Flash Controller

These slides were made by

Prof. Myoungsoo Jung of Yonsei University.

Outline

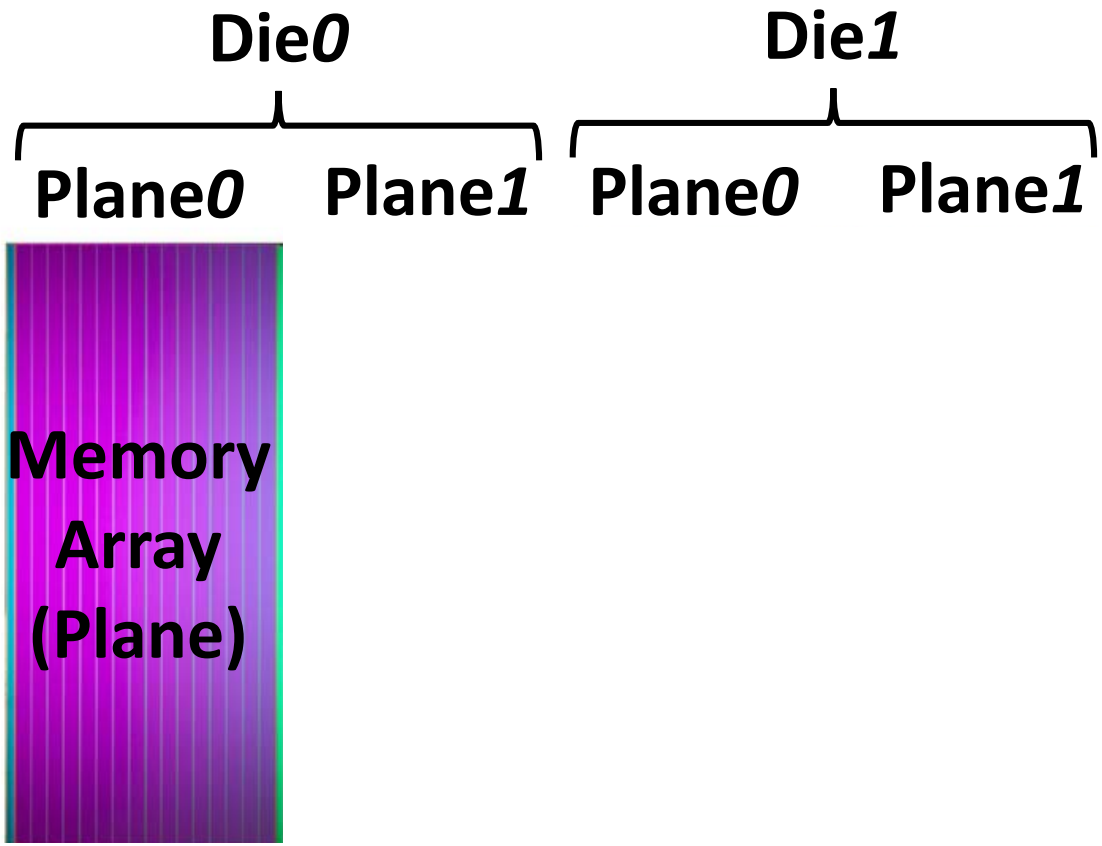
- Flash Microarchitecture
- Basic Operations (Legacy)
- Cache Mode Operations
- Multi-plane Mode Operations
- Copy-back Operations
- Erase Operations
- Evaluation Studies
  - Simulation-based studies
  - Real implementation (FGPA) and evaluation

Outline

- Flash Microarchitecture
- Basic Operations (Legacy)
- Cache Mode Operations
- Multi-plane Mode Operations
- Copy-back Operations
- Erase Operations
- Evaluation Studies
  - Simulation-based studies
  - Real implementation (FGPA) and evaluation

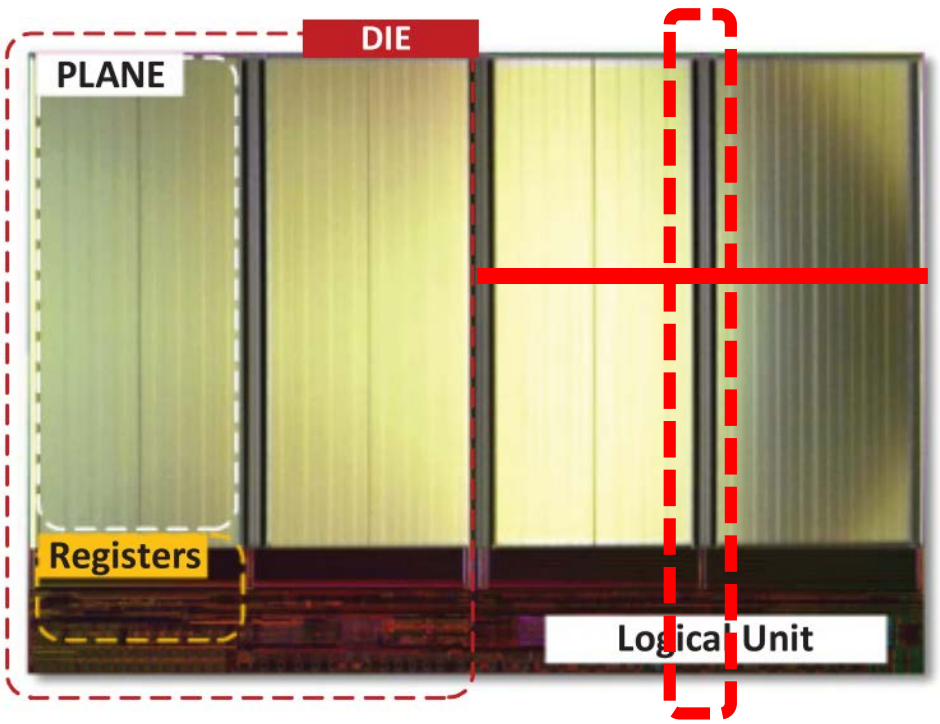
NAND Flash Architecture

- Employing cache and data registers
- Multiple planes (memory array)
- Multiple dies



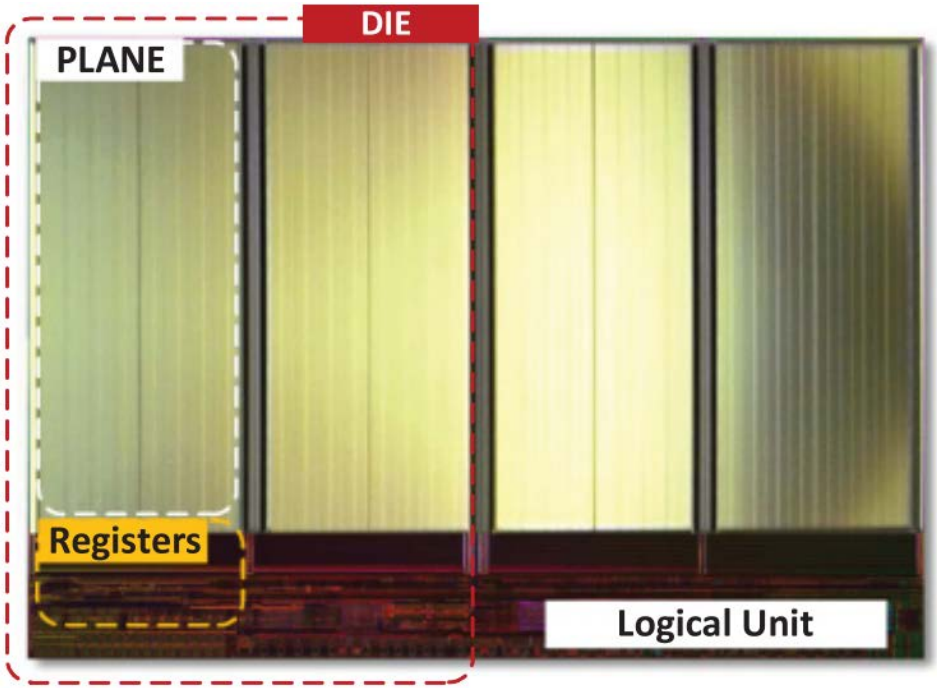
Package Organization

- Flash package can employ multiple **Dies**, each consisting of one or more flash memory islands, referred to as **Plane**
  - Planes in typical share multiple peripherals, which enable all the target pages connected to their wordlines/bitlines
  - Please review the bottom lines of flash memory operations at our first lecture
- Each plane employs a set of **registers** to cache/buffer the data brought by flash interface



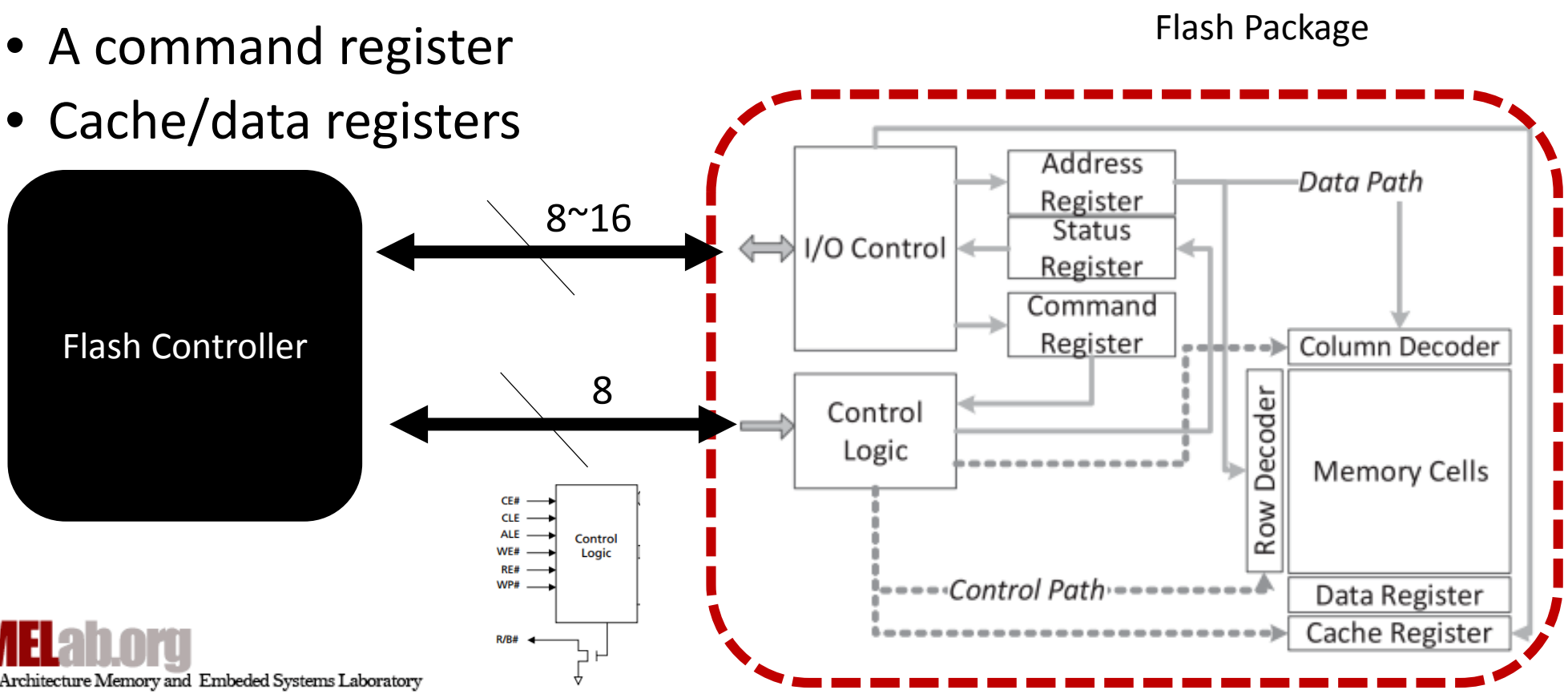
Physical Plane and Die

- Each plane defines a different set of block addresses, and the pages/blocks on different planes operate in parallel
  - The operation type for the requests across different plane should be same
  - The page and plane addresses should be identical
- All individual dies can be simultaneously activated, but share datapath (described at the next page)



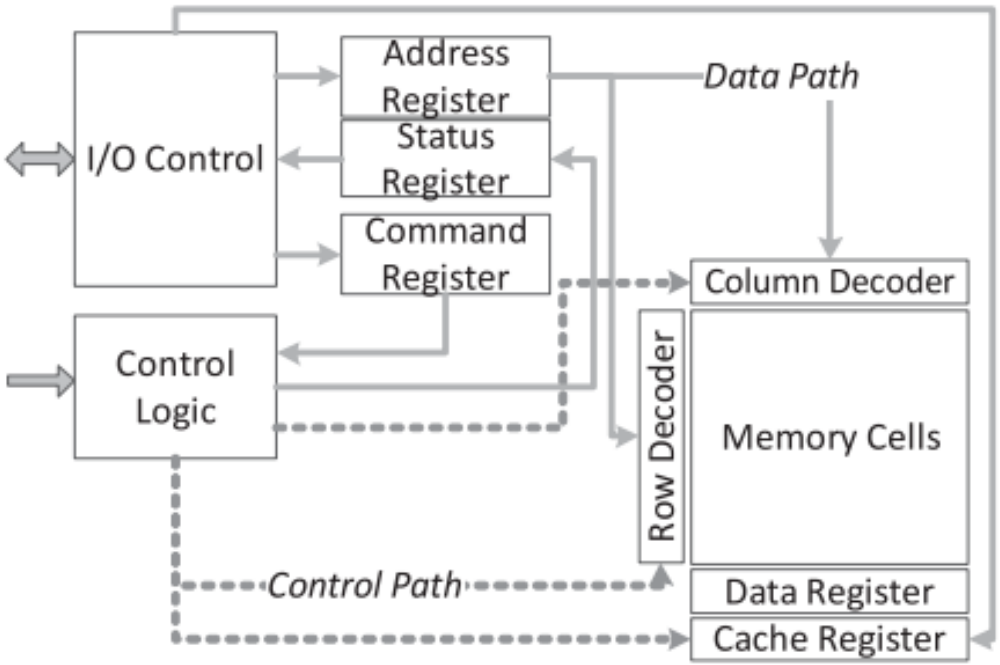
Flash Microarchitecture

- All data, commands and addresses are multiplexed onto same I/O pins and received by I/O control circuit
- Each component of flash transactions is latched by
  - An address register
  - A status register
  - A command register
  - Cache/data registers



Flash Interface and Protocol

- Thus, to issue a read/write request through the multiplexed I/O pins, users (i.e., flash controllers) should obey the protocol that flash interface defines for all memory transactions
  - Basic operations
  - Cache mode operations
  - Multi-plane mode operations
  - Copyback, etc.

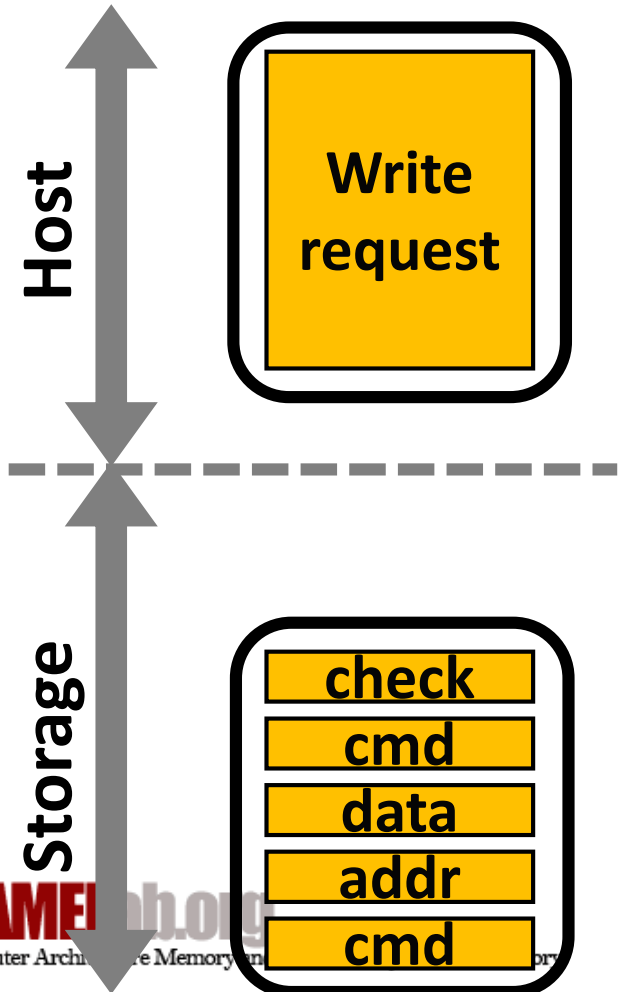


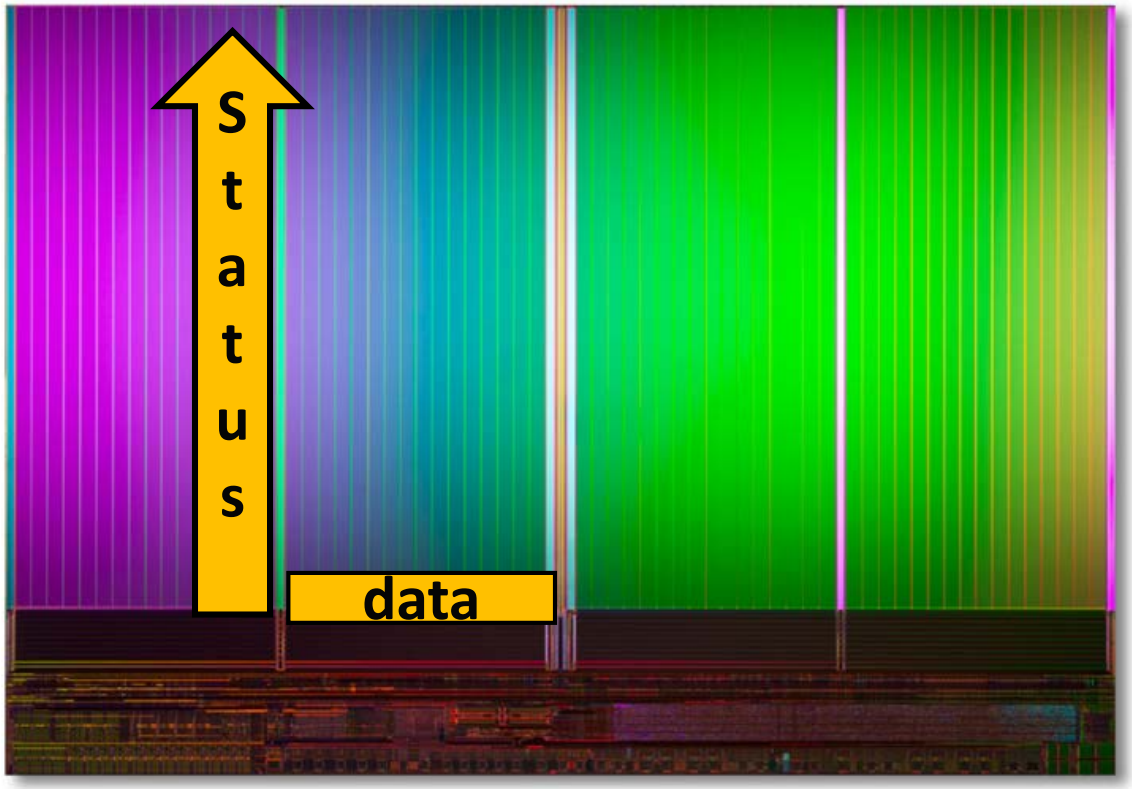
Outline

- Flash Microarchitecture
- Basic Operations (Legacy)
- Cache Mode Operations
- Multi-plane Mode Operations
- Copy-back Operations
- Erase Operations
- Evaluation Studies
  - Simulation-based studies
  - Real implementation (FGPA) and evaluation

Legacy Operation

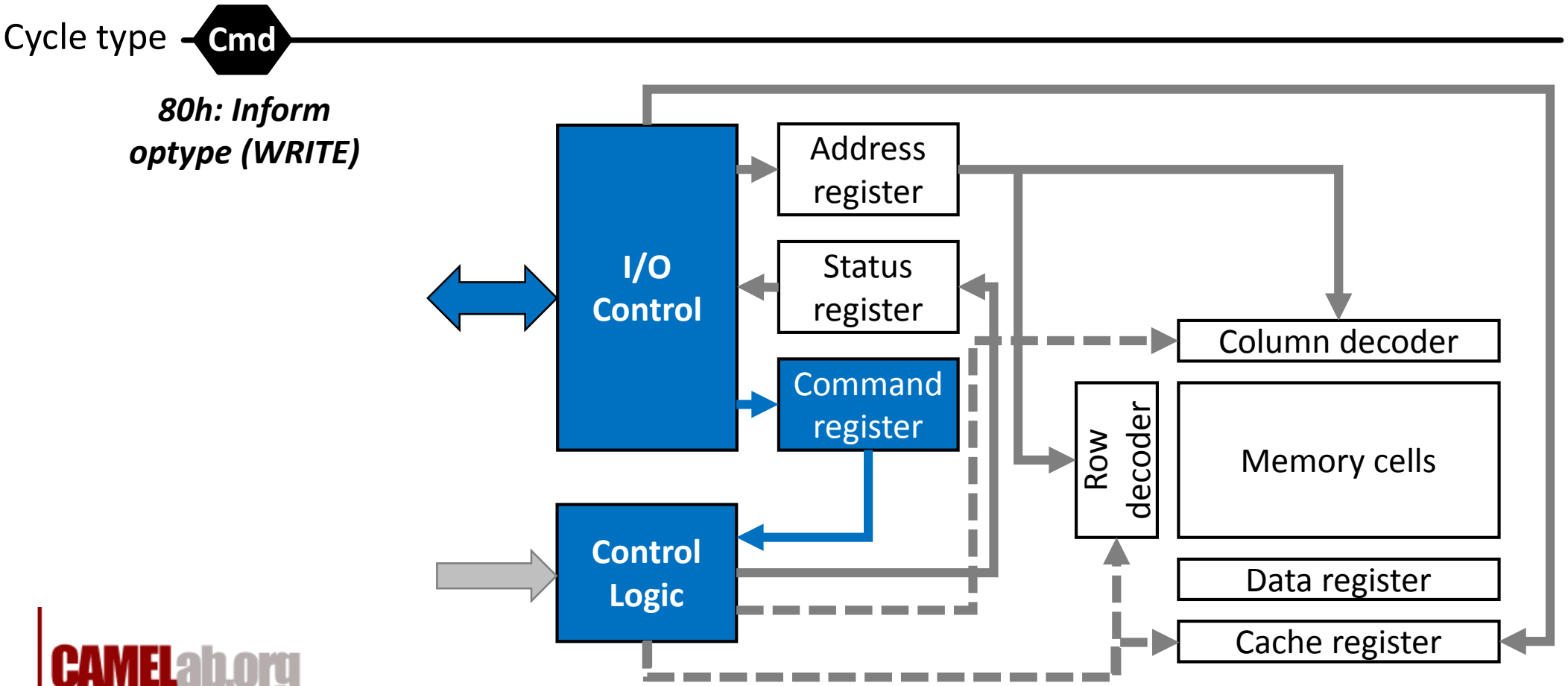
- An I/O request splits into several operation stages (defined by flash interface protocols such as ONFi)
- Each stage should be appropriately handled by flash controller(s)





