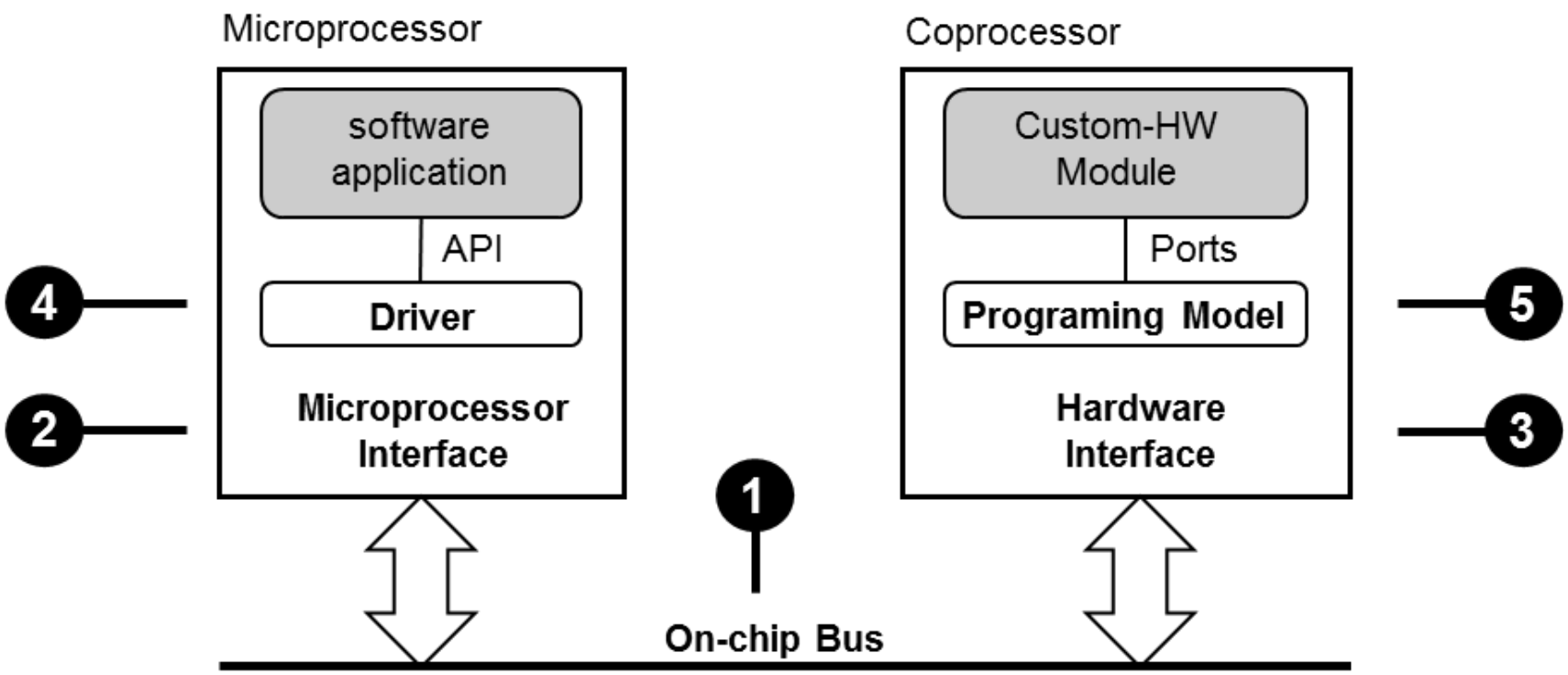
System-on-Chip Design
Microprocessor Interfaces

