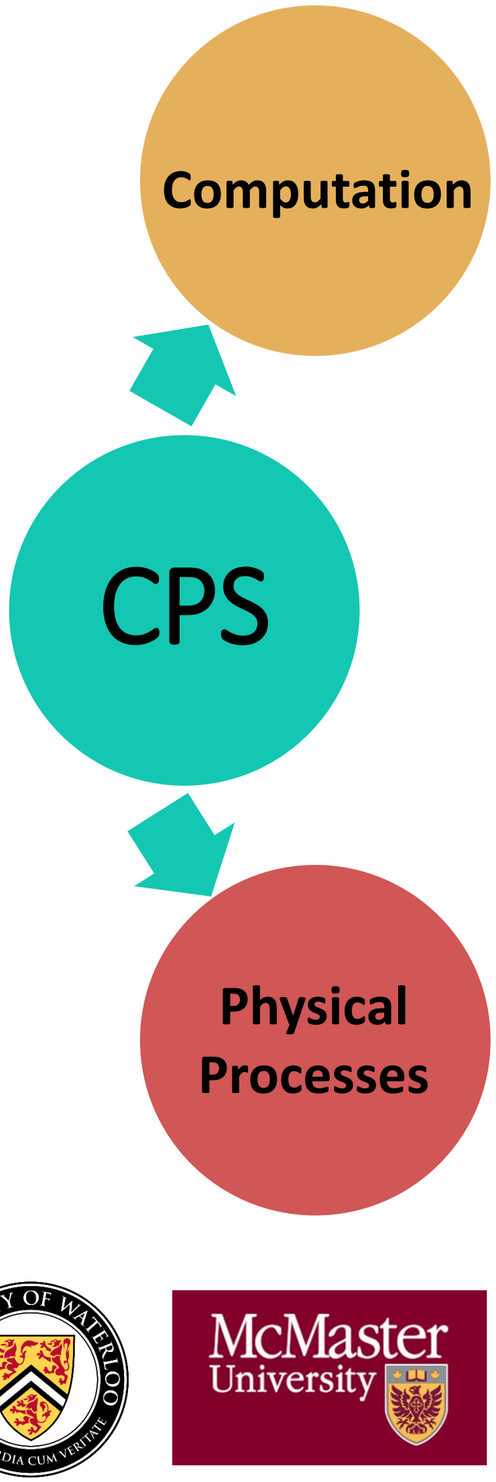
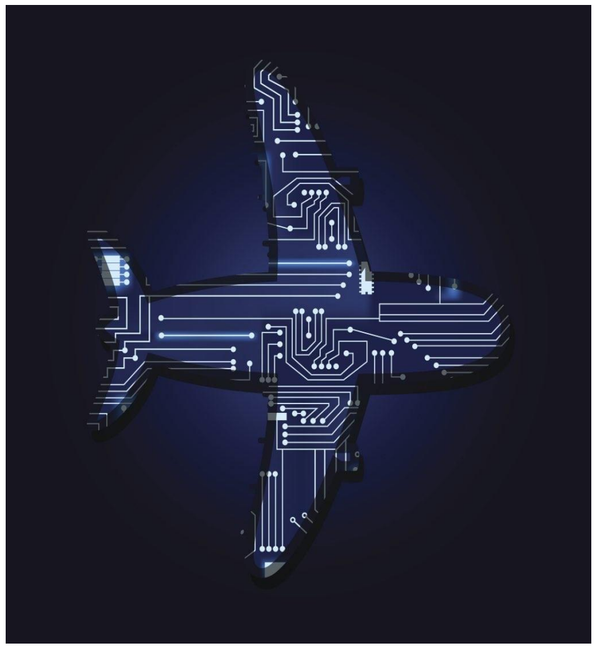
DR*AMbulism*: Balancing Performance and Predictability through Dynamic Pipelining