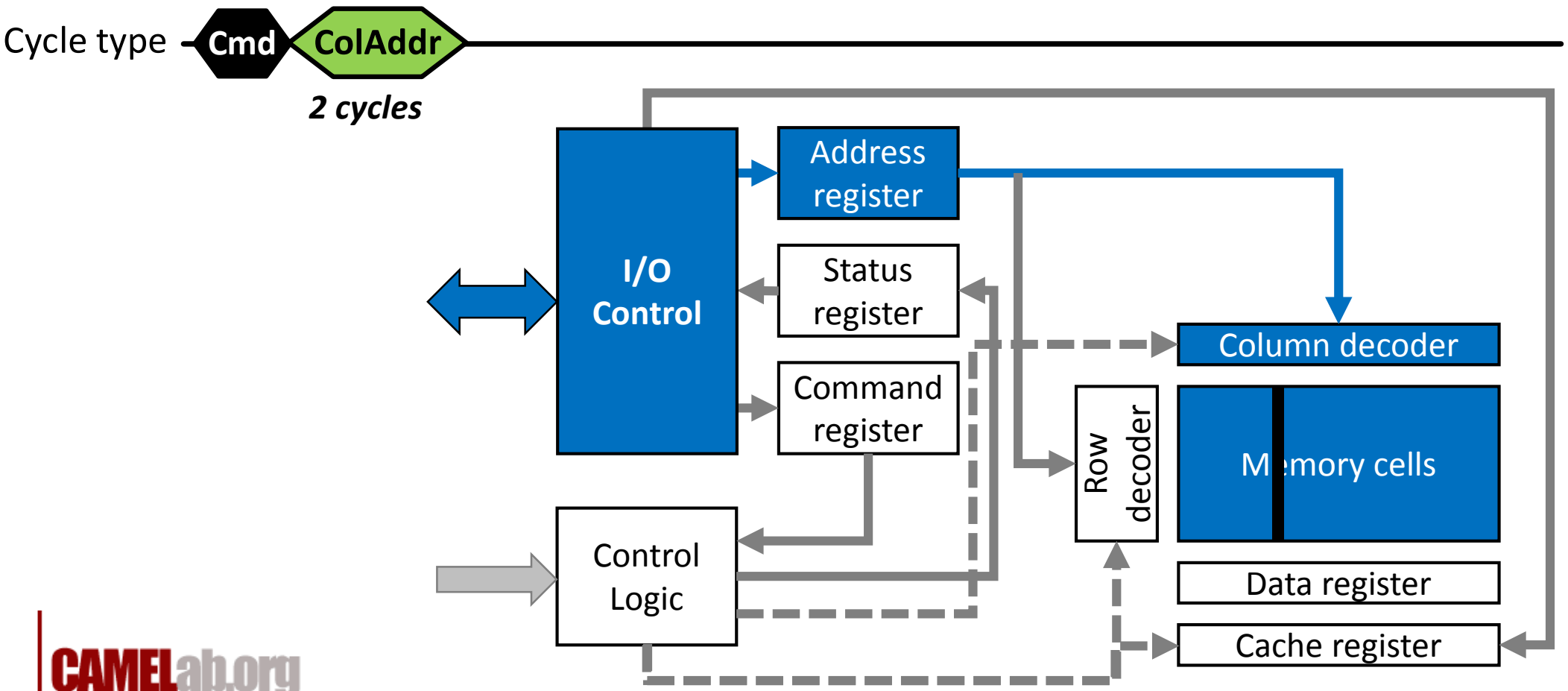
A Basic Write

- The first command is needed to indicate that this is a legacy page write
- The column address (C-ADDR) and row address (R-ADDR) are shipped in a serial order
- Data should be then transferred, and the second command initiates a write
- Once the write is performed, users need to send the last command to check up the status of target



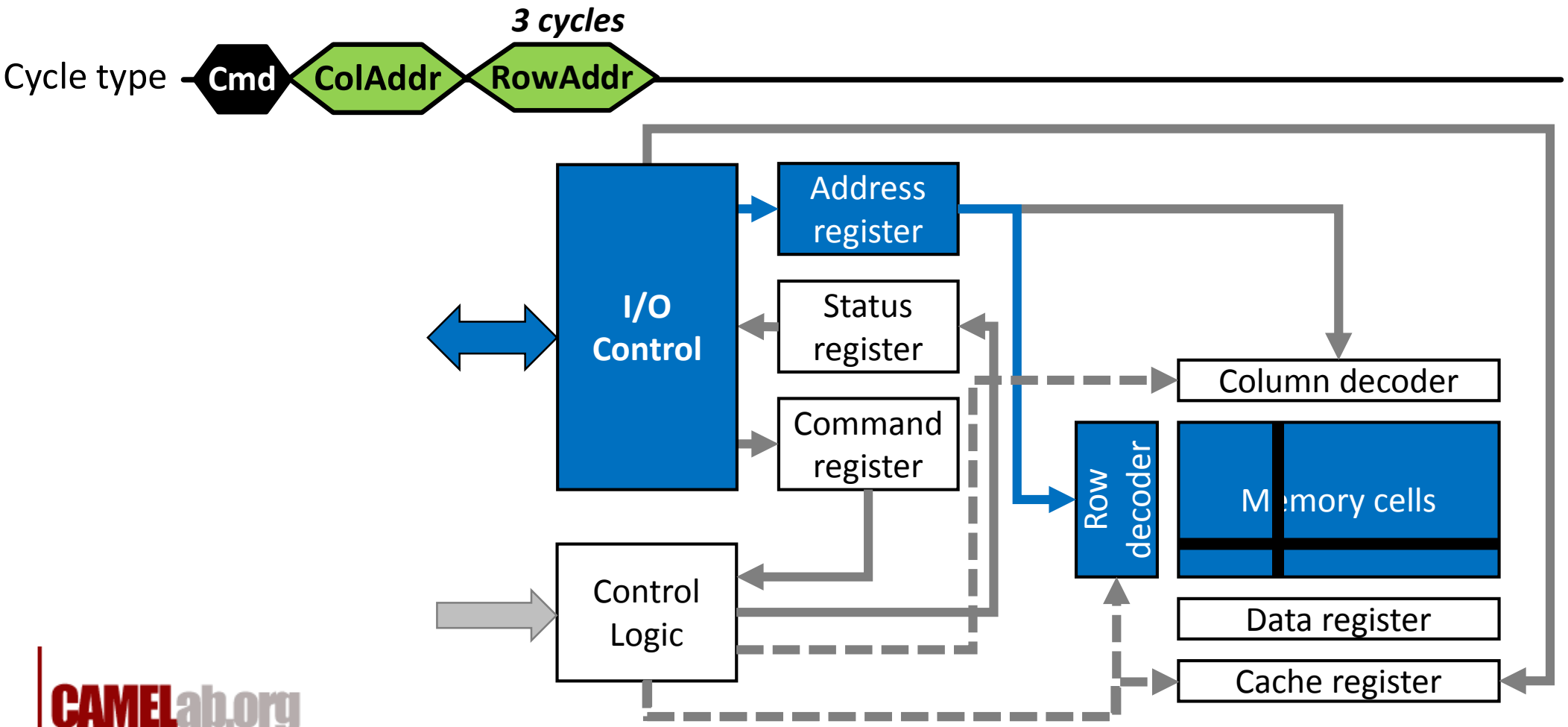
A Basic Write

- The first command is needed to indicate that this is a legacy page write
- The column address (C-ADDR) and row address (R-ADDR) are shipped in a serial order
- Data should be then transferred, and the second command initiates a write
- Once the write is performed, users need to send the last command to check up the status of target



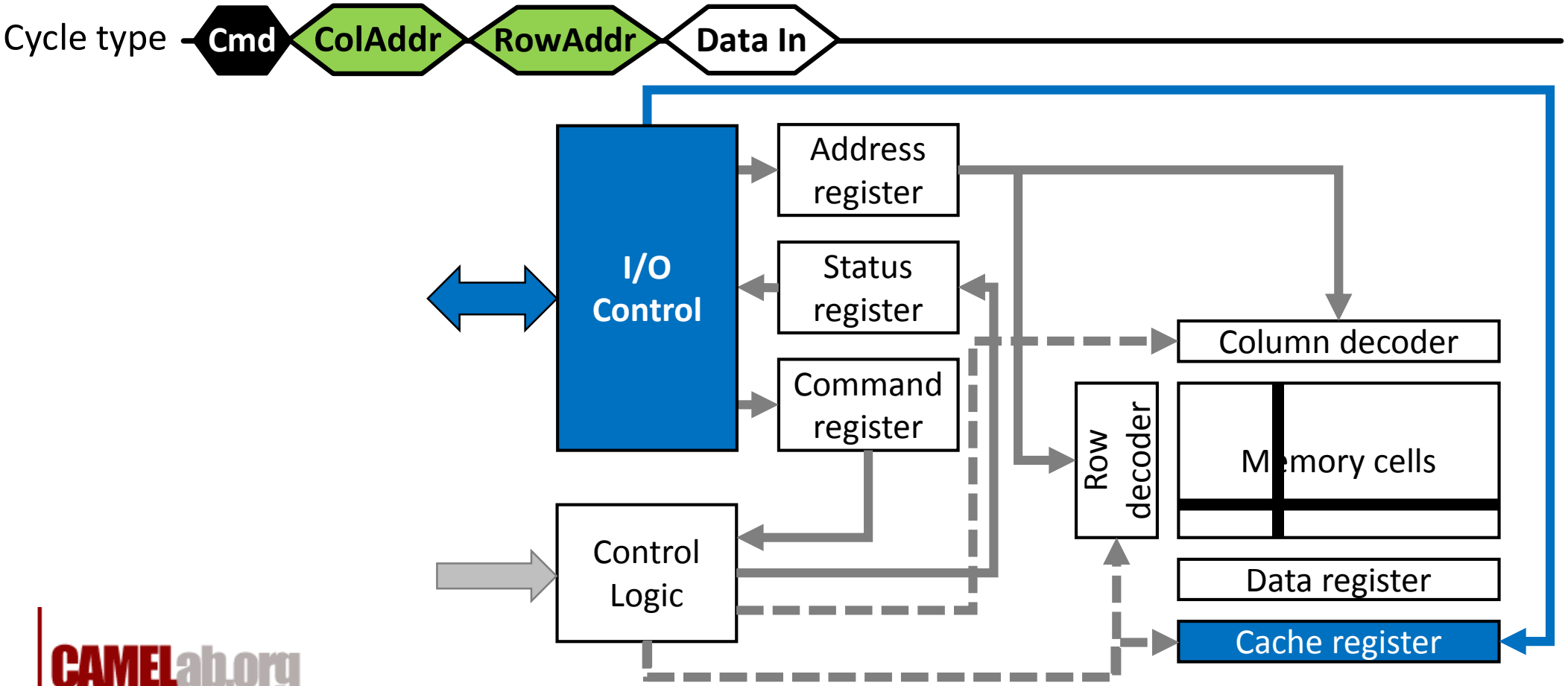
A Basic Write

- The first command is needed to indicate that this is a legacy page write
- The column address (C-ADDR) and row address (R-ADDR) are shipped in a serial order
- Data should be then transferred, and the second command initiates a write
- Once the write is performed, users need to send the last command to check up the status of target



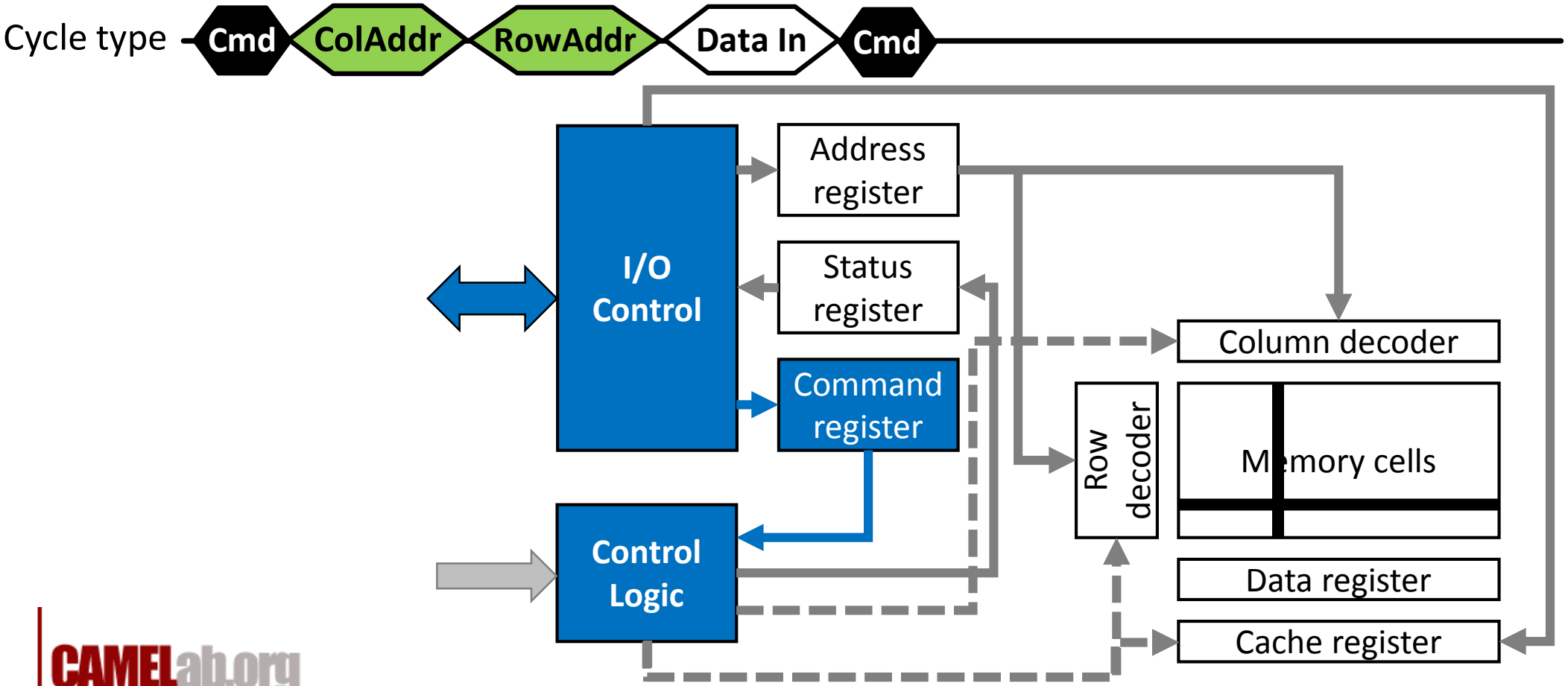
A Basic Write

- The first command is needed to indicate that this is a legacy page write
- The column address (C-ADDR) and row address (R-ADDR) are shipped in a serial order
- Data should be then transferred, and the second command initiates a write
- Once the write is performed, users need to send the last command to check up the status of target



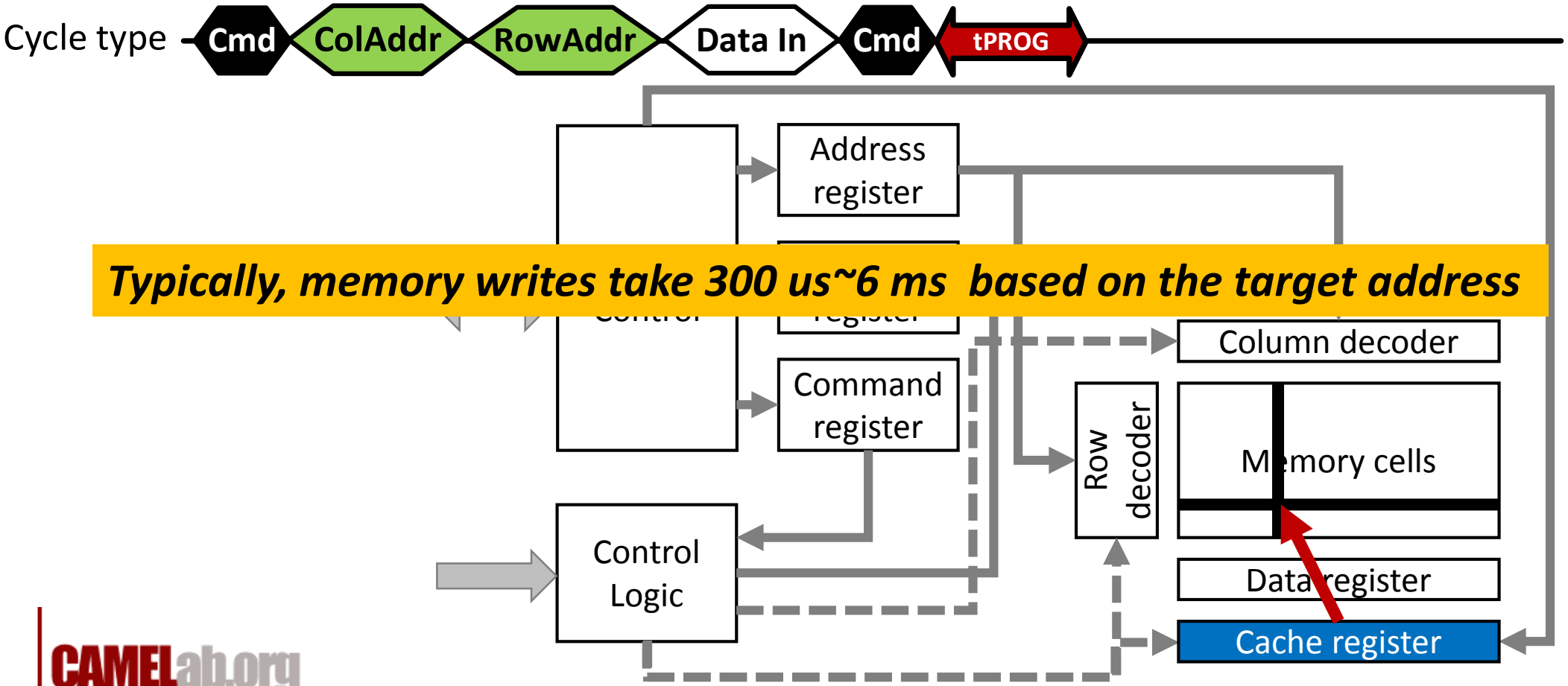
A Basic Write

- The first command is needed to indicate that this is a legacy page write
- The column address (C-ADDR) and row address (R-ADDR) are shipped in a serial order
- Data should be then transferred, and the second command initiates a write
- Once the write is performed, users need to send the last command to check up the status of target



A Basic Write

- The first command is needed to indicate that this is a legacy page write
- The column address (C-ADDR) and row address (R-ADDR) are shipped in a serial order
- Data should be then transferred, and the second command initiates a write
- Once the write is performed, users need to send the last command to check up the status of target

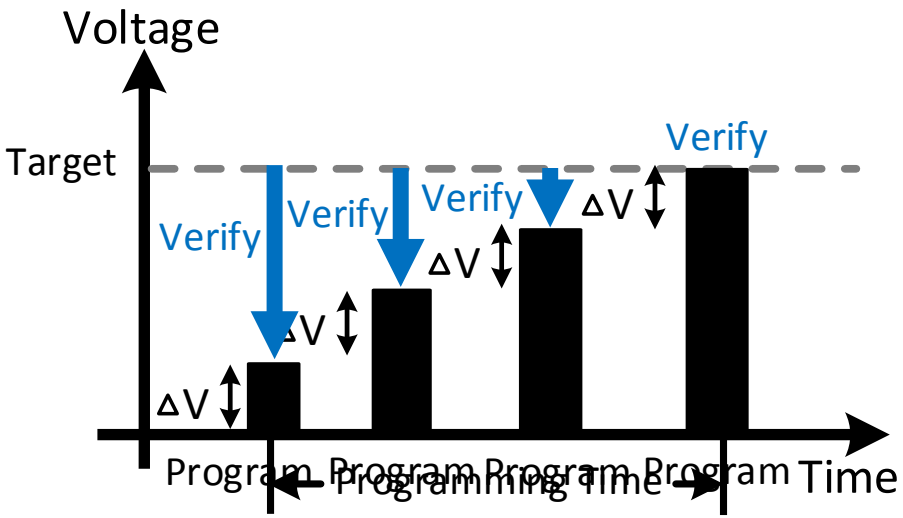


Intrinsic Latency Variation

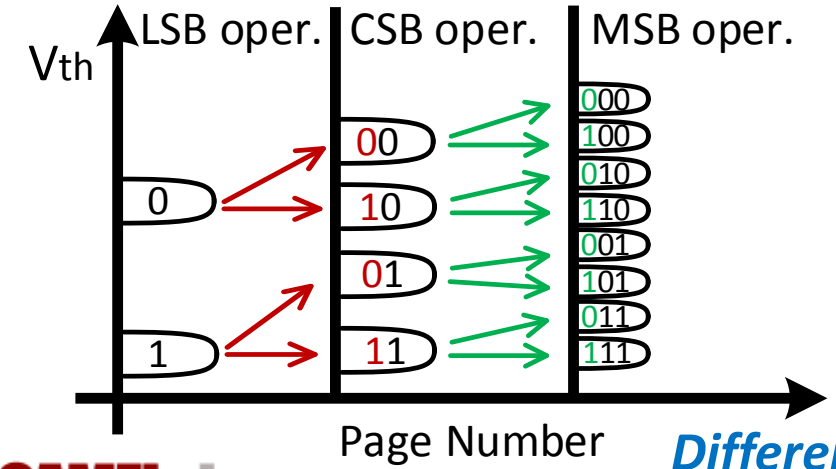
- Incremental step pulse programming (ISPP)
- Fowler-Nordheim Tunneling
  - Making an electron channel
  - Voltage is applied over a certain threshold

