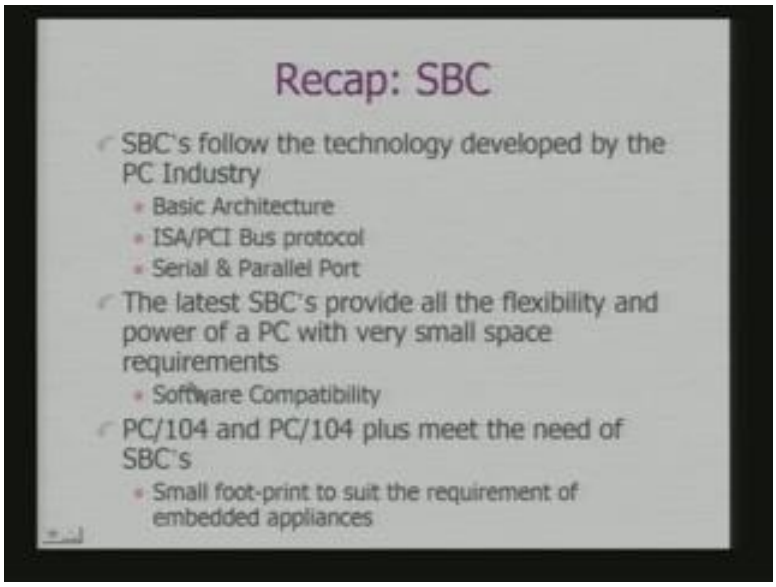
Embedded Systems
Dr. Santanu Chaudhury
Department of Electrical Engineering
Indian Institute of Technology, Delhi

Lecture - 15
Bus Structure – 2 SOC bus Serial Bus

In the last class, we have studied bus structure and bus is the most important component which connects CPU with memory and peripherals. Today, we shall study more specific bus structures both parallel as well as serial buses to recapitulate what we did in the last class.

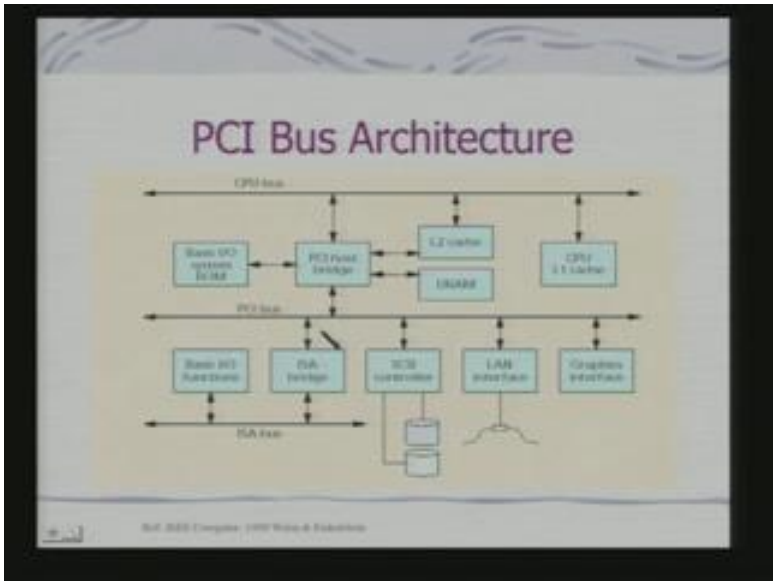
(Refer Slide Time: 01:24)



We studied single boat computers as examples of Embedded Systems. Single board computers are nothing but a PC put on to a single PCB. So, the single PCB satisfies all the basic characteristics of a PC architecture. The bus which runs on their board is typically ISA bus or PCI bus that we had studied earlier and you have serial and parallel ports. So, in fact, SBC's provide all the flexibility and power of a PC with very small space requirements. So, basically what has been done is that foot print of that card which is being used has been reduced. So, that it can be used in an appliance. The basic motivation of doing this is that your software can be very easily ported. And you do not spend time as well as money in using tools targeted for specific microcontrollers like

peak enough. In fact, PC 104 and PC 104 plus meet the need of SBC's. In fact, these are nothing but these specification of ISA and PCI bus suited for small foot print devices.

(Refer Slide Time: 03:04)

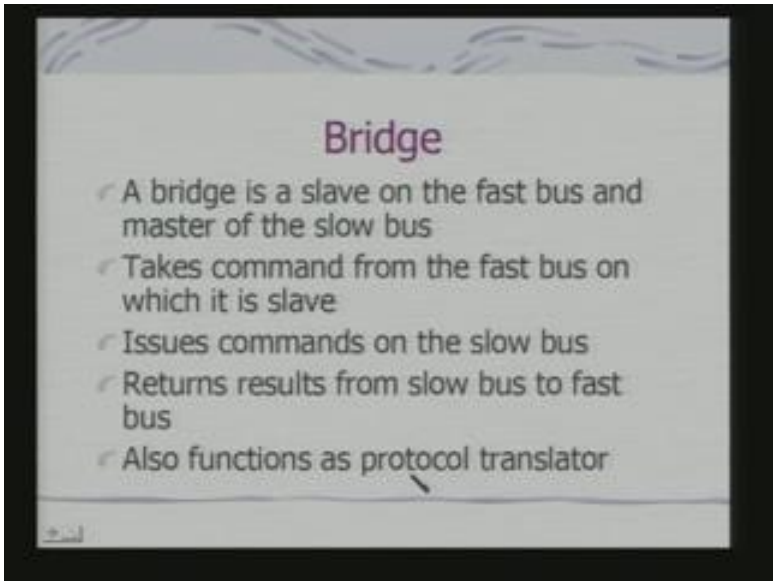


In fact, the basic bus architecture is that of PCI bus. In fact, we looked at these diagram in the last class also. We are just looking it again in order to understand it is dynamics. So, the moves important thing to note is that this is an hierarchal bus structure. In fact, there are three levels of hierarchy. We have got the CPU bus we have got PCI bus which is the bus which connects the CPU with other peripherals and there is an ISA bus which is for the slower devices. In fact, among the three buses in this hierarchy CPU bus is the fastest bus. And you will find that we have got a bridge of PCI host bridge which actually does the communication between the CPU bus and other peripherals whenever it is required, the most important and interesting thing to note the communication with the DRAM.

We have already studied the DRAM is a slower device and the CPU bus will be pretty fast because CPU is working at a higher clock rate. So, the PCI host bridge provides the communication interface between the fast CPU bus and that of the slow DRAM device. In fact, it also provides connectivity to the basic IO system ROM as well further this PCI host provides connection with PCI bus. The motivation is a same here you have got slower devices. So, this slower device access will be on this PCI bus and this bridge will actually control the communication between these two buses. In fact, these basic

architectural model, you will find today in most of these buses which are targeted for connecting high speed processors with peripherals and memory.

(Refer Slide Time: 05:26)

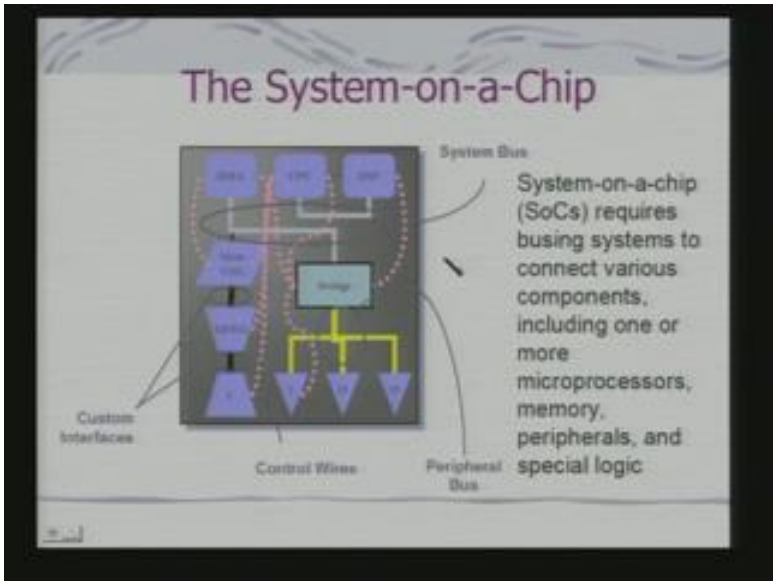


What is really a bridge? A bridge is a slave on the fast bus and master of the slow bus, because bridge is actually sitting on both the buses. And if I look at a typical master slave configuration then you CPU will be the master for the faster bus and bridge will be slave over there. So, the bridge will take commands from the fast bus which as got the CPU and it will become master in the slower bus for carrying out the transactions read and write transactions with the slower devices. So, it will issue common on the slow bus so, effectively it can become a master on the slow bus. So, the basic responsibilities returning results from slow bus to fast bus as well as if something as to be foot from fast bus to the slow bus. It also functions as what is call a protocol translator because I can have the 2 buses working with different transaction protocols.

So, I request from one bus has to be translated into a request compatible with the protocol of the slower bus. So, the translation is also carried out by the bridge so. In fact, bridge you can consider is actually an additional hardware. In fact, in very simple case a bridge would be a simply implementing a protocol machine that is a finite machine, but in more complex cases like PCI bridge it will as good as that of a controller with it is own registers and control logic. In fact, if I now look at system on chip what is a system on chip. It is actually what we have seen in terms of discrete components on a PCB that

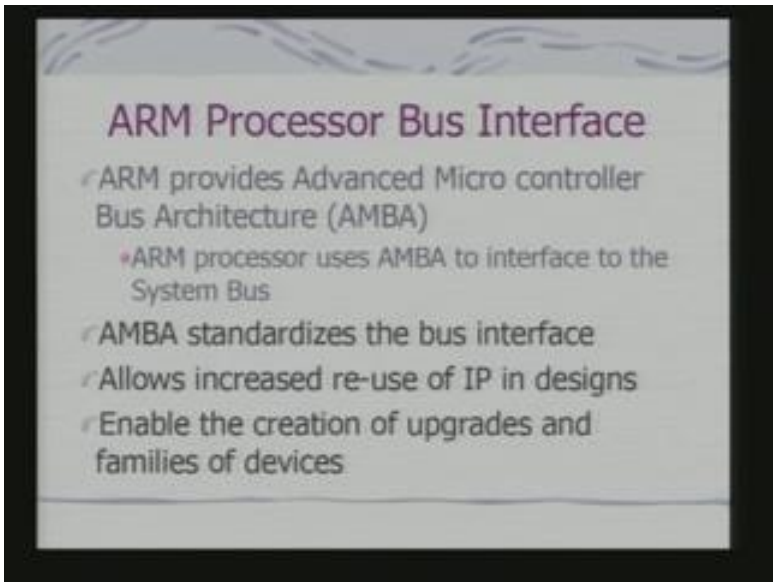
being put onto a silicon element and a System-on-a-Chip can have multiple processors as well.

