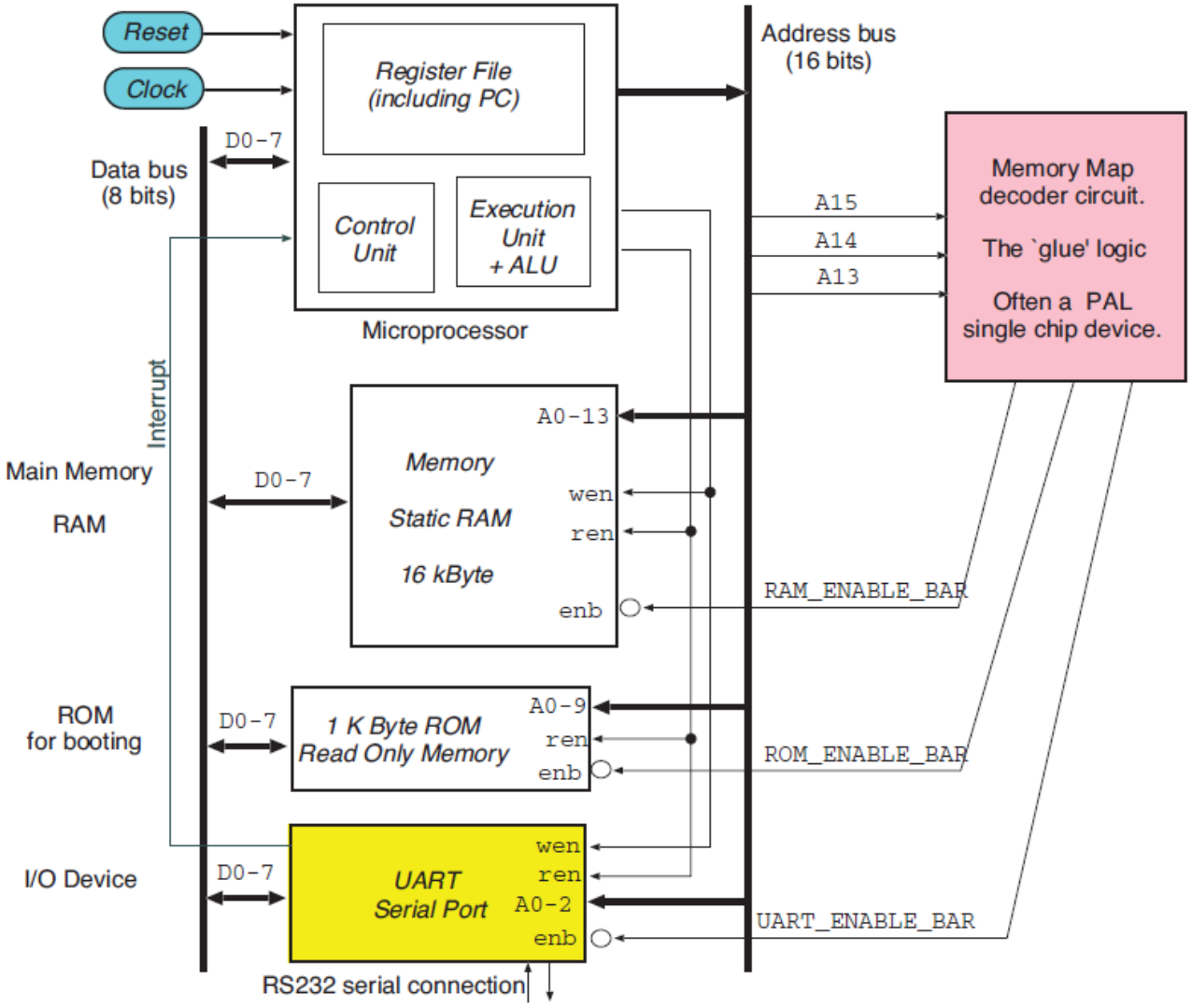
Gestione Periferiche - DMA e BUS

Andrea Bartolini – a.bartolini@unibo.it

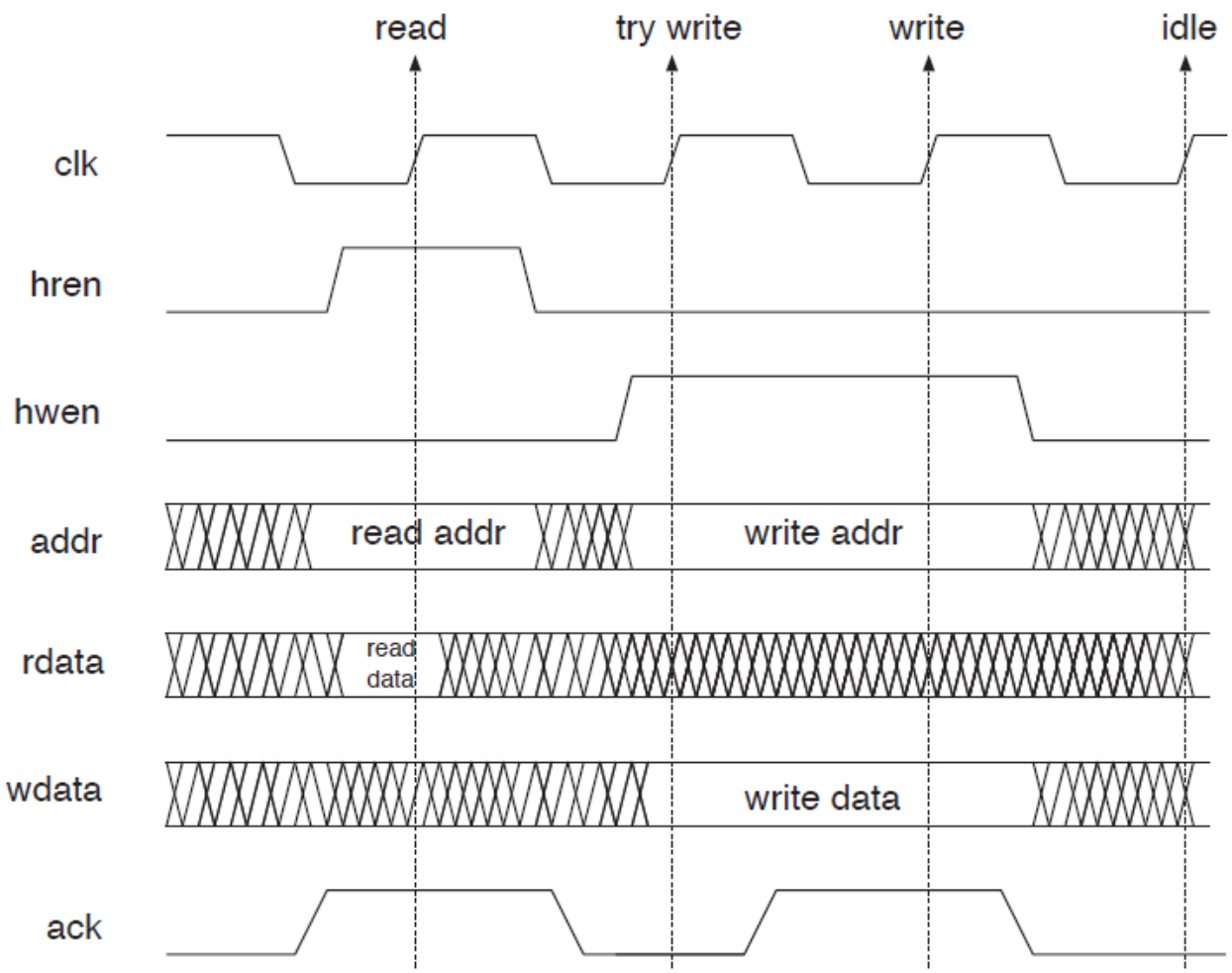
Architettura dei calcolatori T



Tri-state bus not used in on-chip interconnect.

Point-to-point wiring are preferred to achieve a lower switched-capacitance and reducing the leakage energy in logic gates where the input is floating between logic levels.

Simple syncronous bus



Read followed by a write. The write is extended by one clock cycle as the ack signal was not present on the first positive edge when hwen was asserted.

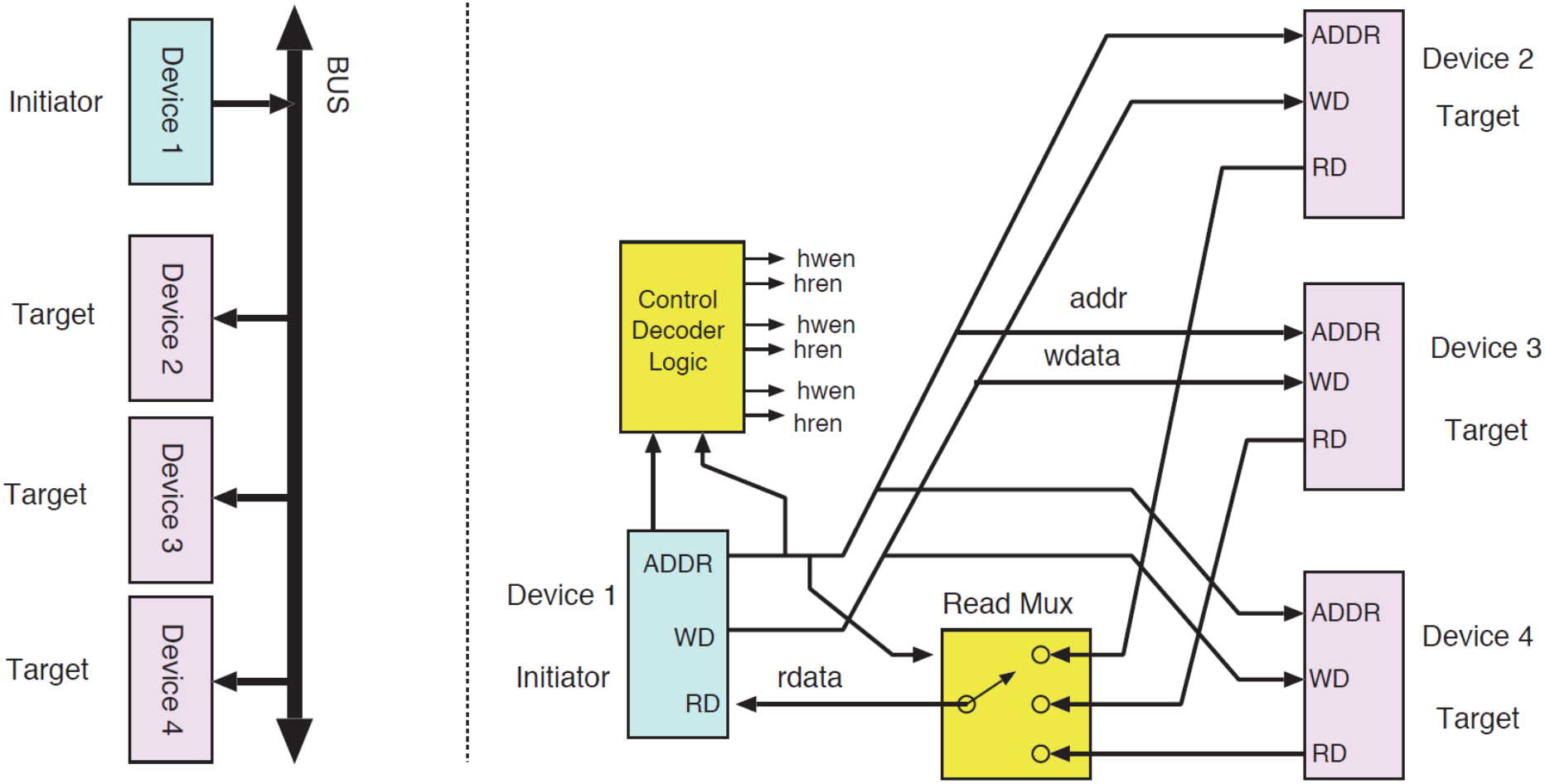
Synchronous BUS => transactions happens on positive edge of clk signal

