

Computer Organization and Architecture: A Pedagogical Aspect

Prof. Jatindra Kr. Deka

Dr. Santosh Biswas

Dr. Arnab Sarkar

Department of Computer Science & Engineering

Indian Institute of Technology, Guwahati

Lecture - 23

Different Internal CPU Bus Organization

Hello and welcome to the last unit of this module of control unit and here in this unit we will be discussing about different internal CPU bus organization. So, throughout this module, on control unit we have seen how we can generate different type of control signals and then we have seen that how different instructions involve what type of control signals, and then we have discussed how this control signals can be generated using hardwired control, and then we have seen how it we can do it using a micro program control unit.

But for everywhere we will basically assume that the CPU is single bus organization architecture. That is their single bus which carries the data and control signals there will be a single pair of busses. But now before we end this module, let us try to give you a very short idea that if there are multiple buses then what can be the advantages and what can be the disadvantages. Of course, one clear advantage as you can figure out if you have multiple buses to carry out data and control you require a much less control steps because many operations can be done in parallel.

So, this is a very simple which is logical argument; that if you have multiple paths then, obviously you can reach the destination much faster; because you can do things parallelly. But again also we see some stray cases we are not much advantage is there, but also at the same time you have to appreciate that it involves more cost.

So, if we have very high number of system buses not only cost of the chip design or chip cost will increase, but also at the same time controlling it and we will involve more number of circuit and over that will increase. But we are not going to get give go to a vary in depth design discussion or how different type of CPU buses, then how you can design control units for then rather we will try to give you an idea that what are the different type of control signals required if you have multiple bus architecture and what is the advantage and how things can move faster.

Based on this idea you can very easily recast if you have multiple bus architecture and then how can you recast and generate control signals for different instructions as well as also you will be able to generate circuits for control units and internal and micro program architecture.

Because you already know those concepts and here we will be show you what are the changes if you have multiple bus architecture, and then you yourself can think and make a merge and blend of it if somehow we have you have some point of time, you have to design a micro program control unit for three bus architecture or four bus architecture.

So, let us in the unit I have a very brief a broad overview, that how things change if they have different internal bus organization compared to a single bus architecture. So, in this discussion mainly we will not be considering a two bus architecture, we will rather go for a three bus architecture, which will give you a more elaborate fill that if you have multiple I was possible together what is the advantages.

(Refer Slide Time: 03:07)

Units in the Module

- Instruction Cycle and Micro-operations
- Control Signals and Timing sequence
- Control Signals for Complete Instruction execution
- Handling Different Addressing Modes
- Handling Control Transfer Instructions
- Design of Hard-wired Controlled Control Unit
- Micro-instruction and Micro-program
- Organization of Micro-programmed Controlled Control Unit
- **Different Internal CPU bus Organization**

So, as you see that is the last part of this unit, that how all these basically all these units basically covered the means whatever we covered in all these one int units, basically buses single bus architecture. Finally, in the last we will just give you an overview of three bus architecture so that you can intuitively think that how other conscience will change, if you have a multiple bus architecture.

(Refer Slide Time: 03:28)

Unit Summary

In case of multiple bus systems, as there are multiple buses that interconnect the components of the CPU —


Less number of control steps and temporary registers are required in multiple bus system compared to the single bus system.

Program Counter:

- In three bus arrangement, the PC has two ports unlike one in case of a single bus organization. In three bus arrangement the PC loads values from Bus C and writes into Bus B. In case of single bus the PC reads and writes through the CPU bus.

Memory Address Register:

- The memory address register (MAR) holds the value of the address location which is to be read or written.
- Organization of MAR in three bus organization is similar to the case of single bus organization. MAR takes input from a system (CPU) bus and feeds to the address bus of the memory. So unless there are multiple memories, reading and writing through multiple buses in case of MAR is not required.



So, again a some pedagogical sense let us try to see what is going to means we will try to summarize in this unit what we are going to expect. So, in case of multiple bus systems there are multiple buses that connects the different components of the CPU that is obvious. So, there will be not a single data an address, bus there can be multiple buses to transfer the signals. Obviously, less number of control signals and temporary register will be required if there are multiple buses as we will see that is quite obvious also, because if you have two lines you send the input and two lines you get the outputs you may not require any temporary register because there will be two direct lines which will feed the ALU.

If you consider single bus architecture if you remember then if you want to add two numbers $a + b$, but there is only one path to give the values then you have to intermediate store the value of a at some temporary register, the second operand that is b can be directly fed to the ALU through the bus and then you have to add and this is answer you also be have to be stored temporary in a register, because the bus at present is carrying the value b . Once the addition is done then you can make b free and then you can feed the answer of the $a + b$ to the to the bus.

But if you have three buses then two bus will fed $a + b$, and the third bus will just give the value we will take the value of the output that is $a + b$. So, in one go you can do $a + b = c$. It is as simple as there will be one input A , one input B that the two

buses there will be a ALU and this will be bus. So, this is A, B and C if there are three buses and, but if there is a single first one bus will dump the value of A and stored in a register. Then the same bus will now divert to the; and it will be whole degree value temporarily. Then you will do $A + B = C$ and then if in the third part third state you have to erase the value of B from that bus and dump the value of C to that bus. So, that it can be saved into the memory location.

So, three steps are required, but if you have more buses very easily you can in a single stage you can do multiple operations, which will be parallel and faster. So, that is what is going to be the basic summary of this of this unit. But now let us look at different components like program counter, memory address, registers etcetera and how they will change if there are multiple buses.

So, what is the program counter? So, program counter actually points to the current instruction, and then it will do $PC + 1$. So, that it can program counter plus increment, which will point to the next address of the instruction so. In fact, that is very simple. So, you always do $PC = PC + 1$.

So, in a single bus if you remember you require two stage; first the program counter will be dumped to the will be in the bus and in the second stage you have to do program counter plus program counter plus constant by the ALU, and that has to be stored in a temporary register. And then only in the second stage the temporary register value will be dumped to the bus, which will again go and save it to the program counter because this is single bus which does this.

(Refer Slide Time: 06:20)

Unit Summary

In case of multiple bus systems, as there are multiple buses that interconnect the components of the CPU —

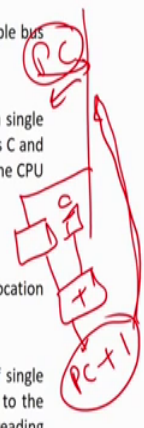
Less number of control steps and temporary registers are required in multiple bus system compared to the single bus system.

Program Counter:

- In three bus arrangement, the PC has two ports unlike one in case of a single bus organization. In three bus arrangement the PC loads values from Bus C and writes into Bus B. In case of single bus the PC reads and writes through the CPU bus.

Memory Address Register:

- The memory address register (MAR) holds the value of the address location which is to be read or written.
- Organization of MAR in three bus organization is similar to the case of single bus organization. MAR takes input from a system (CPU) bus and feeds to the address bus of the memory. So unless there are multiple memories, reading and writing through multiple buses in case of MAR is not required.



And first program counter is feeding this value here, which you are storing in a temporary register and this may be your constant and then you are going to add it by the ALU. So, this is equal to PC plus 1 then again it has to go and write back to the program counter.

But this is a single bus. So, in stage 1 the program counter value will go as an operand to the ALU, addition will be done as stored in a temporary register, which is this one is a temporary register and then in the next stage you will dump the value of PC plus 1 or the PC plus constant into this bus it will again write back to the PC or at the we have seen the constant. So, two stages are required.

But in a three or multiple bus system mainly you are keeping or discussion here in a three bus system, and we are assuming that there are two buses we will check take the output and one bus will give the input to the registers. We will see in the details, but of course, in two bus system all the registers actually have multiple ports because this is very obvious, because if you only one gateway and therefore, multiple paths then there will be no advantage. So, here different registers are different input and output port somebody has multiple output ports, that the register can give output to three parallel buses together and it can; obviously, read only so, one port, you cannot read for multiple ports together.

So, there will be multiple in a multiple output ports which we basically assume.

(Refer Slide Time: 07:33)


Unit Summary

In case of multiple bus systems, as there are multiple buses that interconnect the components of the CPU —

Less number of control steps and temporary registers are required in multiple bus system compared to the single bus system.

Program Counter:

- In three bus arrangement, the PC has two ports unlike one in case of a single bus organization. In three bus arrangement the PC loads values from Bus C and writes into Bus B. In case of single bus the PC reads and writes through the CPU bus.



Memory Address Register:

- The memory address register (MAR) holds the value of the address location which is to be read or written.
- Organization of MAR in three bus organization is similar to the case of single bus organization. MAR takes input from a system (CPU) bus and feeds to the address bus of the memory. So unless there are multiple memories, reading and writing through multiple buses in case of MAR is not required.

So in fact, it can be something like this one register can give multiple outputs to two different bus or more number of buses, and there will be also one input port which means takes the data in parallel.

So, the all these three activities can be done in together. So, one data can done in or the previous data can be sent out together and so forth. Of course, you cannot have two inputs that does not have any meaning basically. So, the program counter has two ports. So, what happens we will see later in details? So, we will see that how program counter can be made more efficient using two ports.

(Refer Slide Time: 08:05)

Unit Summary

In case of multiple bus systems, as there are multiple buses that interconnect the components of the CPU —

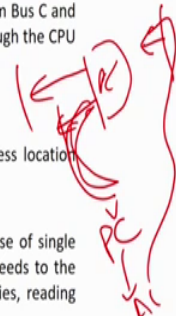
Less number of control steps and temporary registers are required in multiple bus system compared to the single bus system.

Program Counter:

- In three bus arrangement, the PC has two ports unlike one in case of a single bus organization. In three bus arrangement the PC loads values from Bus C and writes into Bus B. In case of single bus the PC reads and writes through the CPU bus.

