Input/Output Subsystems

CS6133 - Computer Architecture I

Vikram Padman

Polytechnic Institute of New York University

vikram@poly.edu

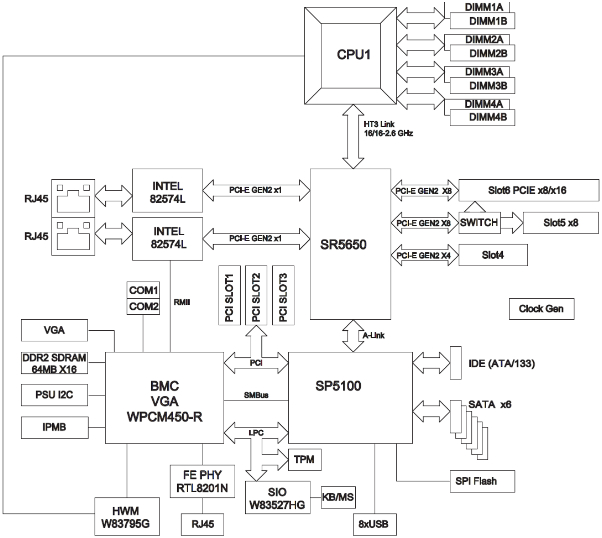
Agenda

1. Introduction
2. Addressing
3. Methodology
    - Programmed
    - Interrupt Driven
    - Direct Memory Access (DMA)
4. Data Transfer
    - Synchronous
    - Asynchronous
5. Interconnects
    - System or Shared BUS
    - Point-to-Point
        - Parallel
        - Serial
6. Performance

Input/Output Subsystem

- CPU's interface to the external world
- Plays an important role in overall system's performance
  - Most application interact with external world
  - Has a profound effect on the CPI
- Essentially, there are three types of Input/Output devices
  - **Human Machine Interface - HMI** devices
    - Keyboard, mouse, joystick, monitors, Google Glass ..etc
  - **Secondary Storage** devices
    - SATA/SAS hard drives, flash disk, SDHC card ..etc
  - **Communication** devices
    - Network Interface Cards, WiFi cards, Modems ...etc

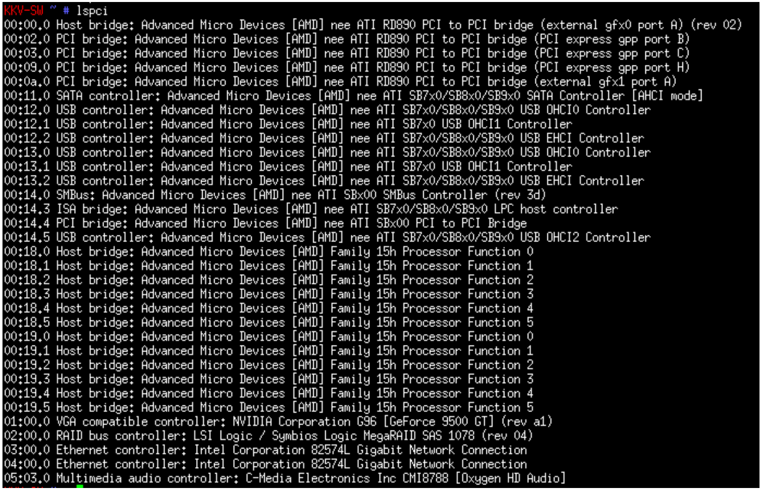
Real World System



Input/Output Addressing

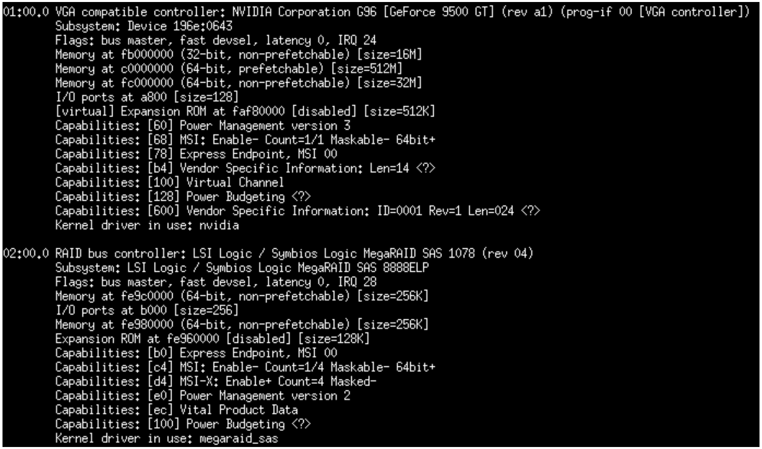
1. **I/O Address Space**
   - A unique address space that could only be accessed by special instruction
   - CPU identifies each I/O using a unique address
   - Uses “IN” or “OUT” instruction to move data
   - Special I/O instruction could only be executed by root user or OS
2. **Memory Mapped I/O**
   - Uses general memory address that could be accessed by any instructions.
   - CPU treats an I/O device as memory and contents of memory mapped I/O device are cache-able, unless stated otherwise
   - Access protection could be changed by root or system user