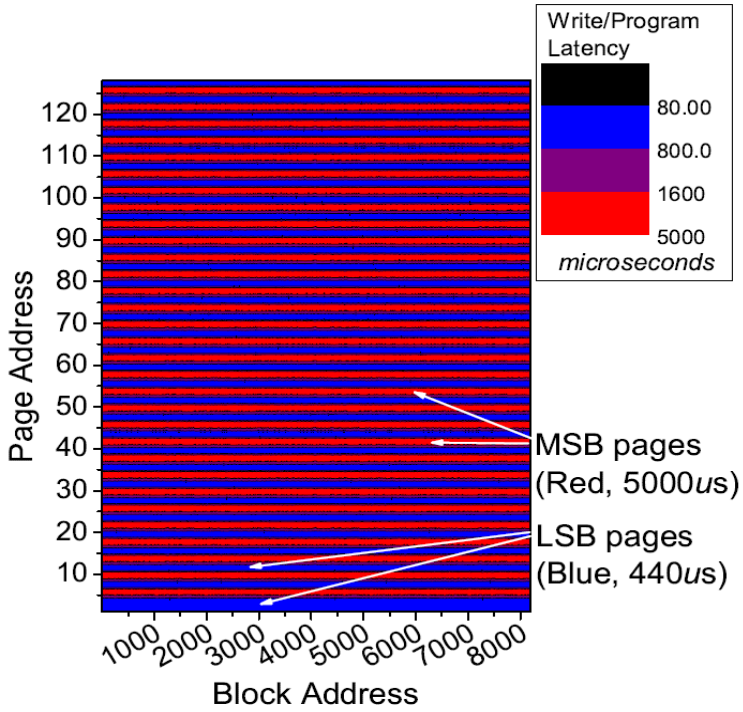
Intrinsic Latency Variation

- Incremental step pulse programming (ISPP)



Ex) TLC NAND programming



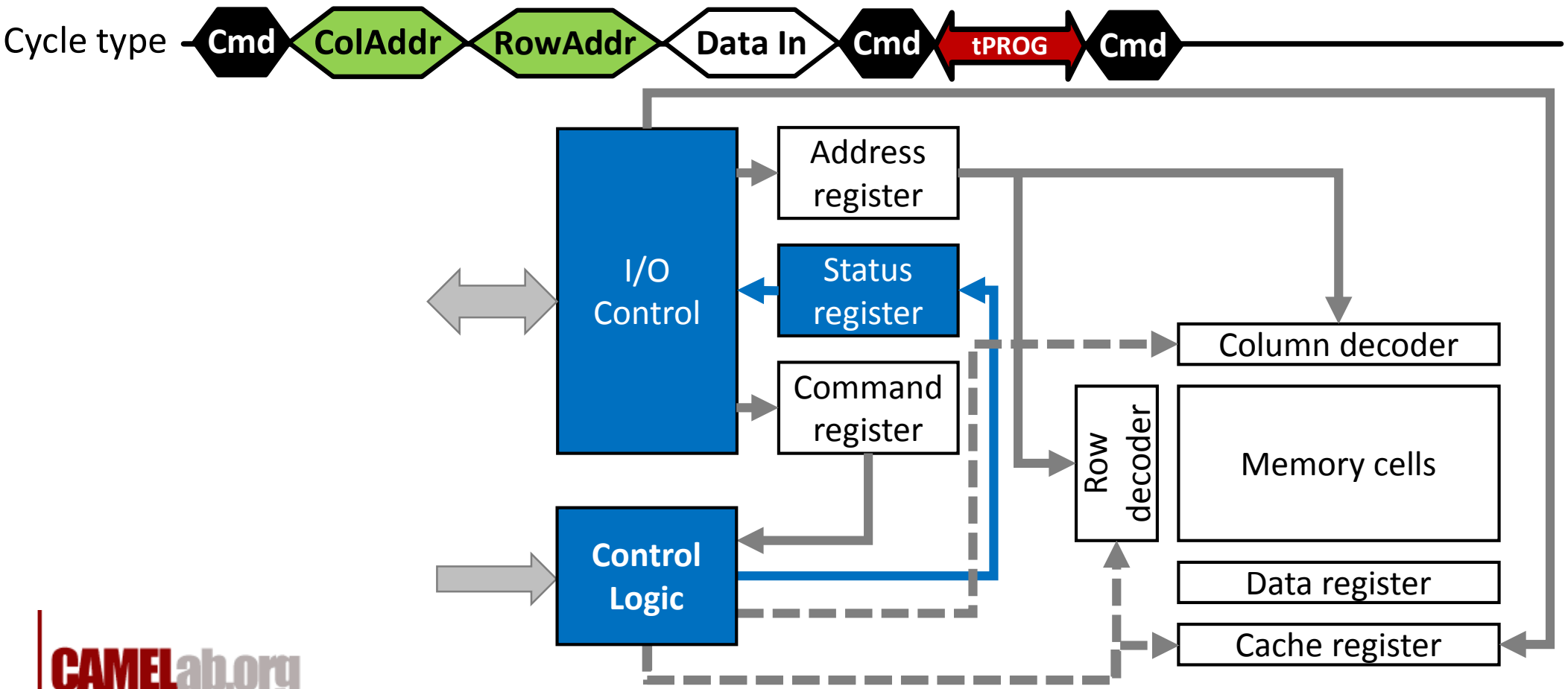


Fluctuating NAND flash latency (Address of the pages in a block)

Different programming time for page types (LSB,CSB,MSB)

A Basic Write

- The first command is needed to indicate that this is a legacy page write
- The column address (C-ADDR) and row address (R-ADDR) are shipped in a serial order
- Data should be then transferred, and the second command initiates a write
- Once the write is performed, users need to send the last command to check up the status of target

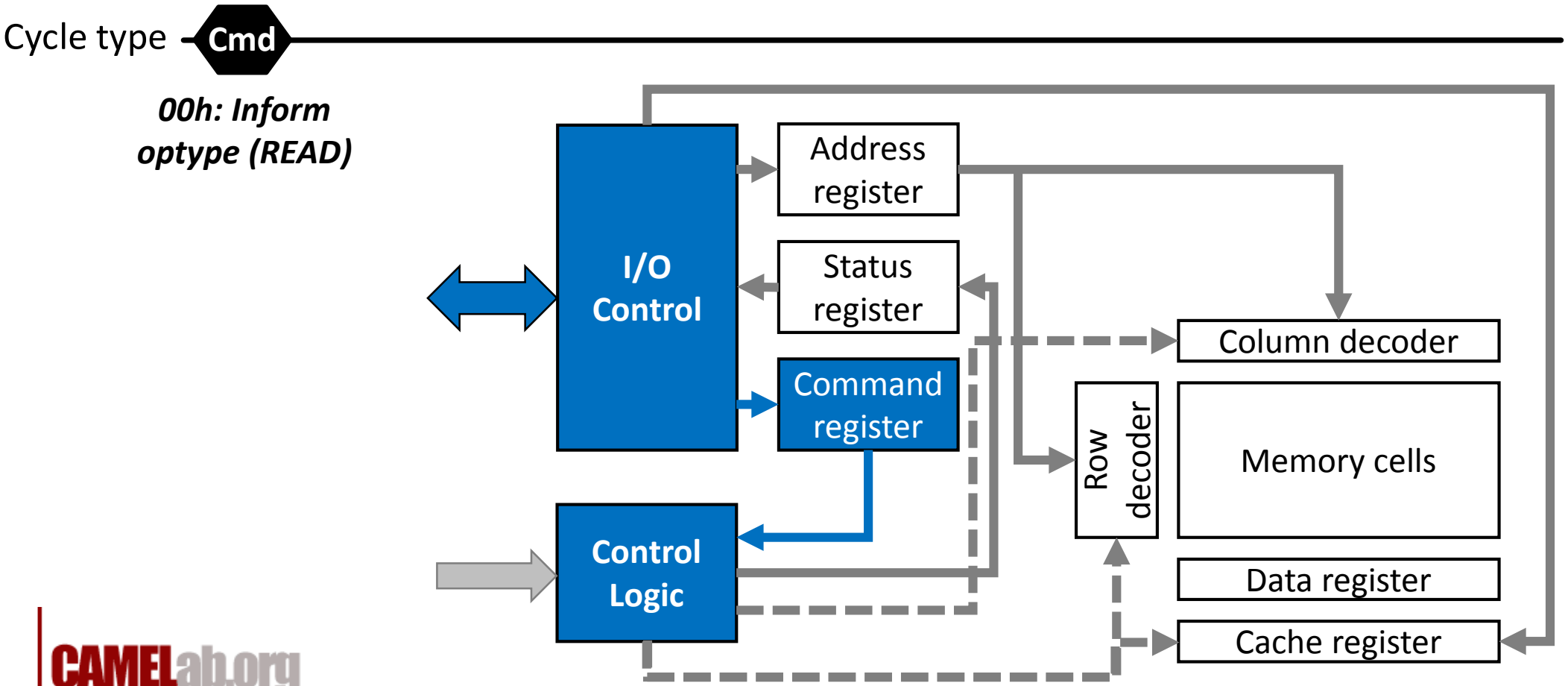


A Basic Write

- Unlike other memory technologies, flash controller(s) should hold the data till the target return a success state.
- If it fails, the controller should retry the write till the target reports a success
  - This indicates that buffer or queue mechanisms within SSDs should appropriately manage the data based on the corresponding flash interface protocol that system employs

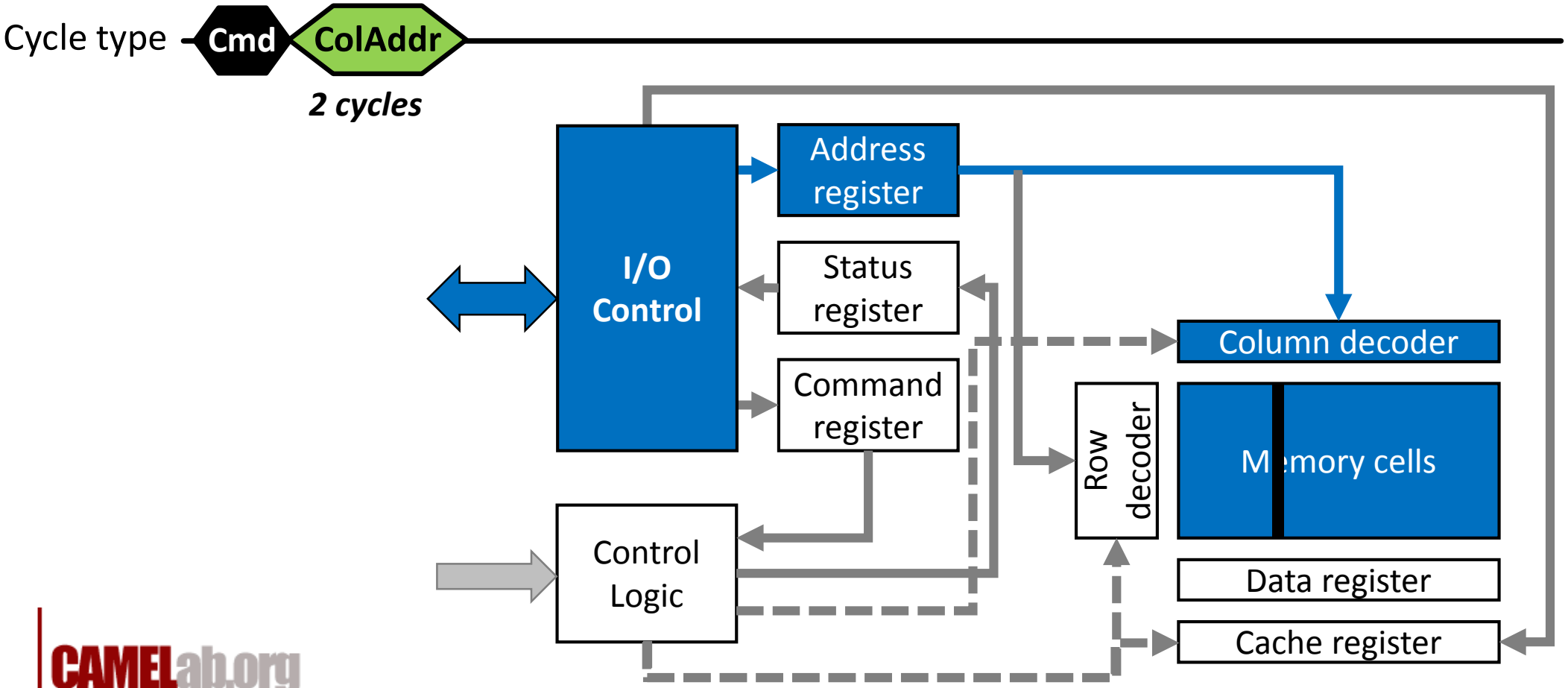
A Basic Read

- The first command and addresses follow the same sequence of a write
- The second command initiates a read
- A read occurs
- Then the data comes out



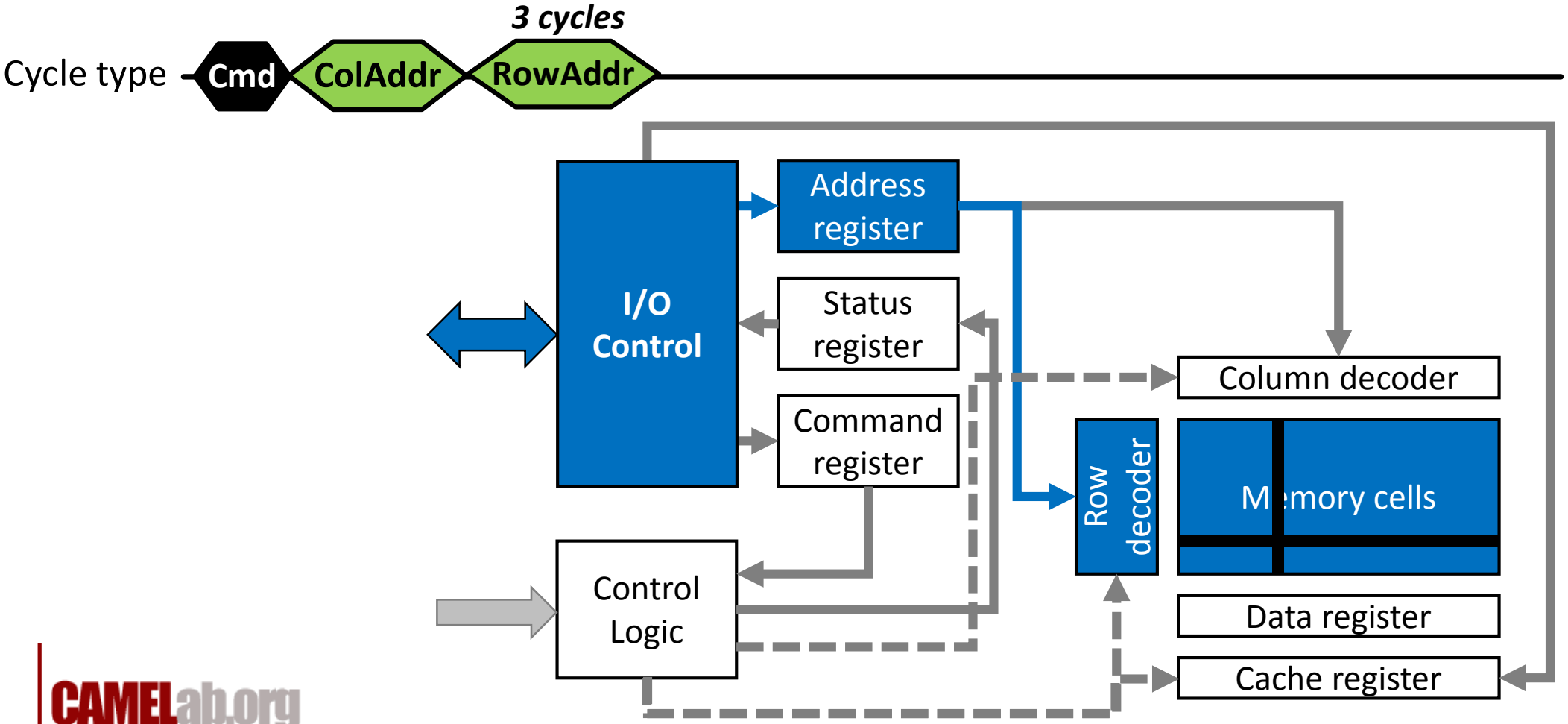
A Basic Read

- The first command and addresses follow the same sequence of a write
- The second command initiates a read
- A read occurs
- Then the data comes out



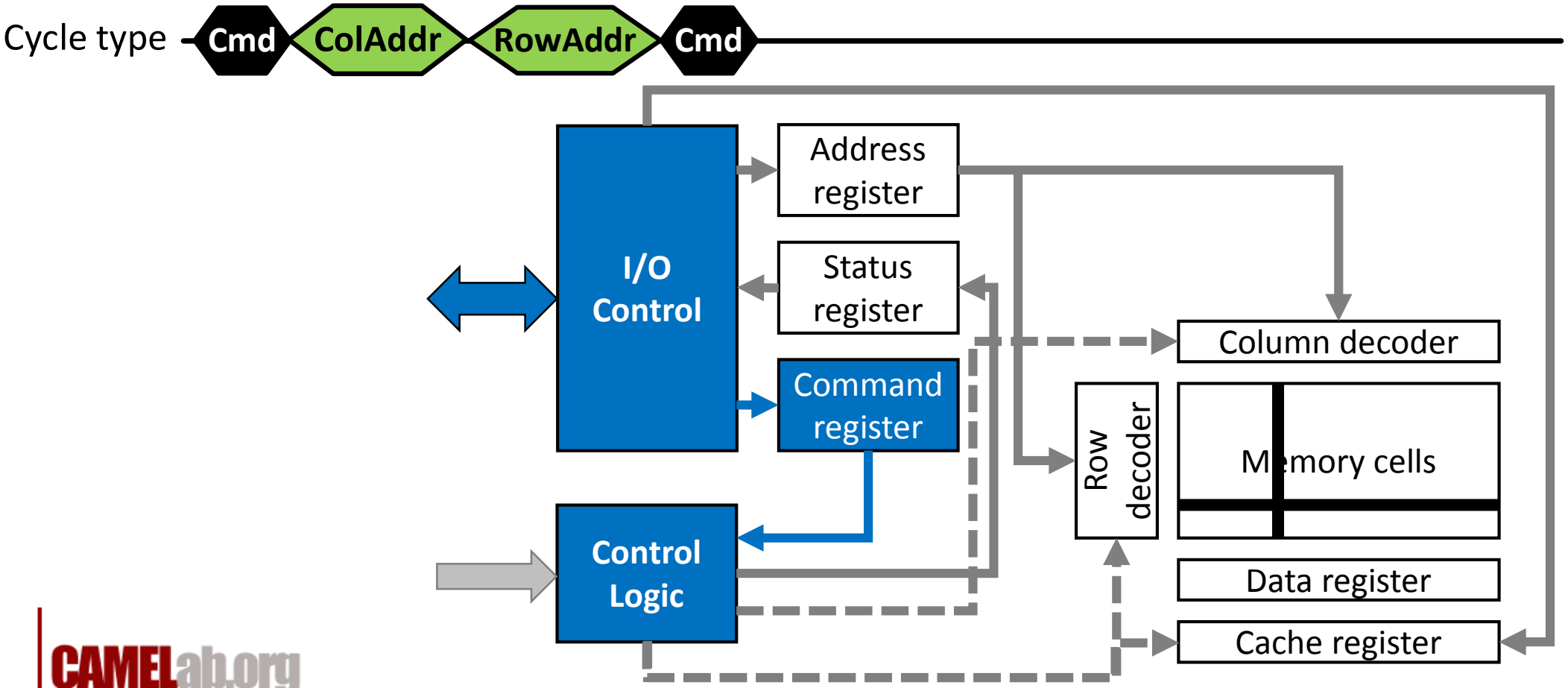
A Basic Read

- The first command and addresses follow the same sequence of a write
- The second command initiates a read
- A read occurs
- Then the data comes out



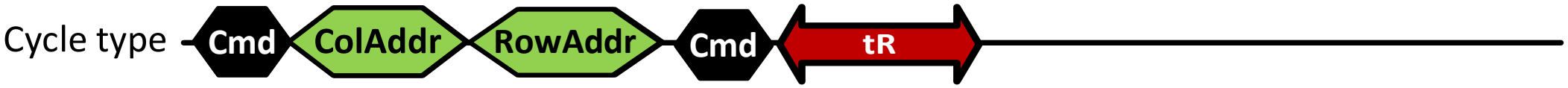
A Basic Read

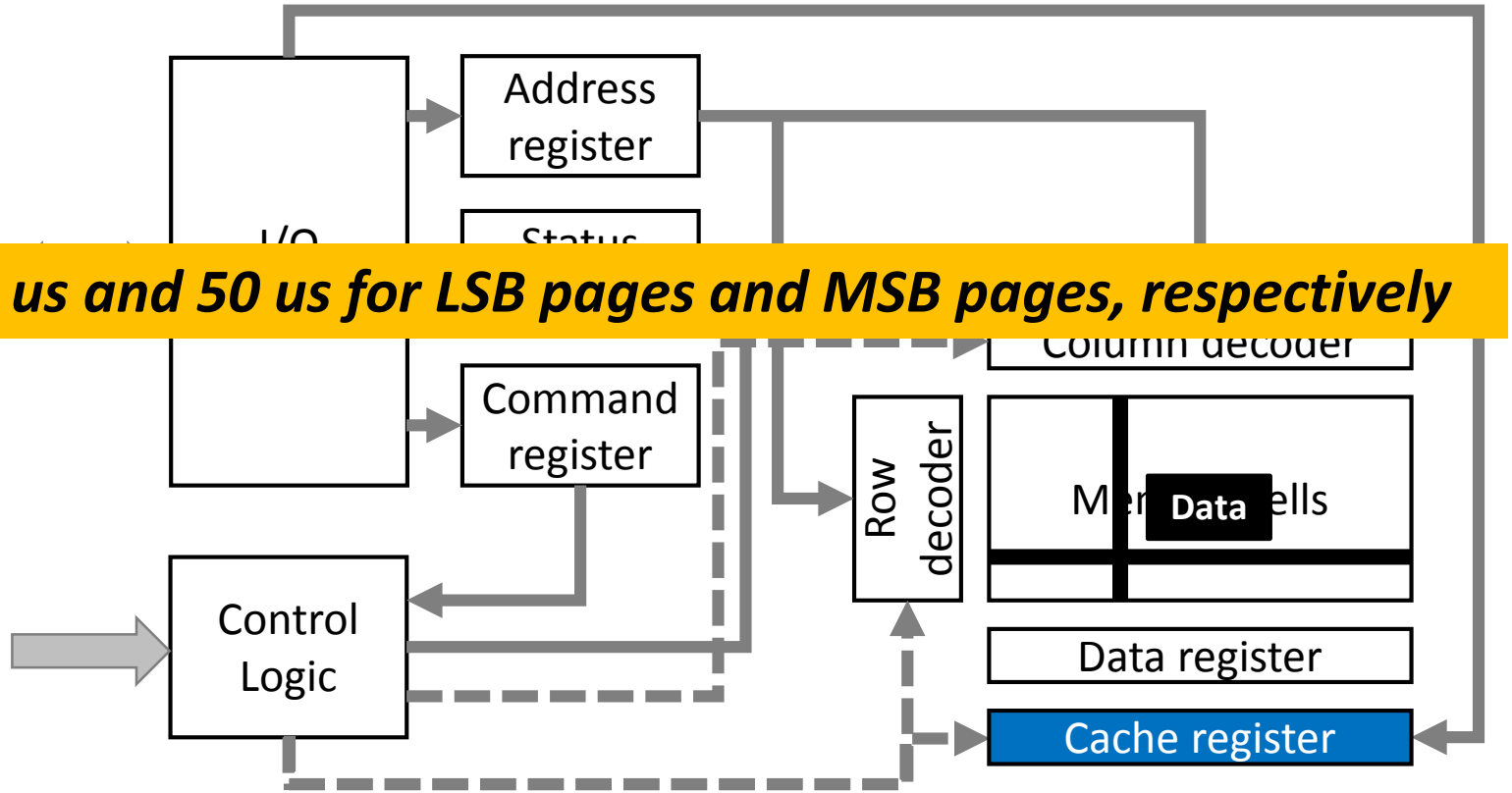
- The first command and addresses follow the same sequence of a write
- The second command initiates a read
- A read occurs
- Then the data comes out



