Von Irrwegen und Zukunftstrends bei Mikrocontrollern

Marcus Gößler

…the "not parallel" era will appear to be a very primitive time in the history of computers when people look back in a hundred years.

Within a decade, a programmer who does not think "parallel" first will not be a programmer.

By James Reinders
October 12, 2006

Dicit ei Simon Petrus: „Domine, quo vadis?" Respondit Iesus: „Quo vado, non potes me modo sequi, sequeris autem postea".

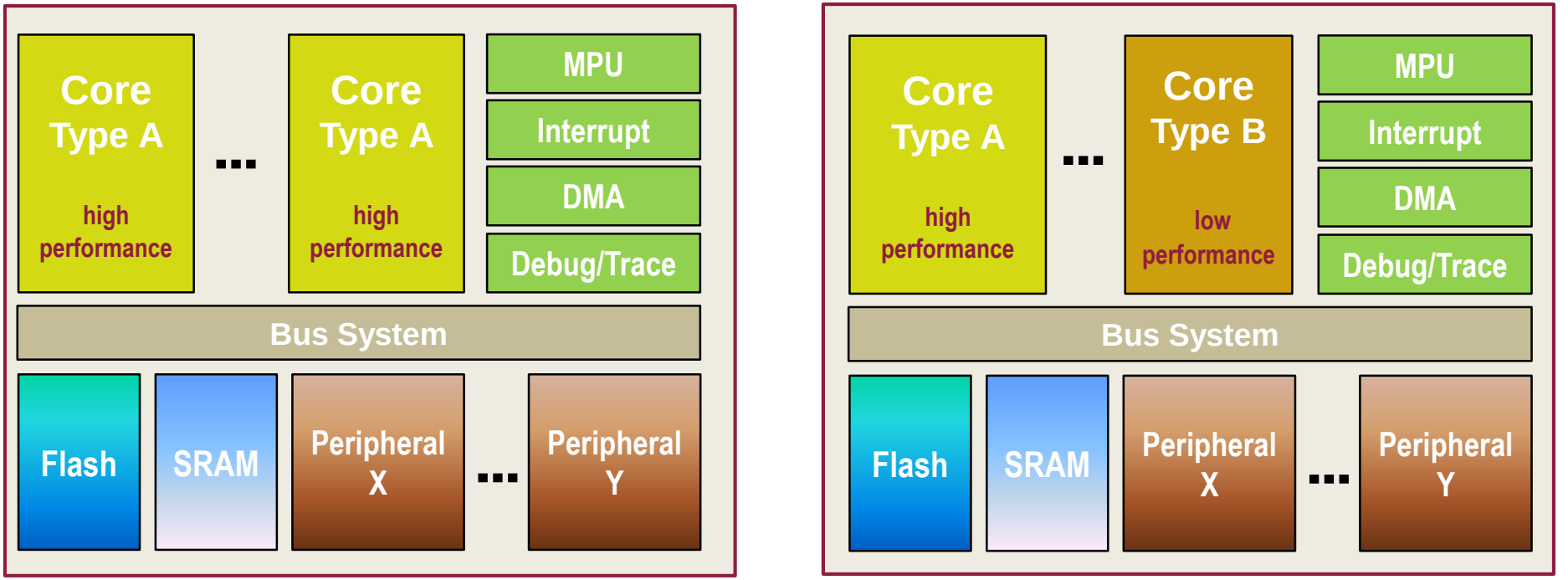
Simon Petrus sagte zu ihm: „Herr, wohin willst du gehen?" Jesus antwortete: „Wohin ich gehe, dorthin kannst du mir jetzt nicht folgen. Du wirst mir aber später folgen."

- Multicore related **concepts** for **decades**
- Driving factors from **customers** and **suppliers**
- Driving factors from pure **physics**
- Driving factors from **standards** and **bodies**
- Multicore brings **benefits**
- Multicore brings new **challenges**

Homogeneous Microcontroller Multicore Architecture

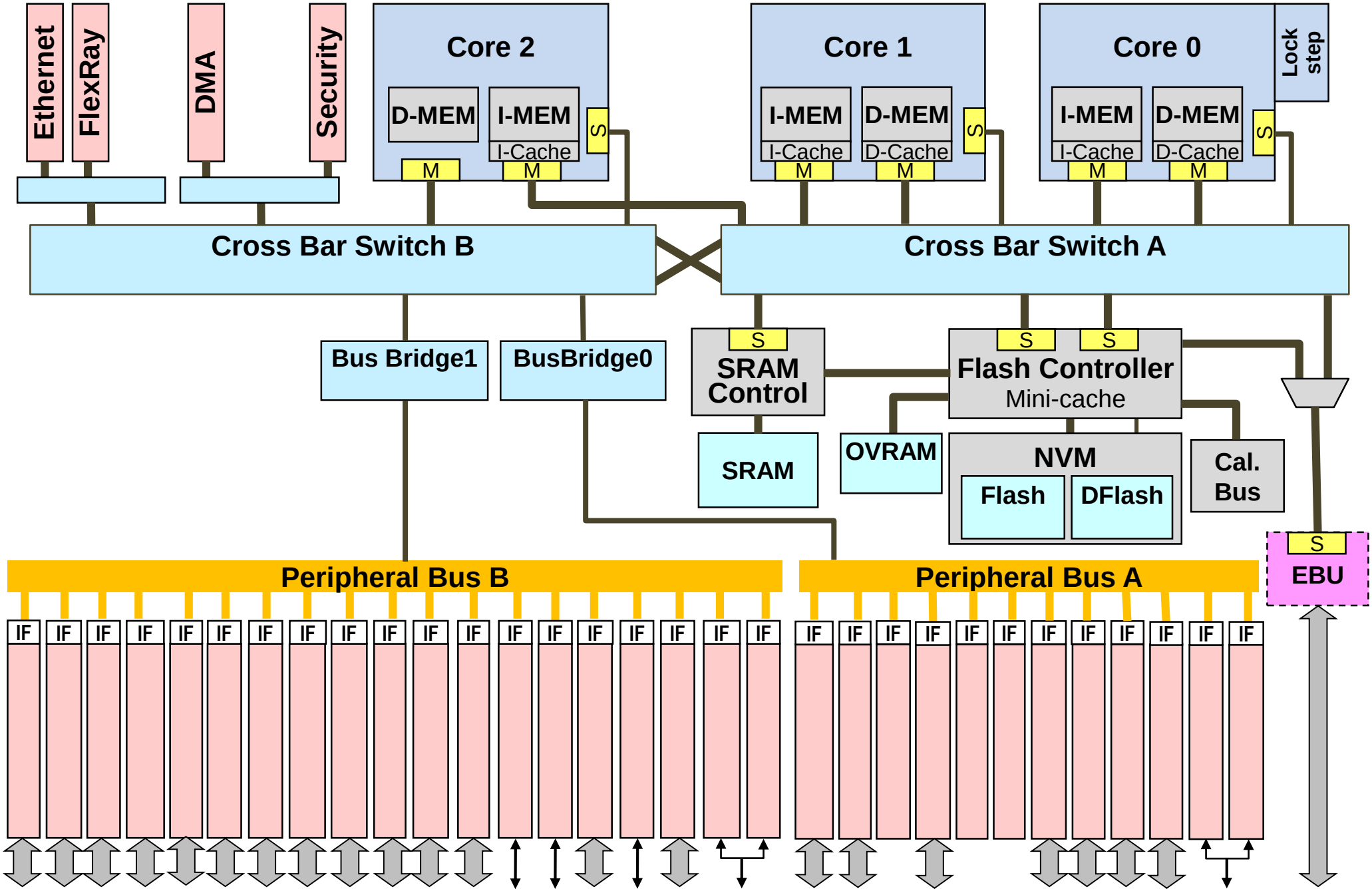
Heterogeneous Microcontroller Multicore Architecture

Mixed Microcontroller Multicore Architecture



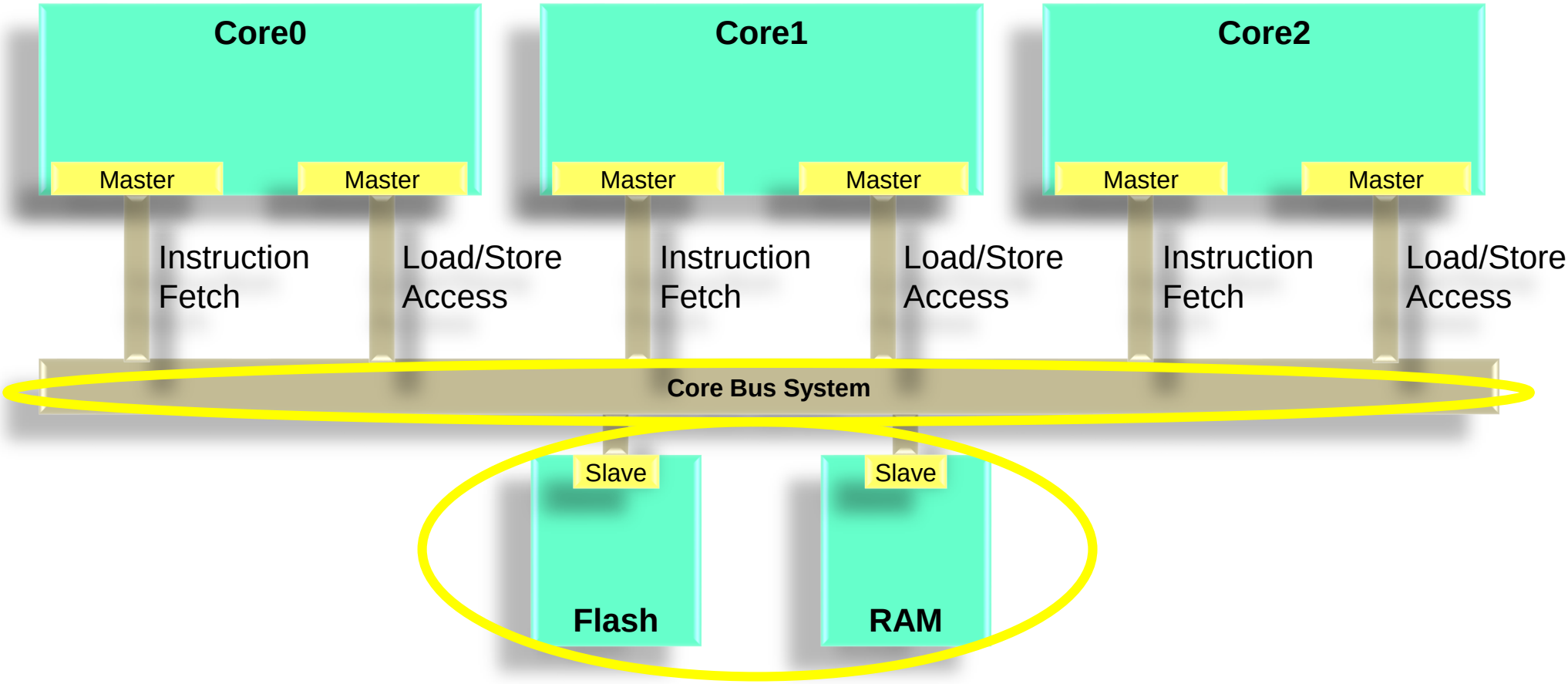
Implementation examples:
- Freescale: i.MX6, MCP56xx
- Infineon Technologies: AURIX TC2xx