Memory Address Register:

- The memory address register (MAR) holds the value of the address location which is to be read or written.
- Organization of MAR in three bus organization is similar to the case of single bus organization. MAR takes input from a system (CPU) bus and feeds to the address bus of the memory. So unless there are multiple memories, reading and writing through multiple buses in case of MAR is not required.



Because in one case the program counter will give the value out as this one as PC, which will be pointing to the instruction and the same time it will also who has the operand to the ALU, you will have ALU a in the ALU will do PC equal to PC plus 1, and it will did not wait that you have to erase this value and then only you can dump the value of PC equal to PC plus 1 because there are two buses here in fact, multiple three buses. So, you can directly feed the value of PC equal to PC plus 1 and the same unit of time.

So, that therefore, there will be two ports and we can actually do this in single stage we will see later in with the figure later. So, but first with the summary is that we will study how program counter efficiency can be increased, using two ports memory address register. So, this is very interesting.

So, here the memory address register in case of multiple bus as well as single bus will be similar now why? Because we have a single bus memory and memory address register what it does? It base basically tells that from which location of the memory data has to be brought in.

(Refer Slide Time: 09:03)

Unit Summary

In case of multiple bus systems, as there are multiple buses that interconnect the components of the CPU —

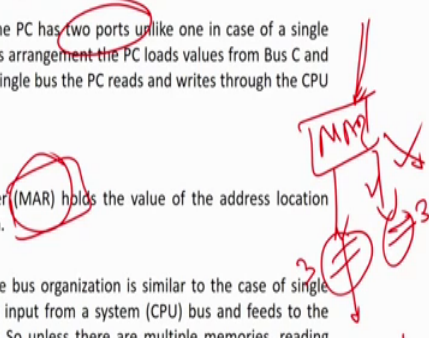
Less number of control steps and temporary registers are required in multiple bus system compared to the single bus system.

Program Counter:

- In three bus arrangement, the PC has two ports unlike one in case of a single bus organization. In three bus arrangement the PC loads values from Bus C and writes into Bus B. In case of single bus the PC reads and writes through the CPU bus.

Memory Address Register:

- The memory address register (MAR) holds the value of the address location which is to be read or written.
- Organization of MAR in three bus organization is similar to the case of single bus organization. MAR takes input from a system (CPU) bus and feeds to the address bus of the memory. So unless there are multiple memories, reading and writing through multiple buses in case of MAR is not required.



So, even if I have multiple ports or even if you have your memory address register can write to multiple outputs and similarly take data, and it can write simultaneous with the output these of course, we can do we can give the value in one go and other values it can give out.

But you see this one will have no sense here, because if there is multiple memory; then only this is going to be a very advanced then it will be a very advantageous concept. Because you can read from one from one main memory location here of course, this is a memory location value we can also need from another memory. So, of course, the memory location will be same, but you can fetch two different data from two different memory blocks and you can get to two different registers. Of course, the memory location if it is 3 over here, it is 3 over here, also in this memory block for the as there are two different independent memory data can be different.

So, you can make a program like that. So, that you can fetch from two different memory locations oh sorry from two different memory blocks you can fetch data together and use you can use it parallelly. But in this constants we will or not handling processors which we are having multiple memory blocks as well. So, we are restricting to three bus architecture, but there is single memory. So, another that case you can you need not modify your memory address register. Because the output of the memory regi address

register goes to the memory. So, if you have a single memory of course, multiple outputs like this from the memory address register will not help.

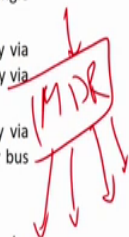
So, we will also appreciate this fact in this unit, that when memory address register has no extra advantage of having more output lines if you have a single memory. Because, but if you have multiple memories then you of course, you can feed it together and get the advantage.

(Refer Slide Time: 10:34)

Unit Summary

Memory Data Register:

- In three bus arrangement, the MDR has four ports unlike two in case of a single bus organization.
- In case of three bus arrangement, the MDR reads from Bus C or memory via "memory bus data lines". The MDR can write to Bus A or Bus B or memory via "memory bus data lines".
- In case of single bus arrangement MDR reads from CPU Bus or memory via "memory bus data lines" and writes to CPU Bus or memory via "memory bus data lines".
- Due to multiple write ports (three bus arrangement), data from MDR can be sent to more number of blocks of the CPU in a single cycle.



Instruction Register:

- Organization of IR in three bus case is similar to single bus organization i.e., reads from a bus and writes to instruction decoder. Unless multiple instructions are processed in CPU more ports are not required in IR.

Then the memory data register. Of course, therefore, many intuitively you can feed that memory register means it will take some data from the memory and of course, you distribute to some others places like, if you have an instruction called load R 1 m. So, data from the memory will be dump to register R1.

So, in this case maybe if I have multiple ports of course, it is very advantageous because you can quickly transmit this data from memory load memory R1 maybe there can be another same instruction load R2 m so. In fact, if you have multiple ports from which you can do the output. So, you can very quickly read from the memory and send value through two wires, because your multiple buses that is R 1 and R 2 can be directly feed.

So, that is very true and very obvious that because you read one from the memory and then the data can go to multiple places like different registers instruction register accumulators. So, if your memory data register this is coming from the memory of

course, this will be 1, but you we can have if it has multiple outputs. So, it can feed different lines in one group

So, in this is one point which we are going to discuss in this unit that is how memory data register, if having multiple ports what will be the advantages. So, in a three bus architecture MDR has 4 ports unlike to in the case of a single bus organization. Single bus means so, the one input from the memory and the other will be to the other and the other will be towards the different places where it wants to send the data.

And of course, in this is the context of when the memory is writing to the registers. In fact, also for reading; what happens then actually things reverses? But in an nutshell I am not going to go into that type of integrations in this unit, but in two single bus architecture from the memory to the registers and vice versa, but in case of multiple ports multiple buses, we have actually four ports four port memory. So, I have 4 ports MDR.

So; obviously, in this case it will help, because in case of three bus arrangement MDR reads from bus C or memory, that is we are assuming there is a 3 pass architecture.

(Refer Slide Time: 12:42)

Unit Summary

Memory Data Register:

- In three bus arrangement, the MDR has four ports unlike two in case of a single bus organization.
- In case of three bus arrangement, the MDR reads from Bus C or memory via "memory bus data lines". The MDR can write to Bus A or Bus B or memory via "memory bus data lines".
- In case of single bus arrangement MDR reads from CPU Bus or memory via "memory bus data lines" and writes to CPU Bus or memory via "memory bus data lines".
- Due to multiple write ports (three bus arrangement), data from MDR can be sent to more number of blocks of the CPU in a single cycle.

Instruction Register:

- Organization of IR in three bus case is similar to single bus organization i.e., reads from a bus and writes to instruction decoder. Unless multiple instructions are processed in CPU more ports are not required in IR.

Handwritten diagram on the right side of the slide shows a central box labeled 'MDR' with four arrows pointing outwards to 'W', 'R', 'A', and 'B'. Below it, there is a circle labeled 'CPU' with an arrow pointing to it. At the bottom, there is a handwritten note 'ADD R1, R2'.

So, A, B and C, A, B and C. So, generally the means organization should be following is that everybody will dump the values to A and B and we will read that is the they will be writing to A and B and they will be reading from C bus C. So, three buses will be there in the organization. So, all the registers etcetera we will write in bus A B and we will read

from bus C and of course, because it is a memory data register. So, MDR it will it can also read from the memory.

So, basically in our previous signal bus architecture, it used to read from the memory and dump to bus A; because there is only one bus. But in this case actually they are in increasing the number and then we will see how it becomes faster, because in memory data register you can read from the memory that is one extra bus C is there, that also can write to the memory data register and of course, it is going to give a output to bus A and B. So, it has multiple ports.

So, in this case it is will be faster because you can take some data from the memory or you can take some data from some bus because in this case it actually writes from the some bus to the memory, and in this case it goes from the memory to the some registers etcetera through the wires A and B. So, multiple port means you read the data from the memory, and write to two locations connected to register two buses A and B simultaneously and it will be a faster operation.

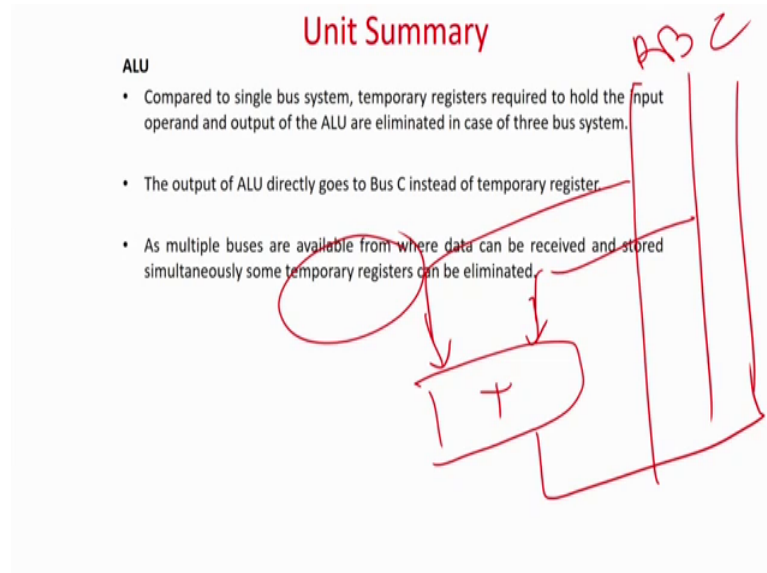
So, due to due to multiple port writes data from MDR can go to more number of blocks a single CPU cycle that is the bus word. More number of ports you read from the memory, you read from another bus called C or any other place and then dumped directly to the components connected through bus A and B. Instruction register again this is very similar to a single bus architecture, because even if you have multiple input output ports from a instruction register, it will not help you to parallelize because we are handling one instruction at a time.

