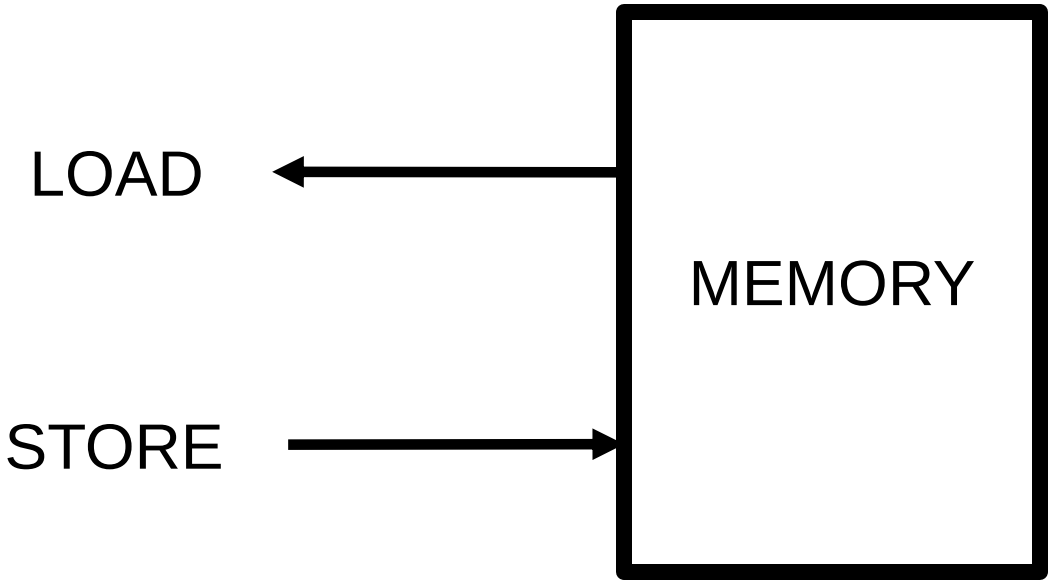
Κύρια Μνήμη
(Main Memory)

Πηγές/Βιβλιογραφία

- “Memory Systems: Cache, DRAM, Disk”, Bruce Jacob, Spencer W. Ng, David T. Wang, Morgan Kaufmann Publishers, 2008

- Onur Mutlu, “Main Memory & DRAM Fundamentals”, Computer Architecture - Lecture 4 – ETH, 2017 (slides & video)
  - https://safari.ethz.ch/architecture/fall2017/lib/exe/fetch.php?media=onur-comparch-fall2017-lecture4-mainmemoryanddramfundamentals-afterlecture.pdf
  - https://www.youtube.com/watch?v=sZjSBFceV_o

Η μνήμη για τον προγραμματιστή



Ιδανική Μνήμη

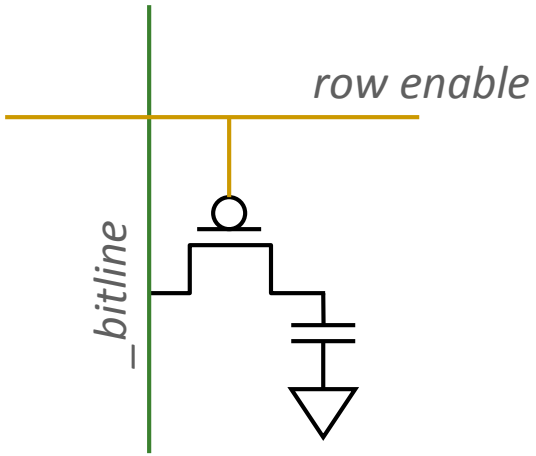
- Μηδενικό κόστος προσπέλασης (zero latency)
- Άπειρη χωριτικότητα (infinite capacity)
- Άπειρο εύρος ζώνης (infinite bandwidth)
- Μηδενικό κόστος ($) (zero cost)

Πρόβλημα

- Οι απαιτήσεις της ιδανικής μνήμης είναι αντικρουόμενες
- Μεγάλη χωρητικότητα → αυξημένος χρόνος προσπέλασης
- Μικρός χρόνος προσπέλασης → αυξημένο κόστος ($)
- Μεγάλο εύρος ζώνης → αυξημένο κόστος ($)

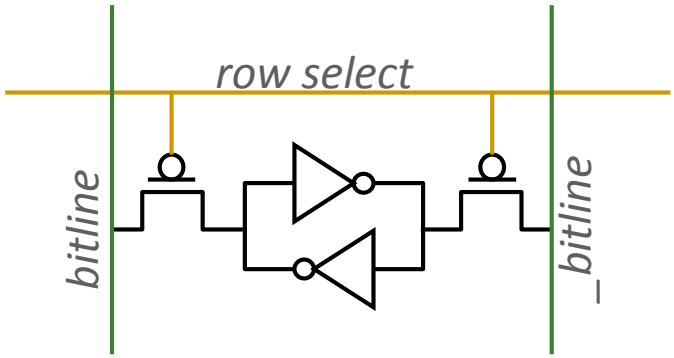
Τεχνολογίες Μνήμης: DRAM

- Dynamic random access memory (DRAM)
  - Η λογική τιμή == φορτίο πυκνωτή
    - » 1 ο πυκνωτής ειναι φορτισμένος
    - » 0 ο πυκνωτής είναι εκφορτισμένος
  - Μικρό δομικό στοιχείο → Μεγάλη πυκνότητα (higher density)
  - Πυκνωτής → Αργή προσπέλαση (slower access)
  - Χρειάζεται αναζωογόνηση (refresh)
    - » ρεύματα διαρροής → ο πυκνωτής μπορεί να αποφορτιστεί σε μερικά ms



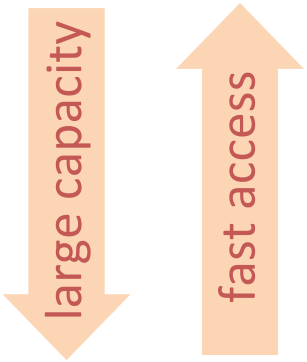
Τεχνολογίες Μνήμης: SRAM

- Static random access memory (SRAM)
  - Η λογική τιμή αποθηκεύεται σε διάταξη 2 αντιστροφέων διασταυρωμένης σύζευξης
  - Μεγάλο δομικό στοιχείο (6T) → Μικρή πυκνότητα (lower density)
  - Όχι πυκνωτής → Γρήγορη προσπέλαση (faster access)
  - Δε χρειάζεται αναζωογόνηση (no refresh)

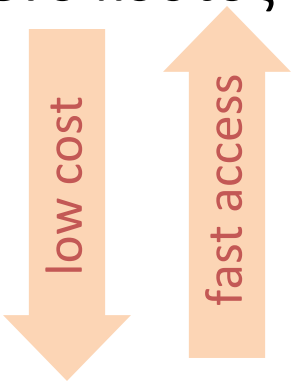


Τεχνολογίες μνήμης – Απαιτήσεις μνήμης Tradeoffs

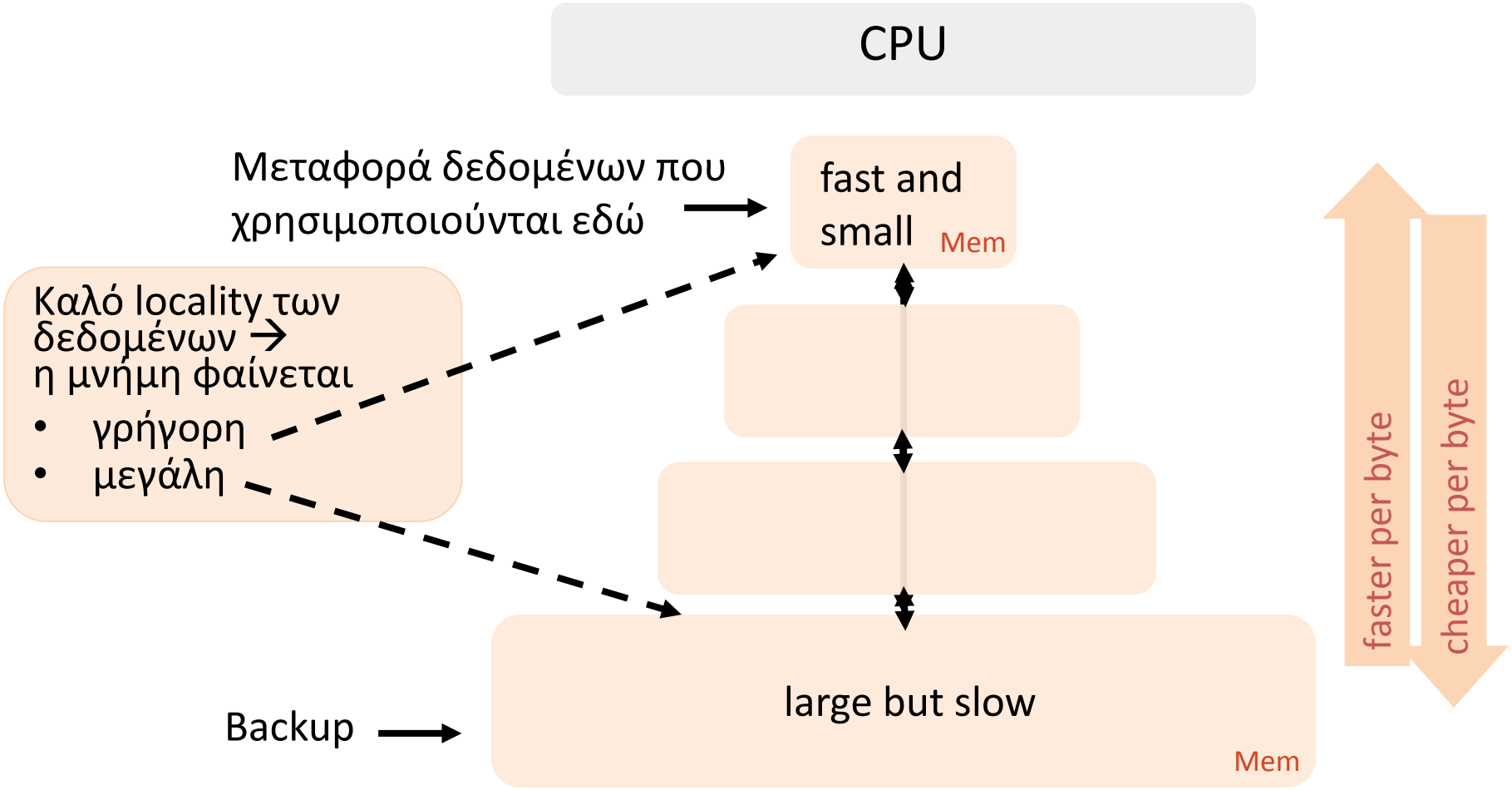
- Μεγαλύτερη χωρητικότητα → αργή προσπέλαση
  - SRAM, 512 Bytes, sub-nanosec
  - SRAM, KByte~Mbyte, ~nanosec
  - DRAM, Gigabyte, ~50 nanosec
  - Hard Disk, Terabyte, ~10 millisec



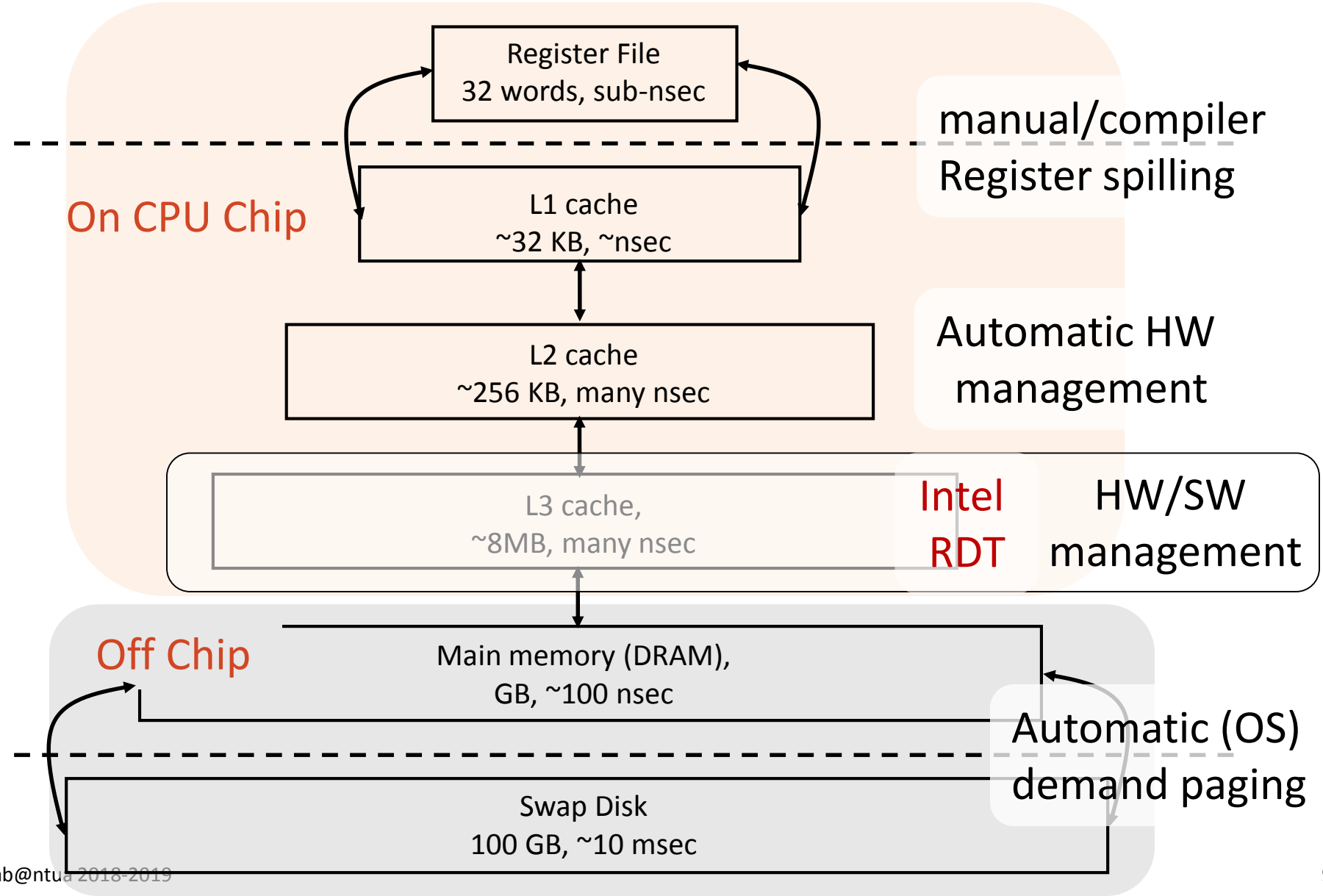
- Γρηγορότερη προσπέλαση → αυξημένο κόστος ($)
  - SRAM, < 10$ per Megabyte
  - DRAM, < 1$ per Megabyte
  - Hard Disk < 1$ per Gigabyte
  - Οι τιμές αλλάζουν με τον χρόνο, αλλά η “τάση” είναι η ίδια



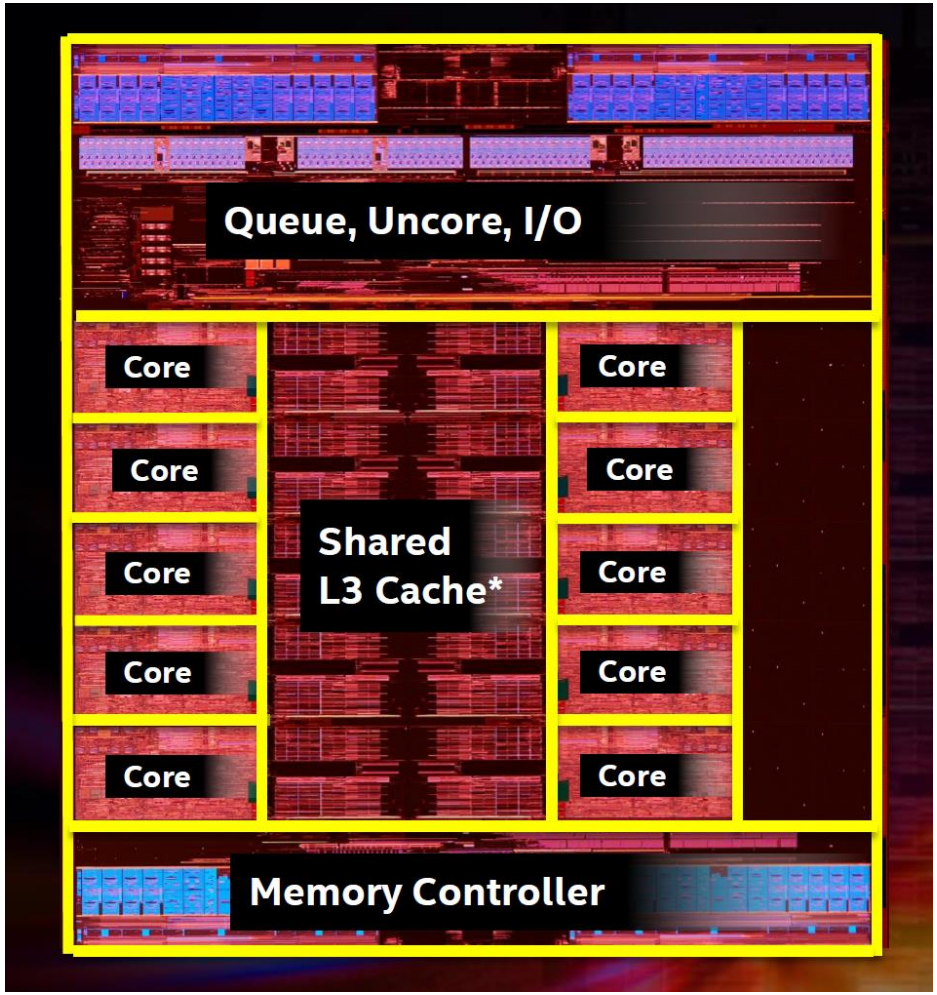
Ιεραρχία μνήμης



Ιεραρχία μνήμης

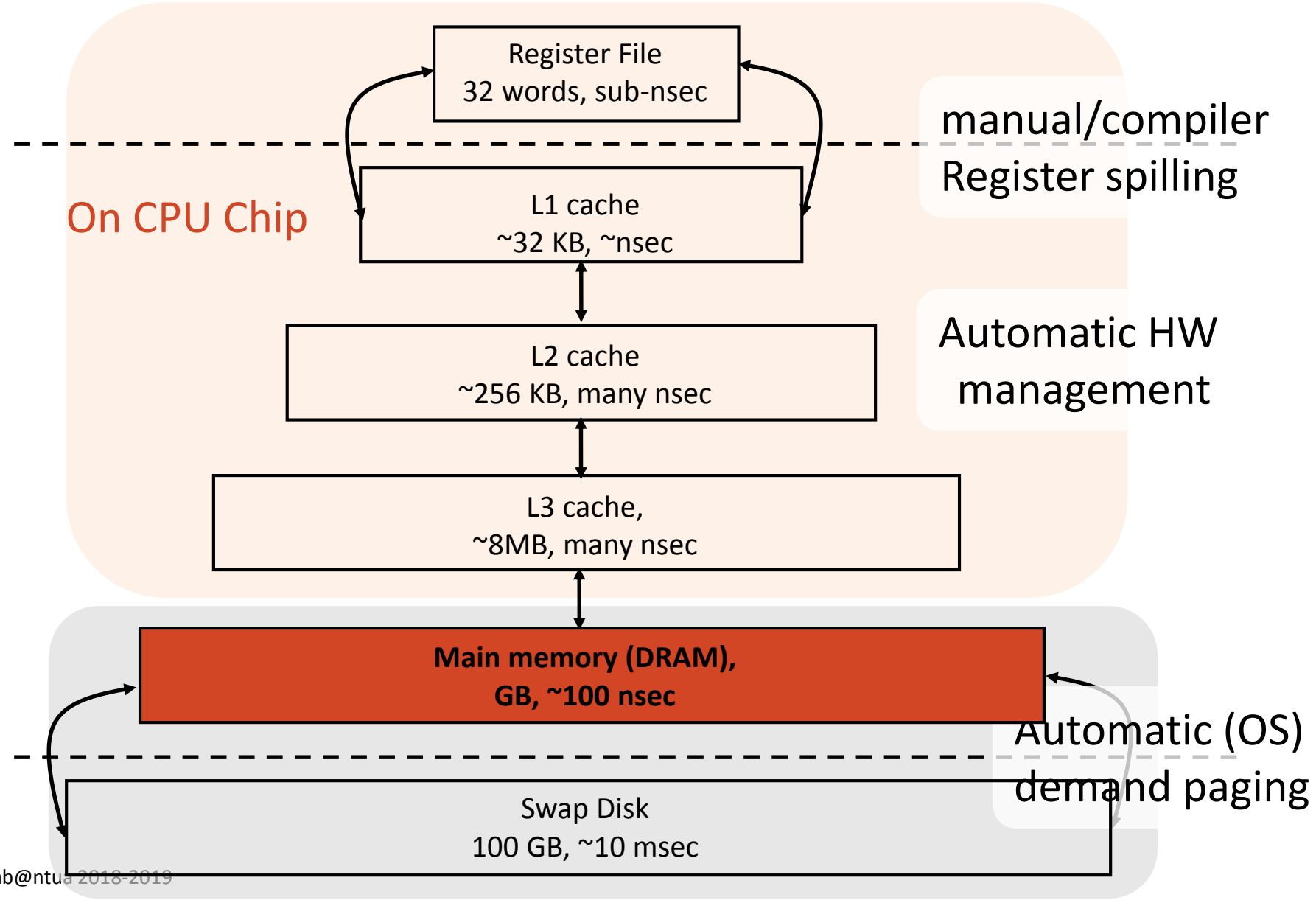


Η μνήμη σε ένα σύγχρονο σύστημα



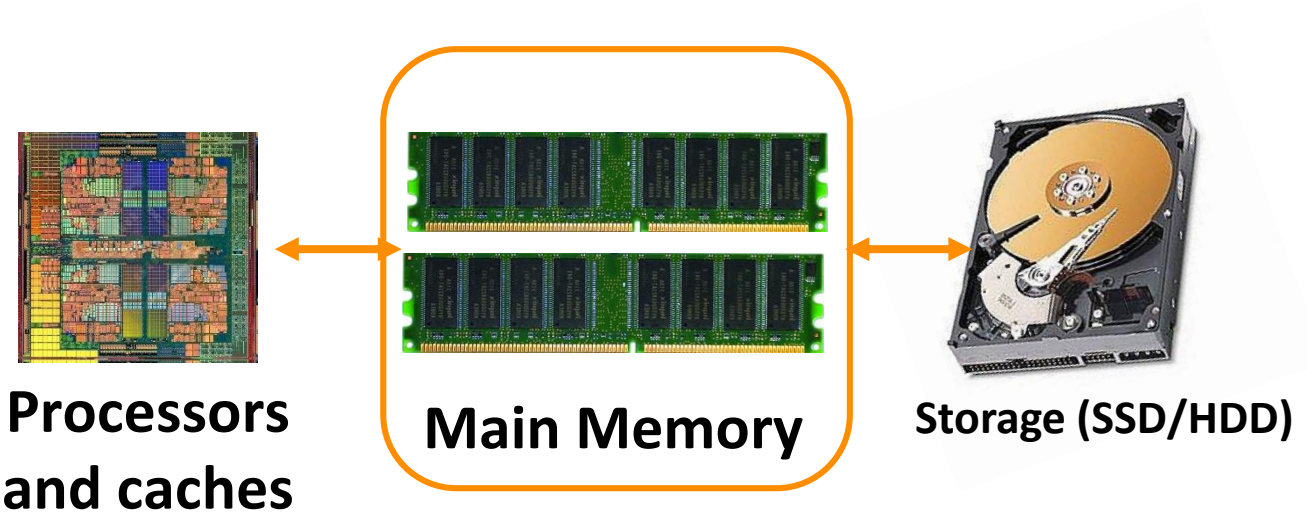
Intel Broadwell-E Core i7 [2016]

Ιεραρχία μνήμης



Γιατί η κύρια μνήμη είναι σημαντική?

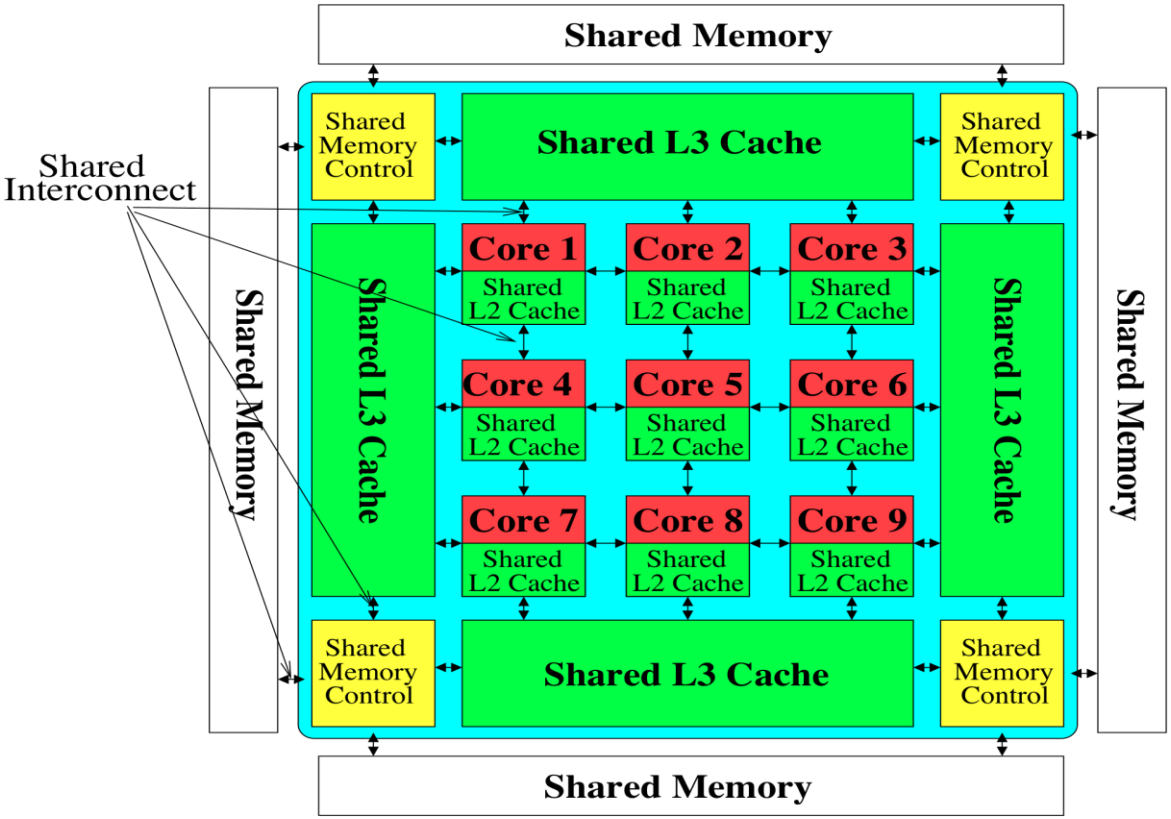
Το σύστημα κύριας μνήμης

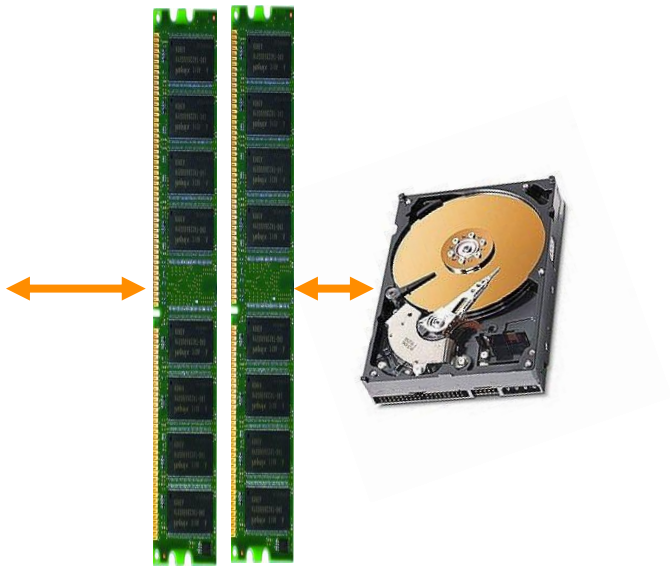


- Η κύρια μνήμη είναι κρίσιμο δομικό στοιχείο κάθε υπολογιστικού συτήματος: server, mobile, embedded etc.

- Το σύστημα της κύριας μνήμης πρέπει να κλιμακώνει (σε μέγεθος, τεχνολογία, απόδοση, κόστος, διαχείριση) για να διατηρούνται τα οφέλη της αυξημένης επίδοσης.

Η μνήμη ως διαμοιραζόμενος πόρος



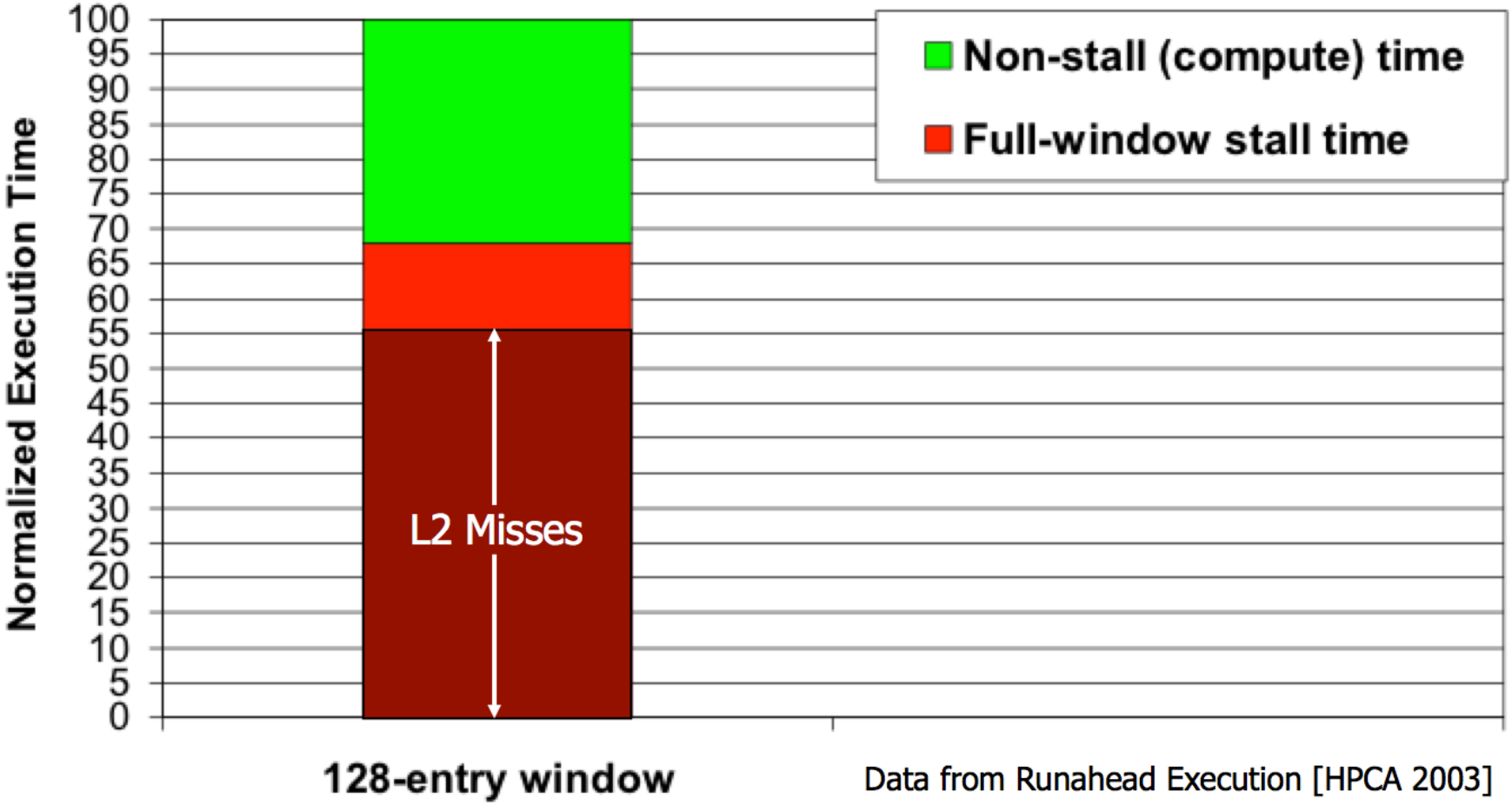


- Η **πλειοψηφία** των πόρων χρησιμοποιείται για την **αποθήκευση** και **μετάδοση** data

Επίδοση

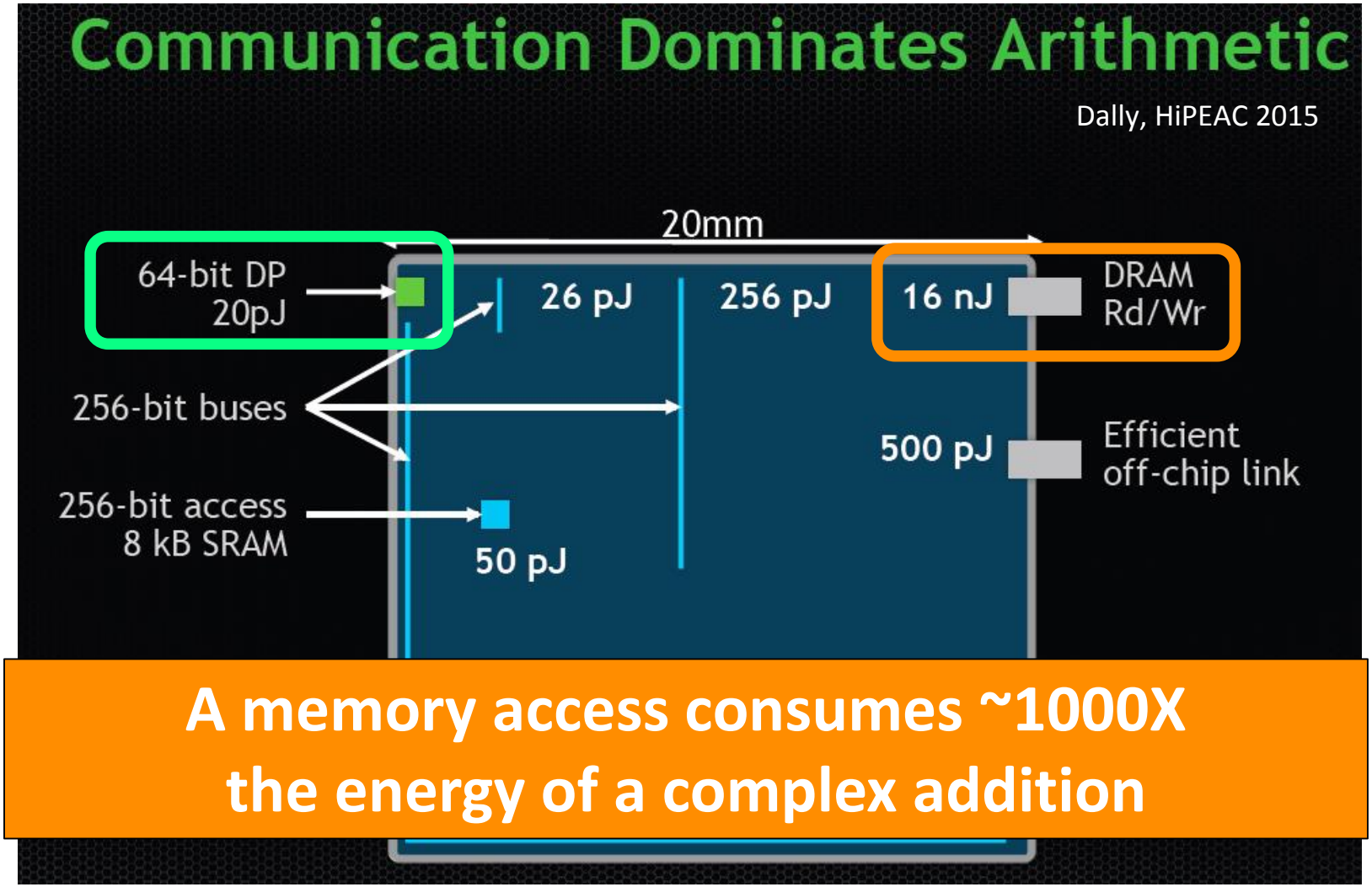
It's the Memory, Stupid!

- "**It's the Memory, Stupid!**" (Richard Sites, MPR, 1996)



Ενέργεια

Ενεργειακό Κόστος Επικοινωνίας

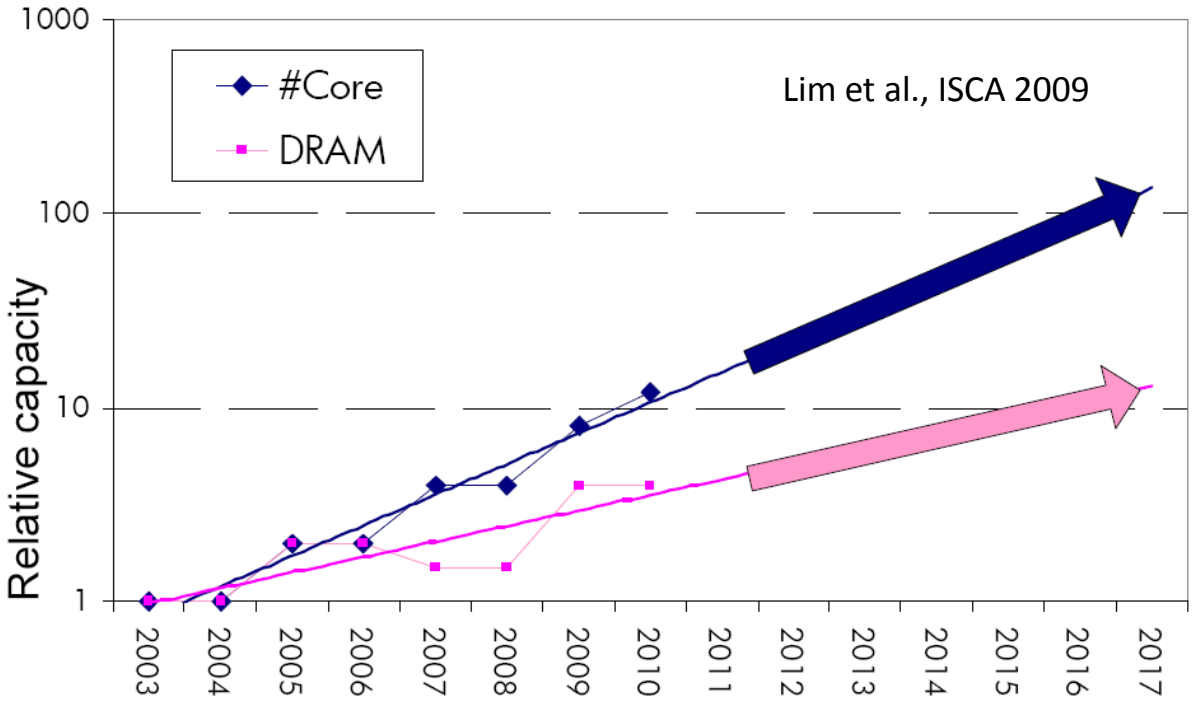


Σύγχρονες Ανάγκες/Προκλήσεις

- Ανάγκη για μεγαλύτερη χωρητικότητα (**capacity**), μεγαλύτερο εύρος διαύλου (**bandwidth**) και καλύτερη ποιότητα υπηρεσίας (**QoS**)
  - Multicore systems: cores ↑
  - Data-intensive applications: data ↑ (size, throughput)
  - Consolidation: ετερογένεια ↑

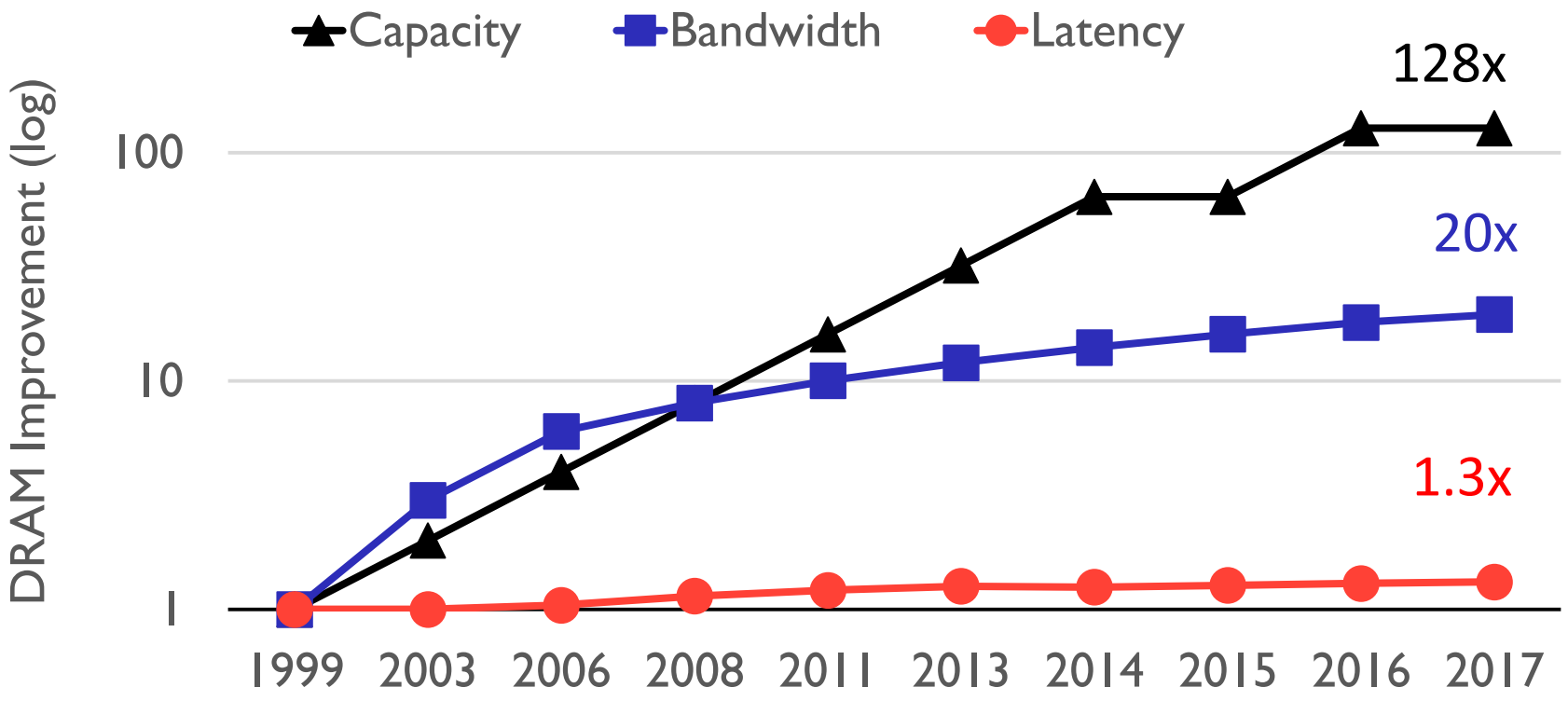
Χωρητικότητα Κύριας Μνήμης

Ο αριθμός πυρήνων διπλασιάζεται ~ κάθε 2 χρόνια
Η χωρητικότητα της DRAM διπλασιάζεται ~ κάθε 3 χρόνια



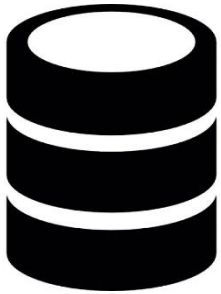
- Η χωρητικότητα μνήμης/πυρήνα αναμένεται να **μειώνεται κατά 30%** κάθε δύο χρόνια.
- Για το εύρος ζώνης τα πράγματα είναι χειρότερα...

Bandwidth & latency

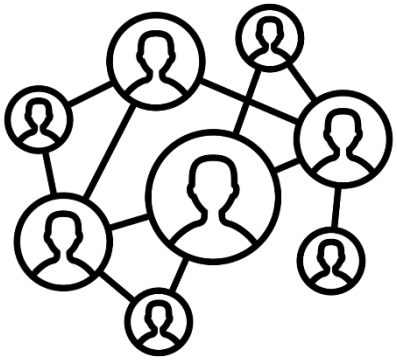


Ο χρόνος προσπέλασης παραμένει σχεδόν σταθερός

Γιατί ο χρόνος απόκρισης του συστήματος μνήμης ειναι κρίσιμός?



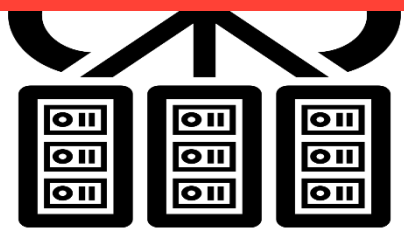
In-memory Databases



Graph/Tree Processing

Εφαρμογές ένταση δεδομένων – το σύστημα μνήμης γίνεται το σημείο συμφόρησης

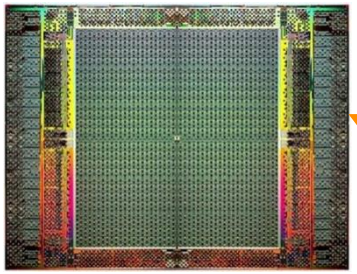
In-Memory Data Analytics



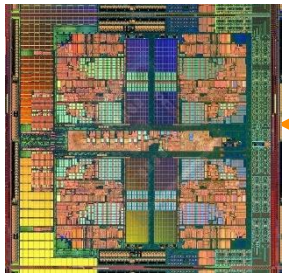
Datacenter Workloads

Η μνήμη ως διαμοιραζόμενος πόρος σε σύγχρονα συστήματα

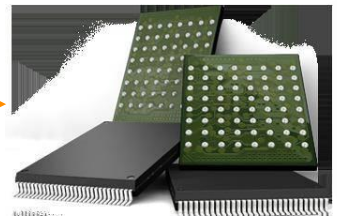
FPGAs



Αυξανόμενη πολυπλοκότητα και ετερογένεια!

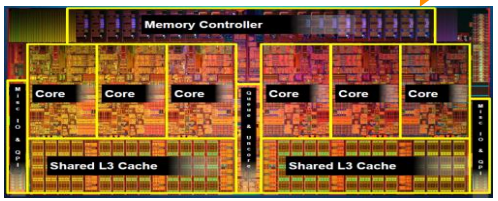


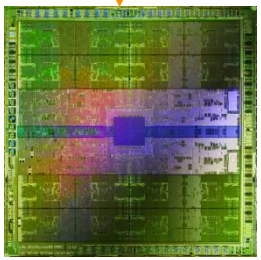
Hybrid Main Memory



Persistent Memory/Storage

Heterogeneous Processors and Accelerators





(General Purpose) GPUs

Σύγχρονες Ανάγκες/Προκλήσεις

- Ανάγκη για μεγαλύτερη χωρητικότητα (**capacity**), μεγαλύτερο εύρος διαύλου (**bandwidth**) και καλύτερη ποιότητα υπηρεσίας (**QoS**)
  - Multicore systems: cores ↑
  - Data-intensive applications: data ↑ (size, throughput)
  - Consolidation: ετερογένεια ↑
- Η κατανάλωση ενέργειας της κύριας μνήμης
  - ~40-50% energy in off-chip (IBM, 2003)

DRAM Power consumption

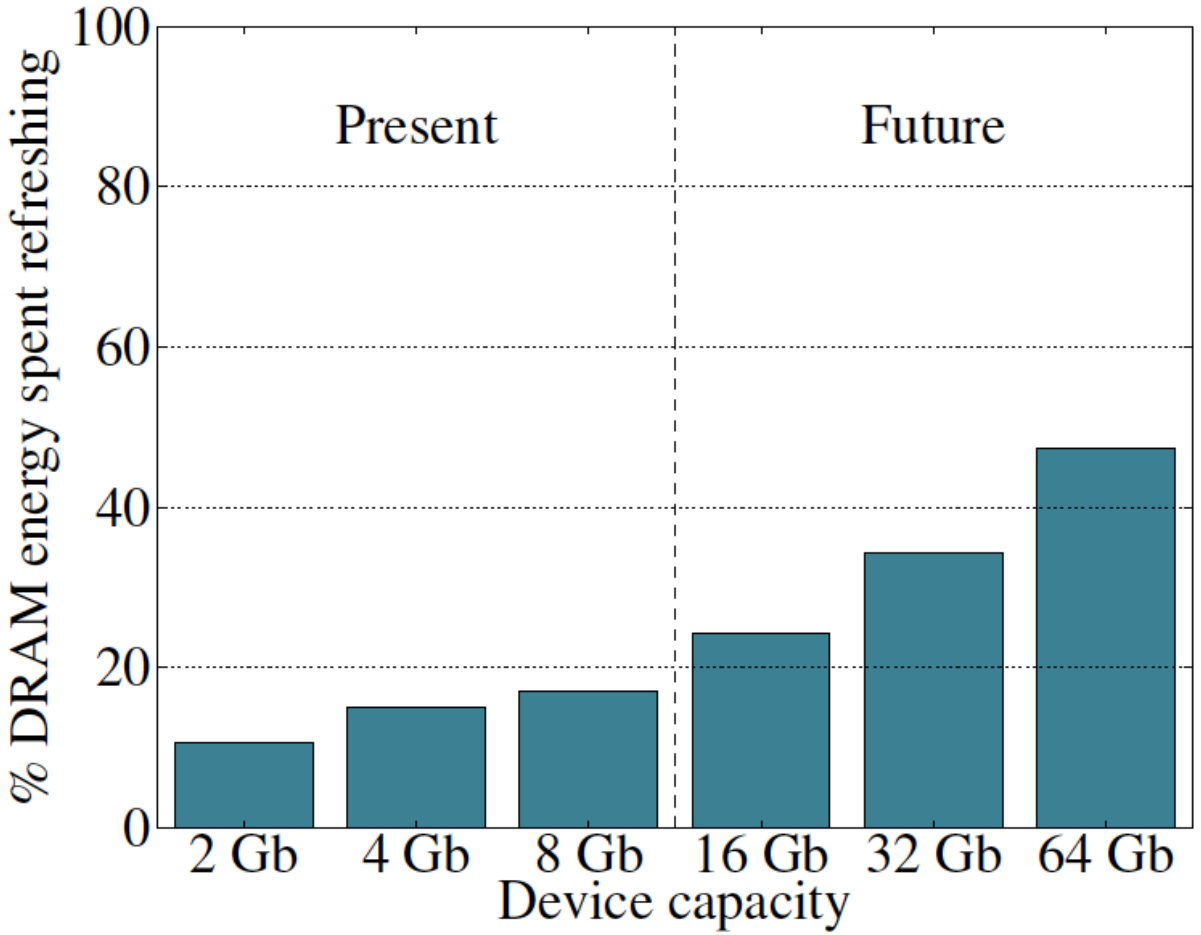
- Υψηλή κατανάλωση ισχύος σε σύγχρονα συστήματα

>40% in POWER7 (Ware+, HPCA'10)

>40% in GPU (Paul+, ISCA'15)

Ενεργειακό Κόστος DRAM Refresh

- Η DRAM καταναλώνει ενέργεια και όταν δε προσπελαύνεται



Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

Σύγχρονες Ανάγκες/Προκλήσεις

- Ανάγκη για μεγαλύτερη χωρητικότητα (**capacity**), μεγαλύτερο εύρος διαύλου (**bandwidth**) και καλύτερη ποιότητα υπηρεσίας (**QoS**)
  - Multicore systems: cores ↑
  - Data-intensive applications: data ↑ (size, throughput)
  - Consolidation: ετερογένεια ↑
- Η κατανάλωση ενέργειας της κύριας μνήμης
  - ~40-50% energy in off-chip (IBM, 2003)
  - >40% power in DRAM
  - Καταναλώνει ενέργεια και όταν δε χρησιμοποιείται
- Η κλιμάκωση (λόγω τεχνολογίας) «τελειώνει»

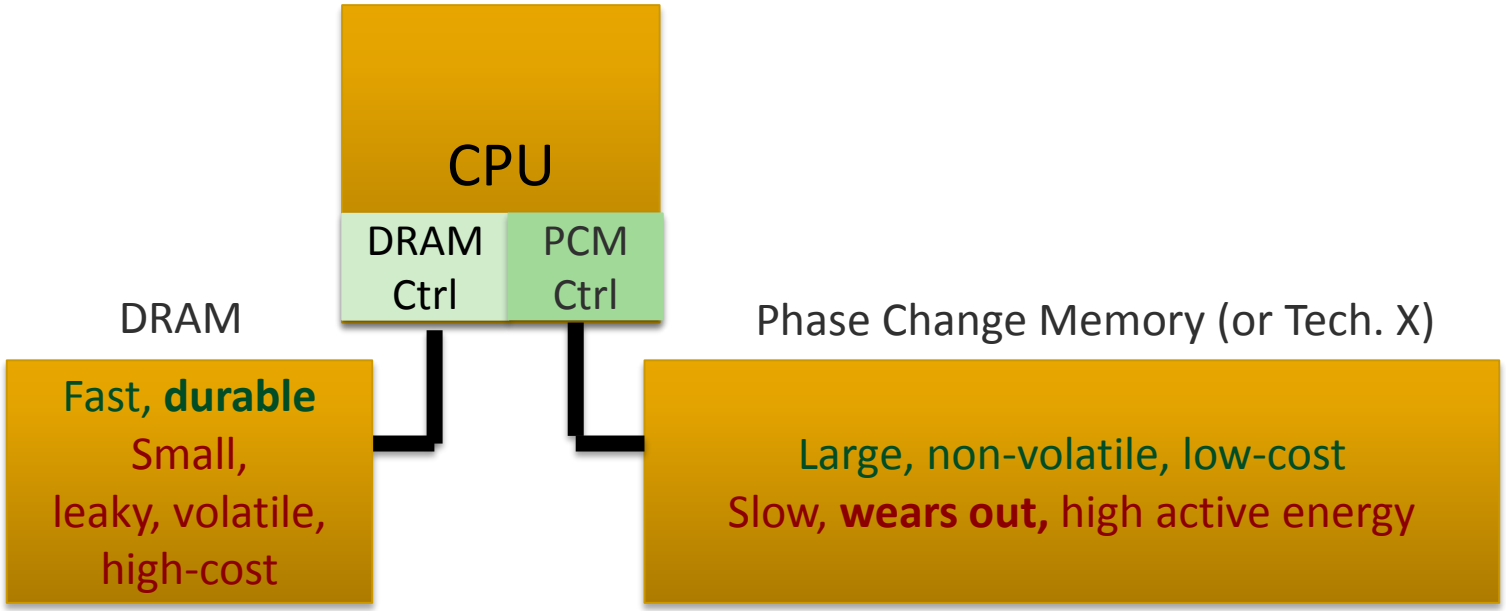
Κλιμακωσιμότητα DRAM

- Ο πυκνωτής πρέπει να είναι αρκετά μεγάλος για αξιόπιστη ανίχνευση φορτίου
- Το τρανζίστορ προσπέλασης πρέπει να είναι αρκετά μεγάλο για περιορισμένα ρεύματα διαρροής
- Η κλιμακωσιμότητα πέρα απο τα 40-35nm (2013) θα είναι πρόκληση [ITRS 09] (18nm, 2017)
- Emerging Memory Technologies

3D-Stacked DRAM	higher bandwidth	smaller capacity
Reduced-Latency DRAM (e.g., RL/TL-DRAM, FLY-RAM)	lower latency	higher cost
Low-Power DRAM (e.g., LPDDR3, LPDDR4, Voltron)	lower power	higher latency higher cost
Non-Volatile Memory (NVM) (e.g., PCM, STTRAM, ReRAM, 3D Xpoint)	larger capacity	higher latency higher dynamic power lower endurance

