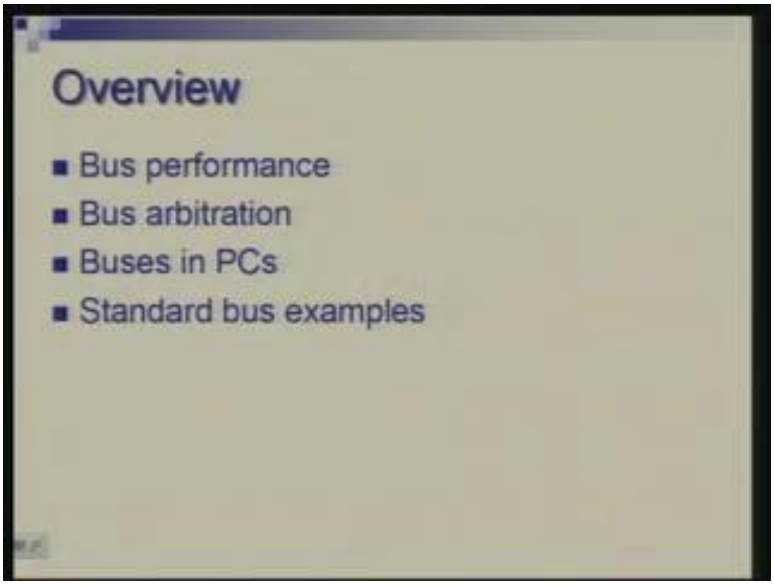
Computer Architecture
Prof. Anshul Kumar
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi
Lecture - 35
Input/Output Subsystem: Interfaces and buses (Contd.)

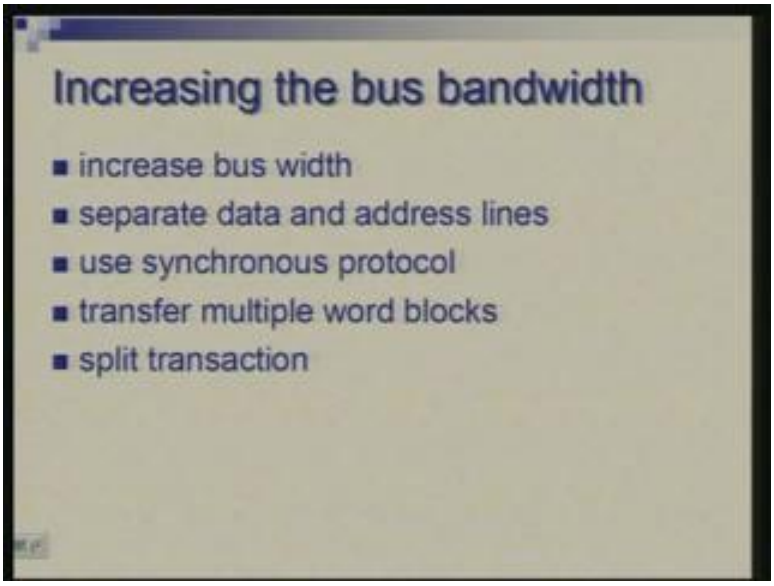
So far we have seen that the input/output subsystem consists of peripheral devices along with their controllers which are connected to rest of the system through buses. So, buses form typically the medium of communication between peripherals, processor and memory. Last time we saw that there are different types of buses and performance of the buses in terms of their throughput or bandwidth was an important issue. We will continue on that discussion, talk about performance, we will begin with the example which we covered in the end of the last lecture, we will repeat that and talk about issue of bus arbitration when there are multiple devices which need to communicate on the same bus. We will we will take specific example of how buses are organized in a modern PC system and we will also look at some standard buses as an example.

(Refer Slide Time: 2:09)



So, while talking of performance of the bus, we notice that there are several factors which can be exploited to increase the performance of the bus. So something which is very obvious is the width of the bus that means how many bits or bytes it can carry at any time. so there are buses on one extreme which are serial buses that means 1-bit of data is carried at any given time, then there are 8-bit buses 16-bit buses, 32-bit buses, 64 and even higher. So the rate at which data moves or the throughput is directly proportional to width of the bus which is very obvious.

(Refer Slide Time: 02:25)



Then different things on a bus maybe multiplexed. For example, typically address and data lines are multiplexed. That means at any given point of time either address is being communicated by the bus or data is being communicated. So if you can provide separate lines which means additional cost of the bus then there is an improvement in performance.

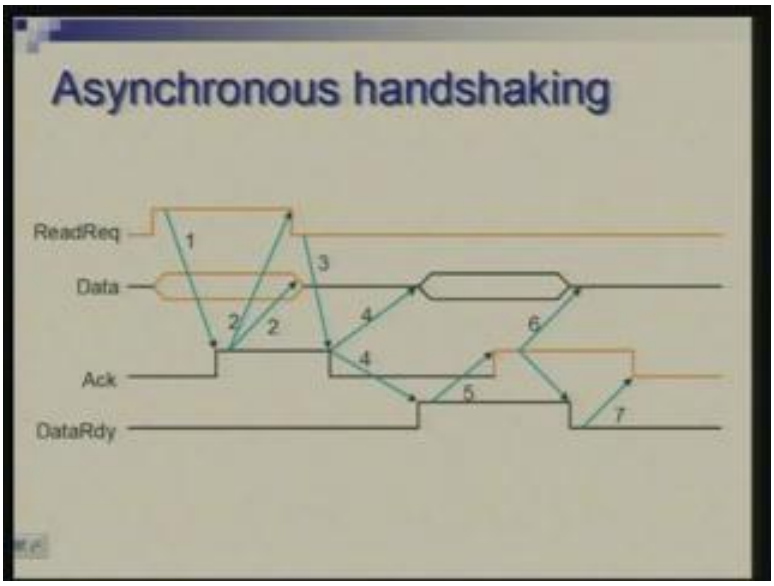
We also looked at two different types of protocols: synchronous and asynchronous. Synchronous protocol implies that everything is controlled by clock. So all events occur at active edge of clock and all times are measured in terms of clock cycles whereas in asynchronous bus there is interlocked response to each other's request from the device and the other party which could be memory or processor. So one event occurs which triggers another event that triggers another event and so on, so the events get chained or interlocked one after another.

Now, because of the need to allow arbitrary amount of time from one event to other event these protocols tend to be slower. They are flexible in the sense that slow and fast devices can be mixed on the same bus but the throughput is comparatively lower as compared to synchronous bus where you do not have to sense signals going up or down, you need to wait for a fixed amount of time and assume that something would have happened. So the higher speed buses typically follow synchronous approach.

In the last example, in the previous lecture, we were seeing the effect of the block size. If you are able to transfer chunks of larger blocks then the overall throughput is faster, we will go through that once again. And finally if you can make use of the bus in the idle period, typically the bus is occupied when you are starting a transaction then it is unoccupied and then it is required again when you are closing it. So in between there is an unutilized period and we can use it to interleave transactions or initiate other transactions that is called a split transaction protocol so that is another device, another

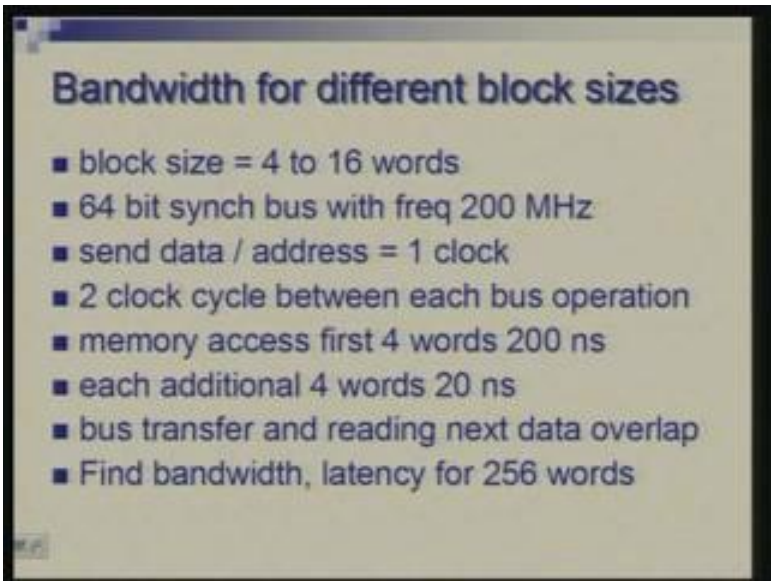
mechanism to improve the bus throughput. So let us get back to that example of block size and its effect on bus performance.

