ADVANCED MICROPROCESSOR ARCHITECTURE

Dr. Ahmed Khattab
EECE Department
Cairo University

ELC 3030

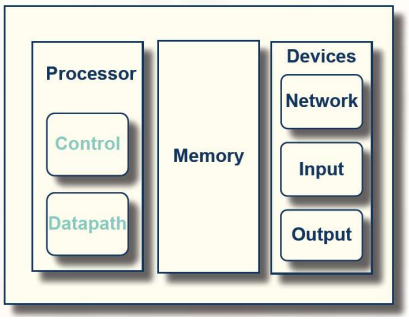
Computer Architecture

- "Computer architecture, like other architecture, is the art of determining the needs of the user of a structure and then designing to meet those needs as effectively as possible within economic and technological constraints."
  - F.P. Brooks, Planning a Computer System, Project Stretch, 1962

Computer Architecture

- Design components:
  - Organizational principles for processors, memory and I/O devices
    - Processors - microarchitecture
    - Overall - system design
  - A “programming” interface for software
    - Instructions
    - State changes
- Goals:
  - Meet functional and performance targets
  - Within constraints, such as cost and power
  - While taking advantage of advances in technology
- Architecture is about making tradeoffs

Components of a Computer



- Example: Desktop design target
  - Processor: 25% of cost
  - Memory: 25% of cost
  - Rest (I/O devices, power supply, enclosure): 50%

Course Goals

- Learn how to make computer systems go fast
  - Pipelining
  - Caching
  - Prediction
  - Parallelism
- Learn how do different components communicate
  - Within a computer system (interconnection)
  - With the external world (interfacing)

Logistics

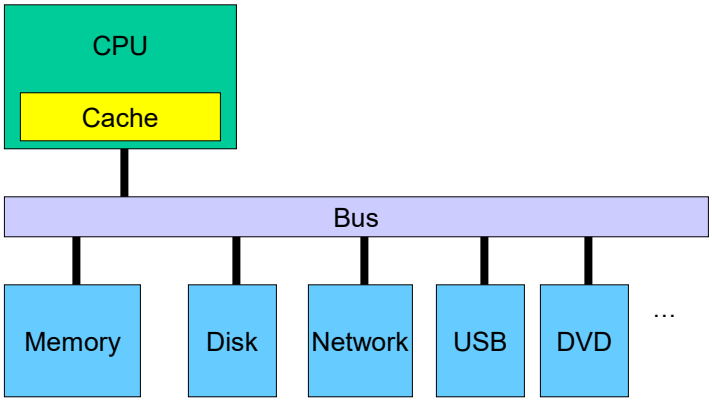
- Instructor of Interconnection/Interfacing Part:
  - Dr. Ahmed Khattab
    - Office Hours: Monday 12:00 to 5:00 (or by appointment)
    - email: akhattab@eng.cu.edu.eg
    - http://eece.cu.edu.eg/~akhattab/elc3030.html
- TAs:
  - Hassan Fakhry (h.elmenier@eng.cu.edu.eg)
- Grading:
  - Quiz/Project/Classwork 20%
  - Midterm 20%
  - Final exam 70%

INTERCONNECTIONS

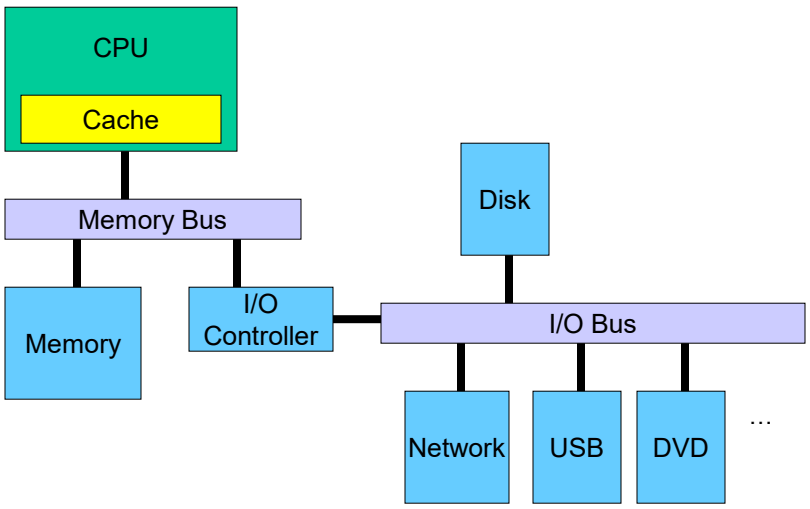
Buses

- Buses are shared communication media used by devices to “talk to” each other. The communication actions which take place can carry both data and control structures.
- Bus variations:
  - on-chip vs. off-chip buses
  - serial vs. parallel buses
  - wired vs. wireless buses

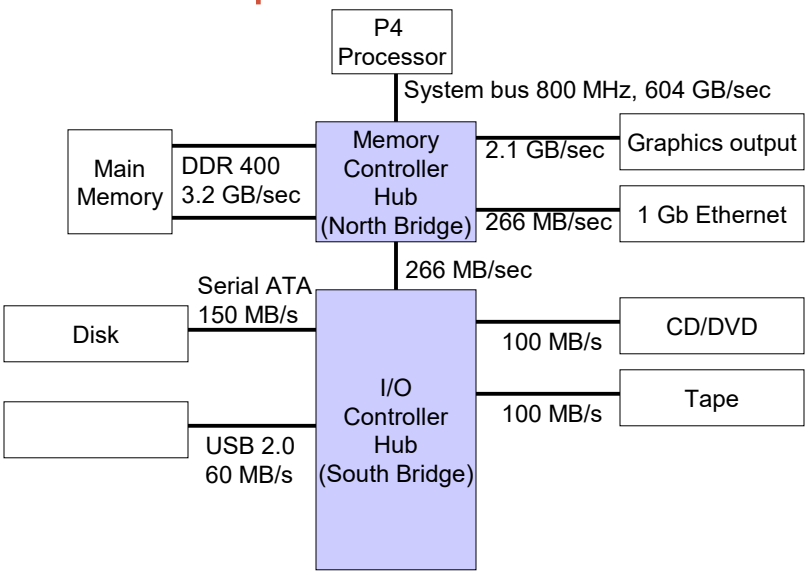
Off-chip Buses: Input/Output



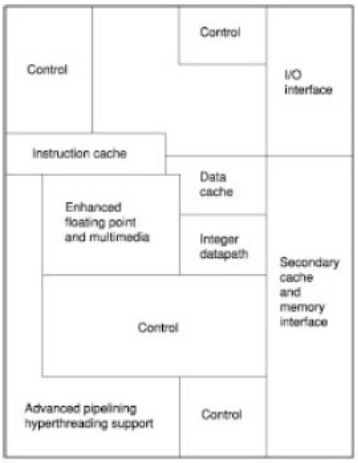
Off-chip Buses: I/O Hierarchy



Intel Example



Pentium 4 Chip



Bus Design

- The bus is a shared resource – any device can send data on the bus (after first arbitrating for it) and all other devices can read this data off the bus
- The address/control signals on the bus specify the intended receiver of the message
- The length of the bus determines its speed (hence, a hierarchy makes sense)
- Buses can be synchronous (a clock determines when each operation must happen) or asynchronous (a handshaking protocol is used to co-ordinate operations)