Implementation examples:
- NXP: LPC 43xx, 54xxx
- STMicroelectronics: SPEAr1310, SPC56xx

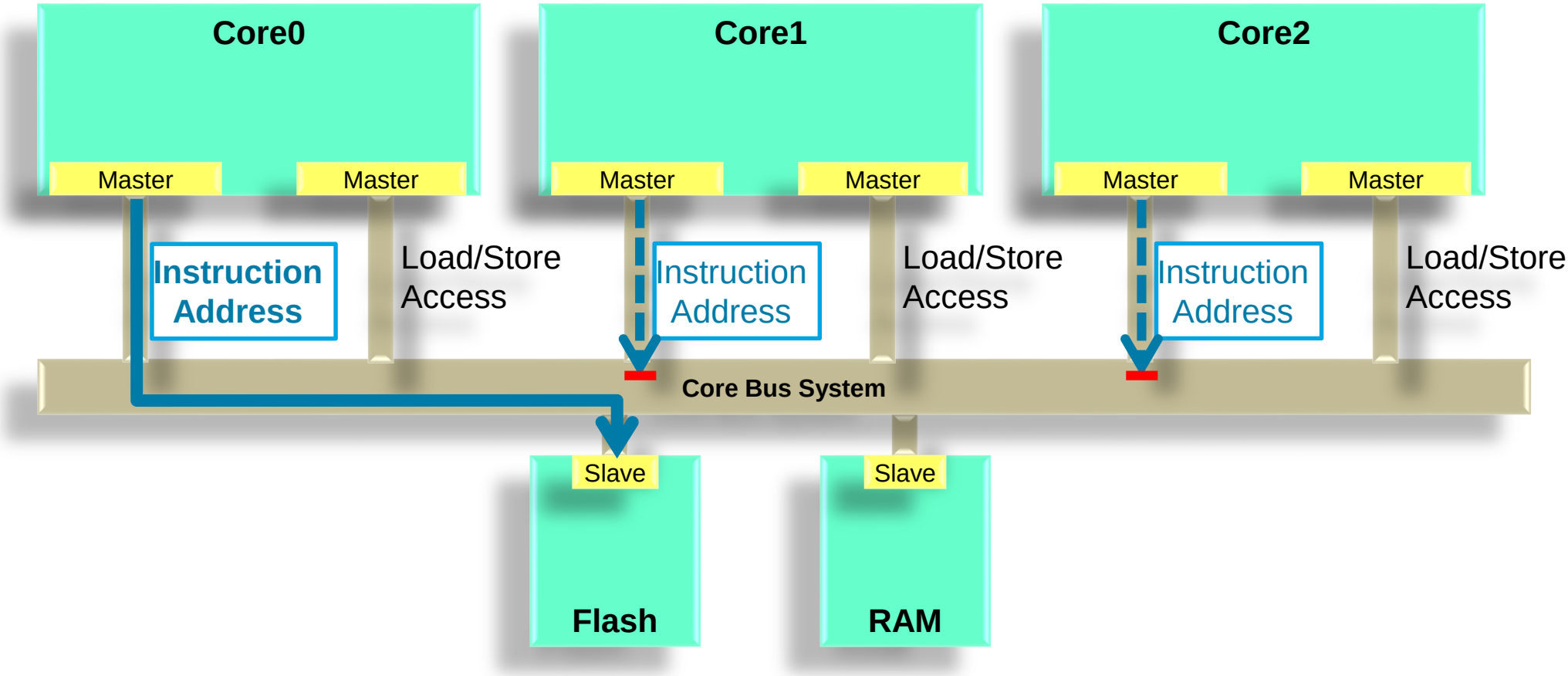


MICRO CONSULT

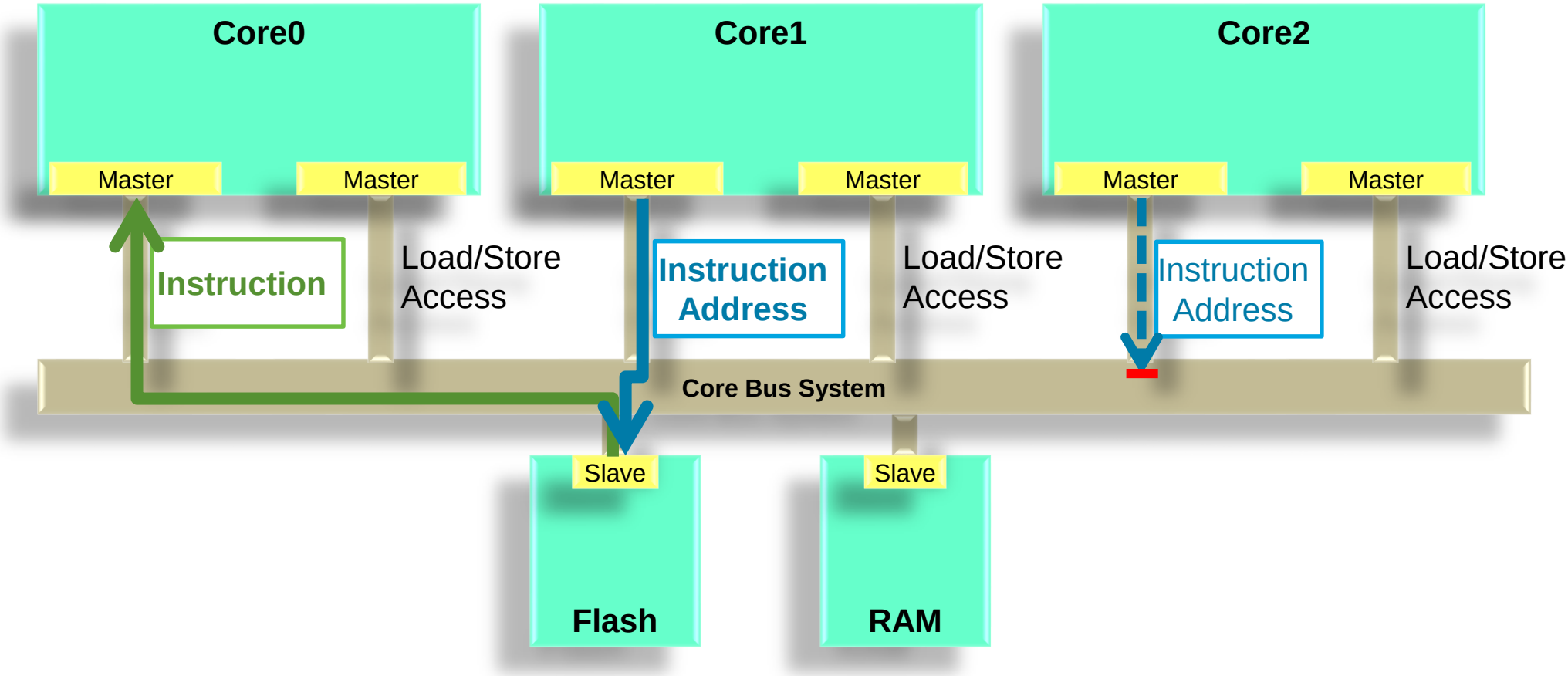
Multicore system with global program and data memory



