

(Refer Slide Time: 53:05)

Three BUS configuration: Instruction Execution

5. WMFC
In the fifth control step we wait for memory to respond i.e., by signal WMFC. Once MFC is high the value of memory to be read has been loaded into the MDR.

In single bus case the control signals are **WMFC**
In this case also, there is no difference compared to the single bus arrangement.

So, now the memory address register will have the value of M. Now we have to wait for some amount of time till the memory is ready, then the value will come to memory data register in fact, that was also similar for the single bus architecture.

(Refer Slide Time: 53:15)

Three BUS configuration: Instruction Execution

6. MDR_{outB} , $Reset_{outA}$, $R1_{in}$, $Select=1$, Add

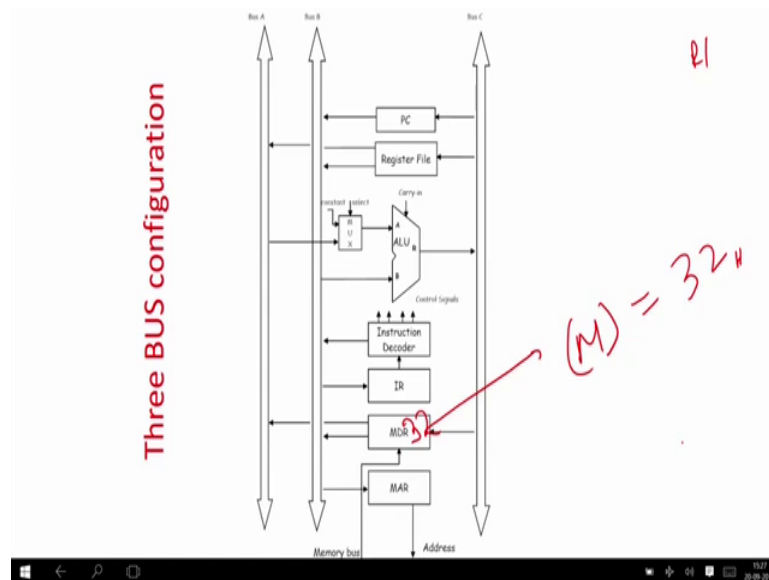
In this step the value present in the MDR (i.e., the operand) is loaded into register R1. This is achieved by an indirect scheme as there is no direct connection between output of MDR (or any register) to the input of any register via a single bus.

If the output of any register R2 is to be given as input to another register R1 say, we

- Output the value of R2 to Bus B using $R2_{outB}$
- Send zero in the Bus A by $Reset_{outA}$. $Reset_{outA}$ outputs the value of a reset register (i.e., reserved register in register file whose all bits are 0) to Bus A.
- $Select=1$ makes the ALU take the first operand from Bus A
- Add makes the ALU to add operands of Bus A with Bus B i.e., $0 + Bus\ B$. So $R2_{outB}$.
- $Reset_{outA}$, $Select=1$, Add---enables the value of Bus A i.e., R2 output to reach the Bus C via the ALU output in an indirect way where value of Bus B is added with 0.
- $R1_{in}$ loads the value of ALU output i.e., content of R2 to R1.

Now, something interesting is going to happen. So, now, let me again go to the architecture and take it from there, here we have to pay attention. Now, what now your MDR is having the value of M sorry.

