Interrupt

- An interrupt is a signal from a device attached to a computer or from a program within the computer that causes the CPU to stop its normal program execution and perform service related to the event.

- Examples of interrupts :I/O completion, divide-by-0, etc.

- **Maskable Interrupt**: It is a hardware interrupt that may be ignored by setting a bit in an interrupt mask register's (IMR) bit-mask.

- **Nonmaskable Interrupt**:is a hardware interrupt that does not have a bit-mask associated with it - meaning that it can never be ignored. NMIs are often used for timers, especially watchdog timers

Interrupt Overhead

The interrupt overhead is caused by context switching (storing and restoring the state of CPU)

On interrupt handler entry, the context of the current process and its thread must be saved. On exit, it must be restored.

On handler entry, memory locations different from the memory locations in the cache are used, and therefore cache updates are required.

-In a similar way, the interrupted process suffers from the interrupt:

-the cache it was using is disturbed, and cache updates are required

Reentrant code

- A computer program or routine is described as **reentrant** if the routine can be re-entered while it is already running (i.e it can be safely executed concurrently).
- To be reentrant, a computer program or routine:
  - Must hold no static (or global) non-constant data.
  - Must not return the address to static (or global) non-constant data.
  - Must work only on the data provided to it by the caller.
  - Must not modify its own code. (unless executing in its own unique thread storage)
  - Must not call non-reentrant computer programs or routines.
- Example of non-reentrant code

```
int g_var = 1;
int f()
    { g_var = g_var + 2;
      return g_var;
    }
int g()
{
    return f() + 2;
}
```

- In the above, f depends on a non-constant global variable *g_var*; thus, if two threads execute it and access g_var concurrently, then the result varies depending on the timing of the execution. Hence, f is not reentrant. Neither is g; it calls f, which is not reentrant.
- Example for reentrant code:

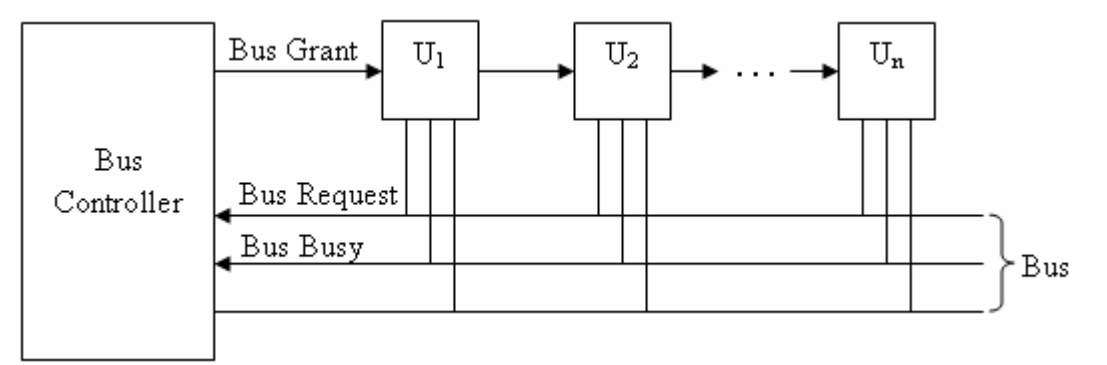
```
int f(int i)
{    return i + 2; }

int g(int i)
{ return f(i) + 2; }
```

