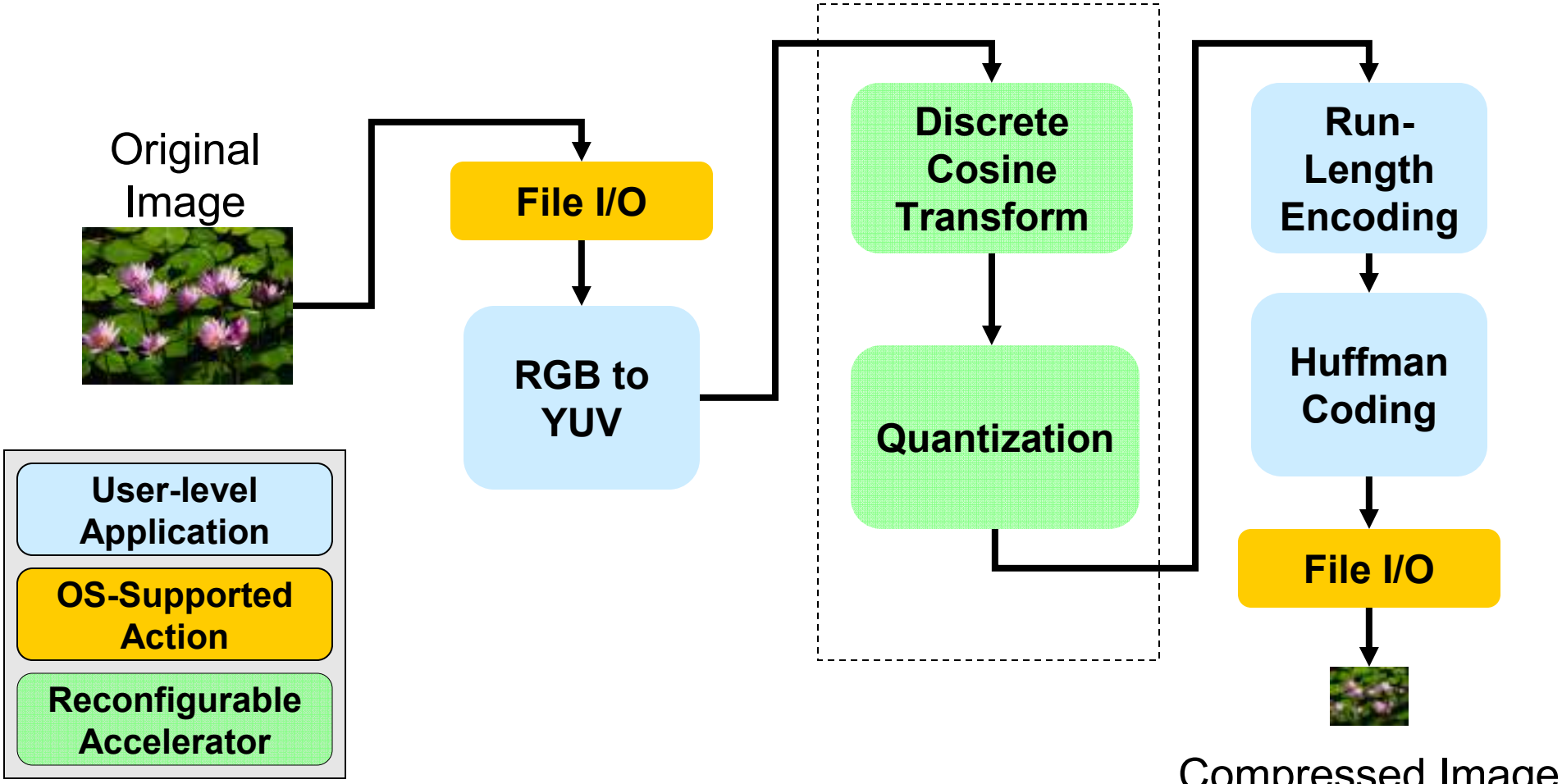
HybridOS: Runtime Support for Reconfigurable Accelerators

