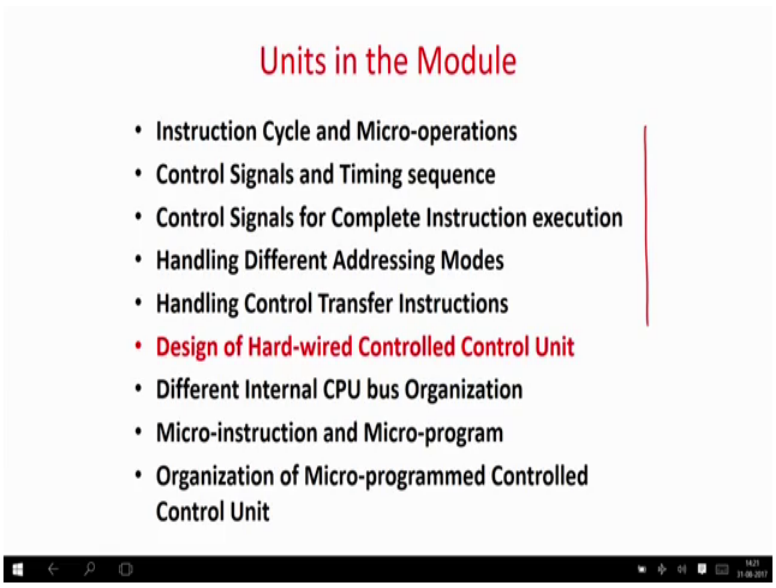
Computer Organization and Architecture: A Pedagogical Aspect
Prof. Jatindra Kr. Deka
Dr. Santosh Biswas
Dr. Arnab Sarkar
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture – 20
Design of Hardwired controlled Control Unit

Welcome to the 6th unit of the module. Here, we will be discussing mainly the design of hardware control unit if you look at the last few units.

(Refer Slide Time: 00:38)



Units in the Module

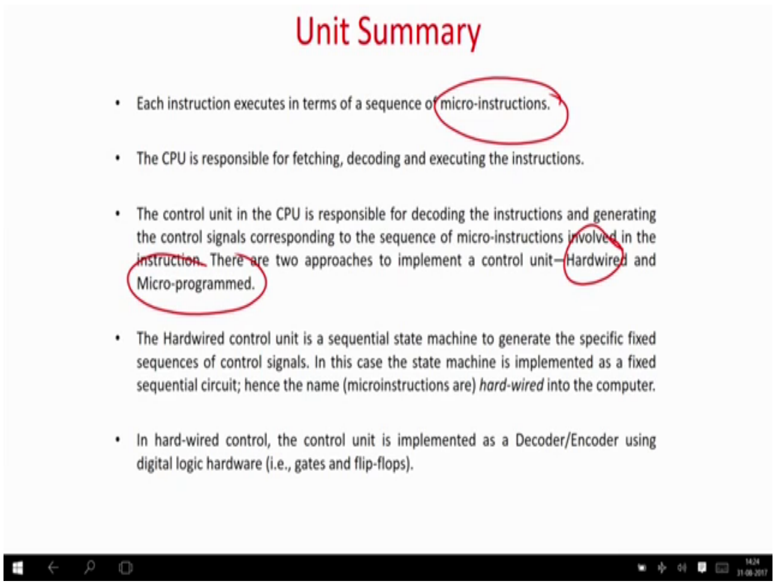
- Instruction Cycle and Micro-operations
- Control Signals and Timing sequence
- Control Signals for Complete Instruction execution
- Handling Different Addressing Modes
- Handling Control Transfer Instructions
- Design of Hard-wired Controlled Control Unit
- Different Internal CPU bus Organization
-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

We will basically discussing that basically for a macro instruction, what are the microinstruction; and for each microinstruction what are the basic control signals required; and how they can be generated; or for each given micro instructions. What are the control signals to be generated; and what is the proper sequence for that.

Then, now onwards with basically looking at how we can generate that what are the basic circuits or what are the basic techniques of; basically generating those control signals as we have already discussed in the summary of the module that basically there are two types of manner in which you can generate the control signals: one is actually the hardware which we are going to do today and another is basically called micro programmed based.

So, one is the hardware based and one is the software based. So, in this unit basically we will be focusing on the hardware based control unit in the which there will be a dedicated hardware which is hardcoded and which will generate the control sequences or control signal sequences based on the control instructions.

(Refer Slide Time: 01:31)



Unit Summary

- Each instruction executes in terms of a sequence of micro-instructions.
- The CPU is responsible for fetching, decoding and executing the instructions.
- The control unit in the CPU is responsible for decoding the instructions and generating the control signals corresponding to the sequence of micro-instructions involved in the instruction. There are two approaches to implement a control unit—Hardwired and Micro-programmed.
- The Hardwired control unit is a sequential state machine to generate the specific fixed sequences of control signals. In this case the state machine is implemented as a fixed sequential circuit; hence the name (microinstructions are) *hard-wired* into the computer.
- In hard-wired control, the control unit is implemented as a Decoder/Encoder using digital logic hardware (i.e., gates and flip-flops).

So, basically if we will first look at the unit summary; we know that basically there is a set of micro-instructions for given any set of macro instructions. Then basically what happens we have seen that for a given instruction; there is fetch decode and execute there are sequence of micro-instructions corresponding to each fetch and for each of the micro instruction there is a sequence of control signals to be generated like program counter out memory address register in set them read mode to the memory at the memory, then wait for the memory to give a signal that it is vary etcetera.

So, basically depending on each of the cycles that is we have seen there is some steps like in step 1, 2, 3 basically corresponds to a face of the instruction after then it basically corresponds to the execution of an instruction.

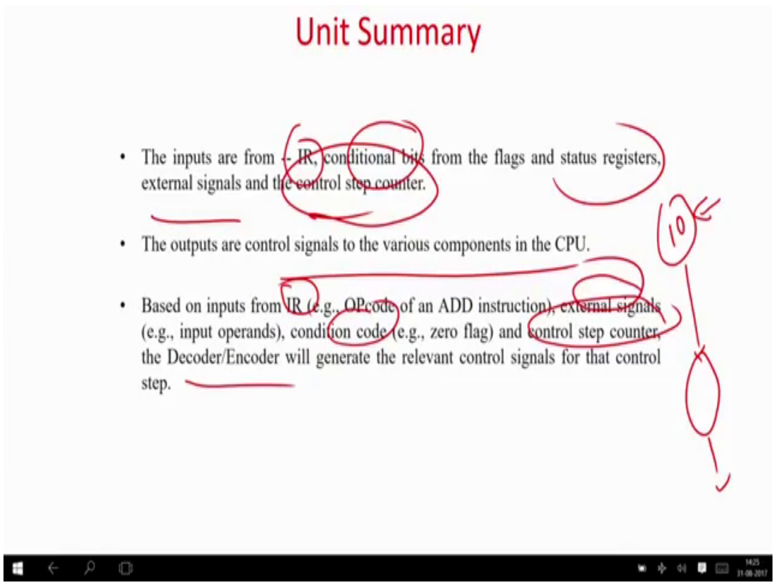
So, based on each of the face or each of the steps we have to generate the signals control signals. So, how we can generate them there can be basically two approaches: one is called hardware and another one is called the micro programmed in hardware. Basically, what is going to happen; we will have a dedicated finite state machine which will move from one state to another and each state will correspond to one time step of the micro instruction or one times step or one micro-instructions.

Basically and the control signals vary then whenever you move from one state to the another state it is basically nothing, but one step of the micro instruction with the another micro-instruction. Basically, we have seen that first three are basically corresponding to this say face a fetch of an instruction. So, first three states of the finite state machine will be corresponding to the fetch face and basically whatever signal it is like, PC out m b r in make read etcetera will be the output of the corresponding final state machine based states, because we know that in this unit. Basically, if you have forgotten you have to basically revise the concepts of designing of mealy and more machine even a truth table implementation. When you have to understand and recollect from your digital design fundamentals; that how a finite state machine can be implemented and based on some inputs how the outputs can be generated.

So, in our case the inputs will be nothing, but the different states output of the instruction registered then flag registers and some signals which will be coming from the data which will be coming from the controll bus that is your signals from the external memory etcetera and then it will generate some outputs which will be corresponding to each state as I as I told you each state correspond to a step or a micro instruction and the output will be signals and this is actually hardwired, why we call it hardwired; because corresponding to each macro instructions we know what are the micro-instructions.

We will generate a finite state machine and it will be hardcoded, but the same thing can also be done using a software approach; which is called the micro programmed base which we will look at later, but in this case basically everything is hardwired in terms of the finite state machine states. Therefore, actually we call it as a hardwired design and. In fact, a hardwired control we have to generate design the finite state machine in terms of gates and flip flops. So, basically as I told you we have to again recollect, but given a sequence of inputs and outputs and states we have to recollect how a finite state machine is.

