Chapter 3
A Top-Level View of Computer Function and Interconnection
Contents

• Computer components
• Computer function
  — Instruction fetch and execute
  — Interrupts & I/O function
• Interconnection structures
• Bus interconnection
  — Bus structure
  — Multiple bus hierarchies
  — Elements of bus design
• PCI
  — Bus structure
  — PCI commands
  — Data transfers
  — Arbitration
Key points

• Instruction cycle
  — instruction fetch
  — zero or more operand fetches
  — execution
  — zero or more operand stores
  — interrupt check

• Major computer components need to be interconnected
  — in most cases, a bus is used

• Key design issues for buses
  — arbitration, timing, and width
Components

• CPU
  — Control unit
  — Arithmetic and Logic unit

• Main memory
  — Temporary storage for code and results

• I/O devices
  — Data and instructions need to get into the system and results out
Computer Components: Top Level View

CPU
- PC
- IR
- Execution unit
- I/O AR
- I/O BR

System Bus

Main Memory
- Instruction
- Instruction
- Instruction
- Data
- Data
- Data
- Data

PC = Program counter
IR = Instruction register
MAR = Memory address register
MBR = Memory buffer register
I/O AR = Input/output address register
I/O BR = Input/output buffer register
Instruction Cycle

- Two steps
  - Fetch
  - Execute
Fetch Cycle

- Program Counter (PC) holds address of next instruction to fetch
- Processor fetches instruction from memory location pointed to by PC
- Increment PC
  - Unless told otherwise
- Instruction loaded into Instruction Register (IR)
- Processor interprets instruction and performs required actions in execute cycle
Execute Cycle

• Interpret & Perform
  — Processor-memory
    - data transfer between CPU and main memory
  — Processor-I/O
    - Data transfer between CPU and I/O module
  — Data processing
    - Some arithmetic or logical operation on data
  — Control
    - Alteration of sequence of operations
    - e.g. jump
  — Combination of above
### (a) Instruction format

<table>
<thead>
<tr>
<th>0</th>
<th>3</th>
<th>4</th>
<th>15</th>
</tr>
</thead>
</table>

### (b) Integer format

Program counter (PC) = Address of instruction  
Instruction register (IR) = Instruction being executed  
Accumulator (AC) = Temporary storage

- 0001 = Load AC from memory  
- 0010 = Store AC to memory  
- 0101 = Add to AC from memory

Figure 3.4 Characteristics of a Hypothetical Machine
Example of Program Execution

Step 1

Step 2

Step 3

Step 4

Step 5

Step 6
Instruction Cycle - State Diagram

Instruction fetch
Instruction address calculation
Instruction operation decoding
Operand fetch
Operand address calculation
Operand address calculation
Data Operation
Operand store
Multiple operands
Multiple results
Instruction complete, fetch next instruction
Return for string or vector data
Interrupts

- Mechanism by which other modules (e.g. I/O) may interrupt normal sequence of processing

- Classes of interrupts
  - Program
    - e.g. overflow, division by zero
  - Timer
    - Generated by internal processor timer
    - Used in pre-emptive multi-tasking
  - I/O
    - from I/O controller
  - Hardware failure
    - e.g. power failure, memory parity error
Program Flow Control

(a) No interrupts

(b) Interrupts; short I/O wait

(c) Interrupts; long I/O wait
**Interrupts and Instruction Cycle**

- **Advantages of using interrupts**
  - do not wait for the actual I/O operation to complete

- **Revised instruction cycle**
  - interrupt cycle is added
  - Processor checks for interrupt
    - Indicated by an interrupt signal
  - If no interrupt, fetch next instruction
  - If interrupt pending
    - Suspend execution of current program
    - Save context
    - Set PC to start address of interrupt handler routine
    - Process interrupt
    - Restore context and continue interrupted program
Transfer of Control via Interrupts

User Program

Interrupt Handler

Interrupt occurs here
Instruction Cycle with Interrupts

START

Fetch Cycle
Fetch Next Instruction

Execute Cycle
Execute Instruction

Interrupt Cycle
Check for Interrupt; Process Interrupt

HALT

Interrupts Disabled

Interrupts Enabled
Program Timing - Short I/O Wait

(a) Without interrupts

(b) With interrupts
Program Timing - Long I/O Wait

(a) Without interrupts

(b) With interrupts
Instruction Cycle (with Interrupts) - State Diagram
Multiple Interrupts

• Multiple interrupts can occur concurrently
• Two approaches
  — Sequential processing
    - Processor will ignore further interrupts while processing one interrupt
    - Interrupts remain pending and are checked after first interrupt has been processed
    - Interrupts handled in sequence as they occur
  — Nested processing
    - Low priority interrupts can be interrupted by higher priority interrupts
    - When higher priority interrupt has been processed, processor returns to previous interrupt
Multiple Interrupts - Sequential
Multiple Interrupts – Nested
Connecting Components

