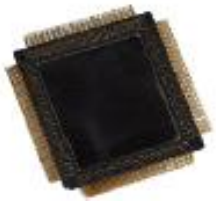
SoC and DSP Architecture

Shao-Yi Chien

SoC, System-on-a-Chip



ASIC

Yesterday

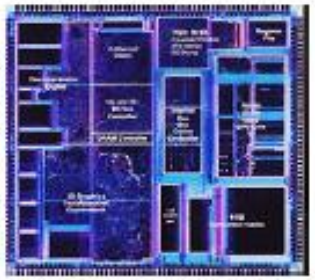
Assembly

System-Board

IP

Today

Integration



SOC

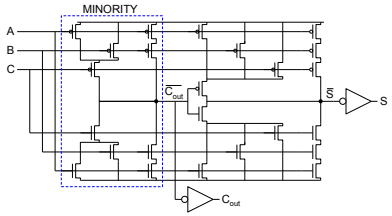
- **Silicon IP** as the components
- IPs may come in several forms: **Hard, Soft, Firm**

Hard IP : Layout / Technology dependent
Soft IP : RTL
Firm IP : Net List : still Technology dependent.

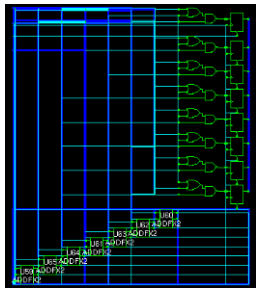
SoC, System-on-a-Chip

Yesterday, ASIC

- HW only
- Perfect interconnection

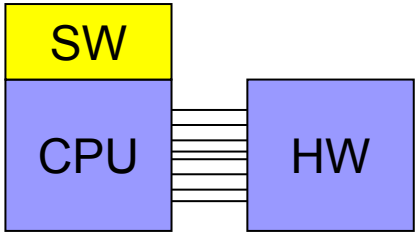



```
module counter(
clk,                    //Clock signal.
reset_n,        //Async. reset signal, active low.
enable,                 //Sync. enale signal. The counter won't act without enable==1'b1.
data                    //output counter data.
);

input clk;              //Clock signal.
input reset_n;                  //Async. reset signal, active low.
input enable;           //Sync. enale signal. The counter won't act without enable==1'b1.
output [1:0] data;      //output counter data.

reg [1:0] data, data_w;

.....

always@(posedge clk or negedge reset_n)
begin
  if(!reset_n)
  begin
    data <= #1 2'd0;
  end
  else
  begin
    data <= #1 data_w;
  end
end

endmodule
```

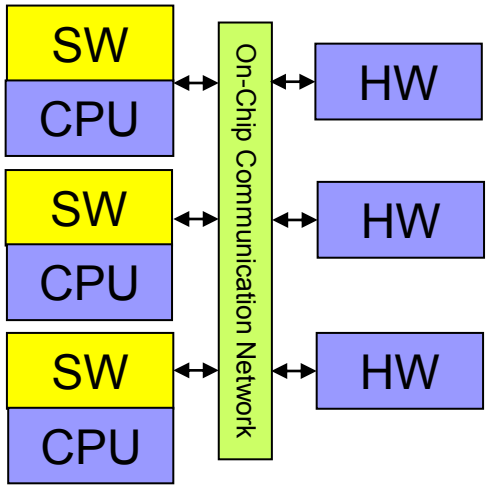


Today, SoC

- Heterogeneous
- CPU + dedicated HW

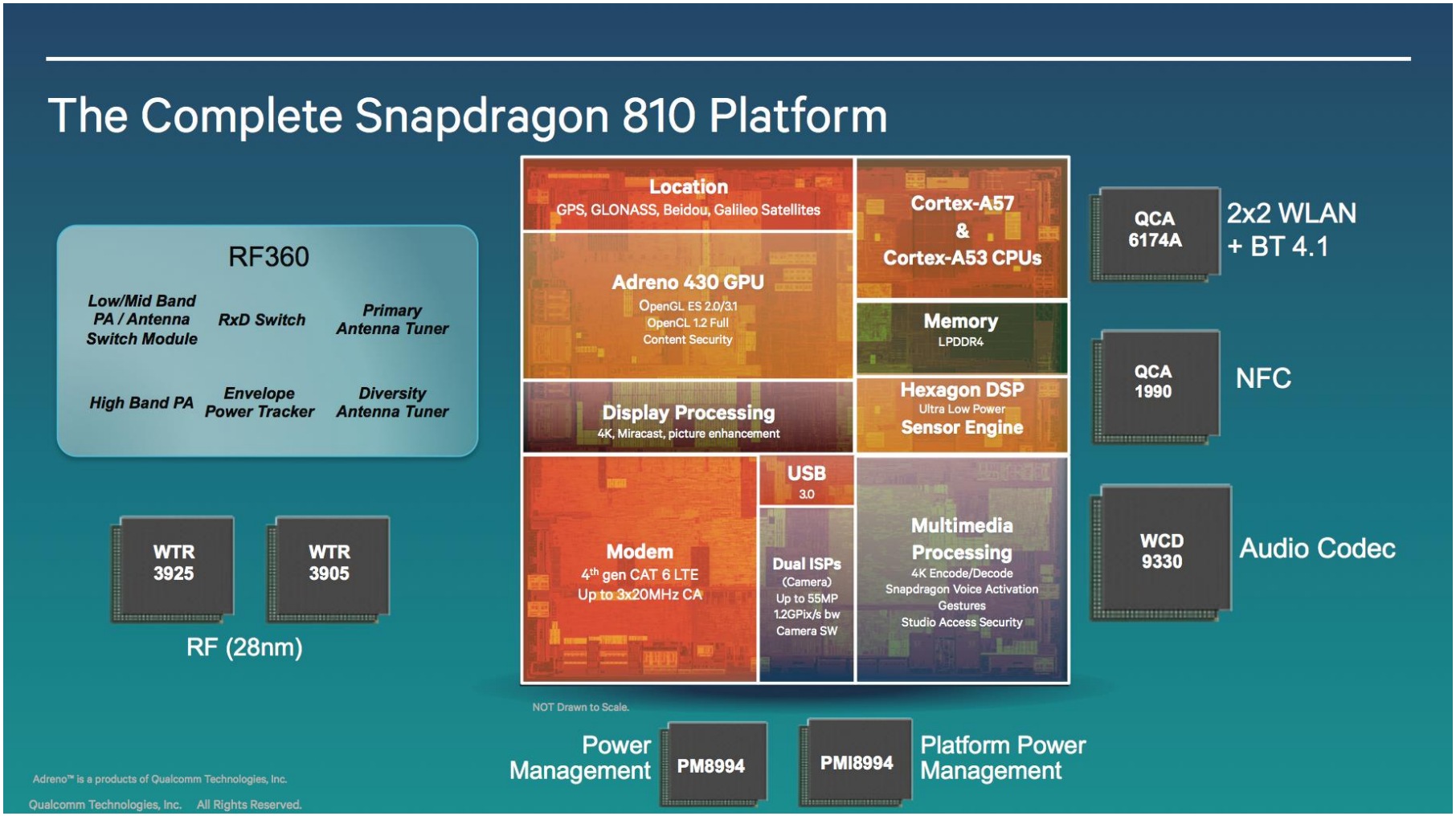


- Multiple SW stacks
- Non perfect interconnect

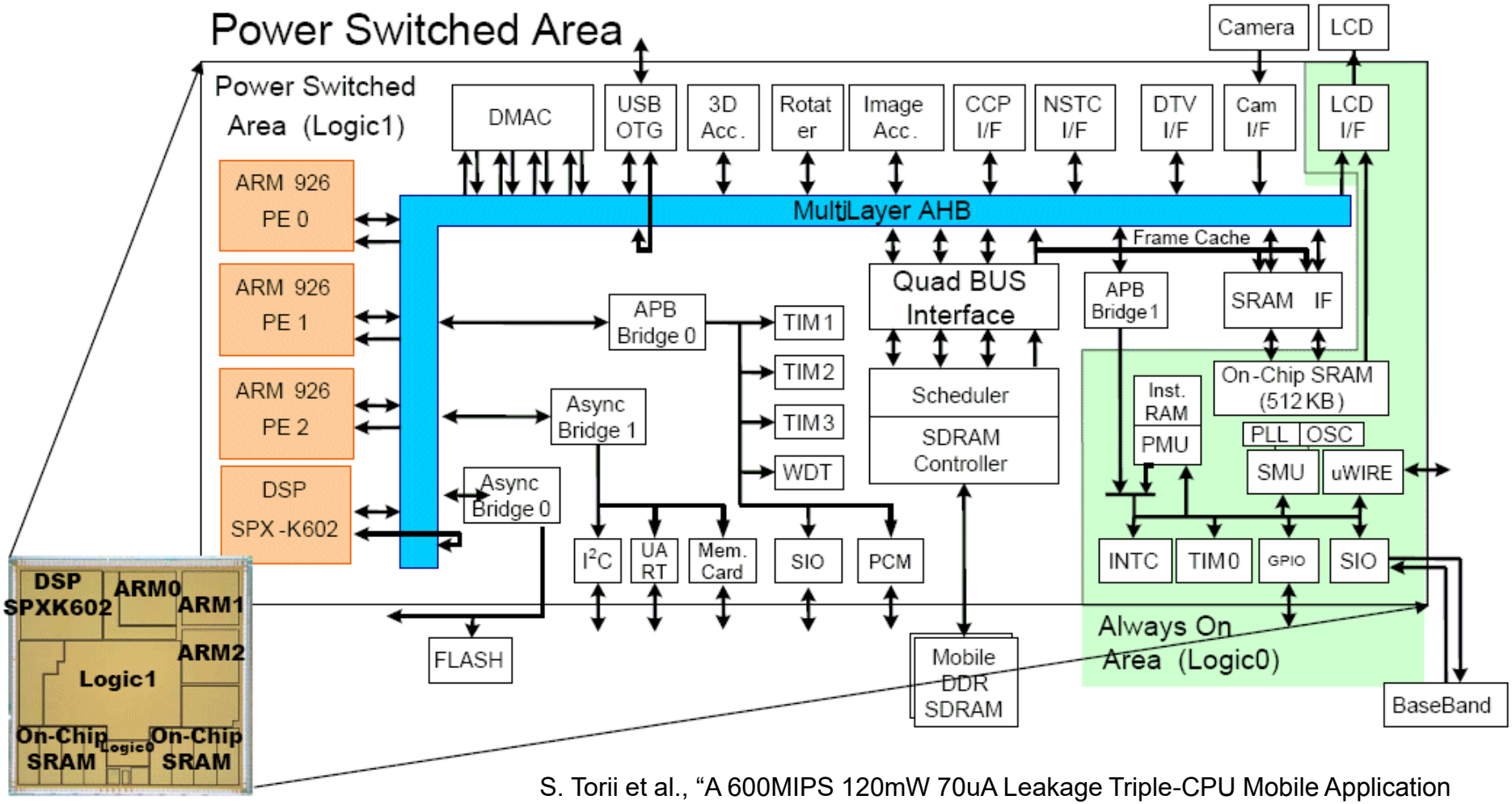


In an SoC, the Role of DSP Architecture?

DSP Architecture
Computer Architecture.

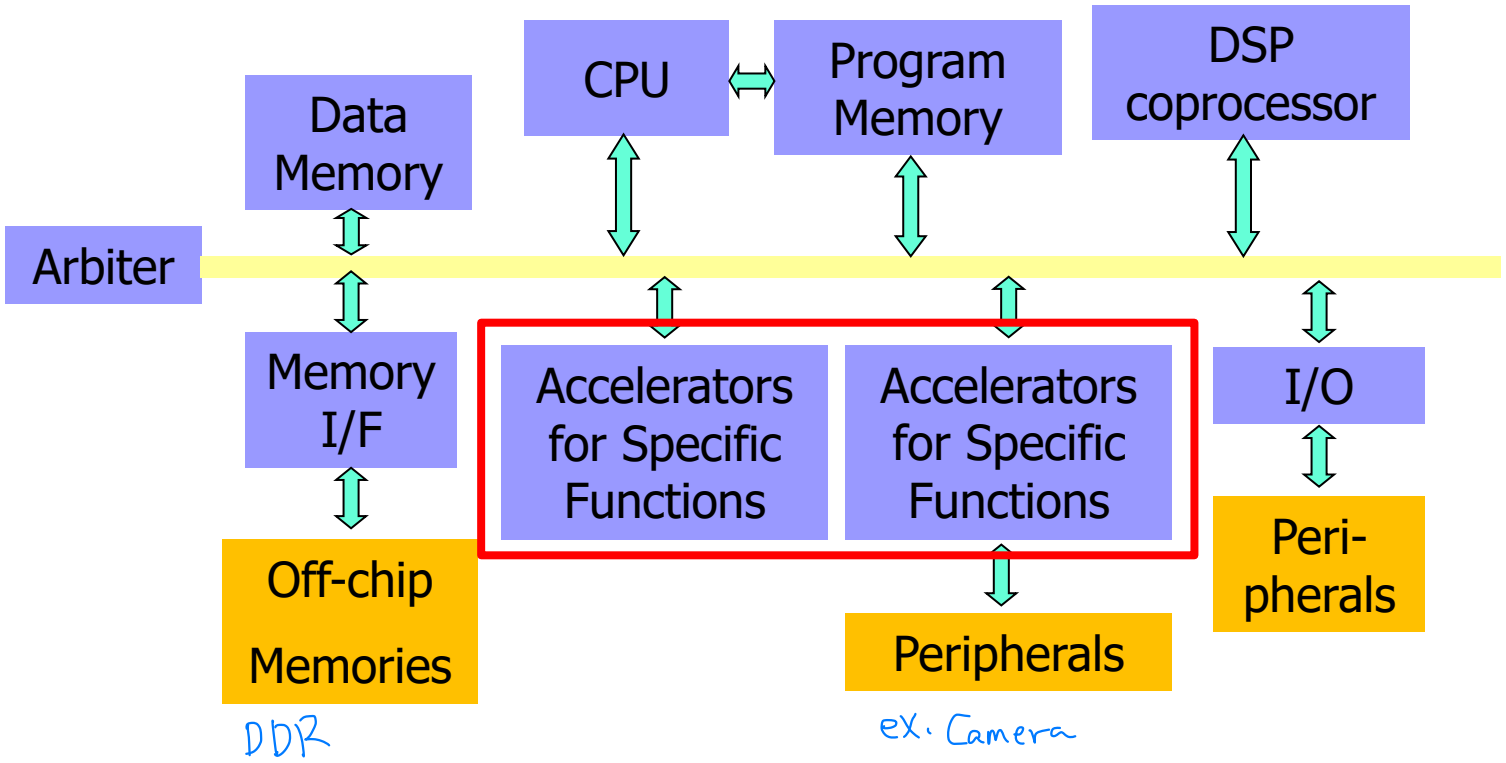


In an SoC, the Role of DSP Architecture?



