Computer Organization

Douglas Comer

Computer Science Department
Purdue University
250 N. University Street
West Lafayette, IN 47907-2066

http://www.cs.purdue.edu/people/comer

© Copyright 2006. All rights reserved. This document may not be reproduced by any means without written consent of the author.

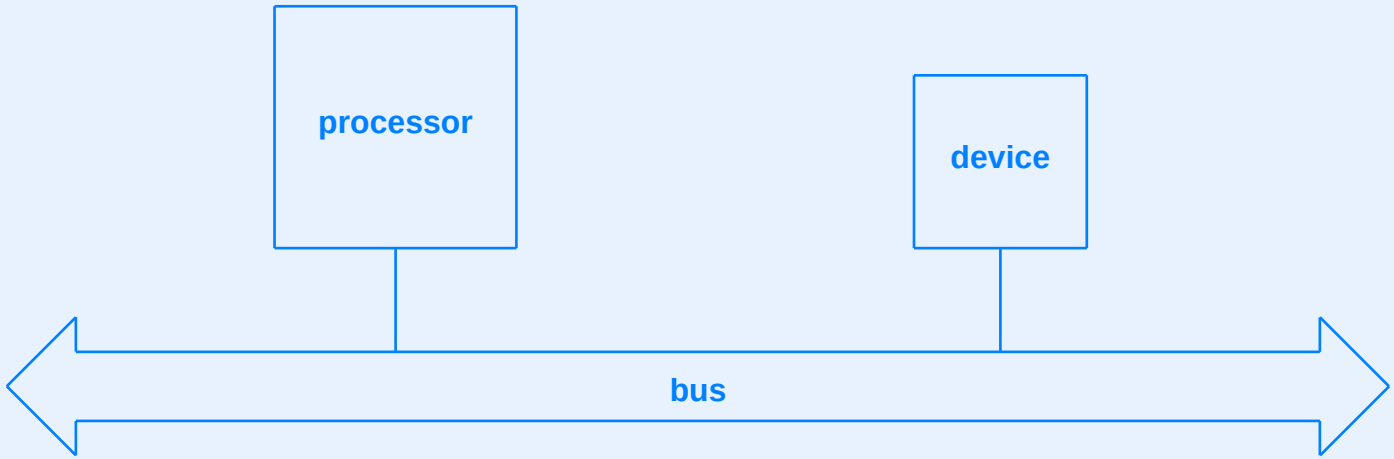
XIV

Buses
And
Bus Architecture

Definition Of A Bus

- Digital interconnection mechanism
- Allows two or more functional units to transfer data
- Typical use: connect processor to
  - Memory
  - I/O devices
- Design can be
  - Proprietary (owned by one company)
  - Standardized (available to many companies)

Illustration Of A Bus



Sharing

- Most buses shared
- Need an *access protocol*
  - Determines which device can use the bus at any time
  - All attached devices follow the protocol
- Note: can have multiple buses in one computer

Characteristics Of A Bus

- Parallel data transfer
  - Can transfer multiple bits at the same time
  - Typical width is 32 or 64 bits
- Passive
  - Bus does not contain many electronic components
  - Attached devices handle communication
- Conceptual view: think of a bus as parallel wires
- Bus may have *arbiter* that handles sharing

Physical Bus Connections

- Several possibilities
  - Wires on a circuit board
  - Sockets
  - Combinations