• All the components must be connected
• Different type of connection for different type of component
  — Memory
  — Input/Output
  — CPU
Computer Components

Memory
- Read
- Write
- Address
- Data

$I/O$ Module
- Read
- Write
- Address
- Internal Data
- External Data

CPU
- Instructions
- Data
- Interrupt Signals
- Address
- Control Signals
- Data
Memory Connection

• Receives and sends data
• Receives addresses (of locations)
• Receives control signals
  — Read
  — Write
  — Timing
Input/Output Connection (1)

- Similar to memory from computer’s viewpoint
- Output
  - Receive data from computer
  - Send data to peripheral
- Input
  - Receive data from peripheral
  - Send data to computer
Input/Output Connection(2)

• Receive control signals from computer
• Send control signals to peripherals
  — e.g. spin disk
• Receive addresses from computer
  — e.g. port number to identify peripheral
• Send interrupt signals (control)
CPU Connection

- Reads instruction and data
- Writes out data (after processing)
- Sends control signals to other units
- Receives (& acts on) interrupts
What is a Bus?

• A communication pathway connecting two or more devices
  — shared among devices
• Usually broadcast
  — only one unit at a time can transmit over the bus
• Often grouped
  — Address information (address bus)
  — Data information (data bus)
  — Control information (control bus)
  — Miscellaneous
    - power, ground, clock,...
Address bus

• Identify the source or destination of data
  — e.g. CPU needs to read an instruction (data) from a given location in memory

• Bus width determines maximum memory capacity of system
  — e.g. 8080 has 16 bit address bus giving 64K address space
Data Bus

• Carries data
  — Remember that there is no difference between “data” and “instruction” at this level

• Width is a key determinant of performance
  — 8, 16, 32, 64 bit
Control Bus

• Control and timing signals are carried
  — Memory read/write
  — I/O read/write
  — Transfer ACK
    - indicates that data have been accepted from or placed on the bus
  — Bus request/grant
  — Interrupt request/ACK
  — Clock
    - used to synchronize operations
  — Reset
    - initialize all modules
Bus Interconnection Scheme
Single Bus Problems

• Lots of devices on one bus leads to
  — Greater propagation delays
    - affect performance
  — Bus may become a bottleneck as the data transfer demand approaches bus capacity

• Most systems use multiple buses to overcome these problems
Multiple Buses in a System

• Generally laid out in a hierarchy
  — bridge/interface between buses

• High-speed buses closer to the processor
  — high-capacity I/O devices are attached here

• Lower-speed buses farther away from the processor
  — low-capacity devices are attached here
Traditional Bus Architecture
High Performance Bus

Processor → Local Bus → Cache/Bridge → System Bus → Main Memory

SCSI → FireWire → Graphic → Video → LAN

High-Speed Bus

FAX → Expansion bus interface → Modem → Serial

Expansion Bus
Bus Design Elements

- **Type**
  - Dedicated
  - Multiplexed

- **Method of arbitration**
  - Centralized
  - Distributed

- **Timing**
  - Synchronous
  - Asynchronous

- **Bus width**
  - Address
  - Data

- **Data transfer type**
  - Read
  - Write
  - Read-modify-write
  - Read-after-write
  - Block
Bus Types

• Dedicated
  — Separate data & address lines

• Multiplexed
  — Shared lines
    - using same lines for multiple purposes
    - time multiplexing
  — Address valid or data valid control line is needed

— Advantage
  - Fewer lines, save space and cost

— Disadvantages
  - More complex control
  - Potential reduction in performance
Bus Arbitration

• Several modules may need control of the bus
  — e.g. CPU and DMA controller
• Only one module may control bus at one time
  — need some method of arbitration
• Arbitration
  — centralized
  — distributed
Centralized Arbitration

• Single hardware device controlling bus access
  — Bus Controller or Arbiter

• May be a separate module or part of CPU
Distributed Arbitration

- No central controller
  - Each module contains access control logic
  - They act together to share the bus
Timing

• Refers to the way in which events are coordinated on the bus
  — Synchronous
  — Asynchronous
Synchronous Timing

- Events are determined by clock signals
- Clock signal
  - Bus has a clock line
  - 1s and 0s of equal duration
  - A single 1-0 transmission is called a clock(bus) cycle
  - All devices can read clock line
  - All events start at the beginning of a clock cycle
  - Most events occupy a single clock cycle
Synchronous Timing

• An example (case of reading)
  — during the 1st clock cycle
    - CPU places a memory address
    - CPU issues an address enable signal
  — during the 2nd clock cycle
    - CPU issues a read command
  — during the 3rd clock cycle
    - memory module places the data on the data lines
Synchronous Timing Diagram