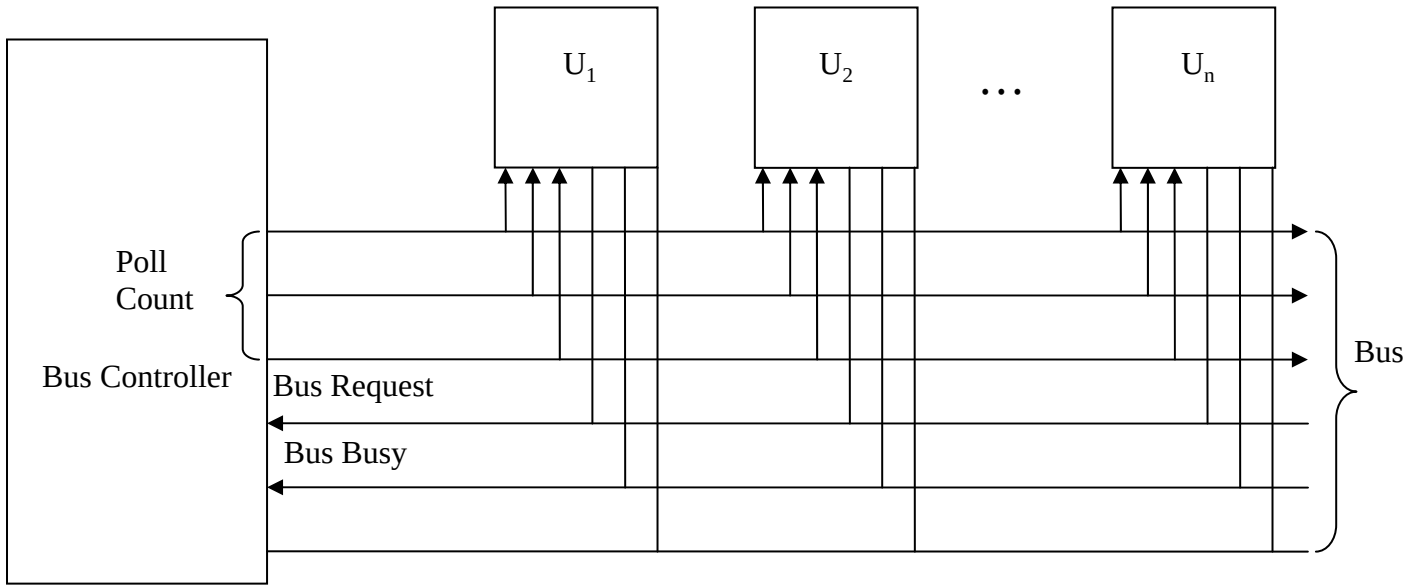
Bus Arbitration

- If several units can generate requests for bus access simultaneously, the bus master (bus controller) needs a way to select one of the units. This selection process is called Bus arbitration
- Three different arbitration schemes: (number of control lines, speed of bus controller)
  - Daisy chaining
  - Polling
  - Independent requesting
- Some bus systems combine several distinct arbitration techniques
- Bus arbitration schemes usually try to balance two factors:
- – Bus priority: the highest priority device should be serviced
- first
- – Fairness: Even the lowest priority device should never
- be completely locked out from the bus.
- Types of Arbitration:
- static: (priority fixed) dynamic: (flexible priority)
- - Daisy chaining - polling
- - Parallel arbitration - Time slice
- - Independent Requesting. -LRU/FIFO
- - rotating daisy chain

