# Input and Output

# Chapter 14

# Bus And Bus Architectures

# Topics

- Introduction
- Definition of a Bus
- Processors, I/O devices, and Buses
- Proprietary and Standardized Buses
- Shared Buses and an Access Protocol
- Multiple Buses
- A Parallel, Passive Mechanism
- Physical Connections
- Bus Interface

# Topics

- Address, Control, and Data Lines
- The Fetch-store Paradigm
- Fetch-store over a Bus
- The Width of a Bus
- Multiplexing
- Bus Width and Size of Data Items
- Bus Address Space
- Potential Errors
- Address Configuration and Sockets

# Topics

- Many Buses or One Bus
- Using Fetch-store with Devices
- An Example of Device Control Using Fetch-store
- Operation of an Interface
- Asymmetric Assignments
- Unified Memory and Device Addressing
- Holes in the Address Space
- Address Map
- Program Interface to a Bus

# Topics

- Bridging Between Two Buses
- Main and Auxillary Buses
- Consequence for Programmers
- Summary

# Introduction

- This chapter discusses
  - external connections between processor and memory system
  - bus, a fundamental architectural feature in all computer systems
  - motivation for a bus, basic operations
  - a common bus shared between memory and I/O
  - how bus defines address space

# Definition of a Bus

- Digital communication mechanism that allows two or more functional units to transfer control signals or data

- Examples
  - memory bus connects processor and memory
  - I/O bus interconnects processor with I/O devices

# Definition of a Bus

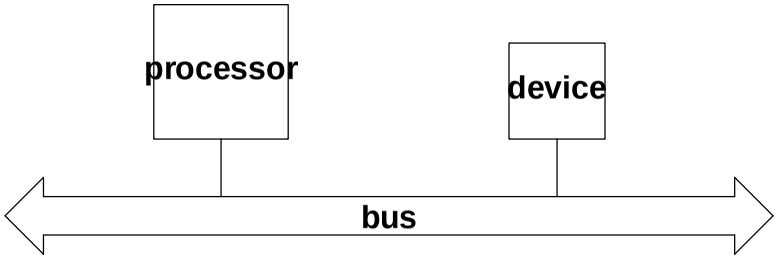



Illustration of a bus used to connect a processor an external device. Buses are used for most external connections.

*A bus is the digital communication mechanism used within a computer system to interconnect functional units. A computer contains one or more buses that interconnect the processors, memories and external I/O devices.*

# Proprietary and Standardized Buses

- Proprietary buses
  - design is owned by private company via patents etc
- Standardized buses
  - specifications available to any company
  - allows multi-vendor devices to work together
  - specifications, hardware, voltage, timing of signals, encoding, all must be strictly adhered to.

# Shared Buses and Access Protocol

- Processor can connect to a set of I/O devices, or multiple processors can connect to single shared bus

- Access protocol
  - how attached device determines if bus is available or busy
  - how devices take turn in using the bus

# Multiple Buses

- Computer Systems may contain several buses, examples
  - memory bus, I/O bus
  - special buses connecting coprocessors
- Reasons: flexibility, convenience
- Internal buses: not visible to computer owner

# Parallel Passive Mechanism

- Bus usually do not contain electronic components
- Device connecting to bus contain circuits for communication
- Bus are like parallel wires to which devices attach

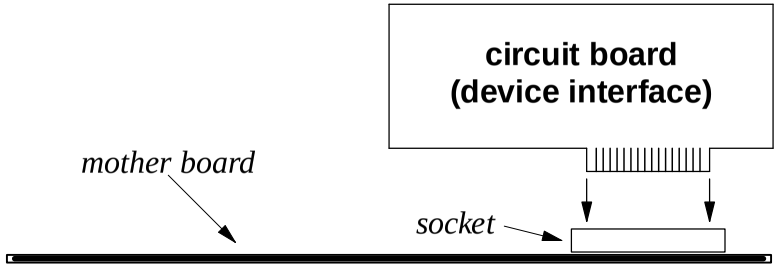
# Physical Connections

- Bus can consists of
  - tiny wires etched in silicon on a single chip
  - cable containing multiple wires
  - set of parallel metal wires on circuit board
- Computer's main circuit board is also called the mother board
  - sockets on mother board connects to the bus.
  - sockets allow devices to be plugged in or removed

# Bus Interfaces

- Bus interface or bus controller
  - additional circuit that attaches to bus and follows the bus protocol
  - device interface have metal fi ngers that touch metal contacts of socket

# Bus Interfaces




Side view of a mother board illustrating how a printed circuit board can plug directly into the socket of a bus. Metal strips on the circuit board press against metal contacts in the socket.

