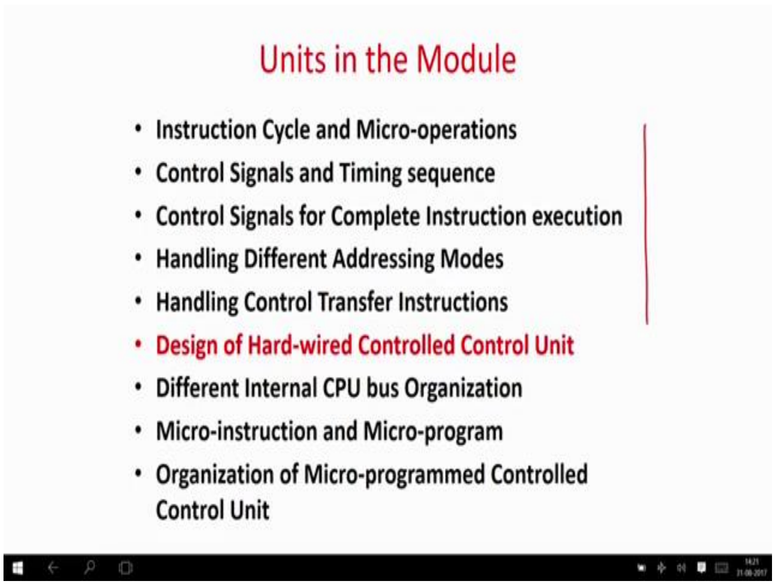
Computer Organization and Architecture: A Pedagogical Aspect

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 hardwired control unit.

(Refer Slide Time: 00:38)



Units in the Module

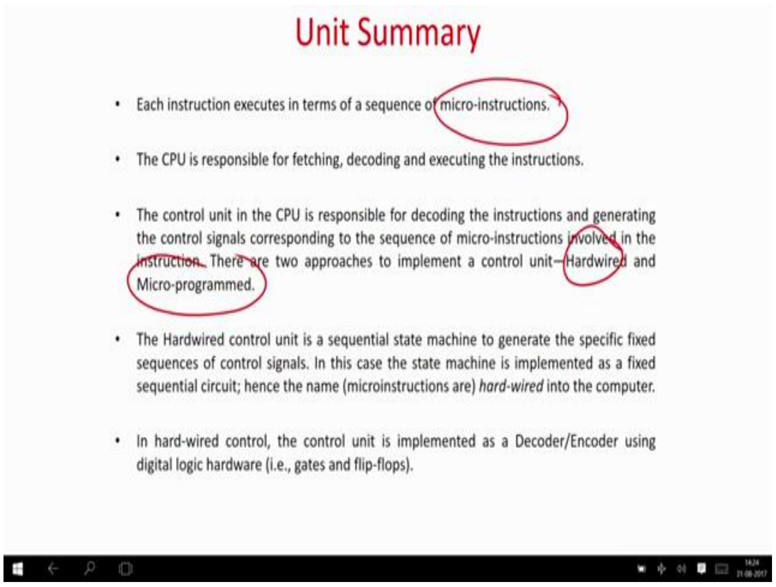
- Instruction Cycle and Micro-operations
- Control Signals and Timing sequence
- Control Signals for Complete Instruction execution
- Handling Different Addressing Modes
- Handling Control Transfer Instructions
- Design of Hard-wired Controlled Control Unit
- Different Internal CPU bus Organization
-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

If you look at the last few units we are 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 we will be 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 we can generate the control signals: one is actually called the hardwired which we are going to do today and another is basically called micro program based.

So, one is a hardware based and one is a software based. So, in this unit basically we will be focusing on the hardware based control unit in 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 phase 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 the read mode to the memory, then wait for the memory to give a signal that it is ready etcetera.