(Refer Slide Time: 07:33)



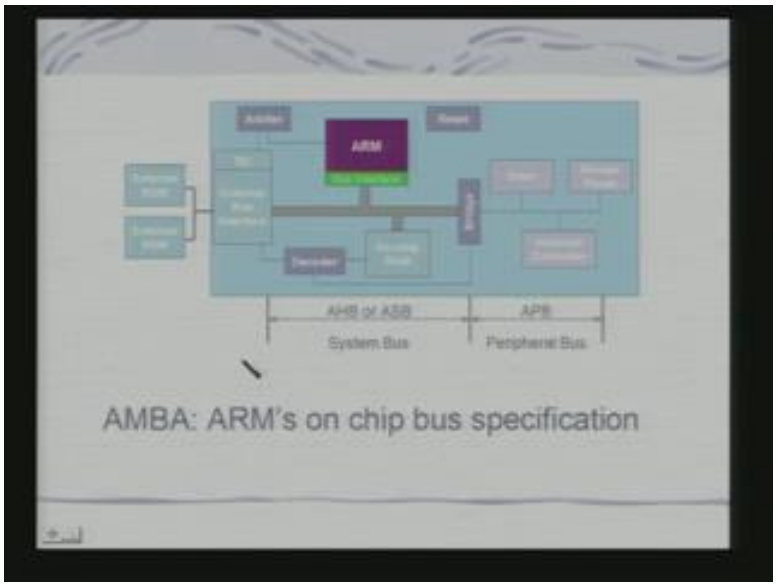
In fact, we have already seen O map which has got a ARM as well as a CP55 DSP. So, this is an example of System-on-a-Chip. So, there will be a fast bus you have a system bus you have a bridge connecting it to a slower bus to which this kind of input and output devices can be connected. There may be also custom interfaces here an example has been given for an encoder. So, if I have the data that empec encoder converts it and it uses a kind of a custom interface to write the data through the memory controller. It may also invoke the DMU DMA controller and DMA controller is actually sitting on these bus transferring the data from this special purpose peripheral to the corresponding devices. It may be even to the memory. So, what we say the System-on-a-Chip requires a busing system to connect various components including one or more microprocessors memory peripherals and special logic like your empec encoder. Now, here; obviously, my bus specification and bus protocols have to be different from what we have consider an seen for a PCB requirement. So, in this context we shall look at ARM Processor bus Interface in many of the SoCs use ARM CPU core as a main driving CPU

(Refer Slide Time: 09:17)



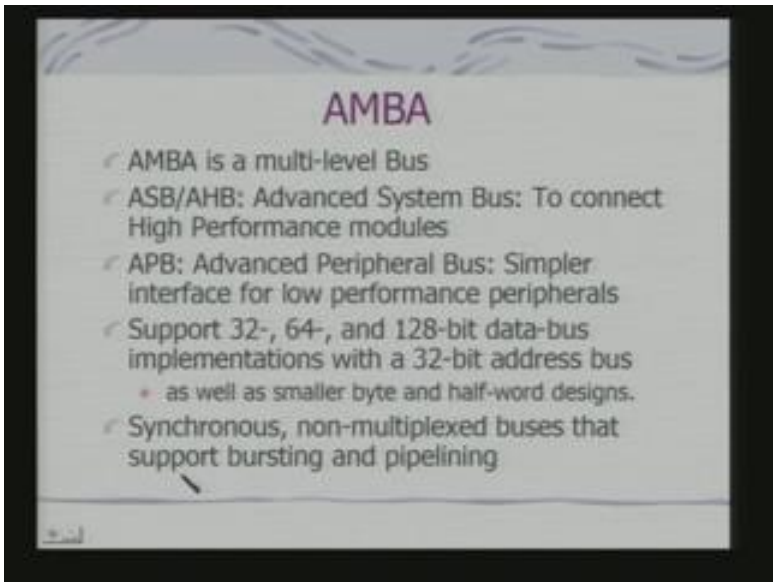
So, ARM provides Advanced Micro controller bus Architecture and this bus is called AMBA bus. ARM processor uses AMBA to interface to the System bus that CPU interfaces to the system bus and the what is the advantages of these kind of interface definition it is standardizes bus interface it allows increased reuse of IP in designs because I can pickup another IP which is compatible to the AMBA interface and I can put them together to design my own SOC and it enabled the creation of upgrades and various families of devices because it devices make to be just compatible in the bus. In fact, this is the basic underlined theme for definition of all the buses and the necessity for defining a kind of a fixed protocol for the buses. So, once I have a devices confirming to the bus protocol they can be used in a system without a necessity for developing special interfaces. In fact, this is the key motivation for defining buses and also studying the bus structures. So, you go back to the picture that we had already seen earlier. So, you have got this ARM processor.

(Refer Slide Time: 10:56)



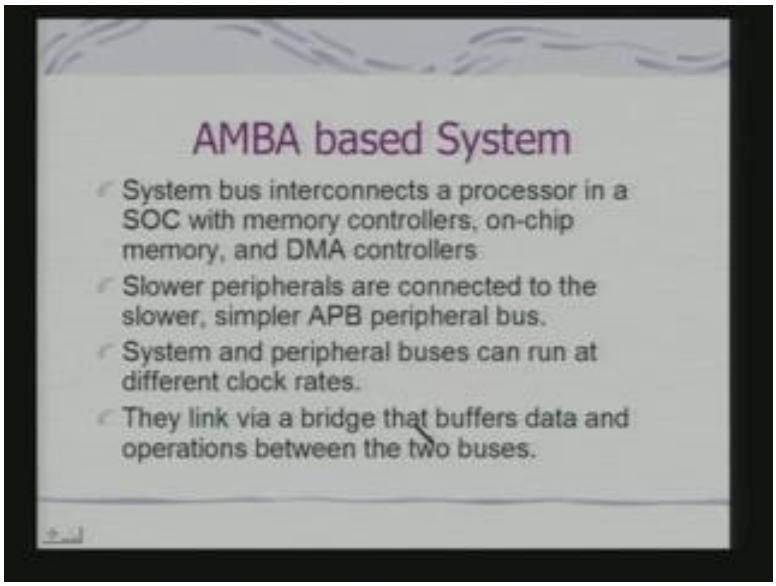
It has got it is bus interface making the interface compatible with what is call AMBA bus. Now; obviously, when I am having a bus and definition of the signals they may not be always same as that of the signals being provided by the core processor core. So, I need to have an interface unit. So, this is the internal bus these bus connect it is what we have called the system bus. These bus connects a processor to the on chip RAM as well as these provides a interface to the external bus interface. So, these are external devices since it is a multi master bus actually there would be an arbiter we are discussed how what is the role of arbiter? Arbiter arbitrates between the request coming from multiple masters sitting on the bus and just like PCI bus here also I have got a bridge. The function of the bridge is a same it is connecting the high speed system bus with the low speed peripheral bus. You have got a timer interrupt controllers and other devices which should be lying on the peripheral bus and what is important this AMBA is basically one chip bus specification. Your PCI 104 or 104 plus why not really one chip bus specification, but actually external to the chip PCB based bus specification.

(Refer Slide Time: 12:46)



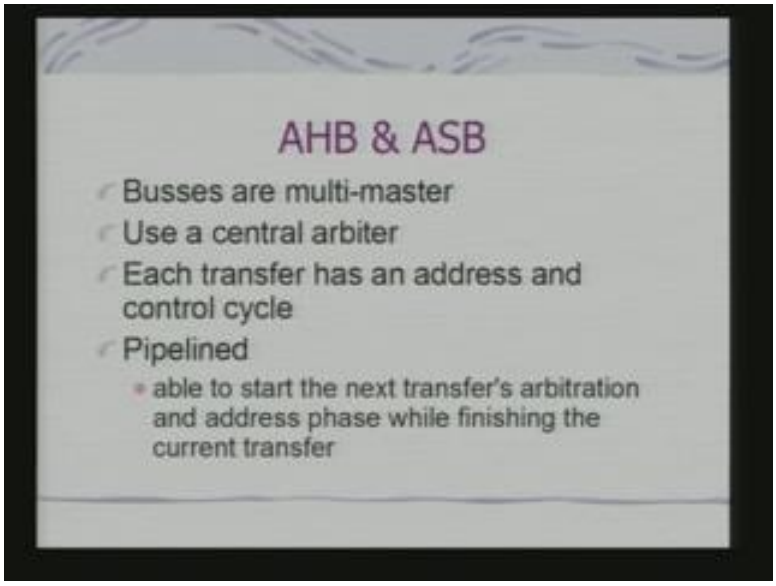
So, it is again as we have seen it is a multilevel bus it has got at the system level the bus is called ASB or AHB. These are two different bus specifications and they have got different capabilities they are they define the structure of the system bus. We shall not really distinguish between them we shall look at their characteristics more in reference to AHB and these parts will translated most of the cases to ASB as well. So, here the objective of this bus to connect high performance modules. The APB is Advanced Peripheral bus simpler interface for low performance peripherals. In fact, the whole bus supports what is called 32 bit 64 bit as well as 128 bit data bus transfers using a 32 bit address bus and this is the basic characteristic of ARM it can also be used for doing smaller bi transfers like that of smaller word transfer though word byte etcetera. It is a synchronous bus non multiplex buses that supports bursting that is bus mode transfer as well as that of pipelining.

(Refer Slide Time: 14:13)



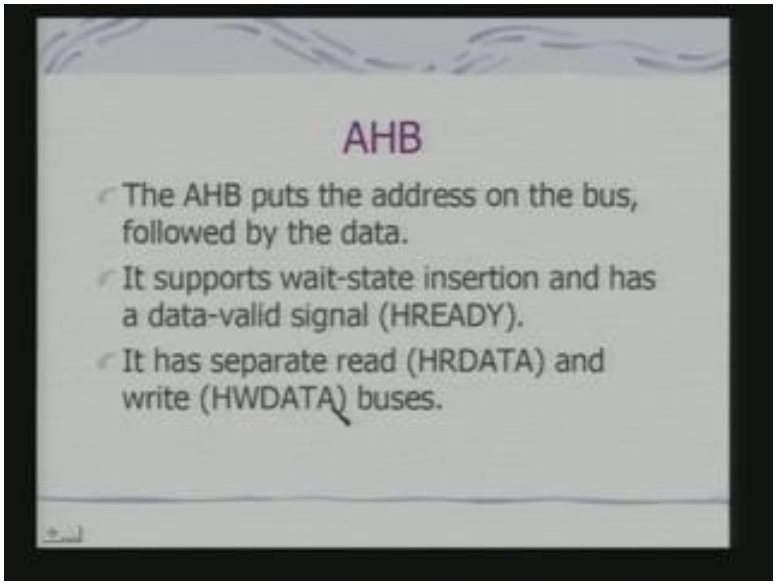
So, system bus interconnects a processor in an SOC with memory controllers on chip memory and typically DMA controllers. Slower peripherals are connected to the slower and simpler APB peripheral bus and this system and peripheral bus; obviously, it is expect to run at different clock rates and they link via bridge that buffers data and operations between the two buses. Why it needs to buffer data? Because, there is a mismatch in the speed between the 2 buses.

