Computer Organization
Part – IV
Buses
Prof. S. Raman
Department of Computer Science and Engineering
Indian Institute of Technology, Madras
Lecture – 31
Buses

In this last part in our series of lectures on computer organization, let us take a look at the bus. What is a bus? So far we had seen the blocks of CPU, memory, I/O and the interconnecting bus. On more than one occasion, I had pointed out that we have to evolve some kind of standardization so that the system from the bus we will see something uniform because you may be having different types of memory systems and you may be having different types of devices but the CPU must see something uniform on the bus on the bus end. In spite of all these, though we talk about standards, we would find that there are different standards; obviously because the CPU and memory work at some rate different from I/O. If at all CPU is directly involved, that is going to be at another rate; then, memory I/O will be a different rate; and in I/O, you have a spectrum from the lowest K speed to the fastest one. So generally, we would find that there are different types of buses and when a processor comes with some brand name in the market, you would find that that particular manufacturer comes out with its own bus also.

(Refer Slide Time: 05:40)

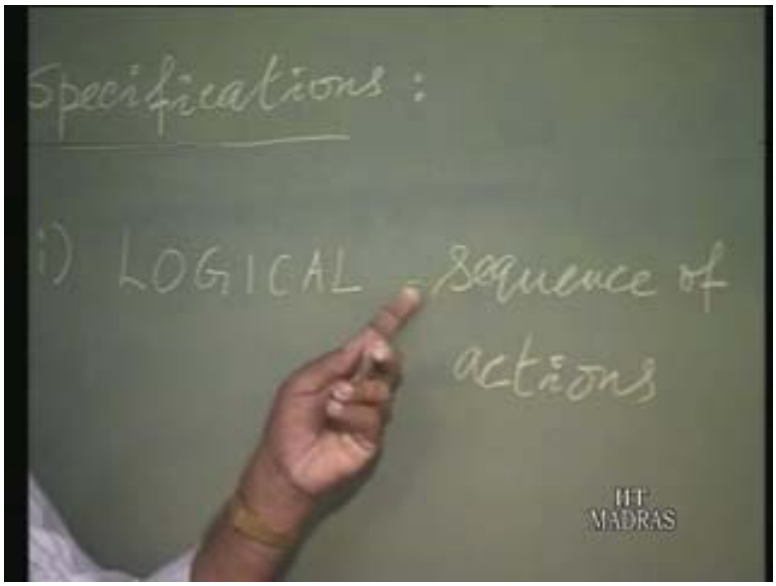
By the time that particular bus gets standardized, you would find that the bus is no longer very relevant, mainly because the processor is outdated. Nevertheless, we can always talk in the case of bus; in general, about the specifications of the bus. Here there are certain things which you may find are common and there are certain things which would keep varying. Now while specifying a bus, what exactly is the bus; what is it we have talked about earlier? We have always said a bus is a set of signal lines. We had introduced the bus as a set of signal lines.

Bus in fact is a term introduced by Americans; earlier the British used to call it highway. Now that particular term is coming in a different context like super highway for information and so on. Earlier they were calling it highway. Essentially on a highway something moves; but Americans used the bus. Now in the case of computer system, we can say the bus is like a highway over which the data moves – there is some communication. But Americans called it bus – since bus is a set of signal lines, these signal lines are carried over lines, which form conductors. So the bus has conductors and a bus is usually long; while trying to meet an electrical specification, you will find that you would need some current amplifiers, we call them current drivers. That means along with the bus or as part of a bus, you have drivers. So you can see that a bus has drivers and conductors and we are quite familiar with the term bus.

Now let us go into certain aspects of the specifications. There are actually three types of specifications but usually in computer organization, or we may say even computer scientists will be concerned more with only one thing. That particular thing is called the logical specification of the bus. In fact all the time when we were talking about some transfer over the bus, we were indicating this. What is it we said earlier? We said the CPU places address on the bus and if it were a read cycle, the CPU also places a read signal on the bus, and then the memory responds with data and it puts it on the bus. CPU, of course, reads it in. So you can see that there is generation of address, signal generation of read signal and strobing the data that's available.

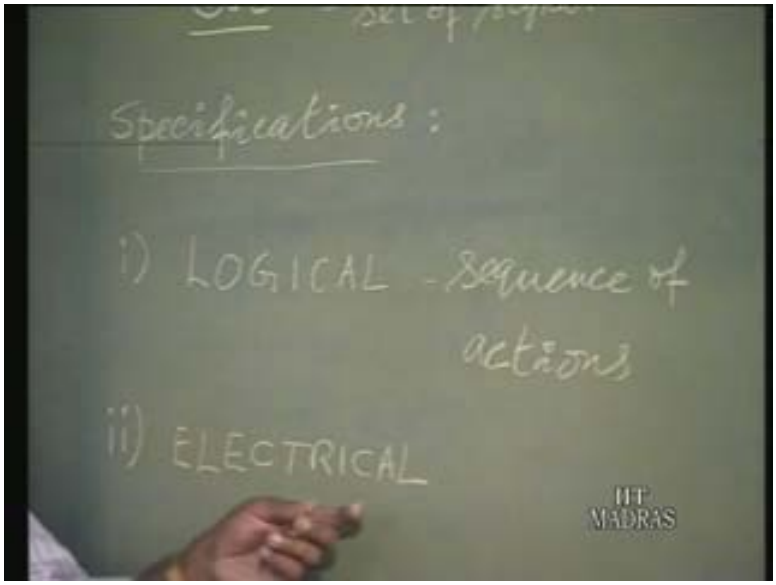
This is what we were talking about all the time. Now what is the data? Data is a pattern of 1s and 0s; it is a string of 1s and 0s. Similarly address also is a string. So all the time we were only talking about the logical aspect of it because we never said the data which is 1 means say 1 volt; 0 means say 0 volts or any such thing; we did not talk about any physical, real-world quantity. We were not referring to any physical quantity. We are talking about a read as a read signal, write as a write signal. We never said read means what must be the voltage and so on and so forth. So that is what we are talking about when we talk about the logical specification; another thing as you would have noticed is that we are talking about sequence of actions.