A Basic Read

- The first command and addresses follow the same sequence of a write
- The second command initiates a read
- A read occurs
- Then the data comes out

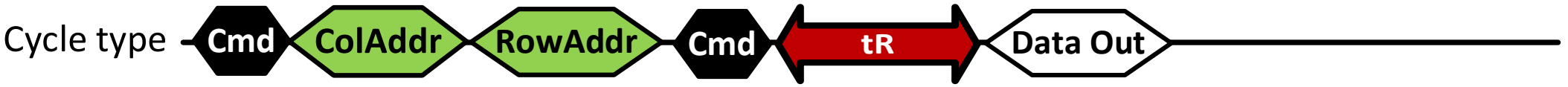


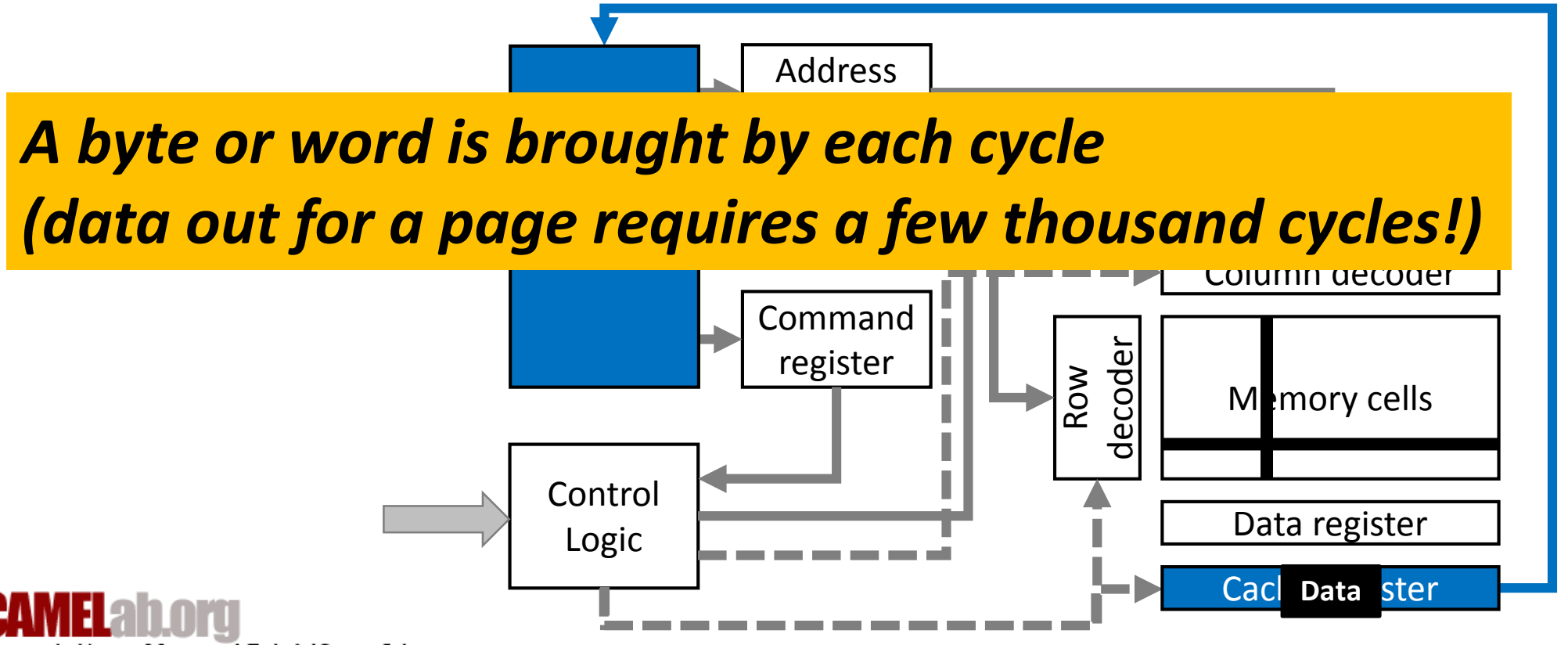


Typically, it takes 25 us and 50 us for LSB pages and MSB pages, respectively

A Basic Read

- The first command and addresses follow the same sequence of a write
- The second command initiates a read
- A read occurs
- Then the data comes out





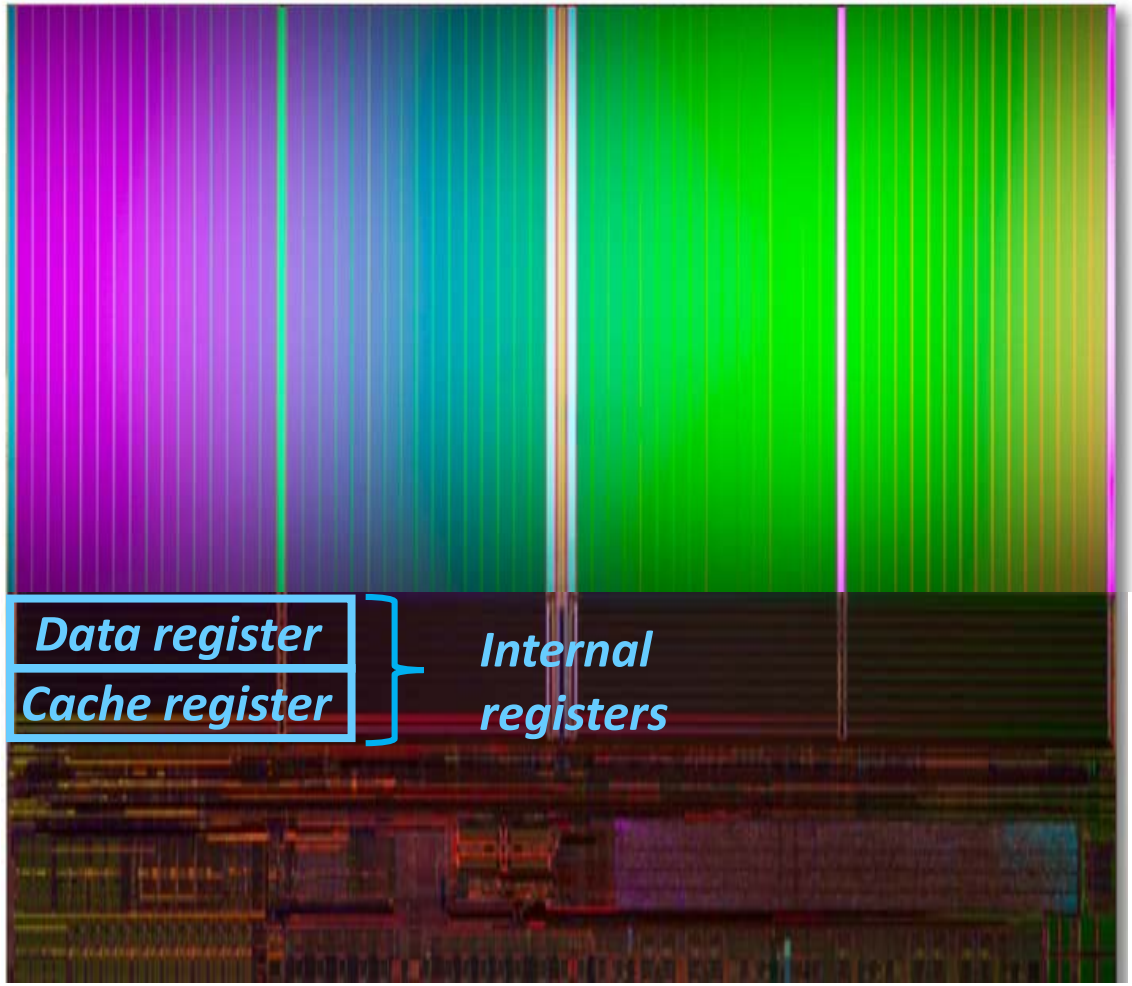
Outline

- Flash Microarchitecture
- Basic Operations (Legacy)
- Cache Mode Operations
- Multi-plane Mode Operations
- Copy-back Operations
- Erase Operations
- Evaluation Studies
  - Simulation-based studies
  - Real implementation (FGPA) and evaluation

Cache Operation

- The cache mode operations leverage internal registers in an attempt to hide performance overheads imposed by data movements

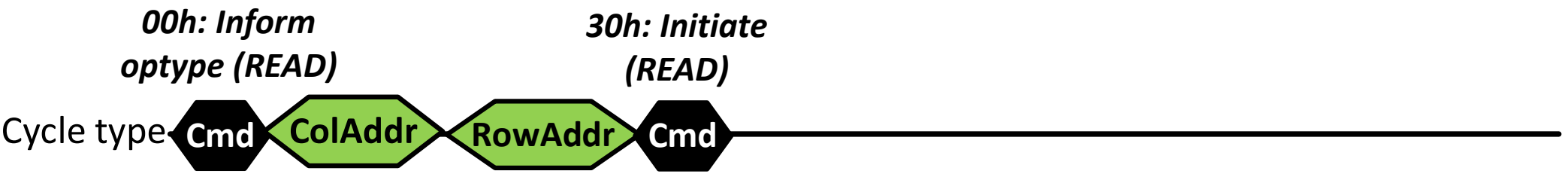
data2
data1

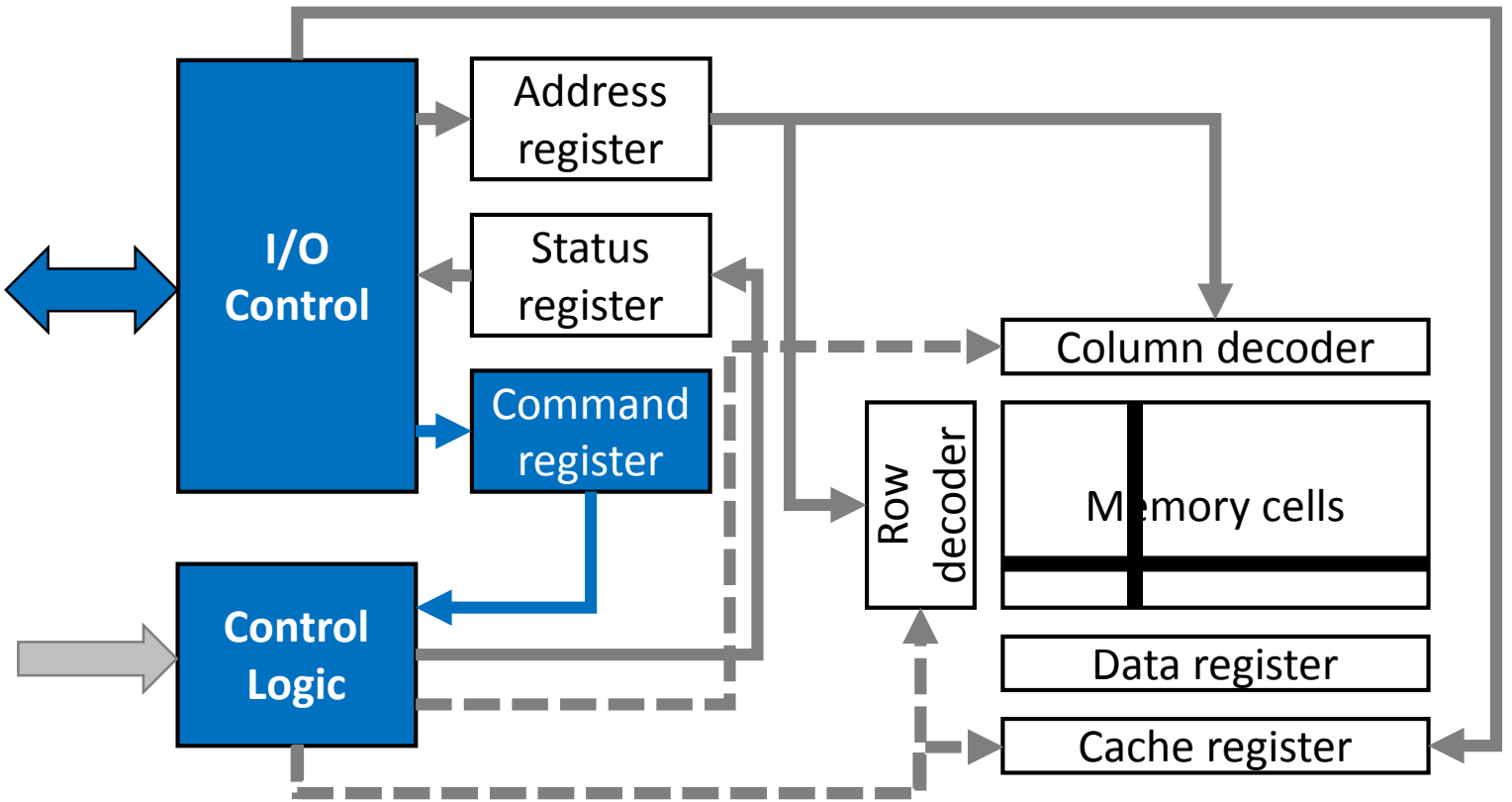


Cache Mode (Reads)

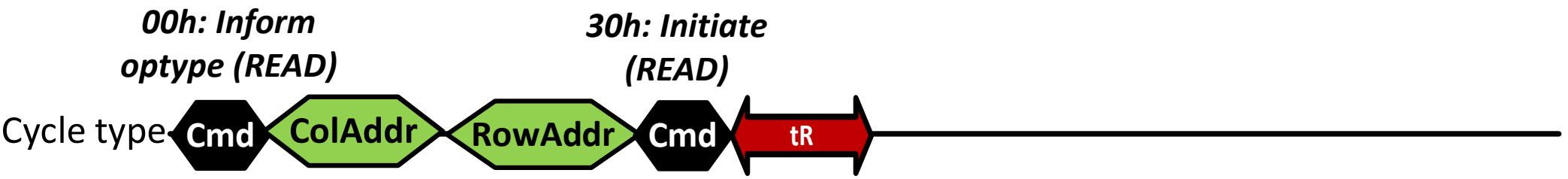
- A sequential cache operation reads a next sequential page by using a data register while the data of the previous page is in transferring from a plane to a cache register
- A cache command should be issued just before the data out occurs
- Random caching is also available by placing the target address in stead of the sequential command

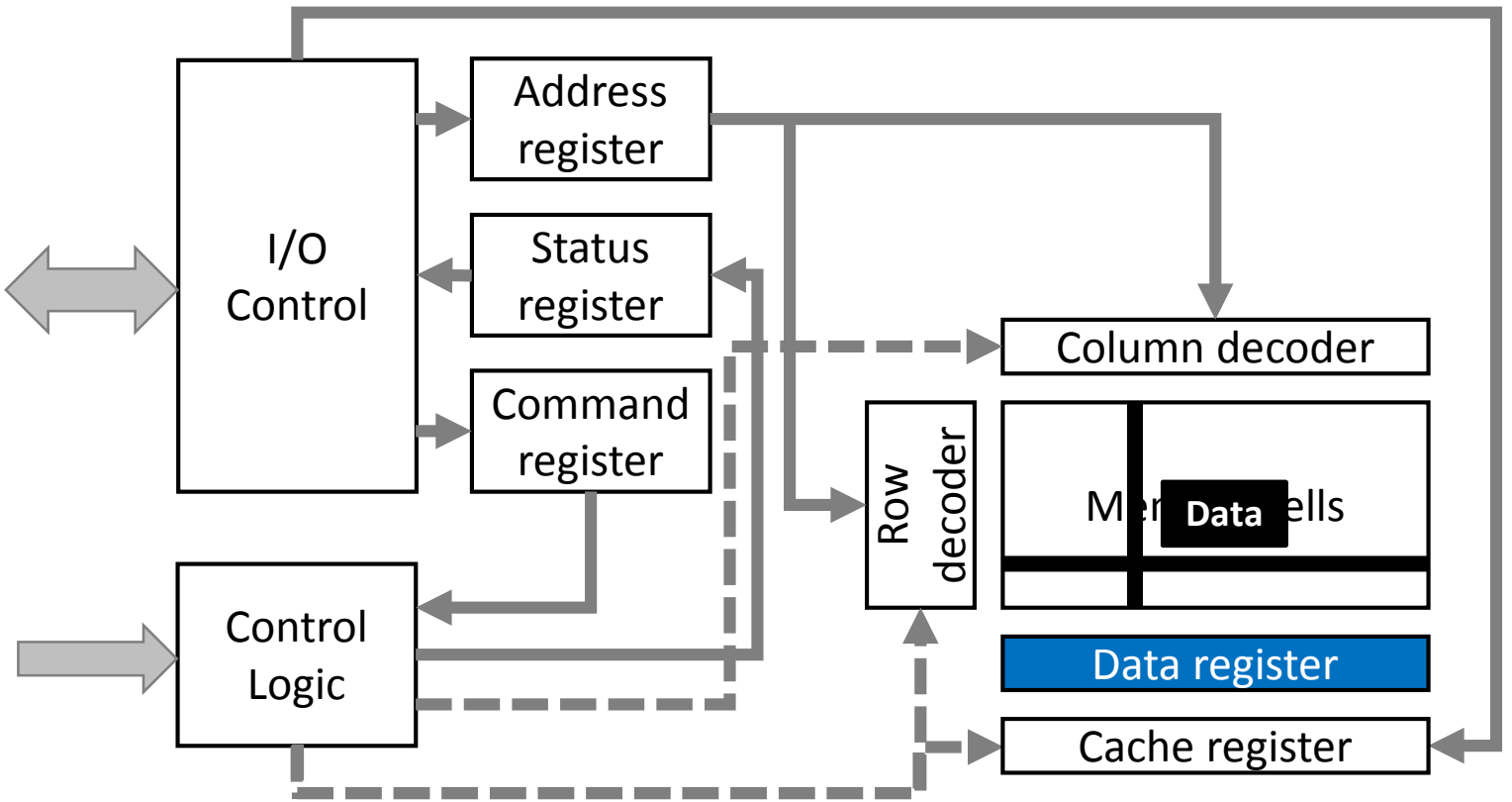
Cache Mode (Reads)





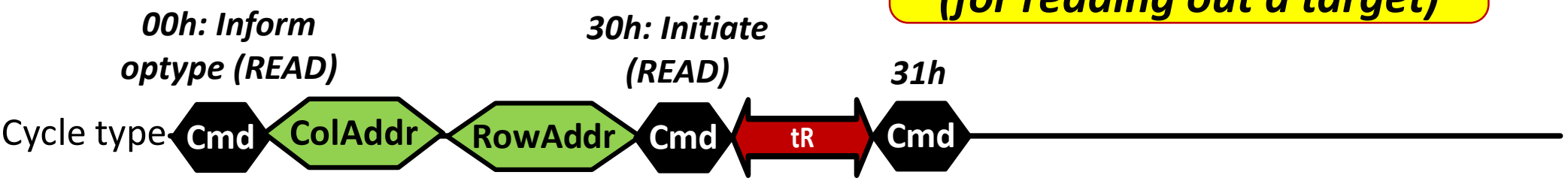
Cache Mode (Reads)

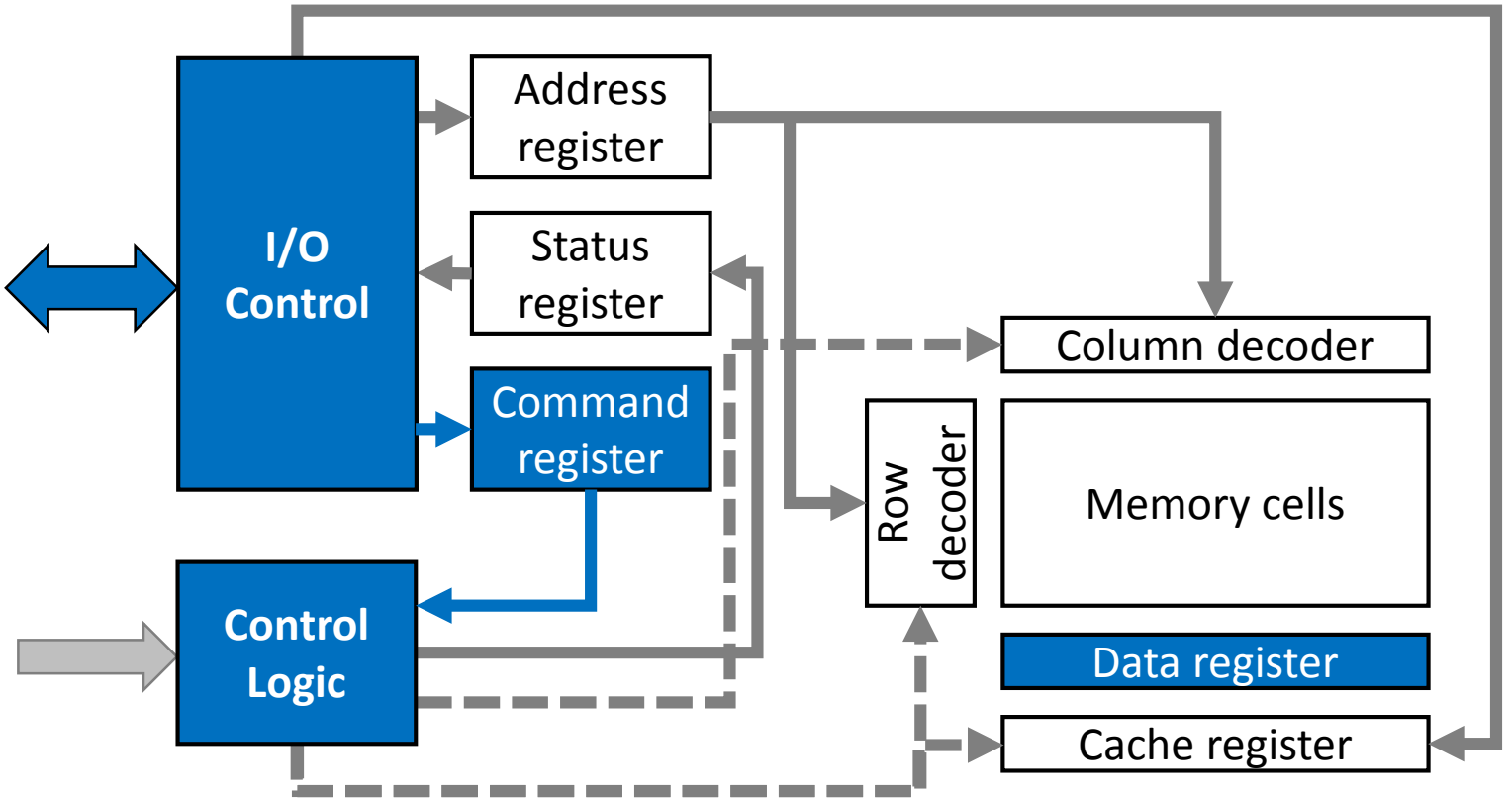




Cache Mode (Reads)