Reza Mirosanlou, Mohamed Hassan, and Rodolfo Pellizzoni

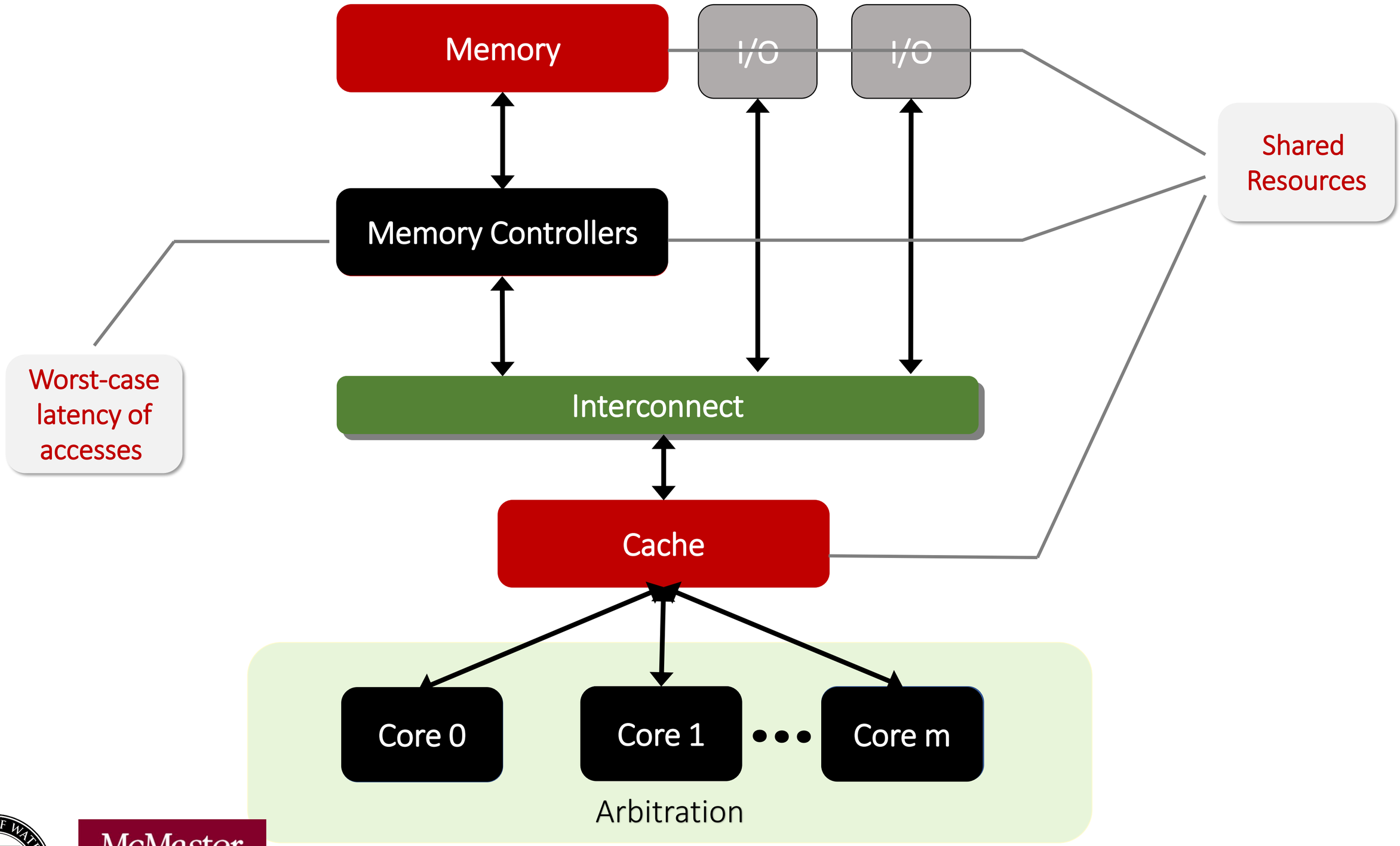
RTAS 2020 @ Sydney

Cyber Physical Systems



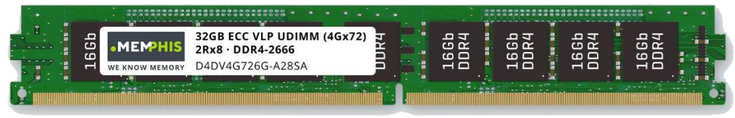


Multicore Processors



Memory

- Worst-case execution time are impacted by WCL
- SRAMs, predictable and fast, but inefficient; hence, we focus on DRAM
  - Cheap 😊
  - Higher bandwidth 😊
  - Lower power consumption 😊
  - High latency ☹
  - Timing variabilities ☹



- Challenge is to provide tight worst-case bounds and good average-case performance