John H. Kelm
Steven S. Lumetta

University of Illinois at Urbana-Champaign

Motivating Example:
JPEG Encoding for CPU/Accelerator Platforms



Roadmap

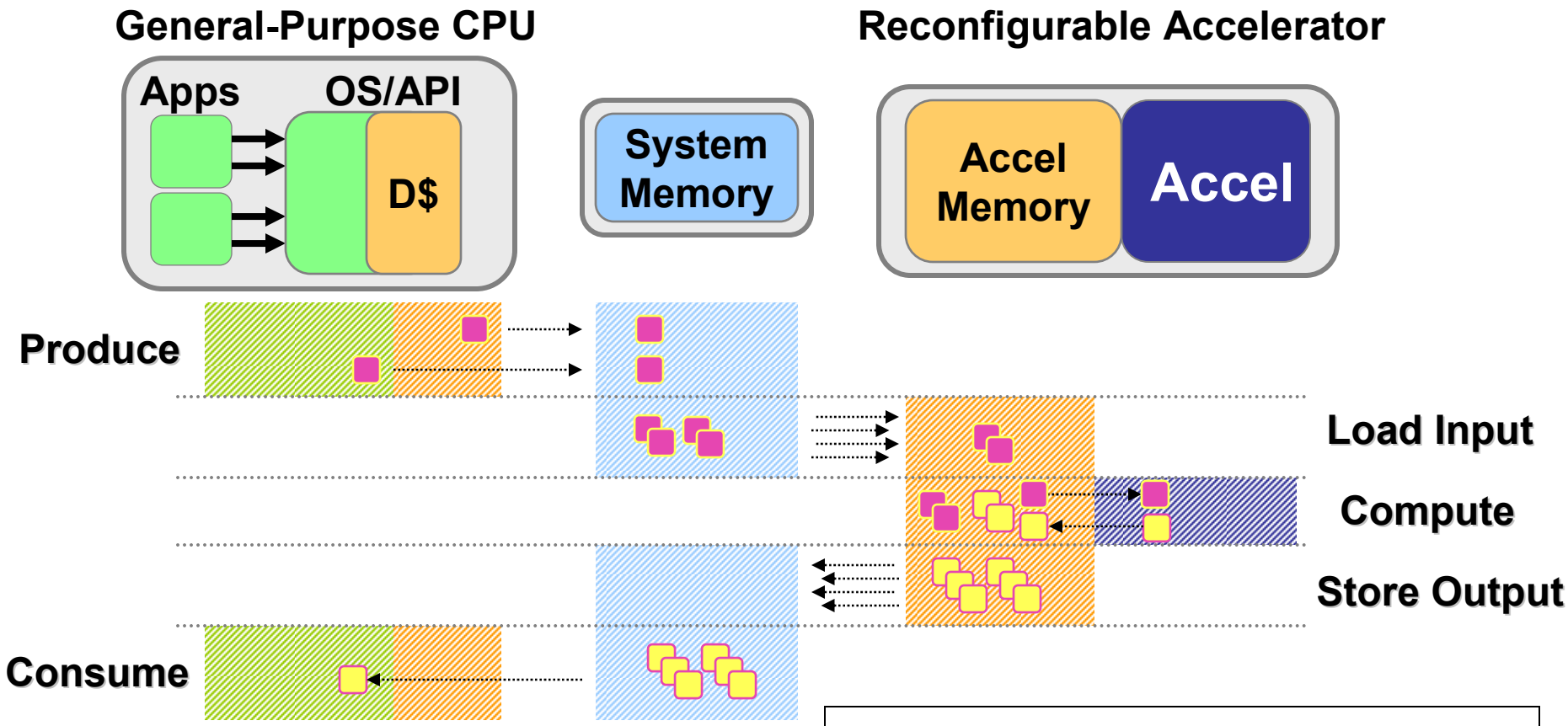
Problem Statement: *How do we add OS support and reduce complexity when integrating accelerators with a general-purpose CPU?*

- Motivation & Introduction
- HW/SW Interfaces
- Accelerator Access Methods
- Case Study Results
- Summary & Conclusions

