

18 I/O and Peripherals

Part 1: Bus Structures

CSC 230

M&H: 8.1, 8.2, 8.5, 5.1 to 5.4

Stallings: 3.4 to 3.8

What is a bus?



**Physical structure
(wires)**

- *Address*
- *Data*
- *Control*
- *Power*

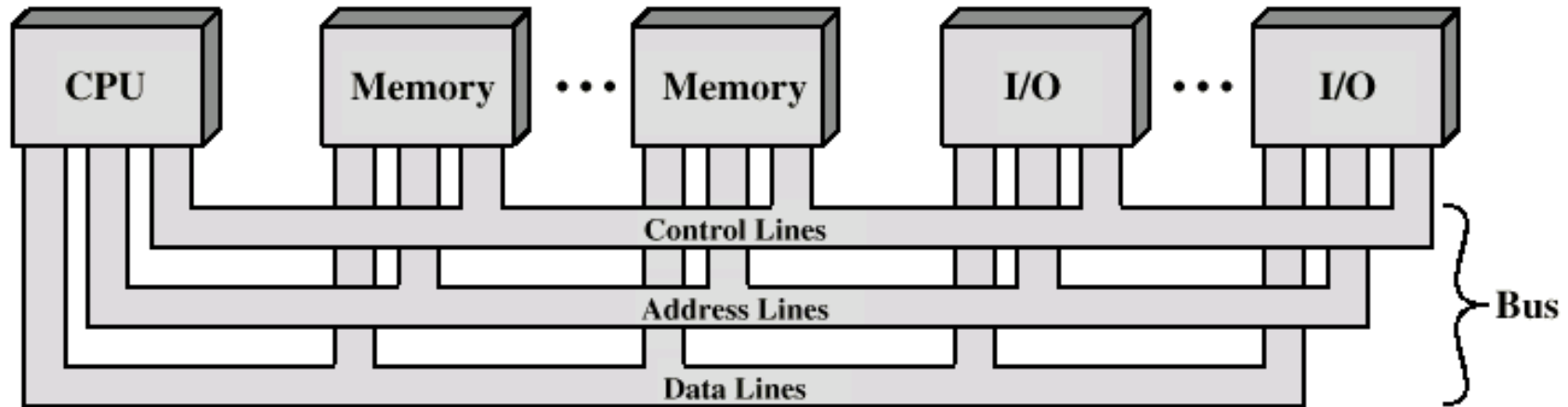
Protocol

*Synchronous
(e.g. memory)*

*Asynchronous
(e.g. peripherals)*

*Many devices
attached
→ Only one can
be Master at
any point in
time*

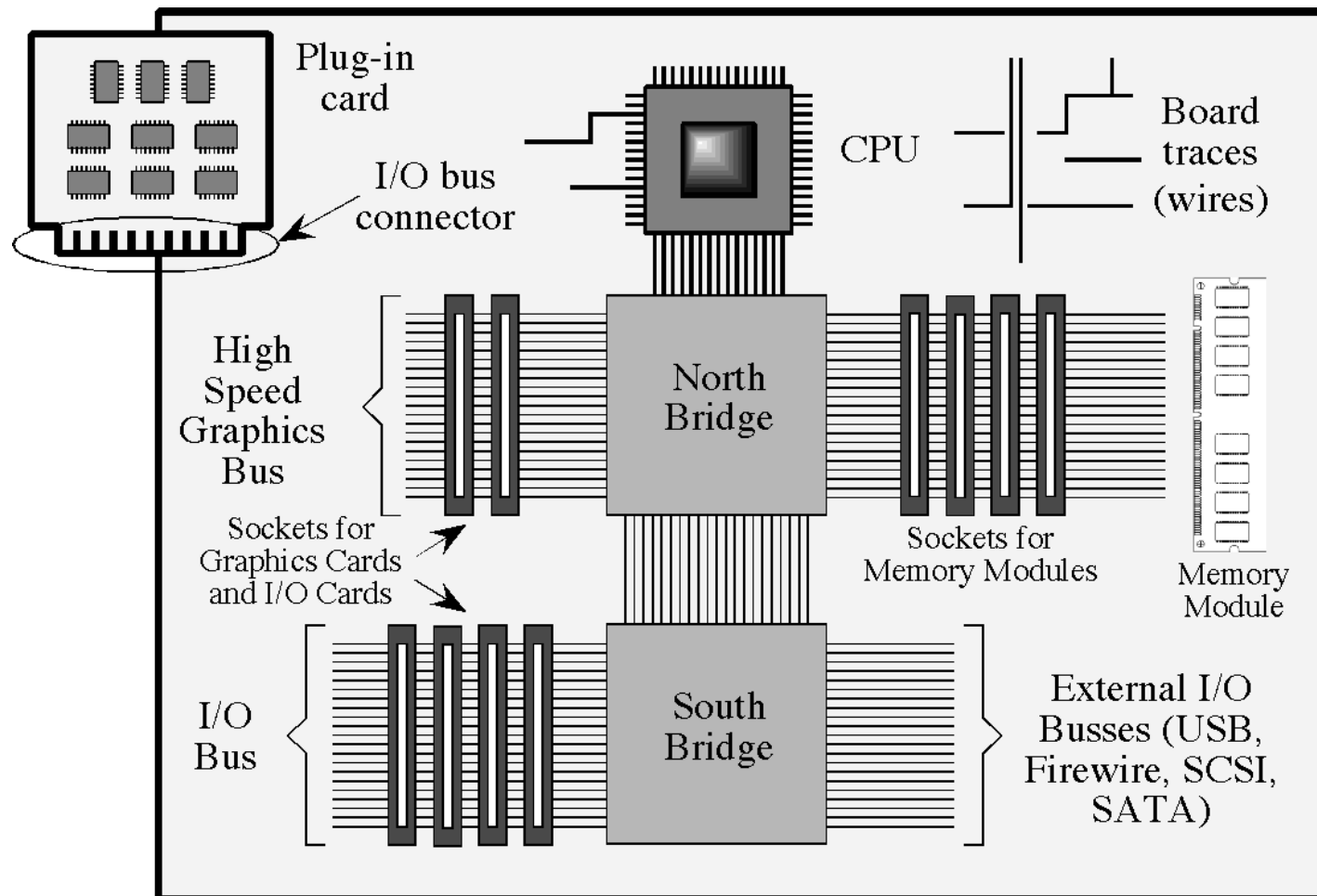
Bus Concepts and Examples



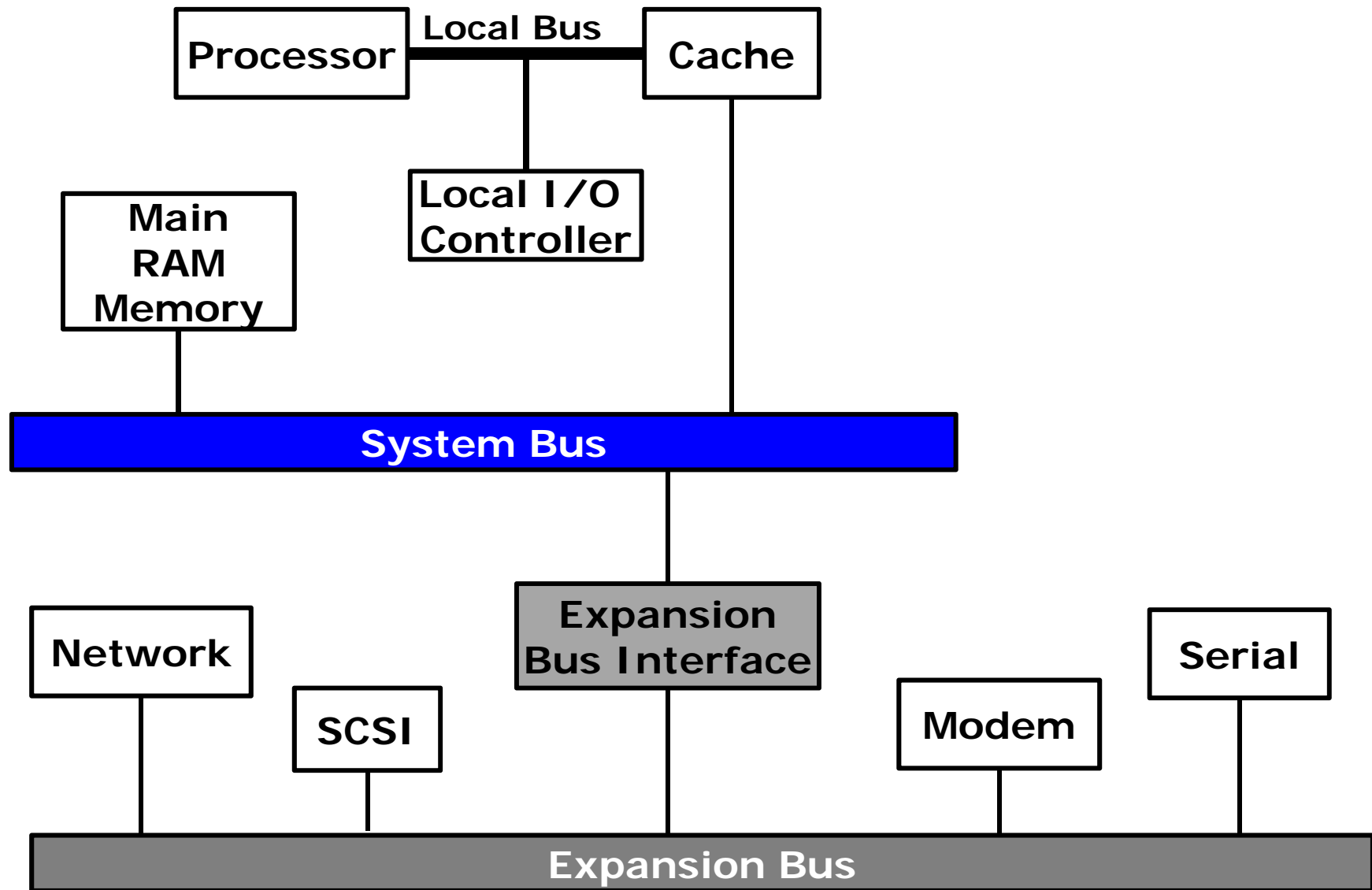
- ☐ Parallel lines on circuit boards
- ☐ Ribbon cables
- ☐ Strip connectors on mother boards
- ☐ Sets of wires

Simple Bus Architecture

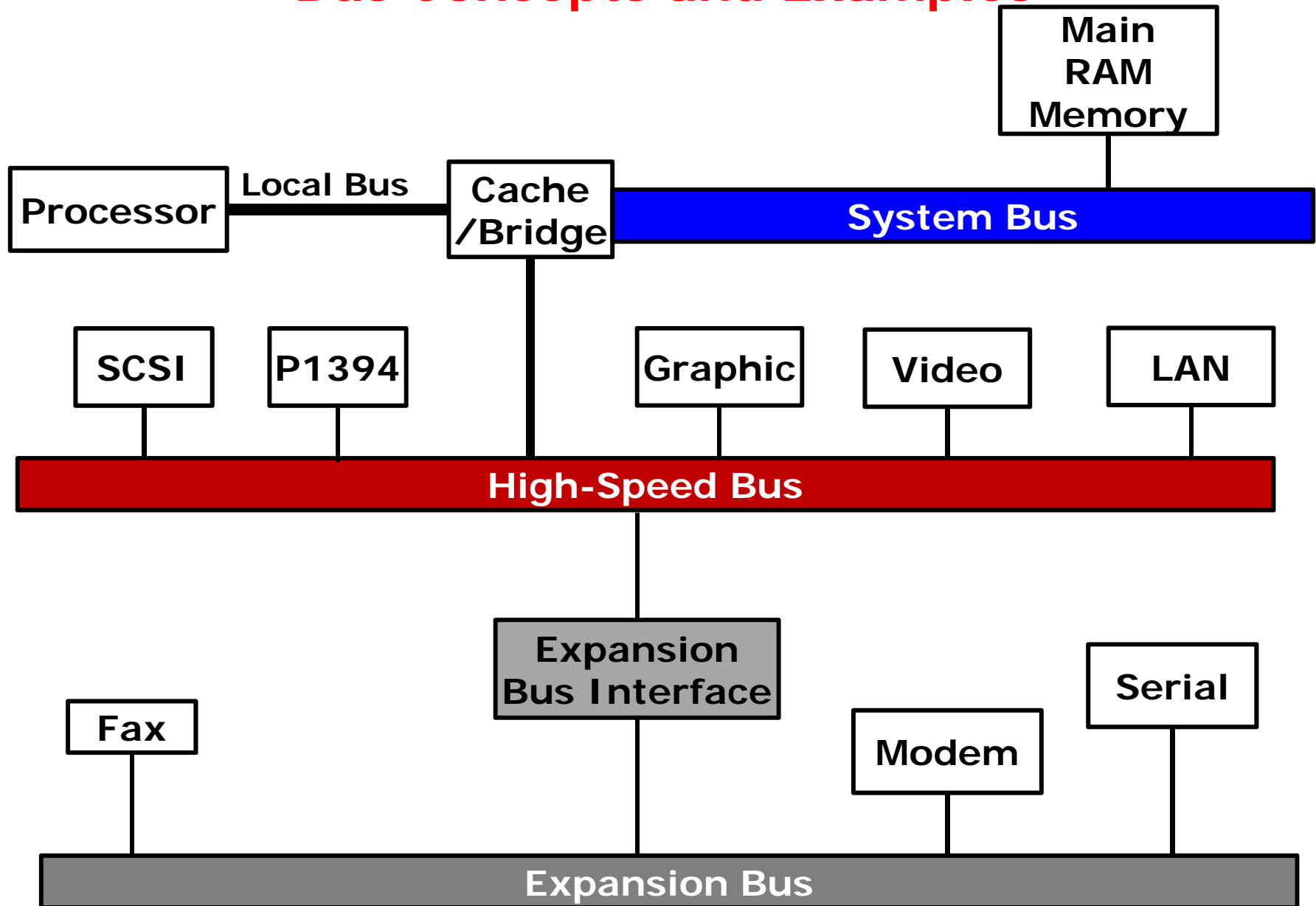
A simplified motherboard of a personal computer (top view)