# Logical Structure of a Bus

- Bus wires are called lines.
- Functions of different lines are
  - control line: control of bus
  - address line: specifi cation of address information
  - data lines: transfer of data
- Lines need not be divided equally, control need fewer lines

# Fetch-Store Paradigm

*Like a memory system, a bus employs the fetch-store paradigm; all control or data transfer operations are performed as either fetch or a store*

- All lines use  fetch or store operation
- Control lines are used
  - to ensure only one pair of devices communicate over the bus at any time
  - allow two communicating devices to interact
- Address lines are used to pass address
- Data lines are used to transfer a value

# Fetch-Store Paradigm

**Fetch**

1. Use the control lines to obtain access to the bus
2. Place an address on the address lines
3. Use the control lines to request a fetch operation
4. Test the control lines to wait for the operation to complete
5. Read the value from the data lines
6. Set the control lines to allow another device to use the bus

**Store**

1. Use control lines to obtain access to the bus
2. Place an address on the address lines

3. Place a value on the data lines
4. Use the control lines to specify a store operation
4. Test the control lines to wait for the operation to complete
5. Set the control lines to allow another device to use the bus

The steps taken to perform a fetch or store operation over a bus, and the group of lines used in each step.

# Width of a Bus

- If a bus can transfer k bits of data at a time (over k lines), width of bus is k.
- Example 32 bit bus

- How wide should bus be?
  - increasing width increases throughput
  - greater width, translates to more space and electronic components
  - architect chooses compromise between space, cost and performance

# Multiplexing

- Helps in reducing the number of lines in a bus
- Data multiplexing
  - device divides data into blocks that are exactly as large as the bus is wide.
  - device uses bus repeatedly, by sending one block at a time.

# Address and Data Multiplexing

- Use same lines for address and data
- Example of address and data multiplexing
  - send address on the line and receive data on the same line later
  - store operation: send address first followed by data
- Commonly, few lines are kept for control and most lines for either address/data

# Advantages and Disadvantages of Multiplexing

- Advantages
  - architect can design a bus with fewer lines
  - higher overall performance for the same number of lines
- Disadvantages
  - more time may be needed, example store takes two bus cycles
  - more sophisticated bus protocol, therefore more complex hardware

# Bus Width and Size of Data and Address

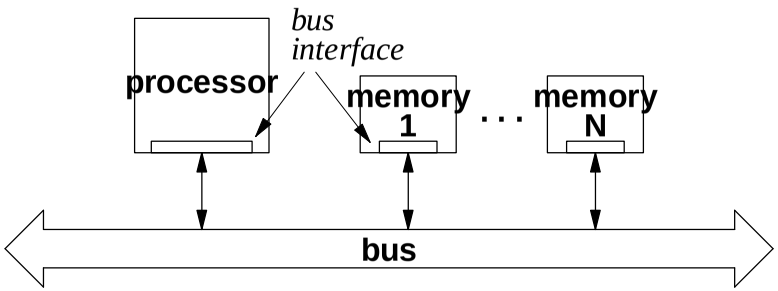
- Choose a single size for all of the following
  - bus width
  - register size
  - data value
  - address

  *Address 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.*

# Bus Address Space

- Example of a bus: memory bus
  - a devices connected to bus via interface (here processor and memories)
  - the interface implements bus protocol, handles all communication
  - the interface uses control lines to gain access to the bus, sends addresses or data values as instructed by device (processor or memory)
  - the interface alone understands bus details

# Bus Address Space




Physical interconnections of a processor and memory using a memory bus. A controller circuit in each device handles the details of bus access.

# Bus Address Space

- Programming interface for bus has only fetch and store operations
- Example
  - data to be moved from memory to register, processor issues fetch to bus interface
  - data to be moved to memory, processor issues store to the interface
- Single address space is divided among different memories. All memory controllers receive the fetch or store, but only one memory whose address ranges matches the request address responds.

# Bus Address Space

*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*

# Potential Errors

- Address conflict
  - Bus error occurs when interfaces are misconfi gured so they respond to the same address.
- Precaution: bus protocols include test for address conflicts

- Unassigned address
  - the processor attempts to access an address that has not been confi gured into any interface
- Precaution: bus protocols use timeout mechanism. If no response is received within a certain time, the processor hardware generates bus error

# Address Configuration and Sockets

- Memory is not installed to cover all addresses, infact memory bus are designed to accommodate memory expansion
- Use of special sockets helps in avoiding memory confi guration problems
- Another alternative is to allow MMU to confi gure socket addresses when the computer boots

*To avoid memory configuration problems, architects can place memory on small circuit boards that each plug into a socket on the mother board. An owner can install memory without configuring the hardware because each socket is configured with the range of addresses to which the memory should respond*

# Many Buses or One Bus?

- High performance computers have many buses, each optimized for a specifi c purpose

- Advantage of single bus
  - Lower cost and more generality
- Disadvantage
  - may not be optimal for given purpose
  - may become the bottleneck

# Using Fetch-Store with Devices

- Bus only provides transfer mechanism, not what the bit means.
- Device can interpret bits uniquely. Example bits may mean control operation, or a data transfer .

