Digital Computer Organization

Prof. P.K. Biswas

Department of Electronic & Electrical Communication Engineering

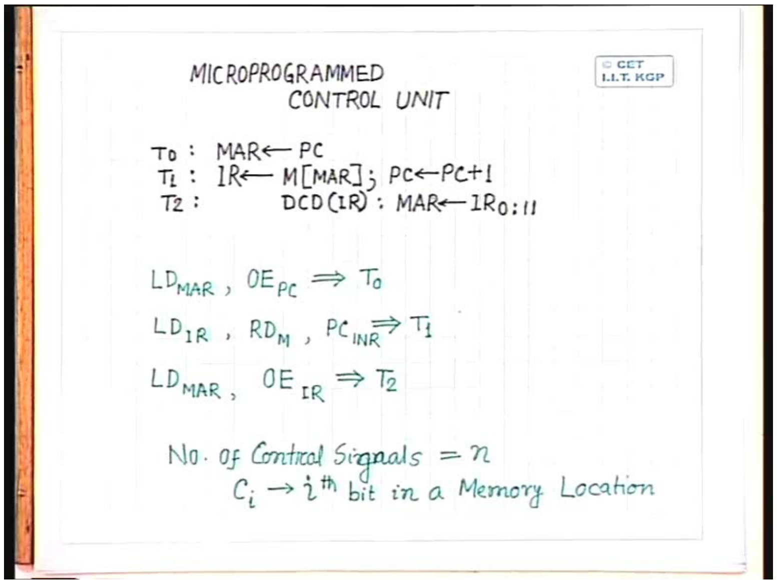
Indian Institute of Technology, Kharagpur

Microprogrammed Control

Lecture No. # 05

In last class the control circuit unit that we have discussed that is called hardware control unit because all the control signals are generated by hardware circuit. Now there is another way of generating the control signals and that is called a micro programmed control unit. In case of micro programmed control unit, the control signals instead of being generated by hardware circuit, it is generated through software and because these control signals are generated through software this is more flexible. So now let us see how you can generate the control signals through software. So we have said during our previous discussion that once you design the hardware resources within the CPU for every hardware resource, you can determine that what are the control signals that will be required.

(Refer Slide Time: 00:02:08 min)



MICROPROGRAMMED CONTROL UNIT

T_0 : $MAR \leftarrow PC$

T_1 : $IR \leftarrow M[MAR]$; $PC \leftarrow PC+1$

T_2 : $DCD(IR)$: $MAR \leftarrow IR_{0:11}$

LD_{MAR} , $OE_{PC} \Rightarrow T_0$

LD_{IR} , RD_M , $PC_{INR} \Rightarrow T_1$

LD_{MAR} , $OE_{IR} \Rightarrow T_2$

No. of Control Signals $= n$

$C_i \rightarrow i^{th}$ bit in a Memory Location

So in our hardware control unit when we have discussed, we have said that during time state T_0 the operation that was performed was the memory address register, gets the content of the program counter. Then during time interval T_1 or machine state T_1 what we had done is instruction register gets the value from memory whose address is in the memory address register. At the same time, the program counter is incremented by one and during T_2 the content of the instruction register is decoded. So what we do is we decode the instruction register content. So these are the operations that were done during the machine states T_0 to T_2. We have also said that simultaneously what we have done is we have loaded the memory address register with the lower 12 bits of the instruction register that is supposed to hold the operand address in memory, if it is a memory reference instruction.

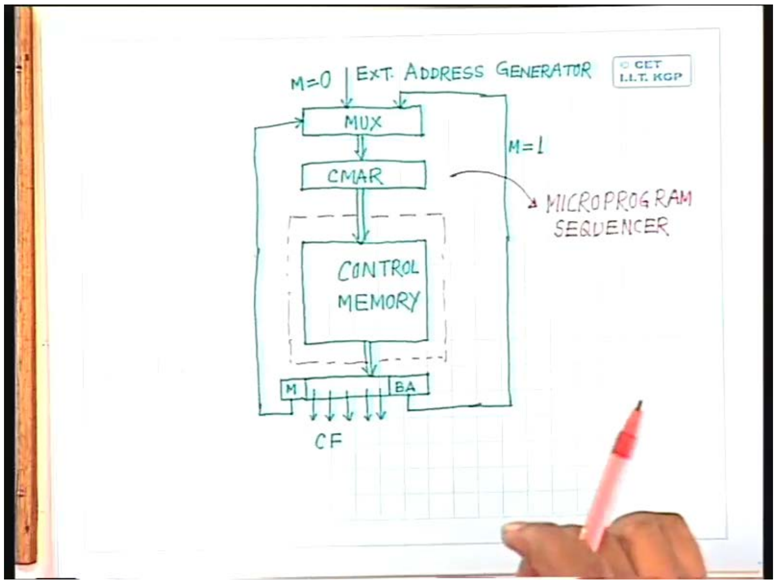
So if I consider only these three machine states, you find that the controls that are invoked is for memory address register, we have to have the load control input of the memory address register. So I have load of memory address register, I have the control input, output enable of the program counter. So both of these control signals are to be activated during machine state T_0. Then for the instruction register, we have to have load instruction register then because we have to perform a memory read operation, so we have to activate the read control signal which will go to the memory. The program counter is incremented, so for the program counter we have to activate the increment control signal. These are the control signals which have to be activated during machine state $T_{1.}$