Input/Output Addressing



```
KKV-SW ~ # lspci
00:00.0 Host bridge: Advanced Micro Devices [AMD] nee ATI RD890 PCI to PCI bridge (external gfx0 port A) (rev 02)
00:02.0 PCI bridge: Advanced Micro Devices [AMD] nee ATI RD890 PCI to PCI bridge (PCI express gpp port B)
00:03.0 PCI bridge: Advanced Micro Devices [AMD] nee ATI RD890 PCI to PCI bridge (PCI express gpp port C)
00:09.0 PCI bridge: Advanced Micro Devices [AMD] nee ATI RD890 PCI to PCI bridge (PCI express gpp port H)
00:0a.0 PCI bridge: Advanced Micro Devices [AMD] nee ATI RD890 PCI to PCI bridge (external gfx1 port A)
00:11.0 SATA controller: Advanced Micro Devices [AMD] nee ATI SB7x0/SB8x0/SB9x0 SATA Controller [AHCI mode]
00:12.0 USB controller: Advanced Micro Devices [AMD] nee ATI SB7x0/SB8x0/SB9x0 USB OHCI0 Controller
00:12.1 USB controller: Advanced Micro Devices [AMD] nee ATI SB7x0 USB OHCI1 Controller
00:12.2 USB controller: Advanced Micro Devices [AMD] nee ATI SB7x0/SB8x0/SB9x0 USB EHCI Controller
00:13.0 USB controller: Advanced Micro Devices [AMD] nee ATI SB7x0/SB8x0/SB9x0 USB OHCI0 Controller
00:13.1 USB controller: Advanced Micro Devices [AMD] nee ATI SB7x0 USB OHCI1 Controller
00:13.2 USB controller: Advanced Micro Devices [AMD] nee ATI SB7x0/SB8x0/SB9x0 USB EHCI Controller
00:14.0 SMBus: Advanced Micro Devices [AMD] nee ATI SBx00 SMBus Controller (rev 3d)
00:14.3 ISA bridge: Advanced Micro Devices [AMD] nee ATI SB7x0/SB8x0/SB9x0 LPC host controller
00:14.4 PCI bridge: Advanced Micro Devices [AMD] nee ATI SBx00 PCI to PCI Bridge
00:14.5 USB controller: Advanced Micro Devices [AMD] nee ATI SB7x0/SB8x0/SB9x0 USB OHCI2 Controller
00:18.0 Host bridge: Advanced Micro Devices [AMD] Family 15h Processor Function 0
00:18.1 Host bridge: Advanced Micro Devices [AMD] Family 15h Processor Function 1
00:18.2 Host bridge: Advanced Micro Devices [AMD] Family 15h Processor Function 2
00:18.3 Host bridge: Advanced Micro Devices [AMD] Family 15h Processor Function 3
00:18.4 Host bridge: Advanced Micro Devices [AMD] Family 15h Processor Function 4
00:18.5 Host bridge: Advanced Micro Devices [AMD] Family 15h Processor Function 5
00:19.0 Host bridge: Advanced Micro Devices [AMD] Family 15h Processor Function 0
00:19.1 Host bridge: Advanced Micro Devices [AMD] Family 15h Processor Function 1
00:19.2 Host bridge: Advanced Micro Devices [AMD] Family 15h Processor Function 2
00:19.3 Host bridge: Advanced Micro Devices [AMD] Family 15h Processor Function 3
00:19.4 Host bridge: Advanced Micro Devices [AMD] Family 15h Processor Function 4
00:19.5 Host bridge: Advanced Micro Devices [AMD] Family 15h Processor Function 5
01:00.0 VGA compatible controller: NVIDIA Corporation G96 [GeForce 9500 GT] (rev a1)
02:00.0 RAID bus controller: LSI Logic / Symbios Logic MegaRAID SAS 1078 (rev 04)
03:00.0 Ethernet controller: Intel Corporation 82574L Gigabit Network Connection
04:00.0 Ethernet controller: Intel Corporation 82574L Gigabit Network Connection
05:03.0 Multimedia audio controller: C-Media Electronics Inc CMI8788 [Oxygen HD Audio]
```