(Refer Slide Time: 04:31)



Unit Summary

- The inputs are from - IR, conditional bits from the flags and status registers, external signals and the control step counter.
- The outputs are control signals to the various components in the CPU.
- Based on inputs from IR (e.g., OPcode of an ADD instruction), external signals (e.g., input operands), condition code (e.g., zero flag) and control step counter, the Decoder/Encoder will generate the relevant control signals for that control step.

Designed basically as I told you the inputs are from the instruction register control flags and status registers and control flags some external signals and the control step counter; that is very important what is the control step counter it is nothing, but which have a finite state machine which goes from one step to another.

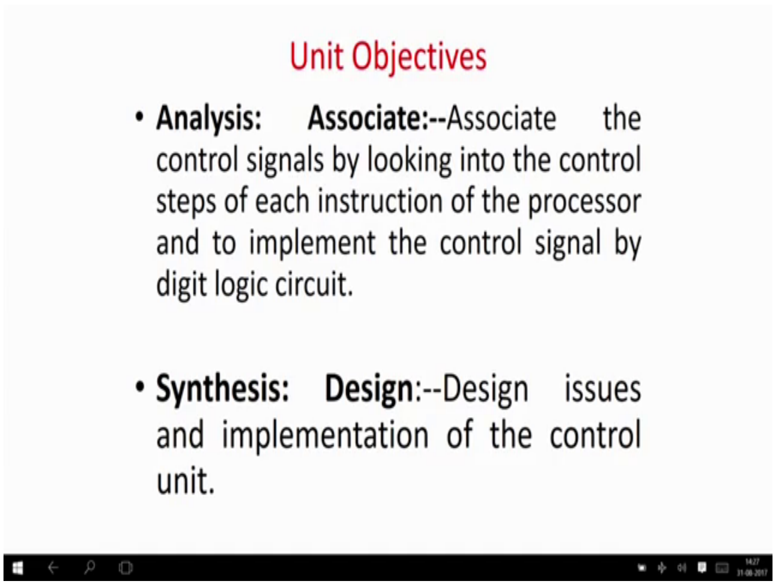
So, also you have to know that what are the values of the state variables at that state that will also depending depend on like for example, in this state the variables are same all zero. So, the output will also depend on the value of the present state is the very standard terminology, when you are saying it was an implementation in terms of circuits.

The outputs are of course, the different control signals which go to the CPU at the memory. So, based on the your instruction register external signals condition code; that is your flag and control step counter control step counter is nothing, but the your state in which you are in the decoder or the encoder that is the your which is going to generate the signal for that based on all the input that the decoder or there will be decoder encoder combination; that is a sequential sorry a combinational circuit will be there which will generate the element signals control signals at each of the finite state machine based state.

So, in nutshell what you are going to see today? We are going to see different type of instructions and the micro-instructions there in we will see and then we will see how a finite state machine is designed which is hardcoded, because anyway a finite state machine is hardcoded because that is not flexible and how it generate the corresponding control signal; that is what is the summary of this unit?.

So, what are the object unit objectives?

(Refer Slide Time: 05:55)



Unit Objectives

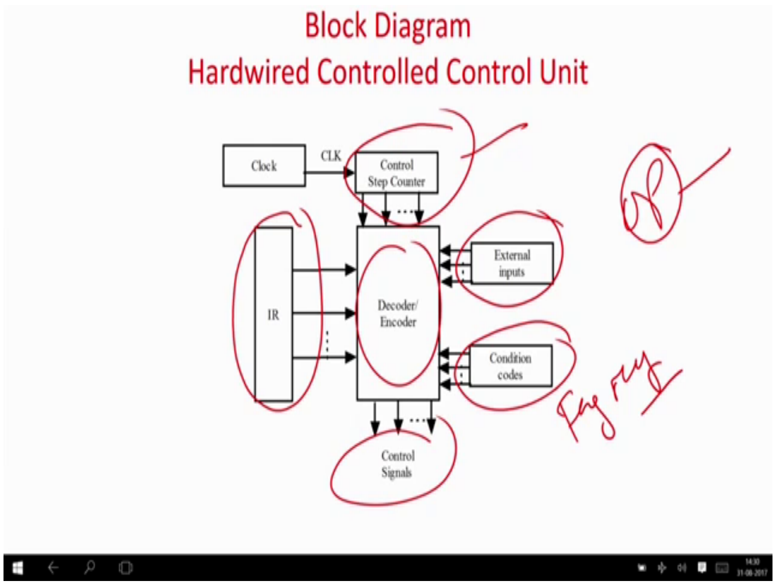
- **Analysis: Associate:**--Associate the control signals by looking into the control steps of each instruction of the processor and to implement the control signal by digit logic circuit.

- **Synthesis: Design**:--Design issues and implementation of the control unit.

At this unit you will be able to this is an analysis objective you will be able to associate the control signals by looking at the control steps of each instruction that is the micro instruction and the processor and to implement the control signal by digital logic circuit. That means, basically you just if you look at the sequence of micro-instructions and the control signals they are in you will be able to basically associate for which state what will be signals will be there; and how the finite state machine can be designed.

So, again for this basic design of finite state machine will not be covering in this unit; because it corresponds to your digital design fundamentals. So, if some truth table is given. So, how you design a finite state machine in terms of gates and flip flops that you have to recollect that. And finally, there is a design objective design the design issues and implementation of the control unit. So, if you I give you a macro instruction you will be able to design the finite state machine based controller for that that is; what is the idea? So, now, let us go into the unit.

(Refer Slide Time: 06:51)



Block Diagram
Hardwired Controlled Control Unit

So, basically this is; what is your block diagram of a hardwired controlled unit? So, what is there as I already told you the heart of the input is instruction register because you will have an op-code and then the operands see the op-code will basically tell you, what is the correct instruction; and what are the control signals required for that.

For example, if it is a load instruction you will have the different if it is the store instruction it will be different; that means, the micro-instructions corresponding to that. So, if it is the fetch part then sub different step has to be done if it is the add you are really going to add the two opponents then different I mean different step has to be done. So, all this things will be basically controlled by the op-code which will be in the instruction register.

So, that is the heart of the controller the control unit then of course, here the external inputs. So, what are the external inputs like your memory block is saying that I have already done with the reading or writing part; then you can access my memory data register. So, that is; what there are some condition codes? So, this condition codes basically are you can understand that this is something like your flags basically the flag registers.

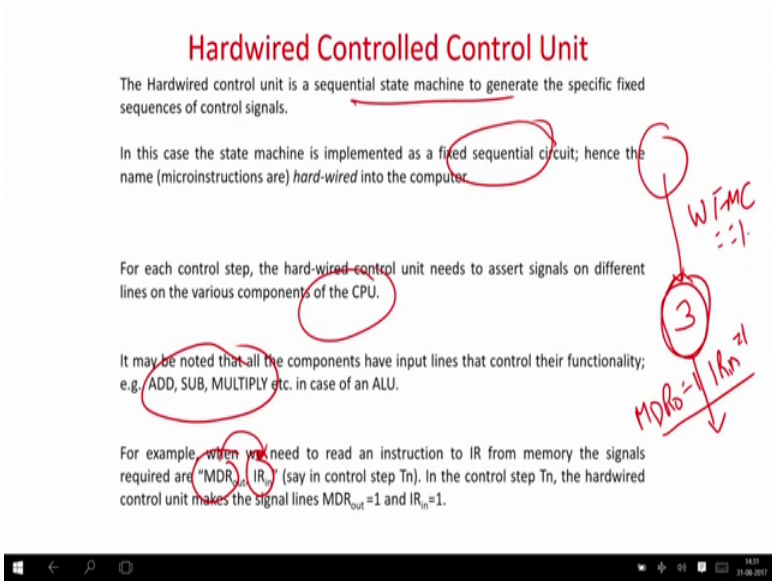
So, the flag registers will directly tell you that whether the 0 flag inside the sign flag inside accessed. So, those inputs also has to be taken if it is the conditional instruction because conditional instructions will always depend on inputs of the I means flag registers you can consider this as the flag register and finally, there is something called a control step counter that is step 1, step 2, step 3 some step there may be any if then else condition like; if 0 is said you do something else.

So, the finite state machine implementation also the state the state bits will also give a this as input taking all these inputs basically you are going to generate some control signals which are going to control generate the control signals. These signals are like PC out 1, PC in 1 there may be memory register out one etcetera. So, basically or add 1 subtract 1 I mean multiplexer bit select 0 or 1.

So, all this things will be generated by looking at all the inputs. So, these were already we have discussed like IR external inputs and conditional codes here one important thing is that; as you have to go in steps like add r 1, r 2 say 5 steps are there add r 1 m may be 6 stages will be there. So, there will be a finite state machine which will be involving this 6 stages and all this and also depending on what state you are the control signals will change. So, that state variable as an input will be given by the control step counter and obvious a clock will be driving it, because it is a finite state machine.

So, now basically if such an input is given like I have inputs external conditions control step (Refer Time: 09:15) then how you can design this decoder or how you can design (Refer Time: 09:18) out of it is the standard digital design procedure which we have to read when you which you might have already read in your digital design fundamentals and you have to recollect them back that is the finite state machine synthesis you can use any kind of flip flop like j k s r d whatever you require accordingly you can design the circuit.