Introduction

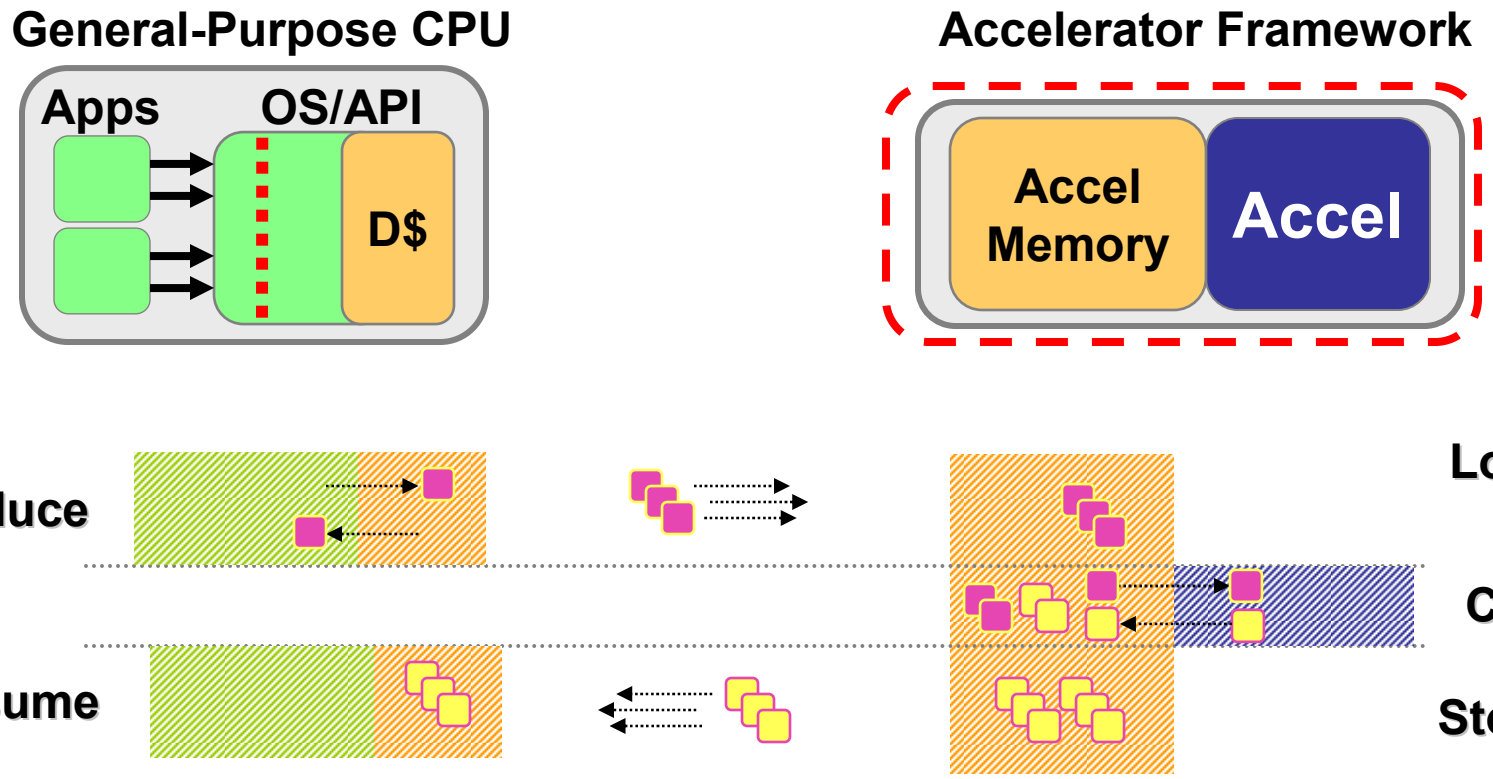
- ***HybridOS* Game Plan:**
  - **What**: Decouple HW design from SW design
  - **How**: Define interfaces; provide infrastructure
- **Design Considerations:**
  - **HW/SW interfaces**
  - **Data movement**
  - Accelerator interconnections
  - Security and protection
  - Debugging

