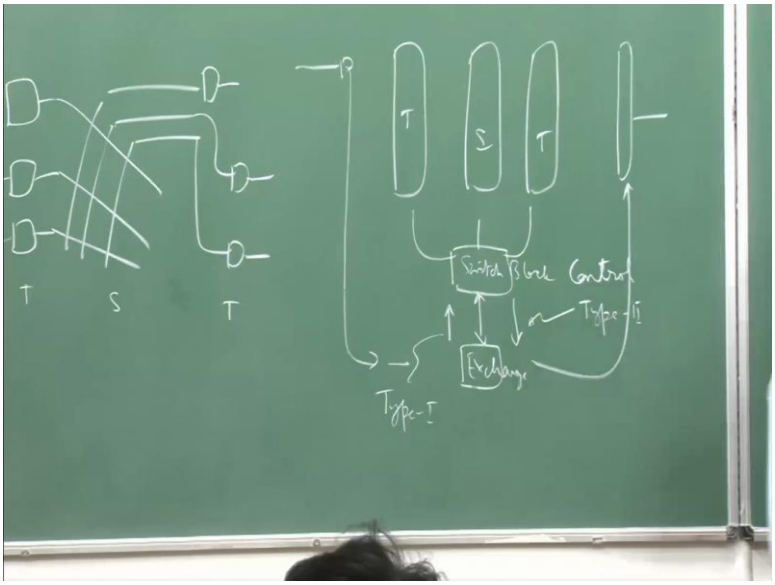
**Digital Switching**
**Prof. Y. N. Singh**
**Department of Electrical Engineering**
**Indian Institute of Technology, Kanpur**

**Lecture – 16**

(Refer Slide Time: 00:27)




Now, we will move from where we left a quick recap of what we were doing last time. We are looking at time space, time configuration, and I gave an example of this configuration. First was a time stage; space and time. This is a time multiplex space switch, and we are looking at how this will be controlled by an exchange control system. Typically, the way I told that there will be a exchange computer, which will be controlling. It will be talking to switch block control. Switch block control in turn, will actually, then control these three different stages; the time part, the space part and the time part. Now, if your switch configuration is going to be different, then this thing will be different, actually. In this switch block control has to understand the all the intricacies of the switch implementation. As far as these messages are concerned, exchange does not bother about it. It specifies only input port and output port and type of operation.

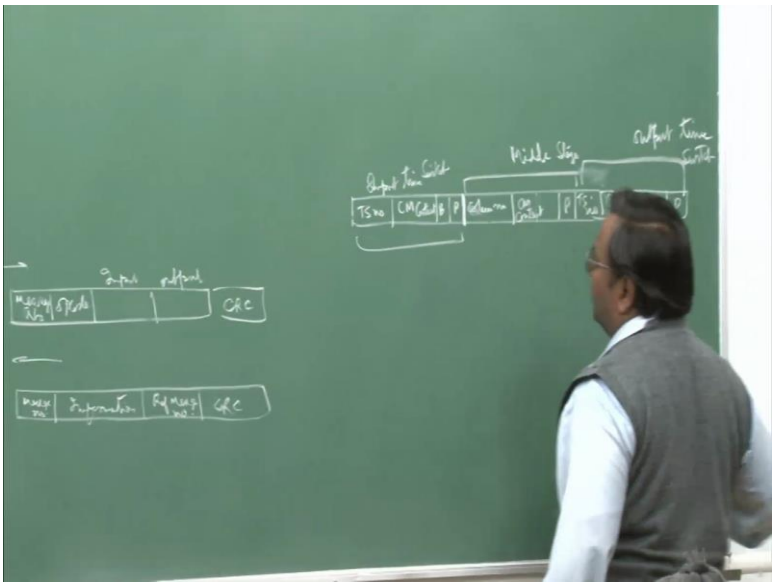
I also mentioned about six possible operations, which can be executed. Since, they can be done in both unidirectional mode, as well as bidirectional mode, in total; twelve operation modes. That will define the packet structure. The messages which are going to go in this direction are known as type 1, and whatever is going to be sent back are known as type 2 messages. I also told that this is always, a master slave configuration. Type 2

message always, has to be in response to some message, which has been sent to type 1; sent as a type 1 message from exchange to switch block control. There will always be a sequence number, given to a type 1 message. While type 2 message will come, it will have its own sequence number, but always refer to a sequence number of a type 1 message in response to whose, that particular packet; the response, this type 2 message has been sent. There can be multiple type 2 messages sent back in response to a type 1 message; that is possible.

Since, I am not talking about anything, which is tended; this is highly vender specific, because you need not, these components are not built by separate; they cannot be, need not be built by separate companies. Telecom operator will not buy this from separate companies and integrate. He will by it as whole switch itself. So, as far as this tended, which is there by which, there will be interface cards there, which will be typing all the information, sending it to the exchange control.