(Refer Slide Time: 15:01)



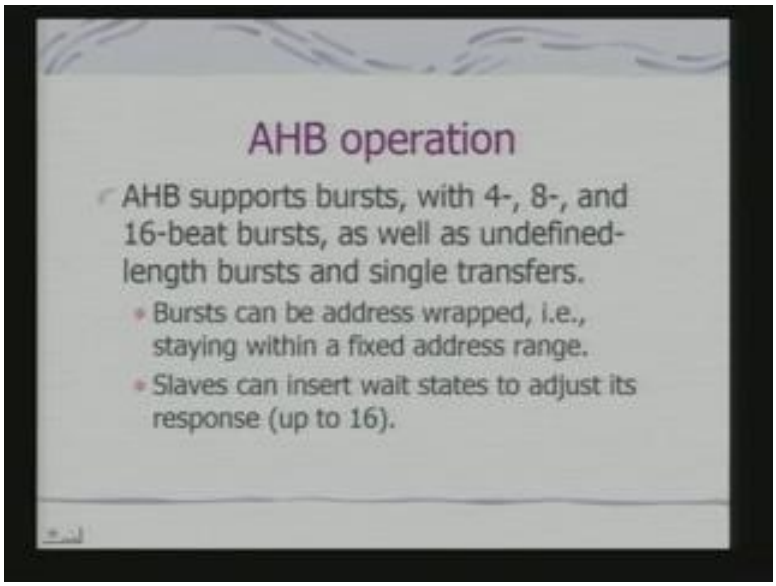
This AHB and ASB which are really the system buses are typically multi master buses. And they use a central arbiter that is the basic arbitration scheme is based upon a central arbiter block and each transfer on these bus has got an address as well as a control cycle. These buses, I already told you is pipelined bus what is that been they can start the next transfers arbitration and address space while finishing the current transfer. While the current data transfer is taking place it can start the arbitration for the next phase so; obviously, since these are pipeline the speed at which the transfer takes place is much more.

(Refer Slide Time: 15:56)



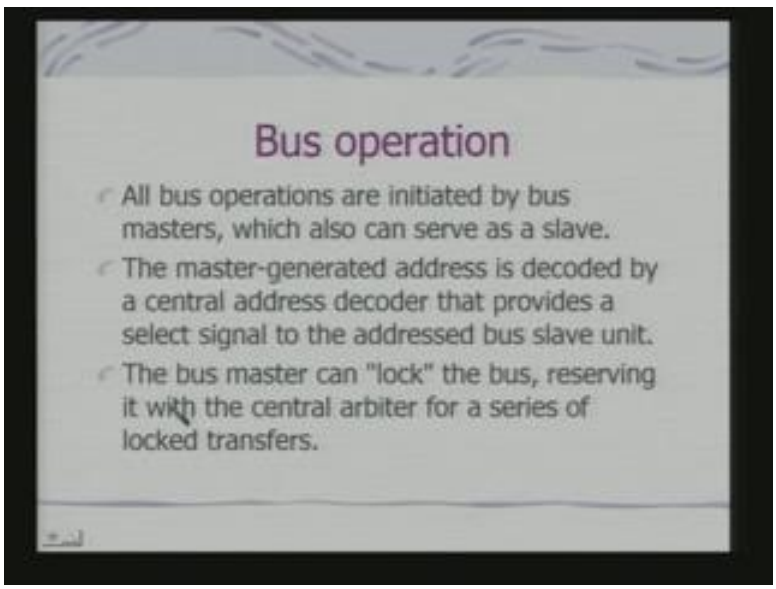
So, how it works? AHB puts the address on the bus followed by the data if it is a write. So, the master provides the data it is supports wait state insertion and uses a data valid signal which is called HREADY which is just take a ready or a wait signal that you fine with ARM.

(Refer Slide Time: 16:28)



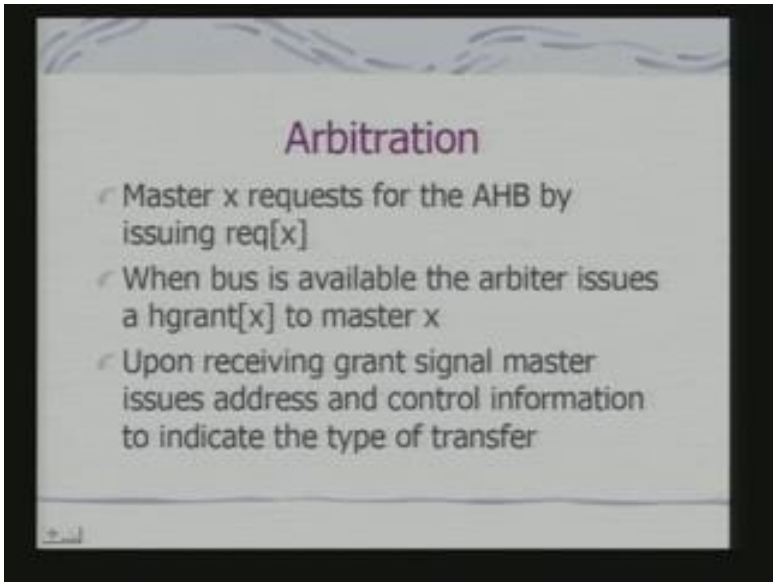
It has a separate read data and write data buses it supports bursts bursts can be in terms of 4 8 and even 16 watt transfers at one go. And these bursts can be addressed, wrapped that is this burst with saying within a fixed addressed range and slaves can insert wait states to adjust it is response. In fact, if you remember we say that if it is a sequential access the whole issue that comes up is you can generate address once you can arbitrate and generate address one and the master can have the bus for transferring up to 16 words at one go. It can be even undefined linked bus as well as there can be simply single transfers.

(Refer Slide Time: 17:21)



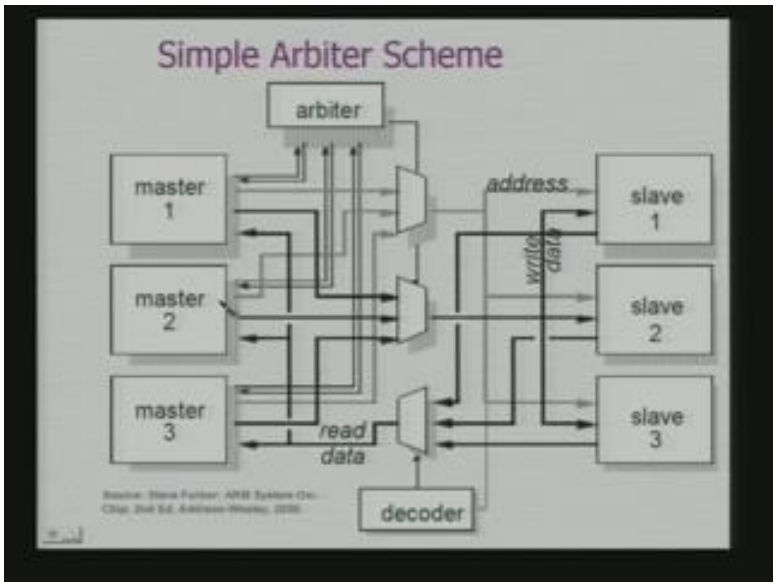
So, all bus operations are initiated by bus masters which can also be slave. So, there is a interchangeable role between master and slave. The master generated address is decoded by a central address decoder that provides a select signal to the addressed bus slave unit. The bus master can lock the bus reserving it with the central arbiter for a series of locked transfers. So, this is this can enable it to take over and an initiate fast transfer on the bus. So, how does really the arbitration takes place each master on the bus has got a unique identifier. So, it is a multi master bus. So, each master has got an unique identifier.

(Refer Slide Time: 18:01)



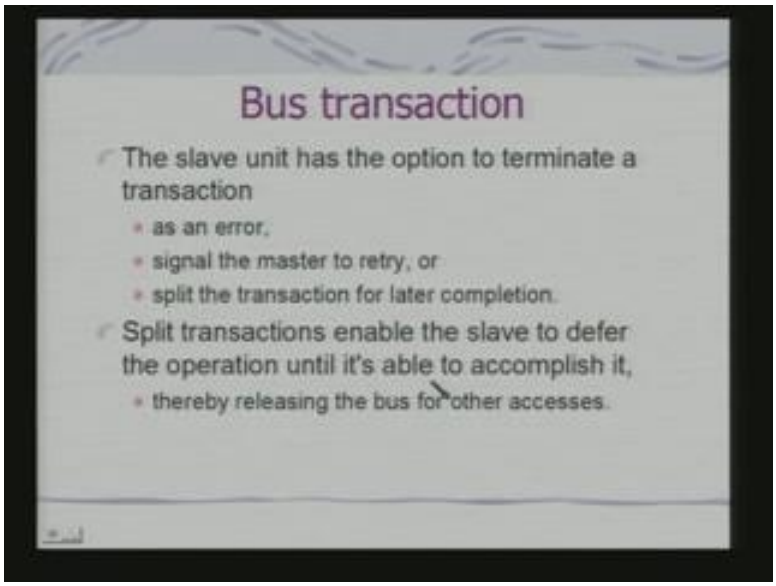
So, master x request for the AHB by issuing what is called a request signal x is the indicating the corresponding the master slave when bus is available the arbiter issues what is called a hgrant[x] to master x. So, master knows that bus is available with him upon receiving the grant signal the master issues address and control information to indicate the type of the transfer. So, the master has to first get the bus and then do the transfer. So, while one transfer is taking place another master can initiate a request for the bus which arbiter can process. So, the whole thing can be actually overlapped in time. So, if you looking do it the basic picture is something like this. So, I have got this as an central arbiter.

