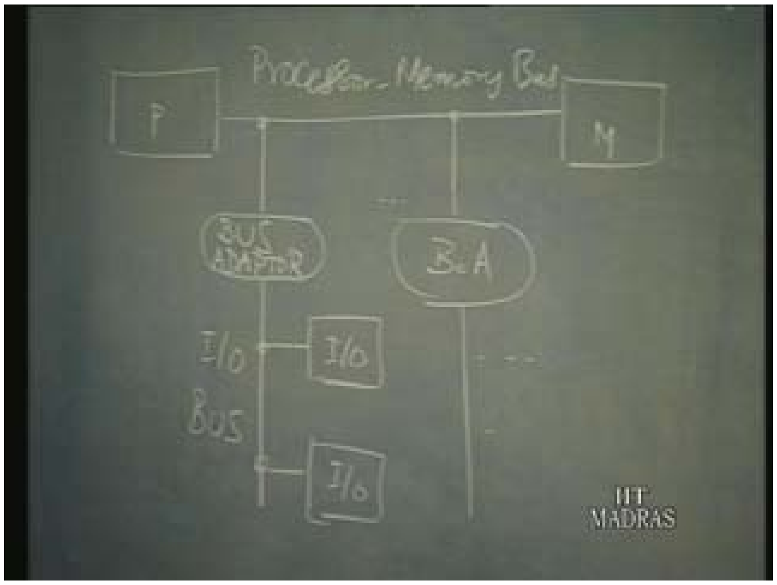
Computer Organization
Part – IV
Prof. S. Raman
Department of Computer Science & Engineering
Indian Institute of Technology, Madras
Buses
Lecture – 32
Buses (contd...)

In the previous lecture we just got glimpse of buses, various types of buses and some aspects of buses which we just saw in general. We will continue the discussion on bus in this lecture also. We were talking about the back plane bus and processor memory bus and I/O bus. Let us just take a look at that.

(Refer Slide Time: 05:25)

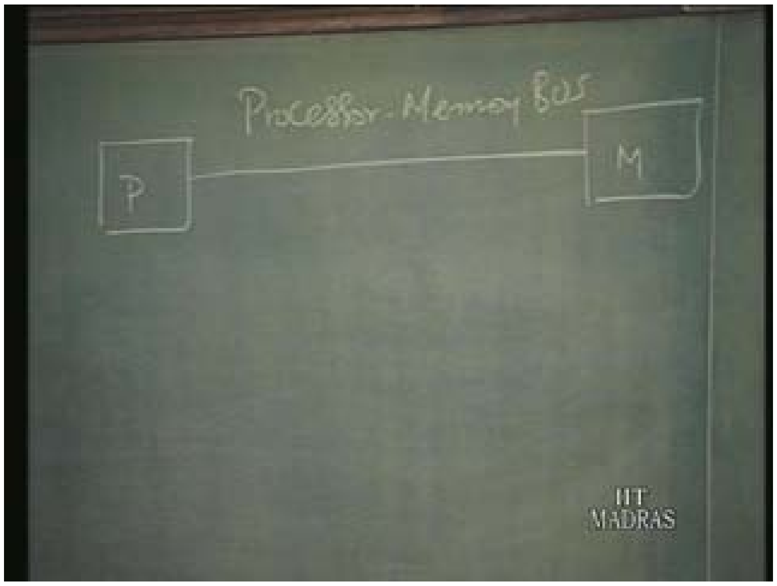


For instance essentially we have processor then we have memory and we have many I/O devices. Normally the bus connecting these, we will refer to as processor memory bus. But then in the case of small systems the whole thing may get sort of integrated which means on this bus itself we may provide the necessary slot so that the I/O may be connected. This is essentially to simplify the matters and you can have many I/O. Though processor memory is very much interconnected you see that I/O is also integrated with that. In which case this particular bus will be there on the back plane itself that is in the printed circuit form itself. On the back plane you have the necessary slots and you would find the processor slot, the memory slot that is a slot in which the processor board and the memory board will go and you will also find the I/O slots on that. Generally that is what you have. Specifically in small systems essentially what you have is a back plane.

As the complexity of the system grows there may be some necessity for separating this somewhat like this. We have the processor, the memory and the processor memory bus. We can see that this processor memory bus will certainly be much faster than back plane bus. The reason is in addition to the processor memory interaction; this back plane bus will also support the I/O. And this will invariably slow down.