(Refer Slide Time: 05:35)



Just for your recollection here is the asynchronous handshaking protocol, one example we had seen and a synchronous protocol. So we are talking of a bus which is a 64-bit synchronous bus and frequency is 200 megahertz and we will see effect of varying the block size from 4 to 16. So now the protocol of the bus is as follows that it takes one clock cycle to send either data or address and we require two clock cycles between each bus operation.

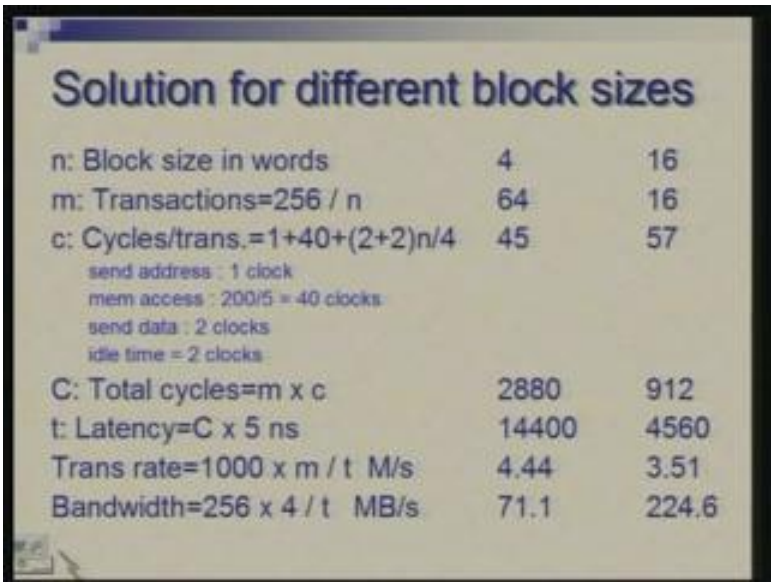
(Refer Slide Time: 06:13)



These are characteristics of the bus (Refer Slide Time: 06:15) and memory which is being accessed is capable of sending first 4 words within 200 nanoseconds and then each additional word requires 20 nanoseconds. So it has a mechanism within it that you do not need to repeat the whole transaction, it can actually get additional words within 20 nanoseconds.

We also assume that bus transfer and reading of next data overlap. That means when one data which is fetched from memory is being transferred on the bus the memory could be busy reading the next data. So now our requirement is to transfer 256 words on the whole either in groups of 4 words or in groups of 16 words.

(Refer Slide Time: 7:21)



So here is the calculation for finding the latency that means how long it takes to transfer or the bandwidth that means the total rate at which data gets transferred. So we are considering two possibilities: the block size is either 4 words or 16 words which I am just denoting by n, the number of transfer or transaction required for 256 words is 256 / n. so when you are transferring 4 words at a time you require this to be done 64 times, when you are transferring 16 words at a time you require this to be done 16 times to make a total of 256 words.

Now the cycles each transaction takes involves sending an address which takes 1 clock, then allowing memory to access the data which takes 40 clocks because 200 nanoseconds is the time given and 200 megahertz means 5 nanoseconds is the clock period so 40 clock periods, the data requires 2 clock cycles to be sent because memory is fetching 4 words at a time and the bus is 64 bit wide. Let us get back to this specification (Refer Slide Time: 8:43) memory access is 4 words at a time; first time it takes 200 nanoseconds and next time it takes 20 nanoseconds and the bus is 2 word wide so it requires 2 cycles to send this data and then between one transfer of data and the next one the bus needs an idle time of 2 clock cycles.

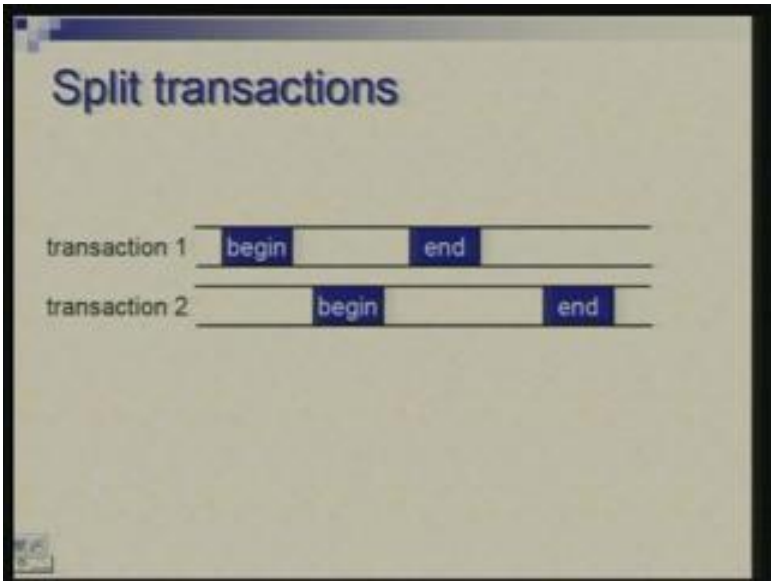
So now it is these figures which have been put here: 1 plus 40 plus this 2 into 2 which will take care of sending 4 words and if n is larger, then this part will be repeated (Refer Slide Time: 9:36). So in this case when the block size is 4 words only this happens only once, when the block size is 16 this happens 4 times so we are multiplying this part by n/ 4 which is the block size in words, n is the block size in words. So it takes 45 cycles here and in this time we have sent 4 words. Here it takes 57 cycles and we have sent 16 words in that time. So the total number of cycles is this c (Refer Slide Time: 10:17) cycle per transaction multiplied by m which is the number of transactions to make up 256 words.

So 45 multiplied by 64, product of these two numbers is the total number of cycles, product of these two numbers is the total number of cycles here. So now this is in terms of cycles. We can convert this into nanoseconds by multiplying it with 5 nanoseconds. So the total latency is this multiplied by 5 so many nanoseconds or 912 multiplied by 5 4560 nanoseconds. This is the total time required to read 256 words from the memory.

We can also talk of transaction rate; how many transactions are being done per second. A transaction means different thing here (Refer Slide Time: 11:15); transaction here means sending 4 words, transaction here means sending 16 words. So number of transaction in million transactions per second is 1000 times m the number of transactions over the time it takes. So it is 4.44 million transactions per second and 3.51 million transactions per second here.

The bandwidth is the total number of bytes being transferred per second. Bandwidth or throughput is, let us say, measured in megabytes per second. So we have 256 words multiplied by 4 which gives you bytes, so many bytes in time t gives you the throughput. So 71.1 megabytes per second is the throughput here and 224.6 is the megabytes per second is the throughput there. So there is a marked difference in terms of the throughput of these two buses.