Motivation

Derive Upper WCL bound on COTS Controllers

- Optimizations

Redesign controllers to tighten the WCL bounds

- Strict rules

Focus of this Work

Pessimistic bounds

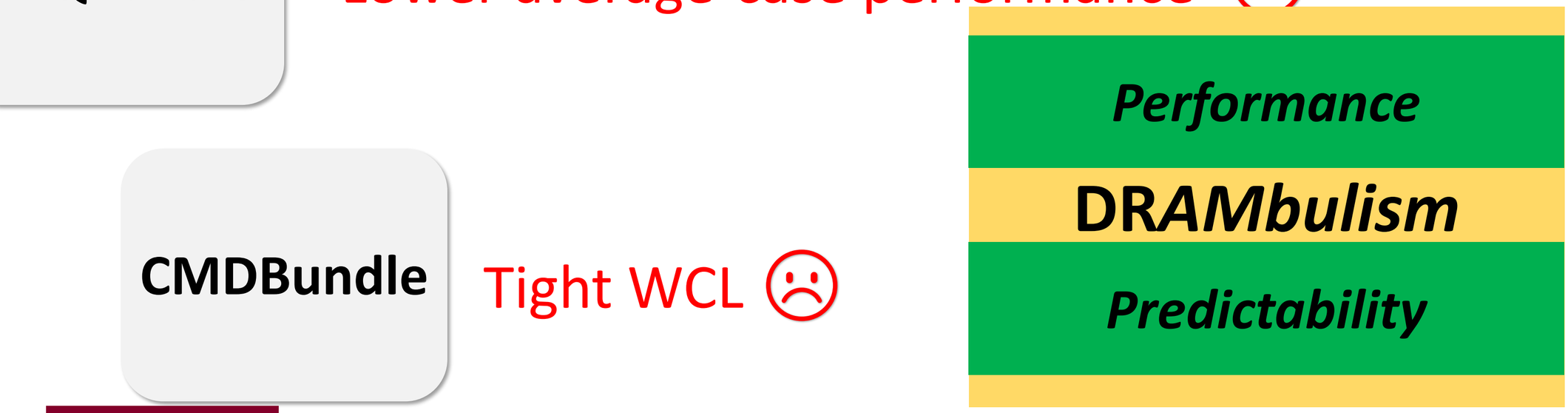
Sacrifice Performance

REQBundle

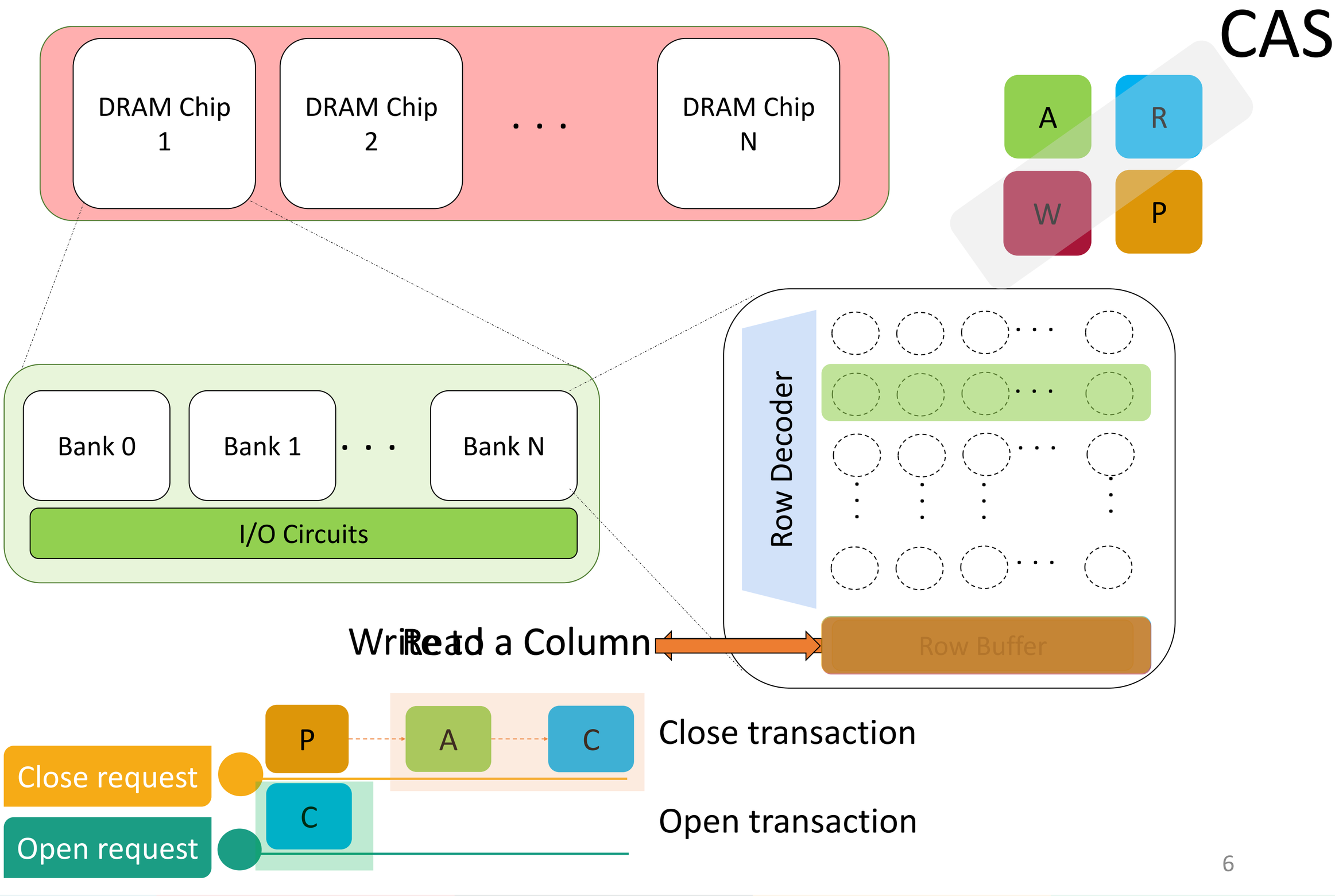
