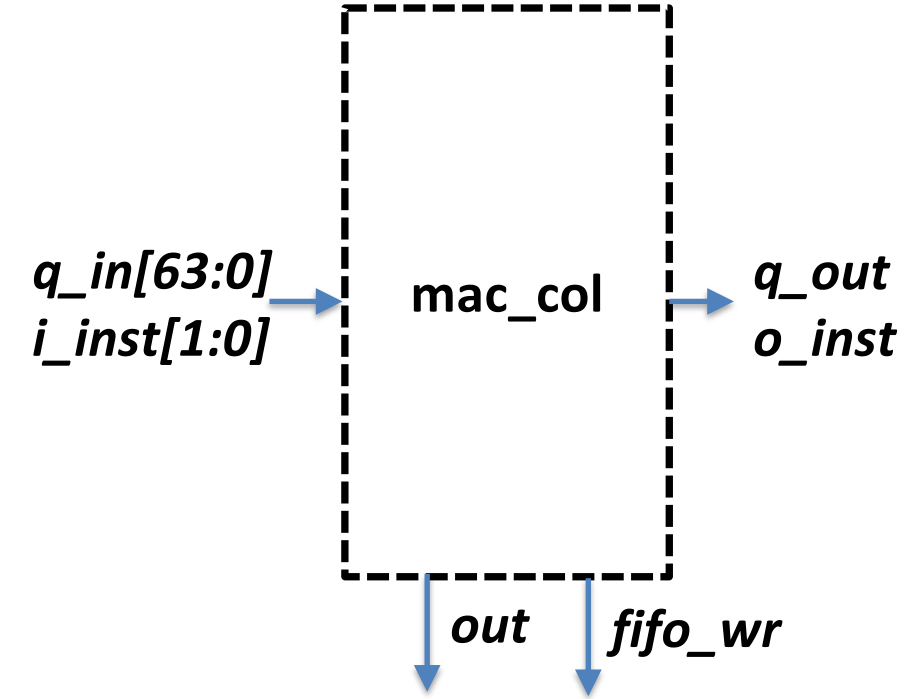
  - move the data from kmem to process’s local register

- Stage3: execute

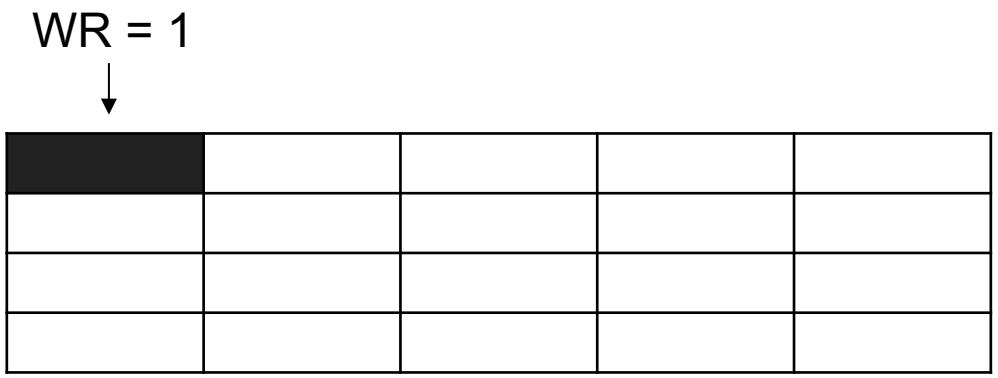
  - move the data from qmem to input of the processor