Simple Coprocessor Model



- Data Movement Serialization
- Application-specific Interface

HybridOS Prototype Platform



- Consistent SW Interfaces
- Protection boundaries
- Embedded memory/CPU caches

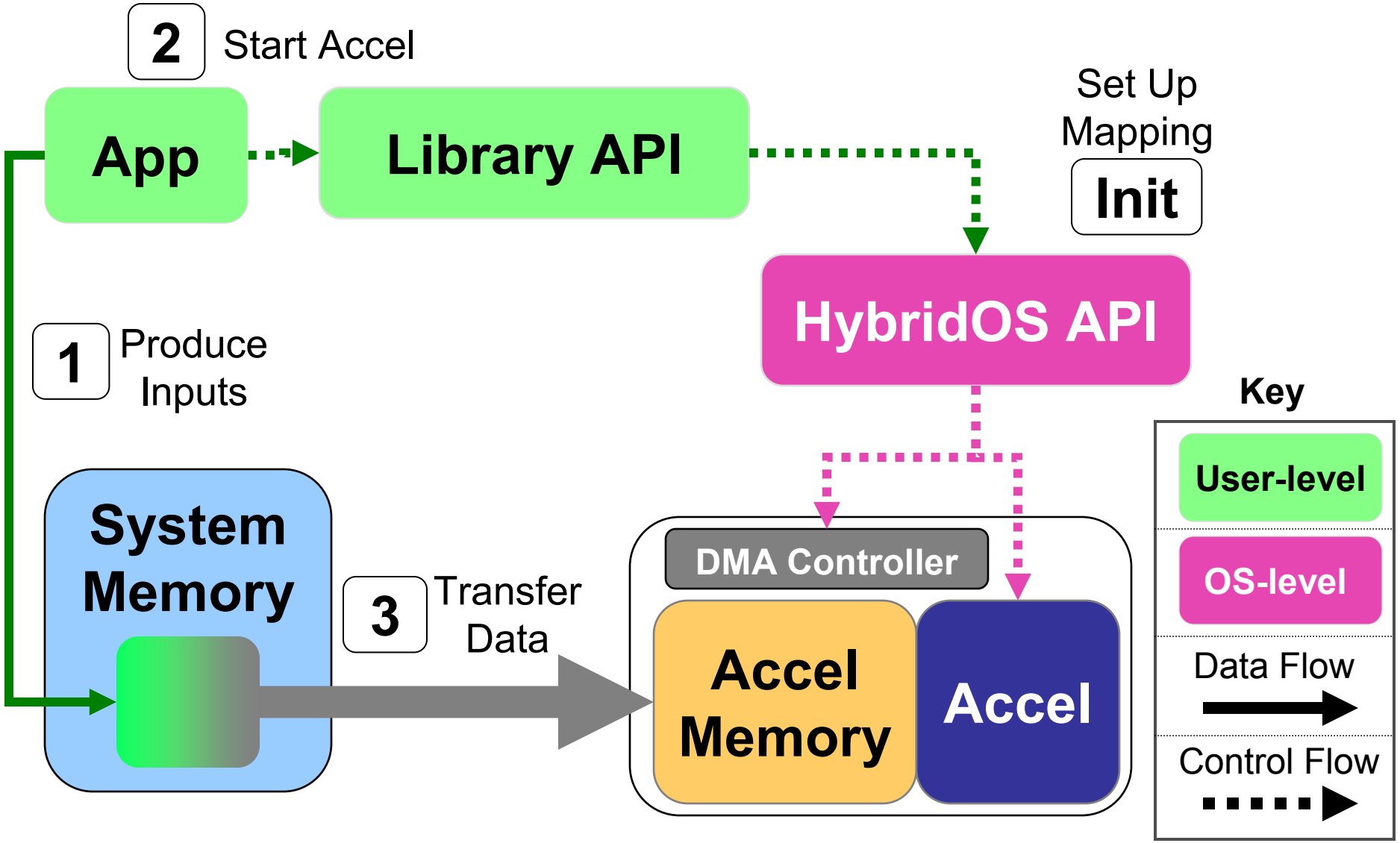
Platform based on Xilinx V2Pro FPGA/XUP Board