Lower average-case performance ☹

CMDBundle

Tight WCL ☹



DRAM Organization

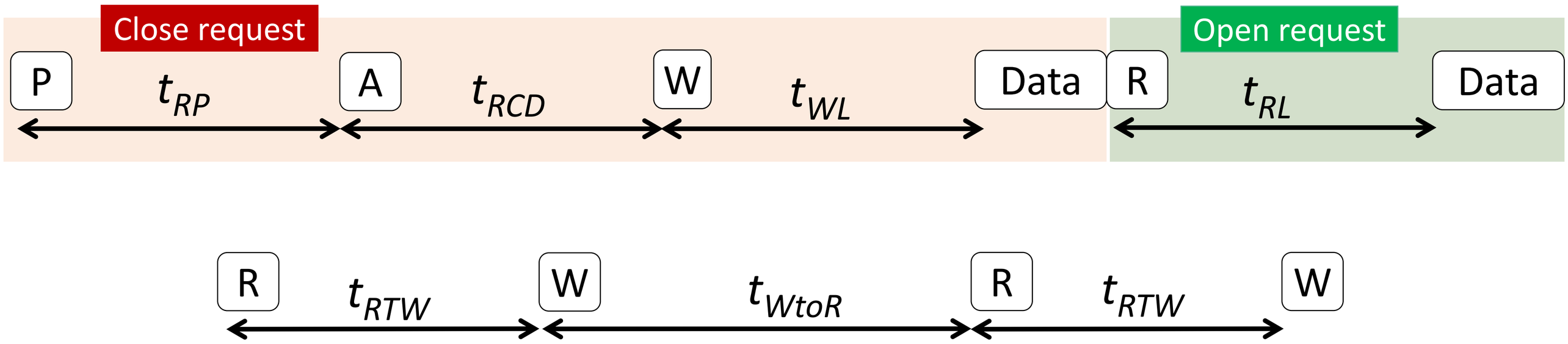


Problems

- Commands must follow constraints by JEDEC

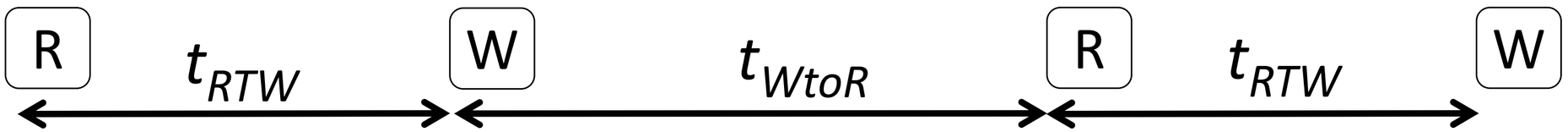
JEDEC DDR3 TIMING CONSTRAINTS FOR DIFFERENT SPEED BINS