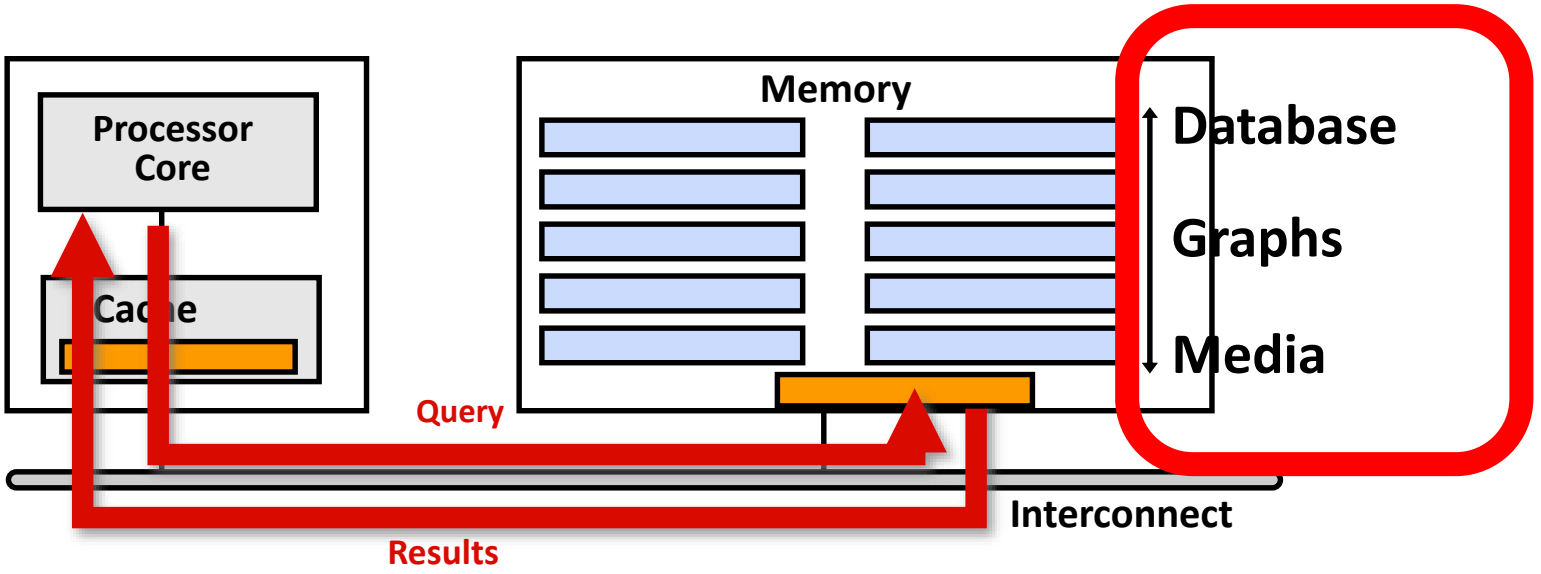
Υβριδικά Συστήματα

- Σχεδίαση συστημάτων → Χρήση πολλαπλών τεχνολογιών και HW/SW ενορχηστρωμένη διαχείριση τους



Near Data Processing

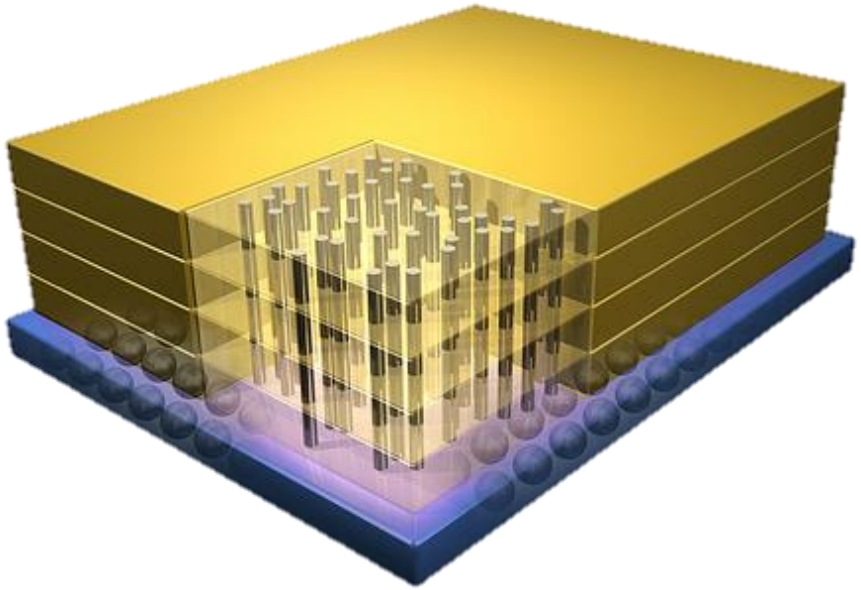
- Η μεταφορά δεδομένων καθοριστικό σημείο συμφόρησης επίδοσης και κατανάλωσης ενέργειας
- Ιδέα: Μείωση της μεταφοράς δεδομένων κάνοντας υπολογισμούς απευθείας εκεί οπου βρίσκονται τα δεδομένα



3D-Stacked memory → Near Memory Computing

- Ευκαιρία: Logic + Memory

Hybrid Memory Cube
CONSORTIUM



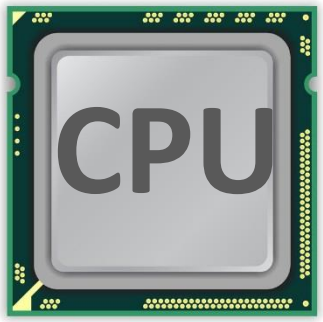
Αξιοπιστία/Ασφάλεια

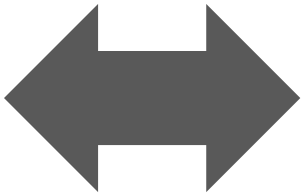
Αξιοπιστία/Ασφάλεια

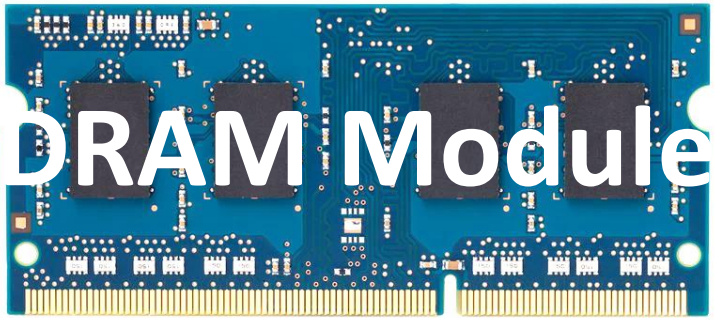
- Όσο η τεχνολογία κατεβαίνει σε nm μειώνεται η αξιοπιστία της
- Μια ιδιαίτερη ανακάλυψη (2014):
  - Πολλές επαναλαμβανόμενες προσπελάσεις μιας γραμμής της κύριας μνήμης (DRAM row), πριν η μνήμη αναζωογονηθεί (refresh), προκαλούν σφάλματα διαταραχής σε ακολουθητικές γραμμές
  - εμφανίζεται σε πολλά σύγχρονα πραγματικά DRAM chips **(RowHammer problem)**

Download from: https://github.com/CMU-SAFARI/rowhammer

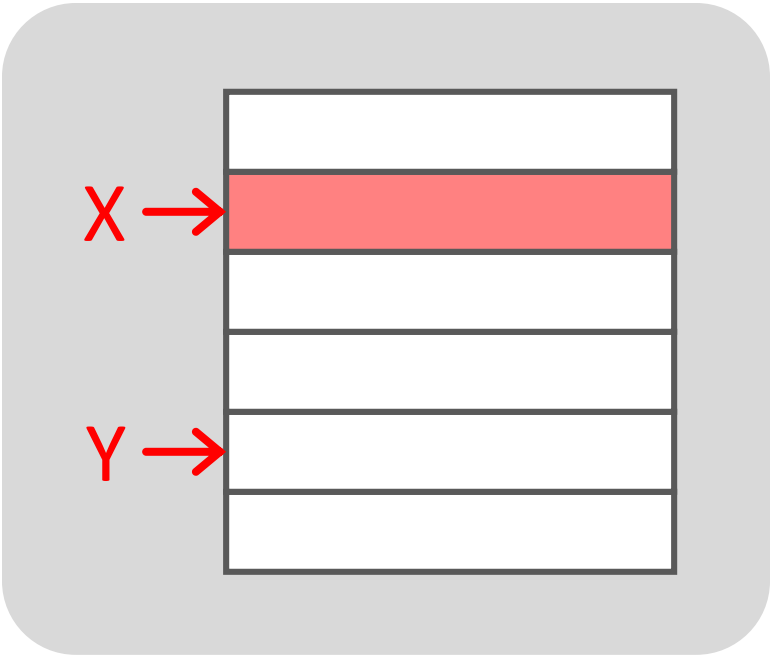
RowHammer





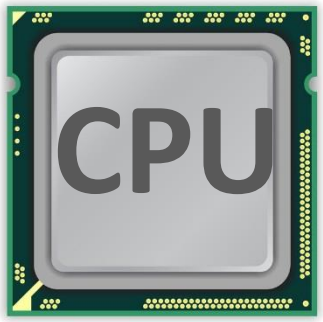


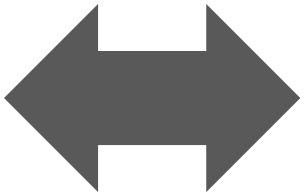

```
loop:
  mov (X), %eax
  mov (Y), %ebx
  clflush (X)
  clflush (Y)
  mfence
  jmp loop
```

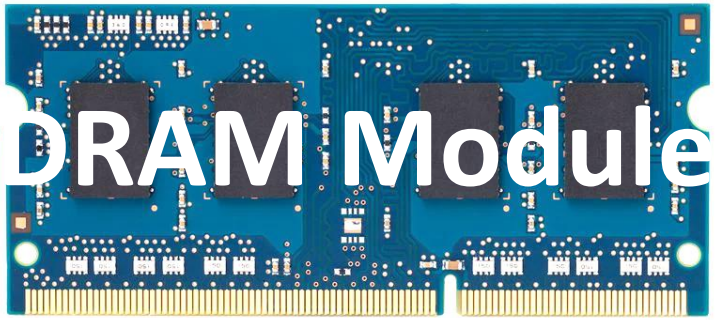


Download from: https://github.com/CMU-SAFARI/rowhammer

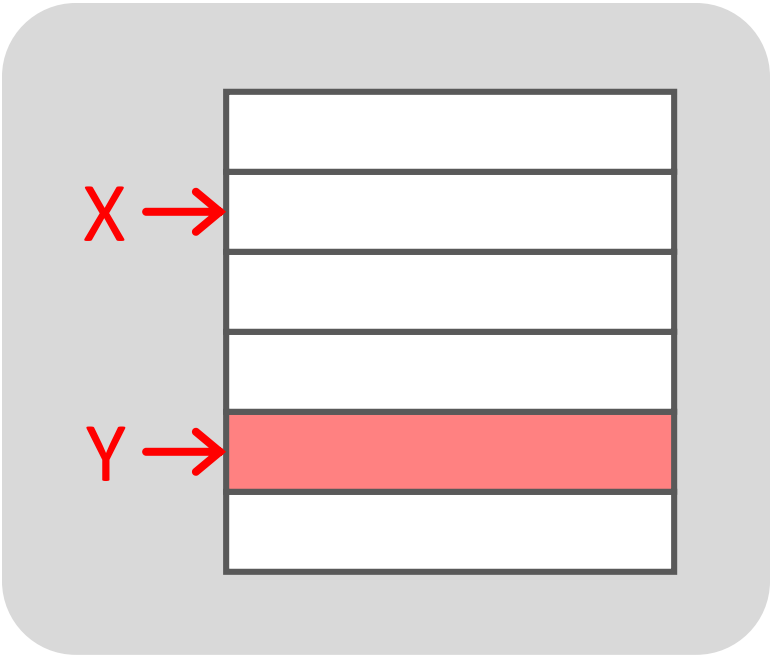
RowHammer





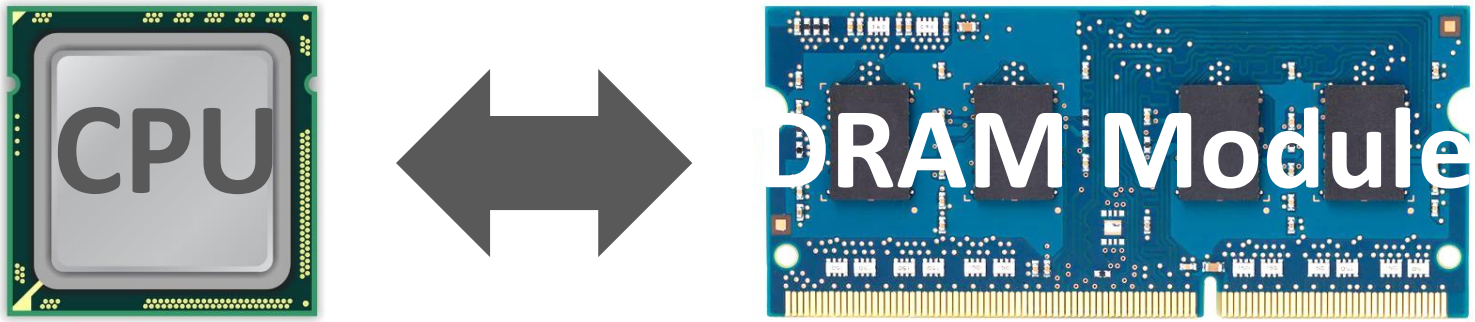



```
loop:
  mov (X), %eax
  mov (Y), %ebx
  clflush (X)
  clflush (Y)
  mfence
  jmp loop
```

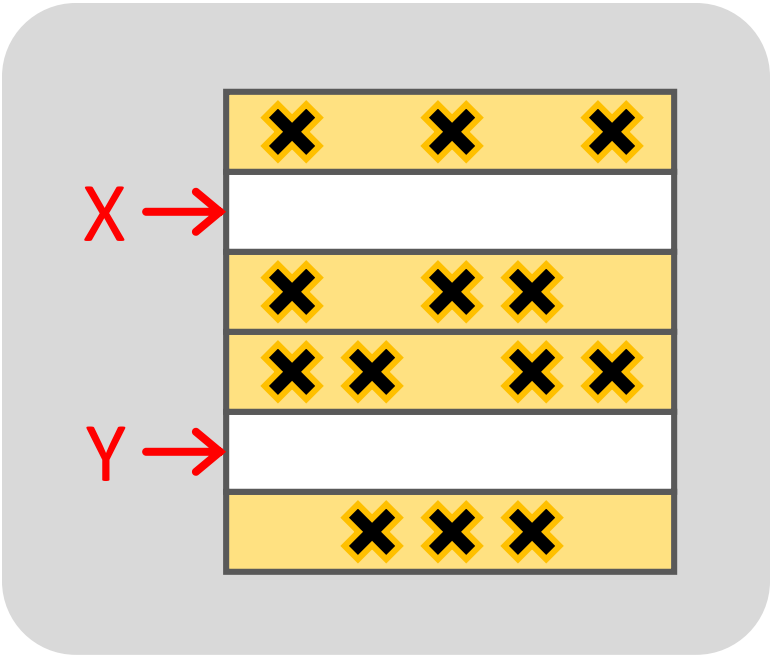


Download from: https://github.com/CMU-SAFARI/rowhammer

RowHammer



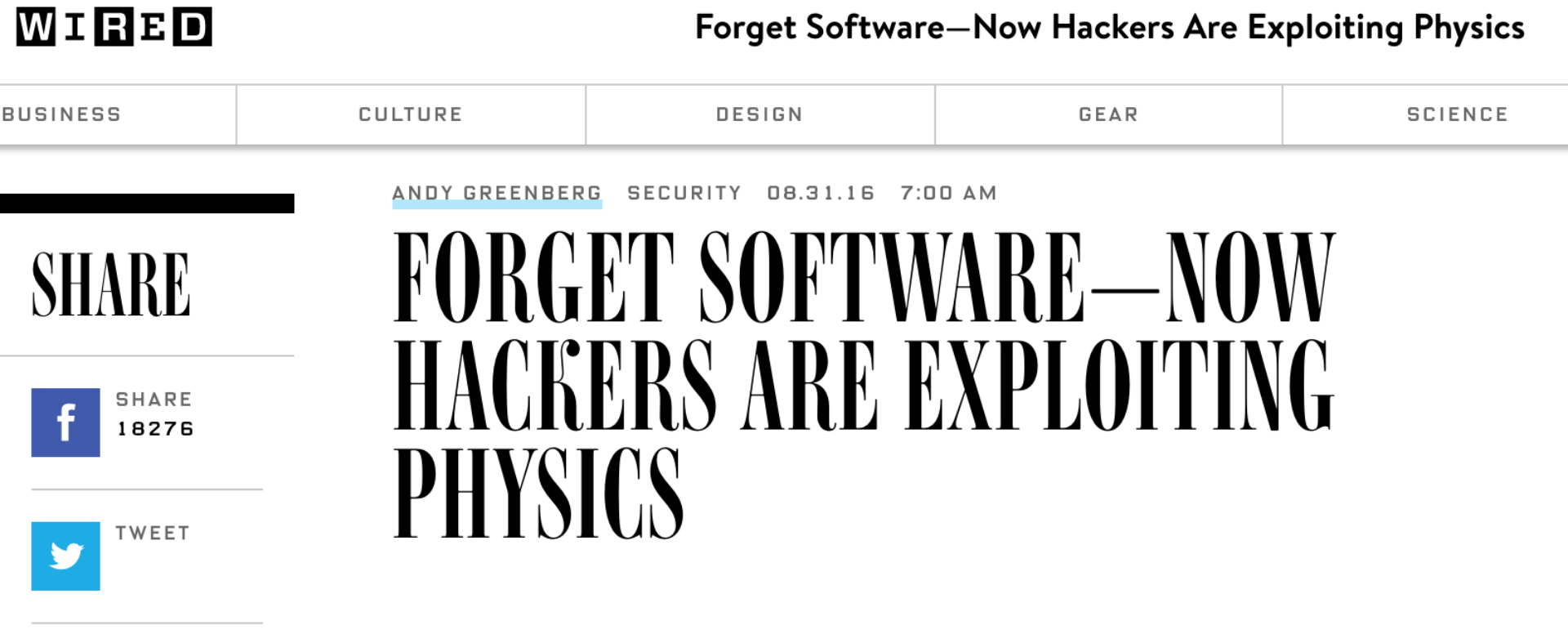

```
loop:
  mov (X), %eax
  mov (Y), %ebx
  clflush (X)
  clflush (Y)
  mfence
  jmp loop
```



Download from: https://github.com/CMU-SAFARI/rowhammer

RowHammer: Security Attack

- Κανείς μπορεί με προβλέψιμο τρόπο να εισάγει σφάλματα σε σύγχρονα DRAM chips
- Ένας απλός μηχανισμός αποτυχίας του HW μπορεί να δημιουργήσει ένα ευρύ κενό ασφαλείας στο σύστημα



RowHammer: Security Attack

- Κανείς μπορεί με προβλέψιμο τρόπο να εισάγει σφάλματα σε σύγχρονα DRAM chips
- Ένας απλός μηχανισμός αποτυχίας του HW μπορεί να δημιουργήσει ένα ευρύ κενό ασφαλείας στο σύστημα

Project Zero

Induce bit flips in page table entries → gain write access to the page table from user space → gain read/write access to entire memory

News and updates from the Project Zero team at Google

https://googleprojectzero.blogspot.gr/2015/03/exploiting-dram-rowhammer-bug-to-gain.html

Monday, March 9, 2015

Exploiting the DRAM rowhammer bug to gain kernel privileges

Apple Patch for RowHammer

- https://support.apple.com/en-gb/HT204934

Available for: OS X Mountain Lion v10.8.5, OS X Mavericks v10.9.5

Impact: A malicious application may induce memory corruption to escalate privileges

Description: A disturbance error, also known as Rowhammer, exists with some DDR3 RAM that could have led to memory corruption. This issue was mitigated by increasing memory refresh rates.

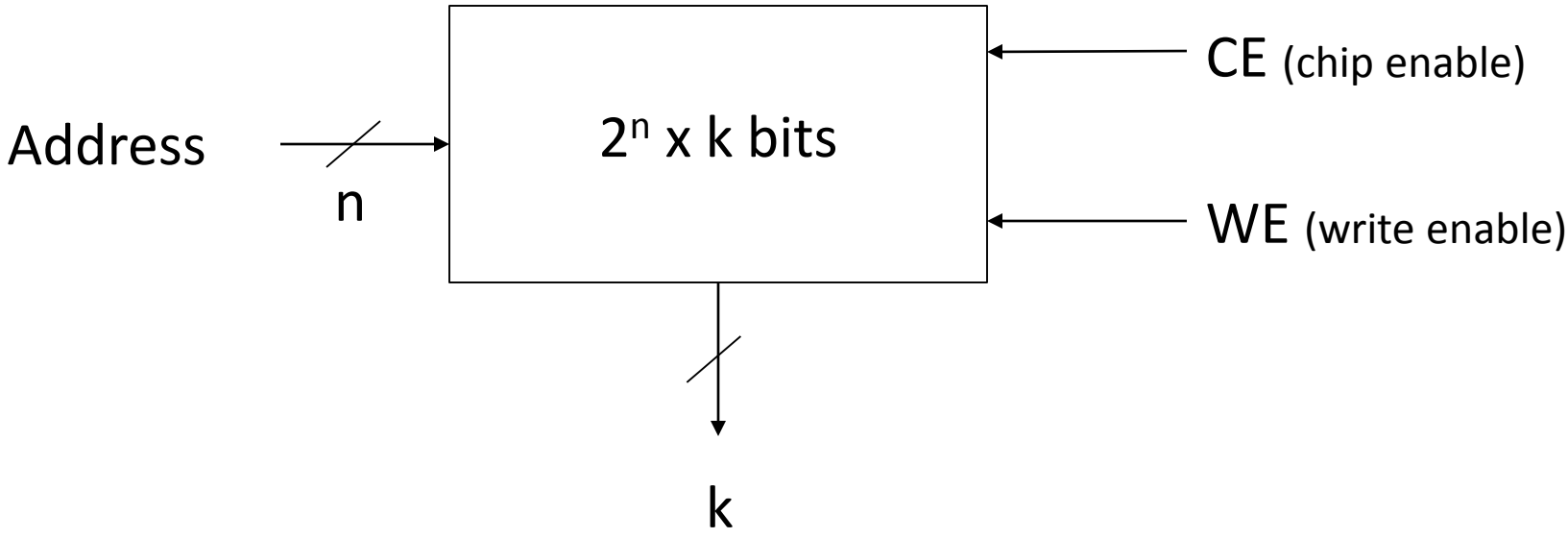
CVE-ID

CVE-2015-3693 : Mark Seaborn and Thomas Dullien of Google, working from original research by Yoongu Kim et al (2014)

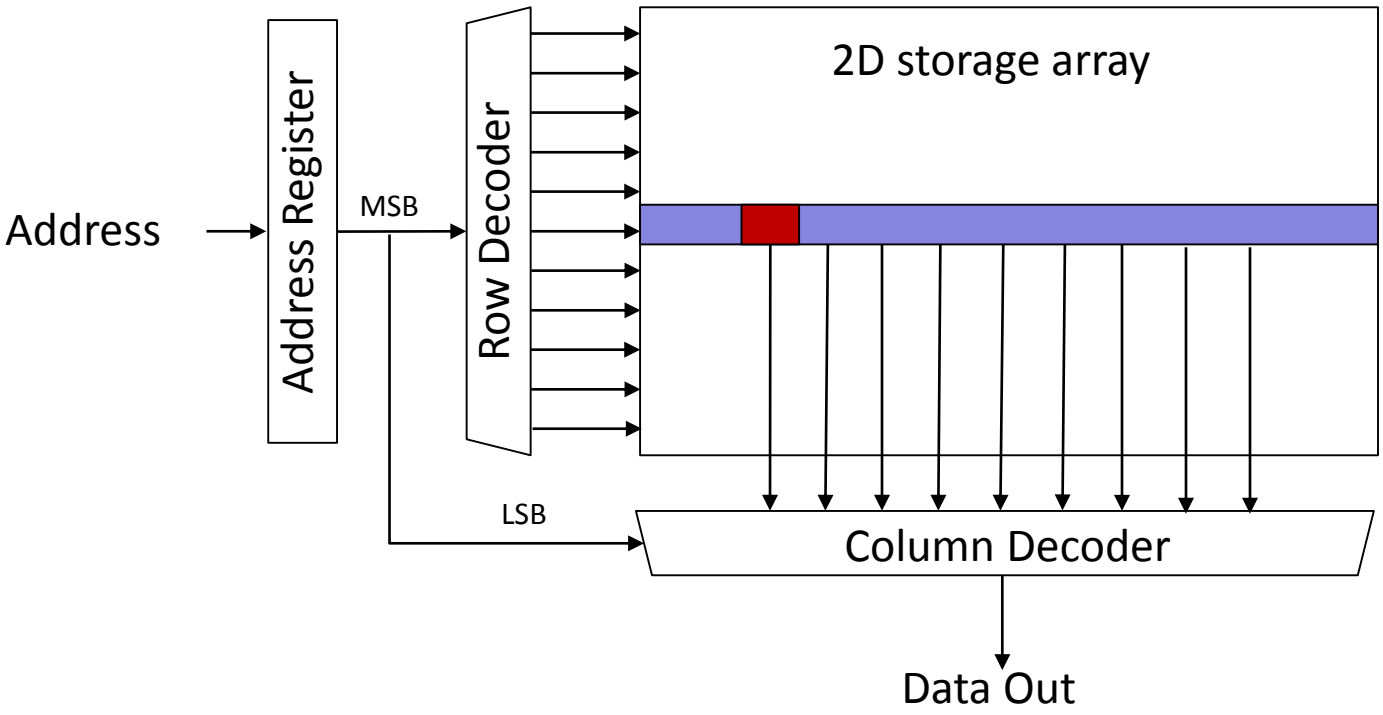
HP, Lenovo, and other vendors released similar patches

Οργάνωση DRAM

Memory chip abstraction



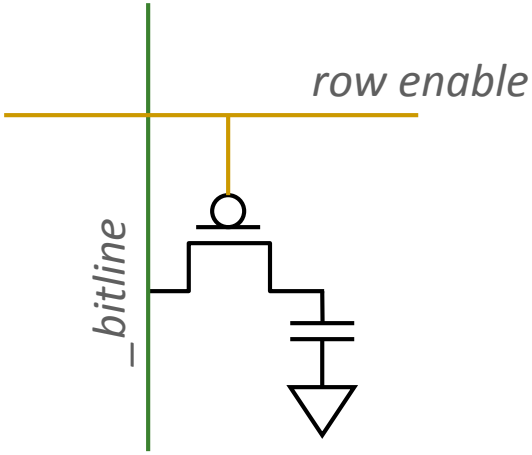
Memory Bank


Address
Address Register
MSB
Row Decoder
2D storage array
LSB
Column Decoder
Data Out

Οργάνωση DRAM

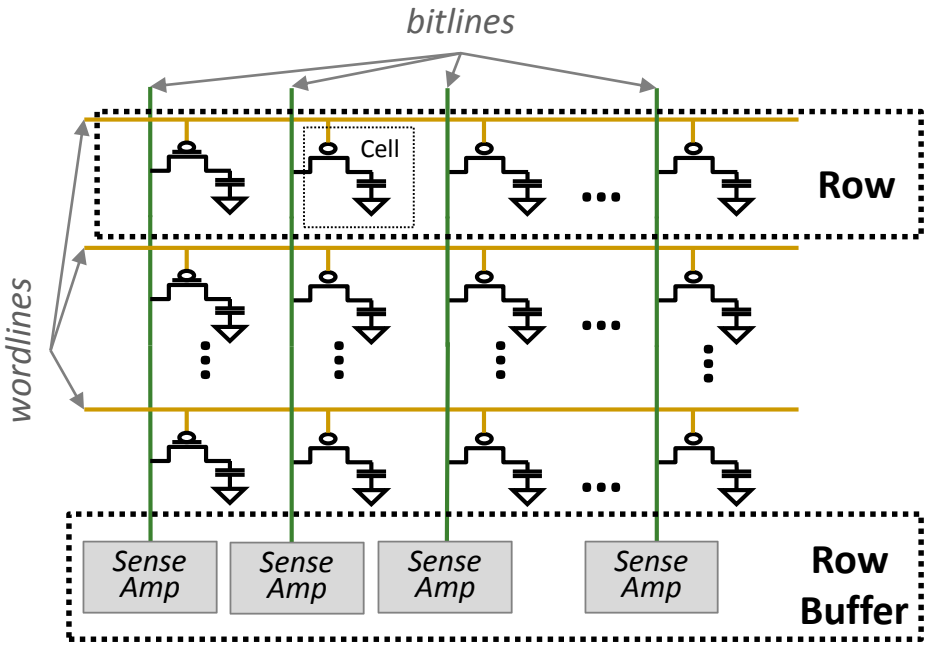
Οργάνωση DRAM
Bottom Up

Δομικό στοιχείο DRAM



- Κυψελίδα (cell)
  - 1 Πυκνωτής
    Λογική τιμή αποθηκεύεται με τη μορφή φορτίου πυκνωτή
  - 1 Τρανζίστορ προσπέλασης
    On/Off → συνδέει/αποσυνδέει τον πυκνωτή σε καλώδιο που ονομάζεται bitline

DRAM πίνακας (array)



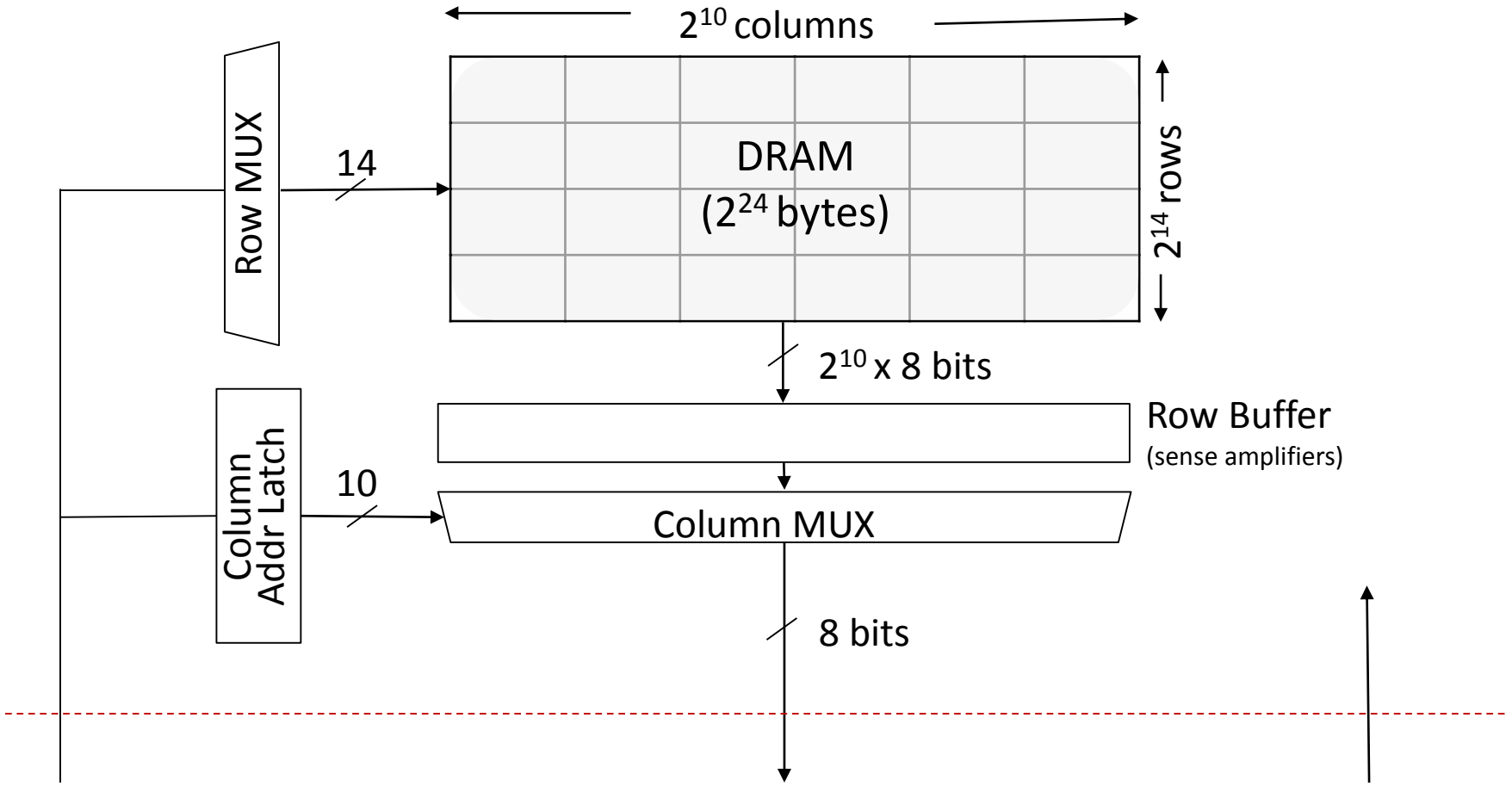
- 2D πίνακας κυψελίδων (cell)
- DRAM row ή DRAM page
  Κοινό καλώδιο (wordline) ελέγχει τα τρανζίστορ προσπέλασης.
- Row Buffer
  Κάθε bitline συνδέεται σε αισθητήρα-ενισχυτή για την αξιόπιστη ανίχνευση του μικρού φορτίου πυκνωτή ενός cell.

Προσπέλαση θέσης μνήμης:

- Ενεργοποίηση ολόκληρης γραμμής → cells συνδέονται σε bitlines
- "Άνοιγμα"/τοποθέτηση γραμμής σε Row Buffer
- Read/Write εντολές → διαβάζουν/γράφουν στο row buffer

Μόνο μια γραμμή μπορεί να είναι "ανοιχτή" τη φορά!

DRAM πίνακας (array) -- αφαίρεση

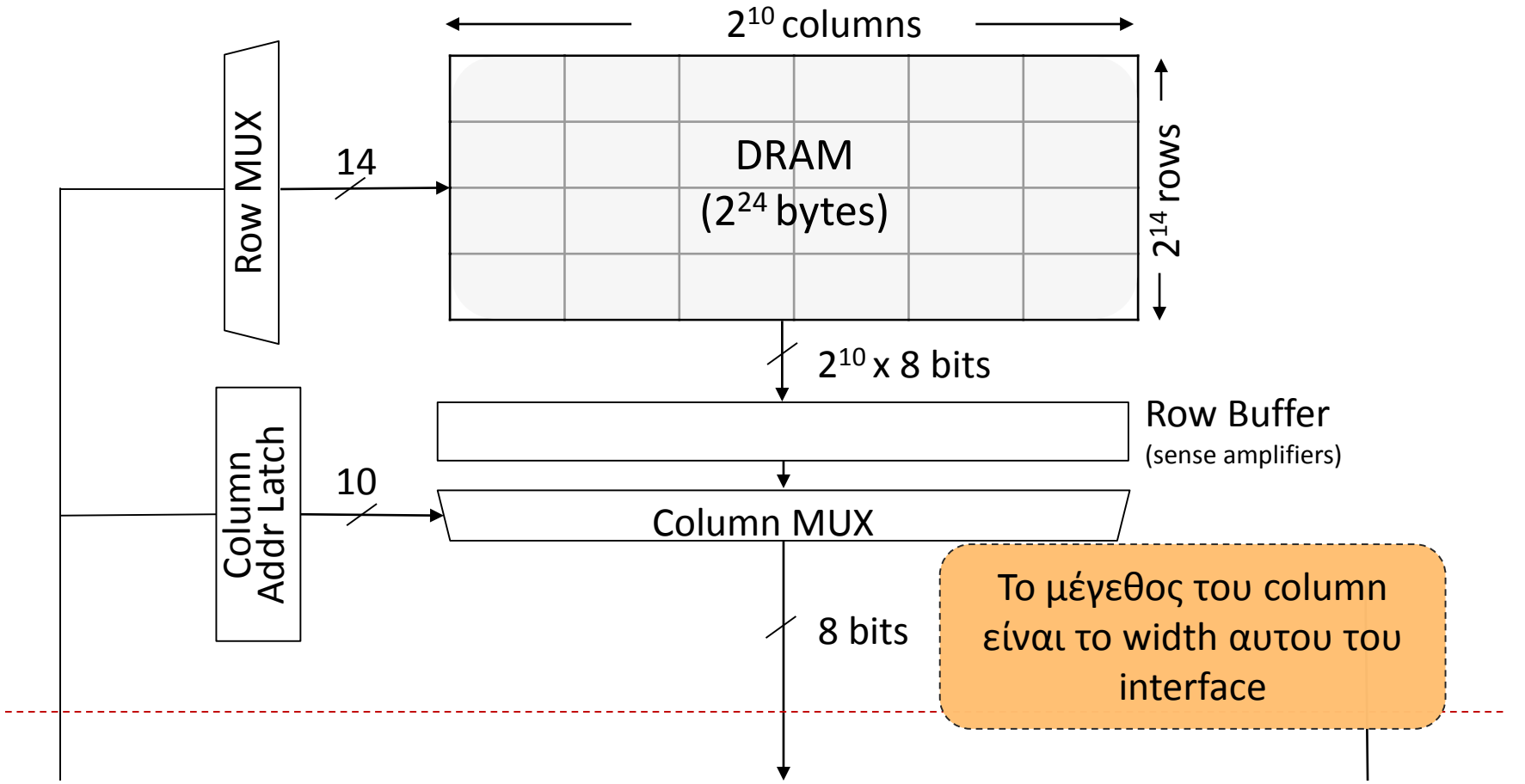


DRAM διεύθυνση από το memory controller

Δεδομένα προς το memory controller

Εντολή από τον memory controller

DRAM πίνακας (array) -- αφαίρεση

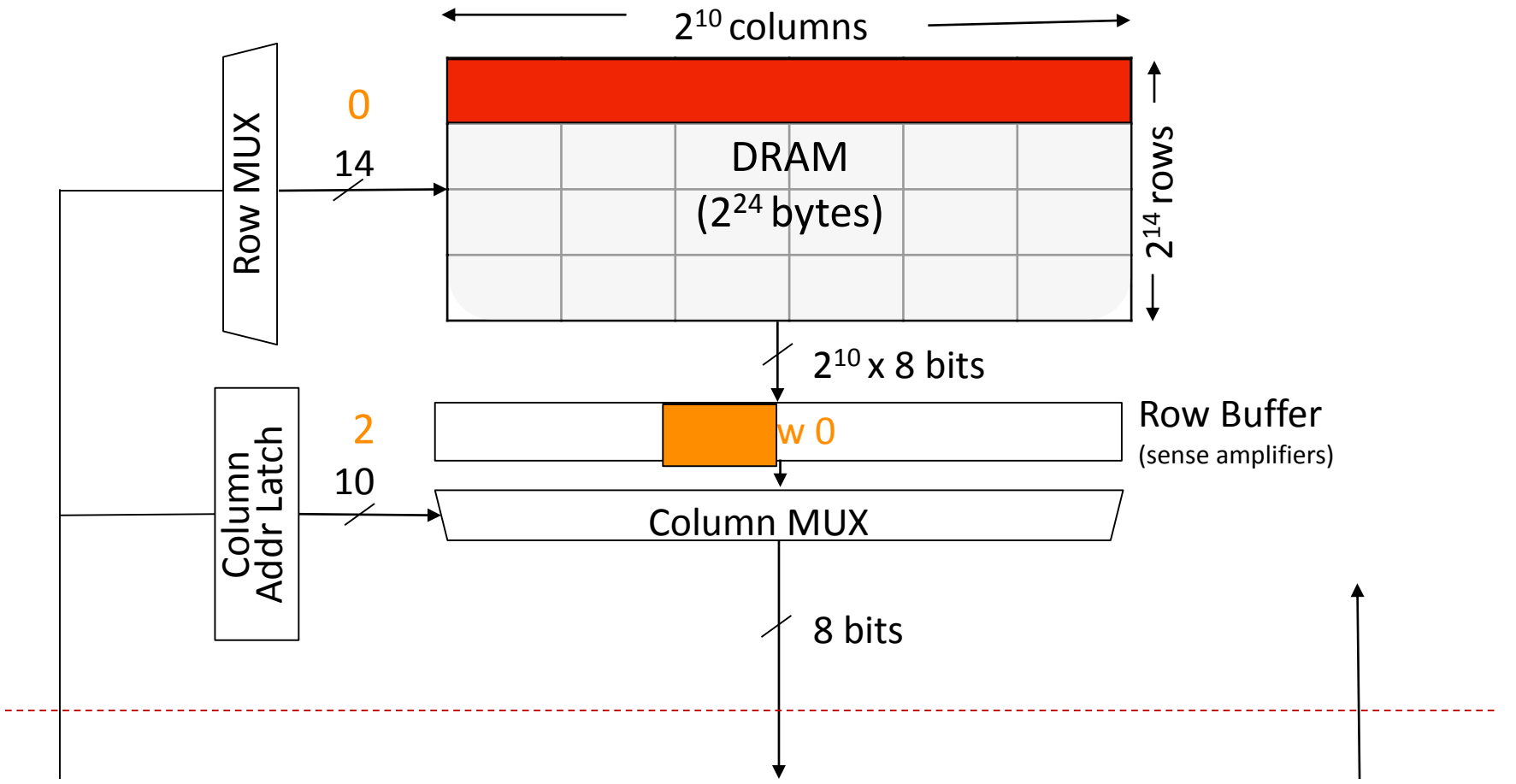


Κάθε διεύθυνση είναι ένα ζεύγος <row, column>

Δεδομένα προς το memory controller

Εντολή από τον memory controller

Προσπέλαση DRAM array



Διαστρωμάτωση (interleaving/banking)

Πρόβλημα

- Ένας μεγάλος μονολιθικός πίνακας μνήμης
  - Έχει μεγάλο χρόνο απόκρισης (latency)
  - Δεν επιτρέπει πολλαπλές παράλληλες προσπελάσεις (memory level parallelism – MLP)

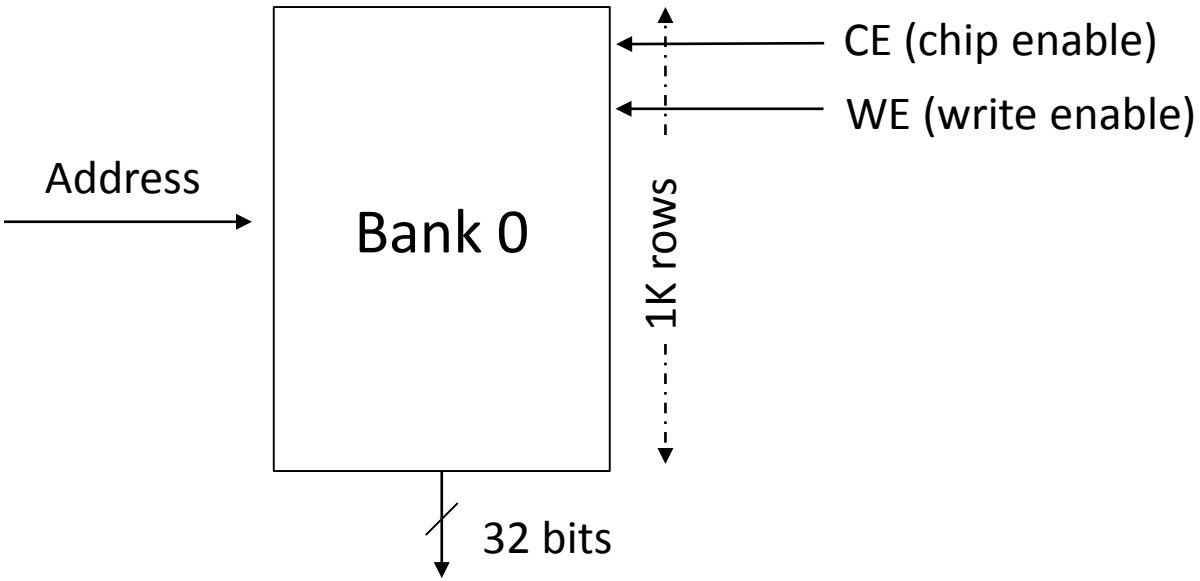
Ιδέα

- Διαίρεση πίνακα σε N επιμέρους banks που μπορούν να προσπελαστούν στον ίδιο ή σε ακολουθητικό κύκλο (cycle)
  - Κάθε bank έχει μέγεθος *συνολική μνήμη/N*
  - Προσπέλασεις σε διαφορετικά banks επικαλύπτονται
  - Bits της διεύθυνσης καθορίζουν σε ποιο bank θα αντιστοιχηθεί

DRAM Bank

- DRAM bank : ένα memory array
- Η μικρότερη δομή μνήμης η οποία μπορεί να προσπελαστεί παράλληλα (bank level parallelism)

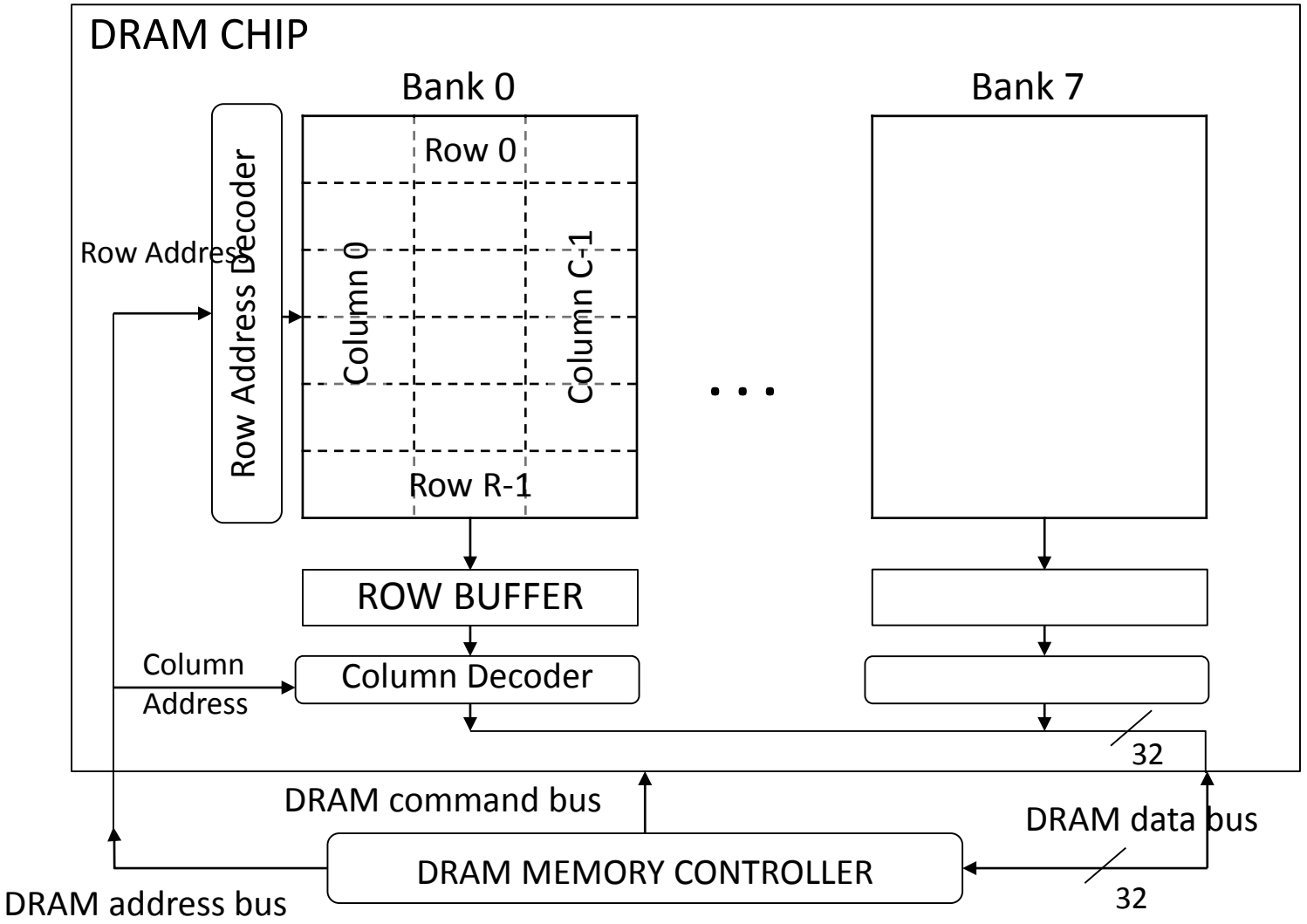
Αφαιρετική λογική αναπαράσταση bank



DRAM chip

- DRAM chip: διάταξη μνήμης που αποτελείται από πολλαπλά banks (8 συνήθως)
- Τα banks διαμοιράζονται κοινούς διαύλους εντολών/διευθύνσεων/δεδομένων (common command/address/data buses)
  - pipelined accesses

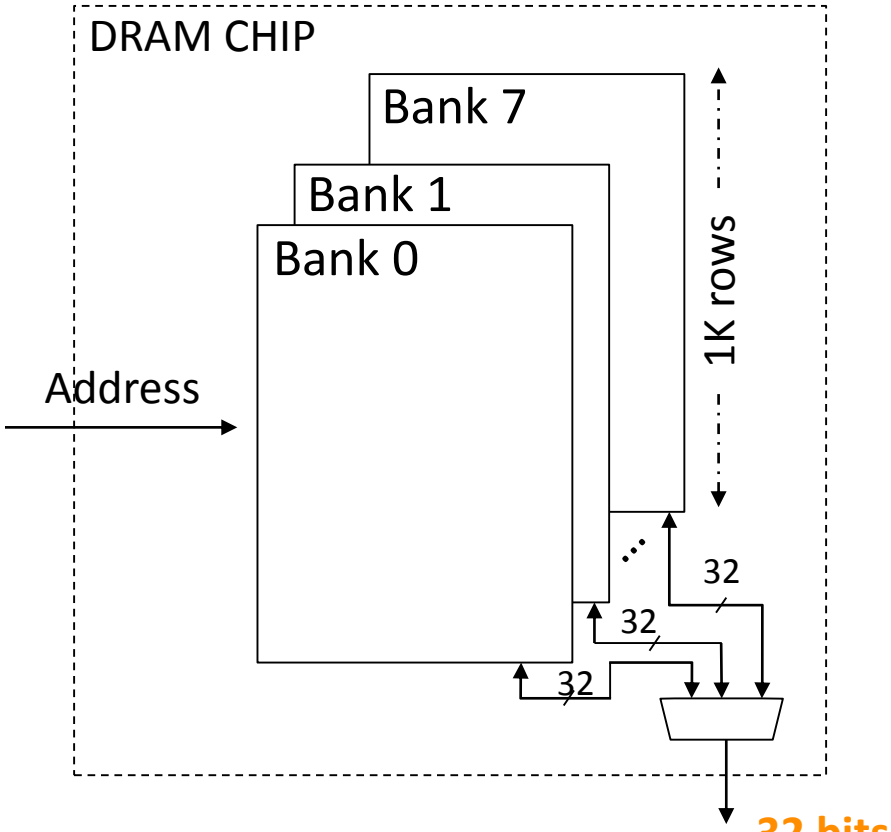
DRAM chip


DRAM CHIP
Bank 0
Bank 7
Row 0
Row Address
Row Address Decoder
Column 0
Column C-1
Row R-1
ROW BUFFER
Column
Address
Column Decoder
32
DRAM command bus
DRAM data bus
DRAM MEMORY CONTROLLER
DRAM address bus
32

DRAM chip

Πρόβλημα:

- Η κατασκευή ενός DRAM chip με “ευρύ” interface (π.χ 32-bit/pin chip) είναι ακριβή ($).



Κατάτμηση: DRAM Rank

Ιδέα:

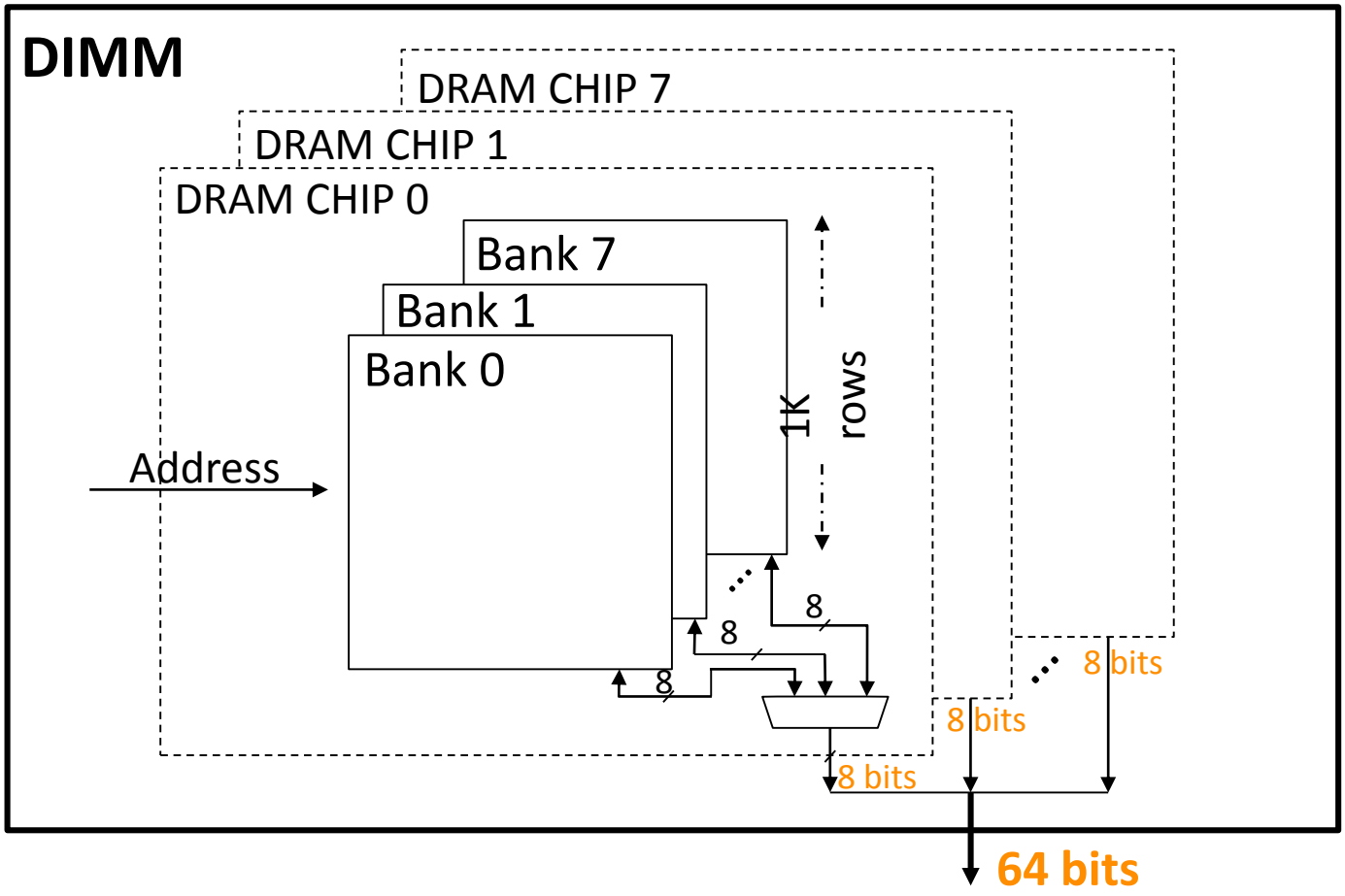
- Συλλογική χρήση σύνολου chip, το καθένα με μικρό interface, για τη δημιουργία μιας ευρείας διεπαφής (wide interface).

- Rank: Σύνολο από chip που
  - αποκρίνονται στην ίδια εντολή και στην ίδια διεύθυνση ταυτόχρονα
  - αποθηκεύοντας διαφορετικό μέρος των ζητούμενων δεδομένων

- Μια δομική μονάδα DRAM (DRAM module) αποτελείται από 1 ή περισσοτερα ranks
  - π.χ DIMM (dual inline memory module)
  - αυτό συνδέουμε στη μητρική μας κάρτα!

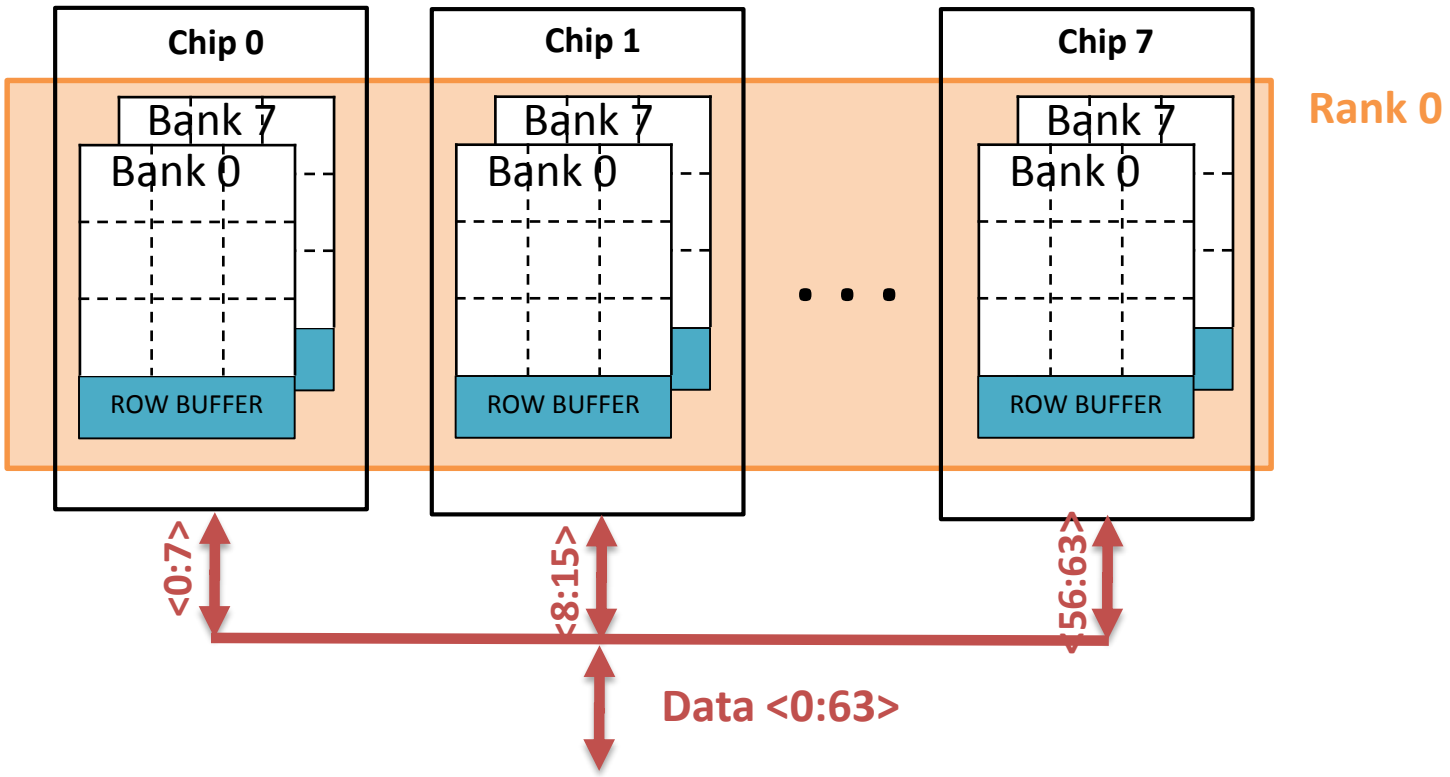
64-bit Wide DIMM (one rank)

Παράδειγμα: δημιουργία 64-bit interface με το memory controller, με ένα rank από 8 chip με 8-bit interfaces. Η σωρευτική έξοδος δίνει τα 64-bit.