Accelerator Access Methods

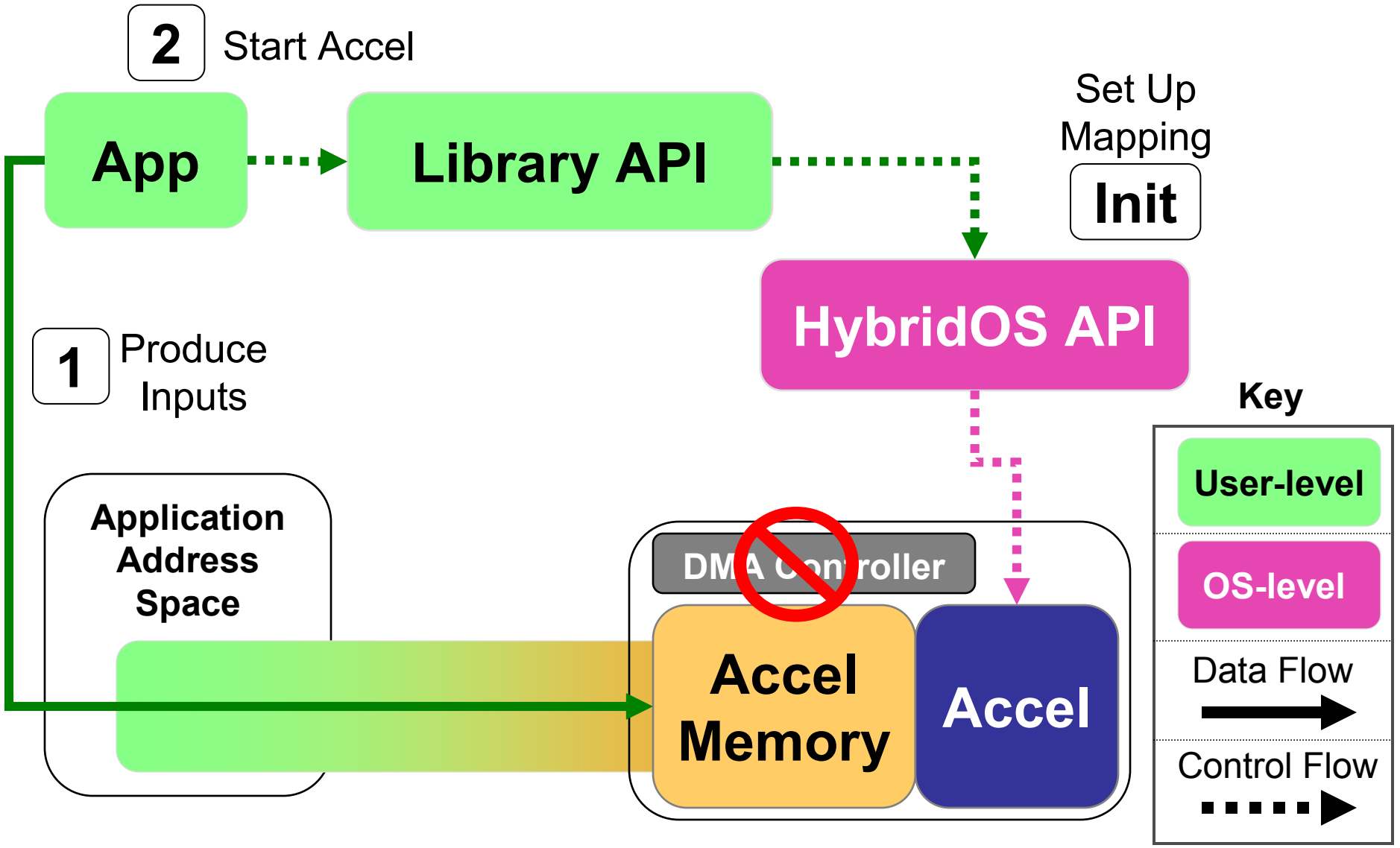
- ***HybridOS*** Access Methods**:**

  Mechanism used to transfer data and control between application and accelerator(s)
- Consistent SW Interface, transparency
- Four methods evaluated
  - *User Space Buffers*
  - User Mapped DMA Buffers
  - Uncacheable Direct Mapping
  - Cacheable Direct Mapping

