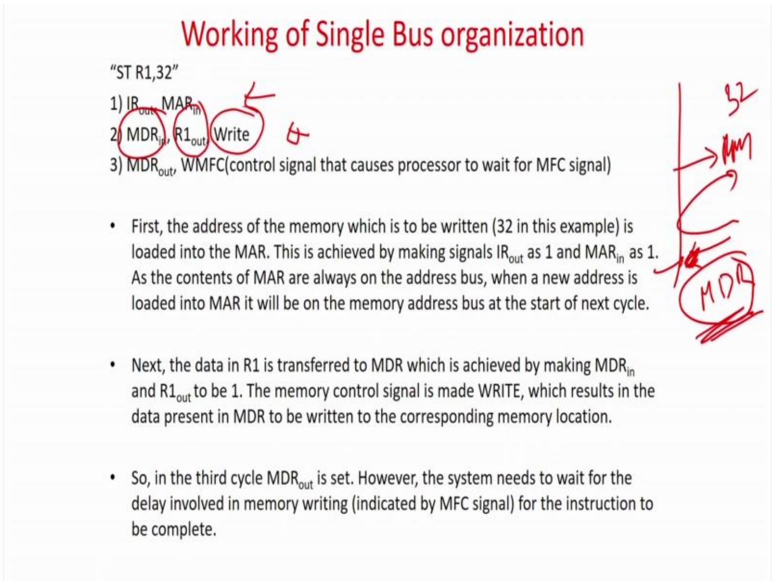
Now, what you will do now we have to read of memory data register to the register $R1$ that is you have to do this part that memory data register value will has to be dumped to register $R1$. So, only after that MFC signal has become 1, you can make the memory data register signal as out. Because before that if you see the memory data signal was a 1 over here in one that is memory data register in was a 1 that means it was reading from the memory.

This MFC signal is saying that the reading is over. So, now, you make $MDR\_out = 1$ that means now it will dump the value whatever was in the memory data register which is taken from the memory to the bus. And then R_{1in} = 1 that means, whatever was in the memory data register will dump to the register $R1$ and this instruction of $MOV\ R1$, 32 will be over.

So, if you can recollect it look at the figure in a slightly you can think it for some time then everything will be very clear to you. Then this was a move instruction. Now, let us say that there is a store instruction move means what we have done, we have taken the value of memory location 32 whatever was there, we have moved to $R1$.

(Refer Slide Time: 42:05)



Working of Single Bus organization

"ST R1,32"

1) IR_{out}, MAR_{in}
2) MDR_{in}, $R1_{out}$, Write
3) MDR_{out}, WMFC(control signal that causes processor to wait for MFC signal)

- First, the address of the memory which is to be written (32 in this example) is loaded into the MAR. This is achieved by making signals IR_{out} as 1 and MAR_{in} as 1. As the contents of MAR are always on the address bus, when a new address is loaded into MAR it will be on the memory address bus at the start of next cycle.
- Next, the data in R1 is transferred to MDR which is achieved by making MDR_{in} and $R1_{out}$ to be 1. The memory control signal is made WRITE, which results in the data present in MDR to be written to the corresponding memory location.
- So, in the third cycle MDR_{out} is set. However, the system needs to wait for the delay involved in memory writing (indicated by MFC signal) for the instruction to be complete.

Now, let us very quickly see that if this is the reverse one that is if there is something in memory register $R1$ sorry if there is some value in $R1$, we want to dump it to memory looking at 32; one was the read operation, next was the right operation very simple. Of course, first the value of $R1$ has to be written to 32. So, the register value R_{out} instruction register has to be made 1, because the default idea is that whatever instruction is there will be first in the instruction register. So, therefore, any instruction in a general thumb rule, what is there you have to first make the instruction register out that means, the value of the instruction register have to go to the memory address register.

Because whenever there is a non immediate mode of operation, so what happens like in this case the memory location is 32; that means, you have to do something with memory location 32, but where is that values 32 will be there in the very initial case it will be in the instruction register. So, generally the first micro operation always says that the instruction register out that means, you have to get the value of the memory location from the instruction register. And it has to go into the memory address register that is a very de facto standard for most of the cases. So, the value of 32 from the instruction register will be dumped to the memory instruction register.