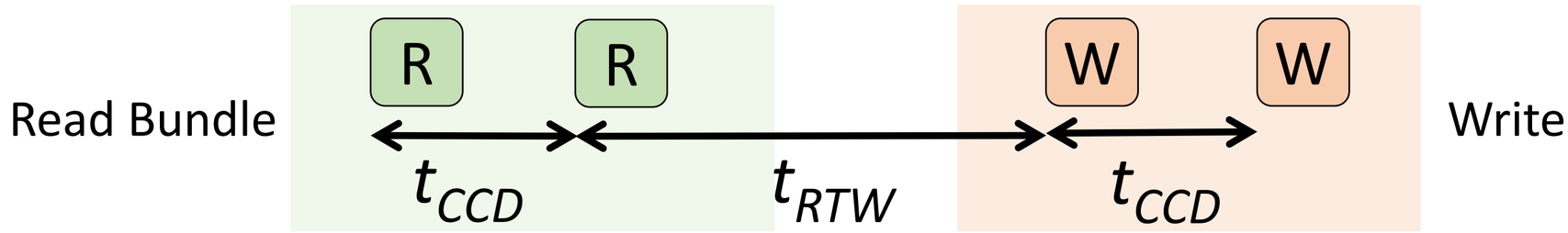
Constraints	1066E	1333G	1600H	1866K	2133L
Inter-bank Constraints (cycle)					
t_{RRD}: ACT to ACT	4	4	5	5	5
t_{FAW}: 4 ACT window	20	20	24	26	27
t_{RTW}: read CAS to write CAS	6	7	7	8	8
t_{WTR}: write data to read CAS	4	5	6	7	8
t_{WtoR}: write CAS to read CAS	14	16	17	20	22
t_{CCD}: CAS to CAS	4	4	4	4	4
t_{Bus}: data transfer length	4	4	4	4	4
Intra-bank Constraints (cycle)					
t_{RL}: read CAS to data	6	8	9	11	12
t_{WL}: write CAS to data	6	7	8	9	10
t_{WR}: write data to PRE	8	10	12	14	16
t_{RCD}: ACT to CAS	6	8	9	11	12
t_{RP}: PRE to ACT	6	8	9	11	12
t_{RTP}: CAS to PRE	4	5	6	7	8
t_{RC}: ACT to ACT	26	32	37	43	48
t_{RAS}: ACT to PRE	20	24	28	32	36



Solutions

- Keep a row buffer open to leverage form open requests
- Private banks and dedicated queues
- Bundle the requests