- Daisy Chaining



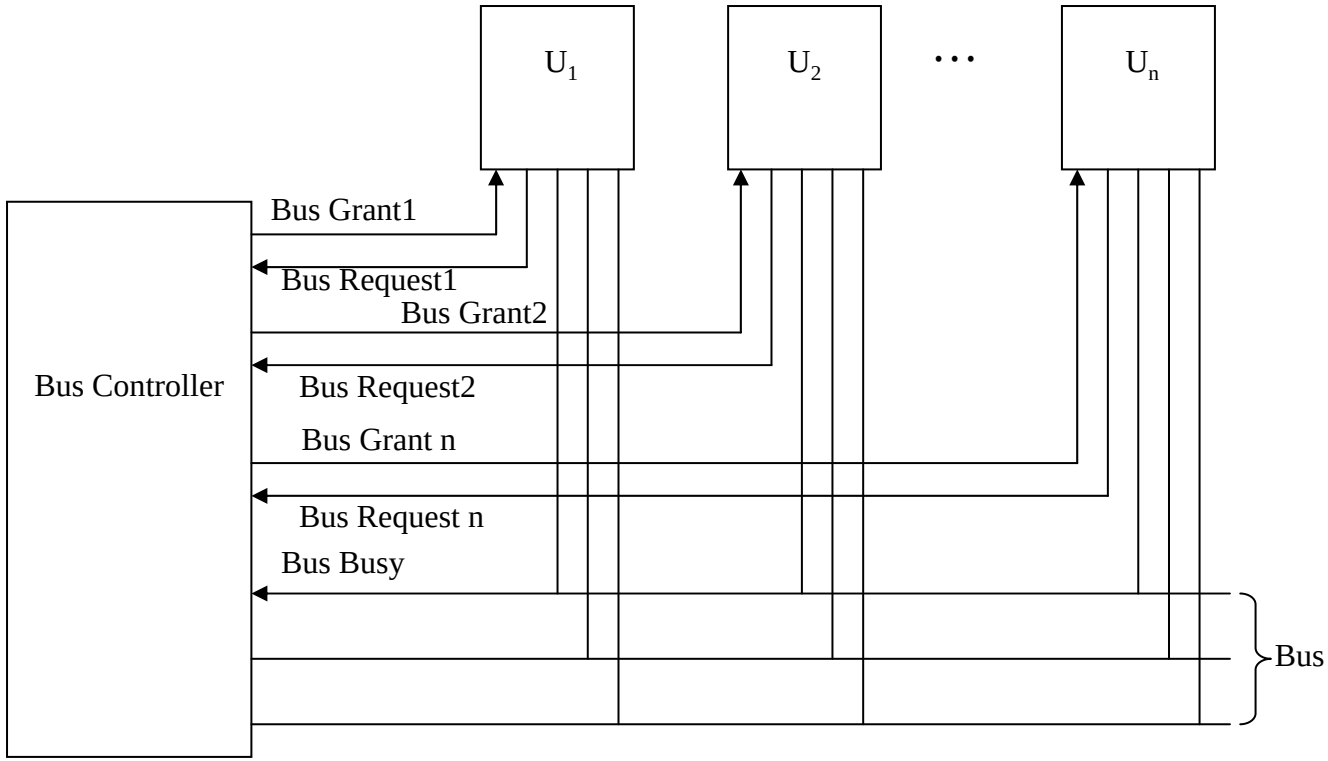
- Polling



polling

- Unit requests the bus via BUS request line.
- In response, the Bus controller proceeds to generate a sequence of numbers on the poll-count lines.
- Each unit compares to the unique address assigned to it.
- When requesting unit finds the match, Bus Busy signal is activated.
- In response, bus controller terminates the polling process and U_i connects to the bus.
- Advantage: failure of one unit need not affect the other units.
- Disadvantage: expensive because of more control lines. Number of units are limited based on the poll-count lines capability

Independent Requesting



- The bus controller determines priority, which is programmable.
- Drawback: 2n Bus request and bus grant lines to control n devices, whereas daisy chaining requires 2 such lines and polling requires $\log_2 n$ lines approximately.

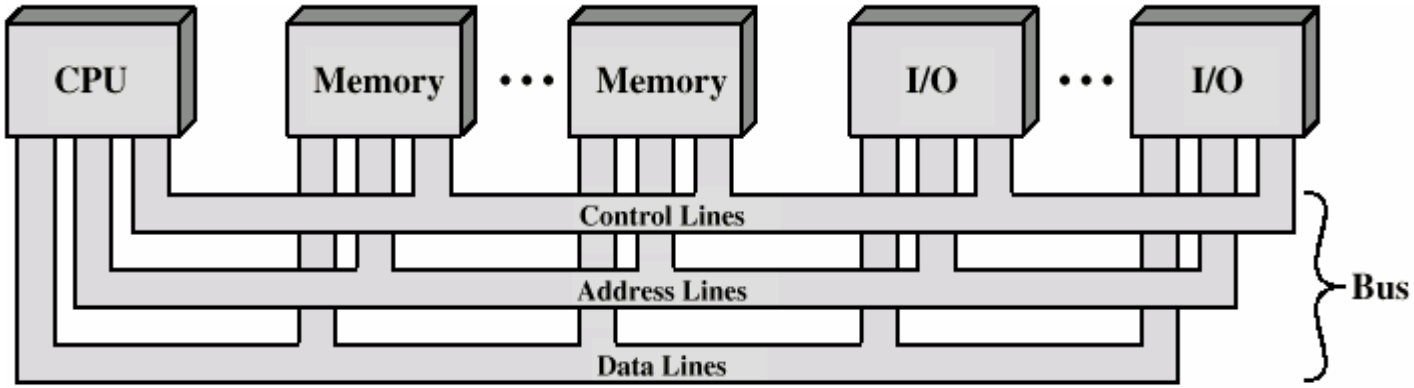
What is a Bus?