Similarly, there will be interface cards on this side, which will be pumping the information back, and going over a signaling channel. So, these patterns are usually, the standards; they are governed by this standard. So, that if you buy an exchange from some other company, and when the control signaling comes all the way through a signaling channel to this particular exchange thing, it has to be in a standard format, and some kind of a common language has to be there. That is where the standardization has to come into picture. Wherever, two different vendors' equipments have to interoperate, not when it is from a single vendor. So, usually, this has never become a sender; I am only giving you a sample thing.

(Refer Slide Time: 04:12)



Then of course, I came up with the type 1 message format, which was typically, as; I think it was a sequence number, which was their first one.

Student: Message number

Message number, yes. In this technically, message sequence number, then we had of course.

Student: Opcode.

Operation code; this looks like an instruction set. A instruction set will usually, never have a sequence number of instruction. Whenever you look into computer organization, they never have message number, but here you are doing, because instructions technically, are going on a serial line. This will be a serial line; serial communication (( )). This can be parallel, serial; it does not matter. As far as the instruction is concerned, obstruction for us is a serial communication; it can be over parallel bus also. So, you will always, start with this bit, and keep on transmitting to leverage to the end; framing, deframing is always is there with this is not, I am not discussing, but this can be implemented with the distances, quite large; where the frame begins and the where the frame ends. Usually, this can be of variable length; that is important, but I think most of the instruction, which I am showing is of fixed length duration; fix length, actually, in this case. There will be input time slot; there will be output time slot, as I have explained.

This format will depend on the structure there, and then of course, ultimately the CRC. So, this is all the forward direction; the reverse direction was the response thing, which is a variable length thing; this fixed one is variable, because there is no other variable length field here. So, this one usually, will be its own message number. There will be an information field and this can be variable, actually. When you send it over the data link control, that that has to be taken care of by framing.

Then there is a reference message number, this is the message number for; the type 1 message known the response, which has been generated, and then of course, you will have a CRC. Now, let us come to what happens; what is type 3 message. Type 3 messages are not communicated over serial lines; that is one important thing. It is sent over parallel lines, is basically, it is not like a packet, actually. So, this structure typically, will consist of; I am just going to draw the packet, this frame structure, but this is like each bit will be a separate physical line over which, something will be sent. This will be split and sent over to the switch's TST.

Usually, there will be; you have now, three components, three stages. So, there has to be component for this; there has to be component for this, and there has to be component for this. So, the component for the first time stage will be this one; we call it input time switch. So, I have to specify which time switch number; it can be multiple of them. For example, this configuration has 0, 1 and 2. I am not writing the address of the control memory where, I am going to write, is very important. So, address of the control memory where, I write, will be decided by the counter output. So, that will be taken care of in that fashion. That is the reason why, remember, I have used this thing as write cyclic and read cyclic; here, it is write cyclic and read cyclic. So, all the three memory locations are going to be exactly same; they will be written at the same time.

This is what the actual reason why it was done. Then, you will put that control memory content. You will have two more fields. I think, this I will expand it as that; one is busy, and other one is parity; there is no CRC. So, parity is of whatever, you have written here; that is the parity; parity over all these conflicts will be done. B is the status; you want to reserve something; you say set this particular value to busy. Now, it is free; only two statuses will be always maintained. So, if we want to release something, it will be set to free, when you are querying something, this value will comeback from whatever is there in the control. You can use the same bus for writing as well as for reading.

Then, you will have the second one, because there are multiple columns which will be there; 1, 2, 3; you have to specify a location, control memory Id, and then of course, what is going to go in that. There is a control memory; this is known as column number; control memory Id, column number. So, this is one block. This is the middle stage, now. Depending on the stage, this hard wired stuff will actually be built. This will have a c m content. So, remember, when I have drawn the circuit for a time switch, I have drawn a bus over which, what is going to be injected in the control memory, is being coming. So, that will be coming from these parallel wires or these parallel wires. Then of course you will have a parity bit.

There is nothing like a busy or free start in this case. Just put the value; its either connection is going to be on or off. The control memory content will tell, whether in which, particular slot, which particular switch point has to be on. Only one of the three will be on, actually in a column. That only gives the memory values. There even, if you put a memory value, it may be free it or it may be busy occupied; both ways. There will be always some control value. So, best is, when you are going to release certain channel; do not worry about writing into control memory; just simply, set the value to be free. Next time you can do it. You can actually, even make a local data structure and maintain this value, or keep on fetching it from the switch. It was for that purpose, and then there is something similar, will be there again, for the output time. We will have p s number.

Again, you will have same content, and you will have a busy bit, and you have a parity bit; similar format, because again, the control memories are here. What you are going to do is this control memory content tells; thus, for this time switch number in whichever, time slot you are putting; writing this, because in every time slot, the address, which you are going write, is going to change. That is coming by the counter, which is there in the frame, and of course, this will tell where, you are going to write.

Student: What is going to?