Handshake

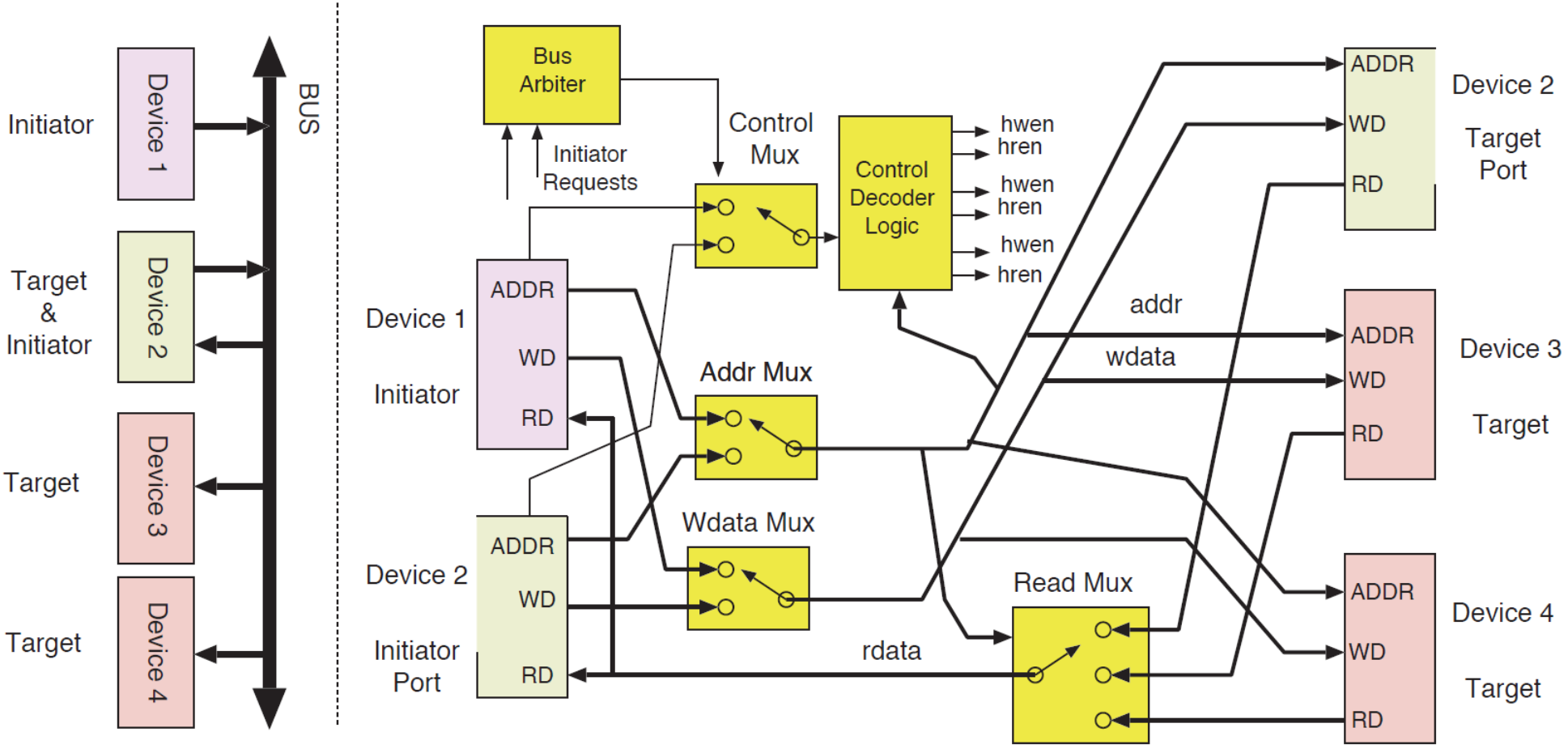
1. Initiator issues hwen/ hren signals
2. Target acknowledge the command with ack signal

- addr[31..0]: select address
- hwen: asserted by initiator during a write to target operation
- hren: asserted by initiator during a read from target operation
- wdata[31..0]: data from initiator during a write operation
- rdata[31..0]: data from target during a read operation
- ack: asserted by target when ready.

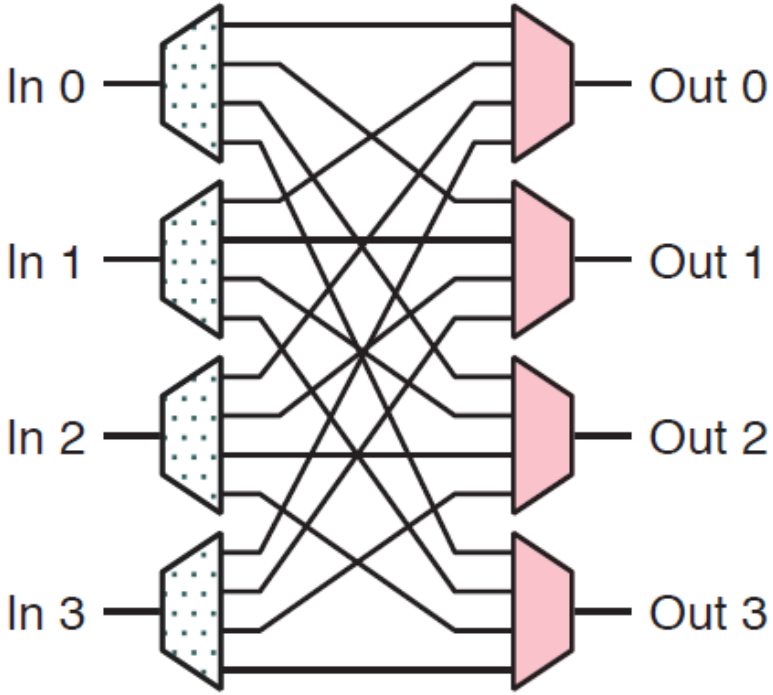
Bus Fabbric – Simple w. one initiator



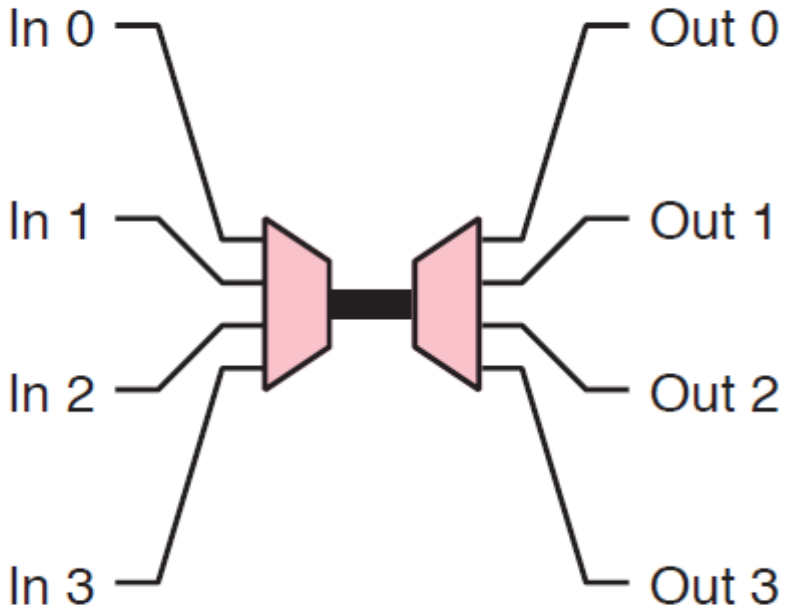
Bus Fabbric – Simple w. multiple initiator



Crossbar

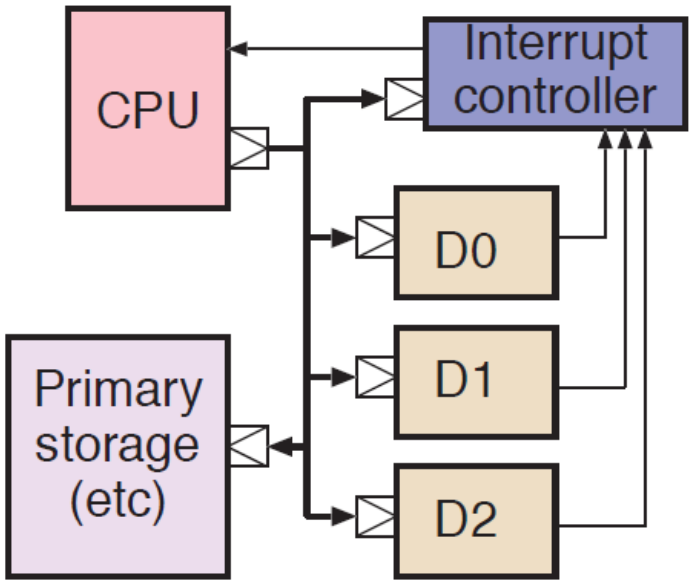


Multiplexer - demultiplexer



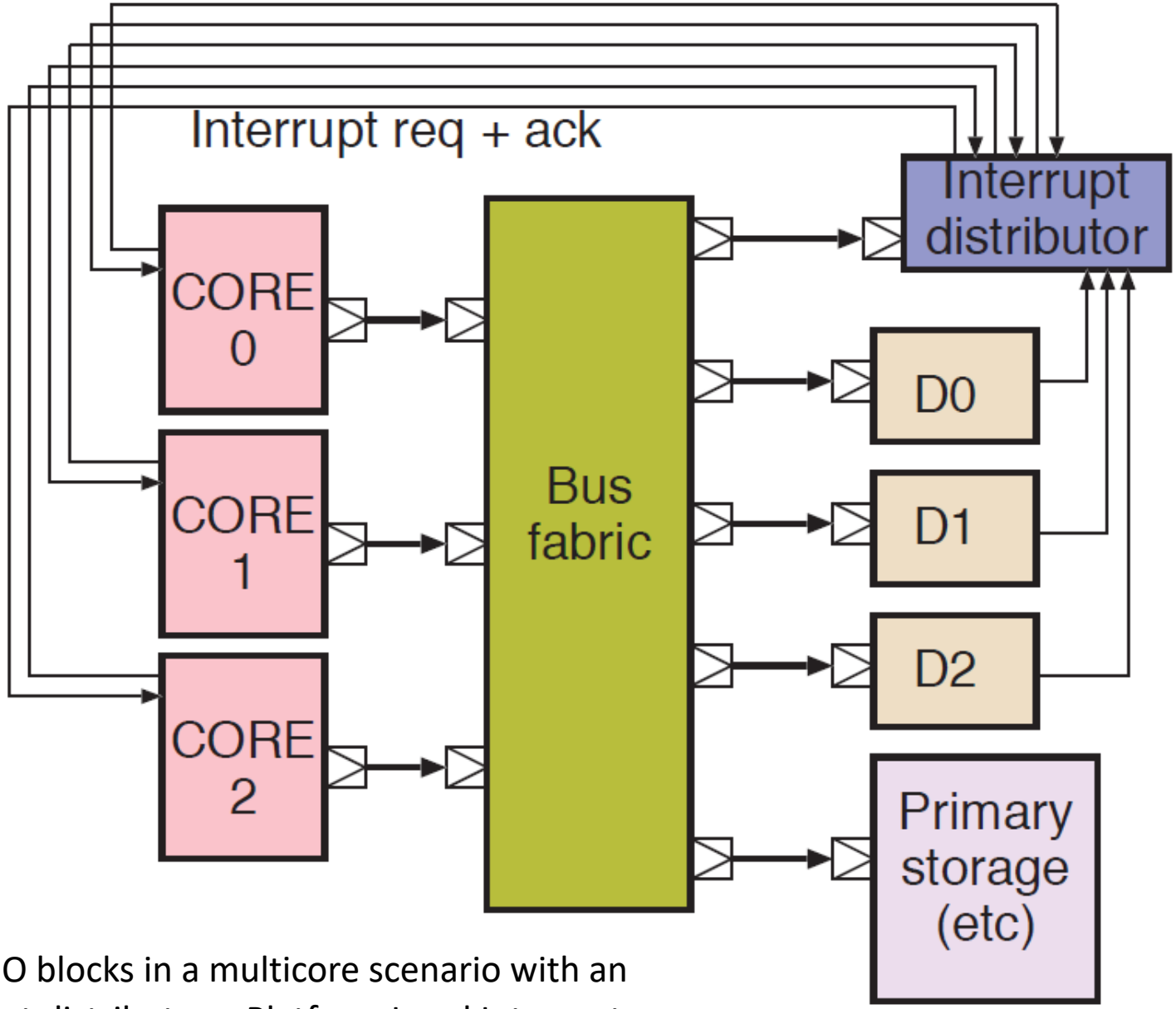
TDM: Time-division multiplexed

Interrupt Controllers



Three IO blocks connected to a CPU, MEM, Core level Interupt Controller (PIC)