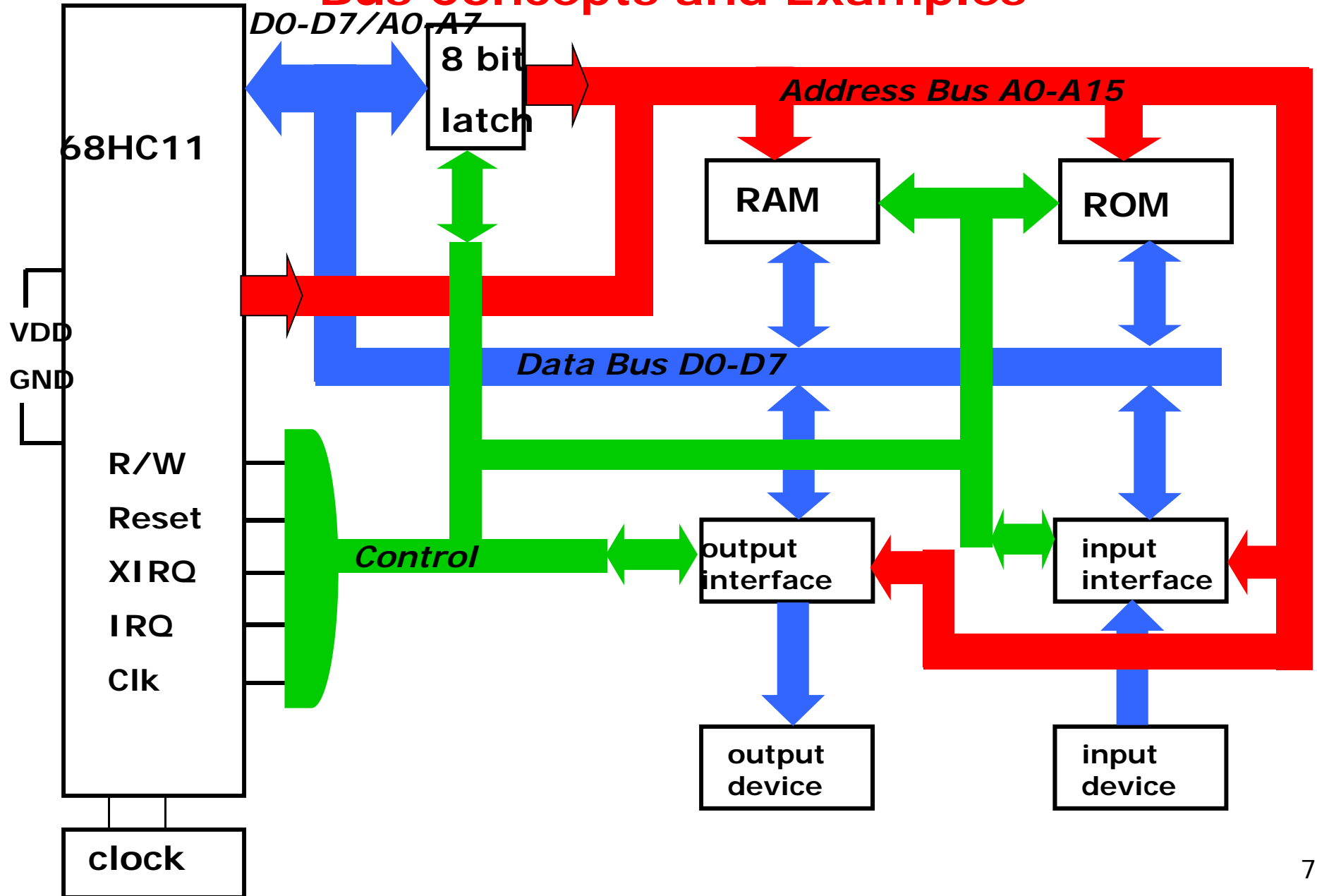
Bus Concepts and Examples



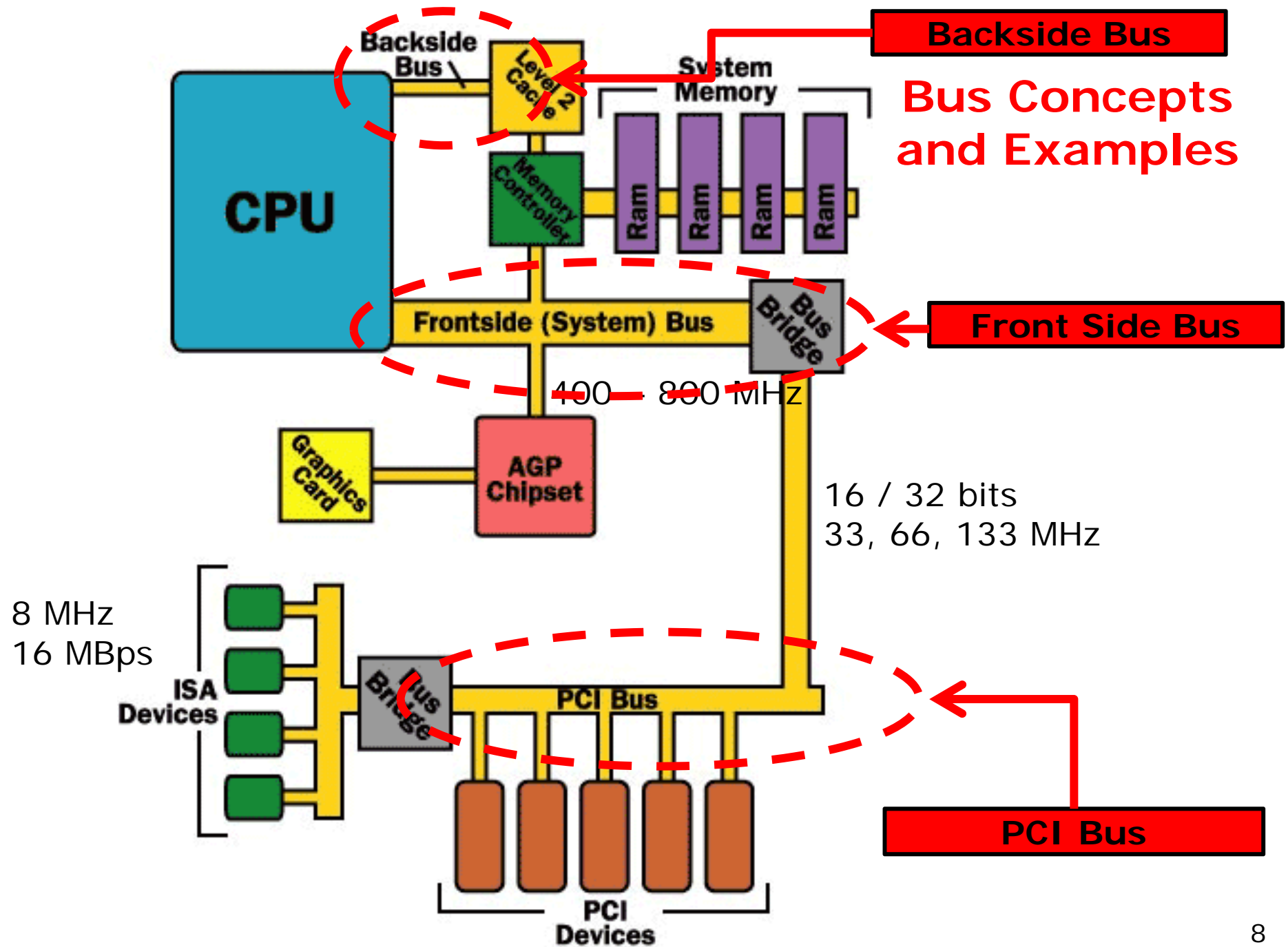
Bus Concepts and Examples



Bus Concepts and Examples



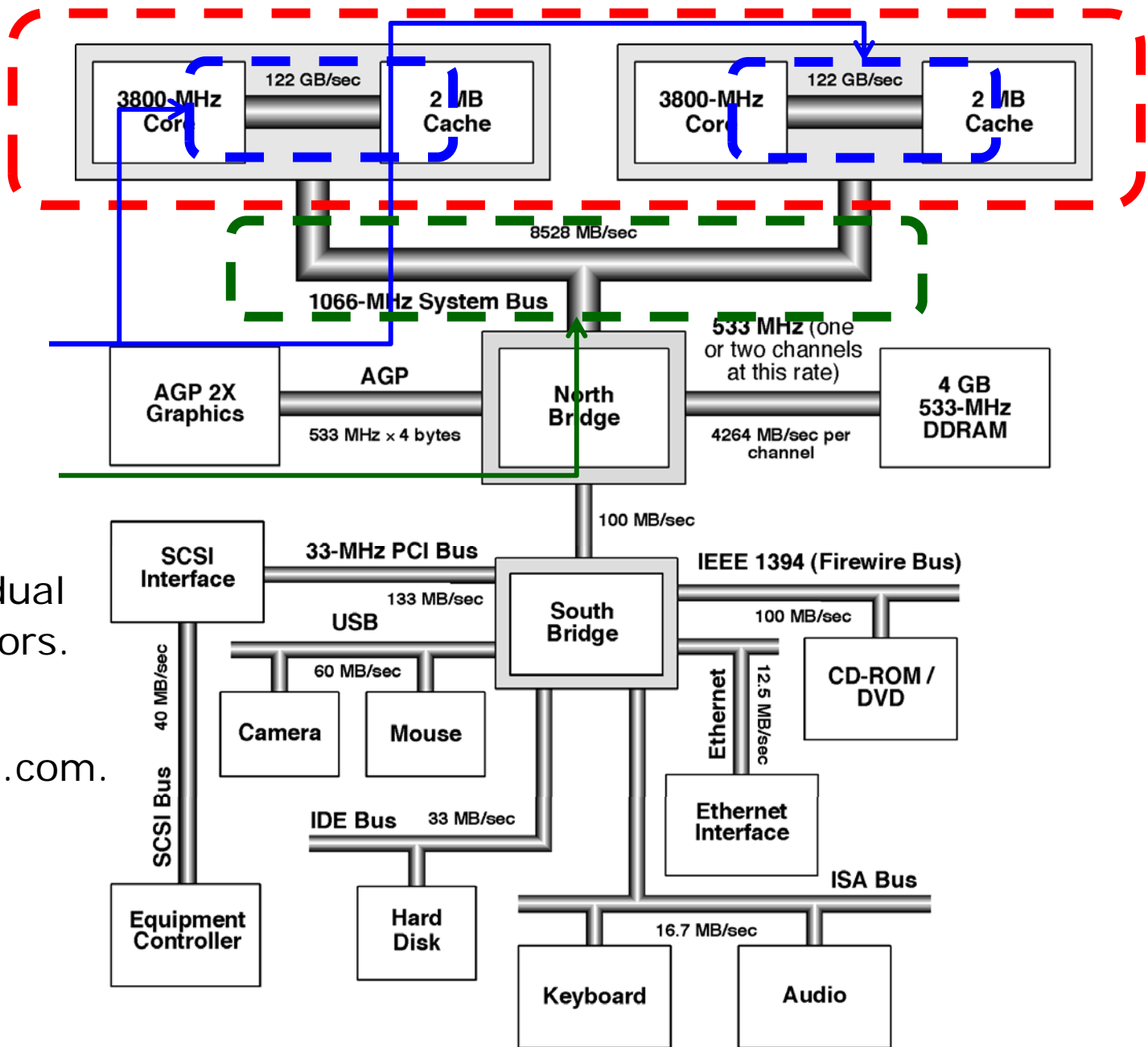
Bus Concepts and Examples



Bridge Based Bus Architecture

backside bus

frontside bus

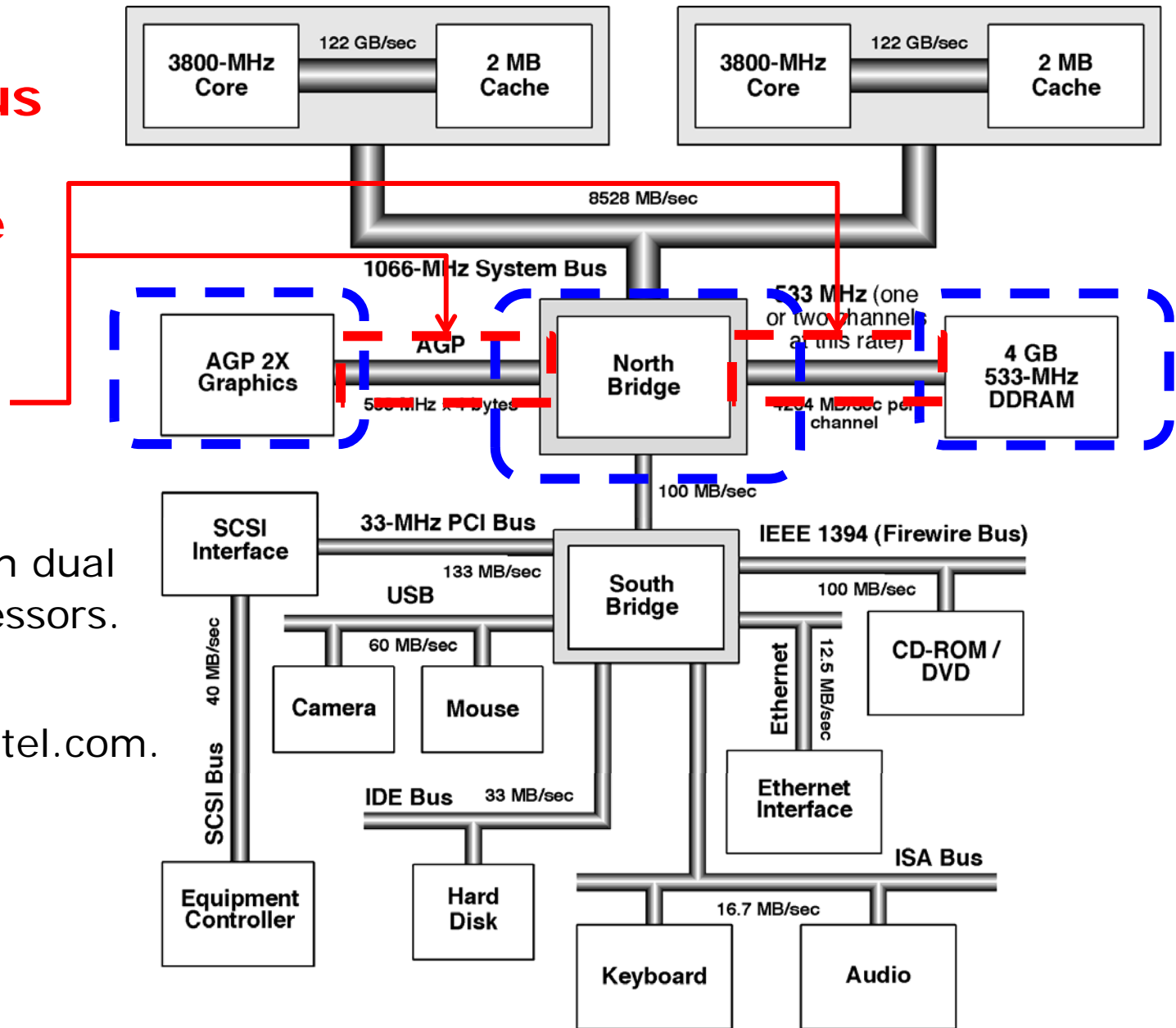


❑ Bridging with dual Pentium processors.

❑ Source:
<http://www.intel.com>.

Bridge Based Bus Architecture

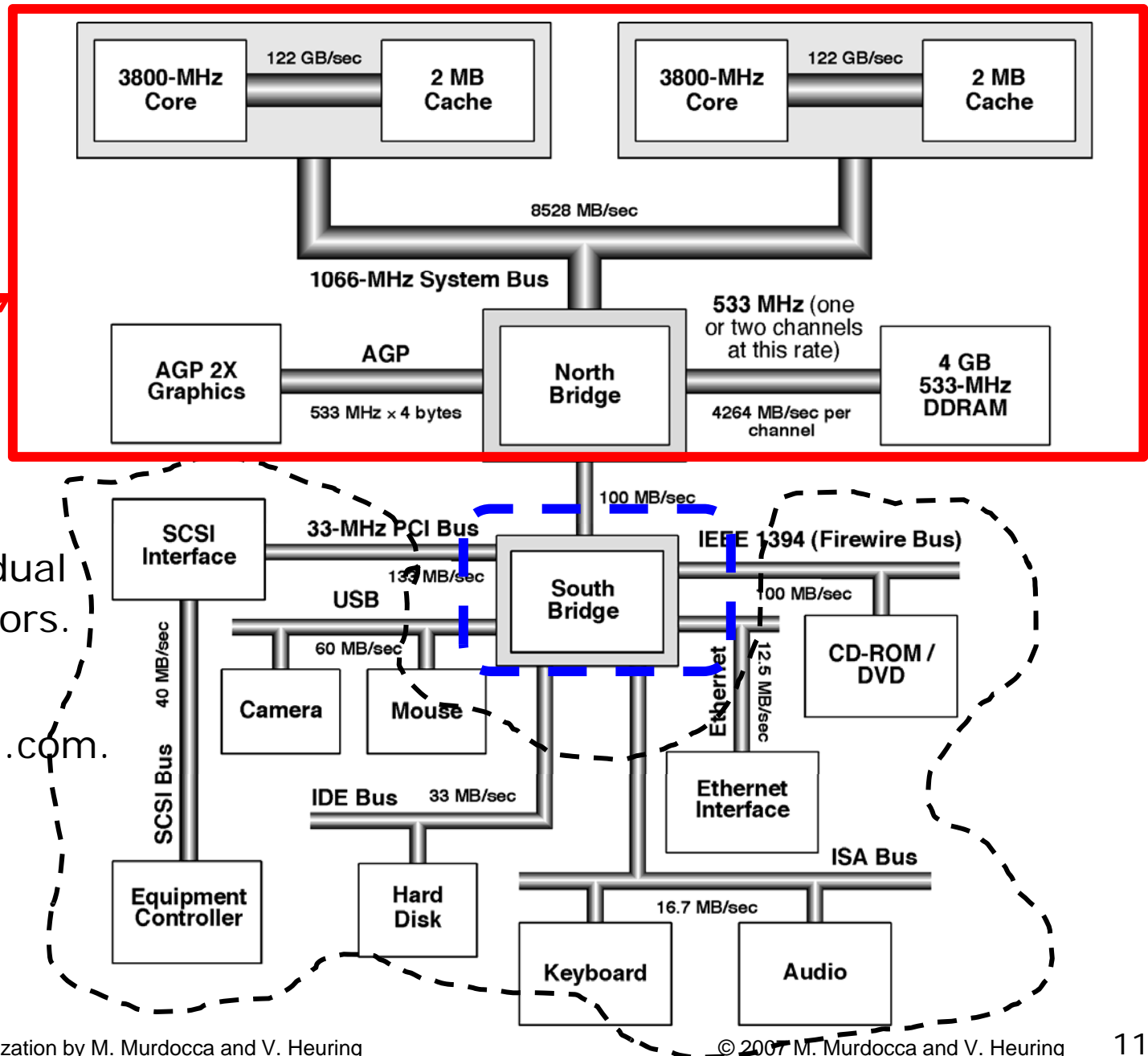
high-speed bus



- ❑ Bridging with dual Pentium processors.
- ❑ Source: <http://www.intel.com>.