Hao Zheng
Comp Sci & Eng
U of South Florida

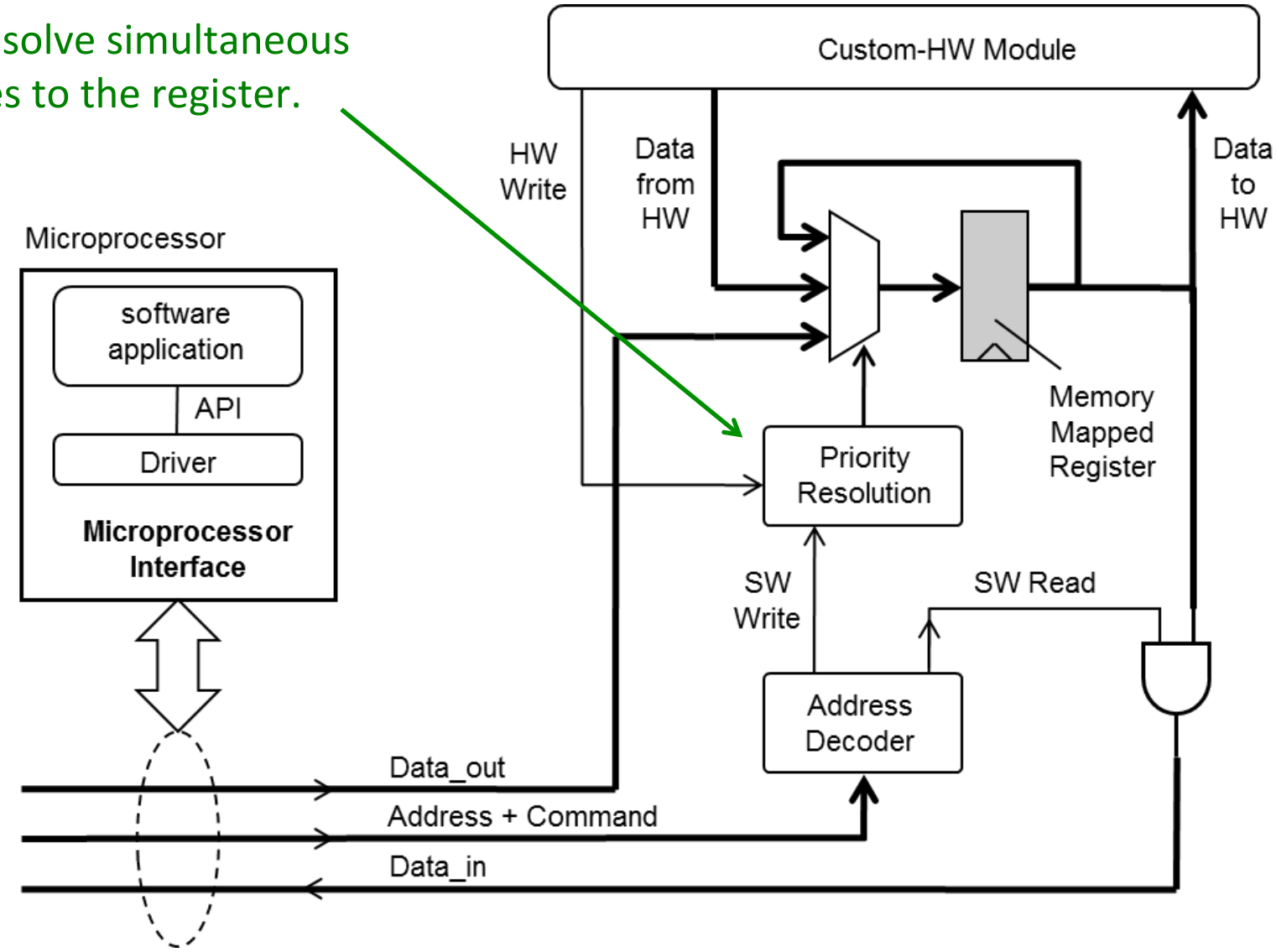
Basic Elements of HW/SW Interfaces



1. On-chip communication fabrics, ex. buses

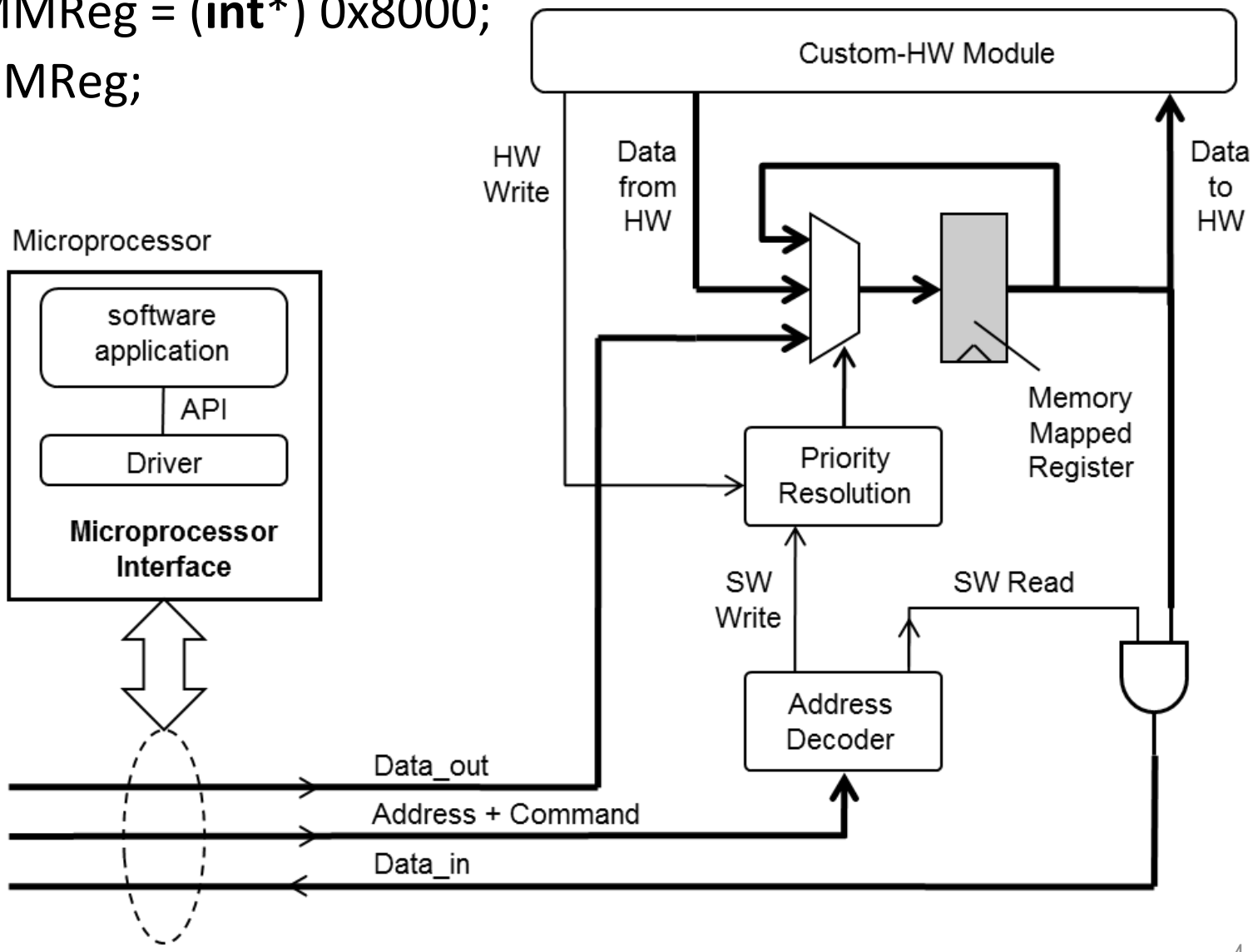
Memory Mapped Interfaces

To resolve simultaneous writes to the register.

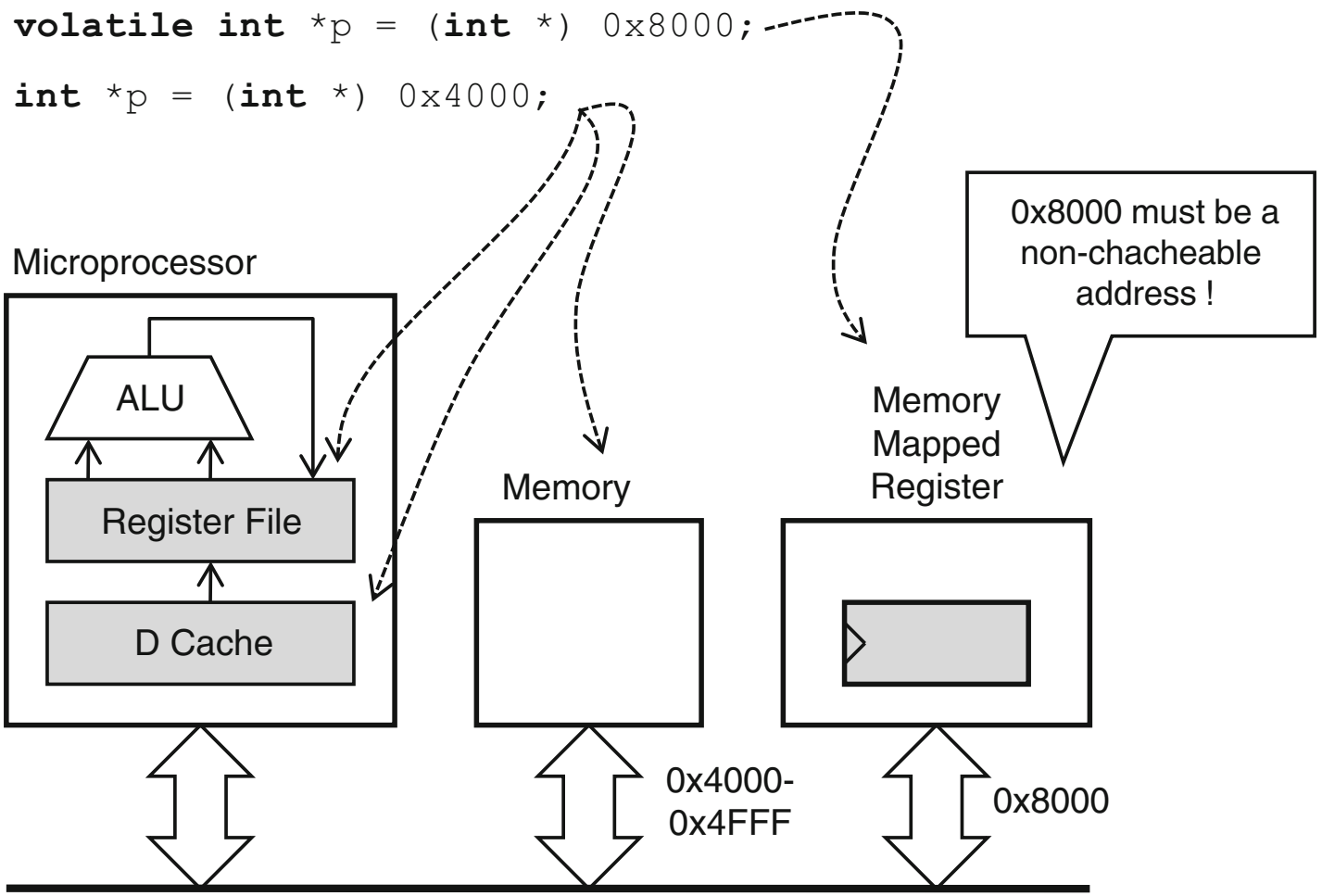


Memory Mapped Interfaces

volatile int *MMReg = (**int***) 0x8000;
int value = *MMReg;
*MMReg = 5;