Interrupt Controllers



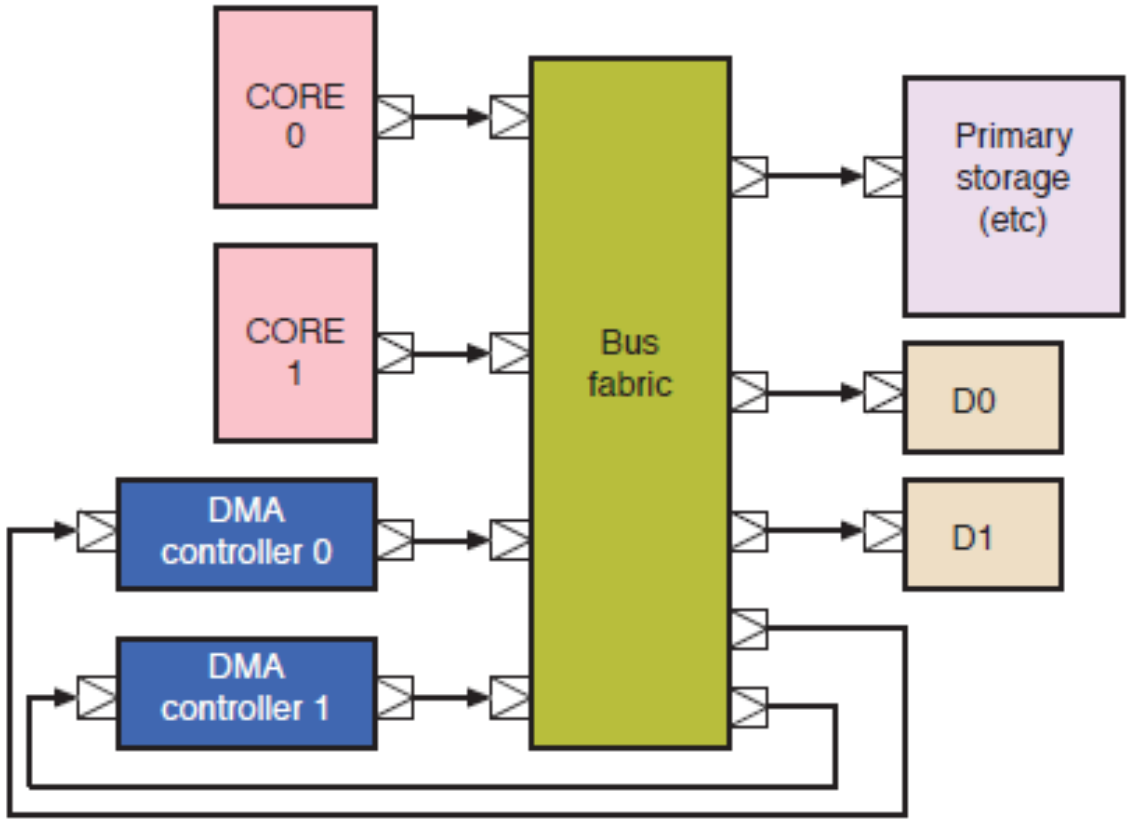
Three IO blocks in a multicore scenario with an interrupt distributor – Platform Level Interrupt Controller (PLIC)

Gestione periferiche

:

Direct Memory Access:

- Introduciamo un controllore HW per gestire direttamente i trasferimenti dati tra periferica e memoria centrale. Senza coinvolgimento della CPU.
- Direct Memory Access Controller (DMAC)



Direct Memory Access Controller (DMAC):

L'architettura di un DMAC prevede come minimo:

I. Un contatore del numero di parole da trasferire;

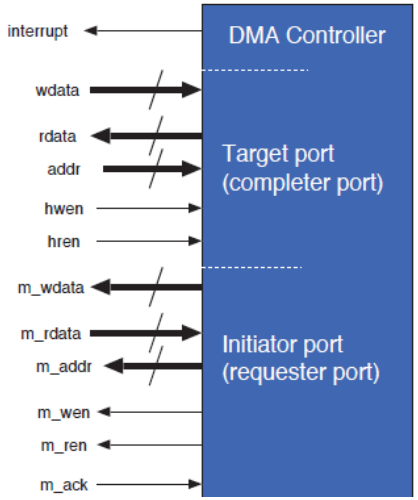
II. Un puntatore alla posizione della memoria in cui verrà letto/scritto il prossimo dato;

III. Un registro di comando che indichi il tipo di trasferimento;

IV. Un eventuale registro di stato.

V. Un registro per mascherare eventi (Mask Register)

Questo fa riferimento ad un ad un DMAC con un singolo canale ciascuno dei quali presenta un suo registro contatore ed un suo registro di puntatore.

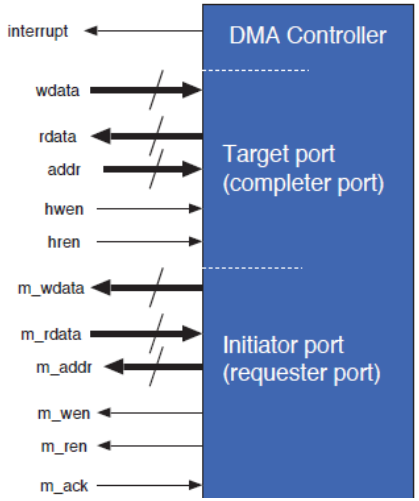


Direct Memory Access Controller (DMAC):

Il funzionamento di un DMAC prevede due fasi mutualmente esclusive (una esclude l'altra):

1) Fase di programmazione: Serve a trasmettere al DMAC le informazioni che definiscono la modalità di trasferimento sul canale.
   - Il controllore viene visto come una periferica dotato di un insieme di registri.
   - Si programmano i registri puntatore, contatore e di comando. Si controlla lo stato.

2) Fase di trasferimento dei dati: Si distinguono due modalità:
   - Trasferimento singolo (cycle stealing): il DMAC occupa il bus per un ciclo (pochi cicli) necessari al trasferimento di un dato da/verso la memoria.
   - Trasferimento a blocchi (burst): prevede l'occupazione del bus per tutto il tempo richiesto a trasferire il numero di parole scritto nel contatore in fase di programmazione.