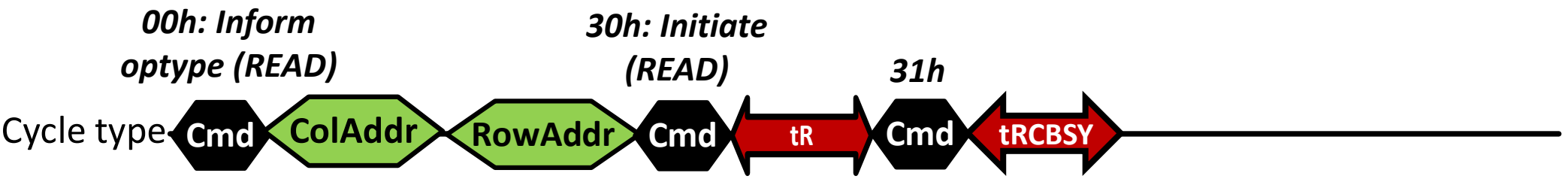
31h: No address is required (for reading out a target)

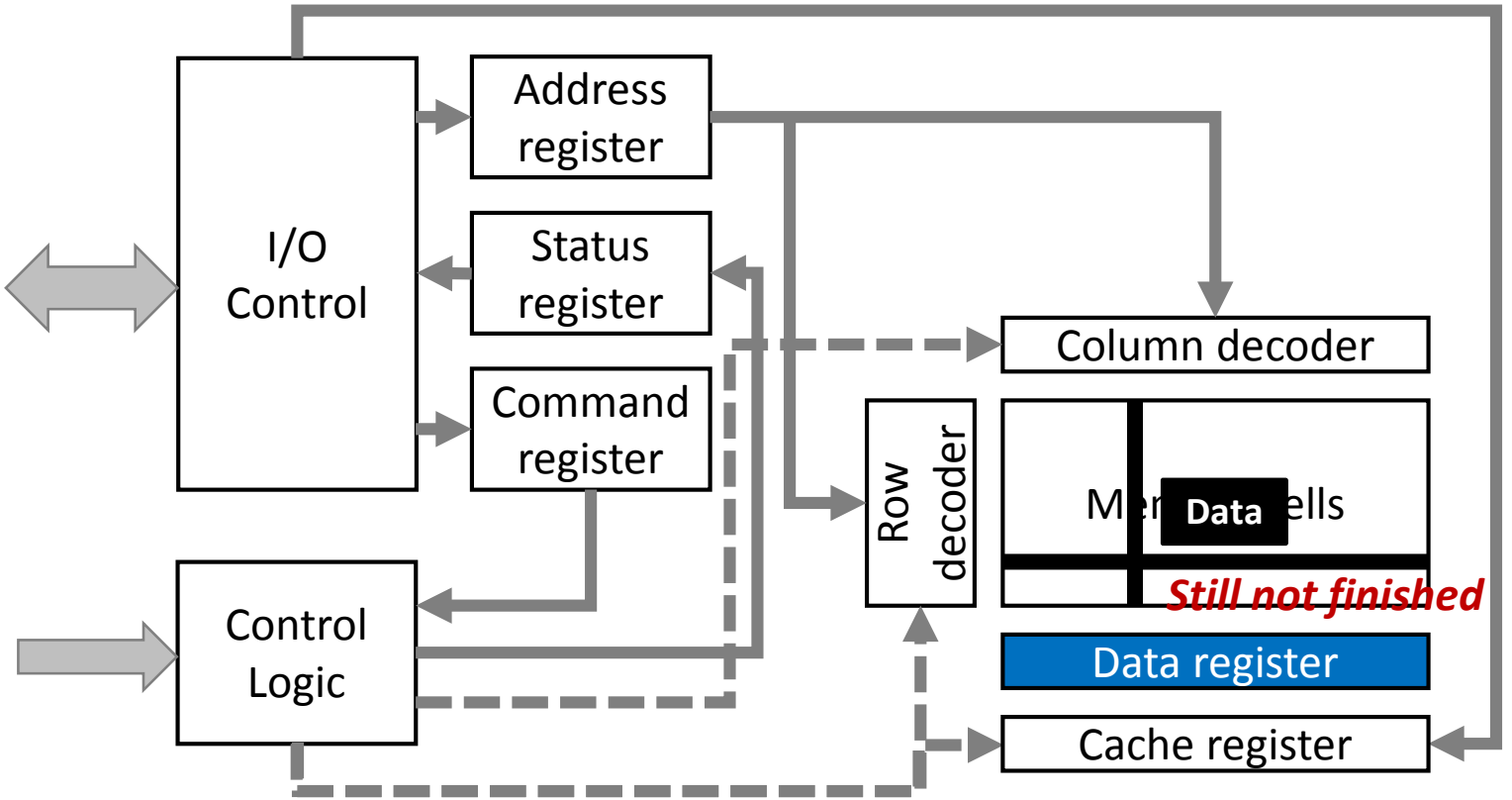




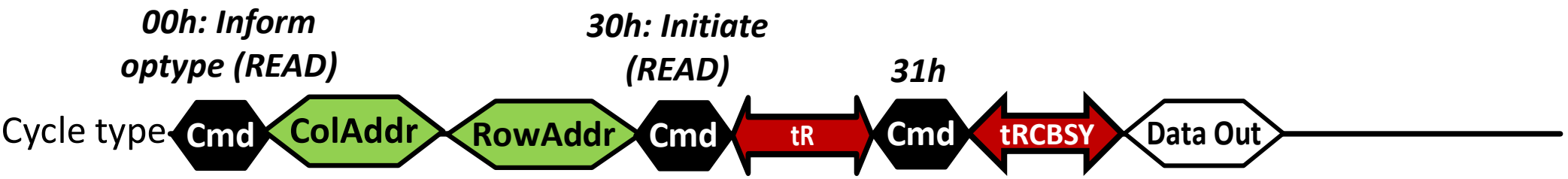
Cache Mode (Reads)

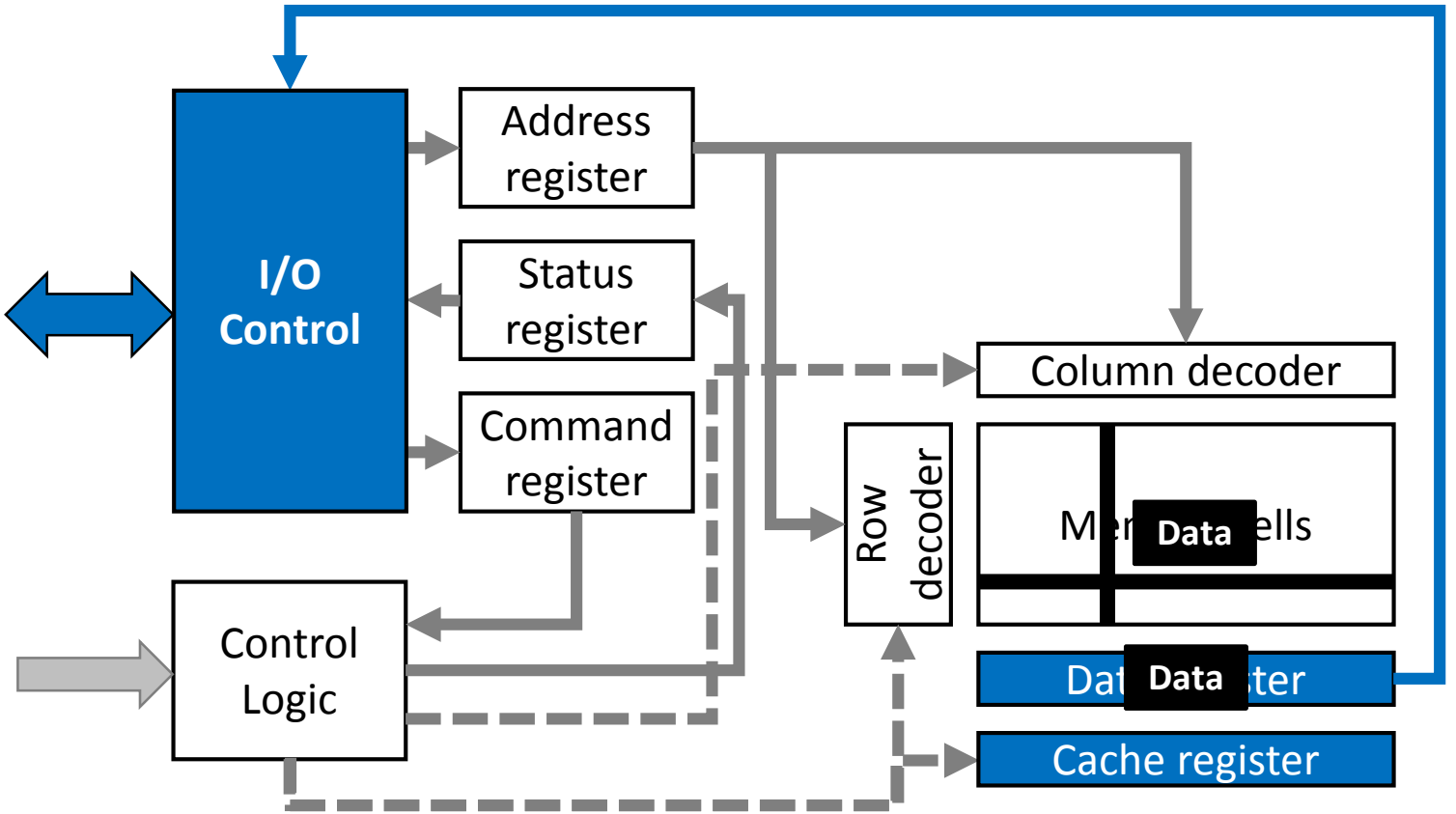
The second tR is invisible (as it is overlapped with data transfers)





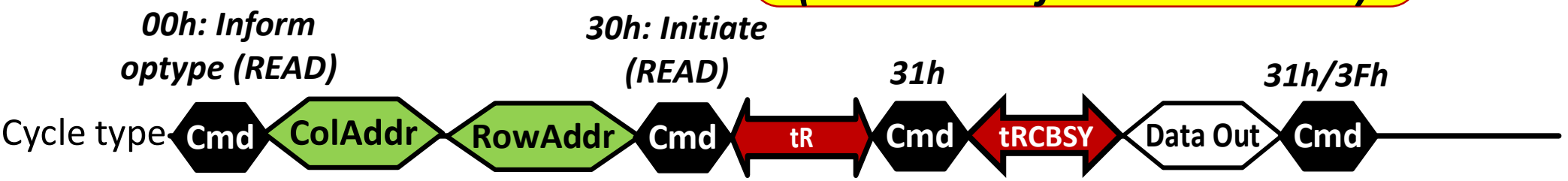
Cache Mode (Reads)





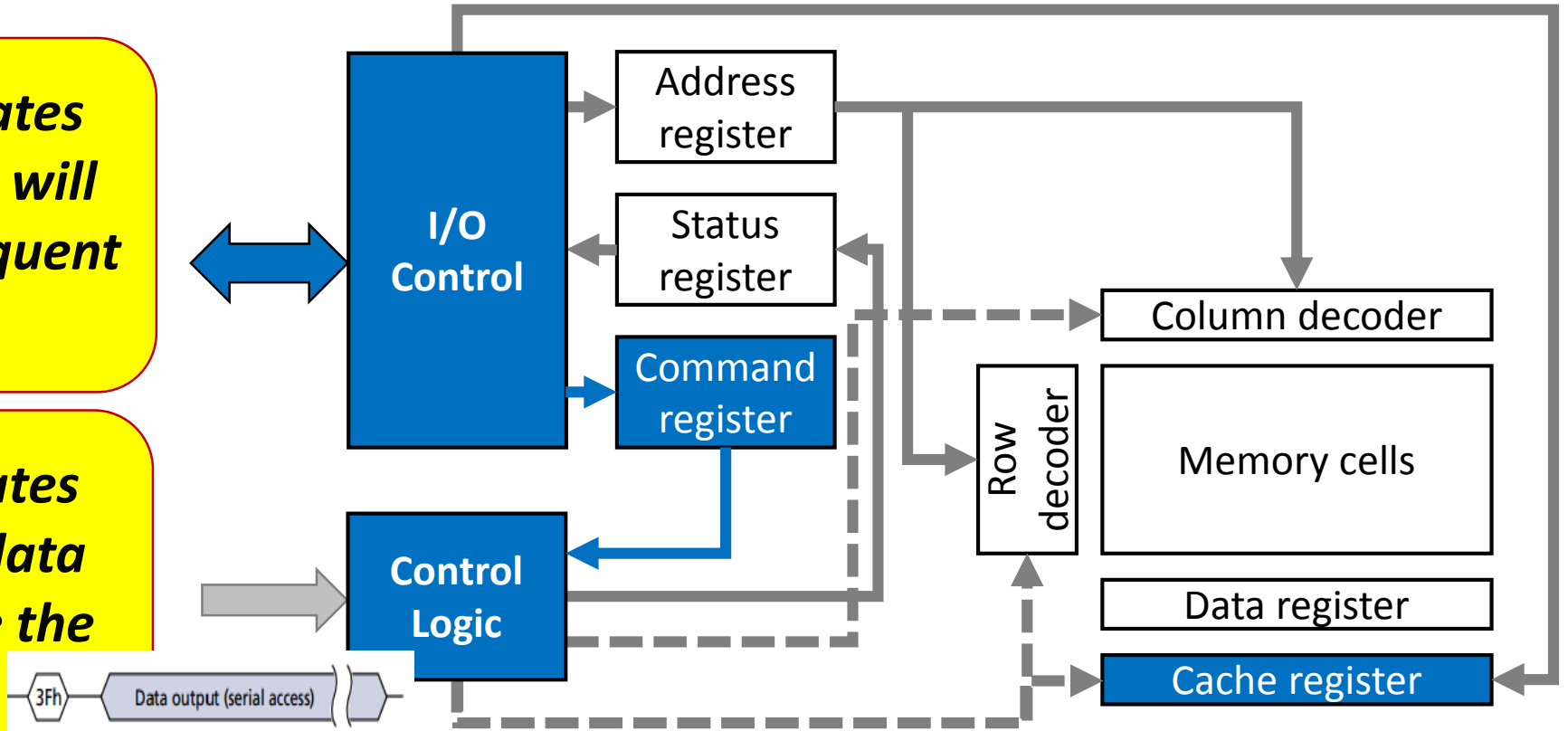
Cache Mode (Reads)

The selection of a register is controlled by control logic (invisible to flash controllers)

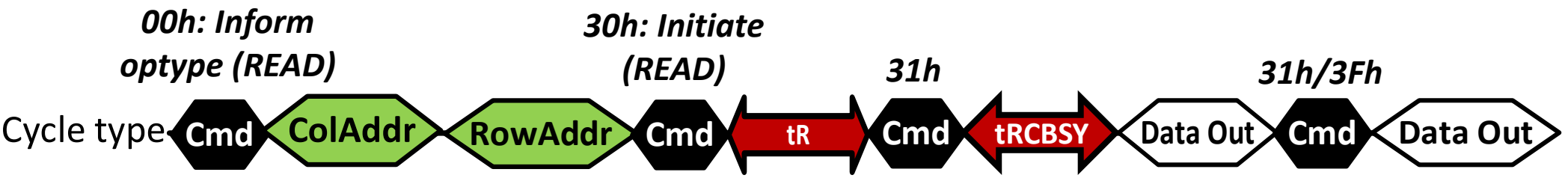


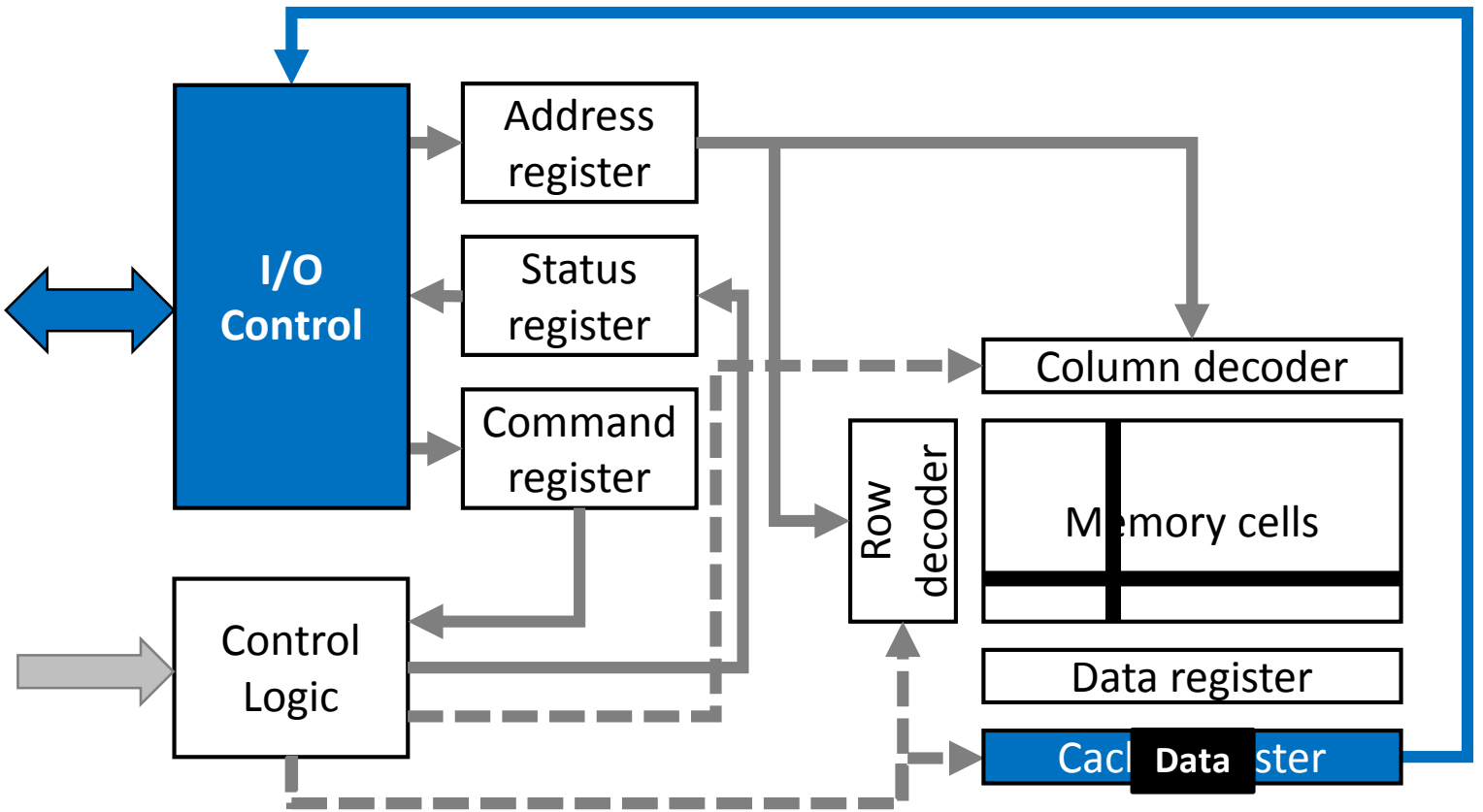
31h indicates that there will be a subsequent read

3Fh indicates that this data out will be the last



Cache Mode (Reads)

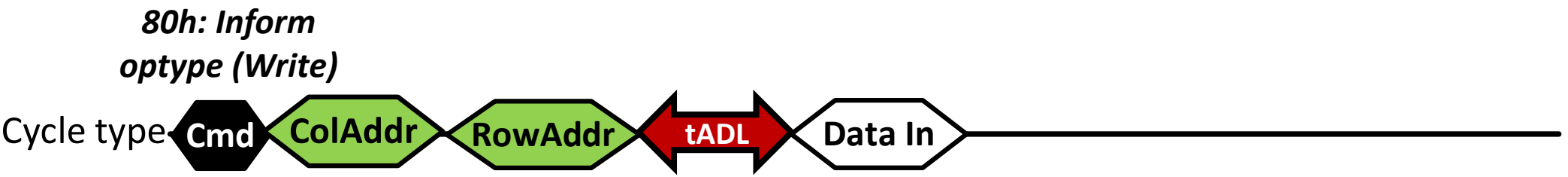


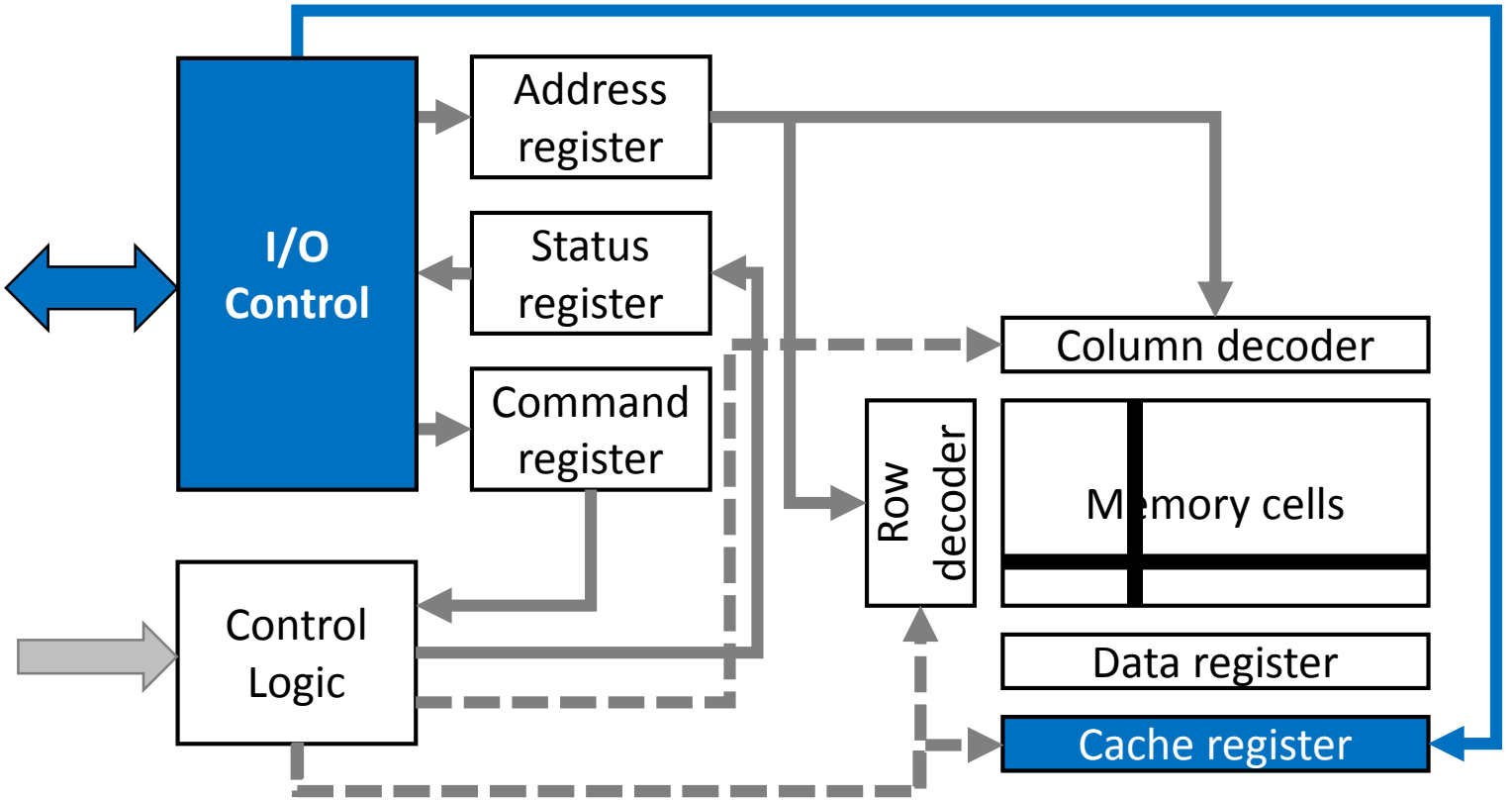


Cache Mode (Writes)