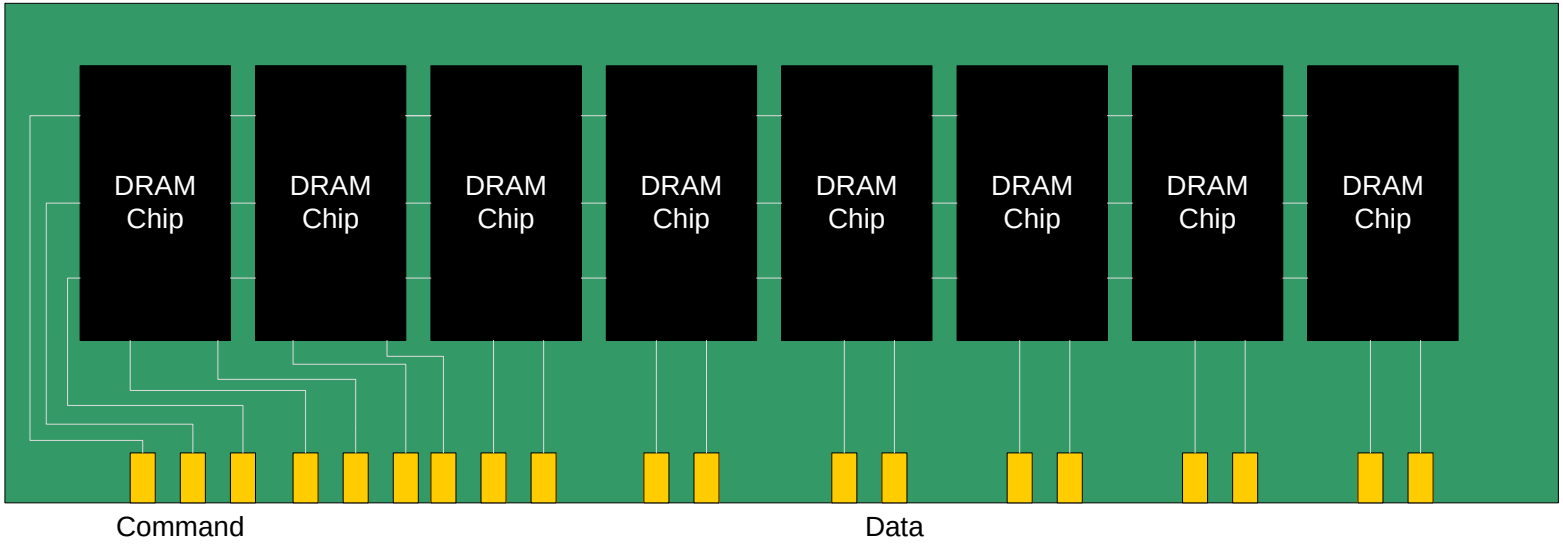
64-bit Wide DIMM (one rank)

- Ένας άλλος τρόπος να το σκεφτούμε/απεικονίσουμε είναι η κατάτμηση (partitioning) του κάθε bank στα διαφορετικά chips του ίδιου rank.



64-bit Wide DIMM (one rank)

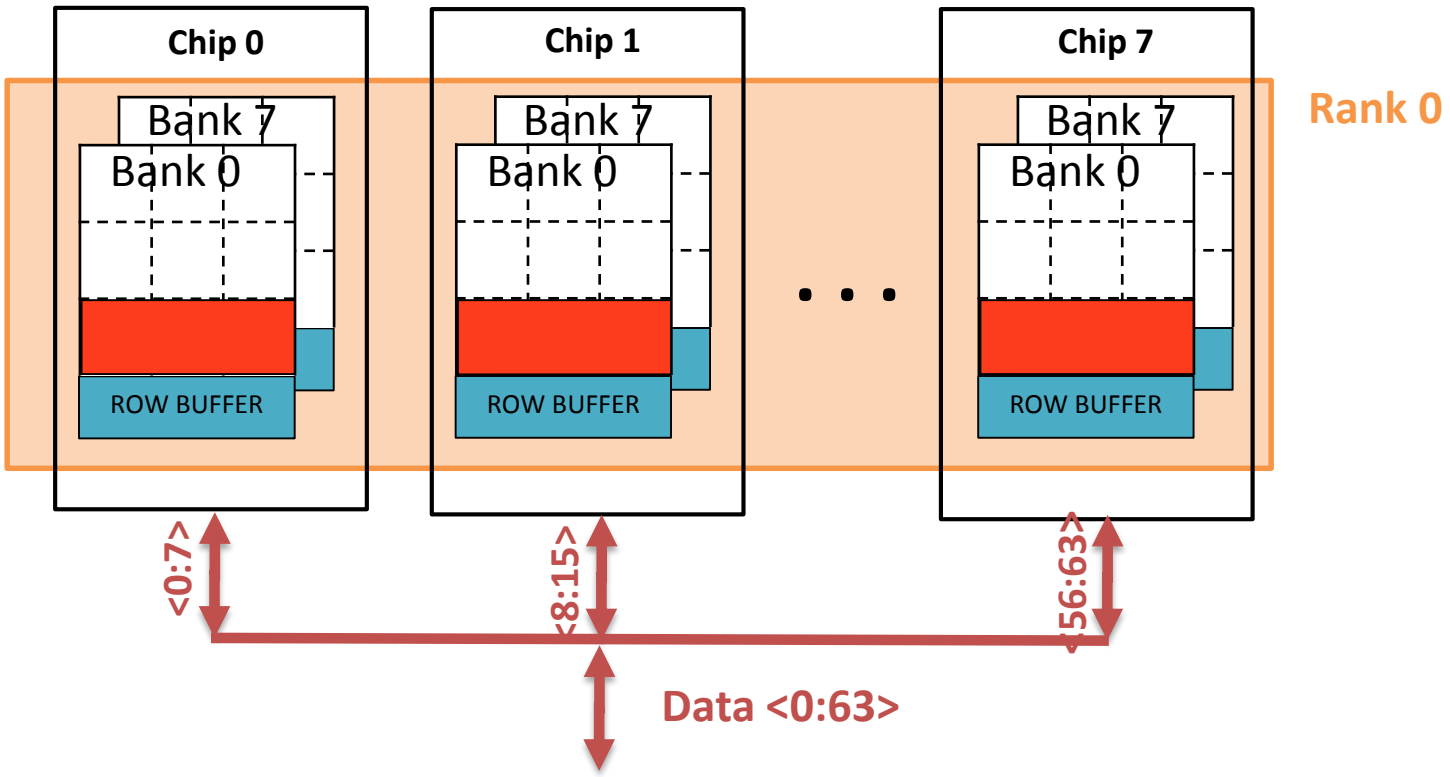
- Τα chip του ίδιου rank έχουν κοινό command και address bus (ανταποκρίνονται στην ίδια εντολή) αλλά έχουν διαφορετικό data bus (παρέχουν διαφορετικό τμήμα των δεδομένων)



64-bit Wide DIMM (one rank)

Το bank εξακολουθεί να είναι η μικρότερη δομή μνήμης που μπορεί να προσπελαστεί παράλληλα, απλώς είναι κατανεμημένο.

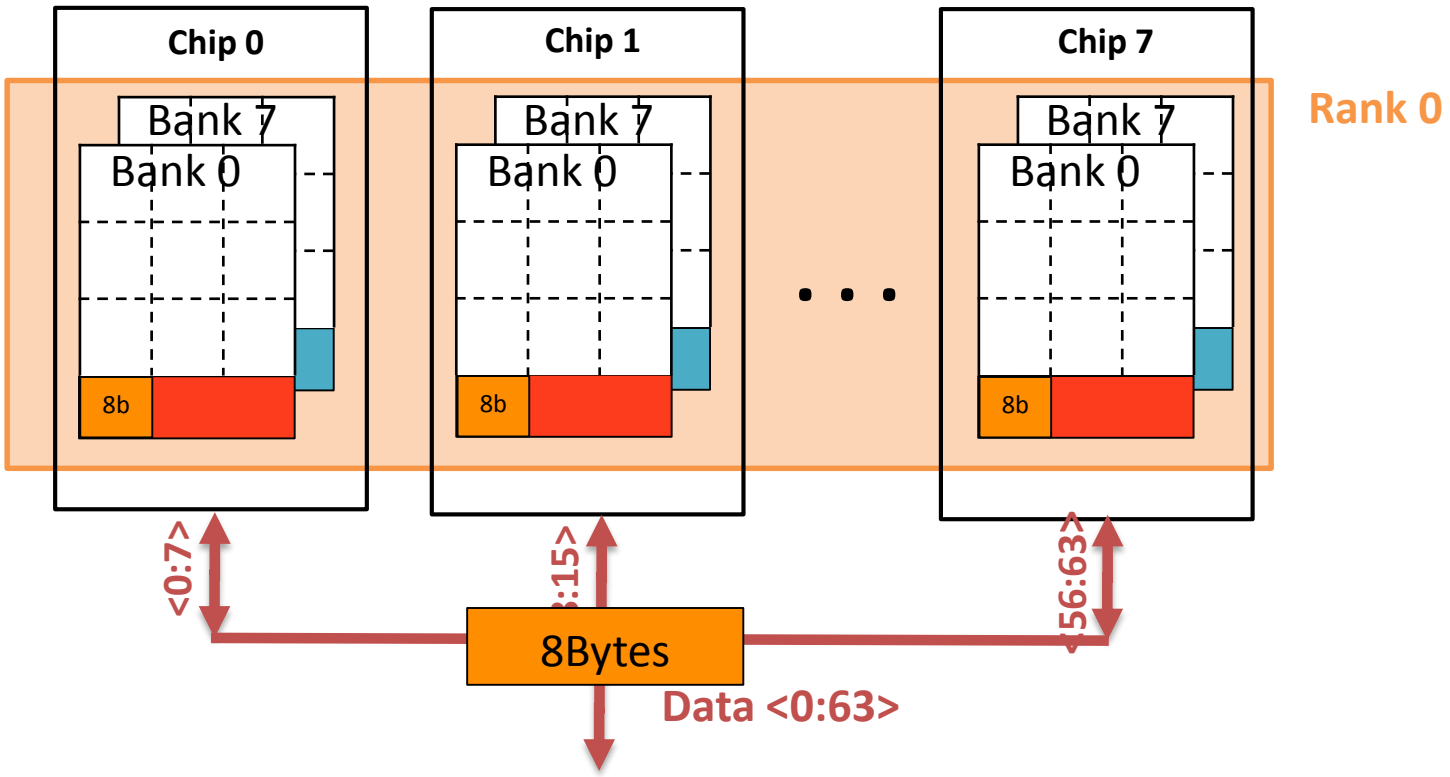
Open
Bank 0, Row 0



64-bit Wide DIMM (one rank)

Το bank εξακολουθεί να είναι η μικρότερη δομή μνήμης που μπορεί να προσπελαστεί παράλληλα, απλώς είναι κατανεμημένο.

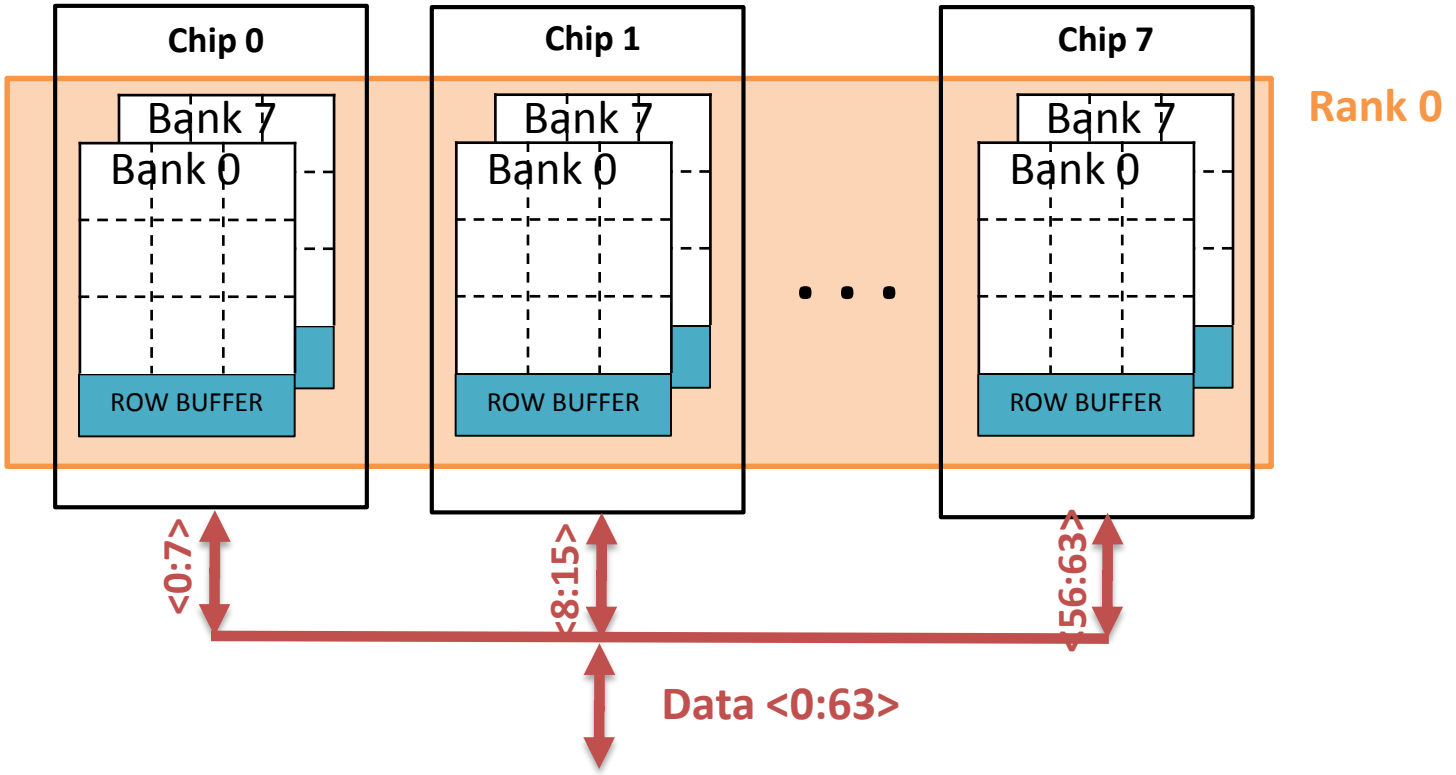
Access data: Row0, Col 0
(8 bytes – dictated by interfaces)



64-bit Wide DIMM (one rank)

Πλεονεκτήματα:

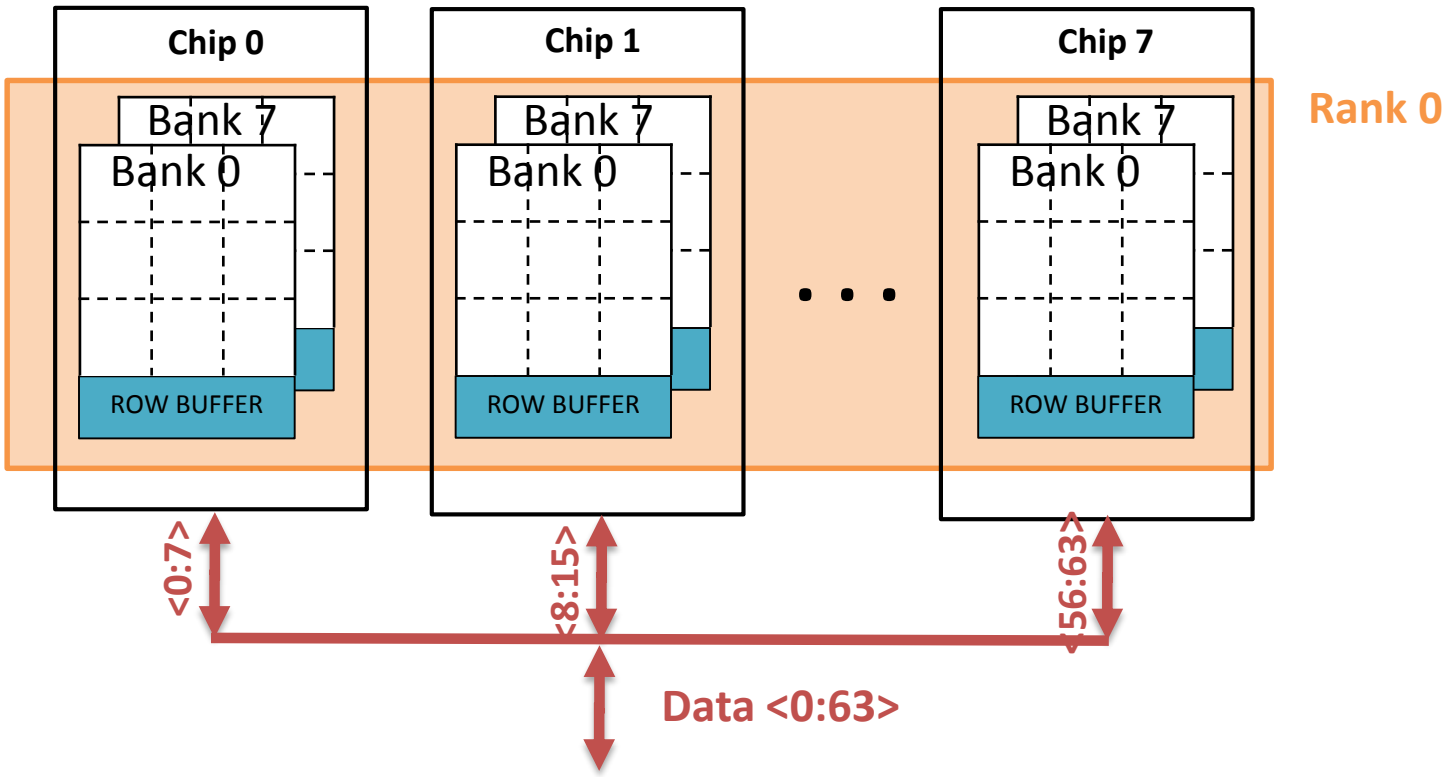
- Λειτουργεί ως ένα chip μεγάλης χωρητικότητας και διευρυμένης διεπαφής (wide interface) έχοντας χαμηλότερο κόστος.
- Ο memory controller δε χρειάζεται να διαχειριστεί τα chip
  Διαχειρίζεται τα banks (η μικρότερη δομή που μπορεί να προσπελαστεί παράλληλα – bank level parallelism) αγνοώντας οτι ειναι κατανεμημένα.



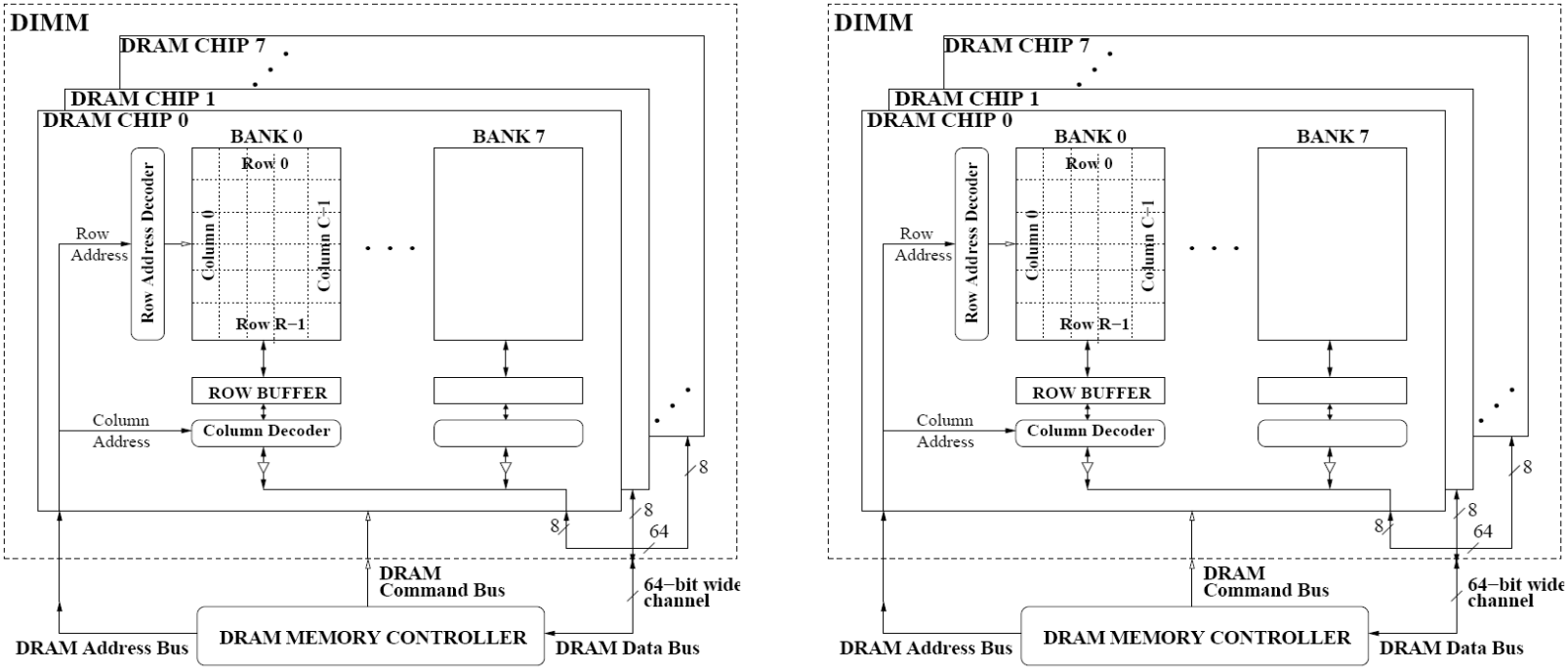
64-bit Wide DIMM (one rank)

Μειονεκτήματα:

- Granularity: δε μπορούν να γίνουν προσπελάσεις μικρότερες από το “εύρος” της διεπαφής (8 bytes)

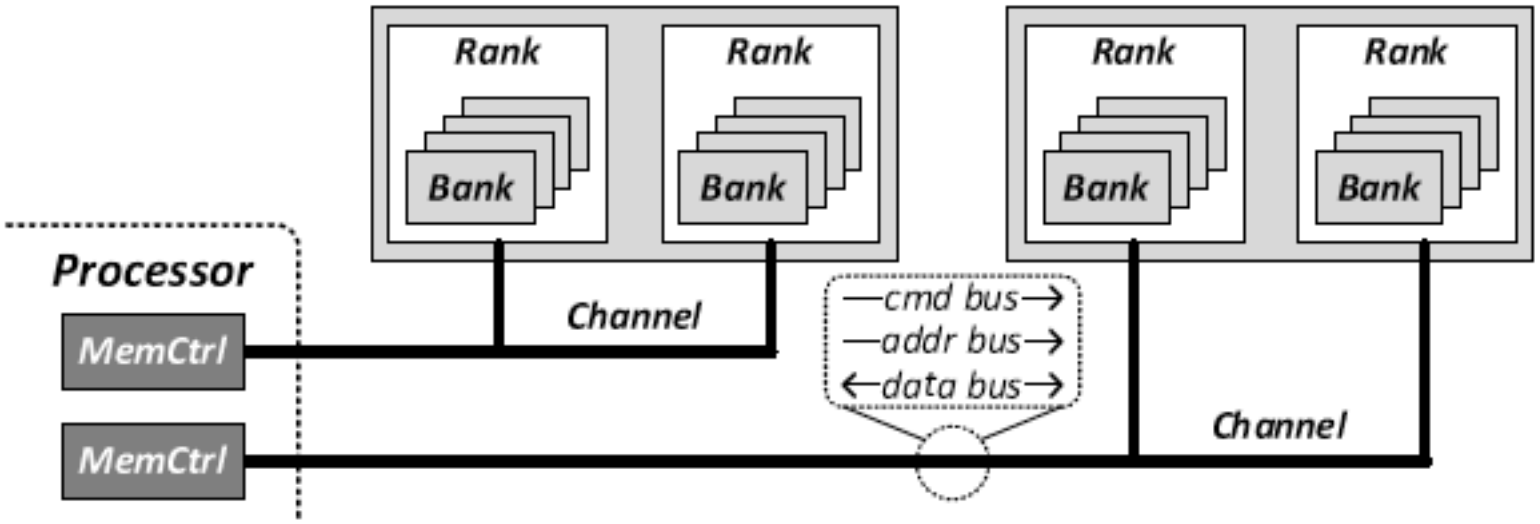


Δίαυλοι DRAM (DRAM channels)



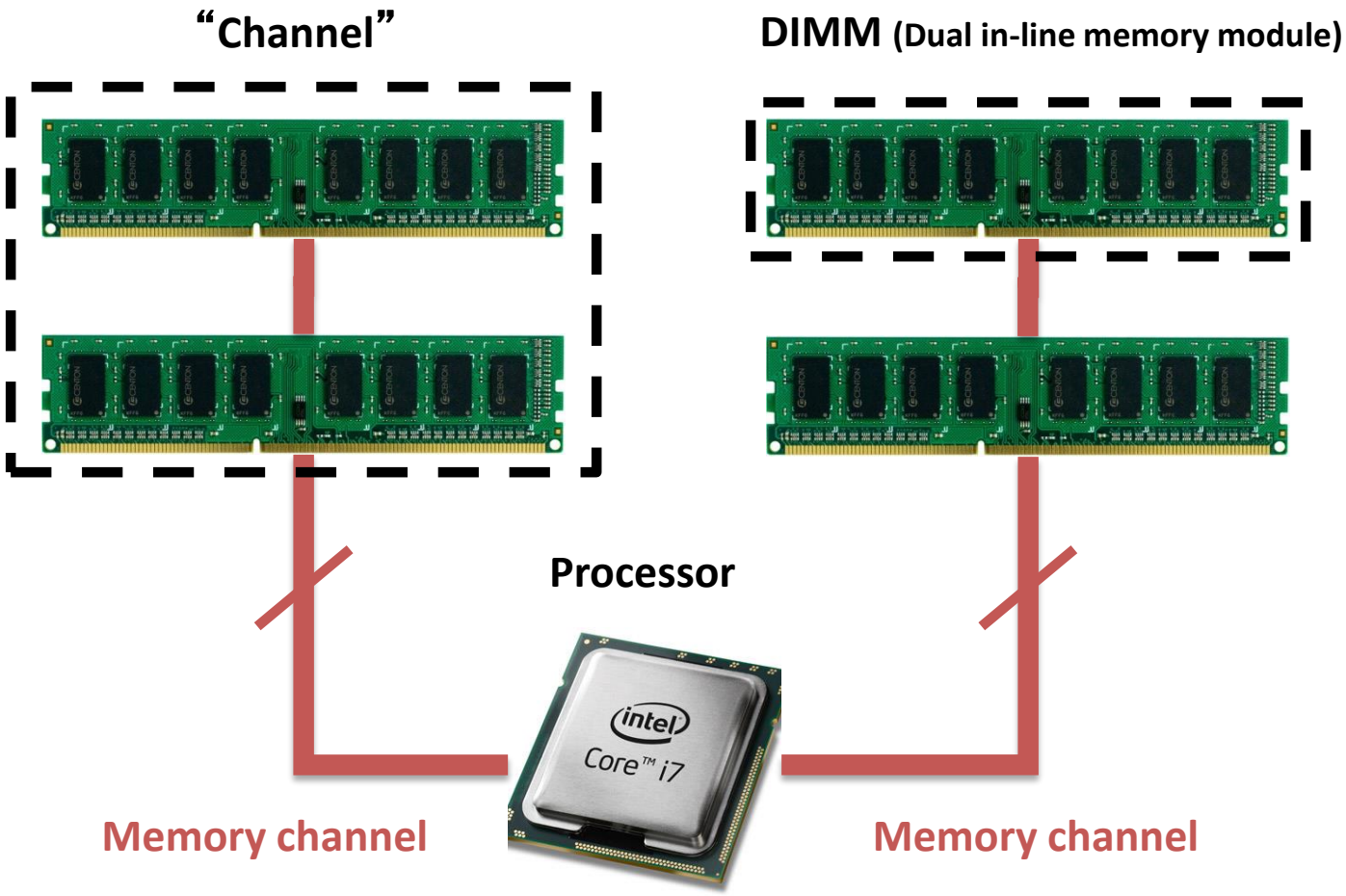
- Channel: Το σύνολο των banks (οργανωμένα σε ranks και DIMMs) τα οποία διαμοιράζονται κοινό link (εντολών, διευθύνσεων, δεδομένων) προς τον επεξεργαστή.
- 2 ανεξάρτητα Channels: 2 memory controllers

Η γενική εικόνα



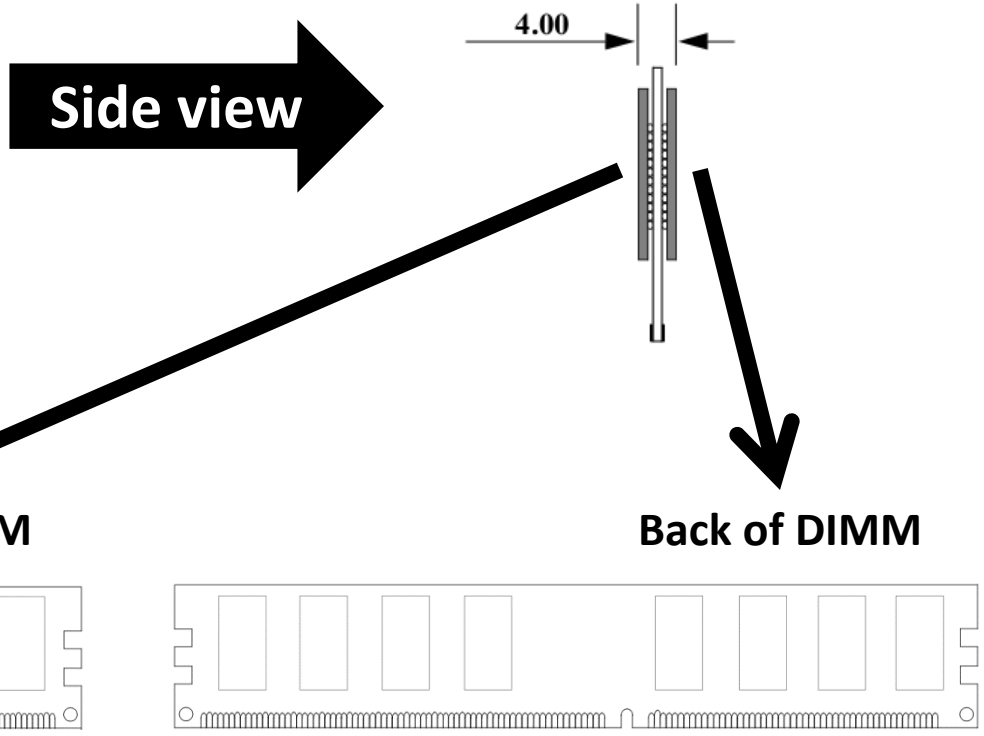
Οργάνωση DRAM
Top Down

Memory Channels



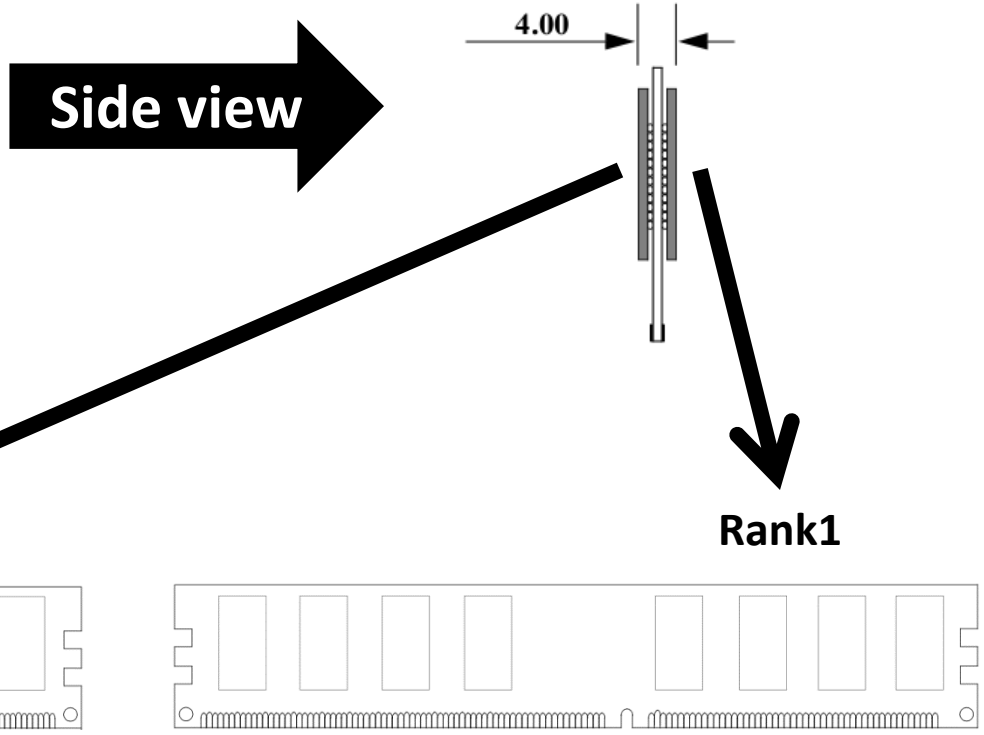
DIMM

DIMM (Dual in-line memory module)

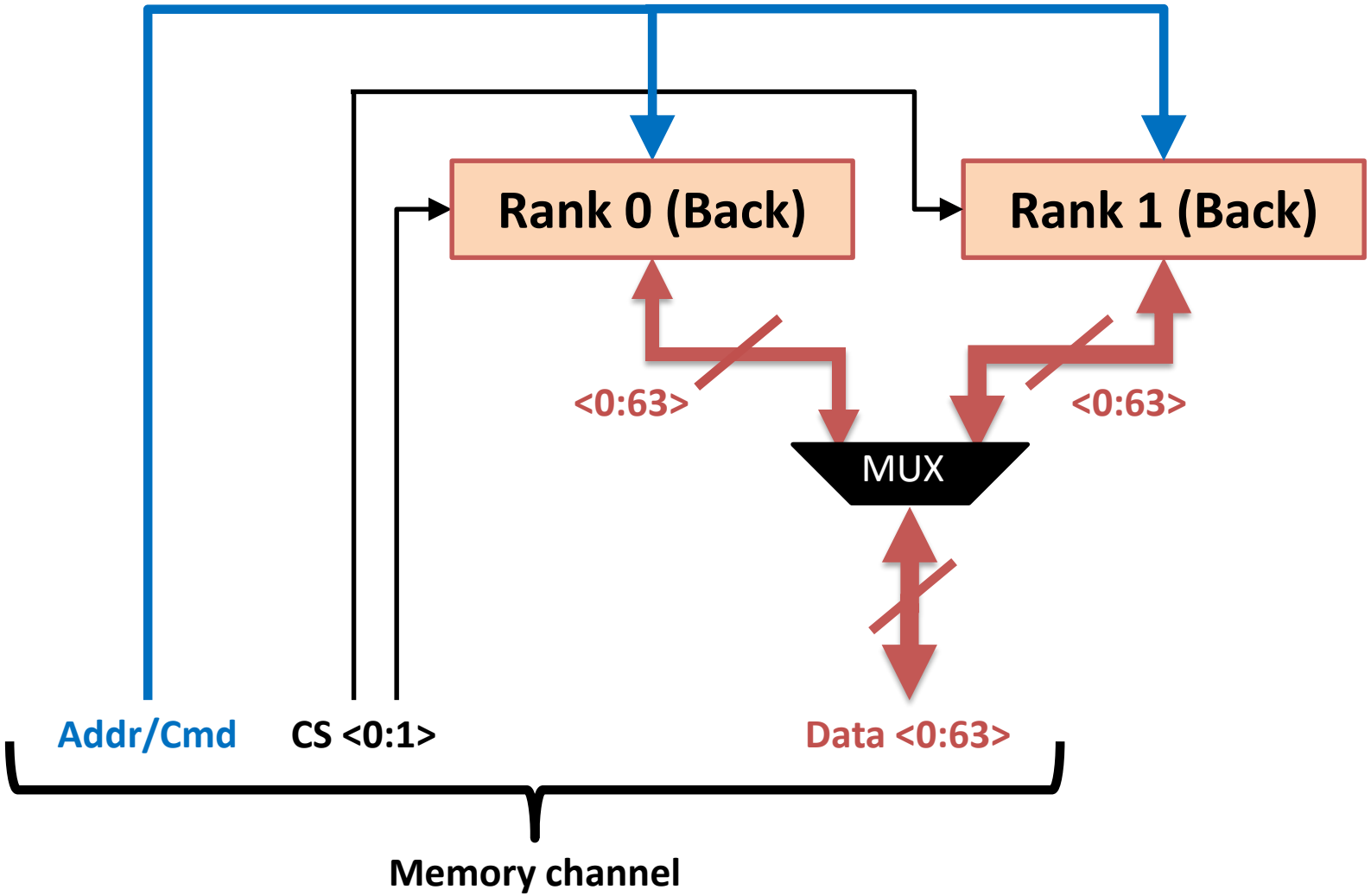


DIMM

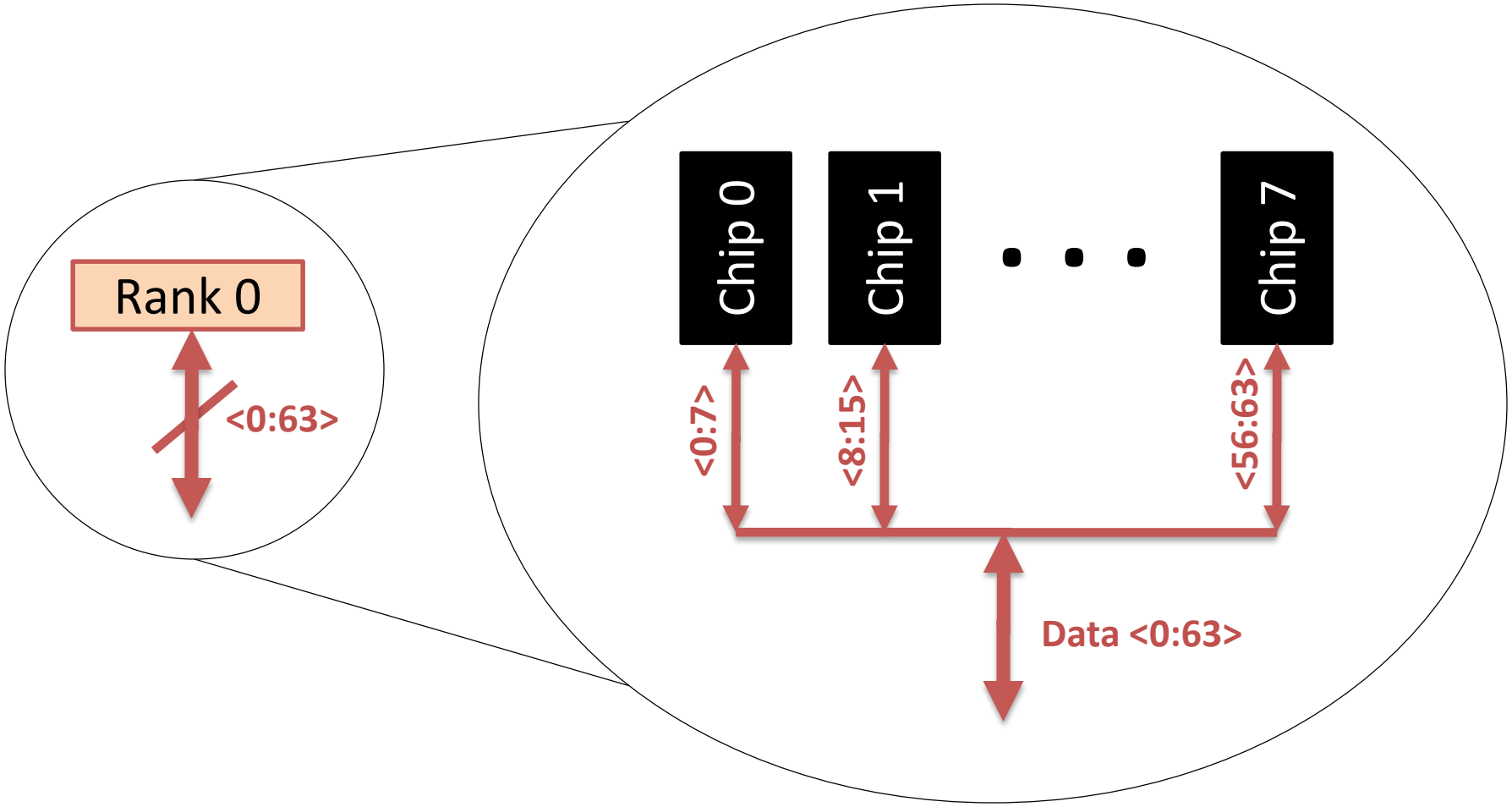
DIMM (Dual in-line memory module)