(Refer Slide Time: 09:37)



Hardwired Controlled Control Unit

The Hardwired control unit is a sequential state machine to generate the specific fixed sequences of control signals.

In this case the state machine is implemented as a fixed sequential circuit; hence the name (microinstructions are) *hard-wired* into the computer.

For each control step, the hard-wired control unit needs to assert signals on different lines on the various components of the CPU.

It may be noted that all the components have input lines that control their functionality; e.g. ADD, SUB, MULTIPLY etc. in case of an ALU.

For example, when we need to read an instruction to IR from memory the signals required are "MDR_{out}, IR_{in}" (say in control step Tn). In the control step Tn, the hardwired control unit makes the signal lines MDR_{out} =1 and IR_{in}=1.

So, basically whatever I told you can read through over this slide it says that hardwired control unit is a sequential state machine; that is the finite state machine which will generate the sequence of controls.

As a it is a finite state sequential machine which is hardcoded. So, we actually called it as a hardwired control unit; because once the finite state machine is synthesised you cannot change that and the output of the finite state machine basically will generate different control signals which will be given to the ALU the read or write part of the CPU sorry write part of the memory block etcetera.

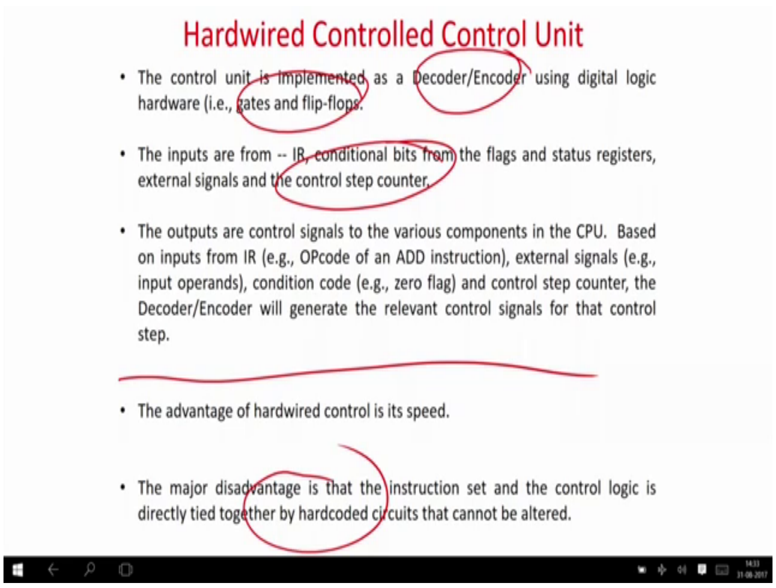
And of course, you can tell add or subtract or multiply etcetera to be done for example, they have given a simple example say when you want to read an instruction to IR from the memory. So, generally write MDR out IR in; that means, basically when saying that memory has already given the value to the memory buffer register or the data register, then I can actually put it into the instruction register say may be it is happening at state number 3.

So, basically what is happening at state number 3 basically the output will be MDR out equal to 1 and IR in that signal will be equal 1 all other control signals will be equal to 0. So, basically in this case we just need to depend there is no input responsible here we just need to see the control bits or the state variables for state number three should match and then you can easily generate out the signal.

So, in this at for this simple thing that the MDR out and IR in there is no require to think about an inputs. In this case, unlike for some other cases like where you have to wait for the input signal that the memory is ready. So, may be at some other stage from coming from here to here with W F M C equal to equal to 1. So, you have just waited that MFC is 1; that means, the memory he has already told that whatever data you are asking from the memory has already being dump to the memory data register and now you can read it.

So, by getting this input signal you come over here this is an input and there is you do not have any output correspondingly, but at this stage what happens you just need to know that I am in state 3 that just depend on the state variables and just generate the output MDR out equal to 1 and instruction register equal to in. So, basically in this case only your hardwired control unit will depend on the state variables as the input you just has various example to basically how it works.

(Refer Slide Time: 11:50)



Hardwired Controlled Control Unit

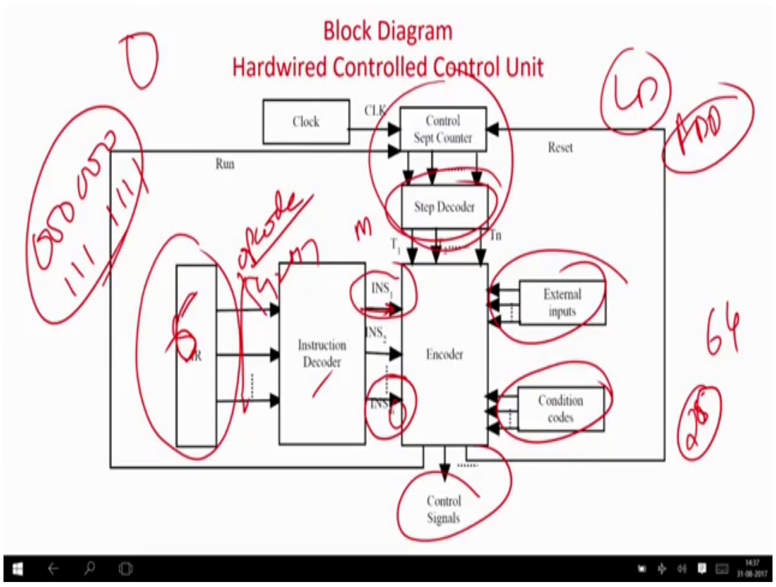
- The control unit is implemented as a Decoder/Encoder using digital logic hardware (i.e., gates and flip-flops.
- The inputs are from -- IR, conditional bits from the flags and status registers, external signals and the control step counter.
- The outputs are control signals to the various components in the CPU. Based on inputs from IR (e.g., OPcode of an ADD instruction), external signals (e.g., input operands), condition code (e.g., zero flag) and control step counter, the Decoder/Encoder will generate the relevant control signals for that control step.
- The advantage of hardwired control is its speed.
- The major disadvantage is that the instruction set and the control logic is directly tied together by hardcoded circuits that cannot be altered.

And then basically what happens we will finally, have a decoder encoder and series of flip flops which will design the basic circuit which will take all this inputs and generate the corresponding output signals and in the proper sequence of steps of a finite state machine. So, it is a finite state machine based synthesis you have to learn as I again repeating in the inputs are IR condition bits flags status registers, and basically your controlled step counter.

Control step counter is nothing, but your steps of the finite state machine the outputs will be basically your some control signals which will be giving it to the different parts of the code like your ALU memory read etcetera again now. So, this is actually very straight forward whatever I have discussed I have written in this slide you can go through it; then what are the advantages of a hardwired control unit it is a speed.

So, whenever you have something implemented in hardwired it is extremely fast disadvantage is that basically it is hardcoded that is you cannot change. So, given a micro instruction set or a macro instruction the corresponding micro instruction is what is very very hardcoded when we will be going for the micro program based control you will find out that is slightly slower, but there is not a flexibility there. So, that you can appreciate only when we will be discussing in the micro program based control unit for the time being this understand that and this is the hardwired based implementation low flexibility and disadvantage, but fast implementation is the advantage.

(Refer Slide Time: 13:11)



So, again slightly more detailed diagram basically what happens. So, there is something called the instruction register. So, the instruction register actually we will feed to a instruction decoder basically this is this part is nothing, but your op-code. So, it one it is going to tell you that say if it is a load instruction or if it is a add instruction which finite state machine basically you have to choose; because for load there will be a several finite state machine for add.

There will be a several finite state machine, because a load means a series of micro-instructions for that fetch the instruction in fetch register then you will decode and lode means you have to again look at the memory and dump the value. So, you have already seen that for a load R 1 R 2 and for a load R 1 M the sequence of micro-instructions will be defined. So, based on the op-code or if else add instruction etcetera.

So, based on the op-code it will give the inputs to a decoder the decoder is actually make the corresponding one bit as high. So, if there are m instructions. So, it will be a decoder of M and input will be log 2 M cap so; that means, the idea is say for example, if there are 60 4 bit 64 instructions possible then 2 to the power 5. So, there will be actually 5 bits will be your op-code. So, op-code can generate sorry 62 to the power 5 is 32 to the power 6 is 64.

So, basically sorry this a 64 six bit input. So, you can generate all the numbers from 0, 2 to the power 6. So, it will generate all the numbers this is 6 bit. So, you can generate all numbers from 0, 0, 0, 0, 0, 0 to all ones correspondingly basically it will generate the means the decoder. So, any out of any 1 bit out of the 64 inputs will be a 1 which will correspond to the micro instruction or basically the finite state machine corresponding to that macro instruction say for example, for load maybe the op-code is all zeros.

So in fact, it will first make this bit as 1 and the INS 1 will actually invoke the finite state machine correspond to the to the load may be 1, 1, 1, 1, 1, 1, 1, 1 is the op-code for the instruction and then whenever all the IR will give the value of 1, 1, 1, 1, 1, 1where will go to the instruction decoder it will make INS M equal to 1; then INS M equal to 1.