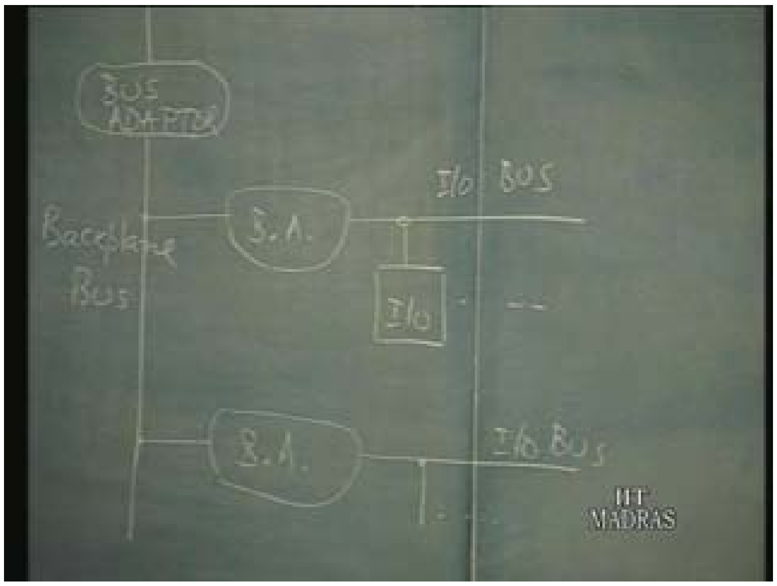
You will have processor memory bus then to this you include the necessary interface circuits called bus adapters. Using this bus adapter you have another I/O bus and I/O will be sitting on this bus. You can see that the function of I/O and the function of processor memory interaction have been separated. Like this particular bus adapter you can have many bus adapters and you can have many I/O buses also. But if you go on adding a lot then you are going to tap on to this processor memory bus and this could affect the speed. What you might have in such a situation?

(Refer Slide Time: 06:07)



With a slight modification, you have the processor, you have the memory and you have the processor memory bus as before. Then instead of having many tapings on this bus, just have one tapping and have the bus adapter on this and have a back plane bus. On this back plane bus you connect the other adapters say another bus adapter and create an I/O bus on which the different I/O will sit. So instead of tapping on the processor memory bus you are tapping on the back plane bus and adding additional bus adapters and creating many I/O buses. Processor memory bus will be short and fast, then I/O bus can be long and it need not be fast. In fact it will have a wide range but then for different type of devices you can have different types of I/O buses and then do not tape too much on the processor memory bus then this will also act like buffers.

(Refer Slide Time: 07:22)



Then have a back plane bus and have the necessary slots for this adapter and go on expanding. You can just see this is precisely what you have. Earlier we had all of them in one, this is how we started with our discussion. So one set of signal lines that is the bus processor memory I/O was connected but then the practical considerations make us go in for this different configuration. That is about the arrangement. We are talking about the synchronous and asynchronous bus. I will show some extra item on these buses. We would start with what is going on in an asynchronous bus.

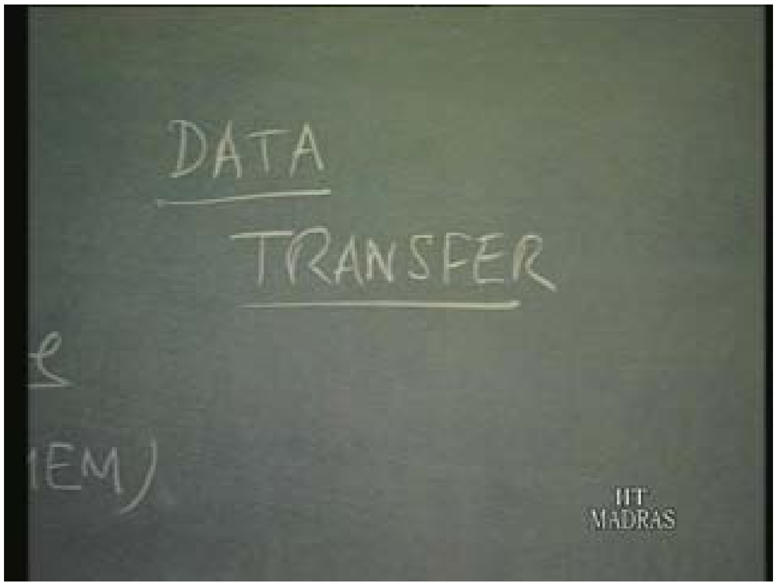
(Refer Slide Time: 09:46)

Asynchronous Bus
Master
(CPU)
Slave
(MEM)
IIT MADRAS

Specifically we were only talking about 2 handshake signals MSYN and SSYN. This is what we were doing earlier.

I would try to present the picture of the communication protocol between the master and the slave. Let us assume that the current master, there can be many masters. Even a device can be a master for instance when it is interrupting. We will just assume CPU is usually the master and memory is always the slave. We will just assume that between the processor and memory the communication is going on. Let us just work out for one thing that is the data transfer part. You remember earlier I was talking about data transfer and then in-parallel priority arbitration can go on.

(Refer Slide Time: 10:15)