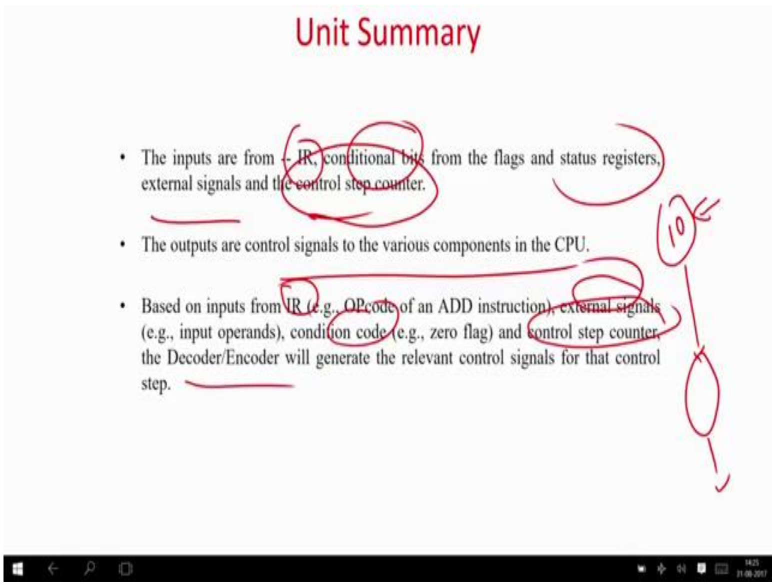
So, basically depending on each of the cycles that is we have seen there is some steps like in step 1, 2, 3 basically corresponds to a fetch of the instruction after that it basically corresponds to the execution of an instruction.

So, based on each of the phase 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 hardwired and another one is called the micro programmed. In hardwired 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 time 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 to the another microinstruction. Basically, we have seen that first three are basically corresponding to this fetch of an instruction. So, first three states of the finite state machine will be corresponding to the fetch phase and basically whatever signal it is like, PC_{out}, MBR_{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 Moore machine given a truth table implementation. Then 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 register then flag registers and some signals which will be coming from the data which will be coming from the 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 told you each state corresponds to a step or a micro instruction and the output will be the 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 program based which we will look at later, but in this case basically everything is hardwired in terms of the finite state machine states. So 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 designed.

(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.

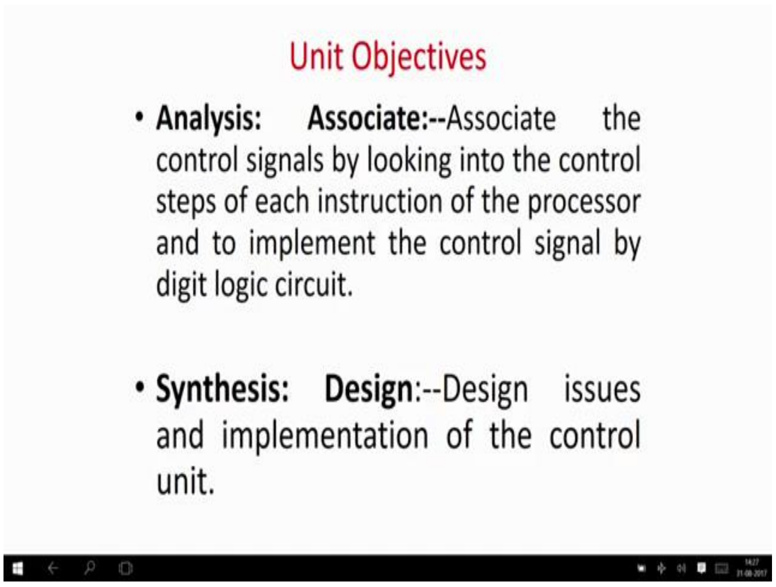
Basically as I told you the inputs are from the instruction register control flags and status registers and control flags some external signals and the control step counter; that is very important what is a control step counter it is nothing but if you have a finite state machine which goes from one step to another so, also you have to know that what are the values of the state variables at that state. That will also depend on like for example, in this state the variables are say all zero. So, the output will also depend on the value of the present state which is a very standard terminology, when you are saying FSM implementation in terms of circuits.

The outputs are of course, the different control signals which go to the CPU and 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 to the decoder or there will be decoder encoder combination; that is a sequential sorry a combinational circuit will be there which will generate the enable signals control signals at each of the finite state machine based state.

So, in nutshell what we 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 generates the corresponding control signals; that is what is the summary of this unit?.

So, what are the object unit objectives?