(Refer Slide Time: 12:20)

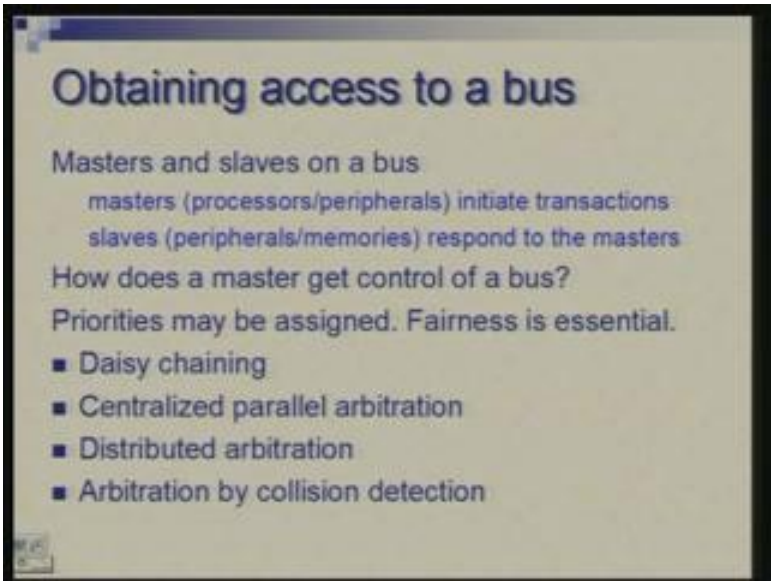


The other thing we talked of was split transaction. So if you look along the time axis, suppose one transaction begins here, for example, sending the address and sending a request to memory that you want to read then memory take some time, so in between the device which had sent this request is not using the bus so bus could be released and made available to somebody else and when the memory signals that the data is ready or if it is synchronous you wait for right number of cycles and then you can come back on the bus and read the data. Now, in between you can allow another transaction to begin which might end later on.

So now you have to properly link the beginnings and the ends. That means the device which send a request here should know exactly when it has to pick up the data as requested and similarly the one which requested here needs to pick up at appropriate time. But as it is very obvious, that utilization of the bus is much better here.

Now we have assumed that you can have many parties connected to a single bus. There could be processor, memory, I/O devices all could be in general sitting on a single bus so there are many conversations or many transactions between different pairs which can go on.

(Refer Slide Time: 14:06)



We have typically a concept of master and slave on the bus. Master is the one which initiates a transaction. it will initiate a request for read or write and slave is the one which will respond to this request. So typically let us imagine three different situations; processor talking to memory. So processor wants to get a block of data containing instructions or wants to write a block of data to the memory. So in this conversation processor would be the master and memory would be the slave.

Another scenario is that a disk drive wants to write into memory or read from memory. In this case the disk drive controller would be the master, memory would be the slave.

There could be another scenario that processor wants to initiate a transfer; processor wants to instruct the disk drive that from track number so and so, set number so and so send 1000 bytes of data to memory so that is the initiation process and there processor would be the master and the peripheral controller would be slave.

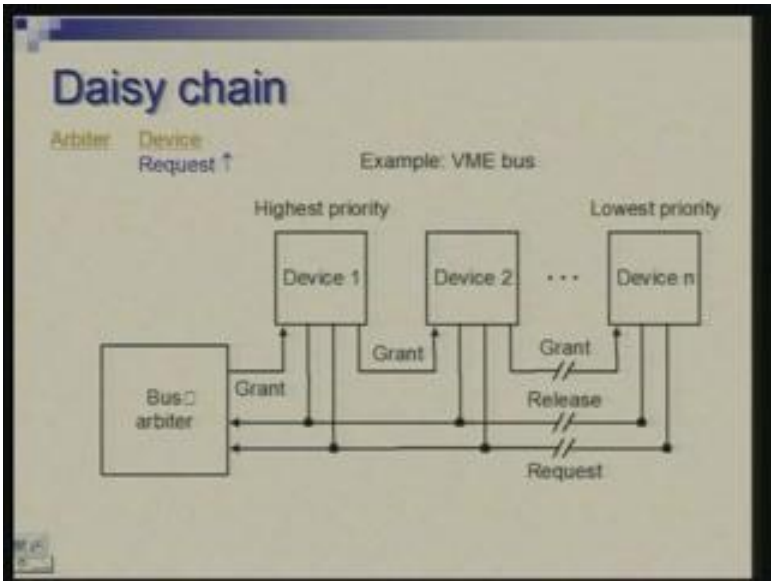
So among masters I have listed processors or peripherals and for slaves it is peripherals or memory. So peripherals could be slaves while talking to processor;
would be master while talking to memory. And processors are always masters, memories are always slaves.

Now with the possibility of multiple masters on a bus how do we coordinate among them, what happens if multiple masters have a need to transact on the bus simultaneously, so what is the discipline which has to be followed. It is an issue of getting access to the bus or getting control of the bus and using it and then releasing it. So, after bus has been released by one master, another master can use it. There has to be typically an arrangement of priorities. Somebody may have higher priority, somebody may have lower priority and this priority could be used to resolve the conflict; when multiple requests are there simultaneously one with higher priority needs to be given and these

priorities would depend upon what is the urgency, so there may be some transactions which cannot wait which have to be done on an urgent basis. At the same time fairness is essential. Whether a particular party as low priority or higher priority, it should eventually get its chance, so the usage should be reasonably distributed and everyone should get a chance.

We will look at four different mechanisms called daisy chaining, centralized parallel arbitration, distributed arbitration and arbitration by collision detection. Different mechanisms are used in different buses and we will see how these work.

(Refer Slide Time: 17:22)



Here is a scenario showing the first approach called daisy chain for resolving the access issue to the bus. here we assume that there are multiple devices; I am using the term device here in a generic sense, one of these could be processor, so these are all potential masters who wants to access the bus, we are not showing the bus completely, bus will have data lines and address lines and so on we are not showing those, we are only showing a few signals which will define the discipline of transfer of control of bus from one master to other master.

We assume that there is a block here called bus arbiter which will actually coordinate the whole thing. So the devices are arranged in decreasing order of priority. Highest priority is sitting closest to the arbiter and the one with lowest priority sits farthest away from the arbiter. There are bus request signals and bus release signals. There is a common signal on which every device can send a request and there is another signal on which a device which had the bus can indicate that it does not need the bus any longer.

Then the interesting part here is that a bus grant signal which comes from the arbiter is chained through all devices in a manner which is called daisy chain. So grant signal goes to device 1 which is the highest priority device, it may use it or it may pass it on to the

next one which may block it or pass it on to the next one and so on . So let us say a request comes from some device say device number 2, in response to that the arbiter will send a grant signal and device 1 does not need it so it will allow the signal to pass through to device 2; device 2 needs this so it will block it and all the devices further down will not see grant signal. It is a same signal which is trying to propagate through from one end to the other end and at some point it gets blocked. If there are multiple devices requesting for the bus, the one which is closest to the arbiter will block it first and the one which is downstream will not be able to see it. So that is how priorities are managed.

Therefore, the exact sequence of events which will go on is as follows:
The device, one or more devices, they will send a request and request is sent by let us say raising this line, bus request line to 1, one could have an opposite convention but just for the sake of explanation I am assuming that the request line is normally 0 and it will go to one indicating that some device is requesting. in response to this if the bus is free the arbiter will activate the grant signal and assuming that the device which needs the bus gets the signal it will then lower the request and then start using the bus. So it will use the bus for certain interval of time and then activate release signal. So I am assuming that release signal was also 0 initially and now it has become 1. When the bus arbiter sees that the bus has been released it will lower the grant signal and in response to that the device will also lower its release signal. So this is a complete transaction of acquiring the bus and releasing the bus.

Now let us try to understand what will happen if there multiple devices which are requesting the bus. So at any point of time let us say bus was free and the bus arbiter notices that there is a request for the bus. Now you can imagine that both these signals ‘release and request’ are wired, or in the sense that each device may send its individual request but effectively what you see on the bus what you see on this line is OR of all those. So a 1 here means that one or more devices are requesting for the bus; you do not know which one but all you know is that there is at least one device which is requesting for the bus.