Illustration Of Bus On A Motherboard

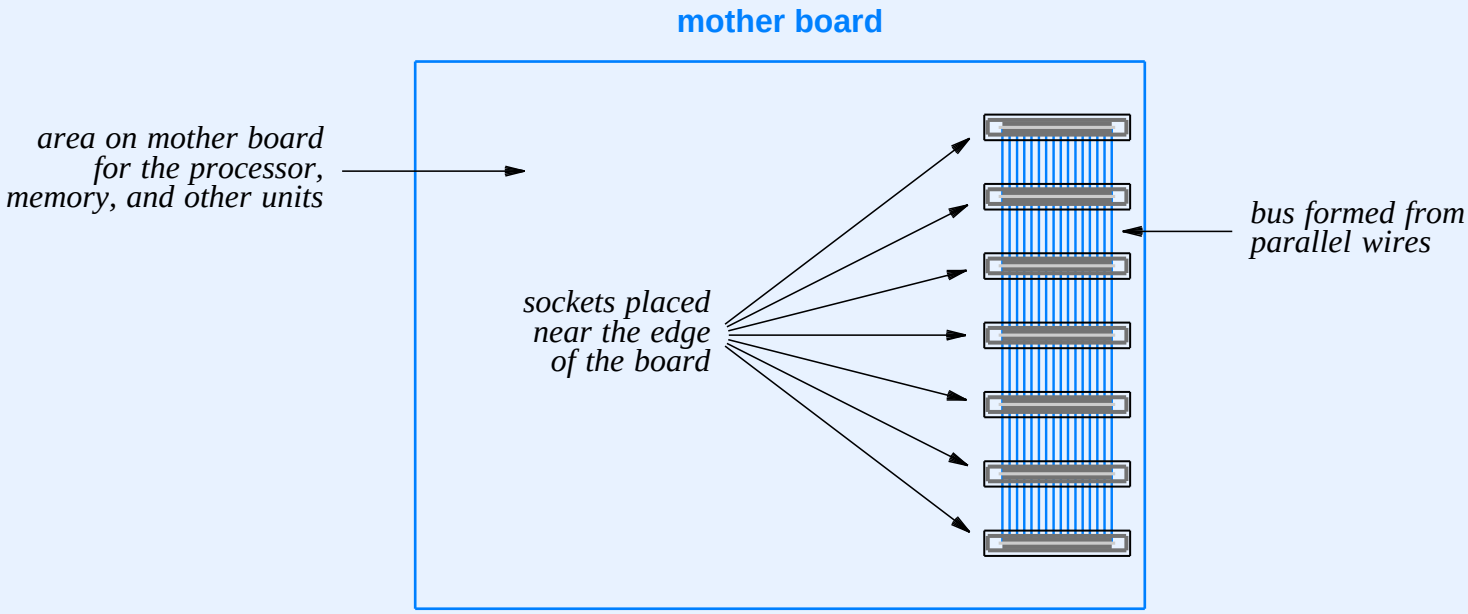
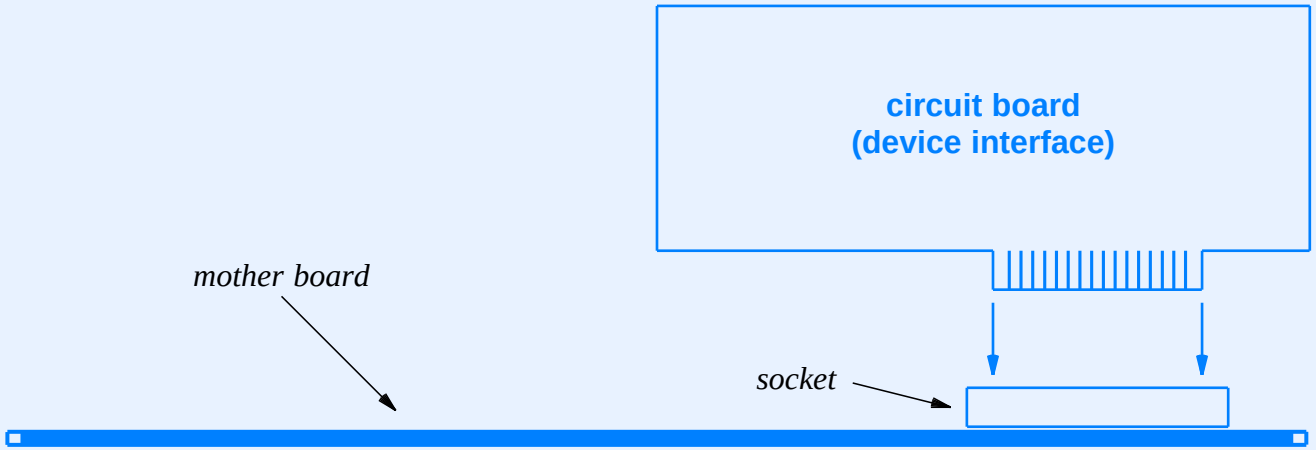


Illustration Of Circuit Board And Corresponding Sockets



Bus Interface

- Nontrivial
- Controller circuit required

Conceptual Division Of A Bus

- Need three functions
  - Control
  - Address specification
  - Data being transferred
- Conceptually three separate groups of wires (*lines*)

Illustration Of Lines In A Bus

Bus Access

- Bus only supports two operations
  - *Fetch* (also called *read*)
  - *Store* (also called *write*)
- Access paradigm known as *fetch-store paradigm*
- Obvious for memory access
- Surprise: all operations, including I/O must be performed using fetch-store paradigm