More Detail

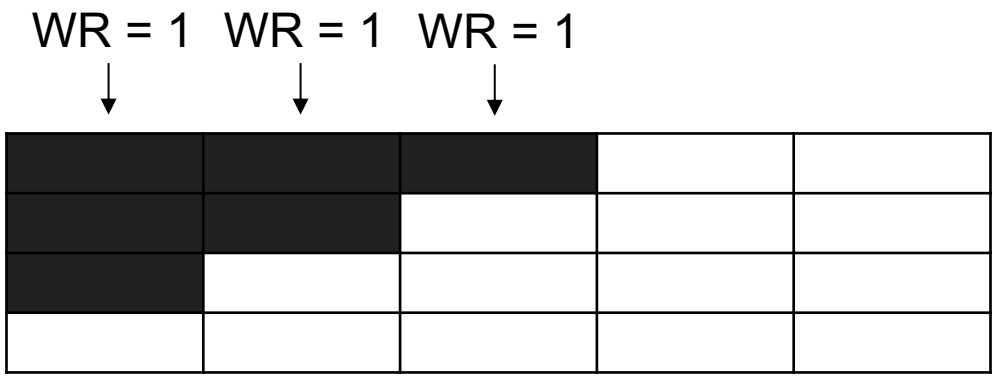

K_1
K_2
K_3
K_8
$Q_{1\text{-}8}$
q_1
q_2
q_3
q_N
$k_{1,1}$ $k_{2,1}$ $k_{3,1}$ ... $k_{8,1}$
$k_{1,2}$ $k_{2,2}$ $k_{3,2}$ ... $k_{8,2}$
$k_{1,8}$ $k_{2,8}$ $k_{3,8}$ ... $k_{8,8}$
SFU
qmem
ofifo
pmem
mac_array.v
mac_col.v