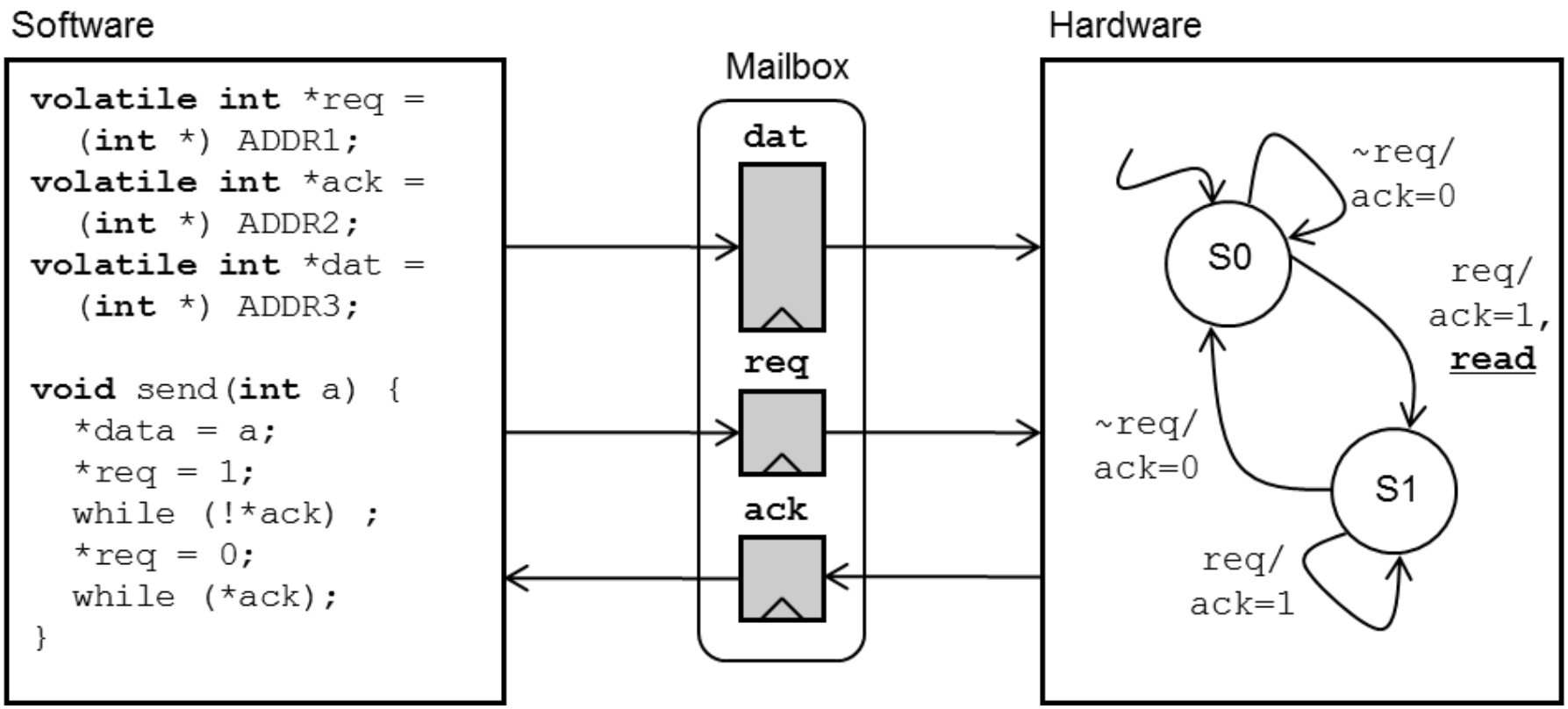


Keyword Volatile

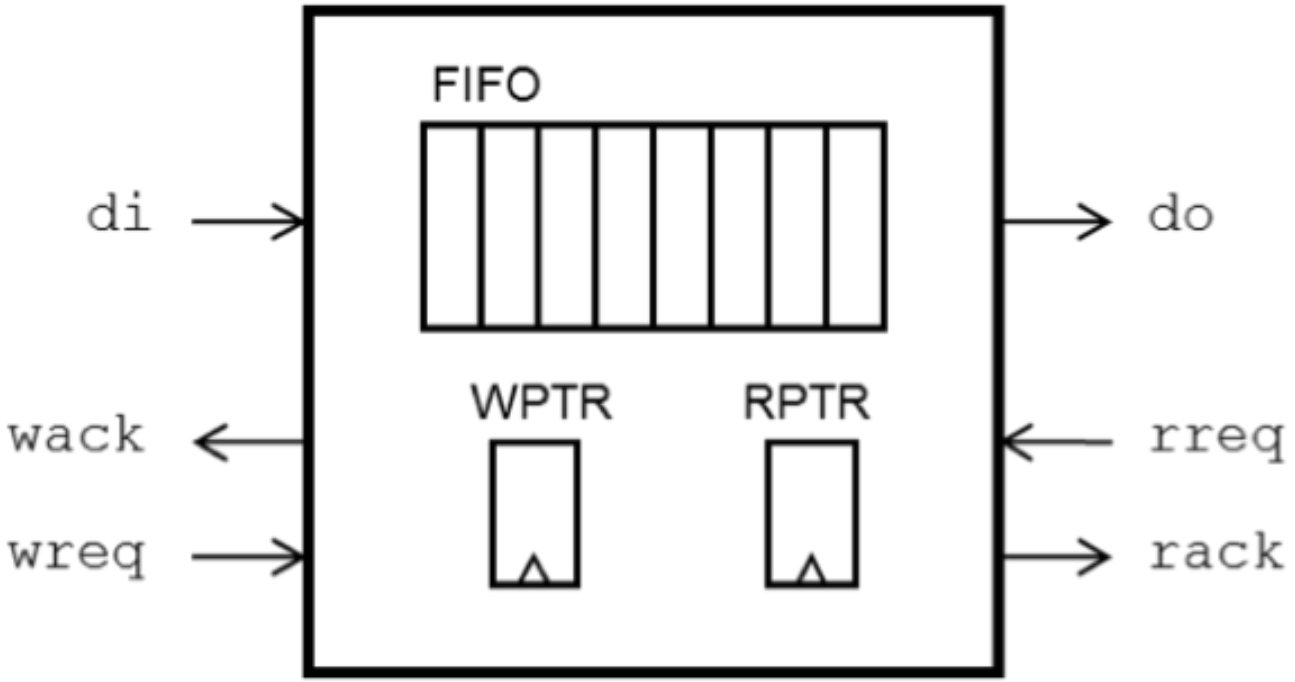


refer to Wiki for a good example of using volatile.

MailBoxes


Software
volatile int *req =
(int *) ADDR1;
volatile int *ack =
(int *) ADDR2;
volatile int *dat =
(int *) ADDR3;
void send(int a) {
*data = a;
*req = 1;
while (!*ack) ;
*req = 0;
while (*ack);
}
Mailbox
dat
req
ack
Hardware
~req/
ack=0
S0
req/
ack=1,
read
~req/
ack=0
S1
req/
ack=1

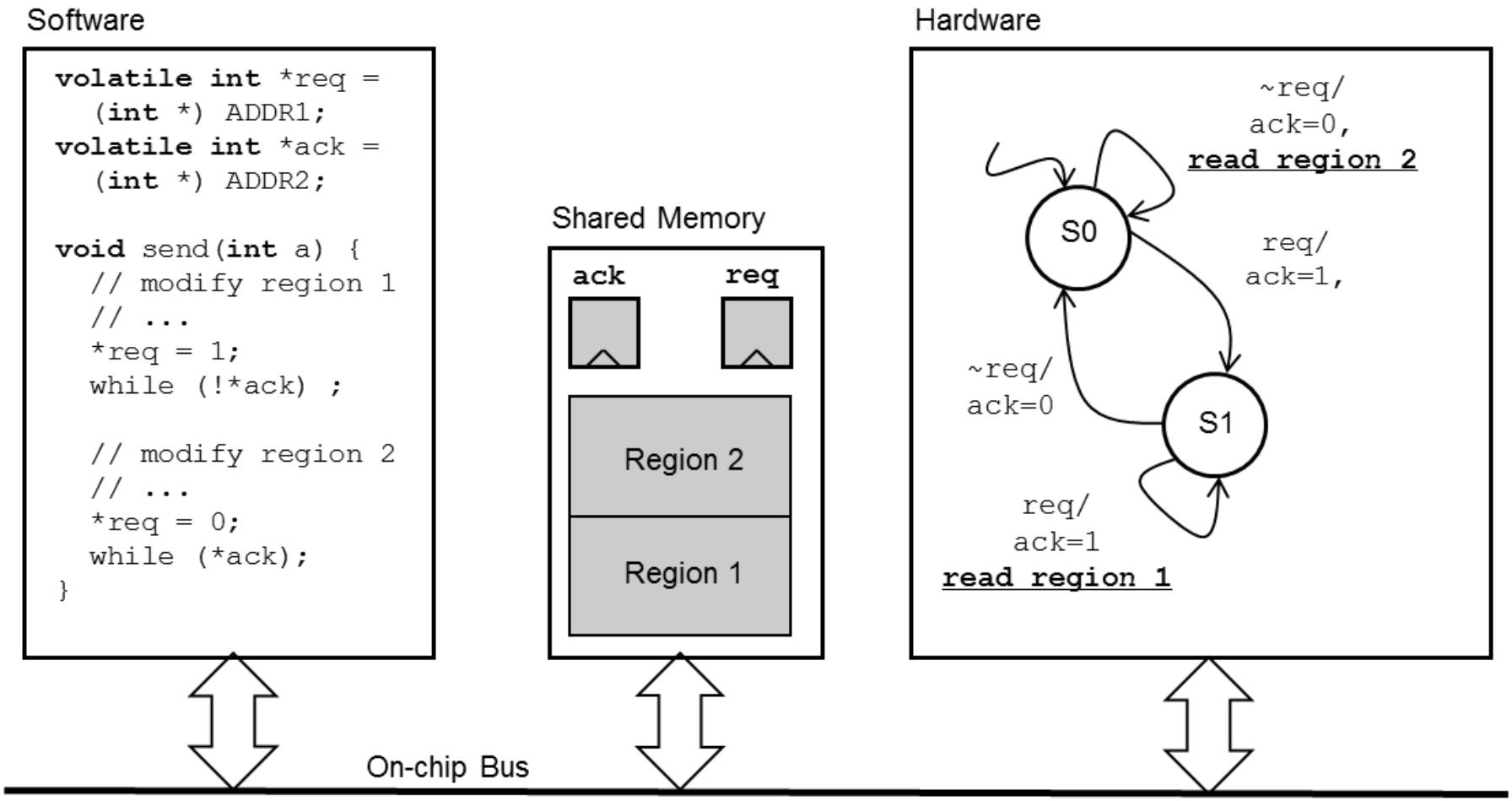
FIFOs



Allows masters to perform non-blocking operations on a slave.