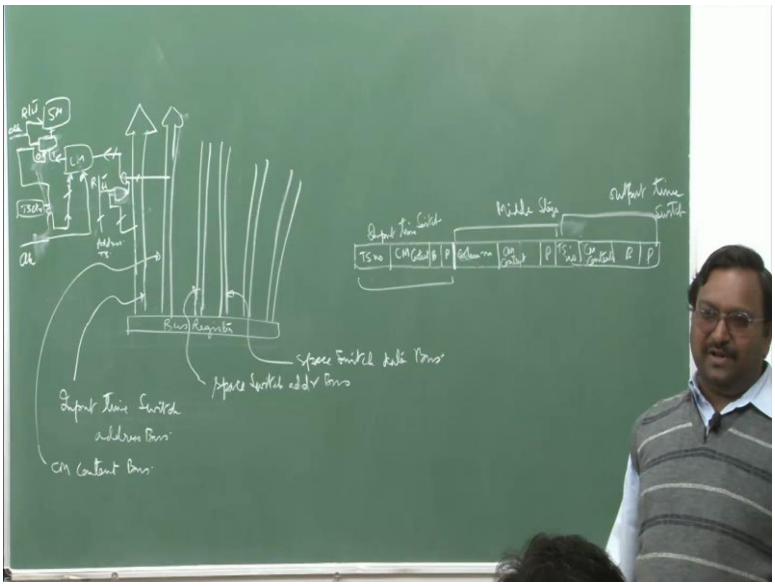
Prof: This will tell from where, you are going to read. See that this information has to be inserted in control memory, depending on a switching functionality. This is a control memory content. That is why, it is c m; control memory content. This tells in that switch what is going to be written in that control; in that particular time slot, where you are writing this.

.

Student: These are in stages sir in input stage is…

Now, I am coming to that particular thing, that whole diagram, which is there. So, typically, the circuit for the type 3; this is type 3 message we call it, but this is never sent over a serial line.

(Refer Slide Time: 13:24)



Usually there will be a bus register, which will be driving this particular thing into the bus. This is input going time switch address bus. I have to write it; I will write it. Horizontally, I think that is better; I should not write this. When I am drawing two parallel lines, this is a bus, actually. Input plain switch, and there is a control memory content. Now, for a switch which will be there, literally you will have some control memory, they will be having and then there is a time slot counter.

Now, I am talking about; this not a switch memory; it is a control memory. In turn, it will be now; the output will be fetched to the switch memory. Same counter will be actually, putting up the address there for whatever, is the cyclic. For example, in case of input stage, it is a cyclic read, is write cyclic. So, for read operation, this time start counter will be invoking this, and this control memory will be going out. When you are going to do a cyclic write into the switch memory, this will be directly, giving the address. At the same time, you can also write here, because both are attending, cyclically. In this case, when

you are actually reading, cyclically, you are also going to write at the same time, because counter will be feeding the address for cyclic read in this time memory, in this switch for switch memory. That time, there is nothing being used from control memory; that time, you can write into control memory.

So, whenever, time slot counter's value output is being used as an address for the switch memory, at the same time, you are going to write into control memory, because when you are reading from control memory, you cannot write at that same time, because this is one single data bus, which is coming out. So, remember, the clock cycle will be positive or negative. If the clock cycle is positive and you are using time slot counter value to write into a speech memory; the same address will also be used for writing into control memory, if the cyclic write is there. That depends on which I have; what is cyclic and what is acyclic; basic base, depending on that combination, you have to use it.

For this one, I can just tell; the remaining thing will come for the other one. So, let me draw this switch memory part here. This counter is, my node does not have these things; so, I am drawing it, separately. So, this is the output, which is going to come; time slot counter value will come; a clock which will drive this; clock which will put it; it is read-write bar. Because I am looking at the input size; input side time switch; this is going to write in an acyclic manner or cyclic manner.

Student: Cyclic.

Cyclic manner.

So, it is a right cyclic. Wherever, this 0 is there, this is going to write, and I have to take my counter from time slot counter. So, this is 0; this has to be activated; 1, this has to be activated, and this is what is going to be fed as an address. The same clock, I am going to use here. When I am writing cyclically, at the same time, I can also write into control memory. So, this time slot counter will be fed. I do not require an addressing, because somebody actually asked me at that time; why I am actually using a separate address bus here, for writing. We are not using that. So, this will remain active; I do not require this selector; that will be the modification. I do not require this selector, because either you are going to read the address or you are going to write, and both the addresses, which will be used, are here. So, in this case, I do not require this, actually. So, counter will be always providing the address for read as well as write; both operations. Because I am

reading acyclic, here; write cyclic and read acyclic operation. I might have reversed. Input time switch is write cyclic and read acyclic, right. Yes, it is ok. At this moment, I have to now, insert the clock; clock has to be inverted, actually in time figure, which is again, read-write bar, or I have to put read bar write. When I put this, I have to be inverting this clock.