Bridge Based Bus Architecture

On motherboard



- ❑ Bridging with dual Pentium processors.
- ❑ Source: <http://www.intel.com>.

Bus Basics again: REVIEW

Bus Transaction:

- a sequence of bus operations
- send address + send/receive data
 - ➔ to memory or to peripherals

Synchronous:

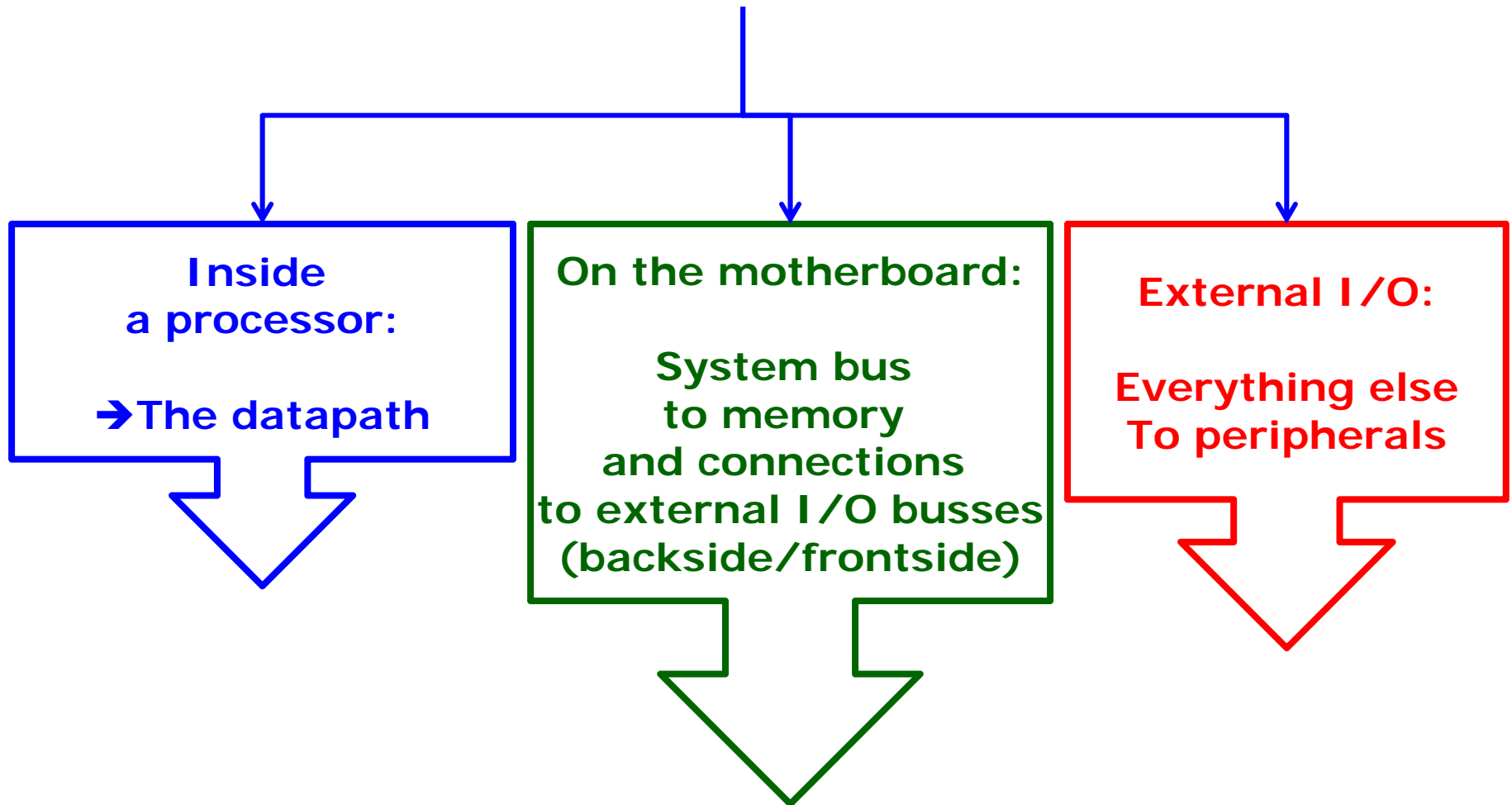
- clock as part of control lines
- fixed protocol relative to clock
- fast, little hardware
- usually only for memory

Asynchronous:

- not clocked
- good for variety of devices
- needs handshaking protocol, which means
 - a series of steps to coordinate transfers

Where do we find a “bus”?

BUS is at times an overused label → be careful



Microprogramming

What is microprogramming?

Programming at the level of the control unit in the CPU, the ALU signals, the transfers between registers

→ The micro steps which actually implement instructions

Microarchitecture design and microprogramming can change without changes needed in the ISA itself

There is a lot more in the textbook than necessary for this course

1. Read (superficially) for the overall content sections 5.1 to 5.4 in M&H and 3.4 to 3.8 in Stallings
2. Example for datapath behaviour given in the lecture notes here with a different (simpler) implementation

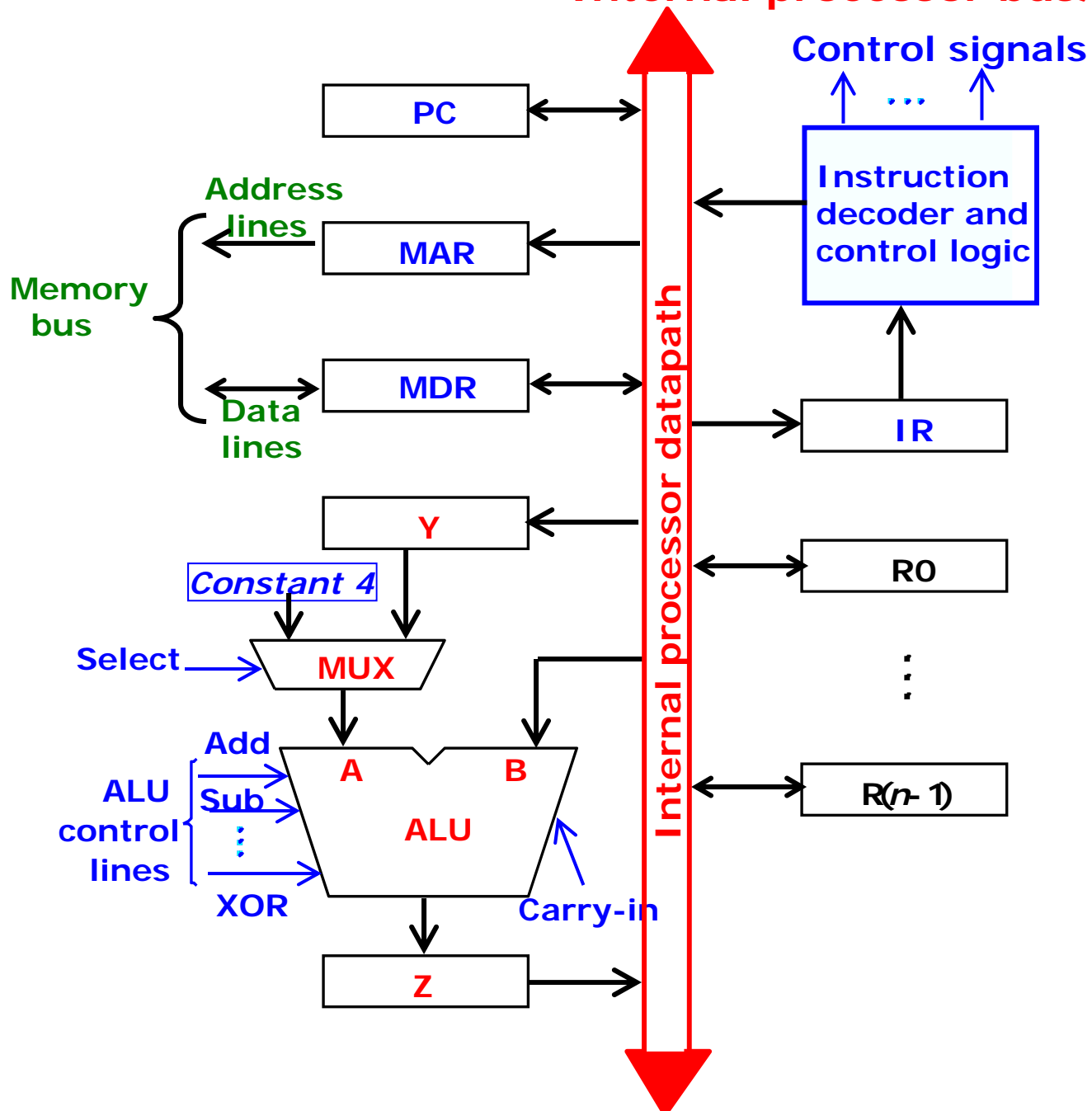
Look again inside the CPU: Datapath and Control

- ❑ **Datapath** - supports data transfer and processing operations
- ❑ **Control Unit** - Determines the enabling and sequencing of the operations

DATAPATH DESIGN:

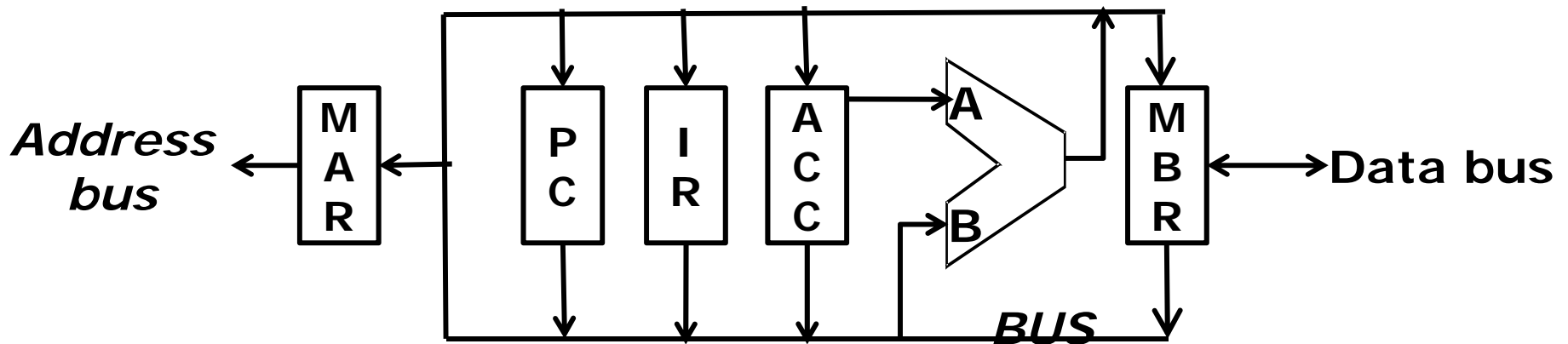
- ❑ A difficult task consisting of:
 - Choosing number of registers
 - Choosing number of busses
- ❑ Tightly related to the instruction set (register operations)
- ❑ Choices impact performance (cycles per instructions) and complexity (number of states, output signals) of controller
- ❑ Three choices for datapath bus structures:
 1. Point to point (*infeasible in most cases*)
 2. Single bus
 3. Multiple bus

Internal processor bus/datapath



As seen before for
examples in Fetch
– Decode –
Execute cycle

Datapath Design Example

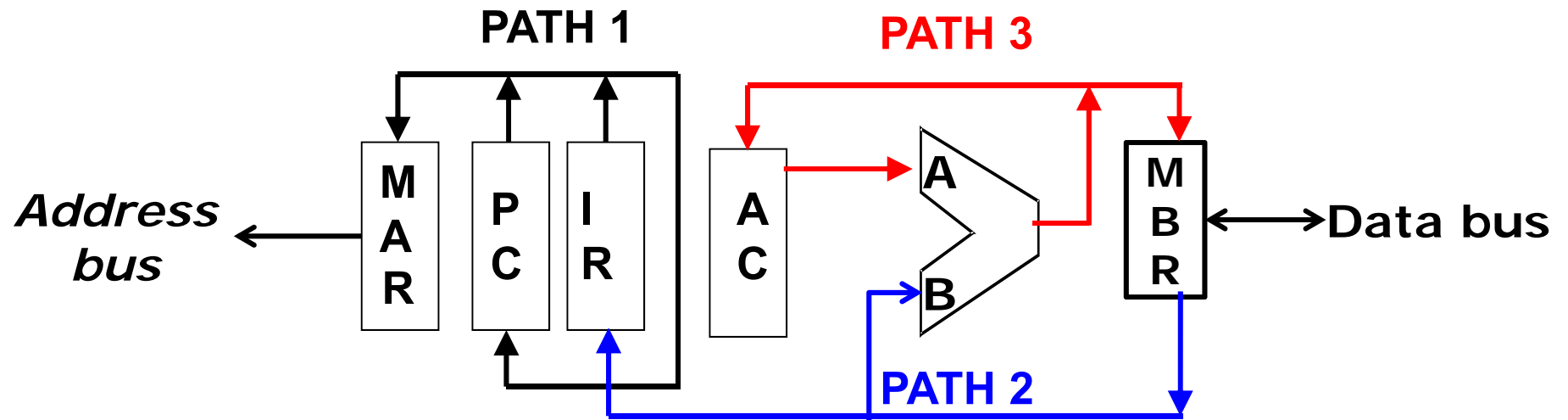


Architecture 1

Can we do better than this basic structure with one path?