Transaction

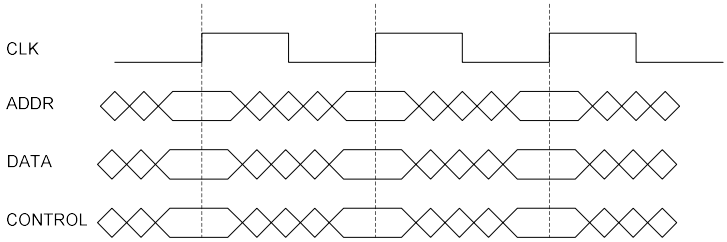
- Transaction is a complete piece of communication. All the transfers which take place across the bus are split across transactions.
- There are three distinct phases of every transaction:
  - **Arbitration** – it is decided which device will own the bus (drive the common medium) for the time of transaction, thus becoming a master. It is only applicable to multi-master buses.
  - **Addressing** – the master activates another listening device by broadcasting (via bus) its address (i.e. device address or control register address) for reception.
    - Can be unicast (message applies to single device) or multicast (applies to many). The addressed devices are traditionally called slaves.
  - Actual **data transfer**.

Synchronization

- Synchronization methods are needed for assurance that data presented in each of the aforementioned phases are valid (not corrupt or changing) when being read. We can distinguish three types of synchronization protocols across the bus:
  - Synchronous protocols
  - Semi-synchronous protocols
  - Asynchronous protocols

Synchronous Protocols

- There is a clock signal, which informs that all the data is stable, and can be read safely
- As vast majority of logic circuits is synchronous, the idea of extending it to the buses seems natural

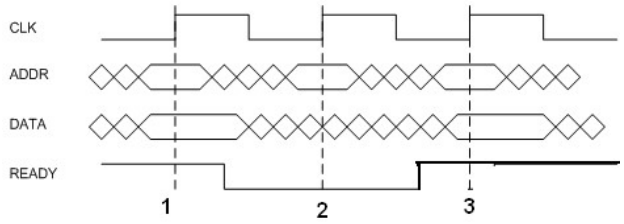


All information is read only on the CLK edge

Challenge: how to synchronize different devices operating at different frequencies?

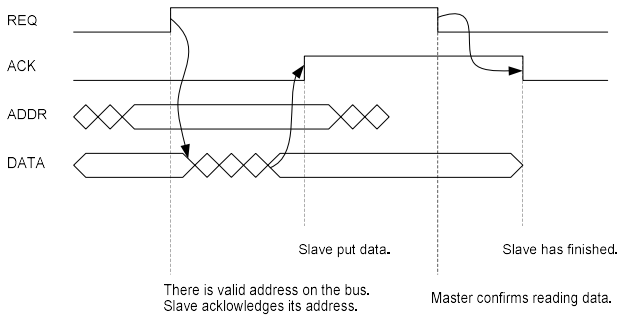
Semi-synchronous Protocols

- There is a clocking signal synchronously to which a request from master occur.
- The response from slave is indicated on dedicated line (i.e. READY or WAIT), whose value is sampled by the clock signal.



Asynchronous Protocols

- There is no implicit time constraints – the asynchronous events are issued both by master and slave.

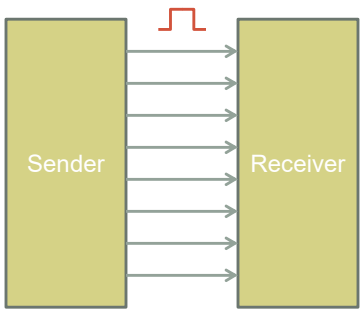


Is gaining popularity in on-chip communications as it does not suffer the constraints/delays imposed by the slowest signal.

PARALLEL AND SERIAL BUSES

Serial vs Parallel Communications

- Parallel communication uses multiple wires running parallel to each other, and can transmit data on all the wires simultaneously.
- Serial communication uses a single wire to transfer the data bits one at a time.



Serial interface transmits a series of bits

Serial vs Parallel Communications: Data Transmission Speeds

- Speed of the parallel data transfer is extremely high compared to serial data transfer.
  - An 8-bit parallel data transfer is 8-times faster than serial data transfer.
- However, clock skew reduces the speed of every link to the slowest of all of the links.
  - The propagation conditions for each may be slightly different (the distributed reactance can differ from one to another) thus resulting in different travel time across whole line.
  - It is difficult to balance many lines, so that they have equal propagation times. This effect is called the skew.

Serial vs Parallel Communications: Data Transfer Quality: Crosstalk and Interference

- Crosstalk between the parallel lines
  - Due to the capacitance and mutual inductance between the wires of a parallel system
  - Leads to inter-symbol interference (ISI) and noise
  - Reduces the bandwidth of link (i.e. lower transmission rates)
- The effect worsens with the length of the communication link
  - This places an upper limit on the length of a parallel data connection
  - Usually parallel buses are much shorter than serial buses

Serial vs Parallel Communications: High Frequency Performance

- In the competitive consumer electronics technology, the speeds of data transfer are increasing day by day
- Unfortunately parallel buses are hard to run at high frequencies for a number of reasons the greatest of which are:
  - It is hard to route many signals across a board without introducing timing variation (clock skew) between them – the more variation the lower maximum frequency is.

$$f_{CLK} = \frac{1}{T_{DMAX} + T_{SKEW}}$$

  - Many wires switching simultaneously at high frequency produce more EMI and interfere and limits the maximum frequency.