Similarly during machine state T_2, we said that decoder is a combinational logic circuit. So for that we don't need any control signal to be activated. The moment, instruction register is loaded with the opcode of the instruction, the decoder starts working and the output of the decoder will be available after few gate delays but for the memory address register, when it is getting loaded from the instruction register, we have to again activate the load control input of the memory address register and we also have to activate the output enable control signal of the instruction register. So these are the control signals that have to be activated during time state T_2. So in case of hardware control circuit, what we have done is we have ensured generation of these control signals in this proper sequence by making use of the instruction decoder and the sequence counter.

Now there is another way of generating these control signals. That is given the architecture and the hardware resources that you have in the architecture, you identify total number of control signals that are required. So for the given circuit, if we say that the number of control signals that will be required, if number of control signals is equal to say n then we can generate these control signals in the given sequence, in the required sequence by having a memory whose every location will contain n number of bits. So a control signal C_i will be represented by the ith bit in a memory location. So since we are assuming that we are having a memory where every location in the memory will contain n number of bits, so every bit I can use to represent a control signal.

So when ith bit is equal to zero in a particular memory location and when that memory location is read, the control signal C_i will not be activated. Whereas if ith bit equal to one and I read that particular location in the memory then control signal C_i will be generated. So accordingly for each of these control signals, I can associate every control signal with a corresponding bit in a memory location and read the memory locations in the desired sequence. So as you read the memory locations in desired sequence, you find that the desired control signals will also be generated in the desired sequence. The total architecture of such a micro programmed control unit will be something like this.

(Refer Slide Time: 00:09:20 min)



So for storing these control signals I need a memory. So in this case the memory is called a control memory because it is storing the control signals. Now once I want to read a particular location in the control memory then I must have a register which tells you what is the address in the control memory that is to be read. So for that what we use is a control memory address register. We have a control memory address register or CMAR. Control Memory Address Register provides the location address where from this control memory is to be read. Then output of that particular location, I can load into another register. A part of this will give you the control signals. So this is a control signals and we call this field in the memory location as control field or CF and within this memory, there are other locations which can be divided in other fields.

For the simple architecture, let us assume that this field gives you the branch address or the next address in the control memory that is to be read. So we assume that in every location, whenever I put the control signals that are to be put in that particular location, in addition to that we also put what is the next address of the control memory location that will be read, after execution of the micro operations initiated by this control signals. So this branch address because this is the next location in the control memory that has to be read, I must have a provision for giving back these branch address to the control memory address register. I should also have a provision of loading some address from the external source to the control memory address register. Why that is needed? Because whenever a new instruction is to be executed, for the first three time states, machine states the operations are identical but after that the micro operations that are to be performed depends upon the instruction that is going to be executed.

So I must have an external resource generator or address generator which after decoding the instruction, identifies the location in the control memory from where the required control signals will be obtained. On this side I have to have a multiplexer. One of the inputs to the multiplexer is the branch address which comes from the control memory location itself. The other input to the multiplexer comes from the address generator or external address generator. So I call it external address generator.