(Refer Slide Time: 07:47)



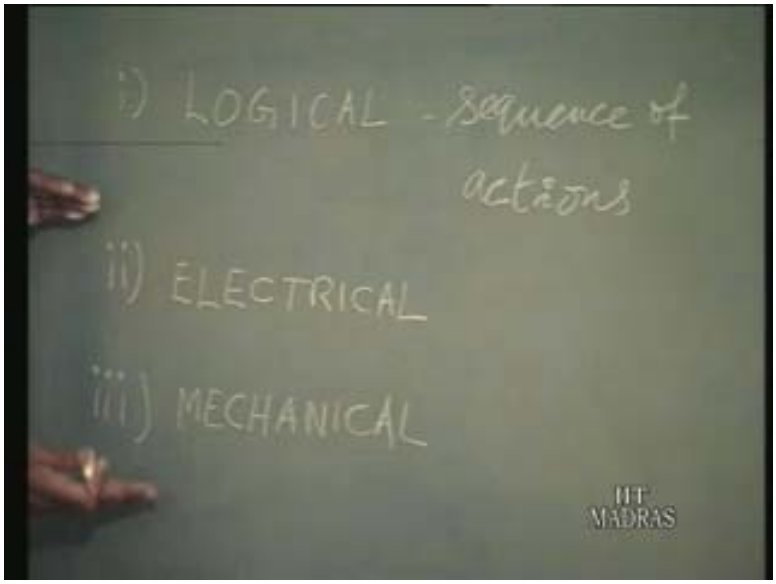
What are the things we were mentioning? We said first address, then read and then the data comes, which is strobed, so we were talking about sequence also. In other words, the sequence of actions which is part of what we refer to as bus protocol, is only one specific way in which this read or write can be performed. So we talk about logical signals involved and then we also talk about the sequence in which the signals must be generated. Now generally when we discuss the bus in computer science, or from a computer scientist's point of view, we will always be concerned with these logical aspects. But later as I was just trying to point out, there are two things: one is the electrical specification.

(Refer Slide Time: 08:51)



So when we talk about the electrical specification of the bus, I have to say for instance what does 1 mean? Do 1 and 0 there refer to say some voltage signals or current signals or some other signals? And if for instance 1 and 0 will be represented by plus 5 V and 0 V or 3 V, plus 3 V and plus 0.2 V or let us say plus 15 V and minus 15 V. All these things really refer to the physical quantities. They would come into the electrical specification of the bus. It is not enough if we say that the address is placed. We have to specify exactly what must be the levels of the voltage or current signals. Essentially we can just put it as physical specification, but then the mechanical specification of the bus also has to be specified.

(Refer Slide Time: 10:19)



So together, these two form the physical part of the bus and this forms the logical part of the bus. One talks about the electrical signal levels and so on, the other one talks about the mechanical thing. So this would say how long the bus conductor can be and what sort of edge connector is used – is it something like 32 pin or 62 pin and if it is this pin, is it a parallel or are the pins parallel or staggered, etc. For instance, if we talk about the standard bus, we have specification along all the three dimensions. If we say multi-bus, there is a specific multi-bus connector and each of the multi-bus signals is going to be defined in terms of some electrical voltage current and so on. And then we also talk about this particular one.

So essentially we will only concentrate on this. But let us not forget that there are these specifications also; somebody must concern with this; otherwise there is no standard. Now why must we be concerned even down to the level of mechanical? That is mainly because this is the one which is going to give freedom to the user. So if we say use multi-bus one connector or multi-bus two connectors or some other X bus, new bus, and then all he does is he goes to the shop and ask for that connector and then brings it and then machine properly without any difficulty.

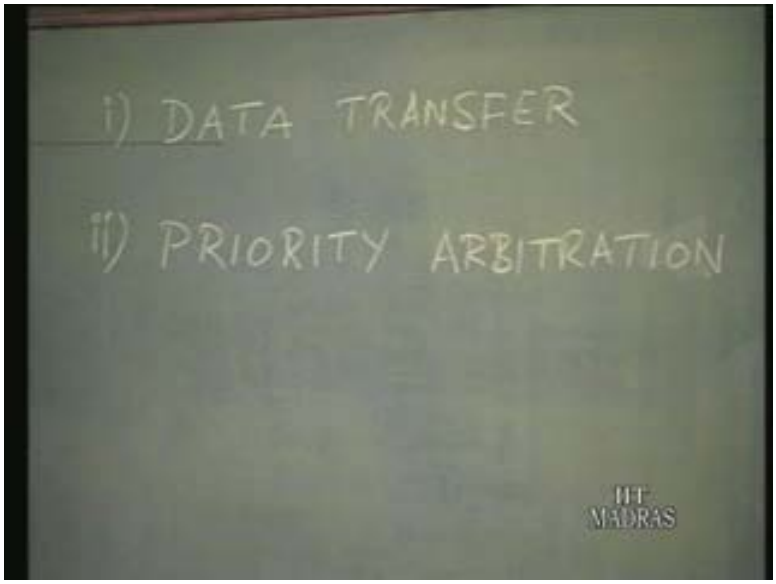
He is not going to be bothered about either logical aspect or electrical aspect; for him, when he keys in, there must be a display and that particular display must mean something to him at a higher level. Having said that, we will be concentrating on the logical aspects of this bus. We can note that essentially there are again three sections. The first one we may say is concerned with the data transfer.