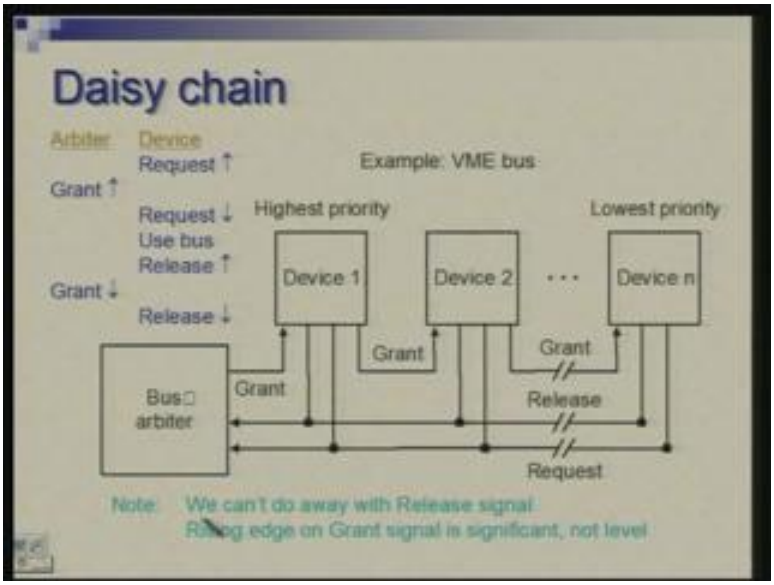
Bus arbiter does not worry about who is requesting and it simply activates the grant line indicating that yes among all of you whosoever has the highest priority can now use the bus. So naturally the device which is lower down in the chain will wait, the one which is higher up in the chain will get the bus. When the high priority device completes this cycle (Refer Slide Time: 22:58) it would have lowered down its own request but this line will stay high because there is a lower priority device which is still persisting and eventually when the bus grant has been lowered down, release has been lowered down the bus arbiter will still see that there is a request and it will give the grant once again which will be now seen by a lower priority device.

So eventually, if multiple devices had requested for the bus simultaneously or in overlapping manner then they will get the bus grant one by one; one device gets served and then when it is through, another device in the chain will get served, then next device will get served and so on.

What would happen if a device is getting served and a higher priority device comes up with a request now?
So, if a higher priority device comes it will find that grant signal is 1 and it might think that the grant has been made 1 for its own request but that could be a disastrous situation because the high priority device will hijack away the bus while low priority device was still using it. To avoid that we follow a rule that the device has to look at a transition on the grant signal, it should not just assume that a one level on the grant signal is sufficient to ensure that it has the bus granted, it has to see the grant signal going from 0 to 1. So now with this assumption if you have a high priority device coming in between when bus was already being used by a low priority device it will find grant signal 1 but it will not see a transition 0 to 1 on the bus on the bus grant and therefore the situation which I mentioned will get avoided.

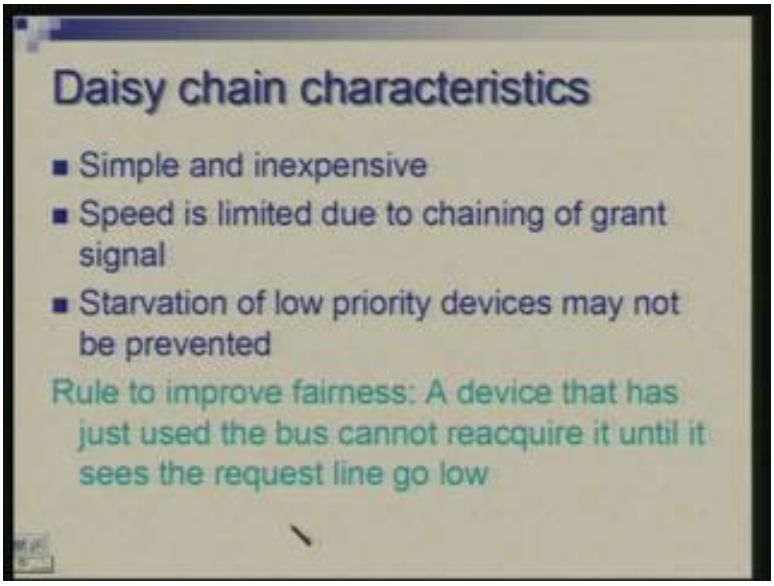
I have already mentioned; raising edge on the grant signal is significant and not the level.

(Refer Slide Time: 25:23)



There is another point here; one may question that why do we need release signal, why not we simply manage with a grant signal and a request signal. If you do not have a release signal then the bus arbiter will not have a clear indication of when the usage of bus is over because a device which is using the bus would have removed its request but since other request are still persisting you will not know this change so you need another signal to indicate that bus is being released so this is important.

(Refer Slide Time: 26:10)



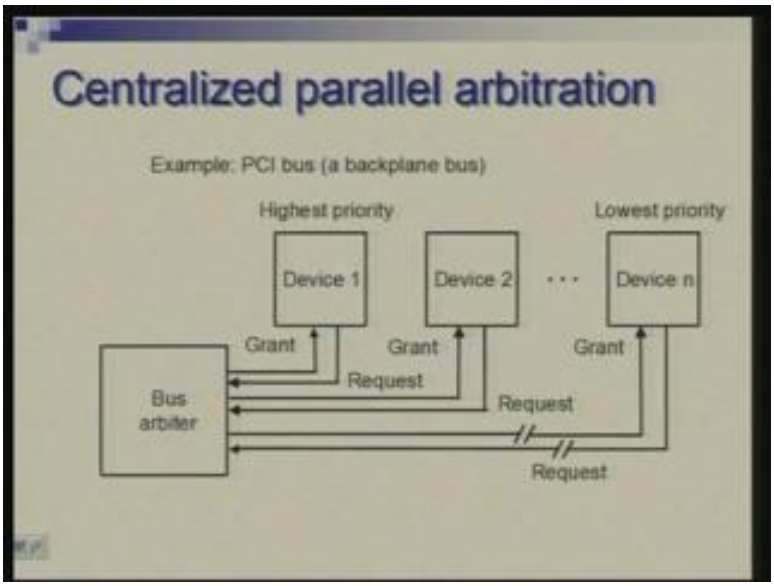
Now this arrangement is very simple and inexpensive, the arbiter is very simple it does not have to worry about who has what priority, it simply looks at the request and release and activates grant or deactivates grant signal. The problem with this is that the speed could be limited here if the chain is very long. If the chain is longer, as we have seen in carry propagation, there is a delay, so we have to allow for the maximum delay before we can actually decide whether the signal has changed from 0 to 1 or 1 to 0. So, if the chain is long the operation will get slowed down.

The other problem is that in this we have not ensured that a low priority device does not starve. What could happen is that suppose it could happen that high priority device can keep on shuttling the bus in between them and a low priority device can always keep waiting. Suppose device 1 and 2, let us say 1, 2 and n (Refer Slide Time: 27:22) all had requested so 1 gets served then 2 gets its chance but before 2 finishes, 1 as another request. So as soon as 2 releases grant is given again, device 1 gets it and meanwhile request from 2 could come and so on.

So bus could keep getting passed on between high priority devices and some devices at low priority could get starved. So the solution to that could be as follows; that a device that has just used the bus should be disallowed to reacquire it until it sees the request line go low. The meaning of this is that if you have used the bus there may be other requests pending so request line will continue to be 1 till everyone down the line has been served. The request line goes low only when no device is requesting. That means everyone who as requested some point of time is served. So if a device does not make a second request before everyone else who was in the queue has been served then this problem will go away.

The second solution which I mentioned about arbitration is the centralized parallel arbitration.