(Refer Slide Time: 19:17)



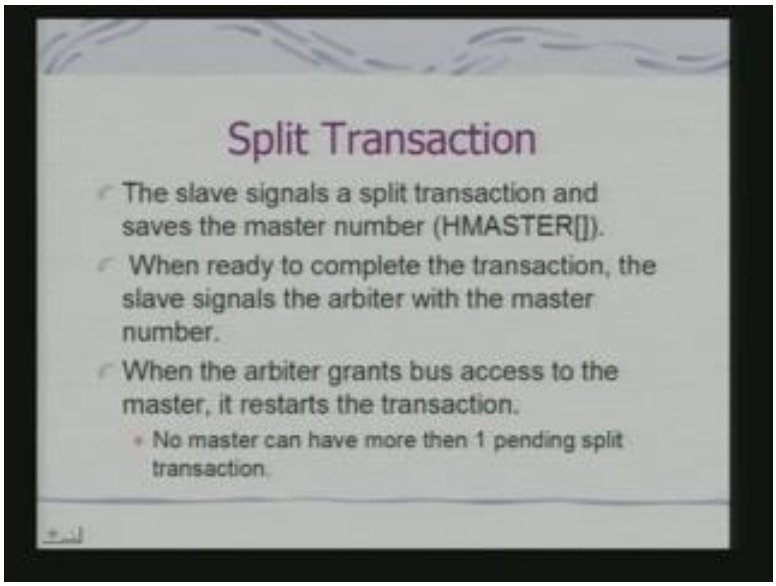
These are multiple masters which are actually sitting on the bus they will generate the request and arbiter will generate the grant once it resolves the priority. If there are multiple request coming in an the same point in time arbiter will use it is own priority scheme. We had discussed a priority scheme in the last class will use it is own priority scheme to identify which master now gets the bus. Once it knows, it enables these buffers. Once it enables these buffers the address would flow from the corresponding master to the target slave device. First the address would go. In fact, the whole point is once you have the address the decoder actually knows which of these is to be enabled an accordingly which slave device is now targeted to receive the data. So, which slave device is provided with an address and using that address master refers to the slave device?

(Refer Slide Time: 20:31)



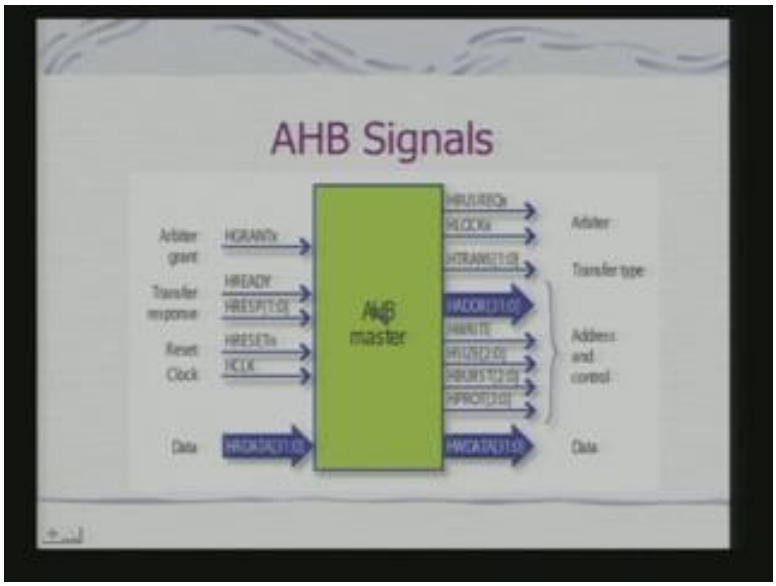
In fact, the slave unit has some options as well. So, slave unit has option to terminate a transaction as an error if it fines there is an error condition currently valid with that peripheral it can flag an error. It can signal the master to retry or split the transaction for later completion. In fact, split transaction is an advanced feature which is available with edge bit. Now, the split transaction enables the slave to defer the operation until it is able to accomplish it and thereby releasing the bus for other accesses. So, what does split transaction really mean? In fact, the slave once it has received a request for a transaction and it fines then it is not to able to complete the transaction. It will indicate to the master that the transaction can be splitted.

(Refer Slide Time: 21:50)



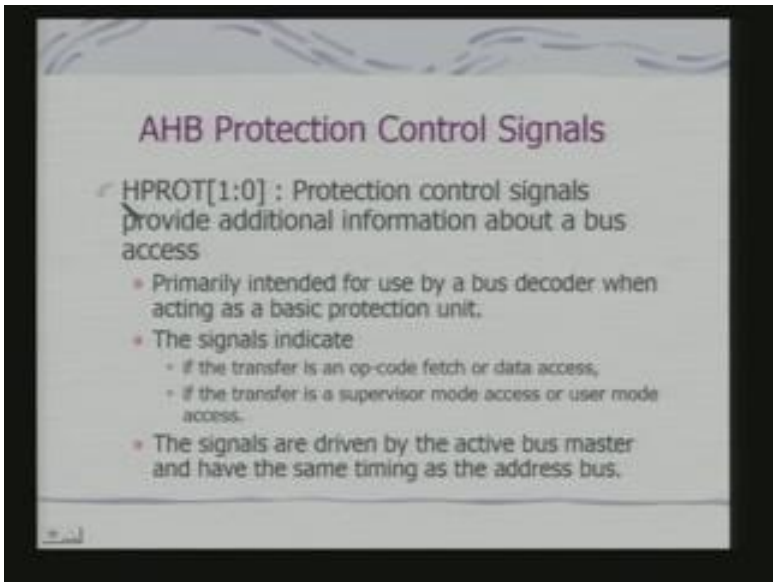
So, the slave signals as split transaction and saves the master number, because each master has got a unique ID I told you that earlier. When ready when it is ready to complete the transaction the slave signals the arbiter with the master number. When the arbiter grants bus access to the master it restarts the transaction. In fact, the convention is no master can have more than 1 pending split transaction. I hope you of understood why this transaction is got a split transaction because master had initiate at the transaction. For some reason slave is currently not able to complete the transaction. So, it signals as split transaction and retains the master number then the slave request arbiter that I am now come ready. So, please provide the bus to the master to carry out the transactions. As long as it is not ready the bus becomes available for other operation. So, this is a more sophisticated way of doing a transaction with a peripheral. So, what are the AHB signals if I have if I look at these as an AHB master?

(Refer Slide Time: 23:14)



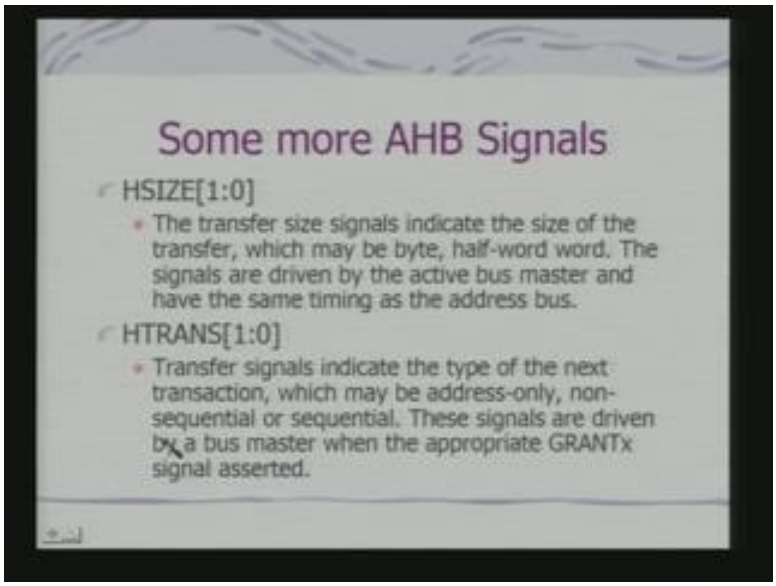
This a kind of a conceptual model they can be any CPU or a DSP can be an AHB master. The whole issue is that the interface bus interface unit of the AHB master shoot have compatibility with these kind of signals. Already, we have seen that it will generate the request this is the request to the arbiter. It will get the grant from the arbiter if the slave wants to introduce waits state this is the input which will come from the through the wait state. The response keeps the states the response from the slave and these are; obviously, address and the control signals. These are address and control signals this is the data this is your you have got your write data bus as well as read data bus which are two distinct buses and these indicates the transfer type.

(Refer Slide Time: 24:20)



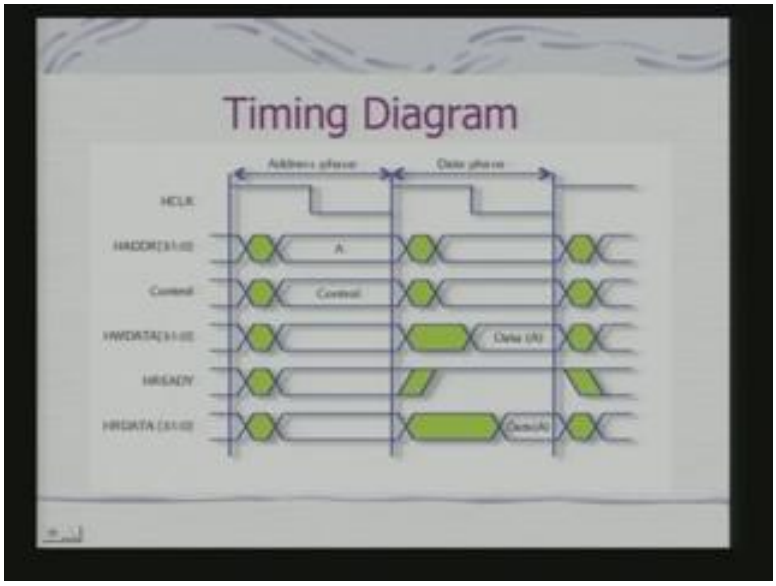
You have seen that there was signals called HPROT. In fact, this is point we had not discussed earlier. These are protection control signals provide additional information about a bus access it is primarily intend for use by a bus decoder when acting as a basic protection unit. So, the signals indicate if the transfers is an op-code fetch or data access if the transfer is a supervisor mode access or user mode access. The signals are driven by the active bus master and have the same timing as that of the address bus. So, this a protection information and we call this as a protection data these signals really do not provide the production, but these signals actually encode what the staters of the current transaction by making use of these bits. The bus decoder can function as a basic protection unit a simple example is if a transaction is taking place in user mode. It will not be permitted may be to access register of another processor or may be another memory area which is granted simply supervisory access write. So, that protection can be ensure in hardware through the bus decoder and that is why the protection signals from part of your bus signals specification.