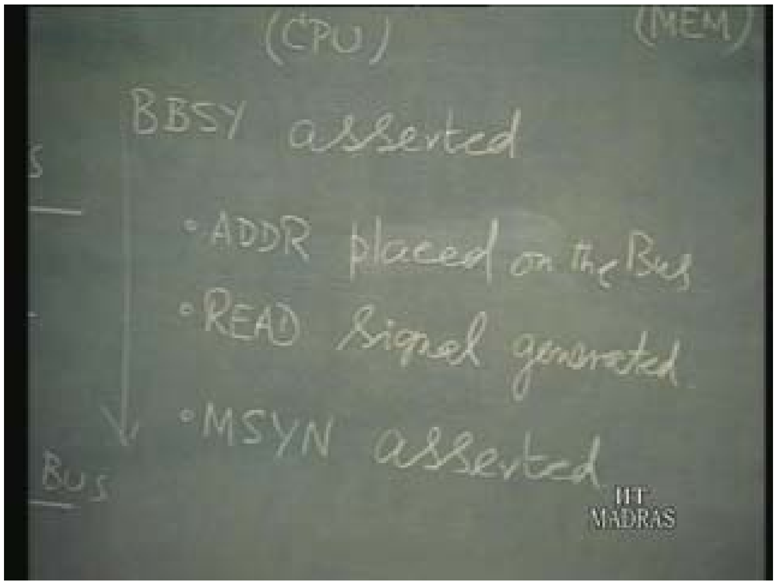


First let us see what is going on during the data transfer. In fact essentially I was introducing this while discussing the asynchronous bus. There can be many signals. I will start with one new signal and call it as bus busy. A master will remain as a master on a bus as long as it can assert that particular signal. When we say bus busy asserted that means for one of the masters there can be many asserts and usually CPU is the master. That is what we say the CPU remains as a master. To become a master in other words a device must assert that signal. And it will have to continue to assert that signal which means hold the signal. Logic level may be high you can also have negative logic and make it low it does not mater, that is why we say asserted. The asserted just means it generates the signal it does not mean its logical one or logical zero, it can be either.

A master remains as master of the bus by asserting the signal bus busy. Let us say this particular data transfer is concerning read cycle which is easy for us to discuss. What it does? The CPU asserts the bus busy and then since I said it as read, the master will place the address on the bus that is address will be placed on the bus and then read that also. Let us give it separately, they can go simultaneously. I will say it asserts the address on the bus. What you mean by asserting address on the bus? Basically it means the master places the address on the bus. That is what it means. So I think we will reserve this assertion and negation for the handshake signals which is more meaningful. We will use the same terminology what we were using earlier, that is address placed on the bus. Then of course we know read is a control signal so let us say generate which is placed on the bus. We were earlier using the timing diagrams. It is some what like this. We did it in the previous lecture assuming address, read and we say address is placed on the bus at time strength and then at some time say read is generated. Suppose time axis is the usual thing.

At t1 address is placed on the bus, at time instant t2 read signal is generated. (Refer Slide Time: 13:53-14:30)That is what I am just putting it in the form of text in the below slide. Now after it has done, in fact you can say that the time axis flows in the downward direction. That is the time sequence. First the master remains master by asserting the bus busy signal and assuming there is a read cycle, the CPU places the address on the bus and places the read signal on the bus. After this MSYN is the handshake signal. So let us use that terminology, asserted. Where is the need for this MSYN or handshake signal? We briefly saw it earlier.

(Refer Slide Time: 15:13)


(CPU) (MEM)

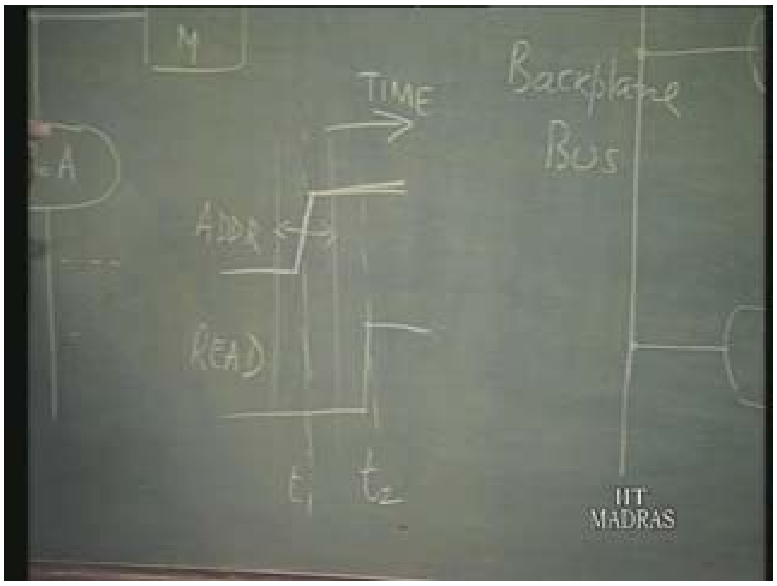
BBSY asserted

- ADDR placed on the Bus
- READ Signal generated
- MSYN asserted

BUS
IIT MADRAS

Let us see in detail. You have address as we said that is the CPU places the address and let us assume that it is a 16 bit address. Essentially it is a 16 bit number placed on 16 different signal lines. If you see the timing diagram say at instant t1 we say the address is placed. But who can give the guarantee that address is placed exactly at time instant t1? Because there are 16 different signals on 16 different lines. Some may have been placed a little earlier and some may have been placed a little later. Actually something like a time bandwidth. Over this time the signal may have been placed at the CPU end. Then you have the signals traveling down the line.

(Refer Slide Time: 16:26)



You have the transmission line effects. When it reaches the memory, we generally think it reaches immediately. The transmission line problem comes. You can just see when we draw a timing diagram we only talk about the logical aspects. When you start considering the physical realities, you find that there are lots of nasty thing that we have to be taken care of. There is going to be some delay and that delay is going to be different for each of those 16 address bits. Assuming that for the given set of signal lines this maximum delay is going to be 40nano seconds. By that time the address that is placed, all the 16 bits reach the memory end. At that time that is I said 40nano seconds, if this MSYN signal generated 50nano seconds later, then it is assured that by the time memory sees this MSYN the complete address is available at the memory end.