Student: That did you see that at one what it is providing that is read of reaction that is it is read a cyclic sir.

It is read acyclic.

Student: It is. So, control memory is giving when it is on that one the selected to switch memory.

Yes. So, when clock is one.

Student: When I write is that that time slot form to should be said straight away put to control memory with this 0.

Yes, I am putting time slot counter all the time, now.

Student: 0 here, sir, when 0 is.

Because I am writing also on the same address, I am writing in that as per the time slot. I am also reading as per the time slot. So, I do not require actually, selector, here. So, that will be the modification, which will be come into the circuit. Only thing is that whether you are writing or reading; this has to be taken care of. So, when you are reading here, from the switch memory, that time, you should also read from here. So, it has to be there; hence, to be not getting here in this case.

Student: That means, we are reading and writing from same divisions, time slot.

Yes, same time, but it cannot happen simultaneously.

Student: Half.

Half cycle, you will be doing. You can, when you are reading from here, you are not writing into that.

Student: This is write cyclic, slightly, then read cyclic. One will say this is out dated; this is write cyclic.

This is a write cyclic.

Student: This is a write cyclic.

Write cyclic and read acyclic; input time switch is; let me write down this. I think, because what happens, the diagram is not there. This is a write cyclic and read acyclic, and this has to be write acyclic and read cyclic. Then, you get all the addresses are going to be as per the, same, actually, at any point of time, when they will be activated. So, there is going to be a change in this circuit, because of this, because I am all the time now, feeding address, only from time slot count, for reading and writing; both purposes. Because the clock where, it is going to be 1, when it is going; clock is going to be 0, you are writing into this. At that time, when clock is 0 here, you are also writing into this, but when you are reading from this; because it is an acyclic read; I am also reading from this control memory; I cannot write into this, and this is going to work. So, input data only has to be. Then, there is no selector is required. This data will come from this particular bus.

Now, what this address bus is going to do; there are multiple time switches. Those have to be selected. So, when you are, this data is going to come from here, but there will be an ending operation, depending on what is the switch Id; 0, 1, 2, 3; I took this address will be mapped on to that particular Id. So, there is going to be some logic here, which will convert it to a, which will map, if you compare basically, this thing with, I will call it (( )) data.

Student: This will be the actual what we use

Yes. There will be what we call address of this TS. So, if this address and this address, both match; it will become 1, and then data will be written in that place. The important thing is that in one single time slot, I cannot write in all three control momeries, simultaneously. There are three control memories; they are all feeding from the same bus. When I am writing the control memory here, other two control memories; I cannot write; that is the only problem here. They are not independently addressed.

Student: In this AND gate the fast address is coming from that.

This one; this is the configured address. For this, it will be 0. For this, it will be 1. For this, it will be 2.

Student: There are two control lines what we say.

Which one?

Student: This one, this data.

That one is coming from this address first. See, I can now, address 0, 1, 2, from this bus register; from this switch block control. Switch block control can write in any one of the three control memories. There are three control memories here. For each one of these time switches, there will be three memories.

Student: Hence, the second input, how many concept.

Second is data. Data will be again, now, similar thing; 3 times, it will be repeated. For this case, there are 16 input time switches; it will be repeating 16 times. So, all data will be coming from here; everybody will be getting, but only one of them will be able to, you write only in one of them, and that will be chosen by this.

Student: That is sir, that AND gate is having two inputs.

This will have a bus; otherwise, how you will do. If it is only binary, you can only have two times just. If you repair multiple, it will be bus. This output will be only one single bit, but this is a bus; this is a bus. So, it will be repeated in same fashion. Then, what will happen is you could have similarly, two more buses. This one will contain, this is space switch address bus, and this is space switch data bus. I am actually, kind of doing multiplexing; I am not accessing all memories at the same time. So, if I have to in the same time slot, I have to write in all three memories. This requires three cycles, three frame cycles, but this is fine; control traffic is always going to be less, because I am talking about a circuit string system.

Student: There will be delay will be different you gave output

Yes, for setting up. If I ask you what is the call setup delay; this also should be accounted for, if it needs writing in three memories.

Student: Even speech will be delayed by.

No, speech will never; speech depends only on the control memory, speech memory. Setting up of a circuit; you do not part, actually, unless that complete circuit is setup. So, when you make a dialogue something, when the whole part is setup; after that only you start talking. Once you start talking, there is no delay. It is only speech memory, which is participating; control actions are not. This is only when, the path is setup or path is released. At that time, this will be operating. So, this structure is basically, for circuit switching system; not for packet switching. Packet switching has a problem, actually. I will tell once, I come to packet switching. Implementing this kind of centralized control for a switch is a very complicated thing, and this is actually, not desirable in that case. Circuit switching, its time, because you can take care of the delays; because even, if delay is there, you talk for 5 minutes, delay of hardly 20 to 30 seconds does not matter.