This bit will correspond to the encoder it will actually activate the finite state machine which corresponds to the micro-instruction implementation for add instruction that way it goes external inputs are always similar as I told you they have nothing to change; that is your instructions or signals from your process memory etcetera condition codes are the flag registers which will be given to the encoder and this is the step decoder step decoder is nothing, but T 1, T 2 up to T n.

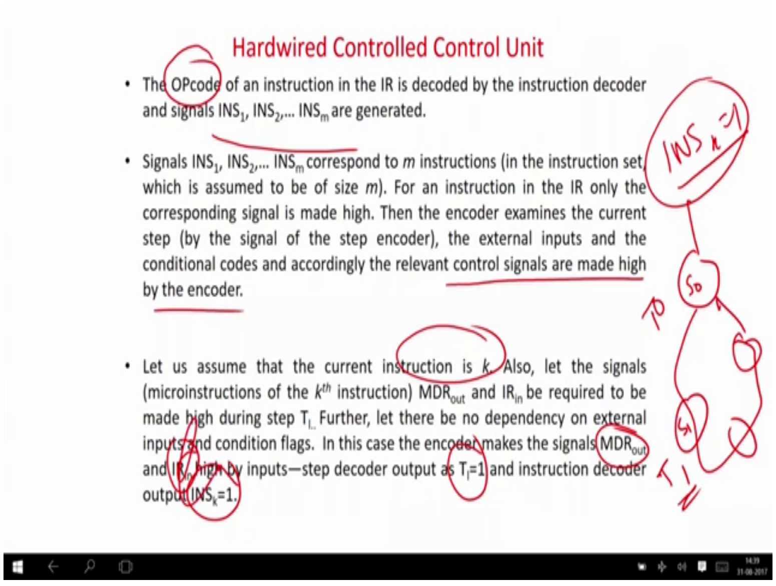
So, T 1, T 2 T n are basically the steps. So, for example, if you are at state 0 T 1 will be high if you are at state one T 2 will be high and so forth or you can even think of a binary encoding like for example, if the state 0 is there all T ns will be 0 if it is state 1. So, it will be 0, 0, 0, 0, 1. So, that can be a binary coding or it can be a hot one encoding kind of a thing like in state 1 it is a 1 in state 2 it is a 1 and so, forth.

So, any way it is encoding is the way to minimize the circuit for, but for explanation just you can understand that for given any state a proper encoding will going step controller is counter is basically it basically generates the moment in the finite state machine taking all this inputs it will generate the control signals so. In fact, what happens; then in a nutshell your instruction register will give the op-code corresponding to an instruction based on that any one of the signal will be made high that corresponds to the macro instruction based on this you will invoke the finite state machine which will correspond to that macro instruction and then you will move to the sequences of steps based on the external inputs.

And the condition codes and as well as the steps, because step 1 some instruction some of signals will be generated step 2 some instruction will be generated; and it will keep on actually moving 1, 2, 3, 4, 5, 6 if there if then else condition then you can have to move from one state to another or otherwise there is a incremental value of the steps.

So, I have written that basically in the text you can read over it. So, what it says that?

(Refer Slide Time: 17:26)



Hardwired Controlled Control Unit

- The OPcode of an instruction in the IR is decoded by the instruction decoder and signals INS_1, INS_2,... INS_m are generated.
- Signals INS_1, INS_2,... INS_m correspond to *m* instructions (in the instruction set, which is assumed to be of size *m*). For an instruction in the IR only the corresponding signal is made high. Then the encoder examines the current step (by the signal of the step encoder), the external inputs and the conditional codes and accordingly the relevant control signals are made high by the encoder.
- Let us assume that the current instruction is *k*. Also, let the signals (microinstructions of the k^{th} instruction) MDR_{out} and IR_{in} be required to be made high during step T_l. Further, let there be no dependency on external inputs and condition flags. In this case the encoder makes the signals MDR_{out} and IR_{in} high by inputs—step decoder output as T_l=1 and instruction decoder output INS_k=1.

The op-code will actually generate signals I 1 to I m. So, if there are m signals based on op-code any one of them will be high and that will actually invoke the corresponding finite state machine correspond, then the decoder examines the signal and accordingly the relevant control signals are made high by the encoder. In fact, actually it will select a corresponding finite state machine.

Say for example, they are saying that the current instruction is k. So, that is INS k will be invoked for example, that is what they saying INS k is invoked. So, if the if the k instruction then this is going to be invoke and then let us saying that assume that you are in T equal to 1; that means, may be this is we are in the first stage. So, whenever INS k equal to 1 so; that means, the k-th instruction corresponds to the op-code of the current instruction which is in the instruction decoder. So, INS 1 will be the 1 followed by the decoder.

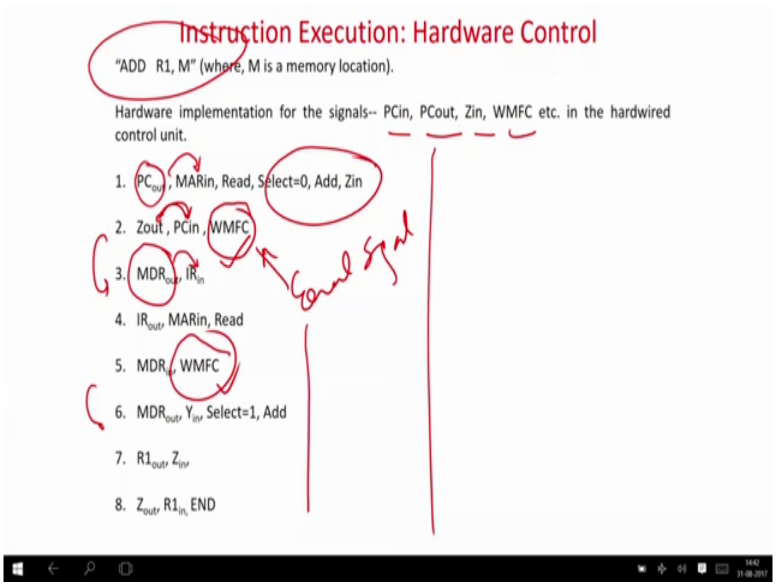
So, when it is the case the corresponding finite state machine like we will have lot of states in this. So, this one is going to be invoke and may be you are in state S 0, then whenever I will move to state this is T 0. Assume this is T 1 then at time T 1 first in after first step I will go to T 1 and may be in that state basically you are going to take appropriate actions like you can make the MDR out to IR in; that is you are going to make this zero some steps can happen. So, basically this how it happens; so, corresponding to the INS; that is instruction kind based on the op-code of corresponding state machine will be invoked that is you are going to be initial state.

After that depending on the depending on this one that is your step controller you will go to step 1, step 2, step 3 and therefore, that mean in this state you not have any input you just generate MDR out and IR in. So, that is corresponding signals will be generated and we will go forth. So, that is how we basically happen; you can read the slide and also I have explained you. So, basically you can get a very quick understanding what is happening whenever you are going to now take some complete examples which will make this step absolutely clear all that thing you remember.

Then based on the op-code only one bit for the corresponding instruction will be made help that finite state machine will be invoked and after that you will go step 1, step 2, step 3 based on the counter that is your stage counter we call it that is your step counter and then you at each step you are going to read some flag values some input values some other values and you are going to generate the corresponding control signal that is how it happens.

Now, we will take some very concrete examples and the thing will become very very clear for us.

(Refer Slide Time: 19:48)



Instruction Execution: Hardware Control

"ADD R1, M" (where, M is a memory location).

Hardware implementation for the signals-- PCin, PCout, Zin, WMFC etc. in the hardwired control unit.

1. PC_{out}, MARin, Read, Select=0, Add, Zin
2. Zout , PCin , WMFC
3. MDR_{out}, IR_{in}
4. IR_{out}, MARin, Read
5. MDR_{in}, WMFC
6. MDR_{out}, Y_{in}, Select=1, Add
7. $R1_{out}$, Z_{in},
8. Z_{out}, $R1_{in}$, END

So, first let us take an example called add R 1 m that is you are taking the memory location value R 1 dumping it to R 1 after adding it to whatever the content in R 1 if you look forget about last class we have discussed that is the series of control instructions, what are the signals involved? Program counter in program counter out Zin FMC all the signals whichever we are listed here are actually utilized for this micro-instruction, sorry; the micro-instructions which will be involved in this macro instructions. So, they are all will be generated.