Serial vs Parallel Communications: Applications

	Parallel	Serial
Distance	Short	Long
Frequency	Low	High

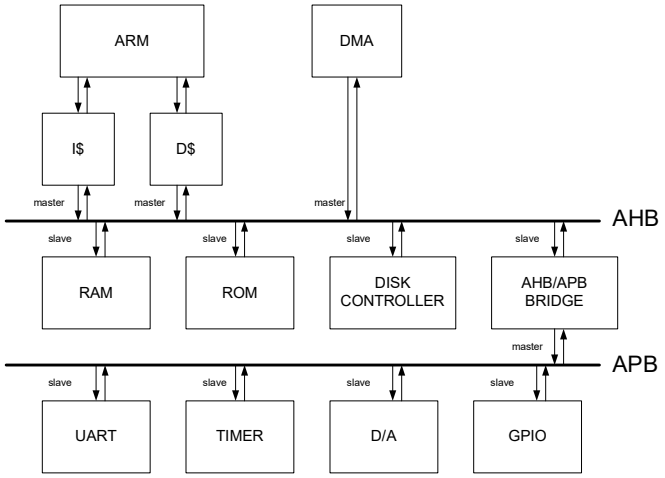
- So, the parallel buses are option of choice for on-chip buses, while in off-chip communication one often resorts to serial protocols.

PARALLEL BUSES

AMBA

- AMBA is a feature-rich example of parallel on-chip bus standard. It is defined by ARM company and used in μC's with ARM cores, thus being the most popular 32 bit on-chip bus.
- The standard defines two buses serving different roles:
  - **AHB** – which stands for ARM High-performance Bus – it links fast peripherals providing high clocking frequency and large throughput. Has very complex hardware and is expensive to implement.
  - **APB** – slow bus, very simple in comparison, cheap to implement in hardware

AMBA Architecture



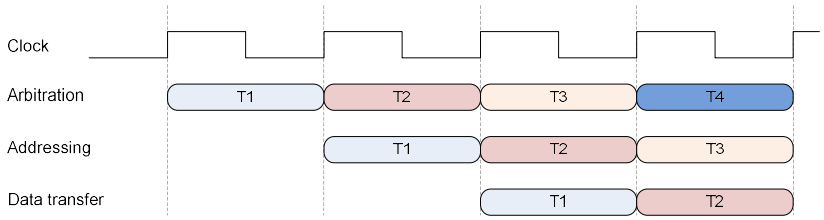
The bridge functions as slave to the AHB and is only master of APB bus. It adapts relatively slow APB bus (can be 16x slower) to high speed AHB, dealing with timing, split transactions and packing and unpacking the bytes of AHB word.

AHB

- AHB is fast, parallel, multi-master, pipelined bus with support for burst and split transactions.

Pipelining

- Pipelined bus can perform each of three transaction phases simultaneously

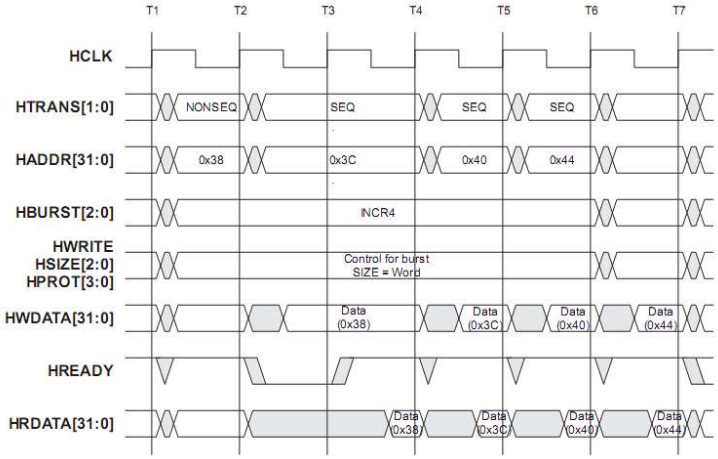


Burst Transaction

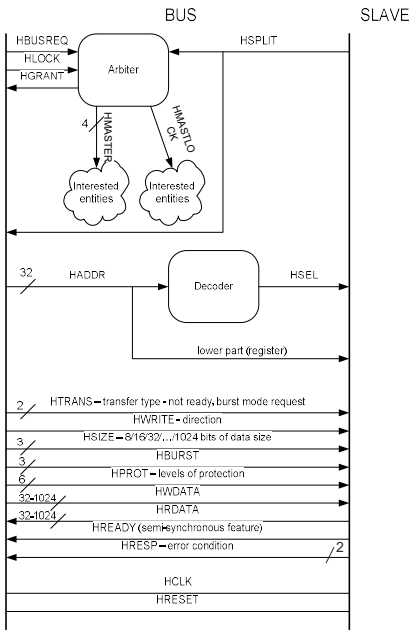
- Special type of transaction with specifying burst request to the slave, which may be able to increment (or decrement) the given address
- Reduce the need of sending many sequential addresses through bus.
  - This reduces power, since computations at the slave are less power consuming, and may increase performance, since address decoding is done at the destination.
- Burst transaction is often used by cache controllers when fetching a block of data from memory.

Burst

- Used to fill buffers of peripherals with sequential data



Connectivity in AHB



Split Transaction

- Its primary goal is to improve performance if following scenario:
- A master initiates a transaction which potentially can take a long time (i.e. communicating with APB peripherals). Without possibility of split transactions it would hold the bus for many cycles, which could not be utilized by other masters (e.g. DMA).
- With split transaction, the masters issues a request, then releases the bus, and waits for notification, which can come many cycles after.

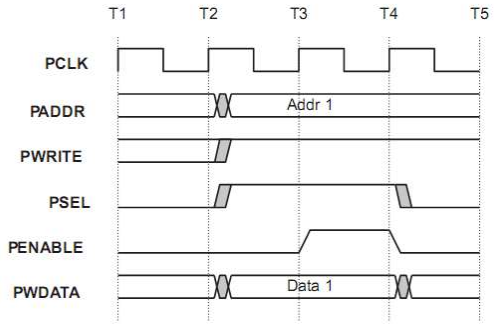
Stages of Split Transaction

1. Master initiates transaction as usual.
2. If the slave is not ready, it asserts split and remembers the active master (provided by arbiter to anyone interested).
3. The arbiter grants the bus to other masters.
4. Slave asserts HSPLIT line to the arbiter, telling which master can resume.
5. Arbiter restores bus grant to interrupted master.