- Clock
- Status lines
- Address lines
- Address enable
- Read cycle
  - Data lines
  - Read
- Write cycle
  - Data lines
  - Write

- Status signals
- Stable address
- Valid data in
- Valid data out
Asynchronous Timing

• The occurrence of one event on a bus follows and depends on the occurrence of a previous event

• An example (case of reading)
  — CPU places address signals on the bus
  — CPU issues a read command
  — Memory module responds by placing the data on the data line
    - acknowledged line is asserted to signal CPU that data are available
  — After reading data, read signal is deasserted
    - this causes the memory module to drop the data and acknowledge lines
  — CPU removes address information
Asynchronous Timing – Read Diagram

- Status lines
  - Status signals

- Address lines
  - Stable address

- Read

- Data lines
  - Valid data

- Acknowledge
Comparisons of Timing Methods

• Synchronous timing is simpler
  — but all devices are tied to a fixed clock rate
• Asynchronous timing is more flexible
  — a mixture of slow and fast devices can share the bus
Bus Width

• The wider the data bus, the greater the number of bits transferred at one time
• The wider the address bus, the greater the range of locations that can be referenced
Bus Data Transfer Types

- Data types
  - read and write
    - multiplexed or non-multiplexed
  - read-modify-write
    - a read followed immediately by a write to the same address
    - the whole operation is typically indivisible
    - used to protect the shared memory resources in a multiprogramming system
  - read-after-write
    - indivisible operation
    - used for checking purposes
  - block data transfer
    - one address cycle is followed by n data cycles
Bus Data Transfer Types (1)

<table>
<thead>
<tr>
<th>Time</th>
<th>Address (1st cycle)</th>
<th>Address (2nd cycle)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Write (multiplexed) operation</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data and address sent by master in same cycle over separate bus lines.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Address</th>
<th>Access time</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Read (multiplexed) operation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Address</th>
<th>Data read</th>
<th>Data write</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Read-modify-write operation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Read (non-multiplexed) operation</td>
<td></td>
</tr>
</tbody>
</table>
## Bus Data Transfer Types (2)

<table>
<thead>
<tr>
<th>Address</th>
<th>Data write</th>
<th>Data read</th>
</tr>
</thead>
</table>

Read-after-write operation

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
<th>Data</th>
<th>Data</th>
</tr>
</thead>
</table>

Block data transfer
PC Buses

- ISA
- Micro Channel Architecture
- EISA
- VESA Video Local Bus
- Futurebus+
- PCI
ISA Bus

- Industrial Standard Architecture
  - First open system bus architecture for PCs
  - 8-bit and 16-bit ISA buses
- 8-bit bus
  - first used in the PC-XT
  - 62 pins
  - 4.77 MHz clock
  - 20 address lines
    - 1 MB addressable memory
  - 8 data lines
  - 6 interrupt lines, 2 DMA channels
ISA Bus

- 16-bit bus
  - 8-bit bus was very limiting
  - 16-bit bus introduces with the PC-AT and 80286
  - augmented the existing 8-bit bus’ 62-pin connector with a 36-pin connector
  - 8.33 MHz clock
  - total of 24 address lines
    - 16MB addressable space
  - 16 data lines
  - 5 more interrupt lines and 4 more DMA channels
ISA Bus

- CPU
- Memory
- Bus drivers logic
- Keyboard
- ISA Bus
- Bus slots
MCA Bus

• Micro Channel Architecture
  — Introduction of 386 and 486 processors put a strain on the performance of ISA bus
    - slow to pass 32-bit data words in 2 bus operations
  — IBM wanted to put ISA to rest and introduced the MCA in their PS/2 series of machines (late 80s)
  — Offered many improvements over ISA
    - higher speed
    - bus arbitration
    - automatic configuration
  — 16 and 32-bit implementations
EISA Bus

- Extended ISA
  - Introduced in 1988-89 to provide enhancements to the ISA bus
  - 16/32-bit data
  - 24/32-bit address
  - 8.33 MHz clock
  - Backward compatible with ISA equipment
  - Roughly twice the data throughput of ISA
  - More interrupts and DMA channels
  - Never really caught on
    - viewed as a bus for high-end machines
VESA Bus

• VESA Video Local Bus
  — Video Electronics Standards Assoc.
  — Give video and graphics peripherals quick access to main memory
  — Implemented in conjunction with ISA or EISA for support of other peripherals
  — 32/64-bit data, 24/32-bit address
VESA Bus

- CPU
- Memory
- Bus drivers logic
- Keyboard
- VL Bus slots
- ISA Bus
- Bus slots
Futurebus+

• High-performance asynchronous bus
  — Introduced in the late 80s
  — Architecture, processor, and technology independent
  — Support
    - fault-tolerant and high-reliability systems
    - cache-based memory
  — Has the potential to supplant other buses because of its flexibility
  — Flexibility comes at a higher implementation cost than PCI bus
    - would appeal to a different target user
PCI Bus