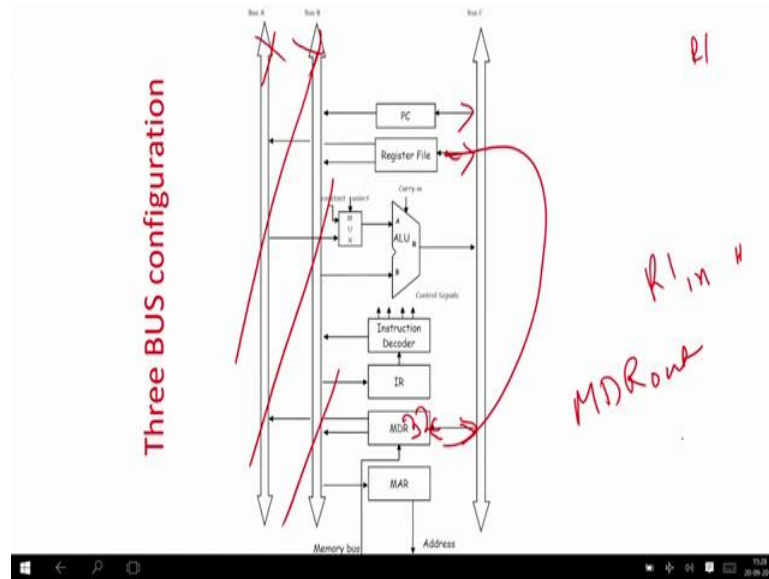
(Refer Slide Time: 53:27)



Let us assume that the memory location M has the value of 32 let us assume this. So, now, the MDR has the value of 32. Now we have to write it to basically register R₁. So, this step is more simpler in a single bus architecture, let us assume that we have a single bus architecture here and we have only C, let us assume that only C is available and 32 is available over here.

So, these two buses are not there. So, in this case what will happen as I already told you in this case it should be a two bus bidirectional bus, because anyway these things are not available data there is only a single bus and of course, everything will be a bidirectional port and of course, you can take this, but anyway at this point of time we do not require the ALU. So, I am not drawing it because the two inputs will again come from bus C only.

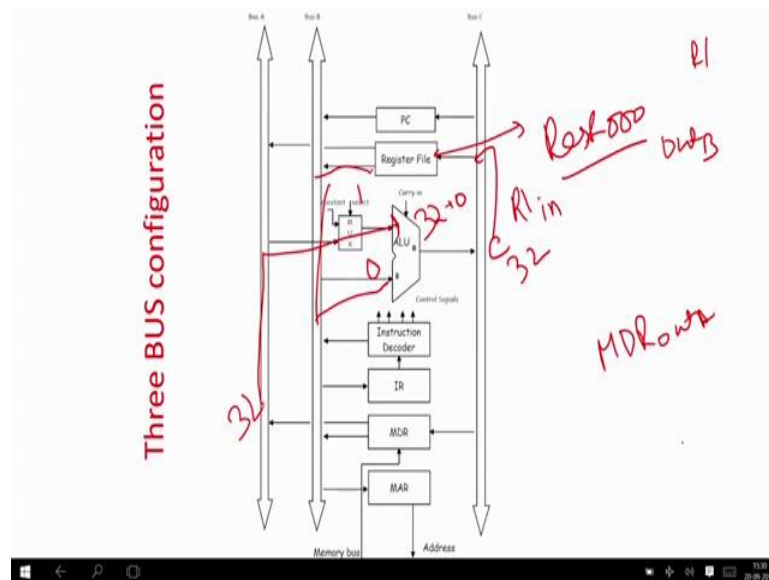
(Refer Slide Time: 54:22)



Now, this 32 has to go to some register file R_1 , in this case it will be very simple it will be MDR_{out} and it will go to R_{1in} . So, this value will go here it is simpler single bus architecture as simple as that, but in three bus architecture in this stage it's a more completed way of solving the problem. As you can see the MDR is going to dump the value to bus A and C by that way any register is going to dump the value of where value in A and B, and they are reading it through another bus called C. That means, the MDR either through this bus or this bus has somehow to route it from here to here and sorry route it from here to the register file, that is not going to be a very easy task because we do not in this configuration we do not have any connection from bus A, B and C. So, how you do it? So, here we do it in a roundabout way.

So, what we do? This 32 sorry this 32 let us assume that I dump it over here that I can do. So, and then or to make its write and then what I do? I take it from here and I can connect it here, that is simple then I dump the value of MDR at 32, that will be actually MDR_{out} and you are going to MDR_{outA} . So, it will be MDR_{outA} so; that means, the value of MDR is now at bus A.