APB

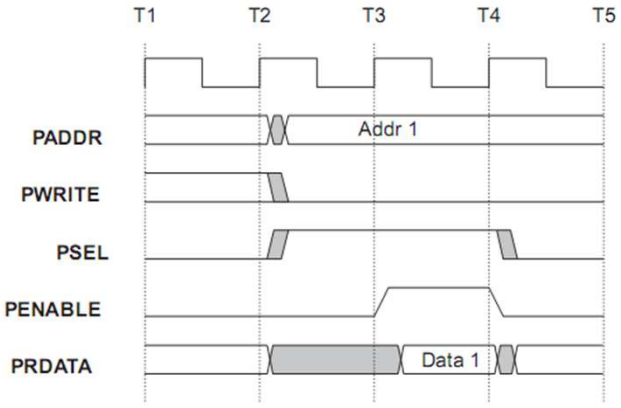
- APB stands for ARM Peripheral Bus. It is a slow, parallel bus with no pipeline, burst or split transaction with single master. The peripherals connected to it won't utilize high speed provided by AHB, so it can be i.e. 12x slower than APB. Otherwise it would be a pure waste of power and chip space.
- Typical buses used in 8-bit µC's are very similar to APB – this is due to the fact, that they are mostly legacy devices fabricated in older technologies, and where simplicity values higher.
- The width of APB data mismatches data width of AHB. The bridge responsibility is to split and merge bytes/words.

Write Transfer



- The PSEL duplicates some of PADDR part to simplify decoding logic in peripherals, thus reducing power.
- PENABLE high indicates data phase.
- If PCKL = 10MHz, the actual data throughput is 5MHz, cause data is held by 2 clock cycles.

Read Transfer

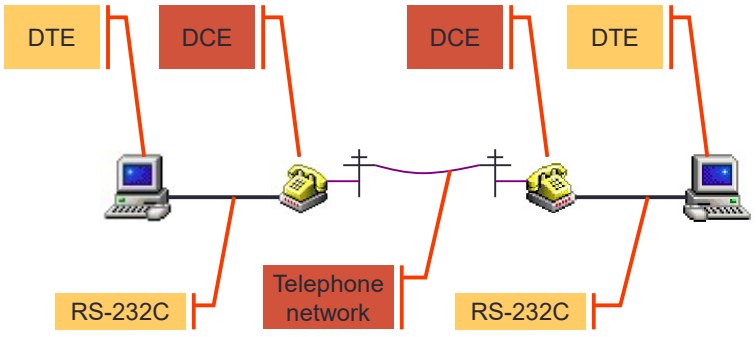


SERIAL BUSES

RS-232C

- History
  - Well-established standard, developed by the EIA (Electronics Industry Association) in 1960s
  - Originally intended as an electrical specification to connect computer terminals to modems
- Defines the interface between a DTE and a DCE
  - DTE = Data Terminal Equipment (terminal)
  - DCE = Data Communications Equipment (modem)
  - A "modem" is sometimes called a "data set"
  - A "terminal" is anything at the "terminus" of the connection
    - VDT (video display terminal), computer, printer, etc.

"Traditional" Configuration



RS-232C Specifications

- Data rate
  - Maximum specified data rate is 20 Kbits/s with a maximum cable length of 15 meters
  - However…
    - It is common to "push" an RS-232C interface to higher data rates
    - Data rates to 1 Mbit/s can be achieved (with short cables!)
- Configuration
  - Serial, point-to-point

Serial Data Transmission

- Two modes
  - Asynchronous
    - The transmitting and receiving devices are not synchronized
    - A clock signal is not transmitted along with the data
  - Synchronous
    - The transmitting and receiving devices are synchronized
    - A clock signal is transmitted along with the data (and is used to synchronize the devices)
  - Most (but not all) RS-232C interfaces are asynchronous!

Asynchronous Data Transmission

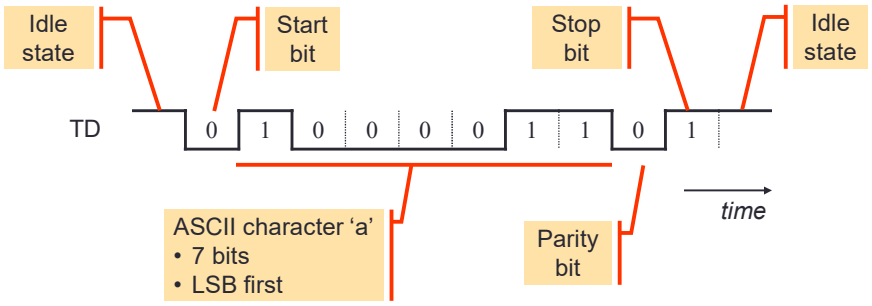
- Data are transmitted on the TD (transmit data) line in packets, typically, of 7 or 8 bits
- Each packet is “framed” by a “start bit” (0) at the beginning, and a “stop bit” (1) at the end
- Optionally, a “parity bit” is inserted at the end of the packet (before the stop bit)
- The parity bit establishes either “even parity” or “odd parity” with the data bits in the packet
  - E.g., even parity: the total number of bits “equal to 1” (including the data bits and the parity bit) is an “even number

1’s and 0’s in RS-232C

- A “1” is called a “mark”
- A “0” is called a “space”
- The idle state for an RS-232C line is a 1 (“mark”)
  - Idle state is called “marking the line”
- Voltages on an RS-232C line
  - Well… that’s another story, and it’s not really a concern to us

Data Transmission Example

- Plot of the asynchronous RS-232C transmission of the ASCII character ‘a’ with odd parity:

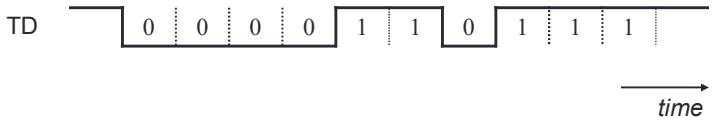


Exercise – RS-232C

- Plot the transmission of the ASCII character “X” over an asynchronous RS-232C channel with 7 data bits and even parity

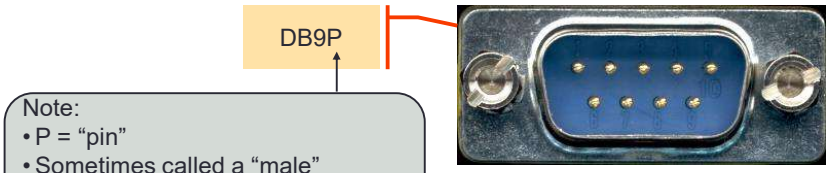
Exercise – RS-232C

- Plot the transmission of the ASCII character “X” over an asynchronous RS-232C channel with 7 data bits and even parity



RS-232C Connectors

- The original standard specified a 25-pin connector
- Today, a 9-pin connector is more common
- E.g.,