(Refer Slide Time: 12:39)

This is the one we keep talking all that time about. That is, we say that the CPU places address, indicates what type of transfer, read or write, input or output, whatever it wants, and then memory or I/O responds. So all these things: placing the address, placing the appropriate control signal, and then passing on the data will come under the data transfer aspect. Now as the CPU is involved in this data transfer because what is going on in any instruction cycle again and again is the same thing, fetching an instruction, interpreting, executing, as part of executing fetching a data, that is, instruction or data fetching – it all comes under the data transfer.

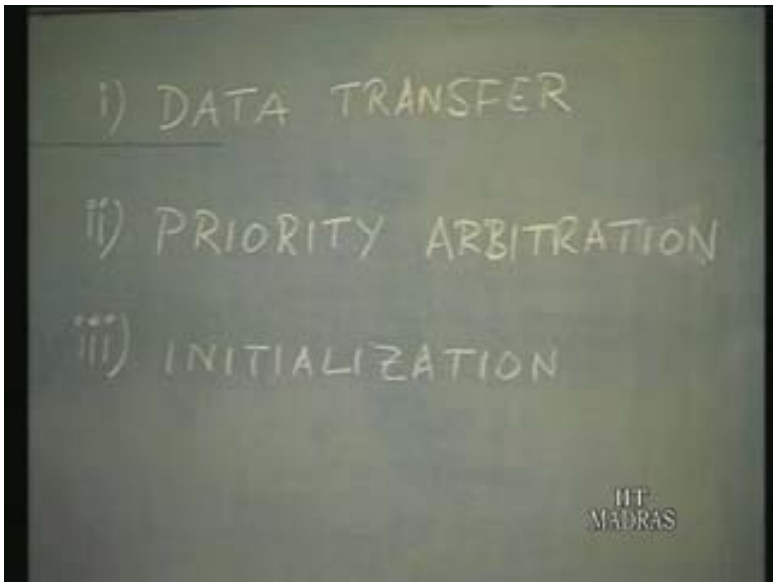
This is what is going on in every instruction cycle. Now this is the essential thing as far as the processor is concerned. As this goes on some other device may indicate that it is ready; in the case of interrupt, is it not. Now in a system which has n number of devices, that is, multiple devices, we also said that we have to look in to the priority among these when there is a multiple request for this CPU attention. So we have to basically see that there is priority checking and so on. We may put this particular one as priority arbitration; meaning as CPU is busy with the transfer, then possibly there is some other specialized hardware unit, which looks into the action. Priority arbitration can go on in parallel with the data transfer. For instance as part of the some instruction cycle, when one of the n devices or two of the devices indicate that readiness, then there is a conflict. Now the priority arbitrator will look in to the requests and then will choose the higher priority device between those two so that when the instruction cycle is complete it can go ahead, that is, in the case of interrupt.

(Refer Slide Time: 14:24)



So priority arbitration is another piece of action that goes on and we will have dedicated signal lines as part of the bus because then only two things can go on in parallel; otherwise it is not possible. If the same sets of signal lines are going to be used, one has to keep idling. For instance that was the situation in the case of DMA. The I/O is directly accessing the memory so CPU is going out of action. The third aspect of this bus we may just put in general as initialization.

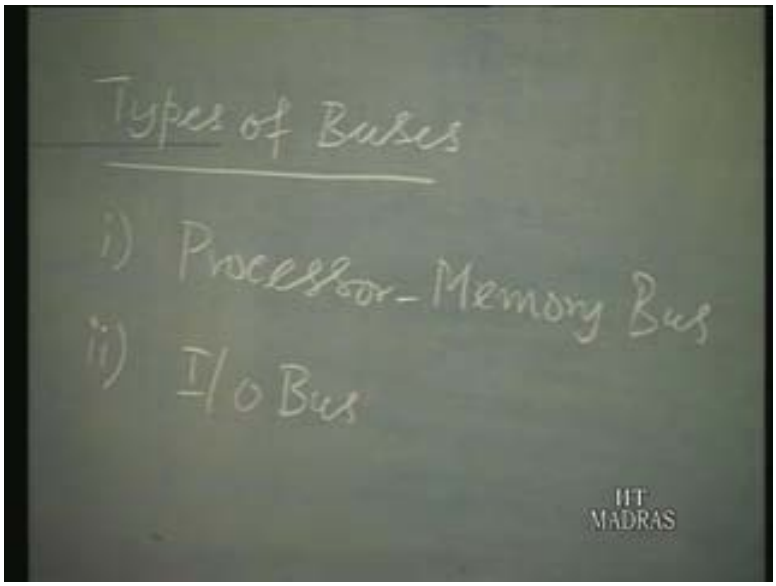
(Refer Slide Time: 16:05)



Different people may call it differently. What exactly we mean here is something to do with checking about the power, the system clock, and few other signal lines, which really not take direct part in data transfer or priority arbitration. There will be another set of signal lines.

We may just call them for instance system reset signal. So that can come under the initialization and a few other things: system clock, system reset, some special signal lines, monitoring the power – all these things will come under this. So we as we talk about the data transfer; we should also take a look at what is going on in this and also know the functions of some of these because there is no standard about this particular thing. So generally when you study any bus, you may be able to identify a group of signals belonging to this or this or this category. Now there are different types of buses, we may say. Actually I am not talking about just the standard buses here. I am just trying to give what you may call a generic or general classification. What are the various things involved? For instance, we are not talking about multi-bus or a new bus or uni-bus or a mass bus and so on; we are discussing purely from functional point of view. Now we know that CPU and memory, both work at electronic speed. So it is meaningful to have a processor memory bus and have it somewhat different, distinct from what you may call the second one as an I/O bus mainly because in the case of processor memory, the transfer is going to be very fast whereas in the case of I/O bus, we do not know.