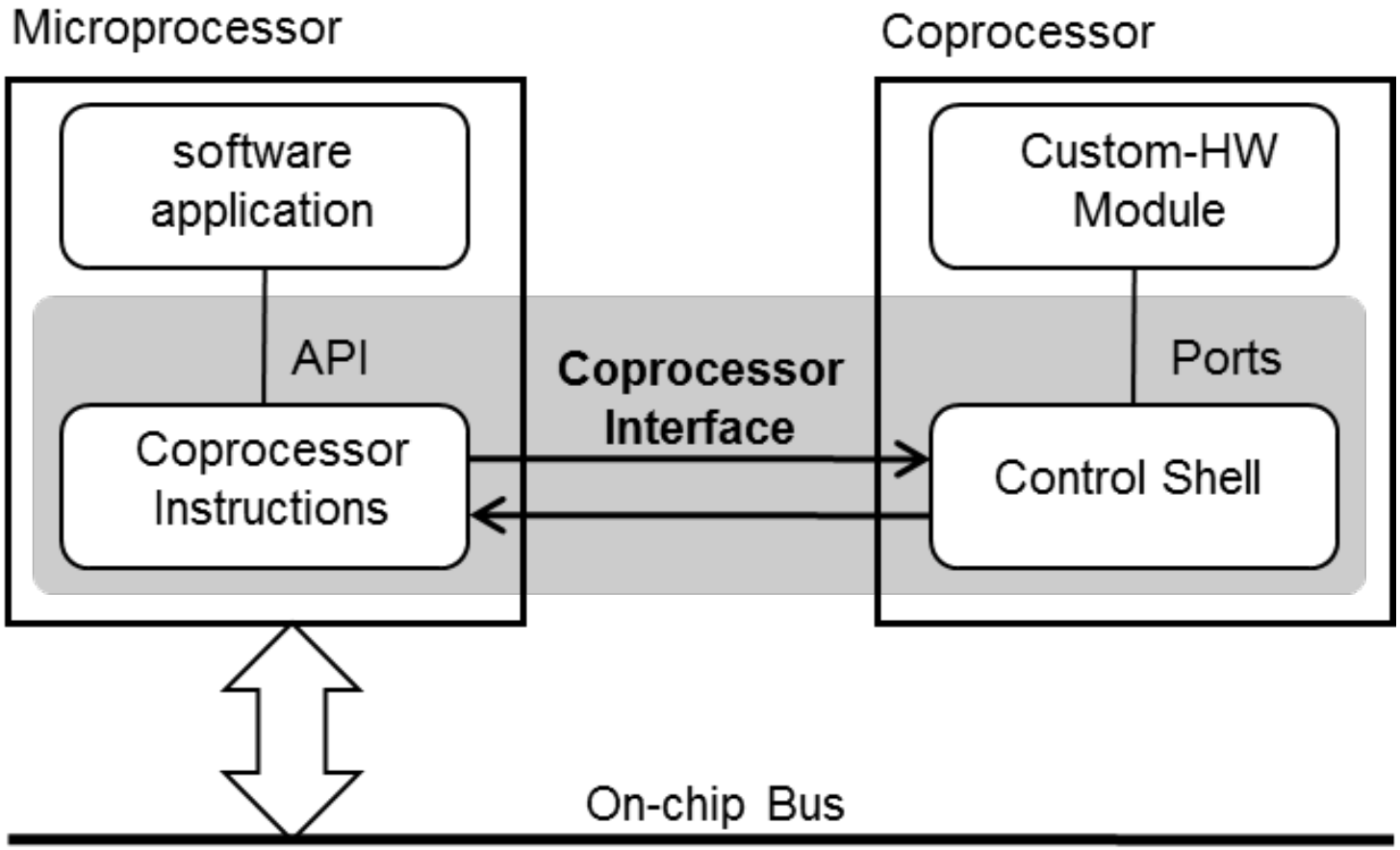
This FIFO has two slave interfaces.

Shared Memory



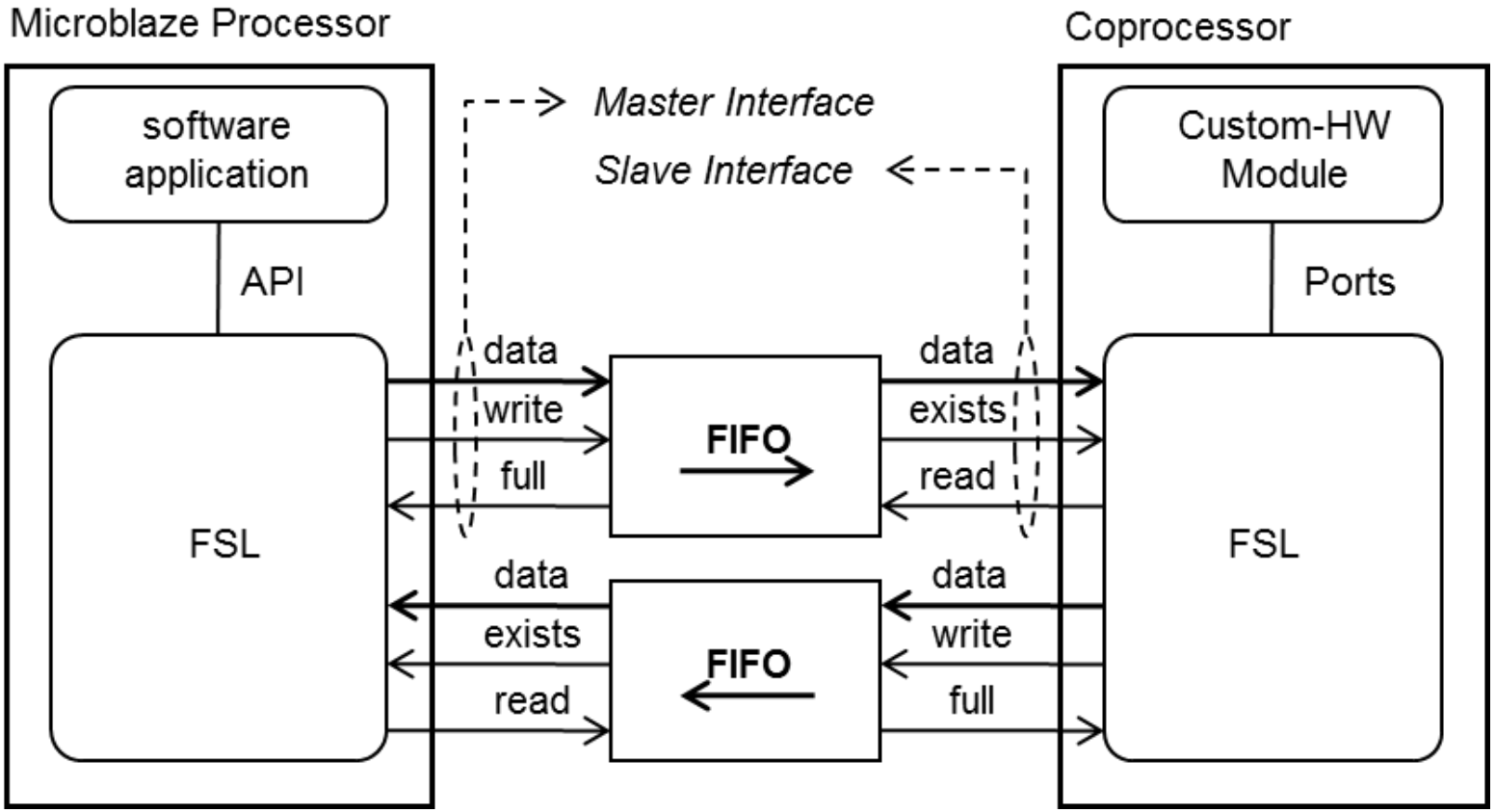
To transfer large chunks of data between SW and HW.