But if it is a parallel processing kind of an architecture, then multiple instruction can be fetched and executed together. Like for example, if you have one instruction call say add R1, R 2 another instruction say add R3 R4 and there is no inter interdependence in between. So, you can execute add R 1, R 2 and a add R 3, R 4 in parallel. So, in that case if you have multiple instruction registers and multiple input output ports somehow it can help. But unless multiple instructions are there we are actually it is not better to have anything in the instruction register; it can be it is very similar to that of a single bus architecture.

So, here our assumptions are slightly different, it is not actually totally parallelized it is a single bus architecture like, but topology, but only the number of buses are 3 compared

to 1. Memory is also not multiple and only single instructions are handled at a time. So, so in that context we are going to discuss this unit like arithmetic and logic unit.

(Refer Slide Time: 15:37)



Of course; similar because we are not handling multiple instructions. So, ALU can do only one operation at a time.

So, a ALU just like this if you look at it, it is something like this and something like this. You cannot there is no a means point in putting multiple input output ports and because we are actually not going to handle multiple instructions, but one actually changes there, there is something interesting.

So, as I told you if you remember it is a single bus architecture then first the data comes here and gets stored in a temporary memory. Second stage this value gets directly fed over here the addition or subtraction is done, and the output is also stored in a temporary register. And then in the third phase this one will go over there this one will be erase the erase means it is nullified, and the value will travel to the bus and go to the register. But if you look at if there are multiple registers multiples words, then actually all these things all these temporary registers can be done a (Refer Time: 16:29). So, we can have three. So, this one will go over here, this one will go over here and this one will go over here BC. So, that is the only difference in the context of ALU, that there is no buffering registers or there is no temporary registers are eliminated. So, that will be the difference.

So, in summary we are going to look at three bus architecture, how control signal changes and how the different components basically requirement of different components like ALU, program counter, memory data register, memory buffer registers change in this context that is what we are going to learn in this unit.

(Refer Slide Time: 17:02)

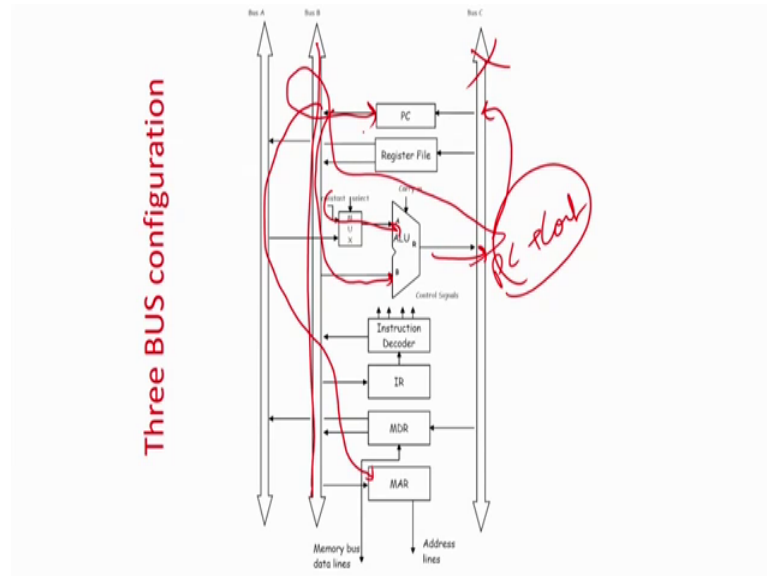
Unit Objectives

- **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.

So, one of the basic objectives of the unit, the basic objective of the unit is one is a comprehensive objective, that is we will be able to describe about the different internal CPU bus organization and placement of components. That is if I give you a single bus architecture, two bus architecture, three bus architecture, even there will be design the entire system and place the different components like ALU registers etcetera.

And then analysis is there you can compare the performance of the processor, while executing an instruction depending on the internal organization of the processor. That is whether this is the single bus architecture, two bus architecture, three bus architecture then if I am executing some instructions, what are the number of steps required, how the control signals will change, what are the advantages etcetera you will be able to compare so, that is these are the two basic objectives of this unit.

(Refer Slide Time: 17:47)



Before we start this one let us be a very careful attention on this three bus architecture. So, again I will first show you that is the three bus architecture, we have three buses A, B and C and these are the internal components like PC register, ALU decoder instruction register etcetera. But now we will try to assume here the bus A and B if you look at it, bus A and B are going to take the data from the output of the registers and blocks and bus C mainly basically is going to take the data and write it to the registers. And for ALU it is just the reverse, again I am repeat it general trend in this architecture whatever values you have to dump from program counter any other register will be dumped to bus A and B. So, mainly bus A and B are going to read the values and you are going to write the values sorry bus A and B are going to take the values from program counter register files etcetera and the registers are going to read from bus C.

And only for the arithmetic and logic it will be just the reverse, it will read from bus A and B and write to bus C and we will see why it will help. So, with this assumption we will take some time and look at this bus in elaboration. So, there are two buses A and B. So, you can just assume that for all the registers they will be writing the values from the registers to bus A and B and they have two ports. So, if you see their register file that mean R 1, R 2, R 3, R 4 and so forth. So, there are two output ports.

So, same value can be dumped at register A and. So, bus A and bus B. So, same register values or the register output will be now available in two buses simultaneously. And bus

C is generally going to give the output input to the registers. Program counter generally is writing to bus B you can also think that I am going to write to bus A, but generally as the program counter give the values only to memory address register to find out the next instruction is not advantageous or not necessary that you done the values in both A and B not necessary. But in bus C is going to give the inputs through program counter or any other register.

Now, let us look at the ALU here is slightly different. So, in the in all other cases basically the registers they are going to dump the values to A and B and read from C, but ALU is slightly reverse they will take the values from A and B and the output will dump to C this is obvious basically its a mirror image right because you are going to give one some registers will going to give the values which has to be computed upon. So, the register will write the value to this bus and this bus they are going to take the inputs and the output is going to be dump to C. So, that is very obvious that registers are connected to the inputs of the ALU and the output of the ALU is again going to writes write to the registers or the variables that is $A + B = C$.

So, what are these are the outputs A, B and this is the going to be the output with the C. So, just you are remembering as default that register files are writing register bank is writing to bus A and B which are inputs to the ALU or some other blocks and the composition output will go via C and it will go to and write the register blocks. Again this is very similar to the single bus concept mux with a select it is a constant, then program counter will be incremented, otherwise it is going to data operand from bus A. Bus second operand B is directly connected to the ALU.

Instruction register is going to take the value from bus C again for instruction register slightly the other way around it is going to take the data from instruction register from bus B instruction register is going to take the value is going to for the instruction decoder, and it is going to write the values of the instruction decoder value to bus B. So, this is the cycle over here we will later see how it is advantageous and why the connection is in this manner.

Memory data register again it will take the value from bus because they will be register is also a register. So, it will take the value from bus C and it will dump the value together to bus A and B. So, in fact, basically what happen all the registers excepting instruction

register and ALU are connected in a very similar kind of a fashion. That is the dump the values to bus A or B or both and read from bus C. ALU is just the reverse it will take the values from A and B and write to C, because $A + B = C$. So, whatever our output for the registers will become input from the ALU and what are the input for the register file is the output for the ALU that is obvious.

Just memory data register is also the same thing and the interconnectivity of the instruction register is slightly different, it takes just the value from IR from bus B and it dumps the value basically to bus B or C in fact, I mean this is in fact, I mean if you say, this is in fact, not very much required because what happens is that it generally generates this multiple signals. So, basically it reads from bus B and generally it generates multiple signals.

So, this signal basically whatever in this part is not that very relevant, then basically if you look at it the last block which is important here is the memory address register. So, sorry in memory data register remember there is one more port basically which is the input which is coming from the memory. So, memory data register and has four ports, it dumps the value to two register two writes A and B takes the input from line numbers C and also it can the bidirectional bus if you see it goes to the memory. Of course, if you remember what is the memory data register? memory data register writes to memory by this line and also reads from the memory if you reads from the memory. So, this will be the bus. So, its a four port architecture and its a bidirectional bus.

And this is something called the memory address register. So, memory address register has nothing, but it just reads the value from bus B and it the it feeds this to the memory. So, memory writes address register has only a single port, because we are assuming a single memory kind of in a block here. So, it takes the address register from one from a bus B it will take the address from where you have to read the instruction or data and this is the single address line because there is a single multiplayer block sorry single memory block.

So, they this in a nutshell shows what is the basic architecture we are following, but in fact, slight changes you can yourself do like for example, memory address register is reading from bus B. You can make it to read from bus A and drop bus B like. So, this is why some interconnection changes can easily be done over this. But in our examples will

be strictly following this three bus architecture, and then we will on this one will try to see basically what are the signals what are the number stacks etcetera required to execute the instruction. So, for the time being look at this slide for one minute and then drawing your mind that this is going to be the basic architecture and how I am going to actually execute the instruction on this three bus architecture

(Refer Slide Time: 24:36)

Three BUS configuration

Program Counter
In three bus arrangement, the PC has two ports unlike one in case of a single bus organization. In three bus arrangement the PC loads values from Bus C and writes into Bus B. In case of single bus the PC reads and writes through the CPU bus.

Memory Address Register
The memory address register (MAR) holds the value of the address location which is to be read or written. The MAR holds two kinds of addresses—(i) the address of an instruction (ii) address of operand.

Organization of MAR in three bus organization is similar to the case of single bus organization.

It may be noted that MAR takes input from a system (CPU) bus and feeds to the address bus of the memory. So unless there are multiple memories, reading and writing through multiple buses in case of MAR is not required.

So, again just read to these slides. So, what are the difference in between a single bus architecture and multiple bus architecture? Let us look at the program counter you can read the text over here. So, I will just tell you; what are the difference? I will then again we can come back. So, this is the program counter. So, what the program counter does? So, the program counters basically write the value to bus B, which can actually feed the value to the instruction register. So in fact, this program counter is writing something and it is going to the value of instruction register.