Packet switching; you send a packet. It takes few micro seconds to transport a packet. You cannot setup a circuit and take 2 to 3 seconds, for transmitting only, for few micro seconds; that is highly, inefficient. Then, if you come to those special structures, which will do self routing; this class benz a class network, usually, will not or benz network will not work out in this scenario, because in this case we require centralize control. Well, that has been the problem. That is why these configurations are not used. Even, if the blocking is there, you go for a banyan kind of configuration. The third one is again, very similar to this. It will have an output time switch address bus from addressing each switch; not the control memory address. Control memory address is going to come from time slot. So, somebody was asking this is actually, the way it will happen. I do not generate an address from switch block control; of the control memory address, we never can be. This is the output time switch data bus, and this whole thing together, is what is known as type 3 message.

Student: Database data control memory content it will get a is different.

Yes, right. It is a control memory content.

Student: So, now there will be.

c m content bus.

Student: Now, there will be no what this divided it is going parallely then now when we know.

But only, you can write only, in one control memory. Usually, for setting up a circuit, you need not write in all the three control memories. You write in one control memory here; one control memory here, and one control memory here; all these can be done in one single time slot, but if you have to do lot of rearrangements and everything, then it makes, you have to then compute the time. If you want to make it, for example, here, you will be non blocking switch. Mostly, you will never be implementing still a non blocking. It will be mostly rearranged on blocking, or a blocking configuration. Then, of course, there are issues, but there, it must be something more to the circuit; this is not the complete thing. So, I am going to change the scenario.

Student: Bits parity in this which I have parity and with.

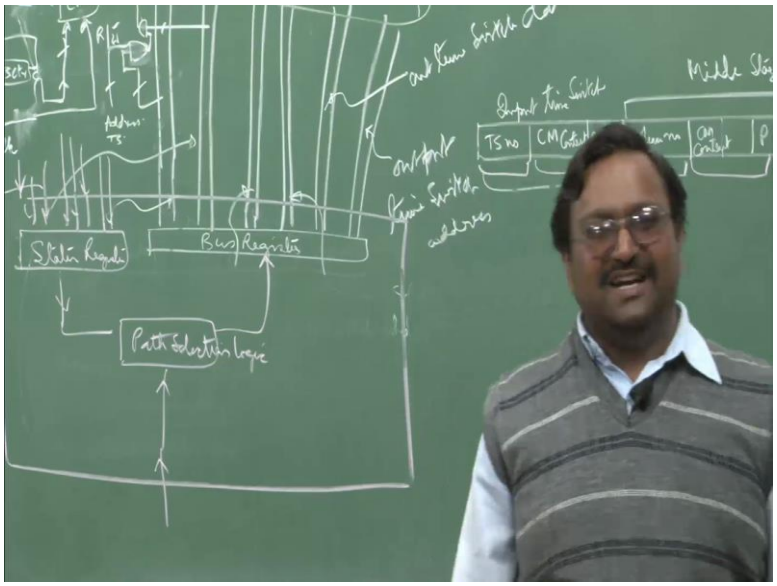
Prof: Those also go on the data bus only.

Student: But in the address bus itself, there is a control memory as per the format, you have given there.

Address bus no this is what is not address bus; this is what goes on the data bus.

This is what goes on the data bus. Similarly, here, this is what goes on the address bus; this is what goes on the data bus, address bus and for this, address bus, and this is what goes on data bus. Parities always use to check, verify, whether things are correct. If parity break happens, some wire might have broken on the bus. So, that guy also, always getting 0 or 1 depending on whether it is a basically, pull up or pull down kind of configuration. If it is connected with the register to a higher voltage, it will always become default 1. If I pull down; you will make it to 0. If it is rounded through a register, it will be pull down kind of configuration. If you leave it floating, it is pull down to 0, or pulled up to 1. Now, there is something more here in the box. I am going to just erase this text. I think you have already noted down, and draw the remaining component of the box.

(Refer Slide Time: 32:16)




Usually there will be a status register, which I call a record of whatever, is happening everywhere. So, all busy and parity, and all those bits, keep on coming from this side, from all of them, and they will be pumped here, because that is the only thing, which you require from the control memories. So, those can come to again, by putting up the wires, extra bus lines, and this will maintain a status registry here. So, there will be a separate switch block control processor. Usually, that processor can stand through the status register, or this can be maintained as a data structure in the software. This value will be taken, whenever, a message will come from here; there is a path selection logic. The path selection logic will decide what has to be written into this bus register, or what will be the type 3 message. This has to keep on changing at every 125 micro seconds.

Every slot, actually, it will change, because slot will decide where, we are going to write it. So, if you want that in these particular control memories; it is specific location you want to write; in that slot only, this bus register would have that value; this will be enabled. So, this also, needs to have a clock synchronization coming here. So, clock will also come down in this box; switch block control, because you will apply only in that particular slot. So, for every slot, this can change. In every frame, you can actually, write 125, sorry, if it is an even frame, you can write 32 variations. Actually, 30 variations technically, and those will be kept in the bus register; it will keep on flipping every time slot. So, it is also, kind of sending frame; it has to be in the frame change.