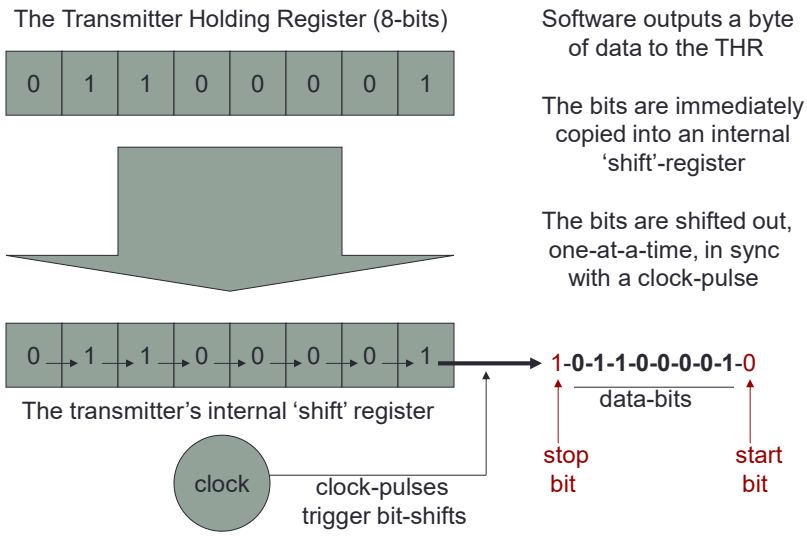


Note:
- P = “pin”
- Sometimes called a “male” connector
- The mate for this is a DP25S, or “socket” connector – the “female”

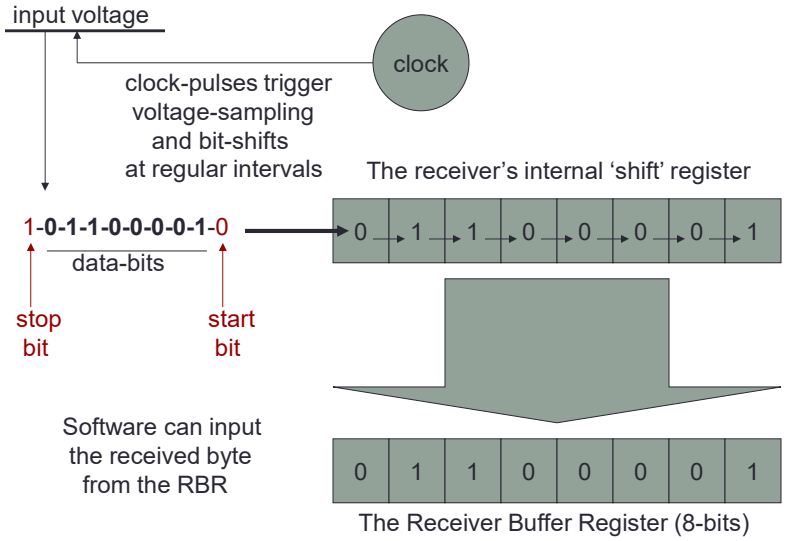
Universal Asynchronous Receiver-Transmitter (UART)

- Transforms parallel data into serial format
- The UART has a transmission engine, and also a reception engine (they can operate simultaneously)
- Software controls the UART's operations by accessing several registers, using the CPU's input and output instructions

Serial Data Transmission



Serial Data Reception



Universal Serial Bus (USB)

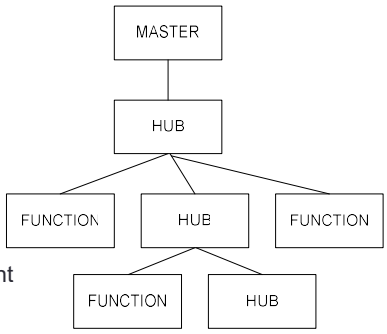
- It was meant to replace huge variety of serial protocols which existed at the time of its design in PC's and embedded systems. It become quite a success with PC, in the world of embedded systems it still competes with others.
- I^2C – one of competitor. Very simple, has only data and clock lines. Used i.e. with boot control, when CPU talks to external ROM.
- CAN, LIN, others – competitors in automotive world. Provide i.e. guarantee of service, multiple masters.

Universal Serial Bus (USB)

- Universal Serial Bus is a synchronous serial protocol for low to medium speed data transmission
  - USB1:
    - Full speed signaling 12 Mbps
    - Low Speed signaling 1.5 Mbps
  - USB2
    - Targets maximum signaling of 480 Mbps (effective 35 Mbps)
  - USB3
    - Targets maximum signaling of 5 Gbps (effective 625 Mbps)
- Intended devices are keyboards, mice, joysticks, speakers; other low to medium speed I/O devices

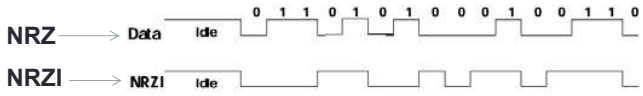
USB Technical Details

- Has one master (mostly a PC). Some smartphones can act both as master or as slave.
- It is tree-structured. Master is called a host. Slaves are called functions. There are also nodes, that only implement connectivity, called hubs.
- Functions cannot request anything from Master. Every communication is initiated by Master. There are no interrupts. Receiving data from functions is done by polling.
- USB provides “hot plugging and unplugging” capability, which is fairly unique, but is dependent on good-written software.
- As much as 128 devices can be plugged-in at the same time.



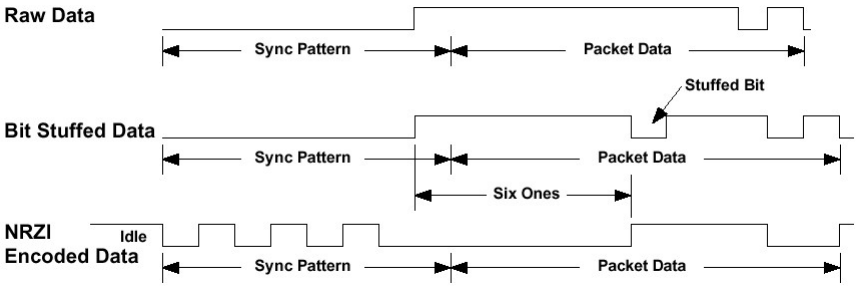
USB Physical Interface

- Differential Signaling, Half duplex
- Input: Binary stream
- NRZ with **bit stuffing**
  - A zero is stuffed after a sequence of 6 ones (why?).
- Differential encoding (NRZI)
  - Zero is encoded as transition
  - One is encoded as no change
  - In USB specification they are called J and K.
  - The transmission is differential, and when both lines are in the same state it signals special condition – synchronization.