S. Torii et al., "A 600MIPS 120mW 70uA Leakage Triple-CPU Mobile Application Processor Chip," ISSCC Dig. Tech. Papers, pp.136-137, 2005. (NEC)

Essential Components in an SoC



Essential Components in an SoC

- Processors: CPU, DSP, GPU APU, NPU.
- Bus system
- Memory system
  - MMU, DMA Direct Memory Access
  - On-chip memory: cache, buffer (scratch pad memory)
  - Off-chip memory: DRAM, SRAM, NVM
- I/O peripherals
- Hardware accelerators

Topics to be Covered in this Unit

- Communications between Different IPs
- Hardware accelerator
- Important concept

Communications between Different IPs

BUS Brief

- In a system, various subsystems must have interfaces to one another
- The bus serves as a shared communication link between subsystems
- Advantages
  - Low cost
  - Versatility
- Disadvantage
  - Performance bottleneck

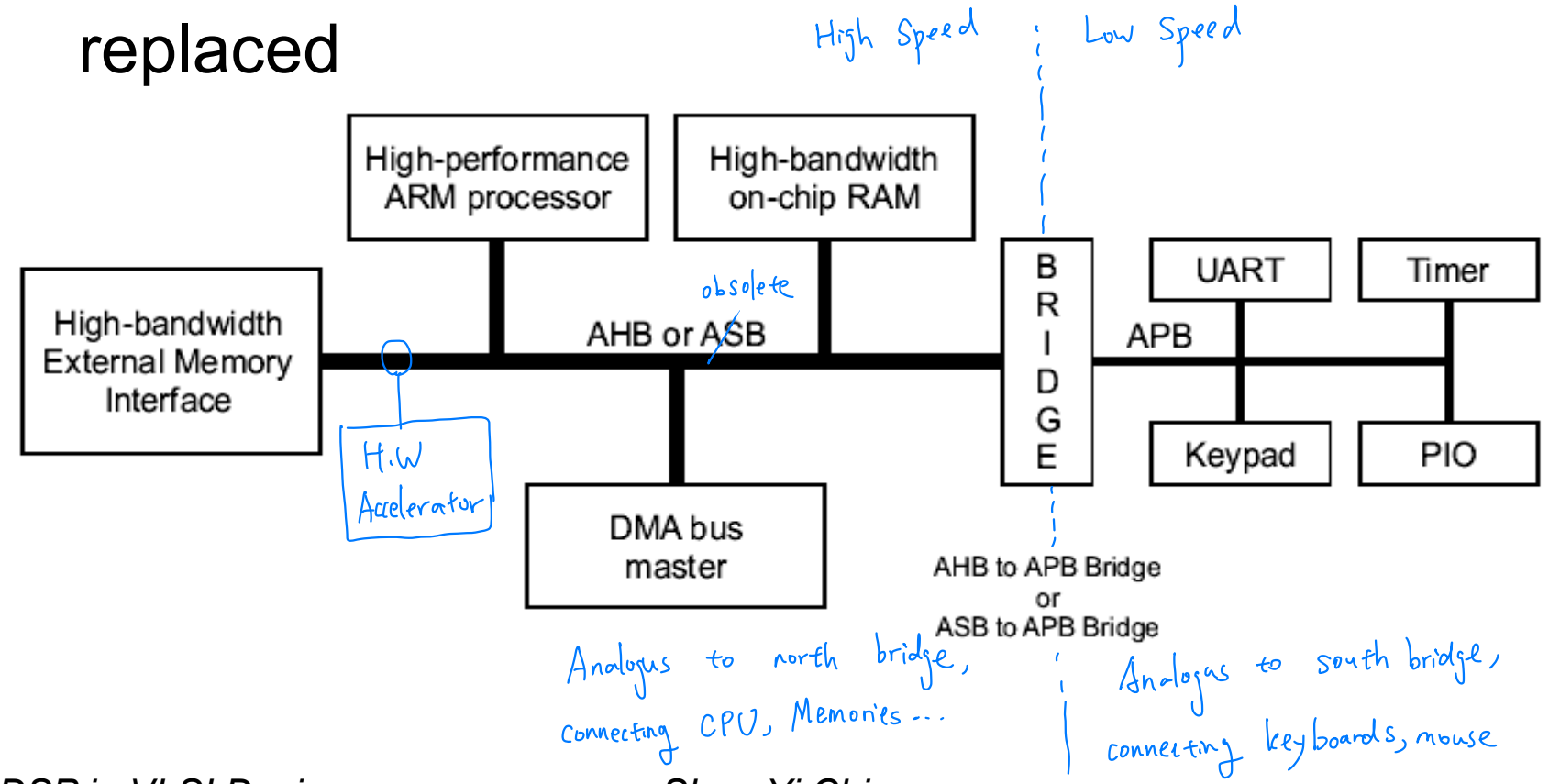
AMBA Introduction

- Advanced Microcontroller Bus Architecture
  - An on-chip communication standard defined by ARM
- AMBA 1.0—5.0
  - AHB, APB, AXI, ACE, CHI, …

There are several bus existing in an chip at the same time.

A Typical AMBA 2.0 System

- Processors or other masters/slaves can be replaced



AHB Components

- AHB master
  - Initiate a read/write operation
  - Only one master is allowed to use the bus
  - uP, DMA, DSP, …
- AHB slave
  - Respond to a read/write operation
  - Address mapping
  - External memory I/F, APB bridge, internal memory, …
- AHB arbiter
  - Ensure that which master is active
  - Arbitration algorithm is not defined in ABMA spec.
- AHB decoder
  - Decode the address and generate select signal to slaves

APB Components

- AHB2APB Bridge
  - Provides latching of all address, data, and control signals
  - Provides a second level of decoding to generate slave select signals for the APB peripherals
- All other modules on the APB are APB slaves
  - Un-pipelined
  - Zero-power interface
  - Timing can be provided by decode with strobe timing
  - Write data valid for the whole access

Communications

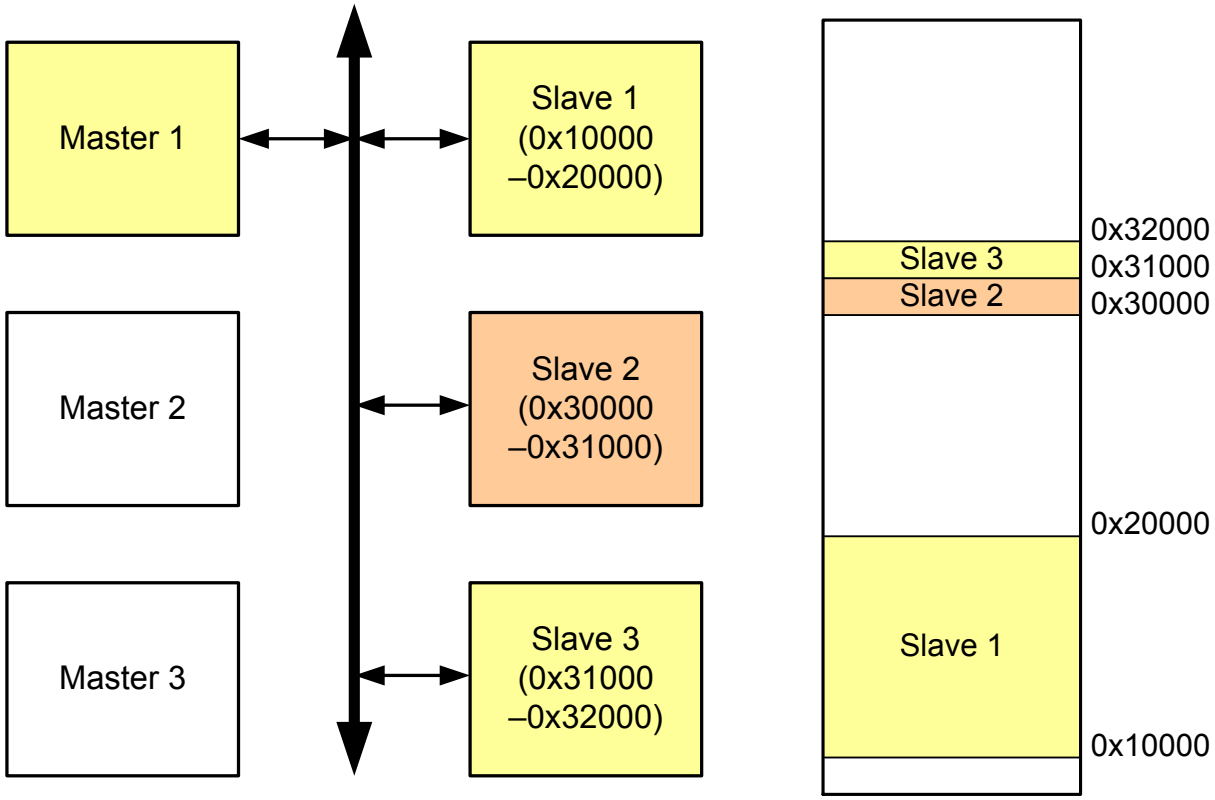
- CPU (master) ←→ IP (slave)
- IP (master) ←→ IP (slave)

Memory Mapped I/O

- Each **slave** occupies a range of (>1KB) address space in the system
- All the slaves are **addressable**
- Memory mapped register/memory
- CPU/IP and read/write data to other IP **as read/write data from/to memory**

Communication between IPs

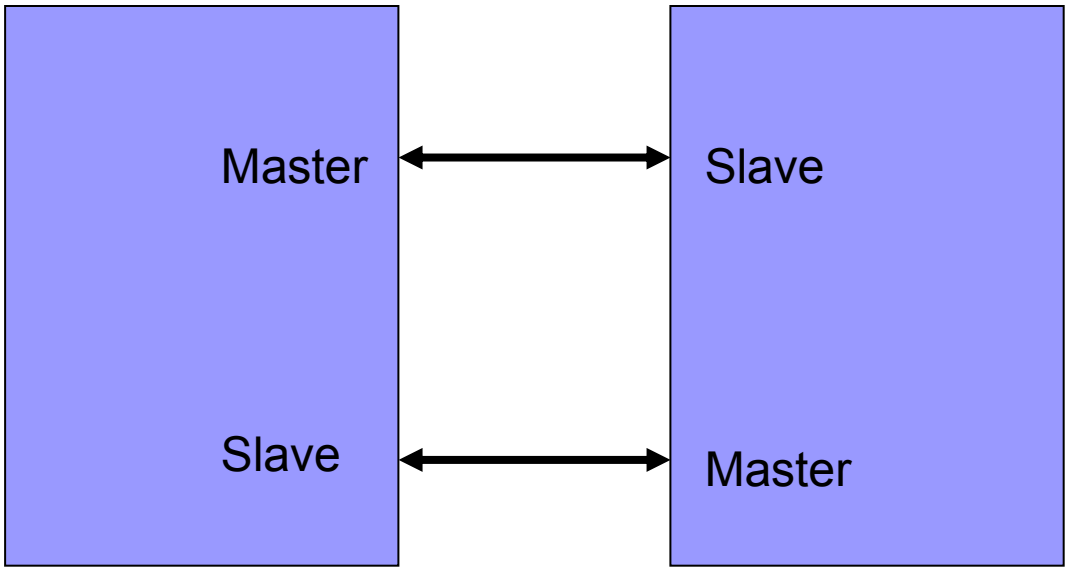
- After the master is granted by the arbiter, it can access all the slaves on the bus



Write(address, data)
Read(address, data)

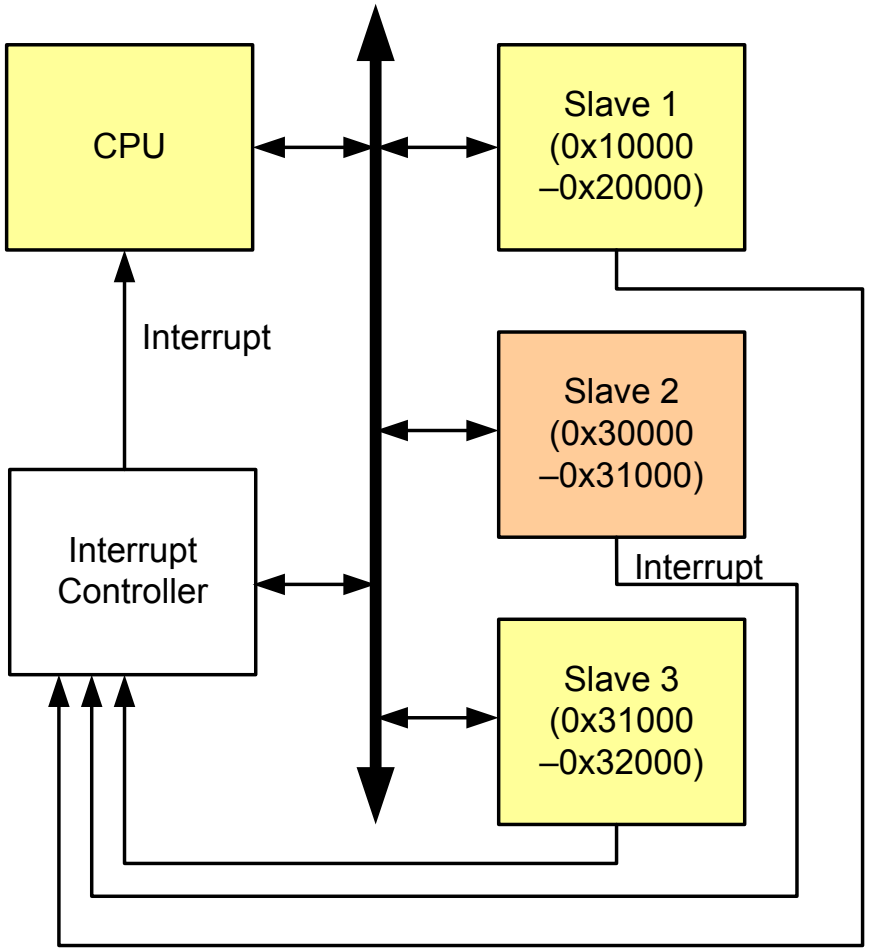
Ex:
Write(0x30020, 0x0)
Read(0x30100, &temp)