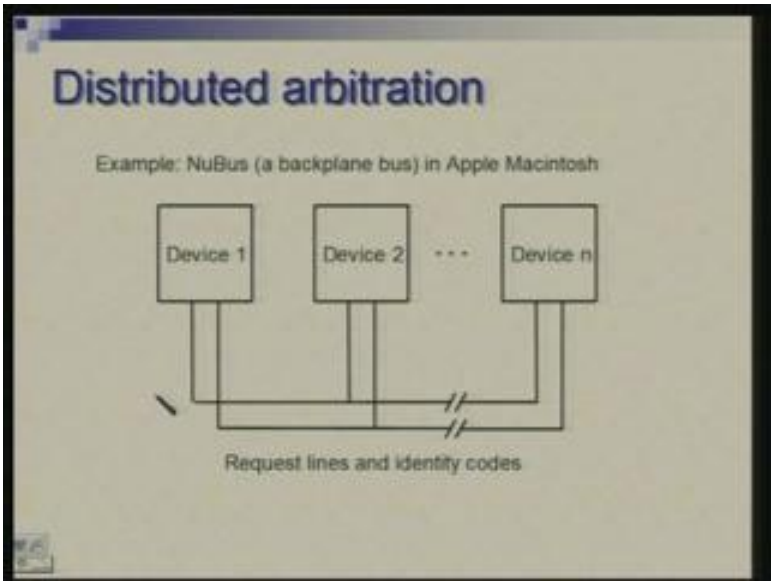
(Refer Slide Time: 28:50)



Here we have, again many devices which can request and there is a central arbiter which can send grant. Now we assume that there is a separate request line for each device and a separate grant line for each device. So this is the request line for device 1 and a grant line (Refer Slide Time: 29:12), request line for 2 grant line for 2, request line for n grant line for n. Now the whole logic is contained in this arbiter. Arbiter is supposed to look at all the requests and issue them grants individually. Then the priorities of the devices could be hard coded into this arbiter or could be defined dynamically and this could in a fair manner assign the bus or give the grant signal to various devices turn by turn.

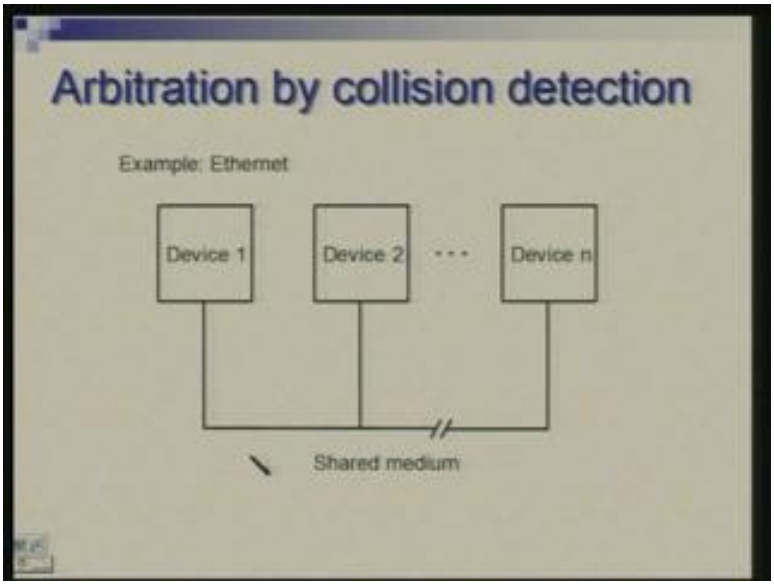
So here the priority need not be defined by the position the way we did earlier; although in the picture I have shown that this is the highest priority and this is the lowest priority but the position where you are located is not necessarily the priority because there is no chaining here (Refer Slide Time: 30:13) so you can arbitrarily define priorities of various devices and the resolution of multiple request will take place here. So the kind of mechanism which is being followed in the previous case in somewhat a distributed manner would now get concentrated in a single arbiter. So arbiter here is more complex than what we assumed in the previous case.

(Refer Slide Time: 30:40)



We can distribute this task of arbitration among the devices themselves. So it is a more democratic kind of system where all the devices sort of negotiate among themselves or collectively decide who gets the bus. So you have several control lines on which devices can send their request and also put their identity on some specific lines. So each one can see who all are requesting at any given time and the identities are all available. So let us say device 2 sees that device 1 is requesting and device 5 is requesting and then if it understands that among these three among 1, 2 and 5 it is not the highest priority it would stay back and allow the one which is highest priority among all these to go through. So the priorities of various devices are understood by each other and in a honest manner all devices are supposed to look at what is going on the bus, what requests are there on the table and make a choice accordingly.

(Refer Slide Time: 32:02)



Then lastly we have arbitration by collision detection. Example of this is Ethernet where you have, similar to the previous case you have shared medium over which all devices are connected and here you notice that each device simply sees that if the bus is free, you try to initiate your transaction and if multiple devices happen to do so then there is a need to figure out that whether collision occurred or not. So what you do is you first check if the bus is free, then try to start and then check if what you sent did collide with something else or not. If no collision takes place that means what you see on the bus is what you wrote that means there is no collision and it means you have the access to the bus and you can continue.

On the other hand, if you collide that means you try to write something but because somebody else also wrote something on the bus what you saw was a combination which is different from what you wrote, that is an indication of collision so you back of and try again after some time. To ensure that both devices which collided do not try again at the same time, this delay is modified randomly. One device may try after 5 microseconds and the other may try after 15 microseconds and therefore unlikely that they will clash again. For some reason there is a clash again you just repeat the process.

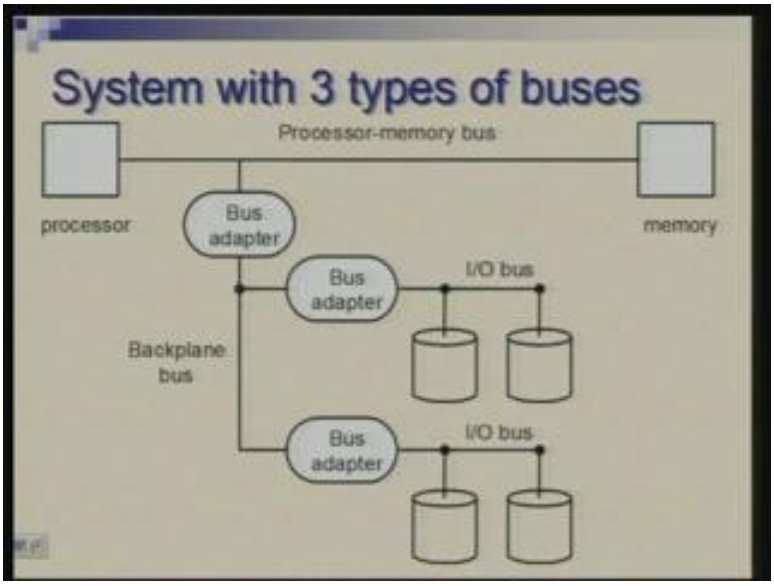
So it is again very simple but obviously time gets lost in because of collision. If the density of the traffic, if the usage requirement for the bus is not very heavy the collision will not be too many and therefore it will work efficiently.

[Conversation between student and Professor: (34:08)]..... yes, it could get starved. If the request keep on coming from high priority devices and priorities are fixed and we do not follow a rule of the kind I mentioned earlier then starvation could occur. So you could either have somewhat restricted rules so that repeated request actually gets held back or you could also talk of modifying the priorities dynamically, the priorities could rotate.

For example, a device which let us say use the bus kind of goes back to the queue end of the queue and stand in the queue again that could be the lowest priority and therefore the priority could dynamically modify.

Now, we talked of different types of buses and one scenario which I had shown was where we had three types of buses.