- a) Consider the operations and their intersections
- b) IF some operations do not use the same components ever at the same time, the datapath can be subdivided into groups

Datapath Design Example: Change 3



Part 1:

- $MAR \leftarrow IR$
- $MAR \leftarrow PC$
- $PC \leftarrow IR<13:0>$

Note that these operations are never done in the same state

Part 2:

- $IR \leftarrow MBR$
- $AC \leftarrow MBR$
- $ALU [B] \leftarrow MBR$

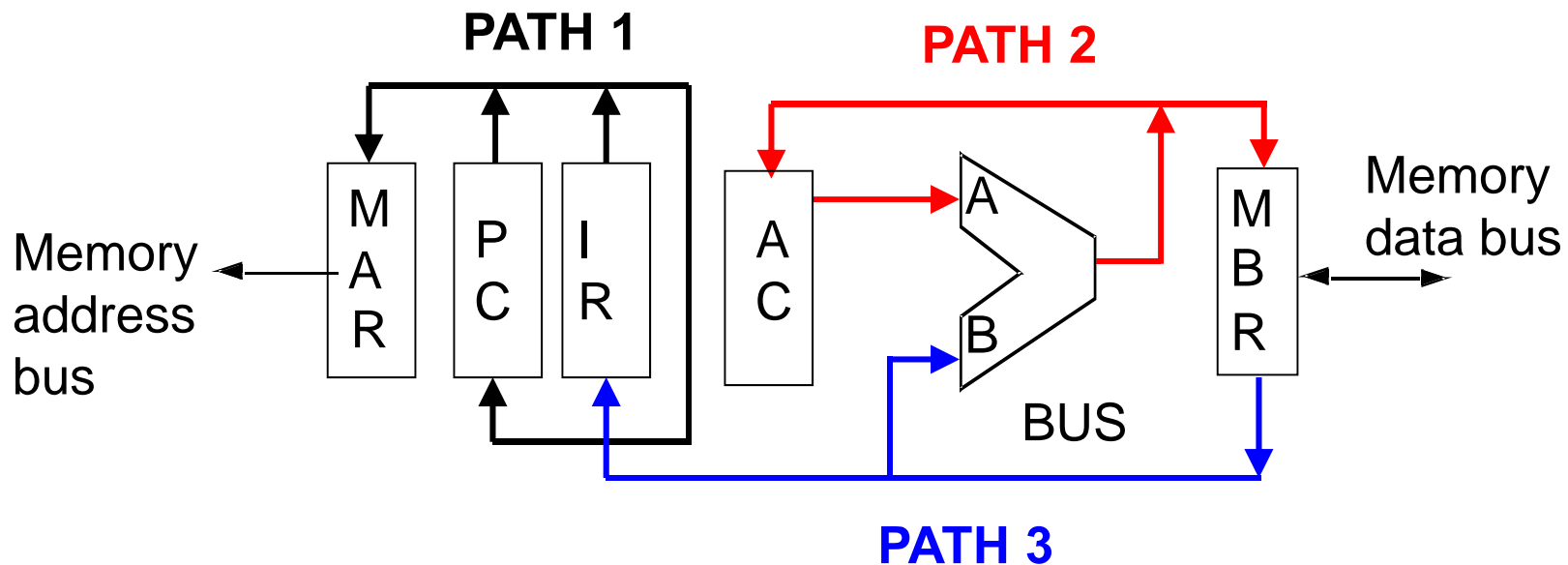
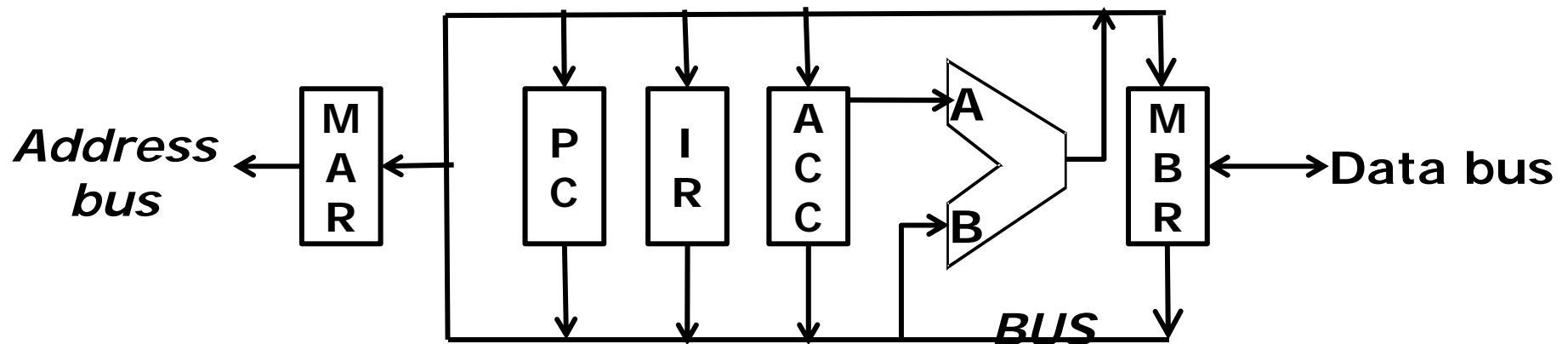
Note that these operations are never done in the same state

Part 3:

- $MBR \leftarrow AC$
- $AC \leftarrow MBR$
- $AC \leftarrow ALU$

Note that these operations are never done in the same state

Datapath Design Example: final



Architecture 2: Higher parallelism – for example, the result of ALU can be transferred to AC in same cycle

Guiding Principles for Datapath Design

□ The set of registers:

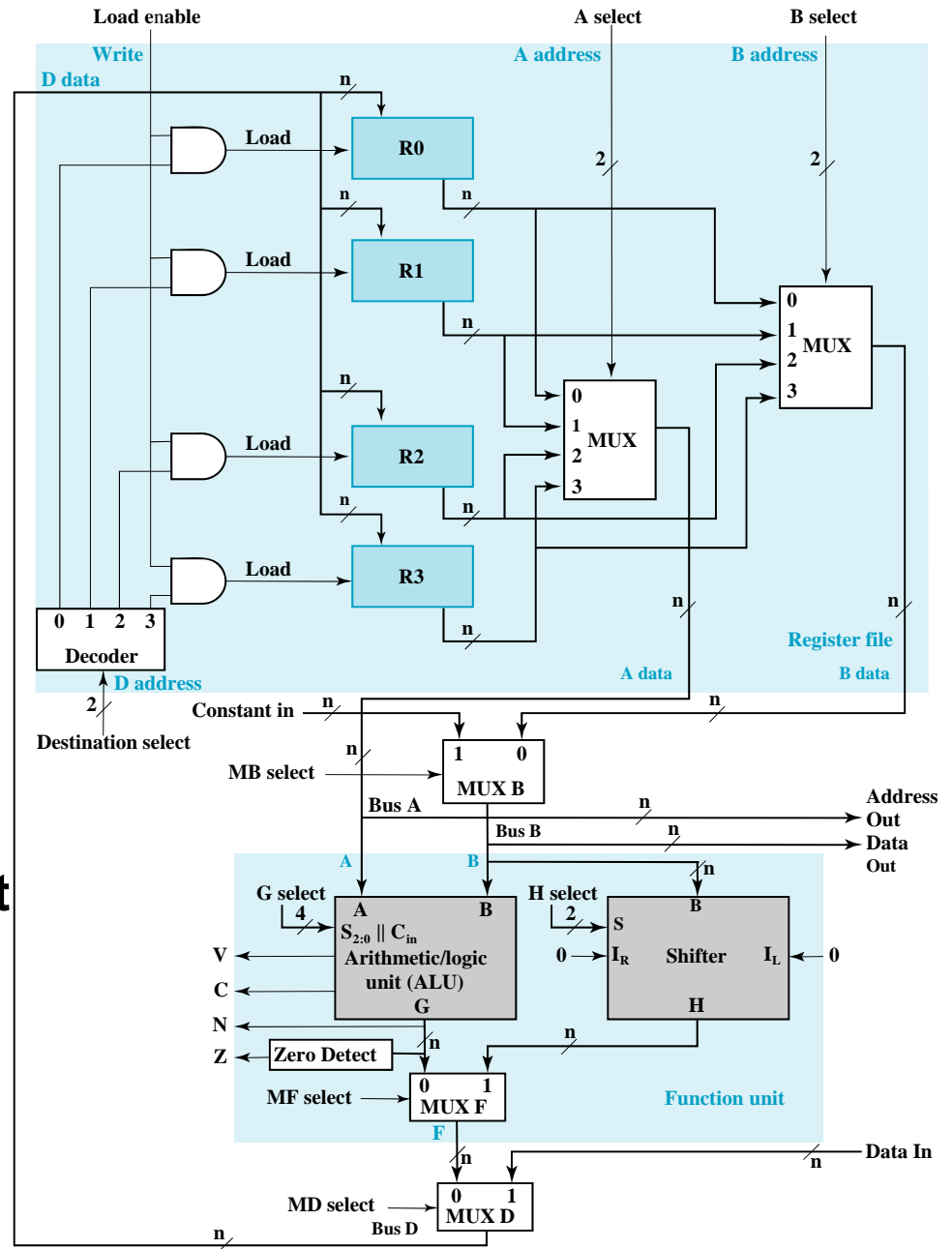
- a) Collection of individual registers
- b) A set of registers with common access resources called a *register file*
- c) A combination of the above

□ Microoperation implementation

- One or more shared resources for implementing microoperations
- Buses – as the shared transfer paths
- *Arithmetic-Logic Unit (ALU)* - shared resource for implementing arithmetic and logic microoperations
- Shifter (barrel shifter) - shared resource for implementing shift microoperations

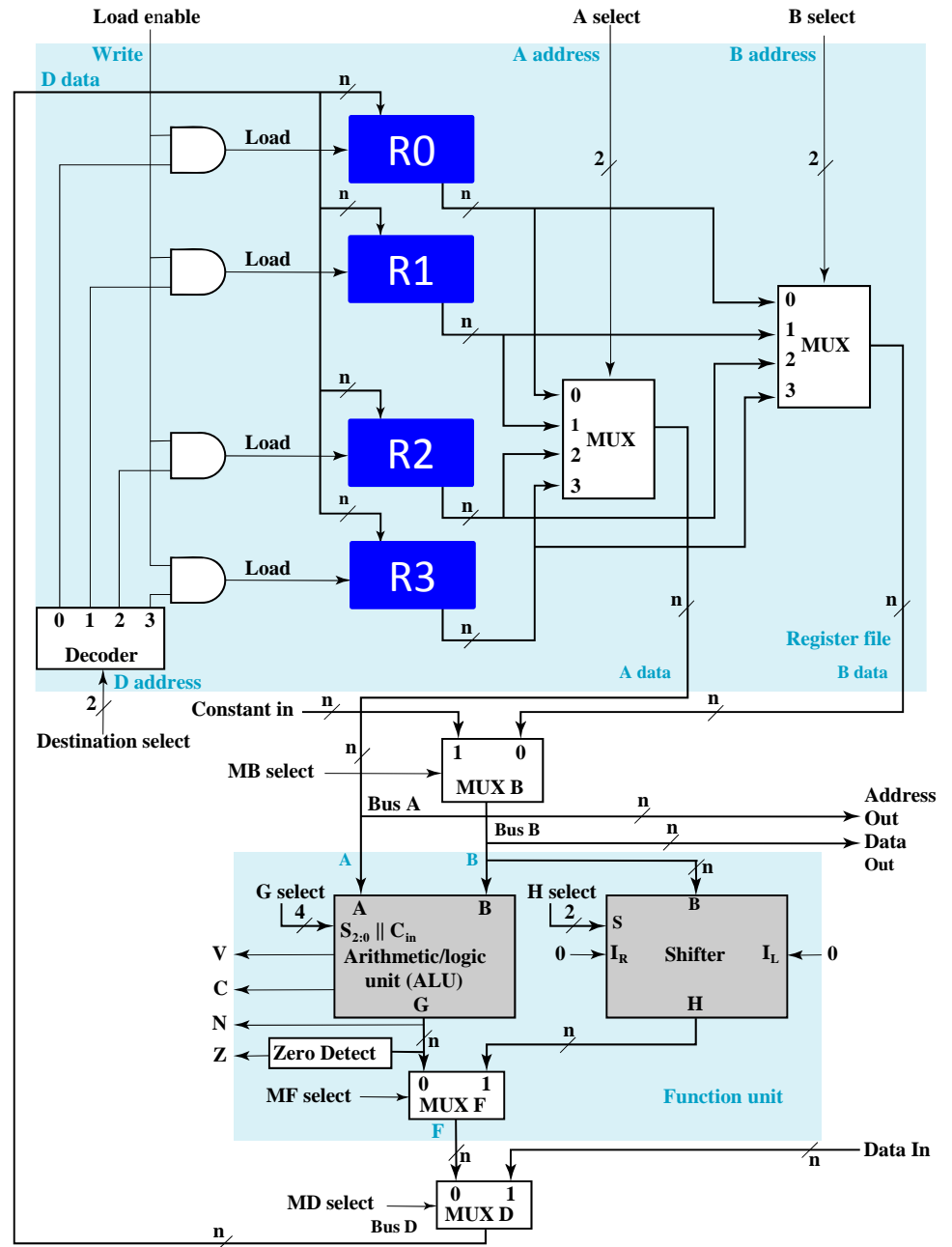
Datapath Example

- Four registers
- Two mux-based register selectors
- Register destination decoder
- Mux B for external constant input
- Buses A and B with external address and data outputs
- ALU and Shifter with Mux F for output select
- Mux D for external data input
- Logic for generating status bits V, C, N, Z