(Refer Slide Time: 05:55)



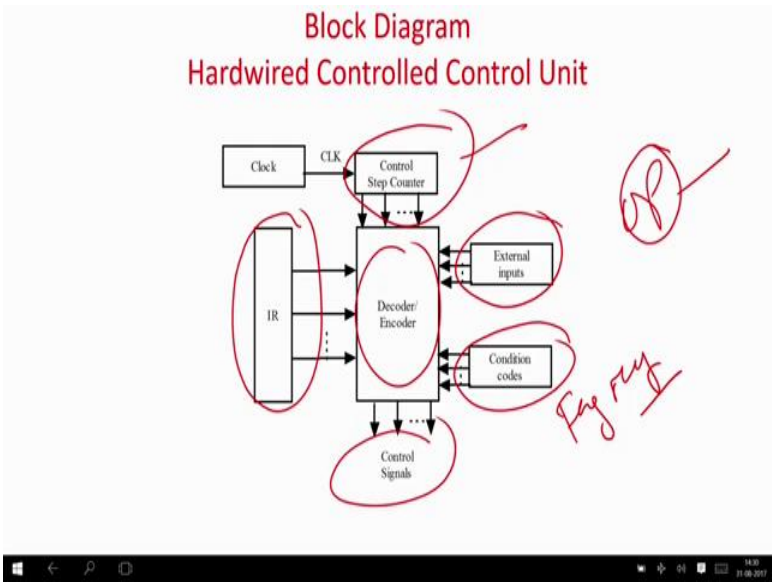
Unit Objectives

- **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 there in you will be able to basically associate for which state what are the signals will be there; and how the finite state machine can be designed.

So, again for this basic design of finite state machine we 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 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 control unit? So, what is there as I already told you the heart of the input is instruction register because you will have an opcode and then the operands see the opcode 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 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 some different step has to be done if it is add, you are really going to add the two operands then different I mean different step has to be done. So, all these things will be basically controlled by the opcode 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 these condition codes basically are you can understand that this is something like your flags basically the flag registers.

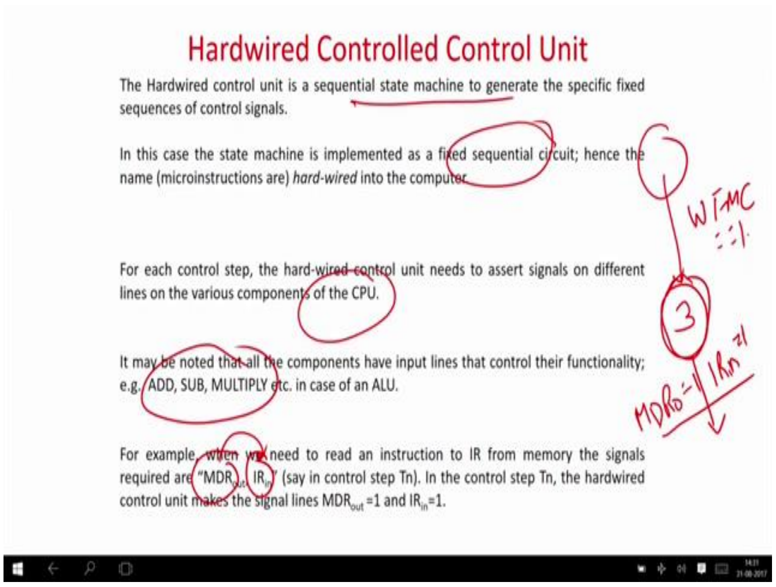
So, the flag registers will directly tell you that whether the 0 flag is set, the sign flag is set etcetera. So, those inputs also has to be taken if it is a conditional instruction because conditional instructions will always depend on inputs of the flag registers you can consider this as the flag register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

So, the finite state machine implementation also the state the state bits will also give this as an 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 $ADD\ R1, R2$ say 5 steps are there $ADD\ R1, M$ may be 6 stages will be there. So, there will be a finite state machine which will be involving this 6 stages and all these 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 of course 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 stuff then how you can design this decoder or how you can design FSM out of it is a standard digital design procedure which we have to read which you might have already read in your digital design fundamentals and you have to recollect them back that is a finite state machine synthesis, you can use any kind of flip flop like JK, SR, 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 it is a finite state sequential machine which is hardcoded. So, we actually call 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.