Fetch-Store Over A Bus

- Fetch
  - Place an address on the address lines
  - Use control line to signal *fetch* operation
  - Wait for control line to indicate *operation complete*
- Store
  - Place an address on the address lines
  - Place data item on the data lines
  - Use control line to signal *store* operation
  - Wait for control line to indicate *operation complete*

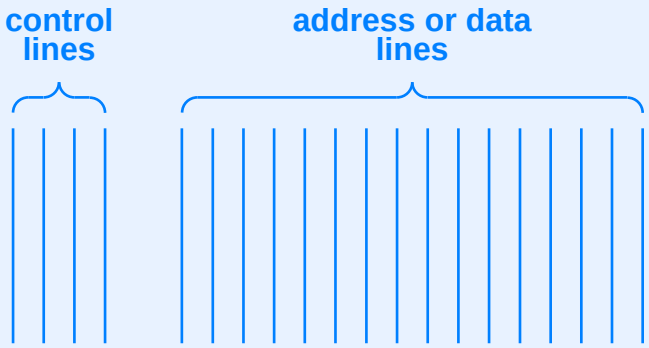
Width Of A Bus

- Larger width
  - Higher performance
  - Higher cost
  - Requires more pins
- Smaller width
  - Lower cost
  - Lower performance
  - Requires fewer pins
- Compromise: multiplex transfers to reduce width

Multiplexing

- Reuse lines for multiple purposes
- Extreme case
  - Serial bus has one line
- Typical case
  - Bus has *K* lines
  - Address can be *K* bits wide
  - Data can be *K* bits wide

Illustration Of Multiplexing On A Bus

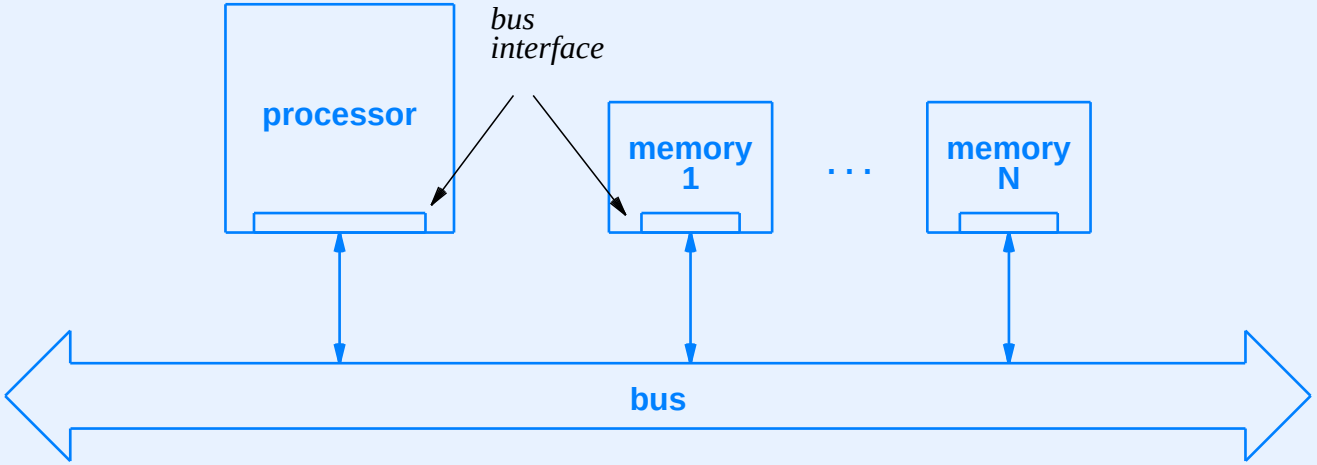


- Transfer takes longer with multiplexing
- Controller hardware is more sophisticated

Effect Of Bus Multiplexing On Design

Addresses and data values are multiplexed over a bus. To optimize performance of the hardware, an architect chooses a single size for both data items and addresses.

Illustration Of Connection To Memory Bus



- Bus defines *address space*

Control Hardware And Addresses

Although an interface receives all requests that pass across the bus, the interface only responds to requests that contain an address for which the interface has been configured.