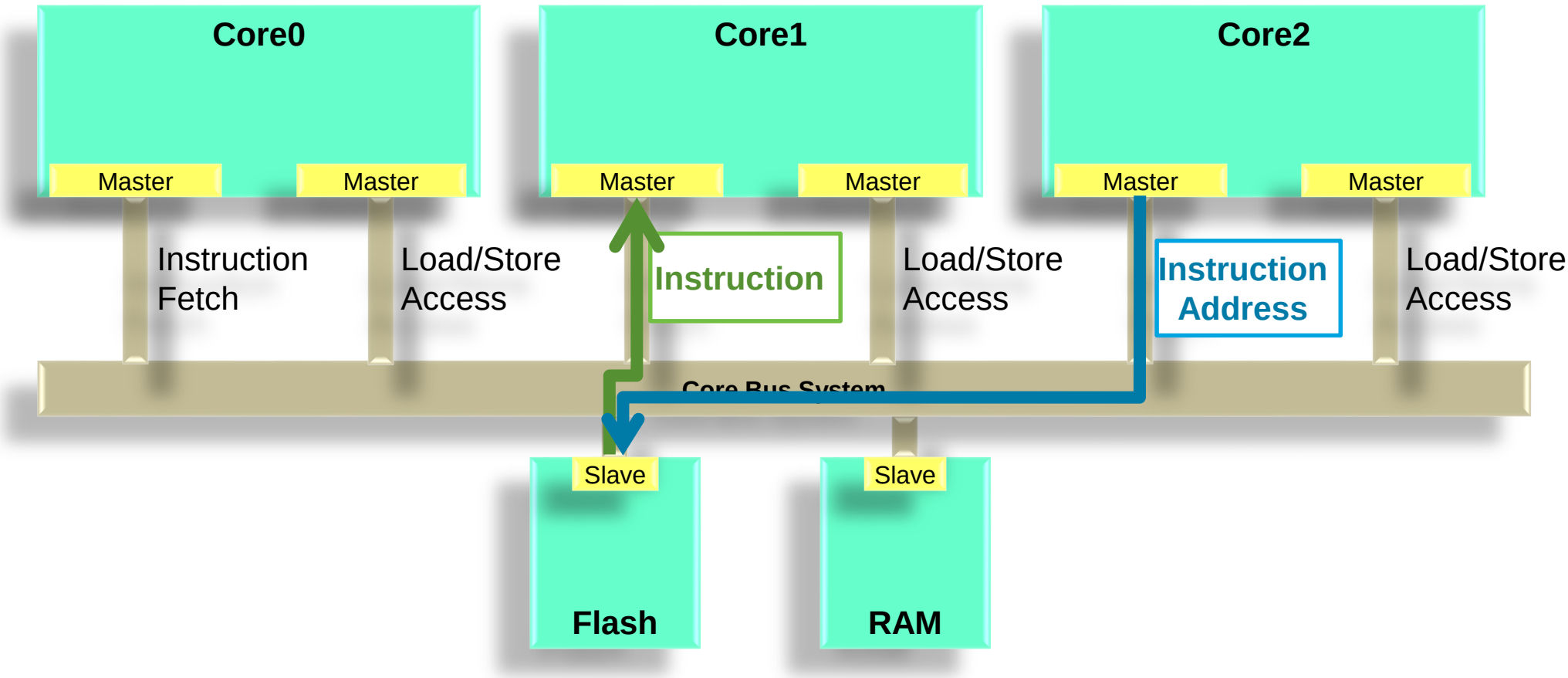
Multicore system with global program and data memory



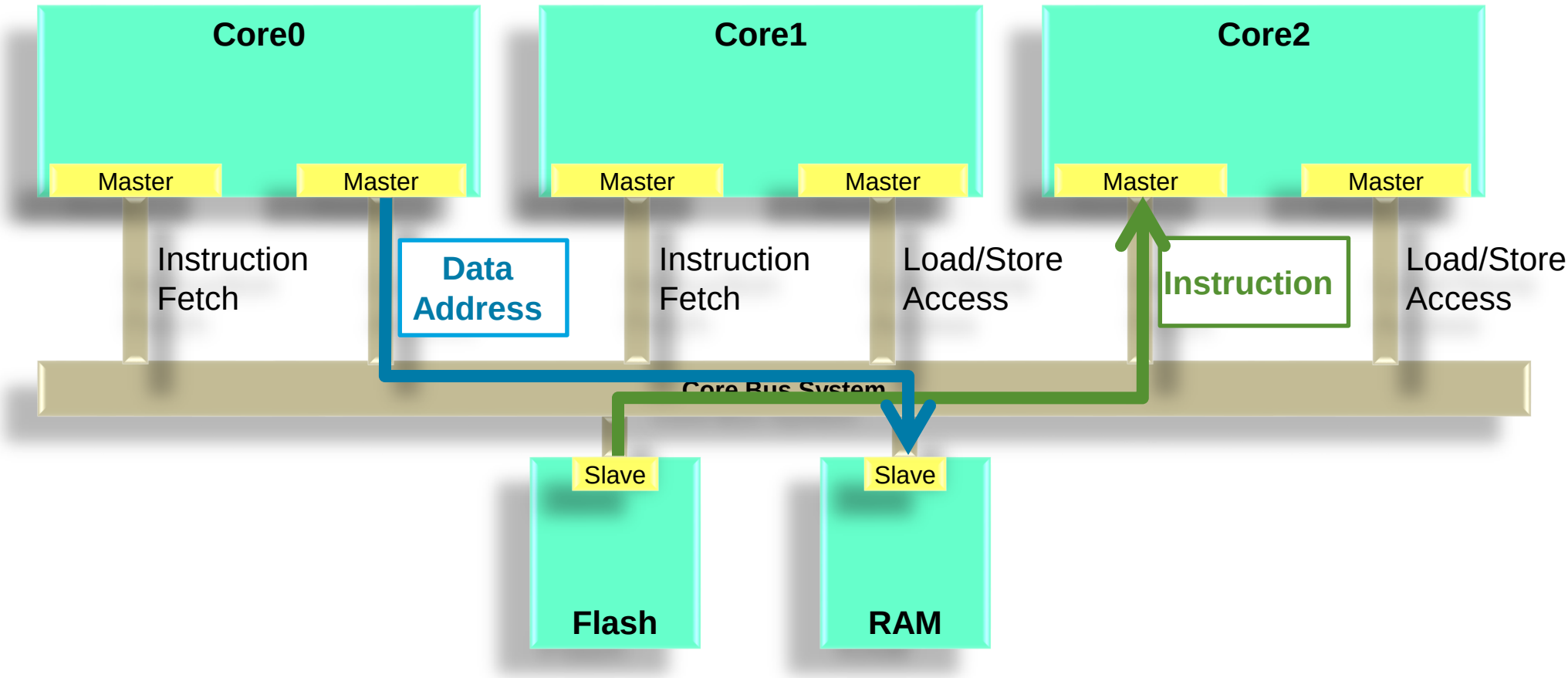
Multicore system with global program and data memory



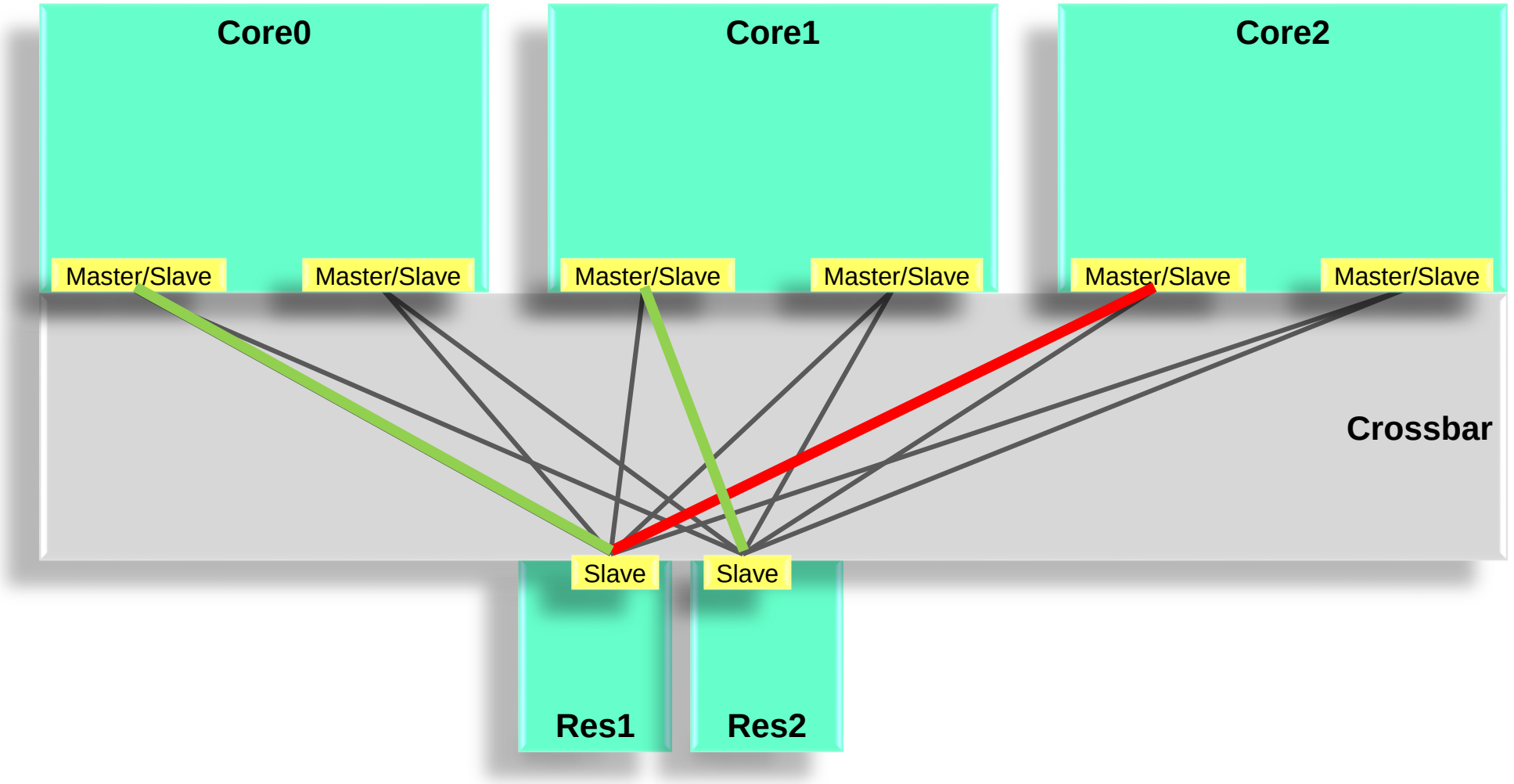
Multicore system with global program and data memory