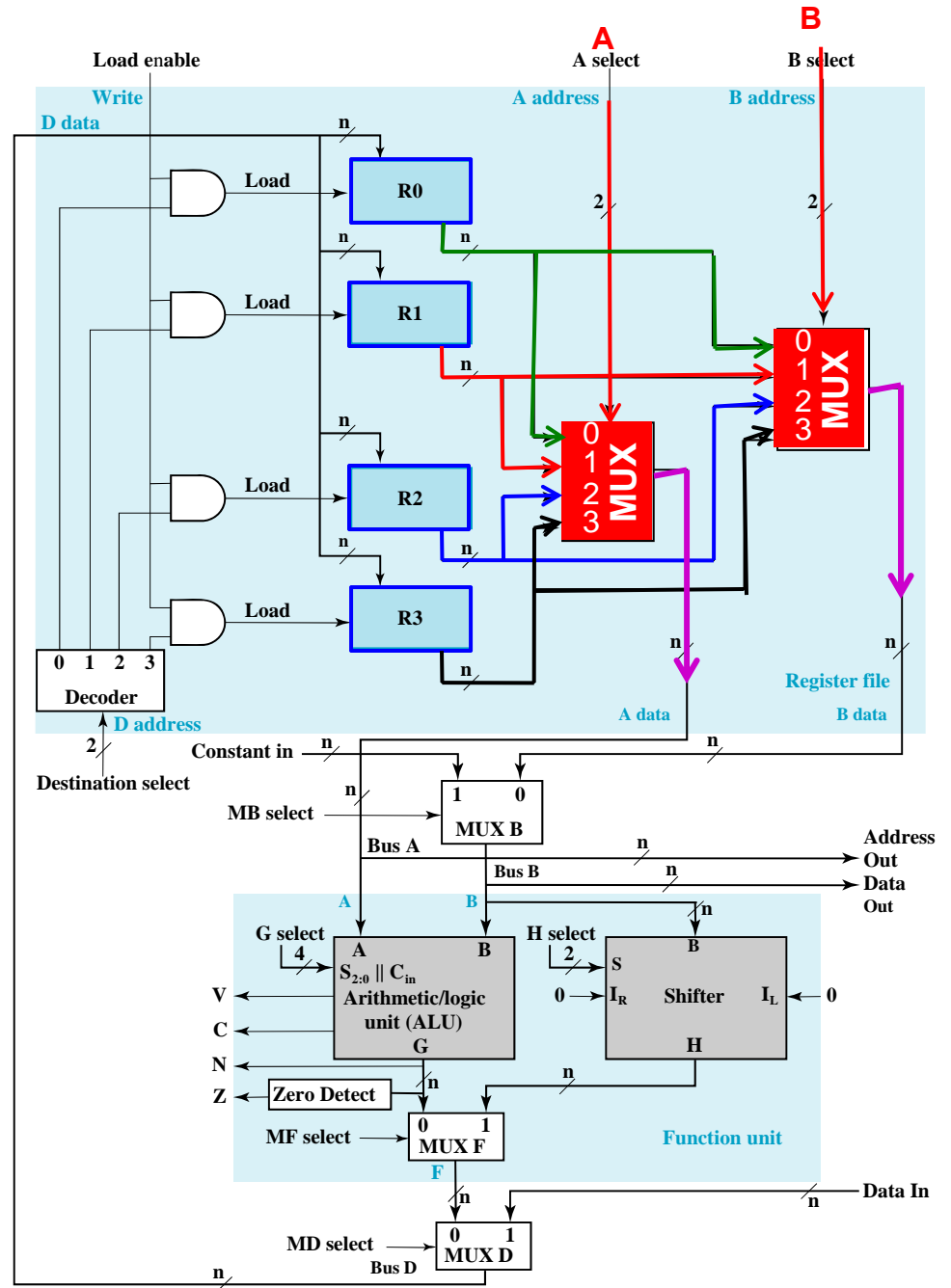
Datapath Example

➤ Four registers



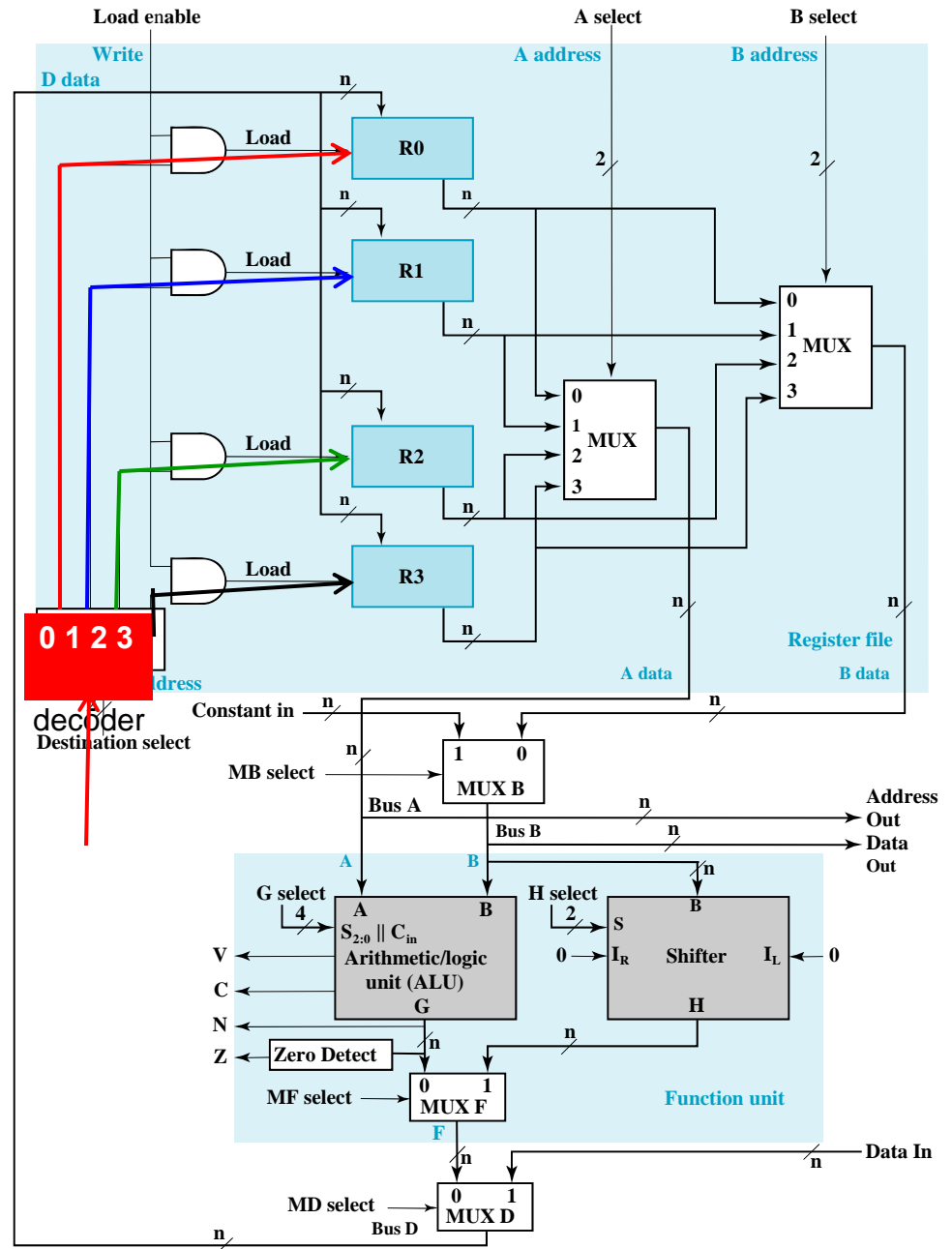
Datapath Example

- Four registers
- Two mux-based register selectors



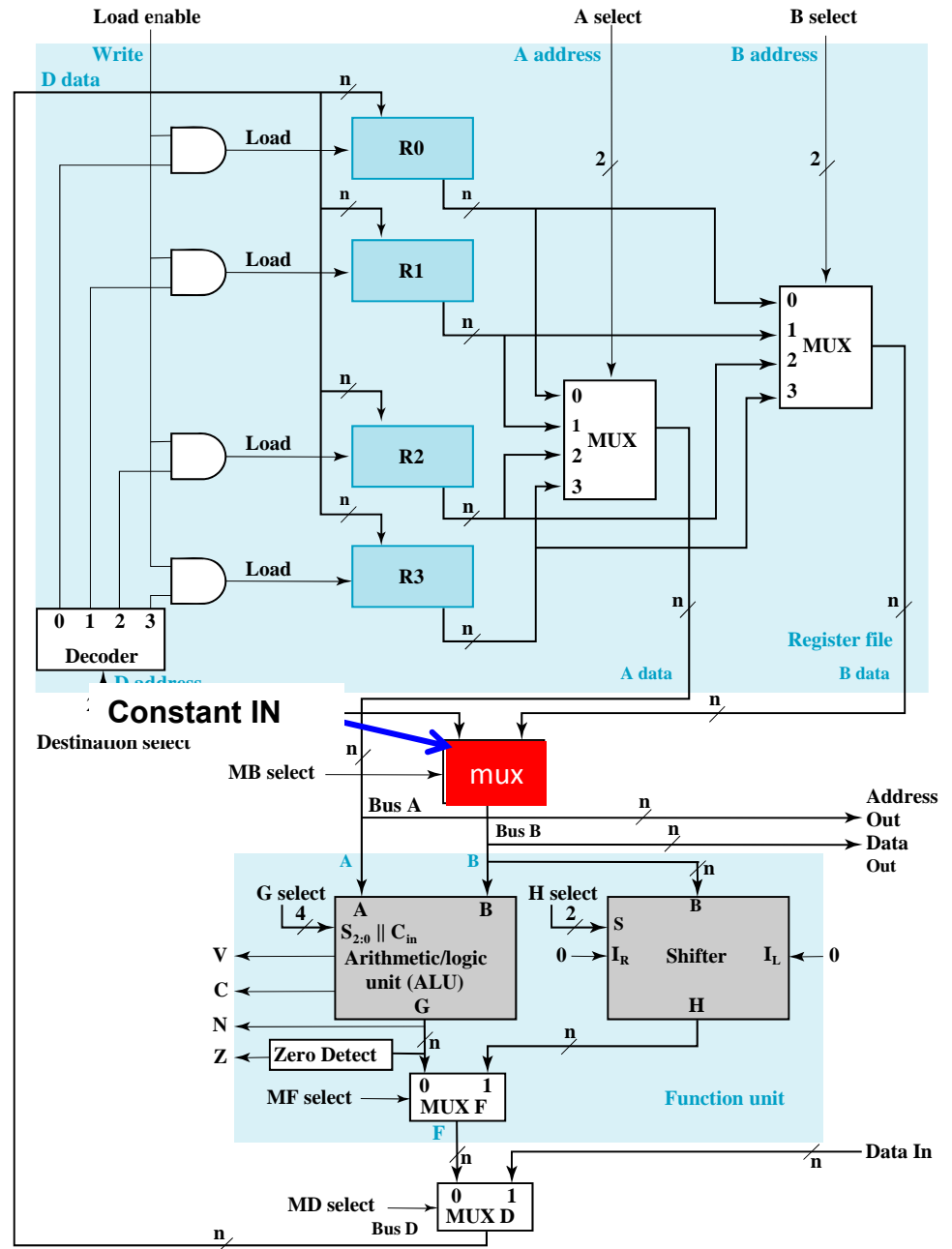
Datapath Example

- Four registers
- Two mux-based register selectors
- Register destination decoder



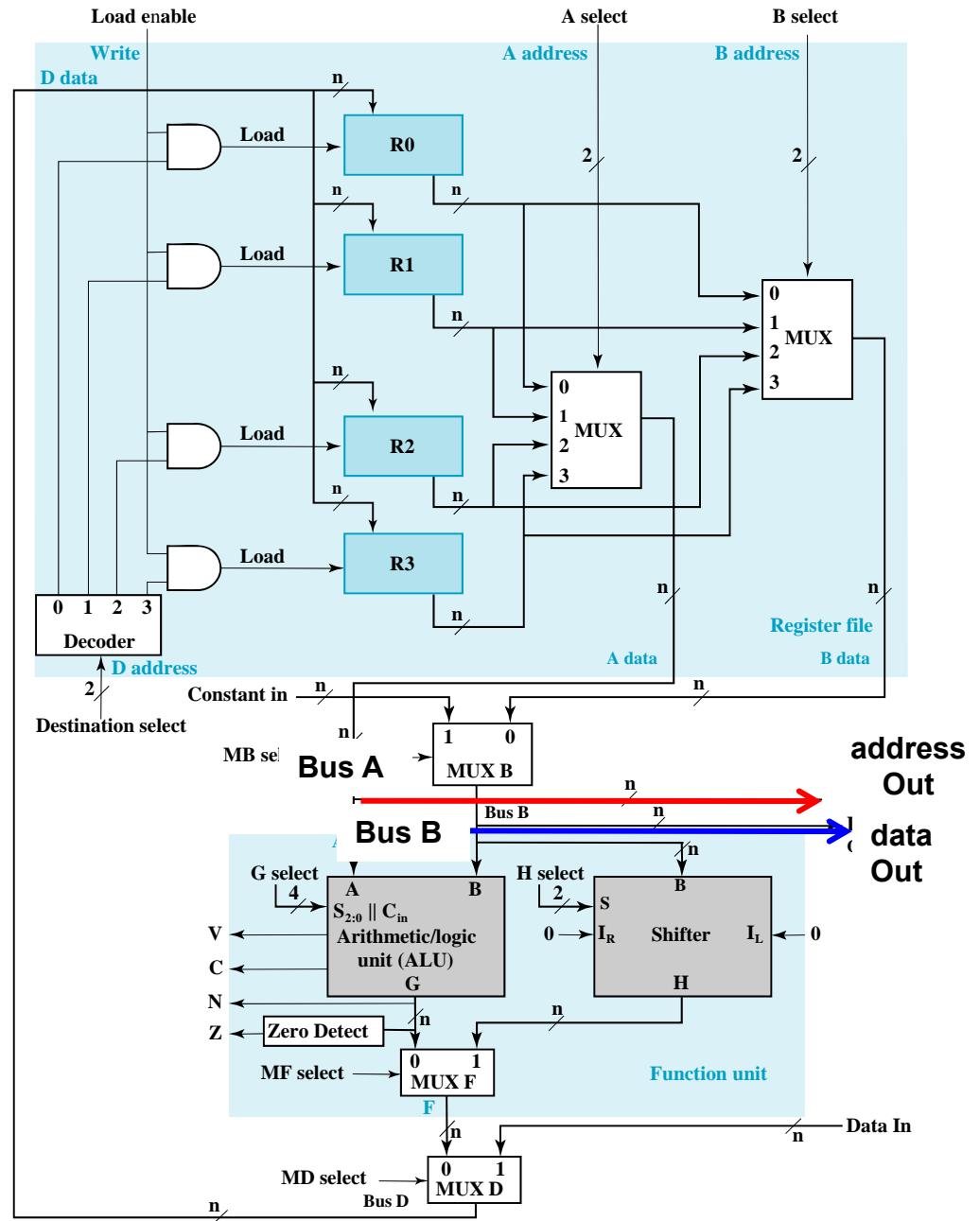
Datapath Example

- Four registers
- Two mux-based register selectors
- Register destination decoder
- Mux B for external constant input



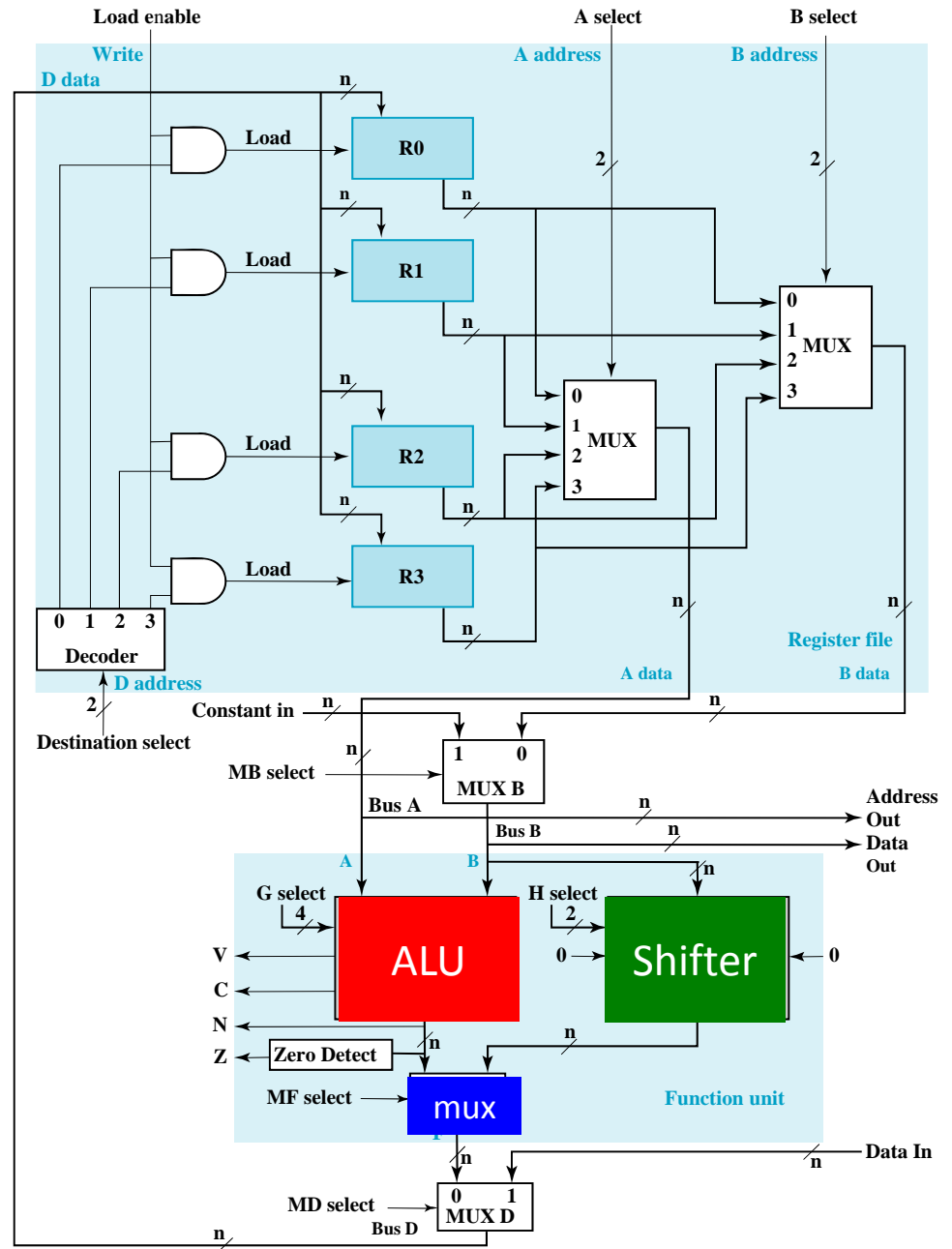
Datapath Example

- Four registers
- Two mux-based register selectors
- Register destination decoder
- Mux B for external constant input
- Buses A and B with external address and data outputs