Example Of Steps A Memory Interface Takes

```
Let R be the range of addresses assigned to the memory
Repeat forever {
    Monitor the bus until a request appears;
    if (the request specifies an address in R) {
        respond to the request
    } else {
        ignore the request
    }
}
```

Potential Errors On A Bus

- Address conflict
  - Two devices attempt to respond to a given address
- Unassigned address
  - No device responds to a given address

Address Configuration And Sockets

- Two options for address configuration
  - Configure each interface with set of addresses
  - Arrange sockets so that wiring limits each socket to a range of addresses
- Latter avoids misconfiguration: owner can plug in additional boards without configuring the hardware
- Note: some systems allow MMU to detect and configure boards automatically

Example Of Using Fetch-Store With Devices

- Imaginary status light controller
- Connected to 32-bit bus
- Contains sixteen separate lights
- Desired functions are
  - Turn the display on
  - Turn the display off
  - Set the display brightness
  - Turn the i^{th} status light on or off

Example Of Meaning Assigned To Addresses

Address	Operation	Meaning
100 – 103	store	nonzero data value turns the display on, and a zero data value turns the display off
100 – 103	fetch	returns zero if display is currently off, and nonzero if display is currently on
104 – 107	store	Change brightness. Low-order four bits of the data value specify brightness value from zero (dim) through sixteen (bright)
108 – 111	store	The low order sixteen bits each control a status light, where zero sets the corresponding light off and one sets it on.

Interpretation Of Operations

- Semantics are

```
if (address == 100 && op == store && data != 0)
        turn_on_display;
```

- And

```
if (address == 100 && op == store && data == 0)
        turn_off_display;
```

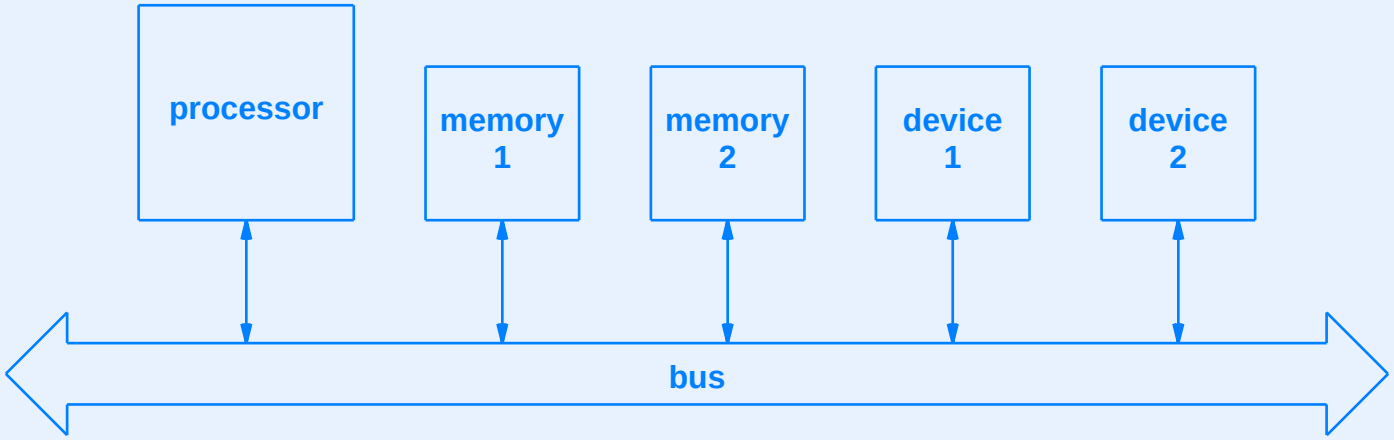
Asymmetry

- *Fetch* and *store* operations
  - Are independent
  - Need not be defined for all addresses

Unification Of Memory And Device Addressing

- Single bus can attach
  - Multiple memories
  - Multiple devices
- Bus address space includes all units