(Refer Slide Time: 25:58)



Some more AHB signals are size. In fact, this size we have already seen in the context of ARM signals memory interface signals also an it is consistent in that context and edge trans. This transfer signals indicate the type of next transaction which may be address only non sequential or sequential. These signals are driven by a bus master when appropriate GRANTx signal is asserted. So, that to prepare the devices to expect what kind of transactions would ((refer time: 26:36)) next.

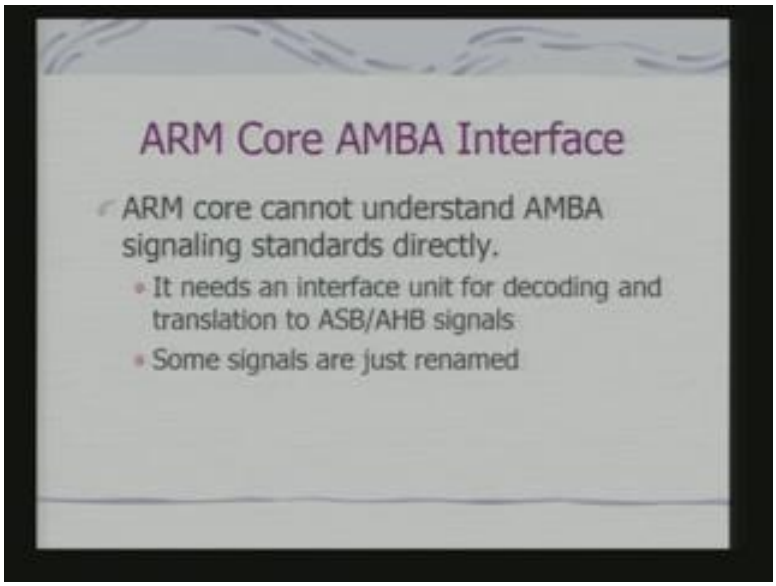
(Refer Slide Time: 26:47)



Let us look at it is timing diagram. In fact, I said that they are synchronous bus. So, if it is a synchronous bus the whole thing is actually driven with respect to a clock. So, all transactions I said as got a address and control phase. This is followed by data phase. In fact, the address is provided on your edge address line and you have got the control information coming in. So, these control information has also provided in the address phase to prepare the devices for the targeted transaction and these effects are write operation being initiated by the master. The data should be available at this point itself and if and when there is edge ready signals what happens if you have these when you have these edge ready signal depending on the status of the edge ready signal. There will be wait states introduced. So, in these case your data which is here gets it is delayed I introduce wait state of are here. Similarly, for the read also there will be the data phase which will get delayed.

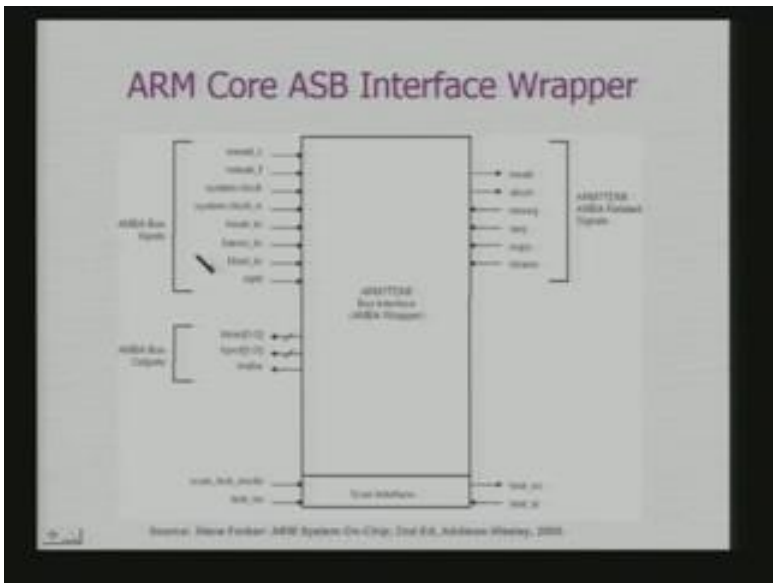
Now, these timing diagrams, is very very similar to what we have seen in the context of a simple memory interfacing. But the overall transaction definition with respect to the bus which involves communication with the arbiter and grand signals as well as the response signals coming from the slaves makes the transaction much more complex than that of accessing a simple memory. So, the bus structure since it needs to deal with a variety of devices. So, the signal definitions the protocol definitions is much more complex than that of a simple memory interfacing. In fact, memory also is a different speed. And that is precise their reason why we need these kind of a variety of signals and to speed up. In fact, to club transactions together you have the information about the next transaction which you takes place to implement the protections. You have got the protection signals defined as part of the bus itself. Now, the interesting feature is also you should note is this this is the one chip bus. So, for these bus there are know mechanical parameters really specified. So, now how does ARM core interface with among them.

(Refer Slide Time: 29:48)



Let us take simple example of ARM7 and we have seen ARM7 signals. So, ARM7 signals is not directly are strictly compatible with AMBA signal in standards. In fact, they do not translate directly. So, typically you need on interface unit for decoding and translation to ASB or AHB signals. Some signals may be just renamed when it is directly their and. So, this is what is important when I am using IP core. So, if I choose to design and IP core compatible with the bus I need to define also the interface unit which will make the core compatible with the bus.

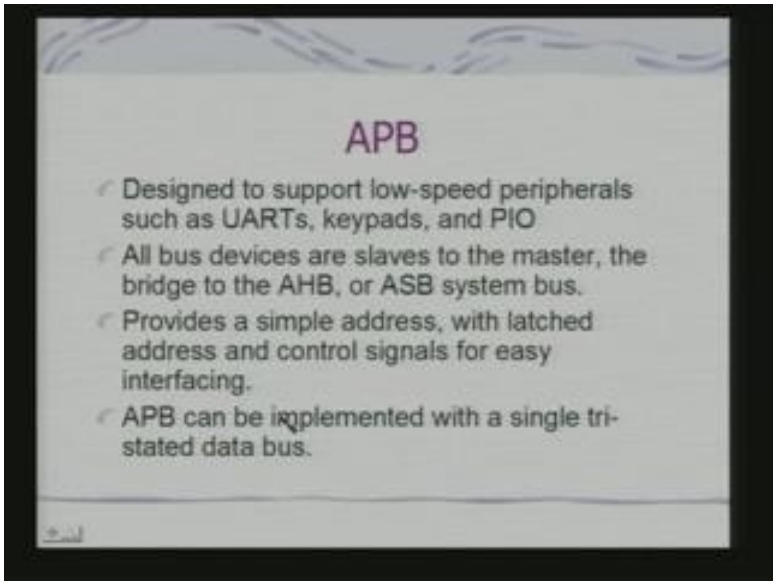
(Refer Slide Time: 30:43)



So, here it is an example of what we say that bus interface which is nothing but a ((refer time: 30:48)) it is wrapping the basic CPU. So, what we are looking at is you have got this amber bus inputs these are your amber bus outputs and this is primarily with relation to ASP interface. So, these ASP interface is. So, you will find some difference in the signals with respect to the AHB signals that we had discussed earlier. Now, here what we have got? You have got the signals coming from ARM7 this is basically the ARM7 ((refer time: 31:23)) amber related signals which are directly available and these are interface onto the amber bus. So, you will find that here in fact, the signal HREADY that we had discussed in case of AHB bus here it is ((refer time: 31:45)) because that definition slightly different in the AHB bus.

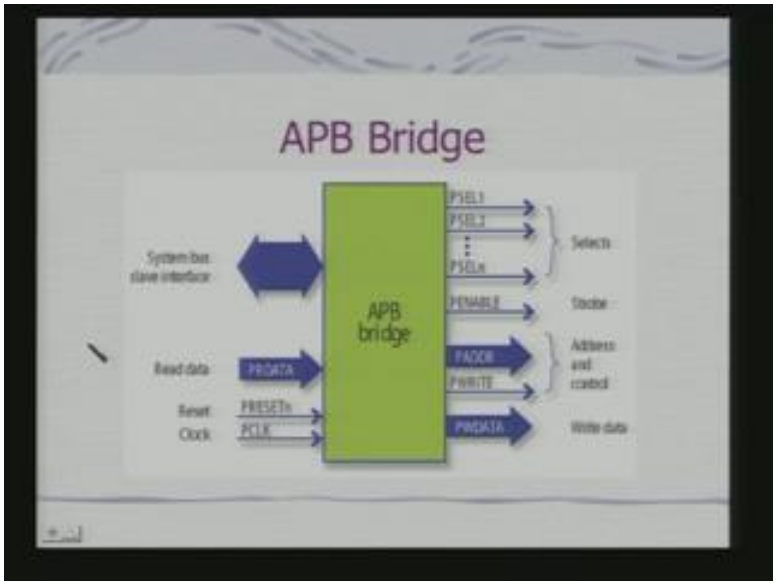
So, this is ((refer time: 31:52)) and which is equivalent to that of N wait on the chip itself similarly you have got the sequence the ((refer time: 32:03)) you. In fact, we have used to see how this can be used for accessing BRAM in a in a bus mode where I can get data from after selecting a row from consecutive locations. So, this can this should be used in kind of define defining what kind of transaction is actually taking place also the size information when we provide in AHB bus that size information would be obtained from the outputs provides by the ARM chip. So, this is basically an interface unit. So, with the CPU code typically you design an interface unit to provide it to provide it with the ability to communicate on the AMBA bus.