In fact this signal is an addition signal which ensures that the valid address which means all the 16 of that address are available at the memory end. Remember MSYN is also another signal generated by the master. It is very much like address signal or read signal. There is going to be some delay of this also. But that is fine. So on seeing the MSYN assertion we can just say there is going to be some delay. On seeing MSYN assertion now the slave responds. All the things which we have been talking about the earlier are for the master.

We got to write for the slave, the slave that is the memory. On seeing MSYN assertion the slave responds, it basically sees some signals are there on the bus. Obviously it will see the read signal, MSYN and then it has the address on the bus. On seeing read signal the slave knows that the read cycle must be performed. In other words we can say that MSYN assertion is a signal to the slave, that master wants some specific thing to be performed.

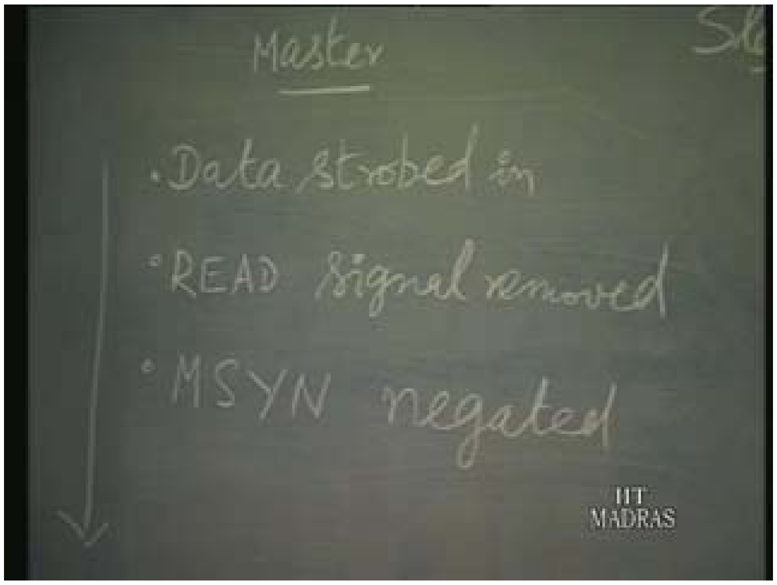
(Refer Slide Time: 20:35)

On seeing MSYN the slave responds and it will do whatever the master wants it to do. Master wants reading to be performed from the address location. We can say that memory responds with that data. You can say that on seeing MSYN assertion data is placed on the bus. Because it is a read cycle which we had assumed, the CPU places the address and the memory responds with that data. If it were a write cycle the CPU will not only place the address it will also place the data but then it will generate the write signal. On seeing MSYN asserted, the slave does the job that is reading from the memory and placing that particular data that is read on the bus. After it has done it SSYN is asserted. That is the slave says I have done my job.

Let us go back, MSYN assertion is an indication that master wants something done. SSYN assertion is an indication that the slave says that it has done its job. These signals must be negated or removed. Let us see the sequence in which this particular thing proceeds. We saw that the MSYN was asserted by the master and subsequently SSYN asserted by the slave. And MSYN assertion is for indicating that the master wants something done and SSYN assertion is an indication that the slave says that it has done what the master wanted. Specifically with reference to read cycle we saw that, that is the address was placed, read control signal was generated and in response to this by the master, the slave was placing the data on the bus and then only it asserts the SSYN.

What is the master doing and then what is the slave doing subsequently? Because we had introduced 2 signals MSYN and SSYN. They now remain asserted, they have to be removed. On seeing SSYN assertion by the slave, the master knows that the valid data is available on the bus. That is because of the read cycle we are talking about. We can say that data strobed in, that is the valid data available on the bus is strobed in. What will you do after that? The first thing is after it has read in the data, the read signal is removed.

(Refer Slide Time: 24:22)

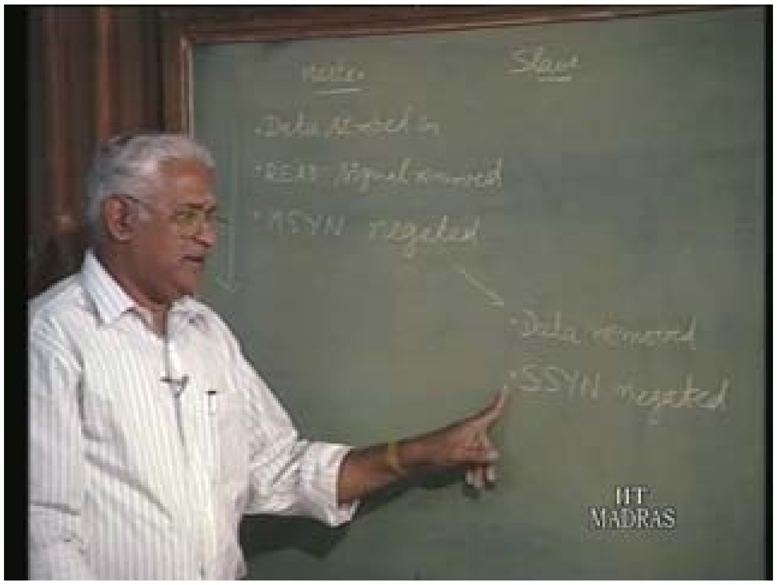


Master

- Data strobed in
- READ Signal removed
- MSYN negated

Earlier the read signal which was placed is removed. What about the address? The address can very well be there, it does not matter but read signal must be removed certainly. After that MSYN that is a hand shake signal will be negated. The time flow is in the downward direction. On seeing SSYN assertion that is by the slave, master responds by strobbing in the data because the data is valid then it removes the read signal and then it negates MSYN. MSYN negation is a signal to the slave. That master indicates that whatever it wanted is over. It has concluded.