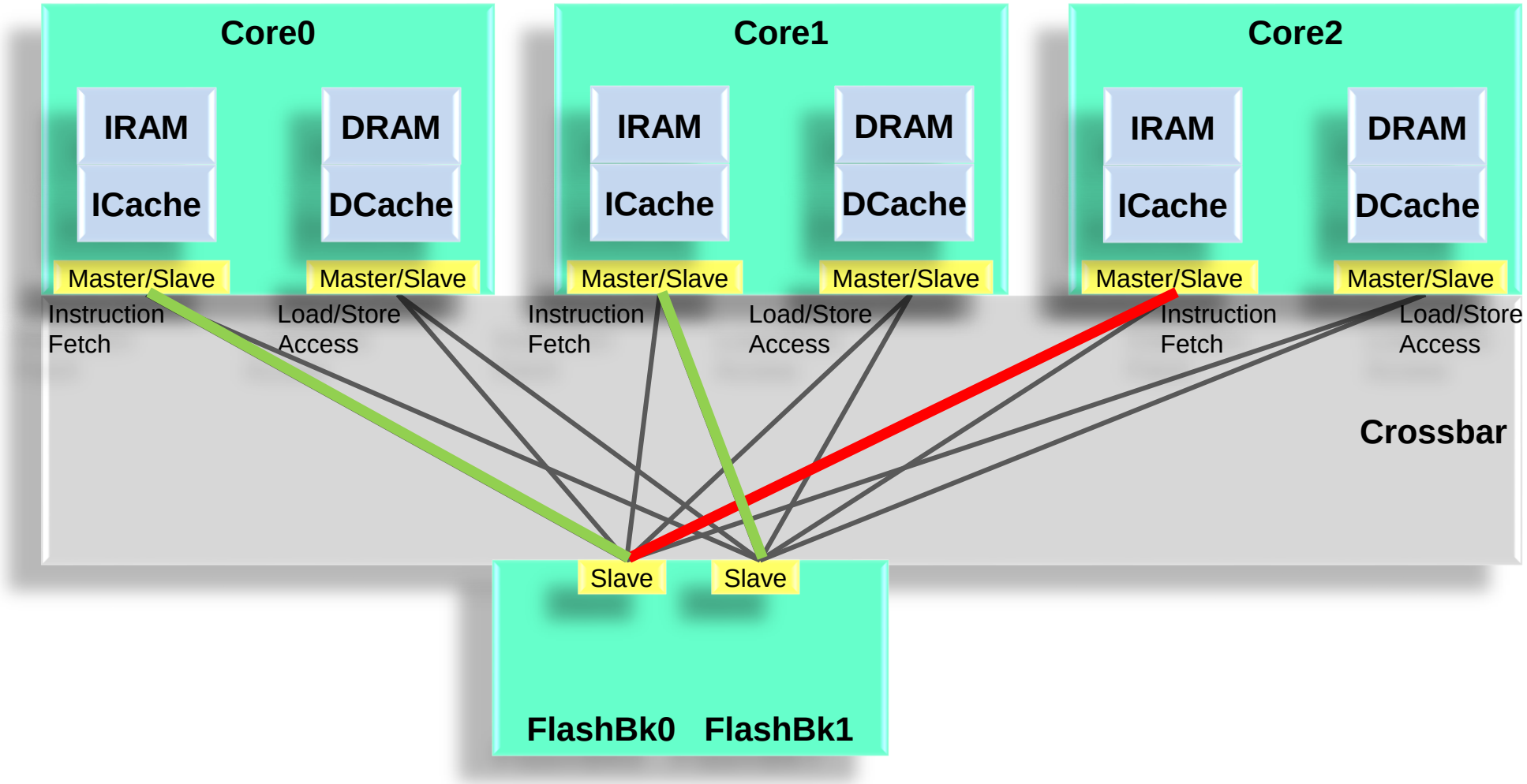
Multicore system with global program and data memory

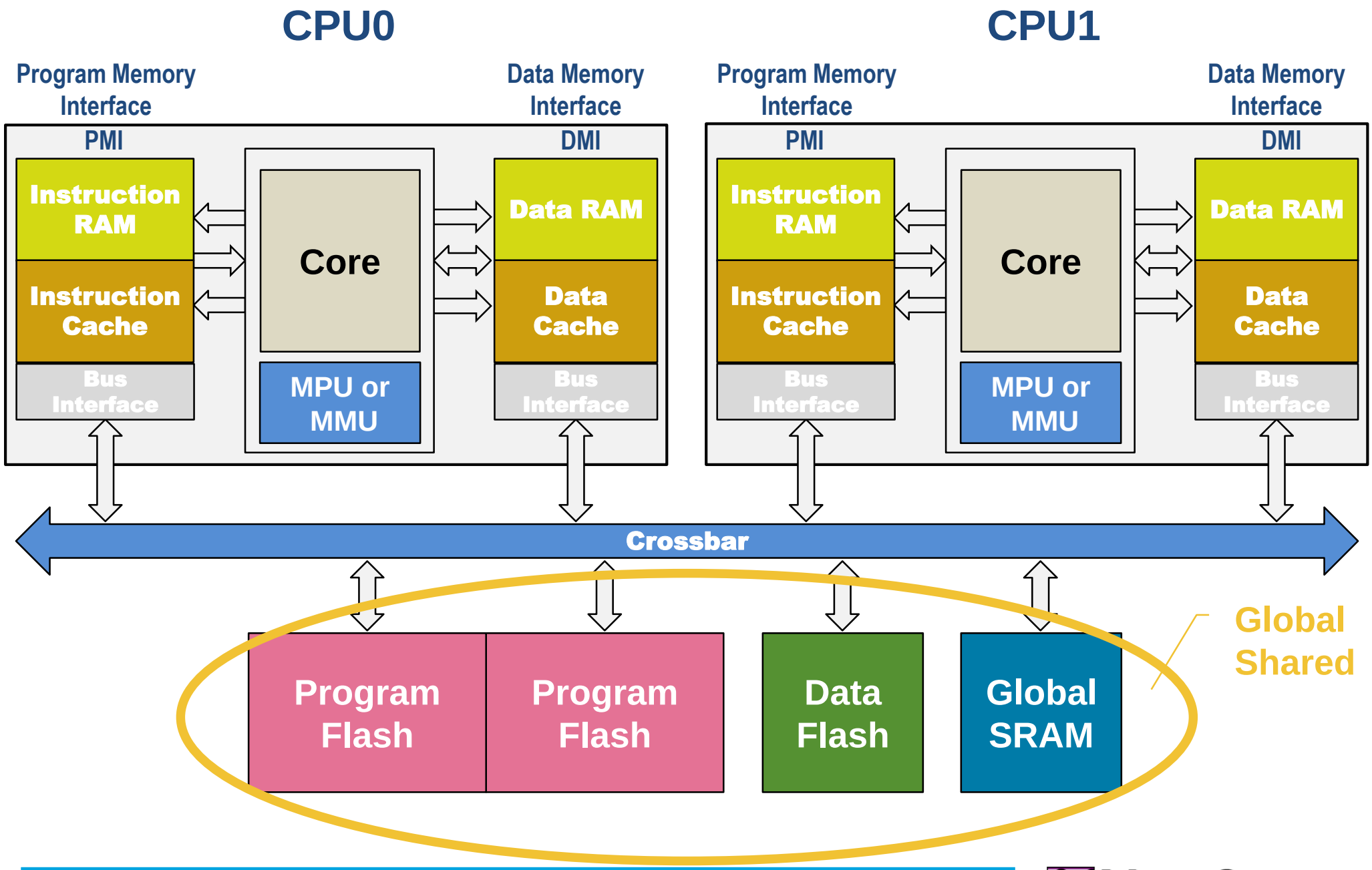


Multicore system with **multiple shared resources** (e.g. global memory):
Crossbar ⇨ concurrent access to **different** resources

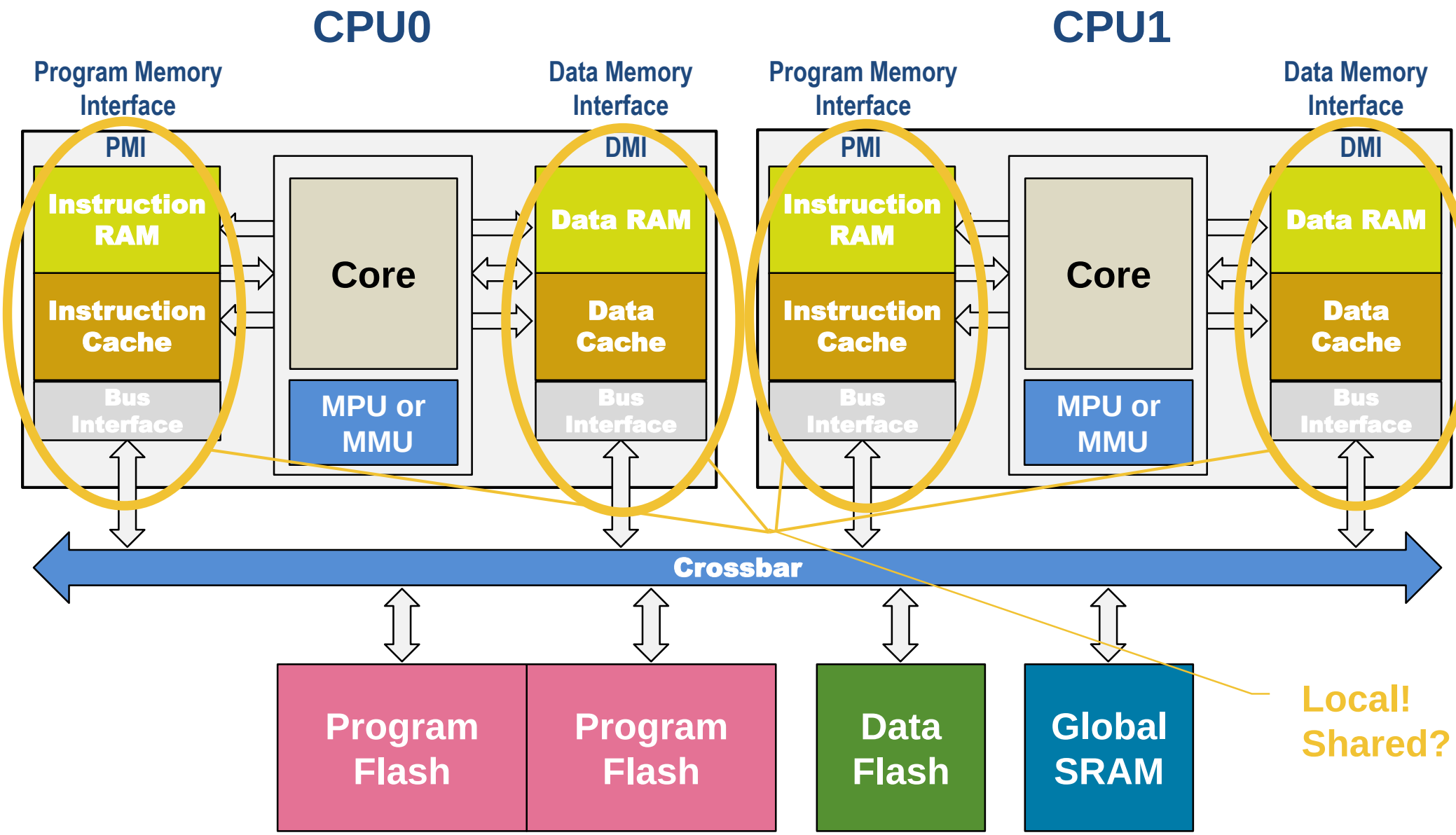


Multicore system with **local and global** memory:
Duplication of resources ➪ **concurrent** access to '**same**' resource