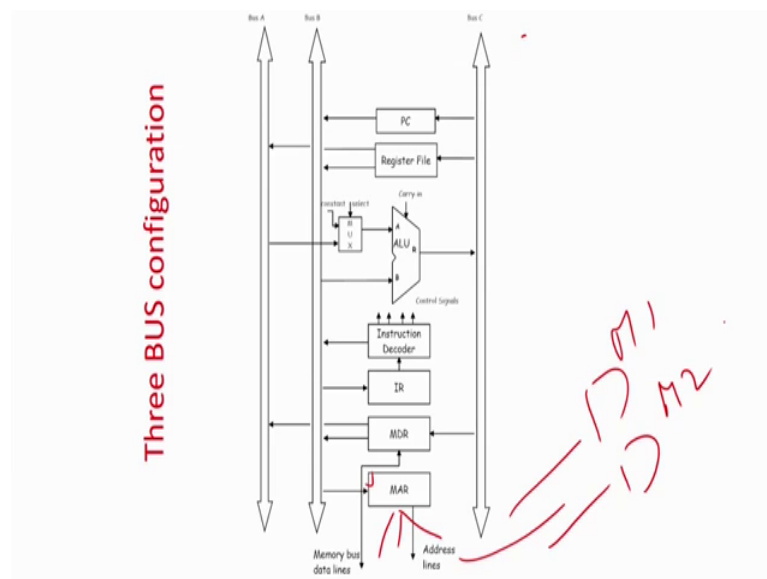
At the same time if you have to also increment the value of program counter equal to program counter plus 1. So, very easily same time you can pass this value to B and you can take this constant, and you can dump the value over here and it is can be directly going to the PC. So, single cycle you can do it. You write the value to instruction register sorry this one will go to the memory address register I am sorry this line will not be there sorry ok. So, this line will be not there PC will write to the memory address register same time you dump the value to B at the second operand from part A, you take the value of

constant and output here will be equal to PC plus constant and same line you can directly feed it to the program counter.

If you compare to single bus architecture, this part was not there basically. So, in this part is not there, in one step you can get the value of program counter to the memory address register this one you can dump it over there PC will be PC plus 1, but this line is not there. So, you have to wait till this bus is free because that is the only available bus, then only you can round it from here to here. So, that was two steps if there is the single bus architecture, but in this as there is a three bus. So, you can very easily do it in a single stage and also if you look at. So, there is a two port program counter

Next it is saying that the memory address register. So, memory address register basically if you look at it, it will take some value from bus B and write it to the memory address to the memory. Of course, single memory is there. So, we cannot have much more ports to help us.

(Refer Slide Time: 26:47)



Because if there is a multiple ports over here; so in fact, in this and they connect to different bus to different lines to the address. So, you should have multiple memory blocks memory 1, memory 2.

So, as you are assuming single memory block, they are not going to help you. So, we are having a single port two port basically mar which takes the value from memory from bus

B and directs through the address line of the memory. But in fact, you instead of bus B you can also take the input from A that you can easily do.

(Refer Slide Time: 27:15)

Three BUS configuration

Program Counter
In three bus arrangement, the PC has two ports unlike one in case of a single bus organization. In three bus arrangement the PC loads values from Bus C and writes into Bus B. In case of single bus the PC reads and writes through the CPU bus.

Memory Address Register
The memory address register (MAR) holds the value of the address location which is to be read or written. The MAR holds two kinds of addresses—(i) the address of an instruction (ii) address of operand.

Organization of MAR in three bus organization is similar to the case of single bus organization.

It may be noted that MAR takes input from a system (CPU) bus and feeds to the address bus of the memory. So unless there are multiple memories, reading and writing through multiple buses in case of MAR is not required.

So, that what is says. So, it is made in bold it says that this one is going to change, if you are assuming a 3 bus architecture. Memory address register is similar and does not require any change. So, to explain it emp to emphasize on that I have not made it bold sorry.

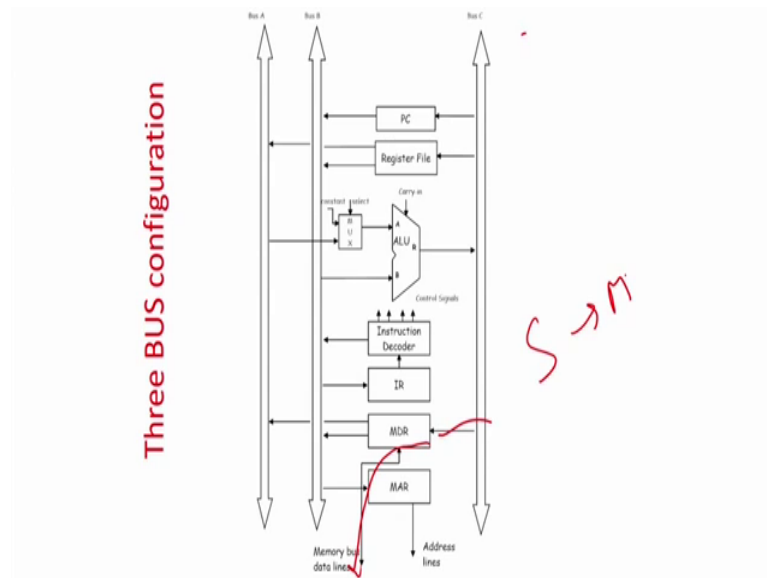
Next memory data register. So, let us look at it. So, this is a MDR Memory Data Register. So, memory data register basically I have put multiple ports over here, because if you consider a single bus architecture it take some data in and out basically it can take this is the case, if you assume that there is a single bus you have to take a bidirectional port and of course, these two buses are not there. So, what happens? It takes a it takes input output from the memory bi directionally, and also it connects to a single line. So, that is what is the architecture? So, its a two port architecture, and both the directions are bidirectional.

But in multiple port architecture, we are not giving a bidirectional bus here, but we are writing to two different lines that is the A and B. So, for the memory data or memory instruction can go to two different places. In fact, it is if it is a instruction generally directly goes to the instruction register, there is no point in having two lines, but mainly we are handling also data in the same memory component. So, if the memory data

register have any data. So, it can directly go to two different memory two different locations or two different registers or it can go to register, it can also go to an ALU or two different locations basically simultaneously because it feeds bus A and B together and it reads from bus C and basically this is the bidirectional bus.

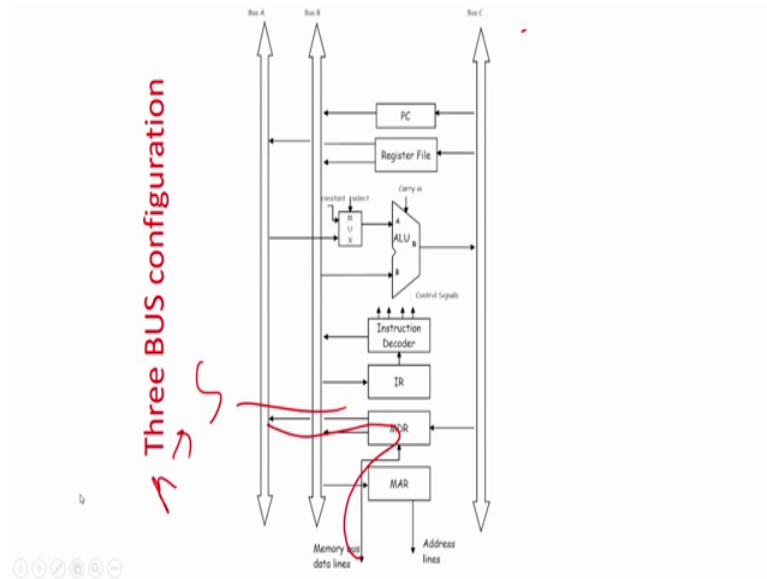
So, for example, if you want to read from the memory. So, it will go over here and then it will go to this bus and this bus. If you want to write from the memory or if you want to write to the memory basically you will take the data from here and it will go here.

(Refer Slide Time: 29:06)



That is a system to memory and then if you are telling the other way around, that is from memory to system.

(Refer Slide Time: 29:13)



So, this is your system and then from memory sorry then it will memory it will come from here and it will that is from the memory to system, system means a you are having two buses over here.

So, that is the change that if from 2 ports from it has become 4 ports and you have to observe that this is a now a unidirectional bus not a bidirectional bus because output is this direction, output is this direction input is this direction from the MDR, but it is a single bus system which we have to generally make it a bidirectional port because these two port lines are because A and B these two lines are not available, that the these two lines are not available actually.

(Refer Slide Time: 29:49)

Three BUS configuration

Memory Data Register

In three bus arrangement, the MDR has four ports unlike two in case of a single bus organization.

In case of three bus arrangement, the MDR reads from Bus C or memory via "memory bus data lines".

The MDR can write to Bus A or Bus B or memory via "memory bus data lines". In case of single bus arrangement MDR reads from CPU Bus or memory via "memory bus data lines" and writes to CPU Bus or memory via "memory bus data lines".

Due to multiple write ports (three bus arrangement), data from MDR can be sent to more number of blocks of the CPU in a single cycle.

Instruction register

Organization of IR in three bus case is similar to single bus organization i.e., reads from a bus and writes to instruction decoder. Since multiple instructions are not processed in the present CPU organization more ports are not required in IR.

So, MDR has an our fourth port block. So, if you look at it memory read register now has two ports. So, it can write two bus A or B or memory data lines and basically or the differences has been mentioned over you can read this slides and instruction register as I told you. So, basically if you look at the figure. So, instruction register we have a we have a single instruction, that is you take the data from the data means it is the instruction from the memory is taken from the memory data register via it will go this.

So, if you want to read an instruction it will come from the MDR, basically it will come from the MDR and it will go to the instruction register. This will be basically the path it will come from memory to MDR from MDR it will directly go over here. Basically and we are as you are telling that you are not in implementing multiple instruction case. So, the instruction register is very similar compared to a single bus architecture ok. So, in this case again the instruction register is very similar to the single bus architecture.