(Refer Slide Time: 25:42)



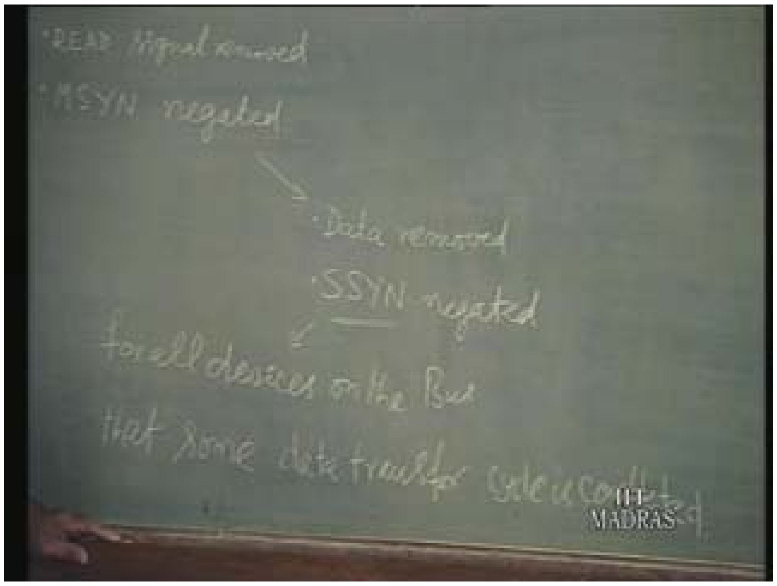
Master | Slave

- Data strobed in
- READ Signal removed
- MSYN negated
- Data removed
- SSYN negated

What did slave do earlier? Slave earlier was placing the data on the bus. Let us say slave responds by removing the data from the bus. We can say data removed and not to worry about how it does.

Usually it is done by disabling the output buffers in the memory. Data removed from the bus and it negates SSYN. That is on seeing MSYN negation, the slave responds by removing the data and then negating the SSYN. MSYN which was earlier asserted is negated and subsequently SSYN which was earlier asserted has now been negated. This SSYN negation in fact is the signal for all devices on the bus. We do not know what exactly it is. Some data transfer cycle is complete or concluded. We had assumed the read cycle so that is what had completed or that particular thing is over.

(Refer Slide Time: 26:59)



•READ signal removed

•MSYN negated

•Data removed

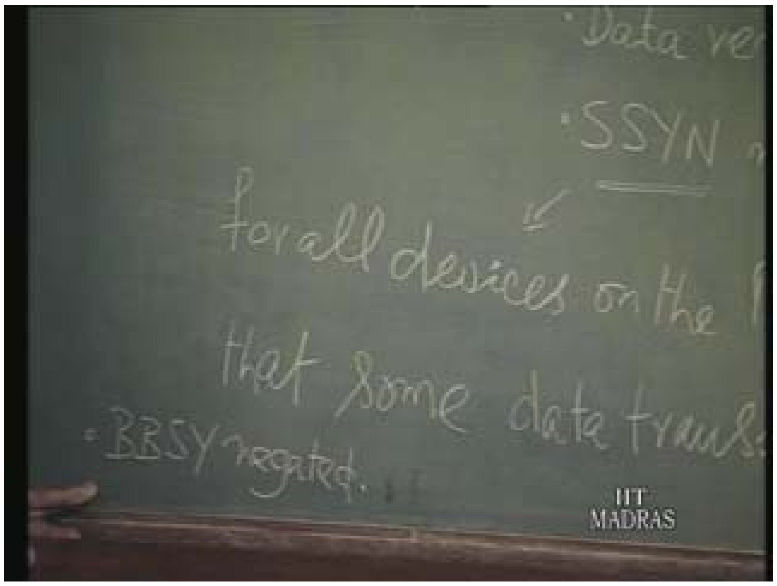
•SSYN negated

for all devices on the Bus that some data transfer cycle is concluded

In case the master that is the CPU wants to further go head with the data transfer, then it has to continue to assert the bus. Because only when it asserts the bus, the master will remain as the master. Bus busy asserted something, MSYN asserted then something done by the slave, SSYN asserted and then some response by the master for the SSYN, then it negates SSYN, then the slave responds and it negates SSYN. SSYN is an indication for all devices on the bus that is some data transfer cycle is completed and some other thing can start. We can see how MSYN and SSYN have been used.