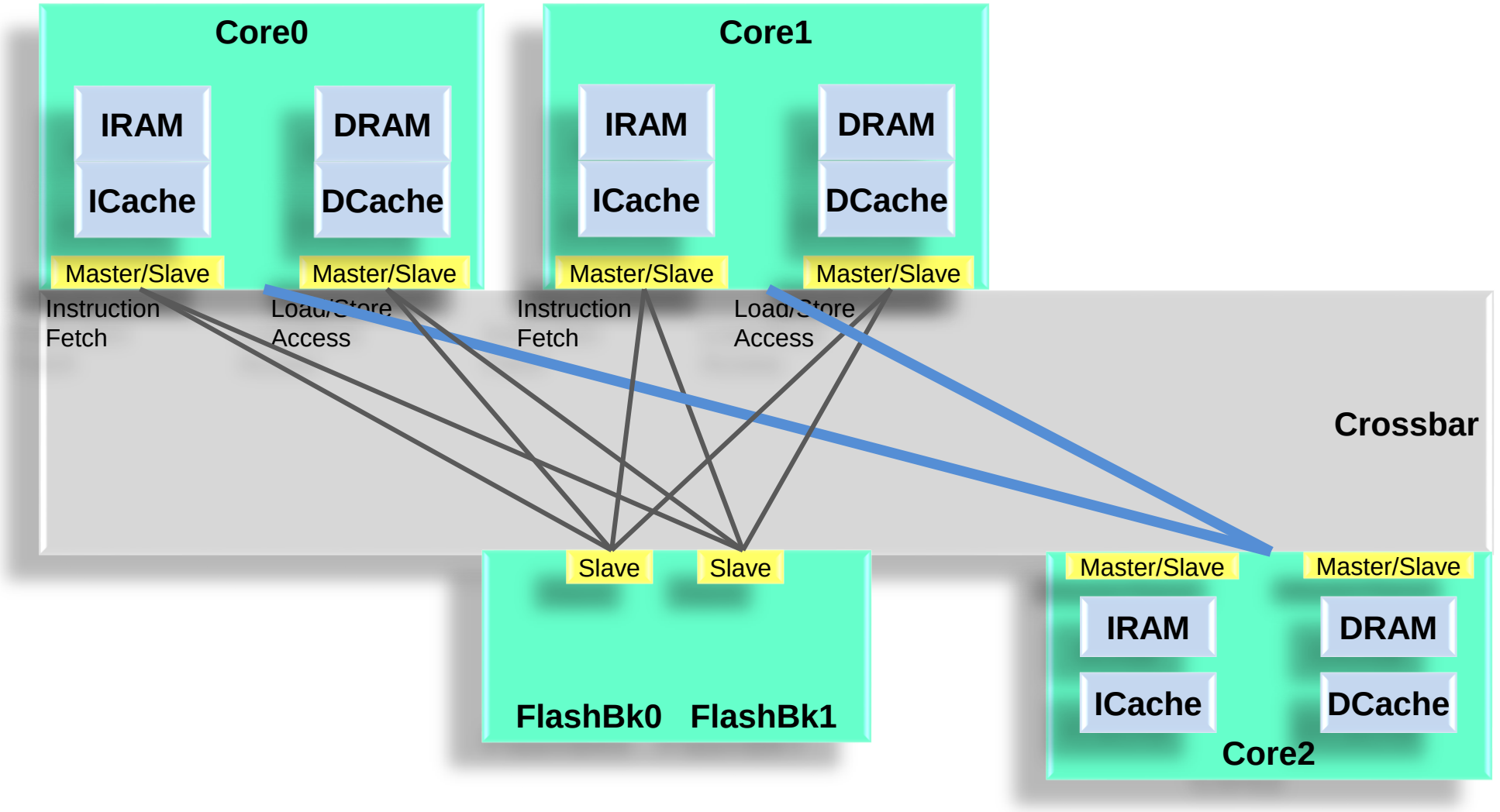




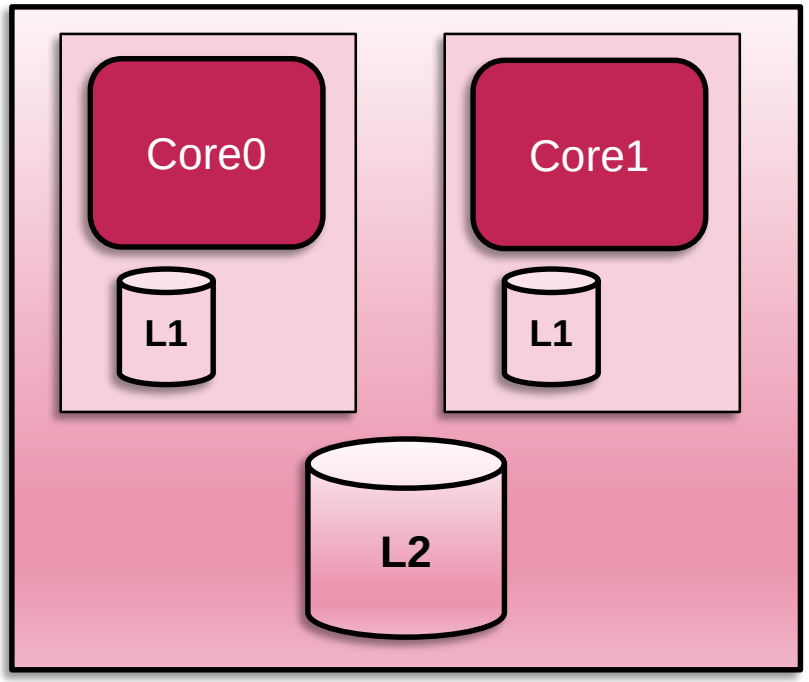
MICRO CONSULT


CPU0
CPU1
Program Memory Interface
Data Memory Interface
Program Memory Interface
Data Memory Interface
PMI
DMI
PMI
DMI
Instruction RAM
Instruction Cache
Bus Interface
Core
MPU or MMU
Data RAM
Data Cache
Bus Interface
Instruction RAM
Instruction Cache
Bus Interface
Core
MPU or MMU
Data RAM
Data Cache
Bus Interface
Crossbar
Program Flash
Program Flash
Data Flash
Global SRAM
Local!
Shared?

Multicore system with local and global memory:
Duplication of resources ➪ concurrent access to ‘same‘ resource



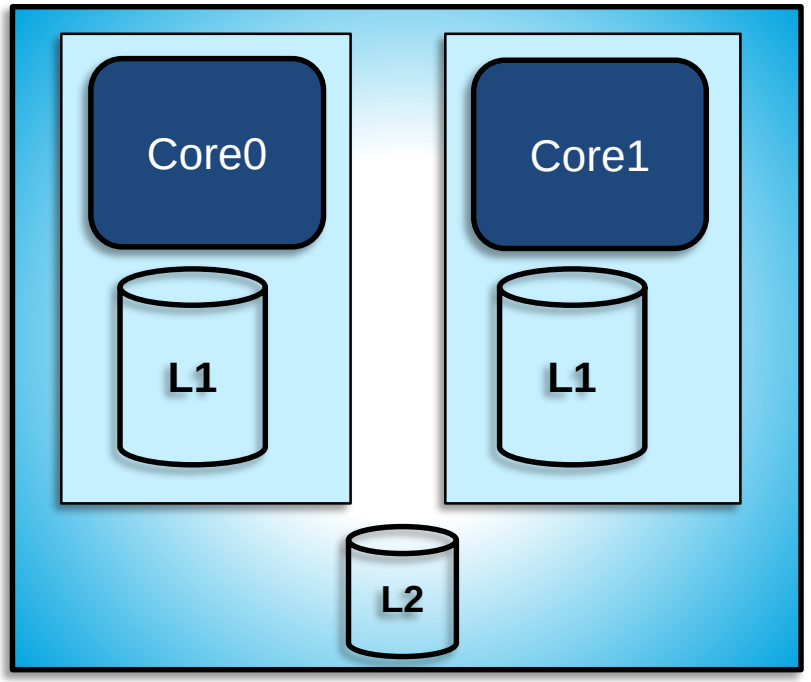
Multicore Microcontroller Type A



- **Small**, fast 1st level memory
  L1 (tightly coupled)
- Larger, slower 2nd level memory
  L2 (global/shared)