An IP Can Have Both Master and Slave I/F

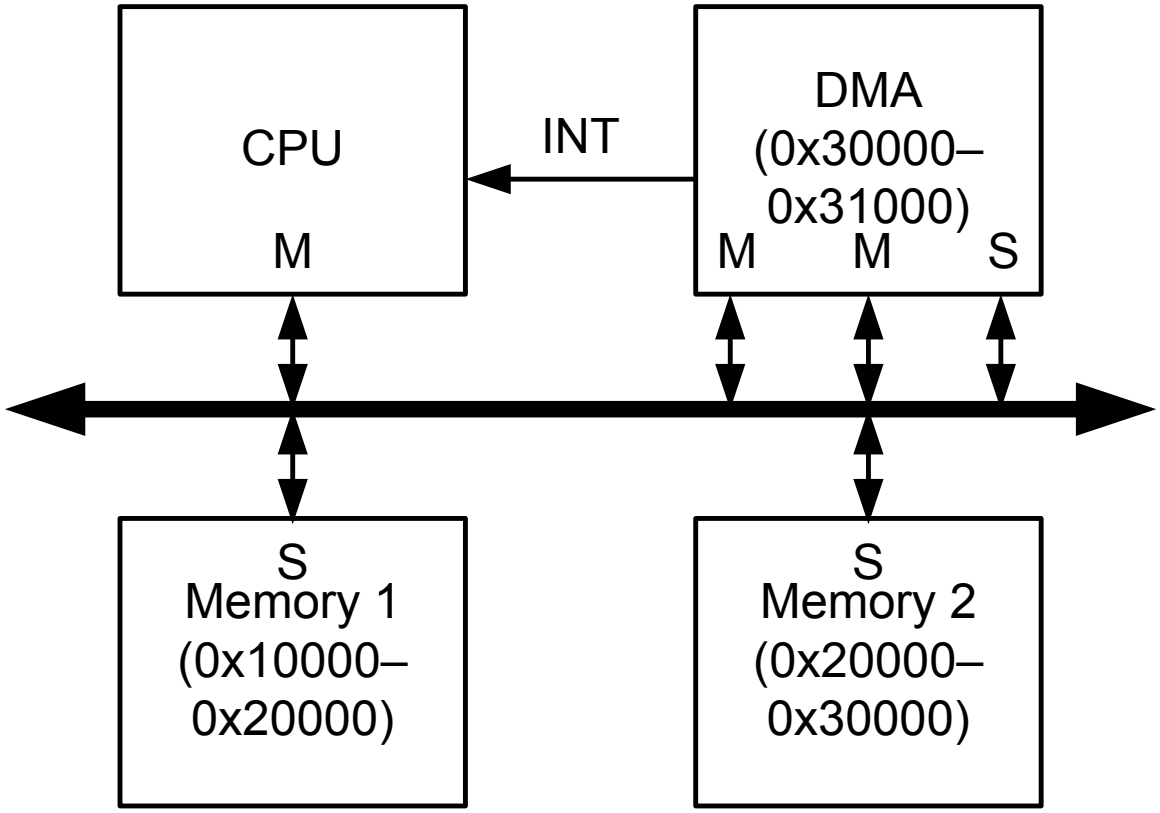


Communication between CPU and IP

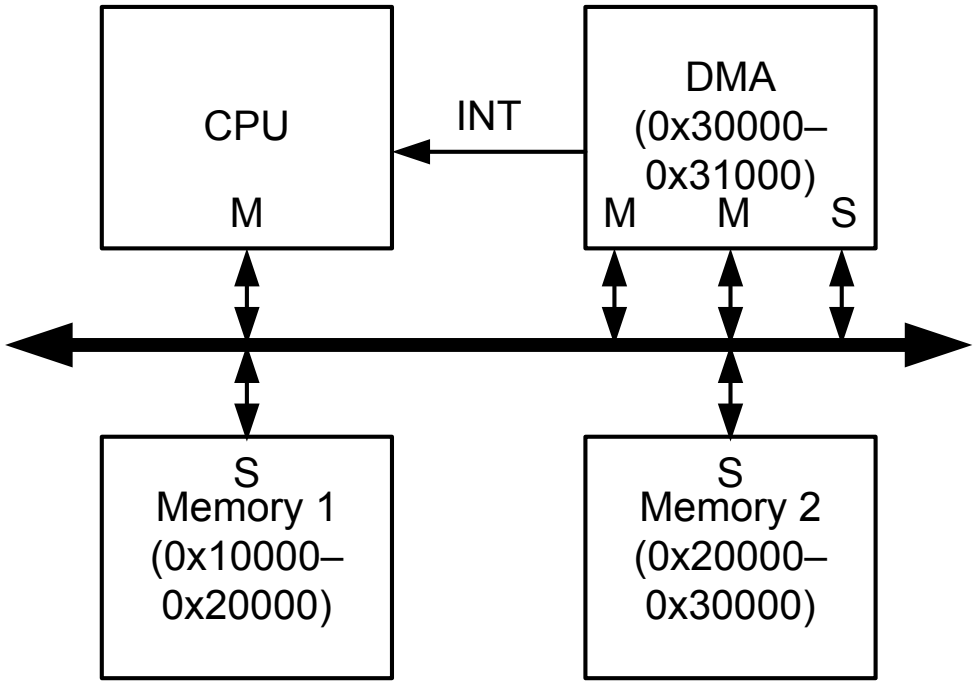
- CPU is always the master
- The IP is always the slave
- The IP can initiate the feedback with **interrupt**
- After interrupt, the CPU enters interrupt mode, and the interrupt is handled with **interrupt service routine (ISR)**



Example: DMA (Direct Memory Access, an accelerator)



Example: DMA

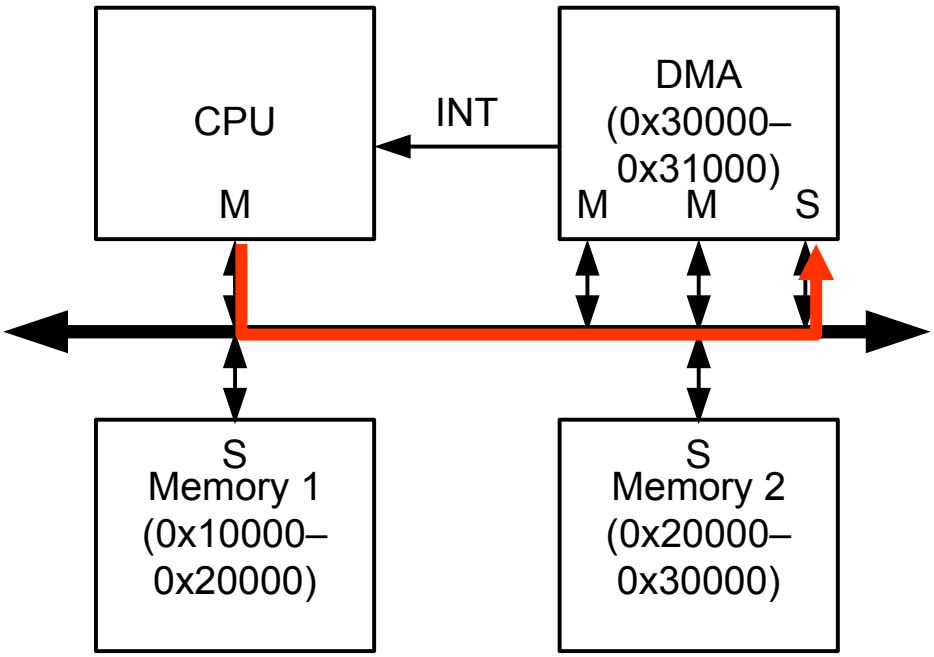


Base = 0x30000

Address + base	Function
0x00	1: start, 0: stop
0x04	Status. 0: ready; 1: busy
0x08	Source address
0x0C	Destination address
0x10	Size

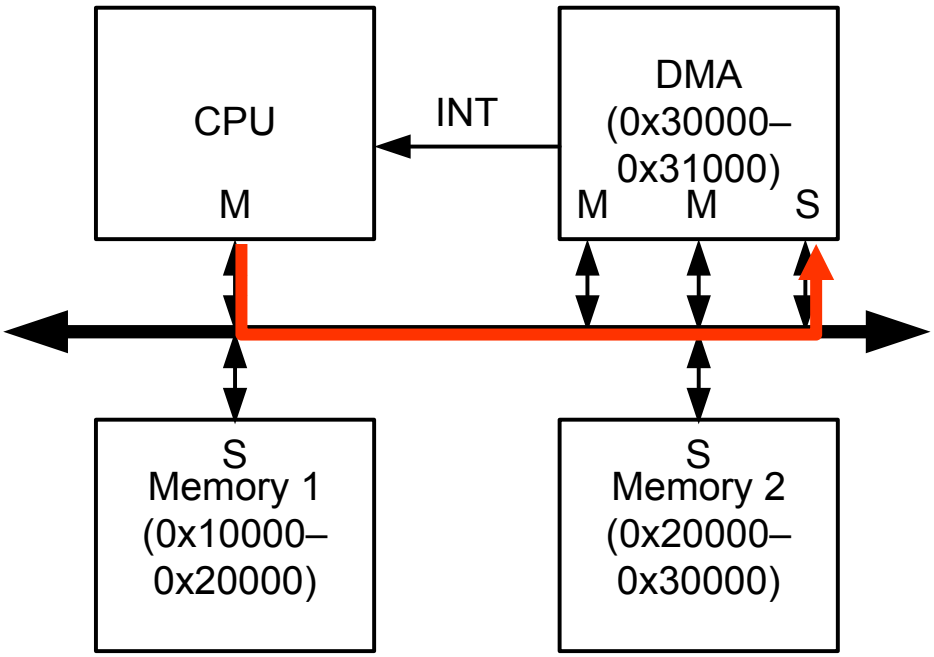
Example: DMA

- Step 0: CPU check the status of DMA to make sure it is ready to be used
  - While(1)
  - {
  - Read(0x30004, &status)
  - if(status == 0)
  - break;
  - }



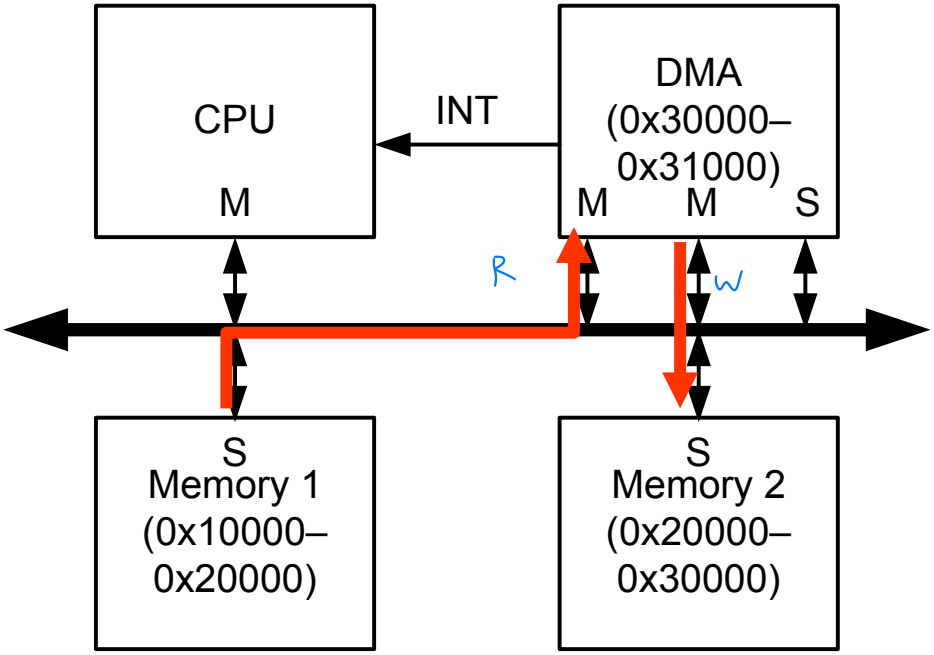
Example: DMA

- Step 1: CPU sets the (source address), (destination address), and (size) with the slave I/F
  - Write(0x30008, 0x10000)
  - Write(0x3000C, 0x20000)
  - Write(0x30010, 0x100)
- Step 2: starts DMA
  - Write(0x30000, 0x1)



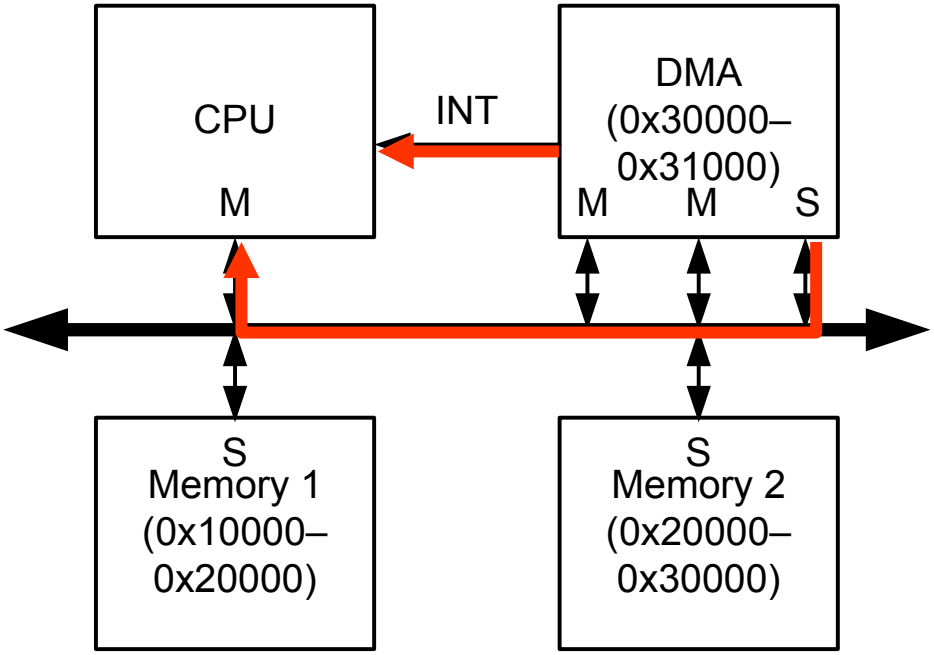
Example: DMA

- Step 3: DMA moves data from memory 1 to memory 2 with the two master I/F



Example: DMA

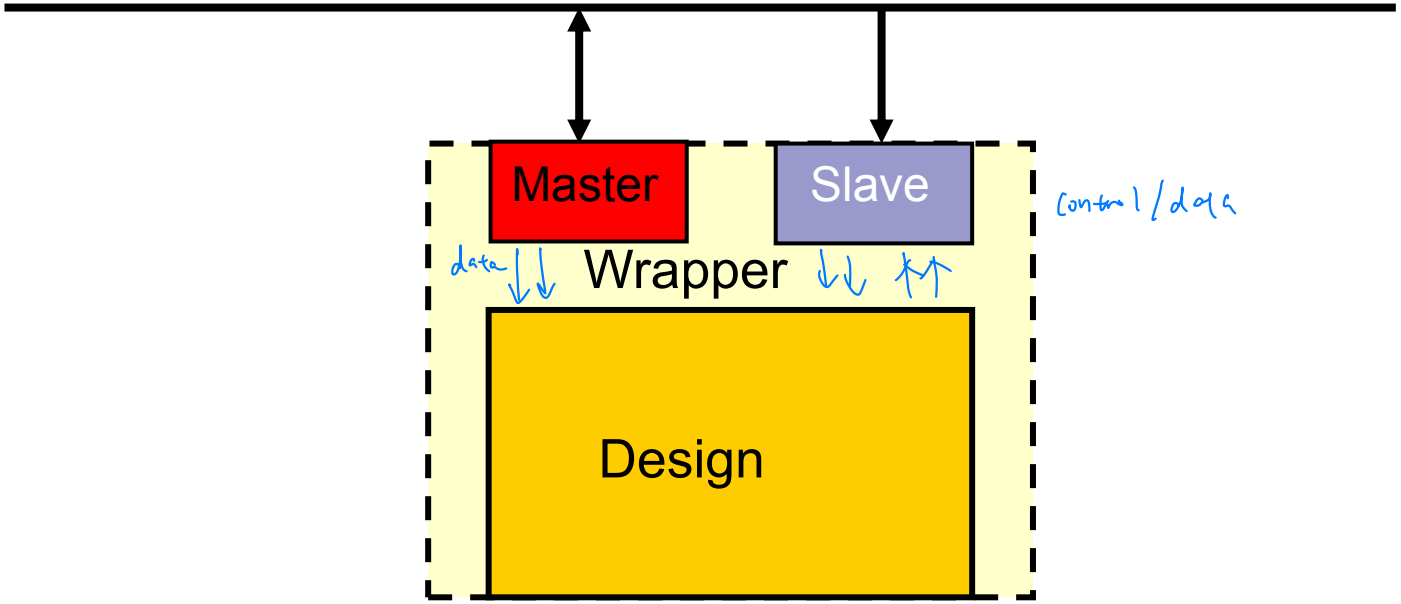
- Step 4: DMA interrupts CPU
- Step 5: CPU checks the status of DMA
  - Read(0x30004, &status)