- A communication pathway connecting two or more devices and also provide the mechanisms for controlling the exchange of signals over the bus.

- System bus comprises of data, address and control bus and handles the intrasystem communication.

- Data bus
  - Carries data
  - Width is a key determinant of performance
  - 8, 16, 32, 64 bit
- Address bus
  - Identify the source or destination of data
  - Bus width determines maximum memory capacity of system
  - e.g. 16 bit address bus giving 64k address space
- Control bus
  - Control and timing information
    - Memory read/write signal
    - Interrupt request
    - Clock signals

Bus Interconnection Scheme



Types of Buses

• Today there are likely to be a number of different buses in the typical machine, supporting various devices. They can be divided into

– Local buses

- **Such as PCI**
- **Used with internal devices such as graphical cards**

-External buses

- **Such as ISA**
- **Used with external devices such as scanners.**

Other types of buses:

• Processor-Memory Bus

– Short and high speed

– Only need to match the memory system

- Maximize memory-to-processor bandwidth

– Connects directly to the processor

– Optimized for cache block transfers

Bus Protocols

The essence of any bus is the set of rules by which data moves between devices. This set of rules is called a "protocol.

- PCI (Peripheral Component Interconnect)
- SCSI (Small computer System Interconnect)

PCI:

PCI bus is often referred as "local" bus which is designed to interface with different microprocessor families, main memory, and a very wide range of I/O devices.

It can support either 32- bit or 64- bit data transfer with maximum clock rate of 66 MHZ and allows a data transfer rate up to 524 MB/s.

PCI BUS

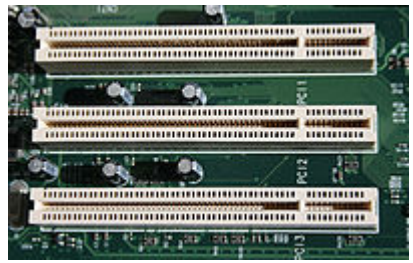
Familiar with I/O bus and standard I/O interfaces— parallel bus PCI (Peripheral Connect Interface) for a synchronous parallel communication interface bus

Devices connected to the PCI bus appear to the processor as if they were connected directly to the processor bus.