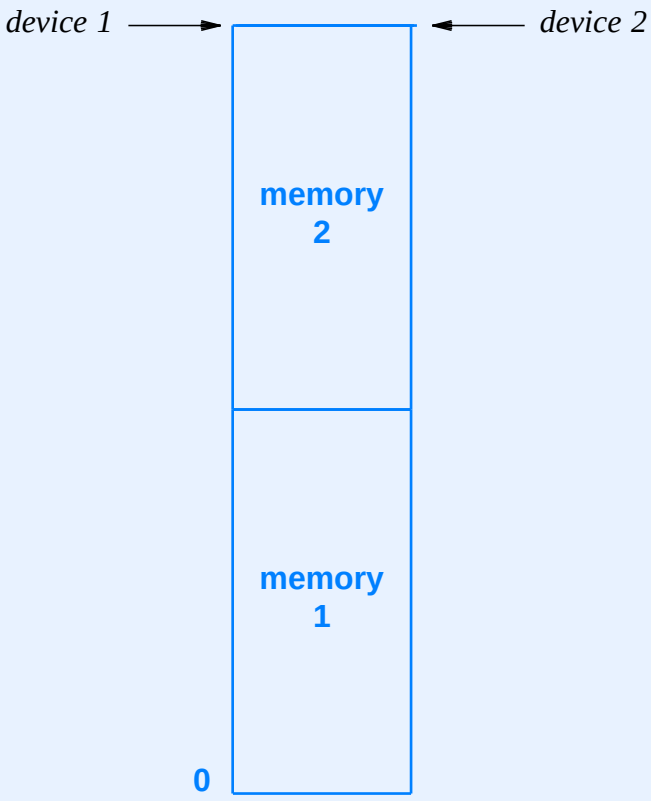
Example System With A Bus



Address Assignments For Example System

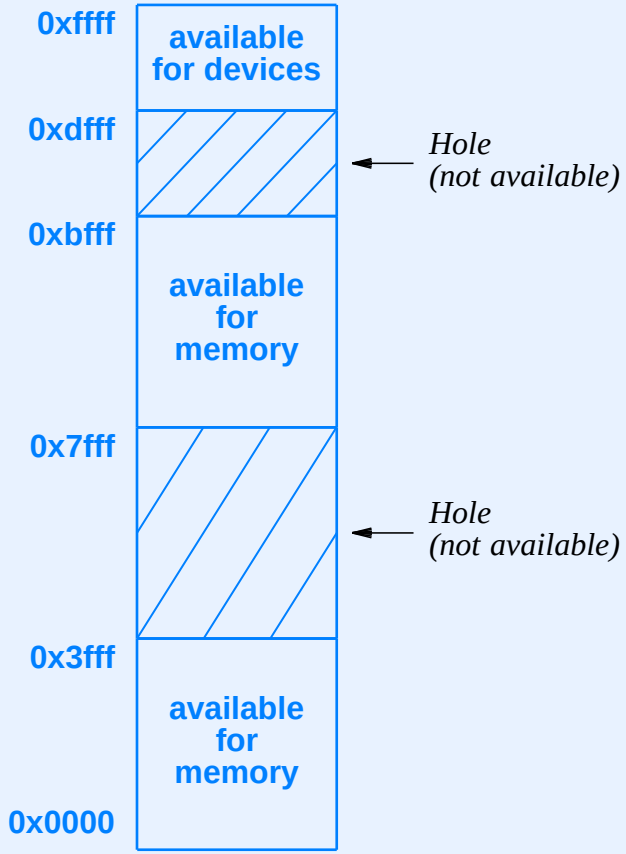
Device	Address Range		
Memory 1	0x000000	through	0x0fffff
Memory 2	0x100000	through	0x1fffff
Device 1	0x200000	through	0x20000b
Device 2	0x20000c	through	0x200017

Illustration Of Bus Address Space For Example System



- Address space may have *holes*

Example Address Map For 16-Bit Bus



A Note About Address Space

In a typical computer, the part of the address space available to devices is sparsely populated — only a small percentage of the addresses are used.