(Refer Slide Time: 18:19)



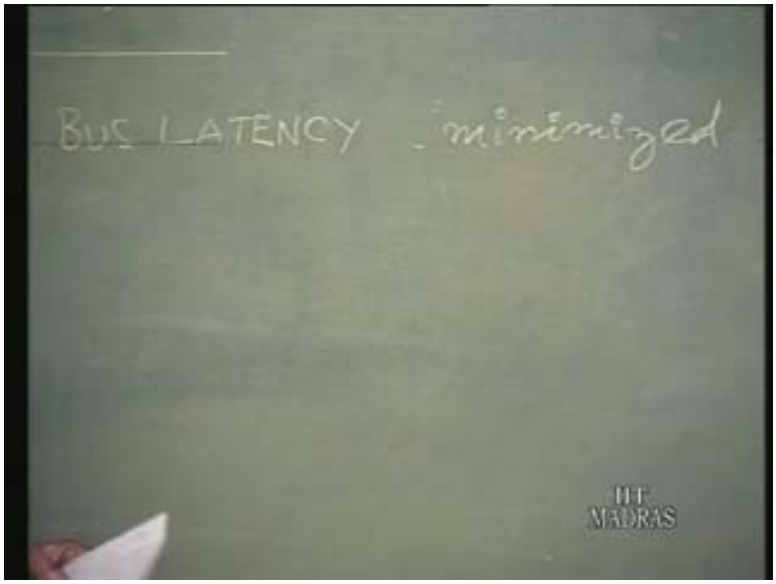
We have devices with varying speeds, characteristics, and what not. Then there is another bus also that we talk about; that is generally called a back plane bus. In some systems we may not able to identify them separately. For instance the processor of the memory bus itself may act as a back plate bus; it is not necessary that all the three must exist in all the systems. Now let us say some one is looking up the system with an Intel processor. Then suited with that particular Intel processor, there may be certain memory chips. It is even possible that the manufacturer of the processor or the CPU has also come up with a set of memory chips, which would directly talk in the sense there will be some special features about that memory. For instance let us say suppose you have a processor with multiplexed bus, there is let us say address and data multiplex. If you have the memory chips or the memory controller, which goes with the chip in the memory bus system, it can take care of the multiplexing, so that internally it buffers and then de-multiplexes that. Then it is meaningful; and so what happens is this processor memory bus essentially we may say is a short bus and also it is a bus which does transaction as fast as possible.

We can understand fast because, for this processor utilization to be the highest, maximal, it must be fast and invariably the processor, memory, and the bus are interconnecting the processor memory bus on a single board itself. That is why invariably we define that particular thing as a short bus and there may not be any standard about this also. For an Intel processor, there may be a set of things; for a Motorola processor there may be another set of things. It is all because of the signals that are generated by the respective processor. We can say that this processor memory bus is a proprietary bus because we have a specific processor and then we have specific memory requirement; it is not open for the general use. Now in the case of I/O bus, it is a different story. First of all, we have different types of devices to be connected; and second thing is that whereas in the case of processor memory even at the time of the design it is known how much of memory it has, at the time of the system installation, we do not know how many devices we want – may be during installation we would like to add a few more devices. Generally we would find this I/O bus is in contrast with the other one. I/O bus will be a long and slow bus; slow because essentially it is concerned with the I/O part.

Generally it is lower compared with the other one, which is the processor memory bus, and it is also a long bus because you may have to have many connectors for the expansion of these devices and so on. So we have a range of speeds to be taken care of in the I/O; it is more or less standardized in the processor memory. The user is not directly concerned with this whereas the user is very much concerned with this. And the third one, the back plane bus, is on the PC board itself. You have the set of signal lines connected that is why it has derived the name back plane. As I said in some systems the back plane bus itself may be the processor memory bus.

So through a few system configurations we will just see the essential difference between these back plane buses. But we take it as the back plane bus is one in which you have the entire set of signal lines on the PCB itself; so that is how it got the name back plane. Now regarding the requirements, it so happens that there are two requirements for a bus and they seem to be also conflicting. We generally talk about bus latency. What is the bus latency? Whenever there is a requirement of the bus for a data transfer, you would like to see that the bus is made available as fast as possible for the specific requirement. So we would have to see that the bus latency time must be minimized. The time associated with the bus latency must be minimized; that is, whenever there is a request for the bus, the bus must be made available with the least delay possible.

(Refer Slide Time: 24:53)

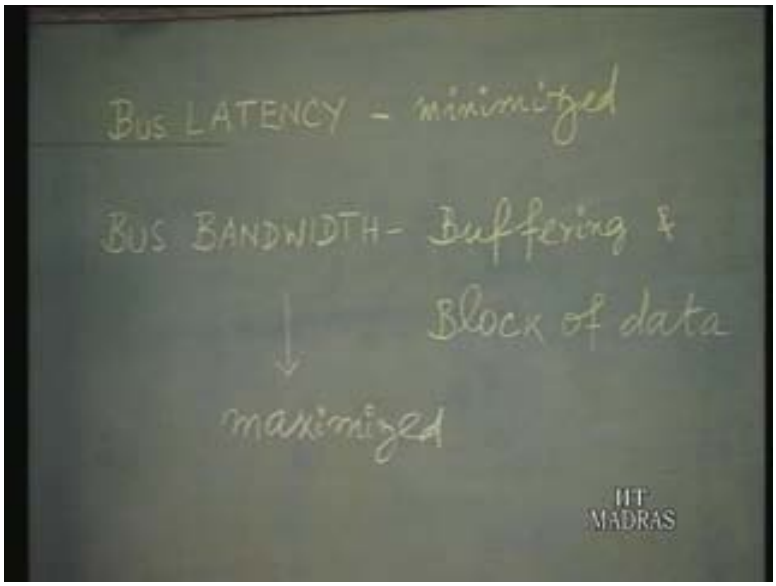


Then the other factor is the bus bandwidth. This particular one conveys to us that if the bandwidth is high, more data can be transferred; that is, there is more efficient utilization. Now more data can be transferred if we can bunch all the data, buffer it and then send it with the least amount of interaction asking for address, control, things like that. As we have seen for instance in the DMA, the data is ready and available and then, like a machine gun, it keeps going. Every time we do not have to keep checking. So the bus bandwidth can be increased by what we say as buffering the data and transmitting block of data so that the time that is generally lost between two blocks or pieces of data can be further minimized.

