Chapter 7
Input/Output
Contents

• External devices
• I/O modules
• I/O techniques
  — Programmed I/O
  — Interrupt-driven I/O
    – Interrupt processing
    – Intel 82C59A controller
  — Direct memory access
• I/O channels and processors
• The external interface: FireWire and InfiniBand
Why do we need I/O modules?

• Wide variety of peripherals
  — Impractical to incorporate the logic within the CPU

• Data transfer rate is often much slower
  — Impractical if the device is directly connected to the bus

• I/O devices often use different data formats
Generic Model of I/O Module

Address Lines

Data Lines

Control Lines

System Bus

I/O Module

Links to peripheral devices
I/O module can be called as...

• I/O channel or I/O processor
  — Takes on most of the detailed processing
  — Presents a high-level interface to CPU

• I/O controller or device controller
  — Requires quite detailed control from CPU
7.1 External Devices

• Types of external devices
  — Human readable
    - used to communicate with the user
    - video display terminals
    - keyboard
    - mouse
    - printer
  — Machine readable
    - used to communicate with electronic equipment
    - magnetic disk and tape systems
  — Communication
    - used to communicate with remote devices
    - modem
    - Network Interface Card (NIC)
7.2 I/O Modules

• Module functions
  — Control & Timing
  — CPU Communication
  — Device Communication
    - Command, status information, and data
  — Data Buffering
    - Data coming from main memory are sent to an I/O module in a rapid burst
    - Data are buffered in the I/O module and then sent to the peripheral device at its data rate
  — Error Detection
    - Mechanical errors
    - Transmission errors
Control and Timing

• Coordination of the flow of traffic between internal resources and external devices

• Example
  — CPU inquires I/O module about device status
  — I/O module returns device status
  — If device ready, CPU requests data transfer
  — I/O module gets data from the device
  — I/O module transfers data to CPU
CPU Communication

• Command decoding
  — I/O module accepts commands from CPU

• Data

• Status reporting
  — Busy and ready
  — Various error conditions

• Address recognition
  — Each I/O device has an address
Typical I/O Data Rates

- Gigabit Ethernet
- Graphics display
- Hard disk
- Ethernet
- Optical disk
- Scanner
- Laser printer
- Floppy disk
- Modem
- Mouse
- Keyboard

Data Rate (bps)
I/O Module Diagram

- Interface to System Bus
- Interface to External Device
- Data Lines
- Status/Control Registers
- Address Lines
- Control Lines
- Data Registers
- External Device Interface Logic
- I/O Logic
- External Device Interface Logic
# I/O Techniques

- Programmed I/O
- Interrupt driven I/O
- Direct Memory Access (DMA)

<table>
<thead>
<tr>
<th></th>
<th>No Interrupts</th>
<th>Use of Interrupts</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O-to-memory transfer</td>
<td>Programmed I/O</td>
<td>Interrupt-driven I/O</td>
</tr>
<tr>
<td>through processor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direct I/O-to-memory</td>
<td></td>
<td>Direct memory access (DMA)</td>
</tr>
<tr>
<td>transfer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(a) Programmed I/O

1. Issue Read command to I/O module
2. Read status of I/O module
3. Check status
   - Not ready: Error condition
   - Ready: Read word from I/O Module
4. Write word into memory
5. Done?
   - Yes: Next instruction (a) Programmed I/O
   - No: Error condition

(b) Interrupt-driven I/O

1. Issue Read command to I/O module
2. Read status of I/O module
3. Check status
4. Ready: Read word from I/O Module
5. Write word into memory
6. Done?
   - Yes: Next instruction (b) Interrupt-driven I/O
   - No: Error condition

(c) Direct memory access

1. Issue Read block command to I/O module
2. Read status of DMA module
3. Interrupt
4. DMA → CPU
5. Next instruction
6. Do something else

CPU → I/O
I/O → CPU
CPU → memory
CPU → DMA
7.3 Programmed I/O

• Overview
  — CPU executes I/O instruction
    - Issuing a command to the I/O module
  — I/O module performs the requested action
    - Set the appropriate bits in the status register
  — CPU checks status bits periodically
    - No interrupts
    - CPU waits for I/O module to complete operation
    - Wastes CPU time
I/O Commands