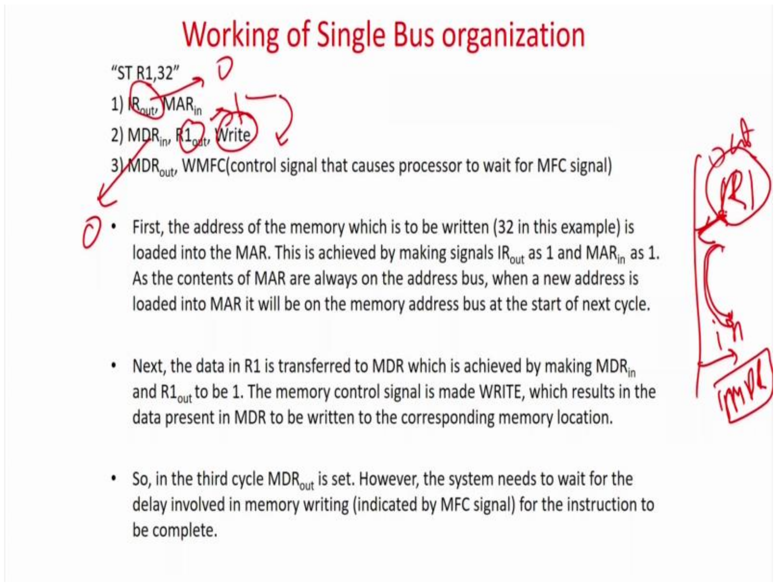
Even if you have looked in this solution also, the same read or write from the memory the first microinstruction will be more or less will be similar. So, the first microinstruction register instruction register out will dump the value to the memory register memory address register; that means, instruction register 32 value will be now dump to the memory address register. So, now, the memory address knows that value of 32 is there. Now, we have to do something with 32.

Now, in this case, what happens, you have to write; in the previous case what happened it was a read now it is basically a write; that means, whatever is in the register $R1$ has to go to the memory. So, in now in this case what happens this is the memory, this is the 32 memory location has to be read and in fact, there is a memory data register.

So, in the read mode from the memory, the memory register is to be read, but is the write operation, so you have to dump the value over here. So, in this case it will be just the reverse compared to the previous analysis. So, MDR will be actually equal to in. So, now, the memory read it has to read. So, the memory data register will be in a read mode because it will read something from the register $R1$ and that has to be dumped to the memory.

So, in the previous case memory data register in was a 1 that means what basically what happens again as I was telling you that the first microinstruction, the second microinstruction more or less looks similar in case of a read write mode. Because in any case, memory data register is to be read either it is from the memory or it is from the CPU register. If you are going for a write read operation, then the memory register will read from the memory if you are going to go for a write operation to the memory it’s going to read from the register. But anyway memory data register has to read some value which is to be transferred to the memory or it has to be transferred to the register. So, in this case, memory data register is equal to 1 and R_{out} = 1.

(Refer Slide Time: 45:12)



Working of Single Bus organization

"ST R1,32"

1) IR_{out}, MAR_{in}

2) MDR_{in}, $R1_{out}$, Write

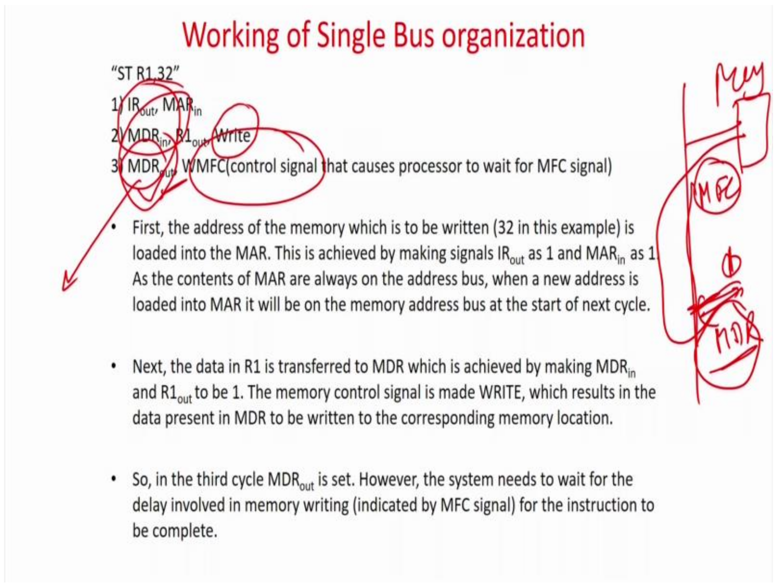
3) MDR_{out}, WMFC(control signal that causes processor to wait for MFC signal)