(Refer Slide Time: 30:48)

Three BUS configuration

Instruction decoder
Instruction Decoder decodes an instruction and generates the corresponding control signals.

↳ Organization of instruction decoder in three bus case is similar to single bus organization i.e., reads from IR and generates control signals.

However, it may be noted that there may be difference in the number of control signals in three bus case compared to single bus organization.

Register File
The register file comprises user programmable registers R0 to R(n-1).

In case of single bus organization, the registers read and write through the single CPU bus. However, in case of three bus organization, registers read from Bus C and writes into Bus A or Bus B.

It may be noted that due to multiple ports, data can be read from two registers through Bus A and Bus B and written to a register through Bus C in a single cycle.

Now, instruction decoder I just need not elaborate here, the instruction decoder is also a very similar stuff because you are considering a single instruction. So, single instruction will be taken in the instruction register, it will go to the decoder and signals will be generated. Of course, the internals of the instruction decoder will change because here the outputs of the instruction decoder will be different because of course, the control signals here we will not we will they have to control three buses instead in old case it was just controlling one bus.

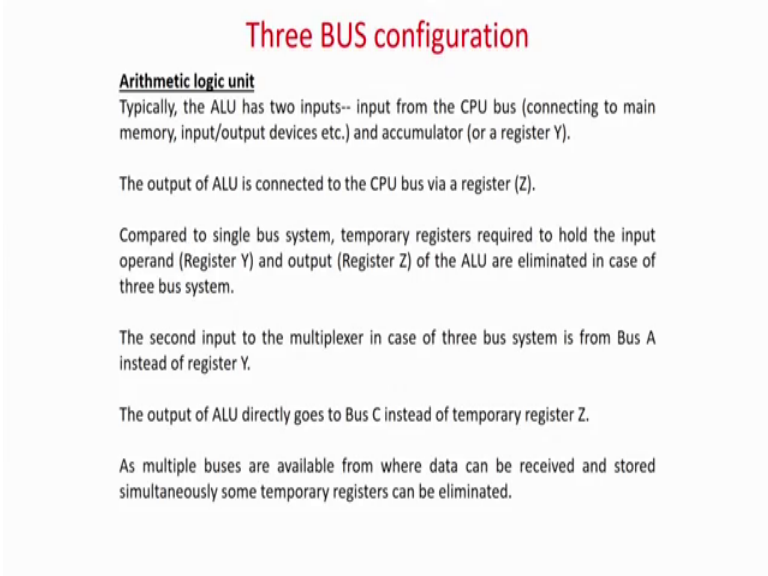
So, in that case the instruction decoder internals will change, but the input output ports or the IO behavior of the instruction decoder will be similar. Because the instruction decoder, decoder instruction and generate the corresponding control signals. So, in this case multiple number of control signals will be generated and it will feed to different parts, but basically if you look at it that if you look at this part not much we will change, because it is going to take the data from the instruction register and it will generate multiple instructions and instruction values or control signal values basically.

So in fact, the control signals of course, will be different because there are different buses. So, they will be different input output configurations, but the overall topological will look very similar. That it will take some data instruction from the instruction decoder and generate the corresponding inst corresponding control signals.

So, therefore, they say that not nothing much changes any instruction decoder of course, the register file registers have lot of changes. If you look at it again let us go to the figure sorry, if you look at the figure the register changes because in the old case there is a single bus like say bus C, and the registers file have a bidirectional bus. It reads and writes from the same bus, but in this case number of ports are increased because the registers are writing to bus A and B simultaneously. So, you can dump the values in two different locations simultaneously and it can, but and the input comes from a register.

So in fact, the number of ports has increased from 1, 2, 3 and secondly, please know bidirectional bus requirement here because in this case it reads from port this port and dumps the value the output port. So, simultaneously you can read from here and the old value can over here like a shift register kind of a thing you read from this port and write it to this valuable. So, this is this slight change in the architectural register files.

(Refer Slide Time: 33:01)



Three BUS configuration

Arithmetic logic unit
Typically, the ALU has two inputs-- input from the CPU bus (connecting to main memory, input/output devices etc.) and accumulator (or a register Y).

The output of ALU is connected to the CPU bus via a register (Z).

Compared to single bus system, temporary registers required to hold the input operand (Register Y) and output (Register Z) of the ALU are eliminated in case of three bus system.

The second input to the multiplexer in case of three bus system is from Bus A instead of register Y.

The output of ALU directly goes to Bus C instead of temporary register Z.

As multiple buses are available from where data can be received and stored simultaneously some temporary registers can be eliminated.

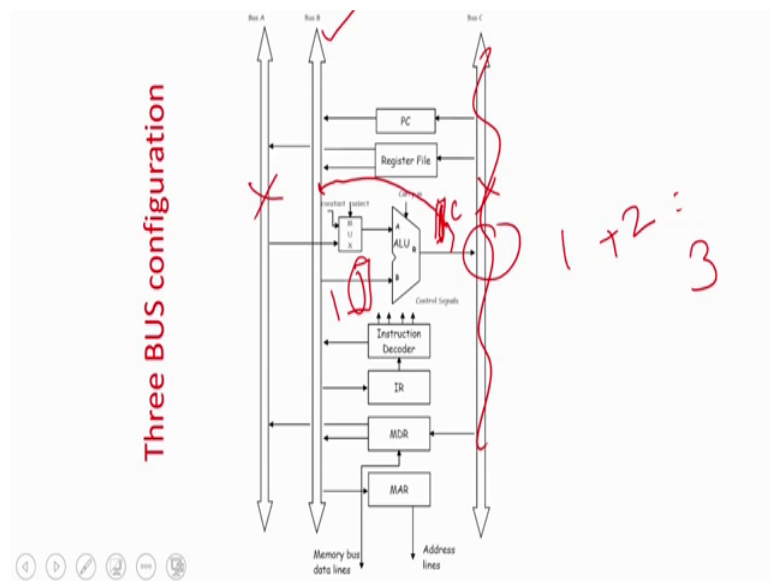
Arithmetic logic units, that is again the arithmetic logic unit is similar it comprise of adder, subtractor, multiplier, comparator or all the logical instructions there can be executed that hardwired is present, but what changes easy there is no requirement of any kind of a temporary register. You can read in this slide and then we will basically look at why there is no requirement of any this slide, why there is no requirement of any temporary registers let us again look back here.

So, if you just see a single bus architecture; in single bus architecture what happens? This is the only bus available or let us assume that B is the only bus available A and C are not there. So, what will happen? So, first some data if it has to come. So, let us assume that this bus is also not there this bus is not there. So, of course, this one will have to again go back here with some multiplexing arrangement.

So, what is going to happen? So, there is only one bus that is B. So, first operand actually we will come over here and say it will be stored in a temporary register, because there is only one bus which will carry the data. So, this is data 1 which is actually stored in a temporary register. Second one it can come over here and multiplexing I am not taking the constant. So, this one will be with the data operand two will be directly fed over here

So, first step you have to take a temporary variable, temporary register over here stored variable number one because there is only a single bus. Next is you connect this bus or from data from some register data and you can connect it to the ALU via mux done.

(Refer Slide Time: 34:32)



Then you are we are going to have operand 1 plus operand 2 the value will be equal to say 3 and taking the value 1 and 2. So, value three or operand output 3 will be ready, it will be waiting over here see there is A plus B, C or whatever the output is there.

But this bus is not available over here. So, it is waiting over here. So, weight has to be again output you have to have a temporary register that will be y we generally call the

register y. So, after this operation has been done the temporary variable or the temporary register is going to store the value of C, that is three in this case then in the third stage basically this output variable to input register temporary register C will be done to the single bus and it will go to the corresponding places and place basically. So, its a multiple stage process, but in a multiple bus architecture it is very simple. So, one line is connected from the first operand, second line is connected the second operand second bus and output is going to a third bus.

So, there is no requirement of any temporary registers as such compared to a single bus architecture so, but basic ALU structure remains same that is not much changes over here and input output ports in that terms is also similar, but only thing is there is no requirement of any temporary register. So, that is what is the in a three bus configuration, the multiple registers, temporary registers job is minimize.

So, that is why we are explaining this whole unit using a three bus architecture. If you are using a two bus architecture that is flavor is slightly gone because generally ALU means A plus B and the output has to go to C. So, three bus architecture actually makes this things very complete from 4 bus, 5 bus things are more sophisticated and not that simple to exam explained and if you are having a two bus architecture this advantages cannot be illustrated.

So, therefore, we are considering a three bus architecture for explanation multiplexer line constant need not to say. So, this multiplexer and constant need not to say. So, this multiplexer and constant if you look at it. So, this multiplexer and constant, these are same because the job is just to take an operand or the constant if you want to increase the value of program counter. So, there is no change in that. So, therefore, if you look at it. So, what changes basically? Configuration changes are program counter, memory data register, registers and you know arithmetic logic unit in terms of temporary registers. So, these changes are happening because there are more buses into picture.

(Refer Slide Time: 36:52)

Three BUS configuration: Instruction Execution

ADD R1, R2

1. PC_{out}, MARin, Read, Select=0, Add

In the first control step the value in the PC is loaded into the MAR (via Bus B) and the control signals are PC_{out} and MARin.

Make the memory control signal as READ because the contents of the memory location specified in the PC needs to be loaded into the IR (in 3rd control step after WMFC).

Also in this control step we initiate to increment PC to point to the next instruction. The ALU adds the constant with present value of PC (fed through the Bus B). Output of ALU (i.e., PC+ constant) is fed automatically to Bus C.

In single bus case the control signals are PC_{out}, MARin, Read, Select=0, Add, Zin

So it may be noted that in case of three bus system we do not require the "Zin" as the output of the ALU is directly fed into Bus C.