(Refer Slide Time: 26:16)

So there is buffering or storing more data before the actual transmission starts. So what we exactly gain here is that before the transmission, there may be some overheads and delay, but then there should not be any delay once the transmission starts, and once it starts, it goes very fast. While buffering a block and transmitting blocks of data, you maximize the bandwidth. Now what happens? When there is a block data transfer, the bus is not going to be available for some other device which requires it. That is because when the bus is being used by some other device, the one which requires the bus is going to wait; that is the reason. Now as we said the latency must be minimized; that is, the wait period must be minimized. We also say that the bus bandwidth must be maximized.

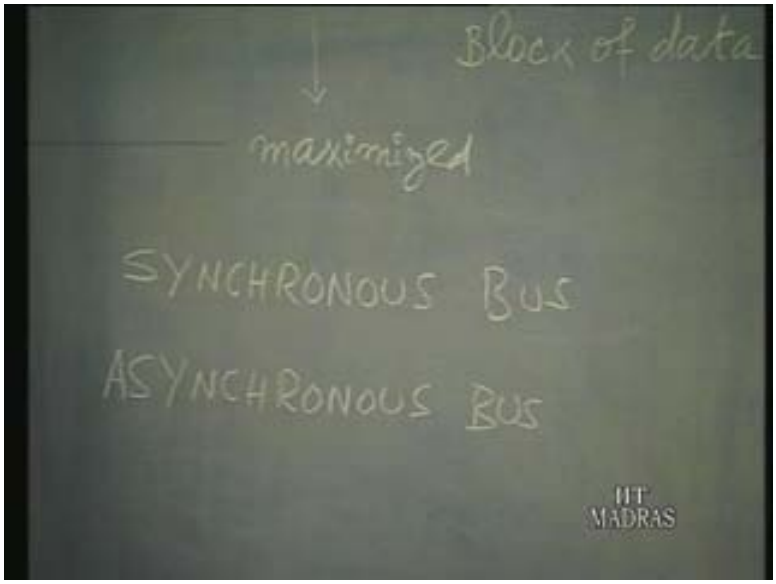
(Refer Slide Time: 27:41)



Now while trying to maximize this, we ended up buffering the data and then transmitting the blocks of data; and while trying to maximize this, we see that the latency gets affected. That is, the device which requests the bus has to wait because some other large transfer is going on. So these two – bus latency and bus bandwidth – are actually conflicting requirements, so there must be some compromise between these; now this is very important. Talking about different types of buses from the timing point of view we talk about synchronous buses in which the transmission takes place.

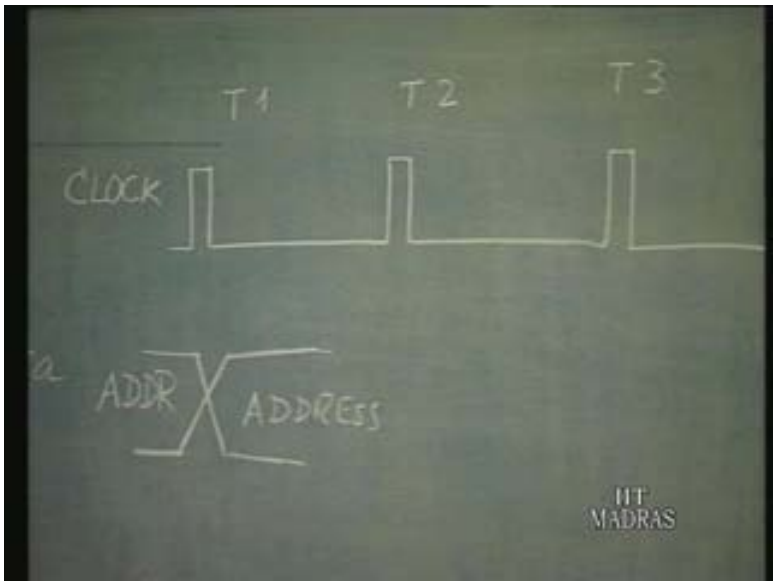
It is synchronized with some clock and of course the asynchronous bus. We have both types of buses; generally you will find that when synchronous bus is used everything must be known a priori. For instance, in the case of processor–memory interaction, the speed of the processor and how exactly memory is organized is known, whereas when it comes to I/O device we were not sure.

(Refer Slide Time: 28:51)



We may add slow device and fast device later on also; a priori we will not have everything. So it may be better to say that it depends on the individual characteristic of the device. The bus transmission must be flexible; for this asynchronous is better. If everything is known a priori, then we can make those elements in synchronous, or rather work in synchronous. That is, for instance, we have talked about synchronous action earlier. Remember in the initial period we are talking about the states and in each state we were saying some minimum action was going on. The minimum action is going on and then the state itself is being defined by the clock of the system.