User Mapped DMA Buffers



Unached Direct Mapped Buffers

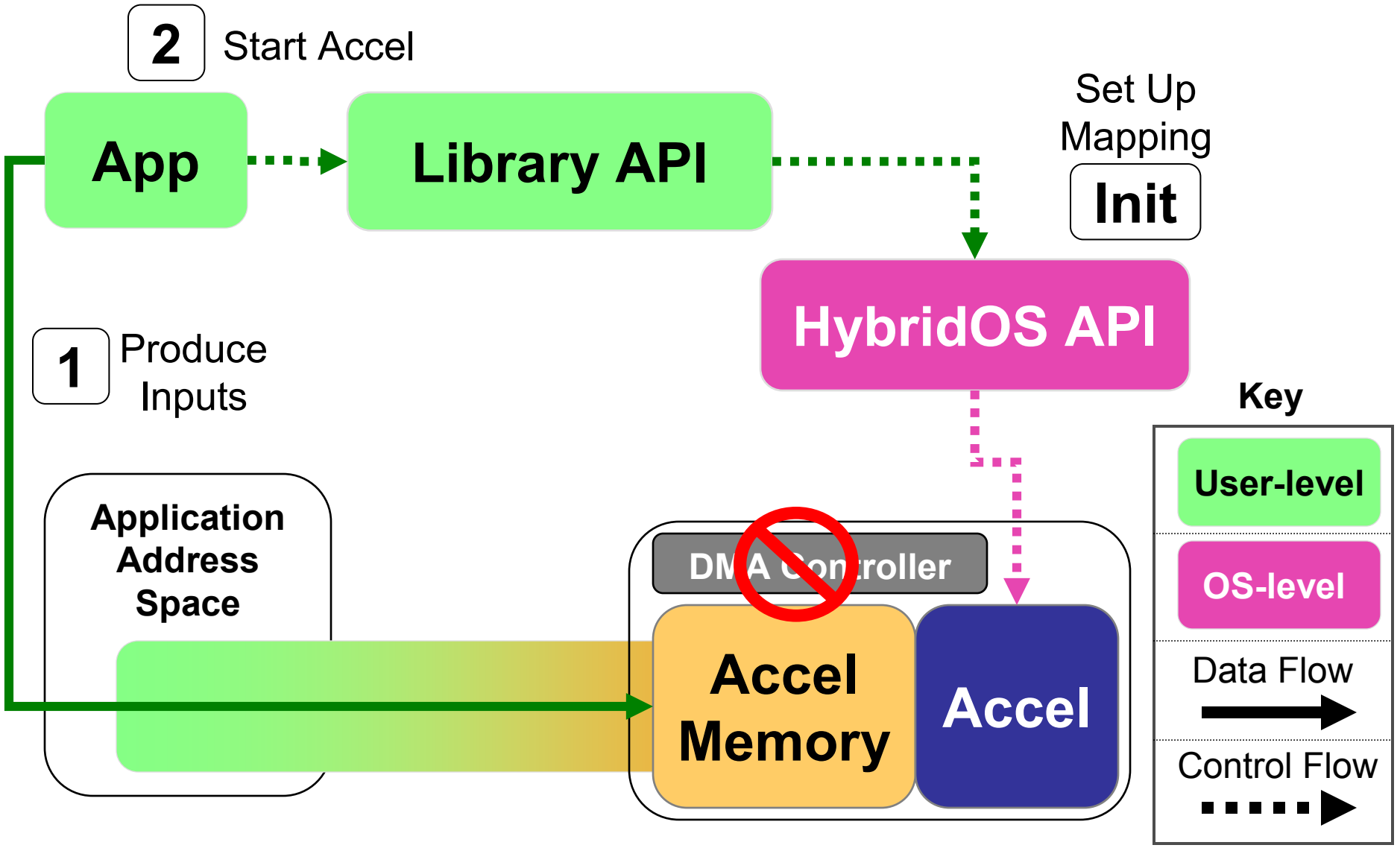


Caching of Buffers

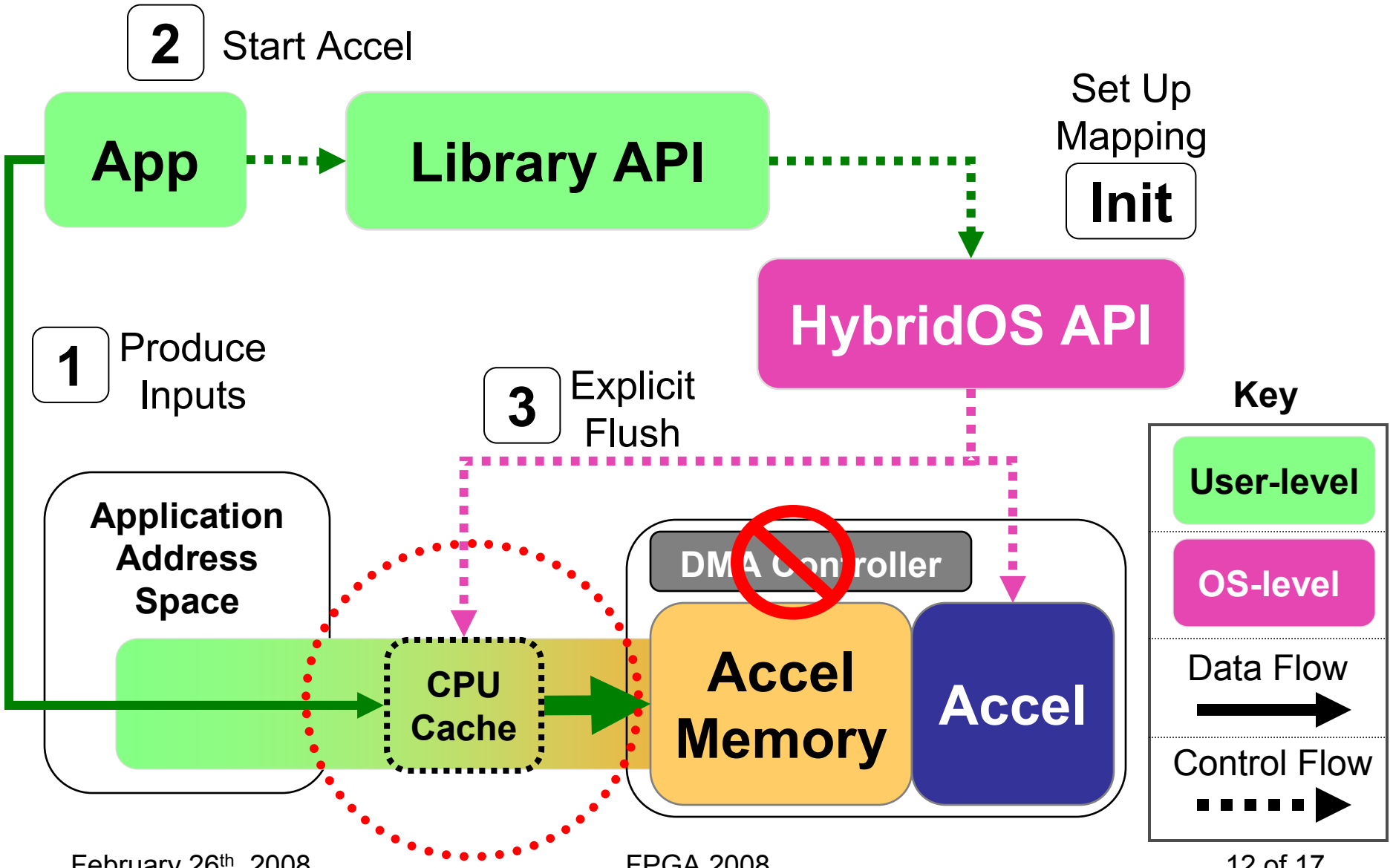
- **Problem**: Direct mapping reduces number of copies, but high per-transfer cost
- **Solution**: Cached buffers

- Advantages of caching:
  - Cache line transfers
  - Reduced latency access for CPU
  - Leverage HW prefetching
  - Read sharing and natural data partitioning