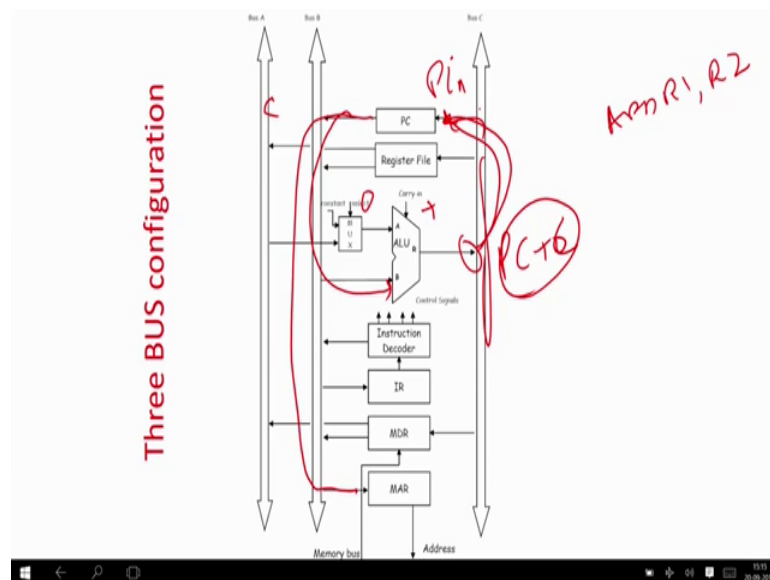
Now, we are going to take two examples; in one example will show that what are the advantages of having three buses and another case we will show that we do not get so much advantage, if you are considering a multiple bus architecture. To extremes can be two different instructions will take and show, but for most of the cases we will always going to have an advantage, because it is very obvious because if you have multiple buses things will go parallelly, but first one or two stray examples we can see where the advantage is not there in fact, you are having more hardwired, but still the number of stages are not reducing.

So, the first case we are going to take is add R 1 into R 2. So, what is the thing? So, two variables already available in R 1 and R 2 and then one you have to do it. So, first is you have to fetch the instruction. So, how do you fetch the instruction? Basically program counter output value will go to memory address register in, that is simple as or single bus architecture then you put the memory in read mode here you select 0; that means, you want to add the constant and increment program counter at add. So, what this step does? Already I have discussed so many times in some previous units, that program counter value is the address where the instruction is there that you put in the memory address register, make the memory to read mode, select is 0 that you have to add the constant and make it to add mode.

But if you look at a single bus architecture, we had another signal that is called Zin. Because the output of program counter plus constant has to store in a separate temporary register which we call it z or y and we have to wait this step is over then only you can write the value to the program counter. But in this case as you have already seen who do not require any kind of a temporary register. So, we do not have anything called Zin or Yin something like that to hold the value.

So, what happens we will go to the figure and see what happens basically? So, the program counter is going to feed to the memory address register, this is the first step which can be done directly, which are going through let me let me just erase it basically we are doing it through bus B.

(Refer Slide Time: 38:58)



So, PC out is we are going we are basically doing add R 1 and R 2.

So, program counter out is going to get MAR in to that is done then we are going to select this equal to 0, so that the constant goes over here and program counter of course, if you see we are also using it, to give it to the value here. So, this is equal to PC. So, if you can see the program counter is connected to bus B, and the bus sorry bus; A program PC value is going to bus B and we are feeding it to memory address register and also you are feeding it to the second component of the ALU.

So, therefore, output of the PC is connected to bus B and memory address register is taking the value from bus B. So, the as I was saying that slightly sometimes you have to think that if you are instead of doing this if you are taking the value from bus A that will not be a very good design, because most of the time the program counter value is taken as the input to the memory address register. So, therefore, we will whichever output bus you are dumping the value of program counter, that should be considered as a input to the memory address register.

So, even if I can take the input from here, but we are not doing it, because the pcs writing through bus B; but it can be done if you are sorry you can also be done, but if you have to you have to take the output for program counter to bus A, you have to disconnect this line and you have to disconnect this line.

So, any option you can take. So, as I told you this is the very flexible architecture you have your own design in this case we are taking this. So, the program counter value is going to come to the memory address register, same time it is going to B and you are taking select equal to 0 memory is in add mode and here is equal to PC plus constant is already there, and you can see and it is already feeding back to the program counter. So, there is no need to store anything.

So, program counter out you will select 0 memory address register in and at the same amount of time sorry and same amount of time you can say that the addition is being done. So, just you can update the value of PC through bus C. So, in a single step we are able to do it. So, this is; what is the stage PC? Program counter out memory address register in read the memory select 0 because PC plus constant and add it, there is no constant of properly temporary register to hold the value PC equal to PC plus 1.

(Refer Slide Time: 41:15)

Three BUS configuration: Instruction Execution

2. PC_{in}, WMFC
In the second control step the updated value of PC that is in Bus C is loaded into the PC; this is achieved by control signals PC_{in}.

Also, in this step we wait for memory to respond i.e., by signal WMFC. So now the MDR contains the instruction what was present in the memory location pointed by the PC (in the 1st step).

In single bus case the control signals are Zout, PC_{in}, WMFC
In case of three bus system we do not require the "Zout" as the output of the ALU is directly fed into Bus C.

3. MDR_{outB}, IR_{in}

Value present in the MDR (i.e., the instruction) is loaded into the IR.

In single bus case the control signals are MDR_{out}, IR_{in}

As input of IR is connected to Bus B so the MDR needs send its output through Bus B; hence the signal MDR_{outB}. Such an option is not present in the single bus system and MDR can connect to IR only via the CPU bus; hence the signal MDR_{out}.

Next step is simple you have to make PC in read the value of output to the PC, and where till the memory response. And if you look at the single bus architecture, these things were very similar you have to read the value of PC WmFC, but also you have a sim single instruction called Zout. Zout will go to the value of PC in, but here PC in will not require any Zout, because already bus C is kind the value of the new value of the program counter. So, again I will look back the figure and then it will be clear.

So, in this case what happens we are not having any temporary variable out which is going to the PC in, here directly you just make PC in PC in. So, whatever the value is in C will directly go over here. So, one control instruction is saved in this case and of course, WMFC same in both the case, because we have given a you want to read from some memory you have given the value to the memory address register. So, sometime you have to wait till the value comes from the memory bus and it will be dump to the memory address register. That is the instruction add R 1 and R 2 will come to memory data register after wait of some amount of time.

Then next what is going to happen? The memory register data will be out and it will go to the instruction register and this step will be very similar to the single bus architecture. So, this is your point, memory data register out instruction register in it will be also similar for a single bus architecture, because we are not handling as I told you we have a single memory and we are not havely handling any kind of multiple instructions together.

(Refer Slide Time: 42:47)

Three BUS configuration: Instruction Execution

4. $R2_{out}$ $R1_{out}$ $Select=0$, Add

The output in R2 is fed to the first operand of the ALU via Bus A by control signals $R2_{out}$.

The output in R1 is fed to the second operand of the ALU via Bus B by control signals $R1_{out}$.

The control signal to the ALU is Add.

In single bus case we require two control steps for the above operation.

$R2_{out}$ Y_{in} The value in R2 is loaded into the register Y (by control signals $R2_{out}$, Y_{in}).

$R1_{out}$ $Select=1$, Add, Z_{in}

In this step, $R1_{out}$ is enabled which results in the value of register R1 being present in the CPU bus. The CPU bus is fed as the second operand to ALU.

Select=1, which ensures that the first operand to ALU is the value at Y (i.e., the R2 content). ALU to perform addition

Control Z_{in} enables loading of the ALU output (content of Register R2 (Y) + content of Register R1) to register Z.

The diagram illustrates a three-bus CPU architecture. It features three registers: R2, R1, and Z. Register R2 is connected to a multiplexer that outputs to Bus A. Register R1 is connected to a multiplexer that outputs to Bus B. Both Bus A and Bus B are connected to an ALU. The ALU also receives a control signal 'Add'. The output of the ALU is connected to a multiplexer that outputs to Register Z. Handwritten red annotations include a circle around 'R2_out', a circle around 'R1_out', and a circle around 'Z'.

Now, let us see we have to now do real addition? So, if you look at it. So, what is the addition? So, we are assuming that the two registers R 1 and R 2 already has the value, and the instruction that is R 1 add R 1, R 2 is going to the instruction register from instruction register it goes through the instruction decoder decoding has been done, and it will have to generate the signals. So, what you have to generate? Register value R 2 register value R 1, they have to be dumped to two different buses they have to be added and the value will be out. So, very simple R 2 out R 1 out B.

So, now,; that means, what as I told you here you have to observe that basically here the signals are R 2 out a and R 2 out B, unlike it is a single bus architecture you had something called r out R 1 out R 2 out R 3 out, here we have R out A and r out B now why they are different because the register R 2 can give the value to two different ports port A and port B there are two different buses.

In this case R 2 is giving the value at A and R 2 is giving the sorry R 2 is giving the value at A and R 2 at A and R 1 at B; that means, there are two words that is A and B this is two buses basically R 2 is giving at A and R 1 is giving at B. So, simultaneously two datas are available there and these are basically if you look at the figure they are going to the two different ports of the ALU and you make select equal to 0 that is very obvious, because in this case you are not taking PC increment, but you are taking the operand 0 and you have a add.

So, you have to very nice you have to observe that we do not have to store the output; this is directly it will directly go to bus C. So, you need not have to store any temporary variable over here because we have directly two buses available which can give input to the ALU, and also the output that is equal to R 1 plus R 2 can directly go and feed bus C. So, you also do not require any kind of a temporary registers over here. So, those things are not required.

Now, so, this is the same thing we are doing, we have taking two words A and B, and dumping the values of R 2 and R 1 and making the add operation. Now if you look at a single bus architecture, it would be in slightly more complicated.

(Refer Slide Time: 45:10)

Three BUS configuration: Instruction Execution

4. $R2_{outA}, R1_{outB}, Select=0, Add$

The output in R2 is fed to the first operand of the ALU via Bus A by control signals $R2_{outA}$.