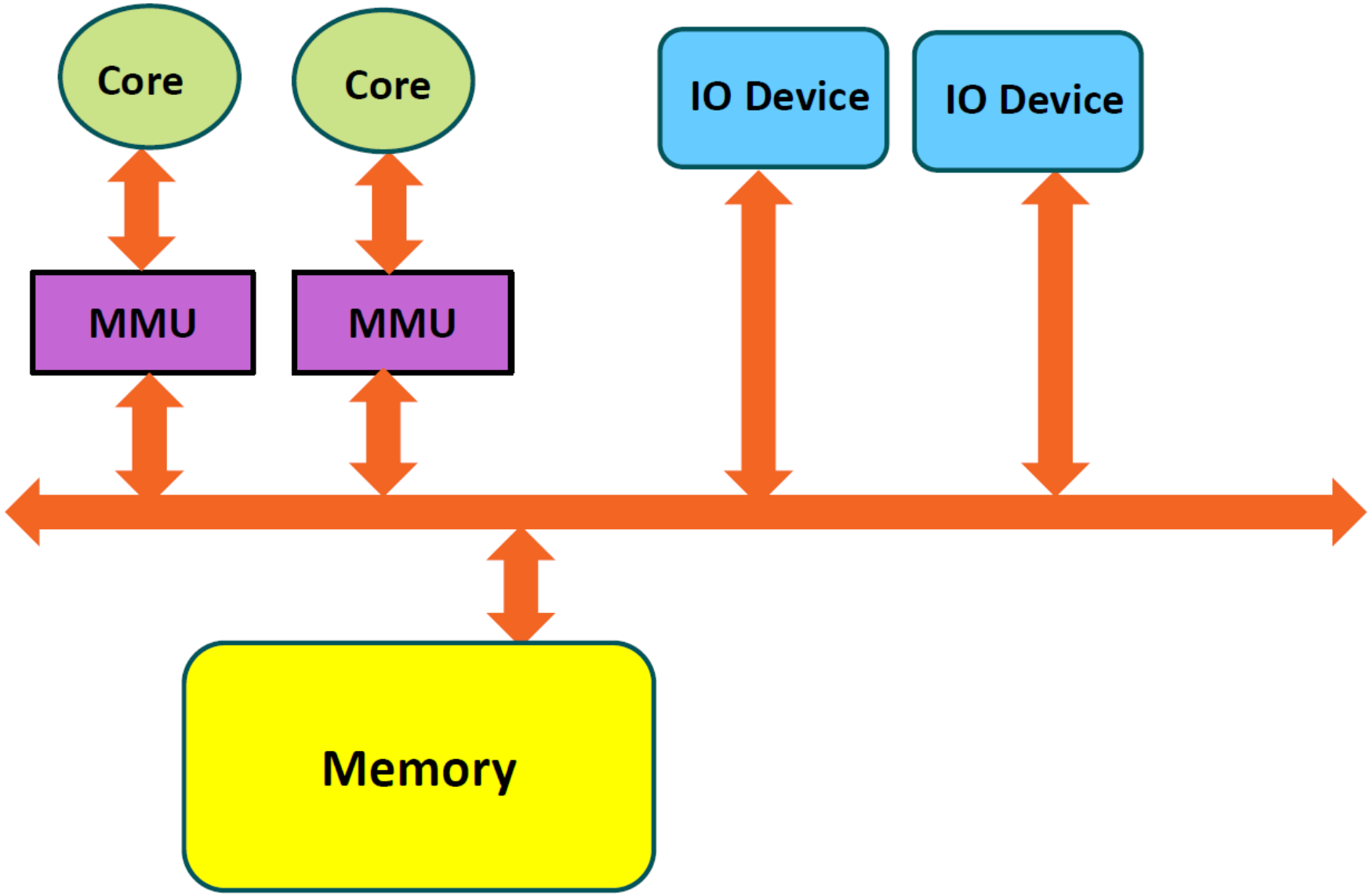


Virtual Memory e periferiche

Andrea Bartolini – a.bartolini@unibo.it

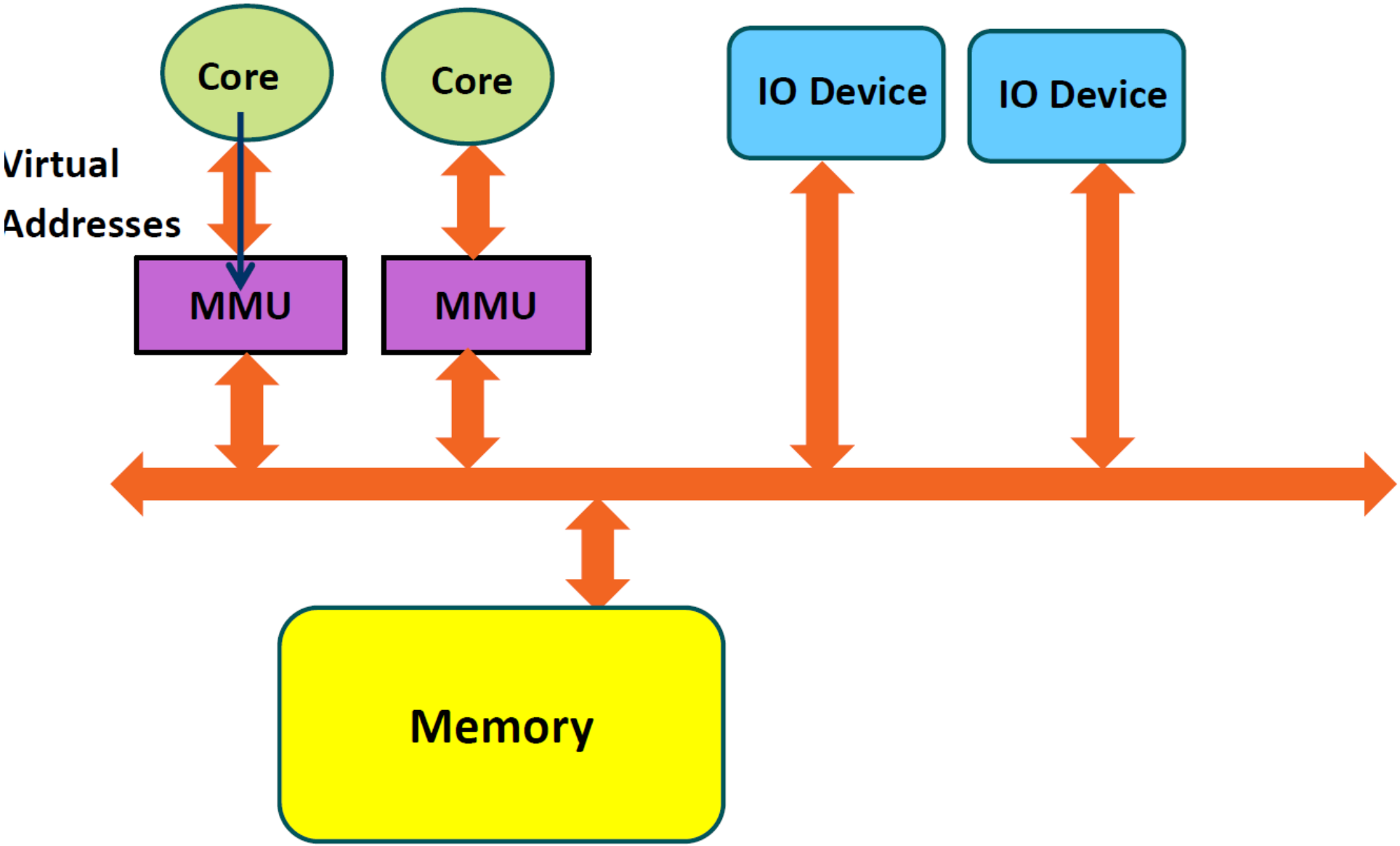
MOTIVATION: TRADITIONAL DMA BY IO

NO SYSTEM VIRTUALIZATION



MOTIVATION: TRADITIONAL DMA BY IO

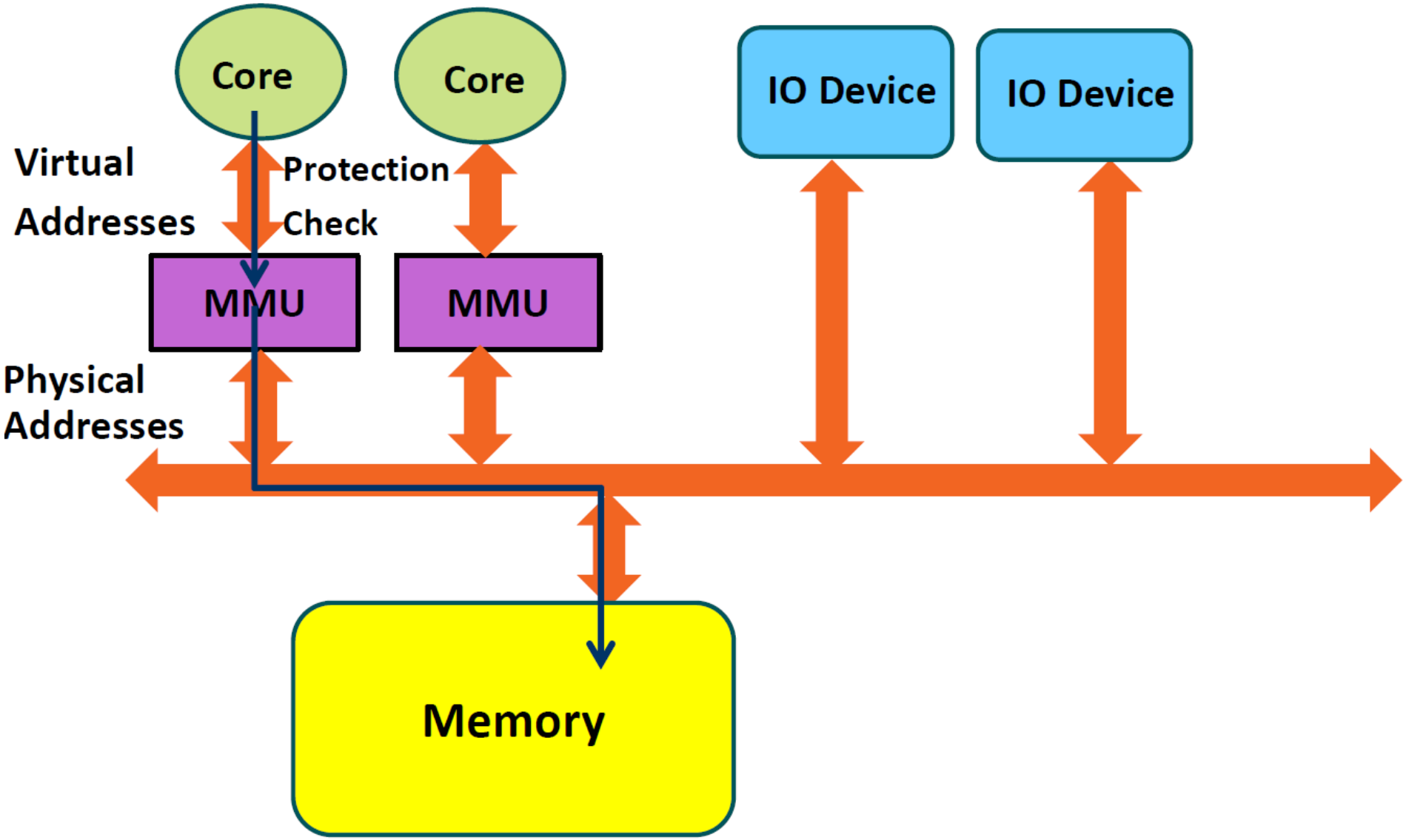
NO SYSTEM VIRTUALIZATION



MOTIVATION: TRADITIONAL DMA BY IO