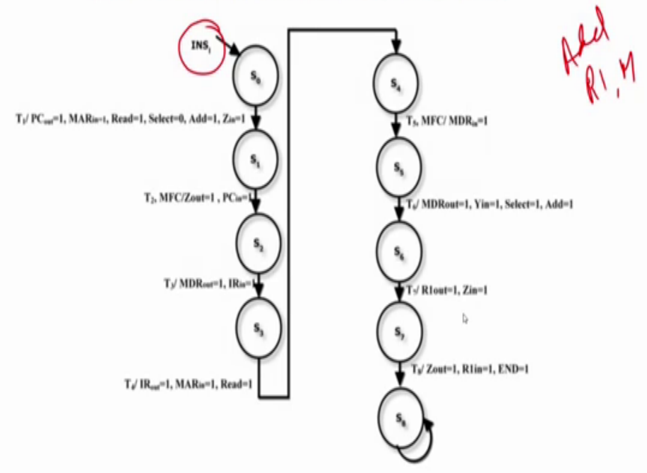
So, now, first we will say if we remember that first was program counter out that is the PC value will be fed to the memory address register and then you will make it select zero z add in this part corresponds to incrementing of the value of PC. So, whenever I say PC equal to Zin out; that means, the PC is increment, sorry this way. So, whenever Zout means the value of incremented value of PC will be dumped into PC output, because Z output actually was storing the value of accumulated sorry the value of the ALU which was added the value of program counter plus the constant.

Now, we have to put it in PC that now PC is updated. Now, we have to a l is important we are waiting for an external signal this is an external signal. So, this is an external signal. So, we have to wait for the external signal all the once when it is ready in stage three basically what you can do the instruction which is now in MDR can be loaded to register in and similarly you can go ahead. So, they can again there is a external signal and all others are internal signals like PC out MARin read they are all signals which are generated and here we are waiting for something which is again a condition which I am depending on which comes from an external one.

So, all this is like PC out, MARin, read, select 0, ADD, Zin that the control signals which you have to generate and then WFMC is something on which you have to wait till you can go to the third stage similarly from the 5th stage to go to the 6th stage basically you have to again wait for the WFMC; that means, this round this special this two are basically inputs which on which the movement of the final state machine will depend and all other signals are basically being generated which has to be generated by the your hardware control unit which in this case is a finite state machine. Now, we will first basically see the circuit and then we will go for the discussion.

(Refer Slide Time: 22:03)

Instruction Execution: Hardware Control



INS_1
S_0
T_1/ PC_{out}=1, MAR_{in}=1, Read=1, Select=0, Add=1, Z_{in}=1
S_1
T_2, MFC/Zout=1 , PC_{in}=1
S_2
T_3/ MDR_{out}=1, IR_{in}=1
S_3
T_4/ IR_{out}=1, MAR_{in}=1, Read=1
S_4
T_5, MFC/ MDR_{in}=1
S_5
T_6/ MDRout=1, Yin=1, Select=1, Add=1
S_6
T_7/ R1out=1, Zin=1
S_7
T_8/ Zout=1, R1in=1, END=1
S_8
Add R1, M

So, let us assume that this add R 1 M whatever may be the op-code for this corresponds to first instruction. So, what is going to happen INS 1 will be 1; that in the decoder the first line will be 1 which corresponds to this instruction which is the first instructions, so wherever the INS is 1, because it is a decoder. So, only one line will be 1. So, only this finite state machine will be invoked before look at the initial stage it is input is INS 1.

So, only if the value of INS 1 equal to 1, then this basically an invoke and you know there is already a decoder. So, based on the input of the op-code only one of this machines will be invoke like there will be another machine for INS 2. Another machine for INS 3 dot dot dot up to INS m so, based on which instruction is in the instruction decoder the op-code it will be correspondingly invoking only one finite state machine, because correspondingly this input is equal to 1.

Now, next what let me zoom this part? So, it will be clear. So, next is what I am doing? So, you are invoking this finite state machine based on the in which based on the output of the instruction decoder INS 1 equal to 1. Next, what next I am going to go to state 1. So, what is the output? Initial state is there from state 0 to state 1, what happens; basically you should know that now I should be in state T 1 that is from 0.

Next state has come that is counter has now become one then what are the outs if you remember there is only one thing over here after initial state basically you need not depend on any other input only thing is that state 1 corresponds to the fact that at present the instruction is INS 1 that is ADD R 1 M register to memory. Then next is nothing just you need to wait till T 1 comes T 1 comes means; that is sequence counter has now become 1.

So, what is that now after that there is no other input required only time has passed the clock has come that from stage 0 to stage 1 you can go and what are this signals out PC out MARin read 1, select 0, ADD 1, Zin 1. So, all this signals will be correspondingly generated at this state and all other signals which are not shown like MARin MARin is 1. So, what about MARout it will be equal to 0 like PC out equal to 1. So, what about PC in it will be 0? So, whatever I am not shown here are made 0 now it is important. So, now, in this case you have said that I want to read the memory in the memory address register you have given the value of PC; that means, you are going to fetch the instruction.

Second state is instruction second state is interesting basically. So, when I am in state one again you are waiting till you go to the second state that is second clock first that the counter has become two not only here you have to again wait here for another external symbol that is WFMC that MFC that MFC you have to wait this another signal which will be coming from the memory through the bass it was is going to tell that. Now the memory has been read everything is fine.

Now you can go ahead only after this you can go to the next state. So, what is the next state; you are saying that PC in equal to 1 and Zout equal to 1 that is you are going to dump the value of program counter with an updated value from Z which is nothing, but the constant value of ALU, but before initially there is only one input which are we are depending on; that is your clock count that is state one here you are depending on two that is clock counter as well as you are waiting till the memory is ready.

So, whenever the memory is ready you can go to the next state S 2, there is nothing else you could depend on just the timer; because already the memory signal has been ready in the previous state just you dump the value of memory data register to the instruction register after that just the sequence because there is no if then else condition you did not wait for any memory read here. So, the memory data register out; that is basically instruction has been now loaded to the instruction register.

For state three nothing you have to do just you have to wait for the next clock pulse which will make the state equal to 4 here instruction register out will go to the memory address register in, because you have to fetch another opponent from the memory if you remember your instruction is R 1 M the next instruction is available in the memory location. So, what you are going to do instruction register out that is m in the instruction register is dumped into the memory address register and you are making the register mode as 1.

So, these are the control signals which will be generated all other not mention like memory out Rin they are all zeros then again I go sequentially in next state here also I do not need to depend on anything else except is the important over here I have given a read signal to the memory. So, I cannot only depend on the input. So, only input here is T 5; that is state 5 is going to come the counter is 5, I cannot depend only on that here again I have to wait till the memory gives a green signal that I have done you can read it.

So, whenever it says MFC; that means, I am done. So, then you can go for here actually MDR in is 1; that means, you are going to generate the signal that I want to read the memory; because till here it is memory address you have given the value of the register value M and you are saying read equal to 1. So, you have said that the memory in a read mode. So, this one says that memory is being read a green signal has been there then only you can give the corresponding signals like here already MFC has been 1 right.

MDR in means you are going to get the value of the memory into memory address register, but now you cannot read here, because you cannot read at this at this stage you cannot read basically the value of MDR, because I am waiting for the memory location to be done here. I am giving a read signal, but when I am going to state six; that means, already this has been over long back. So, I can go for MDR out equal to 1 and all other similar procedure like this signal has been done.

So, if you remember for all other states similarly; I can complete all the sequences of this state machine design like for 6 this is the state 6 that counter is 6. I will generate the corresponding counter instruction 7, I will generate the control signals correspondingly 8; I will generate the corresponding signals correspondingly. Finally, m equal to 1 is you have to stop this set of micro-instructions and the macro instruction is over.

So, what is to be emphasized is very easy to understand only some clocks are there first is this machine is invoked that is the initial state condition enabling is INS 1 that is based on the op-code only, when the signal is one you can go to that corresponding state machine if there are m instruction m state machines will be there and based only on the output of the instruction register via the decoder you can invoke that corresponding state machine.

Now, for all cases generally if you do not wait want to wait for any kind of an external input like; your condition flags or your memory values output of the memory that is memory is ready or there is some signals like you want to wait that if the flag is 0 or flag is 1. So, if no such conditions are there you just need to wait for the time stage 1, stage 2, stage 3 these are generated by the clock and the if is in counter go to stage one and generate all the corresponding signals which are required to do that only here like whenever you have given a read signal to the memory.

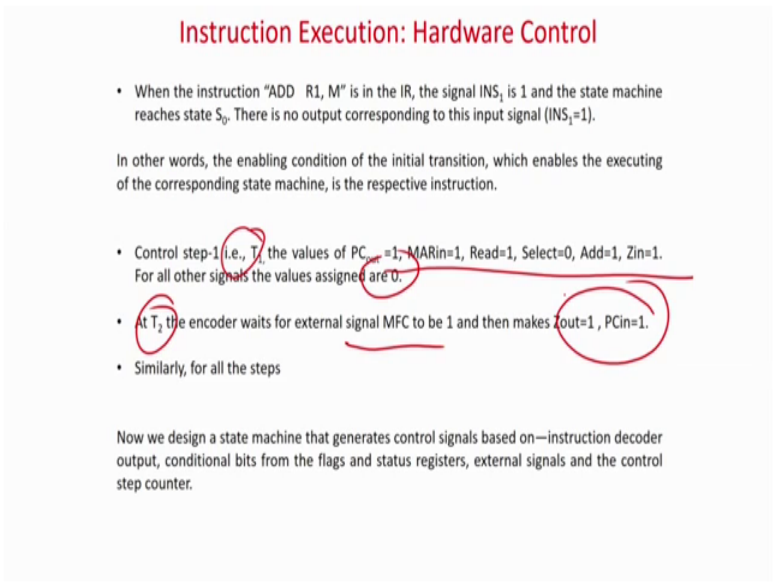
So, you have to wait till the memory says that I am ready. So, I that case the input from this state to this state the transition will depend on two inputs one is the state and one is the external signal and then you generate the corresponding signals like on this case it will be similar, but here again I have to wait for the memory to be ready, because here I have given a memory read this is a memory read state that T 4 and you are going for IR out that is the instruction register you want to read from the operant from the memory location m.