It is not kind of which I supply something; it remains there forever; no. It is changing every time slot, because you are not sending the control memory address. That is the penalty, which you pay, because you are not sending control memory from switch block control. If you can actually, bear the cost, additional cost; you can choose to actually, have time slot address, plus actually, control memory address also, in the bus. That will increment only number of bits. Say, in case of that 30 channel system, you require 5 more bus wires here, and 5 more bus wires here; that is it.

Then you will have a selected, and all that here, and this TS counter will not be used for writing into this thing, when you will do a selection. Only for read purpose, this will be used; for write purpose, you will have an address, generated from here. Advantage is that you just, can decide here, and put on the bus; you need not fetch this flop, which has to be in synchronization, as in all the three time switches; remember, all three time switches, I require frame synchronization. All three frames have to start at same time. They have to end at same time. So, that requirement is one, but by putting additional wires.

Even though, it looks simple and trivial, but it is not, actually. So, all timing has to be synchronized at all three. Usually, it will be time synchronized; otherwise, if these frames are not synchronized, you cannot do switching here. Asynchronous operation actually, is not possible in this case. So, this is switch block control, but processor; I have not shown. Here, the block of codes, are hard wired logic, which it be always used. Whatever, the type 1 message will come to pass selection logic, outcome, will be sent by it. Actually, it is a processor, software, which will do; OS software. So, these are like separate code, OS will maintain the queues that will just pass the message between the queues, and execute the processing logic for each queue of them; sort the messages. Worst part; I am not discussing here; I am just leaving it to your imagination.

So, this is your switch block control. Now, the path selection logic; what can be used here? I think whatever the strategy you tell; most likely, it is going to work. Only thing is it may not be efficient. What is the meaning of having a pass selection logic? Remember; type 1 message only tells; this is the input port; this is the outgoing port; setup a collection; it does not tell, how. When it has to put a how, you have to decide what is going to be the common slot between TS and T, and this will decide what is the control memory addresses, which have to be put. Incoming and outgoing sequence numbers;

these physical ports will anyway, fix, switch, which control memory, which control memory here, and which control memory; three control memories are identified by that port number, but the location inside have to be searched. One very simple strategy can be that let me start from 0.

Student: 21.

Prof: If the 0 time slot being used in all the three control memories or not; I am trying to find out something, which is common; free 0 in all the three. If 0 is free in all the three control memories, coming from this status register, if 0 is free; fine, I can use that based on that I will decide on my type 3 message and apply. If 0 is not free, choose 1. If 1 is not free; 2, 3, 4, 5, and so on. Then, you find something which is free. If you cannot find something is free, you say, I cannot set up the connection; it is all busy. This is known as sequential search, but starting always, from the same starting point. What is the harm in this kind of strategy, if you start using this?

Student: Same 0 will be used over and over, again I mean.

Prof: Fine, what happens is most of the powers will now be compacted towards 0; 1, 2, and 3; higher side will not be used. As a result, whenever, you are trying to make call, whatever, is the average number of calls, which are there at any point of time, you have to search at least, those many number of times; you are restly wasting your computation power. I can probably, do something still better; what could we that still better?

Student: Random.

Prof: Random; do a point task; do a random memory generation, based on that decide. There may be chances of collision, and if there is a collision again, do a random point task, till you find something, which is vacant, but you cannot be sure that all slots are busy; you will never be able to tell that actually; that is one problem. So, what you will do is, because you are not scanning at all possibilities; then how, when you will stop. Will you keep on tying it all the infinite time? Will it gets you randomly, find and call gets through? No. So, what should be the strategy, then?

Student: Can we help to alternate?

Prof: Now, that is a search state, but I must be doing random. So, you will do random search, but only certain number of times. You would say, try after ten number of times; does not happen, you say, why it cannot be done; would be loading is very high. So, that comes from the probability estimate, actually. From there, you can figure that thing out.

Student: Sir, instead of searching, we can have a list of free slots; whenever a slot is free that.

No, our slots has to be free in all three (( )).

Student: When all the three are free.

But which three pairs; that is a problem. These three pair is not free; pepple is not fixed; that depends on the actual physical input and output port.

So, the all possible combinations you are looking then. So, it will be 3 into 3 into 3; there are 27 sets which, you have to manage at any point of time; this is going to be cumbersome, actually. It is never done, because of this, and then the switch size; I am going to talk about 3 ports. When you buy, how many physical ports are there; think of that. It is going to be; it is not going to work, actually. Second possibility is that sequential search, which I had. You do a sequential search; this controller will actually, remember what was the last; where I finished my last search. So, write it 0, 1, 2, 3, 4; 4, I finished; 4 was perfect, fine and then I remember the 4. Next time, a call request come; I will start not from 0, but from 4. So, my initial value of the search, will keep on changing, which will be where, I last finished, and I keep on doing it this way, and I will do round robin. So, one of the good things, I will try to utilize all cross points, all memory locations, equally.