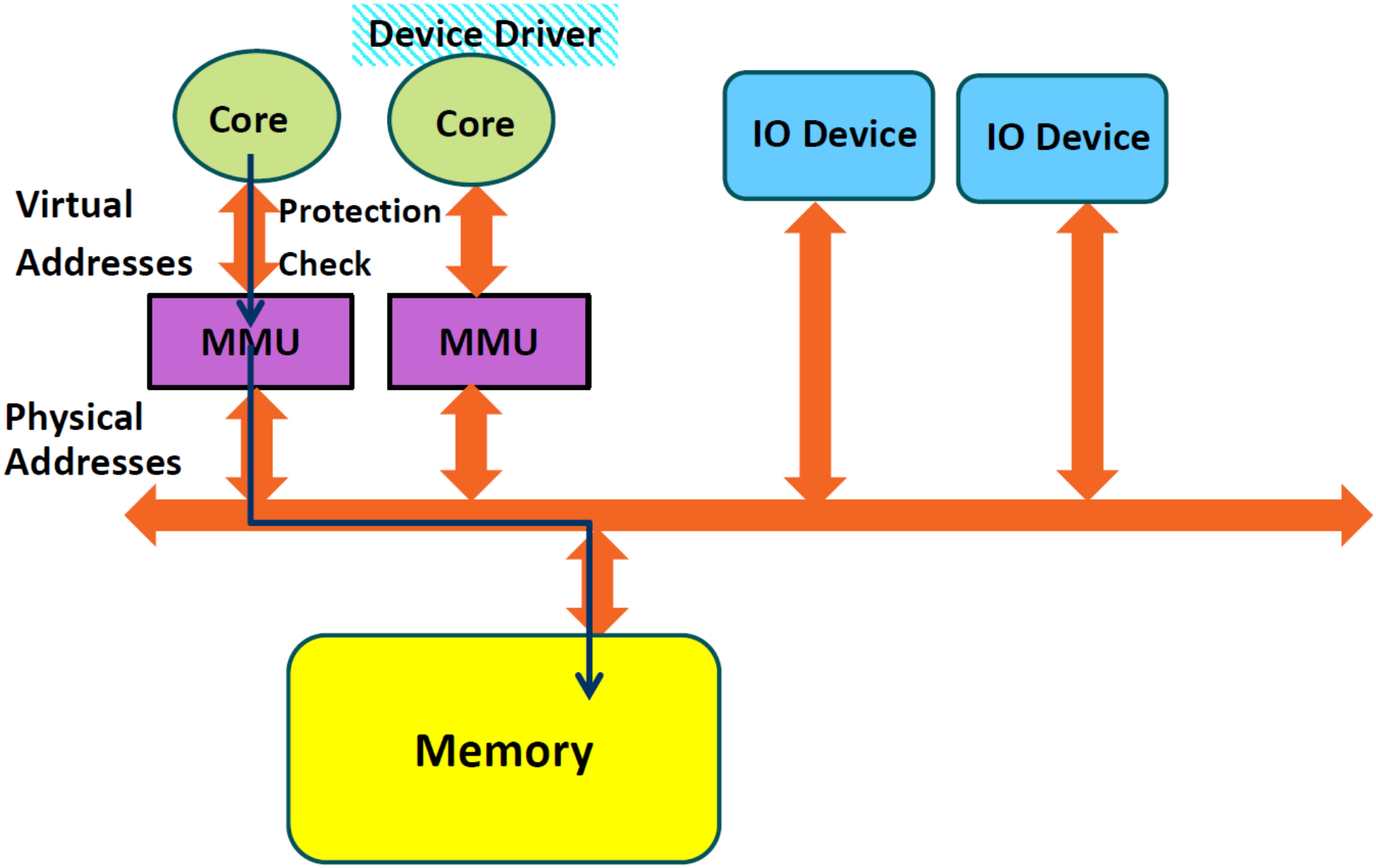
NO SYSTEM VIRTUALIZATION

AMD



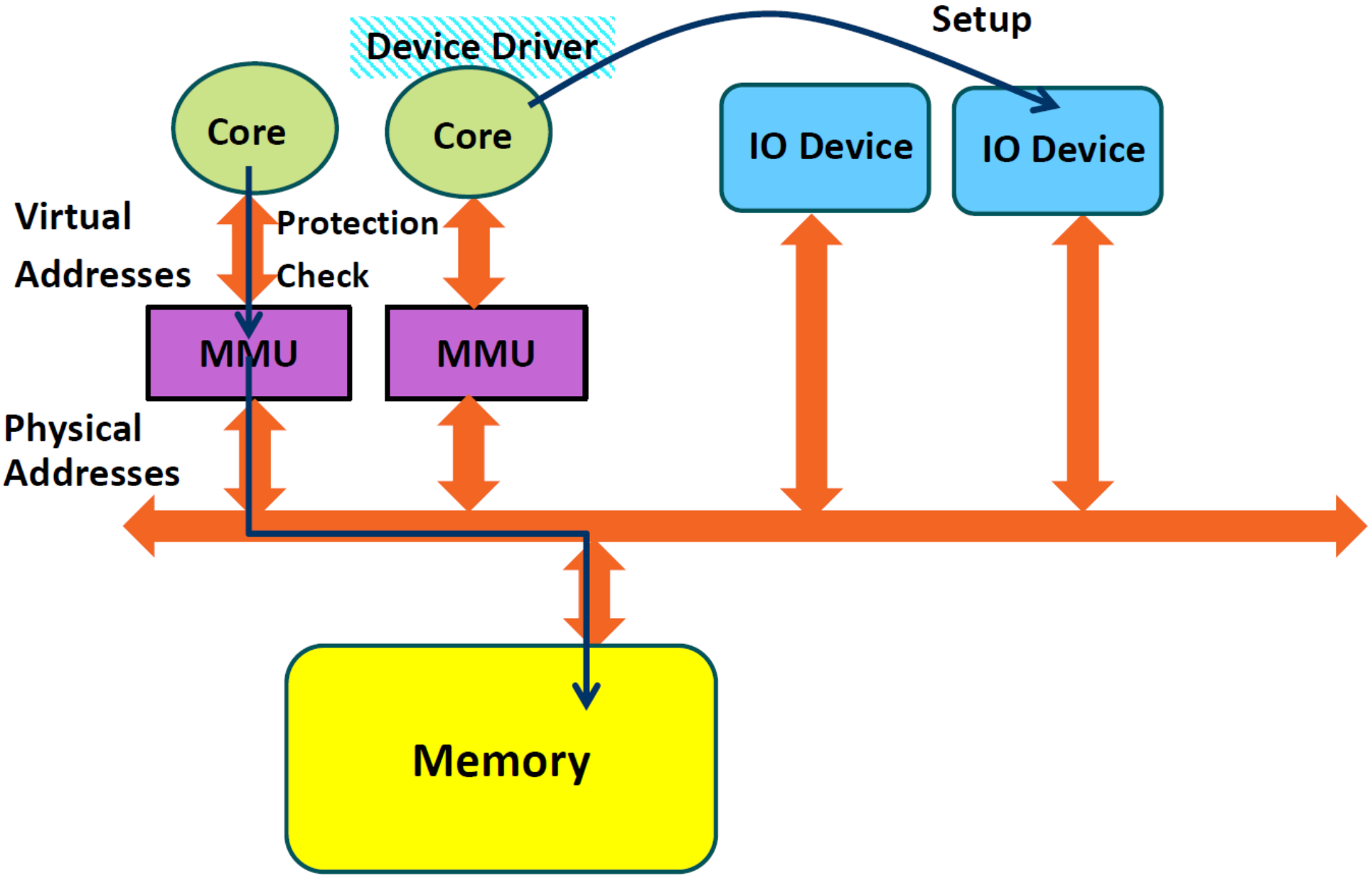
MOTIVATION: TRADITIONAL DMA BY IO

NO SYSTEM VIRTUALIZATION



MOTIVATION: TRADITIONAL DMA BY IO

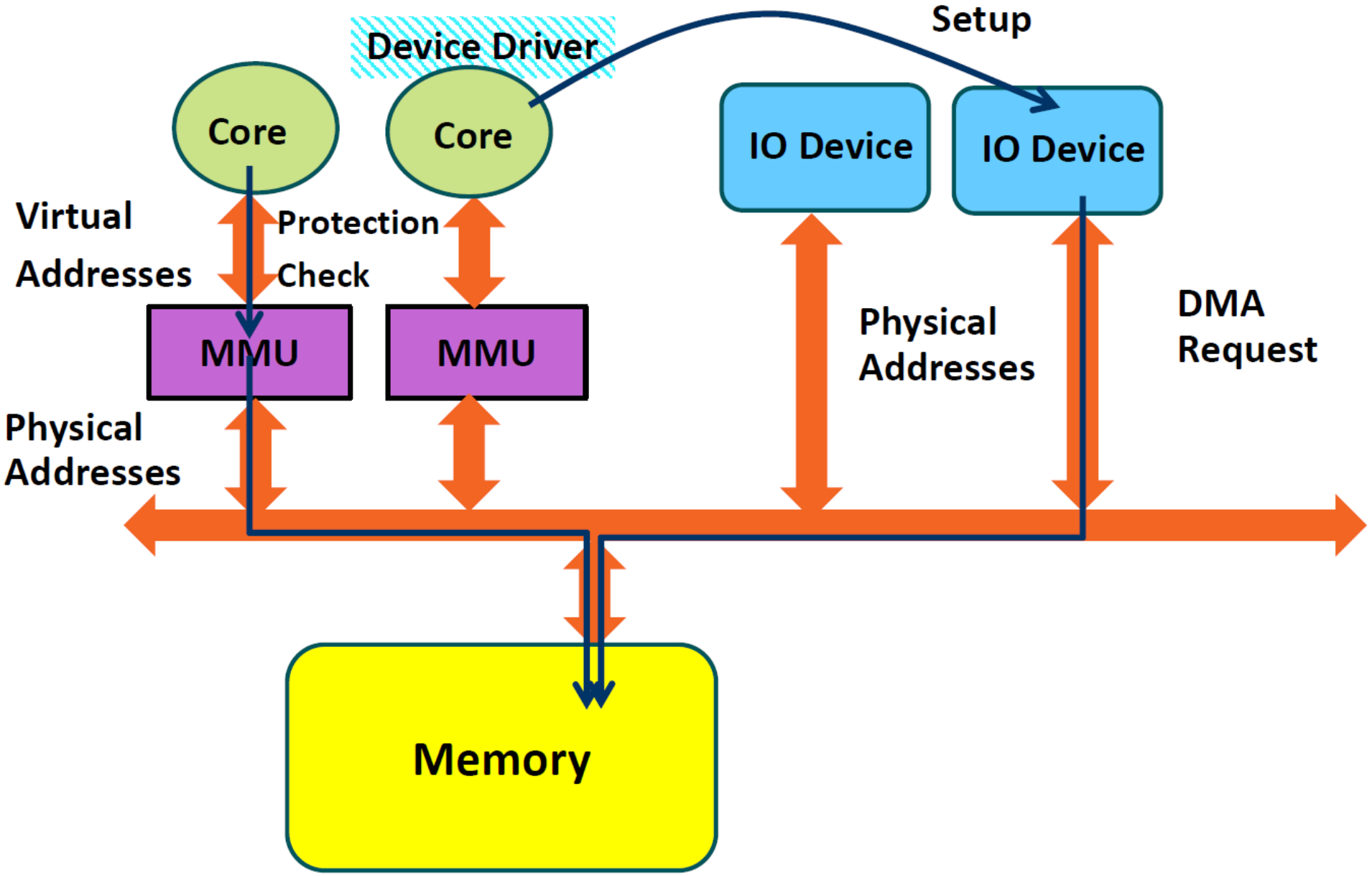
NO SYSTEM VIRTUALIZATION



MOTIVATION: TRADITIONAL DMA BY IO

NO SYSTEM VIRTUALIZATION

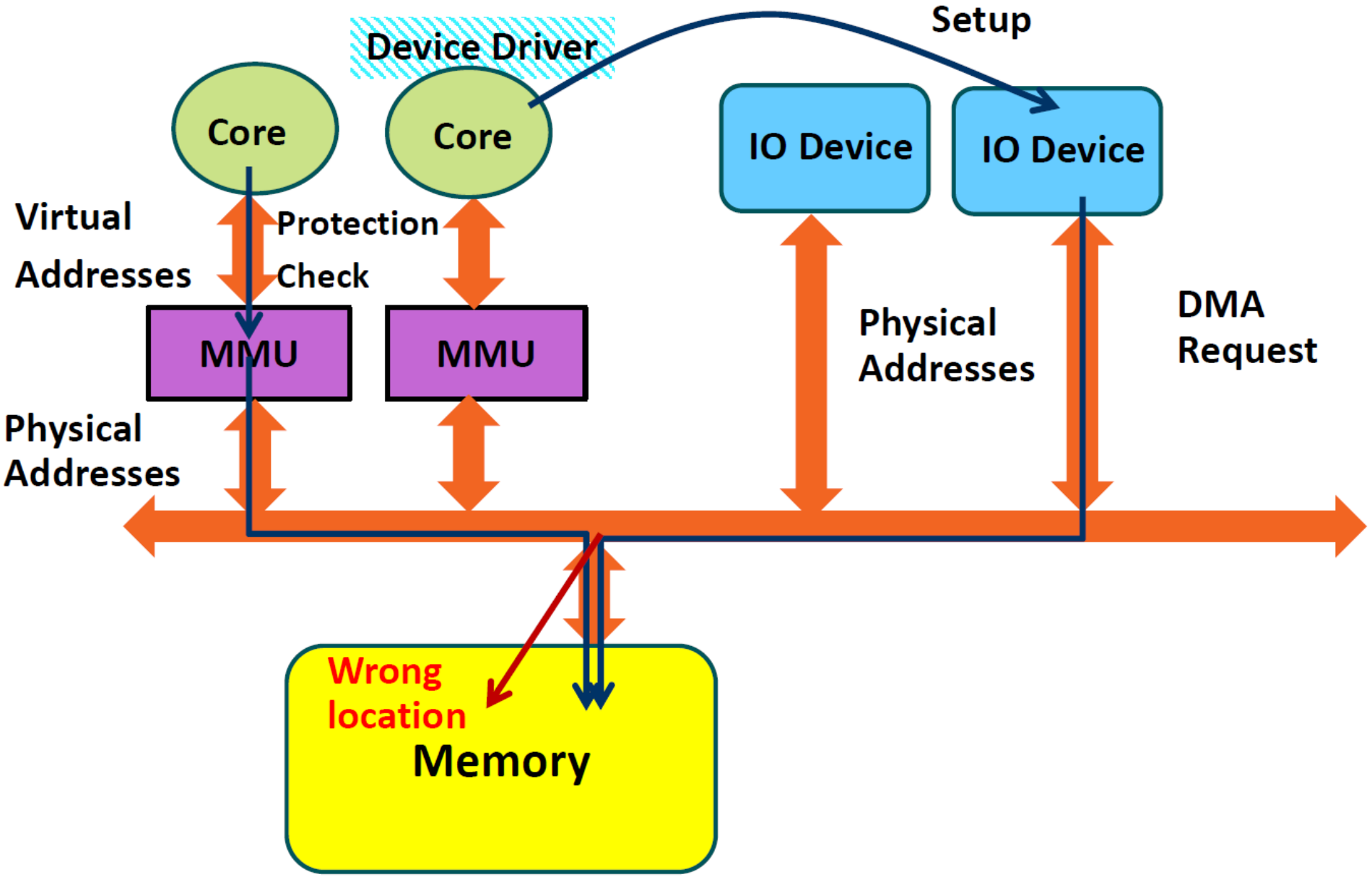