• CPU issues address
  — Identifies module (& device if >1 per module)

• CPU issues command
  — Control
    - used to activate a device
    - e.g. spin up disk
  — Test
    - used to check status conditions of a device
    - e.g. power on? error?
  — Read/Write
    - module transfers data via buffer from/to device
Addressing I/O Devices

• Memory-mapped I/O
  — Devices and memory share an address space
  — I/O looks just like memory read/write
  — No special commands for I/O
    - Large selection of memory access commands available

• Isolated I/O
  — Separate address spaces
  — Need I/O or memory select lines
  — Special commands for I/O
    - Limited set
### ADDRESS | INSTRUCTION | OPERAND | COMMENT
--- | --- | --- | ---
200 | Load AC | "1" | Load accumulator
200 | Store AC | 517 | Initiate keyboard read
202 | Load AC | 571 | Get status byte
202 | Branch if Sign = 0 | 202 | Loop until ready
202 | Load AC | 516 | Load data byte

(a) Memory-mapped I/O

### ADDRESS | INSTRUCTION | OPERAND | COMMENT
--- | --- | --- | ---
200 | Load I/O | 5 | Initiate keyboard read
201 | Test I/O | 5 | Check for completion
201 | Branch Not Ready | 201 | Loop until complete
201 | In | 5 | Load data byte

(b) Isolated I/O
7.4 Interrupt Driven I/O

• Problems of programmed I/O
  — Processor has to wait a long time for I/O module to be ready

• Interrupt driven I/O
  — I/O module will let processor know when it is ready
    - CPU can do other useful things in the mean time
  — Sends an interrupt when ready
Interrupt Driven I/O

• Overview
  —Point of view of the I/O module (Input)
    - receives a READ command
    - proceeds to read data from the device
    - signals an interrupt and waits a request from CPU
    - places the data on the bus and is ready for another I/O
  —Point of view of CPU (Input)
    - issues READ command
    - does something else
    - checks for interrupt at the end of each instruction cycle
    - if interrupted,
      + save context (registers) of the current program
      + process interrupt
        o fetch data & store
      + restores context of the interrupted program
(a) Interrupt occurs after instruction at location N

(b) Return from interrupt
Design Issues

• How do you deal with multiple interrupts?
  — Sequential and Nested processing

• How do you identify the module issuing the interrupt?
  — Multiple interrupt lines
    - Different line for each module
    - Limits number of devices
  — Software poll
    - CPU asks each module in turn
    - Slow
  — Daisy chain (hardware poll, vectored)
    - Interrupt Acknowledge sent down a chain
    - Module responsible places vector on bus
    - CPU uses vector to identify handler routine
  — Bus arbitration (vectored)
    - Module must claim the bus before it can raise interrupt
82C59A Interrupt Controller

- 80x86 has two pins for handling interrupts
  - one for INTR and the other for INTA
- 82C59A interrupt controller is used to handle a variety of devices
  - external devices are connected to this, which in turn connects to 80386
  - has 8 interrupt lines
  - can be cascaded to handle up to 64 modules
  - programmable
    - fully nested
    - rotating
    - special mask
Intel 82C59A Interrupt Controller

• Management of interrupts
  — 82C59A accepts interrupts from modules
  — 82C59A determines priority
  — 82C59A signals 80386 (by raising INTR line)
  — CPU acknowledges via INTA
  — 82C59A puts appropriate vector on data bus
  — CPU processes the interrupt
Intel 82C55A
Programmable Peripheral Interface

- An example I/O module
  - Single chip, general-purpose module
  - External interface
    - 24 I/O lines
    - divided into three 8-bit groups
    - group C can be subdivided into 4-bit groups
      + used in conjunction with A and B ports
  - Internal interface
    - 8-bit data bus
    - two address lines
    - CHIP SELECT line
    - READ and WRITE lines
    - RESET line
Intel 82C55A
Programmable Peripheral Interface