The output in R1 is fed to the second operand of the ALU via Bus B by control signals $R1_{outB}$.

The control signal to the ALU is Add.

In single bus case we require two control steps for the above operation.

$R2_{outA}, Y_{in}$ The value in R2 is loaded into the register Y (by control signals $R2_{outA}, Y_{in}$).

$R1_{outB}, Select=1, Add, Z_{in}$

In this step, $R1_{outB}$ is enabled which results in the value of register R1 being present in the CPU bus. The CPU bus is fed as the second operand to ALU.

Select=1, which ensures that the first operand to ALU is the value at Y (i.e., the R2 content). ALU to perform addition

Control Z_{in} enables loading of the ALU output (content of Register R2 (Y) + content of Register R1) to register Z.

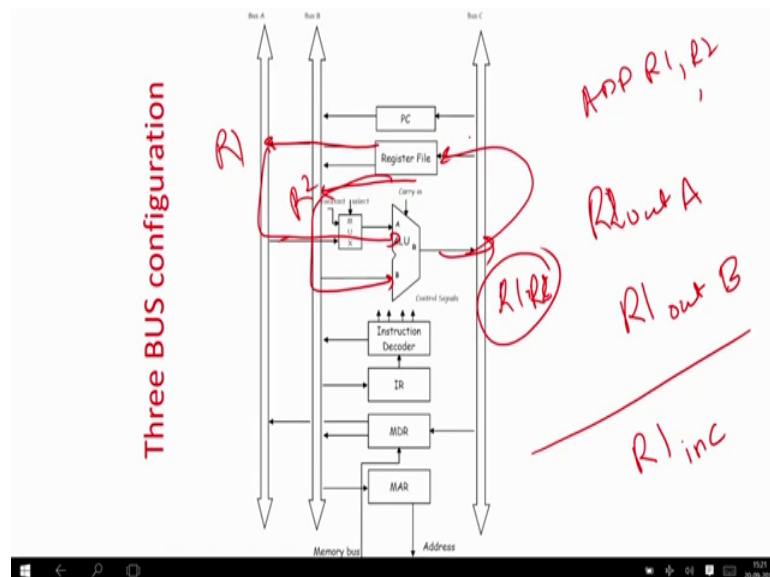
So, what we have to do? We have to take R 2 out and Y in. So, what was that? We have to take the value of R 2 and store in a temporary register that is actually equal to Y it will remember it is a temporary register Y we are now having the value of R 2 over here one step is gone.

Next step you are connecting R 1 here directly to the bus then here you are going to have the answer that is R 1 plus R 2. But then again we required to store this in a temporary register and in third stage only you can write back the value using the bus this is the single bus. So, first one you will store the value, then the second stage you do the add and you write in the temporary variable and finally, only after this one finally, you have to make sorry finally, basically the content of R 2 and content of will be stored in the

register Zin, and then this Z this temporary register Z, the third stage can only dump the value to wherever required.

But in this case if you look at there is nothing called such type of any registers, you dumped the value of R 2 and R 1 in bus A and bus B and just add it the value will be available in the memory register C. So, again let me just again look back the figure. So, what I have doing. So, may be consider this is register R 1. So, one register R 1 will dump the value here that is R 1 out A. So, that is may be saying R 1 out A or out B; that means, in which bus they are going to give the output.

(Refer Slide Time: 46:25)



So, we can also have R 1 out A, R 1 out B; that means, in that case both the A and B will have the value from the register same register that also can be done, but in this case R 1 out A may be the instruction and that will dump the value of here. So, in our example I think it was two anything is ok. So, R 1 out B. So, in this case 2 and then this is going to have the value of register R 1 this is R 2 and this is going to have the value R 2 and if you look at. So, this one is going over here, B is going over here both the operands are here and the output R 1 plus R 2 is likely available the output.

So, now the instruction one add R 1 R 2. So, very simple the value is already available at C, just write the next control instruction will be that is will be registers R 1, and you have to make in port C or in that is this value will be directly going to register R 1 in this port

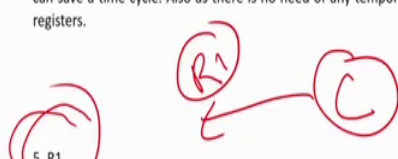
thus after this one more inst control signal required will be R 1 c and basically you are done.

So, if you look at it. So, this was the case. So, we have selected addition is being done, but in case of single bus more number of stages as we have seen more in for more number of intermediate registers and finally, you just put R 1 in because this is a single R 1.

(Refer Slide Time: 47:47)

Three BUS configuration: Instruction Execution

It may be noted that in case of three bus system as there are two output ports for the registers by buses A and B we can feed both the operands to the ALU in a single cycle. So we can save a time cycle. Also as there is no need of any temporary registers we can minimize registers.



5. $R1_{in}$

In this step the value present in the output of ALU (i.e., the result of addition) is loaded into R1 via Bus C. This is achieved by signals $R1_{in}$

In single bus case the control signals are $Z_{out}, R1_{in}$

~~So it may be noted that in case of three bus system we do not require the "Zout" as the output of the ALU is directly fed into Bus C. R1 directly obtains the value from Bus C.~~

Windows taskbar at the bottom shows the date 20-09-2017.

So, R 1 in; that means, the value of bus C will go to basically your register R 1.

But in case of a single bus architecture, you will be Z_{out} plus R in; that means, now the intermediate value was stored in Z. So, it has to be dumped to R in. In this case the value is available in already bus C and it can be given. So, if you study here we will save actually one control step for addition and of course, the overall steps will be reduced as the same time we do not require any kind of internal CPU registers explicitly some temporary registers are avoided and store the less number of control signals are generated.

So, this instruction shows a very explicit advantage of using a multiple bus architecture for most of the designs you will for most of the instructions you will find out there are advantages. You can easily try out on your own, but what I am going to show you now is one case where the advantages are not there, that is same number of instruction time or

timelines will be required of course, you will save in the number of intermediate registers that of course, will be there, but I will show you where the timeline is similar.

(Refer Slide Time: 48:52)

Three BUS configuration: Instruction Execution

Load R1, M

1. PC_{out}, MARin, Read, Select=0, Add

In the first control step the value in the PC is loaded into the MAR (via Bus B) and the control signals are PC_{out} and MARin.

Make the memory control signal as READ because the contents of the memory location specified in the PC needs to be loaded into the IR (in 3rd control step after WMFC).

Also in this control step we initiate to increment PC to point to the next instruction. The ALU adds the constant with present value of PC (fed through the Bus B). Output of ALU (i.e., PC+ constant) is fed automatically to Bus C.

In single bus case the control signals are:
PC_{out}, MARin, Read, Select=0, Add, Zin

So it may be noted that in case of three bus system we do not require the "Zin" as the output of the ALU is directly fed into Bus C.

So, what is the instruction? The instruction says that load some value from memory register memory location M to R 1. So, the first stage is very similar program counter out memory register in read select 0 add A and of course, as I can I told you there is nothing called Zin or something because in this case program counter values are available directly in the bus. So, that is the difference already we have seen this one is avoid already we have discussed because as I told you in some previous lecture, that instruction fetch concept is similar for all the instructions. So, that will be very similar.

(Refer Slide Time: 49:31)

Three-BUS configuration: Instruction Execution

2. PCin, WMFC

In the second control step the updated value of PC that is in Bus C is loaded into the PC; this is achieved by control signals PCin.

Also, in this step we wait for memory to respond i.e., by signal WMFC. So now the MDR contains the instruction what was present in the memory location pointed by the PC (in the 1st step).

In single bus case the control signals are Zout, PCin, WMFC
In case of three bus system we do not require the "Zout" as the output of the ALU is directly fed into Bus C.

3. MDR_{out}, IR_{in}

Value present in the MDR (i.e., the instruction) is loaded into the IR.

In single bus case the control signals are MDR_{out}, IR_{in}

As input of IR is connected to Bus B so the MDR needs send its output through Bus B; hence the signal MDR_{outB}. Such an option is not present in the single bus system and MDR can connect to IR only via the CPU bus; hence the signal MDR_{out}.

Similarly, you have to wait for program count in and wait till the data of the instruction in this case comes to the memory data register. Of course, there will be no Zout already discussed and finally, the memory data out B will go to R in there are slight difference here. So, generally in the when we are having a single bus architecture we have memory register out, but here we are having memory data out B that is the memory data should be giving the output to memory bus B.

So, in this case as your multiple buses. So, we generally make it explicit, otherwise because in our architecture we could have also avoided it because out memory data register if you look at it is just connected to basically single output.

(Refer Slide Time: 50:03)

Three BUS configuration: Instruction Execution

4. IR_{out}, MARin, Read

In the fourth control step the value of M (i.e., the memory location from which data is to be read) is loaded into the MAR from the IR and the control signals are IR_{out} and MARin.

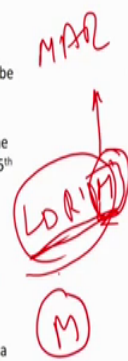
At the same control step we need to make the memory control signal as READ because the contents of the memory location specified in the IR needs to be loaded into the MDR (in 5th control step after WMFC).

There is no direct output of the IR to any of the system buses. The "address of M" part of the IR is loaded to MAR through Bus B and the output is given by instruction decoder.

The IR cannot give any output of its own and the instruction needs to be decoded before a meaningful output can be produced. However instead of making the signal as "instruction decoder_{out}", with slight abuse of notation we denote the signal as IR_{out}.

In single bus case the control signals are IR_{out}, MARin, Read

As the organization of IR and MAR are same in both the single bus and the three bus cases, the control signals are same.



So, the memory data register we are connected to both A and B. So, in this case you have to explicitly specify that where you have to were connect basically because we have to go to instruction register and instruction register is basically connected to B. So, in this case we are saying MDR out B.