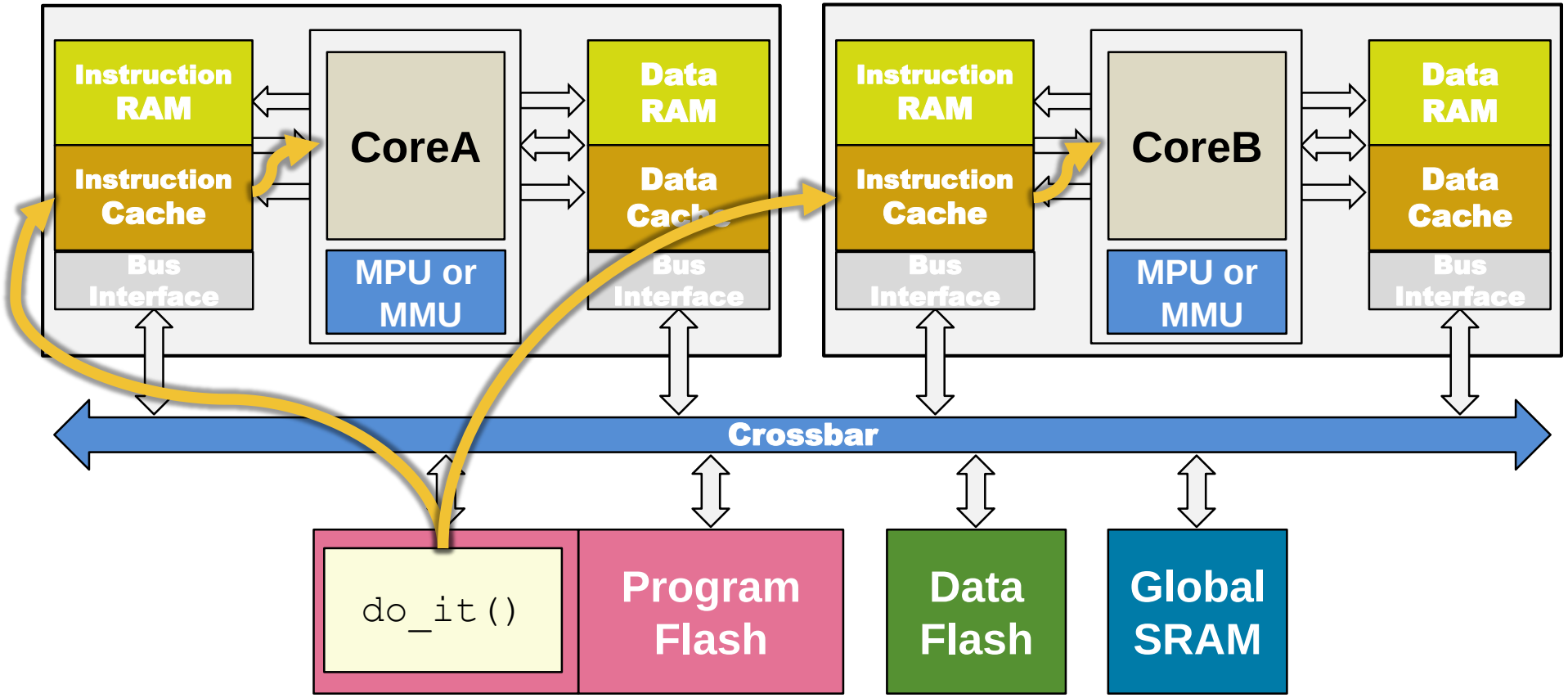
- Typical “desktop” model
- Relying on cache

Multicore Microcontroller Type B

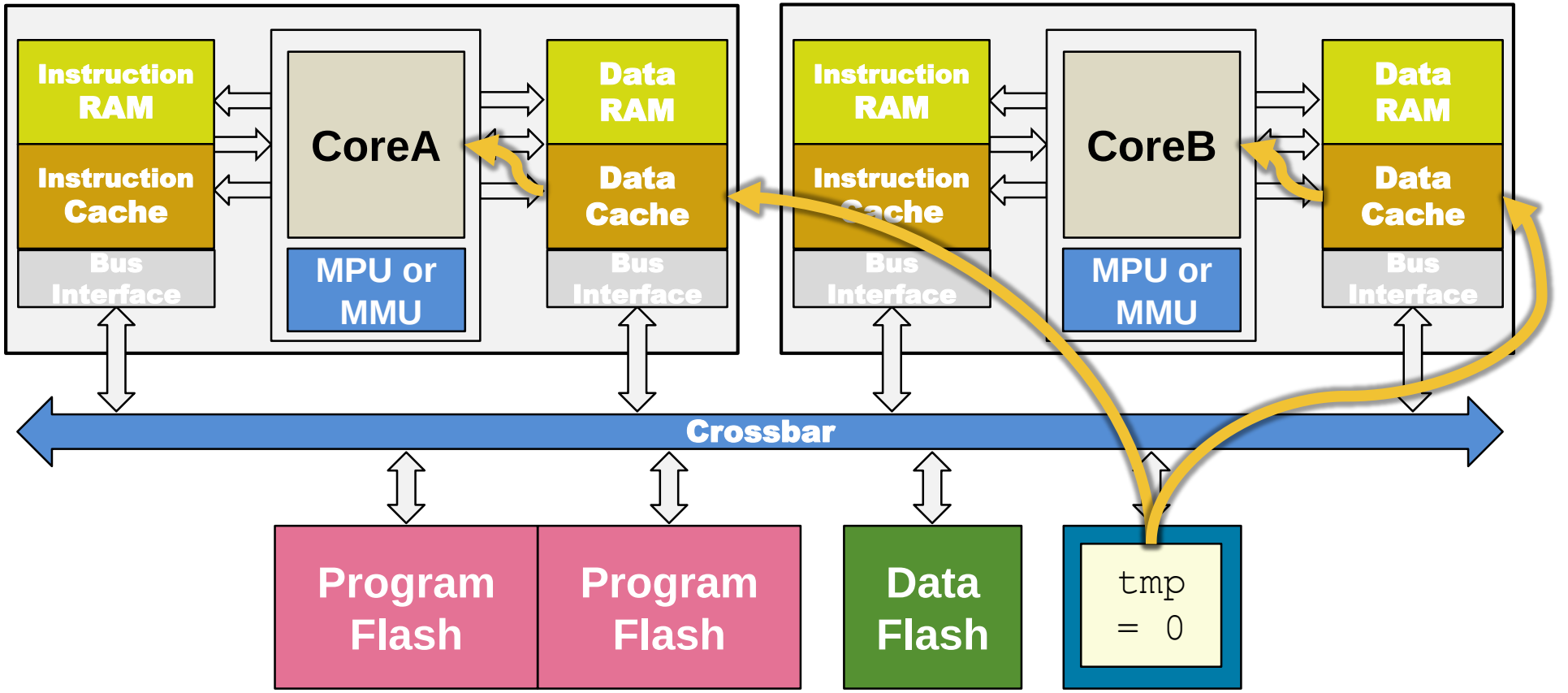


- “**Large**”, fast 1st level memory
  L1 (tightly coupled)
- Smaller, slower 2nd level memory
  L2 (global/shared)

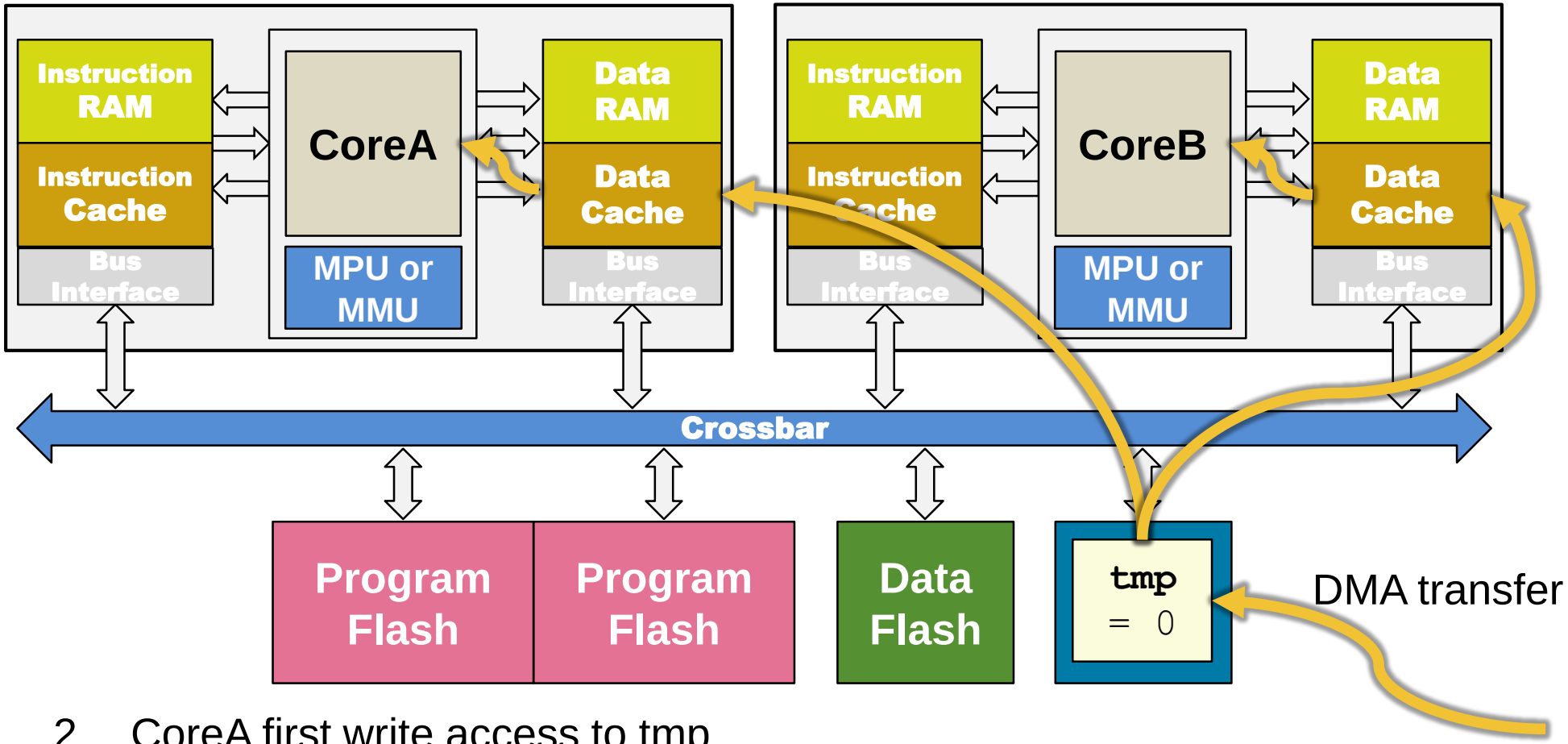
- Easier separation
- Optional cache



1. CoreA first call to do_it()
   - fetch do_something() → locally cached
2. CoreB first call to do_it()
   - fetch do_something() → locally cached
3. CoreA/B second call to do_it()
   - Execution from local caches