Input/Output Addressing



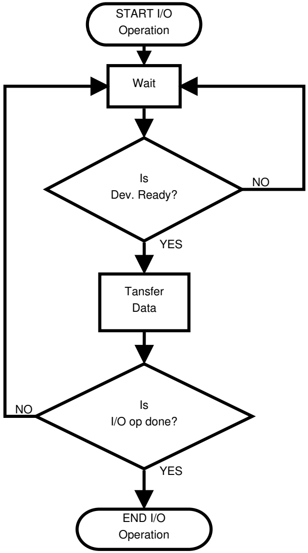
```
01:00.0 VGA compatible controller: NVIDIA Corporation G96 [GeForce 9500 GT] (rev a1) (prog-if 00 [VGA controller])
        Subsystem: Device 196e:0643
        Flags: bus master, fast devsel, latency 0, IRQ 24
        Memory at fb000000 (32-bit, non-prefetchable) [size=16M]
        Memory at c0000000 (64-bit, prefetchable) [size=512M]
        Memory at fc000000 (64-bit, non-prefetchable) [size=32M]
        I/O ports at a800 [size=128]
        [virtual] Expansion ROM at faf80000 [disabled] [size=512K]
        Capabilities: [60] Power Management version 3
        Capabilities: [68] MSI: Enable- Count=1/1 Maskable- 64bit+
        Capabilities: [78] Express Endpoint, MSI 00
        Capabilities: [b4] Vendor Specific Information: Len=14 <?>
        Capabilities: [100] Virtual Channel
        Capabilities: [128] Power Budgeting <?>
        Capabilities: [600] Vendor Specific Information: ID=0001 Rev=1 Len=024 <?>
        Kernel driver in use: nvidia

02:00.0 RAID bus controller: LSI Logic / Symbios Logic MegaRAID SAS 1078 (rev 04)
        Subsystem: LSI Logic / Symbios Logic MegaRAID SAS 8888ELP
        Flags: bus master, fast devsel, latency 0, IRQ 28
        Memory at fe9c0000 (64-bit, non-prefetchable) [size=256K]
        I/O ports at b000 [size=256]
        Memory at fe980000 (64-bit, non-prefetchable) [size=256K]
        Expansion ROM at fe960000 [disabled] [size=128K]
        Capabilities: [b0] Express Endpoint, MSI 00
        Capabilities: [c4] MSI: Enable- Count=1/4 Maskable- 64bit+
        Capabilities: [d4] MSI-X: Enable+ Count=4 Masked-
        Capabilities: [e0] Power Management version 2
        Capabilities: [ec] Vital Product Data
        Capabilities: [100] Power Budgeting <?>
        Kernel driver in use: megaraid_sas
```

Programmed I/O

- Commonly known as Polling I/O
  1. The CPU queries for the readiness of the I/O
  2. When the device is ready CPU transfers one or more data units
  3. Waits for an acknowledgement
  4. Goes back to step 1 to transfer more data
- CPU is in total control of the I/O device
- Busy wait loops are used in code until device is ready
- Many real time systems prefer or require programmed I/O method

Programmed I/O



- Programmed I/O, in general, is not efficient
- CPU spends considerable amount of time waiting
- CPU is in greater control, so PIO method is preferred in some real time systems

Programmed I/O

- **Periodic Polling** CPU periodically checks to see if the device is ready for I/O
- **Busy or Spin Wait** CPU stays in a busy wait loop and continuously checks for I/O device's readiness

Interrupt Drive I/O

- Preferred I/O method in modern computing systems
  1. CPU sends a message to I/O device and proceed to execute a different thread or process
  2. I/O device interrupts' the CPU when it is ready
  3. CPU stops current execution, switches back to I/O process and performs I/O operation
- Traditionally, interrupts are hard wire connecting the CPU and I/O device. Modern systems use message signalled interrupt **MSI**
- An interrupt causes the CPU to jump to interrupt service routine (ISR) to service the I/O device
- Data transfers are done by CPU or using DMA