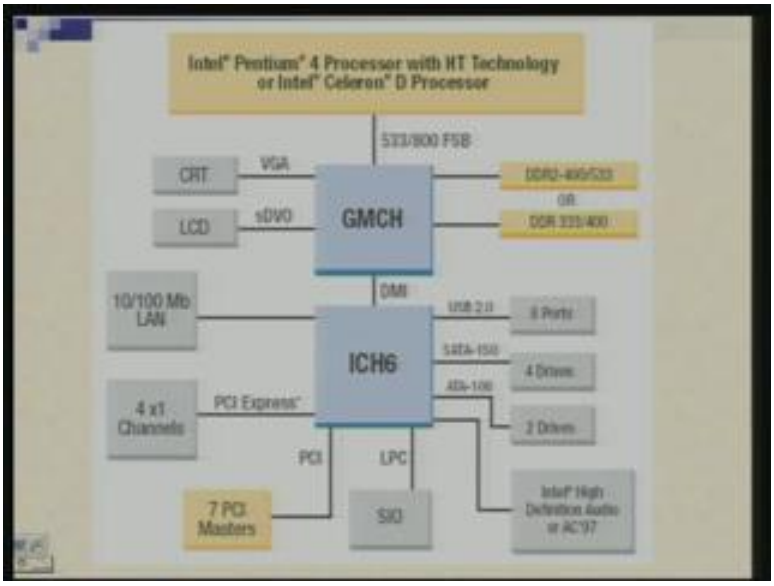
(Refer Slide Time: 35:11)



I am showing it again here. You have a processor memory bus, typically a proprietary bus connected to processor memory and through an adapter you have a backplane bus hooked to it. So backplane bus is the one where I/O bus directly or indirectly and processor memory, directly or indirectly all connected. The multiple I/O buses could be there connected through different adapters to the backplane bus. So for example, one I/O bus could take care of disk drives, CD ROM and so on; the other could take care of maybe printers, scanners and so on.

Typically the I/O buses are standard, processor memory buses are proprietary, backplane buses could be either of these two. More often they are also more and more becoming standard, the speeds obviously are highest at processor memory bus level and lowest at I/O bus level, the PM buses will tend to be synchronous, I/O buses would tend to be asynchronous but there are examples of both kinds. Backplane buses are also now generally synchronous. This is a kind of oversimplified situation. If you look at a real system things may be somewhat different.

(Refer Slide Time: 36:48)

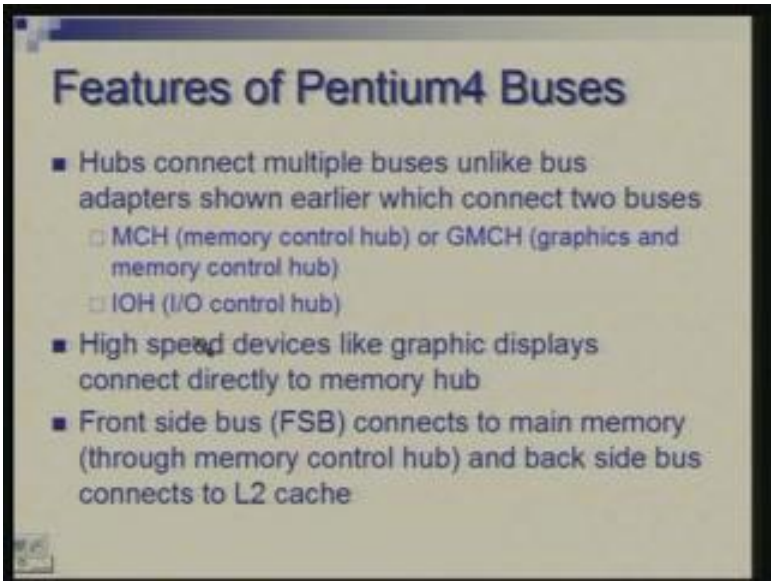


I am showing an example of a typical Pentium 4 type of architecture where you have..... basically this is the processor and you have two controllers or two hubs, instead of adapters we have these devices, actually these two forms what is commonly called a chip set. You have a chip set around which a motherboard is built, you have a processor and you have a chip set which interface most of the other things and these actually characterize a particular motherboard. So there two complex chips here: this is called MCH or GMCH (Refer Slide Time: 37:37), MCH stands for Memory Control Hub or GMCH Graphics Memory Control Hub, ICH is I/O Control Hub. So on this side it connects to memory modules, on this side it connects to the display modules; it could be CRT monitor or LCD monitor or it could be simply a video out and this connects to variety of peripherals including a PCI bus here (Refer Slide Time: 38:10) which is a backplane bus and there are several I/O buses; for example, this ATA for disk drive, USB for variety of devices and so on….. LAN.

In looser terms sometimes some people call this as a north bridge (Refer Slide Time: 38:29) and this as south bridge just because of the way they are typically placed in the diagram. So you would notice in some cases that frequencies are also given. this is a bus which is connecting the hub here and the processor and it would typically run at 533 or 800 megahertz, this is called the front side bus and this is designed to connect to memory modules which will run at 333 megahertz or 400 megahertz or 533 megahertz. I think rest of these.... this is USB, SATA, ATA, these are basically disk drive interfaces, this is audio interface, this is S I/O which is another kind of serial I/O, PCI, PCI express it is again another derivative of PCI and this is LAN.

So now in which way is this different from the diagram we had drawn earlier.

(Refer Slide Time: 39:48)



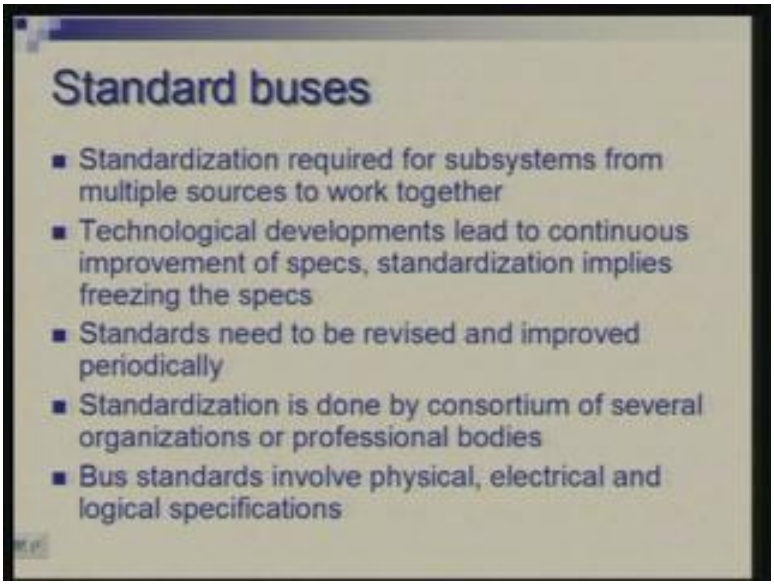
Here we have, what you would have noticed is that we have hubs rather than adapters. We talked of adapters trying to connect two different types of buses but hubs are somewhat more complex, we are connecting multiple buses.

High speed devices like graphic display are connecting directly to memory hub and not to the I/O hub. Here we had an impression that (Refer Slide Time: 40:24) all the I/O devices are coming through I/O bus to the backplane bus and then to the memory processor bus. But here, particularly, these display devices are not connected to I/O control hub, they are connected to the memory hub, the reason for this is that the transfer demand the bandwidth demand here is extremely high; as compared to all these this is the highest.

If you recall, the first lecture on I/O, we looked at a variety of peripherals and tried to get a feel of the speeds or the throughput requirement of various devices and we noticed that the display has the highest throughput requirement and therefore this connects directly here. Then we talked of a processor memory bus here, but actually speaking although it did not show up in this diagram there are two buses: One is called front side bus and other is called backside bus. Front side bus is the one which connects the main memory through the memory control hub and there is also a backside bus which connects L2 cache which is a faster bus that is not accessible outside the motherboard, it is within the processor because L2 cache here is on the chip itself.