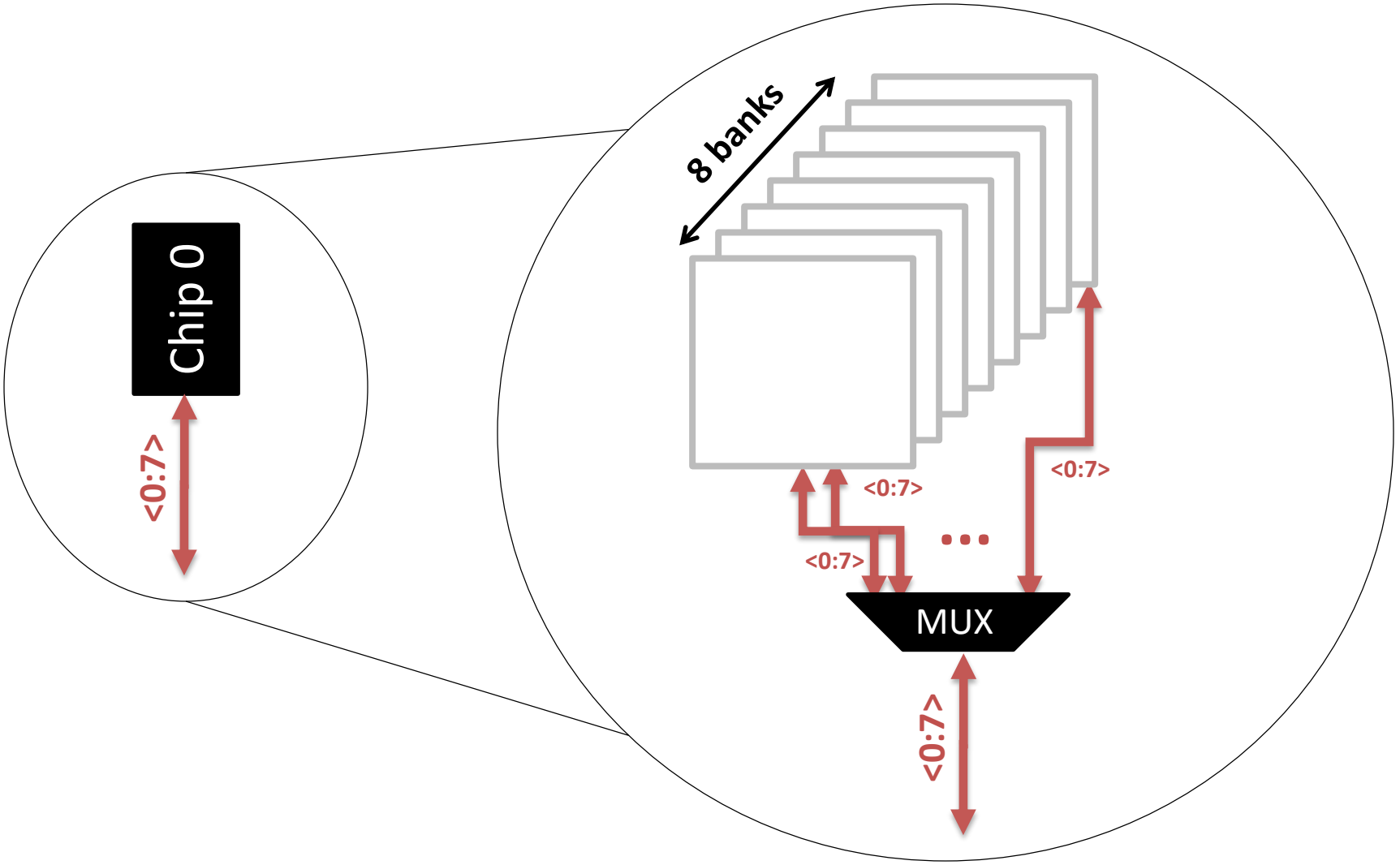
Breaking Down a DIMM → Ranks



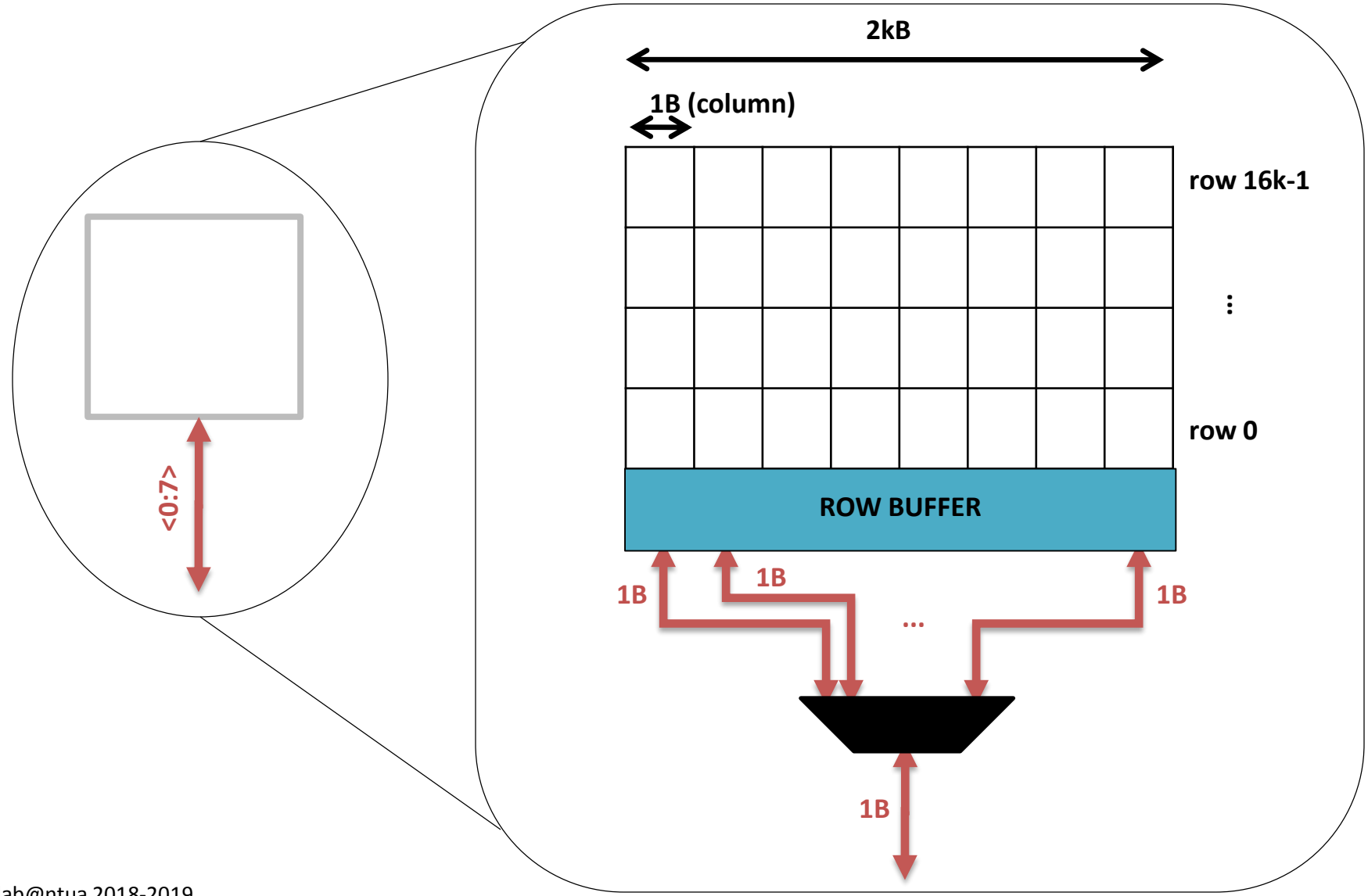
Breaking down a Rank → Chip



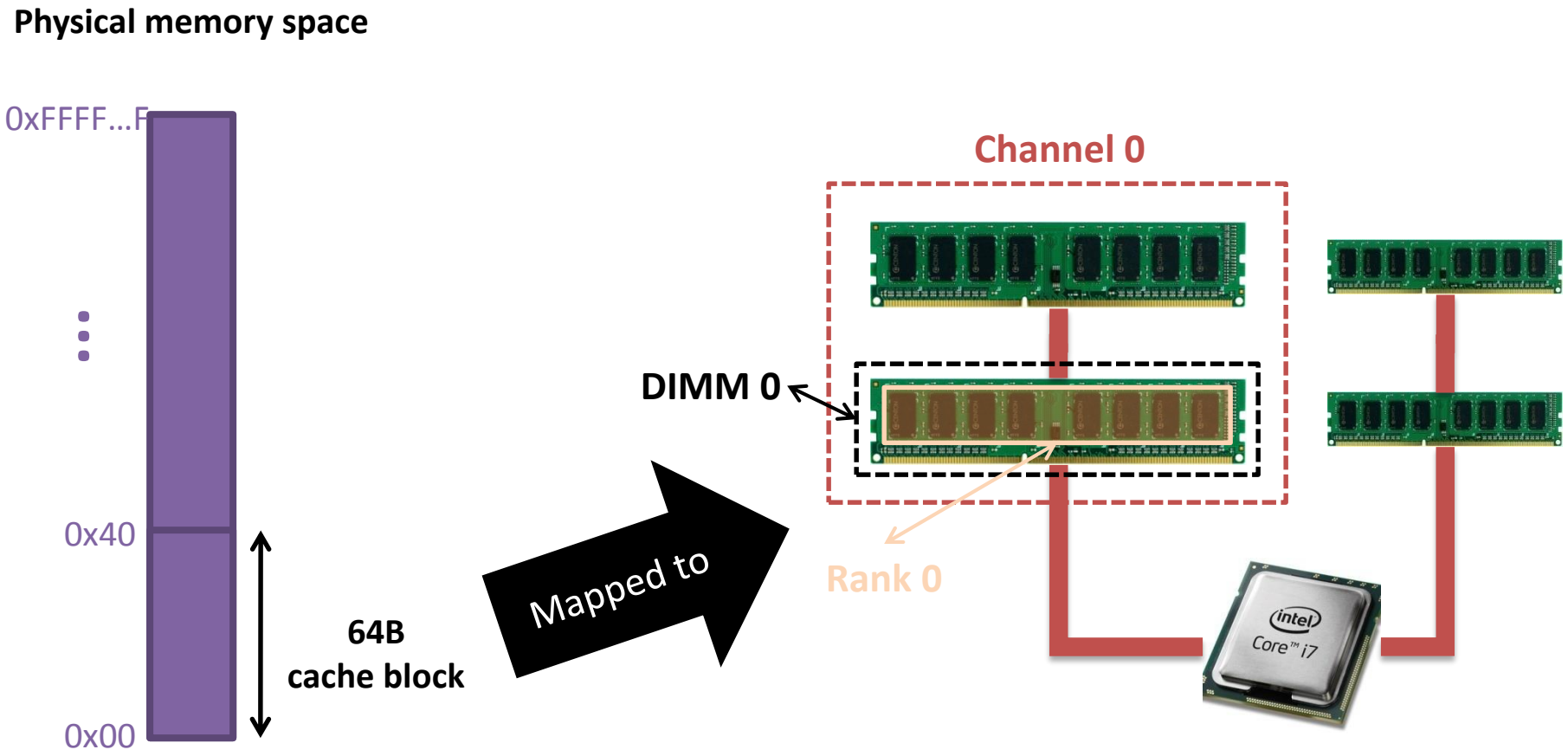
Breaking down a chip → Banks



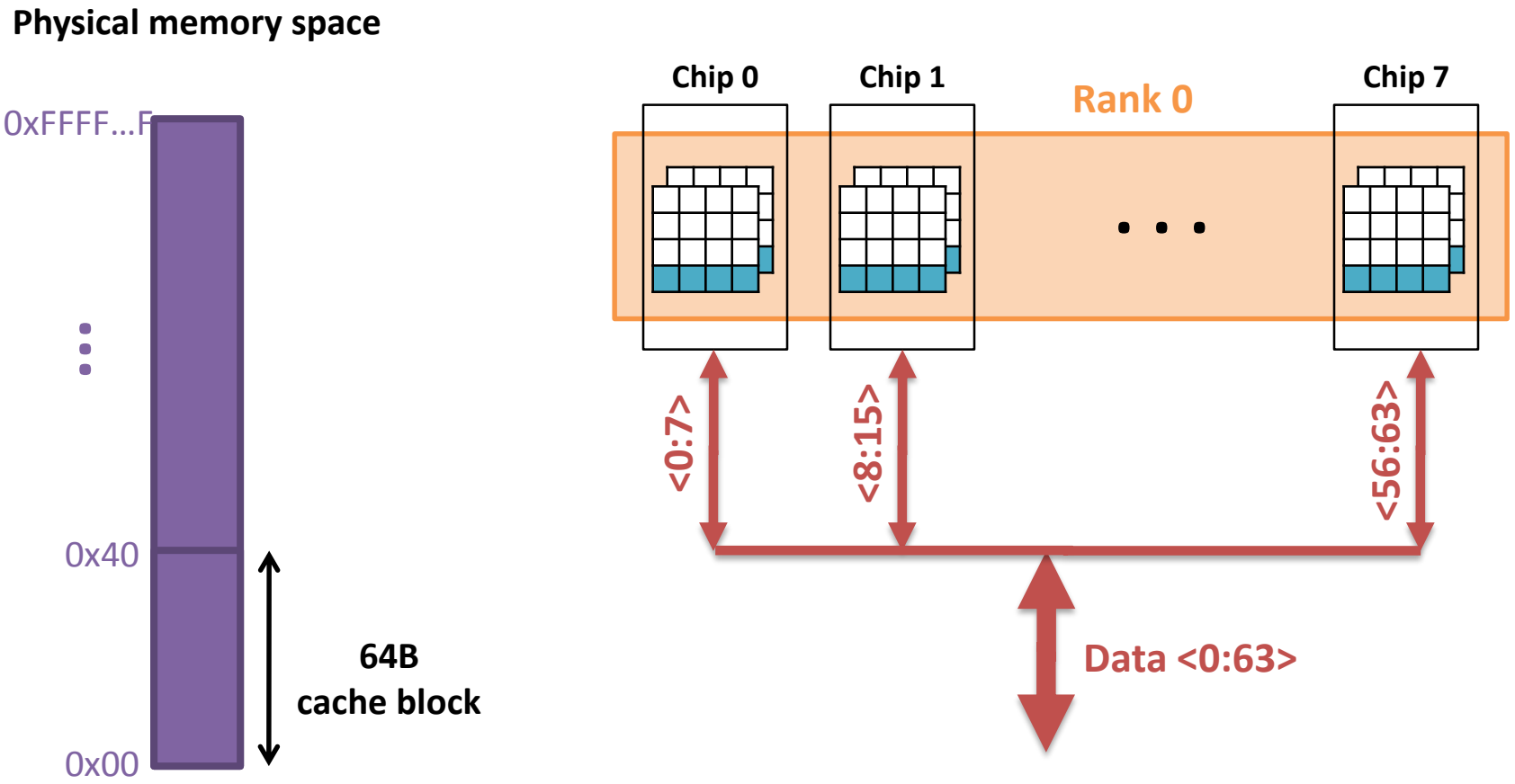
Breaking Down a Bank



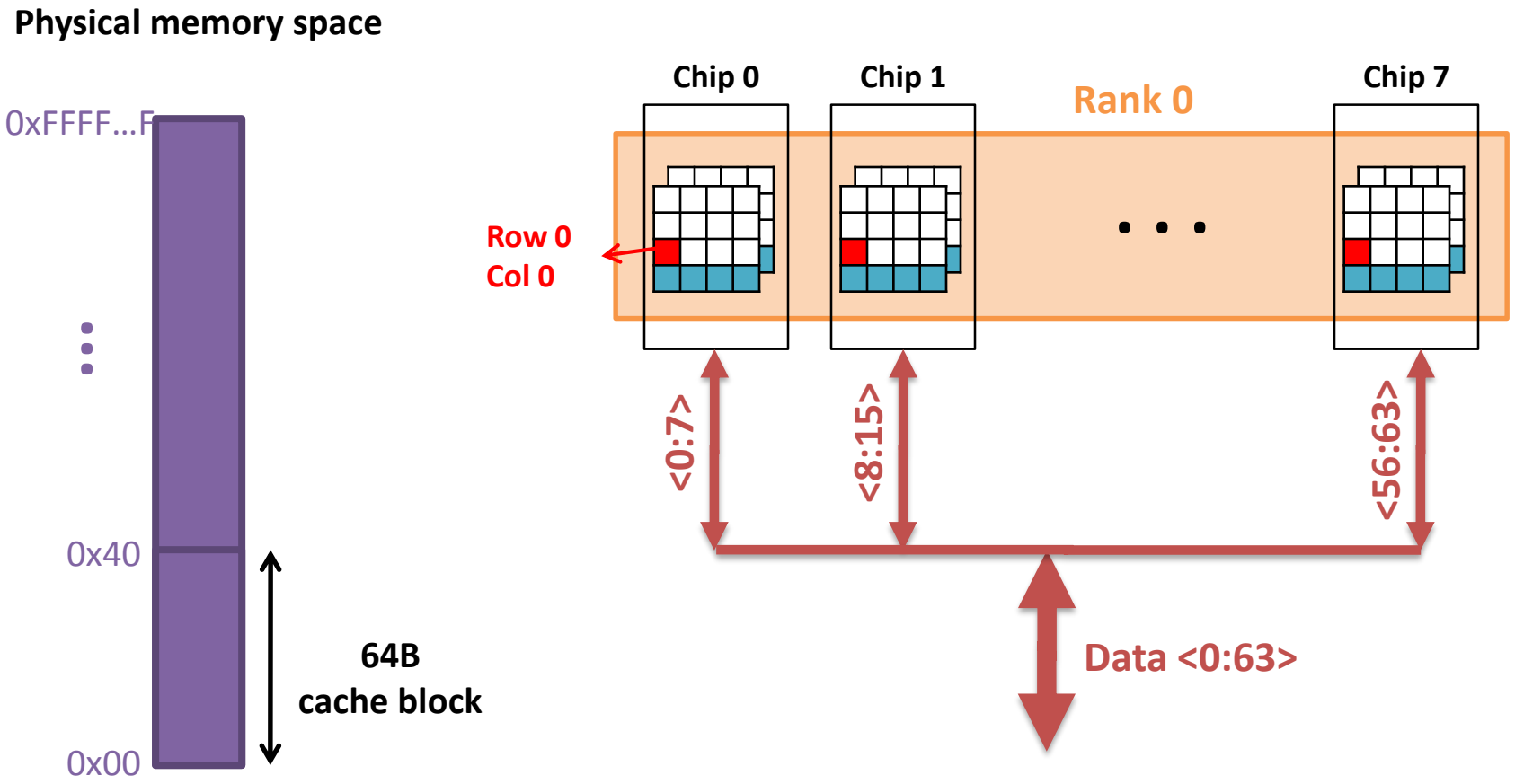
Παράδειγμα: Μεταφορά ενός cache block



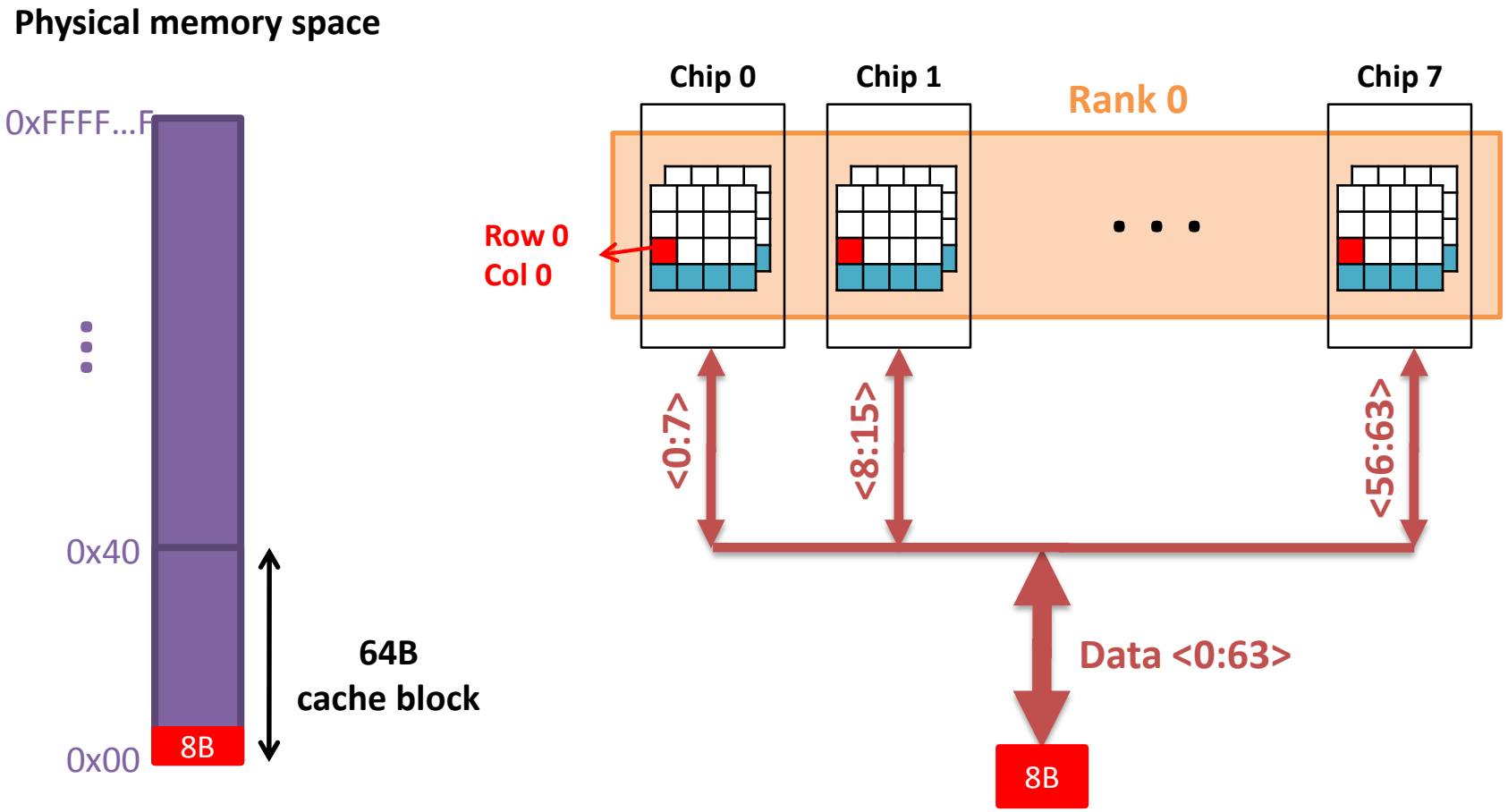
Παράδειγμα: Μεταφορά ενός cache block



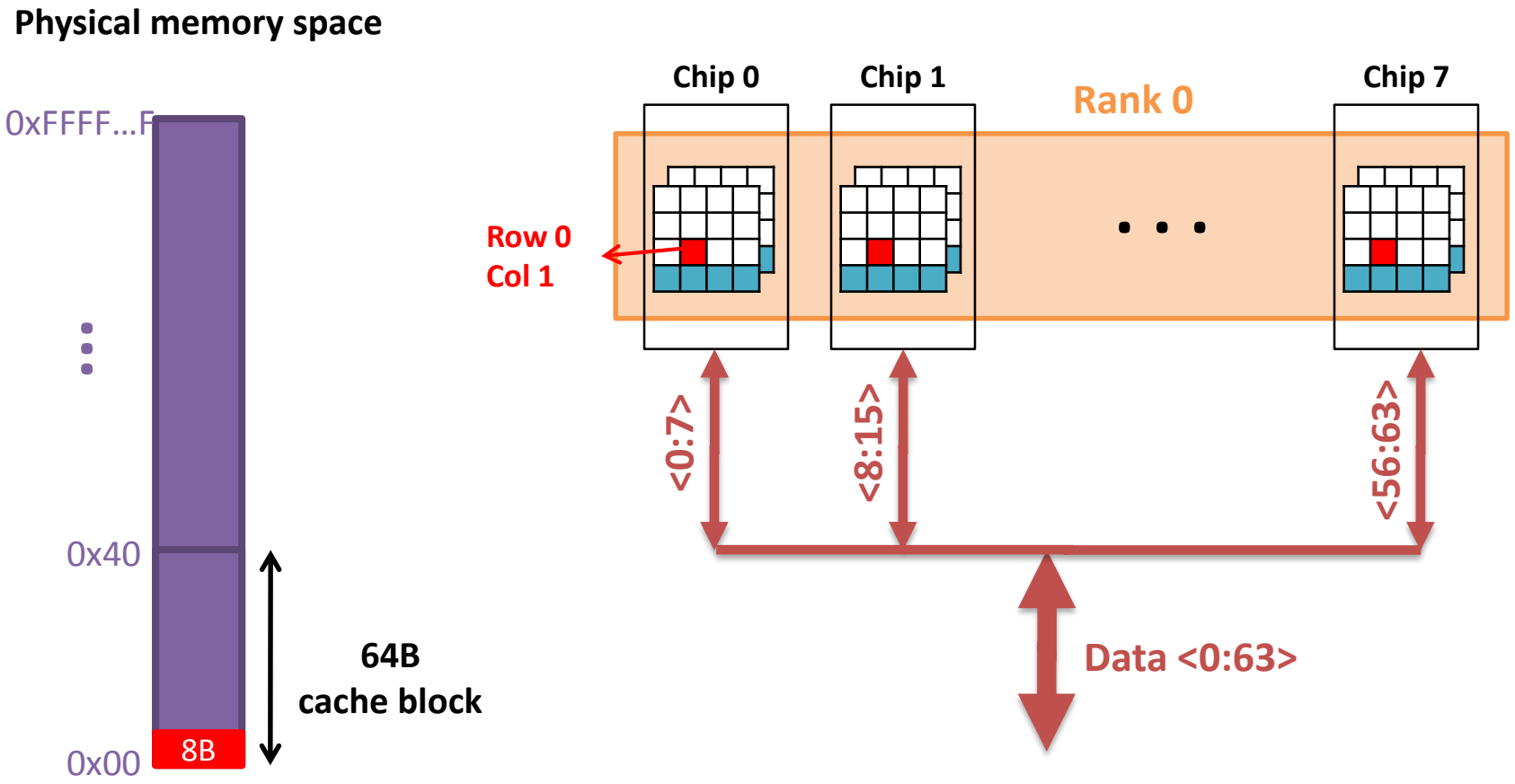
Παράδειγμα: Μεταφορά ενός cache block



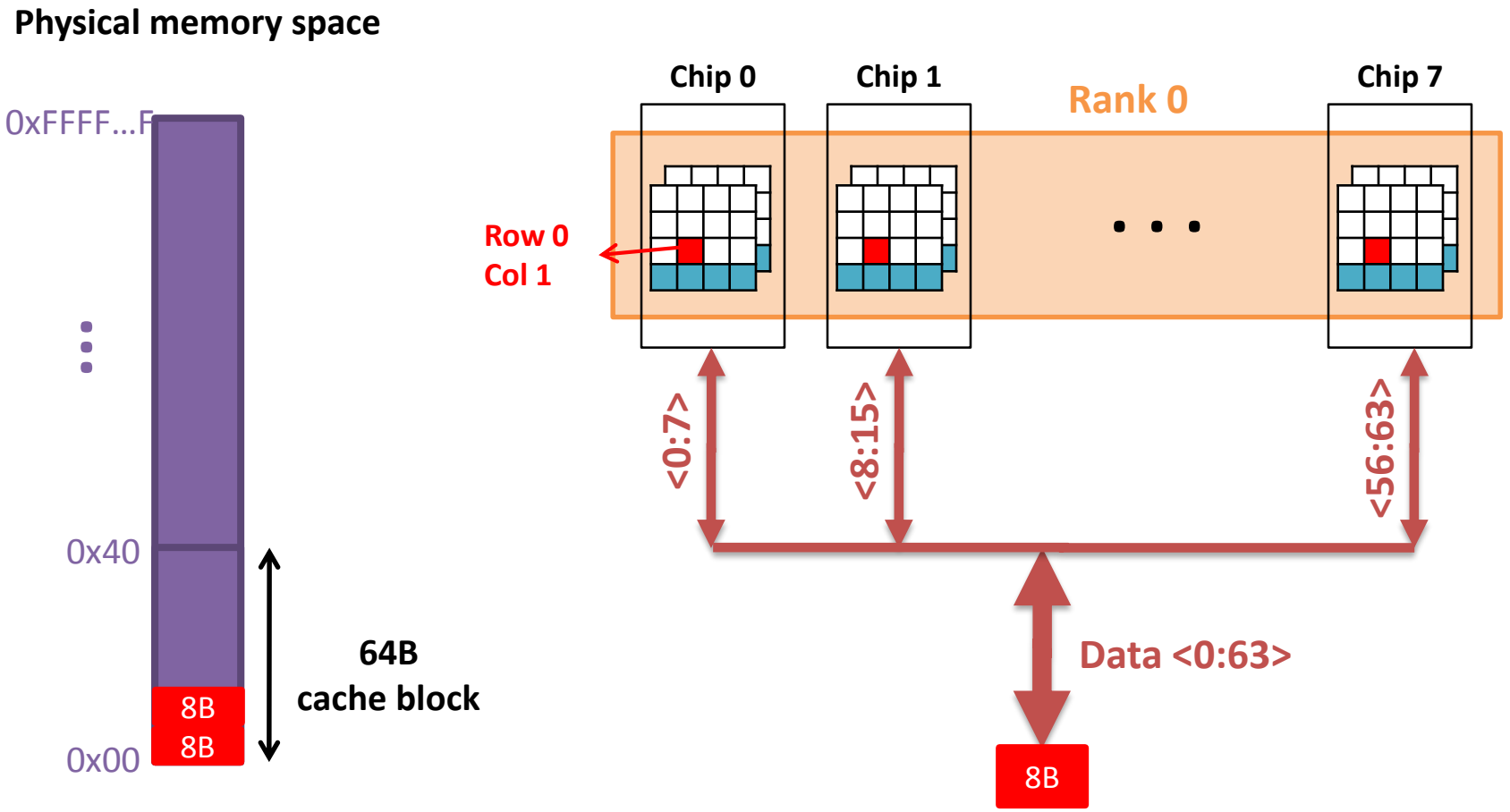
Παράδειγμα: Μεταφορά ενός cache block



Παράδειγμα: Μεταφορά ενός cache block



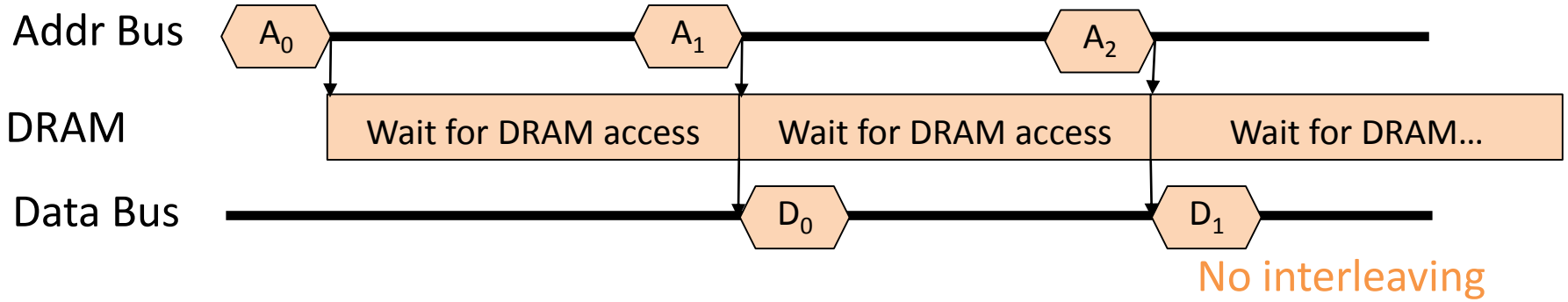
Παράδειγμα: Μεταφορά ενός cache block



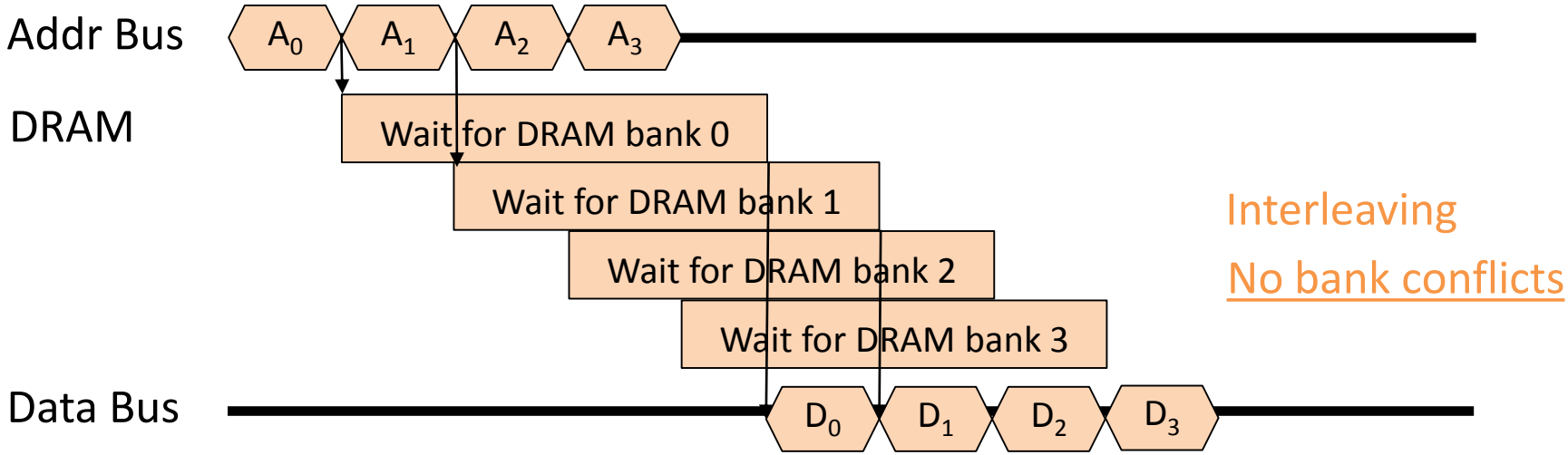
8 κύκλοι I/O χρειάζονται για τη μεταφορά ενος block 64B
Διαβάζονται 8 στήλες του κατανεμημένου bank ακολουθητικά

Γιατί τα πολλαπλά banks (interleaving) είναι σημαντικά?

- Επιτρέπουν πολλαπλές ταυτόχρονες προσβάσεις μνήμης



No interleaving



Interleaving

No bank conflicts

Αντιστοίχιση (mapping) σε banks (interleaving)

- Bits της διεύθυνσης χρησιμοποιούνται για την αντιστοίχιση
- Παράδειγμα: 2GB μνήμη, 8 banks, 16K rows & 2K columns per bank, memory bus 8 bytes

Row interleaving

Row (14 bits)	Bank (3 bits)	Column (11 bits)	Byte in bus (3 bits)

- Ακολουθητικά rows σε ακολουθητικά banks
- Προσπελάσεις ακολουθητικών cache blocks εξυπηρετούνται με pipelined τρόπο

Cache Block interleaving (cache block 64B)

Row (14 bits)	High Col (8 bits)	Bank (3 bits)	Low Col (3 bits)	Byte in bus (3 bits)

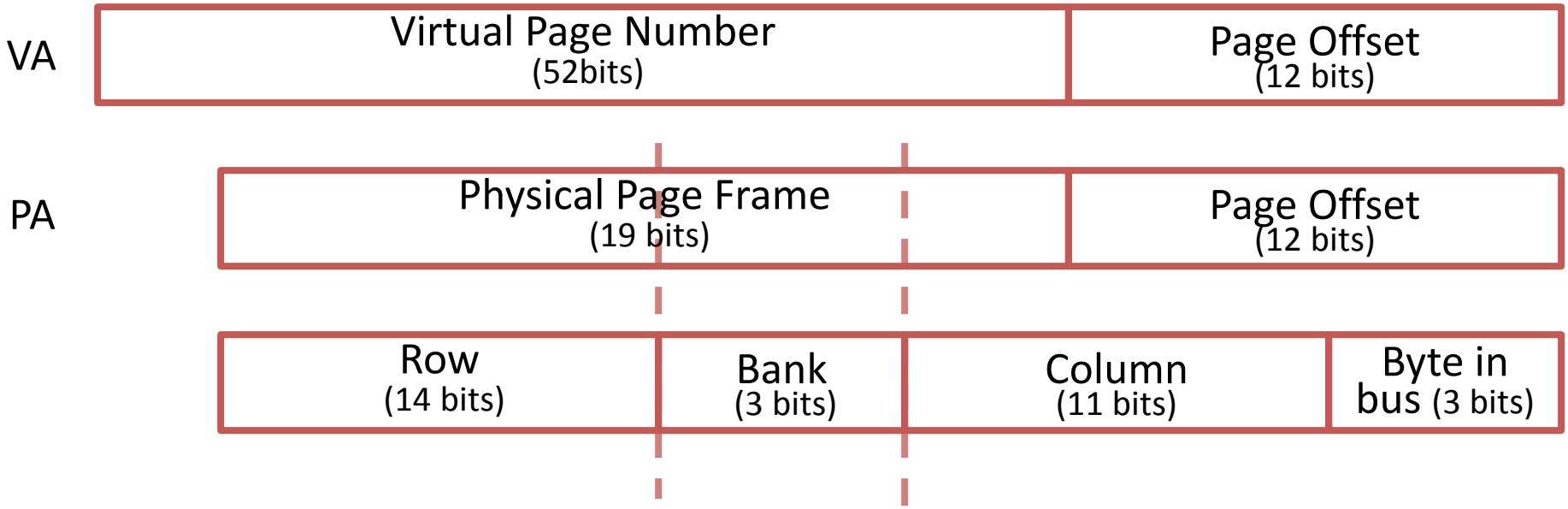
- Ακολουθητικά cache blocks σε ακολουθητικά banks
- Προσπελάσεις ακολουθητικών cache blocks εξυπηρετούνται παράλληλα

Πολλαπλά banks

- Επιτρέπουν ταυτόχρονες προσβάσεις μνήμης (concurrency)
- Bits της διεύθυνσης χρησιμοποιούνται για την αντιστοίχιση
- Στόχος: Μείωση των bank conflicts
- Πως επιλέγω τα bits-δείκτες για διασπορά στα banks?
  - Τα LSB έχουν μεγαλύτερη εντροπία
  - Χρήση συναρτίσεων κατακερματισμού (hash)
    (π.χ XOR διαφόρων bits της διεύθυνσης)

OS interference

- Virtual Memory: το λειτουργικό σύστημα μπορεί να παρέμβει στη διαδικασία αντιστοίχισης χωρίς έλεγχο (uncontrolled)

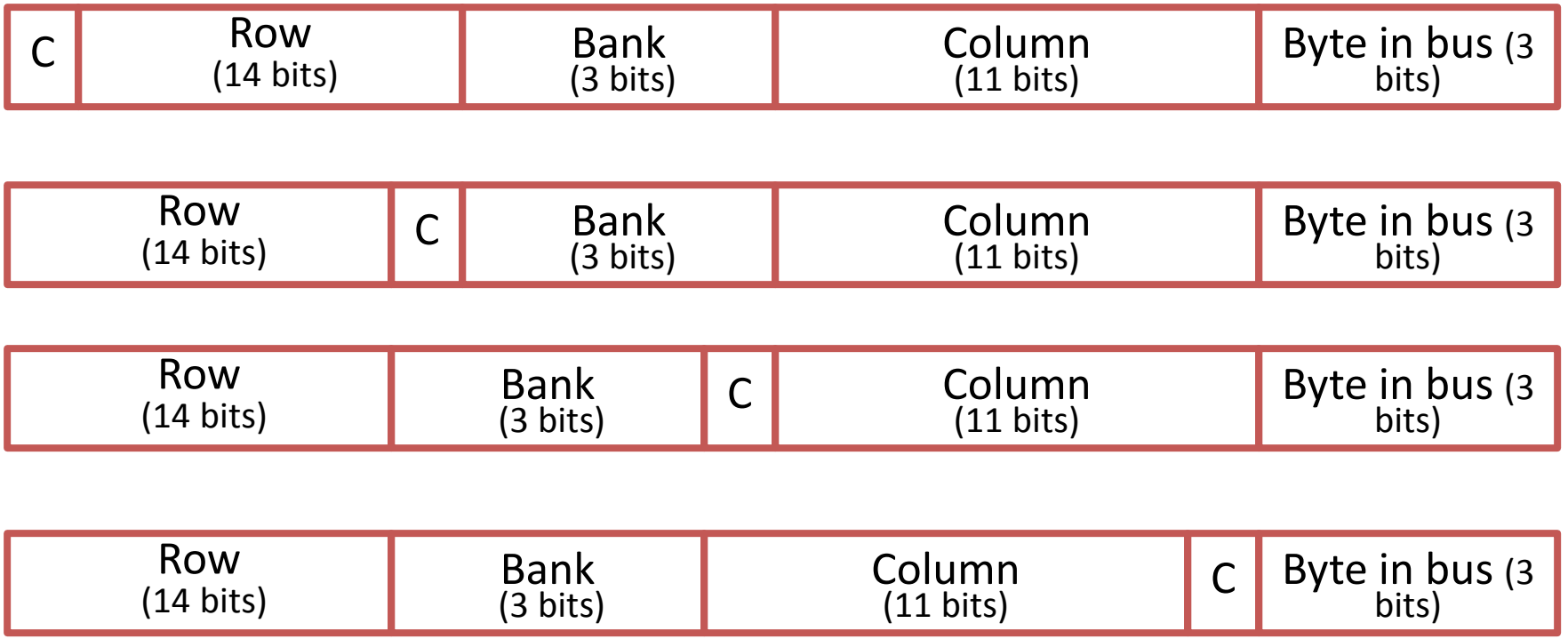


Πολλαπλά channels

- Όμοια οφέλη με πολλαπλά banks (concurrency)
- Ακόμα καλύτερα διότι έχουν διαφορετικά buses
- Αυξημένο εύρος ζώνης (bus bandwidth)

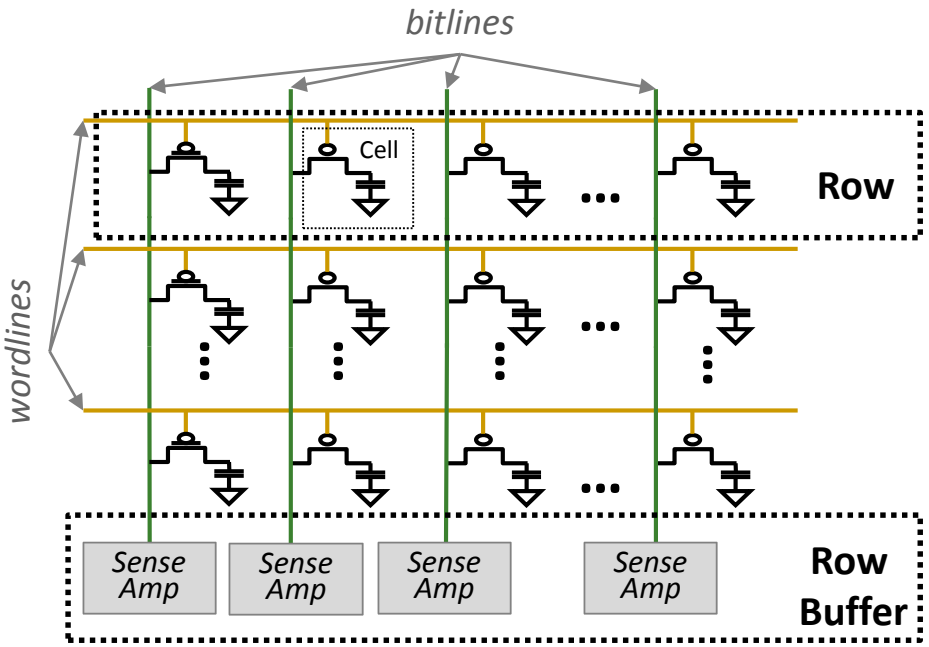
- Μειονεκτήματα:
  - Αυξημένο κόστος σε $ και σε area

Αντιστοίχιση σε multiple channels



Χρόνος απόκρισης DRAM

Επισκόπηση DRAM Bank



Λειτουργία DRAM

3 Βασικές Εντολές Προσπέλασης:

Activate — Open row (place it into Row Buffer)

Read/Write — Read/Write column in the Row Buffer

Precharge — Close the row and prepare the bank for a next access

Προσπέλαση / Περιορισμοί Χρονισμού

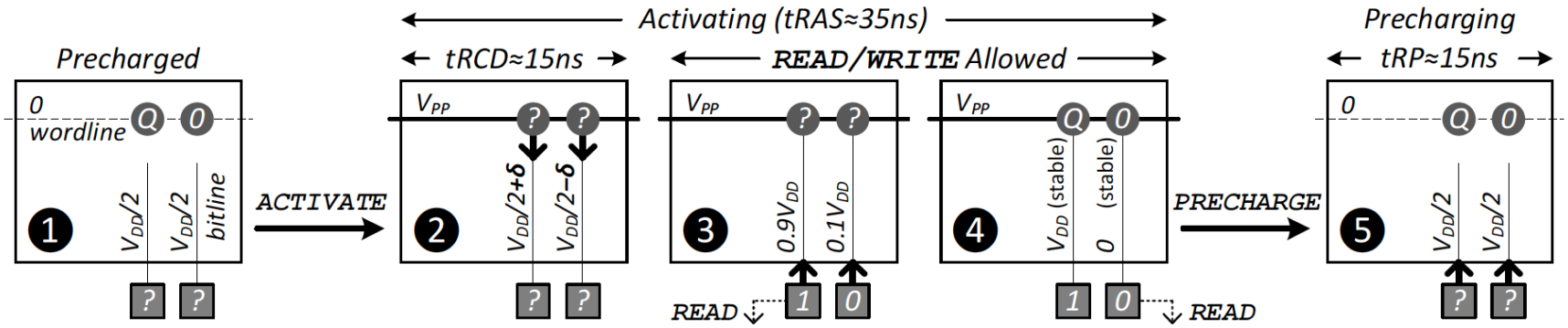


Figure 4. DRAM bank operation: Steps involved in serving a memory request [17] ($V_{PP} > V_{DD}$)

Category	RowCmd↔RowCmd			RowCmd↔ColCmd			ColCmd↔ColCmd			ColCmd→DATA	
Name	*tRC*	*tRAS*	*tRP*	*tRCD*	*tRTP*	*tWR**	*tCCD*	*tRTW*†	*tWTR**	*CL*	*CWL*
Commands	A→A	A→P	P→A	A→R/W	R→P	W*→P	R(W)→R(W)	R→W	W*→R	R→DATA	W→DATA
Scope	Bank	Bank	Bank	Bank	Bank	Bank	Channel	Rank	Rank	Bank	Bank
Value (ns)	**~50**	~35	13-15	13-15	~7.5	**15**	5-7.5	11-15	~7.5	13-15	10-15

A: ACTIVATE– P: PRECHARGE– R: READ– W: WRITE

∗ Goes into effect after the last write *data*, not from the WRITE command

† Not explicitly specified by the JEDEC DDR3 standard [18]. Defined as a function of other timing constraints.

Table 1. Summary of DDR3-SDRAM timing constraints (derived from Micron's 2Gb DDR3-SDRAM datasheet [33])

Προσπέλαση / Περιορισμοί χρονισμού

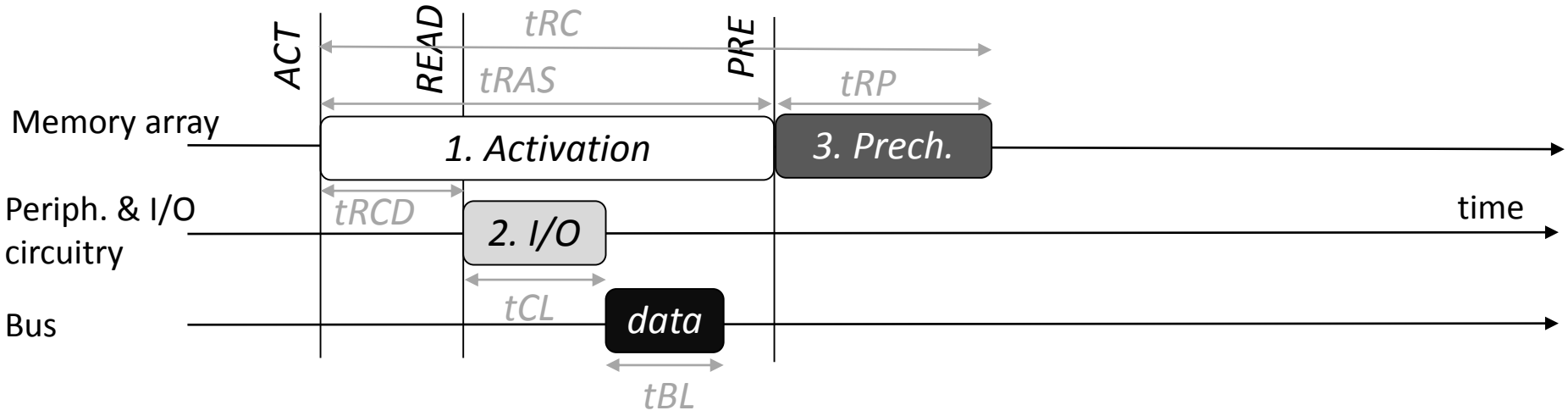


Table 2. Timing Constraints (DDR3-1066) [43]

Phase	Commands	Name	Value
1	ACT → READ ACT → WRITE	tRCD	15ns
	ACT → PRE	tRAS	37.5ns
2	READ → *data* WRITE → *data*	tCL tCWL	15ns 11.25ns
	data burst	tBL	7.5ns
3	PRE → ACT	tRP	15ns
1 & 3	ACT → ACT	tRC (tRAS+tRP)	52.5ns

The three phases of a DRAM access

- Διαφορετικές ακολουθίες προσπελάσεων με διαφορετικές τοπικότητες (i.e bank, row, rank etc hit/conflict) έχουν διαφορετικούς χρόνους απόκρισης

Προσπέλαση / Περιορισμοί χρονισμού

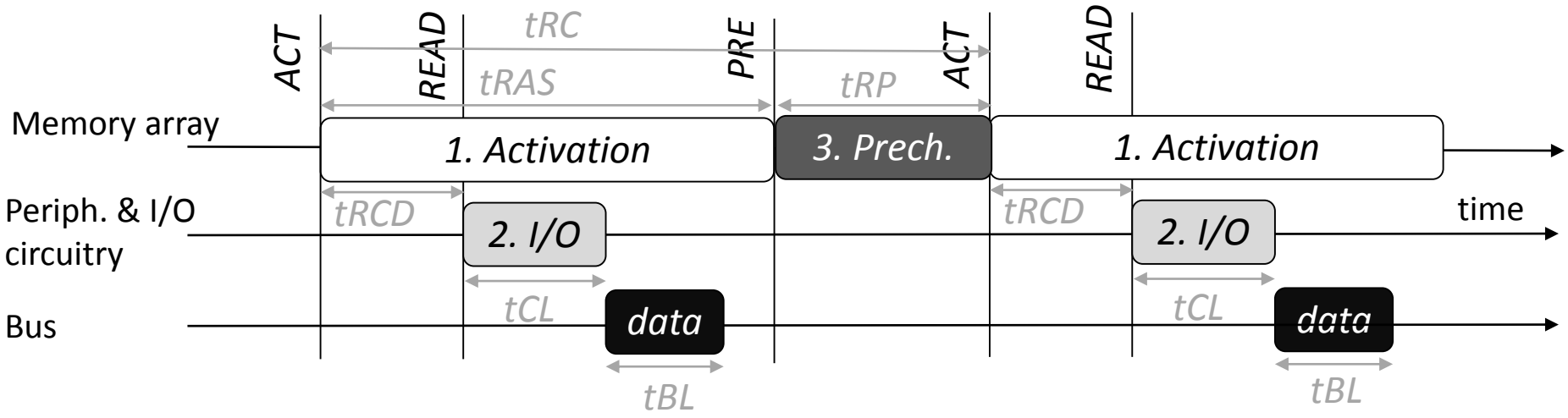


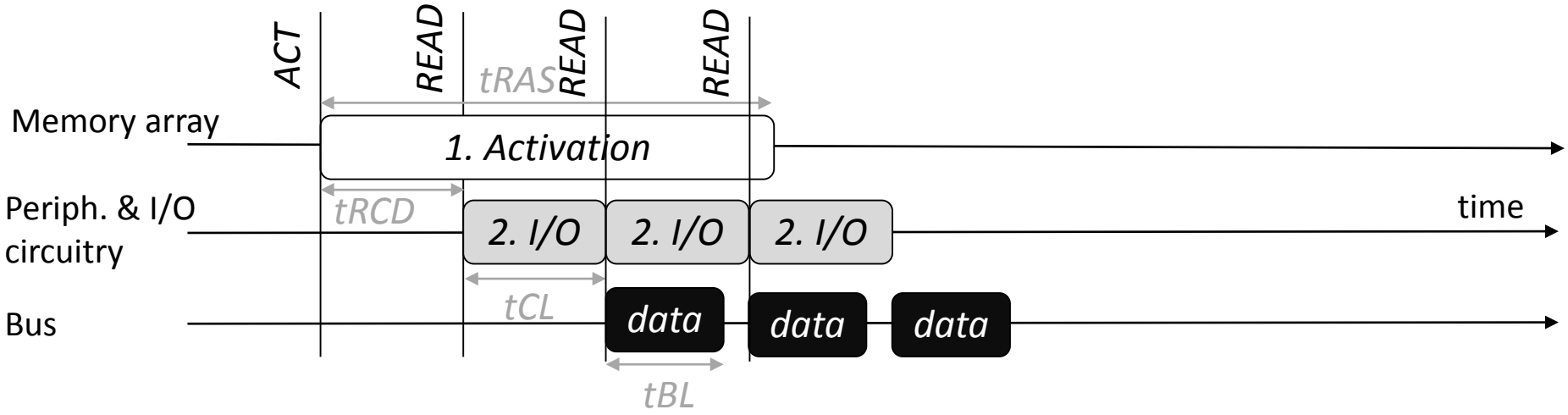
Table 2. Timing Constraints (DDR3-1066) [43]

Phase	Commands	Name	Value
1	ACT → READ ACT → WRITE	tRCD	15ns
	ACT → PRE	tRAS	37.5ns
2	READ → *data* WRITE → *data*	tCL tCWL	15ns 11.25ns
	data burst	tBL	7.5ns
3	PRE → ACT	tRP	15ns
1 & 3	ACT → ACT	tRC (tRAS+tRP)	52.5ns

Consecutive accesses same bank
Row Closed or Row Conflict

- First access → *tRCD+tCL+tBL*
- Second access → *tRP+tRCD+tCL+tBL*

Προσπέλαση / Περιορισμοί χρονισμού



Consecutive accesses same bank
Row Opened and Row Buffer Hit
(only column access)

- First access → *tRCD+tCL+tBL*
- Second access → *tCL+tBL*
- Third Acess → *tCL+tBL*

Table 2. Timing Constraints (DDR3-1066) [43]

Phase	Commands	Name	Value
1	ACT → READ ACT → WRITE	tRCD	15ns
	ACT → PRE	tRAS	37.5ns
2	READ → *data* WRITE → *data*	tCL tCWL	15ns 11.25ns
	data burst	tBL	7.5ns
3	PRE → ACT	tRP	15ns
1 & 3	ACT → ACT	tRC (tRAS+tRP)	52.5ns

Ποιός ελέγχει τη λειτουργία της μνήμης?
Ελεγκτές μνήμης (Memory Controller)

Η DRAM ως παράδειγμα

- Οι μνήμες με μεγάλους χρόνους απόκρισης έχουν παρόμοια χαρακτηριστικά που απαιτούν διαχείριση/έλεγχο.

- Θα χρησιμοποιήσουμε την DRAM ως παράδειγμα
  - Όμως τα βασικά ζητήματα και αρχές χρονολόγησης αιτήσεων μνήμης και ελέγχου μνήμης που θα συζητήσουμε, είναι συναφή για ελεγκτές διαφόρων τύπων μνήμης

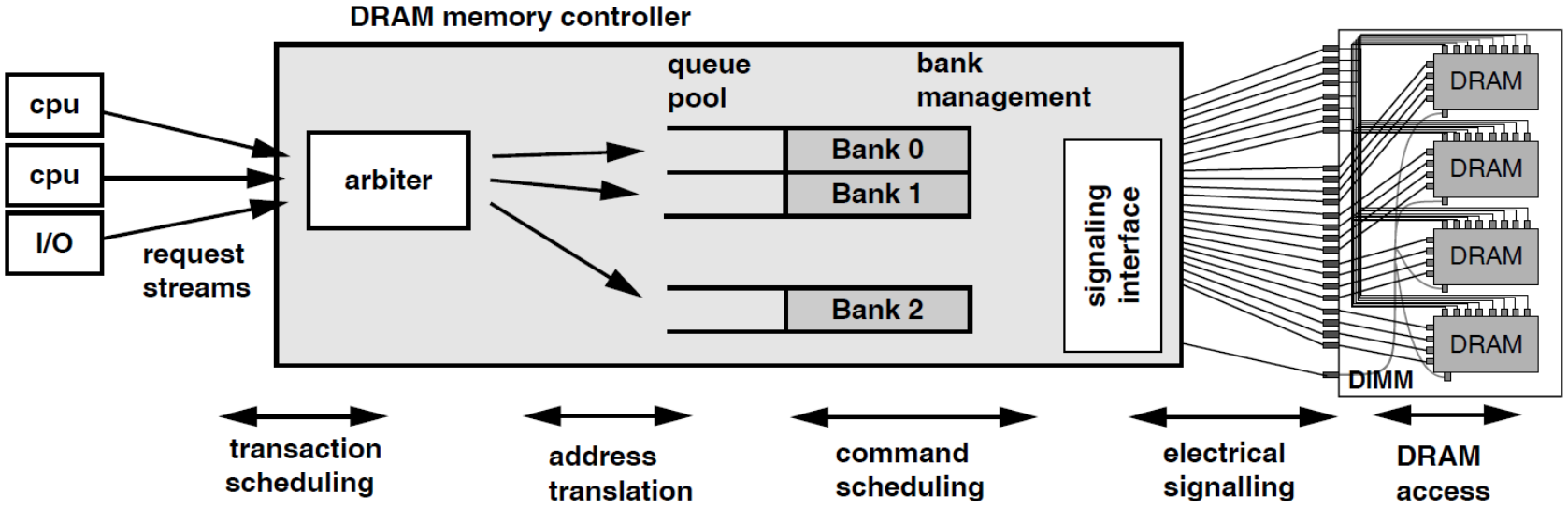
Λειτουργίες Memory Controller

- Διασφαλίζει την ορθή λειτουργία της DRAM (refresh/timing)
- Εξυπηρετεί τις αιτήσεις μνήμης προς την DRAM σεβόμενος τους περιορισμούς χρονισμού του DRAM chip
  - Μεταφράζει τις αιτήσεις μνήμης σε ακολουθίες εντολών DRAM (activate, read/write, precharge etc.)
  - Περιορισμοί: resource conflicts (bank, bus, channel etc)
- Αποθηκεύει τις αιτήσεις σε ενδιάμεση μνήμη (buffering) και τις χρονοδρομολογεί με στόχο την υψηλή επίδοση + QoS
  - Reordering
  - row-buffer/bank/rank/bus etc. management
- Διαχειρίζεται την κατανάλωση ισχύος της DRAM
  - Turn on/off DRAM chips

Τοποθέτηση DRAM controller

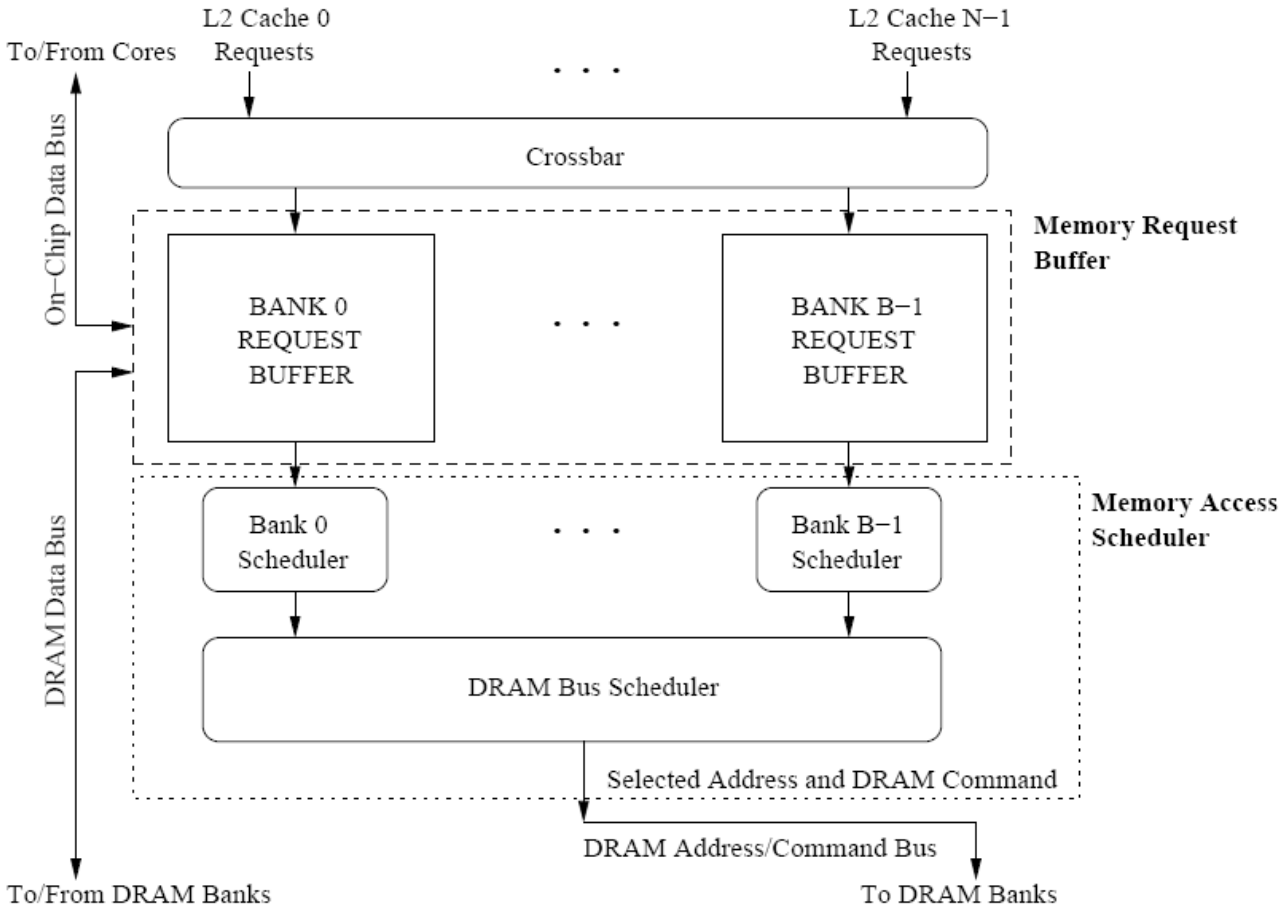
- Στον επεξεργαστή (on CPU chip)
  - Μικρότερος χρόνος απόκρισης για ένα memory access
  - Μεγαλύτερο εύρος ζώνης ανάμεσα στους πυρήνες και τον memory controller
  - Πιο εύκολη “επικοινωνία” πληροφορίας μεταξύ τους (π.χ πόσο σημαντικό είναι ένα request?)
- Στο DRAM chipset
  - Ευκολία πρόσδεσης (plug-in) διαφόρων τύπων μνήμης στο σύστημα
  - Μειωμένο power budget του CPU chip

Ενας σύγχρονος memory controller



Buffering/queueing requests per bank.
Scheduling per bank & across banks

Ενας σύγχρονος memory controller



Διαχείριση του Row Buffer

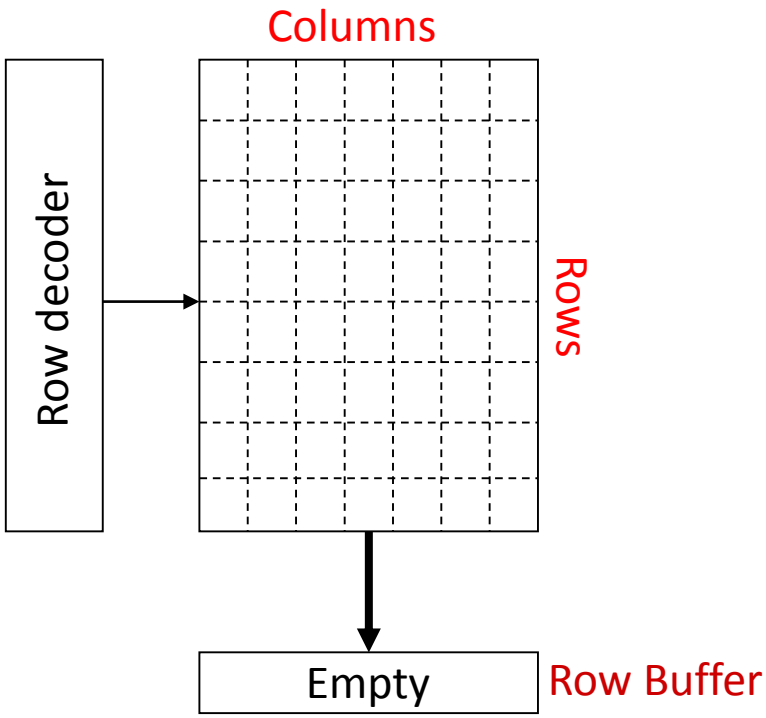
Πολιτικές:

- Open Row: Κρατάω τη γραμμή ανοιχτή στο Row Buffer
  (δεν κάνω precharge αμέσως).
  Ακολουθητικές προσπελάσεις σε στήλες
  ανοιχτής γραμμής έχουν πολύ μικρότερο κόστος.
  Ευνοεί αιτήσεις μνήμης με locality.
  (commodity systems, small agent number)

- Close Row: Κλείνω τη γραμμή στο Row Buffer
  (precharge αμέσως).
  Ακολουθητικές προσπελάσεις σε διαφορετικά rows έχουν μικρότερο κόστος.
  Ευνοεί irregular αιτήσεις μνήμης. (systems with large agent number)

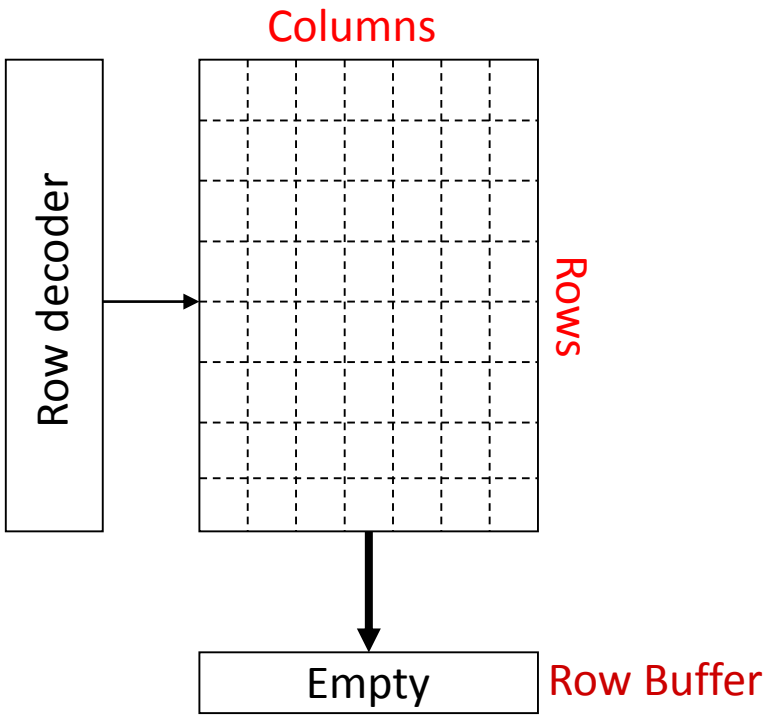
- Hybrid: Δυναμική προσαρμοστική (adaptive) τεχνική

Row Hit / Row Conflict



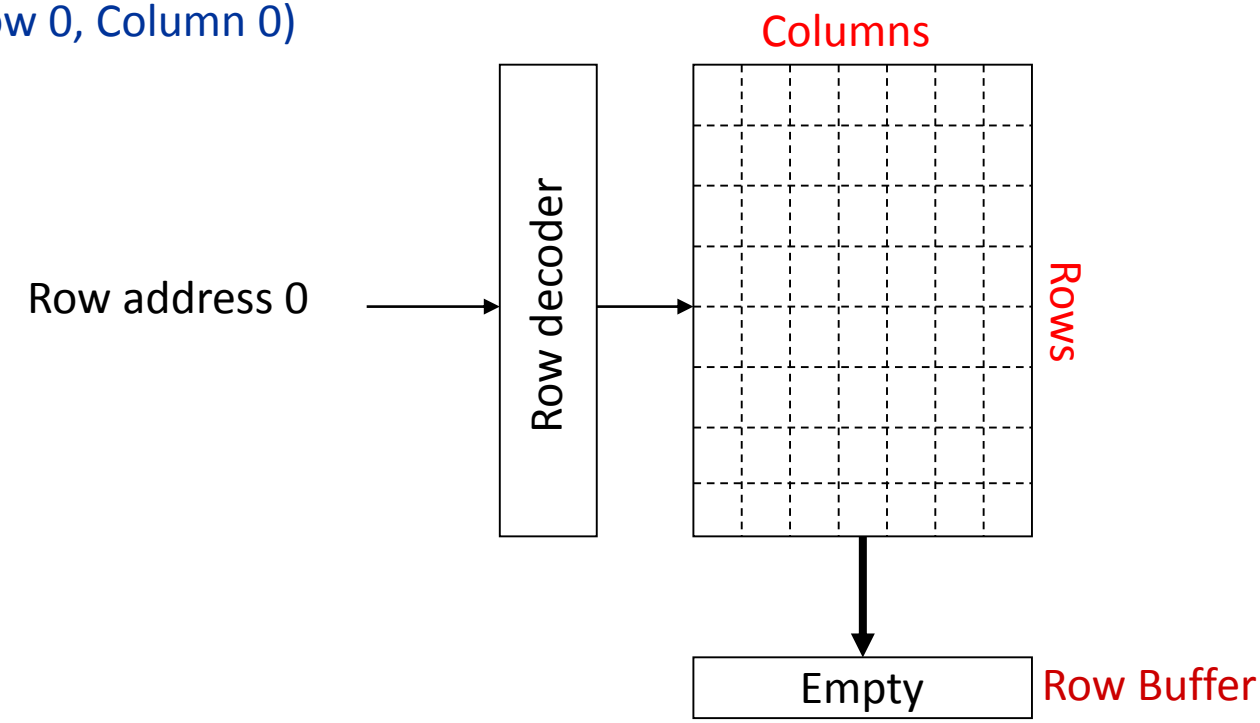
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)