(Refer Slide Time: 33:02)



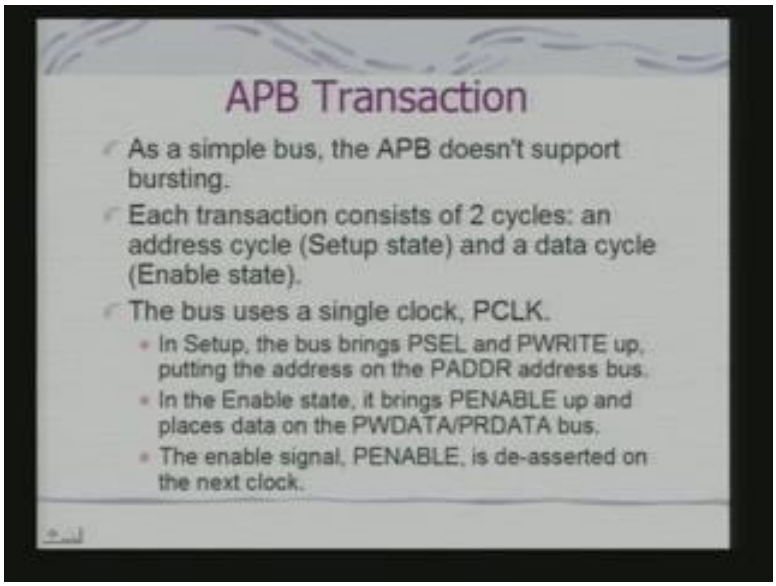
The next thing is your peripheral bus. So, the APB are the peripheral bus is designed to support what we call low speed peripherals such as UARTs keypads and general IO parallel IO. All bus devices in these cases are slave to the master what is the master in these case the bridge. All bus devices are slave to the master if you remember I said basic definition of a bridge bridge is a slave in the high speed bus and master for the low speed bridge. So, here the all devices are slaves to the master the bridge to the AHB or ASB system bus. It provides a simple address with latched address and control signals for easy interfacing and APB can be implemented with a single tri-stated data bus. So, it is very very similar to kind of a bus which you encountered for with 8085 for 8086 family and. So, it has got also the latched address the address gets latched and control signals it is a it is a very simple straight forward bus.

(Refer Slide Time: 34:20)



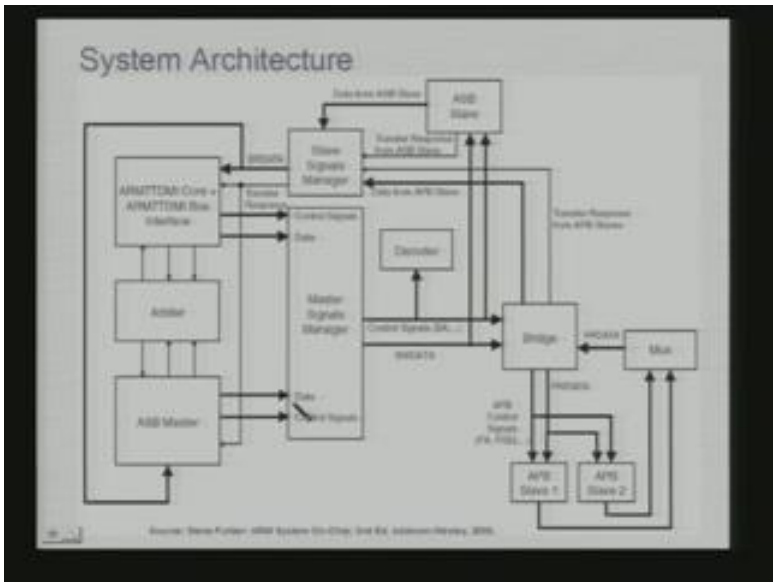
So, an APB bridge now we talked about instead of AHB master now here it is an APB bridge. So, it will have got a system bus. This is a slave interface and these are the signals which goes onto the bus. So, it is basically selects the device. It has got it enables this is the basically the strobing signal for operations on the bus when these are address and control signals. And this is a write bus this is for writing the data and this is for reading the data from the master that is the high speed bus.

(Refer Slide Time: 34:59)



So, the basic operation is very simple. In fact, it does not support bursting because we are not really looking at high speed transparent. In fact, each transaction here also consists of 2 cycles an address cycle which is a setup state and a data cycle which is called an enable state. So, in fact, the basic the cycle definitions are with respect to the enabled signals. The bus uses a single clock call PCLK in setup what happens the bus brings the PSEL and the PWRITE up putting address on the PADDR address bus. These are the address lines. So, the address is put on the address bus in the enable state it brings PENABLE up and places the data on the PWDATA or PRDATA bus. So, enable is. In fact, the data cycle and this data cycle is actually controlled by your enable signal. In fact, the enable signal will be de-asserted on the next clock cycle. So, you can realize it is a very straight forward and simple bus consisting of two cycles and that is the typical feature of most of the peripheral buses.

(Refer Slide Time: 36:33)

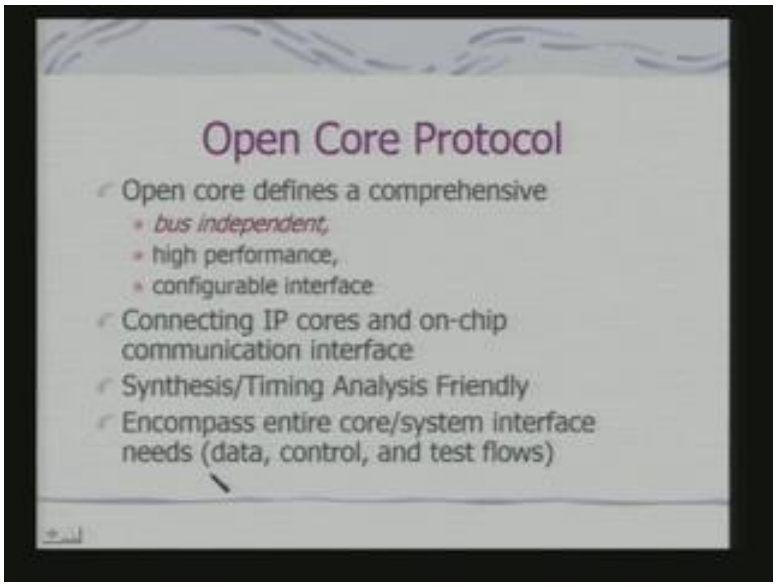


Now, if you look at the complete system architecture this would look something like this. And this is what you should note carefully, because that tails you have a complete SOC architecture internally would look like see you have got the ASB again we are looking at using ASB. These are all ASB masters because and this is your ARM7 core ARM7 core would communicate with you an arbiter. This is another ASB master, because there can be multiple bus masters on these on these bus. You have got these bridge now, this bridge is actually a slave there could be other slaves as well, but bridge is actually a slave with of the master bus. And your peripherals are connected to the bridge and this master the control signals communicate from the master signals manager to the bridge. And the whole idea is these data which comes from here can be to any of these slaves and these response if you remember there was a response signal flowing in. So, the slave sends the response signal.

So, that response signal goes to the ARM core are to be other master. This is a transfer response. So, what is the response to the transfer transaction which is currently being executed? And this is your data which flows from the slave back to the master and this is where the arbiter is. So, once the request comes in the arbiter resolves and gets a grant to the core. Once it gets a grant then only the transaction is initiated. So, this is architecture basic architecture of an SOC using the 2 buses there is a hierarchal bus structure and all data transfers here or in parallel. So, they are both actually parallel buses and this your system bus which can be ASB or AHB is capable of handling a variety of complex

transactions and these gives more flexibility and speed to the bus. In fact, bus as such is the key issue for any kind of SOC designs for Embedded System. I may have a very high speed processor, but if I do not have and efficient bus transaction protocol design then I cannot really exploit the processor features. In fact, I may have multiple processors, but I cannot I shall not be able to explode the characteristic optimally until in the analysis has designed optimal bus connecting them as well as with the memory at peripherals.

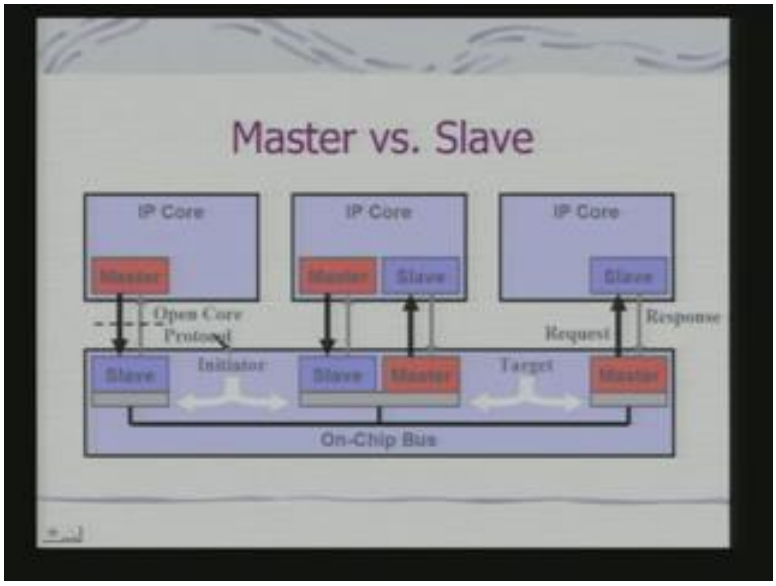
(Refer Slide Time: 39:54)



The other issue, other trend which develop to, relate to bus for SOC is, what is called Open Core Protocol. In fact, the example that we have seen so far is that of a particular bus. In fact, proposed and use then popularize by the ARM processor manufactures. So, thus got a definite transaction definition is that got a specific signal definitions. And when we designing an SOC your picking up the modules compatible with this bus structure. This Open Core Protocol why while discussing it clear is conceptually different what it tries to define is a comprehensive bus independent high performance configurable interface. In the ((refer time: 40:51)) all your interfaces where predefined and your actually making your IP cores compatible with though definitions. Here what we have talking about is a kind of an open core protocol. So, it defines a comprehensive framework and using this framework you can actually design and implement components in an associates.

So, it provides a kind of a communication interface and why it is bus independent because you will be actually defining the bus which is consistent with the protocol. Because a protocol can have a variety of transactions defined depending on your requirements and the device characteristics you may define the bus. But if I pickup an IP core an if I say that my IP core is compatible to the Open Core Protocol; that means, it can support the transactions which have been defined as part of these protocol. So, what in these context. It is synthesis and timing analysis friendly, because I have to put everything to gather to synthesis chip and it encompass entire core and code or system interface needs that is data control and test flows.