There are numerous varieties which are possible when it comes down to real system; although conceptually we have seen two three different possibilities there are lots of variations which are possible.

(Refer Slide Time: 42:12)



I have been mentioning the term standard bus. Actually why we need standards and what these standards are?
Standardization is required so that subsystem from different manufacturers could talk to each other. If that were not the case, if there were no standard buses we would expect the entire system to be built by one manufacturer so that compatibility is ensured. But once you define a standard, it is a common interface, so one company can build the processor, one can build memory, one can build different I/O peripherals so each single company may not necessarily specialize in all the areas and therefore it is a better situation that you can allow multiple parties to build different things in which they are good at.

Now, but as technology develops, these speeds change and various requirements change. On the other hand, when you say something as been standardized, you are freezing all the specifications. So these are two contradictory requirements. On one hand you want things to improve. For example, when you saying that you have defined a bus with 200 megahertz so you are freezing everything at that; you are saying that if I make one device it should be compatible with 200 megahertz, , you also make it compatible to 200 megahertz and we have sort of agreed and frozen it at that.

But suppose I make my devices better and I would like to run at 300 megahertz you also want to run at maybe 350 megahertz, so, by standardizing you are sort of freezing and arresting the growth whereas the technology will like to and commercial pressure will like to push it in the other direction. So what you need to do is you need to keep on revising and refining your standards. You have standard which is version 1 then you get version 2 where some of the things get redefined, all the performance specs go up and this process has to be a continuous process.

Therefore, you have to have formal mechanisms of defining these standards and typically these are done by either groups of industries which are formed, industries and other

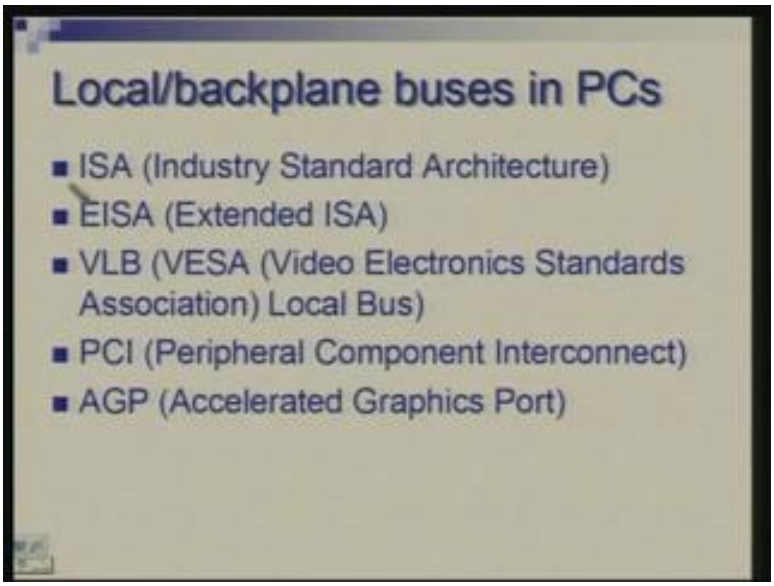
bodies could also be there, so you have consortiums formed which take the responsibility of collectively defining standards or you have professional bodies such as IEEE or ICU and so on, ITU, they have again representation from various organizations and they define the standard in a manner which are acceptable to larger community.

Sometimes what happens is that a proprietary interface or proprietary mechanism which becomes very popular gets adapted as a standard. so others see benefit in following what a popular person is doing.

Now, what exactly is standard?
Standard, as far as buses are concerned, is defined at various levels: at physical level, at electrical level and at the logical level. At physical level you need to define exactly the shape, size, dimension of the connectors or the cables; at electrical level you need to define the voltage and current levels, the impedances; and at logical level you need to define the meaning of each signal and the sequence in which signals change and the events take place. So a bus standard is a complex definition spanning from physical level to the logical level.

Let us look at a few examples of various kinds of buses.

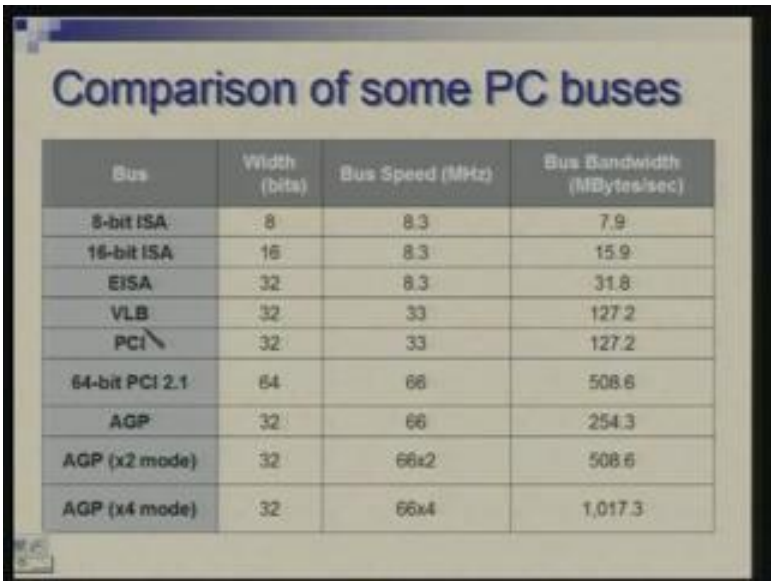
(Refer Slide Time: 46:25)



So, in PC domain, in early stages we used to have ISA bus which stands for Industry Standard Architecture. So, that later on got extended to EISA or Extended ISA bus, then further down the line.....now, ISA and EISA were trying to connect everything but then later it was felt that processor memory could be connected through a faster link and peripherals need to be connected on a slower link. So VLB or VESA local bus; VESA stands for Video Electronics Standards Association; a local bus is defined and ISA or EISA could get linked to this. Then subsequently PCI bus was defined stands for Peripheral Component Interconnect bus, it is a backplane bus and this also has seen

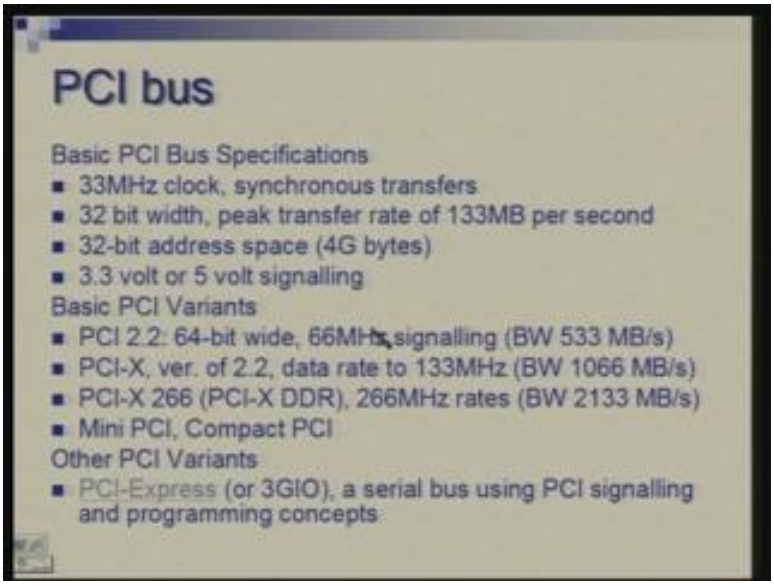
several revisions, I will come to this in a moment and AGP stands for Accelerated Graphics Port on which the display devices get connected.

(Refer Slide Time: 47:40)



So here is a comparison of some of these buses which I just mentioned. ISA bus was earlier 8-bit, later on became 16-bit; EISA is 32 bits; VLB and PCI are also 32 bits, but later versions of PCI are 64 bits and AGP is a 32-bit bus. The frequency for ISA and EISA it is 8.3 megahertz, VLB and PCI 33 megahertz, later versions of PCI 66, AGP has gone from 66 to double of that and quadruple of that and even AGP 8x also there on the same. So, depending upon the frequency and the width the throughput is given in megabytes per second. So here 1 megabyte is meant to represent 2 raised to the power 20 and not 10 raised to the power 6 that is why you will see some discrepancy but approximately you can get this figure by combining this and this (Refer Slide Time:48:55).