q_in[63:0]
i_inst[1:0]
mac_col
q_out
o_inst
out
fifo_wr

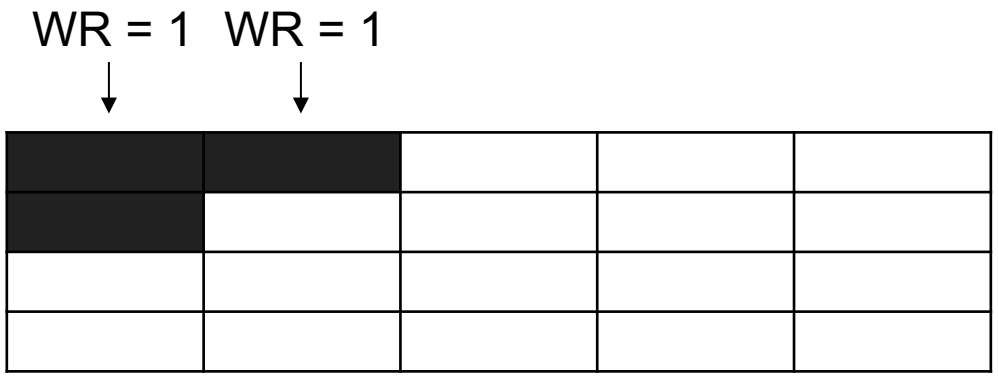
OFIFO Operation



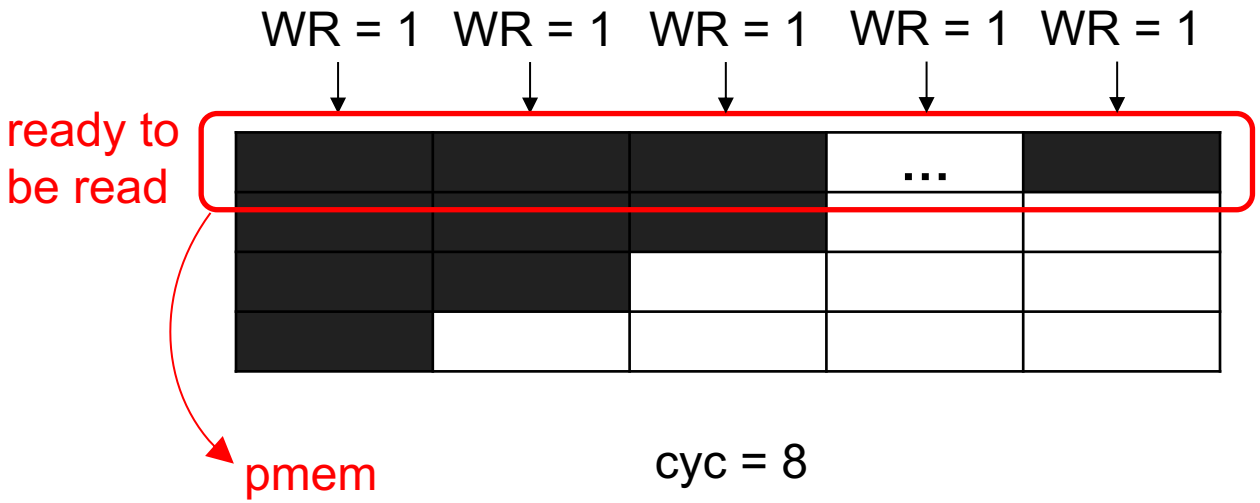
cyc = 1



cyc = 3

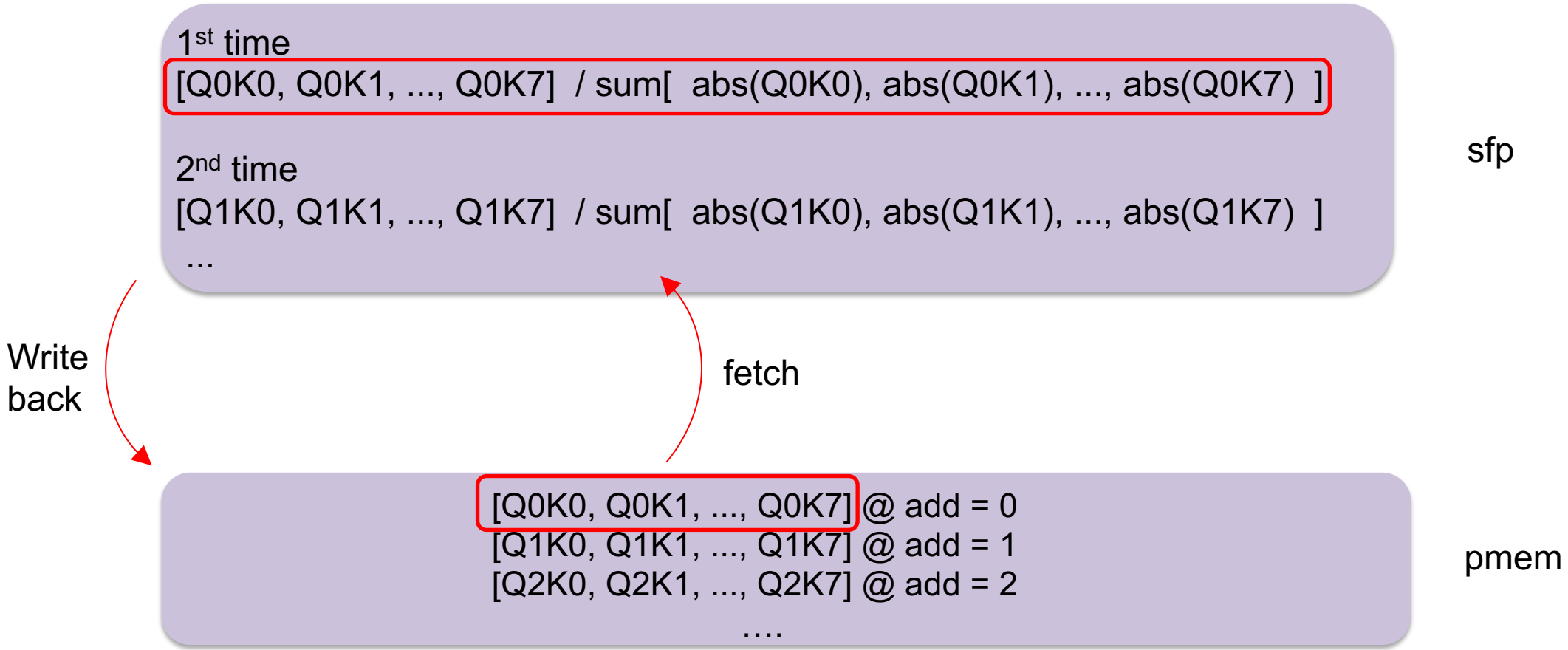


cyc = 2

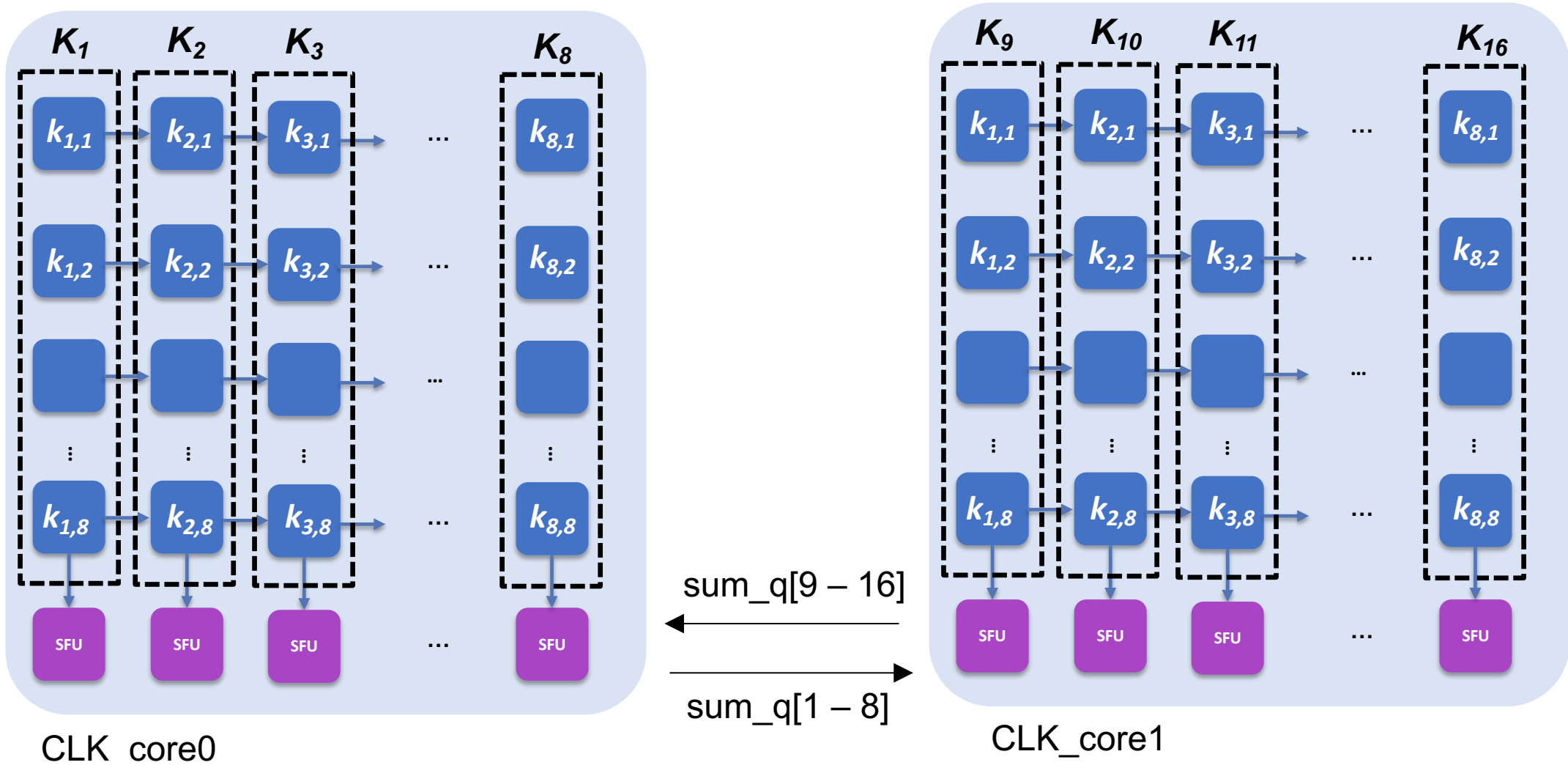


cyc = 8

Normalization



Extension for Dual Core



Normalize by "sum(abs(Q0K0), abs(Q0K1), ..., abs(Q0K7)) from core0
+ sum(abs(Q0K8), abs(Q0K9), ..., abs(Q0K15)) from core1"

Optimization

- Optimize the implementation to maximize the frequency and minimize power
  - However, the project is not a spec competition
- Apply techniques you have learned from this course
- Potential optimization (does not mean you need to do everything below)
  - Pipeline, multi-cycle path, memory double-buffering, clock gating, loop unrolling, more # of cores, any other creative idea
  - Async interface handling (synchronizer, hand shaking, ….)
  - Maximize parallelism between processing stages
  - Design your own controller
  - better verification (with random input)
  - Minimize WNS and DRC errors

How to prove ?

- Prove the efficacy of your idea (why ? how ? how much ?)
  - measure power and check functionality with VCD

    (SDF is not required, but it will be plus alpha)
  - clear comparison before vs. after
  - clear description in the report regarding your design choice

    e.g., why you chose the decision, why you cut the pipeline boundary, ….
  - report is VERY important

Project Policy

- TA & Instructor **will not**
  - solve your project instead of you
  - debug your code (e.g., I got this error with screenshot, can you check my code ?)
  - analyze your issue (e.g., my performance is not so good, do you know why ?)
  - comment on your result (e.g., Can you check my result is correct ?, Can I get A grade with this optimization ?)
  - “fighting against errors” is the biggest part of this project

- However, TA & Instructor will
  - clarify the project problem
  - help any environment issue (e.g., it seems there is no license file, tool does not load, …)
  - provide general guideline for a certain issue

- You can still discuss between students through discussion tab