AMD



MOTIVATION: TRADITIONAL DMA BY IO

NO SYSTEM VIRTUALIZATION

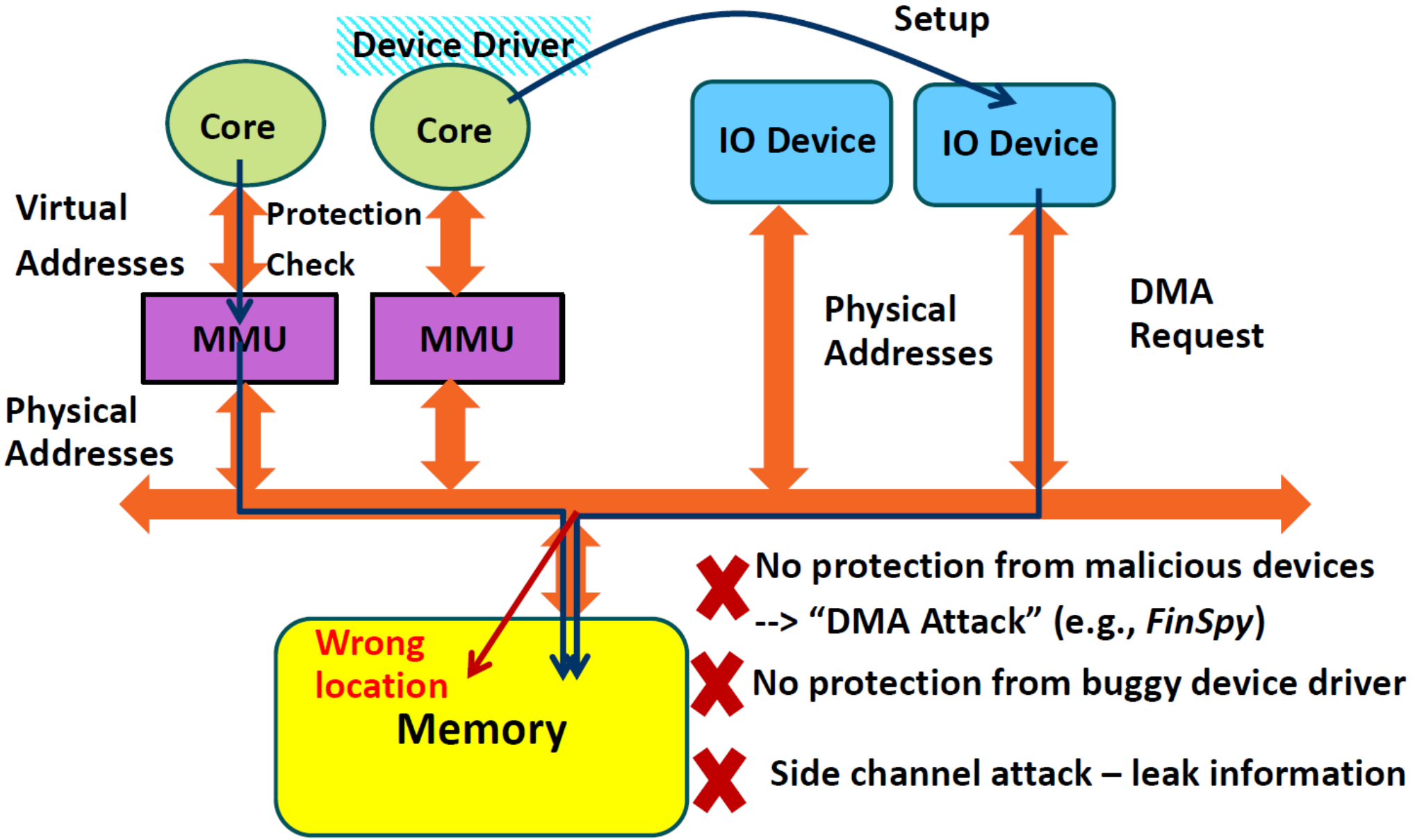
AMD



MOTIVATION: TRADITIONAL DMA BY IO

NO SYSTEM VIRTUALIZATION

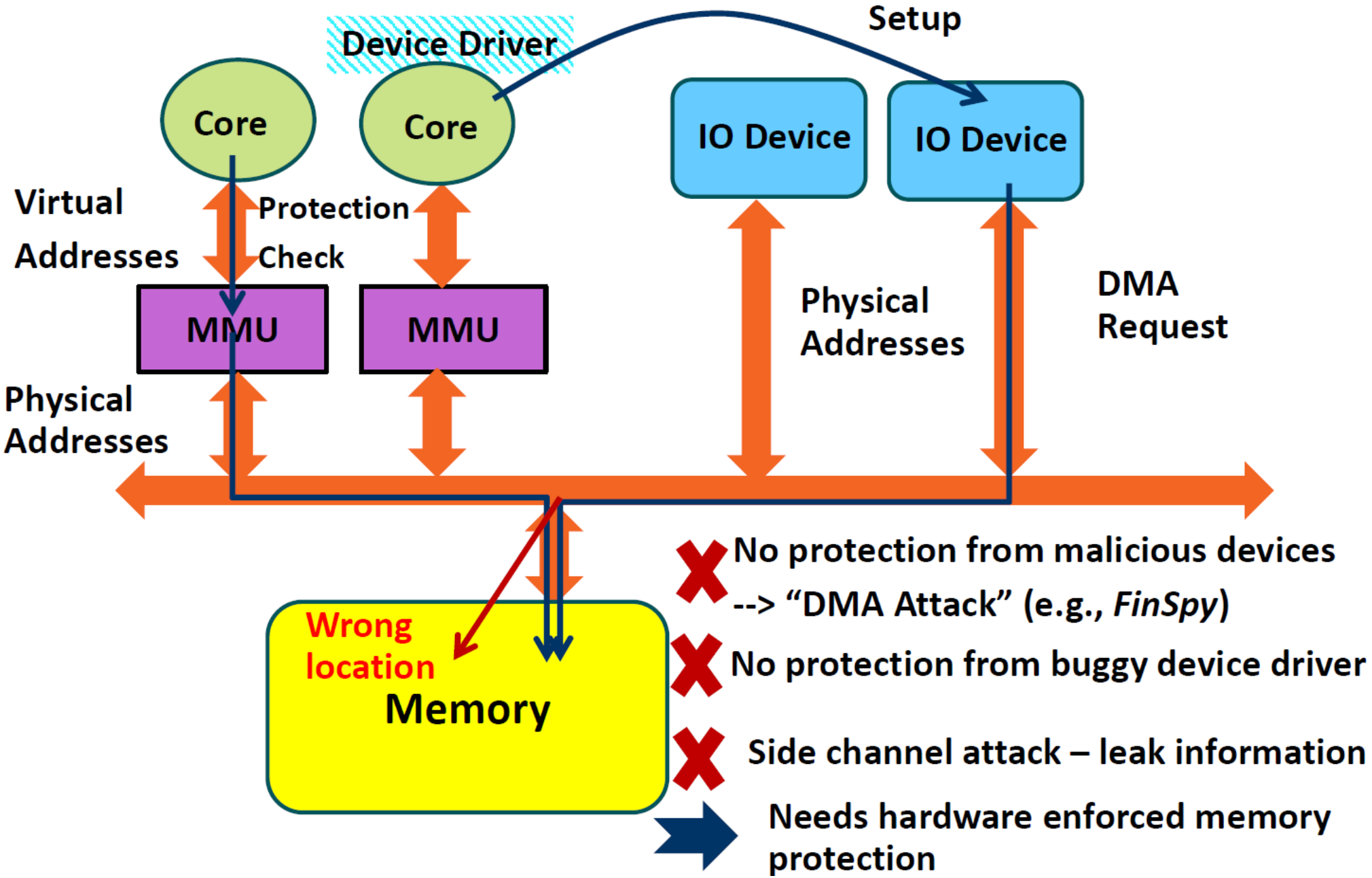
AMD



MOTIVATION: TRADITIONAL DMA BY IO

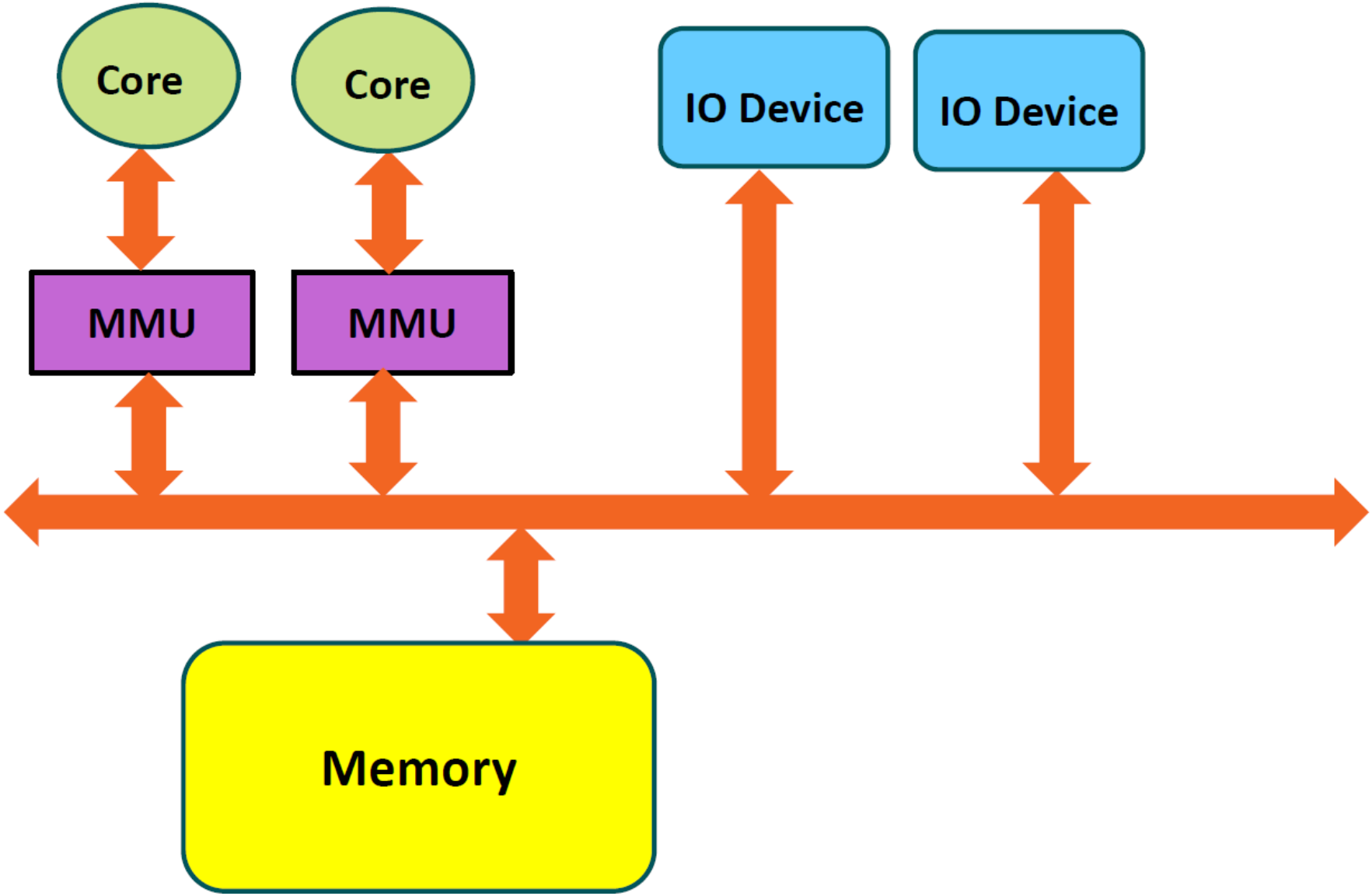
NO SYSTEM VIRTUALIZATION

AMD