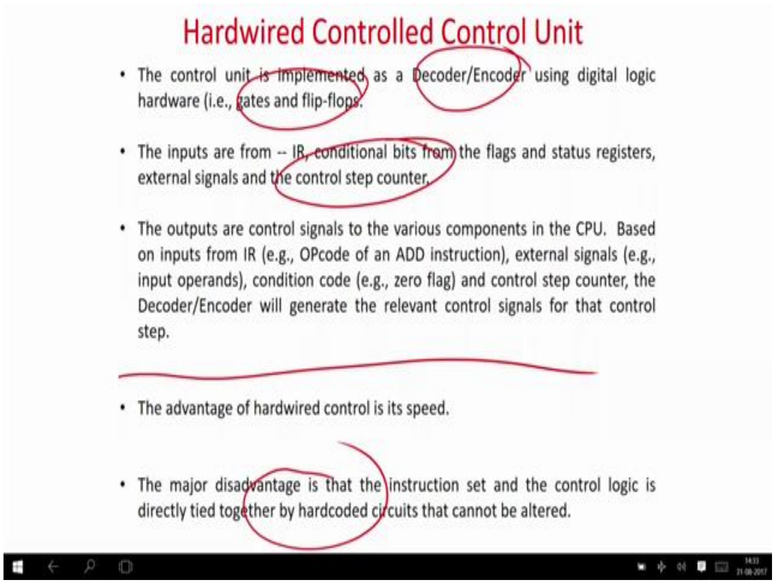
And of course, you can tell add or subtract or multiply etcetera to be done for example, they have given a simple example say when you want to read an instruction to IR from the memory. So, generally write MDR_{out} IR_{in}; that means, basically we are saying that memory has already given the value 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} $= 1$ 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 required to think about any 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 $WFMC = 1$. So, we 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 dumped 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 is does 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 $MDR_{out} = 1$ 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, it is just a very straight example to show basically how it works.

(Refer Slide Time: 11:50)



Hardwired Controlled Control Unit

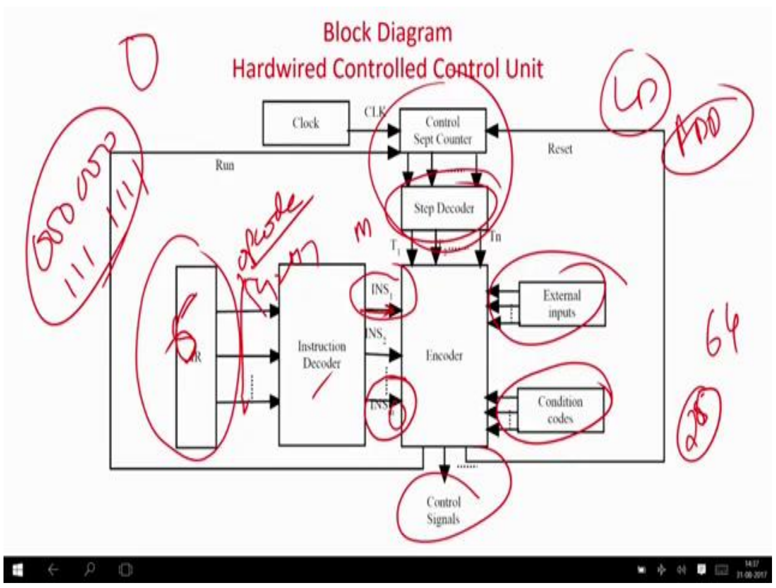
- The control unit is implemented as a Decoder/Encoder using digital logic hardware (i.e., gates and flip-flops.
- The inputs are from -- IR, conditional bits from the flags and status registers, external signals and the control step counter.
- The outputs are control signals to the various components 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's a finite state machine based synthesis you have to run as I again repeating in the inputs are IR, condition bits flags, status registers, and basically your control 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 the speed.