(a) Block diagram

(b) Pin layout

- PA3
- PA2
- PA1
- PA0
- Read
- Chip select
- Ground
- A1
- A0
- PC7
- PC6
- PC5
- PC4
- PC3
- PC2
- PC1
- PC0
- PB7
- PB6
- PB5
- PB4
- PB3
- PB2
- PB1
- PB0
- V
- D7
- D6
- D5
- D4
- D3
- D2
- D1
- PA4
- PA5
- PA6
- PA7
- D0
- Reset
- Write
- +5 volts
- ground
- 8086 data bus
- power supplies
- address A0
- lines A1
- read
- write
- reset
- chip select
- 8-bit internal bus
- Data buffer
- Control logic
- Control register
- Data buffers
- A
- CA
- CB
- B
Using 82C55A To Control Keyboard/Display
7.5 Direct Memory Access

- Interrupt driven and programmed I/O require active CPU intervention
  - Transfer rate is limited
  - CPU is tied up in managing an I/O transfer
- When large volumes of data are to be moved, DMA is the answer
- DMA module
  - Connected to system bus
  - Transfer data to and from main memory
Direct Memory Access

• CPU issues a command to DMA module
  — Read or write?
  — Address of the I/O device involved
  — Starting location in memory
  — Number of words to be read or written

• CPU continues with other work
  — An interrupt is sent when the task is complete
  — CPU is involved only at the beginning and end of the transfer
Typical DMA Block Diagram
DMA and Interrupt Breakpoints
• Single Bus, Detached DMA controller
• Each transfer uses bus twice
  — I/O to DMA then DMA to memory
• CPU is suspended twice
DMA Configurations (2)

- Single Bus, Integrated DMA controller
- Controller may support >1 device
- Each transfer uses bus once
  - DMA to memory
- CPU is suspended once
DMA Configurations (3)

• Separate I/O Bus
  — Easily expandable
• Each transfer uses bus once
  — DMA to memory
• CPU is suspended once
7.6 I/O Channels and Processors

• Evolution of I/O function
  — CPU directly controls an external device
  — I/O controller is added, programmed I/O is used
  — I/O controller is added, interrupt-driven I/O is used
  — DMA is used
  — I/O module is a processor in its own right
    - can fetch and execute I/O instructions without CPU intervention
    - referred to as an I/O channel
  — I/O module is a processor with its own local memory
    - a large set of I/O devices can be controlled with minimal CPU involvement
    - referred to as an I/O processor
Characteristics of I/O Channels

• Operation
  — CPU initiates an I/O transfer by instructing the I/O channel to execute a program in memory
  — Program specify the device, the area of memory, priority, and other actions
  — I/O channel follows these instructions and controls the data transfer

• Types of I/O channels
  — Selector channel
    - dedicated, at any one time, to the transfer of data with one of the devices
  — Multiplexor channel
    - can handle I/O with multiple devices at the same time
    - byte or block multiplexor
7.7 External Interface

• Types of interfaces
  — Parallel interface
  — Serial interface

• Dialogue with the peripheral (writing)
  — I/O module sends a control signal requesting permission to send data
  — Peripheral acknowledges the request
  — I/O module transfers data
  — Peripheral acknowledges receipt of data

• I/O module has an internal buffer
  — Compensate for the differences in speed between the system bus and external devices
Parallel and Serial I/O

(a) Parallel I/O

(b) Serial I/O
External Interface

• Connection
  – Point-to-point
    - dedicated line between I/O module and external device
  – Multipoint
    - external buses actually
    - used to support external mass storage devices and multimedia devices
    - FireWire and InfiniBand
FireWire Serial Bus

- IEEE 1394 for a high performance serial bus
- Fast, low cost, and easy to implement
- Uses serial transmission
  - SCSI uses parallel transmission
  - Disadvantages of parallel transmission
    - require more wires, so more expensive cables and connectors
    - require shielding to prevent interference between wires
    - require synchronization between wires
FireWire Configuration

• Daisy chain
  — Up to 63 devices off a single port
• Provides hot plugging
  — Possible to connect peripherals when system is on
• Provides automatic configuration
  — No need to manually set device IDs
• May be tree structured
Simple FireWire Configuration
FireWire

• Three layers of protocols
  — Physical
    - Defines the transmission media, electrical and signaling characteristics
  — Link
    - Describes the transmission of data in packets
  — Transaction
    - Defines a request-response protocol that hides the lower-layer details from applications
FireWire Protocol Stack