Coprocessor Interfaces



+ high throughput, fixed latency
− non-reusable, HW tied to a specific CPU.

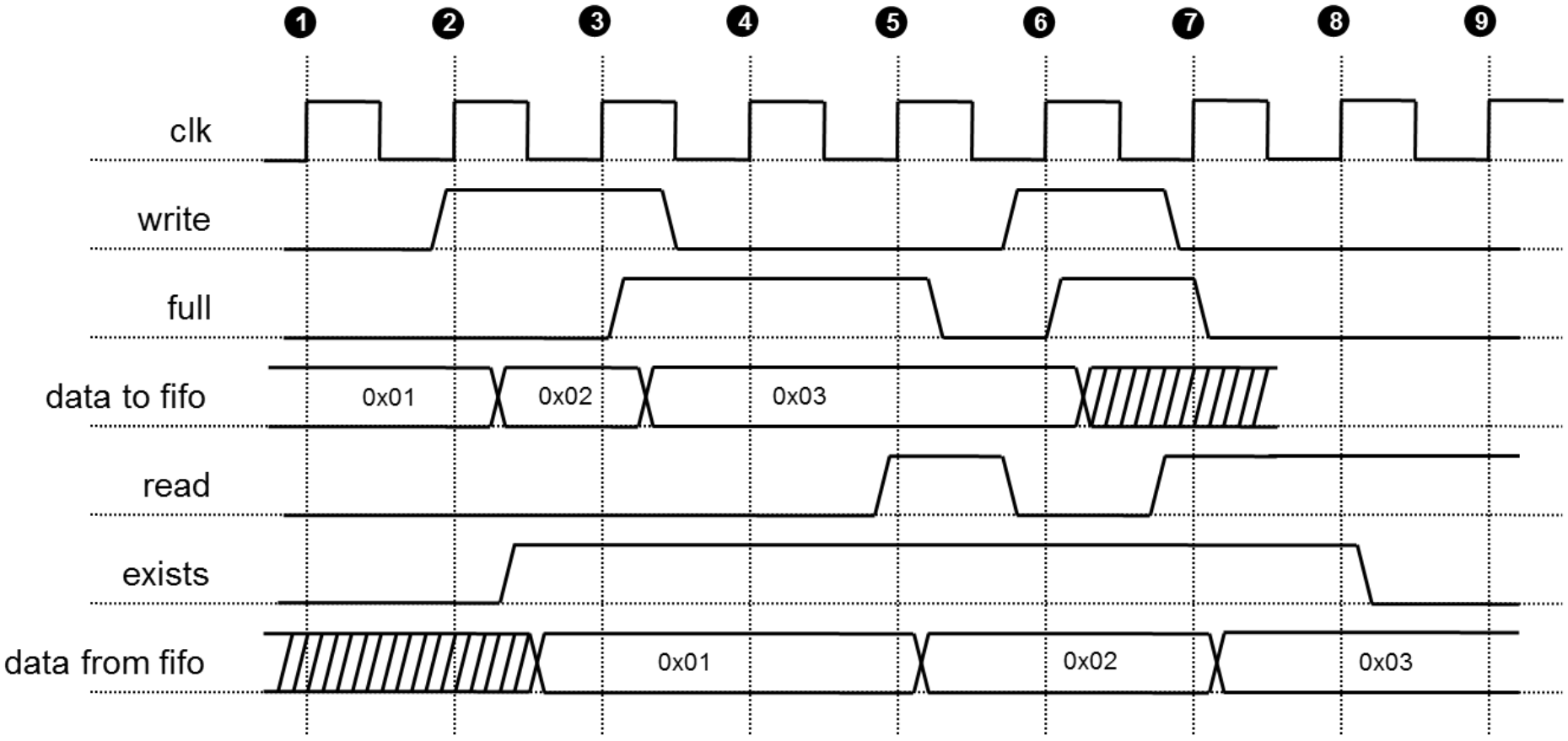
Coprocessor Interfaces: An Example



put rD, FLSx // copy register rD to FSL interface FSLx

get rd, FSLx // copy FSL interface FSLx into register rD

Coprocessor Interfaces: An Example



Assume that the size of FIFO is 2.