(Refer Slide Time: 55:34)



Now, you may $select = 1$. So, that it goes to one port of the ALU. Other thing what you are going to do we will be put all 0's over here, there will a special register actually there is a register here which is called reset register and nothing at but all zero's, that one we are going to take as out B. So, that one this zero register value will be out over B and it will actually connect over here.

So, now, we are going to have 32 plus 0 which value is C over here. Now that C basically is nothing, but 32 and this one then you can feed it here at R_1 . So, it's a roundabout process to get the value from any register to any register. So, dumping value for one register to another register in the three bus architecture, in topology we are considering is a slightly roundabout way.

Of course you can change and make many changes in the architecture and you can have some multiplexing value as over here. So, it can be connected to it all these things you can do, but for this architecture given in place it has to go in a roundabout way. You have to take the value of one register, dump it in one port of the ALU; other port you have to set all zero's output will be given by the ALU to bus C and bus C will go to the some corresponding register. So, it's a longer way, let us see this one and then you can see that in this case we will have a higher number of stages.

(Refer Slide Time: 56:57)

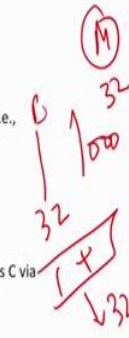
Three BUS configuration: Instruction Execution

MDR_{outB} Reset_{outA} R1_{in} Select=1 Add

In this step the value present in the MDR (i.e., the operand) is loaded into register R1. This is achieved by an indirect scheme as there is no direct connection between output of MDR (or any register) to the input of any register via a single bus.

If the output of any register R2 is to be given as input to another register R1 say, we

- Output the value of R2 to Bus B using R2_{outB}
- Send zero in the Bus A by Reset_{outA}. Reset_{outA} outputs the value of a reset register (i.e., reserved register in register file whose all bits are 0) to Bus A.
- Select=1 makes the ALU take the first operand from Bus A
- Add makes the ALU to add operands of Bus A with Bus B i.e., 0 + Bus B. So R2_{outB}.
- Reset_{outA} Select=1, Add—enables the value of Bus A i.e., R2 output to reach the Bus C via the ALU output in an indirect way where value of Bus B is added with 0.
- R1_{in} loads the value of ALU output i.e., content of R2 to R1.



So, we are here. So, memory data register that is your M, the value of N is 32 over here that is available over here.

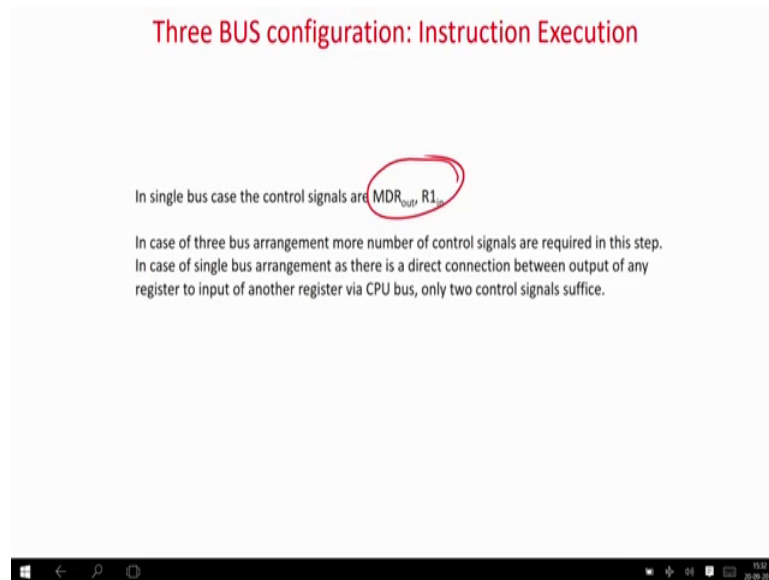
Now, what they are doing this that is going through bus B reset A through output A. In this one reset M is all zero's. So, now, the bus B is having the value 32, bus A and we are setting select equal to 1; that means, we are bypassing the constant. So, it is all zero's they are coming to the ALU and output is basically add. So, output is nothing but 32 plus 0 that is equal to 32, and you are going to dump it to register R_1. So, these are the signals which we have to do it, we can do it in one step that is not a problem, but more number of signals will be required, had it been a single bus architecture you could have just written MDR out equal to R in.