Transaction Layer (read, write, lock)

asynchronous

Link Layer
- Packet transmitter
- Packet receiver
- Cycle control

Serial Bus Management

Isochronous

Link Layer
- Arbitration
- Data resynch
- Encode/decode

- Connectors/media
- Connection state
- Signal levels
FireWire - Physical Layer

• Specifies possible transmission media
  — Data rates from 25 to 400 Mbps
• Converts binary data into electrical signals
• Types of arbitration
  — First come first served
    - natural priority controls simultaneous requests
      + who is nearer to root and who has lower ID number
  — Fair arbitration
    - time is organized into fairness intervals
    - one who gained access in one interval may not again compete during this interval
  — Urgent arbitration
    - devices may be configured as having urgent priority
FireWire - Link Layer

• Two transmission types
  — Asynchronous
    - Variable amount of data and several bytes of transaction layer information transferred as a packet
    - Acknowledgement returned
    - Used when there is no fixed data rate requirements
  — Isochronous
    - Variable amount of data transferred in sequence of fixed size packets at regular intervals
    - No acknowledgement
    - Used for digital sound or video
FireWire - Link Layer

• Typical asynchronous transaction
  — Arbitration sequence
    - exchange of signals to acquire control of bus
  — Packet transmission
    - header + data
  — Acknowledgment gap
    - time delay for the destination to generate an acknowledgment
  — Acknowledgment
    - recipient returns an acknowledgment packet
  — Subaction gap
    - ensuring that other nodes on the bus do not begin arbitrating before acknowledgment packet has been transmitted
(a) Example asynchronous subaction

(b) Concatenated asynchronous subactions

(c) Example isochronous subactions
InfiniBand

• I/O specification aimed at high end servers
  — Result of the merger of two competing projects: Future I/O (Cisco, HP, Compaq, IBM) and Next Generation I/O (Intel)

• Architecture and specifications for data flow between processor and intelligent I/O devices
  — Intended to replace PCI bus in servers

• Increased capacity, expandability, flexibility
InfiniBand Architecture

• Devices are attached in a central fabric of switches and links

• No need to have the basic I/O interface hardware inside the server chassis
  — Independent nodes are added as required

• I/O distance from server up to
  — 17 m using copper
  — 300 m using multimode optical fiber
  — 10 km using single mode optical fiber

• Transmission rates up to 30 Gbps
InfiniBand Architecture

• Key elements
  — Host channel adapter (HCA)
    - links the server to InfiniBand switch
  — Target channel adapter (TCA)
    - links other devices to InfiniBand switch
  — InfiniBand switch
    - provides point-to-point physical connections between devices
  — Links
  — Subnet
    - switch plus the links that connect the devices
  — Router
    - connects InfiniBand subnets to a network
InfiniBand Switch Fabric

HCA = host channel adapter
TCA = target channel adapter
InfiniBand Operation

• Each physical link supports 16 logical channels, called virtual lanes
  — One lane for management, rest for data
  — Virtual lane temporarily dedicated to end-to-end transfer

• Data sent in stream of packets

• Layered protocol architecture is used
  — Physical: defines 3 link speeds and physical media
  — Link: defines packet structure; sets up the virtual links
  — Network
    - routes packets between different subnets
  — Transport: reliability mechanisms for end to end transfer
Figure 7.21  Infiniband Communication Protocol Stack

WQE = work queue element  
CQE = completion queue entry  
QP = queue pair
# InfiniBand Links and Data Rates

<table>
<thead>
<tr>
<th>Link</th>
<th>Signal rate (unidirectional)</th>
<th>Usable capacity (80% of signal rate)</th>
<th>Effective data throughput (send + receive)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1- wide</td>
<td>2.5 Gbps</td>
<td>2 Gbps (250 MBps)</td>
<td>(250 + 250) MBps</td>
</tr>
<tr>
<td>4-wide</td>
<td>10 Gbps</td>
<td>8 Gbps (1 GBps)</td>
<td>(1 + 1) GBps</td>
</tr>
<tr>
<td>12-wide</td>
<td>30 Gbps</td>
<td>24 Gbps (3 GBps)</td>
<td>(3 + 3) Gbps</td>
</tr>
</tbody>
</table>