(Refer Slide Time: 49:00)

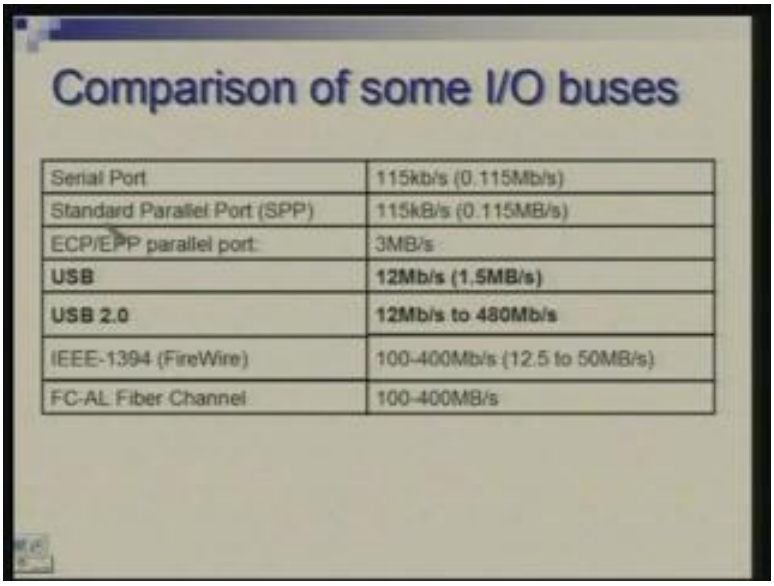


Let us spend a couple of minutes on PCI bus which is invariably there in all the PC systems now. The basic PCI was 33 megahertz synchronous bus, width was 32 bits and accordingly peak transfer rate is 133 megabytes per second, so this is the peak transfer rate. That means if the bus is continuously transferring data it could transfer at that rate but actually because of protocol delays and idle time, the transfer rate would be much less. The address which flows on this is 32 bits which can address 4 gigabytes of memory, in terms of voltage it could have 3.3 volt or 5 volt there are two variations that are possible.

Then later development on PCI lead to PCI version 2.2 which is 64-bit wide 66 megahertz and therefore the overall performance is roughly four times both these factor get doubled (Refer Slide Time: 50:02) then there is a PCI - X version also which is having a data rate of 133 megahertz so bandwidth is twice that and it has a 266 version 266 megahertz and bandwidth is more than 2 gigabytes per second.

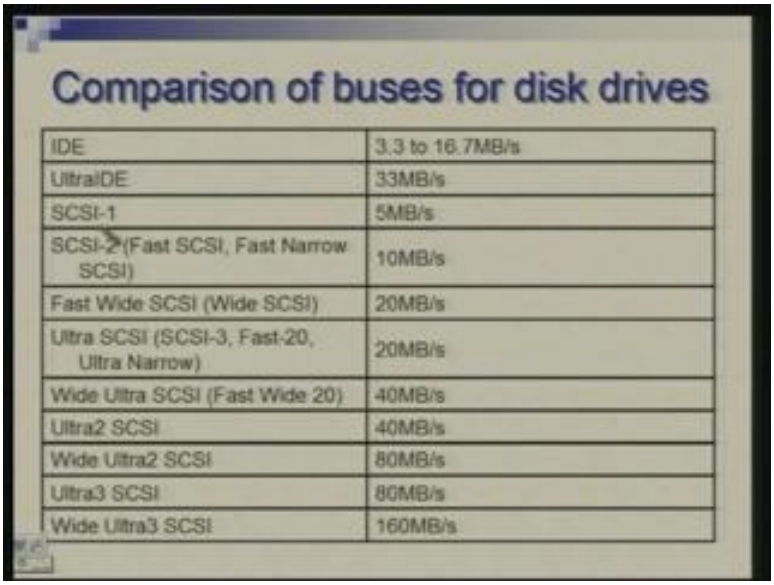
There are other variations like mini PCI or compact PCI and there is also serial version PCI express. It is a serial bus which follows signaling like PCI. We are not going into details of what signals are there, that itself will take several hours if we have to go through that in detail but that is just a serial version. Serial buses are typically cheaper because they have to carry very few wires and therefore the cables are cheaper, the connectors are cheaper and on the whole, cost is lower. But obviously if you are sending one bit at a time the total data rate gets reduced. So here is the comparison of some of I/O buses.

(Refer Slide Time: 51:25)



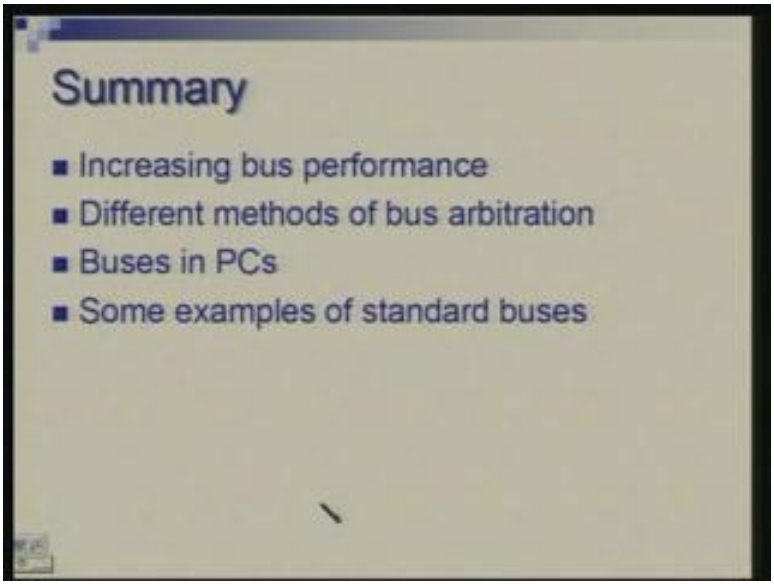
You have serial port on which sometime you connect external modems, parallel port on which you typically connect printer, this is some extended parallel port, USB on which I have connected this flash memory device, you can connect devices like cameras, printers, scanners and so on. Then there are other high performance serial ports like Fire Wire fiber channel. You can see that there is a wide range in terms of throughput rate starting from a fraction of megabytes per second going all the way upto something like 400 megabytes per second. Then these are some of the I/O buses to which disk drives, CD ROM drives, DVDs they connect.

(Refer Slide Time: 52:25)



IDE is a very old one, ultra IDE, SCSI which stands for Small Computer System Interface, it has again, you can see how the standards have evolved from SCSI -1 to 2 to 3 they are ultra, wide, fast you know all these are prefixes, they keep on getting added and the standards improve. So, starting with the 5 megabytes per second all the way upto 160 megabytes per second, there are various steps in between. So this is not all, these are only some of the standard buses which we have seen. I will close at point.

(Refer Slide Time: 53:00)



In summary, what we have seen today is we had seen the factors which influence performance of a bus. In particular, in detail we saw how variation of block size would change the performance, we saw it quantitatively. We looked at different methods of arbitration of bus when there are multiple masters trying to get hold of the bus and the requirement there is that it should be a mechanism which can support priorities but at the same time it should not lead to starvation there should be a fairness.

We looked at the organization of buses within PCs and we very briefly looked at some examples of the buses at the backplane level and at I/O level. Within I/O we saw series of buses which are for hard disk drive or CD ROM drive or DVDs and we have seen that as time progresses the buses have to be refined and redefined, the standards have to keep on changing to keep pace with the technology. Thank you.