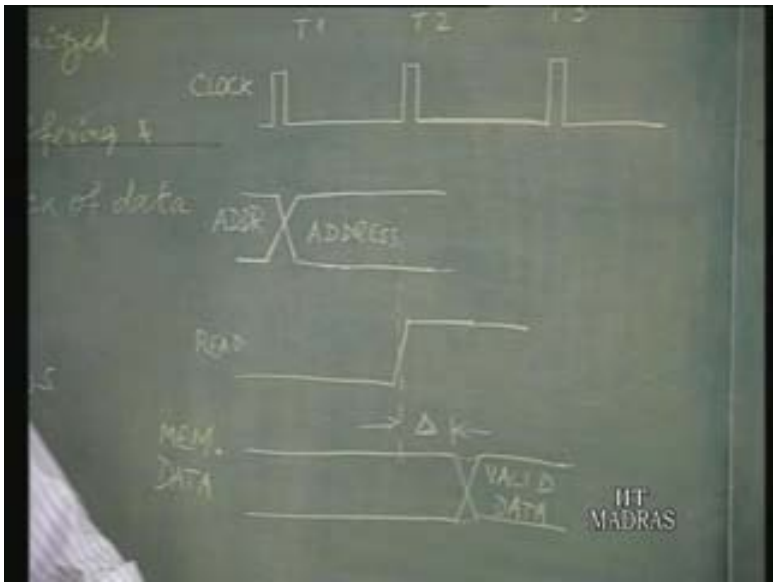
(Refer Slide Time: 31:03)



That is in connection with CPU, we are talking about it. Remember then we were saying in state T1 the address is placed; let us say the address line may be either 1 or 0. So either it may be this way or it may this way; actually this particular one refers to rise time and fall time. So in T1 the addresses is placed on the bus and, let us say, in T2 the read control signal is generated. The read control signal is generated in T2. The address has been placed; the read control signal is generated; let us say that particular going signal is 0 to 1. That is, T1 address is placed; T2 wait control signal is generated; on seeing read, the memory responds with the data from the location indicated by the address. So from now from the memory side this is all this from the CPU side. Now to indicate that the memory is different from these we will call these as the data coming from the memory or memory dot data.

The data that is coming some time after the memory sees the control signal read. So let us say there is some delay. I am just indicating the delay by this delta. There is some delay from the time the control signal is generated to the time the data is generated. This data actually refers to the data being 0. We do not know; may be some bit is 0, some bit is 1, so we would represent both. For instance if the memory responds to the 8-bit data, some bits will be 1; some bits will be 0. Some bits may continue to be 1; some bits may continue to be 0. So when we mark this way, it basically means it can be 1 or 0; this delta is the delay. This delay is due to the memory responding to the control signal read. There can also be delay introduced by the memory – that is when we talk about reading the data; delay with reference to the address also is possible. It is not shown here; here only the particular delay is shown. Now here you can see that assuming this delta, the delay, is less than one clock period, then we say that before T3 comes, this data can be read. This indicates that the read control signal in this says data is available. That is, we may refer to this as valid data.

(Refer Slide Time: 33:59).

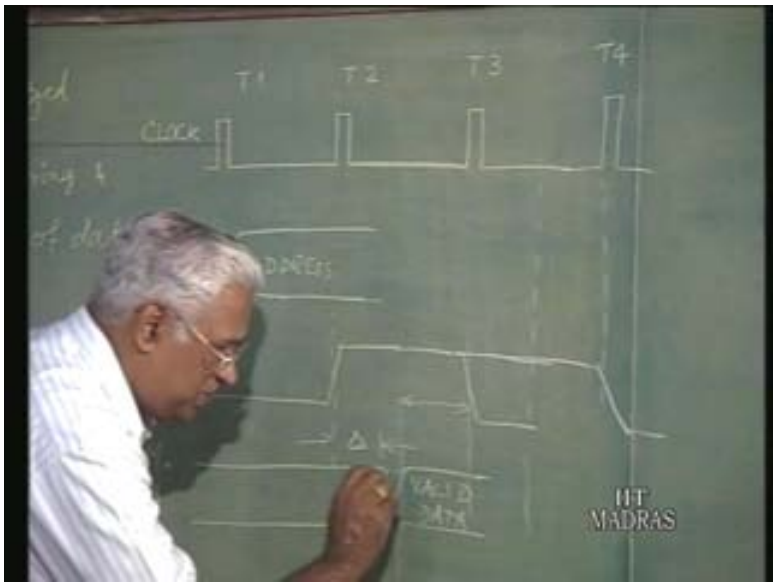


We say valid data because before this instant, the data was not valid; during this instant, there is some transition. Now the valid data is available. The CPU can actually read any time after this and even before T3. In this duration, the CPU can strobe it in, but if you want to be very careful you can see that at T3 this information is strobed, meaning, let us say something like this. For the read control this edge is used; this edge is used for reading.

If that is so, we say that reading of the data is synchronized with the clock. Here this is a clear picture of synchronous transmission, synchronizing with the T1 clock, the address is generated; synchronizing with T2 clock, read control is generated; and in response to the read control, the memory places the data on the bus and synchronizing with T3, the data is read by the CPU. So this is the synchronized transmission, which means we know for sure that this delta is not going to be more than this period. That is, well before T3, the valid data is available. In case this is not available, we had talked about the situation earlier.

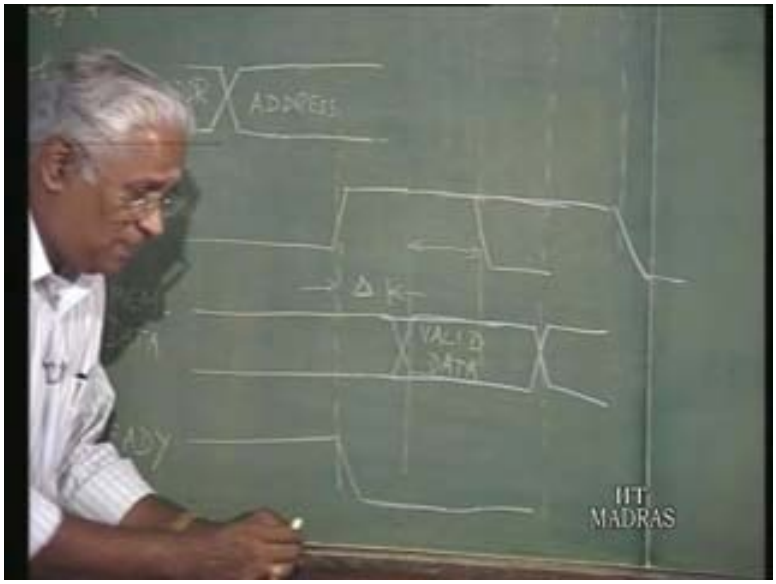
In case before the next clock pulse the valid data is not available, that means memory is not responding to this control and address signals. It needs more time. We assume this particular period is 100 nanoseconds and the memory is delaying let us say by 150 nanoseconds. That is, only 50 nanoseconds later, the valid data will be available, which means well before the next pulse, that is, T4, the data will be available.