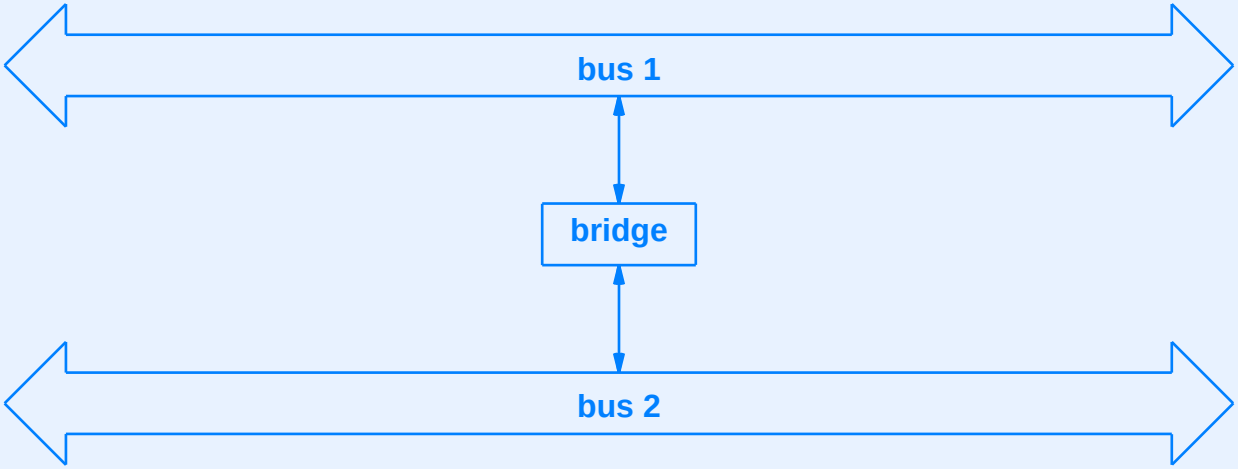
Example Code To Manipulate A Bus

```
int *p;          /* declare p to be a pointer to an integer */
p = (*int)100;   /* set pointer to address 100 */
*p = 1;          /* store nonzero value in addresses 100 - 103 */
```

A Note About Programming With Multiple Buses

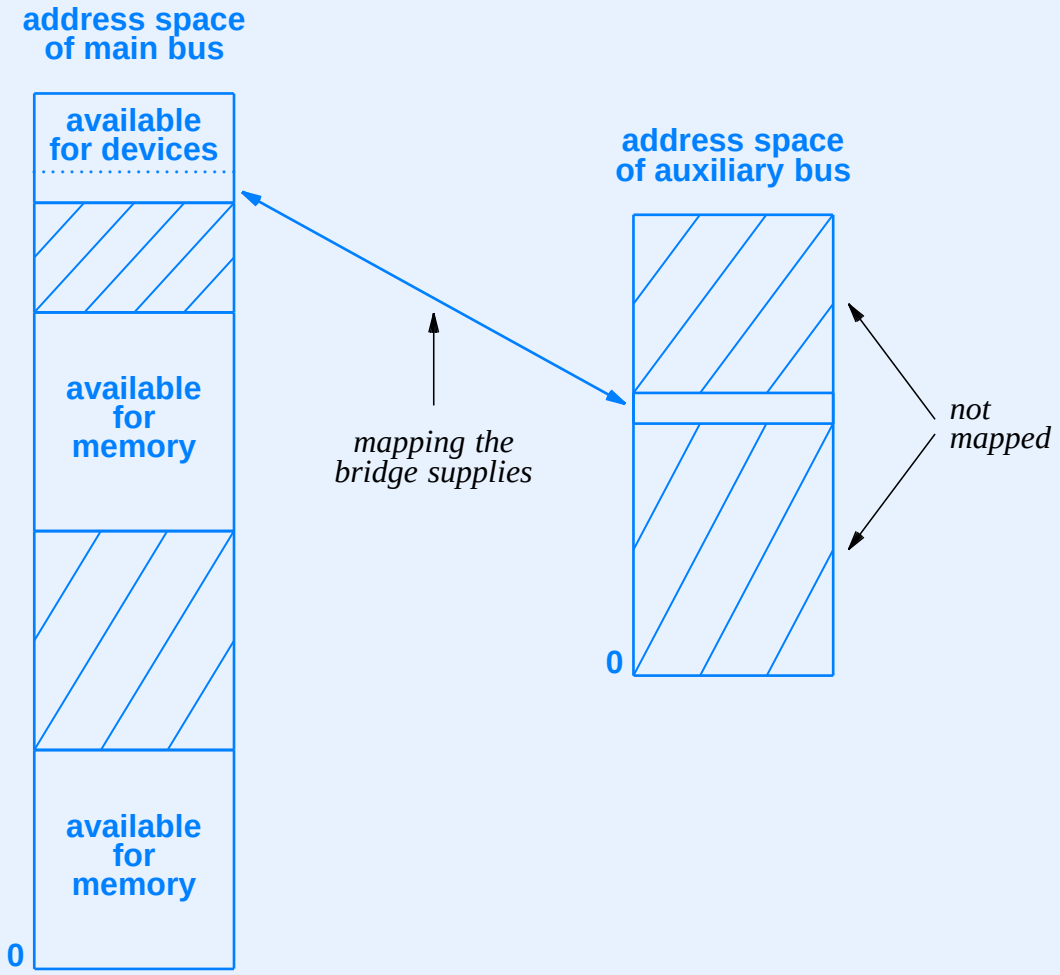
A processor that has multiple buses provides special instructions to access each; a processor that has one bus interprets normal memory operations as referencing locations in the bus address space.