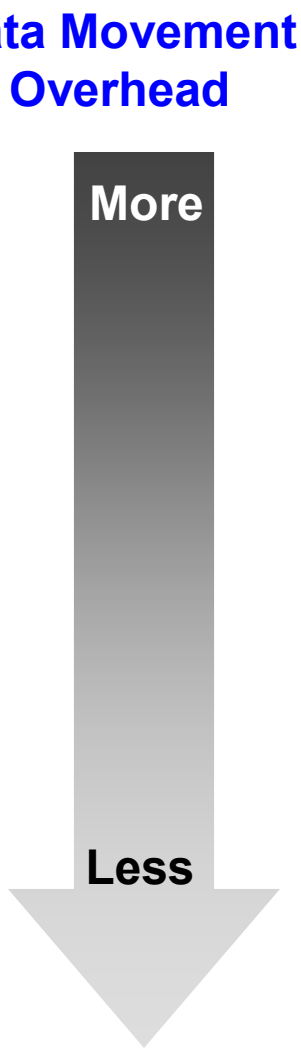
Cached Direct Mapped Buffers



Cached Direct Mapped Buffers



Accelerator Access Methods



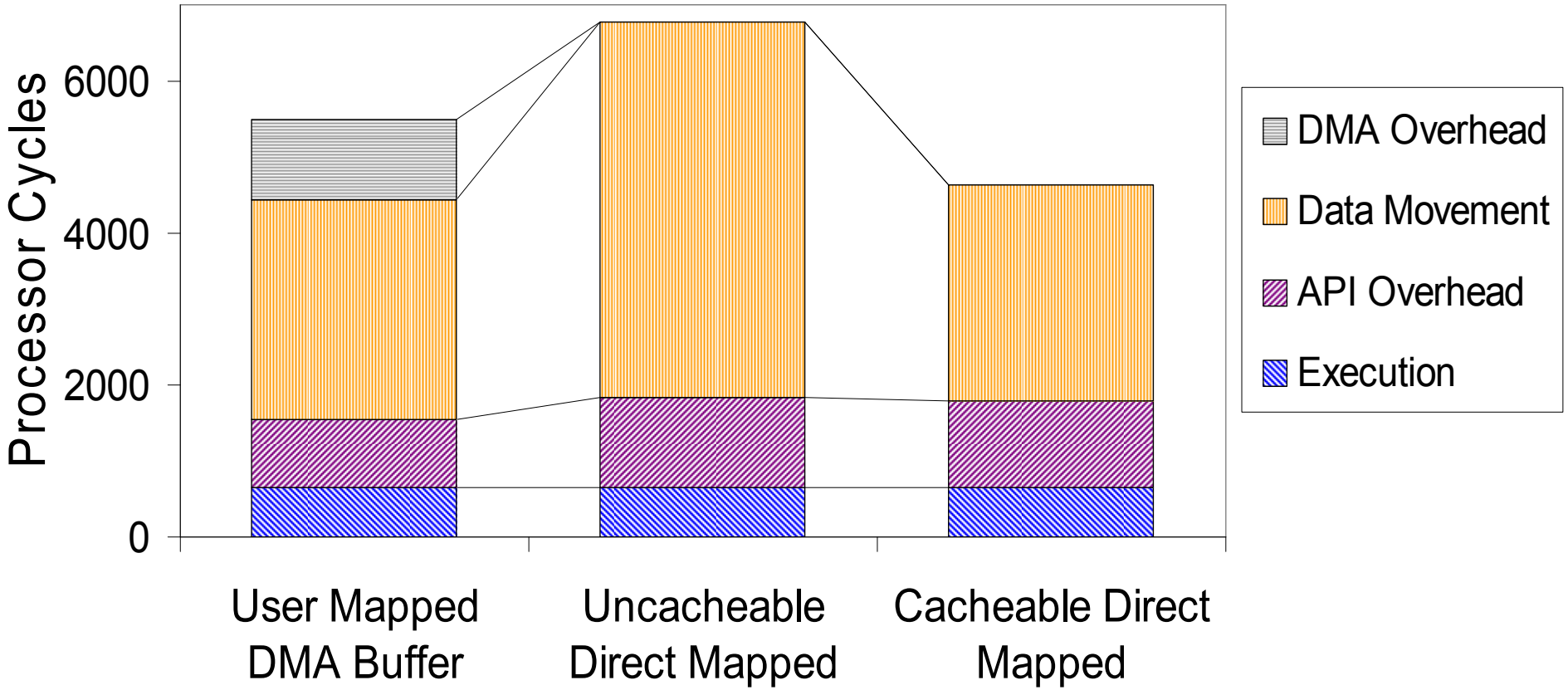
- *User Space Buffers*
- **User Mapped DMA Buffers**
  - + High-throughput transfers
  - – Serialized data transfer, pinned memory
- **Direct Mapping**
  - + No added copies, enables read sharing
  - + Leverage CPU cache, HW prefetching
  - – One-to-one mapping

Accelerator Access Methods

Well-defined interfaces and **OS support** enables **transparent** remapping between access methods.