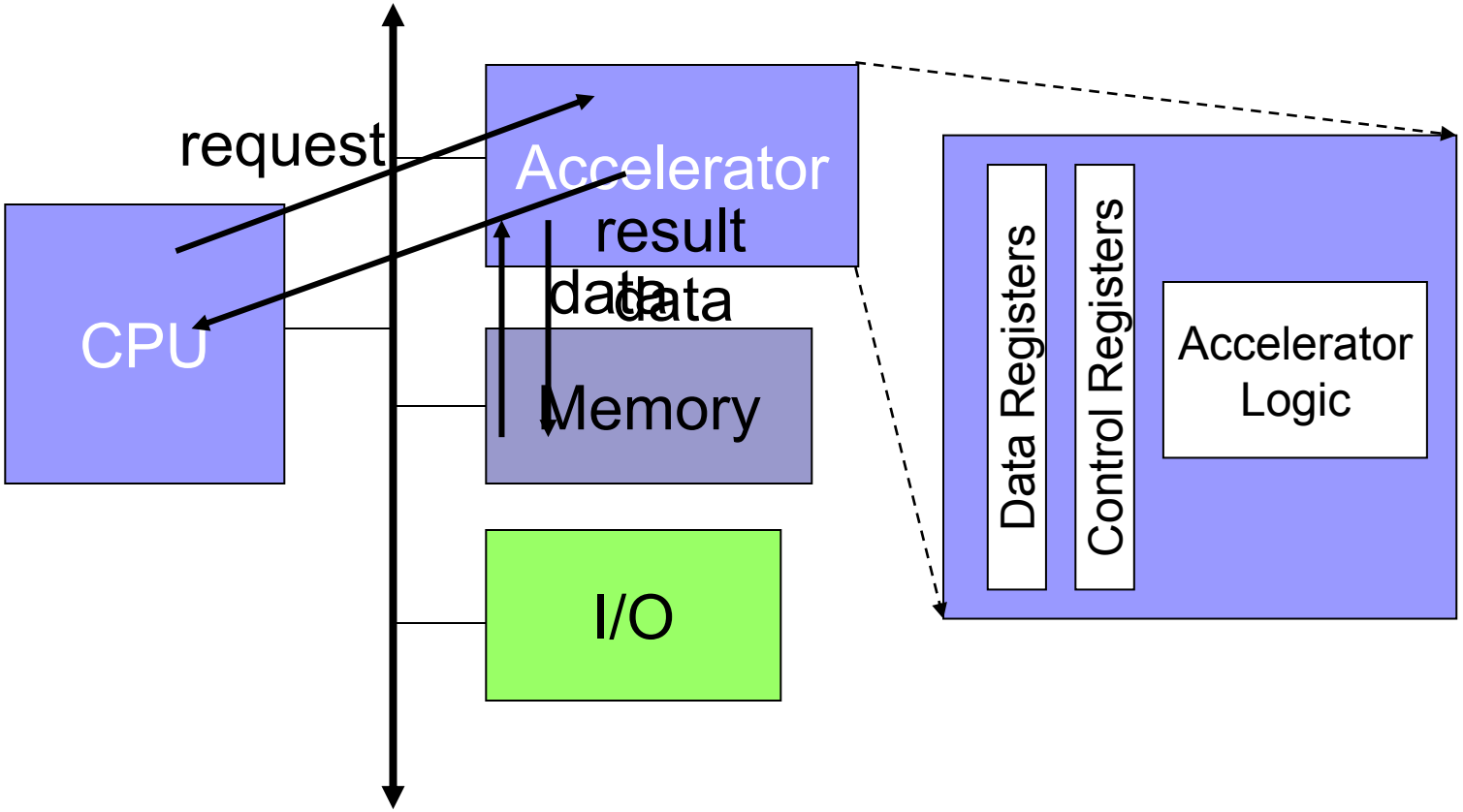
Hardware Accelerator

Hardware Accelerator with both Master and Slave I/F

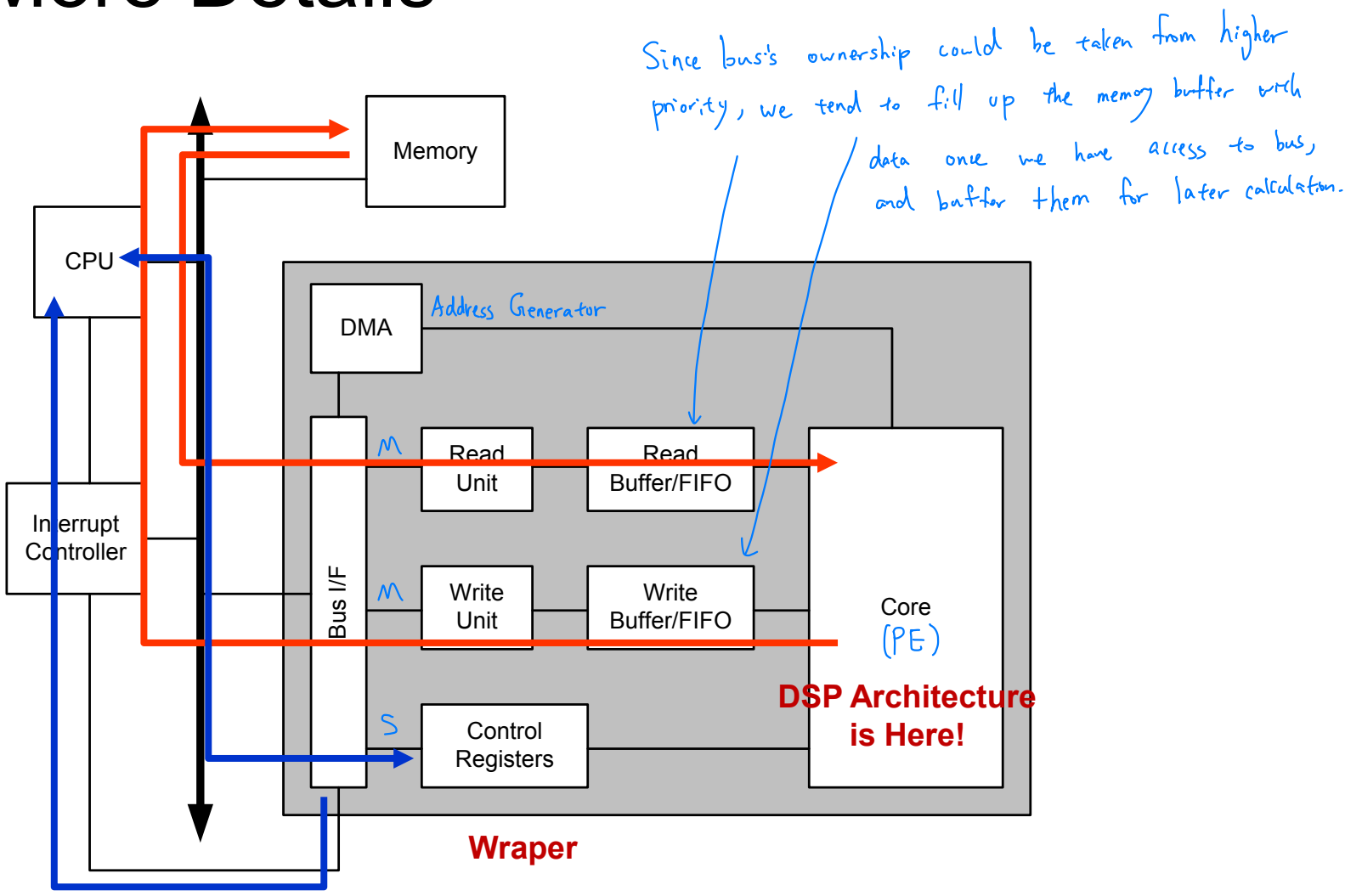
- Both master port & slave port in your design
  - Read/write data via master port
  - Configured by the others via slave port
  - Such as some parameters set by the processor



Typical Accelerated System Architecture



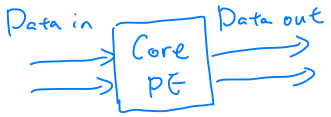
Accelerator Architecture Framework with More Details

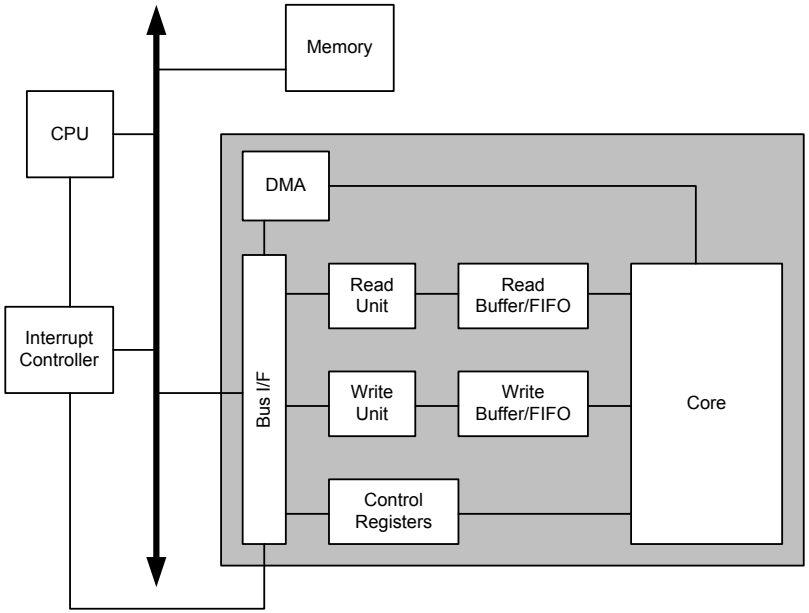


Accelerator/CPU Interface

- Accelerator registers provide control registers for CPU
- Data registers can be used for small data objects
- Accelerator may include special-purpose read/write logic DMA
  - Especially valuable for large data transfers

Differences?

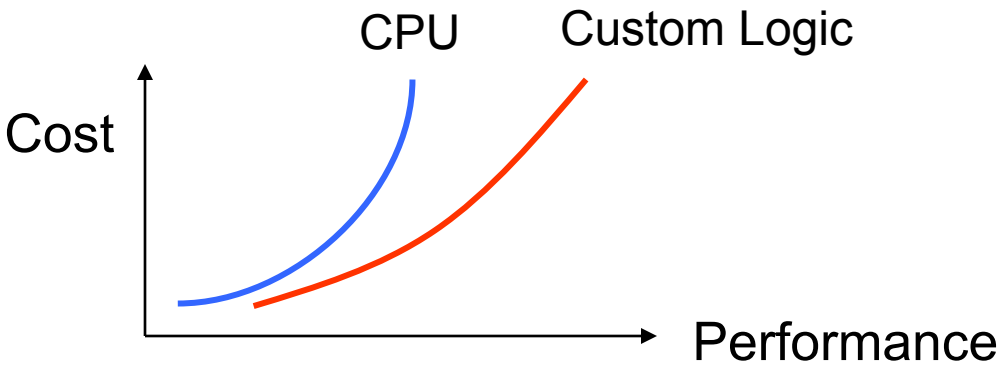




	Conventional DSP Architecture	DSP Architecture in an SoC
I/O	• Direct I/O	• Through system bus -- Memory -- CPU • Direct I/O
Fired by	• Data	• Host processor
Design Consi-derations	• Cost, performance, power	• Cost, performance, power • Communication • Memory system • Cooperation with processors HW/SW Partition.

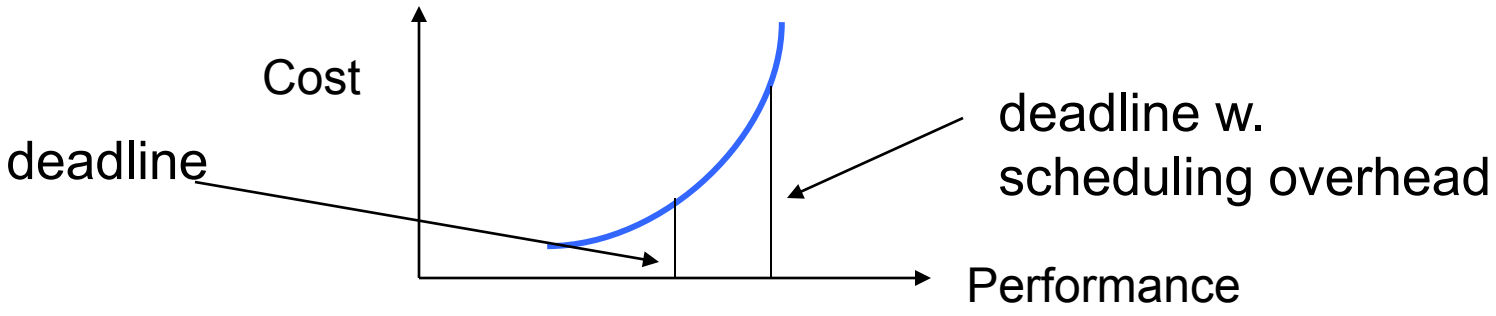
Why Accelerators?