Illustration Of Bridge Between Two Buses



- Interconnection device
- Maps range of addresses
- Forwards operations and replies from one bus to the other
- Especially useful for adding an auxiliary bus

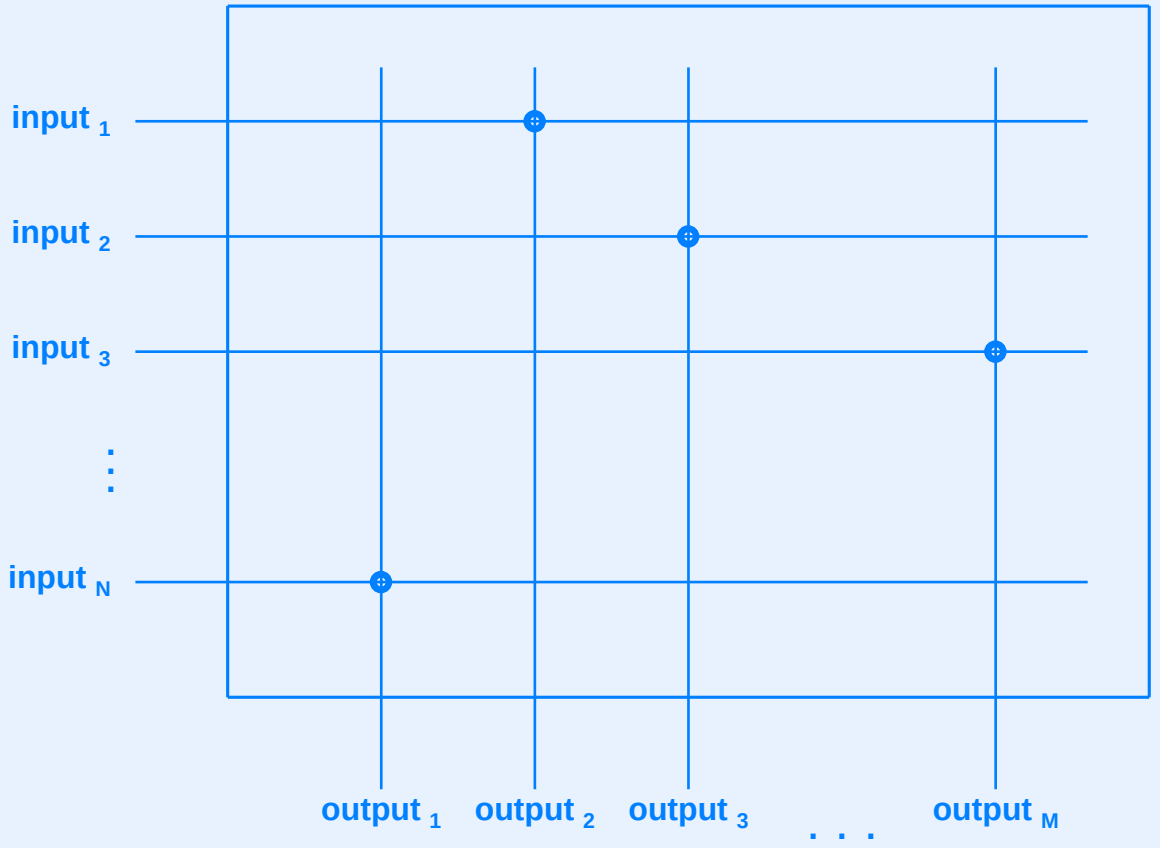
Illustration Of Address Mapping



Switching Fabric

- Alternative to bus
- Connects multiple devices
- Sender supplies data and destination device
- Fabric delivers data to specified destination

Conceptual Crossbar Fabric



- Solid dot indicates a connection

Summary

- Bus is fundamental mechanism that interconnects
  - Processor
  - Memory
  - I/O devices
- Bus uses fetch-store paradigm for all communication
- Each unit assigned set of addresses in bus address space
- Bus address space can contain holes

Summary
(continued)

- Bridge maps subset of addresses on one bus to another bus
- Programmer uses conventional memory address mechanism to communicate over a bus
- Switching fabric is alternative to bus that allows parallelism


Questions?