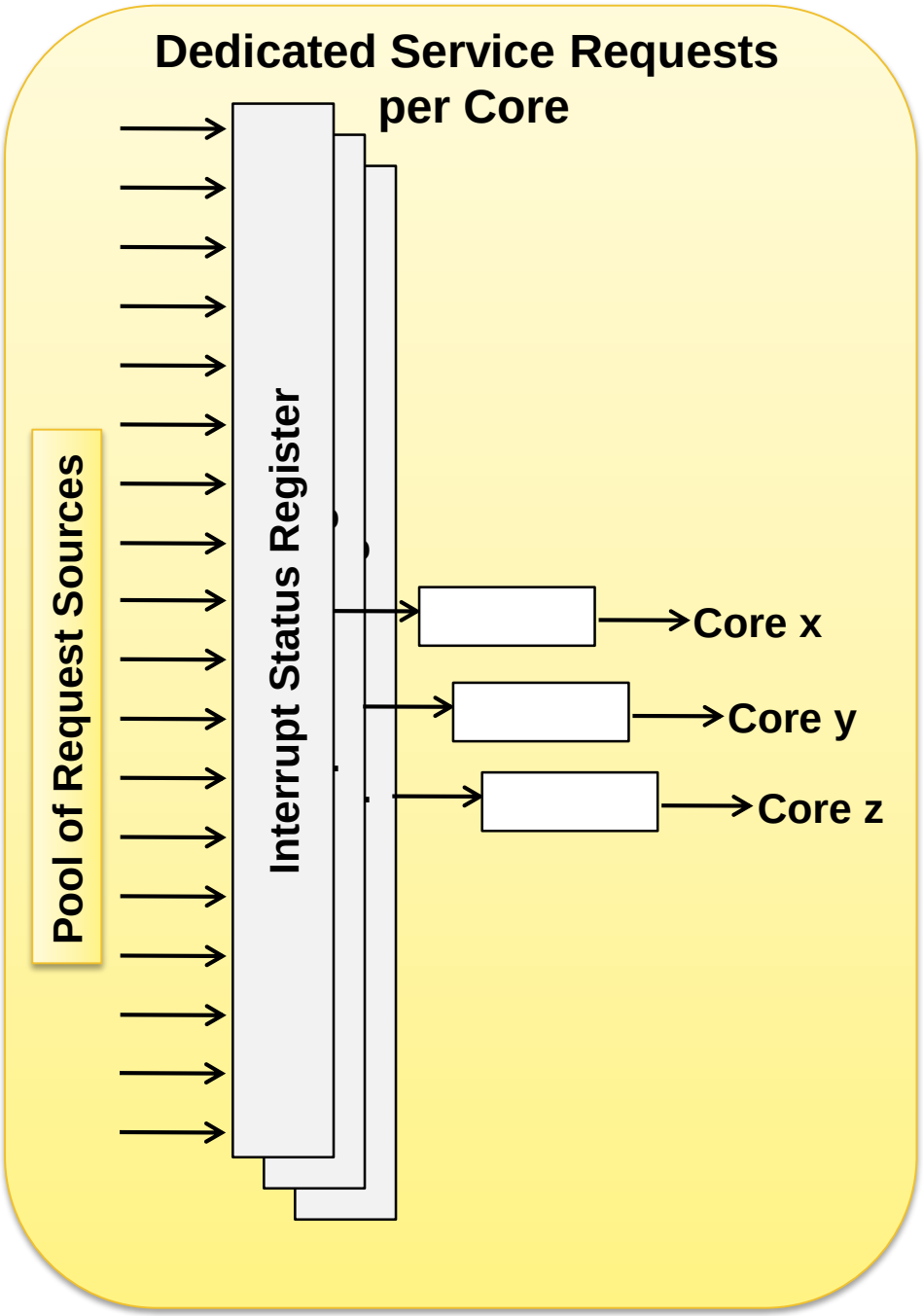


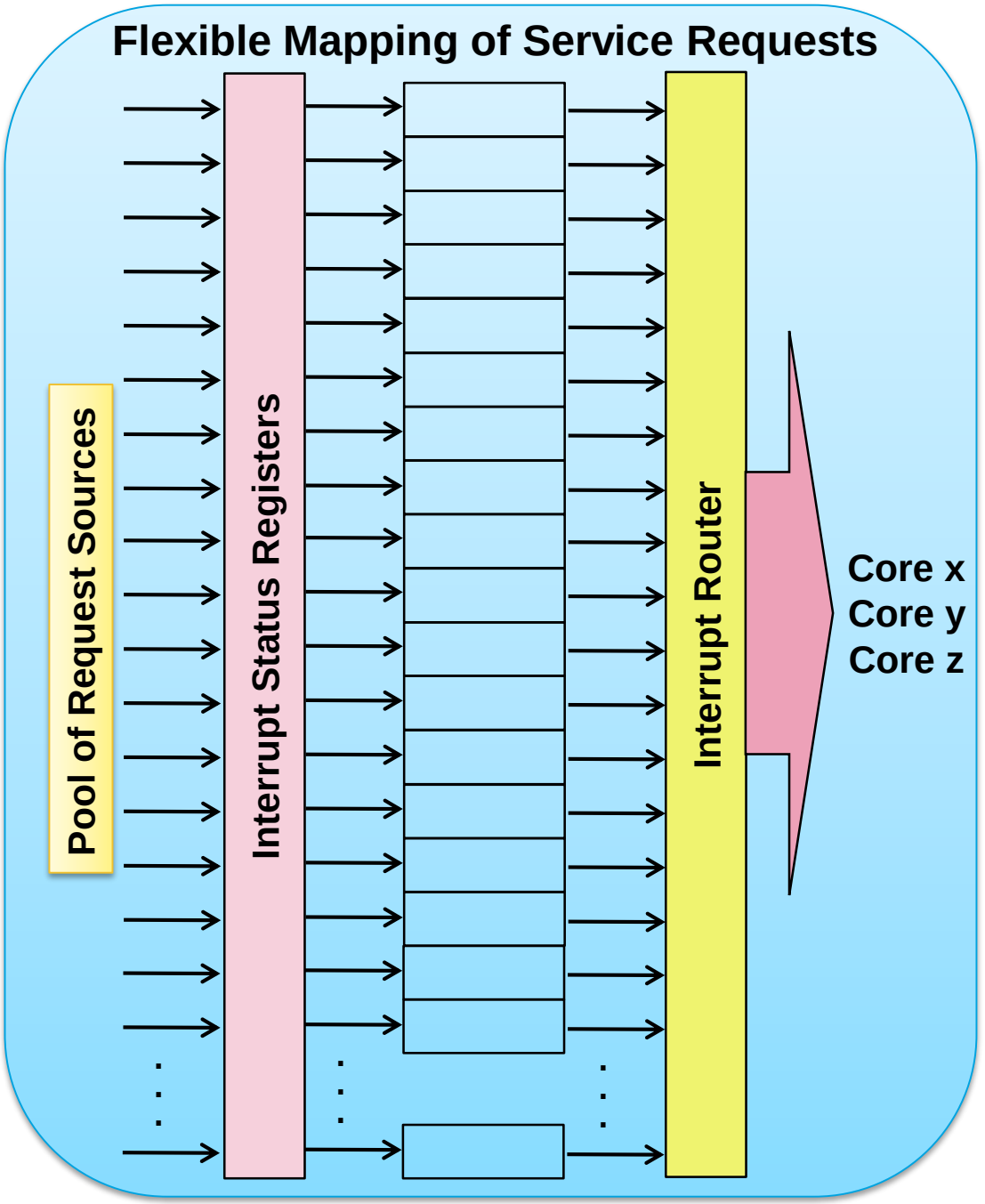
1. CoreA first read access to tmp
   - tmp = 0 → locally cached
2. CoreA first write access to tmp
   - tmp = 1 → locally cached
3. CoreB first read access to tmp
   - tmp = 0 → locally cached