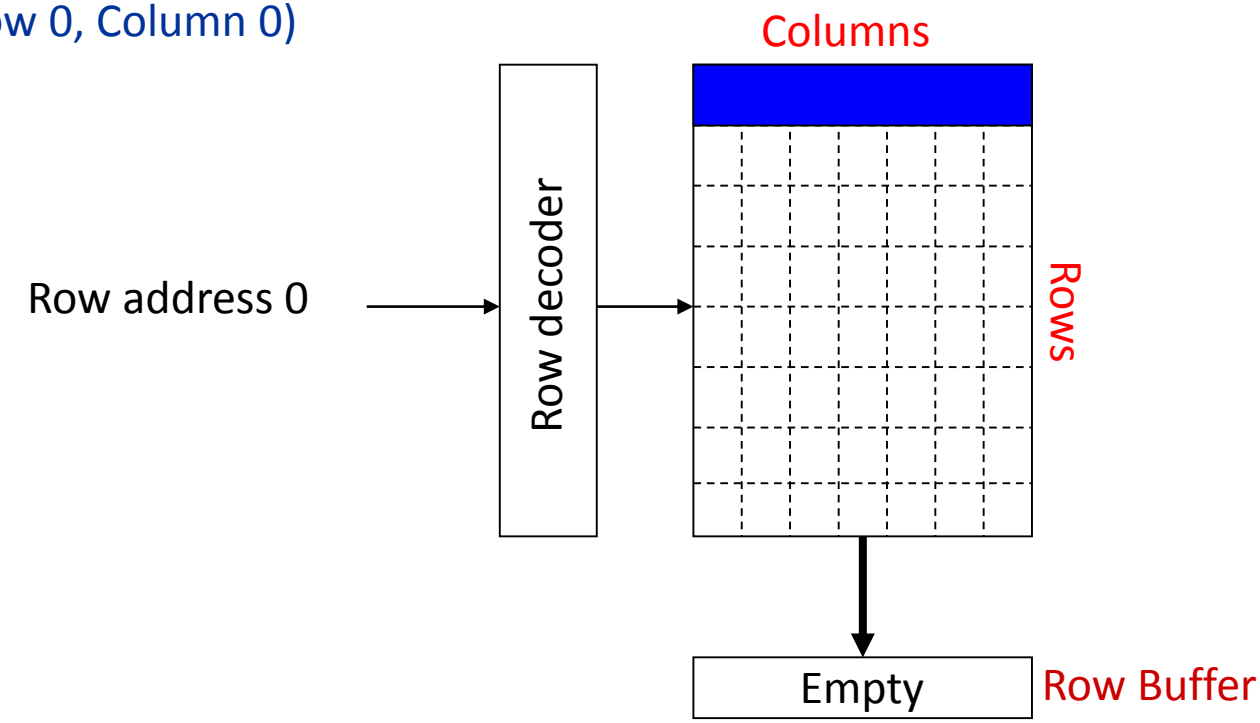
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)



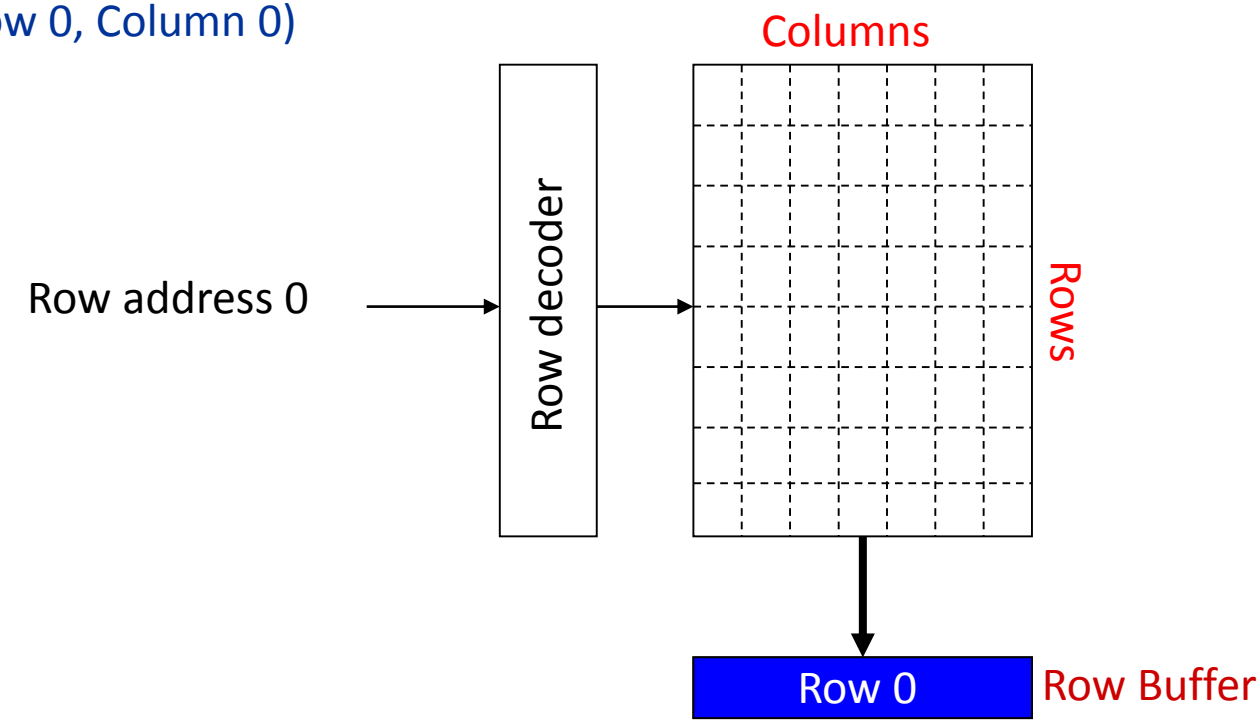
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)



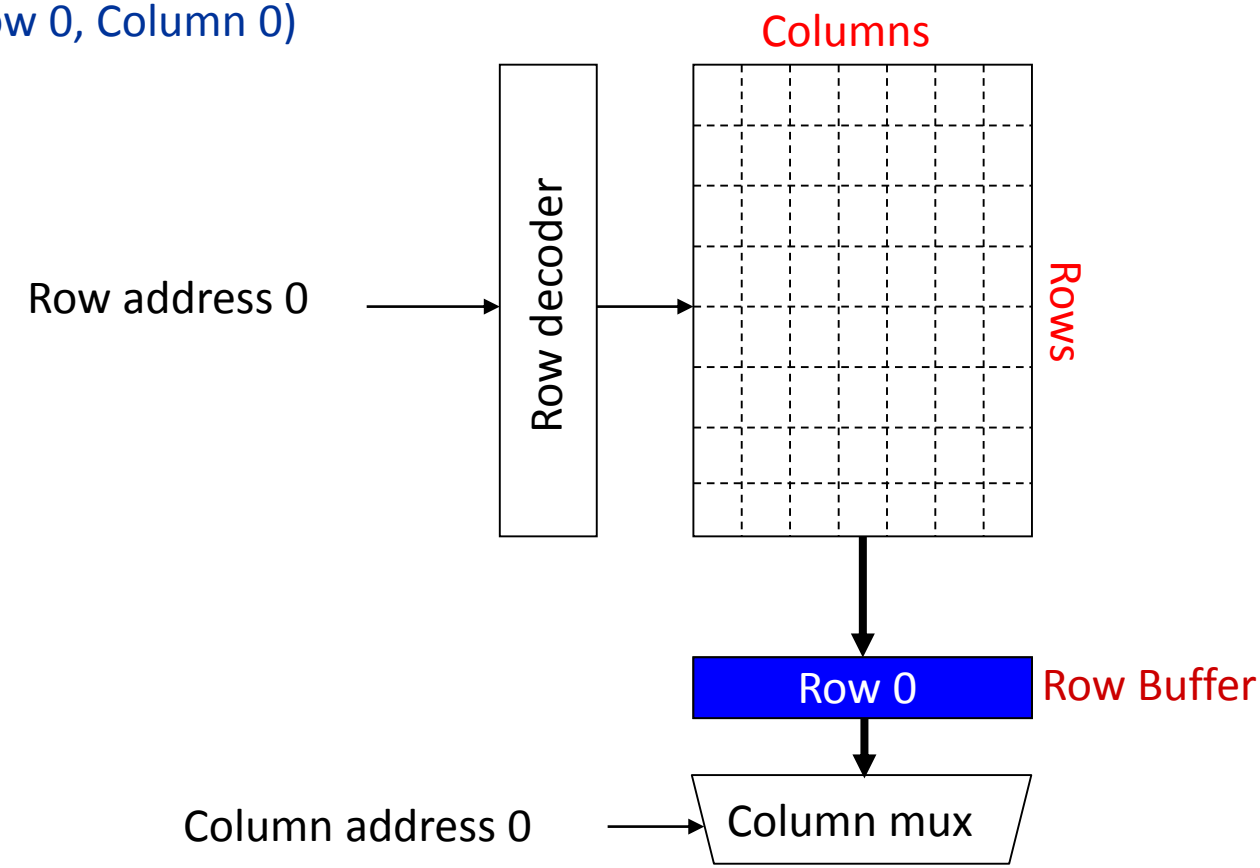
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)



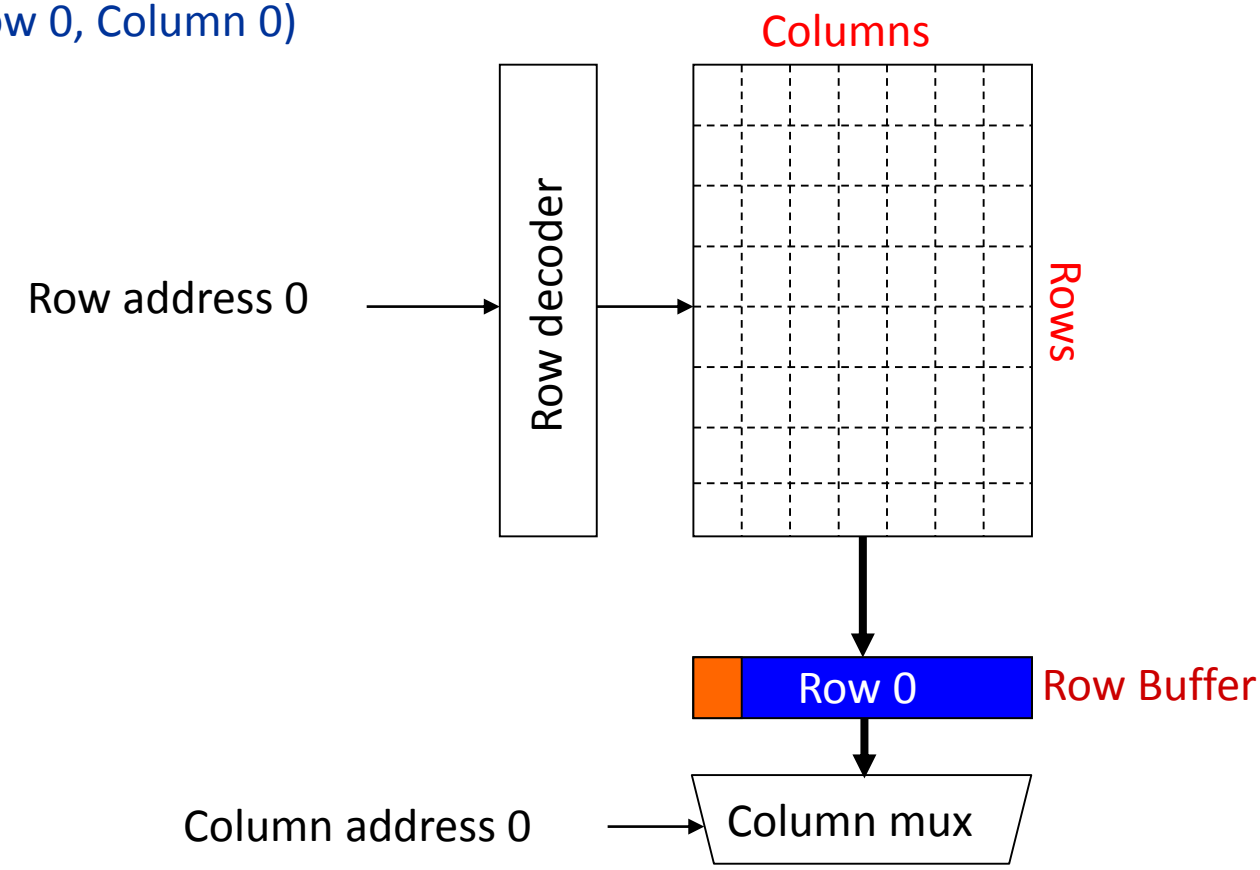
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)



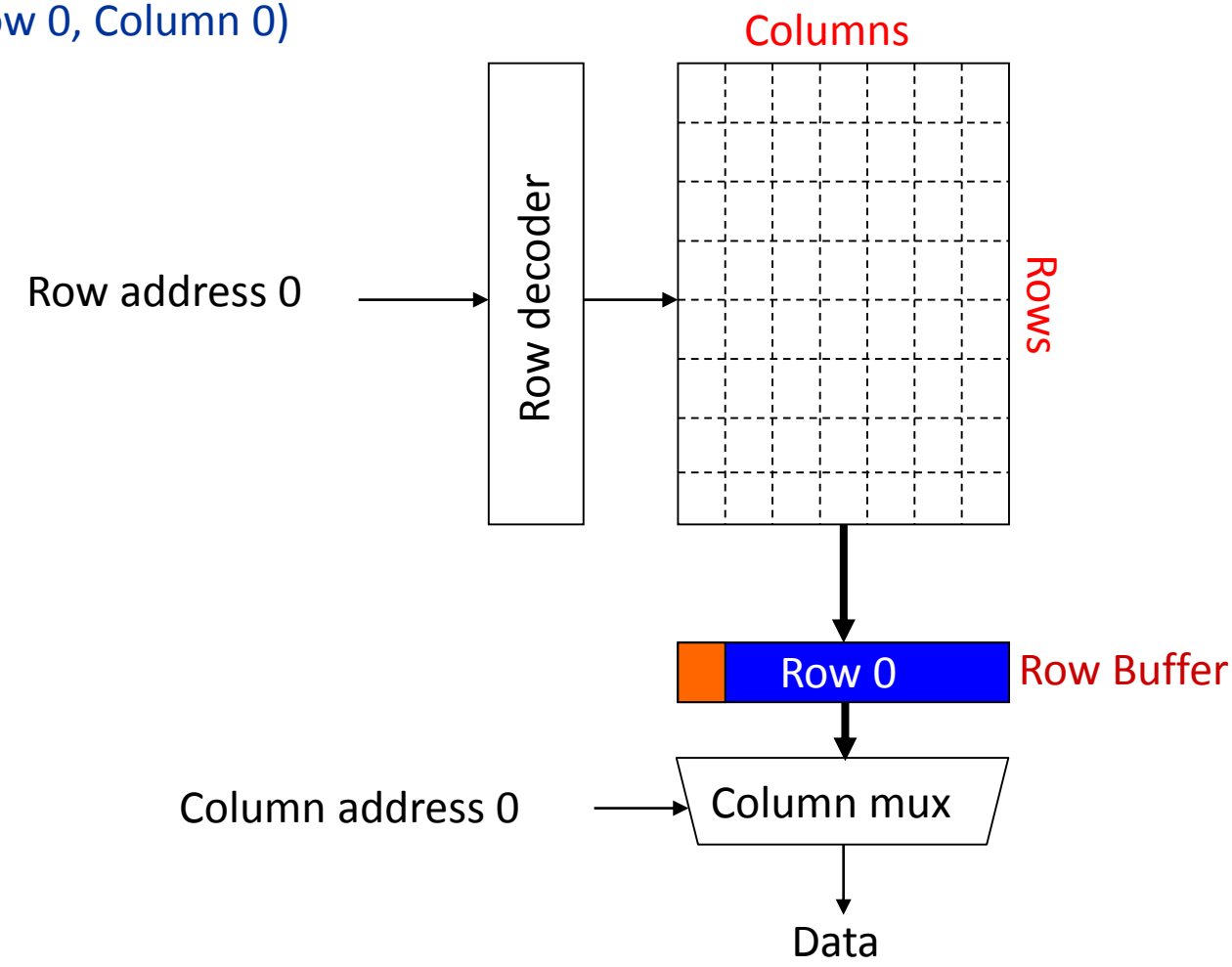
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)



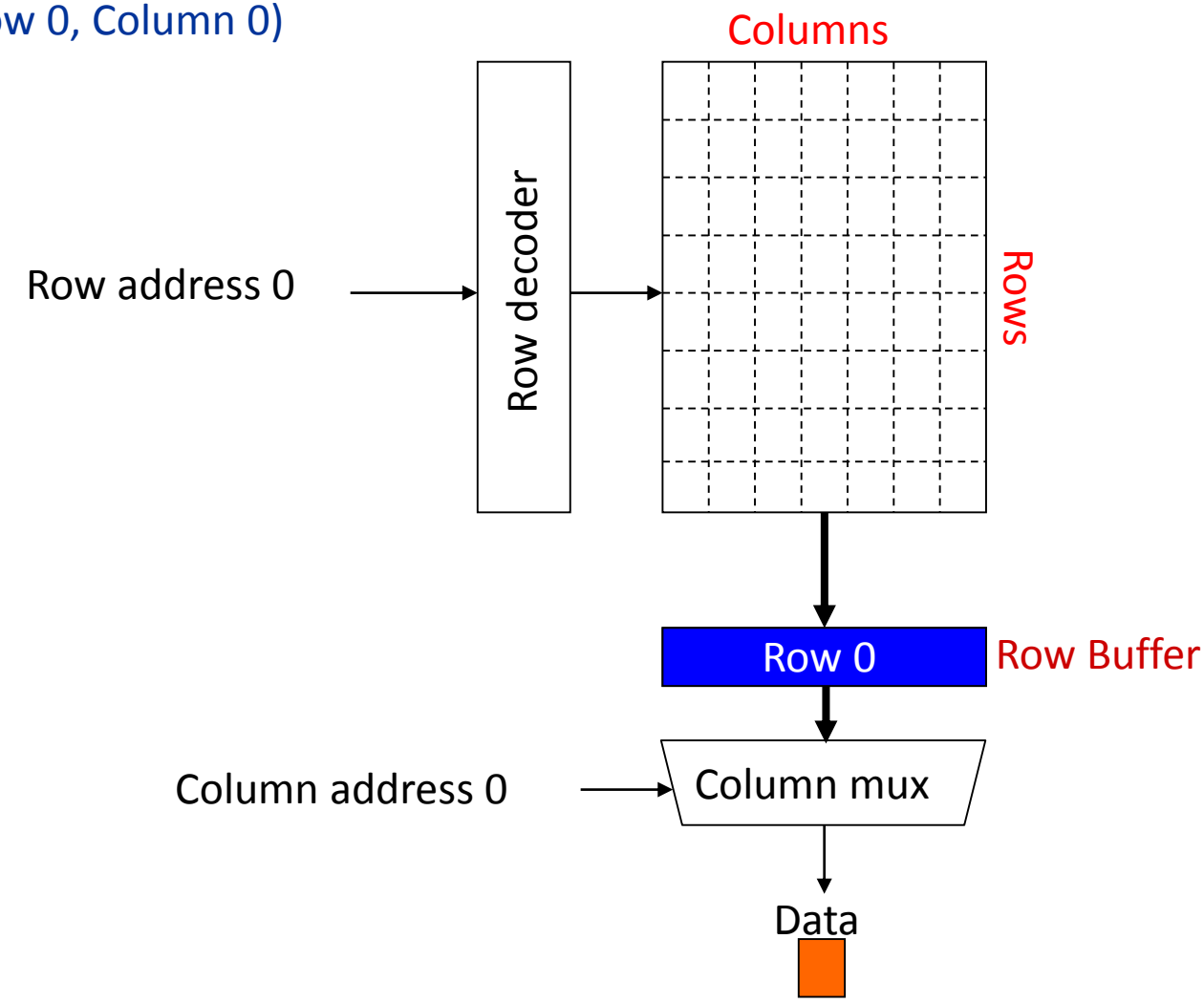
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)



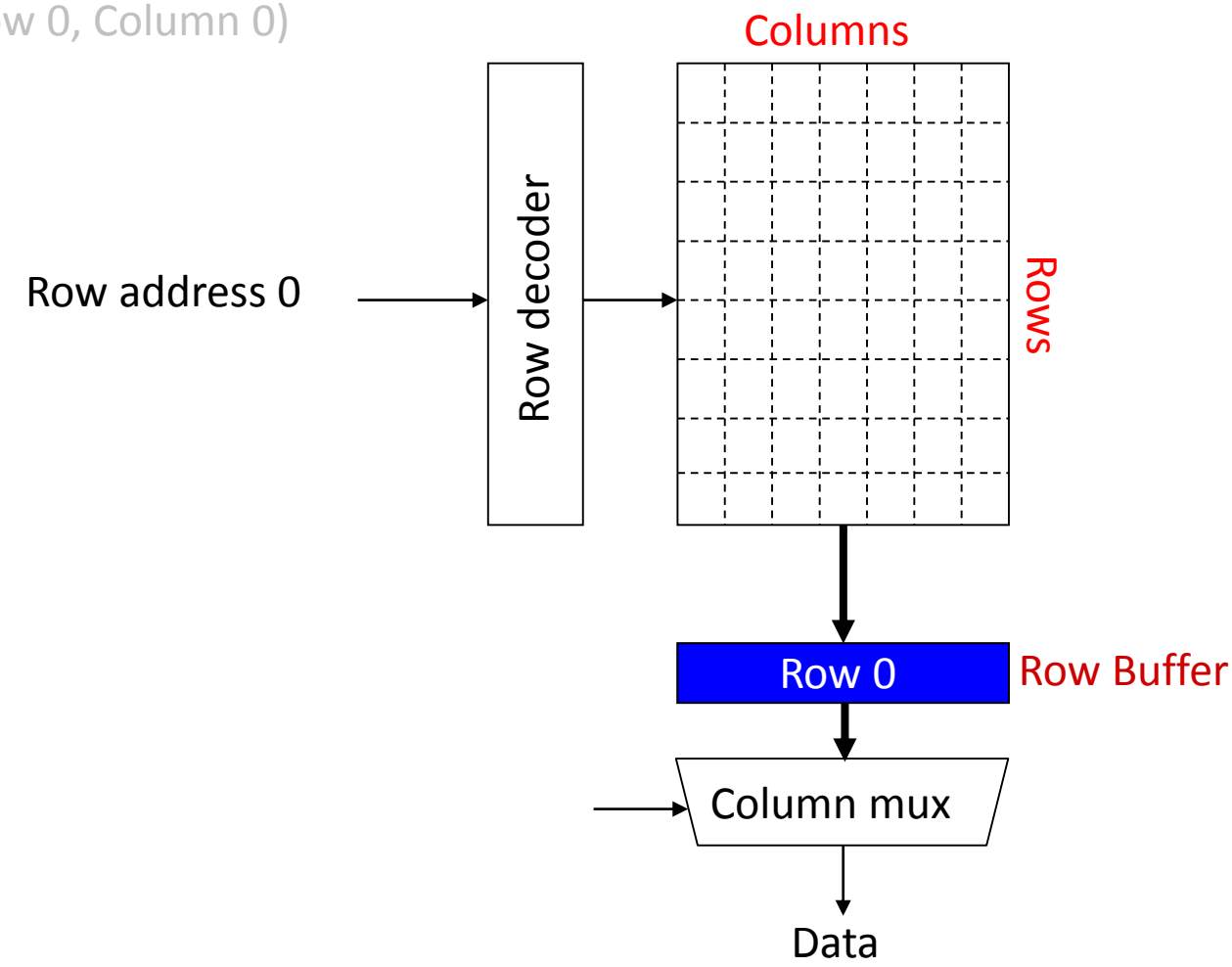
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)



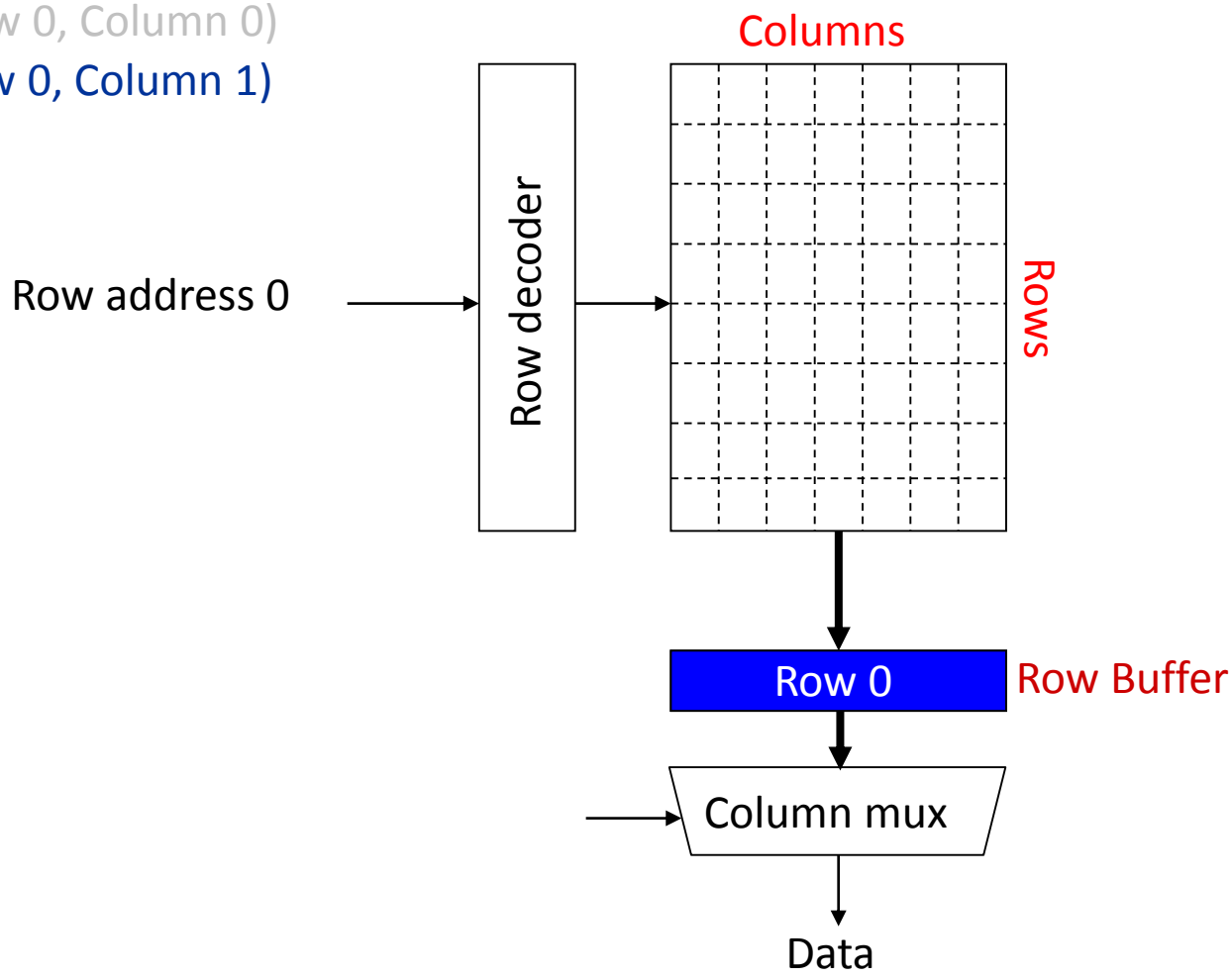
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)



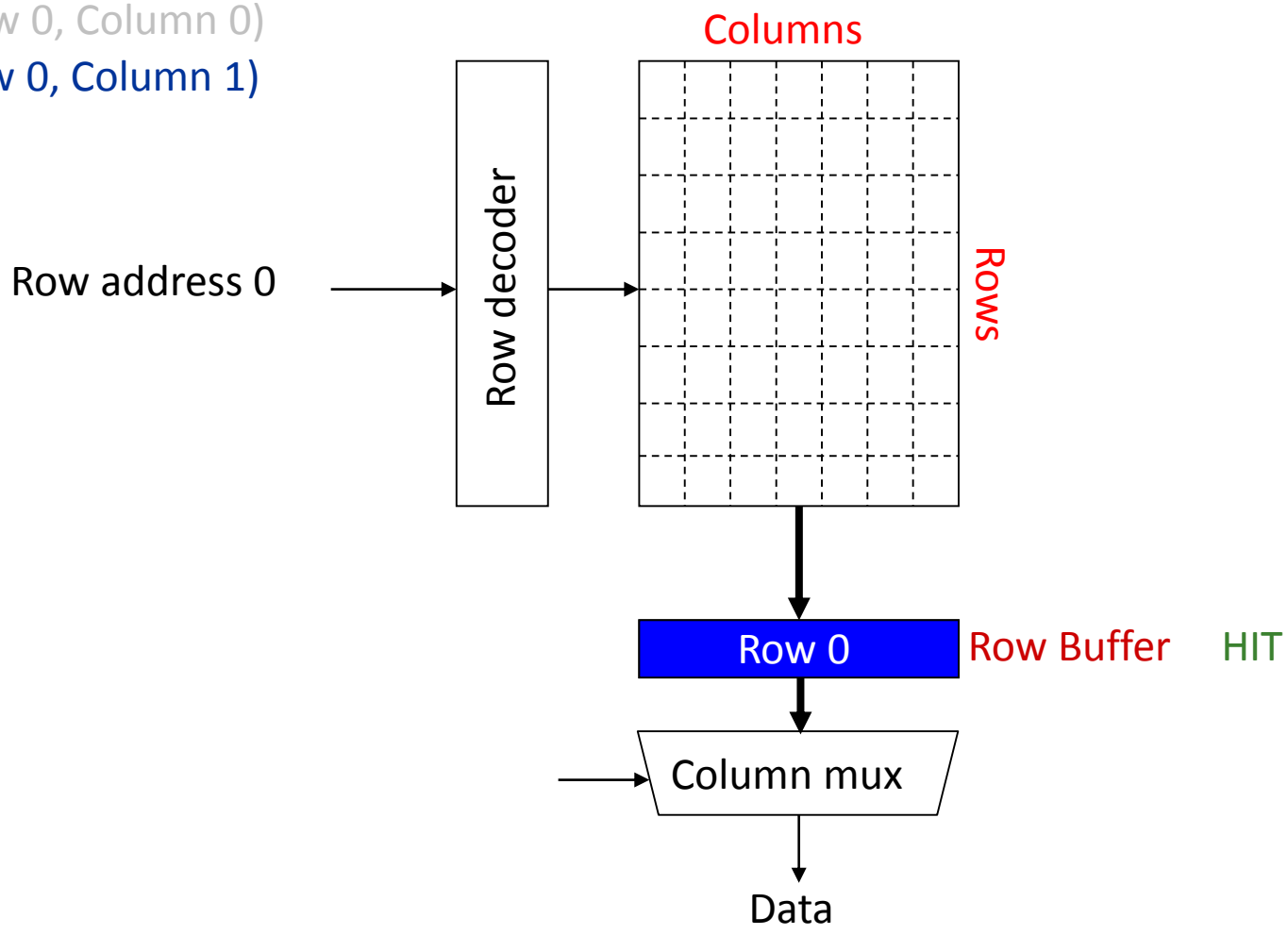
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)



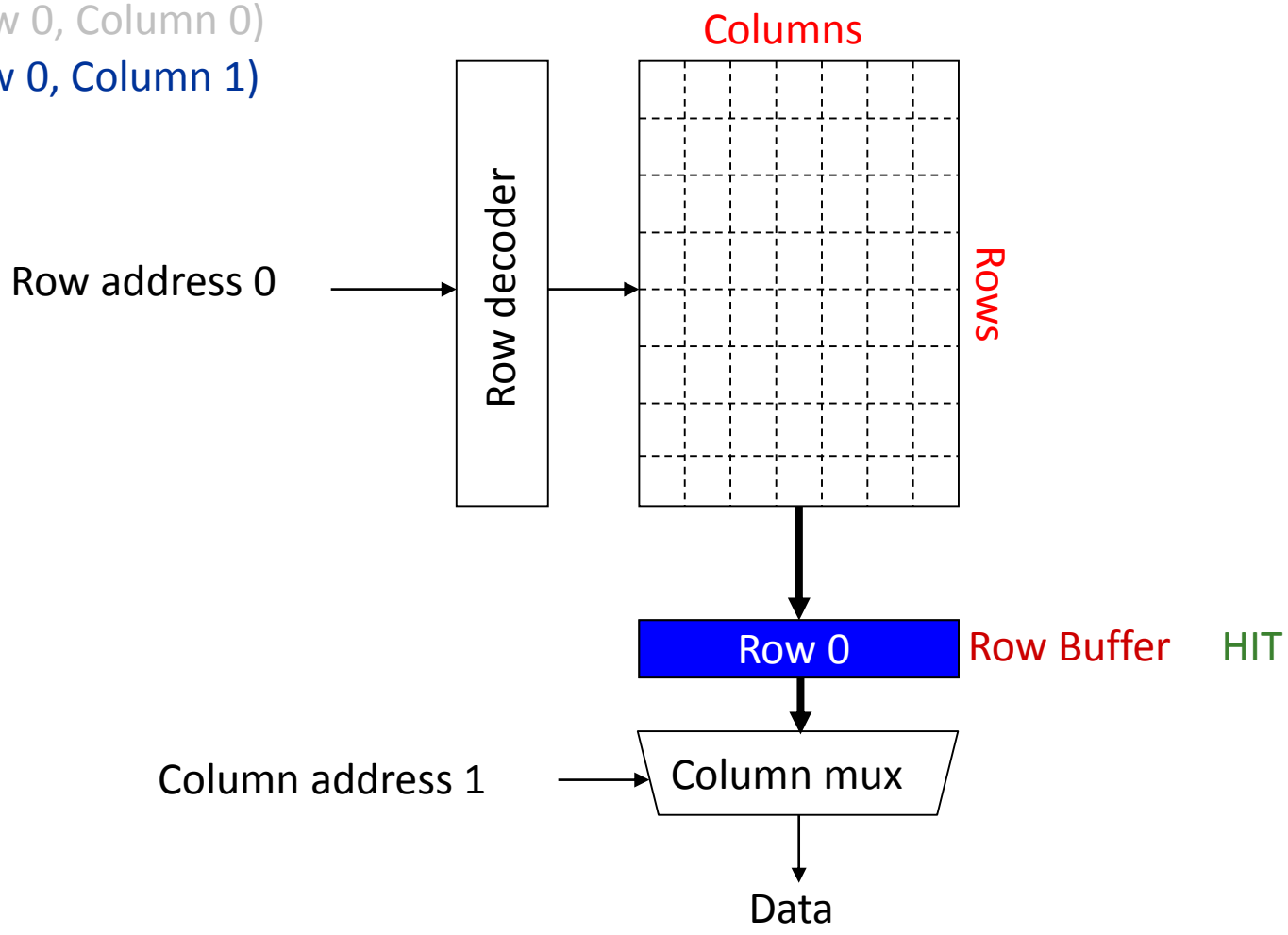
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)



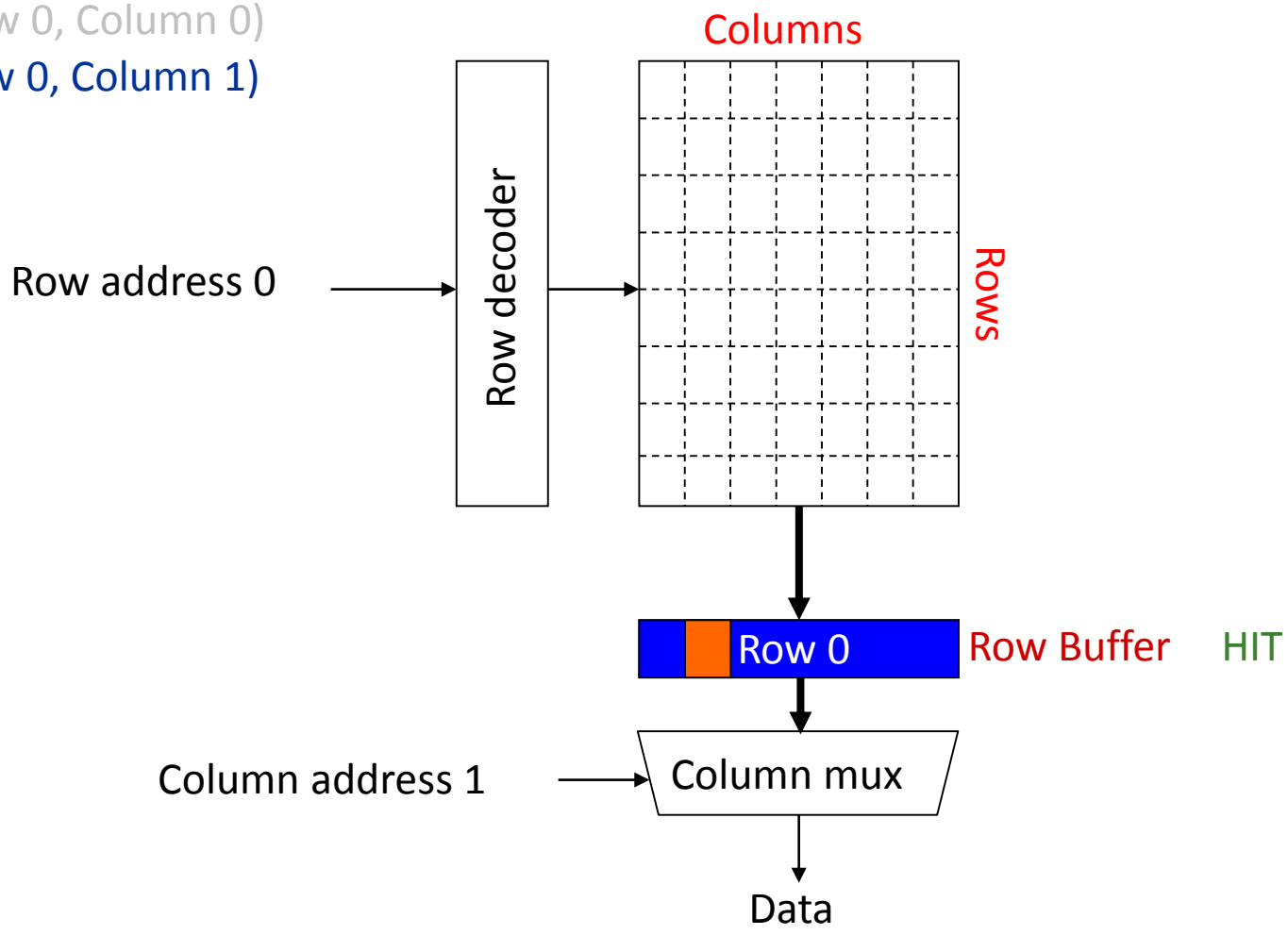
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)



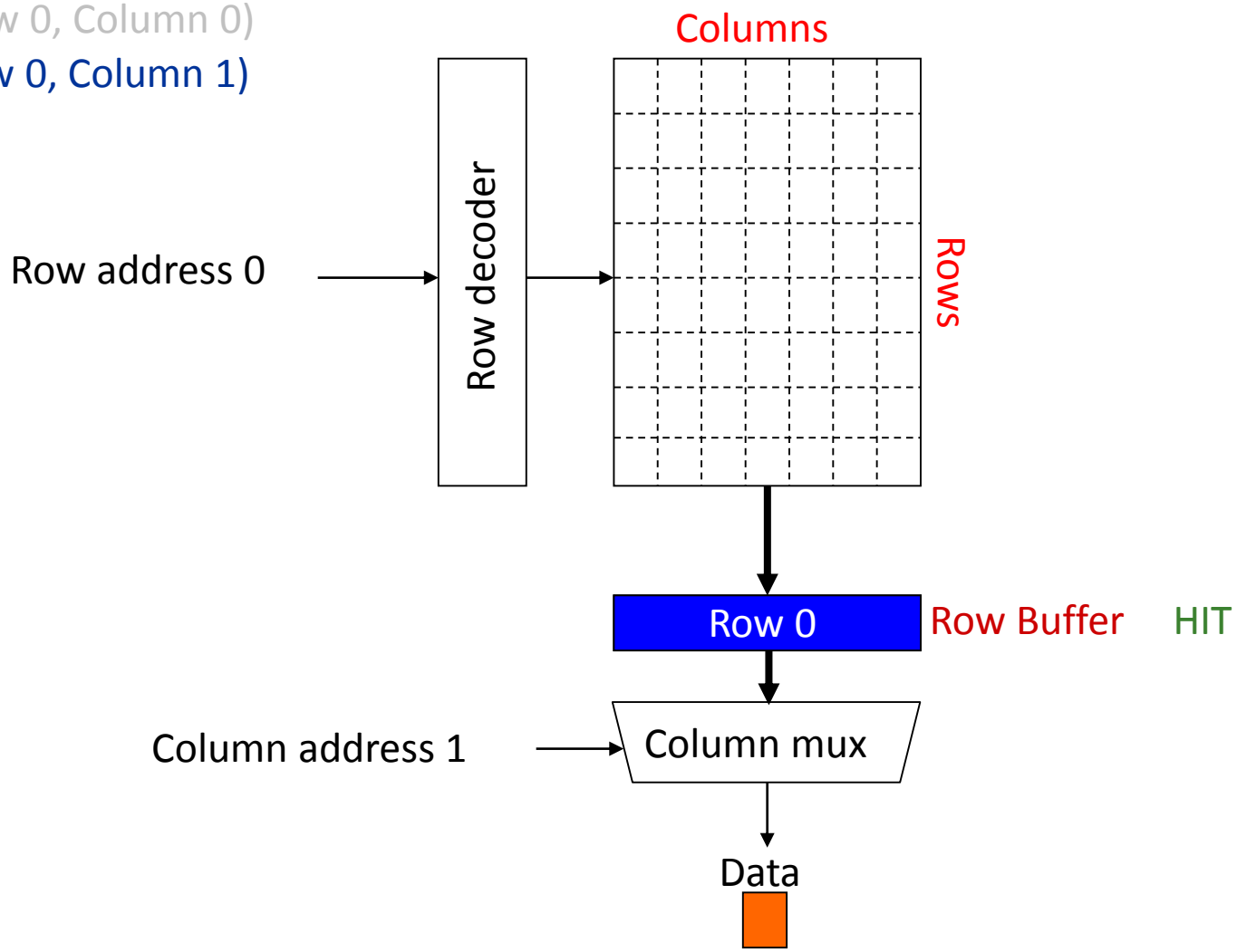
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)



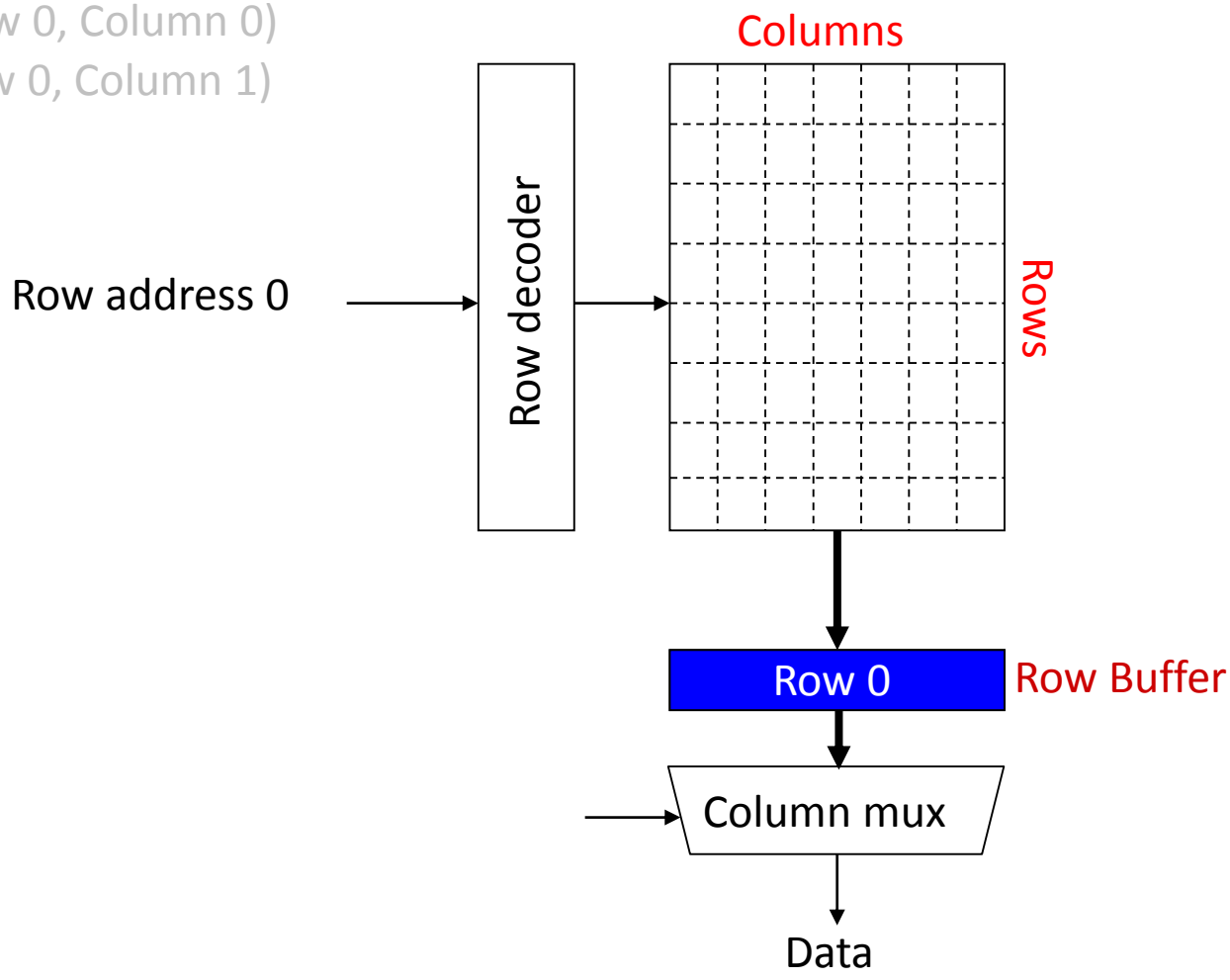
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)



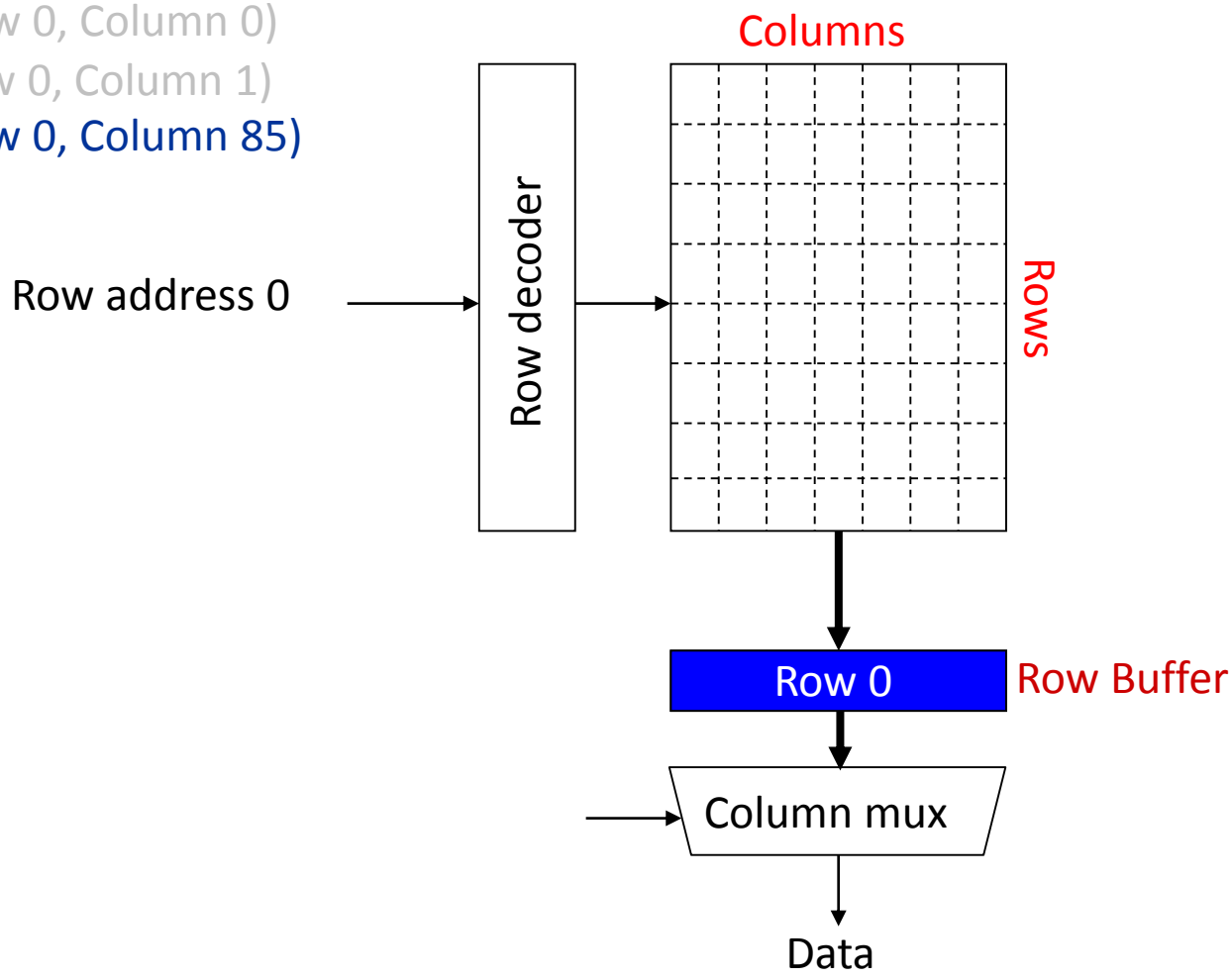
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)



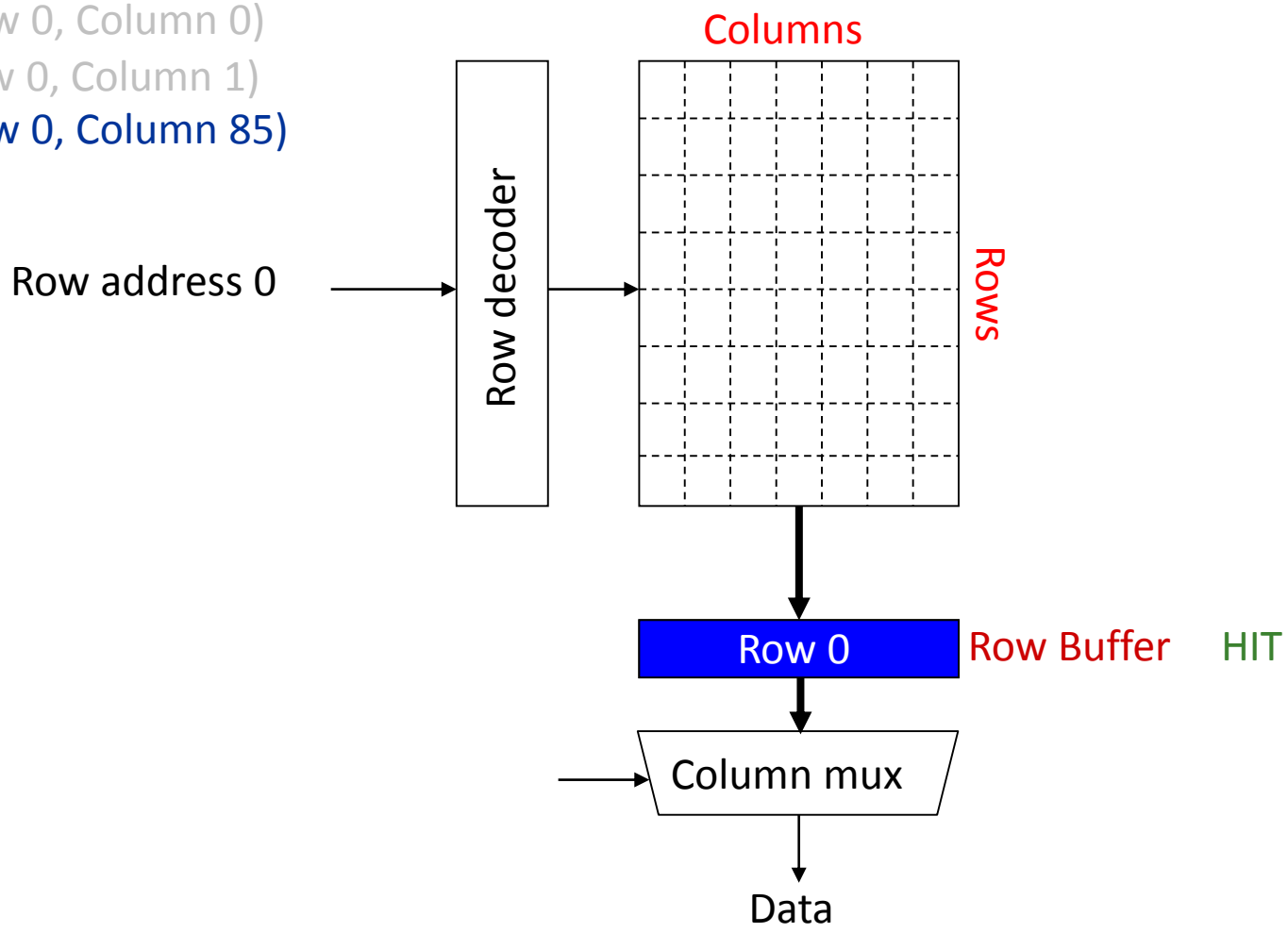
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)



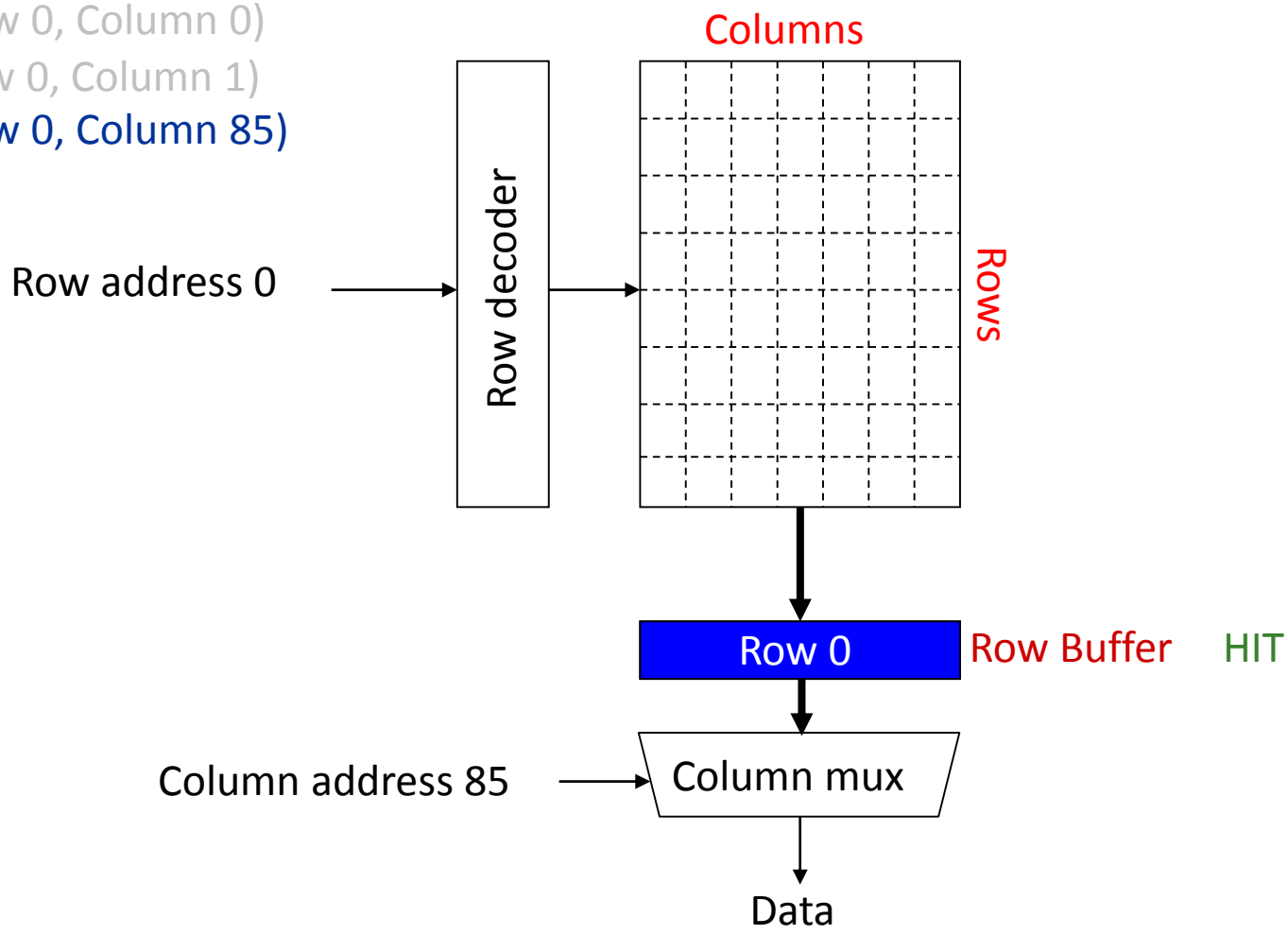
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)