# Using Fetch-Store with Devices

- **Turn the display on**
- **Turn the display off**
- **Set the display brightness**
- **Turn the i status light on or off**

An example of functionality provided by an imaginary hardware device. Before the unit can be attached to a bus, control functions must be implemented using the fetch-store paradigm.

# Using Fetch-Store with Devices

| 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. |

Example assignment of addresses, operations, and meanings for the device control functions listed in the above figure.

# Operation of an Interface

- For a device address is just a set of bits
- Test is done to see if bits match
- If they match, interface enables circuits to a do fetch or store

# Unified Memory and Device Addressing

- A single bus provides access to both memory and I/O devices
- I/O devices have to be given unique addresses to avoid conflict
- Memory obviously needs large address space, but isn't a single address sufficient to identify a device?
  - why sixteen bytes for each device?
  - because, each address for device can be mapped to a different function.

# Unified Memory and Device Addressing

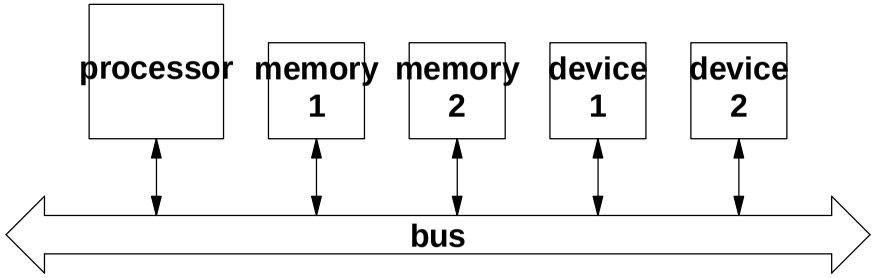



Illustration of a computer architecture that uses a single bus. Both memories and devices attach to the bus.

# Unified Memory and Device Addressing

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

One possible assignment of bus addresses for the set of devices shown in the previous fi gure.

# Holes in the Address Space

- Holes: if there are gaps in the addresses
- Typically lowest addresses reserved for memory, and higher for I/O devices

  *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*

# Program Interface to a Bus

*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.*

# Bridging Between Two Buses

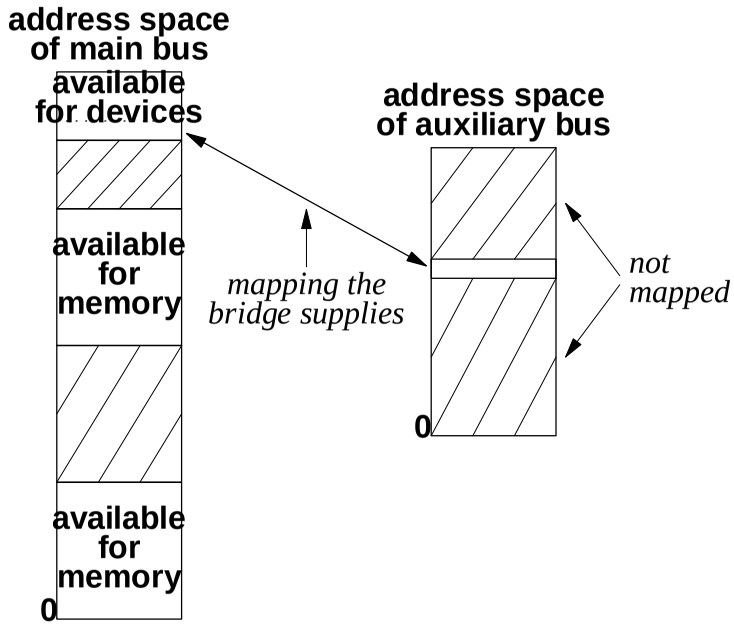
- Bridge interconnects two or more buses
- Unique range of addresses assigned for each bus
- Bridge translates address for request or reply received on one bus and forwards it to the other bus
- Bridge makes a set of address on one bus appear in the address space of another: this is called mapping

# Main and Auxillary buses

- To the processor bridge is transparent, i.e. it doesn't need to know about the auxillary bus.
- However, software must understand about the mapping

  *A bridge is a hardware device that interconnects two buses and maps addresses between them. Bridging allows a computer to have one or more auxillary buses that are accessed through the computer's main bus.*

# Main and Auxillary buses




Mapping that a bridge can provide between the address space of an auxillary bus and the address space of a main bus. Only some bus addresses need to be mapped.

# Summary

- Bus is a fundamental mechanisms to interconnect memory, I/O devices, and processors within a computer system.
- Each bus defi nes a protocol that devices use to access bus. Fetch-store paradigm forms the basis of the protocols
- Control, data and address information is passed over the bus, wires may be shared for better performance
- Bus defi nes address space which may have holes.
- Multiple buses may be connected by bridge with addresses being mapped between the multiple buses or main and auxillary bus.