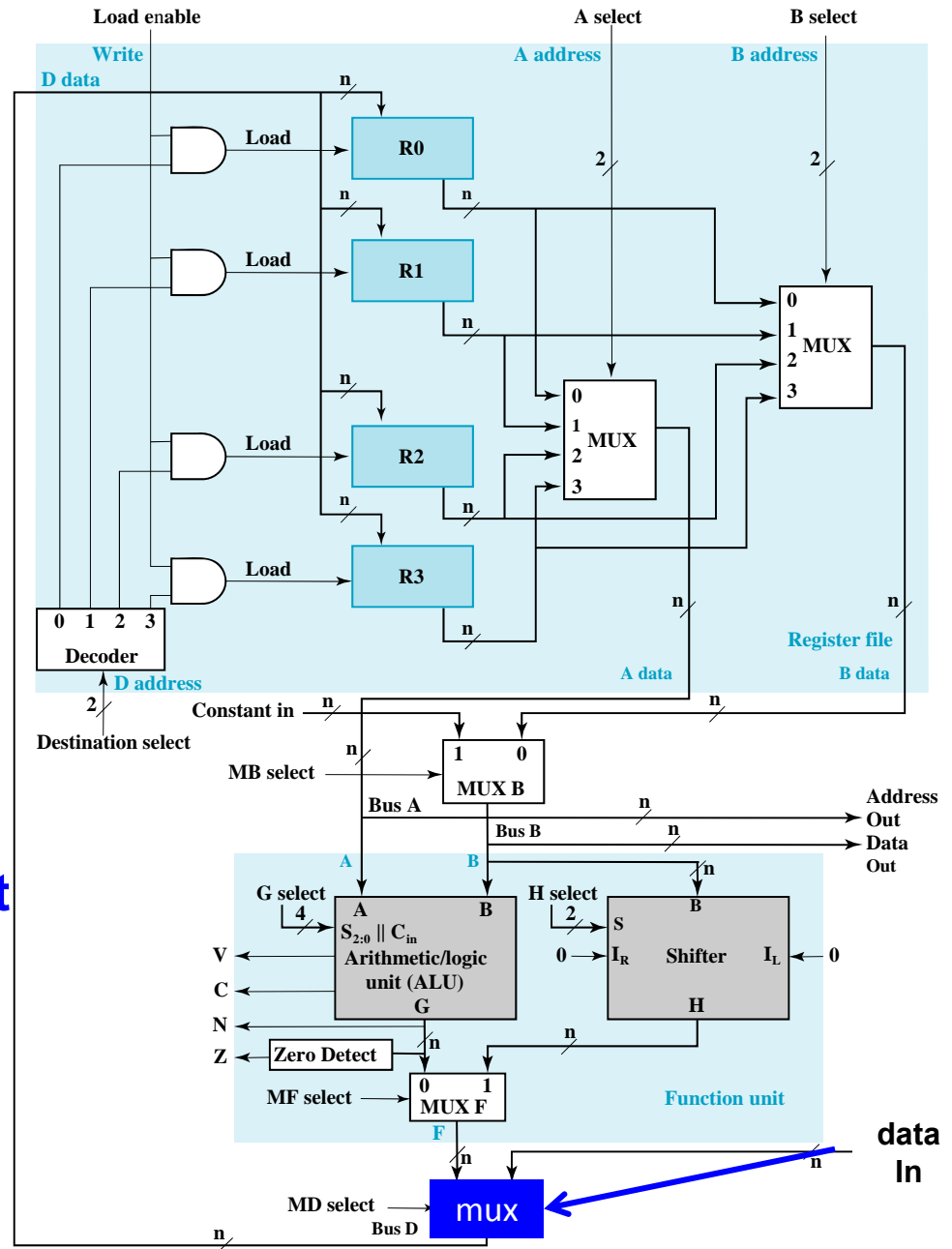
Datapath Example

- Four registers
- Two mux-based register selectors
- Register destination decoder
- Mux B for external constant input
- Buses A and B with external address and data outputs
- ALU and Shifter with Mux F for output select



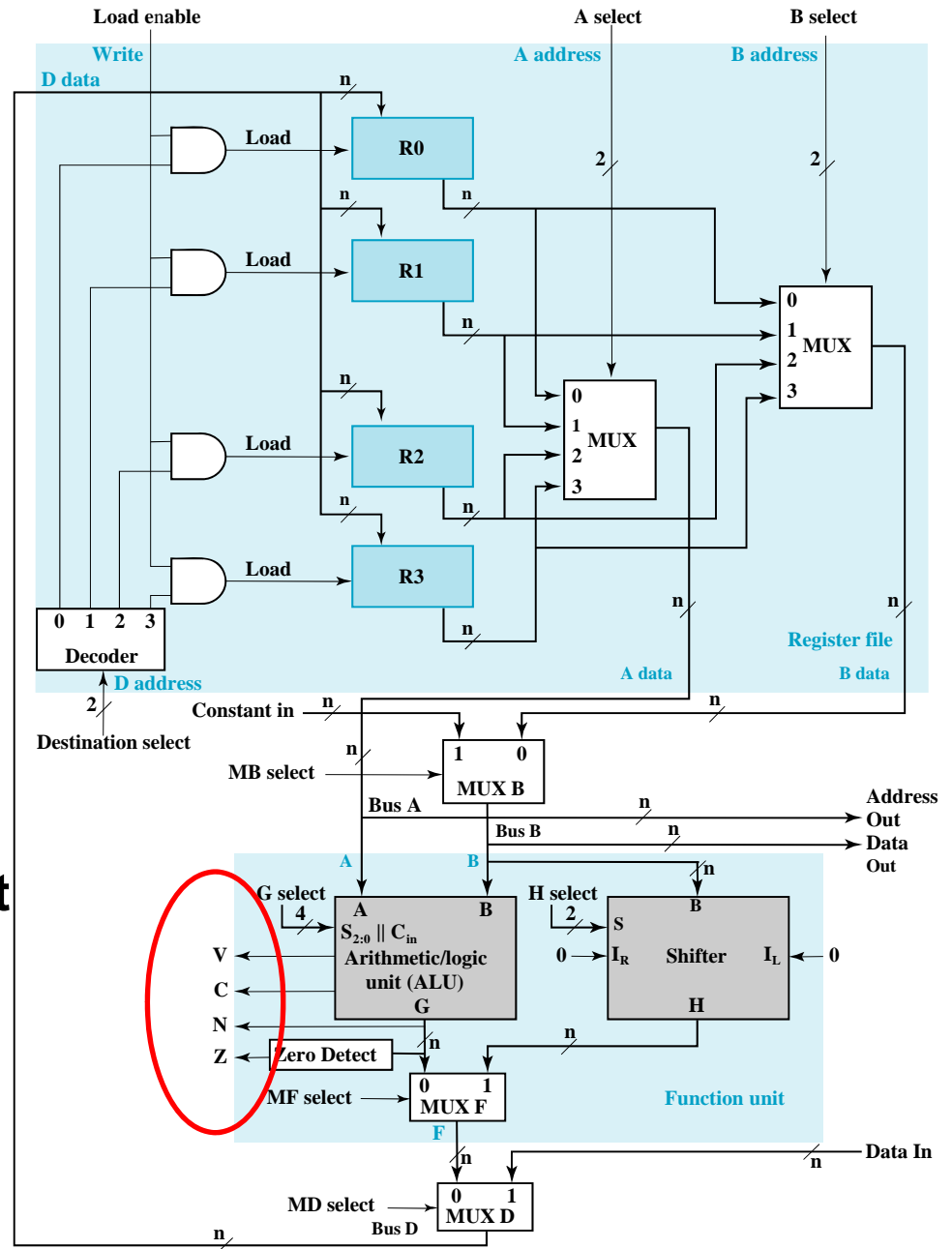
Datapath Example

- Four registers
- Two mux-based register selectors
- Register destination decoder
- Mux B for external constant input
- Buses A and B with external address and data outputs
- ALU and Shifter with Mux F for output select
- Mux D for external data input



Datapath Example

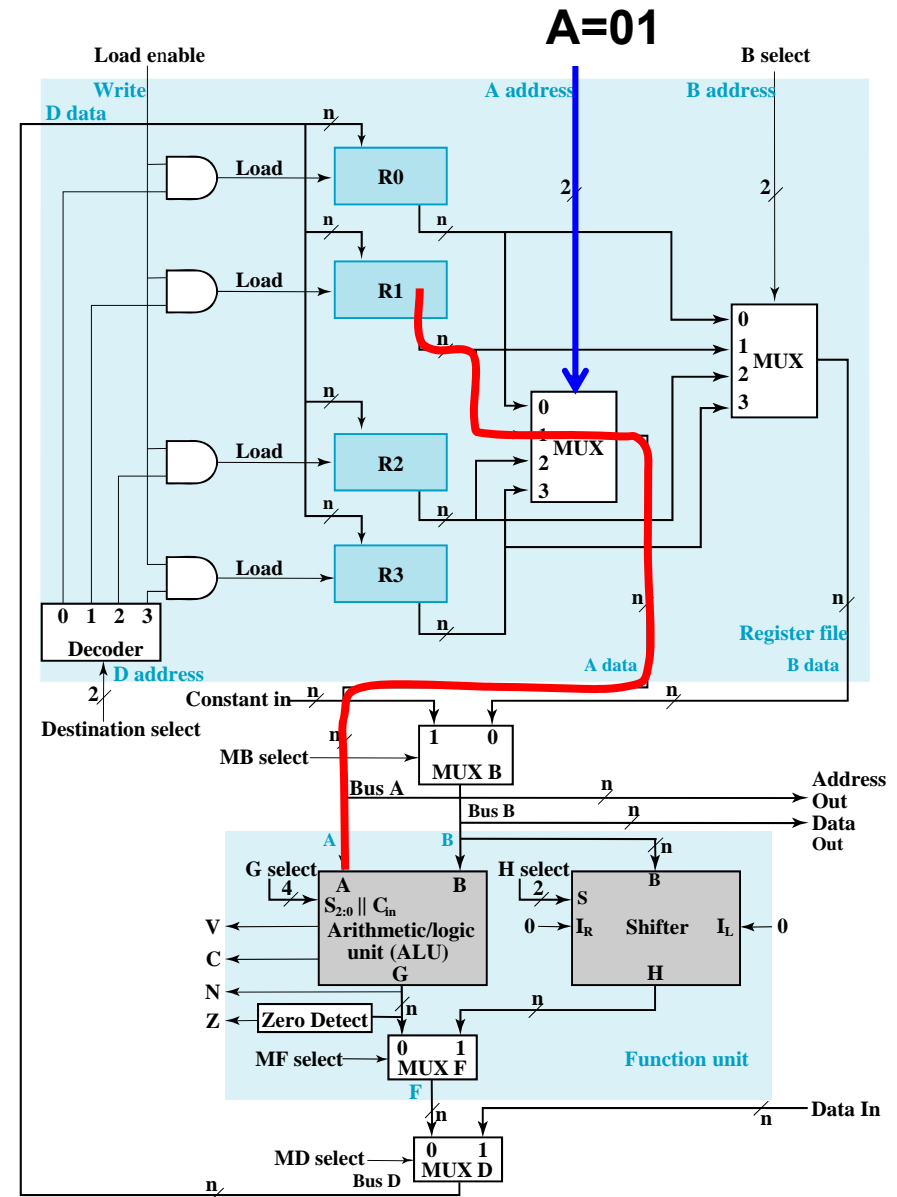
- Four registers
- Two mux-based register selectors
- Register destination decoder
- Mux B for external constant input
- Buses A and B with external address and data outputs
- ALU and Shifter with Mux F for output select
- Mux D for external data input
- Logic for generating status bits V, C, N, Z



Datapath Example: Performing a Microoperation

RTL operation: $R0 \leftarrow R1 + R2$

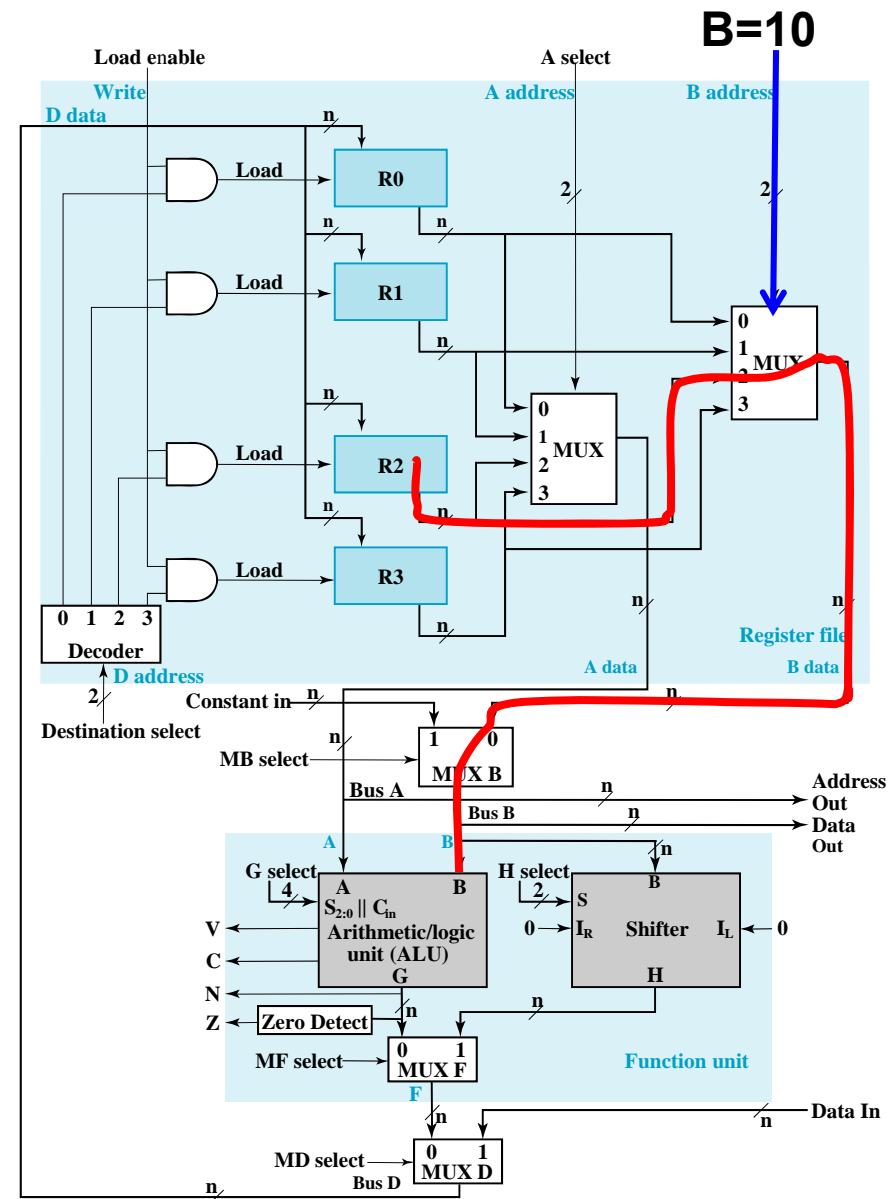
- Apply 01 to A select to place contents of R1 onto Port A of ALU