So, in the memory register address register you are giving the value m and you want to read it. So, at least you have to wait for some amount of time till you can get the value of memory data out to your some kind of the; another operant y where it will be stored. So, this ready can happen only if the MFC is ready. So, only in that way it can be done ok. So, in this way you can explain the whole thing. So, if there are 6, 7, 8 are just sequential part and then finally, you are going to generate the end out of it.

So, this is the finite state machine corresponding to the corresponding the set of macro micro-instructions corresponding to the macro instruction R 1 m that is you first write down this control steps control signals steps then you find out what are the inputs there are two inputs only here WMFC; WMFC and all other signals to be generated and of course, 1, 2, 3, 4, 5, 6, 7, 8 are the sequence numbers. So, you have inputs like sequence number as well as WFMC.

So, these are the two inputs and for all other cases you have to generate. So, similarly you can map this to the finite state machine, I have shown you and then you have to just go for finite state machine base synthesis and your job is done how to generate a finite state machine from a may be the finite state machine design how you will generate in terms of gates and flip flops; is just a simple digital design fundamental which you can read and go back and read your second semester text books this is what is written in written in the language you can read through the slide.

(Refer Slide Time: 30:54)



Instruction Execution: Hardware Control

- When the instruction "ADD R1, M" is in the IR, the signal INS_1 is 1 and the state machine reaches state S_0. There is no output corresponding to this input signal (INS_1=1).

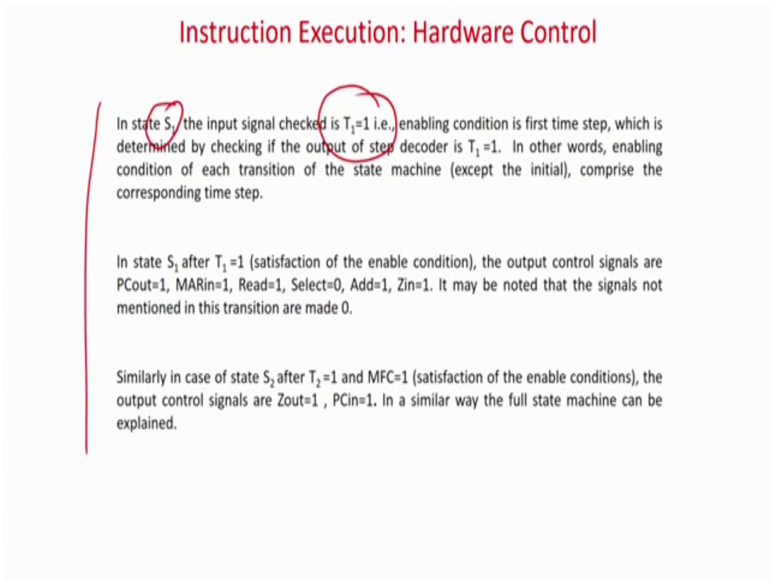
In other words, the enabling condition of the initial transition, which enables the executing of the corresponding state machine, is the respective instruction.

- Control step-1 i.e., T_1, the values of PC_{out} =1, MARin=1, Read=1, Select=0, Add=1, Zin=1. For all other signals the values assigned are 0.
- At T_2 the encoder waits for external signal MFC to be 1 and then makes Zout=1 , PCin=1.
- Similarly, for all the steps

Now we design a state machine that generates control signals based on—instruction decoder output, conditional bits from the flags and status registers, external signals and the control step counter.

Basically and you will get a very clear idea like say in control step one the values are this 1. So, you have to generate these values directly and all other signal values are 0, which I have already told you at T 2 you have to wait for the external signal m f f m c be to be 1. So, basically there are two inputs for the second stage; that is your clock counter stage counter as well as the signal form the memory and then you generate the PC equal to 1. Similarly in this way you can generate the whole circuit whole finite state machine and then you can synthesize.

(Refer Slide Time: 31:21)



Instruction Execution: Hardware Control

In state S_1 the input signal checked is T_1=1 i.e., enabling condition is first time step, which is determined by checking if the output of step decoder is T_1 =1. In other words, enabling condition of each transition of the state machine (except the initial), comprise the corresponding time step.

In state S_1 after T_1 =1 (satisfaction of the enable condition), the output control signals are PCout=1, MARin=1, Read=1, Select=0, Add=1, Zin=1. It may be noted that the signals not mentioned in this transition are made 0.

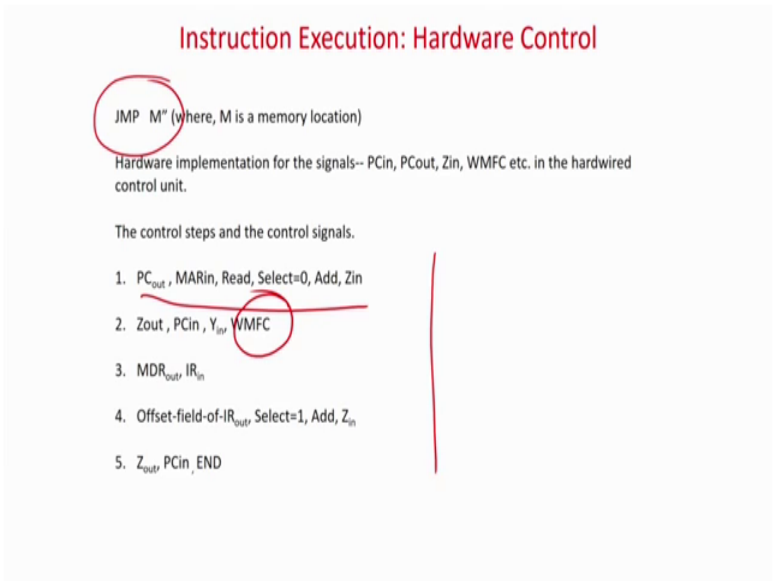
Similarly in case of state S_2 after T_2 =1 and MFC=1 (satisfaction of the enable conditions), the output control signals are Zout=1 , PCin=1. In a similar way the full state machine can be explained.

This right this is this is about basically your state generation in state one, basically you have to you have to check whether the T 1 that is state 1 is equal to 1. So, that is this slide is basically; if saying that from one state to another state the transition depends always on something called the state counter that is T 1, T 2 T 3; that is the states that it will depends on state counter that I can go from state 1 to state 2; only if the state variables have been incremented this is just a finite state machine based implementation.

So, whenever you want to generate implement a finite state machine you remember that always the there are you have to increment this states like 0, 0, 0, 0, 0, 1, 1, 0, 1 if you generate the forward counter. So, that that only you can translate into a sequential machine in states will go through this is just a simple digital design fundamental. So, basically this state slide is showing that for all the cases one primary input for this hardware control is the state counter T 1, T 2, T 3 they are nothing, but the state variable values 0, 1, 2, 3, 4, 5, 6 along with that sometimes you will have extra inputs which in this case the output of the memory sometimes it can be flag register values also ok.

So, now basically you are going to as I have told you we are also see basically how a flag register can be an input till now. The last instruction was a simple non control based implementation just a flow, but here we are going to see two type of instruction in which case if and conditional branch and one will say the conditional jump. So, conditional jump actually the corresponding instruction we will have another input which will come from the flag register so that you can get a basic idea.

(Refer Slide Time: 32:56)



Instruction Execution: Hardware Control

JMP M" (where, M is a memory location)

Hardware implementation for the signals-- PCin, PCout, Zin, WMFC etc. in the hardwired control unit.

The control steps and the control signals.

1. PC_{out} , MARin, Read, Select=0, Add, Zin
2. Zout , PCin , Y_{in}, WMFC
3. MDR_{out}, IR_{in}
4. Offset-field-of-IR_{out}, Select=1, Add, Z_{in}
5. Z_{out}, PCin , END

So, jump to M as you have already seen these are the control signals involved there is much to discuss you just look at the other last previous unit then you will get through what are the signals to be generated. So, these are all the signals to be given as output this is 1 input we have to wait for from the memory and that is it and all others are signals which has to be generated right or at the output signal. So, what will be the step? So, again importantly this is assume that jump in is signal number instruction number 2.