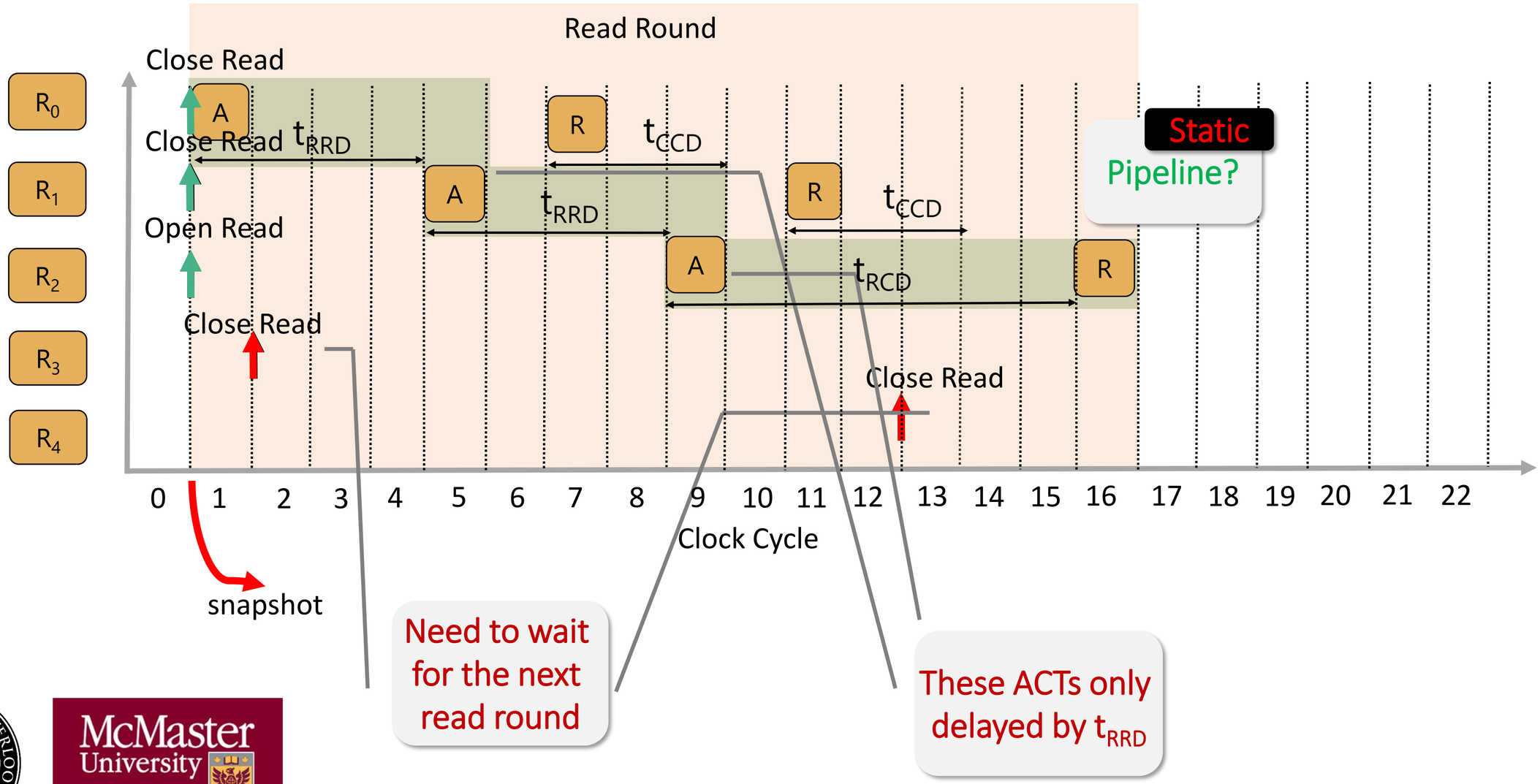
REQBundle Request

CMDBundle Command

- Limit each requestor to have one request in every bundle

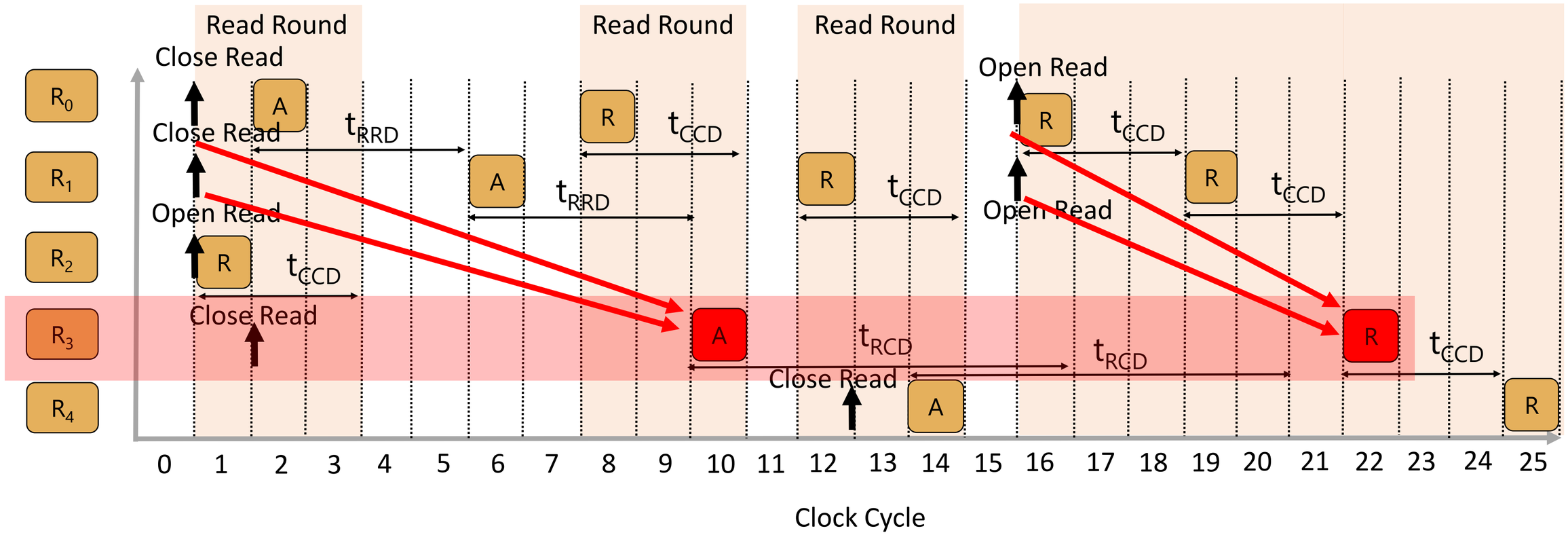
REQBundle

- Accept requests to service in round at snapshot
- Send all the requests as a close requests