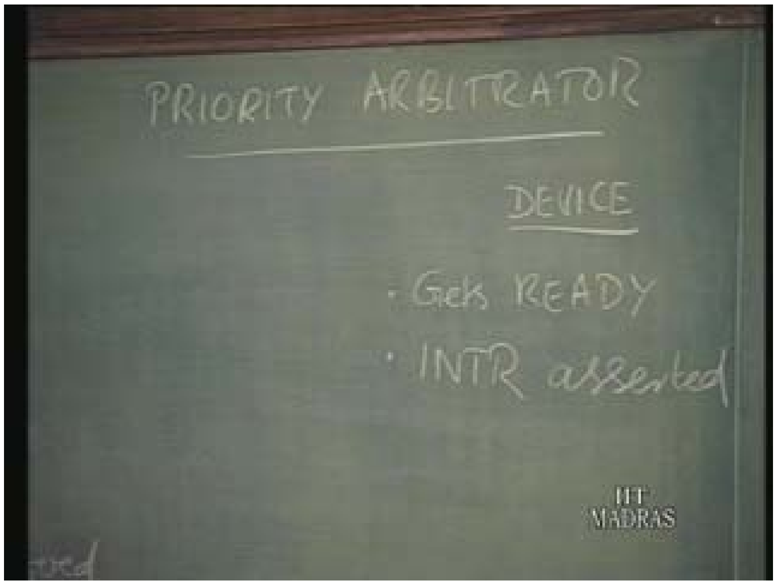
Assuming the master continuous as the master and bus busy will remain asserted. Bus busy is another signal, bus busy will remain asserted. On the other hand at the end of this data transfer, if there is some other device waiting then that particular device will assert bus busy. And so if that is the situation we will see because something else must be said but assuming that some other device is waiting to become the next bus master. Then in that particular case the master will negate bus busy. Bus busy will be negated because that is how the master remains as a master. It was asserting the bus busy, now by negating bus busy the master relinquishes the bus in favor of some other device waiting on the bus.

(Refer Slide Time: 29:02)


·Data ve
·SSYN
for all devices on the
that some data trans
·BBSY negated.
IIT MADRAS

What really decides which of these devices can become the next bus master? Can the master decides that it want to remain the master all the time? Usually that is the situation, it can be, and there is no harm that is it can inhibit everything. But then there will be some problem. Whatever we had talked about holds good for the data transfer. This is how we started, for data transfer this is the protocol. This is in fact the communication protocol. While this is going on in parallel to this? For instance priority arbitrator can work and decide.

(Refer Slide Time: 31:59)

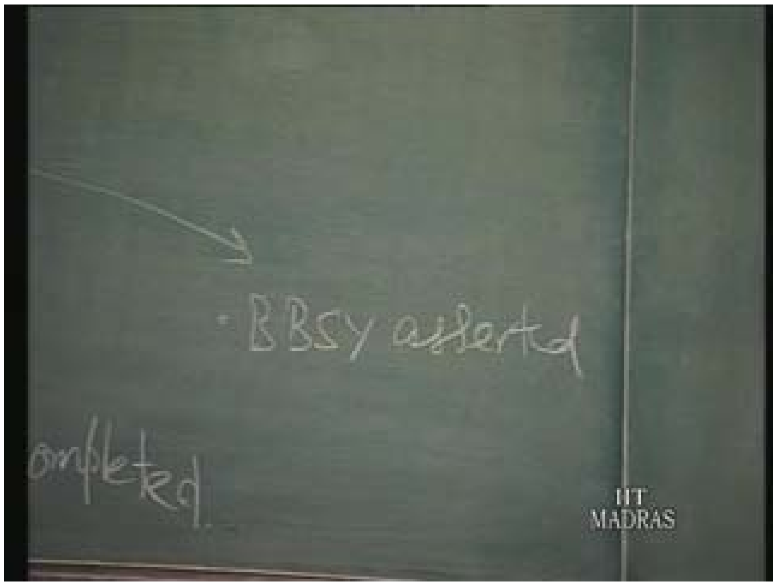

PRIORITY ARBITRATOR
DEVICE
· Gets READY
· INTR asserted
IIT MADRAS

In case some device is interrupting or in case some DMA controller comes up and indicates that it wants a DMA transfer. Then it can decide which must be the next bus master. But this must go on in parallel.

Why? Because the data transfer must go on and you have a separate set of signals. For instance some device, I will just say a device which is not the slave gets ready. Let us assume that this is going to be in interrupt driven mode in which case when the device gets ready it can generate the interrupt signal and let us say interrupt is asserted. An interrupt is an input to the processor. But the processor is busy doing some data transfer. You remember we have always said in the case of interrupt driven mode the processor is going to respond only at the end of the current instruction execution cycle. At the end of the current instruction cycle only it will do it. So obviously the processor can not respond because it is busy. In fact this is priority arbitrator which will accept the signal. This is one method of implementation there can be other methods. This accept the signal and its possible that more than one device interrupts in which case it will look in to the priority of that and then among these devices which is interrupting the processor, it means among the devices which are ready and are ready to interrupt the processor, it will select the one with the highest priority and it will keep it. It will select that particular device and keep it ready for the processor to complete its current transfer as indicated by the SSYN. Then when the SSYN negation comes that device which has been kept ready as the next bus master will assert the bus busy.

So priority arbitrator will have to indicate at this point that there is a next bus master which in fact is an indication that the current master must negate the bus busy and lot of things must be said. The master involved in the data transfer that is the CPU is relinquishing in the bus busy and lot of things must be talked about here. At the end of that on seeing SSYN negation by the current slave, the device which has been selected by the priority arbitrator to be the next bus master will assert the bus busy provided the previous bus master has negated that.