Datapath Example: Performing a Microoperation

RTL operation: $R0 \leftarrow R1 + R2$

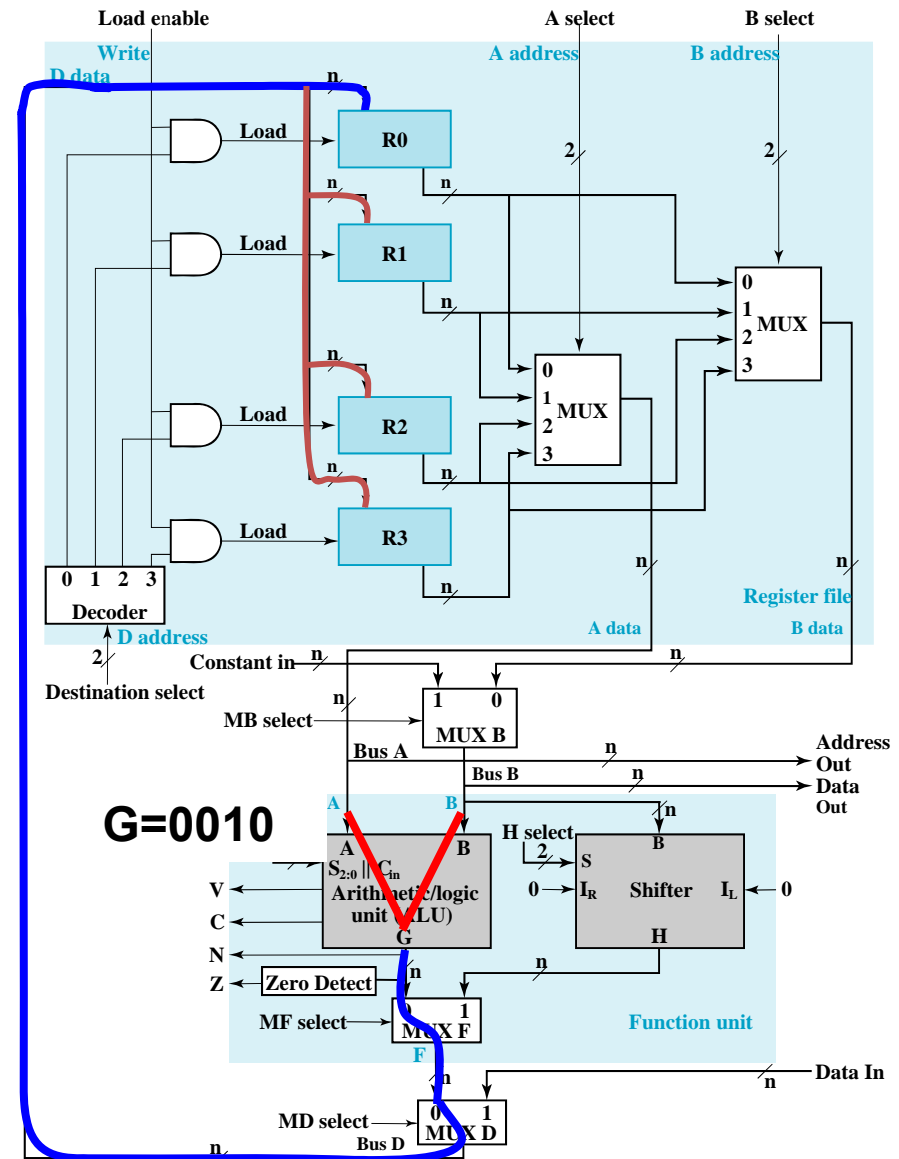
- Apply 01 to A select to place contents of R1 onto Port A of ALU
- Apply 10 to B select to place contents of R2 onto Port B of ALU



Datapath Example: Performing a Microoperation

RTL operation: $R0 \leftarrow R1 + R2$

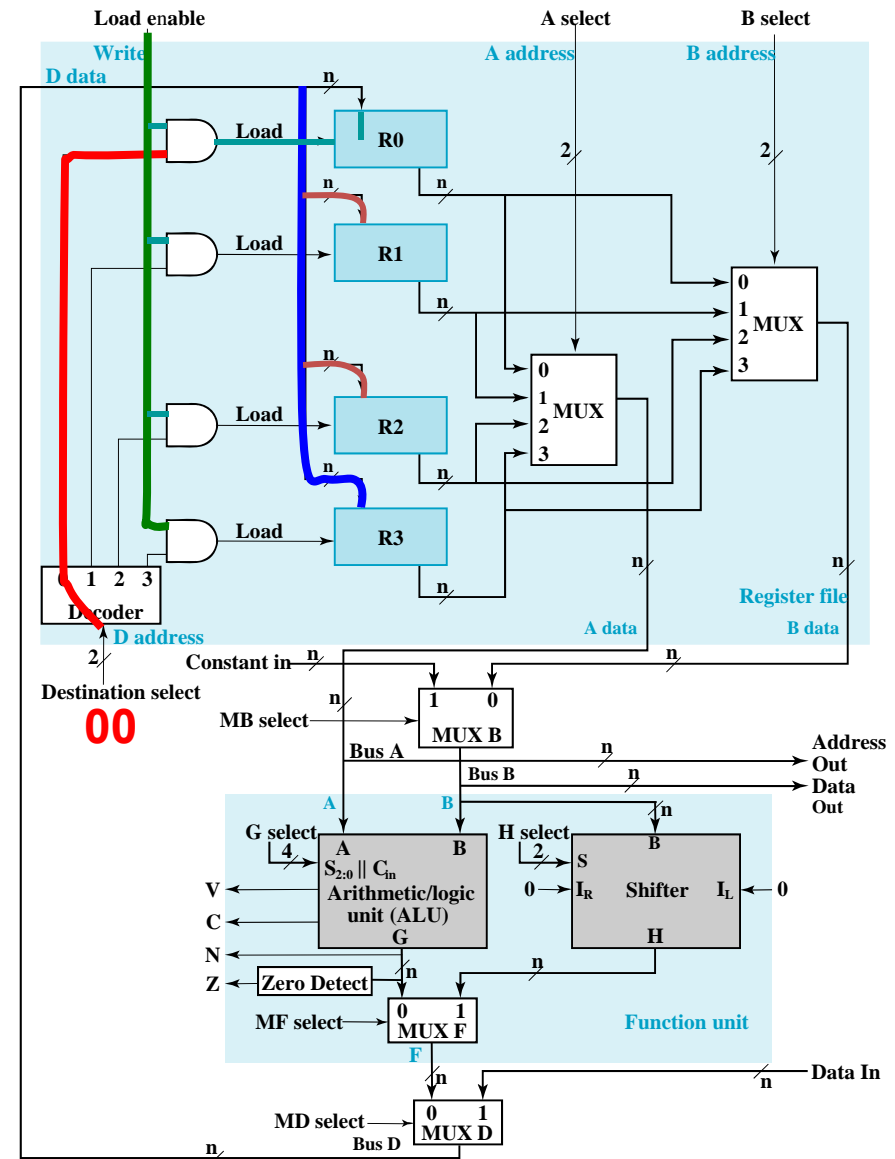
- Apply 01 to A select to place contents of R1 onto Port A of ALU
- Apply 10 to B select to place contents of R2 onto Port B of ALU
- Apply 0010 to G select to perform addition $G = \text{Port A} + \text{Port B}$
- Apply 0 to MF select and 0 to MD select to place the value of G onto BUS D



Datapath Example: Performing a Microoperation

RTL operation: $R0 \leftarrow R1 + R2$

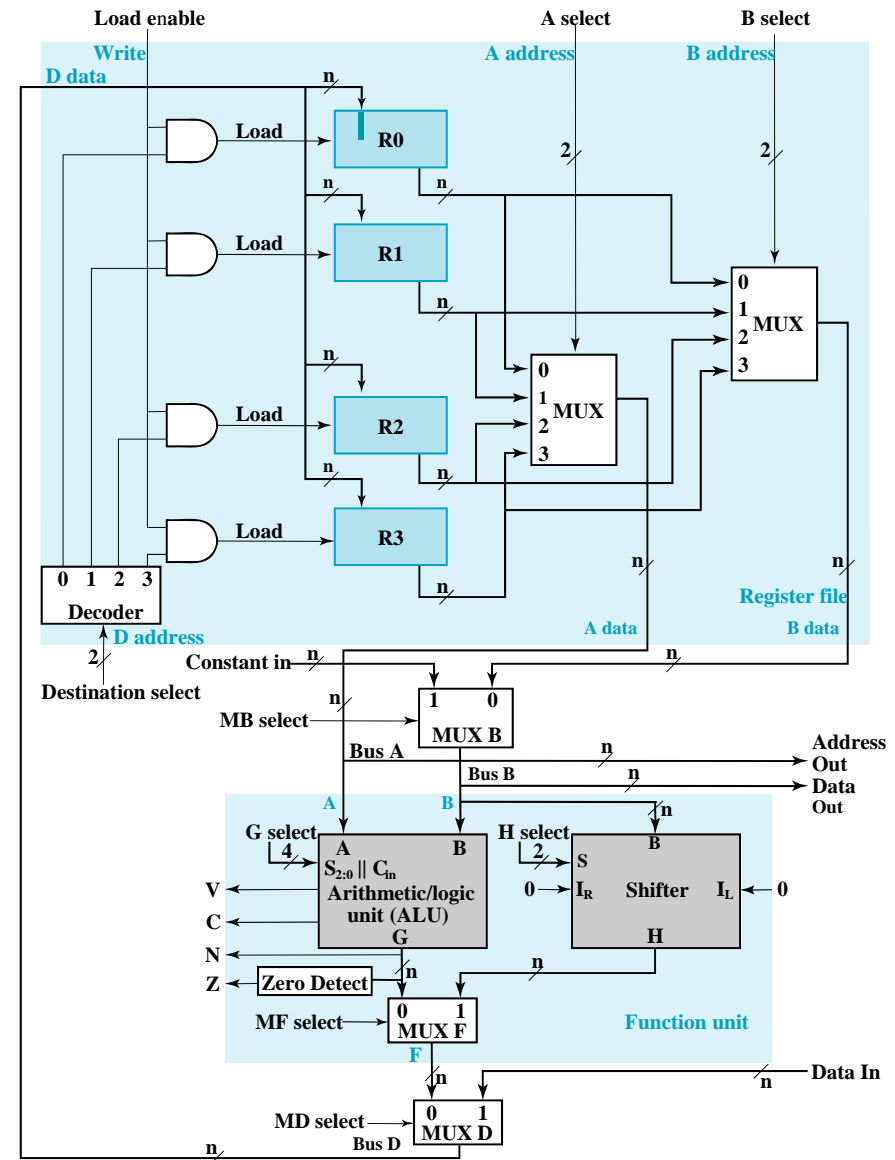
- **Apply 01 to A select to place contents of R1 onto Port A of ALU**
- **Apply 10 to B select to place contents of R2 onto Port B of ALU**
- **Apply 0010 to G select to perform addition $G = \text{Bus A} + \text{Bus B}$**
- **Apply 0 to MF select and 0 to MD select to place the value of G onto BUS D**
- **Apply 00 to Destination select to enable the Load input to R0**
- **The overall set of microoperations requires 1 clock cycle**



Datapath Example: Performing a Microoperation

RTL operation: $R0 \leftarrow R1 + R2$

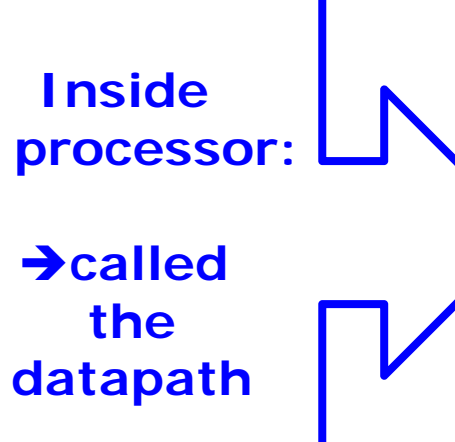
- Apply 01 to A select to place contents of R1 onto Port A of ALU
- Apply 10 to B select to place contents of R2 onto Port B of ALU
- Apply 0010 to G select to perform addition $G = \text{Bus A} + \text{Bus B}$
- Apply 0 to MF select and 0 to MD select to place the value of G onto BUS D
- Apply 00 to Destination select to enable the Load input to R0
- The overall set of microoperations requires 1 clock cycle



Going towards peripherals

- ✓Where does one find busses?
- ✓How many are there?
- ✓What are their protocols?
- ✓What are their features?

BUS is at times an overused label

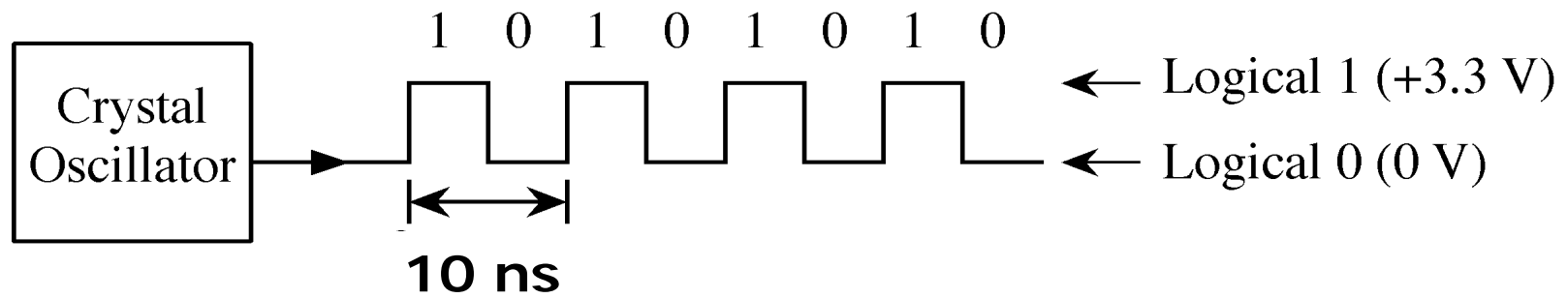


Inside
a processor:

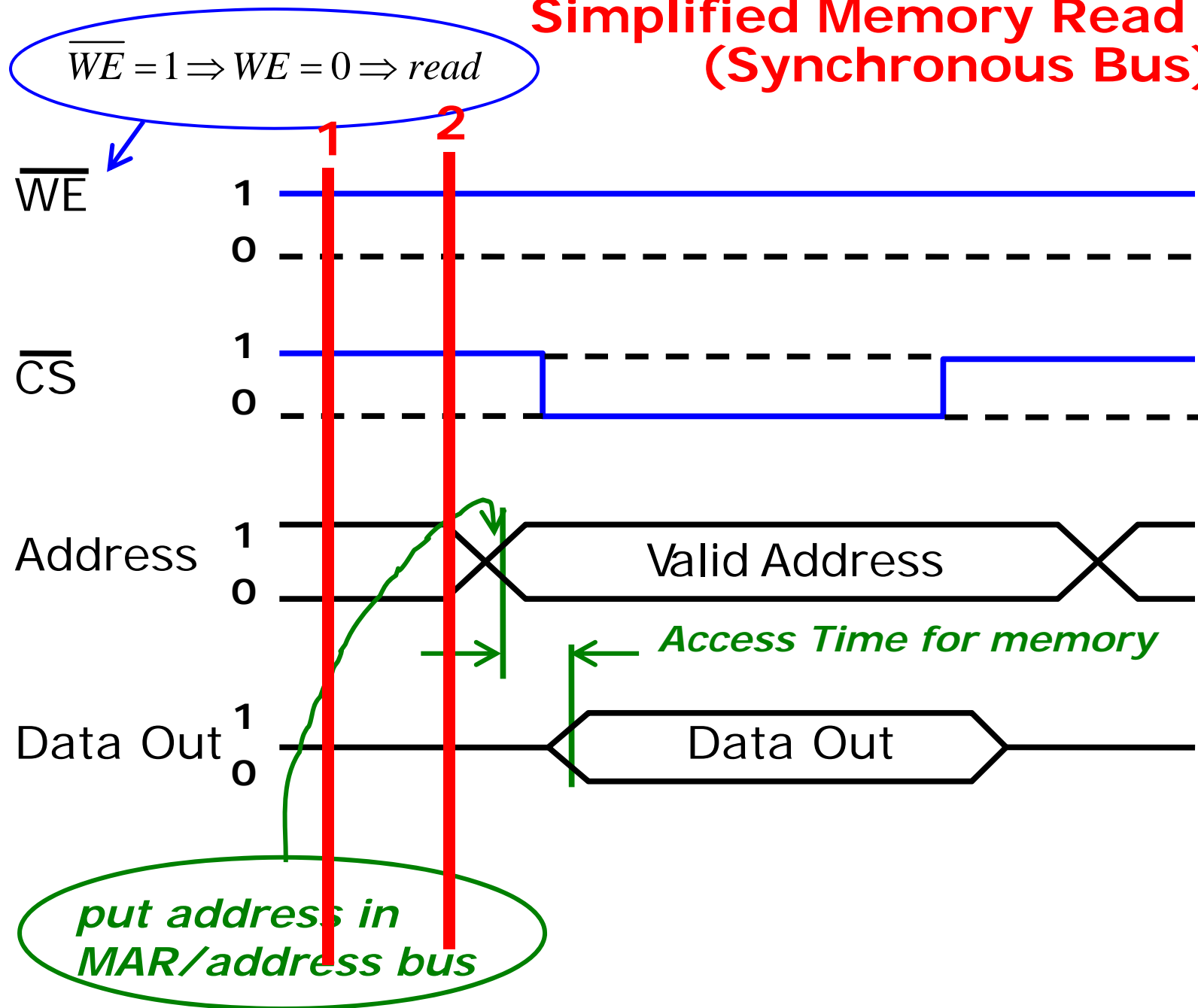
→ called
the
datapath



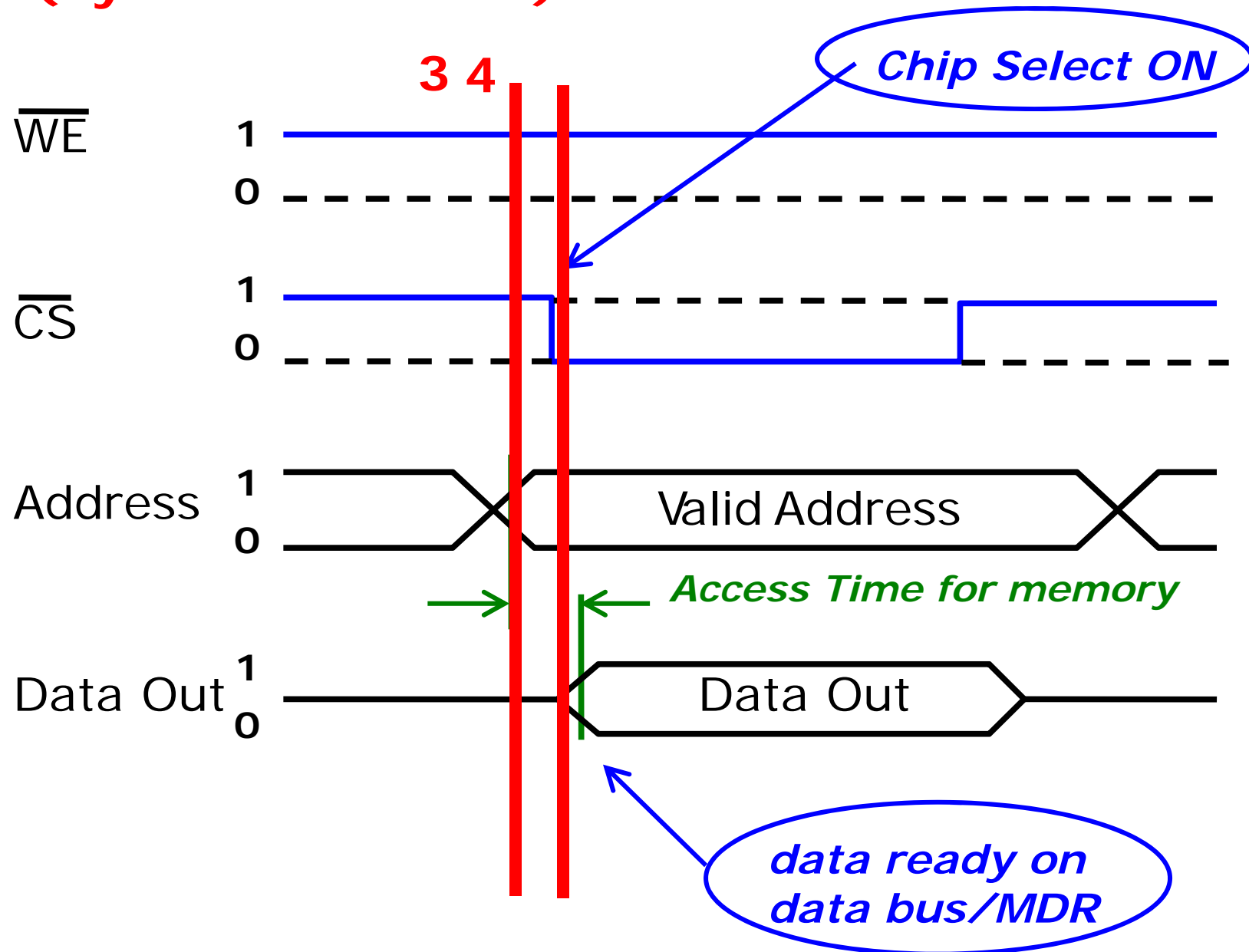
100 MHz Bus Clock



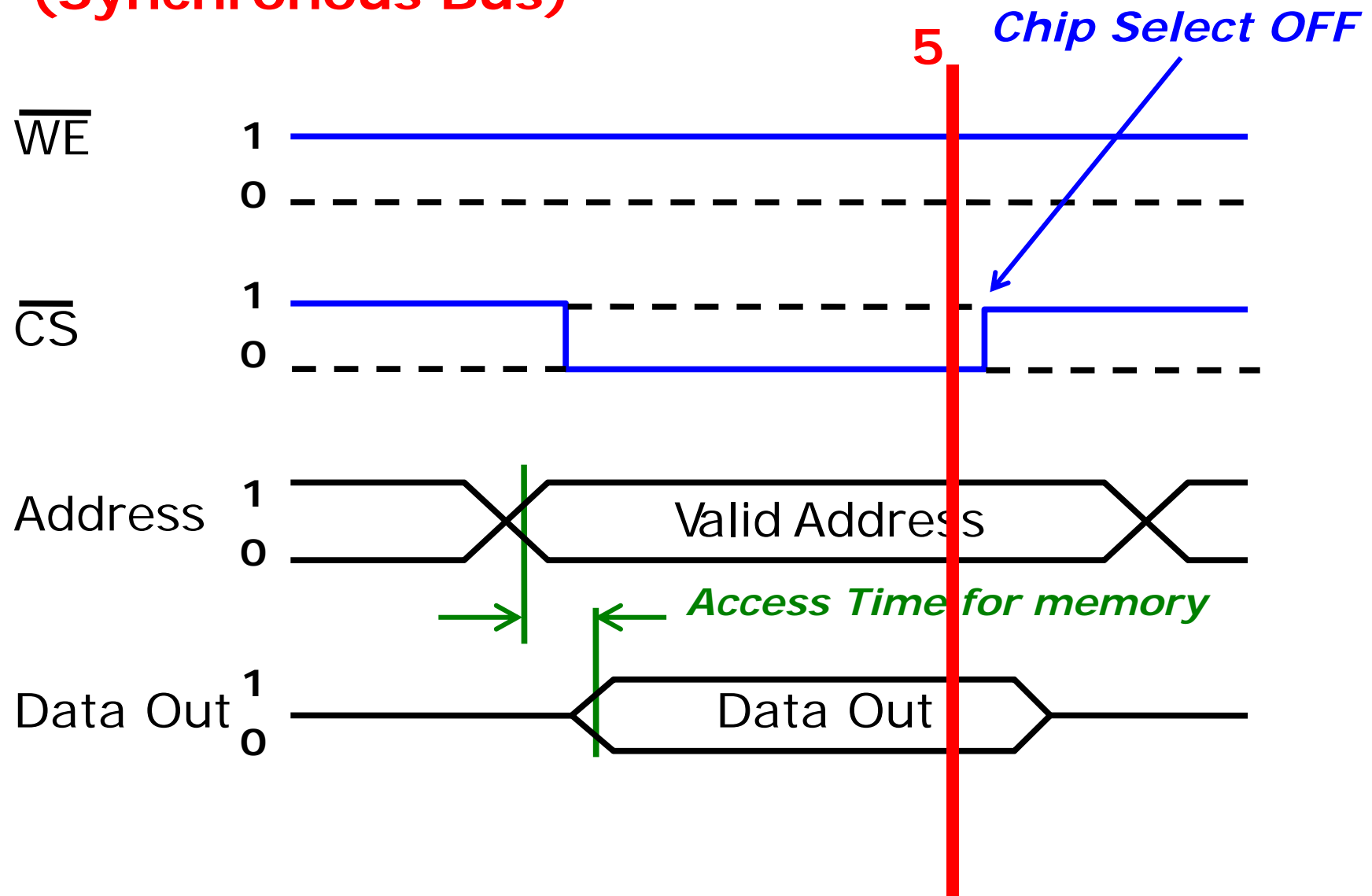
Simplified Memory Read Timing (Synchronous Bus)



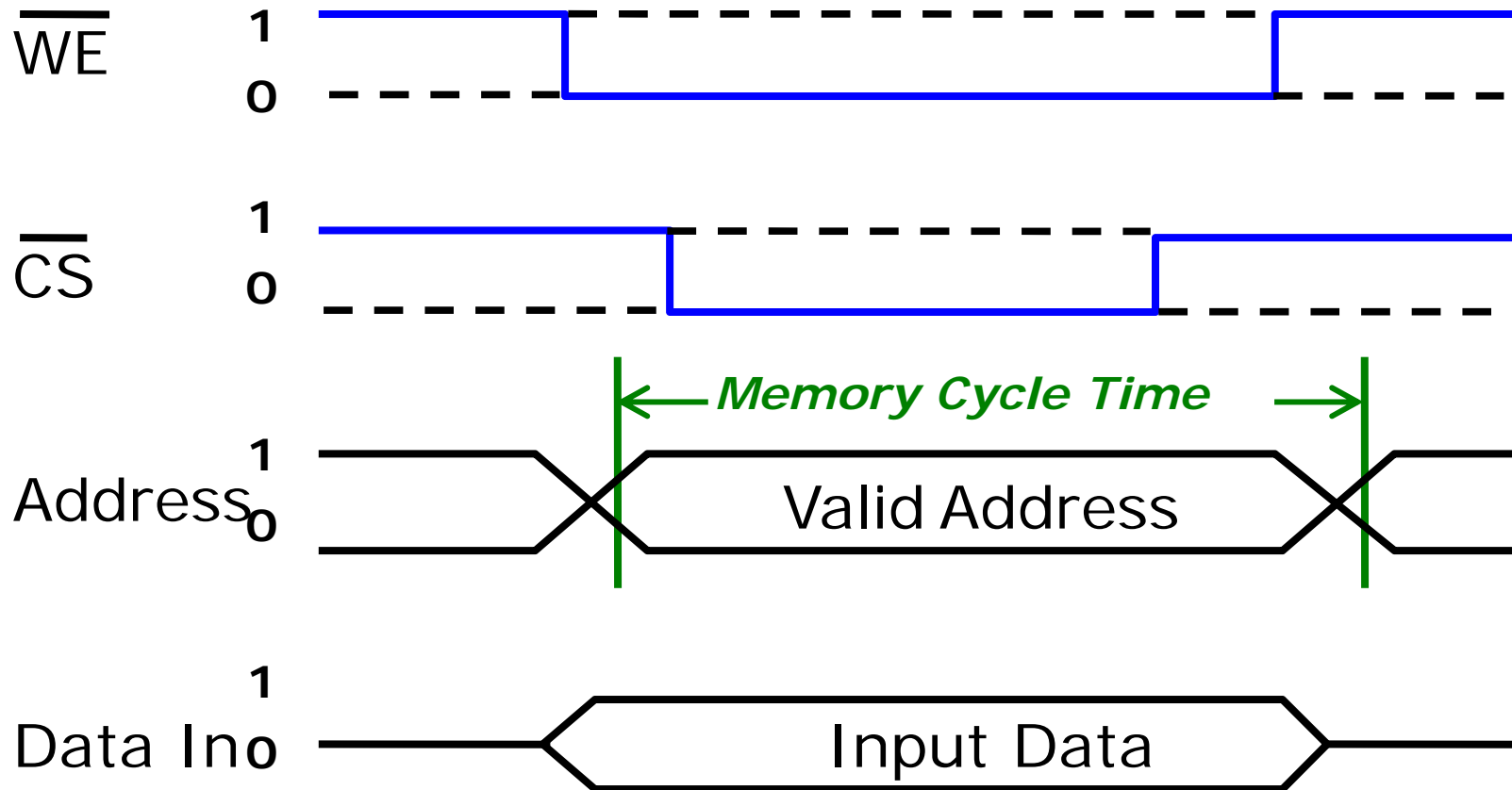
Simplified Memory Read Timing (Synchronous Bus)



Simplified Memory Read Timing (Synchronous Bus)

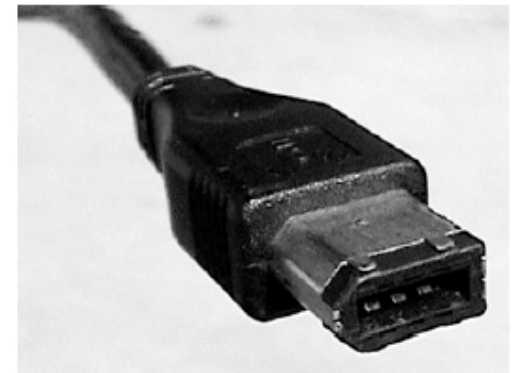


Simplified Memory Write Timing (Synchronous Bus)



USB and Firewire

- ❑ Universal Serial Bus (USB) and IEEE 1394 (Firewire) are groups of standards for interconnecting peripheral devices. USB 2.0 supports data transfer rates up to 480 Mbps, with as many as 127 devices connected to a single host controller through special hub devices in a tree-like manner.
- ❑ Firewire is similar to USB but has traditionally been faster, up to 800 Mbps. A key advantage of Firewire is isochronous data transfer, in which a continuous, guaranteed data transfer is supported at a predetermined rate. This makes Firewire attractive for digital video and digital audio.



(left) USB hub; (middle) USB cable; (right) Firewire cable.

RS-232

The RS-232 standard commonly uses 9-pin and 25-pin connectors, but uses others as well (see the figure).

RS-232 is used for slow-bit-rate devices such as mice, keyboards, and non-graphics terminals.



Examples of bus characteristics

Characteristics	Firewire (1394)	USB 2.0
Bus type	I/O	I/O
Data bus width	4	2
Clocking	asynchronous	asynchronous
Peak bandwidth (low)	50 MB/sec (400)	0.2 MB/sec
(theoretical)	or 100 MB/sec (800)	1.5 MB/sec
(full)		60 MB/sec
(high)		
Hot pluggable	Yes	Yes
Max # devices	63	127
Max bus length (copper)	4.5 m	5 m
Standard name Forum	IEEE 1394, 1394b	USER Imp.

More example data

Bus type	Bus width	Bus speed	MB/sec
ISA	16 bits	8 MHz	16
EISA	32 bits	8 MHz	32
VL-bus	32 bits	25 MHz	100
VL-bus	32 bits	33 MHz	132
PCI	32 bits	33 MHz	132
PCI	64 bits	33 MHz	264
PCI	64 bits	66 MHz	512
PCI	64 bits	133 MHz	1 GB/s

Other issues (read in textbook for details)

Bus Arbitration in general

- ❑ More than one module controlling the bus
 - e.g. CPU and DMA controller
- ❑ Only one module may control bus at one time
- ❑ Arbitration may be centralised or distributed

Centralized Bus Arbitration

- ❑ Single hardware device controlling bus access
 - Bus Controller
 - Arbiter
- ❑ May be part of CPU or separate
 - (e.g. 6811: CPU has total control of bus)

Distributed Bus Arbitration

- ❑ Each module may claim the bus
- ❑ Control logic on all modules