- Hide long latency of memory program time
  - Enable flash controller(s) to transfer data to a cache register while the target is in programming at some extent
  - Internal control logic copies the data from the cache register to the data register
    - No programming operation is involved here, which can be done as quickly as a register copy
  - The logic internally moves the target data sitting on the data register to a specified block and page addresses in the target array of selected die
    - In the meantime, the flash controller is able to transfer data to the cache register (or another one)

Cache Mode (Writes)

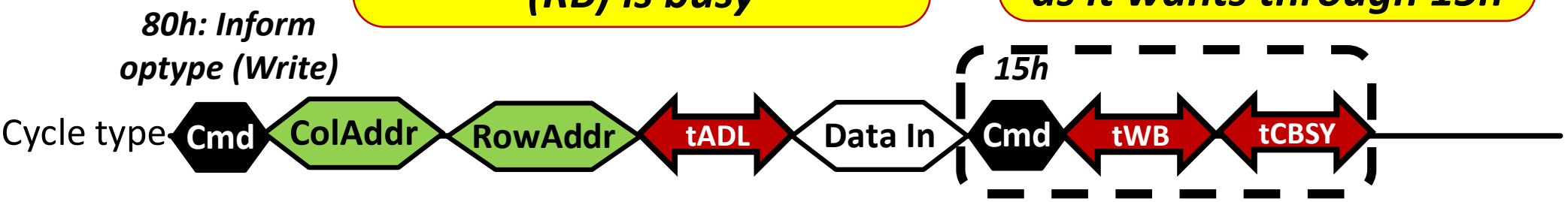


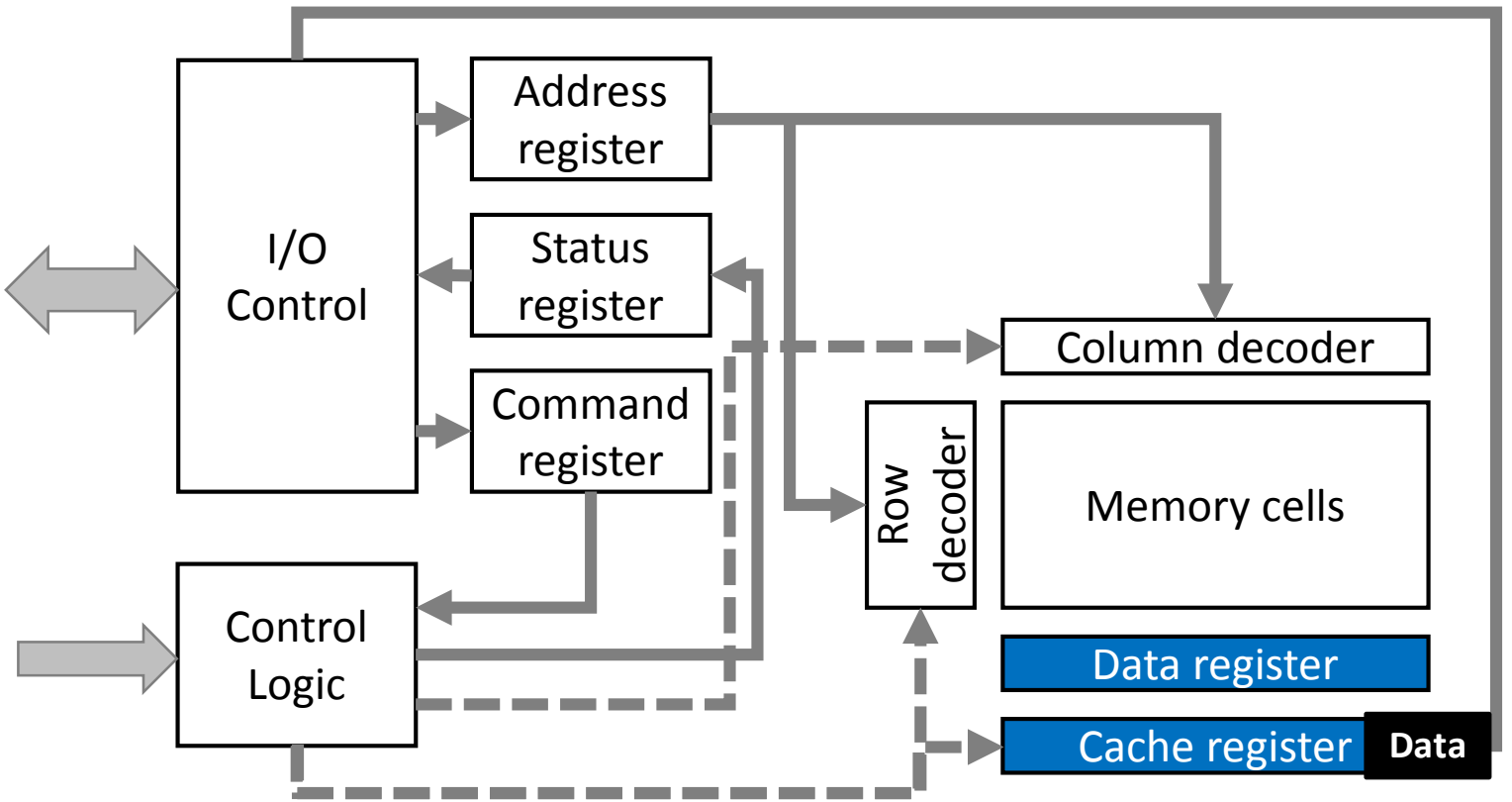


Cache Mode (Writes)

During this time, the status of the target chip (RB) is busy

A flash controller can program data as many as it wants through 15h

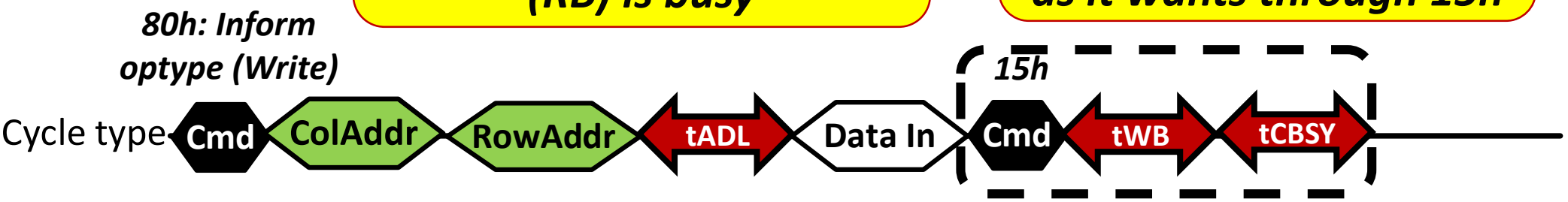


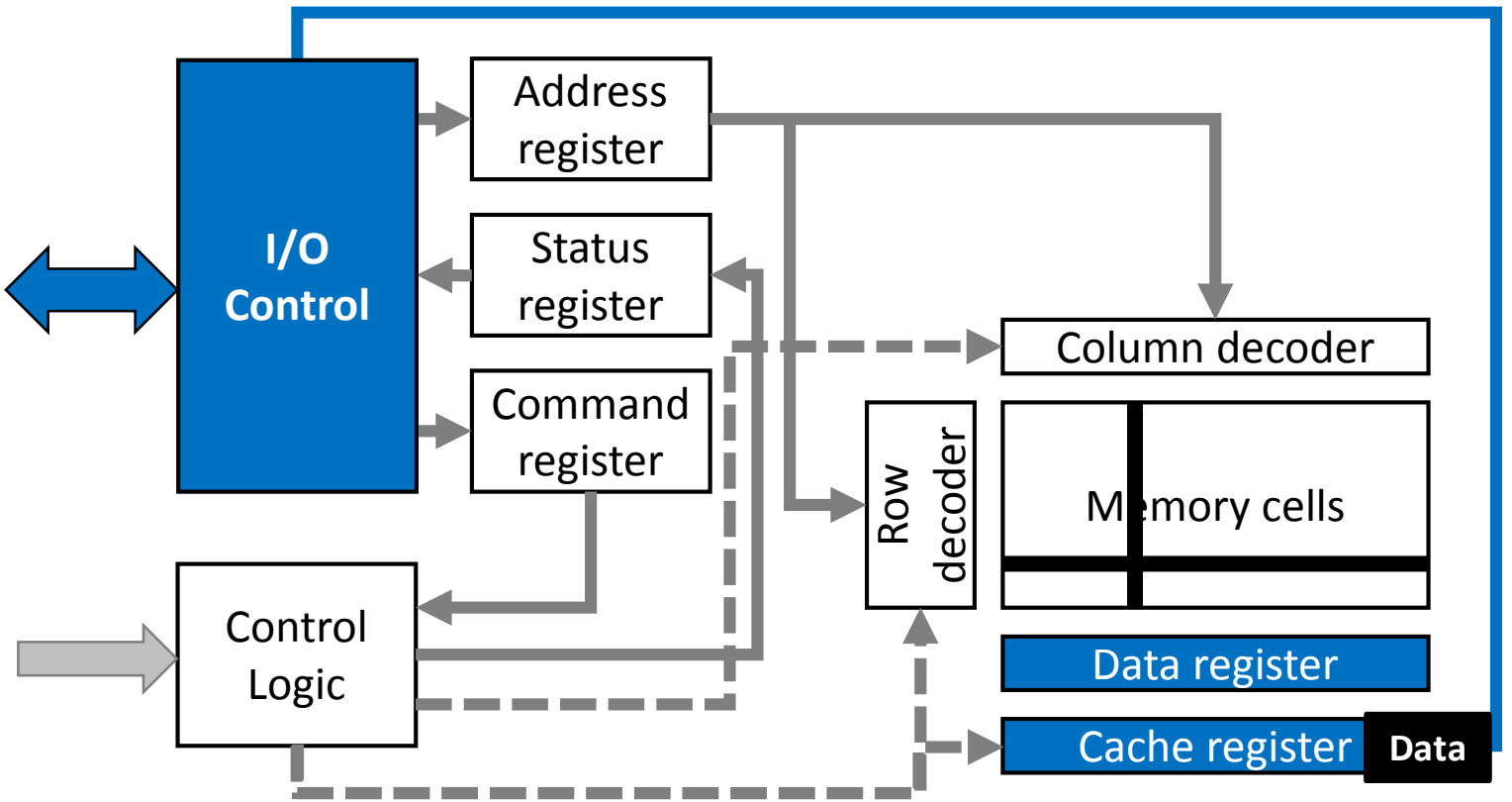


Cache Mode (Writes)

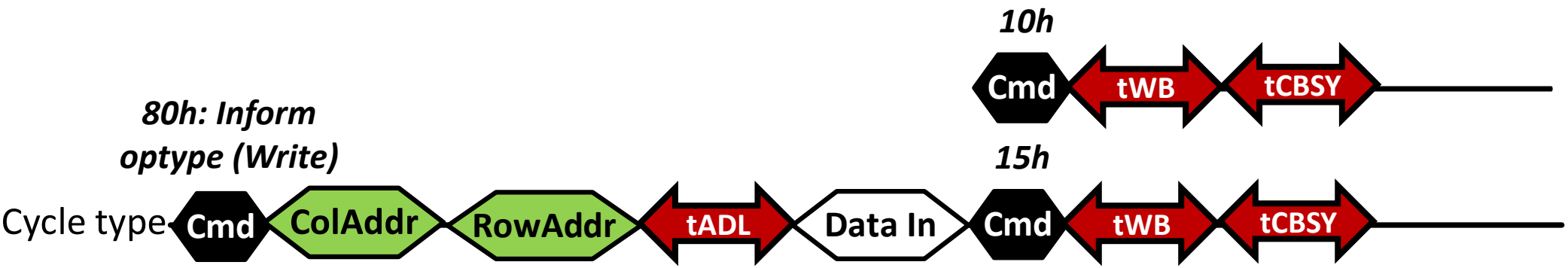
During this time, the status of the target chip (RB) is busy

A flash controller can program data as many as it wants through 15h



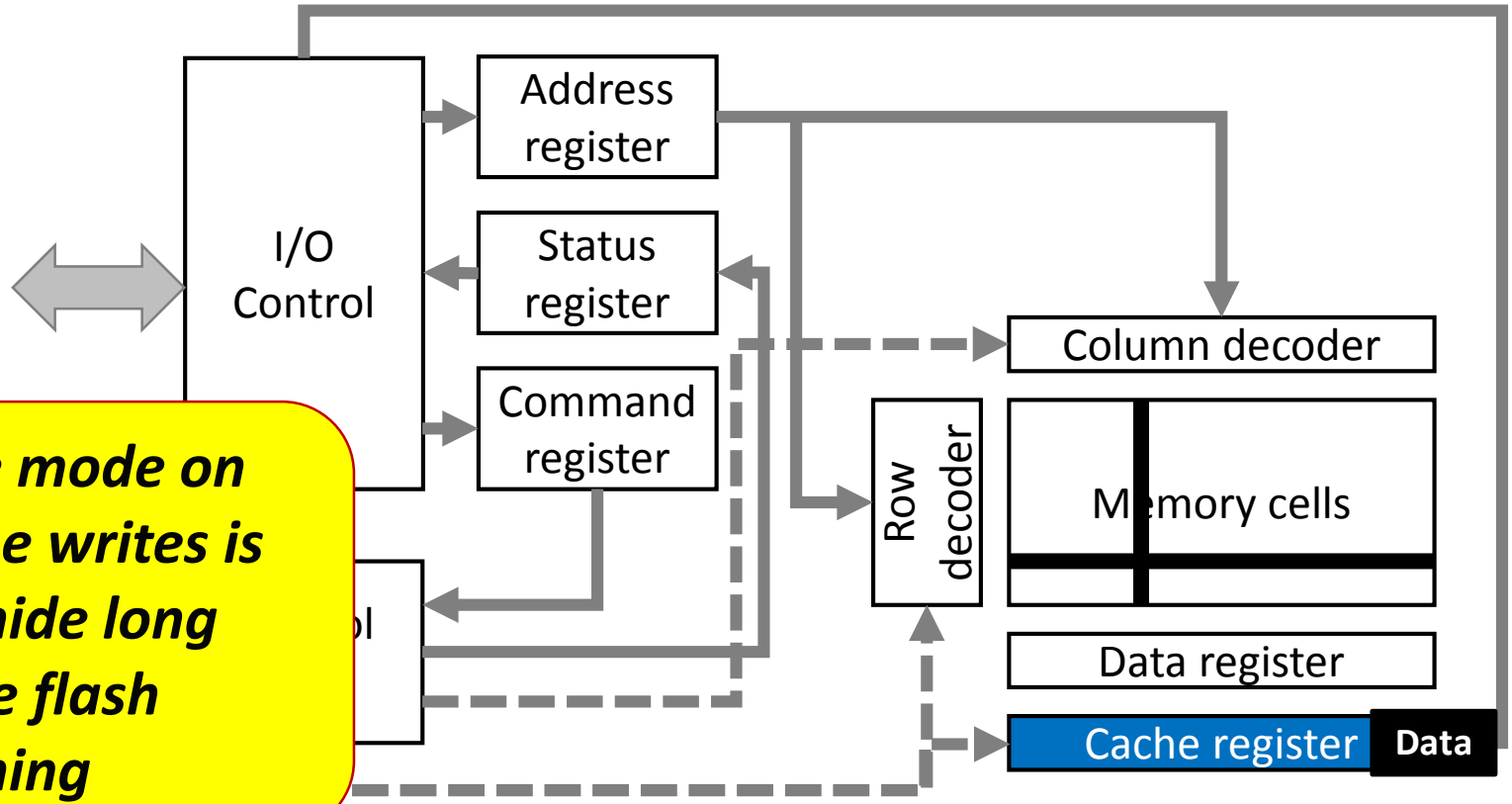


Cache Mode (Writes)



Normal program page command (10h) can be used for the last

Unlike the cache mode on reads, caching the writes is insufficient to hide long latency of the flash programming

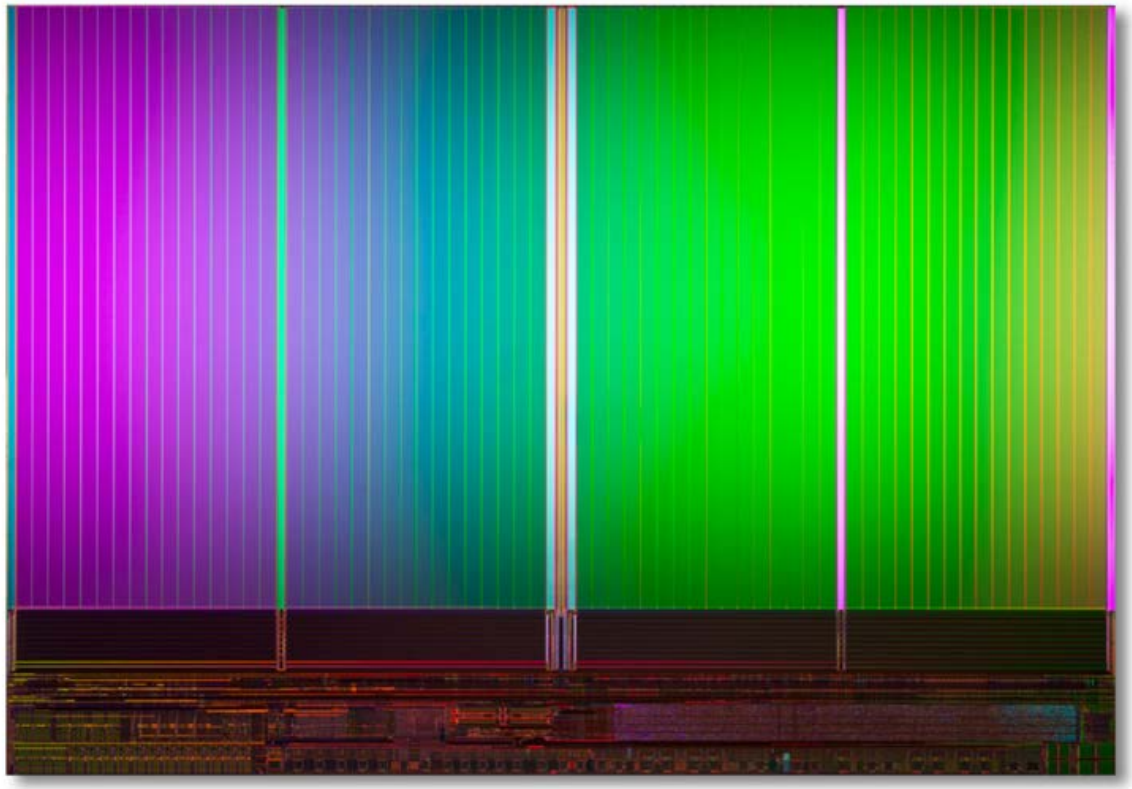


Outline

- Flash Microarchitecture
- Basic Operations (Legacy)
- Cache Mode Operations
- Multi-plane Mode Operations
- Copy-back Operations
- Erase Operations
- Evaluation Studies
  - Simulation-based studies
  - Real implementation (FGPA) and evaluation

Multi-plane Mode Operation

- Two (or more) different pages can be served in parallel
- For some specific flash technologies, the operation should be composed by a same page offset (in a block) and a same die address, but different plane addresses (*plane addressing rule*)



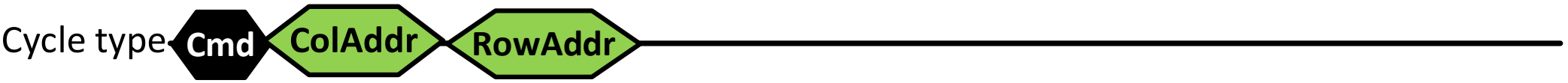
check
cmd
data2
data1
cmd
addr2
addr1
cmd

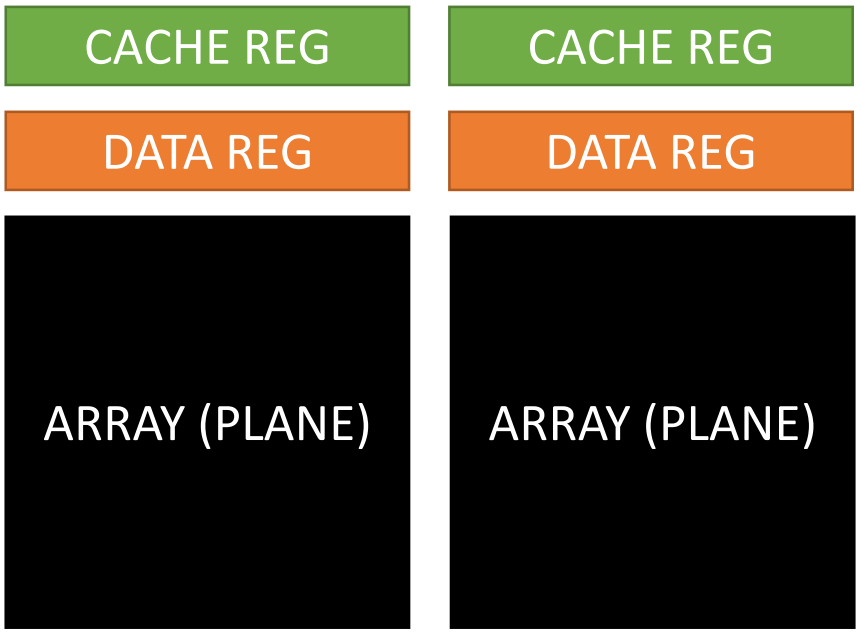
Multi-plane Mode

- Before sending a basic operation command, flash controller(s) needs a request to transfer data from a target plane to its cache register
- Each new plane address is specified, the corresponding request is queued for transferring data
- The address of the final plane should be brought by a basic read or write command
  - The internal flash control logic then serves all queued plane requests at a time (multi-plane operation)