(Refer Slide Time: 36:50).



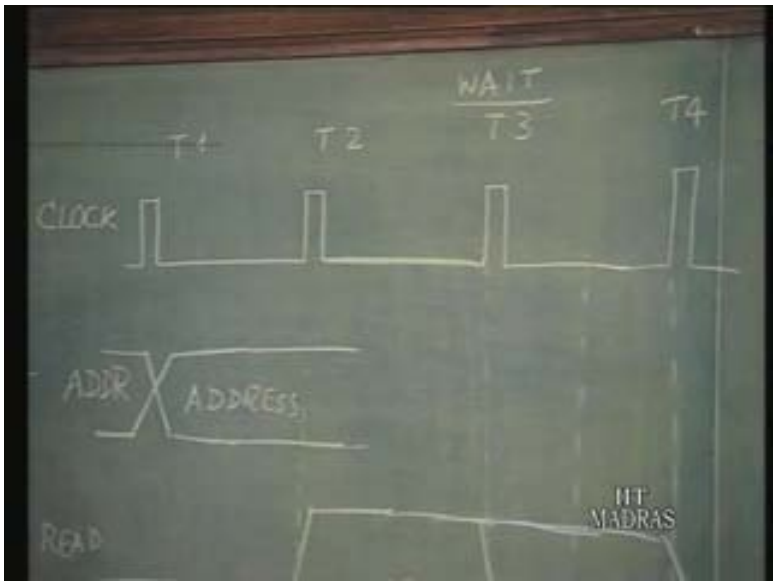
So reading cannot be performed here; so what will be done? What can be done is the read control signal must be further extended beyond and taken up to T4 because the data is not going to be available here. It is going to be available somewhere about 50 nanoseconds later. So the actual valid data will be available here itself; that means 1 clock pulse later, that data can be read. How is this achieved? The extension of this read pulse and delaying the reading is achieved as we had seen earlier. We said that the CPU can have a ready input, which can be used by the memory subsystem, and as soon as the memory system sees the read pulse, it can immediately say that the CPU, rather memory, is not ready.

(Refer Slide Time: 38:29)



Now on seeing this ready input to the CPU, on seeing that memory is not ready, then until it becomes ready for every clock pulse, the signals generated by the CPU will get extended. At this point, when the valid data is ready, that is, somewhere between T3 and T4, when the memory is ready with the data, the memory can pull this up again. So when T4 comes, it sees that the CPU is ready and reading can be performed at that point. So this how it is done. We had seen this earlier. That is, T3 is an extra state that is included as a wait state; that is, the CPU was made to wait during T3, and that was because of the ready input to the CPU.

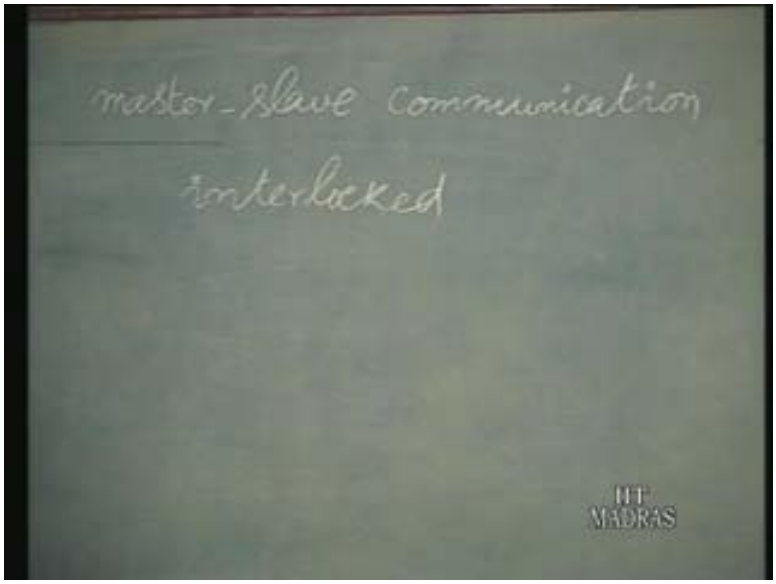
(Refer Slide Time: 39:00)



The ready input is generated by the memory. How and why is it generated? It is known very well that the CPU's fast memory is slow; that means a priori it is known. So, on seeing ready input, which is generated by the CPU as output ready input to the memory, the memory responds immediately, saying that it is not going to be ready. And how much delay is again depending on how many wait states must be introduced. Since it is known that more than one state is not necessary, this will just pan for about one state. This is the very clear case of synchronous transmission.