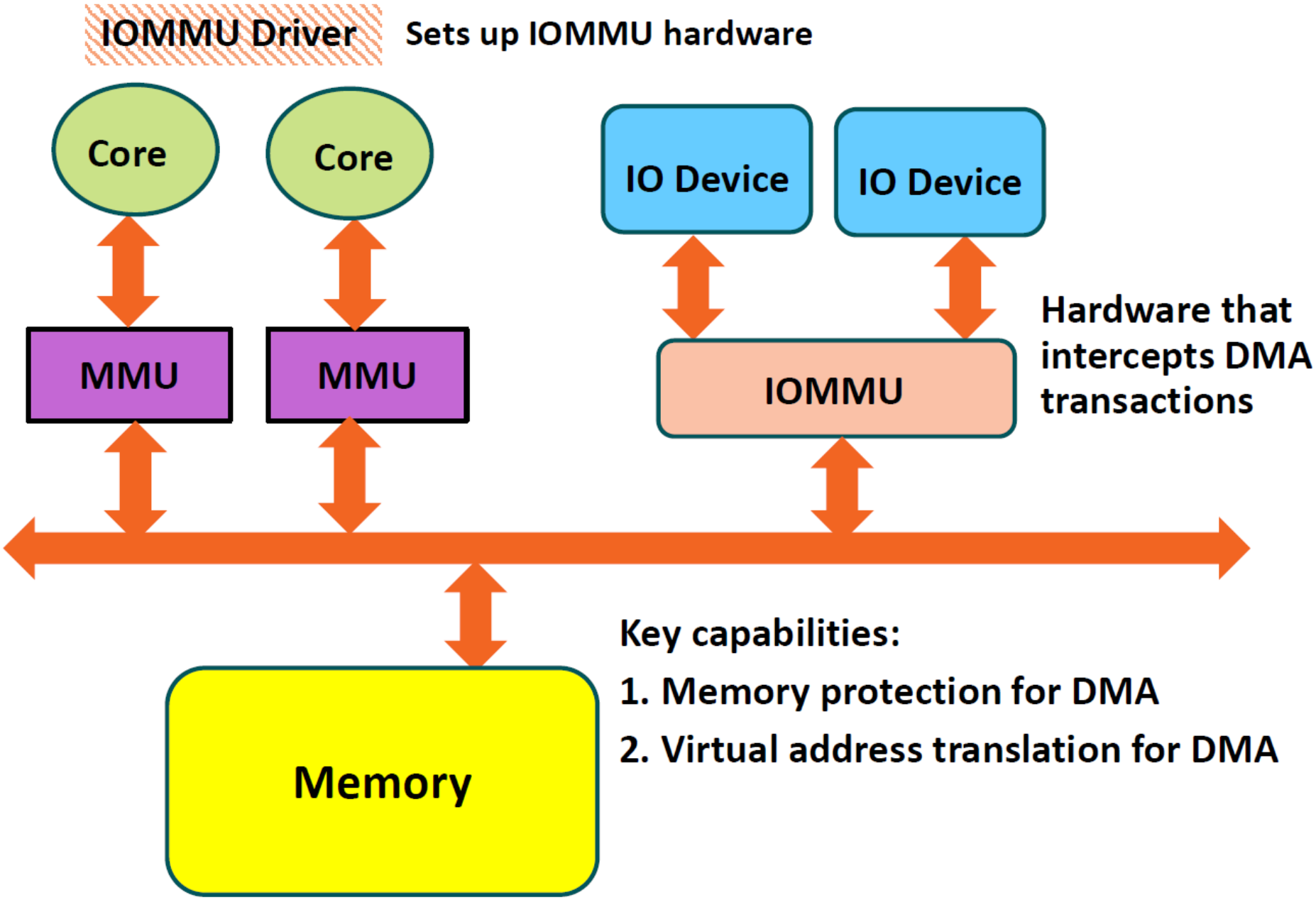
INTRODUCTION OF IOMMU: THE LOGICAL VIEW

AMD



INTRODUCTION OF IOMMU: THE LOGICAL VIEW

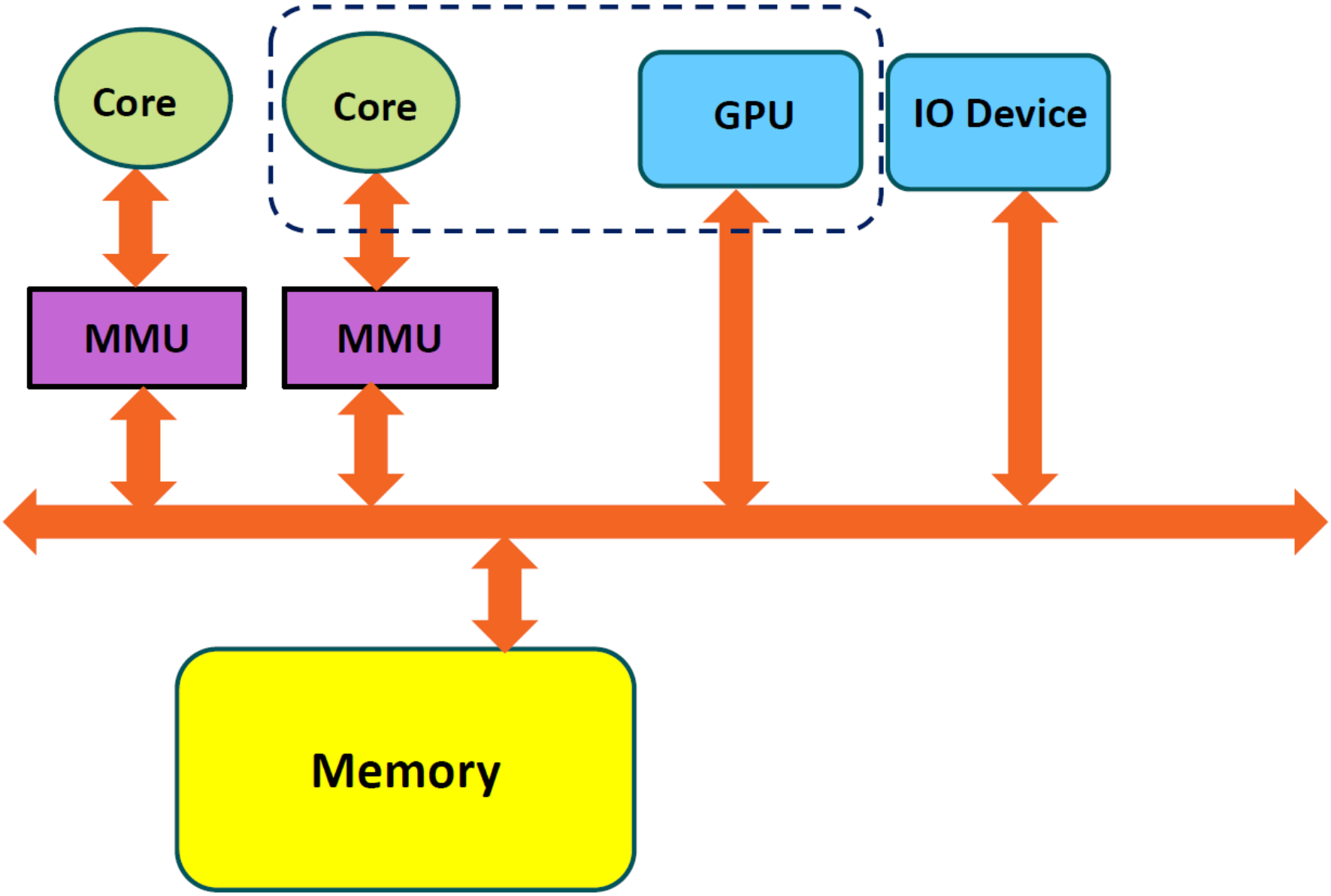
AMD



MOTIVATION: EMERGENCE OF HETEROGENEOUS SYSTEMS

HETEROGENEOUS SYSTEM ARCHITECTURE (HSA)

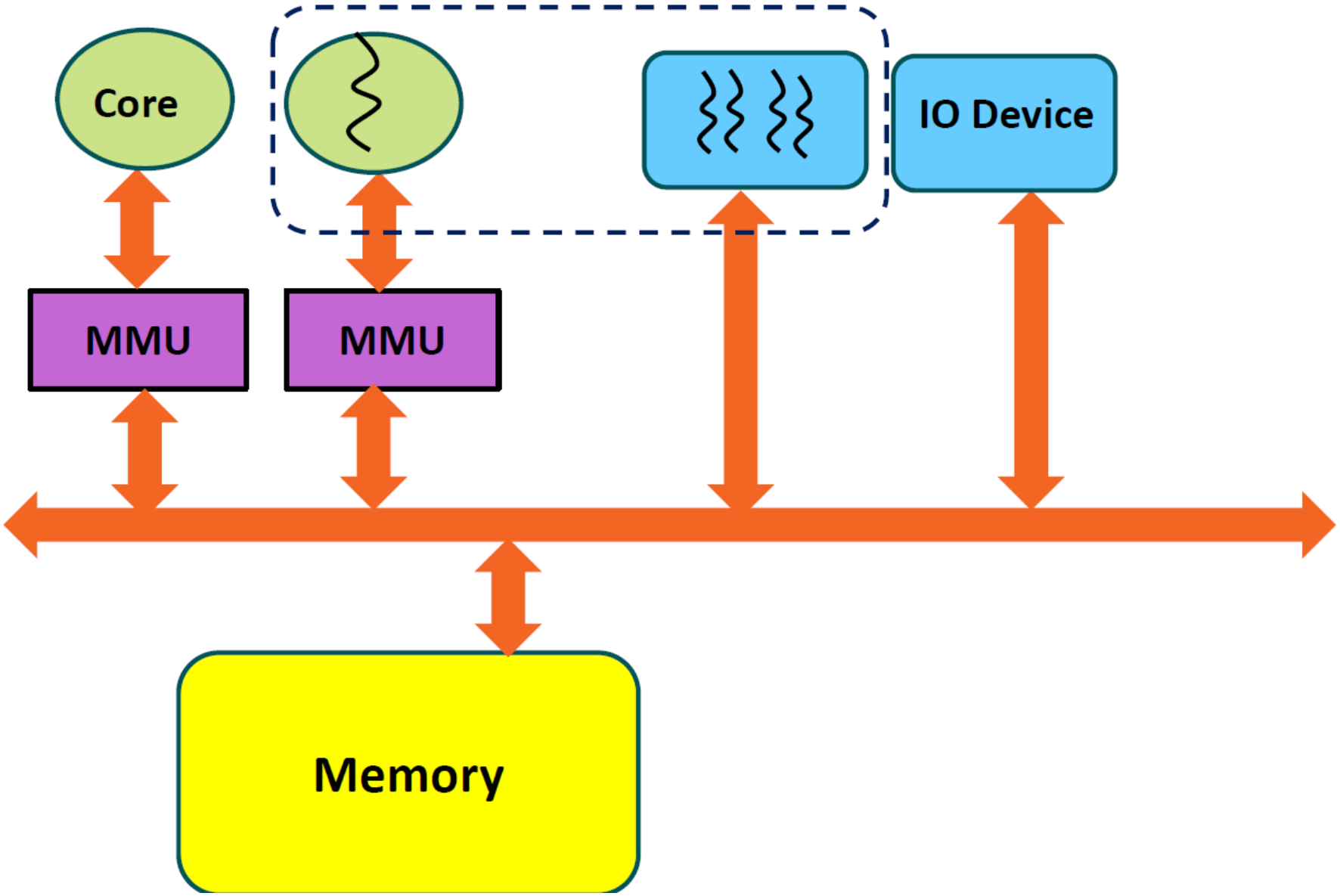
Shared virtual addressing is key to ease of programming