- First, the address of the memory which is to be written (32 in this example) is loaded into the MAR. This is achieved by making signals IR_{out} as 1 and MAR_{in} as 1. As the contents of MAR are always on the address bus, when a new address is loaded into MAR it will be on the memory address bus at the start of next cycle.
- Next, the data in R1 is transferred to MDR which is achieved by making MDR_{in} and $R1_{out}$ to be 1. The memory control signal is made WRITE, which results in the data present in MDR to be written to the corresponding memory location.
- So, in the third cycle MDR_{out} is set. However, the system needs to wait for the delay involved in memory writing (indicated by MFC signal) for the instruction to be complete.

So, in case what happens in the second case, so if you look at it, so this is your register memory data register, which is in mode, and your register $R1$ is in the out mode. So, the register will dump the value which will go to the memory data register, but this is a memory write operation. If it is a memory read operation, then actually in fact this it will not be a $R1$ in fact it will be a memory, which will dump it to the value to the memory buffer register, which will go to the memory that was the previous case.

But it now second case, what happens, memory data register is in mode and who is going to write it the the register $R1$. So, $R1$ out will be one. So, in that case, it will be going in this case. And very important we have to know one point over here, which I am going to emphasize basically first microinstruction R_{out} = 1 that is you are dumping the value of 32 to memory address register. Before going to the second microinstruction, R_{out} IR_{out} has to be made to 0. Otherwise what is going to happen otherwise there will be a conflict what the instruction register as well as $R1$ is going to write together to the CPU bus that should not happen. Before going to the value here this has to be made 0, but now in this case R_{out} is 1, this one is in 0. So, basically now R_{out} is going to write to the memory data register. And now we are giving a signal called write because the memory has to be in right mode.

(Refer Slide Time: 46:40)



Working of Single Bus organization

"ST R1,32"

1) IR_{out}, MAR_{in}
2) MDR_{in}, $R1_{out}$, Write
3) MDR_{out}, WMFC(control signal that causes processor to wait for MFC signal)

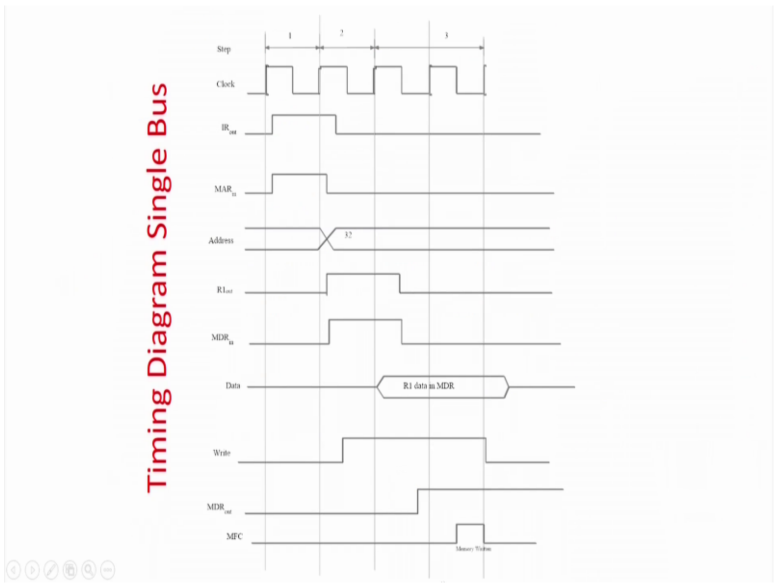
- First, the address of the memory which is to be written (32 in this example) is loaded into the MAR. This is achieved by making signals IR_{out} as 1 and MAR_{in} as 1. As the contents of MAR are always on the address bus, when a new address is loaded into MAR it will be on the memory address bus at the start of next cycle.
- Next, the data in R1 is transferred to MDR which is achieved by making MDR_{in} and $R1_{out}$ to be 1. The memory control signal is made WRITE, which results in the data present in MDR to be written to the corresponding memory location.
- So, in the third cycle MDR_{out} is set. However, the system needs to wait for the delay involved in memory writing (indicated by MFC signal) for the instruction to be complete.