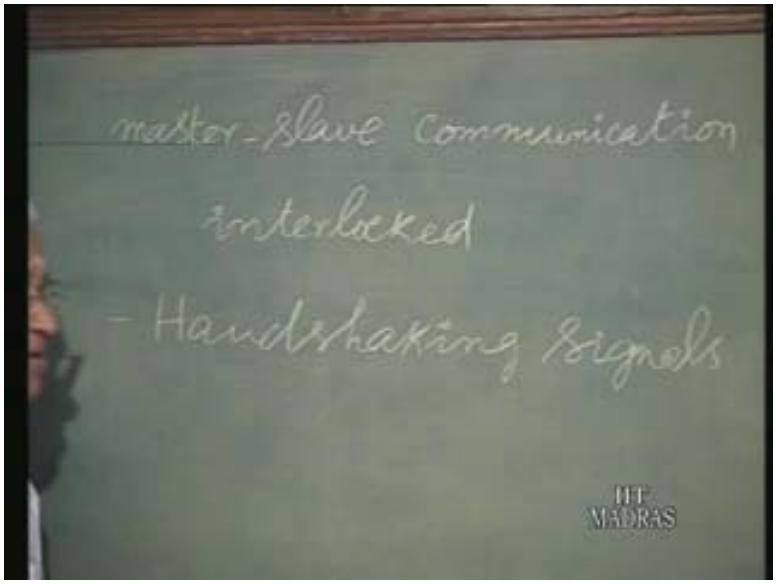
The CPU memory makes use of a set of signal lines and the transmission is going on or communication is going on between CPU and memory in a synchronized manner; it synchronizes with every clock edge. In the case of I/O, we just cannot guarantee that. Some buses or devices may be fast, some devices will be slow. And it may so happen that half way through the life cycle of the system, you may bring in some new device. We may bring in a new device, which may be fast or slow. In those situations, specifically with reference to the I/O bus, it is meaningful to have an asynchronous bus; meaning there will be a signal which says starts the I/O operation and when the I/O has finished with it, it can tell that it has finished this job. If it is a fast device, it is going to tell very fast, and the CPU will note it. If it is a slow device, the device is going to take its own time and then inform. So the master of the bus can initiate a data transfer and I/O will take its own time and then it will communicate saying when it has finished the job, that is, the transfer. In other words we can introduce what we may call us some communication between master and slave in an interlocked manner; what is this communication?

(Refer Slide Time: 42:16)



The master says perform the data transfer and then slave responds to it. So in an interlocked manner you establish the protocol of the communication. That is, the master says the data is ready, now you can take it; the slave says I am taking and this it says taking it own time. So we call this master–slave interlocked communication. It synchronizes with nothing; it will not go by the clock. It need not go by the clock; the clock can very much be there. Certainly it is not going to say that in this time slot something must be done; that restricting is not there. We also say that a set of signals that are used in the interlocked communication would be something like the master and slave shaking hands. So we refer to these signals involved in this as handshaking signals. You may be able to appreciate why we say this.

(Refer Slide Time: 43:34)



When we meet a person, let us say we say hello. And then, he also says hello. Then you shake his hands and say how do you do. And he also says how do you do. It is somewhat like that: the master says hello, are you there? The slave says, yes I am here. Then the master says here is the data; the slave says I have taken the data. That means a set of signals involved in this process are referred to as handshaking signals; they see to it that the communication goes in an orderly manner, and for a person who is not used to speaking very fast, takes his own time and then responds with the hello or how do you do, it may be fast; some may be slow. The same situation exists here too; in other words what we need is a few extra signals, somewhat like this.

The master may place the address – let us just take a read cycle itself – the master may place the address and then it may generate another signal, which says that the address is placed and that signal will be sensed by the slave and it will respond saying I was sensed there, and it will take the address. Then the master will generate a read signal; and then the slave knows that from the address, the slave must read and place the data. After it places the data, it says now the data is ready. The master will respond saying it will take the valid data that is available on the bus; some extra signals are introduced. So in this way, the address is placed; I will avoid this bipolar signal; I will just using only one, just to show you the sequence. Let us say it is something like this. The address is placed; I am just assuming only one this thing at some time edge. Since we have assumed read cycle, let us say that after the address is placed, the read signal is also introduced.

(Refer Slide Time: 50:50)

Now there are different ways in which we can use a handshake signal; I am just assuming one specific sequence. So the address is placed and read is indicated. From the master point of view, it can indicate that it wants the slave to respond by reading the contents of the location, the address of which is given. So after it has performed its job, the master indicates through one hand shake signal; we will call it MSYNC. This in fact is an indication that it wants reading to be performed by the slave and it also gives indication of the address. On seeing the MSYNC signal, the slave understands all this, and in response to this, the slave can respond with the data, that is, the memory. We assume this is memory response data. Whatever may be the delay that delay is because of the memory?

After that delay, it generates the data and this is now available on the bus – valid data. Let us create some space for the other thing. After the data is placed, in this case the memory can generate a similar slave SYNC signal and indicate that after the instant, it will indicate that from the point of view of slave, it has done its job. The master is indicated by placing address and read: it is an indication to the slave that the slave must perform read. On seeing this master SYNC, the slave has responded by generating the data and placing it on the bus. After it has done its job, the slave is indicating. Now this SSYNC is the signal given or generated by the slave. On seeing this SSYNC signal, the master knows that whatever it wants is available on the bus because after this instant, that is, on seeing the SSYNC signal, the master knows that the required information is available. Now the master, on seeing this, will read; that means, this read signal will continue certainly beyond this for some time; after that it will terminate. That means by this time the master has read the data, then after it has read the data the master may terminate its signal, that is, after this instant when the data has been read, the master will terminate its signal and on seeing this, the slave may respond with a few things. Suddenly on seeing this MSYNC going negative, the SSYNC also will be pulled down by the slave. This is the indication that the slave knows that the master has performed its job of reading, now it is closing the whole show.

So we say that there are two hand shake signals, one MSYNC asserted by the master is an indication to the slave that it wants something to be done and on doing that particular work, the slave asserts the signal and on seeing the assertion of SSYNC, MSYNC, the master, concludes its job of reading and then negates the MSYNC and on seeing the negation of MSYNC, the slave also negates it. So we see that the signals are asserted, that means the signals are placed and the signals are negated; that is the signals are removed and you can see the specific sequence.

(Refer Slide Time: 52:07)

Now there is absolutely no clock that need be used here. On seeing the signal, the other signal is generated; on seeing the negation of this signal, the other signal is generated; and in between, the required activity is done. This is the way the ASYNC bus will work and you need a set of extra signals for this. These extra signals are something like a clock because they really do the timing, but it is not strict clock periods like this. So generally these are timing signals; that is about the synchronization.
We will see more about these processes in the next lecture.