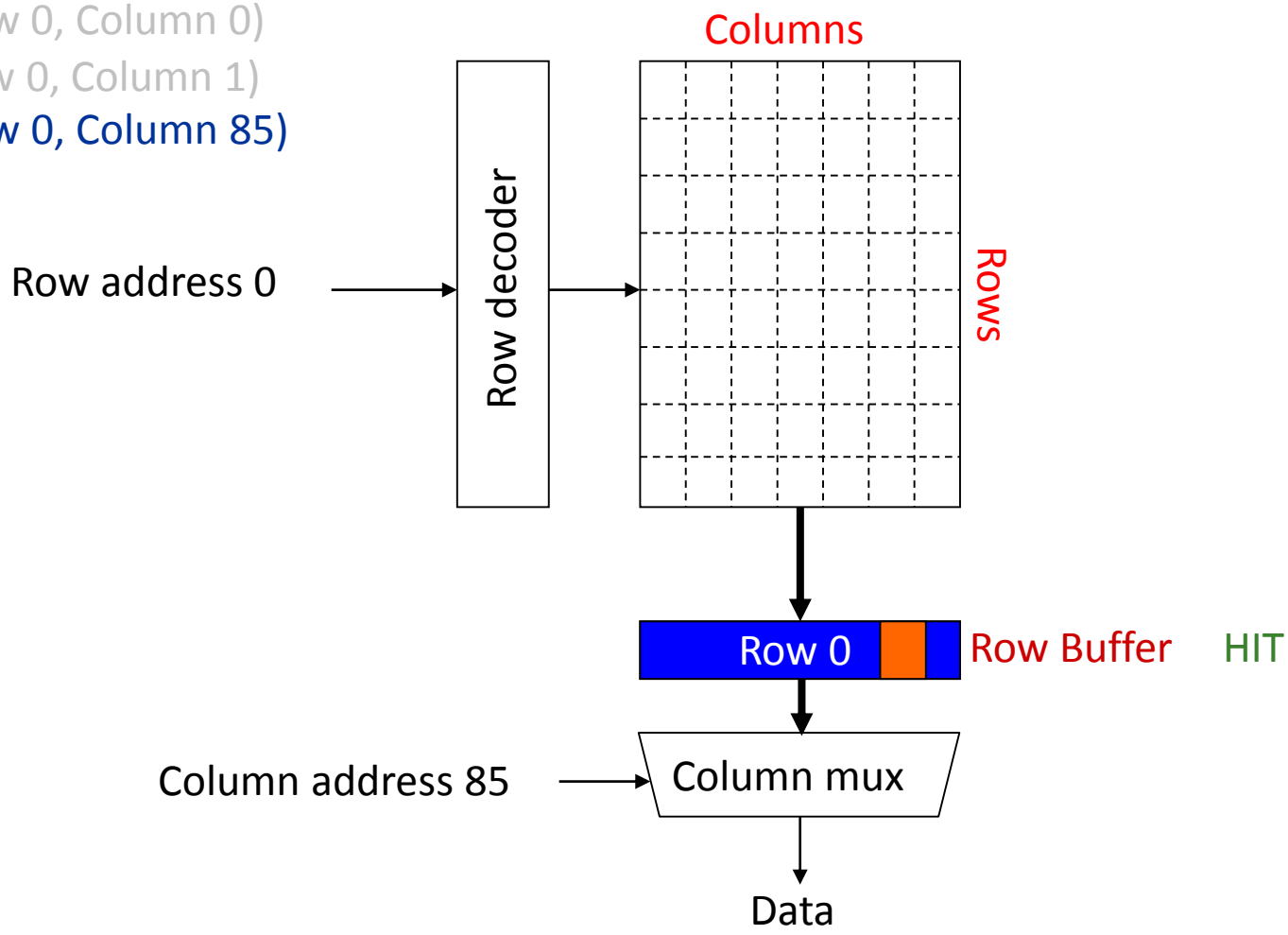
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)



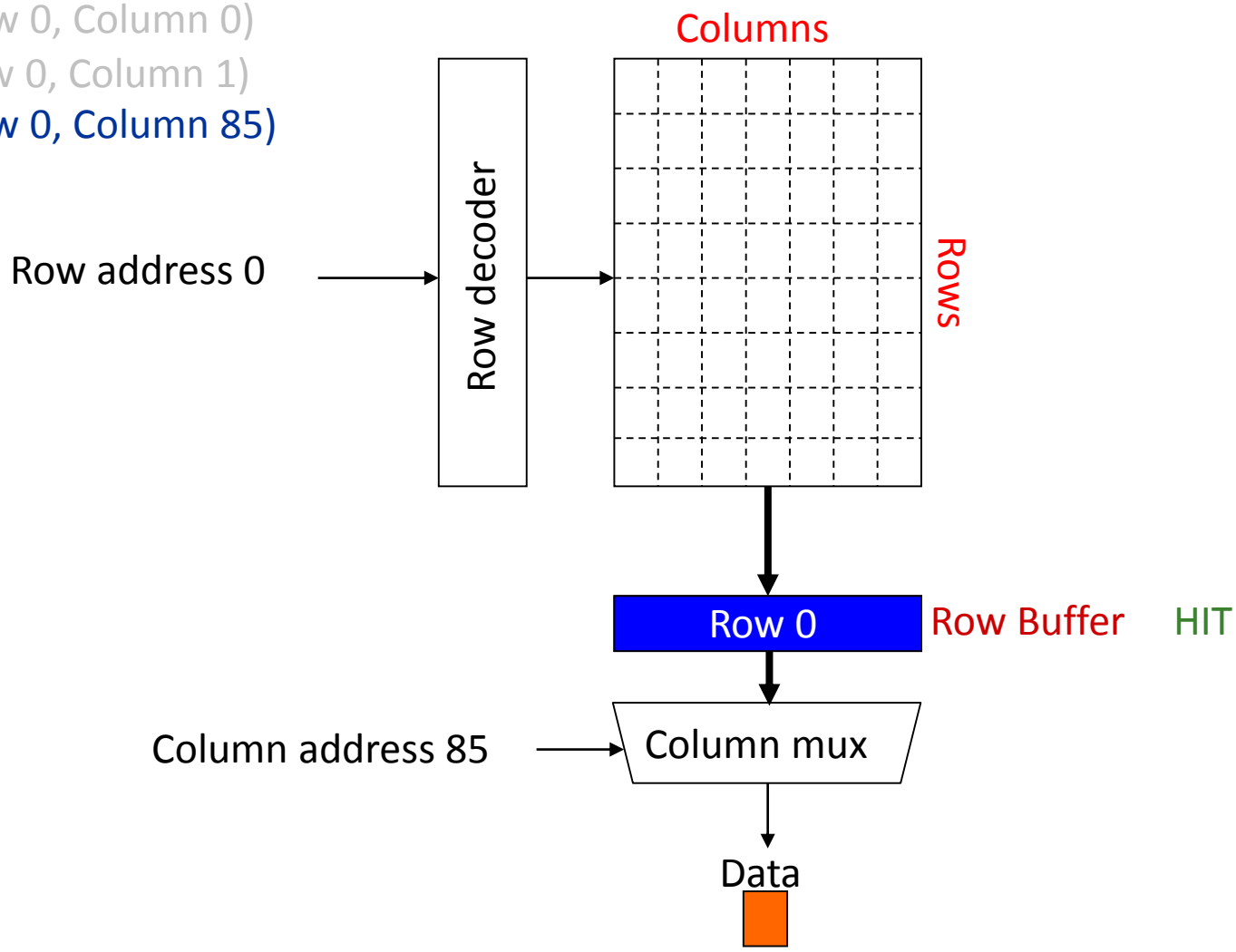
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)



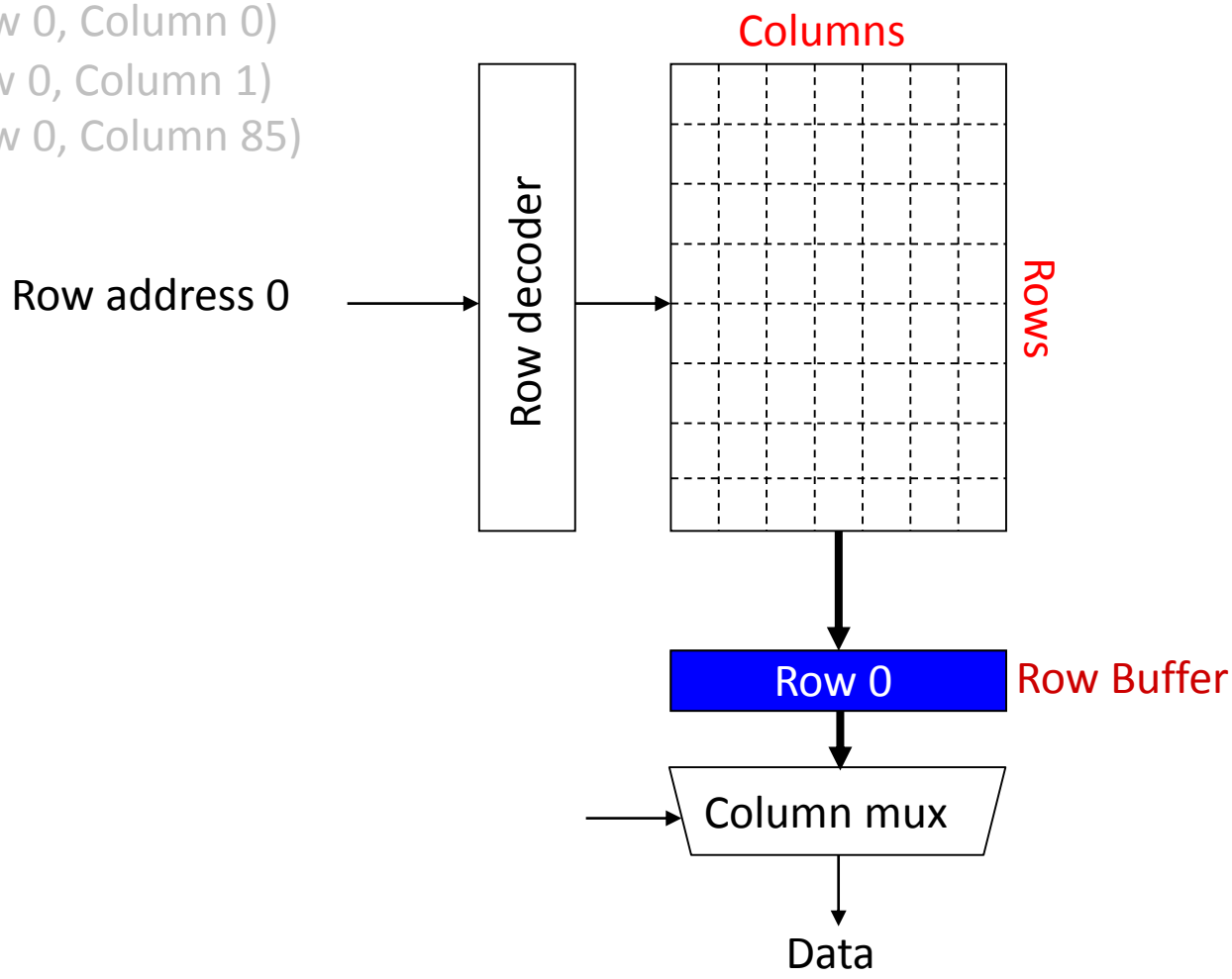
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)



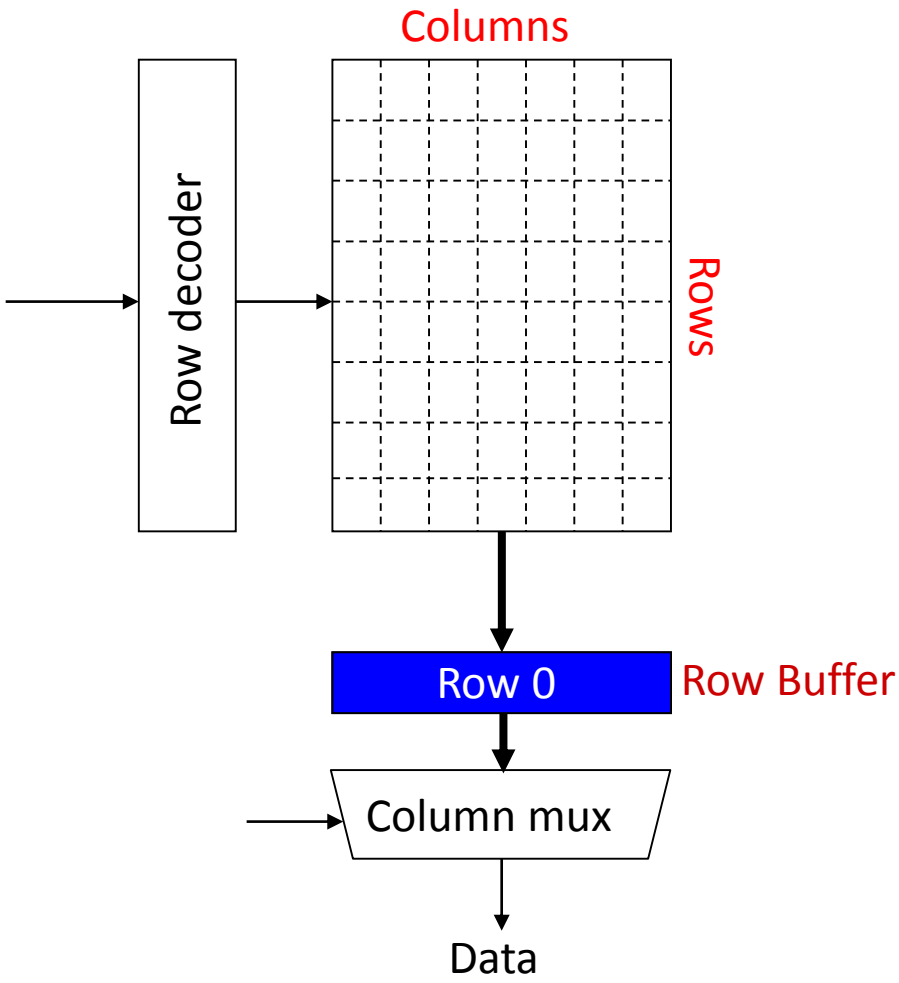
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)



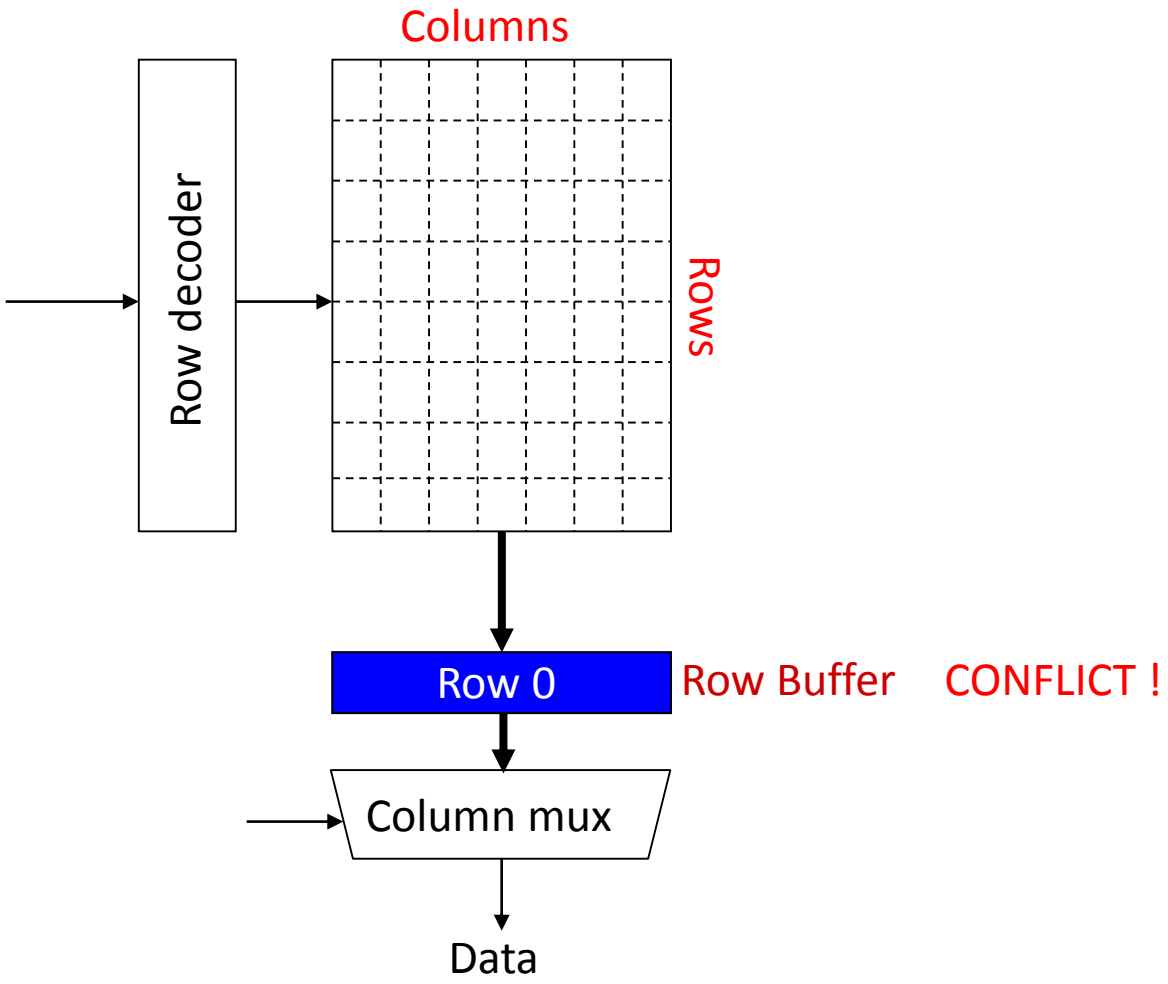
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



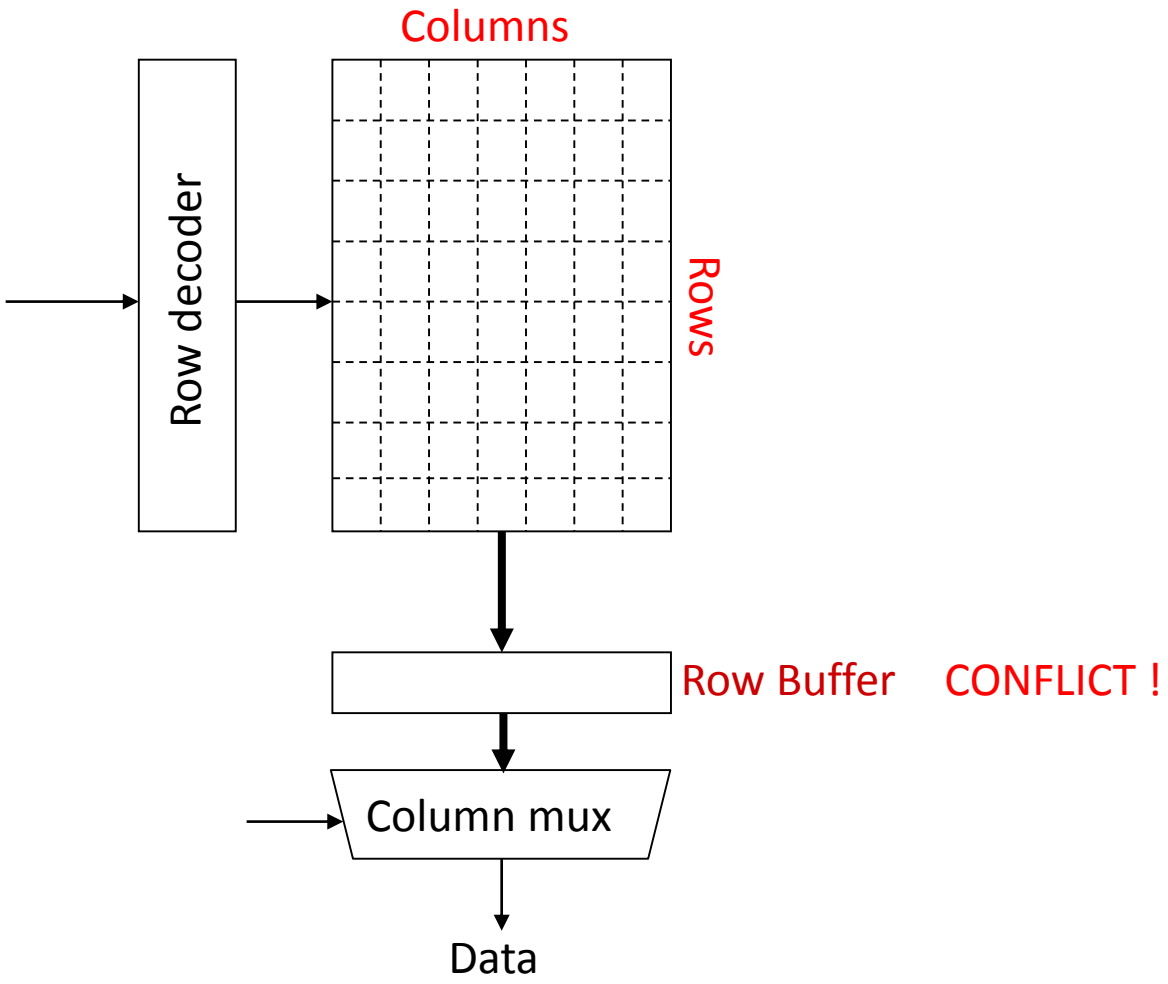
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



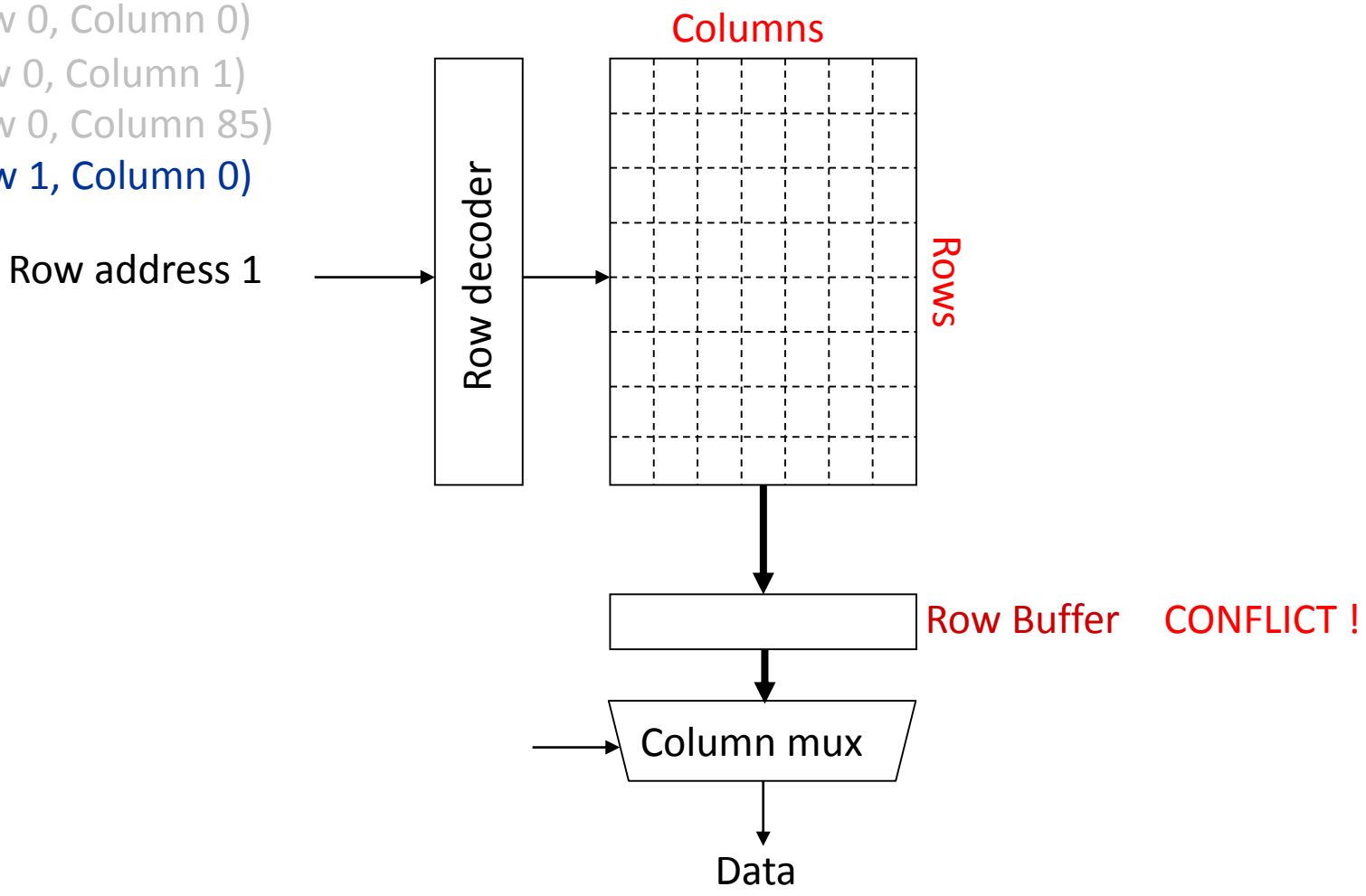
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



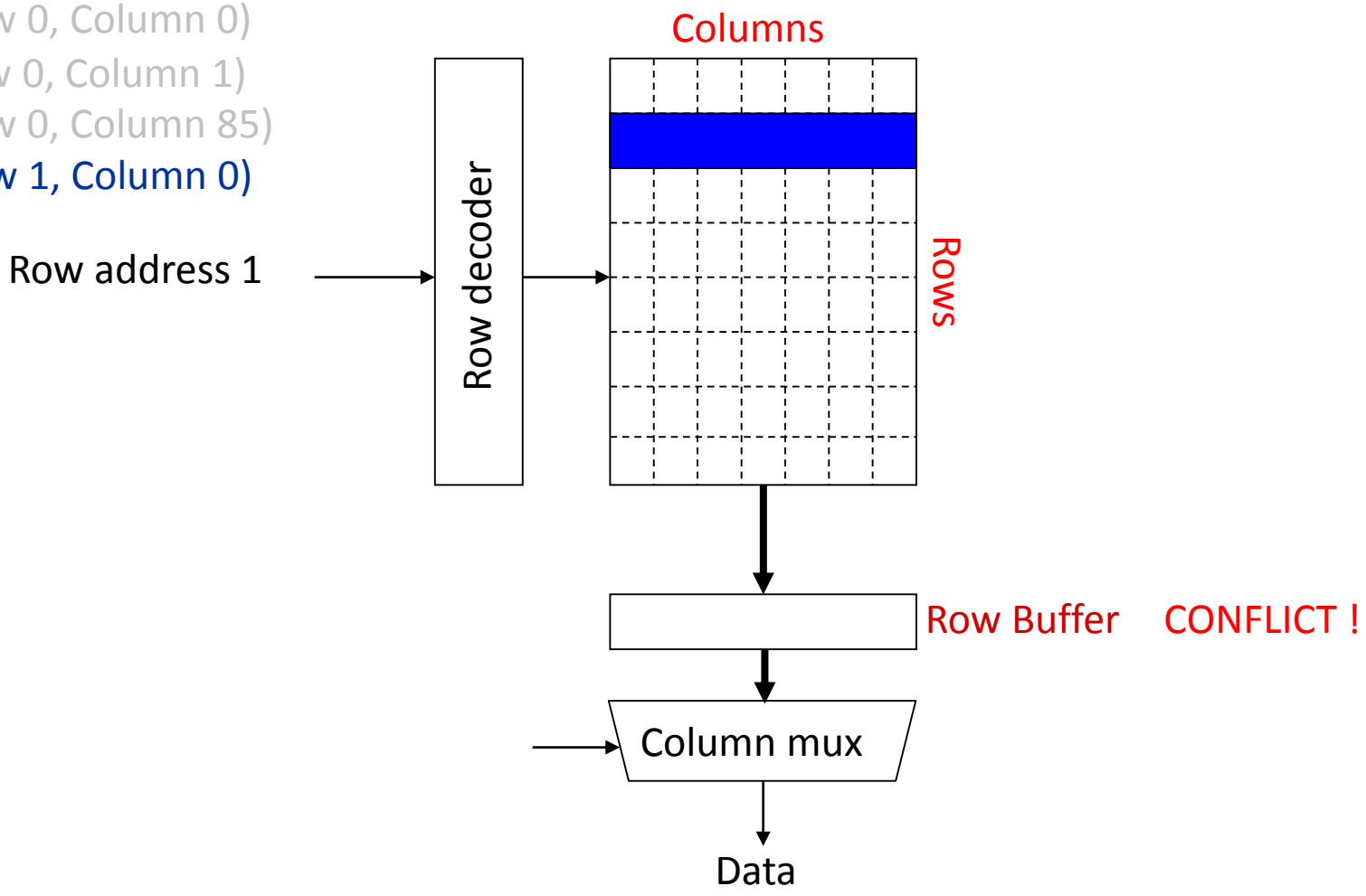
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



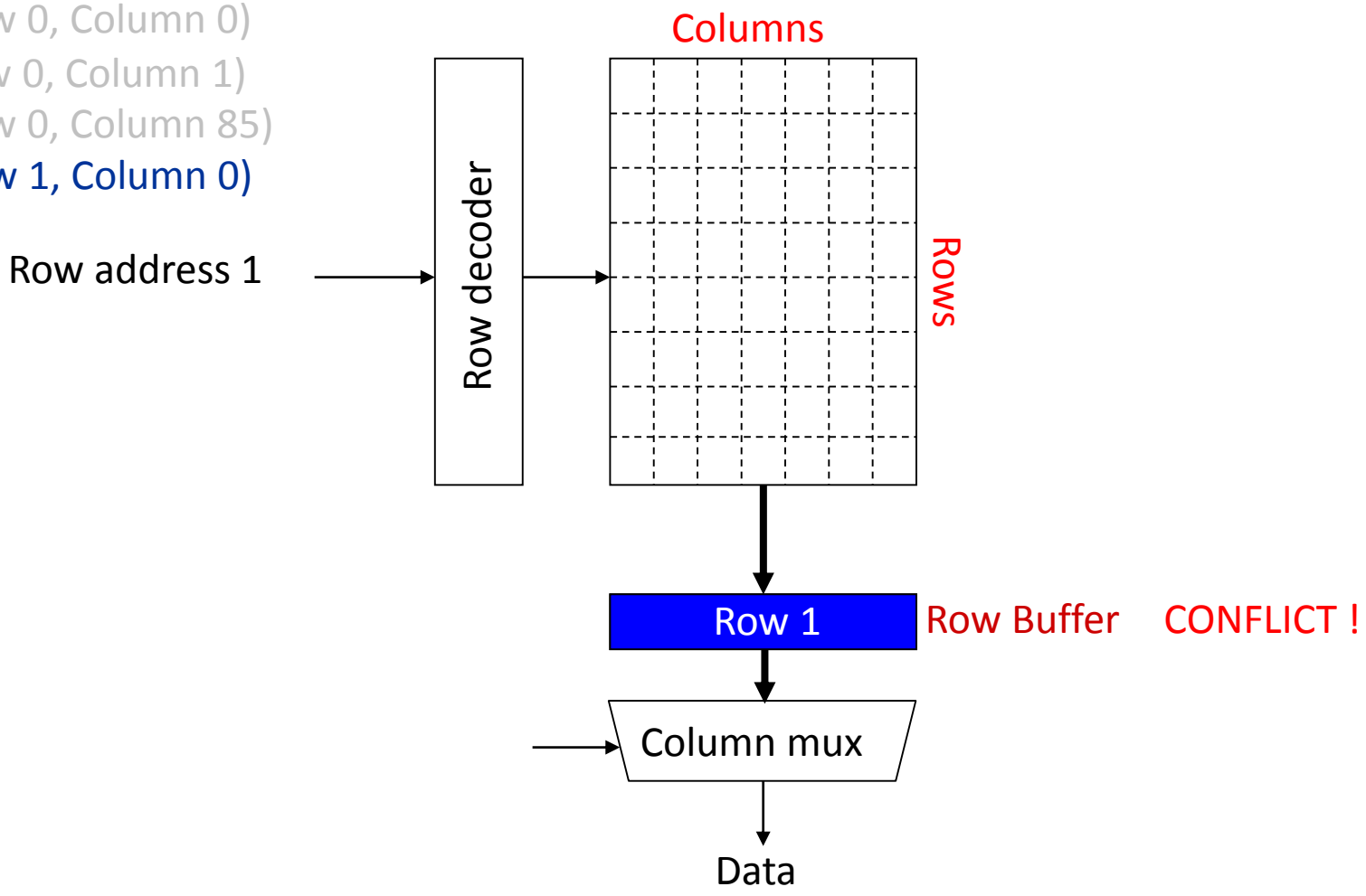
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



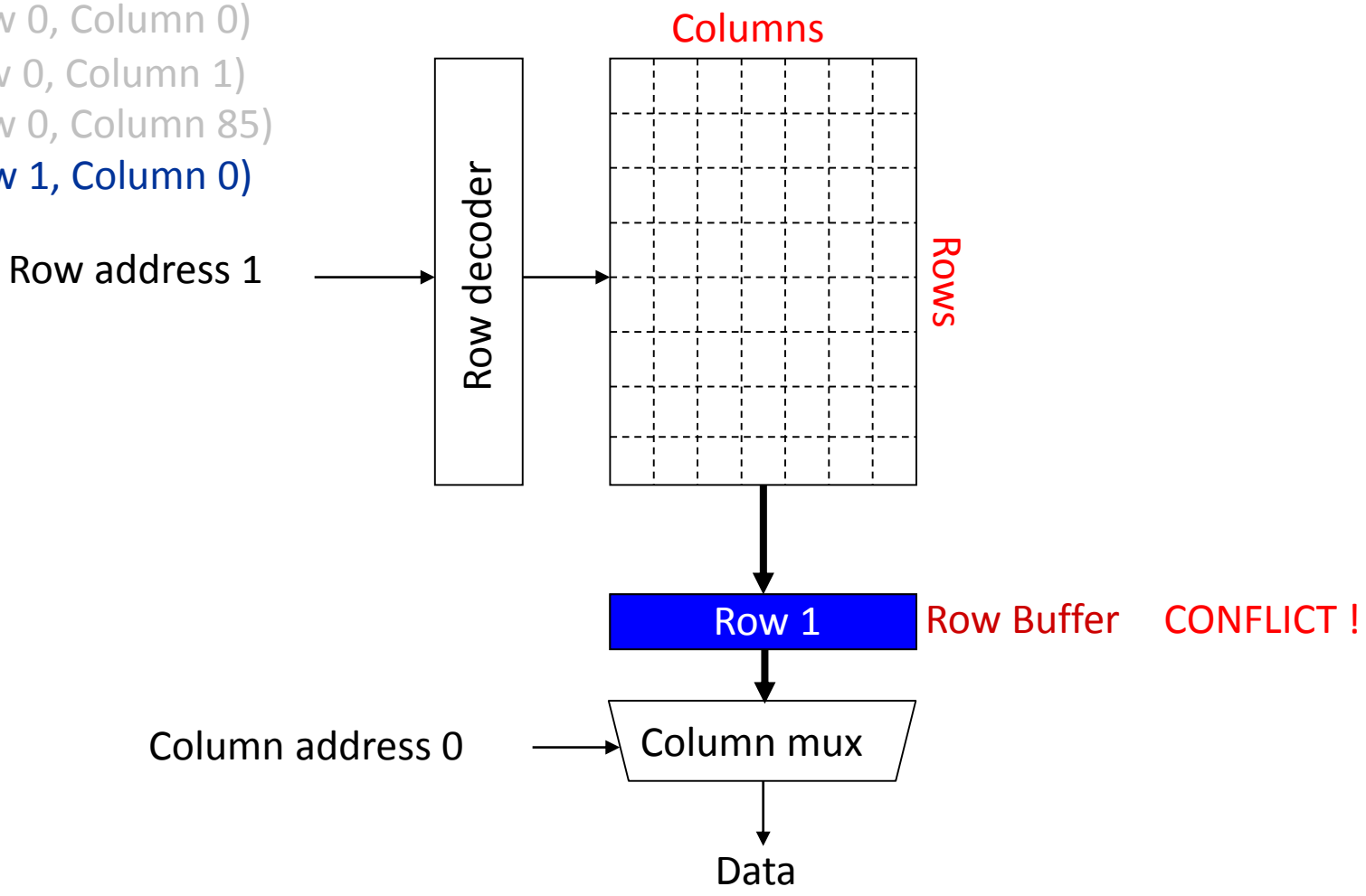
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



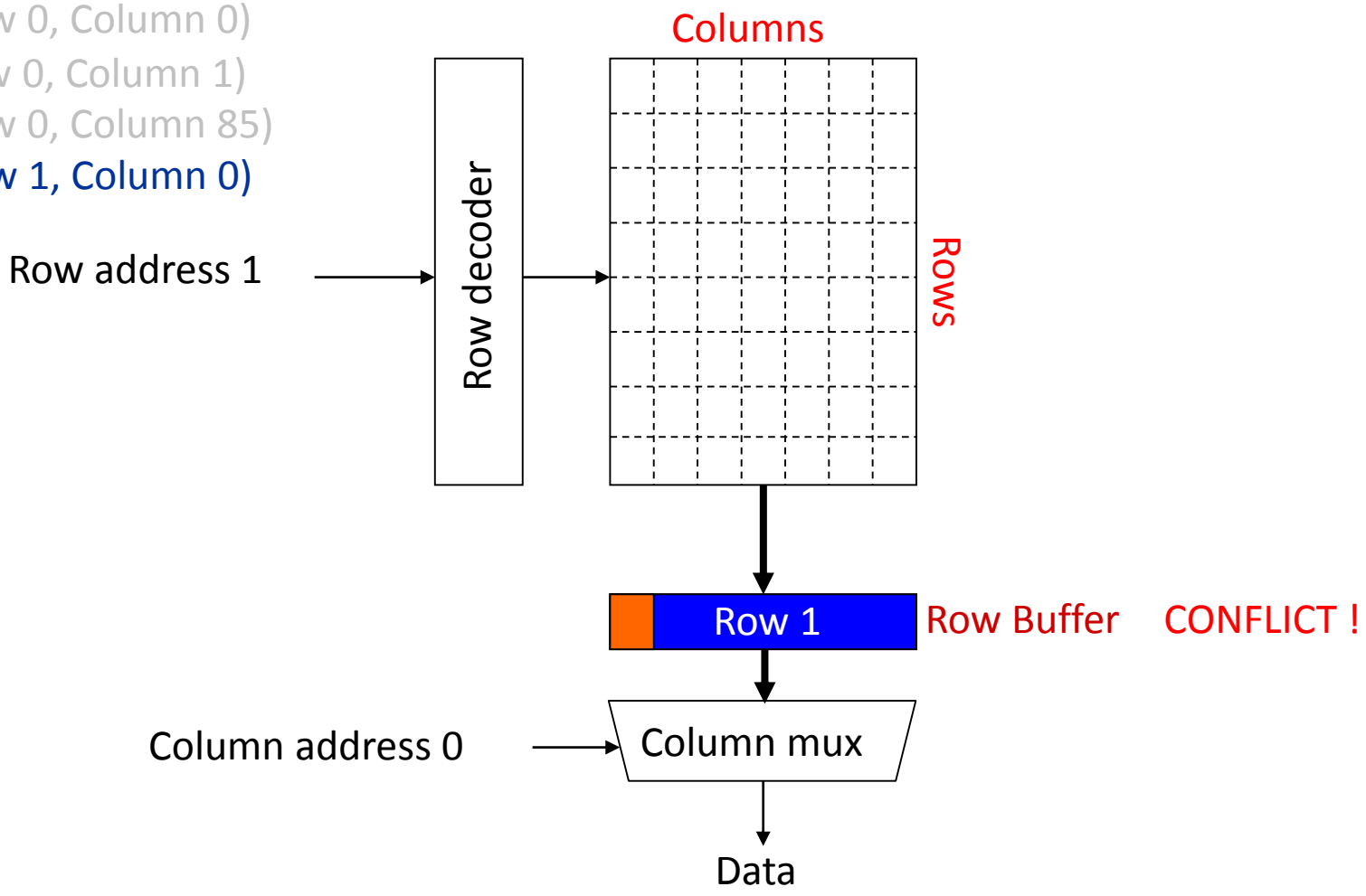
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



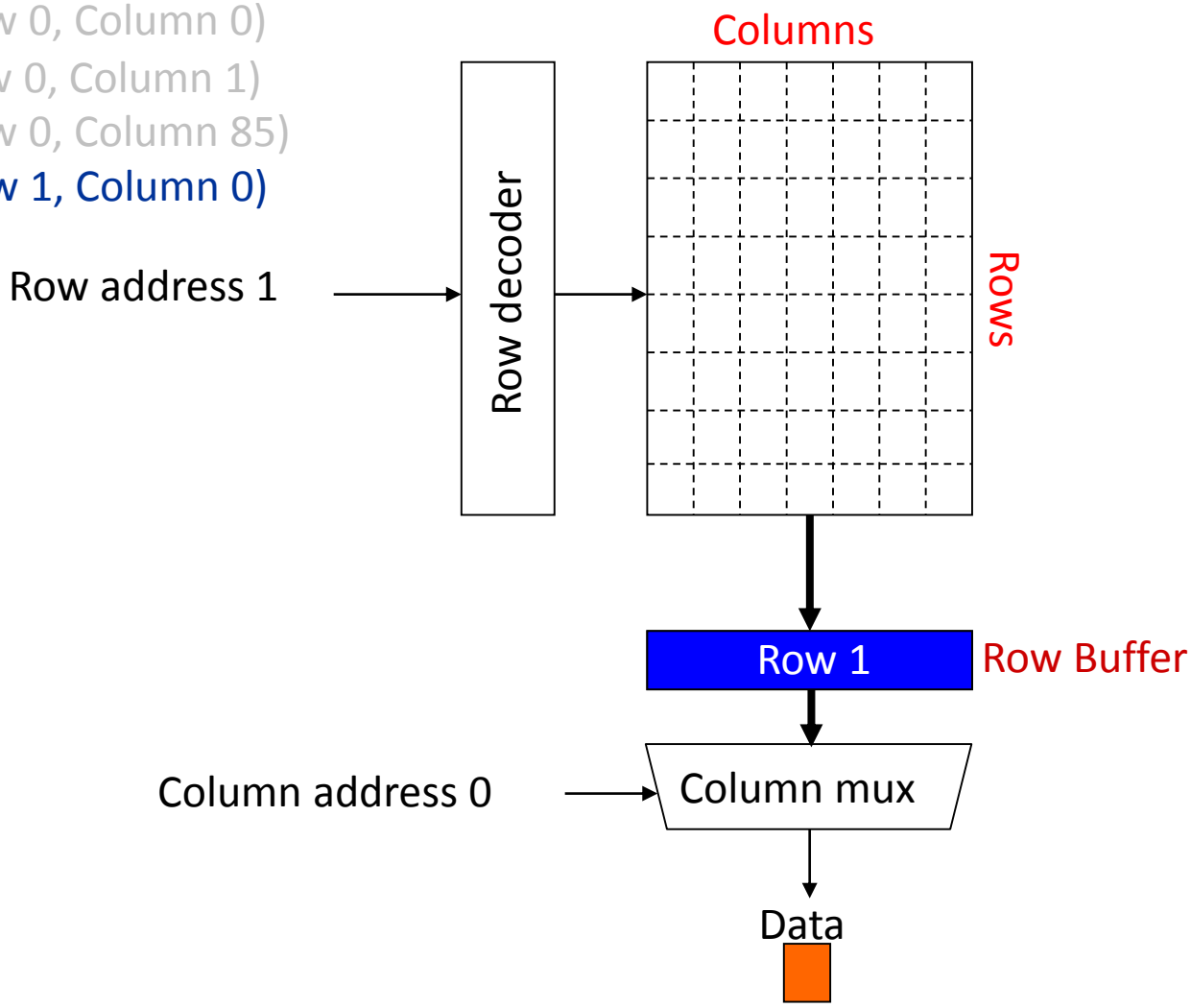
Row Hit / Row Conflict

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



Row Hit / Row Conflict

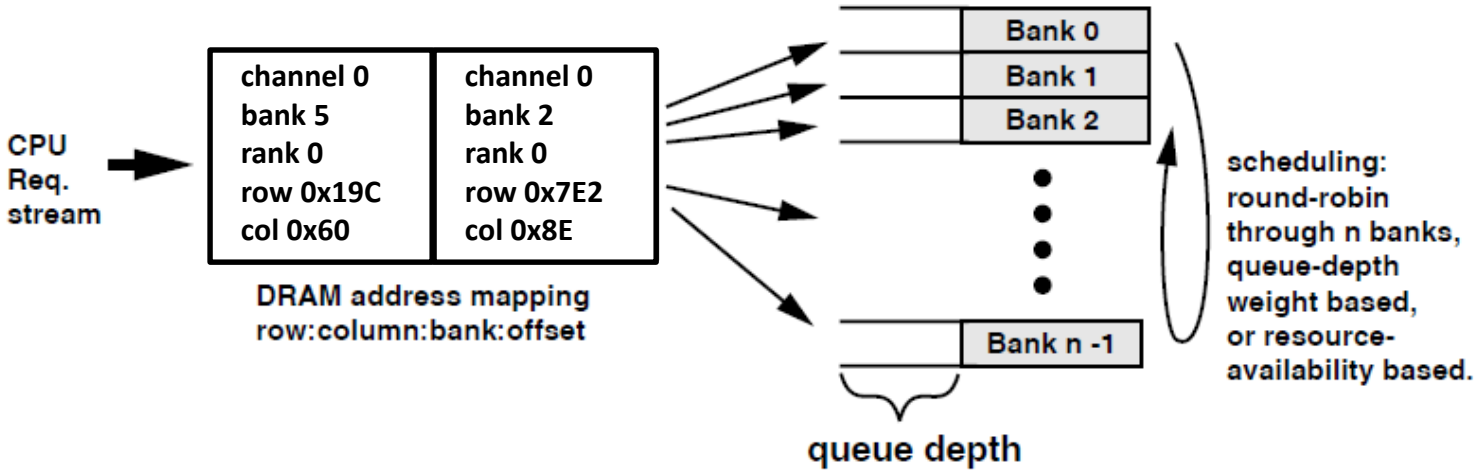
Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



Row Buffer Management Policies

Policy	First access	Next access	Commands needed for next access
Open row	Row 0	Row 0 (row hit)	Read
Open row	Row 0	Row 1 (row conflict)	Precharge + Activate Row 1 + Read
Closed row	Row 0	Row 0 – access in request buffer (row hit)	Read
Closed row	Row 0	Row 0 – access not in request buffer (row closed)	Activate Row 0 + Read + Precharge
Closed row	Row 0	Row 1 (row closed)	Activate Row 1 + Read + Precharge

Αντιστοίχιση (mapping) διευθύνσεων



Πολιτικές χρονοδρομολόγησης DRAM

- Πολιτική χρονοδρομολόγησης == μηχανισμός απόδοσης προτεραιοτήτων σε διαφορετικά requests
- Η απόδοση προτεραιότητας μπορεί να βασίζεται
  - Στην “ηλικία” του request
  - Στον χρόνο που απαιτείται για να εξυπηρετηθεί
  - Στον τύπο του
  - Στην κρισιμότητα του

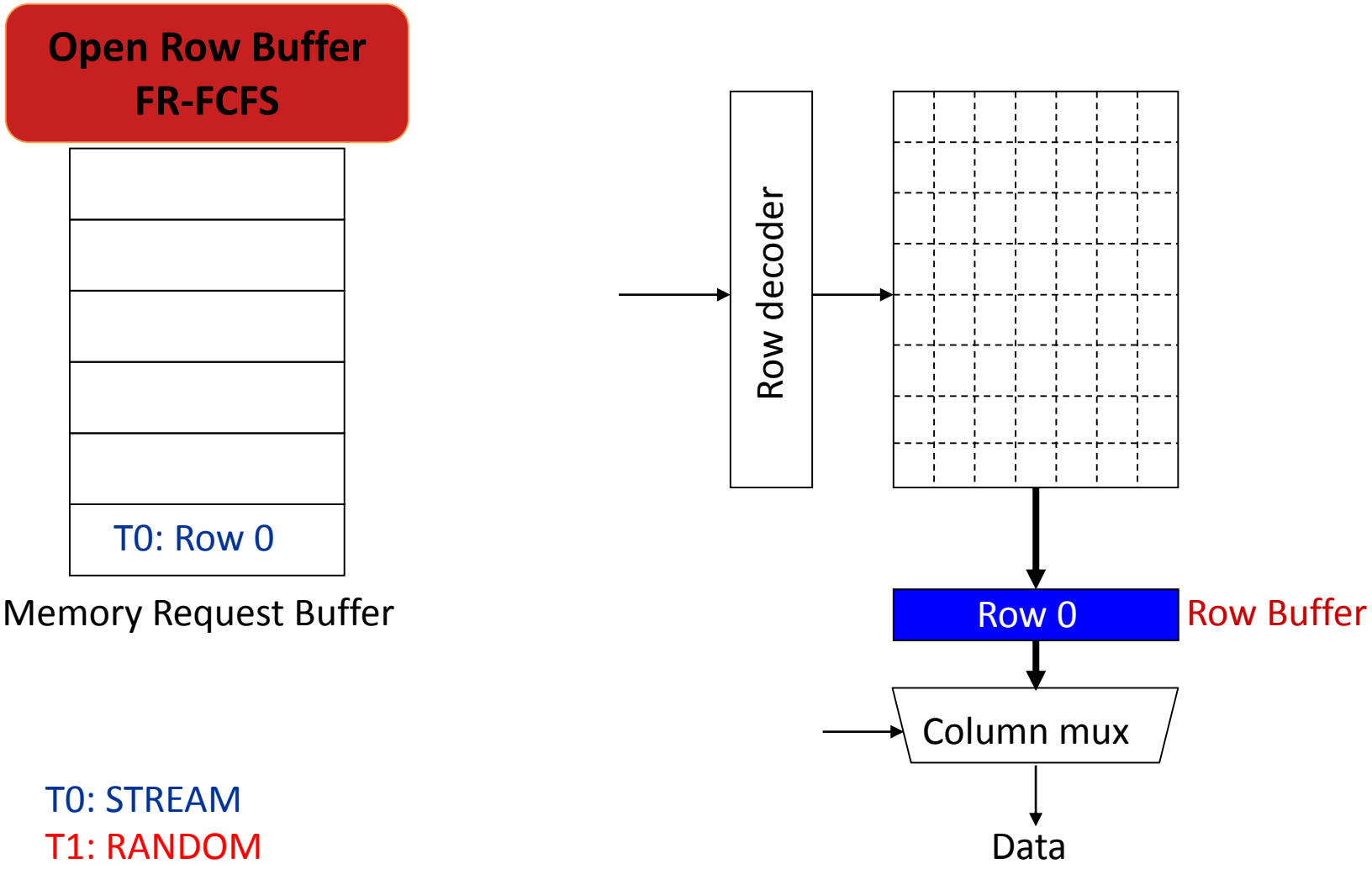
Πολιτικές χρονοδρομολόγησης DRAM

- FCFS (first come first served)
  - Η “γηραιότερη” αίτηση πρώτη
- FR-FCFS (first ready, first come first served)

  1. Αιτήσεις που οδηγούν σε Row-Hit πρώτες
  2. Η “γηραιότερη” αίτηση πρώτη

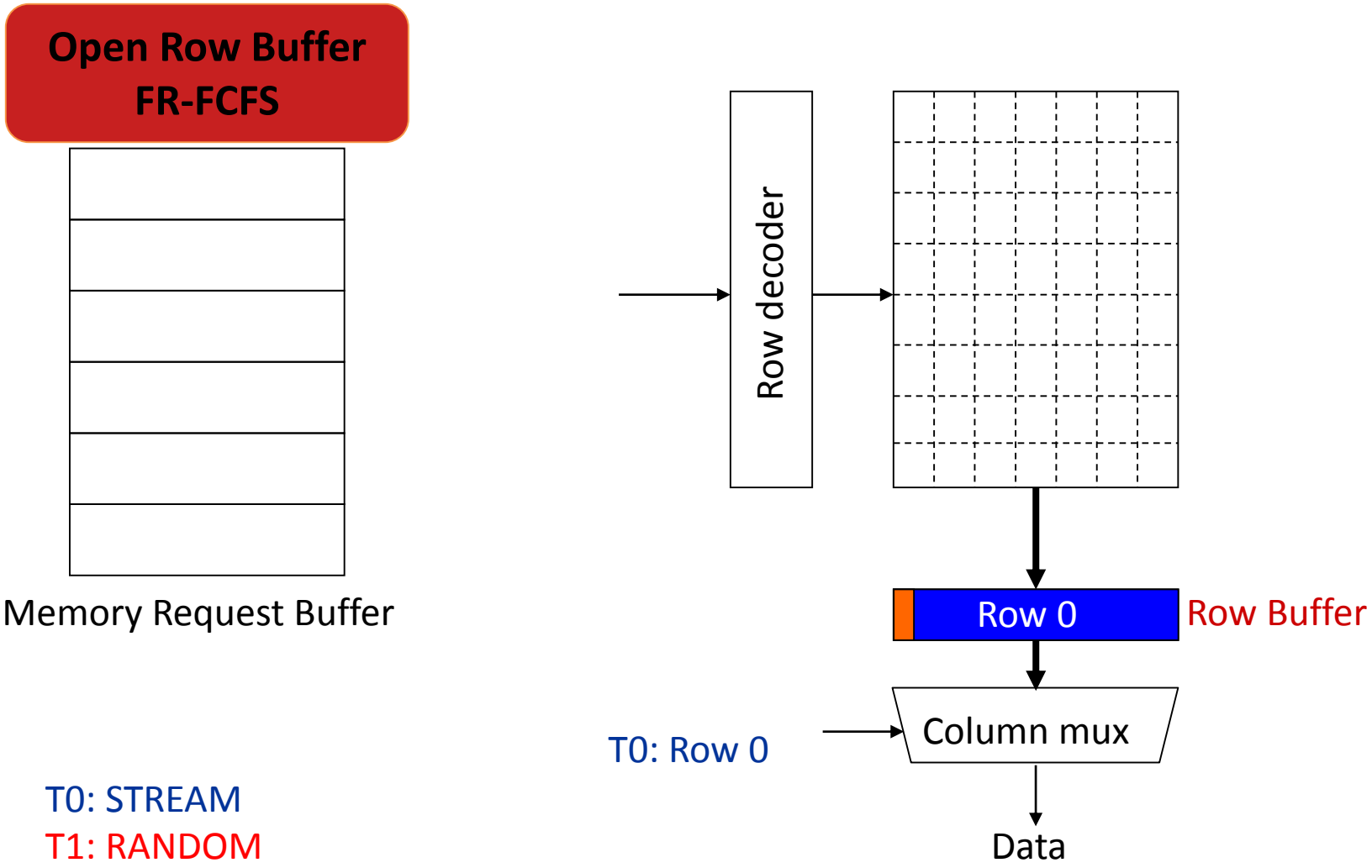
  Στόχος: Μεγιστοποίηση του row hit rate → μεγιστοποίηση του DRAM throughput
- Η χρονοδρομολόγηση γίνεται στο επίπεδο εντολών
  - Εντολές στήλης (read/write) παίρνουν προτεραιότητα από τις εντολές γραμμής (activate/precharge)
  - Σε κάθε group οι εντολές παίρνουν προτεραιότητα με βάση το χρόνο αναμονής στην ουρά (γηραιότητα).