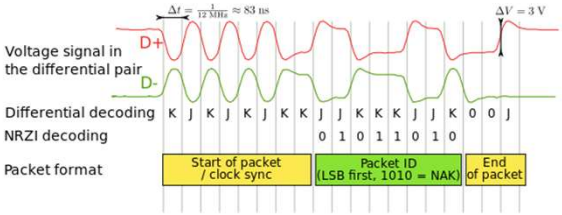
USB Physical Interface

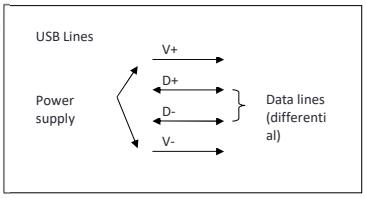
- **Synchronization field**
  - There is sync field which synchronizes the receiver to transmitter.
  - There are 8-bit sequences
    - 7 0's followed by a 1



USB Physical Interface

- Differential Pair Signaling
  - Differential signaling very good at rejecting common-mode noise. If noise is coupled into a cable, then usually it is coupled into all wires in the cable. This 'common-mode' noise (Vcm) can be rejected by input amplifier.





Data Formatting

- Data sent in packets
- Packets will have:
  - Start of Packet Sync Pattern (8 bits, 7 zeros + 1 one)
  - Packet ID (PID) – identifies type of packet. 8 bits total, but only 4 unique bits
  - Address field - 11 bits. 7 bits for USB device (so 128 possible USB devices on bus, host is always address 0), 4 bits for internal use by USB device.
  - Frame number field (11 bits) – incremented by host
  - Data Payload (up to 1023 bytes for high-speed connection)
  - CRC bits - 5 bits for address field, and 16 bits for data field
  - EOP strobe – single ended 0 (160ns-175 ns for high speed, 1.25 us to 1.75 us for high speed)
- Not all packets sent over USB bus have all of these fields (always have SOP, EOP and PID). Packet without data field is a token packet.

What is I²C

- **The name stands for "Inter - Integrated Circuit Bus"**
- **A Small Area Network connecting ICs and other electronic systems**
- **Originally intended for operation on one single board / PCB**
  - Synchronous Serial Signal
  - Two wires carry information between a number of devices
  - One wire use for the data
  - One wire used for the clock
- **Today, a variety of devices are available with I²C Interfaces**
  - Microcontroller, EEPROM, Real-Timer, interface chips, LCD driver, A/D converter

What is I²C used for?

- **Data transfer between ICs and systems at relatively low rates**
  - "Classic" I²C is rated to 100K bits/second
  - "Fast Mode" devices support up to 400K bits/second
  - A "High Speed Mode" is defined for operation up to 3.4M bits/second
- **Reduces Board Space and Cost By:**
  - Allowing use of ICs with fewer pins and smaller packages
  - Greatly reducing interconnect complexity
  - Allowing digitally controlled components to be located close to their point of use

I^2C Bus Characteristics

- **Includes electrical and timing specifications, and an associated bus protocol**
- **Two wire serial data & control bus implemented with the serial data (SDA) and clock (SCL) lines**
  - For reliable operation, a third line is required: Common ground
- **Unique start and stop condition**
- **Slave selection protocol uses a 7-Bit slave address**
  - The bus specification allows an extension to 10 bits
- **Bi-directional data transfer**
- **Acknowledgement after each transferred byte**
- **No fixed length of transfer**

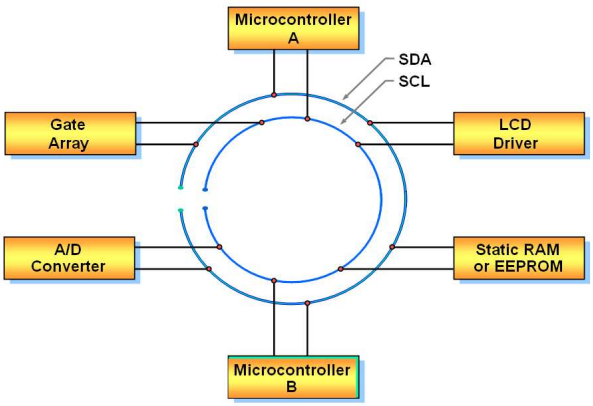
I^2C Bus Characteristics (cont'd)

- **True multi-master capability**
  - Clock synchronization
  - Arbitration procedure
- **Transmission speeds up to 100Khz (classic I2C)**
- **Max. line capacitance of 400pF, approximately 4 meters (12 feet)**
- **Allows series resistor for IC protection**
- **Compatible with different IC technologies**

I^2C Bus Definitions

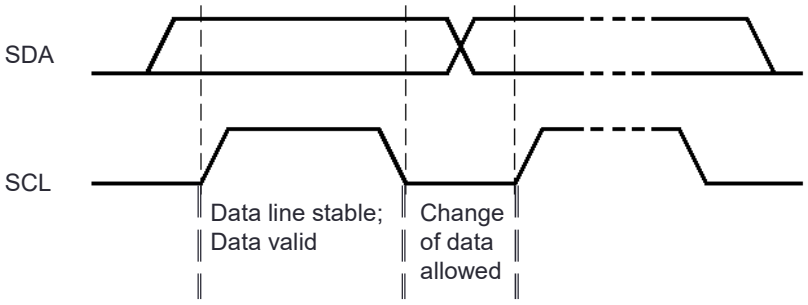
- **Master:**
  - Initiates a transfer by generating start and stop conditions
  - Generates the clock
  - Transmits the slave address
  - Determines data transfer direction
- **Slave:**
  - Responds only when addressed
  - Timing is controlled by the clock line

I^2C Bus Configuration Example



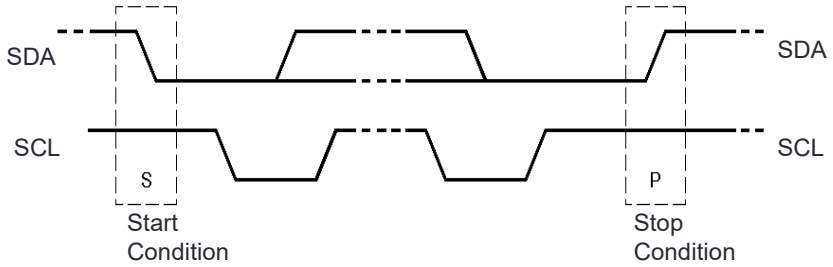
Bit Transfer on the I²C Bus

- In normal data transfer, the data line only changes state when the clock is low