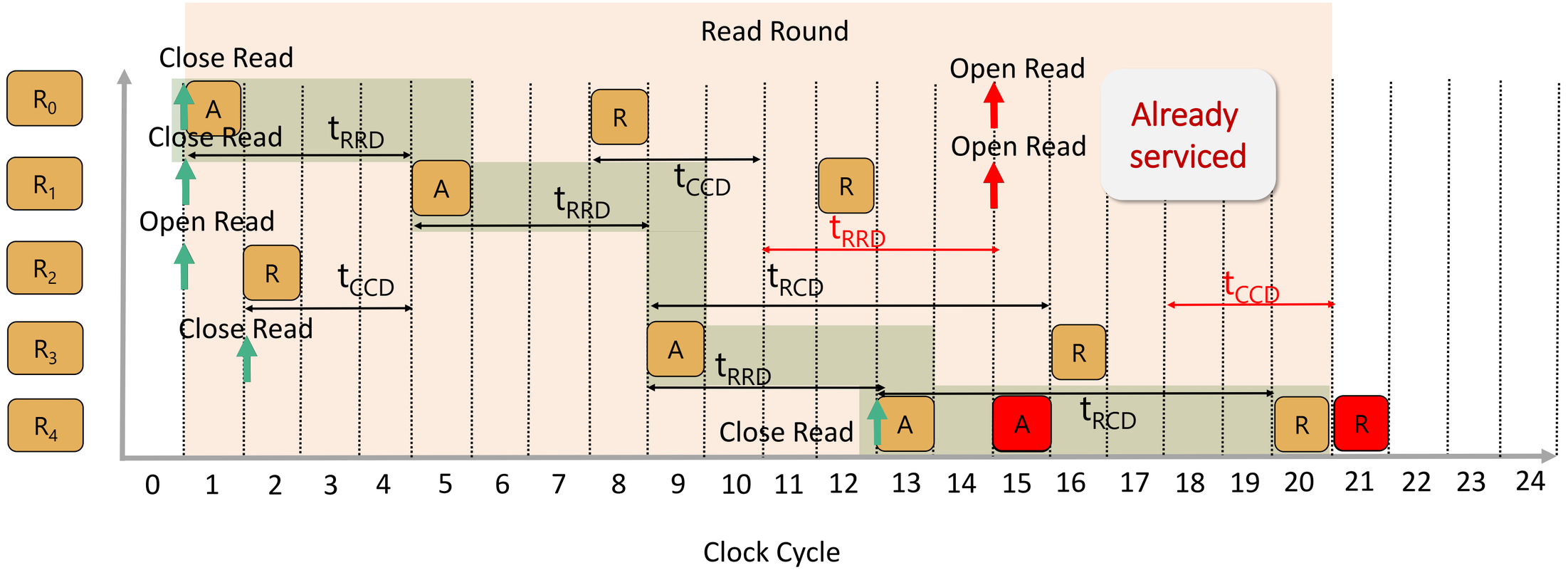
CMDBundle

- Accept commands to service in round dynamically
- It leverage from open requests
- Separate round-robin for ACT, CAS, and PRE commands



DRAMBulism

- Proposed solution: Process requests in round dynamically while maintaining the pipeline
- Take the benefits of both REQBundle and CMDBundle
- Acceptance rules?



DR*AMbulism* Contd.