Now, what happens so now your memory data register already has the value, which is in the register. So, this phase is over. So, last microinstruction basically it will let us see what it will do this is your instruction this is your bus now you are already memory the register has the value which is from $R1$ and you also you have given the write command. The write command is basically an external signal; all others are internal signals which is generated by the control unit. Now, $MDR_{out} = 1$. So, now, it will start dumping the value.

Now, of course, the memory is already in the read mode and memory data register is writing to the control bus or sorry in fact it will be the data bus which is an external bus which will be connecting the CPU to the memory and already the write signal is there. But immediately you cannot withdraw this signal memory data out has been made one. So, the memory register is writing to the control bus, but immediately you cannot make the MDR_{out}= 0 you have to wait for some amount of time. To an external bus external control bus, the memory will tell that MFC that is going to come by a external control bus to the CPU telling that my read is over or sorry yeah my read is over; that means, I have already read the value of register $R1$ from a memory location 32 now you can free your memory data register.

After that case only your memory data register output will be made 0, in fact, initially it was one. So, memory data register was dumping which was going to the memory after some amount of time when this $WFMC$ signal will be coming from the external control bus to the CPU sorry to control unit of the CPU it will relinquish the signal and MDR_{out} will now become equal to basically your 0. No, now it is free and whole value of $R1$ has been dumped to memory location 32.

(Refer Slide Time: 48:10)



So, again I will quickly look at the control sequence because it is more or less similar. So, in fact, this is the positive edge of the clock. First instruction register has to be R_{out} will be one same procedure; in the next clock edge as memory address register is already one. So, the value of 32 will be dumped in the address. From here, it is slightly changing because it is a memory write operation now R_{in}= 1. So, now from this at this clock edge basically as I have made this memory data register in one at this point at this clock edge what is going to happen is that the value of the register $R1$ will be out to the control bus. And already memory data register in have been made 1 that means, at this clock edge at this positive clock edge what is going to happen, the value of $R1$ will be going to the memory data register in.

Now, memory data register in will be fed with the value of register one, but where the memory data register will write to the memory which is in the memory address register which is the value of 32. So, $R1$ out will make the value of register $R1$ being dumped to the bus which will go to memory data register and memory data register will write the value that is what we are saying. The data whatever is in the $R1$ data has now moved to memory data register which is happening in this clock edge. And this memory register will be writing it to the memory. So, the memory write signal has been made one.