If the gaps get come, they will also get filled up over the time. This is one good possibility, which can be sequential search, but the starting point is not always 0; it is where, we left last time. If you do the complete search and come back to the same value, switch is busy; you simply say call cannot be made. So, this can be done within a finite duration. It is not infinite, the way it happens in random. In random, you decide certain numbers for which, you will try. See, at least, now, maybe you can say, you try contrast time, and then you can discard that is also possible. Let the guy try it again, and your call will be actually, dispersed across all the possibilities, uniformly. What happens is you

take these 3 combinations; call is blocking, but you take these three combinations; same slot might workout. So, that is another problem.

Student: To this modified such mentions and how much a b handle this how much efficient will this?

Do not know, you have to estimate it.

Student: Instead of starting from last search, why do not you start from a random point, and then start from the?

That is also a good possible, yes. Directions can be random; that is a good thing.

Student: But even then this switch as the rearrange ably non blocking. So, even if it gets block then.

Yes, if it is rearrangable blocking, then you can do the rearrangement. Then, you have to use all that rearrangement algorithm.

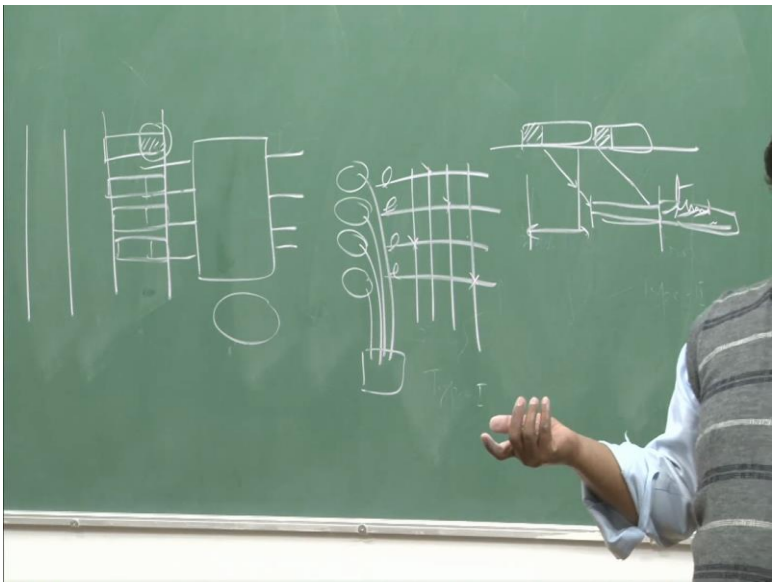
If you are not making any rearrangement, it is a blocking switch, or anything, and I can do random thing. Remember, either what we have done is when we are looking at a blocking probability using (( )) approach, as use a random search there; randomly, things are predictly, lightly, all possible combinations. Now, you can correlate what I thought there and what I am discussing today? So, that algorithm, that formula is now, executed by the processor setting in the switch block control. So, that was not something, fictitious; that was the reality, actually. The moment you change the strategy, your computation or probability of blocking calculation strategy, will also change. That was also a three stage switch. This is also a three stage switch. I can now, increase or decrease my number of slots here. If the number of slots, here and here; it is a rearrangable blocking, but it is still a blocking switch. It is not restricted on blocking. So, blocking probability; you can estimate, and blocking is not happening, because of this switch in those conditions.

So, that that is correlation between the two. I think that is where, we will end our circuit switching discussions, and I think of your own, if you want to pursue circuit switching, we have to do your own study. There, I think lot of papers, which are there, which are come out on various, different kind of switches, talking about their designs, but we are

not going to use any more circuit switching case scenario, expect for optical circuit switches, which are still, a viable thing. But, electronic circuit switching, I think this is not a thing of past, almost, but not in optical domain. Optical domain packet switching is still, very complicated and difficult to implement. Optical circuit switches are going to stay. The reason for that is you can build up reconfigurable topologies, here. So, underline physical; for example, this router is connected to the other router, I can tomorrow, choose and make somebody else, as my neighboring router. So, that topology is nothing, but a graph, which represents the connection between nodes. That itself can be modified by optical circuit switch network.

But that anyway, is not part of this course. We will discuss that an optical network course, whenever, I am going to teach that. What we will do next time. I think till this point, we will have everything coming to mid-sem. So, next lecture will not be part of the mid-sem. That will be on packet switching systems. So, how you will implement packet switching? I am just introducing here, because only that much time is left.

(Refer Slide Time: 46:30)



Packet switch will still be a box, we all know that. How to implement, is the question now? What we want in the packet switching system? What is coming at the input? Let us start from there. There will be some inputs; you all agree, and there will be some outputs. It is like your PCs is being connected into this switch. What comes in? It is not bytes; it is a frame; it is a packet, which is coming in. What typically, a packet will contain?