Early PC's – 8 bit XT Bus

16-bit bus – ISA bus

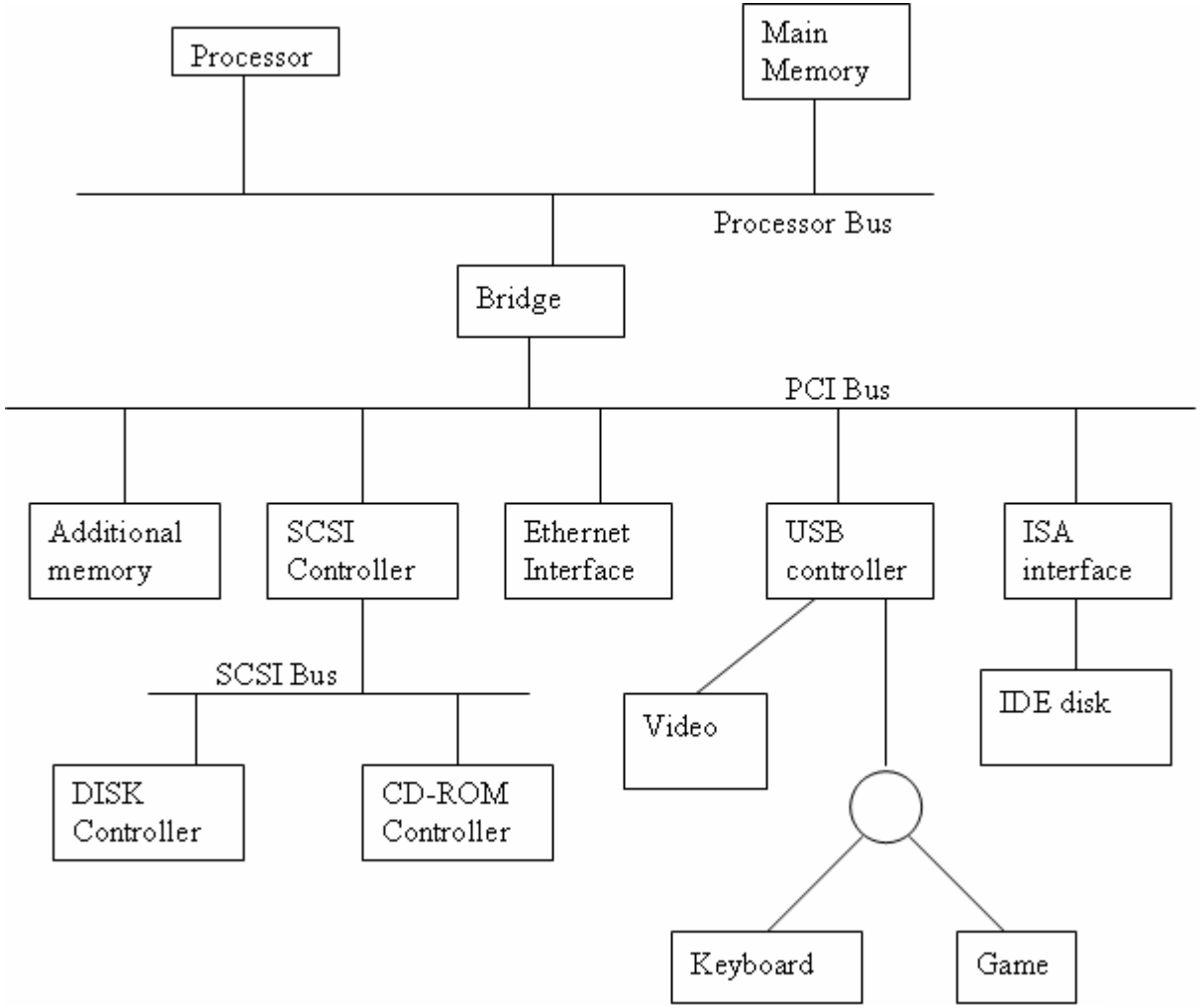
32-bit – EISA bus

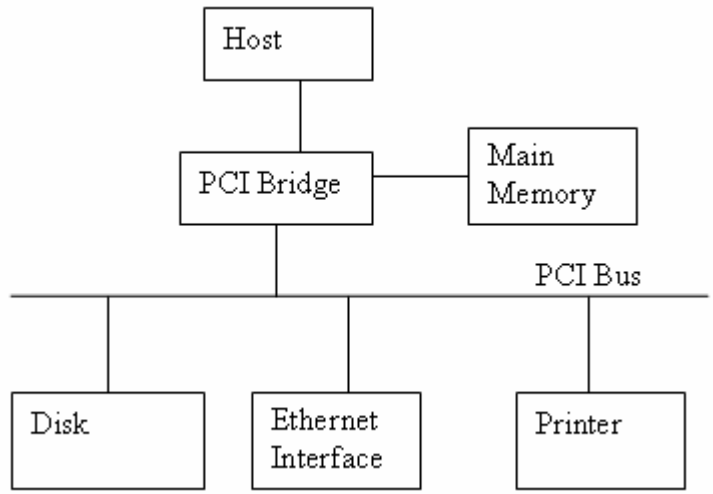


The PCI was developed as a low cost bus that is truly processor independent.

An important feature that the PCI is a plug-and-play capability for connecting I/O devices.

To connect a new device, the user simply connects the device interface board to the bus.