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



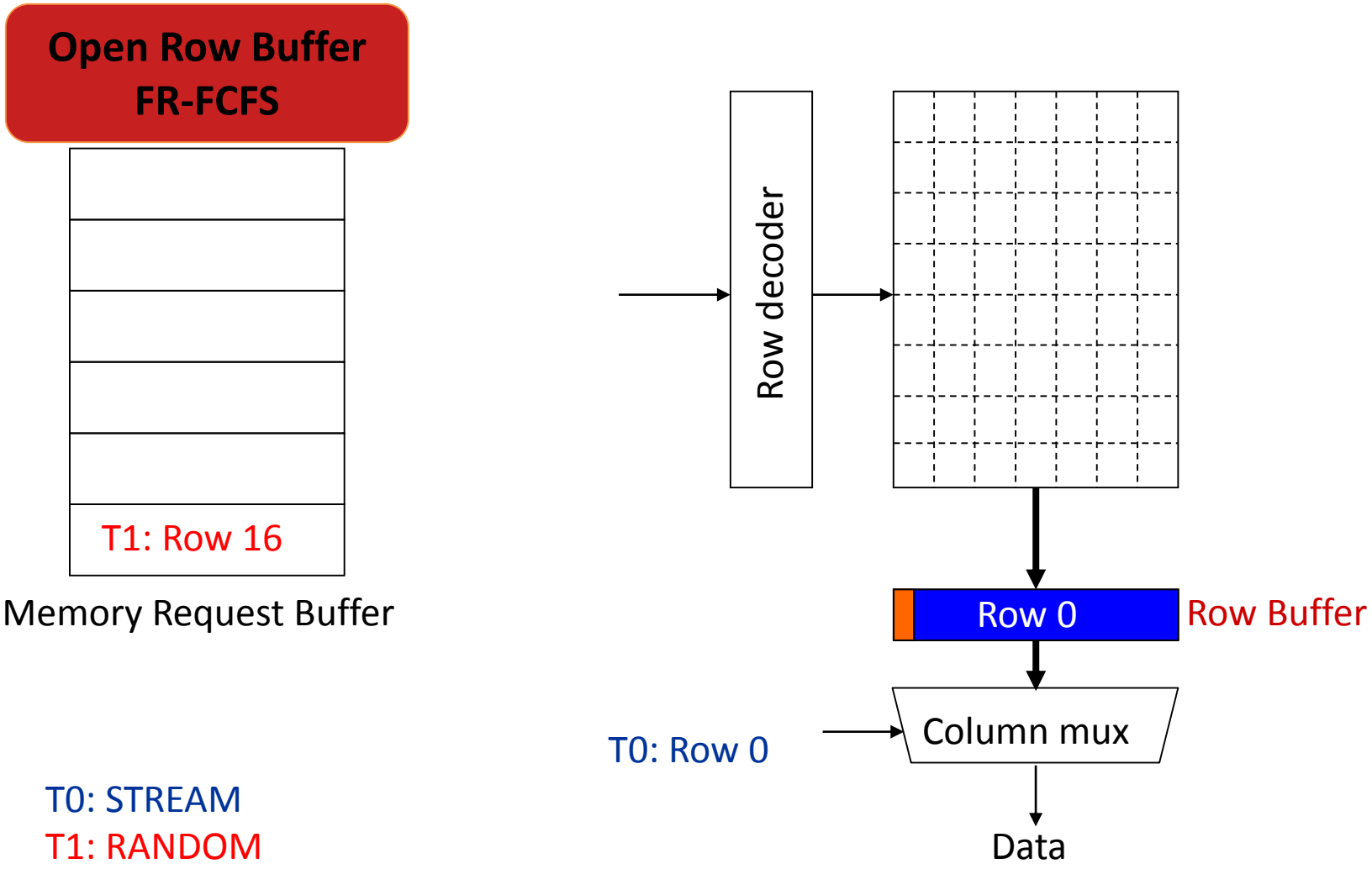
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



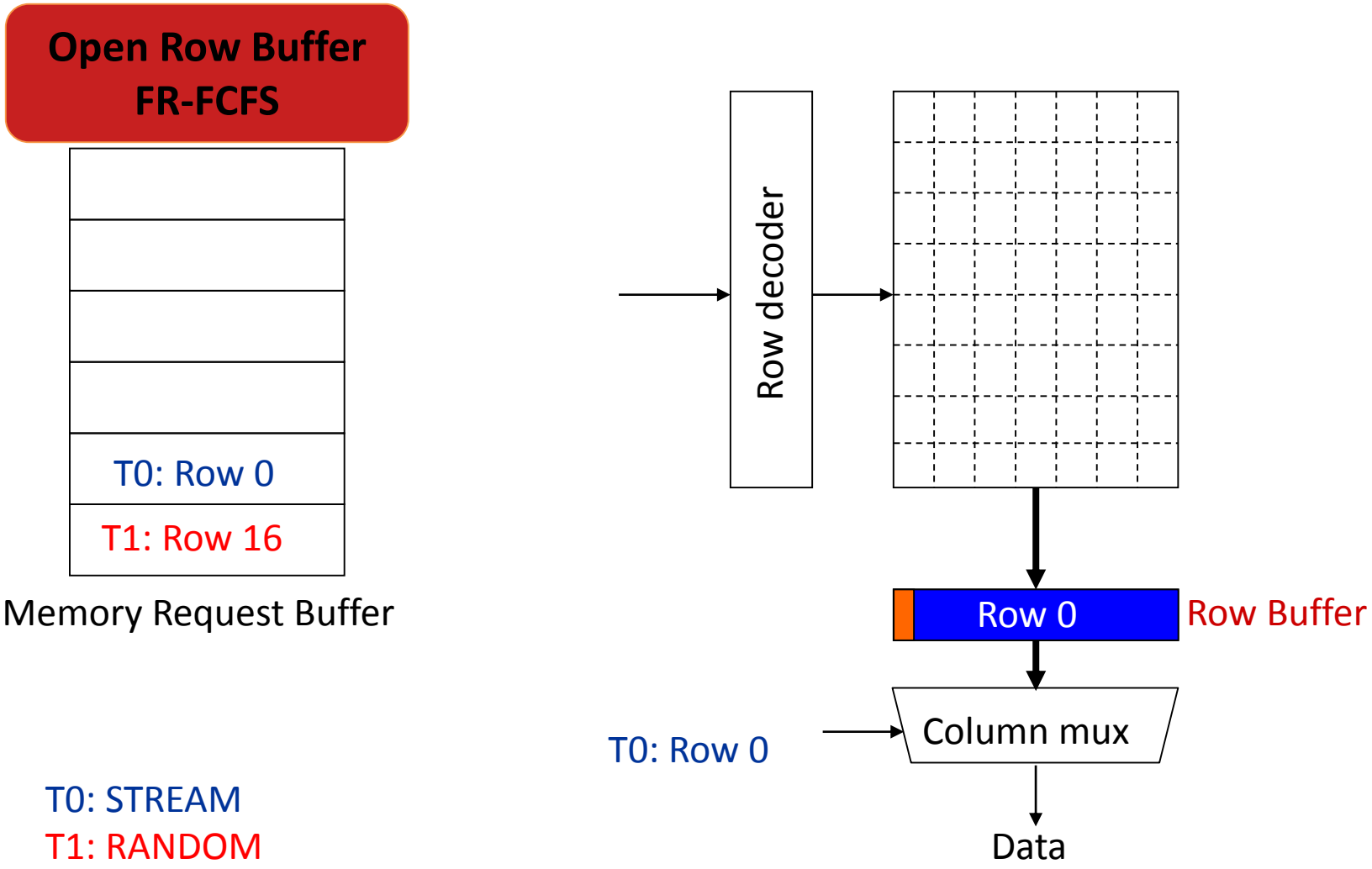
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



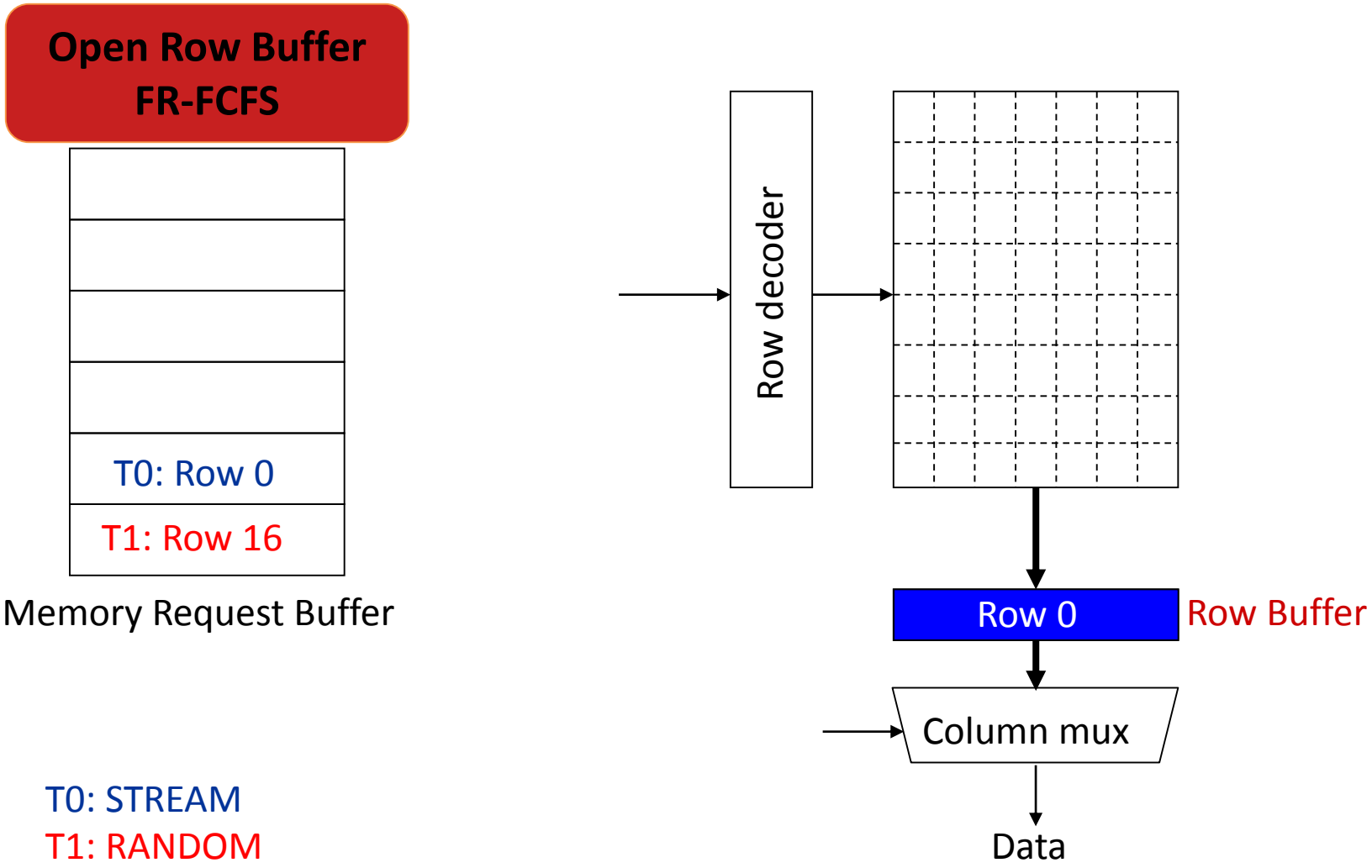
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



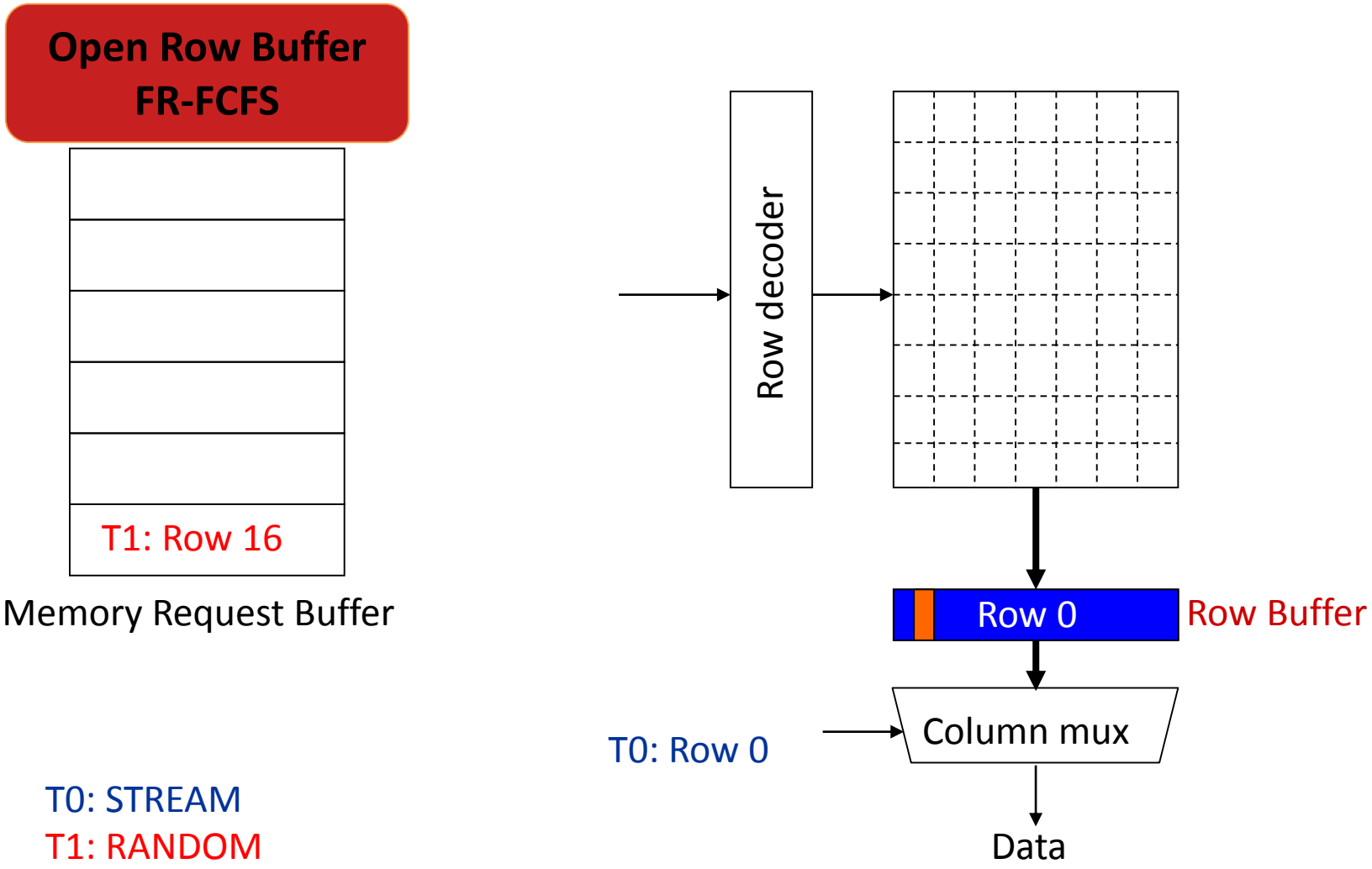
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



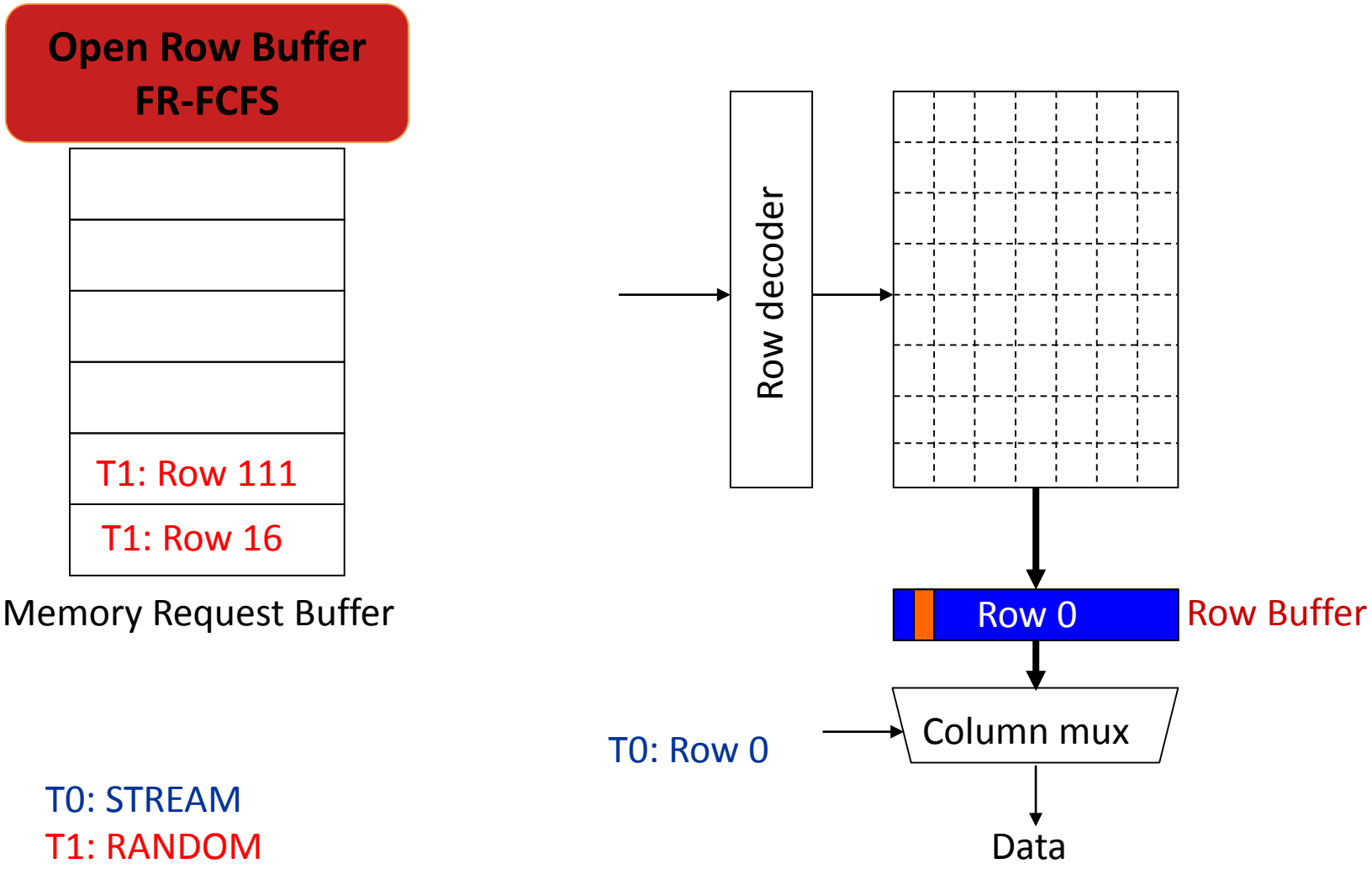
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



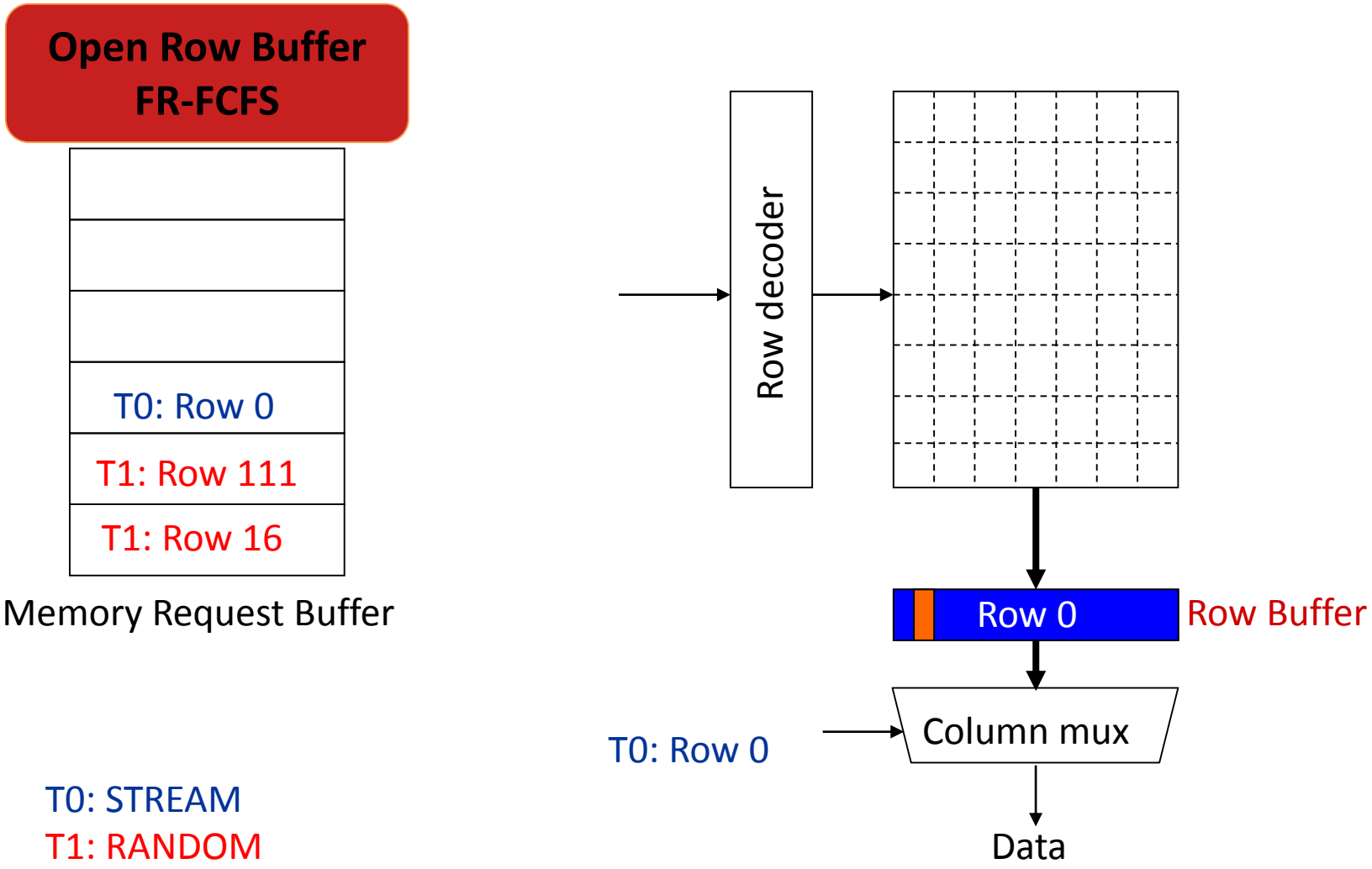
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



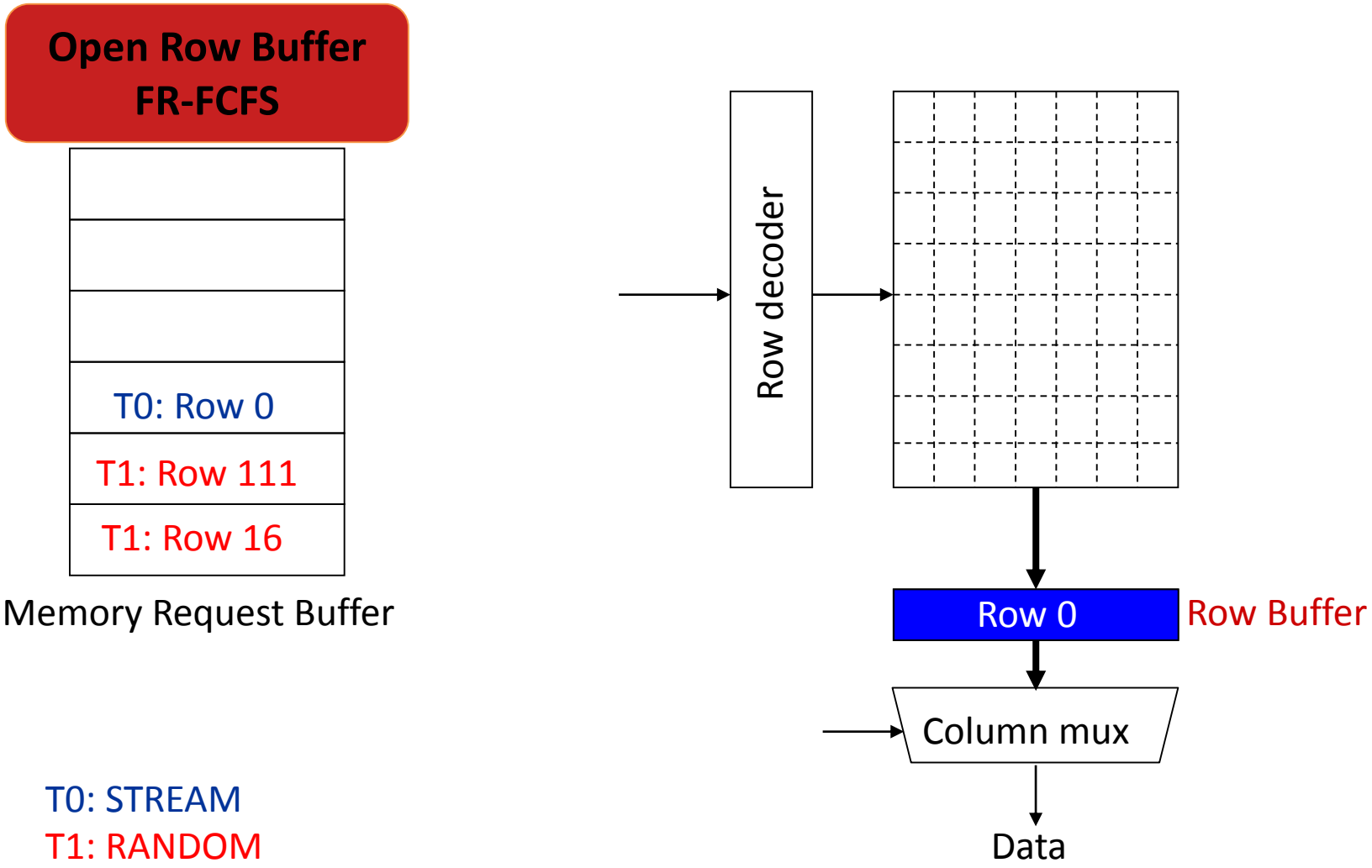
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



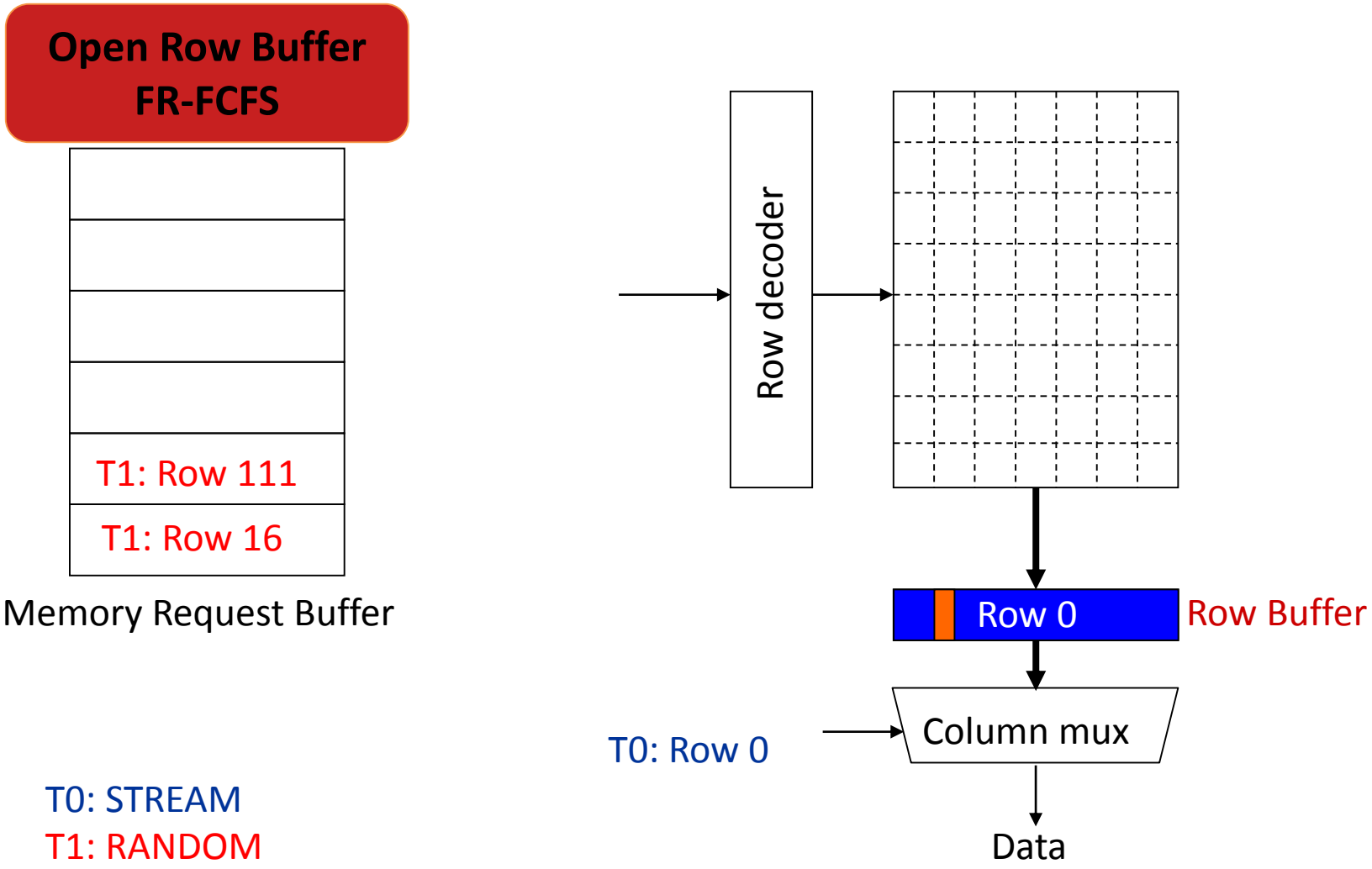
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



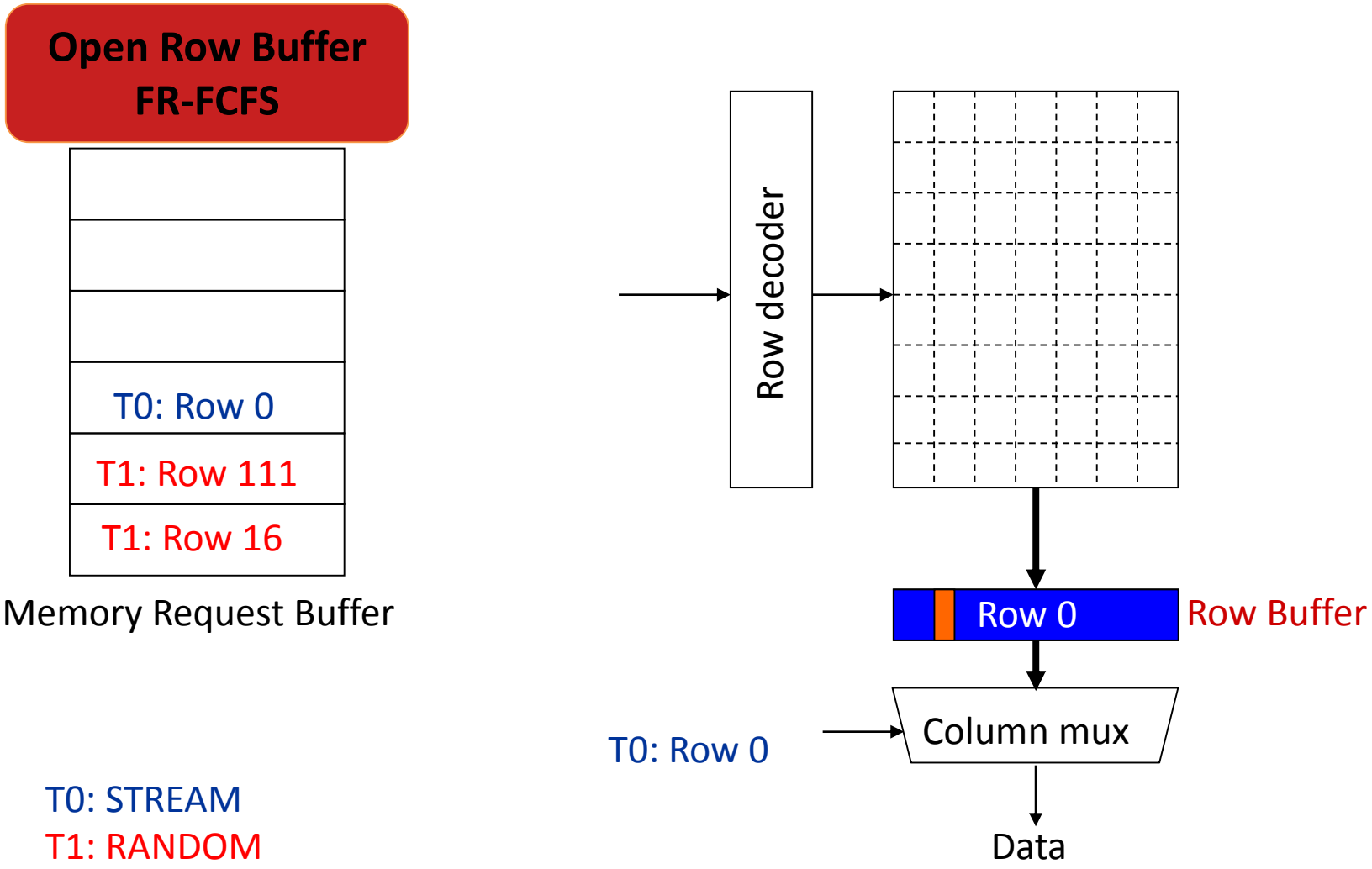
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



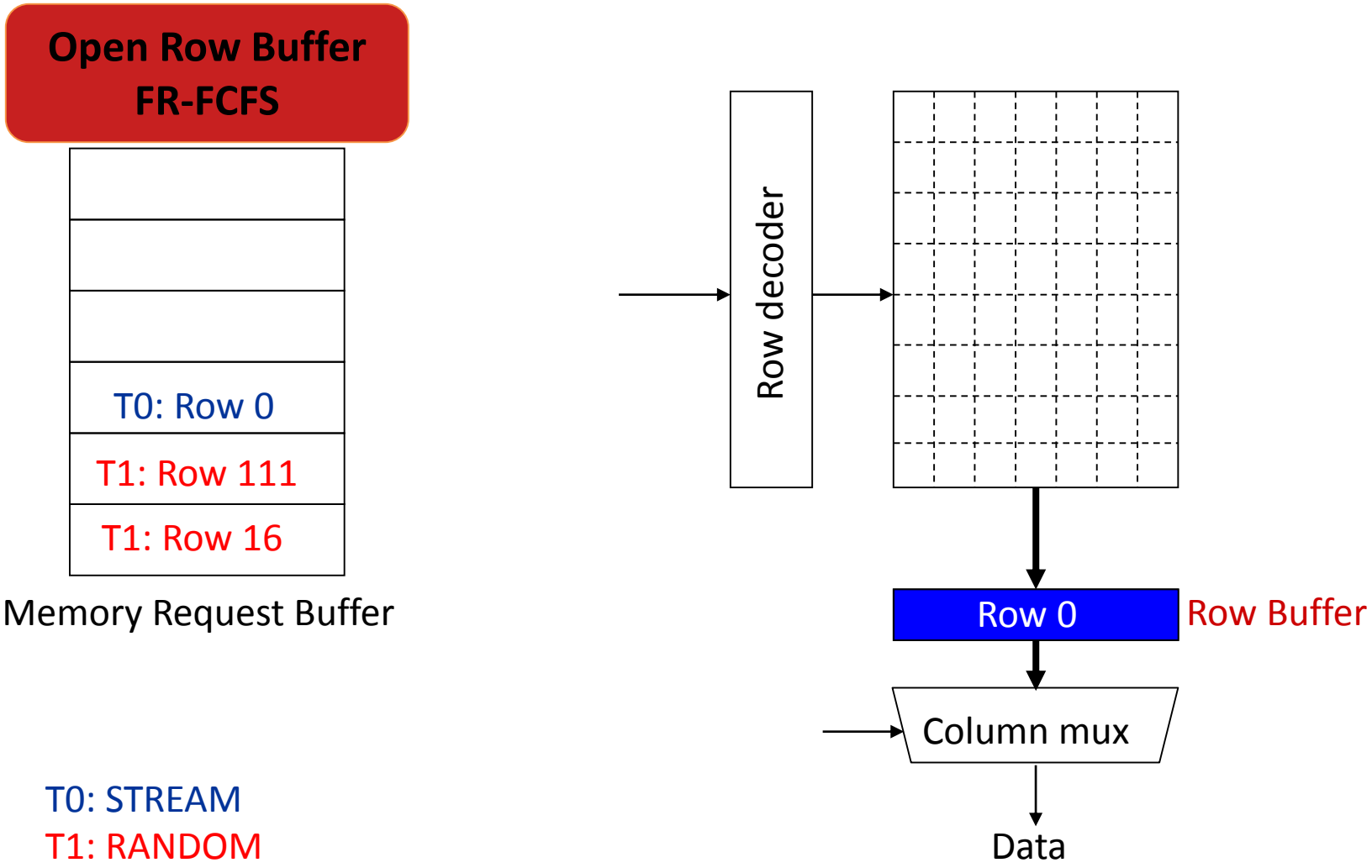
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



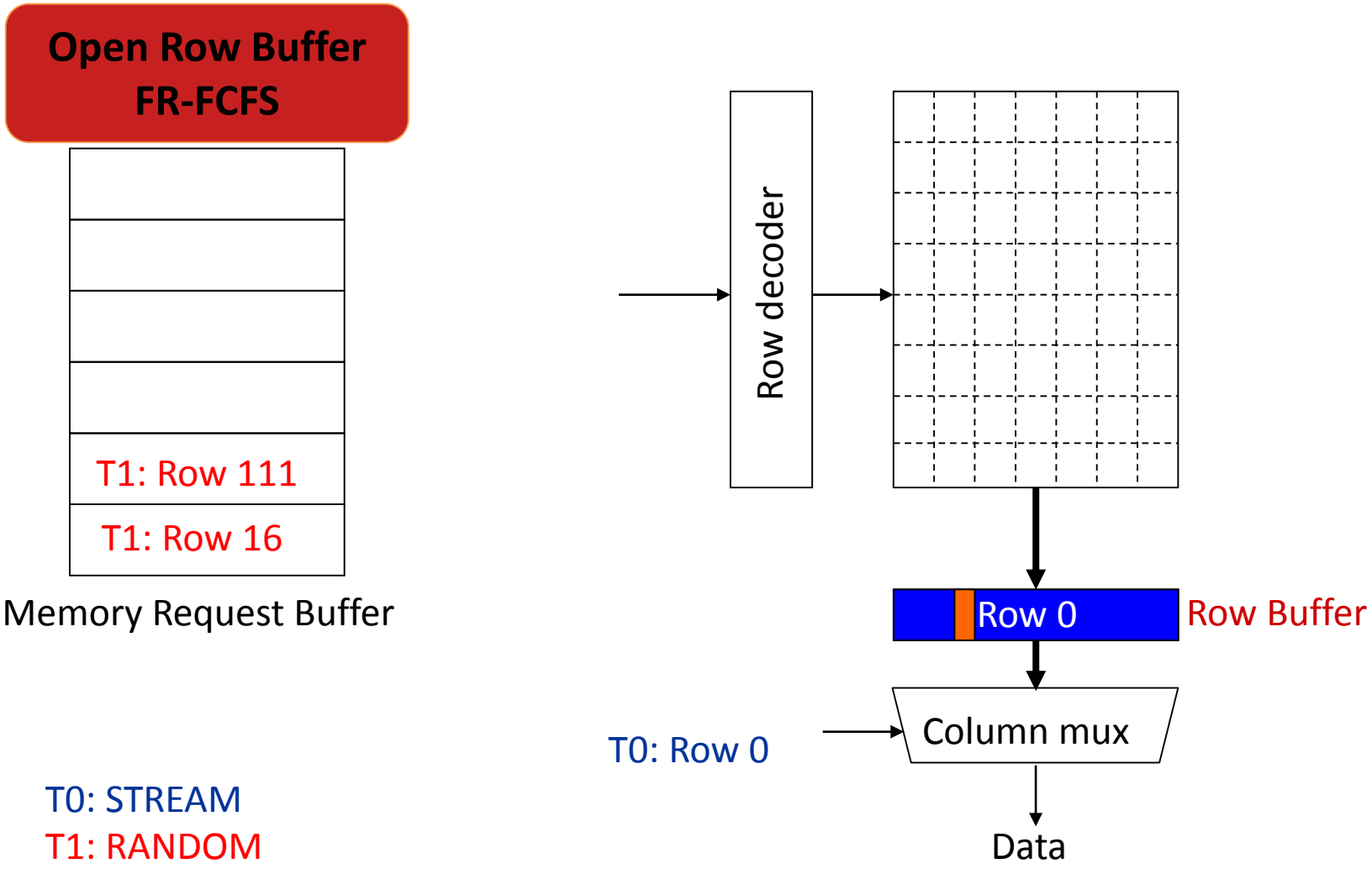
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



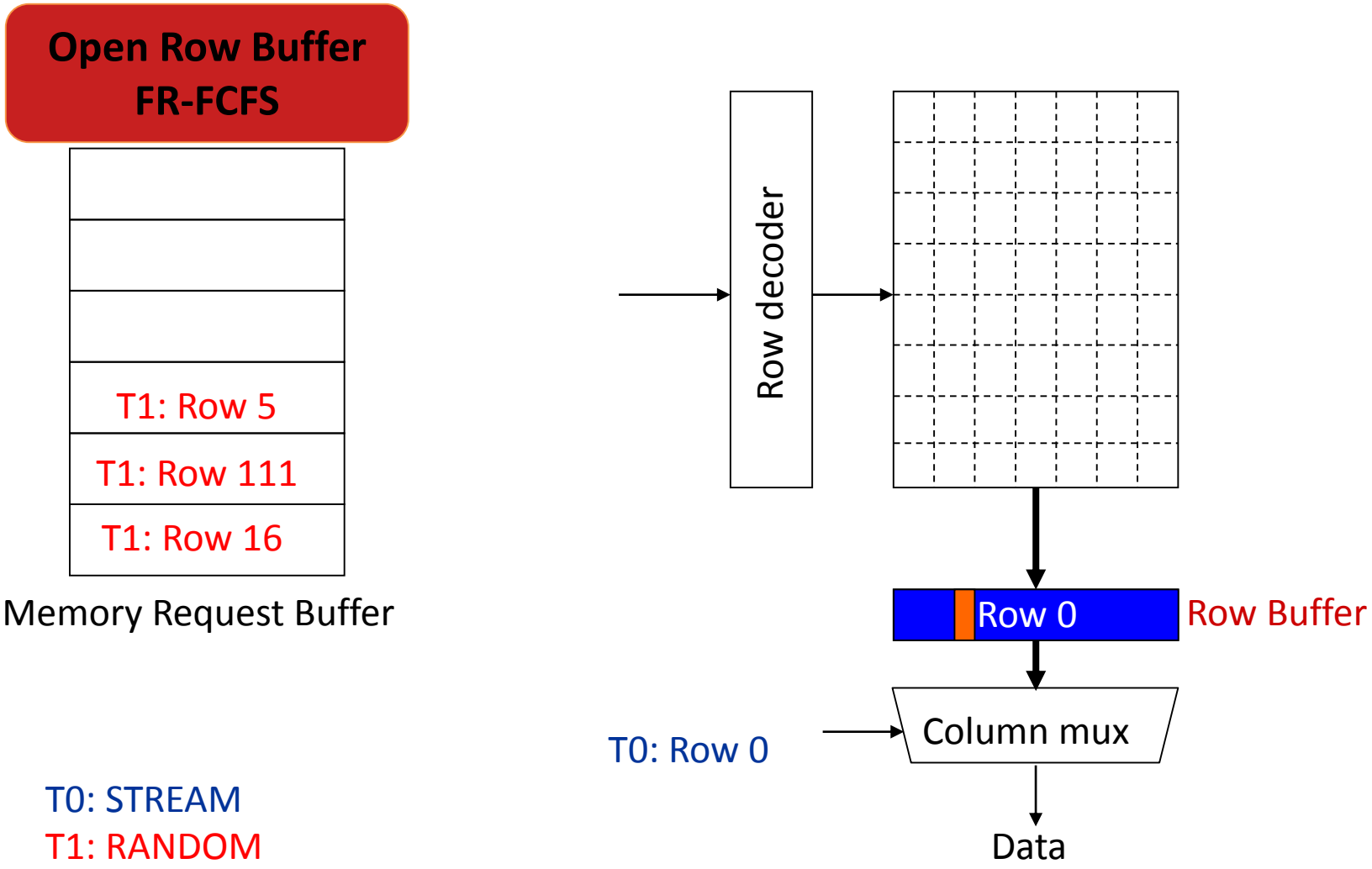
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



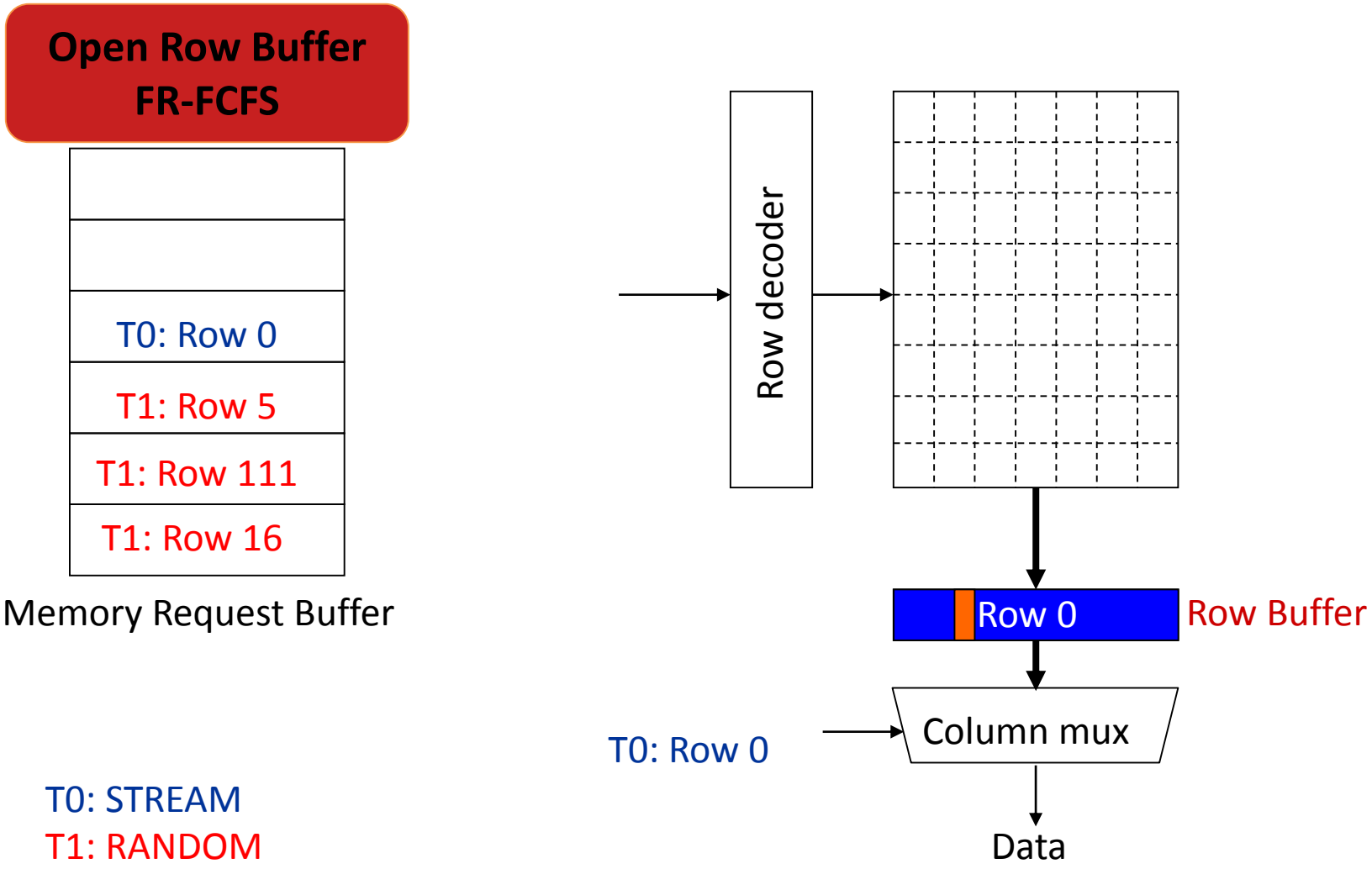
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



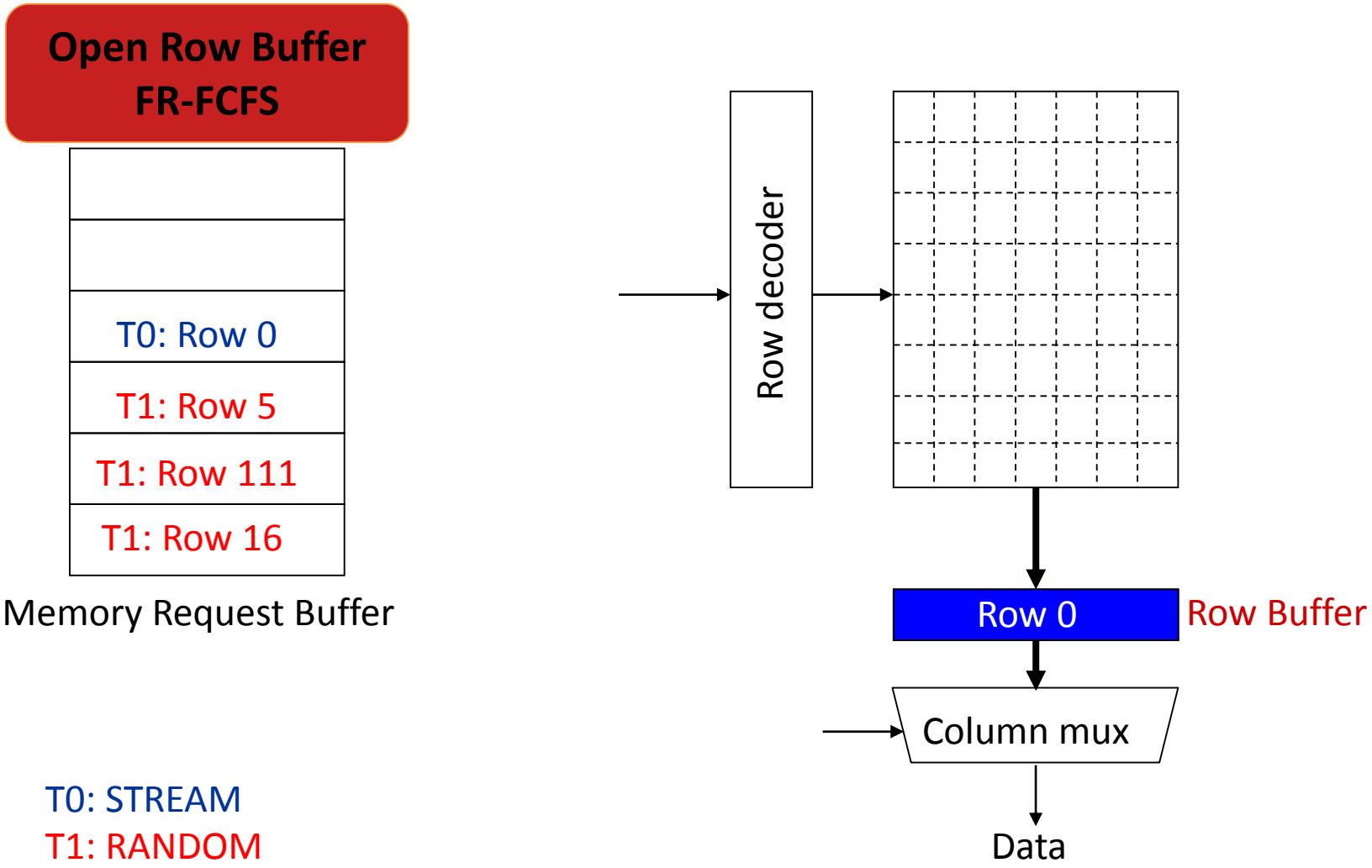
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



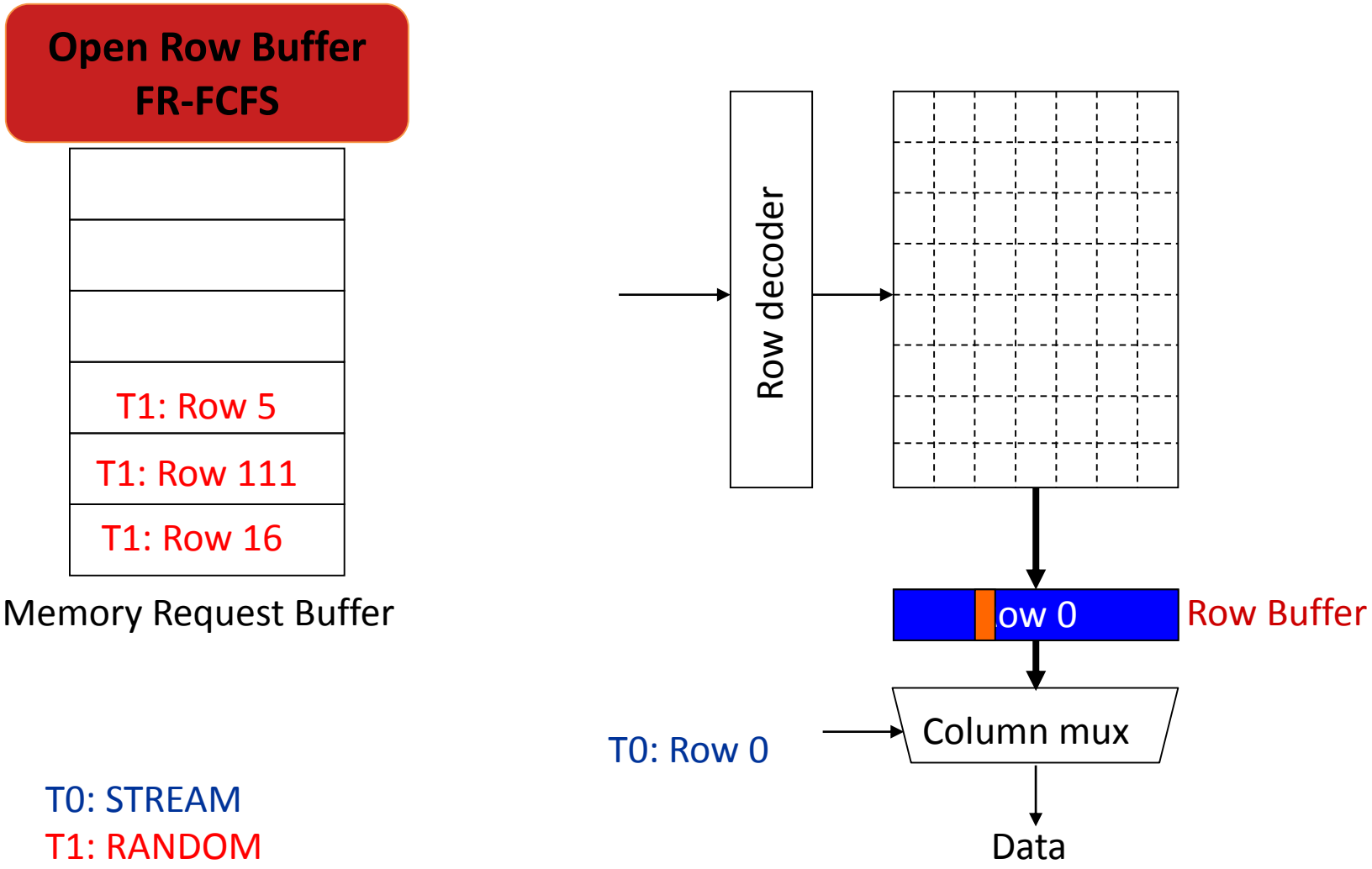
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



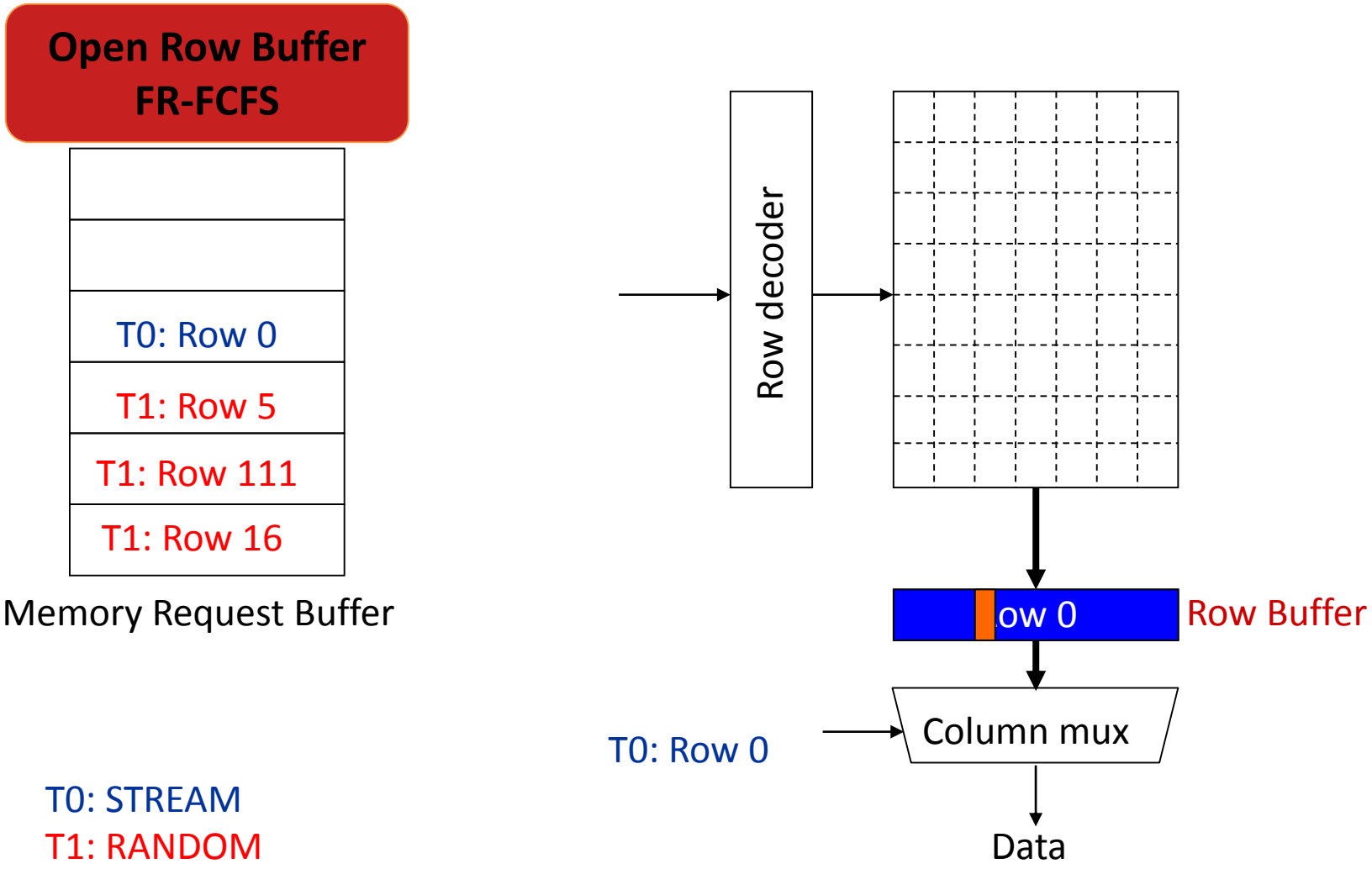
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