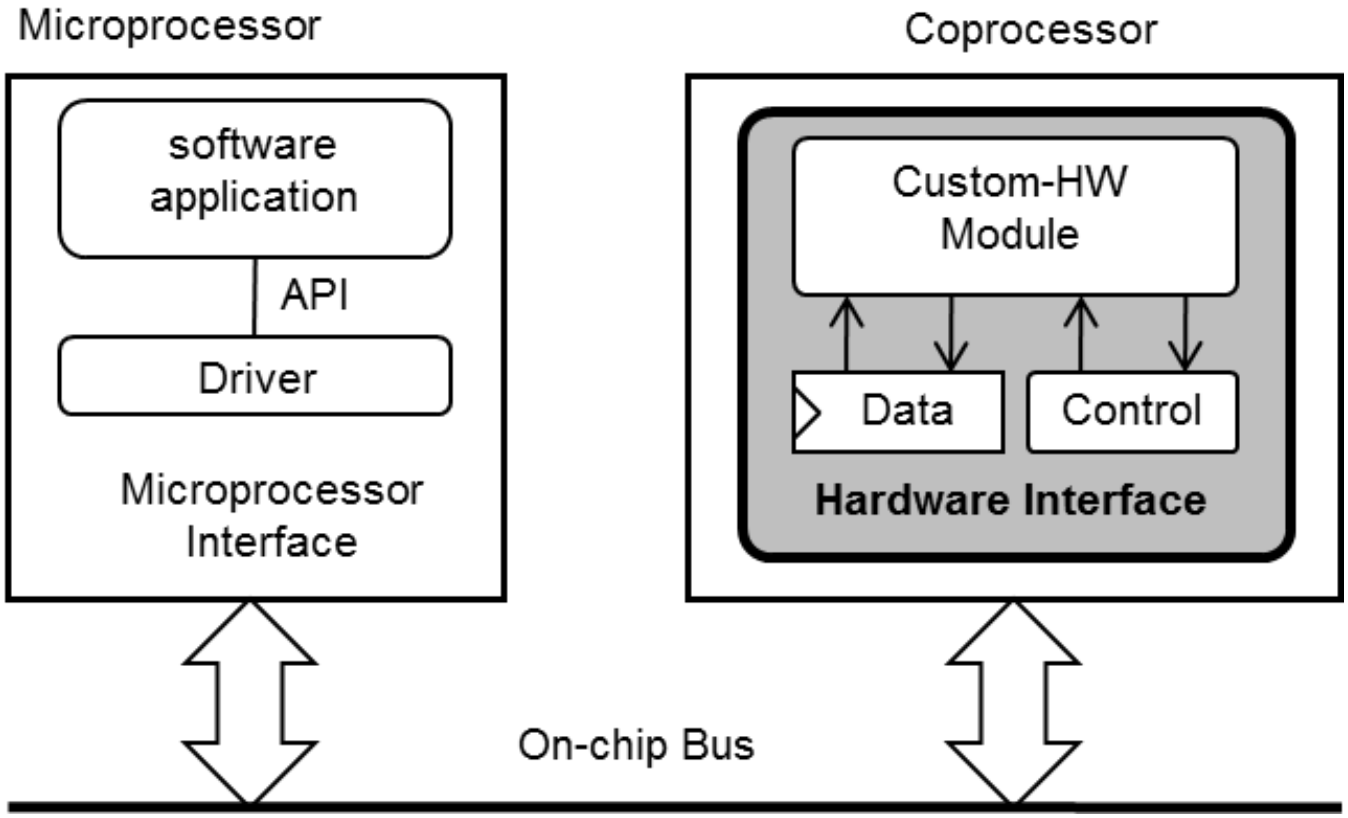
Reading Guide

- Section 11.1 – 11.2, the *CoDesign* book.
  - Skip 11.1.6, 11.2.2

System-on-Chip Design
Hardware Interfaces

Hao Zheng
Comp Sci & Eng
U of South Florida

Hardware Interface



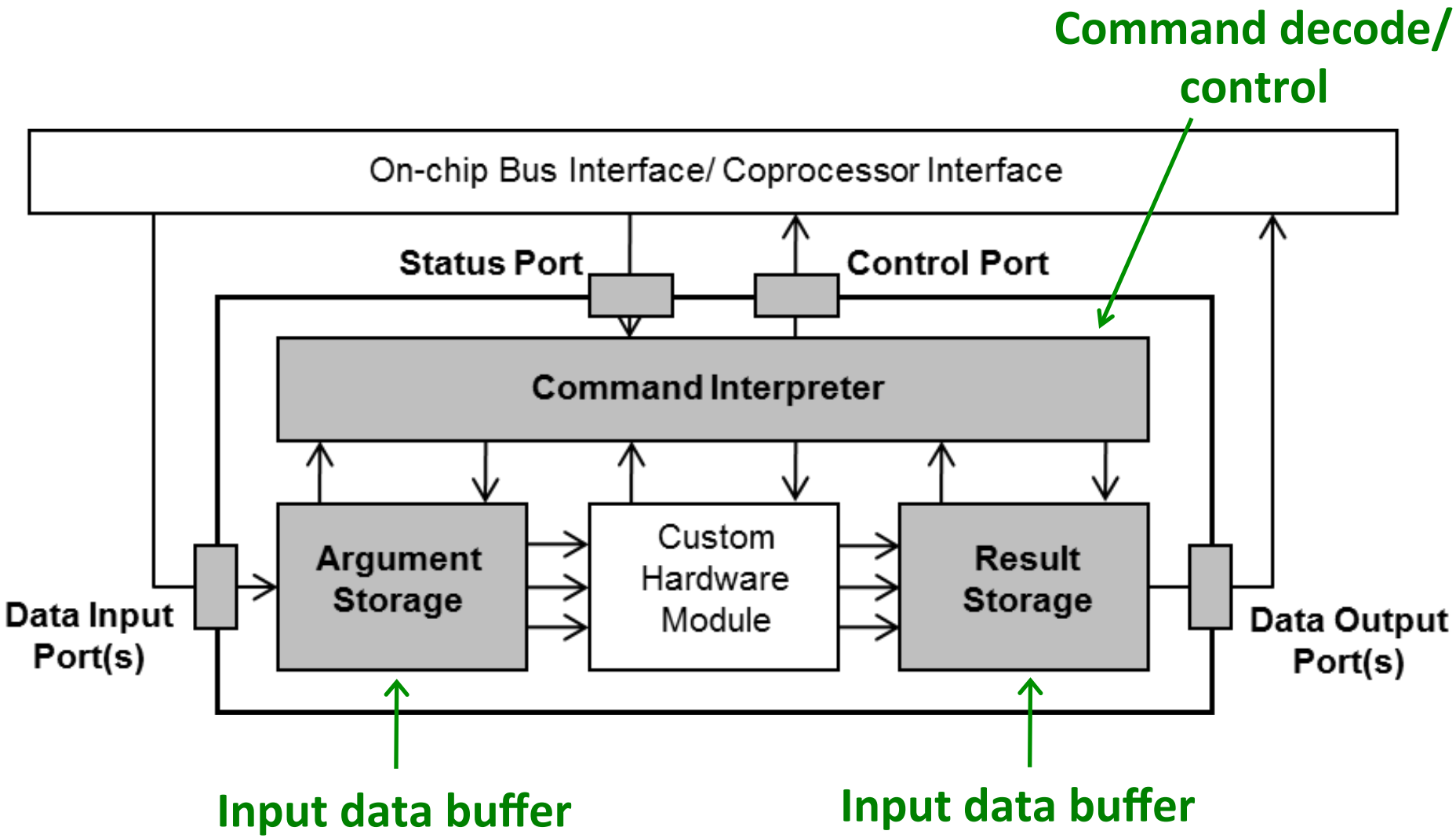
HW interface control comm and comp of HW:

data transfer, **word length conversion**, **local storage**, **instruction set**, **local control**.

Hardware Interface Functions

- **Data transfer** between CPU and HW using a communication protocol.
- **Word length** conversion between the interface and the HW internal format.
- **Local storage** for buffering data.
- **Instruction set** provides a programming interface to SW.
- **Local control** of the HW in response to SW command.