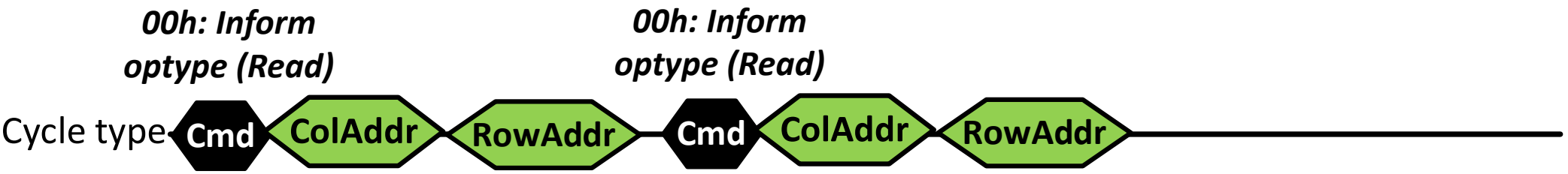
Multi-plane Mode

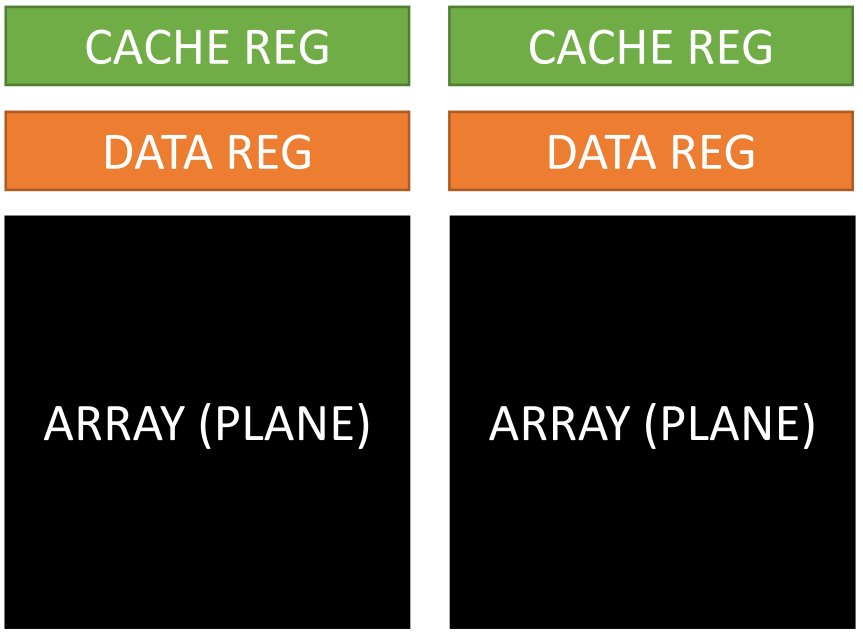
00h: Inform optype (Read)



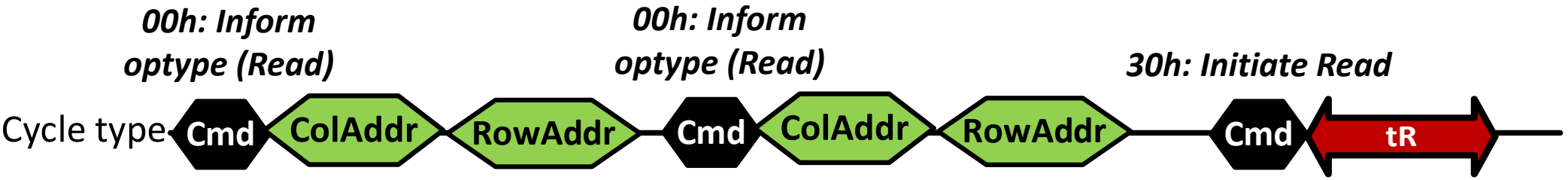


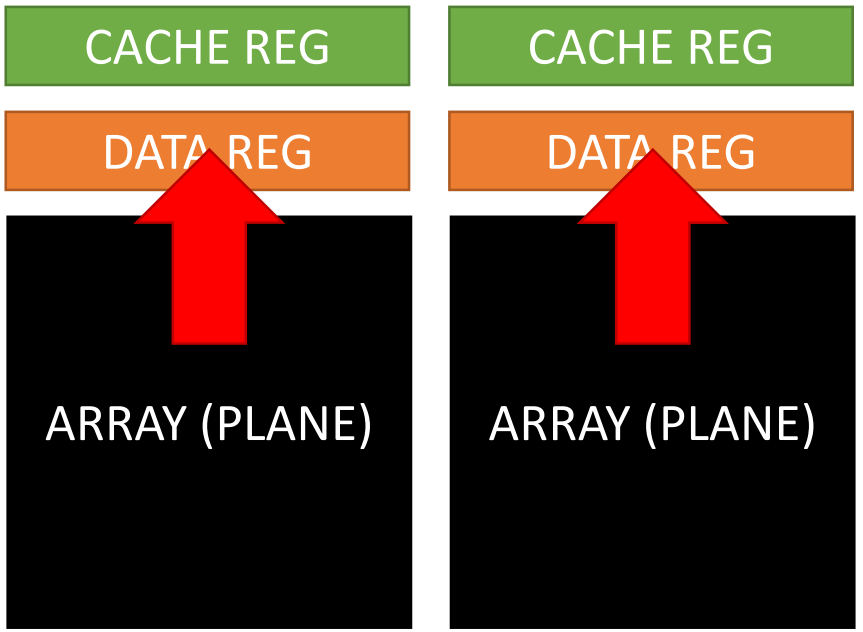
Multi-plane Mode



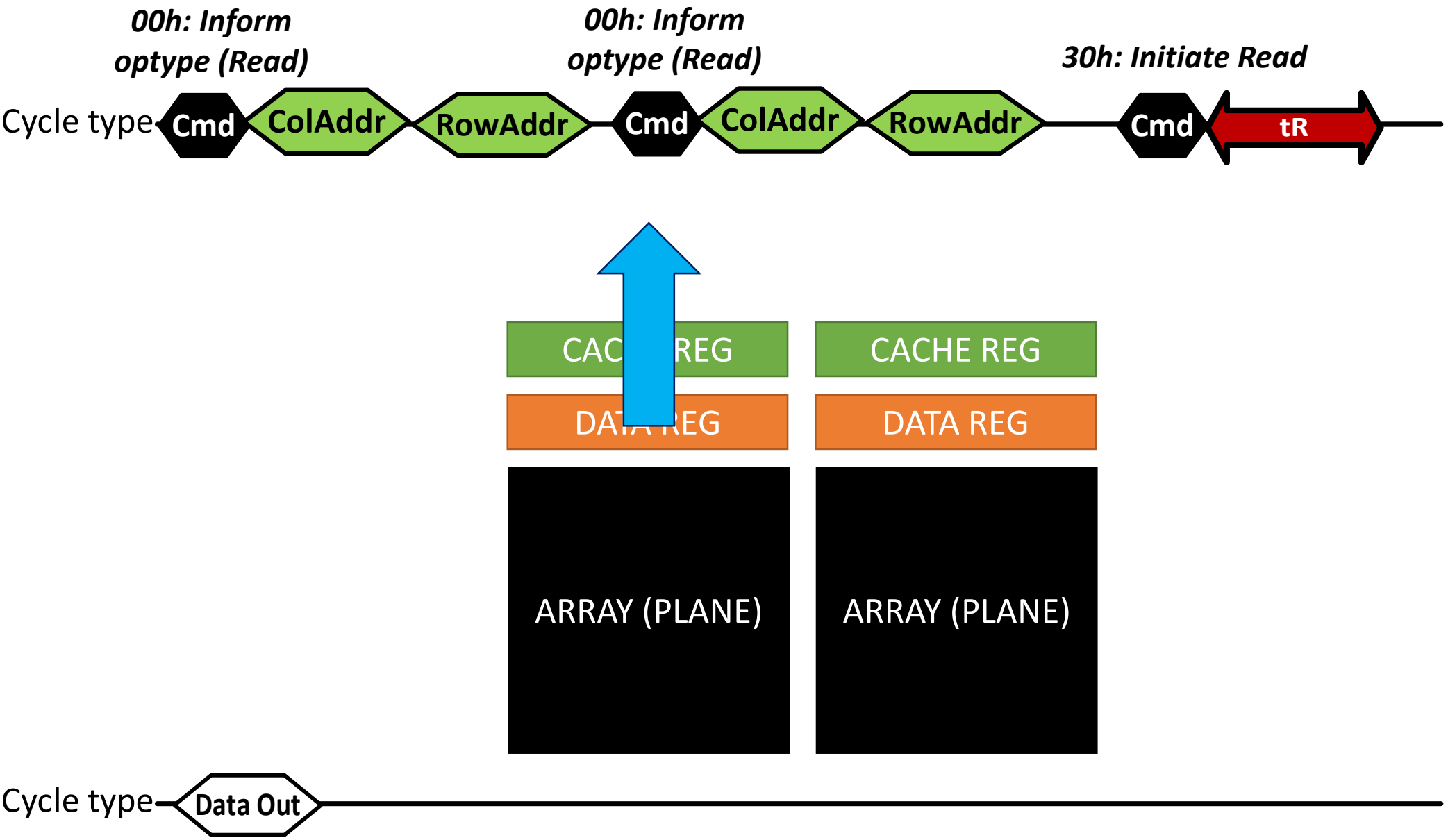


Multi-plane Mode

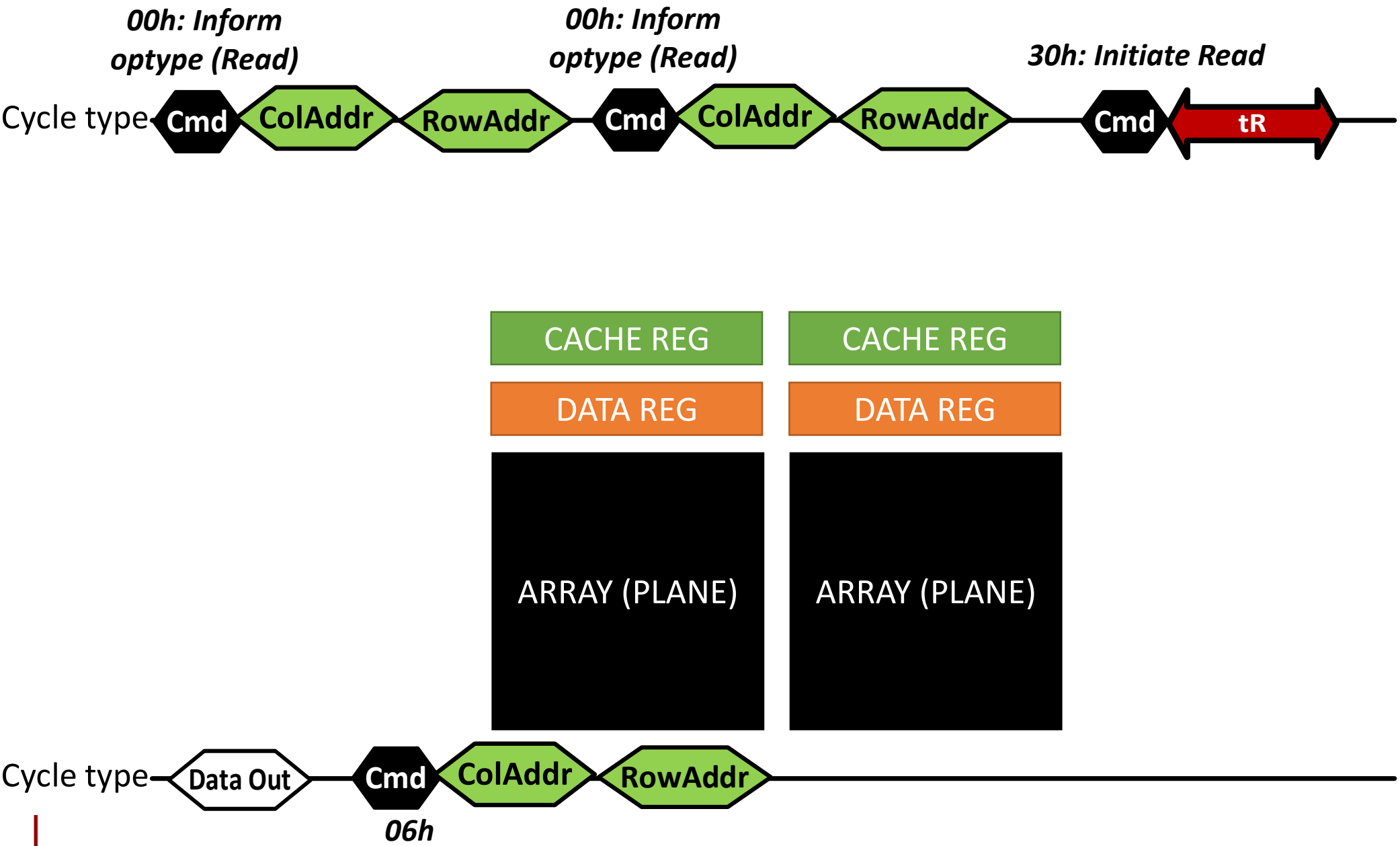




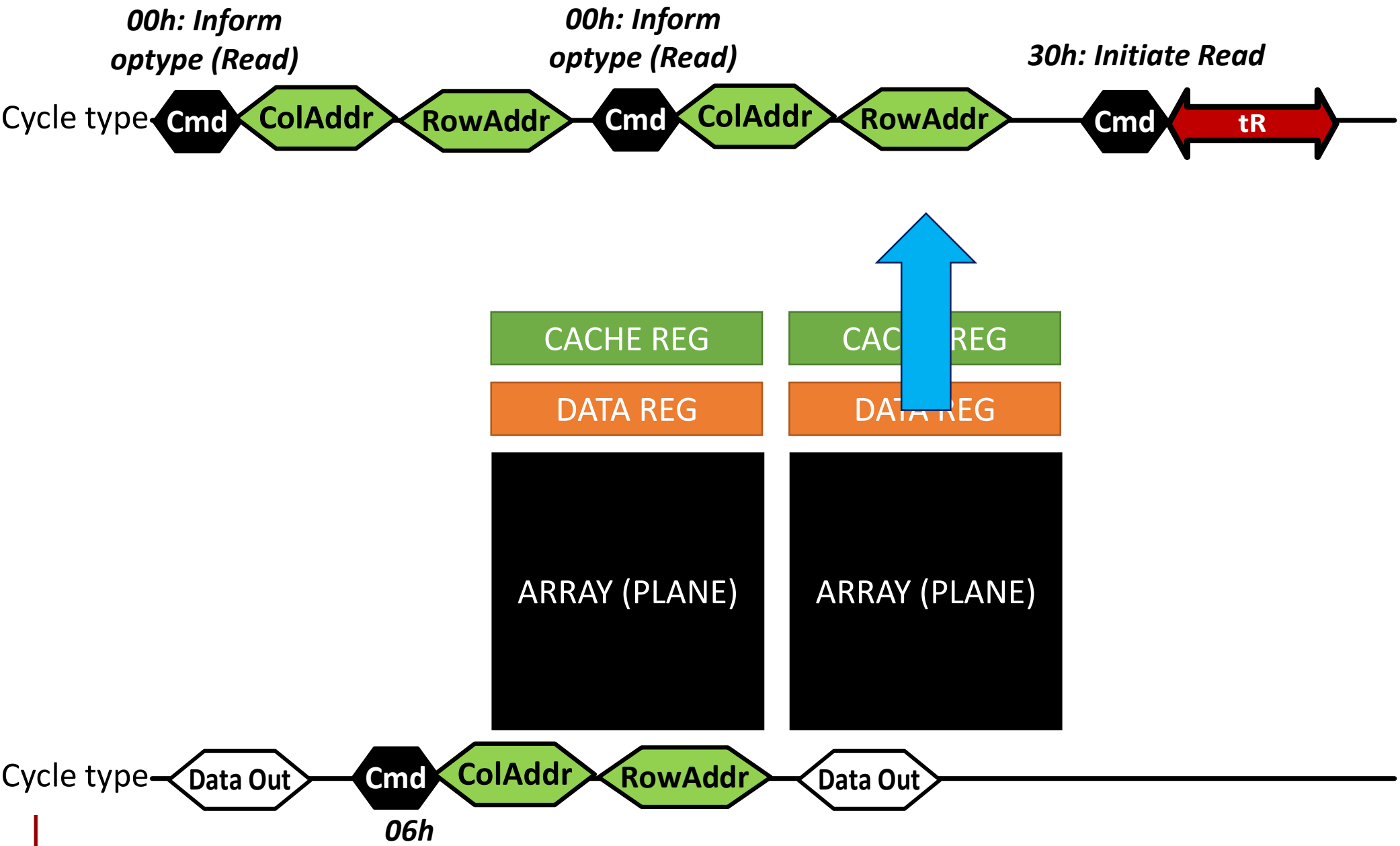
Multi-plane Mode



Multi-plane Mode



Multi-plane Mode

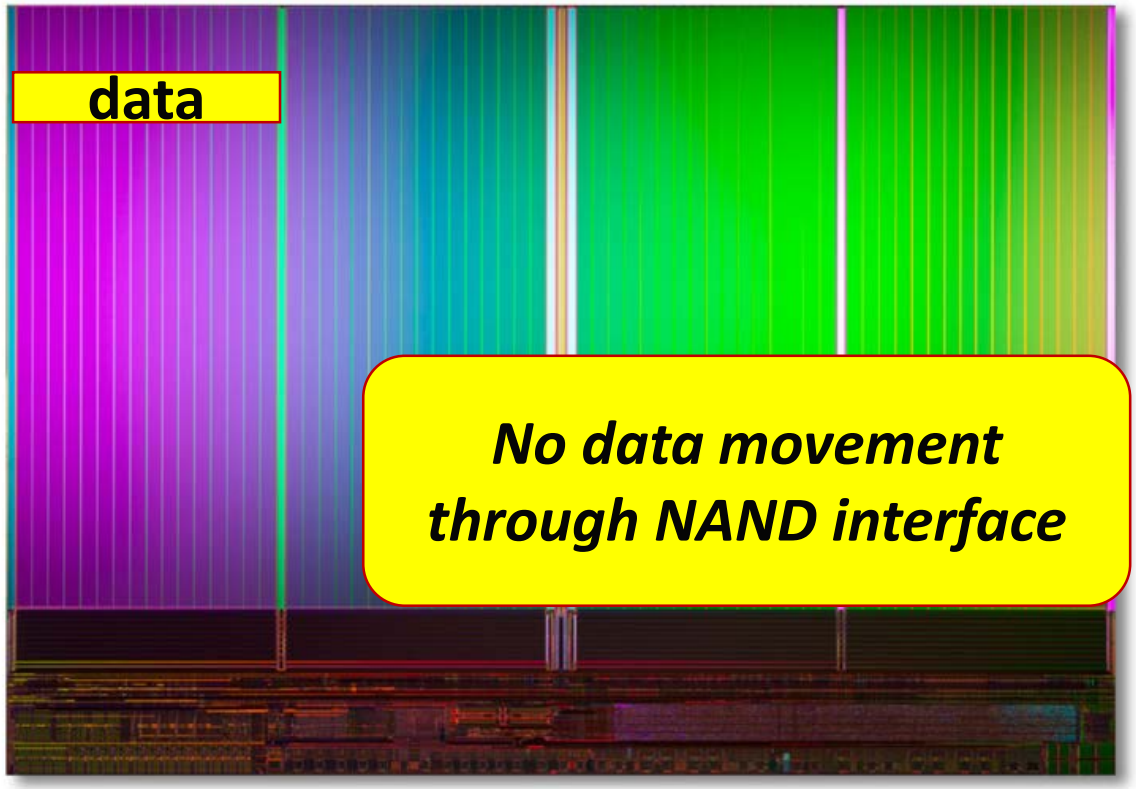


Outline

- Flash Microarchitecture
- Basic Operations (Legacy)
- Cache Mode Operations
- Multi-plane Mode Operations
- Copy-back Operations
- Erase Operations
- Evaluation Studies
  - Simulation-based studies
  - Real implementation (FGPA) and evaluation

Copyback Operations

- This can save space and cycles to copy data
- The source and destination page address should be located in the same die



cmd

dst addr

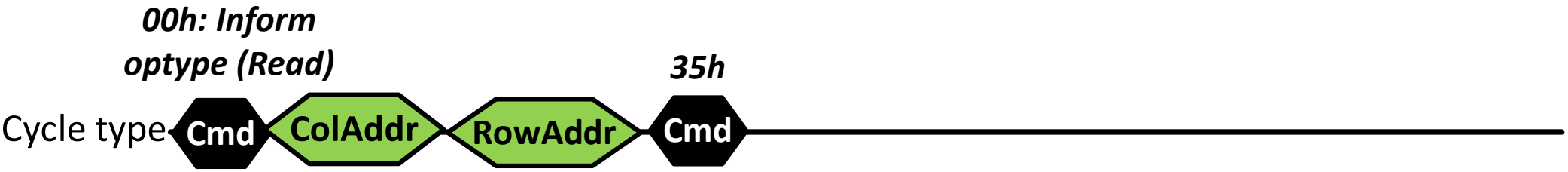
src addr

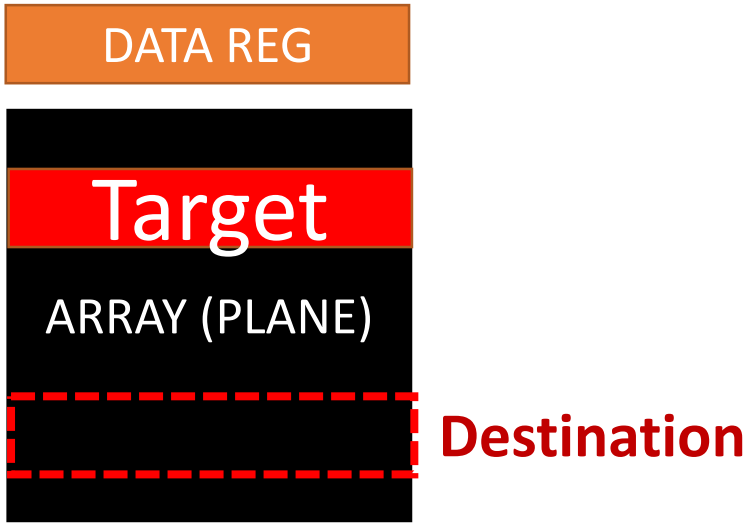
cmd

Copyback Operations

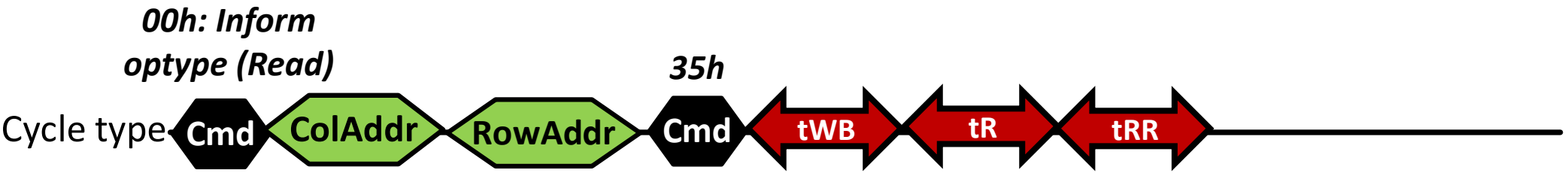
- Block management (like a garbage collection) and wear-leveling mechanisms require to move data from a block to another block through an external interface
- Copyback operations allow to transfer data within a plane from one page to another using the cache register
  - Note that it cannot be performed across different planes (even within a same die)
- AKA Internal Data Movement Mode