Special control information has to be inserted. Each packet is an independent unit. It has to be having a header, and then of course, the information which, you want to transfer. Now, each packet which will be switched inside this; it is not like that somebody is doing a signaling, and it is setting up a path, and then whatever bytes will pump, will always take that path, which is what was happening in telephone.

Here when this comes, this header will decide on which outgoing port, it will go; this header will decide. So, there is no signaling; that is one very good thing; do not require any signaling; your signaling information is going with every chunk of information. This also means, how many packets are coming per unit time; per unit second on each line; those many chunks of control information, it will be coming. That has to be processed by this processor. Circuit switching; how many control information has to be processed; number of calls, which are setup per unit time. Once you setup a call, it is through.

You are talking for two hours. There is no control information being processed, but here, I am not sending in whole packet, which is of 2 hour duration. It will be very short duration. This actually, puts a lot of load on the switch; there is no doubt, I think, this processor. Actually, it takes a lot of load, because of this, because for every packet, which is coming in; it has to get the header out; it has to analyze that and based on that decides where the switch will go; or where the packet will go, on which outgoing port. It has to do something in this box, so that this packet can come here.

How this can be done? Now, I am asking; any idea, how this will be done? You have already done circuit switching; lot of it. You understand the cross bar; you understand class, everything; can I do something with cross bar? Let us start with that question. I think, later on, I will introduce more complicated stuff. Can I do something with the cross bar? Is the 4 by 4 switch, for example, I use a 4 by 4 cross bar; can I use that thing for switching the packets? Can be done.

Student: And dealing the matting.

And most important thing, and I am going to take a simplifying assumption that all these packets are synchronized. So, packets on this port, and packets on this port, and packets on this port; all are of equal length. They always a start at the same time; they always finish at the same time; next set of four packets will come; next set of four packets will come; that is how they are coming in. Actually, this assumption is very important,

because if you are operating in this (( )), this switch will be having maximum efficiency. So, operation becomes very easy; otherwise, implementing a switch is complicated. Most of the designs are basically, based on this kind of structure. I think this is one of the reasons why, asynchronous transfer mode, ATM, actually, came into being, and that was having exactly, fixed cell lengths; 53 bytes. We build up a cross bar, depending on whatever, are the packets, which has to go. So, I have to get all these headers.

There is an interface board, which will do the processing; I have to do the processor at the central command. Based on that I will do all set ups of the circuits. Circuits will be set for exactly, one slot duration, now; cross points. Once the cross points have been set up, there has to be some delay. So, that this setup happens exactly, at one counter, because when the packet, it receives first on time; you receive this packet where, I am saying this is a time excess. You are receiving even others. You will receive the complete header by this time. It will take some time for you to process all the headers, which you have received at all the four ports. You only have to decide by this time, actually now. It will take some more time, and after this, you will be set up the cross point. So, you have to insert this much delay, at each input, and then by this time cross point is setup; packet will go out. Again, there is a synchronous session to the outside. The next packet will come. Again, all addresses will be processed.

By that time, this packet goes out. The same delay will be required by this to process, and I will remove to send the next packet. Packet switching can function. My only condition is that I should be able to process all the headers in one slot, and that actually, means number of ports, which can be there in a switch, is limited by what; processor capacity. So, you cannot keep on increasing your number of ports, unendingly; it is not possible. This becomes still more complex, because you have very high bit rate lines now. You put an optical interface; 10 GBps, and it is now can have a very high throughput; you put a lot of packets, but you have provided, this guy, the processor will not be able to process it. You need to reduce the number of ports. So, throughput is actually, or usually limited by processor capacity; not by ports. So, what I will do is I think, this is a good idea. What we will do is we have to improve on this. There is a problem, near; what? Can anybody guess? Contention.

Student: Output.

Prof: Output contention. There can be more than two packets or more than one packet, trying to go to one single output; what you will do? Only, one can be done?

Student: Yes.

One packet will go; other one has to be dropped. There is no dropping mechanism here, in this case. There is no dropping mechanism. There, what you will do is only certain cross points will be activated; this input cannot go, for example. So, that no cross point will be activated in that case. Multicasting is pretty easy; you set up all the cross points; from 1, you can go to all 4. Remaining, you do not setup and that packet is lost. So, contention cannot be handled in this case; so this blocking in that sense, actually. It can be blocking. So, you have to stagger; stagger in time. We have to stagger it in time.

Student: We can delay, but how?

We have to have a certain queuing strategy, and some strategy by which, it has to be done. So, we will try to analyze; what I can do is I can put buffers, here. So, no packet is lost; I have a memory. So, buffer is first in first out, kind of view. If I put at the input, what is the performance? If I put at the output, what is the performance? This, I think is the first fundamental question, which we will investigate. This is actually, from a paper. That paper, I will upload on the Bruhaspati; you can download that and go through that also. That gives in much more detail the whole thing.