(Refer Slide Time: 34:40)



Here lot of things must be said, the interrupt request comes, and the priority of the interrupt must be checked and so on. There are many ways in which it can be done because the device itself is going to generate the interrupt. I was saying that the hardware priority, then there is a software priority and all those things will have to be sorted out. All these goes on when the data transfer is going on in parallel. Which means basically you have a set of signal lines for the arbitration and another set of signal lines for the data transfer, otherwise it is meaningless.

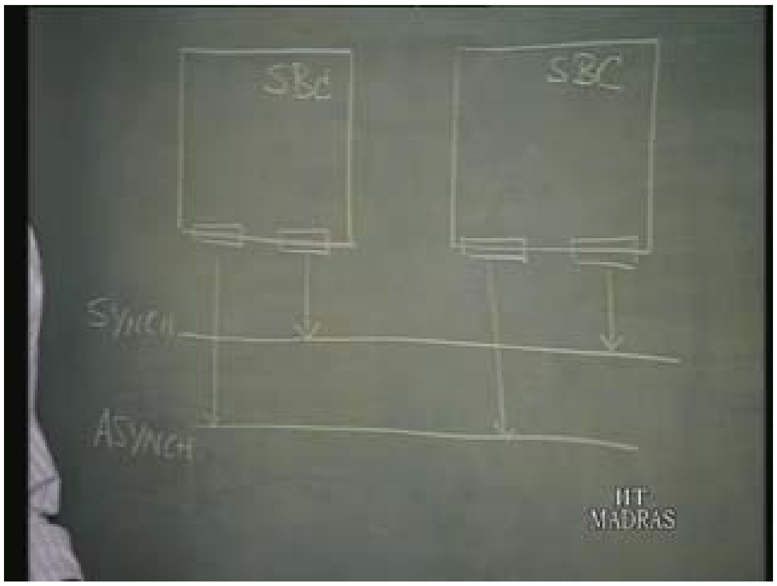
In other words the priority arbitrator is hardware by itself. It will receive the signals then it will process among this and then do something in parallel for which you have a set of signal lines. You can just see that in this particular scheme of asynchronous communication everything is going as per some specific sequence methodology. And if this (slave) device happens to be slow, as we said the memory instead the slave can also be the device. In the other way a device can also be a slave. At that instance what is happening? In case the device is slow then it is going to generate the SSYN leisurely. If it is fast, it is going to respond fast.

So look for MSYN, it generates SSYN, negate MSYN and then do something and negate SSYN. This particular scheme allows asynchronous communication in an interlocked manner. The two things get interlocked and as this goes on other activities also can go on over another set of signal lines. All these signal lines together form the bus. That just gives some idea about asynchronous transfer. Let us take a quick look at the synchronous transfer. We have talked about asynchronous bus; let us also take a look at synchronous bus. Recall while talking about the processor memory interaction, I said essentially it is synchronous bus. But then we also know that it is a proprietary bus in the sense everything that is about the processor speed, memory, response and everything is known as a prior. It is known well ahead and so we can have a synchronous thing.

Wherever full synchronization I mean wherever continuous transfer is not possible, wherever you have to skip a few clock cycles you introduce wait states. But how to do it in case the bus has to support the I/O transfer also. Because we saw that in the case of I/O devices we have arranged, going from the slowest to fastest device. We do not know a priory, what will be the speed and may be after a few years of system usage, something new will come and so on. How this particular thing will be done? There is only one way in which you can cater for this wide spectrum of I/O data transfer rates and still hold on to a synchronous bus. What is that? You already know about it. What is that we were doing when we found that the CPU was not fast enough? We said I/O is fast, memory can respond fast but unfortunately it is the processor which becomes the bottleneck. What is it that we were doing? We were trying to through away the processor and then bring in a fresh DMA controller. Similar thing you would find here too. In other words wherever you want to support the conventional asynchronous thing provide for that also.

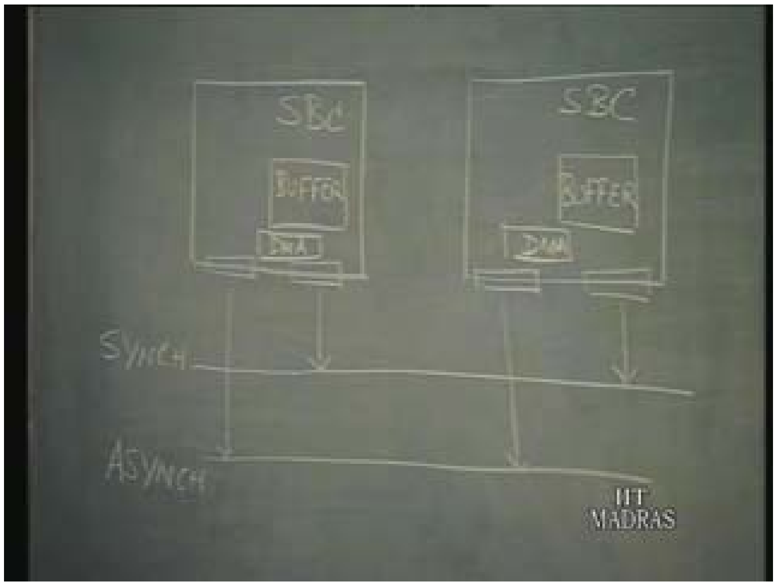
These are just different boards. Let us say 2 single board computers. You can always have 2 connectors to each of these. You can connect say one set like this and another set like this. You can have possibly asynchronous bus that is the conventional one and on this particular one have the synchronous one. Supposing these are single board computers.