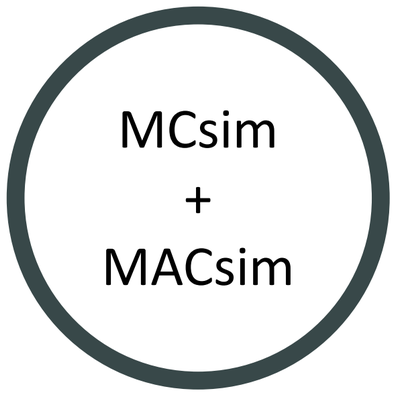
- DRAMbulism do not allow the pipeline to break!
- We are able to prove max round length is bounded by:

$$\max(\text{switch\_time}, t_{RCD}) + (N-1) \cdot \max(\text{CAS-CAS}, \text{ACT-ACT})$$

- Based on these, we construct the worst cases as:
  - Self blocking → Already serviced in round
  - Pipe blocking → Blocked due to maintain pipeline

Evaluation

- We have implemented all controllers in MCsim
- In order to evaluate the average-case performance, we hooked up MCsim to MACsim, a full system simulator

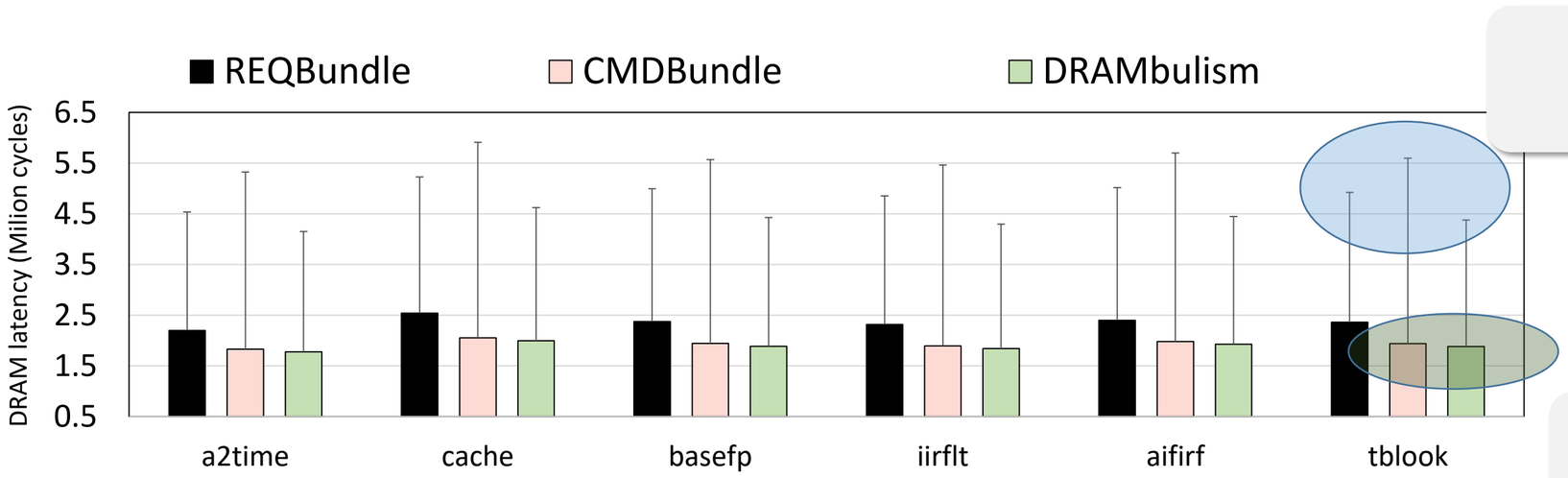


X86, 1GHz

EEMBC benchmark suite (*a2time, cache, basefp, irrflt, aifirf, tblook*), and synthetic tasks

8 Requestors, 1 as core under analysis, 7 interfering cores stressing the cua with open requests

Counterparts: REQBundle, CMDBundle

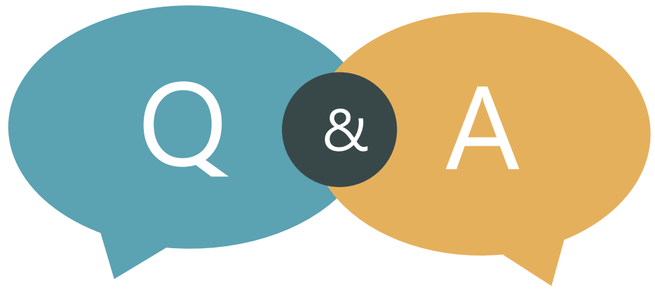


Tighter bounds

Similar performance

Conclusion

- Some conditions guarantee the pipeline in each bundle of same-direction requests
- Evaluation demonstrates that DRAMbulism provides comparable bounds to the most predictable real-time controller while delivering average performance similar to the highest performance real-time controller
- Future work: Multi-rank DDR device with multi mode pipeline



THANKS FOR WATCHING

WE'LL BE ANSWERING QUESTIONS

rmirosan@uwaterloo.ca