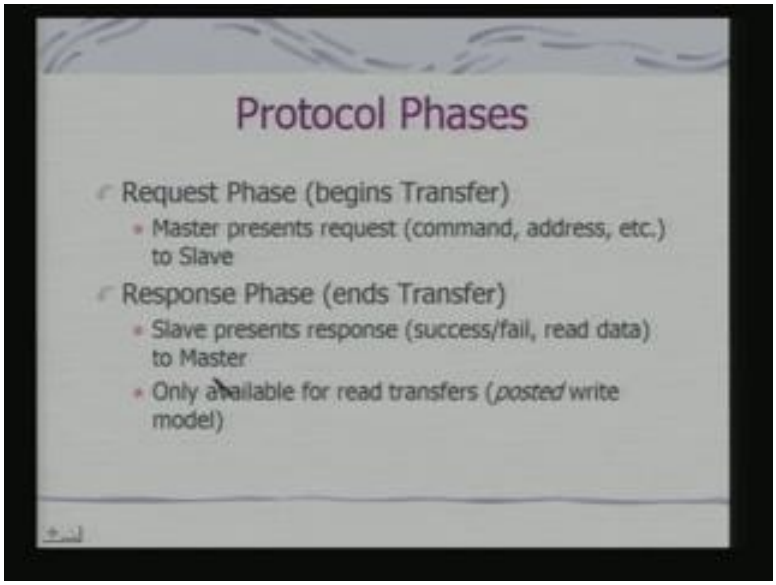
(Refer Slide Time: 42:19)



Test is another important issue. So, basic idea look at these that I have got this IP core these are different IP core. This is basically my on chip bus and these on chip bus have to be definitely in a hardware implementation in terms of a signal. But these request that my master and the slave would may these requires are the transactions are defined through this Open Core Protocol. So, once these transactions are defined through Open Core Protocol. So, there is a standardization in the nature of transactions. So, tried to understand this this in case of AMBA bus AMBA bus had what as set of signal definitions as well as as set of transaction definitions. In fact, we had discussed for AHB bus transactions like burst mode transaction then transaction with security features split transactions. So, all these transactions define as part of that burst definition itself. Here

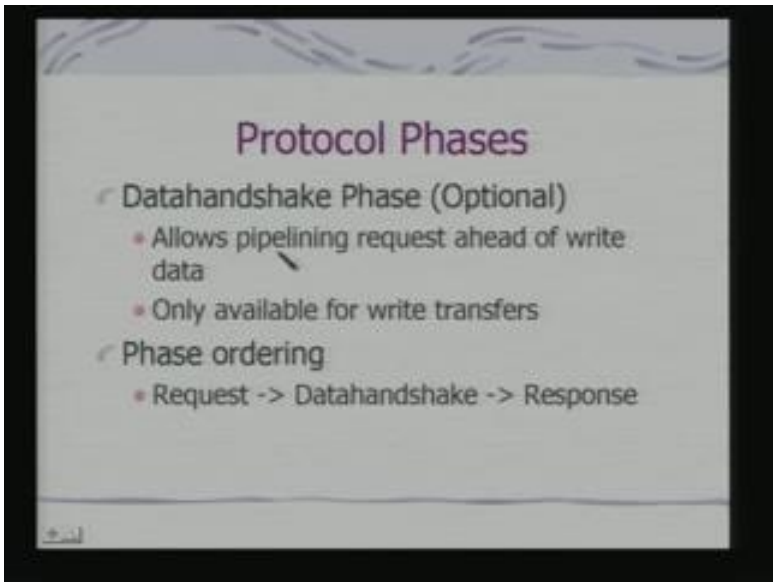
what we are trying to define is basically the protocol for this transaction. So, if you have a standardize protocols basically the IP cores would be compatible with these protocols.

(Refer Slide Time: 43:35)



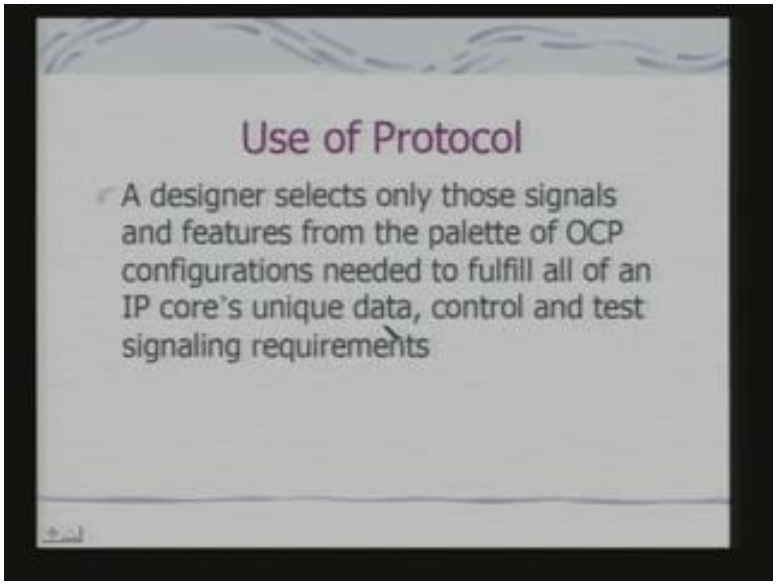
So, they are Protocol Phases in this case in the standard if you look into you will find a very similar kind of a protocol, but just structured in a slightly different way. You have got a Request Phase which begins the transfer master presents request command address etcetera. There is a more the elaborate definitions I am not going entry then just giving the basic idea. Response Phase is ends transfer slave presents a response success or fail or read data if it is a read operation it has to be data to be presented to the master. I am this Response Phase is typically available for read transfers. In fact, if you see that there also similar thing was also there in the ARM, but here we are trying to evolving to a generic protocol phases and protocol definitions.

(Refer Slide Time: 44:32)



Similarly, there is a Datahandshake Phase which is a part optional a part of your protocol phases. So, what we says that allows pipelining request ahead of write data and only available for typically write transfers. So, the Phase ordering typically for a transaction would be request at Datahandshake followed by response.

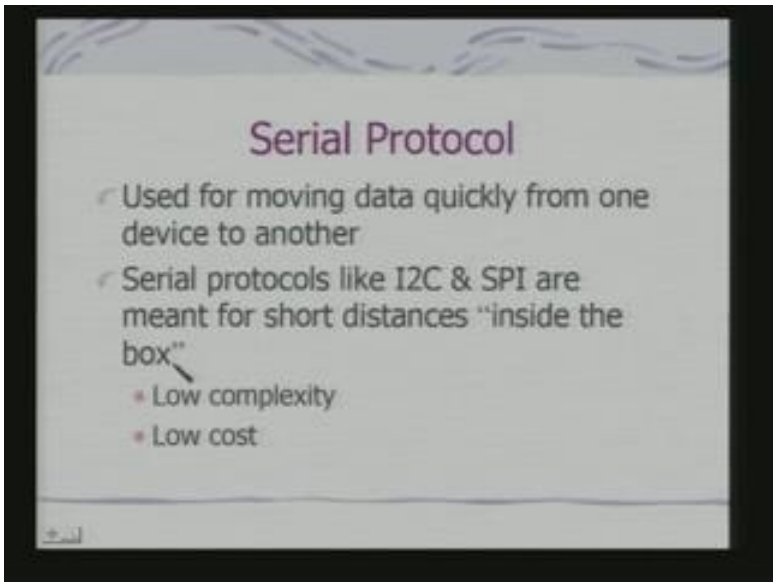
(Refer Slide Time: 45:09)



In fact, this is the generic form of most of these protocols and Open Core Protocol is basically just defining this protocols for transactions on the bus and how do you use this protocol. And designer selects only those signals and features from palette of OCP what

you call open call protocol configurations needed to fulfill all of an IP core's unique data, control and test signaling requirements. So, try to follow the difference in case of AMBA it is a complete specification. In these case it is not a complete specification you a defining a standard and you have got in the devices are the IP codes which a compatible with these protocols depending on what are the features you will be using in an SOC. You can pick up a set of signals and define your actual bus because using those signals you will be implementing those aspects of the protocol which is required for your SOC. So, therefore, your platform becomes more flexible your choice is more flexible and generic. In fact, this is trend which is emerging for defining buses in an SOC application. Next we shall look at serial buses. In fact, when we have looked at these SOC and PCB based application. This whole concept of the buses has been parallel data transfers if you are transferring 32 bit I shall be transferring that on parallel lines. Obviously, this can be use for short distances that we did not cheap or within hard.

(Refer Slide Time: 46:59)

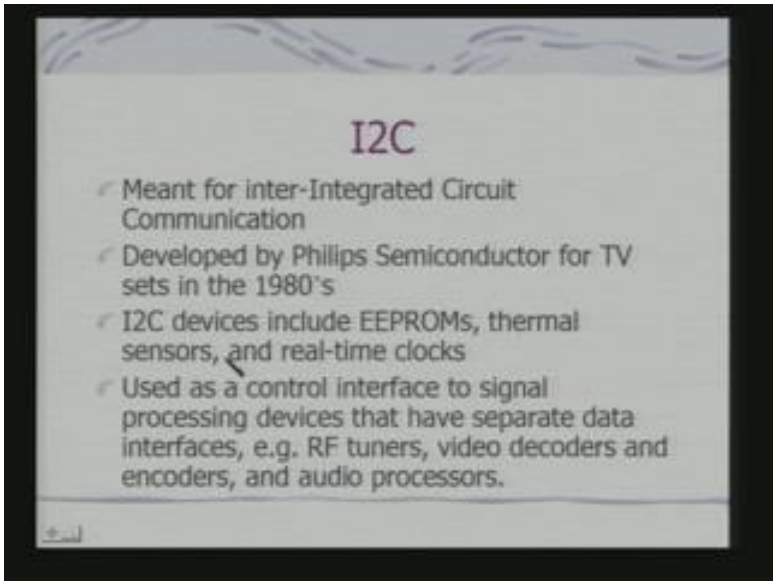


The serial buses, that we are now talking about this is also for short distances is not really very large. And the basic idea is for moving data quickly from one device to another. And the basic objective is; obviously, if you have serial protocol your varying requirement is minimum. So, if you are trying to put a complex system that is you have starting with a pick microcontroller and pick with pick microcontroller. Let us see want to interface additional devices and putting them into an appliance a box. Now; obviously, would like to use minimum number of wires signaling wires to do the connectivity. If

you are using your PCI your ISA it as go an elaborate number signals. And it is signal definitions which itself it would occupy space and your hertz have to be compatible with those definitions.

If I use a kind of a serial protocol I have only very few signals to connect. So, my foot print can be further reduced and. In fact, that is the precisely reason why this kind of I2C SPI. This kind of a serial protocols a serial bus protocols has become popular and being widely used in Embedded Systems. The whole idea is low complexity as well as low cost and it is meant for short distances inside the box itself. So, it is not really on chip. In fact, we can have on chip as well, but typical use is within the box admits may be connecting a big microcontroller with another device.