Start and Stop Conditions

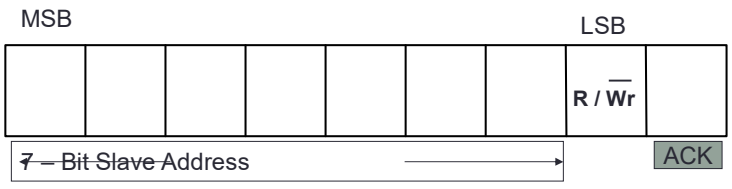
- A transition of the data line while the clock line is high is defined as either a start or a stop condition.
- Both start and stop conditions are generated by the bus master
- The bus is considered busy after a start condition, until a stop condition occurs



I²C Addressing

- **Each node has a unique 7 (or 10) bit address**
- **Peripherals often have fixed and programmable address portions**
- **Addresses starting with 0000 or 1111 have special functions:-**
  - 0000000 Is a General Call Address
  - 0000001 Is a Null (CBUS) Address
  - 1111XXX Address Extension
  - 1111111 Address Extension – Next Bytes are the Actual Address

First Byte in Data Transfer on the I²C Bus



R/Wr

0 – Slave written to by Master

1 – Slave read by Master

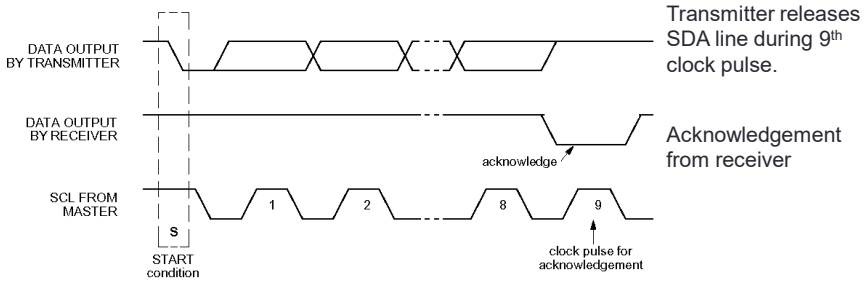
ACK – Generated by the slave whose address has been output.

I²C Bus Connections

- **Masters can be**
  - Transmitter only
  - Transmitter and receiver
- **Slaves can be**
  - Receiver only
  - Receiver and transmitter

Acknowledgements

- Master/slave receivers pull data line low for one clock pulse after reception of a byte
- Master receiver leaves data line high after receipt of the last byte requested
- Slave receiver leaves data line high on the byte following the last byte it can accept

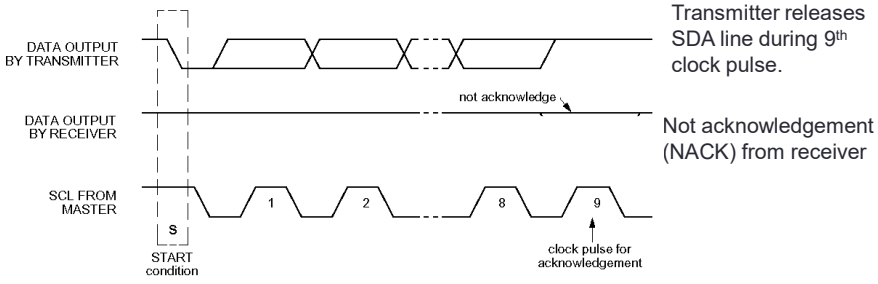


Acknowledgements

- **From Slave to Master Transmitter:**
  - After address received correctly
  - After data byte received correctly
- **From Slave to Master Receiver:**
  - Never (Master Receiver generates ACK)
- **From Master Transmitter to Slave:**
  - Never (Slave generates ACK)
- **From Master Receiver to Slave:**
  - After data byte received correctly

Negative Acknowledge

- Receiver leaves data line high for one clock pulse after reception of a byte

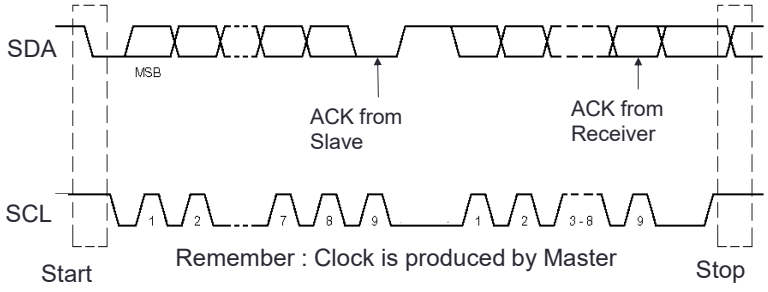


Negative Acknowledge (Cont'd.)

- **From Slave to Master Transmitter:**
  - After address not received correctly
  - After data byte not received correctly
  - Slave Is not connected to the bus
- **From Slave to Master Receiver:**
  - Never (Master Receiver generates ACK)
- **From Master Transmitter to Slave:**
  - Never (Slave generates ACK)
- **From Master Receiver to Slave:**
  - After last data byte received correctly

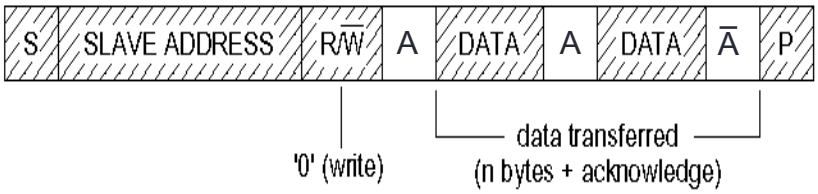
Data Transfer on the I2C Bus

- **Start Condition**
- **Slave address + R/W**
  - Slave acknowledges with ACK
- **All data bytes**
  - Each followed by ACK
- **Stop Condition**



Data Formats

- **Master writing to a Slave**




from master to slave

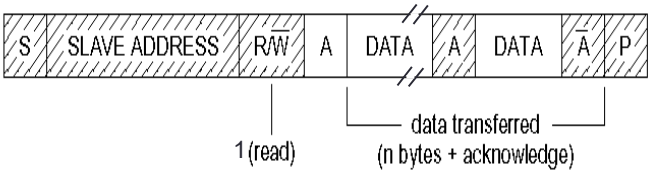

from slave to master

A = acknowledge (SDA LOW)
Ā = not acknowledge (SDA HIGH)
S = START condition
P = STOP condition

Data Formats Cont'd.

- **Master reading from a Slave** :

Master is Receiver of data and Slave is Transmitter of data.