- Better cost/performance.
  - Custom logic may be able to perform operation faster than a CPU of equivalent cost
  - CPU cost is a non-linear function of performance



Why Accelerators?

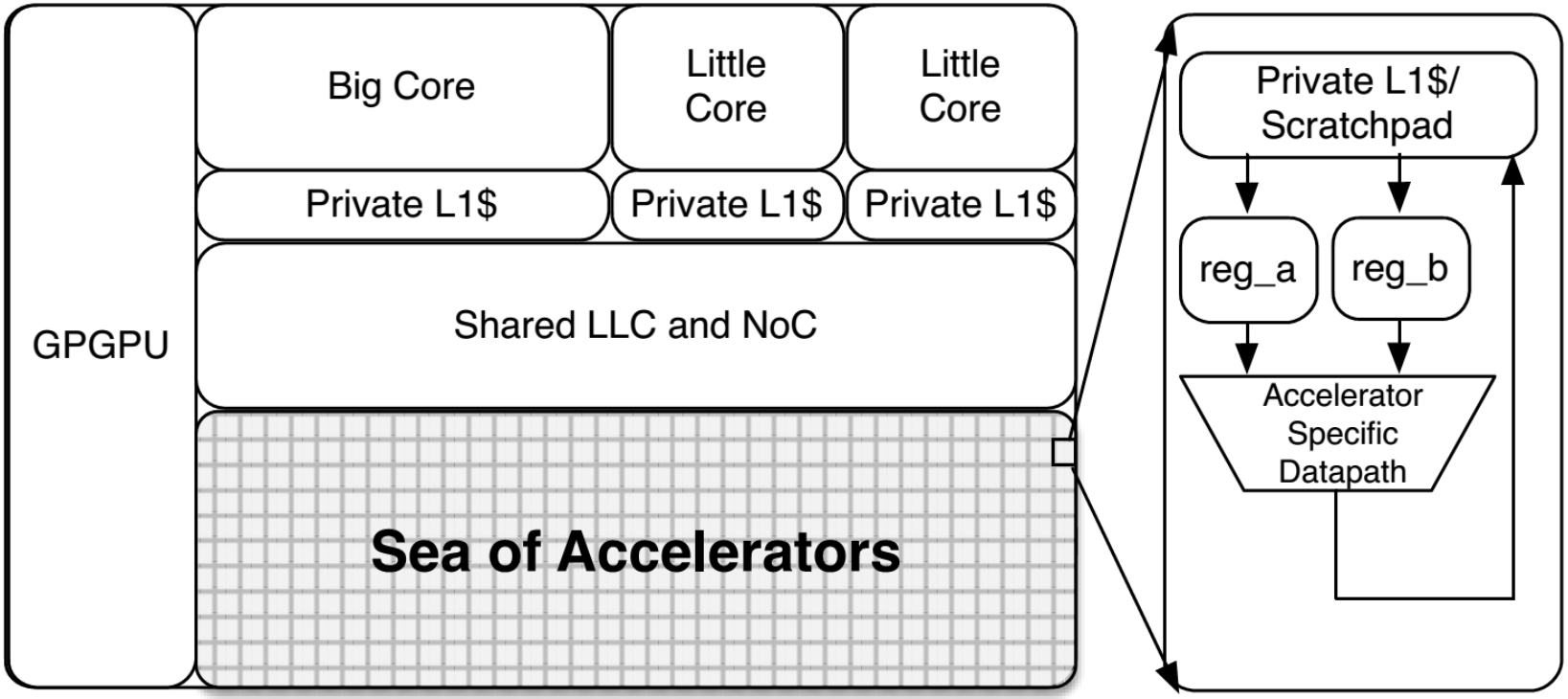
- Better real-time performance
  - Put time-critical functions on less-loaded processing elements
  - Remember RMS utilization---extra CPU cycles must be reserved to meet deadlines.



Why Accelerators?

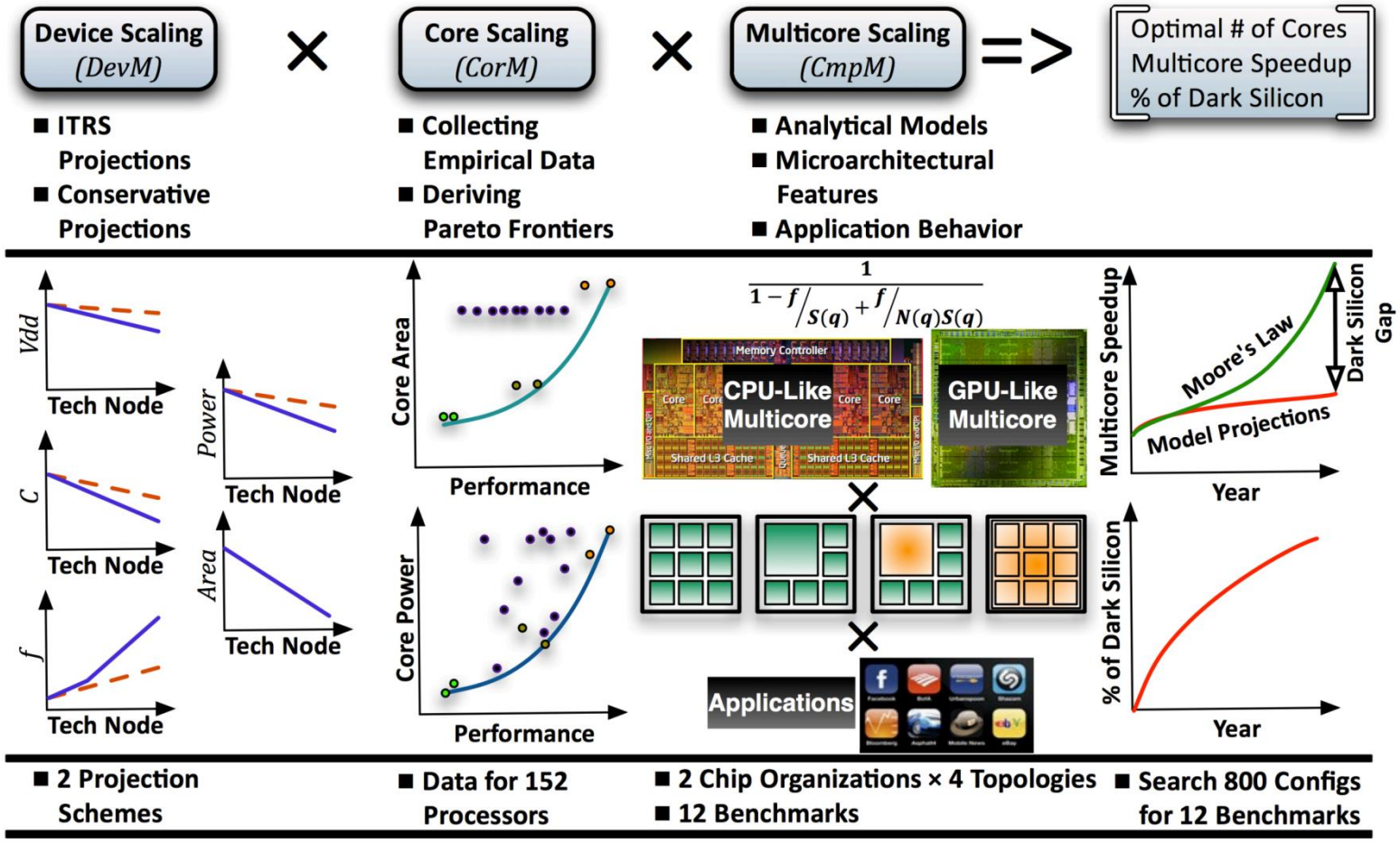
- Good for processing I/O in real-time ex. sensors, displays
- **May consume less energy**
- May be better at streaming data
- May not be able to do all the work on even the largest single CPU

Why Accelerators?



Ref: Yakun Sophia Shao, Brandon Reagen, Gu-Yeon Wei, David Brooks, "Aladdin: A Pre-RTL, Power-Performance Accelerator Simulator Enabling Large Design Space Exploration of Customized Architectures," in Proc. International Symposium on Computer Architecture (ISCA), 2014.

Dark Silicon



Ref: H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," Micro, IEEE, 2012.

Types of Applications Suited to Hardwired Accelerators

- Functions requiring operations that do **not map well onto a CPU's data operation**
  - Bit level operations (CPU operates at word level)
  - Operations requiring too many registers (eg. Image Processing)
  - To control the precision of the arithmetic
- Highly responsive **input and output operations** may be best performed by an accelerator with an attached I/O unit
- **Streaming data**, such as wireless and multimedia

Accelerated System Design

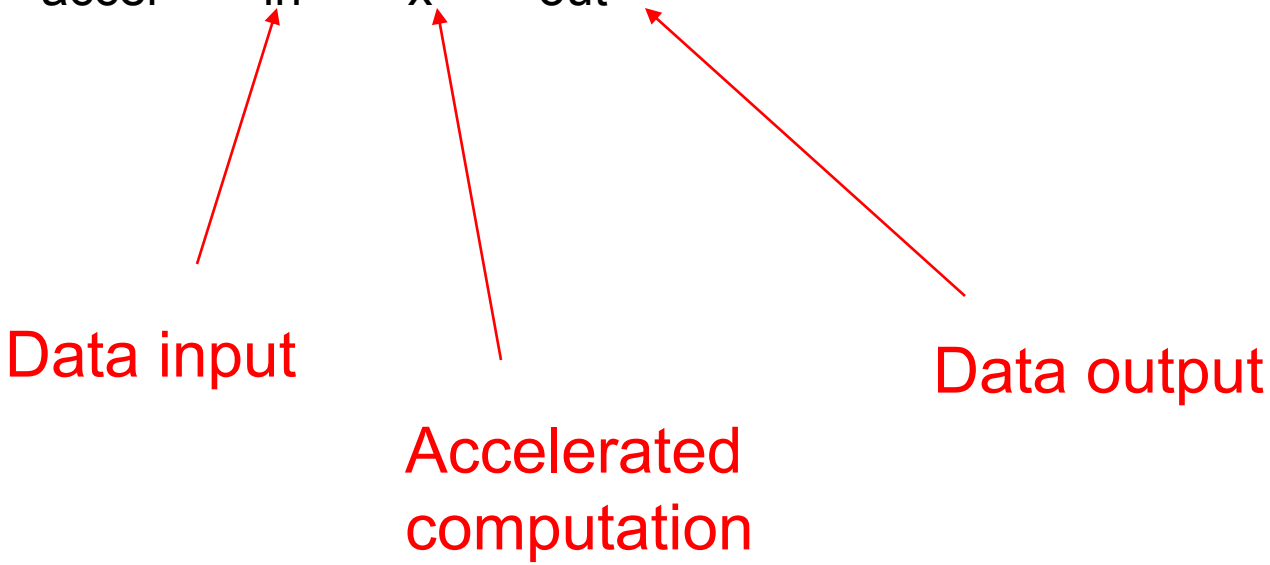
- First, determine that the system really needs to be accelerated
  - How much faster is the accelerator on the **core function**?
  - How much **data transfer overhead**?
- Design the accelerator itself
- Design CPU interface to accelerator

Performance Analysis

- Critical parameter is speedup: how much faster is the system with the accelerator?
- Must take into account:
  - Accelerator execution time
  - Data transfer time
  - Synchronization with the master CPU

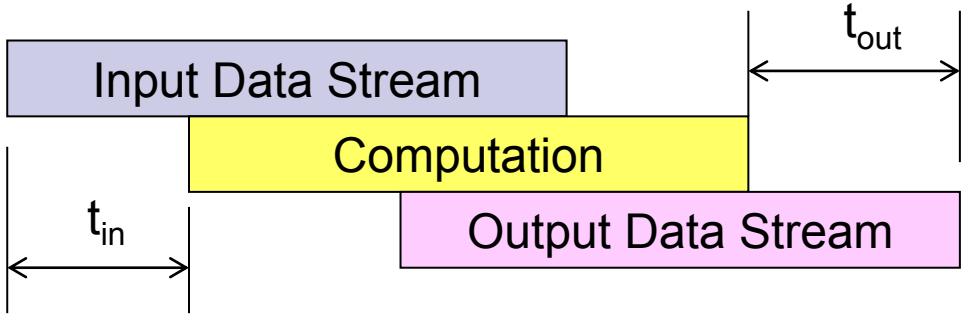
Accelerator Execution Time

- Total accelerator execution time:
  - $t_{accel} = t_{in} + t_x + t_{out}$



Accelerator Execution Time

- A more sophisticated accelerator could try to overlap input and output with computation
- t_{in}
  - Non-overlapped read time
  - Determined by the amount of data read in before starting computation
- t_{out}
  - Non-overlapped write time
  - The length of time between the last computation and the last data output



Data Input/Output Times

- Bus transactions include
  - Flushing register/cache values to main memory;
  - Time required for CPU to set up transaction;
  - Overhead of data transfers by bus packets, handshaking, etc.