(Refer Slide Time: 48:45)

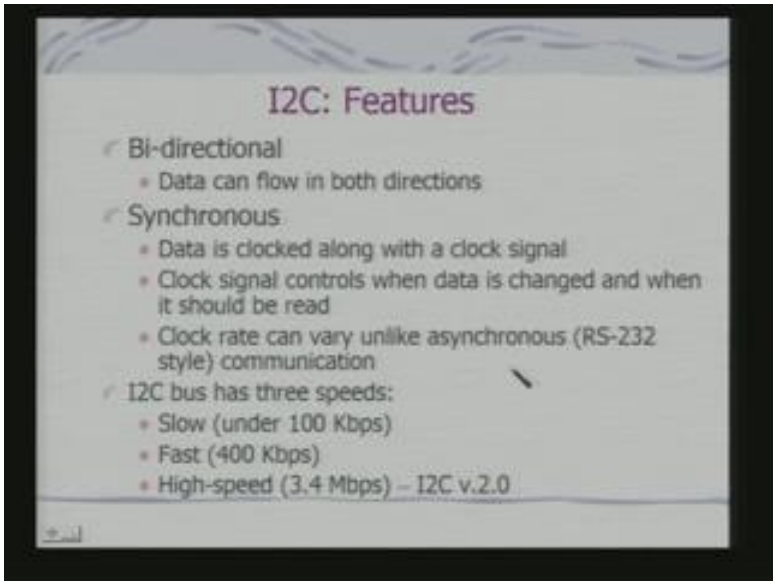


I2C; in fact, the basic name is inter Integrated Circuit Communication this is for communication between 2 ICs. In fact, it was developed by Philips Semiconductor for TV sets in 1980's. So, it is a pretty old kind of a serial protocol. So, originally what was it being used for was it was used as a control interface to signal processing devices that have separate data interfaces. For example, RF tuners video decoders in fact, originally it was being used for this kind of tuners on other applications. Today is also used for RF tuners video decoders encoders and audio processors.

Now, what is this mean it beans that see these devices will be transferring say let us consider a video decoder. So, add an audio processor it would transfer the data through a

separate data bus and. So, they have a separate data interfaces depending on the task. Now, this devices have to be control and this control scheme is not always very simple and straight forward you need a set of control information it there may be a set of instructions which may be sent to the audio processor to do have a write your task. Now, you need a control interface for doing that communication. In fact, I2C is used for that purpose? In fact, I2C devices include EPROMs electrically erasable PROMs thermal sensors real time clocks and lots of lot many sensors are actually used with this kind of I2C devices.

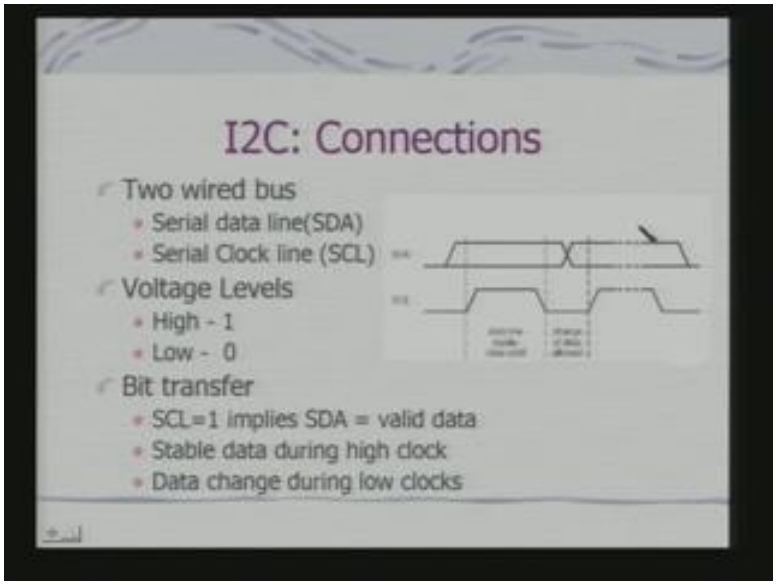
(Refer Slide Time: 50:35)



What are the I2C features here the data can flow in both directions, but it has got base 2 wires just the clock and the data line. So, it is not a full duplex communication, but actually an half duplex communication. It is a synchronous bus. So, the data is clocked with the clock signal. In fact, clock signal controls when data is changed and when it should be rate. In fact, that clock maintains the overall control. The clock rate can a vary unlike asynchronous communication in an asynchronous communication RS 232 C. You have to have a fixed clock rate because you are sampling the bits. That means, you are finding out whether there is a start transmission or not at the clock intervals, but since here the clock is going transmitter along with the data. The clock determines the valid data transitions. So, they are may be change in the clock speed as well. In fact, typically you have got three speeds slow fast high speed this is I2C version 2.0. We shall primarily

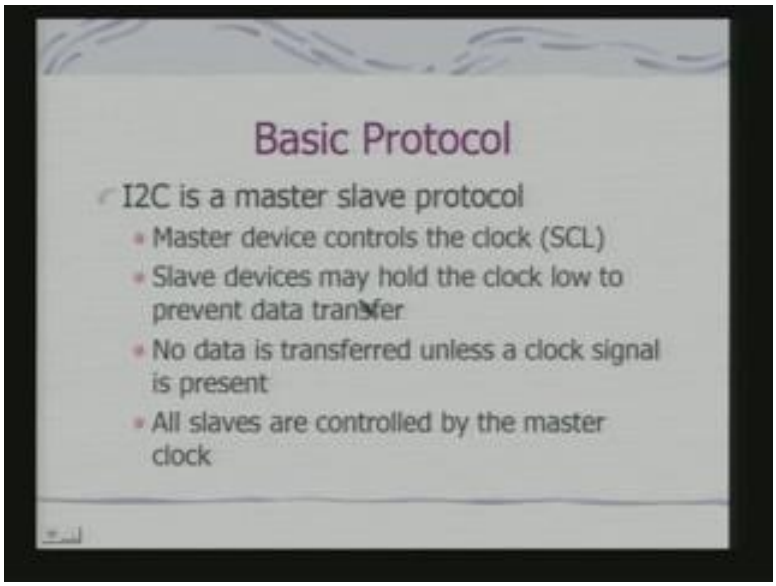
talk about the older version which is slower and this between this two. So, I told you already my I2C is typically a serial connection.

(Refer Slide Time: 52:12)



It has got 2 it is a 2 word bus one is called your serial data line another is serial clock line. In fact, your data so, what we say that actual bit transfer takes place when your SCL that your clock is 1. So, SCL equal to one implies SDA is having valid data. So, the data meats to remain valid during the high clock and the data changes during low clocks. So, the transaction would take place during low clocks. So, the data transition takes place when the clock is low. So, change of the data is allowed when the clock is slow and these clock is actually therefore, controlling whatever bus transactions the takes place on the I2C bus.

(Refer Slide Time: 53:35)

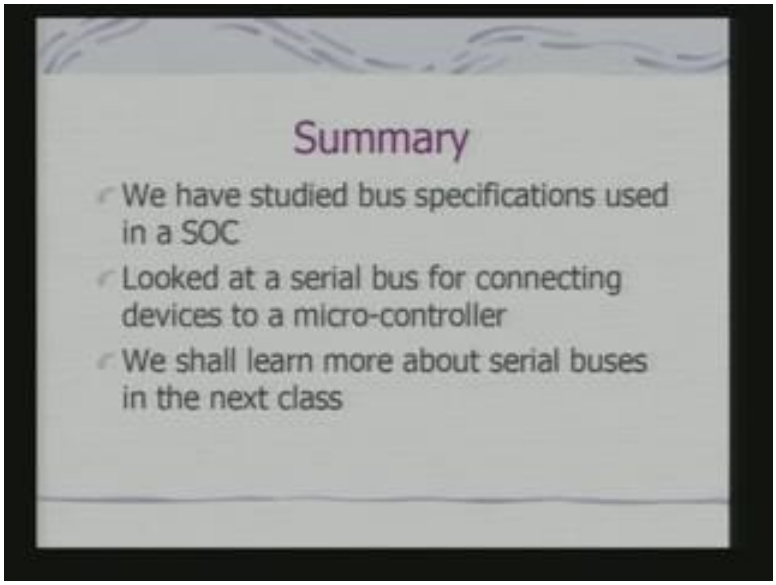


The basic protocol is a master slave protocol. In fact, there is no external clock generated. There is master device the current master. In fact, master and slave can be again interchangeably. Current master can become slave for other application. So, master device controls the clock. This slave devices may hold the clock low to prevent data transfer . So, master is driving the clock. If the slave and there is a single clock line to which the slave is also connected slave can pull the clock low if slaves pulls the clock low then you data is no longer value. So, effectively the master is put into a weight state. So, no data is transfer unless a clock signal is present. So, if a slave pulls a clock low then no transfer can take place. So, if there is a speed mismatch then the slave would do, what simply pull the clock low and all slaves are controlled by the master clock. Obviously, this implementation would require a kind of a wired and connections.

In fact, all these interface lines are part of open collector or open drain connections. So, you have to put pull up registers to connect these interface lines connecting them to the actual supply high voltage so, that they can become operational. So, since it is wired on you can understand that the normal state would be high. So, when there is no transmission there would be high and if there is a one of these slaves pull things down. So, it will be low across the bus. So, you can realize that using just 2 lines just using 2 lines they way by defining the circuit connection appropriate sophisticated protocol can be done. Obviously, it is not as sophisticated in terms of the protocol as that of your ASB or AHB. But still you have the features of multi master control arbitration everything is

built using just 2 lines without involvement of an external arbiter. In fact, this is what we shall discussed in the class.

(Refer Slide Time: 56:22)



So, here what we have done we have studied today the bus specification used typically in an SOC. Looked at the basic serial bus for connecting devices to micro controller. And we shall learn about more about the serial buses in a next class because you have got SPI bus also there will be the bus definitions for connecting peripheral devices. So, other bus definitions in fact, particularly for the IO devices like your USB fire wire we shall look at them in the next class. Any questions? In a AMBA bus change a IP core an like fault a making it compatible with a bus do in the change or IP core. The question is that for making the ARM core AMBA bus compatible do you change need to change the IP core. We are not talking about changing the IP core what I had shown was an interface unit. So, you design an interface logic.

So, that the signals become compatible with the AMBA bus, but for the other bus we do not need the like the open core wire. Open core; it is a bus what we say it is a bus independent it is a definition of a set of protocol for bus transactions. And on the base of the protocols a designer can select the set of signals to define the actual bus requirements. Any other question ((refer time: 57:57)). So, posted write is if a write say the basic idea is that I have asked for a read operation. So, and and then I have got a write posted posted write is I have put in a write request I can have send the data and an

overlapping it with the corresponding read. So, I said that that operation is available only during read. So, if if you look at look at it slave presents a response it is a Response Phase success fail. So, it is a only available for read transfers in the sense that posted write models that is you have posted the request and then you will be bring it later on. So, kind of a Response Phase tells you that whether it is a successful or failure on it can be even done later on.

.