So, jump unconditional to M in signal number 2; so, only when the op-code corresponding to jump M comes; then only this INS 2 will be equal to 1. So, only the initial state input value will be satisfied only for this finite state machine for all instruction there are different-different finite state machines, but whenever you are going to give jump M in the instruction register the corresponding op-code will actually satisfy only the input condition for the finite state machine. So, this will be input and it will not satisfy the initial condition state for the previous instruction or any other instruction in that matter.

So, only this instruction will be activated. So, again as I told you stage 1. So, counter is one you generate all the signals and go to state 1, state 2. Basically similar two is the input and you have to wait for the memory to be ready, because you are going to read the memory value at the instruction in the first stage. So, whenever it is ready then only you can actually start the memory read. So, it is generating the signals and the third stage, because in this stage you have already verified that the memory is ready. So, you are saying that MDR equal to Rin. So, this state says that memory is ready. So, you can actually dump the value of the memory instruction.

T four T 4 what basically as you see the jump unconditional. So, you have to take the offset value and actually you have to put it in Zin already we have seen. So, in this case you have to take the value of offset and select 1 add 1, Zin; that means, what you are going to do you are going to get the updated value of PC you are going to add with the offset value of IR and the you will actually we are going to get the new value of the program counter that is the jump address and you will dump that value to PC and your job is done.

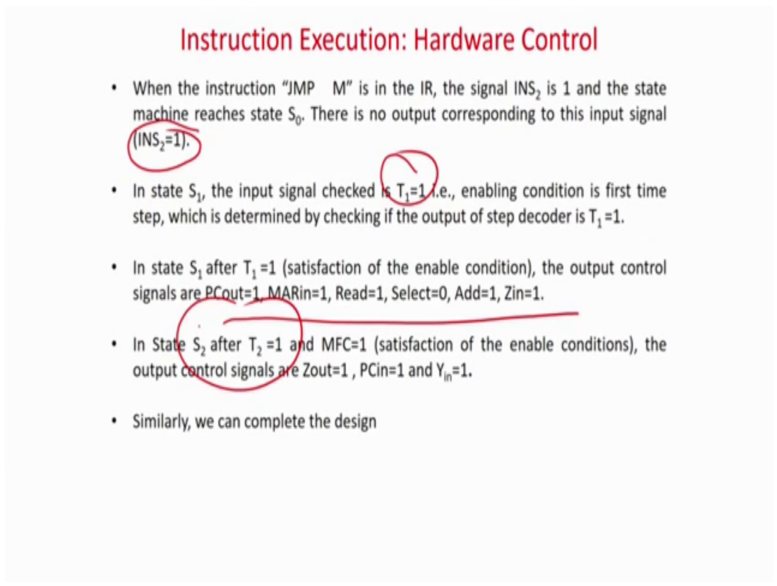
So, basically, but here there is no condition, so whatever maybe state there is no input required from any flag register just you have to take the value of offset value of IR you have to add it with the program counter value which is already saved in a if you look at it. So, the PC out and Zin and of course, which is saved in some temporary register in that manner you have to just get the value added and it will be the new value of the PC so. In fact, there is nothing to depend upon. So, we just say that IR out that is instruction.

Of course, the offset is taken as the output select one add one Zin if you remember it means that I am selecting the wide variable not the constant variable to be added to the there is an updated value of PC which will be in the data bus which will be present in the bus and the output will be saved in Zin that is actually equal to PC plus offset value of PC which is nothing, but equal to jump the variable m will come to the PC the value of M will come to the PC that is state 4.

So, once it is the condition we will come over here then actually a Zout PC in; that means, there is a Zout here is nothing, but the M which will be dumped to the program counter and it is end and; that means, in this case we are just updating the value of PC to M it was jumped M. So, already we have seen that there be just a sequence of step, because the unconditional instruction. So, only everywhere will depend on just the state that is whenever the counter will be four you can go over here and so, forth.

And it will be basically end the machine it is a very simple explanation which we have done to, but to appreciate the fact that when there is another input on which your state machine transition will depend will be clear when you are taking a conditional jump which we are now going to do again you can just go through this slide which I have just told you that what will happen after which state like for example.

(Refer Slide Time: 37:00)

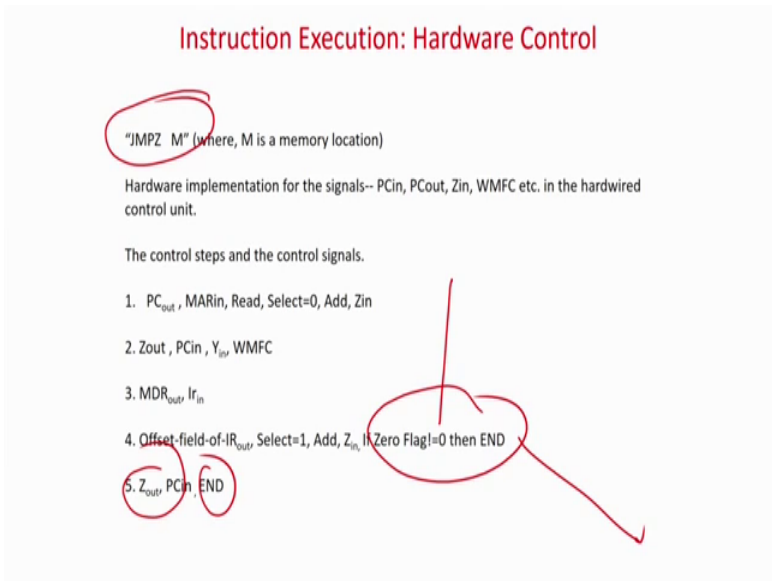


Instruction Execution: Hardware Control

- When the instruction "JMP M" is in the IR, the signal INS_2 is 1 and the state machine reaches state S_0. There is no output corresponding to this input signal (INS_2=1).
- In state S_1, the input signal checked is T_1=1 i.e., enabling condition is first time step, which is determined by checking if the output of step decoder is T_1=1.
- In state S_1 after T_1 =1 (satisfaction of the enable condition), the output control signals are PCout=1, MARin=1, Read=1, Select=0, Add=1, Zin=1.
- In State S_2 after T_2 =1 and MFC=1 (satisfaction of the enable conditions), the output control signals are Zout=1 , PCin=1 and Y_{in}=1.
- Similarly, we can complete the design

In jump state this is the initial condition which is enabled at T 1 state one; when you are going to generate all this signals at T 2 after the second state counter is one you are going to generate the whole signal and then the whole design can be very easily completed most interestingly.

(Refer Slide Time: 37:18)



Instruction Execution: Hardware Control

"JMPZ M" (where, M is a memory location)

Hardware implementation for the signals-- PCin, PCout, Zin, WMFC etc. in the hardwired control unit.

The control steps and the control signals.

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

2. Zout , PCin , Y_{in}, WMFC

3. MDR_{out}, Ir_{in}

4. Offset-field-of-IR_{out}, Select=1, Add, Z_{in}, If Zero Flag!=0 then END

5. Z_{out}, PCin , END

Now, whenever I tell you that the input to the finite state machine will depend on state as well as some external inputs like your memory or your register flags then we have to take a conditional jump. So, this is a conditional jump on Z to M. So, therefore, if you see all other steps will be similar here you are saying that the offset of IR select and everything, but if flag value is not equal to 0, then end if flag equal to 0, then you update the value of 0 a PC with M, but if the flag variable is not set, then you go over here and you do not have to do basically anything or if the flag value is 0, then you have to update this that Zout PC in means the update value or PC with that is M will be dumped to PC.

So, the program counter will have the value m and we will start executing from instruction which is present in memory location M, but if the flag value is not 0. Now from there itself we will come out and the PC will go as forward without having dumped the value of M into the PC. So, pc will be PC plus increment and it will keep on doing it. So, here this will be very interesting to see how we will do it. So, again INS 3. So, you are assuming that jump on 0 to M is instruction 3.

So, only this line will be one. So, only this finite sate machine will be invoke so. In fact, you can see that for each instruction there is a separate finite state machine which will be invoke. So, you can understand that if a finite state machine will be there which will be hardcoded. So, it will take some more area, but. In fact, it will be extremely fast; because here you are mainly depending on the inputs which is nothing, but your states if have lot of or lot of instructions you could have optimized in this way that many of for many of the cases the initial states up to first one two three states are similar then only it is deviating.

So, I can make a more complicated type of finite state machine say for instruction 1, 2 and 3, this is the finite state main which is synthesized; because the first three are similar depending on the input I can then bifurcate into some different braches then again I can do it. So, of course, you will have an optimized hardware implementation because may be the first three states are similar for everybody last two states are similar for everybody. So, I could have made a merged machine and then keep on branching and joining which will give me a better optimized machine, but it will be slower because you will be now depending on the state as well as type of instructions and again you will be joining and merging. So, this hardware size will be a bit small, but it will be a basically more slower design.

So, if I have a full flexibility then it will become a program that is your micro program based controller which we will see later, but for the time being we are actually giving dedicated hardware for each instruction and it is the fastest way of implementation. So, in this case INS 3 is enabled; that means, your instruction at present is jump on 0 to this 1 to some location T 1 will be similar stage one you are going to generate all the signals as output T 2 control state is two you have to wait till the memory is ready. So, that you can get the updated value of program counter from Zout of course, you are storing the value of program count in the temporary variable Y, because you have to add PC to your offset value MDR is very similar instruction you are dumping to instruction register from the memory data register this is instruction fetch.