Accelerator Speedup

- Assume a loop is executed n times
- Compare accelerated system to non-accelerated system:
  - $S = n(t_{CPU} - t_{accel})$

    $= n[t_{CPU} - (t_{in} + t_x + t_{out})]$

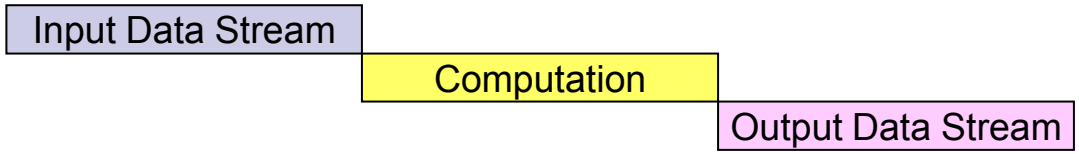
Execution time on CPU

Sources of Parallelism

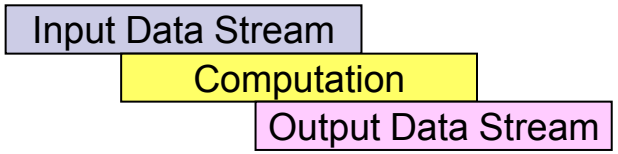
- Overlap I/O and accelerator computation
  - Perform operations in batches, read in second batch of data while computing on first batch
- Find other works to do on the CPU
  - May reschedule operations to move work after accelerator initiation

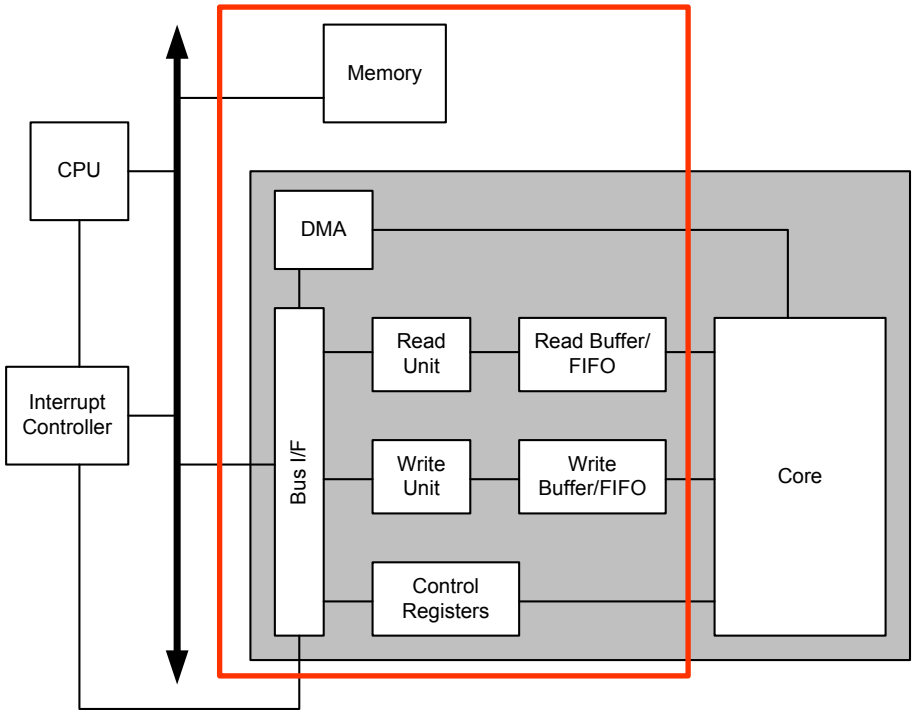
Memory Design is the Key

Single Buffer:



Double Buffer:

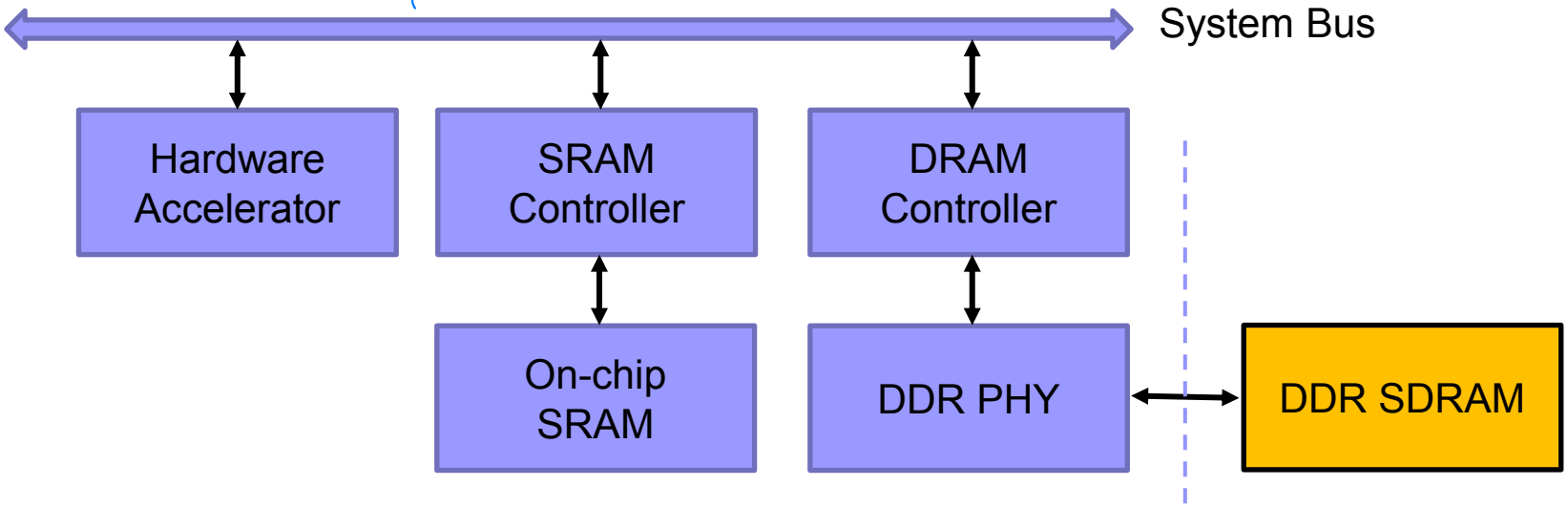




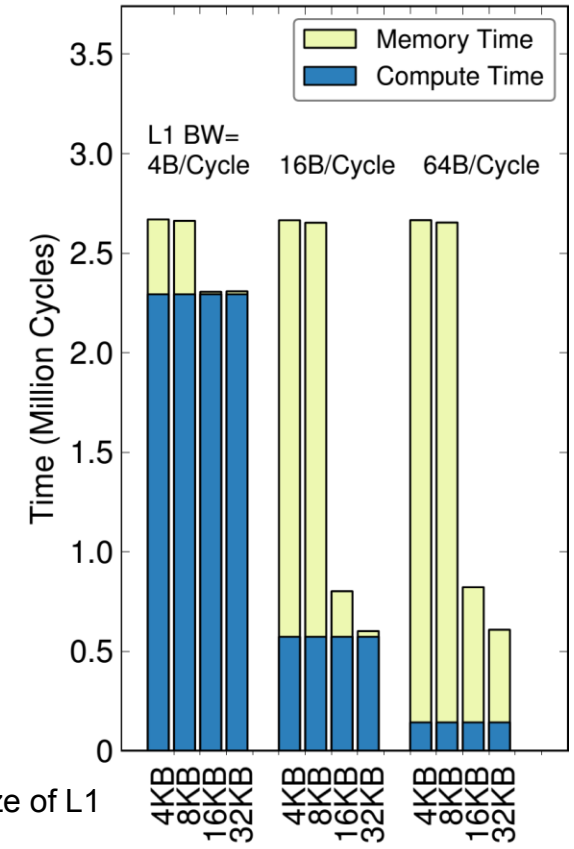
Memory Design is the Key

- Characteristics of bus access
  - Not always available
  - Highly related to the characteristics of memory access
  - Single access vs. burst access
  - Ex. DRAM has variable access time

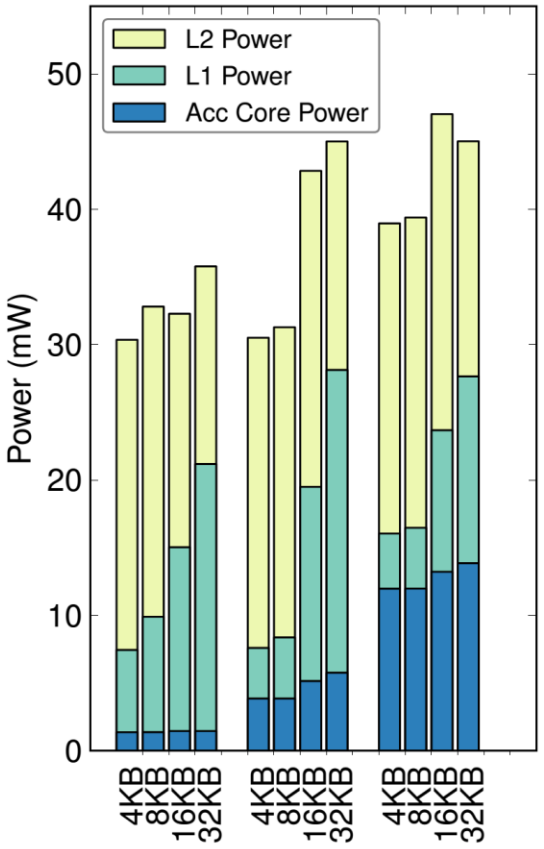
Access DRAM in sequential address is more efficient.



Memory Design is the Key

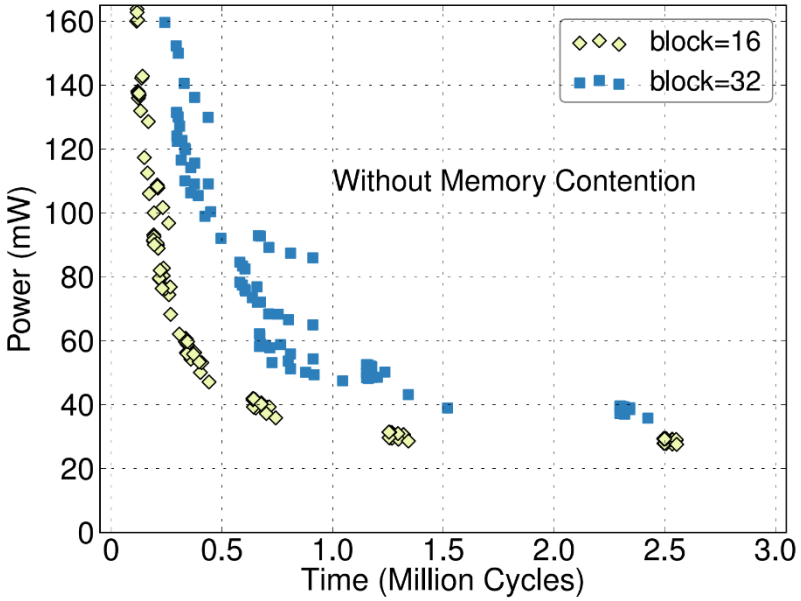


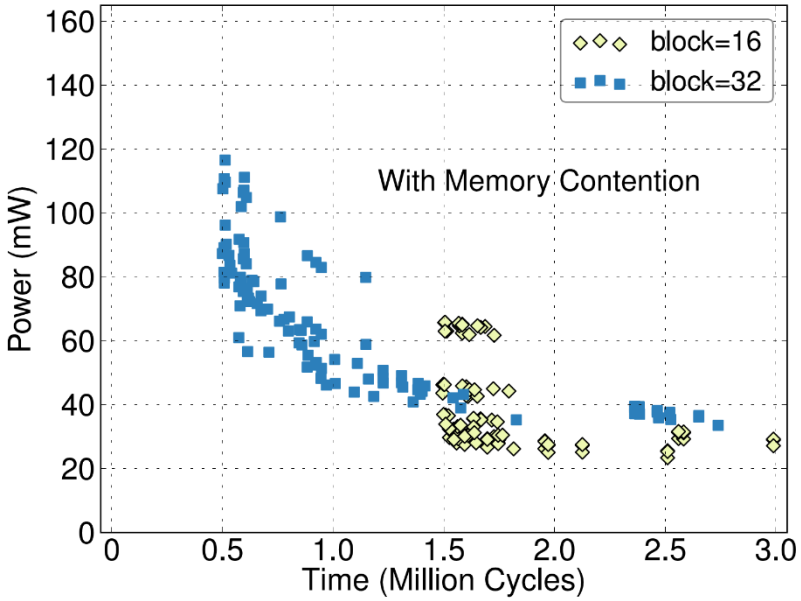
Size of L1



Ref: Yakun Sophia Shao, Brandon Reagen, Gu-Yeon Wei, David Brooks, "Aladdin: A Pre-RTL, Power-Performance Accelerator Simulator Enabling Large Design Space Exploration of Customized Architectures," in Proc. International Symposium on Computer Architecture (ISCA), 2014.

Memory Design is the Key

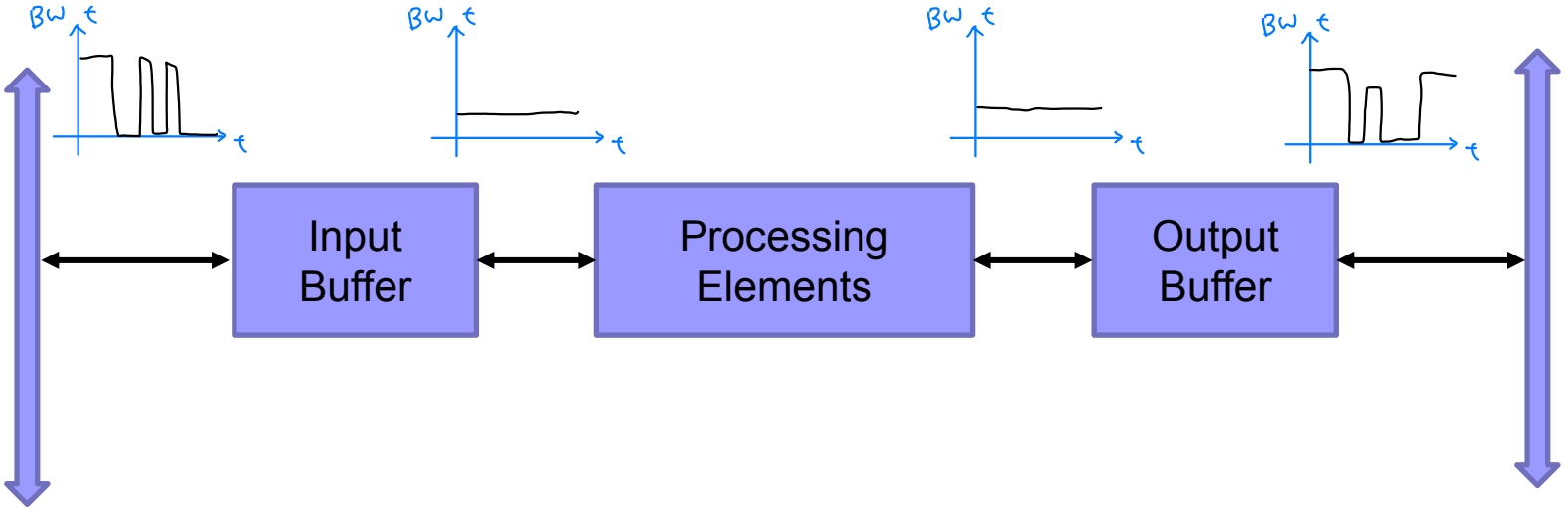




Ref: Yakun Sophia Shao, Brandon Reagen, Gu-Yeon Wei, David Brooks, “Aladdin: A Pre-RTL, Power-Performance Accelerator Simulator Enabling Large Design Space Exploration of Customized Architectures,” in Proc. International Symposium on Computer Architecture (ISCA), 2014.