Direct Memory Access

- Used in conjunction with Interrupt driven I/O method
- A section of memory is allocated to each I/O device
  1. CPU sends a message to I/O device for data transfer
  2. I/O device transfers data to/from the memory allocated to it
  3. When I/O device completes requested I/O operation, it interrupts CPU to signal completion
- Memory marked for DMA are not cache-able, unless its contents cannot be modified by the device

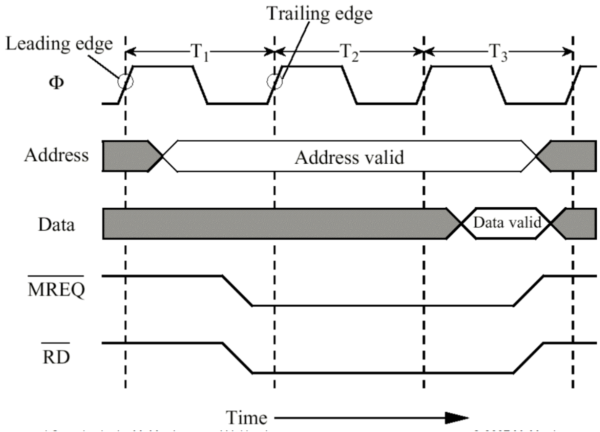
Synchronous vs Asynchronous Data Transfer

- There are two methods to transfer data between the CPU and I/O device
  1. **Synchronous Data Transfer**
     - Data is transferred relative to an I/O clock
     - I/O clock is a multiple of CPU's internal clock
     - Transfer speed and timing is fixed and does not change
  2. **Asynchronous Data Transfer**
     - Dedicated signal(s) are used to control data transfer
     - Validity of data is indicated by a "data valid" signal
     - Transfer speed and timing could vary

Synchronous Data Transfer

- I/O clock is phase aligned to CPU's clock
  - Clock could be $1/n * CPU_{clk}$ where n is a whole number
  - Dedicated wires are used to carry clock signal
  - Transfer protocol is defined with reference to I/O clock signal
- Advantages
  - Very fast, high performance I/O devices are synchronous devices
  - Transfer protocols are simple
- Disadvantages
  - Distance between CPU and I/O must be short due to clock skew
  - In a shared BUS environment all devices must run at the same speed

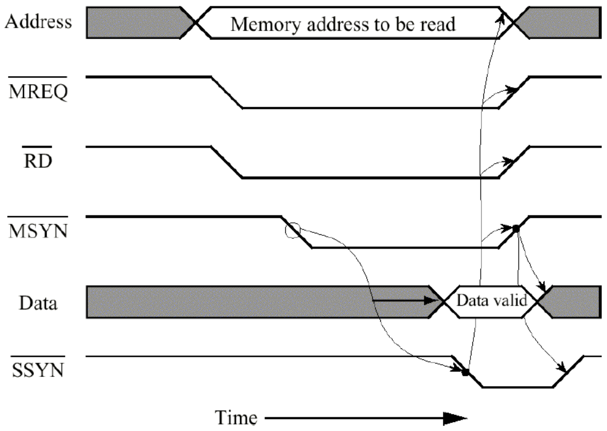
Synchronous Data Transfer



Asynchronous Data Transfer

- Clocks are not used
- Transfers are controlled by handshaking protocols using dedicated wires
- Advantages
  - Distances between CPU and I/O device could be large (meters)
  - Devices sharing the physical medium could run at different speed
- Disadvantages
  - Handshaking protocols could be complex
  - Usually slow due to complexity
  - Latencies could be undeterministic