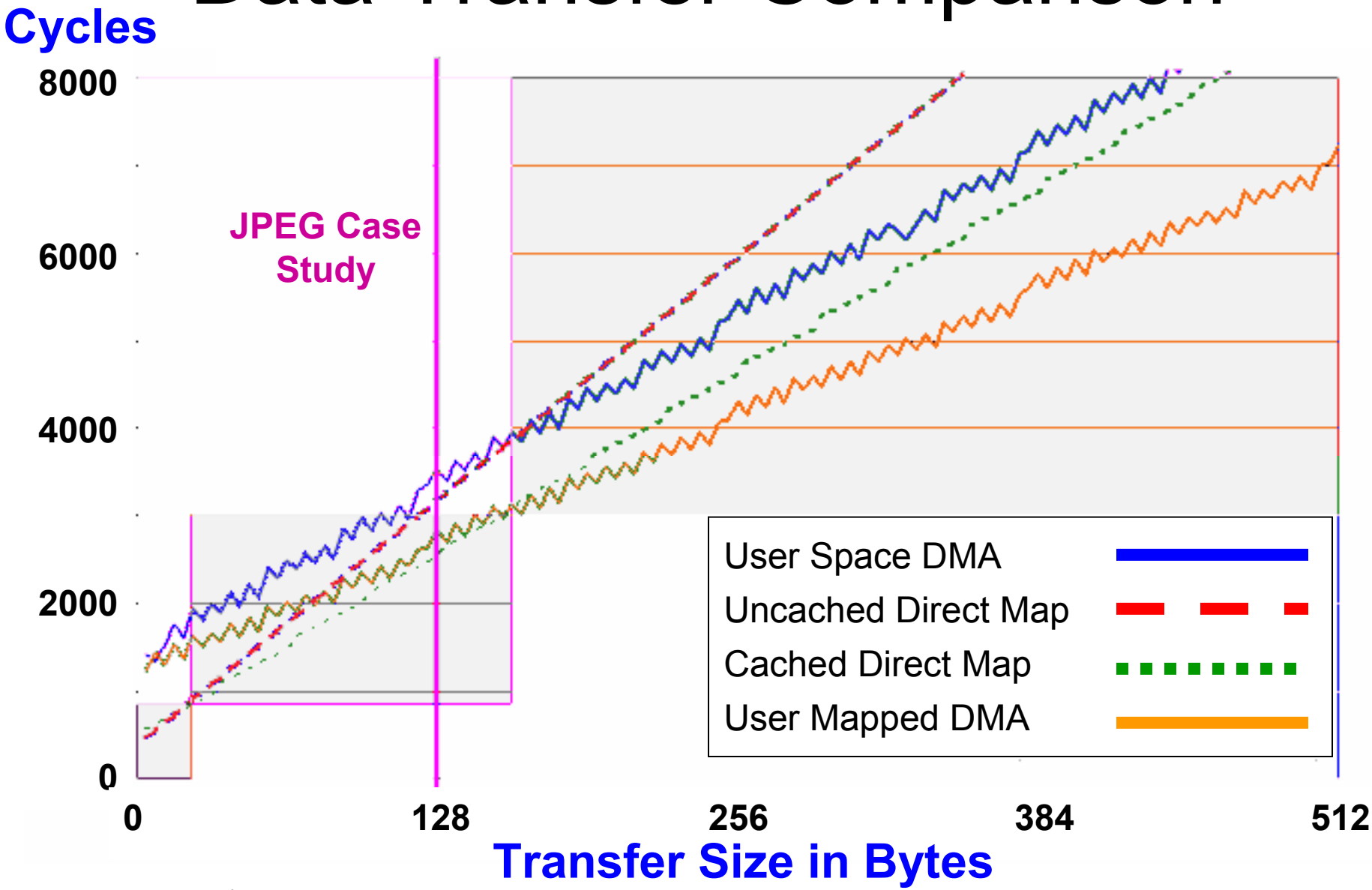
Access Methods Comparison

JPEG encoder **DCT** and **Quantize** Accelerators



Roundtrip cost for processing 128-byte block

Data Transfer Comparison



Summary & Conclusion

- **Summary**
  - CPU/Accelerator platform w/OS Support
  - Framework+API for efficient and consistent accelerator integration with CPU
  - Data access methods/case study evaluation
- **Conclusions**: Reuse, flexible interfaces, reduced overheads, and transparency

Prototype platform available for download:
http://www.HybridOS.crhc.uiuc.edu/