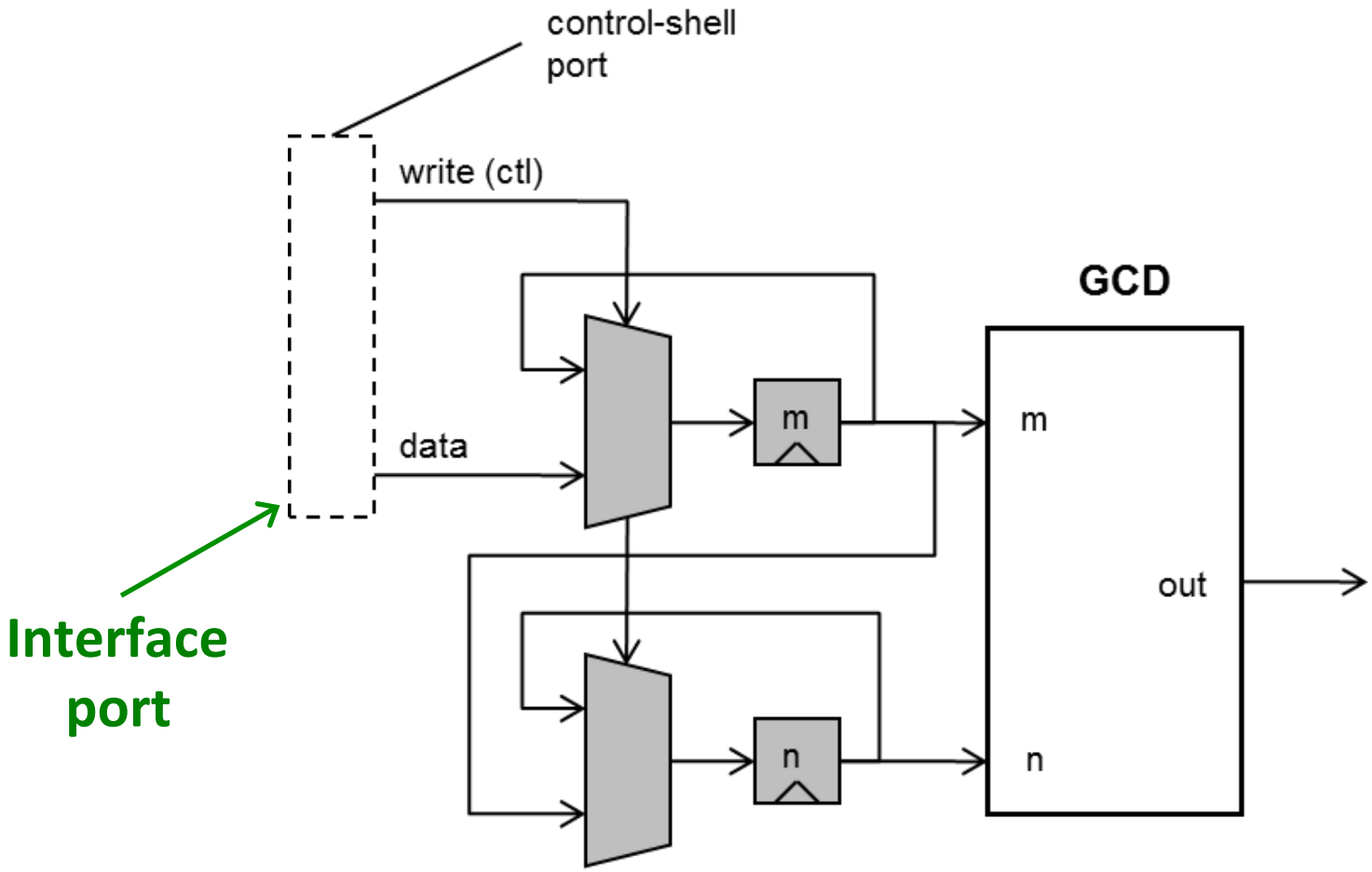
Generic Structure of HW Interfaces



Data Design

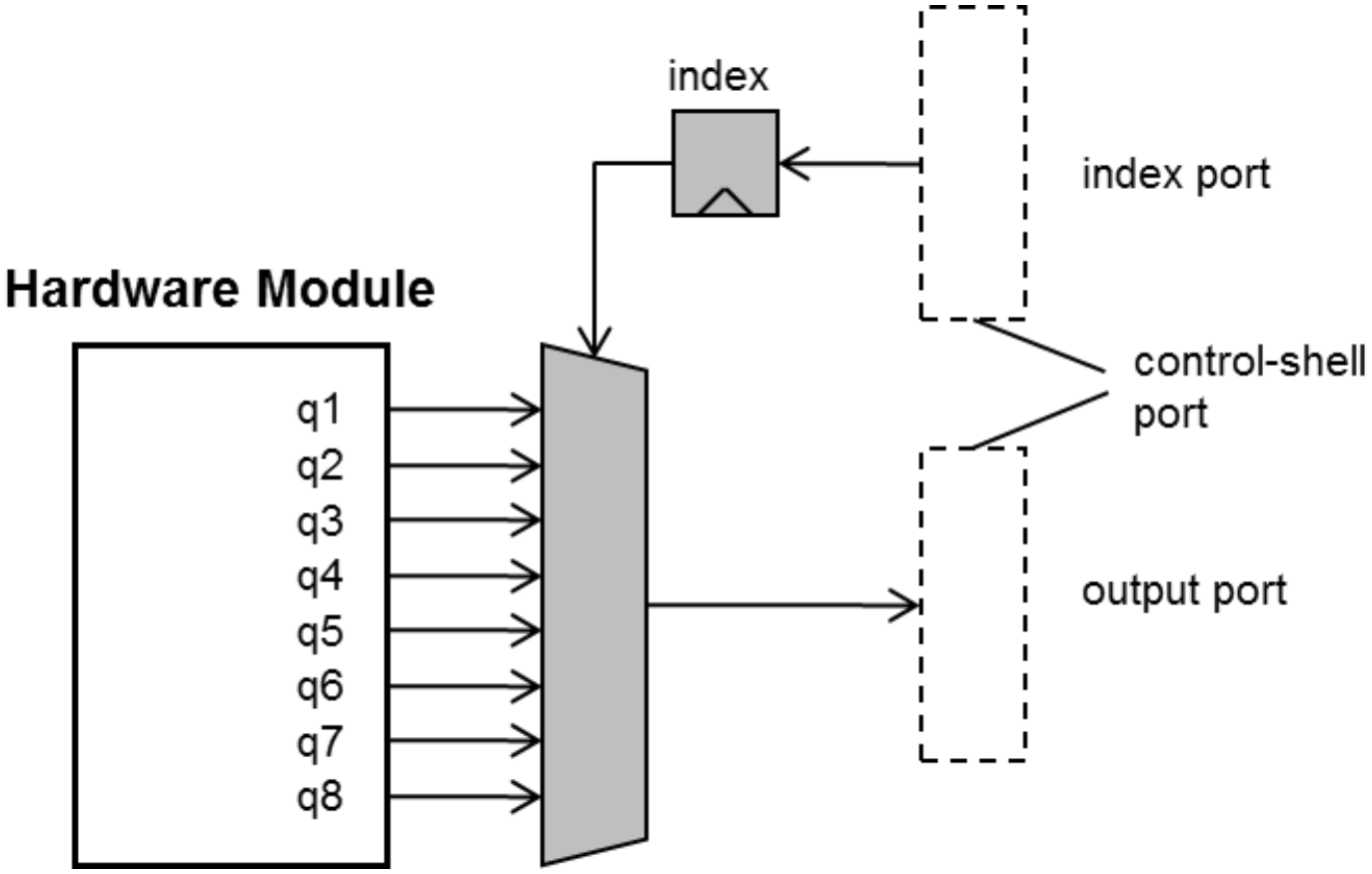
- Concerns passing data from interface to ports of HW module.
  - Three factors: word length, direction, update rate.
- Example: **int** gcd(**int** m, **int** n); *// 3 HW ports*
- Ideally, interface supports three ports:
  - Two input ports: *m*, *n*
  - One output port: *out*.
  - Implemented with three memory-mapped registers.
- But interface does not support all required ports.
  - Or, HW ports used infrequently, making a dedicated interface port less efficient.
- Solution: port multiplexing

Time Multiplexing



SW writes *n* and *m* in sequence to the HW interface.

Index Register

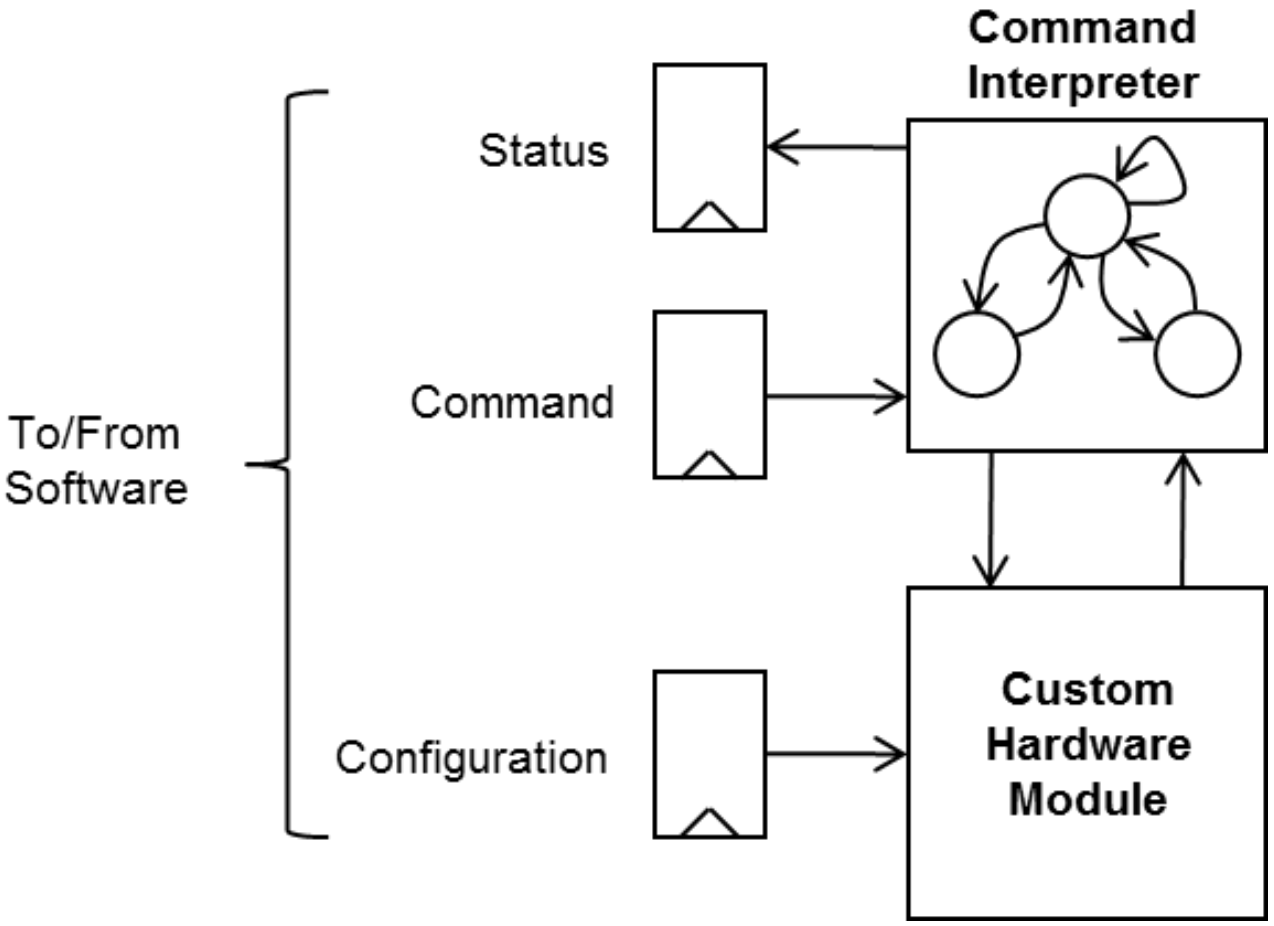


HW interface selects outputs from HW modules to output port by controlling the index port.