So, as I told you we can it is up to you, you can also has slight changes also you can also have something like then it will be slightly different you can take the program counter to B instruction register will come over a and then MDR can overcome. So, slight there is flexibilities you can do. So, in this architecture as you have seen, the instruction register is connected to bus B. So, you have to tell MDR out B. But in a single bus architecture MDR out means MDR out right. So, we were here. So, basically now MDR out B is going to the instruction register.

Now, now it is done. So, the now the instruction that is a load R 1 M that has come to the instruction register now things will basically the different from the previous instruction we have considered. Now what? Now basically your instruction decoder has to tell the address of M and it will has again has to go to the memory address register. That is we are saying that IR out is going to the memory address register in, now it will read the data.

Here one thing we have to know that basically instruction register actually contains the whole thing, but with try it abuse of notation, we are just saying that the instruction

decoder is going basically we are saying that IR out the whole IR means the whole thing, but actually want to just look at this M. So, the instruction register decoder basically gives this value to the memory address register this part, but we are not explicitly writing it over here, because then you should have written something like instruction decoder out and that also for this part.

So, that is that can be very easily implemented, but for ease of notion similar type of notation and keep the things simple, we are just write an instruction register out a memory address register in. In fact, it is basically instruction decoder out which actually considers only the M part it. Anyway with this notation simplicity let us take it through. So, the instruction register out that is the value of M will go to the memory address register in and then you have to read the memory and you have to wait for some amount of time.

In case of a basically a single bus architecture, this would remain the same instruction register out memory in and read. Basically is again emphasized here because we are having a single memory system and at the same amount and then and at the same time basically you are also not having multiple memories, and a single instruction execution at a time. So, when you are taking a data from the memory or instruction from the memory, this type of instructions the control signals will be similar both for three bus as well as a single bus architecture right.

So, in this case the value will be read from the instruction register to the memory address register.

(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 memories ready, then the value will come to memory data register the in fact, that will be 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

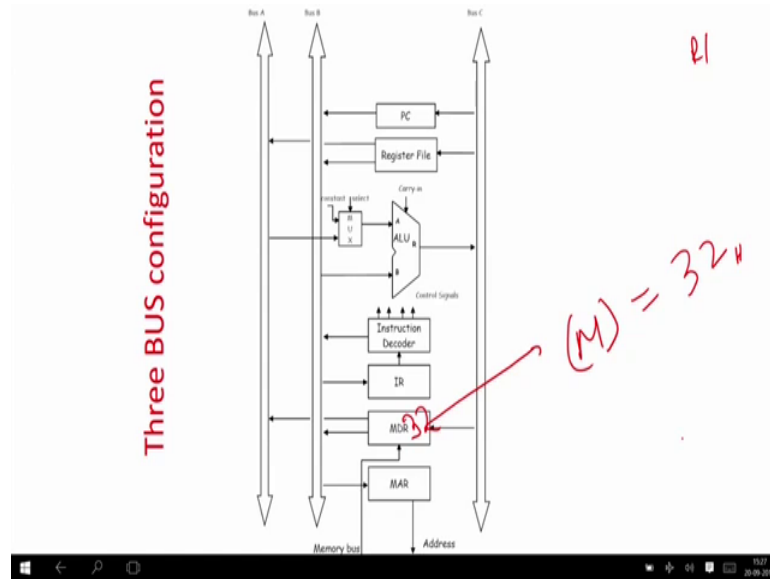
In this step the value present in the MDR (i.e., the operand) is loaded into register R1. This is achieved by an indirect scheme as there is no direct connection between output of MDR (or any register) to the input of any register via a single bus.

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.

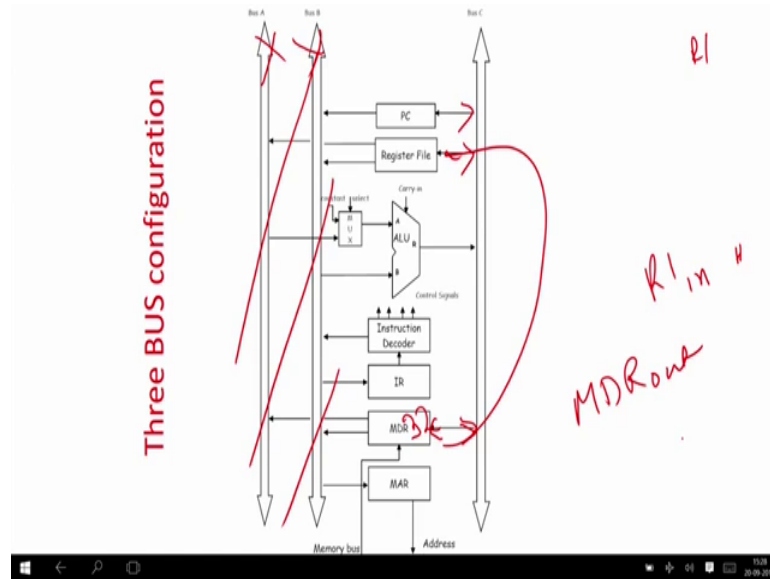
(Refer Slide Time: 53:27)



Let us assume that the memory location M has the value of 32 let us assume this. So, now, the value 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, you have to 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 this 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 these two inputs will again come from bus C only.

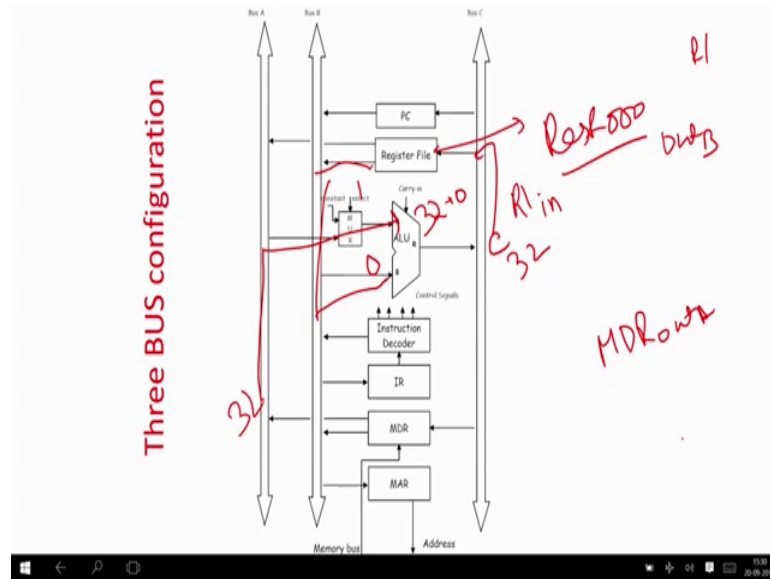
(Refer Slide Time: 54:22)



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 1 in. So, this value will go here it is simpler single bus architecture as simple as that, but in three bus architecture in this stage its a more completed 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 round it from here to here and sorry round 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 then I can. 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 simply then I dump the value of MDR at 32, that will be actually MDR out and you are going to MDR out a. So, it will be MDR out A so; that means, the value of MDR is now at bus A.

(Refer Slide Time: 55:34)



Now, you may select equal to 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 over here, there will a special register actually the register here in the which is called reset register and nothing at by it is all zeros, that one here 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, its 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 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 be all these things you can do, but for this architecture given in place it has to going 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 zeros output will be given by the ALU to bus C and bus C will go to the some corresponding register. So, its 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.

031
20-09-2017

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

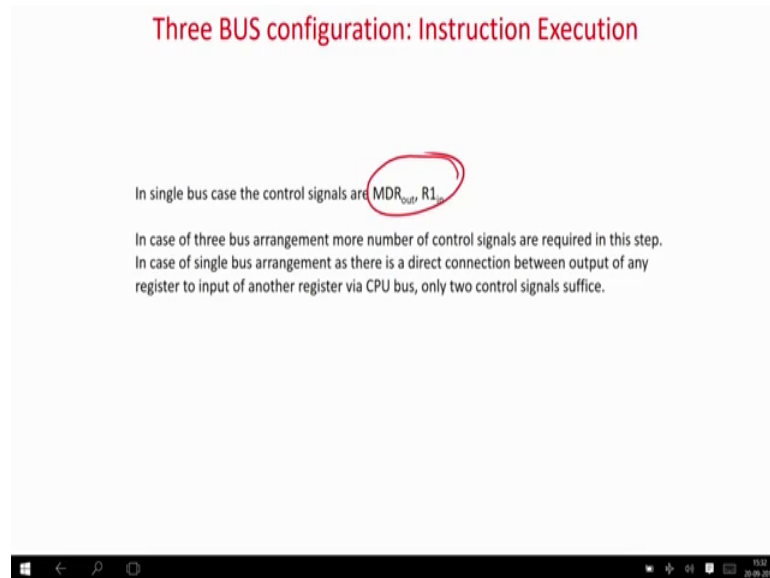
Now, what they are doing this that is going through bus B reset A through output A. In this one reset M is all zeros. So, now, the bus B is having the value 32, bus A and we are set setting select equal to 1; that means, we are bypassing the constant. So, it is all zeros they are coming to the ALU and 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 put have just written MDR out equal to R in.

So in fact, same in single step only you can be able to do it, there 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 or also you do not lose to single bus architecture, but only thing is that you required slightly more number of control signals. So, this is one peculiar instruction which we have shown, we in which case basically you 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 save it that we have save some steps because (Refer Time: 58:22) you do not require any kind of a temporary instructions. So, and belief 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, it 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 examples I have shown you where you lose in the num you do not lose exactly 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 buses 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 it gets 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 program 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

extended this 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~~ 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 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.

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 is equal to R 2 plus R 3. Again try to see how this instruction is executed and compare it is a single bus architecture, whether you save in some number of steps or when 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 portions 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, hardware 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 IO design, which will be a taken over by Professor Arnab Sarkar in few in the next series of lectures.

Thank you.