Processor
Main Memory
Processor Bus
Bridge
PCI Bus
Additional memory
SCSI Controller
Ethernet Interface
USB controller
ISA interface
SCSI Bus
Video
IDE disk
DISK Controller
CD-ROM Controller
Keyboard
Game


Host
PCI Bridge
Main Memory
PCI Bus
Disk
Ethernet Interface
Printer

PCI – Data Transfer Contd.,

- The PCI is designed to support burst data transfer mode of operation and it uses multiplexed address and data lines.
- Advantages:
  - Speed.
  - Reliability.
  - Multiple masters.
  - Configurability
- The bus supports three independent address spaces
  - Memory
  - I/O
  - Configuration
- A 4-bit command that accompanies the address identifies which of the three spaces is being used in a given data transfer operation

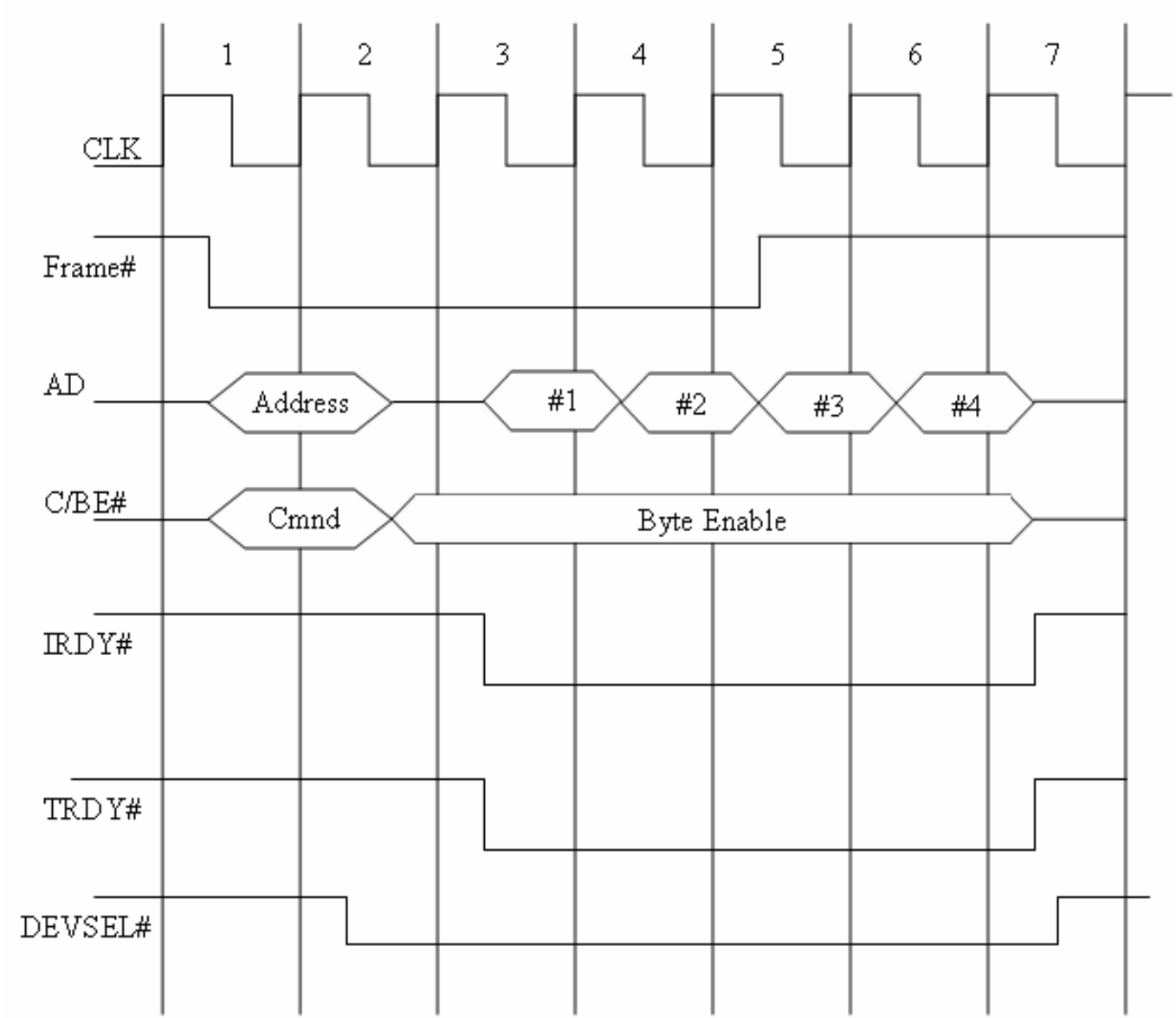
- Before, The master maintains the address information on the bus until data transfer is completed. But this is not necessary
- The address is needed only long enough for the slave to be selected.
- The slave can store the address in its internal buffer.
- Thus the address is needed for one clock cycle only, freeing the bus for sending the data in subsequent clock cycles => reduces cost
- A master is called an initiator in PCI terminology – DMA controller.
- The addressed device that responds to commands is target.