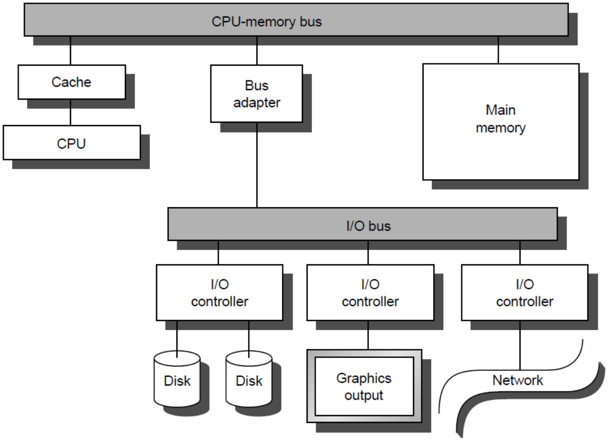
Asynchronous Data Transfer



BUS

- **BUS** A shared communication link between multiple devices
    - Physical wires connecting multiple devices
- Advantages
    - BUS implementations are cheaper
    - Easy to add or remove devices
- Disadvantages
    - Has to be relatively short due to electrical problems
    - Bus shew, crosstalk and dispersion are common
    - BUS contention could arise due to sharing

BUS


CPU-memory bus
Cache
CPU
Bus adapter
Main memory
I/O bus
I/O controller
I/O controller
I/O controller
Disk
Disk
Graphics output
Network

Introduction to High Speed Serial Communication

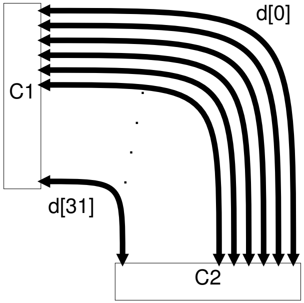
Why talk about High Speed Serial Communication?

- It is the future
- Used in all modern computing systems. Example: SAS/SATA, PCIe, Intel's QPI, USB, ThunderBolt, AMD's HyperTransport, SRIO, HDMI ... etc
- Easy to manufacture, scalable and very flexible
- In most cases it uses only four wires to transfer giga bits of data in a second
- Supported by many chip and PCB manufactures.
- Most high speed serial protocols support automatic speed throttling, error correction and on-board diagnostics.

Parallel vs Serial Communication

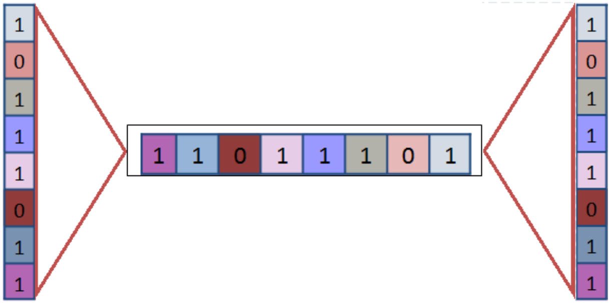
Parallel Communication

- Number of connections
- Data synchronization and alignment
- Resistance to interference
- Chip package size
- Speed limitations

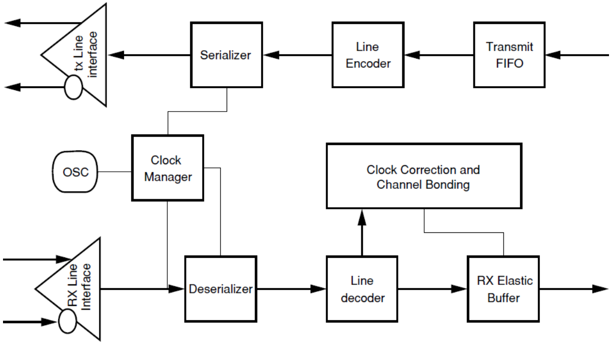


Parallel vs Serial Communication

Serial Communication



SERializer DE-Serializer



Performance

- **Throughput / Bandwidth**
  - Total data exchanged between the processor/memory and the I/O device within a unit time
- **Latency / Response Time**
  - Total elapsed time to transfer one data unit
  - An important measure for real time systems where consistent latency is important
- High throughput, low latency I/O device are desired by many applications

Week 13 Activity 1

Read Chapter 1 to 4 from "High-Speed Serial I/O Made Simple - Xilinx" [1] (on-line book) and answer the following questions

1. What is LVDS and when should it be used. What are the disadvantages of using LVDS?
2. What is the caveat in using Clock and Data Recovery(CDR)? Are there any serial protocols that does not use CDR?
3. Why is 8b/10b encoding used for serial communication? and what is the overhead in using 8b/10b encoding.
4. What is the difference between 8b/10b and 64b/66b encoding?
5. What is an "Eye Patterns"?
6. What is a "Scrambler" and which popular serial protocol uses it and why?

[1] http://www.xilinx.com/publications/archives/books/serialio.pdf