Now, important now we are saying that if t four of course, if zero flag is set. So, now, you see the inputs are 2, 1 is the state another one is your zero flag. So, this is an input which is coming from basically the external register sorry the register input that is the flag register this is an external input that is from the memory. So, if this state depends on two inputs this state depends on only one input that is your state count here also depends on two input that is state as well as zero flag. So, the zero flag is set then what you do you take the offset you add it to your y and basically dump the output to Z 1.

So, Zin that is actually now your Z is actually having the value of M which is offset plus the program counter all there the detail we have seen in the last unit. So, now, Z is actually having the updated value of. So, if 0 is set you update the value of PC with M, and then you come over here and then what you do you dump the value of Z to program count and End and End. So, basically Zout basically Z is Z is having the value of M you dump to PC that is my PC in equal to 1.

So, now, the PC will be having the M and you will start executing from M, but there is another choice if the flag is not 0. So, there is another instruction another branch over here same for the same state if your counter has become 4 and if the zero flag is not set then what you do you just generate the end signal and it will actually come to the final step and look over. So, if the micro-instructions will be stopped.

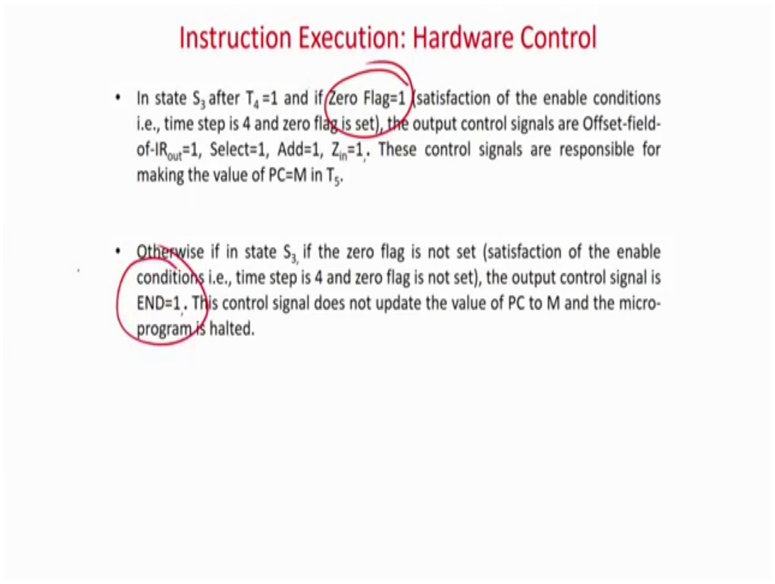
So, this shows very nice example that if there is a conditional jump then you can have a state where there will be a bifurcation. So, here there are two types of inputs state input as well as the inputs from the flags and here then again two inputs one is the flag and one is an external input which is basically your coming from your memory that memory read has been done. So, this shows an hardware control unit generation for a conditional jump.

So, from this discussion you can see that it is very simple to design the finite state machine which corresponds to different instructions for each instruction basically you generate this sequence of inputs which will always be some states as well as the control signals which are to be output and for some particular cases you have to also depend on some external inputs like from the memory; if there are conditional instruction you have to look at the flag registers etcetera.

And then just you have to go for a finite state machine based synthesis which is a standard digital design fundamental and you can make a hardware out of it which will be in terms of your flip flops and gates and it will be done, but only thing is that it takes more area, because for each instruction you will have a different finite state machine if I want to merge and make a optimization of it which will be slightly having lesser area, but it will be slightly slower and we do not try to do a trade off here.

So, what I try to do whenever you have a hardware based control we dedicatedly give for each instruction a finite state machine extremely fast, but you can just see that the area over it is higher as well is the very non flexible design that is for a given macro instruction they are the micro instruction and the hard this finite state machines are basically hardcoded. So, again if you just look at it whatever I have explained I have given you in the thesis I will give you this description in this slide you can read through it like that is the initial state is not the enabling condition is this in T 1 state these are the signals and so, forth and T 2 you have to wait for WFMC and you have to go through and most importantly you have to actually think about these state when we depend on the output of the flag registers. So, that is what has been discussed over here.

(Refer Slide Time: 43:57)



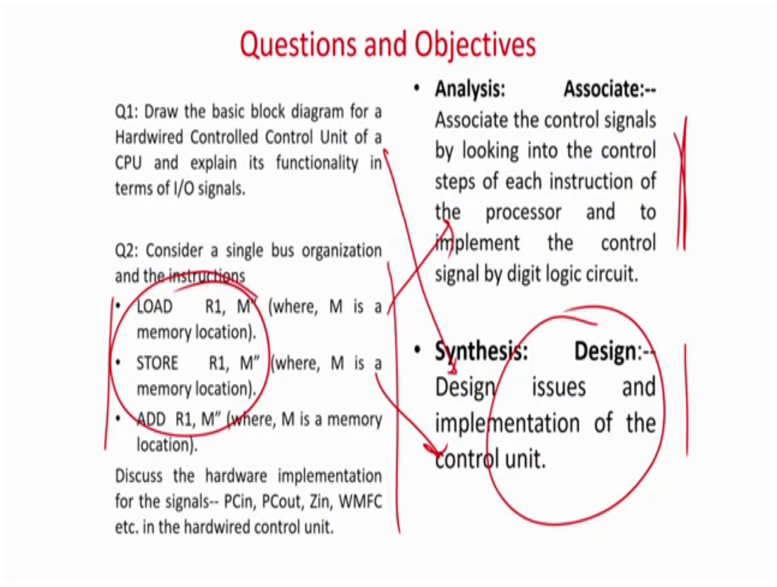
Instruction Execution: Hardware Control

- In state S_3 after T_4 =1 and if Zero Flag=1 (satisfaction of the enable conditions i.e., time step is 4 and zero flag is set), the output control signals are Offset-field-of-IR_{out}=1, Select=1, Add=1, Z_{in}=1. These control signals are responsible for making the value of PC=M in T_5.
- Otherwise if in state S_3, if the zero flag is not set (satisfaction of the enable conditions i.e., time step is 4 and zero flag is not set), the output control signal is END=1. This control signal does not update the value of PC to M and the micro-program is halted.

That, if the zero flag is set which is one of the inputs what happen; and if the zero flag is not set you directly go to END.

So, with this we come to the end of this unit and just have a little look at some of the sample questions like for example.

(Refer Slide Time: 44:14)



Questions and Objectives

Q1: Draw the basic block diagram for a Hardwired Controlled Control Unit of a CPU and explain its functionality in terms of I/O signals.

Q2: Consider a single bus organization and the instructions

- LOAD R1, M" (where, M is a memory location).
- STORE R1, M" (where, M is a memory location).
- ADD R1, M" (where, M is a memory location).

Discuss the hardware implementation for the signals-- PCin, PCout, Zin, WMFC etc. in the hardwired control unit.

- **Analysis:** **Associate:--** Associate the control signals by looking into the control steps of each instruction of the processor and to implement the control signal by digit logic circuit.
- **Synthesis:** **Design:--** Design issues and implementation of the control unit.

We say the first question is draw the basic block diagram for a hardwired controlled units of a CPU and explain it is functionality the theoretical question if you are able to answer this question you should be able to do it; because we have both discussed using examples as well as theory of how a hardwired control unit work if you are able to solve the question you will be are basically within the objectives of associating control signals by looking at the control steps and basically the design and design issues and implementation of a control unit.

So, mainly this actually focuses on this one then also slightly on this in objective, because you have to also know what control signals are depending on what, but mainly if you were able to explain the whole control unit in terms of block diagram you are able to synthesize the idea of how to design a control unit.

Then by different examples like load R 1 M store R 1 M, ADD R 1 M different types of macro instructions are there when I ask you to design a hardwired based controller out of it. So, by if you are able to solve this question of course, you are actually meting both the objectives because the second objective is the design objective where you are ask to design a micro controller oh, sorry; I mean say finite state machine based controller for a given set of macro instructions and of course, the you have to also have a good analysis idea, because you have to associate different control signals with different micro instruction steps.

So, if you are able to solve this your job is done. So, with this we come to the end of this unit from next unit on words basically; we will try to have a look at slightly more integrated details like what happens if you have multiple buses how things you are going to change how many less number of steps are there; how the finite state machine synthesis will change? So, we will be having a look at multiple bus architecture may be a 2 bus or a 3 bus architecture and also you have to look in details about your very importantly your micro program based control because you have seen that in fsn based control it is very fast, but everything is hardcoded and are non flexible.

So, we want to give more flexibility to which because you can see that many of the instructions have several parts has common. So, if we want to give some commonality and some flexibility we will go for micro program based design and then also in the end we will see how all the things get change if you have a multiple based architecture we just give an idea of it.

Thank you.