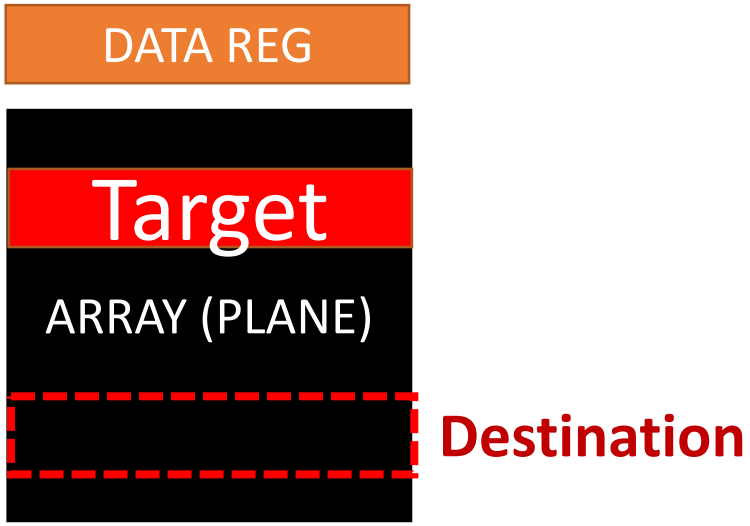
Copyback Operations



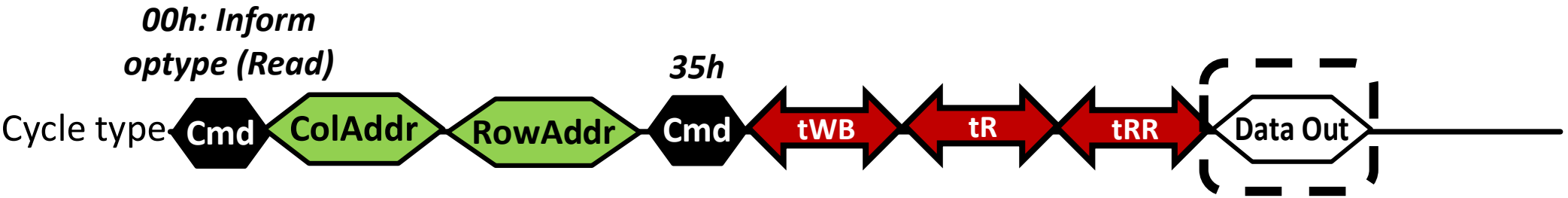


Copyback Operations

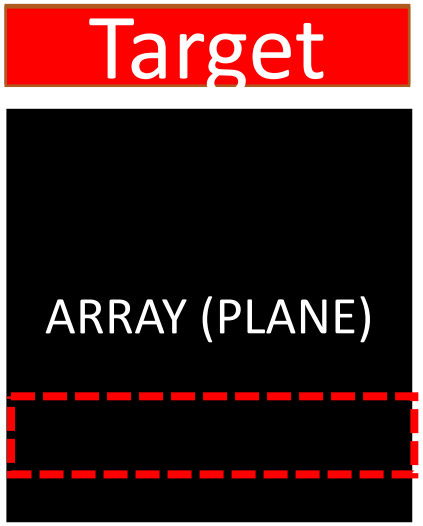




Copyback Operations

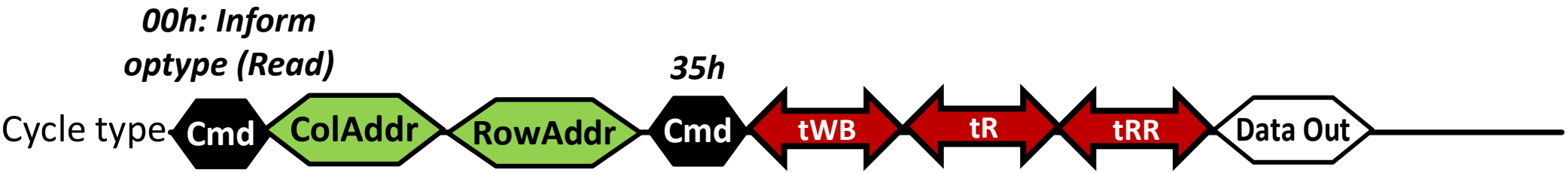


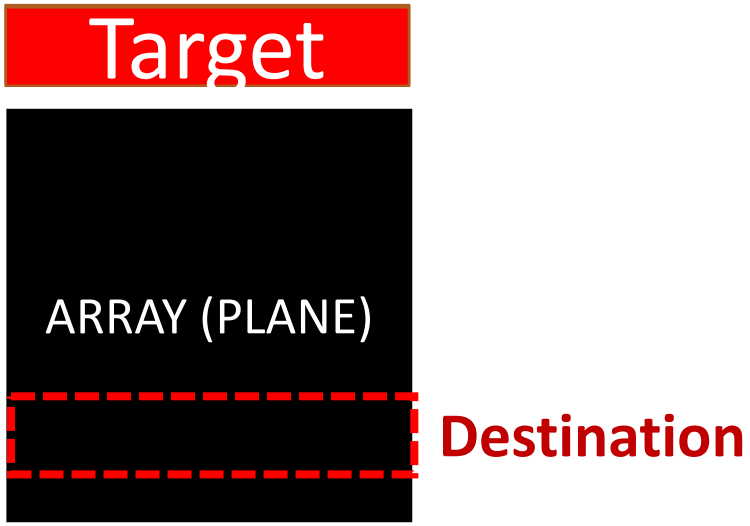
This is optional



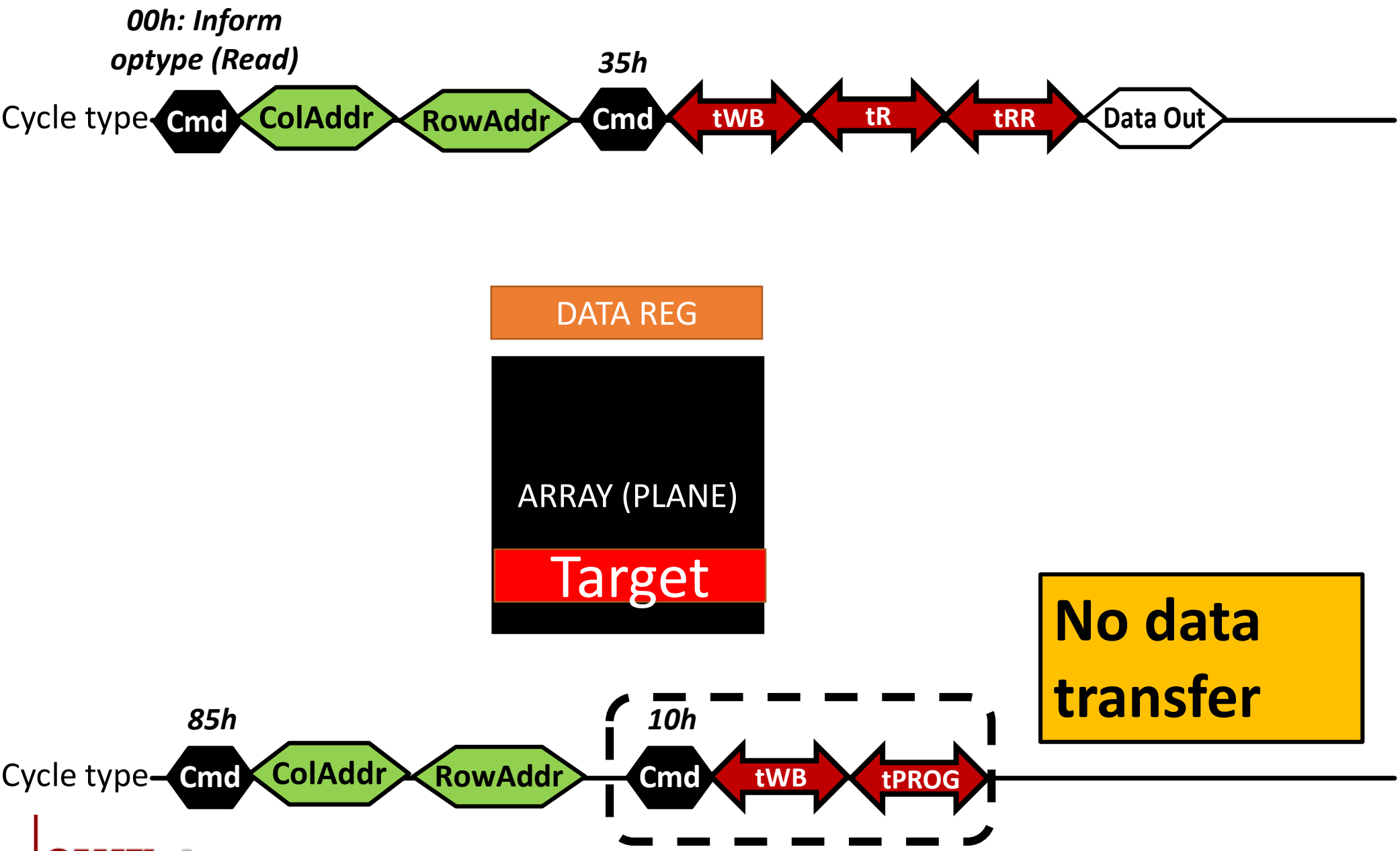
Why do we need this data transfer in a copyback?

Copyback Operations





Copyback Operations

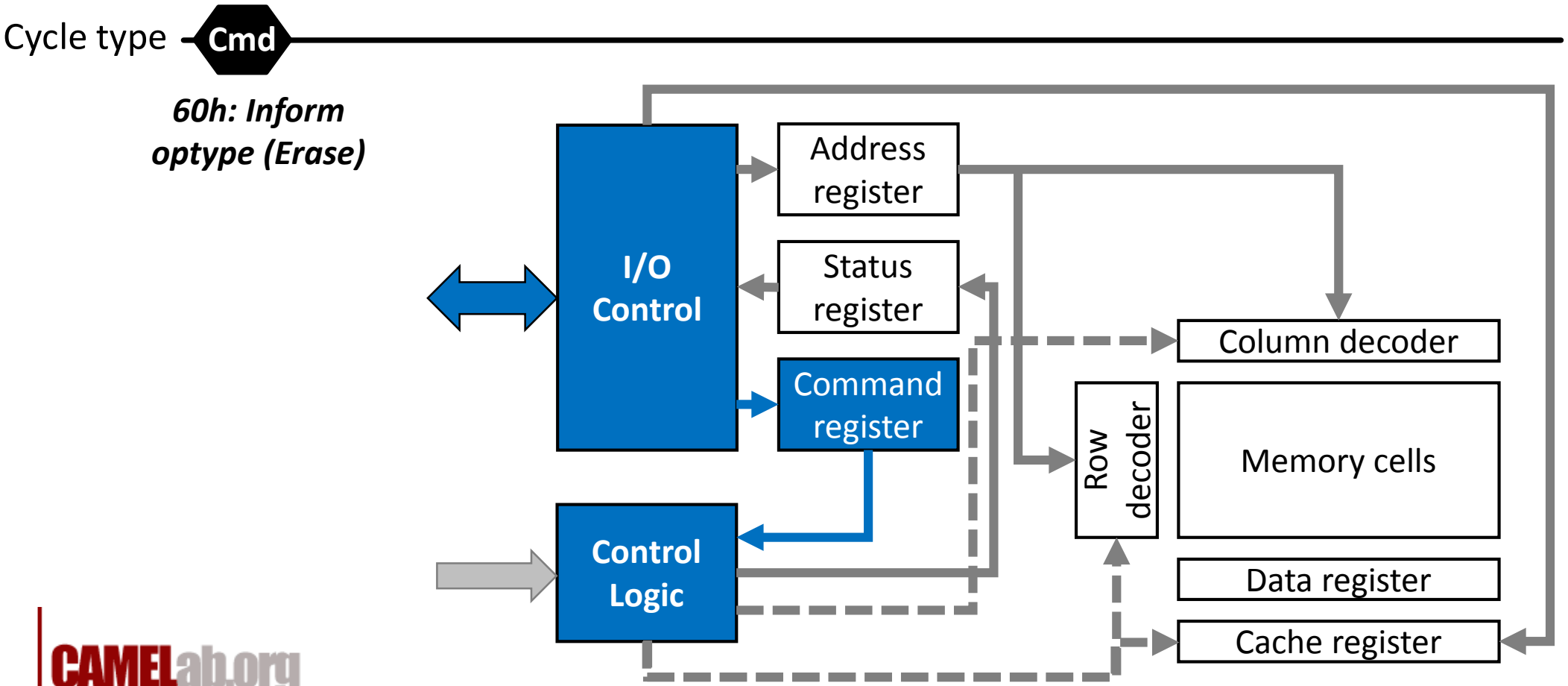


Outline

- Flash Microarchitecture
- Basic Operations (Legacy)
- Cache Mode Operations
- Multi-plane Mode Operations
- Copy-back Operations
- Erase Operations
- Evaluation Studies
  - Simulation-based studies
  - Real implementation (FGPA) and evaluation

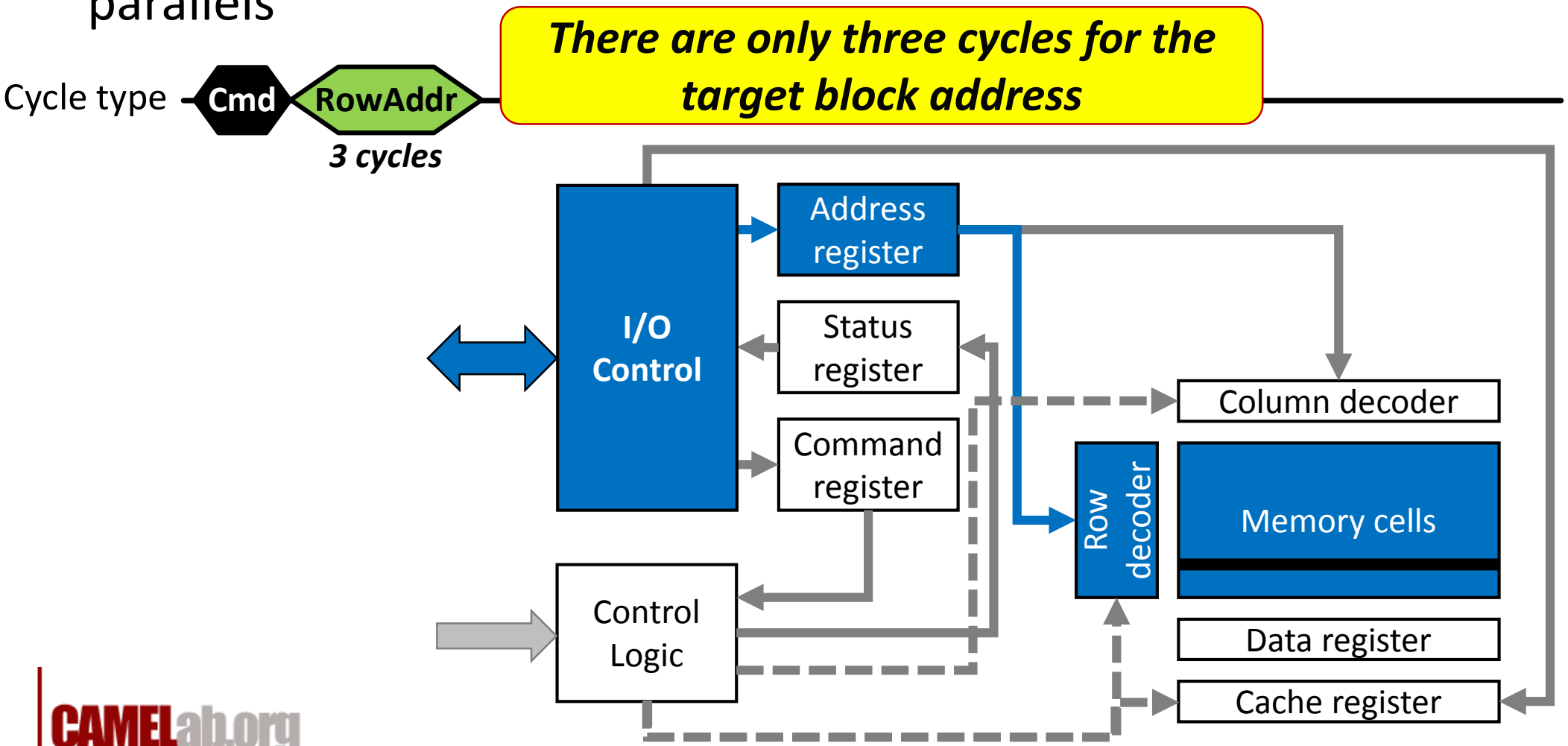
Erase Operations

- The command sequence for erasing a block is not different with basic operations, but the page address is ignored (column)
- It also supports multi-plane mode for erasing multiple blocks in parallels



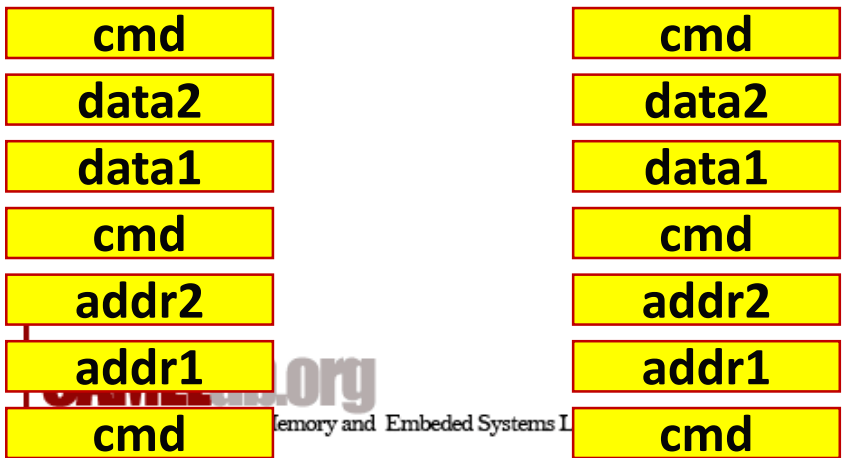
Erase Operations

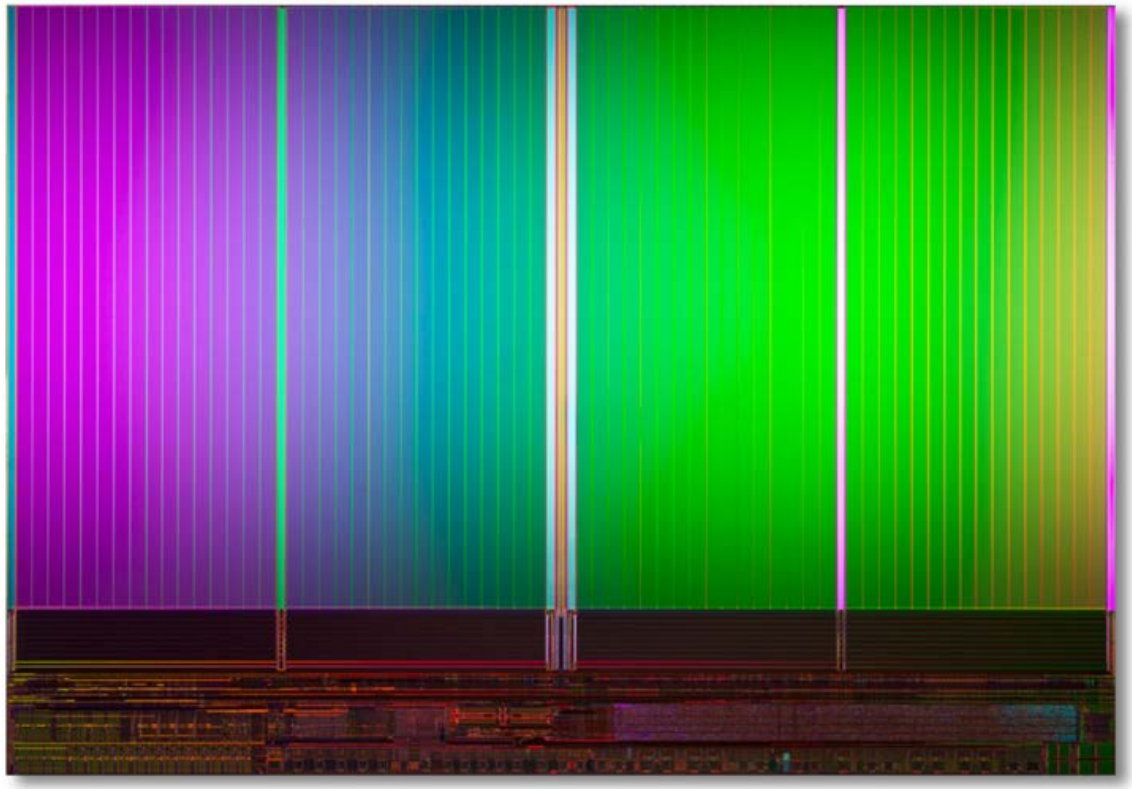
- The command sequence for erasing a block is not different with basic operations, but the page address is ignored (column)
- It also supports multi-plane mode for erasing multiple blocks in parallels



Combination

- All NAND flash memory operations are used in any combination
- Scheduling NAND flash command is an important design parameter to determine system performance





References

- NANDFlashSim: High-Fidelity, Microarchitecture-Aware NAND Flash Memory Simulation, TOS
- Internal Parallelism of Flash Memory-Based Solid-State Drive, TOS
- Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity, ICS
- OpenNVM: An Open-Sourced FPGA-based NVM Controller for Low Level Memory Characterization, ICCD
- Exploring Design Challenges in Getting Solid State Drives Closer to CPU, TC