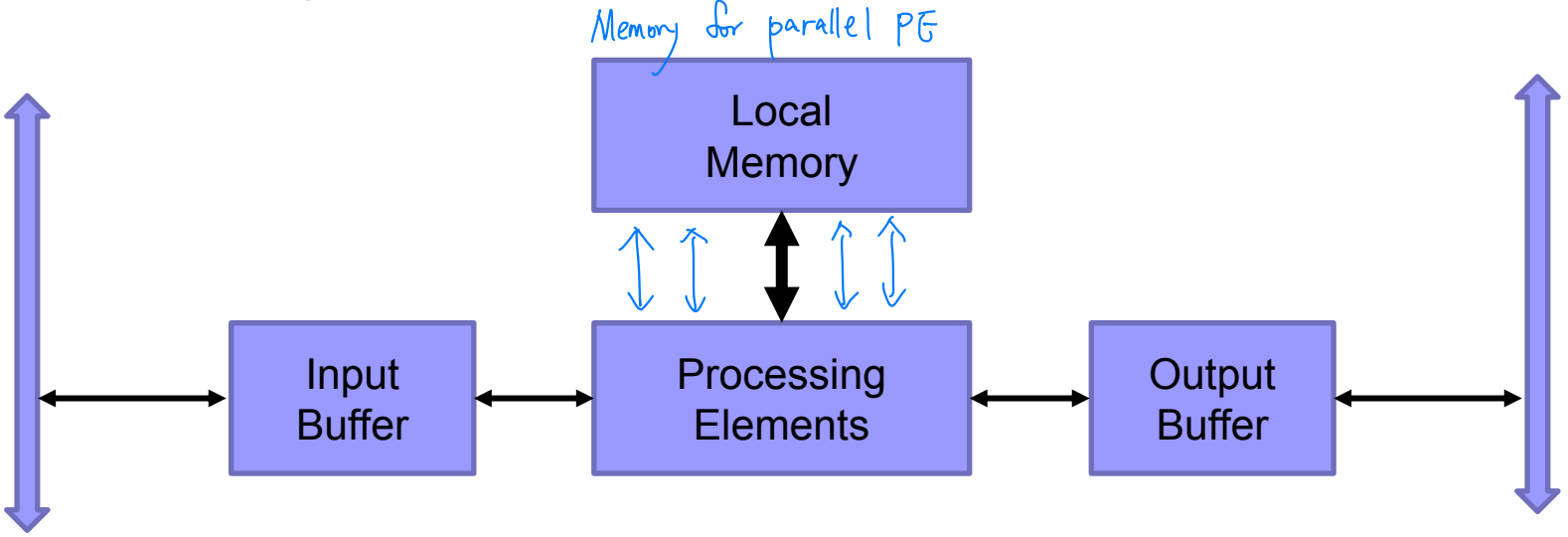
Different Design Characteristics

- Communication/bus/memory bound
  - Fully-pipelined design
  - Ex: filter
- Carefully design input/output buffer (FIFO) and secure stable bus bandwidth



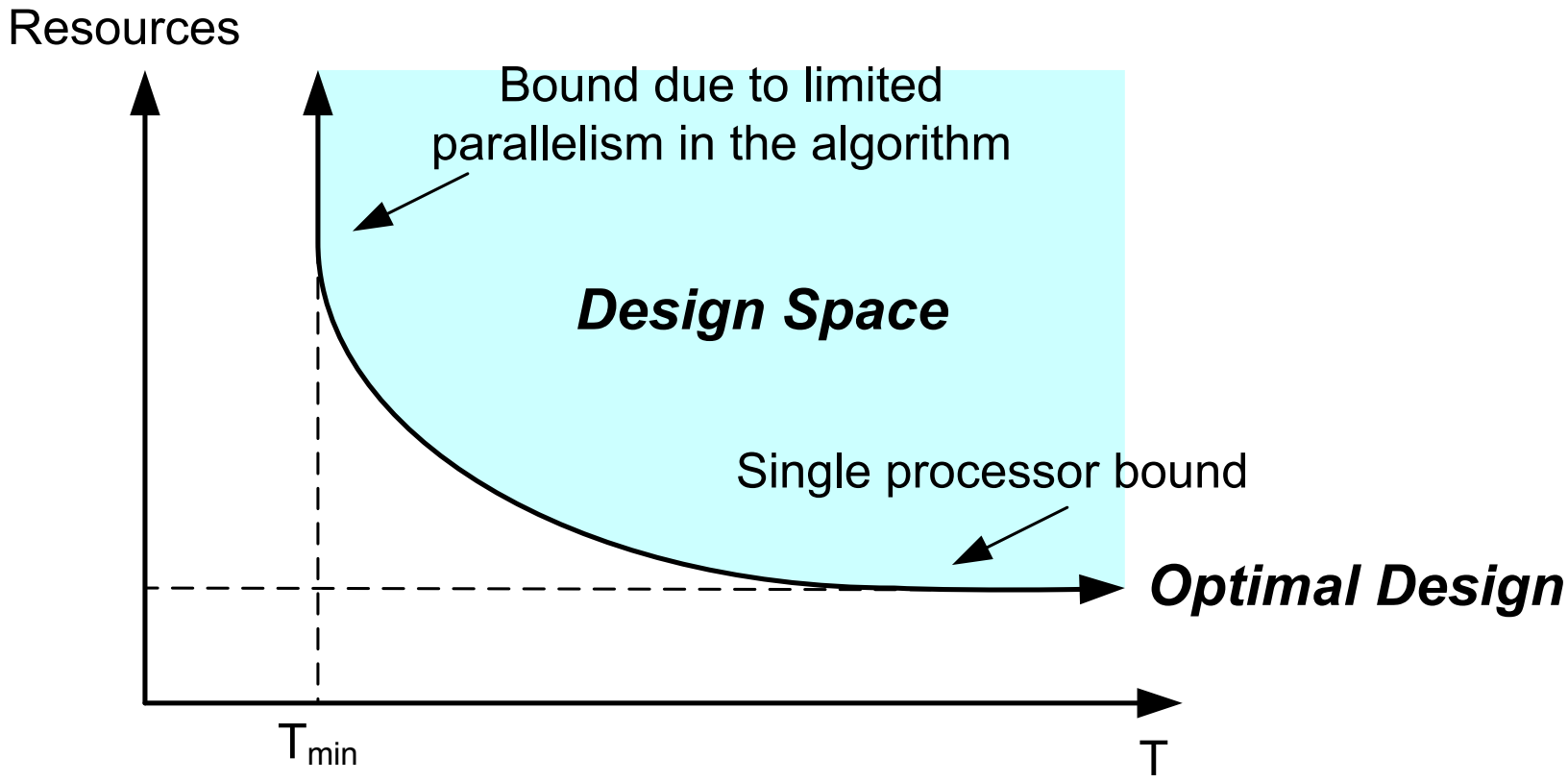
Different Design Characteristics

- Computation bound
  - Parallel design
  - Ex: image analysis
- Carefully design local memory to support parallel processing elements