So in fact, same in single step only you can be able to do it that is not a problem. So, number of steps is not saved in this instruction. So, if some instruction involved transform from one register to another plus in this case, you do not save any kind of a step, but also you do not lose to single bus architecture, but only thing is that you require slightly more number of control signals. So, this is one peculiar instruction which we have shown, in which case basically we do not save on the steps.

But whenever we have taken this in this case you do not save any steps, but whenever you have taken this one, we can say that we have saved some steps because we do not require any kind of a temporary registers. So, and be the you try we you can yourself try with different type of instructions on this bus architecture, and you are definitely going to find out that, in most and

most of the cases the three bus architecture is going to give you much less time cycles that is obvious because more parallel buses and of course, you do not require any kind of temporary registers.

(Refer Slide Time: 58:31)



So, there are more advantages whenever you are going to higher bus architecture, but one stray example I have shown you where you lose in the number of time steps, but you also do not gain in the number of time steps. But in that more complexity is there in the terms of number of control signals and bus management.

So, I have given you two different stray examples, but you yourself can take different type of instructions and try to see how it gets executed in a three bus architecture. As again I told you we will not going to too much depth in a multiple bus architecture, because it is very intuitive because if you know how to design a single bus architecture place components and how to execute instructions, you yourself can modify that and go for a multiple bus architecture.

So, you and then obviously, you can extend these concepts and you can also go for micro program development hard micro programmed control architecture, then you can also call micro instructions and all these concepts which you have learned for single bus architecture can be directly applied to the multiple bus architecture you can easily extend these concepts. And in fact, we are not going to cover more details than that, but we have just given you the idea how things can be extended to a multiple bus architecture in terms of a three bus architecture you can yourself take different configurations and do it.

(Refer Slide Time: 59:57)

Questions and Objectives

Q1: Draw the diagram of a CPU with ~~three~~ ^{two} bus organization. In that design explain the need of each component Also, compare with a CPU having single bus.

Q2: Consider the CPU with ~~three~~ ^{three} bus organization. Write the control steps for fetching and executing the instruction ADD R1, R2, R3 in the three bus organization. Also, compare with a CPU having single bus.

Comprehension: Describe:-
-Describe about the different internal CPU bus organizations and placement of components.

Analysis: Compare:-
Compare the performance of the processor while executing an instruction depending on the internal organization of the processor.

$R1 = R2 + R3$

So, towards the end we always go for some kind of questions. So, we see that first question is draw a CPU diagram with three bus organization, you can also go for two bus organization, four bus organization, in that design explain the need of each component and compare with a bus and in the single bus architecture and compare among yourselves.

So, of course, it will satisfy these two objectives, describe about the different internal components and also compare the performance in different multiple bus architectures. Then second question one is consider a CPU with three bus and an instruction which is saying R_1 , R_2 , R_3 add; that means, in this case $R_1 = R_2 + R_3$. Again try to see how this instruction is executed and compare it with a single bus architecture, whether you save in some number of steps or whether the similar control number of control steps are required what are the number of control signals required you can make a study among this. You can take different instruction, different bus architectures and see how it compares among themselves.

So, once you are I think by after doing this lecture, you will be able to solve these questions and you will be able to which will actually meet these two objectives which we have targeted in this unit. So, with this we come to the end of this module on control unit which covered basically, how an instruction is exactly implemented or how it exactly executes in terms of hardware control signals.

And we have gone through a large spectrum of different instruction type, different addressing modes, hardwired mode of control, micro program mode of control, different type of bus

architectures, and we have dealt in details with the internals of a control unit and how specifically instructions get executed in a very micro observation in this module. So, with this we come to the end of this module again and towards the next module will be on memory components and I/O design, which will be taken over by Professor Arnab Sarkar in few in the next series of lectures.

Thank you.