• Peripheral Component Interconnection
  — designed by Intel in 1990
    - all patents are released to the public domain
  — high-bandwidth, processor-independent bus
  — uses synchronous timing and centralized arbitration

• Specification
  — up to 64 data lines at 66 MHz
    - 528 MB/s
  — designed to meet the I/O requirements of modern systems
    - require few chips to implement
    - support other buses attached to PCI bus
Typical Desktop PCI System

- Processor
  - Bridge/memory controller
  - Cache
  - DRAM

PCI Bus

- LAN
- SCSI
- Expansion bus bridge
- Base I/O devices
- Expansion Bus

- Audio
- Motion Video
- Graphics
PCI Bus Structure 1

- May be configured as 32- or 64-bit bus
- PCI Bus Lines (mandatory)
  - Systems lines
    - Including clock and reset
  - Address & Data lines
    - 32 time multiplexed lines for address/data
  - Interface Control lines
    - control the timing of transactions & provide coordination among initiators and targets
  - Arbitration lines
    - Not shared
    - Each PCI master has its own pair of arbitration lines that connect it directly to PCI bus arbiter
  - Error lines
PCI Bus Structure 2

- PCI Bus Lines (optional)
  - Interrupt lines
    - Not shared
  - Cache support lines
  - 64-bit bus extension lines
    - Additional 32 lines
    - Time multiplexed
    - 2 lines to enable devices to agree to use 64-bit transfer
  - JTAG/Boundary scan lines
    - For testing procedures
PCI Commands 1

• Bus activity
  — transactions between initiator (master) and target
  — master determines the type of transaction

• Commands
  — interrupt acknowledge
  — special cycle
  — I/O read/write
  — memory read, read line, read multiple
  — memory write, write and invalidate
  — configuration read/write
  — dual address cycle
PCI Commands 2

• Interrupt acknowledge
  — read command intended for the device that functions as an interrupt controller

• Special cycle
  — used to broadcast a message to one or more targets

• I/O read/write
  — used to transfer data between master and I/O controller

• Memory read/write
  — used to specify the transfer of a burst of data

• Configuration read/write
  — enable master to read and update configuration parameters in a device

• Dual address cycle
  — used by master to indicate that it is using 64-bit addressing
Data Transfers

• Every data transfer is a single transaction
  — one address phase + one or more data phases

• An example (read)
  — Event a
    - begin transaction by asserting FRAME
      + this line is asserted until master is ready to complete
    - put the start address on the address bus and read command on the C/BE lines
  — Event b
    - at clock 2, target recognizes its address
Data Transfers

— Event c
  - master ceases driving AD bus
  - master designates which AD lines are to be used
  - changes the information on C/BE lines
  - master asserts IRDY to indicate that it is ready

— Event d
  - selected target asserts DEVSEL to indicate that it will respond
  - target places requested data and asserts TRDY

— Event e
  - master reads the data at clock 4 and changes the byte enable lines as needed in preparation for the next read
Data Transfers

— Event f
  - target needs some time to prepare the next block of data
  - target deasserts TRDY to indicate that there will not be new data during the coming cycle
  - the block of data is read at beginning of clock 6

— Event g
  - during clock 6, target places third data on the bus
  - but master is not ready, so deasserts IRDY
  - target maintain the third data for an extra clock cycle

— Event h
  - master deasserts FRAME
  - master asserts IRDY

— Event i
  - master deasserts IRDY, returning the bus to idle state
  - target deasserts TRDY and DEVSEL
PCI Read Operation
Arbitration

- Centralized, synchronous arbitration
- Each master has a unique request and grant signal
  - signal lines are attached to central arbiter
  - simple request-grant handshake is used
- Master must arbitrate for each transaction
PCI Bus Arbiter
An Example of Arbitration

— Event a
  - Before clock 1, A has asserted REQ signal
  - Arbiter samples this signal at the beginning of clock 1

— Event b
  - During clock 1, B asserts REQ signal

— Event c
  - At the same time, arbiter asserts GNT-A

— Event d
  - A samples GNT-A at the beginning of clock 2
  - A finds IRDY and TRDY deasserted
    + bus is idle
  - A asserts FRAME and places address and command
  - A continues to assert REQ-A
    + it has a second transaction to perform
An Example of Arbitration

— Event e
  - arbiter samples all REQ lines at clock 3
    + makes a decision to grant the bus to B
    + asserts GNT-B and deasserts GNT-A

— Event f
  - A deasserts FRAME
  - A puts data and asserts IRDY
  - target reads the data at the beginning of next clock cycle

— Event g
  - At the beginning of clock 5, B finds IRDY and FRAME deasserted
  - B asserts FRAME
  - B deasserts REQ
    + because it wants to perform one transaction
PCI Bus Arbitration