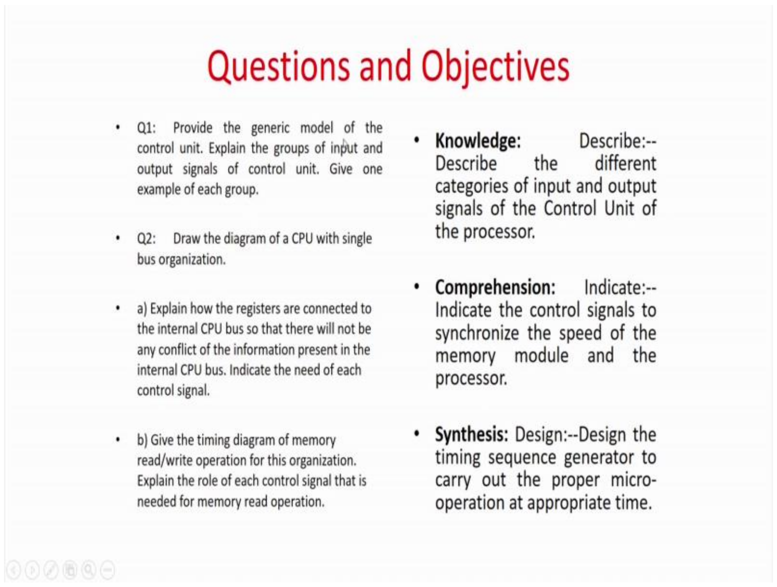
Now, at this clock edge, basically as I told you at this clock edge, the data of $R1$ has been moved to the memory data register write signal has been enabled. So, now, I am making the MDR = 1 that means, now the memory data register is going to write to the memory. So, I am making the memory data register from input mode to output mode which I am making it 1. So, in fact, at this clock edge if you compare this is the rising clock edge after this signal after the MDR signal equal to out, this is the clock edge. At this clock edge, your memory will read the value from the memory data register, because memory data register out signal = one and we are in a write mode. So, at this clock edge, this is the positive clock edge your memory is going to read the data from the memory data register.

But again immediately we cannot make MDR_{out} = 0, we have to wait for some amount of time when the MFC signal will be say memory has been written. So, after this clock edge basically you can again I have not shown this, but again you can in this point you can make this signal down over here. This signal can be made down at this point, because there is no point of keeping the value over here. So, simply the memory has been written.

So, that brings us to the end of this unit we have taken a single bus architecture and we have taken some very simple instructions which are macro instructions. We have broken down into the microinstructions, and we have seen what are the basic microinstructions or what are the control signals generated, and how data transfers operates happens between one register to another register, one memory to another memory and via register. And also we have seen that if you read to a memory, if you write to the memory what are basically controls signal involved.

So in fact, in a nutshell, today we have got some idea using a very concrete digital fundamentals how basically a control unit works in terms of signals. So, before we close down as we are all discuss about some of the questions and try to understand how the objectives are satisfied. So, let us take some few examples like provide a generic model of the control unit.

(Refer Slide Time: 51:22)



Questions and Objectives

- Q1: Provide the generic model of the control unit. Explain the groups of input and output signals of control unit. Give one example of each group.
- Q2: Draw the diagram of a CPU with single bus organization.
- a) Explain how the registers are connected to the internal CPU bus so that there will not be any conflict of the information present in the internal CPU bus. Indicate the need of each control signal.
- b) Give the timing diagram of memory read/write operation for this organization. Explain the role of each control signal that is needed for memory read operation.

- **Knowledge:** Describe:-- Describe the different categories of input and output signals of the Control Unit of the processor.
- **Comprehension:** Indicate:-- Indicate the control signals to synchronize the speed of the memory module and the processor.
- **Synthesis:** Design:--Design the timing sequence generator to carry out the proper micro-operation at appropriate time.

Explain the groups of input output of these signals and give some examples, of course, which will satisfy your knowledge objective like different categories of input and output signals as a comprehension. You should be able to design it as of sorry the comprehensive objective will be able to indicate the control signals to synchronize the speed etcetera. So, if you are going if you are able to answer the first question that what are the different signals these two objectives are satisfied. If I ask you to draw a CPU with a single bus organization, of course, this will again satisfy the next two objectives.

Then in this question 2 we have some parts we are saying that how registers are connected to the CPU and how basically signals move over there, and then we see that how they are synchronized in time. So, if you are able to design a single bus CPU, then we explain how data is moving from register to register, register to memory, and how they are all synchronizing with time as we have seen that everything happens at the positive edge of a clock. So, even if I have changed the signals like R_{in} out, memory buffer register is going to in, but they will all take effect in the next coming edge.

So, if you are able to design that of course, you are going to satisfy this synthesis objective that is design the timing sequence generator to carry out the proper micro operations at the proper time. So, in fact, if the four questions you are able to answer, you are going to you are actually meeting all the objectives. With this, we come to this end of this unit; and from next unit onwards, we will be looking more into the depth of how control signals are generated if there is multiple buses, what are the changes expected and so forth.

Thank you.