Port Multiplexing

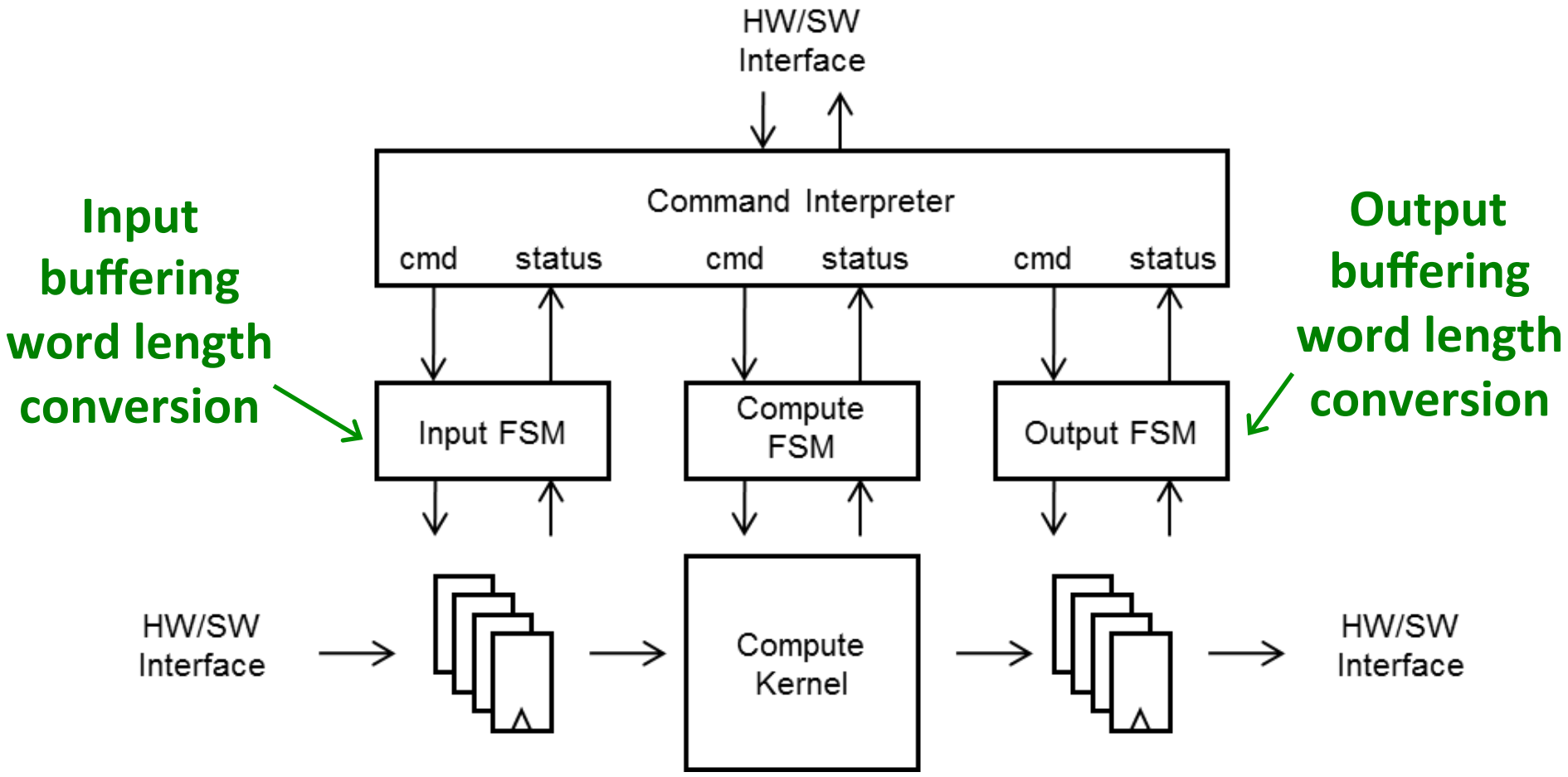
- Mapping HW ports to limited interface ports.
- Interface control becomes more complicated.
- Useful to word length conversion.
  - HW port is 128 bit wide, but interface port is 32 bit wide. How the conversion be done?
- Additional bit mask register is used to map individual bits from interface to a HW port.

Control Design



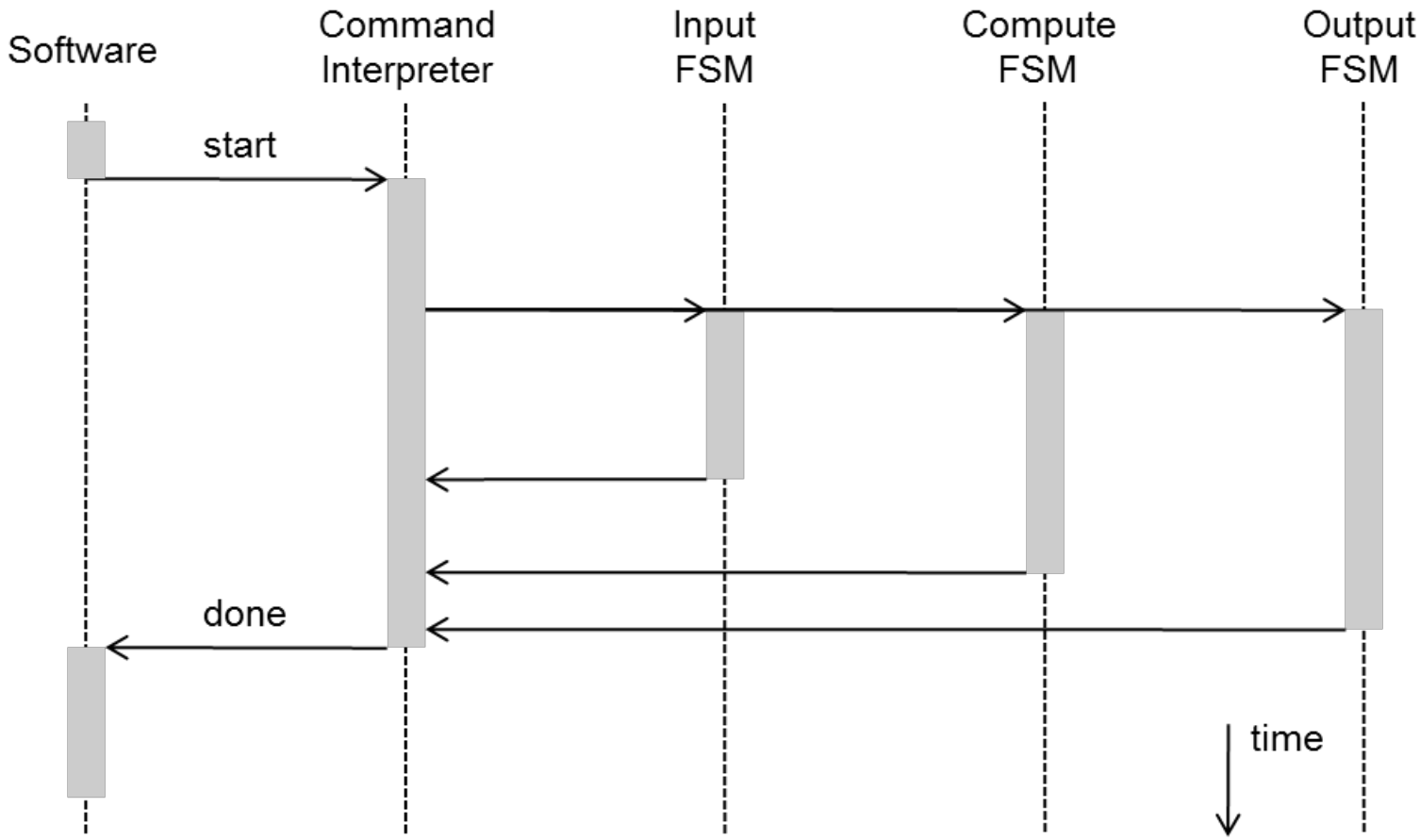
1. Interpret commands to generate control signals.
2. Capture status signals

Hierarchical Control



A command from SW is converted to a sequence of micro-commands for FSMs

Execution Flow



Can be pipelined to improve performance.

Programmer's Model

- **Address map** – organization of SW accessible storage, memory or HW registers
  - SW views a single register for a memory address.
  - HW may have separate registers for write/read.
- **Instruction Set**
  - Need to consider trade-off between flexibility for SW and efficiency for HW.
  - Should include commands for SW/HW synchronization and HW initialization.

Reading Guide

- Read Section 12.1 – 12.3, the *CoDesign* book.
  - Skip 12.3.2
- Skim section 12.4.