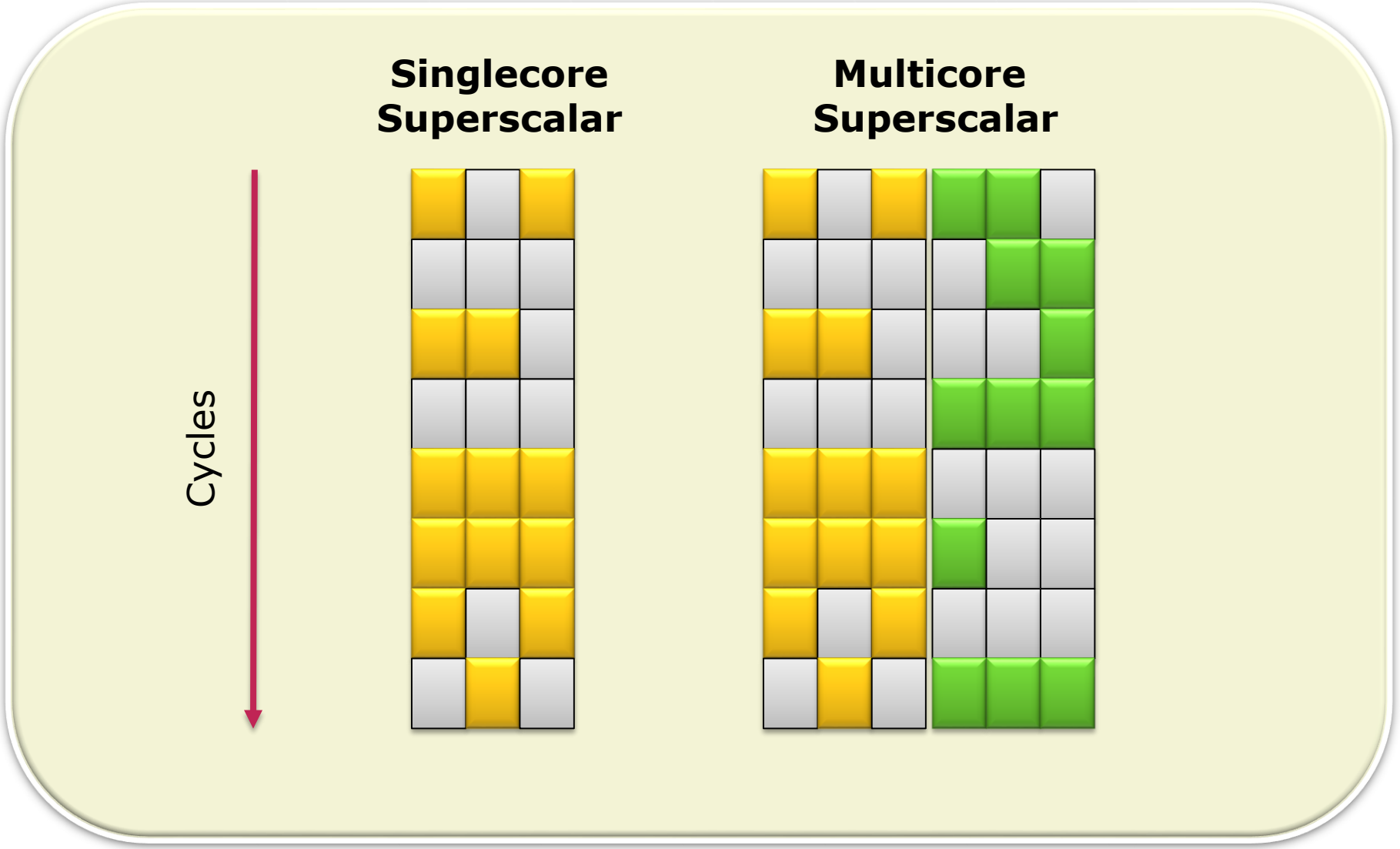
2. CoreA first write access to tmp
   - tmp = 1 → locally cached
3. CoreB first read access to tmp
   - tmp = 0 → locally cached
4. DMA write access to tmp
   - tmp = 2 → global SRAM

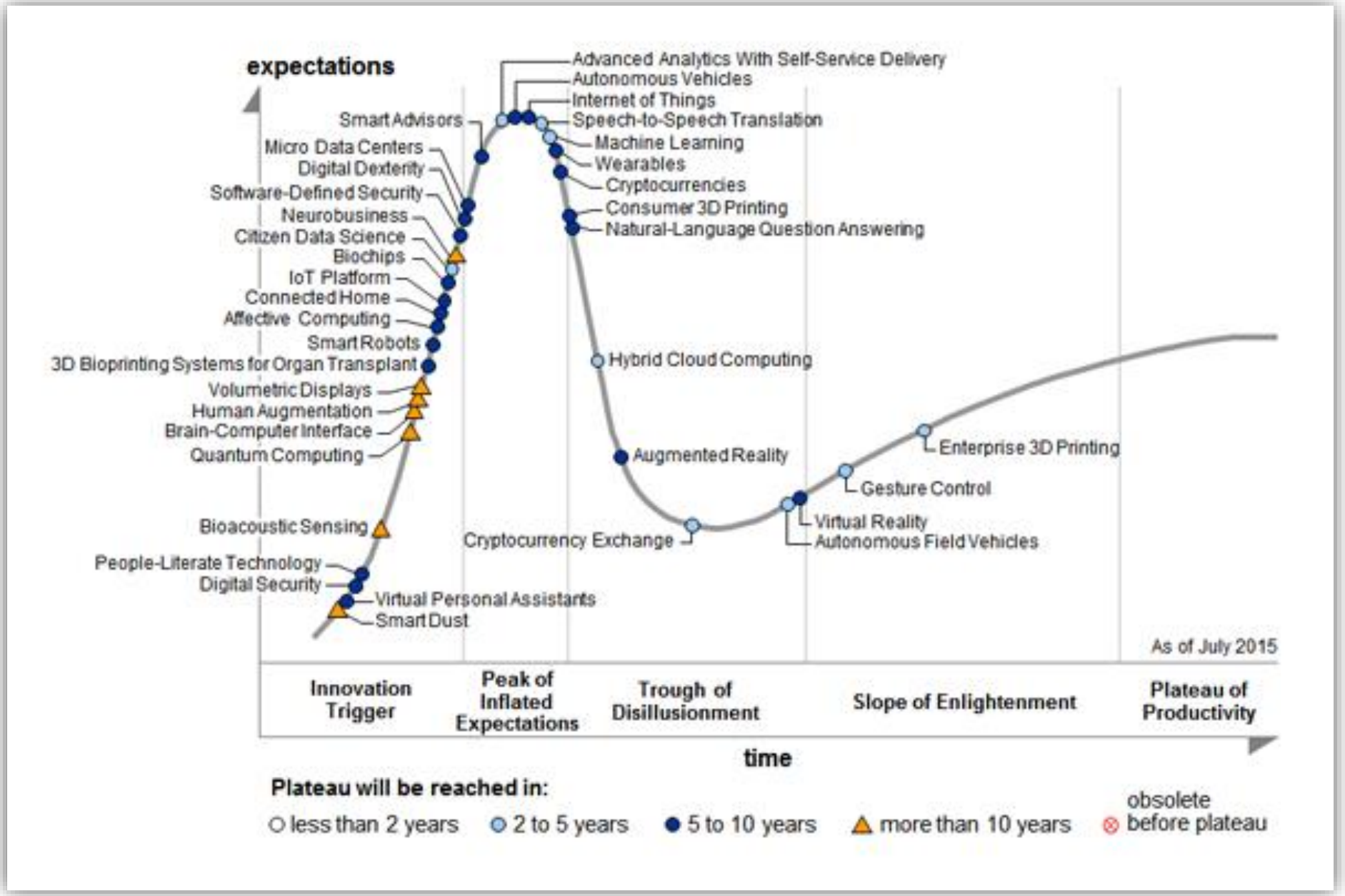
- **Disable** cache
  - Enable instruction cache
  - Enable data cache if absolutely needed
- Utilize **cache APIs**
  - Invalidate
  - Write back
  - Flush (write back invalidate)
- Depending on memory map and cache architecture
  - Define which memory **ranges** to cache
  - Same memory addressable in cached and non-cached **mode**
- **Snooper** (on global bus)
  - Snoop-hit on write
    - e.g. DMA write to global SRAM → cache invalidate
  - Snoop-hit on read
    - e.g. CoreB read from global SRAM → cache write back before read


Dedicated Service Requests per Core
Pool of Request Sources
Interrupt Status Register
Core x
Core y
Core z


Flexible Mapping of Service Requests
Pool of Request Sources
Interrupt Status Registers
Interrupt Router
Core x
Core y
Core z

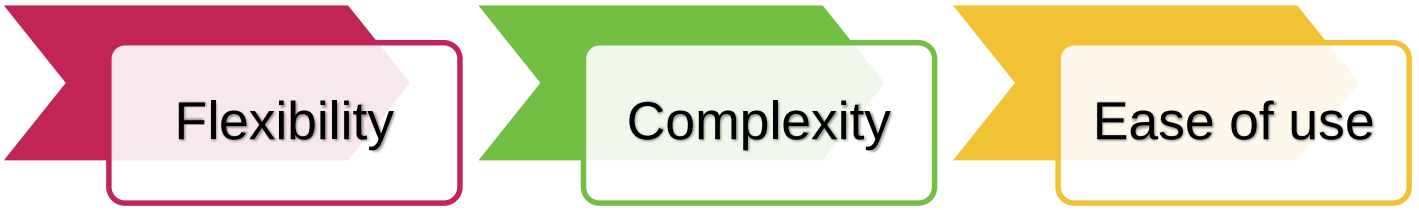
Aspect	Comment
Crossbar	Still bottleneck, hardware dependency is high
Memory protection	Needs to cover every resource, granularity and protection ranges
Inter-core communication	Only one core should pay the bill for shared memory
Synchronization	Additional instructions, atomic accesses
CPU	More of the same (not just look and feel)
Memory	Shared vs distributed model
Portability	Multicore ain’t multicore
Share the pain	Higher software efforts


Singlecore
Superscalar
Multicore
Superscalar
Cycles



Ref. http://www.gartner.com/newsroom/id/3114217

From	To
Multicore	Manycore
Heterogeneous	Homogeneous
High-speed IO	Higher-speed IO
SW/HW virtualization	HW/SW virtualization
Security	Holistically secure
Power modes	Automatic power scaling
Crossbar	Interconnection networks



Take mother nature as a reference:

Mapped to the space of multicore

- Many small cores
- Largely scalable
- Universal approach

→ Homogeneous multicore systems
→ Dynamically adjustable
→ Software friendly

- Food for thought and action
- Predictions may become true or prove false
- Statements are personal but driven and backed by experience
- The future remains to be seen

It's hard to make predictions, especially about the future.

…but be aware that you influence the future as well.