(Refer Slide Time: 40:58)



They sit on these buses appropriately that is you have asynchronous bus that is also supported and then wherever there is a need you also support this asynchronous bus. In fact specifically Intel multi bus too follows this pattern. How do we synchronize? This is important because we do not know at what rate this is transmitting and at what rate this is receiving which is at the left side. Just put it a single board computer for instance you can have a separate say an I/O controller board instead of an SBC. An I/O controller board taking care of the I/O. Main problem is this is transmitting at one rate and this is transmitting at another rate. How do we do it? The best way of doing it is take the slowest device and then suppose this is the controllers for that device then introduce some buffering arrangement here.

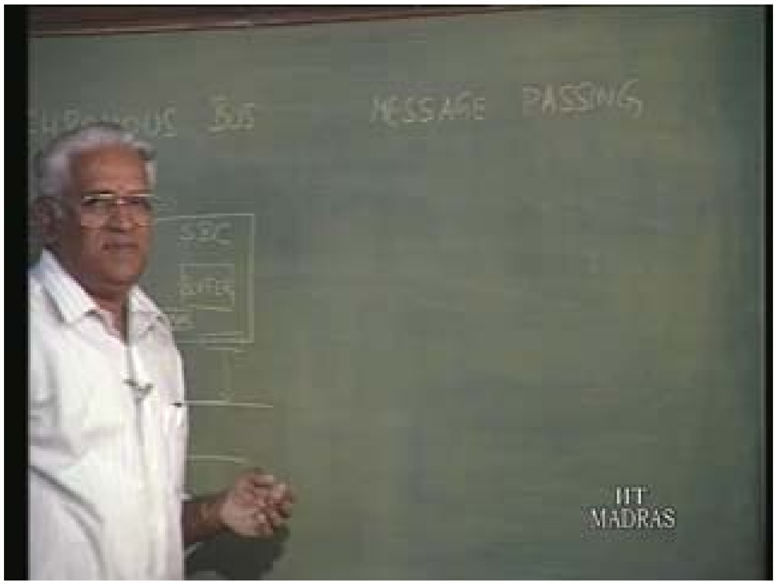
(Refer Slide Time: 42:51)



Similarly have buffering arrangement in the other one. This (right side) may be a slow one and the other may be a fast one. Then provide a DMA feature that is direct memory access feature. Let this particular controller or computer assemble all the data and make use of the DMA controller to pass it on at some standard rate. Because synchronous means it must be synchronized to some standard rate and not directly to the device or to the CPU for that mater. Just from this (right side) buffer pass on to this (left side) buffer make use of the DMA controller. In other words if you have something put them all together and pass it on. We are quite familiar with this where it is something like passing mail from one station to another station or one city to another city. The post officers collect the mails and put them in a post back and pass it on. The same way here too. These mails may be coming at different rate, different speed goes to different places that is a different thing. So collect them all in one place and 12:30 means go and clear whatever that is there pass it on, put it in a train which leaves at that time to move it on. Something like that and it is the same way here.

So as for as the transmission over the bus is concerned, you maintain the synchronization. At the computer end or the controller end they still continue at their own speeds. That is only in the bus. You buffer them and then over the bus you would transfer at a fixed rate, synchronize. Basically what is needed is information that the buffer has data or it does not have the data. The buffer is full or empty that information is needed. Once that is there just passing, in fact this is called message passing. That is you do not talk about signals at this level. Why? Because signal will be at very low level.

(Refer Slide Time: 45:41)



So you collect all the information and packet that information and then pass it on as a message. At this bus level there is no problem mainly it is the DMA and the buffering takes care of the varying speeds. Essentially this is the idea in the synchronization. Really we are not taking care of the individual problems or individual speed, variation associated with this various things. You define a set of standard communication protocols or just say steps, the sequence in which what should follow what and so on.

(Refer Slide Time: 46:49)

So with these what happens is whichever thing is ready can assert the bus and then become a master and then it can pass on and there are other things also we can go on doing it. For instance one of this part can be the master, in fact it is happening in case of multi bus too. One of the things is always in a specific slot and keeps looking at the various things that are there. They would keep sending commands to the other things and configure them in whatever way it wants. And then it will keep monitoring it and then in which case it becomes something like a centralized control. For instance it can send the command and then make this particular thing work, once as a slave processor and some other time as the master processor and so on. It is possible. That is precisely the trick behind the whole thing where we have not introduced any innovative thing in this.

The simple act of buffering and bunching and passing on that is it. In other words you remember what all we used to say earlier in the case of I/O interface. The I/O interface will take care of all the problems associated with the device. So that the buses end something uniform can be seen this is the same thing here. There is no difference in that. This bus signal means we have to really go in to the details of a specific bus because it varies widely. But over all this is the idea. That is the difference between asynchronous and synchronous bus.