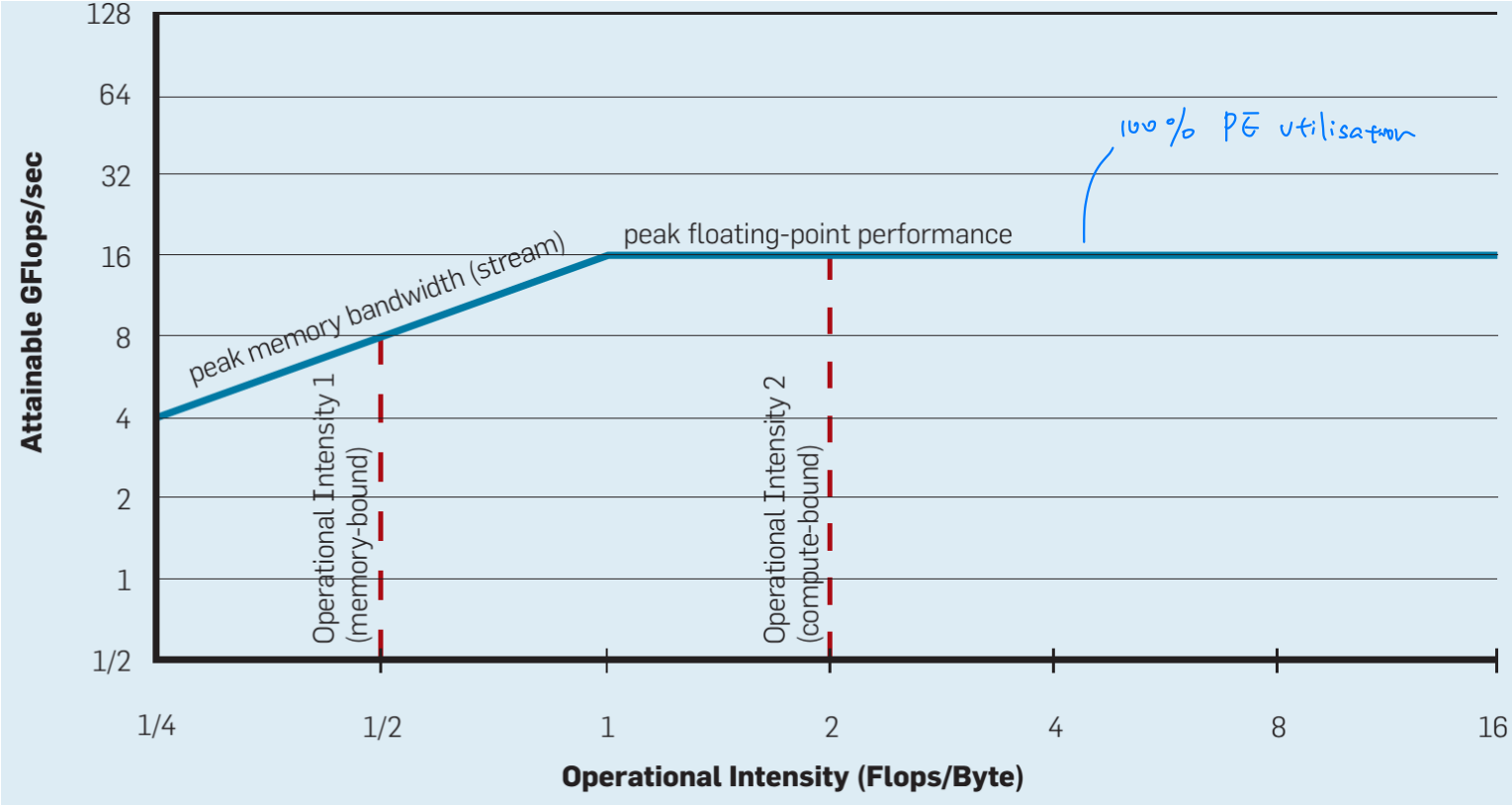


Important Concepts

Design Space: Time and Resources

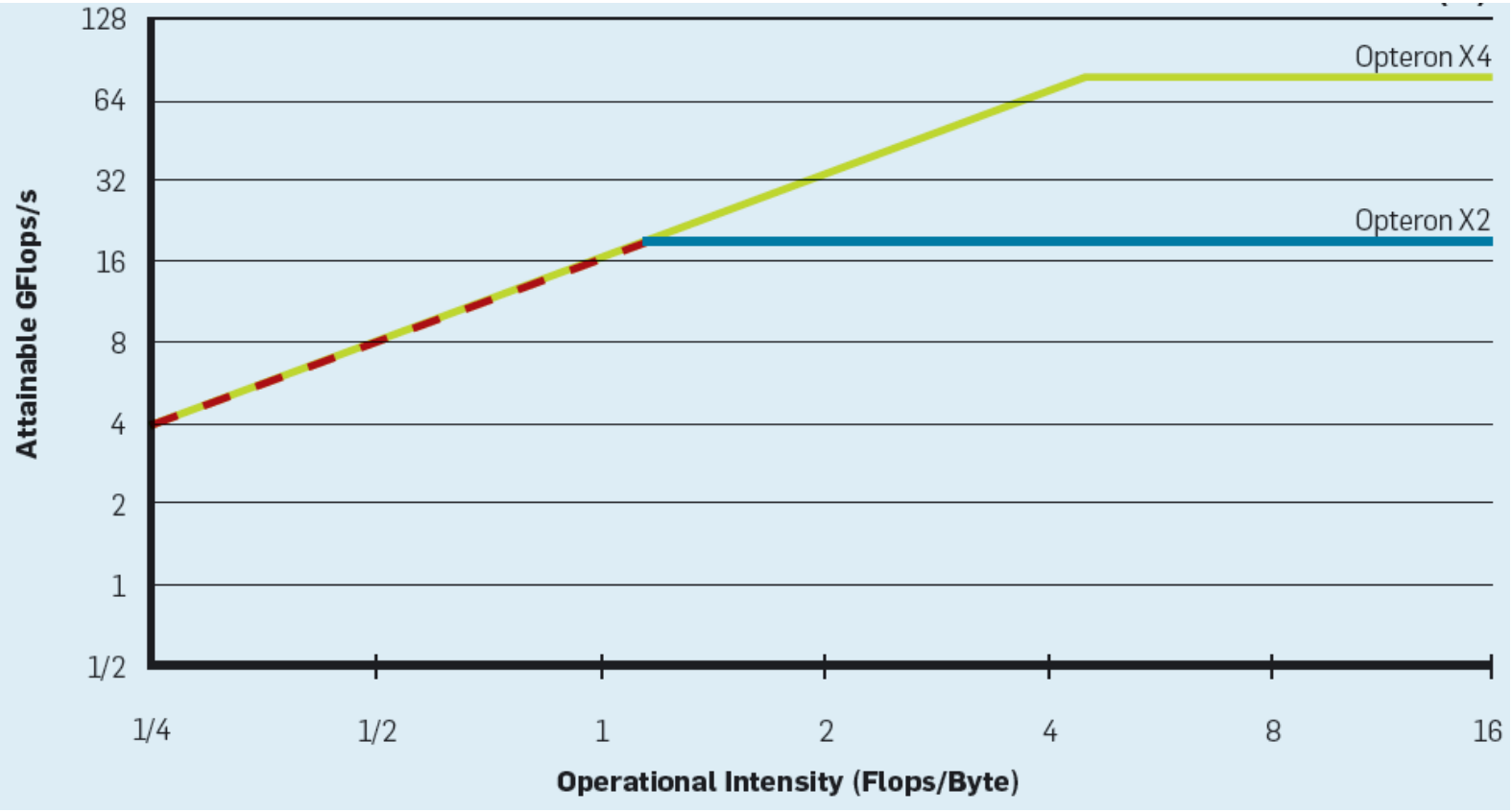


Roofline Model

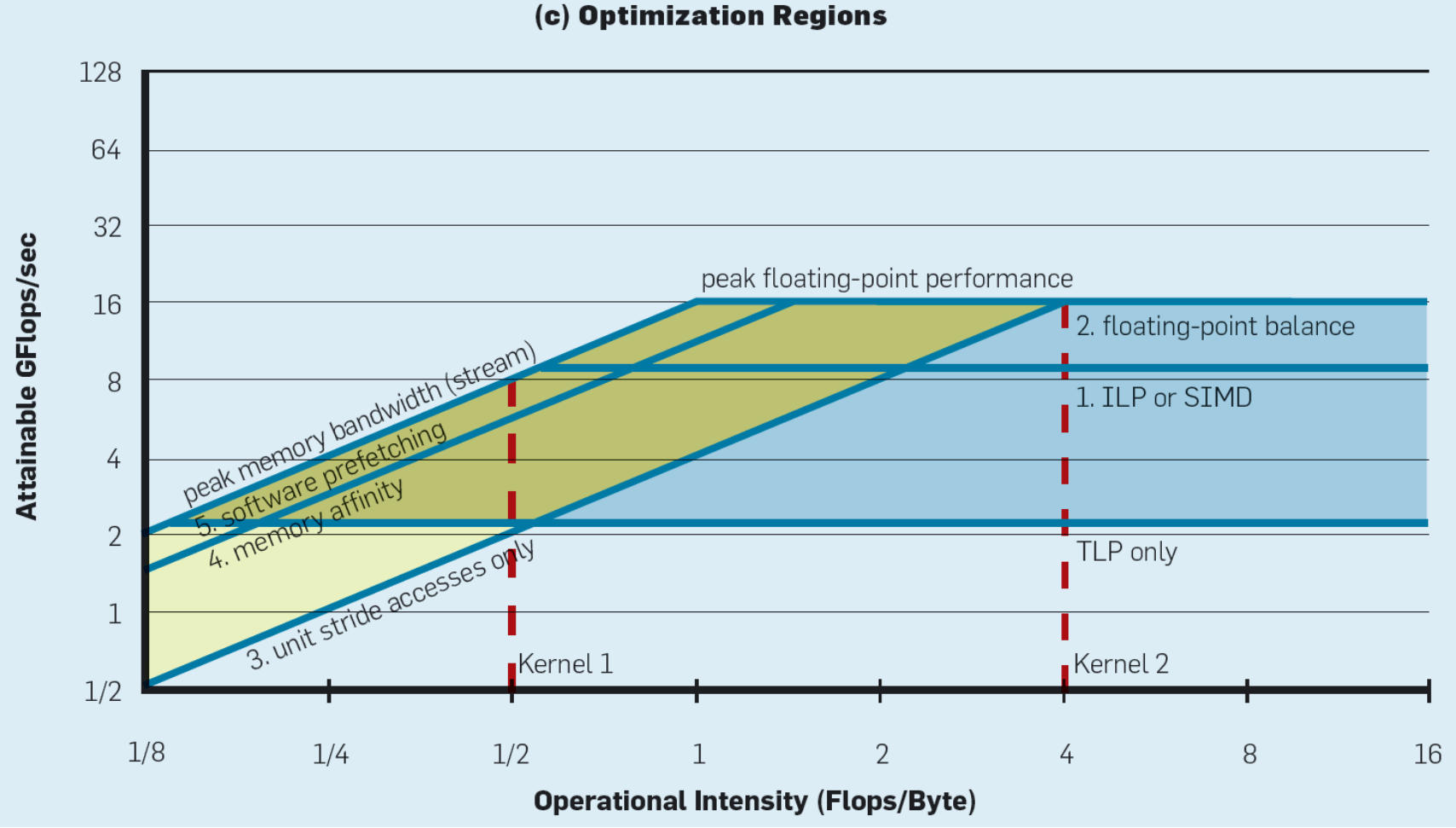


Ref: S. Williams, A. Waterman, and D. Patterson, "Roofline: An Insightful Visual Performance Model for Multicore Architectures," Commun. ACM, April 2009.

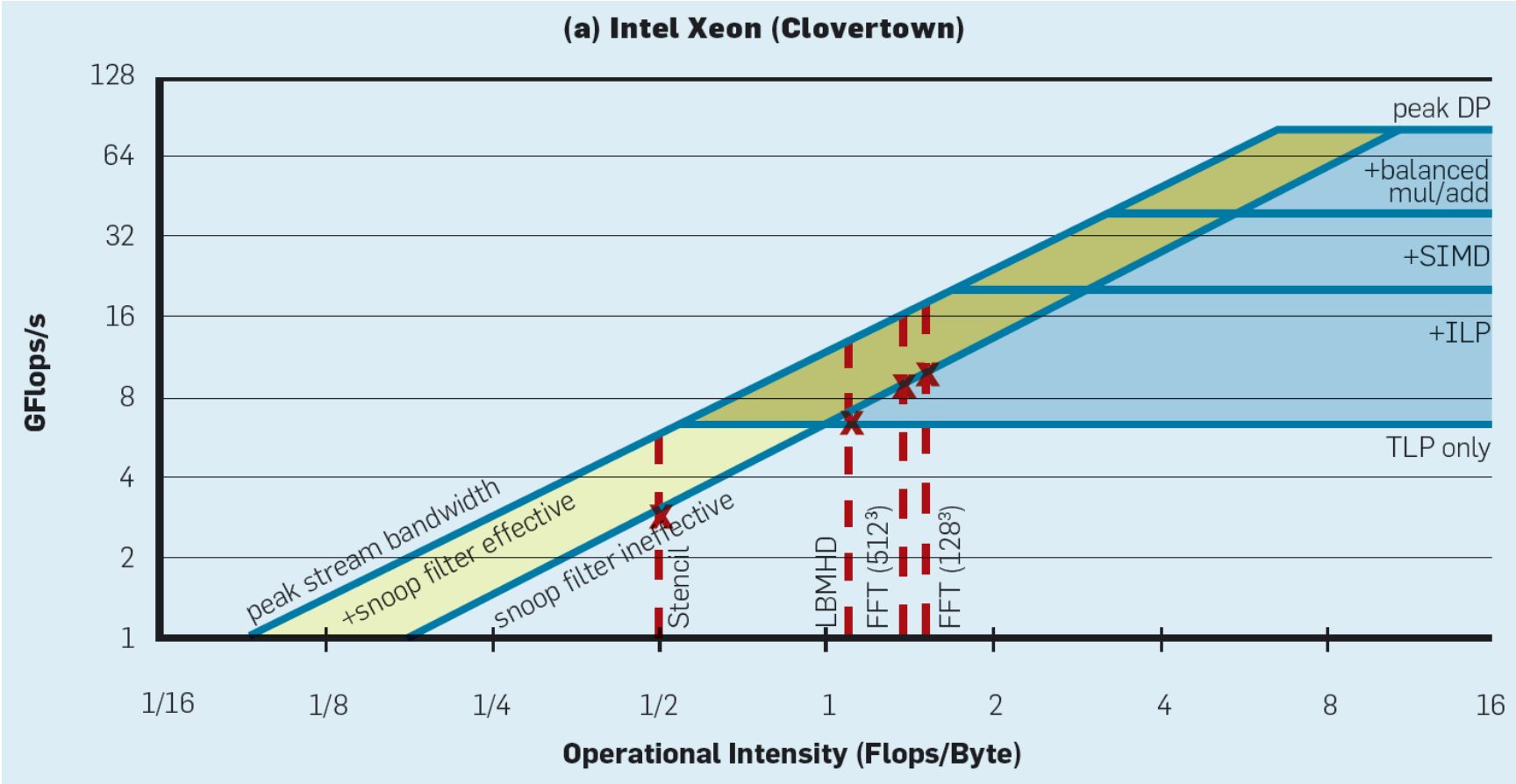
Roofline Model



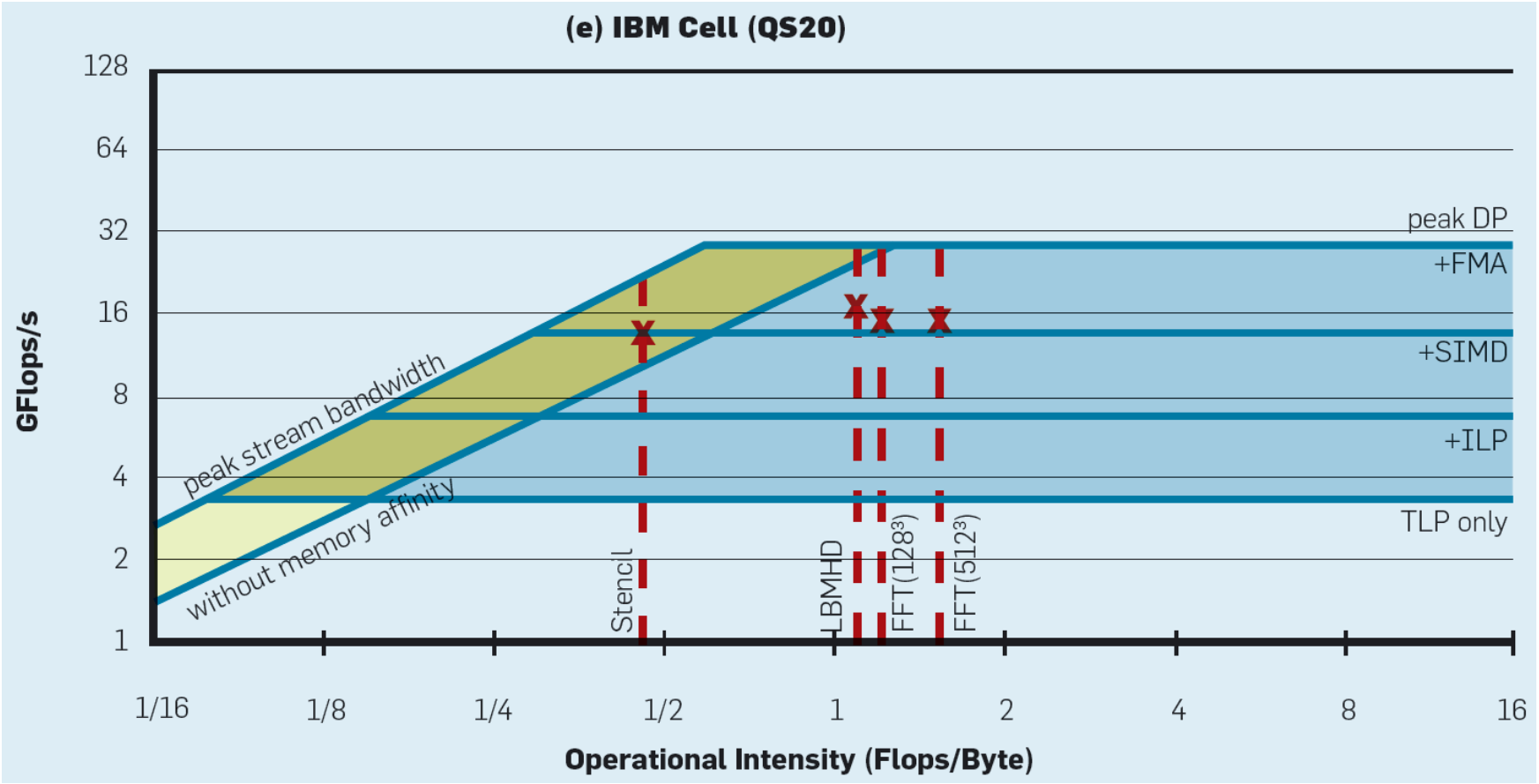
Roofline Model



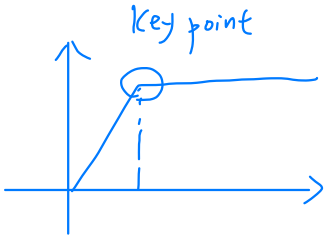
Roofline Model

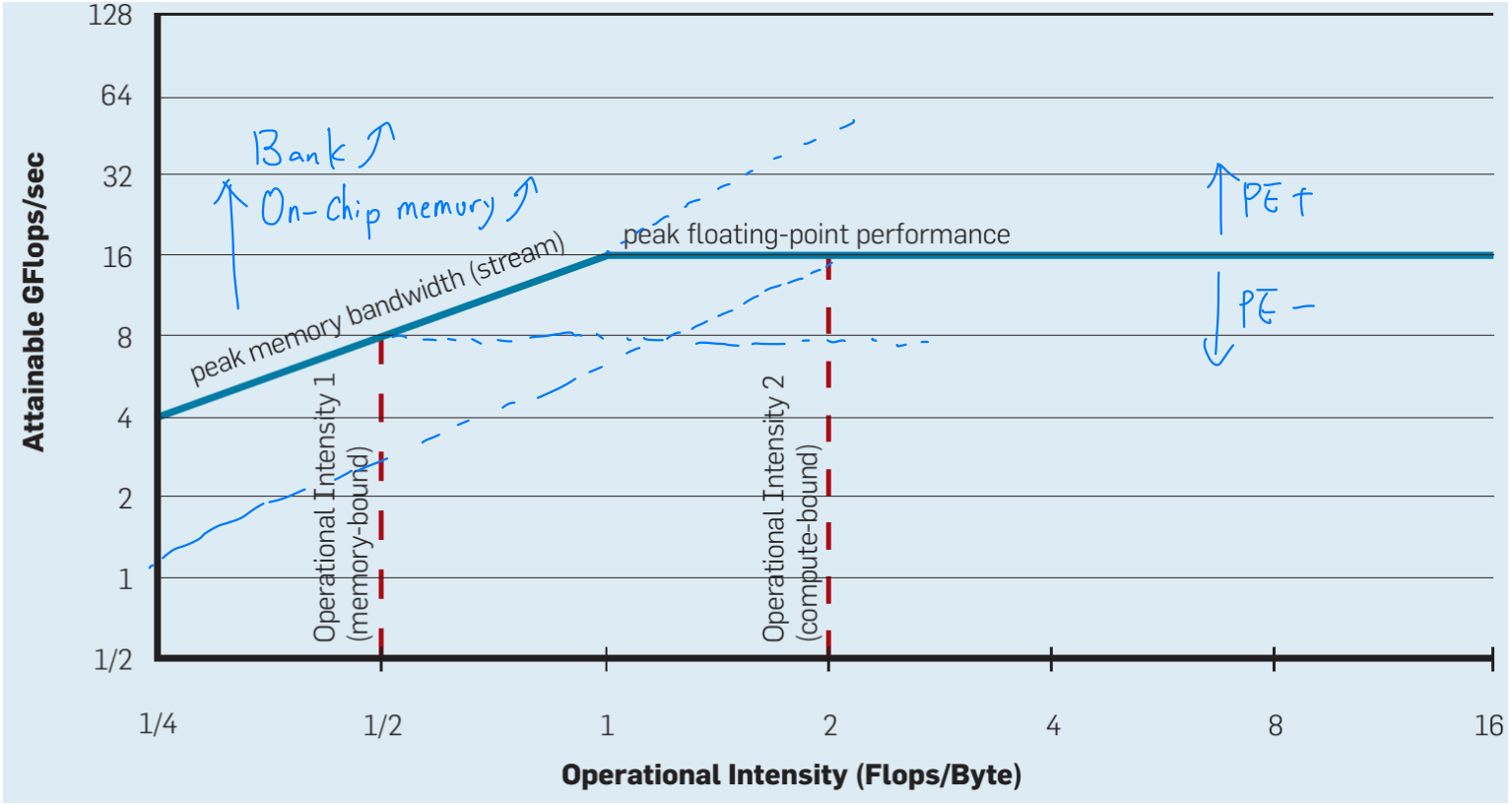


Roofline Model



Roofline Model for Accelerators?





Ref: S. Williams, A. Waterman, and D. Patterson, "Roofline: An Insightful Visual Performance Model for Multicore Architectures," Commun. ACM, April 2009.