Data transfer signals on the PCI bus

Name	***Function***
CLK	A 33-MHz or 66 MHz clock
FRAME#	sent by initiator to indicate the duration of a transaction
AD	32 address/data lines (may be optionally increased to 64)
C/BE#	4 command/byte enable lines (8 for a 64-bit bus)
IRDY#, TRDY#	Initiator ready and target ready signals
DEVSEL#	a response from the device indicating that it has recognized its address and ready for a data transfer transaction
IDSEL#	Initialization Device Select

Signals whose name ends with the symbol # are asserted when in the low voltage state

A read operation on the PCI Bus



Example of a Bus transaction

- Processor reads 4 32-bit words from the memory.
- Initiator – processor, target – memory
- A complete transfer operation on the bus, involving an address and a burst of data is called ***transaction***
- Individual word transfers within transaction are called ***phases***
- Clock cycle 1 – FRAME# is asserted by the processor, address is sent on AD lines, a command on the C/BE# lines
- Clock cycle 2 – AD bus lines are off => processor removes address from AD lines, DEVSEL# is asserted, C/BE# (4 lines are associated with one byte on the AD lines.

- The initiator sets one or more of the C/BE# lines to indicate which byte lines are used for transferring data
- Clock cycle 3 – IRDY# is asserted to indicate that initiator is ready to receive the data, TRDY# - to indicate that target has data to send at this time.
- The initiator loads the data into its input buffer at the end of the clock cycle.
- Clock cycle 4 to 6 – target sends 3 more words
- Frame# is negated during clock cycle 5
- After sending the 4^{th} word in CLK cycle 6, the target disconnect drivers and negates DEVSEL# at the beginning of CLK cycle 7.

SCSI Bus

- SCSI – Small Computer System Interface – defined by ANSI – X3.131 – pronounced as *SCUZZY* - is a set of standards for physically connecting and transferring data between computers and peripheral devices
- Narrow bus – 8 data lines
- Wide bus – 16 data lines
- Earlier versions – High voltage differential (HVD) – 5V
- Latest versions – Low voltage differential (LVD) – 3.3V
- The maximum transfer rate in commercial devices – 5mb/s to 160mb/s
- Maximum transfer rate of the standard – 320mb/s, 640mb/s.
- The maximum transfer rate on a given bus is often a function of the length of the cable and the number of devices connected
- SCSI is most commonly used for hard disks and tape drives, but it can connect a wide range of other devices, including scanners and CD drives.(8 or 16 devices can be attached to a single bus)

- The maximum capacity of the bus is 8 devices for a narrow bus and 16 devices for a wide bus.
- This controller uses DMA to transfer data packets from the main memory to the device, or vice-versa. A packet may contain a block of data, commands from the processor to the device or status information about the device.
- A controller connected to a SCSI bus is one of the two types – initiator or a target
- An initiator has the ability to select a particular target and send commands specifying the operations to be performed.
- Initiator – SCSI controller, target – disk controller
- The initiator establishes a logical connection with the intended target

Category	Name	Function
Data	-DB(0) to –DB(7)	Data lines – carry one byte of information during the information transfer phase and identify device during arbitration, selection and reselection phases
	-DB(P)	Parity bit for the data bus
Phase	-BSY	Busy – asserted when the bus is not free
	-SEL	Selection: Asserted during selection and reselection
Information type	-C/D	Control/Data: Asserted during transfer of control information
	-MSG	Message: indicates that the information being transferred is a message
Handshake	-REQ	Request: Asserted by a target to request a data transfer cycle
	-ACK	Acknowledge: Asserted by the initiator when it has completed a data transfer operation
Direction of transfer	-I/O	Input/Output: Asserted to indicate an input operation
Other	-ATN	Attention: Asserted by an initiator when it wishes to send a message to a target
	-RST	Reset: Causes all device controls to disconnect from the bus and assume their start-up state