MOTIVATION: EMERGENCE OF HETEROGENEOUS SYSTEMS

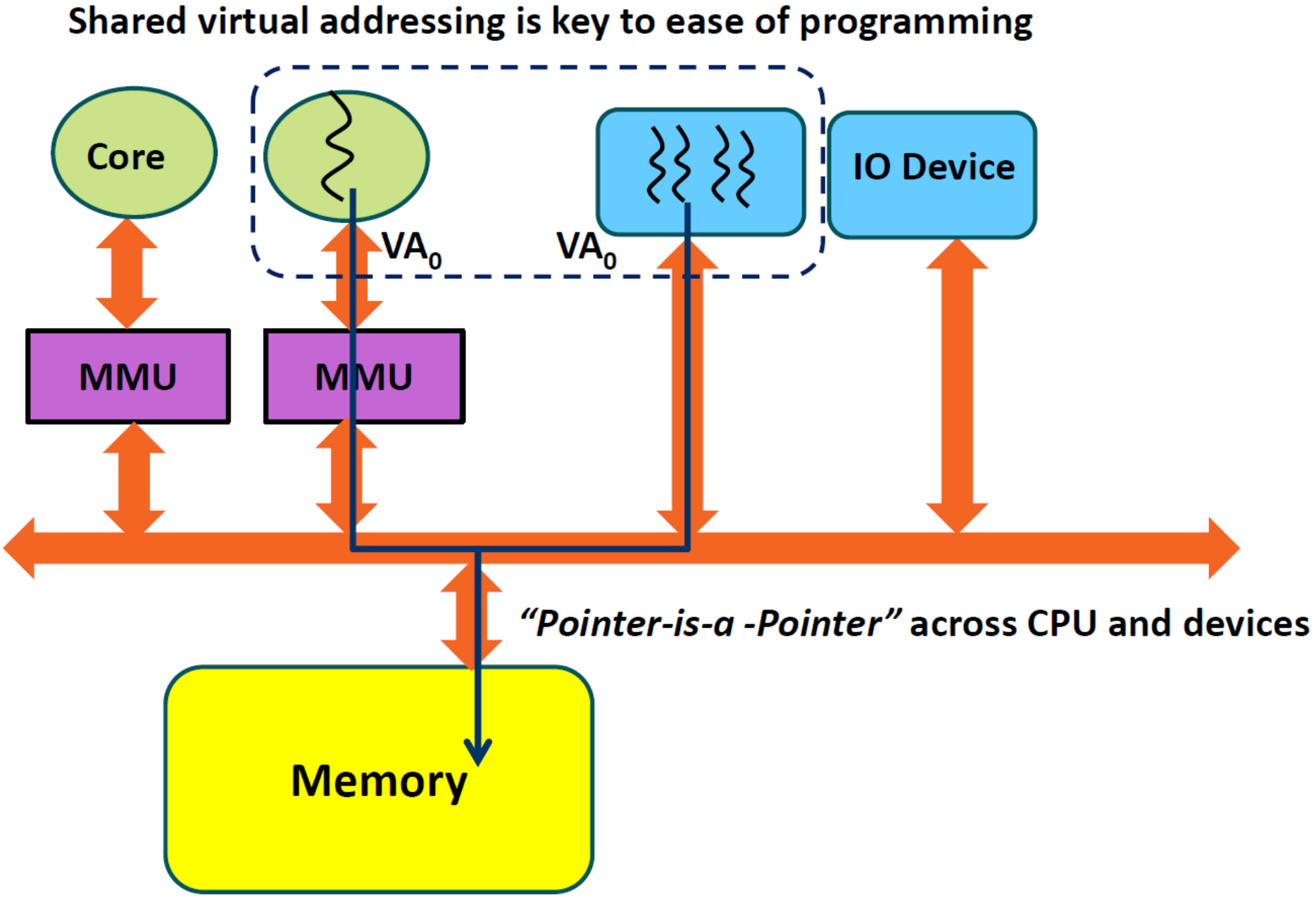
HETEROGENEOUS SYSTEM ARCHITECTURE (HSA)

Shared virtual addressing is key to ease of programming



MOTIVATION: EMERGENCE OF HETEROGENEOUS SYSTEMS

HETEROGENEOUS SYSTEM ARCHITECTURE (HSA)

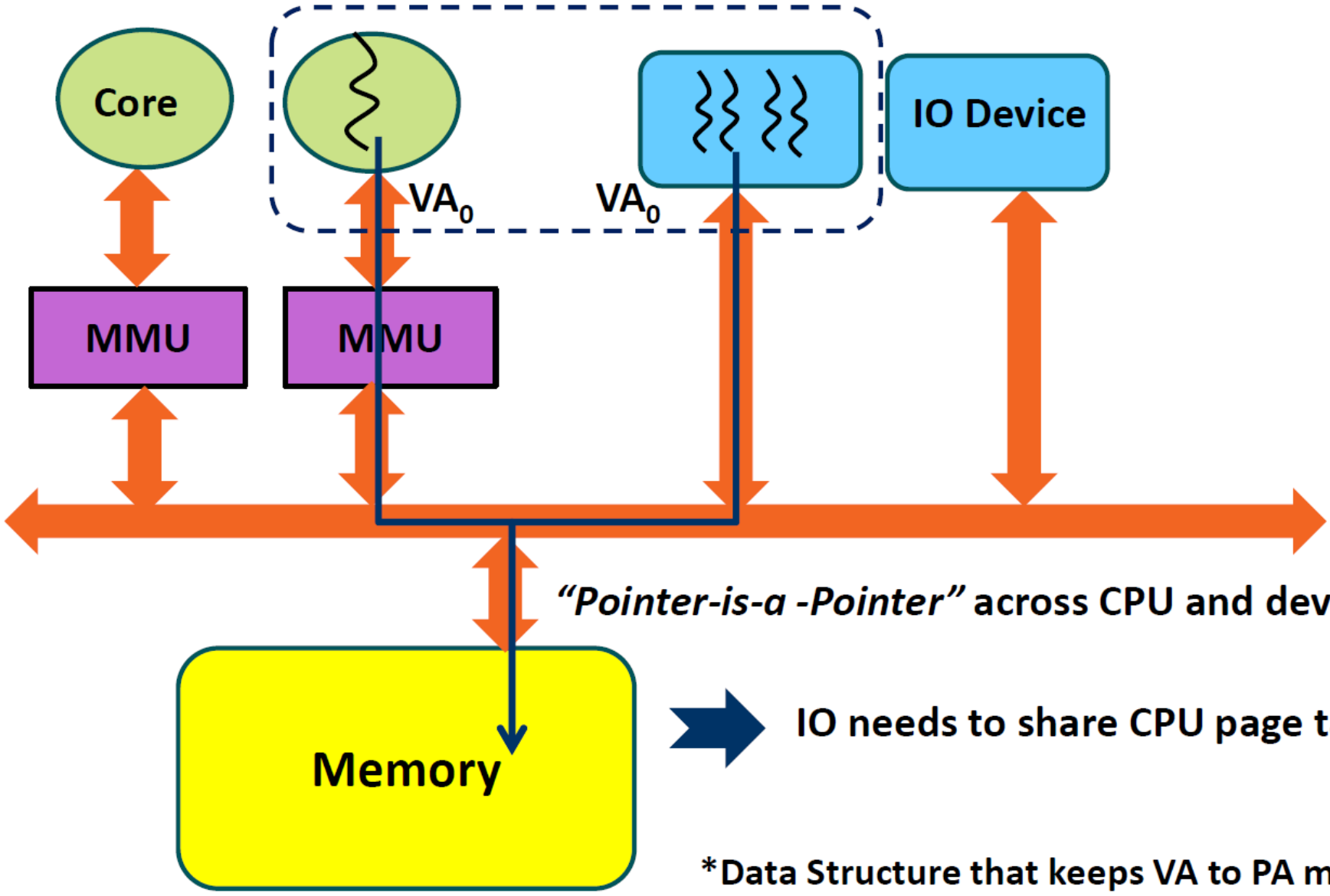


MOTIVATION: EMERGENCE OF HETEROGENEOUS SYSTEMS

AMD

HETEROGENEOUS SYSTEM ARCHITECTURE (HSA)

Shared virtual addressing is key to ease of programming



"Pointer-is-a -Pointer" **across CPU and devices**

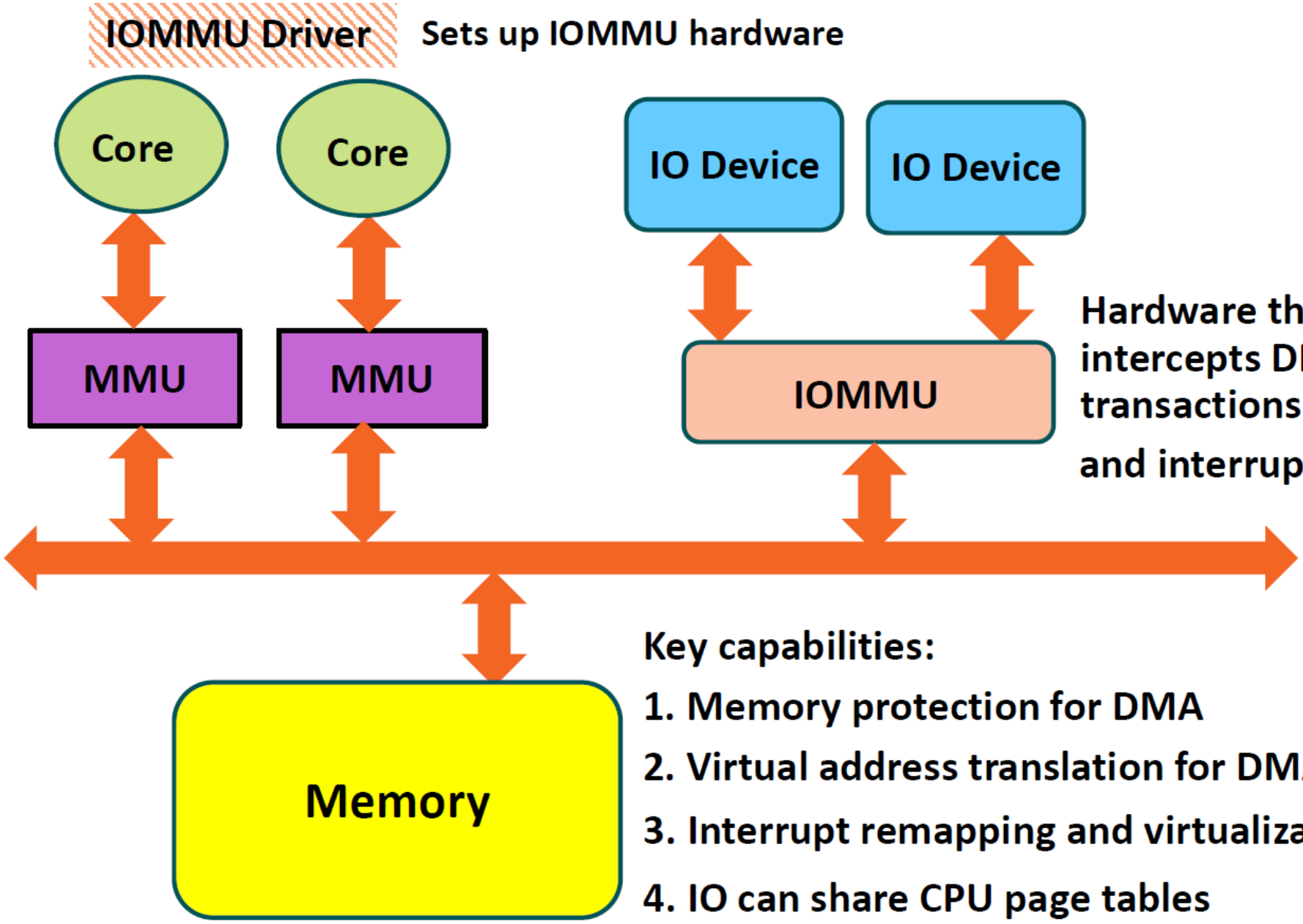
IO needs to share CPU page table*

***Data Structure that keeps VA to PA mapping**

INTRODUCTION OF IOMMU: THE LOGICAL VIEW

ADDING ABILITY TO SHARE ADDRESS SPACE IN HETEROGENEOUS SYSTEM

AMD



Key capabilities:

1. Memory protection for DMA
2. Virtual address translation for DMA
3. Interrupt remapping and virtualization
4. IO can share CPU page tables