So, whenever you have something implemented in hardware 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 lot of flexibility there. So, that you can appreciate only when we will be discussing in the micro program based control unit, for the time being just understand that as this is the hardwired based implementation low flexibility a 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 opcode. 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 separate 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 instruction register then you will decode and load means you have to again look at the memory and dump the value. So, we have already seen that for a $LOAD\ R1, R2$

and for a $LOAD\ R1, M$ the sequence of micro-instructions will be different. So, based on the opcode or if it's an add instruction etcetera.

So, based on the opcode it will give the inputs to a decoder the decoder will 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 $log_2\overline{M}$. So that means, the idea is say for example, if there are 64 bit 64 instructions possible then 2^5. So, there will be actually 5 bits will be your opcode. So, opcode can generate sorry 2^5 is 32, 2^6 is 64.

So, basically sorry this 64 six bit input. So, you can generate all the numbers from 0 to 2^6. So, it will generate all the numbers this is 6 bit. So, you can generate all numbers from 000000 to all 1's correspondingly basically it will generate, this is a decoder. So, any out of any 1 bit out of the 64 inputs will be 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 opcode is all zeros.

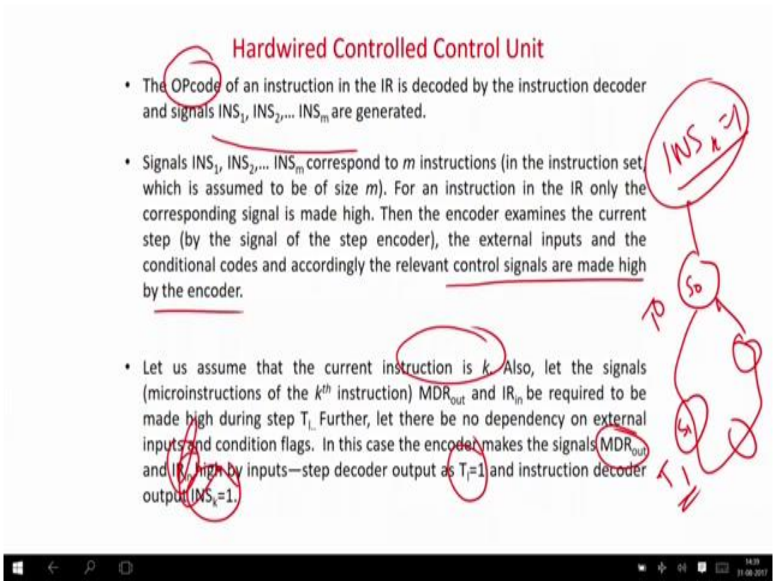
So in fact, it will first make this bit as 1 and the INS_1 will actually invoke the finite state machine corresponding to the load may be 1, 1, 1, 1, 1, 1, 1, 1 is the opcode for the instruction and then whenever all the IR will give the value of 1, 1, 1, 1, 1, 1 where it will go to the instruction decoder it will make $INS_M = 1$; then $INS_M = 1$ 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_n's 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 go in, step controller is counter is basically it basically generates the movement in the finite state machine. Taking all these inputs it will generate the control signals so. In fact, what happens then? In a nutshell your instruction register will give the opcode 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 through 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 is an if then else condition then you can have to move from one state to another but otherwise there is an 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_1. 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_1=1 and instruction decoder output INS_k=1.

The opcode will actually generate signals I_1 to I_m. So, if there are m signals based on opcode any one of them will be high and that will actually invoke the corresponding finite state machine. Correspondingly 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 is saying INS_k is invoked. So, if the if the k instruction then this is going to be invoked and then let us say that assume that you are in $T_1 = 1$; that means, may be this is we are in the first stage. So, whenever $INS_k = 1$ so that means, the k-th instruction corresponds to the opcode of the current instruction which is in the instruction decoder. So, INS_1 will be the 1 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 invoked and may be you are in state S0, 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 read the signal some steps can happen. So, basically this how it happens; so, corresponding to the INS; that is instruction type based on the opcode of corresponding state machine will be invoked that is you are going to the 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 that way, may be in this state you will 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 it 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 concrete examples which will make this step absolutely clear.

Only thing you remember that based on the opcode only one bit for the corresponding instruction will be made high, 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 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.