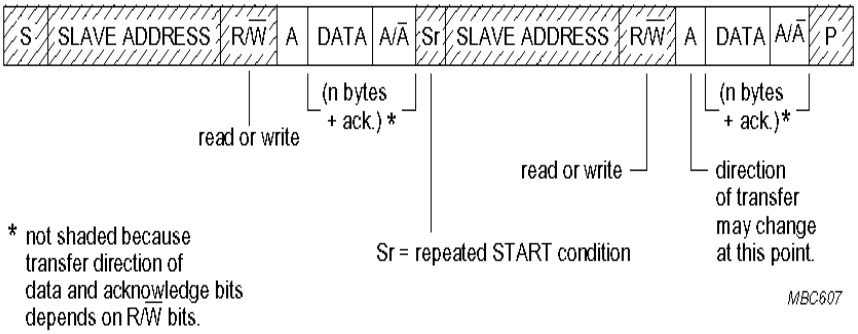

from master to slave


from slave to master

A = acknowledge (SDA LOW)
Ā = not acknowledge (SDA HIGH)
S = START condition
P = STOP condition

Data Formats Cont'd.

- **Combined Format**



- A **repeated start** avoids releasing the bus and therefore prevents another master from taking over the bus

Available I^2C Devices

- **Analog to Digital Converters (A/D, D/A):** MMI functions, battery & converters, temperature monitoring, control systems
- **Bus Controller:** Telecom, consumer electronics, automotive, Hi-Fi systems, PCs, servers
- **Bus Repeater, Hub & Expander:** Telecom, consumer electronics, automotive, Hi-Fi systems, PCs, servers
- **Real Time Clock (RTC)/Calendar:** Telecom, EDP, consumer electronics, clocks, automotive, Hi-Fi systems, FAX, PCs, terminals
- **DIP Switch:** Telecom, automotive, servers, battery & converters, control systems
- **LCD/LED Display Drivers:** Telecom, automotive instrument driver clusters, metering systems, POS terminals, portable items, consumer electronics

Available I²C Devices

- **General Purpose Input/Output (GPIO) Expanders and LED Display Control:** Servers, keyboard interface, expanders, mouse track balls, remote transducers, LED drive, interrupt output, drive relays, switch input
- **Multiplexer & Switch:** Telecom, automotive instrument driver clusters, metering systems, POS terminals, portable items, consumer electronics
- **Serial RAM/ EEPROM:** Scratch pad/ parameter storage
- **Temperature & Voltage Monitor:** Telecom, metering systems, portable items, PC, servers
- **Voltage Level Translator:** Telecom, servers, PC, portable items, consumer electronics

End use

- **Telecom:** Mobile phones, Base stations, Switching, Routers
- **Data processing:** Laptop, Desktop, Workstation, Server
- **Instrumentation:** Portable instrumentation, Metering systems
- **Automotive:** Dashboard, Infotainment
- **Consumer:** Audio/video systems, Consumer electronics (DVD, TV etc.)

Applications

- There are some specific applications for certain types of I^2C devices such as TV or radio tuners, but in most cases a general purpose I^2C device can be used in many different applications because of its simple construction.

I/O Buses and Interfaces

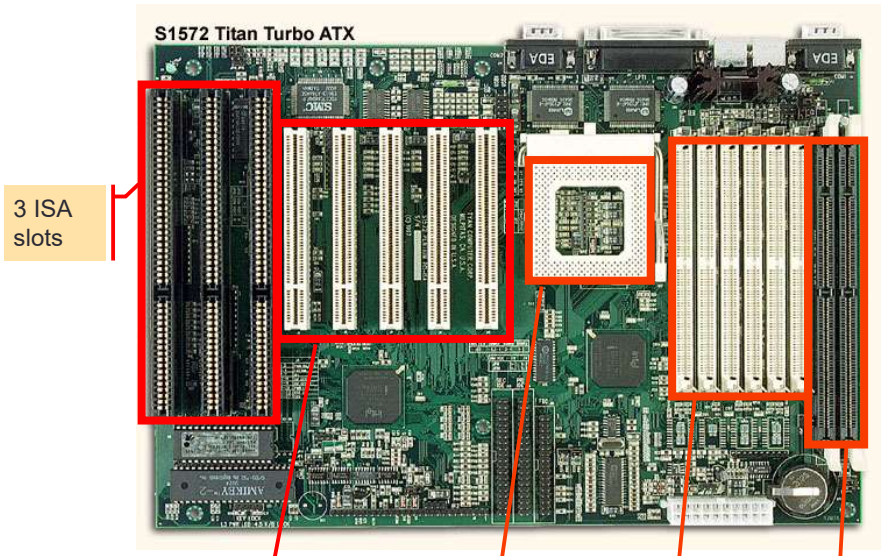
- There are many “standards” for I/O buses and interfaces
- Standards allow “open architectures”
  - Many vendors can provide peripheral (I/O) devices for many different systems
- Most systems support several I/O buses and I/O interfaces

Examples

- Expansion buses or “slots”
- Disk interfaces
- External buses

Expansion Buses

- These are “slots” on the motherboard
- Examples
  - ISA – Industry Standard Architecture
  - PCI – Personal Component Interconnect
  - EISA – Extended ISA
  - SIMM – Single Inline Memory Module
  - DIMM – Dual Inline Memory Module
  - MCA – Micro-Channel Architecture
  - AGP – Accelerated Graphics Port
  - VESA – Video Electronics Standards Association
  - PCMCIA – Personal Computer Memory Card International Association (not just memory!)



Disk Interfaces

- Examples
  - ATA – AT Attachment (named after IBM PC-AT)
  - IDE – Integrated Drive Electronics (same as ATA)
  - Enhanced IDE
    - Encompasses several older standards (ST-506/ST-412, IDE, ESDI, ATA-2, ATA-3, ATA-4)
  - Floppy disk
  - SCSI – Small Computer Systems Interface
  - ESDI – Enhanced Small Device Interface (mid-80s, obsolete)
  - PCMCIA

External Buses

- Examples
  - Parallel – sometimes called LPT (“line printer”)
  - Serial – typically RS232C (sometimes RS422)
  - PS/2 – for keyboards and mice
  - USB – Universal Serial Bus
  - IrDA – Infrared Device Attachment
  - FireWire – new, very high speed, developed by IEEE

Conclusions

- Buses connect different components of a computer system
- Parallel buses are suitable for high speed short range applications such as on-chip interconnects
- Serial buses are suitable for low speed long range applications such as off-chip interconnects
- Communication can be synchronous or a synchronous