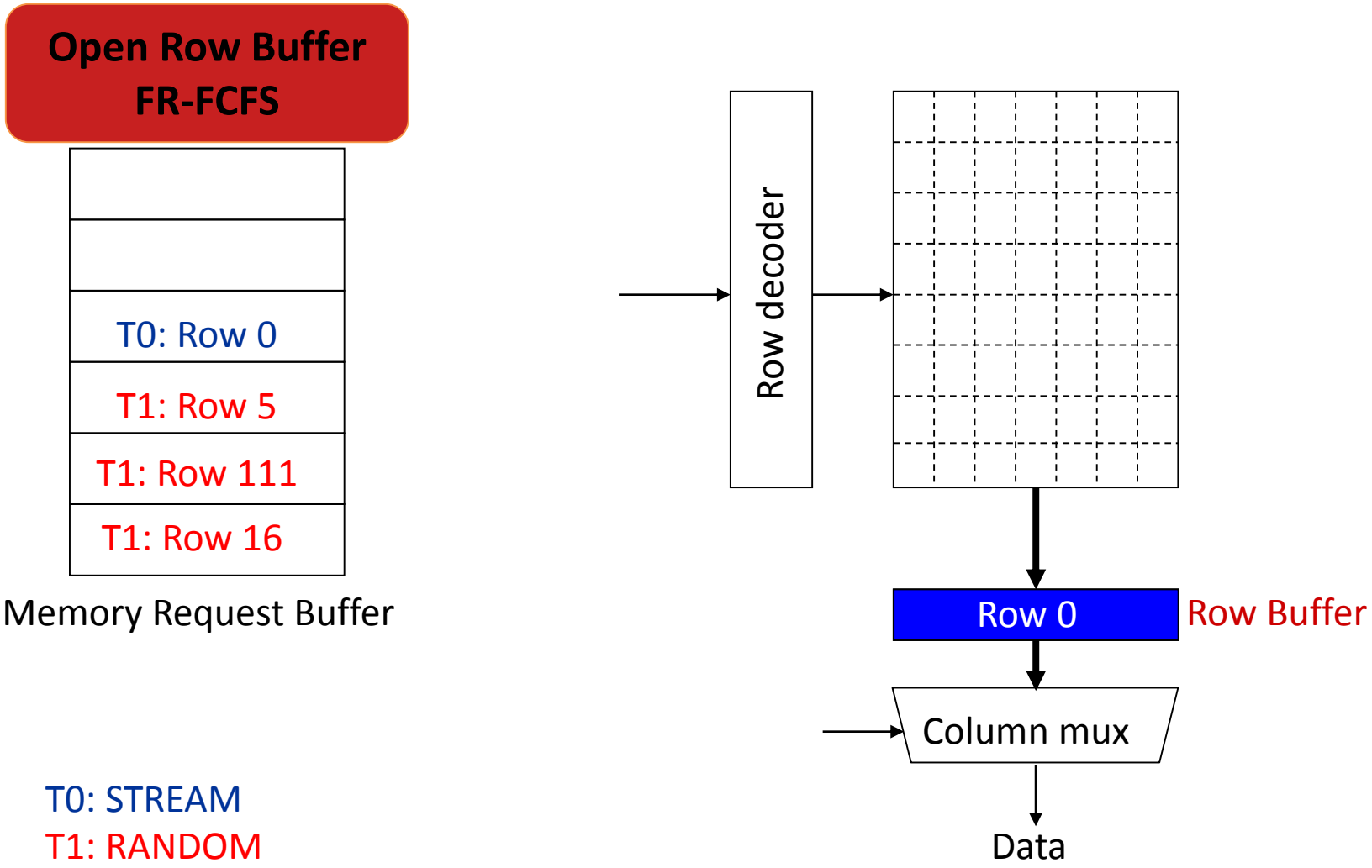
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



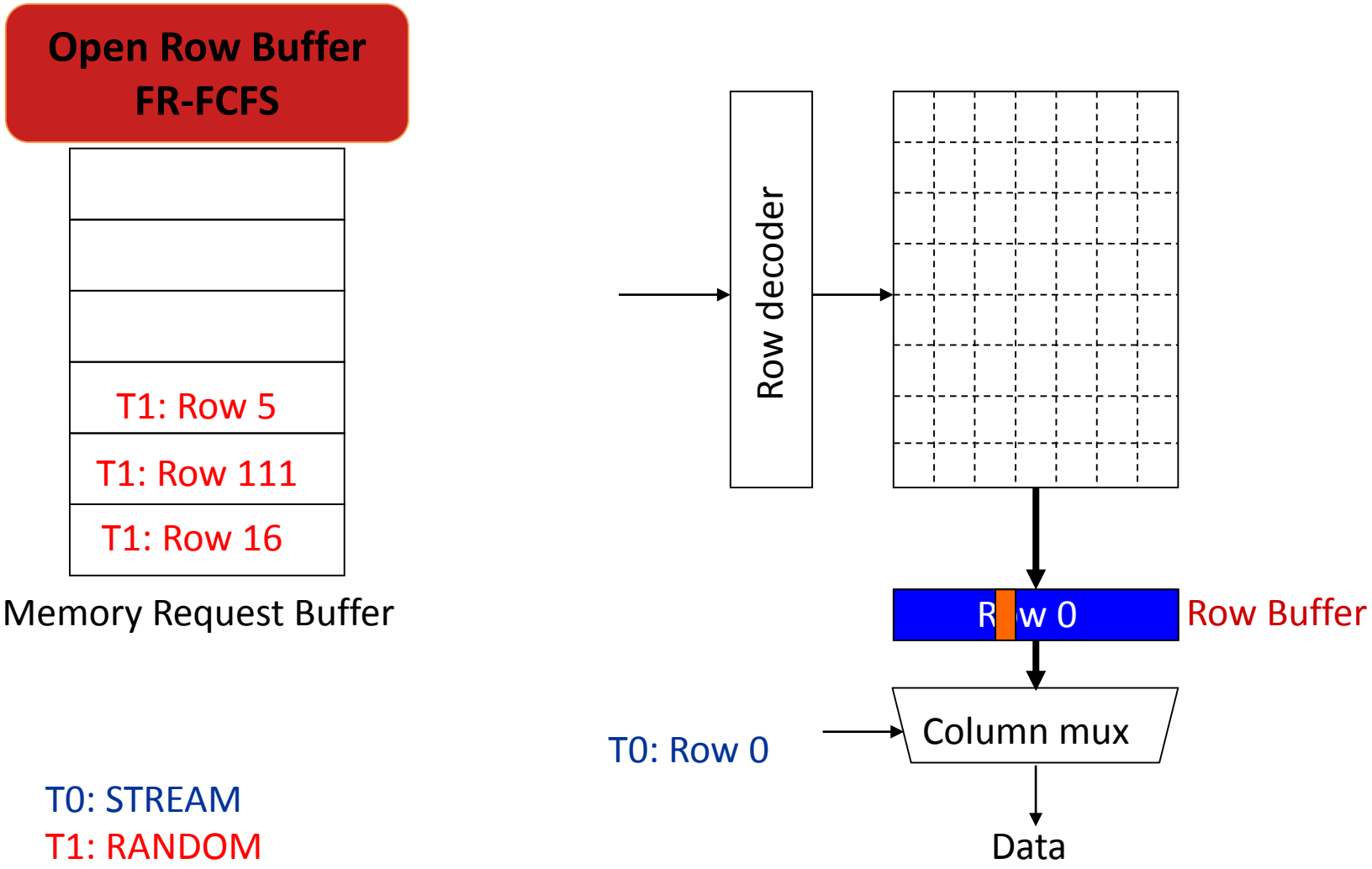
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



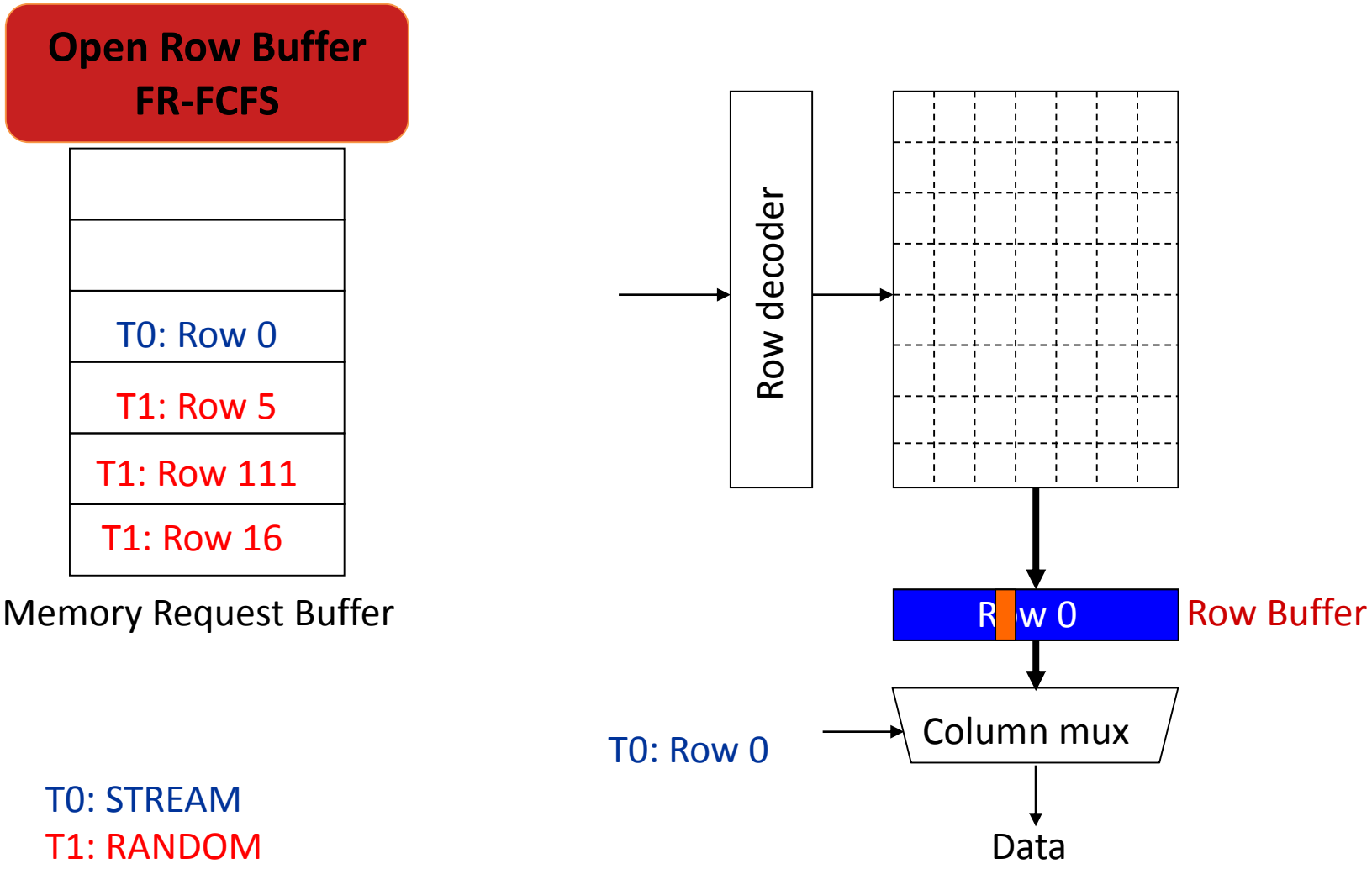
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



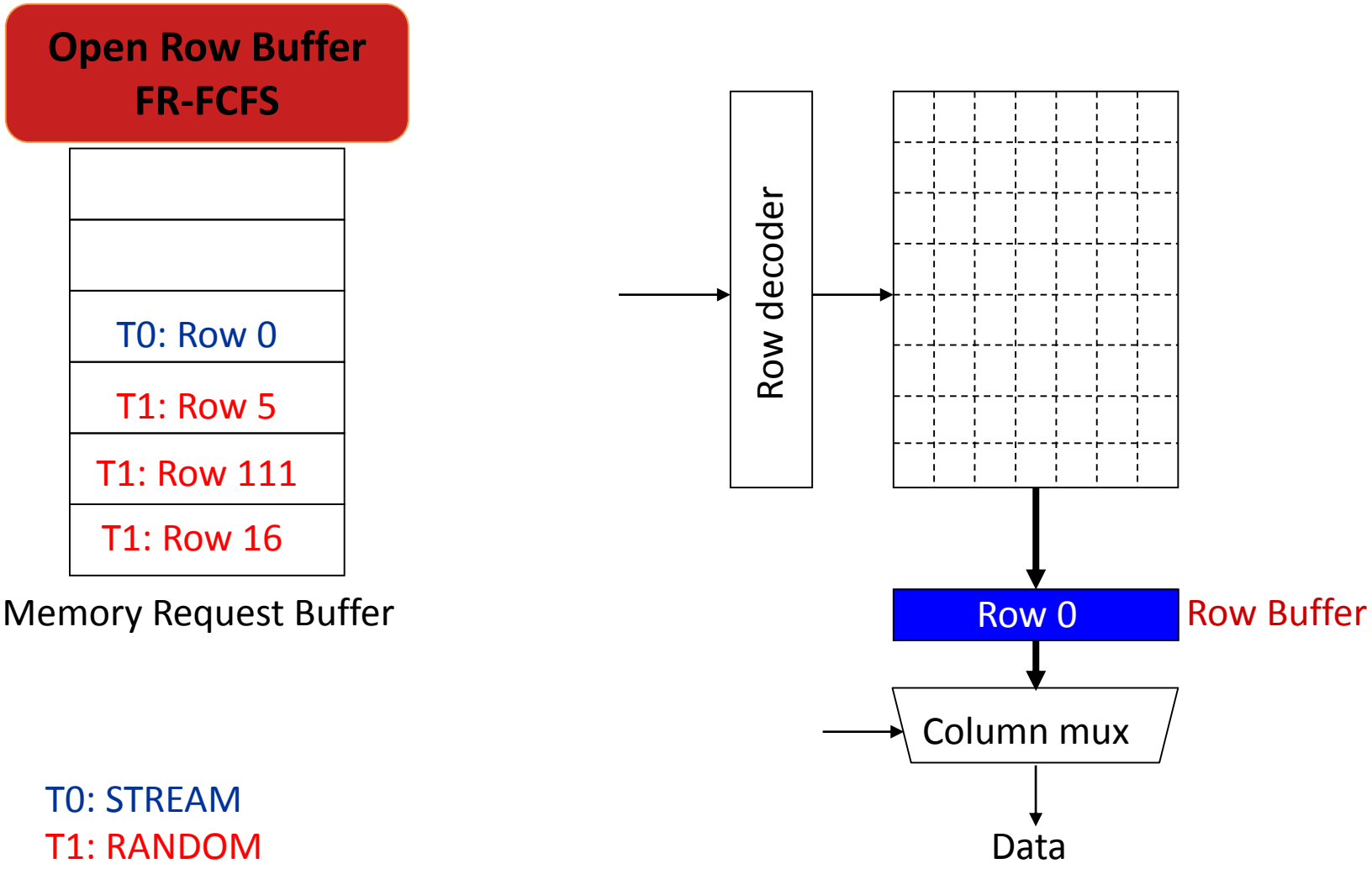
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



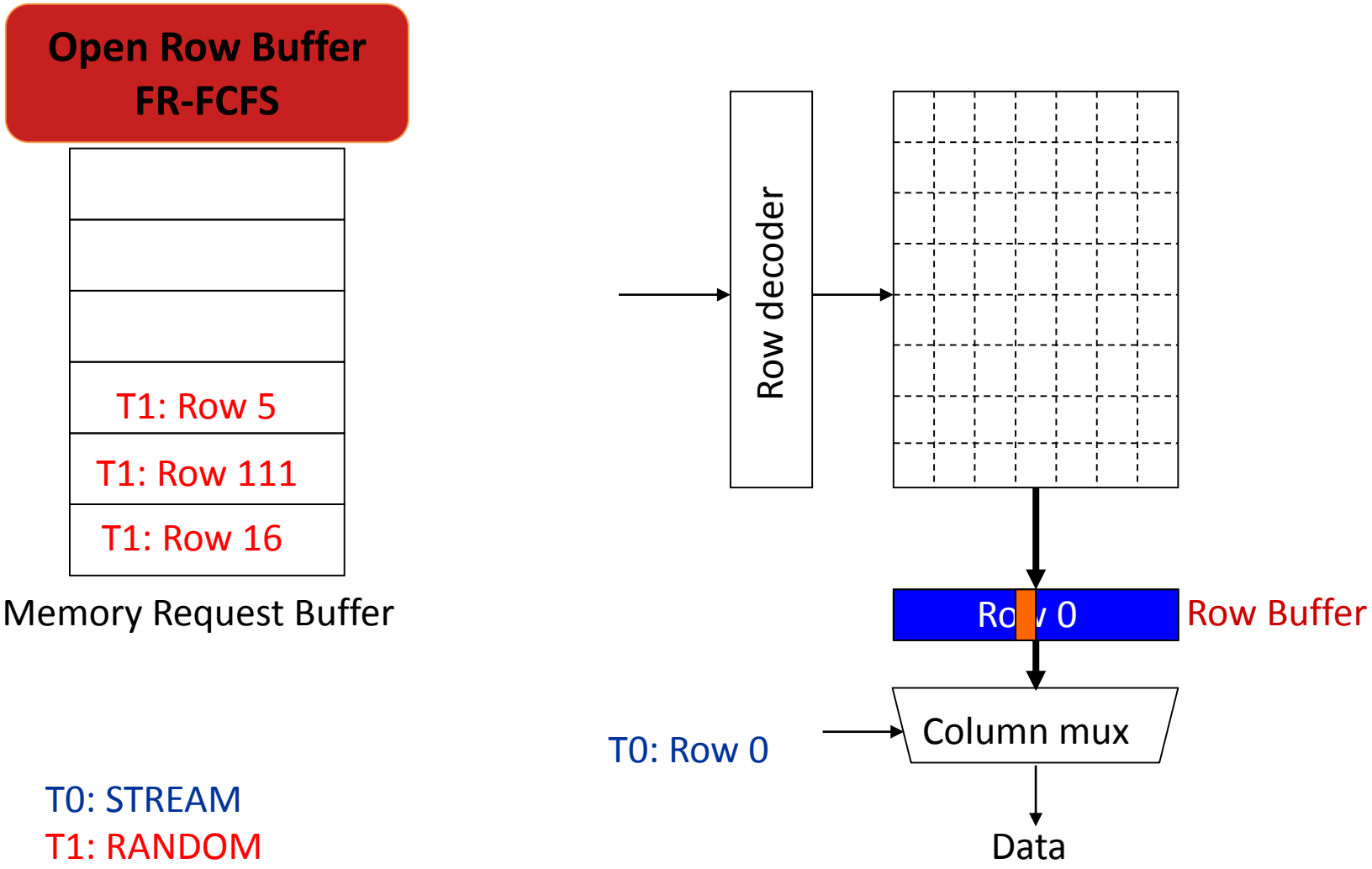
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



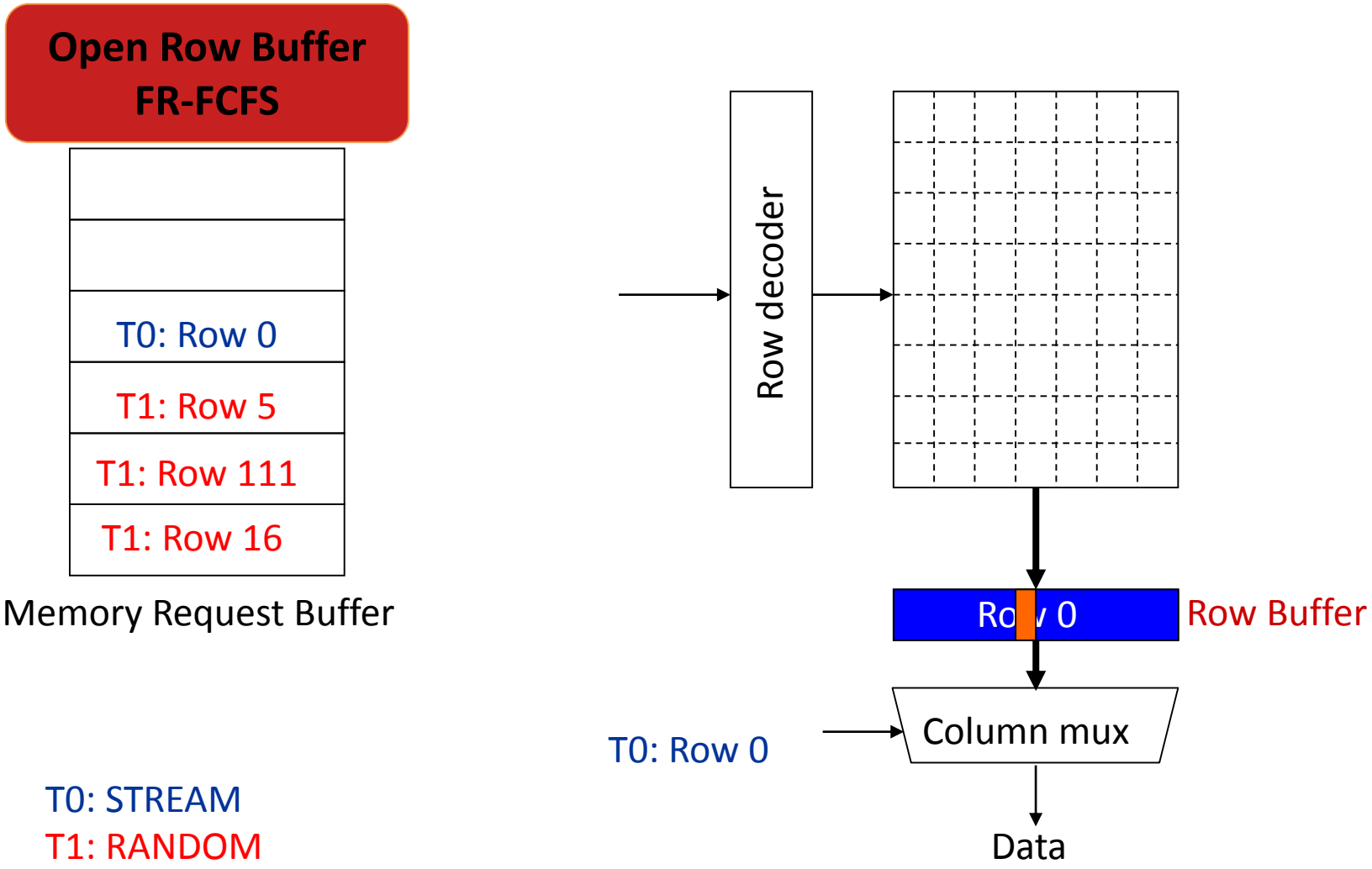
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



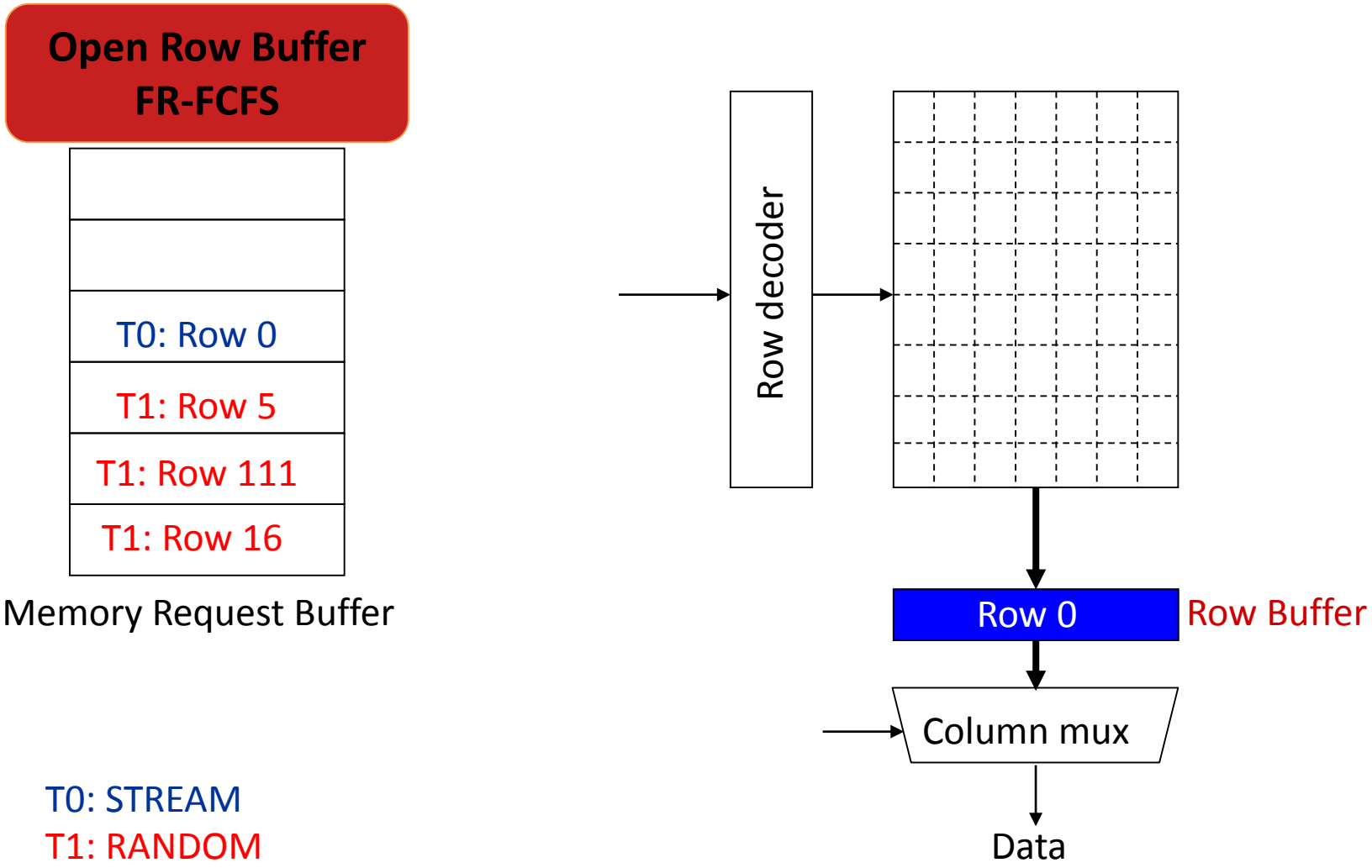
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



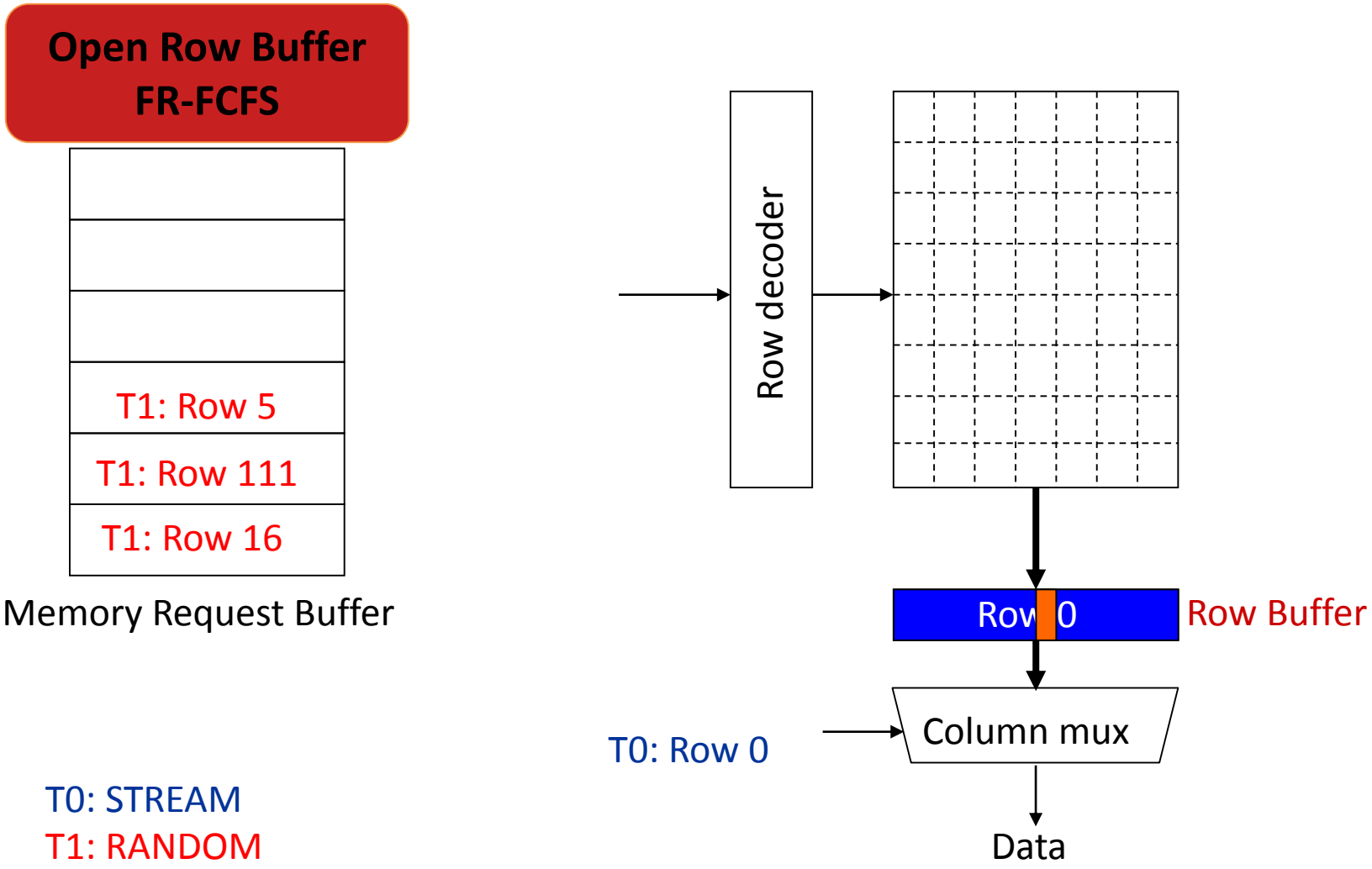
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



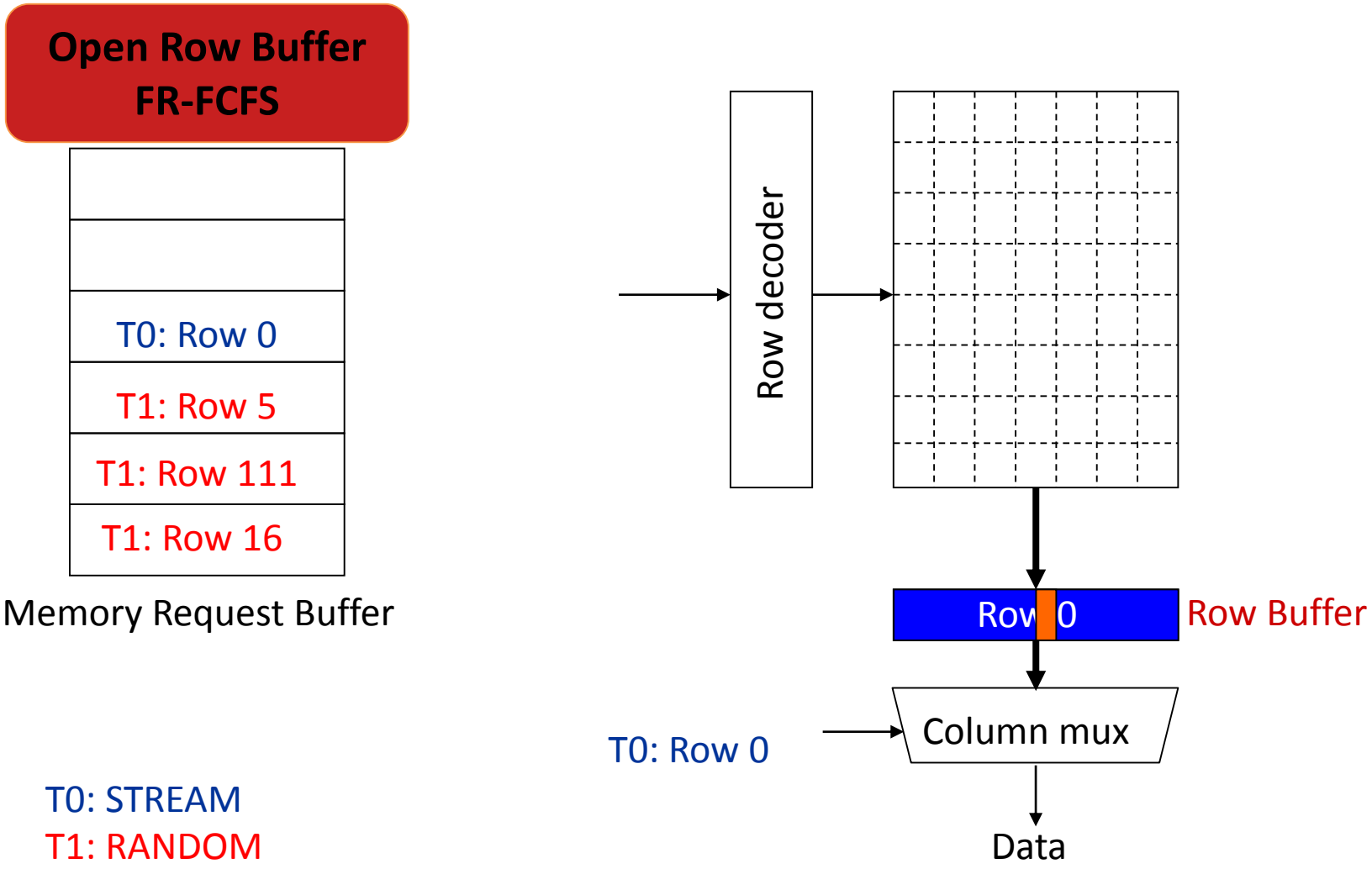
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



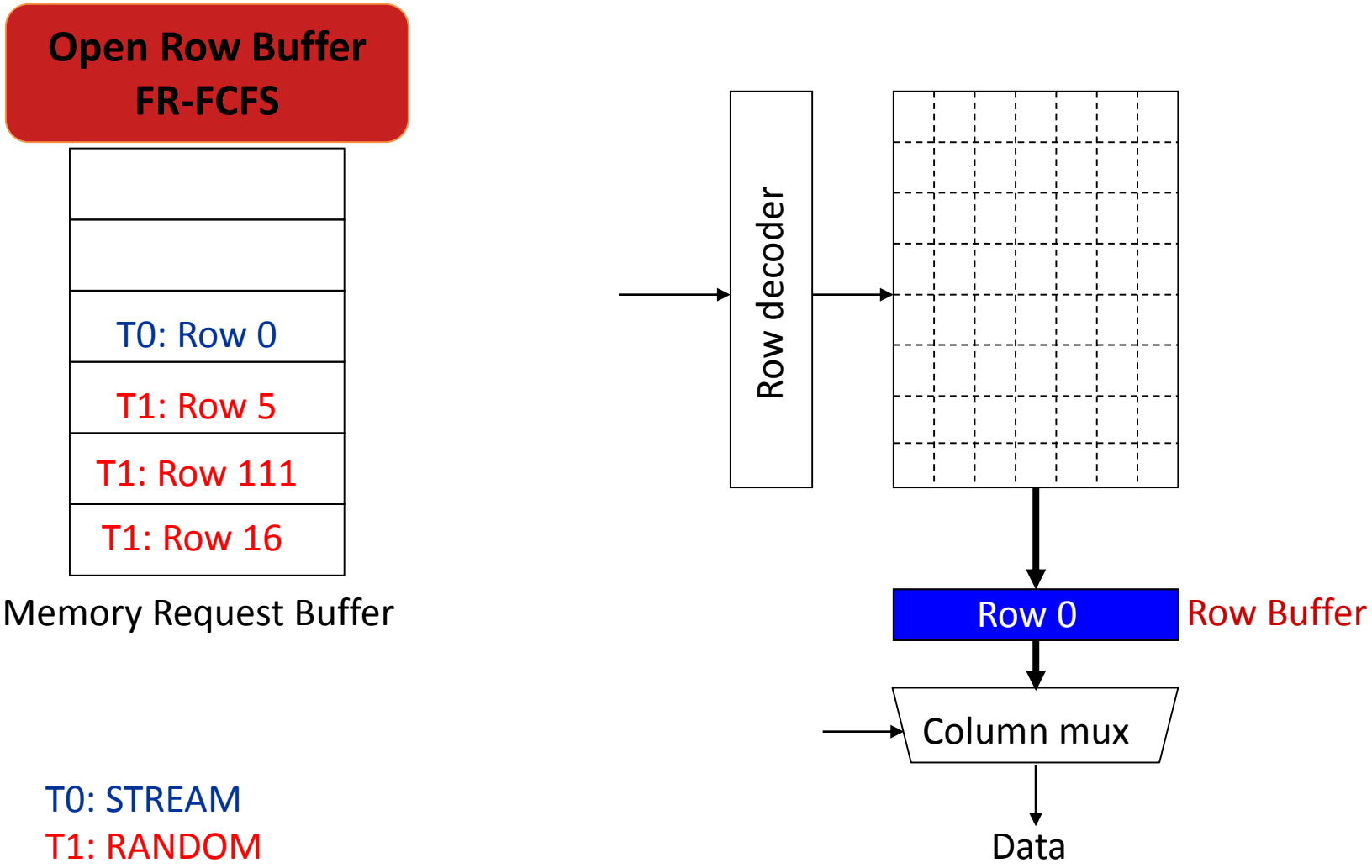
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



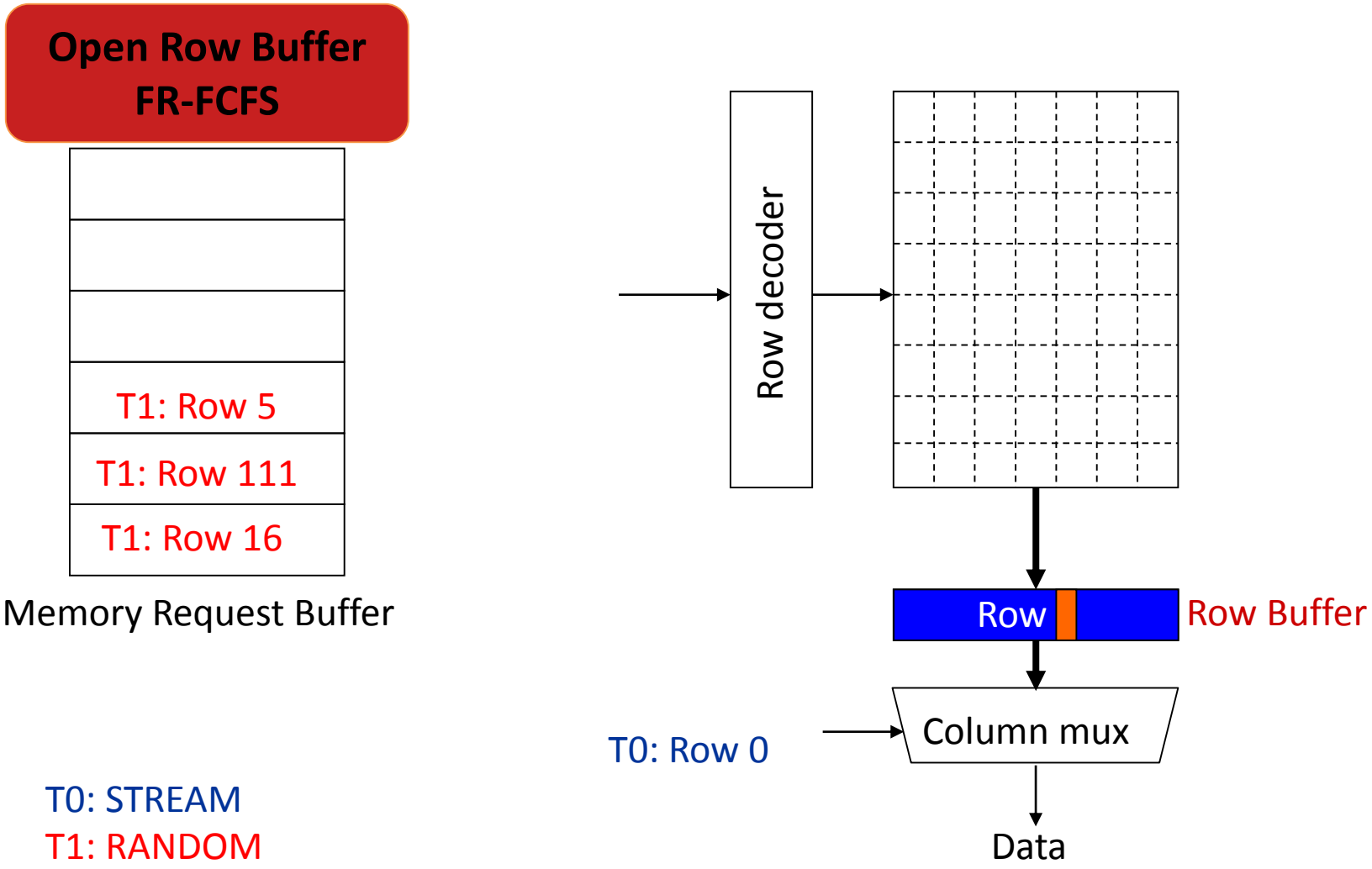
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



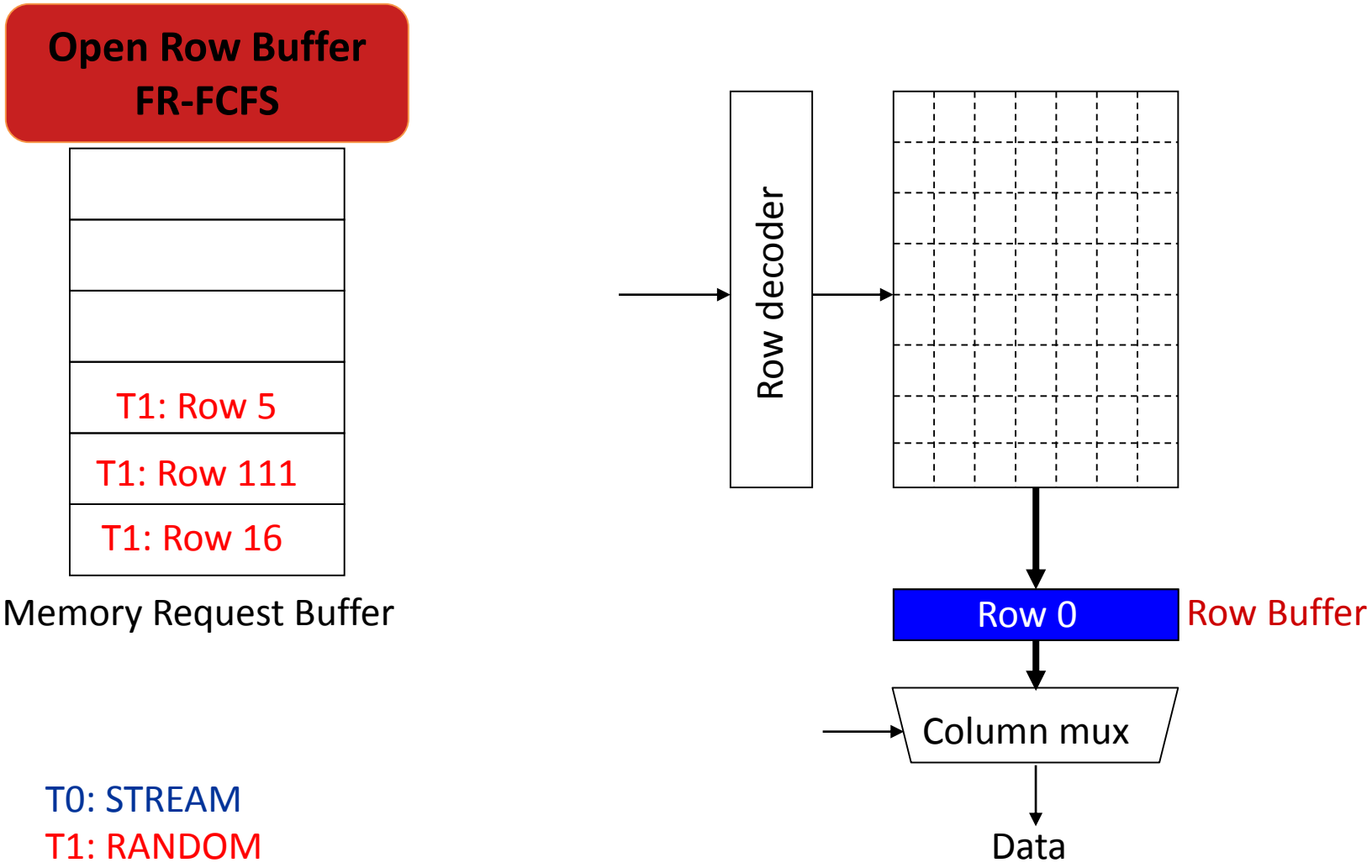
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



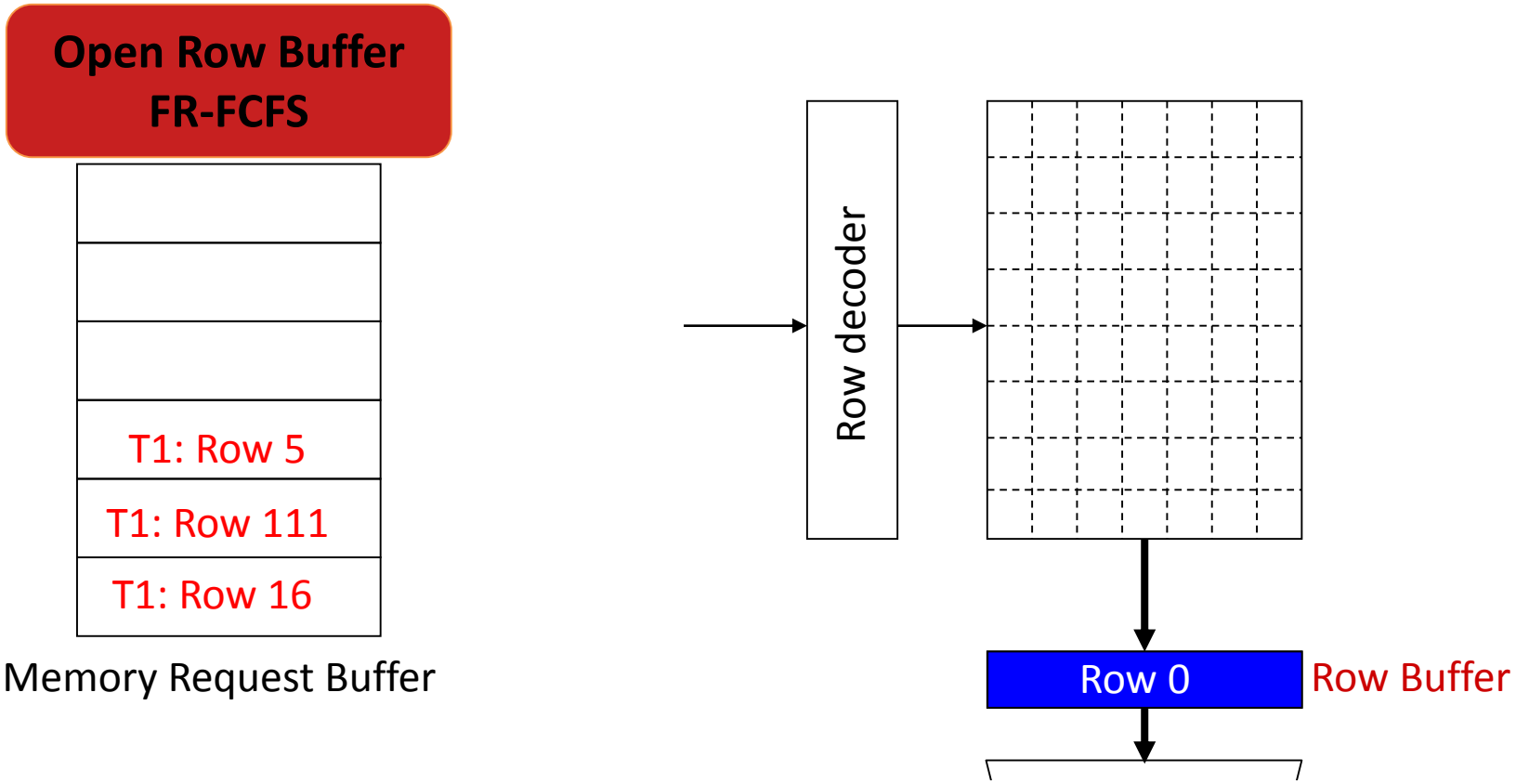
Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Memory Interference σε πολυπύρηνες αρχιτεκτονικές



Row size: 8KB, cache block size: 64B

128 (8KB/64B) requests of T0 serviced before T1

Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Διαχείριση του Memory Interference

- Ο memory controller πρέπει να λαμβάνει υπόψη του και:
  - Fairness
  - Predictability
  - Quality of Service (QoS)

- Μείωση/έλεγχος της “παρεμβολής” στο σύστημα μνήμης
  - Έξυπνες τεχνικές χρονοδρομολόγησης σε ετερογενή πολύπλοκα συστήματα: π.χ εφαρμογή τεχνικών εκμάθησης
  - Data/application partitiotining/mapping στα banks/ranks/channels
  - Μεταφορά πληροφορίας μικροαρχιτεκτονική ←→ controller π.χ κρισιμότητα των requests, thread awareness κ.α

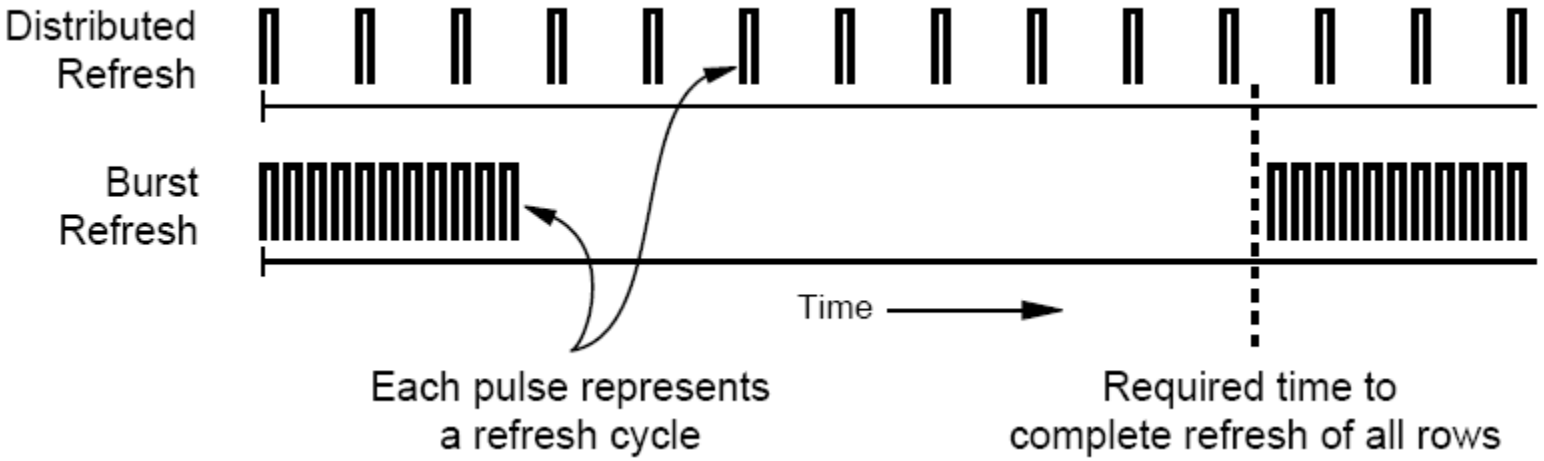
DRAM Refresh

DRAM Refresh

- Ο πυκνωτής της κυψελίδας DRAM μπορεί να αποφορτιστεί
  - ρεύματα διαρροής
- Ο memory controller πρέπει να διαβάζει περιοδικά κάθε row για να αποκαθιστά το φορτίο (refresh)
  - Activate + Precharge κάθε row άνα N ms (συνήθως N=64)
- Επιπτώσεις στην επίδοση:
  - Το DRAM bank δεν είναι διαθέσιμο όσο αναζωογονείται
  - Μεγάλοι χρόνοι αναμονής: Αν όλα τα rows αναζωογονηθούν μαζί (in burst), η DRAM δεν είναι διαθέσιμη κάθε 64ms μέχρι να ολοκληρωθεί το refresh

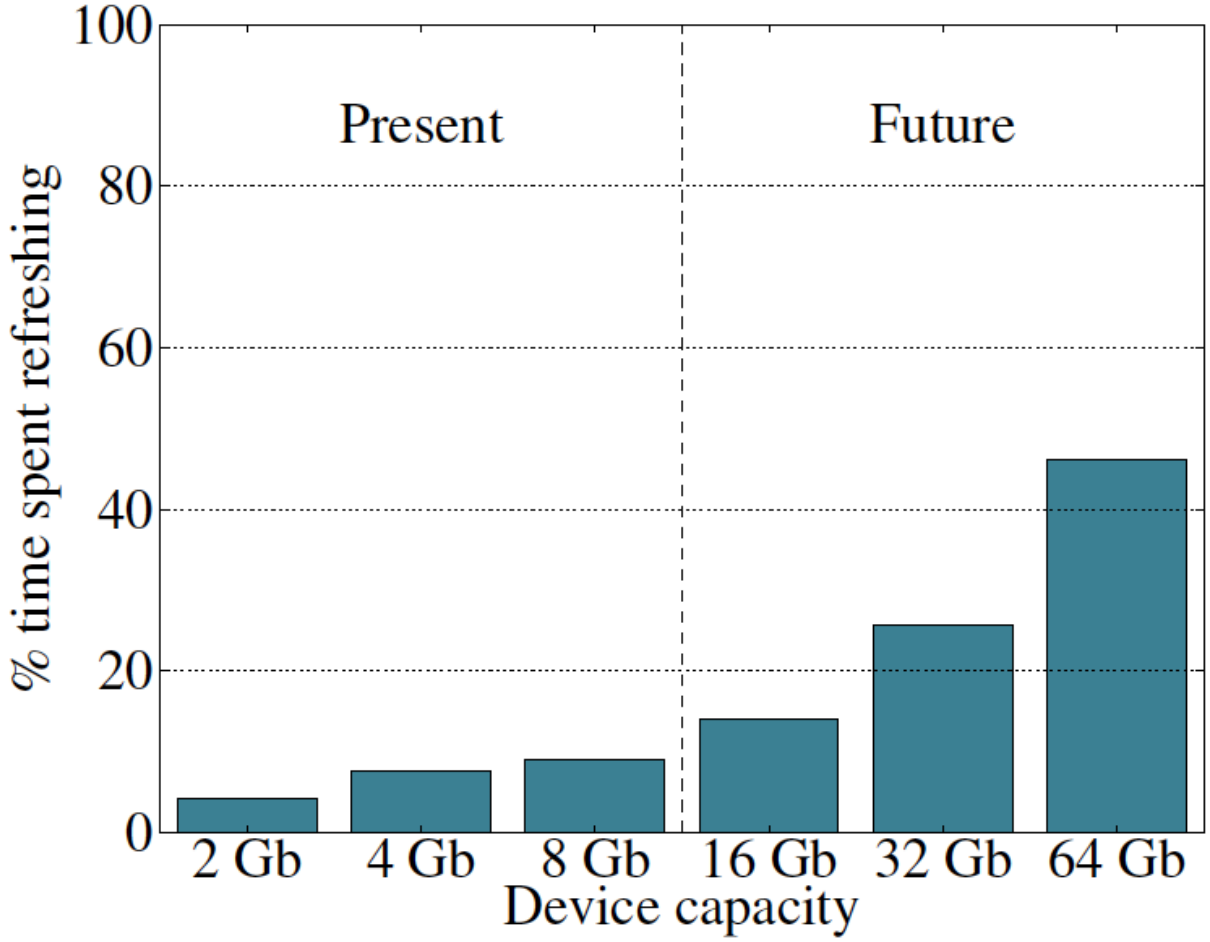
DRAM Refresh

- Burst Refresh: όλα τα rows αναζωογονούνται κατευθείαν το ένα μετά το άλλο
- Distributed Refresh: κάθε row αναζωογονείται σε διαφορετική χρονική στιγμή ανά τακτά χρονικά διαστήματα



- Το distributed refresh εξαλείφει τους μεγάλους χρόνους αναμονής

Refresh: Κόστος σε επίδοση



Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.