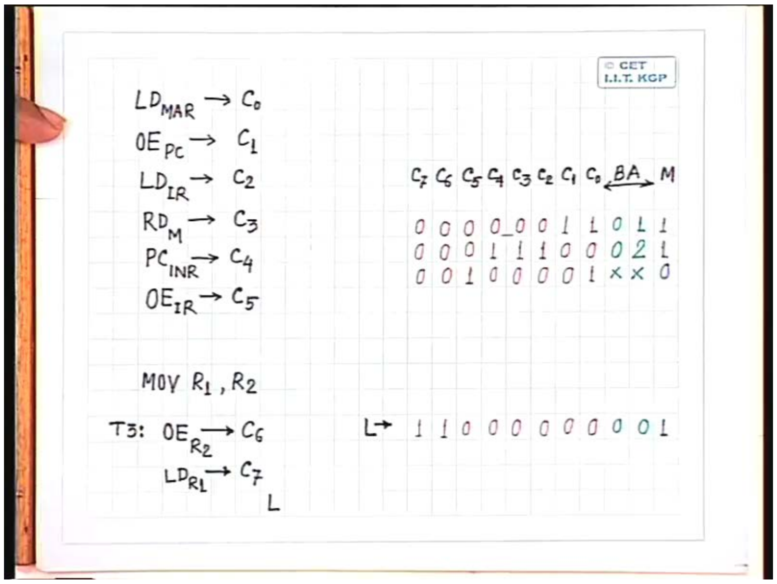
So this external address generator will generate the address depending upon the instruction that is going to be executed. I have one input to the multiplexer as the branch address which is provided within the control memory location itself and the other input to the multiplexer comes from the external address generator. This address is loaded into the control memory address register. I must also have a selection mechanism or the select input to the multiplexer which will decide whether the address that is going to be loaded in the control memory address register should be loaded from the external address generator or it should be loaded from the branch address provided within the control memory itself. So for that what we just done is within the same control memory location, I put a one bit field. Let us call it a mode field or m bit.

Our logic will be that whenever this m bit is equal to 0, the address will be taken from the external address generator. If it is equal to 1 then address will be taken from the branch address field of the control memory location. So this is the one which will be used when M is equal to 1 and this one will be loaded into control memory address register when M equal to 0. So this will be the over all architecture of a micro programmed control unit where the control signals are encoded within the control memory itself and this can be a round. So by analysis of the instructions, you decide that how many control signals are needed and in which sequence all the control signals will be generated. So accordingly you program this ROM, so that every location will contain control signals which are to be generated simultaneously.

Now in this architecture all the components except the control memory, so this is the one which is called the control memory. So all the components except the control memory take part in deciding the next location in the control memory that is to be read. So all those components are called as micro program sequencer. So this whole thing multiplexer, control memory address register then this output register all these are taken together becomes a micro program sequencer and the micro program sequencer along with a control memory becomes the entire micro program control unit.

Now let us see that how these control signals that is needed for all purpose can be generated through this micro program control unit. So as we said that for execution of the instructions in our machine, we need the control signals during T_0 to T_2. We need control signals, load memory address register, output enable of the program counter, load instruction register, read memory, increment program counter, load memory address register again, these are identical then output enable of the instruction register. So let us call these, associate these control signals with some bits in the control memory.

(Refer Slide Time: 00:18:41 min)



$LD_{MAR} \rightarrow C_0$

$OE_{PC} \rightarrow C_1$

$LD_{IR} \rightarrow C_2$

$RD_M \rightarrow C_3$

$PC_{INR} \rightarrow C_4$

$OE_{IR} \rightarrow C_5$

MOV R_1, R_2

T3: $OE_{R2} \rightarrow C_6$

$LD_{R1} \rightarrow C_7$

L

	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0	B	A	M
	0	0	0	0	0	0	1	1	0	1	1
	0	0	0	1	1	1	0	0	0	2	1
	0	0	1	0	0	0	0	1	×	×	0
L→	1	1	0	0	0	0	0	0	0	0	1

So let me say that load memory address register that is associated with bit number C_0 that is the zeroth bit in the control memory. Output enable of the program counter that is associated with bit number one in the control memory or C_1 then load instruction register is associated with bit number C_2 in the control memory. Then read control signal for memory is associated with C_3 in the control memory. Then we have program counter increment that is associated with bit number C_4 in the control memory then load memory address register is already considered. Then what we need is output enable of instruction register is associated with bit number C_5 in the control memory.

So if I put these control signals in different control memories consecutively, so I assume that starting from the zeroth location in the control memory, the control signals that will be stored are the control signals which are common for all the instructions, that means the control signals that are required during the machine states T_0, T_1 and T_2. So let me put it this way, that I put control signal C_0 here, C_1 here, C_2, C_3, C_4 and C_5. So these are the control signals that are required till now. Then during time state T_0, the control signals that are required are output enable of program counter and load memory address register. So we have said that load memory address register is associated with bit number C_0.

So in the zeroth location, I will put C_0 equal to 1 because that is the control signal that is required during time state T_0 and I have to have output enable of program counter because program counter is loaded into control memory address register. So output enable of program counter which is associated with bit number C_1, so I also said C_1 equal to 1. Rest of the bits are equal to 0 because I don't need these control signals during machine state T_0. Then during machine state T_1, the control signals that are required is load instruction register which is associated with bit number C_2. So I said C_2 equal to 1 in the next control memory location and I also need read control signal of the memory which is associated with bit number C_3. So I said C_3 equal to 1 because these are the control signals which are to be generated simultaneously and at the same time, the program counter is to be incremented by one.

So increment of the program counter which is associated with bit number C_4 that also is to be activated during time state T_1. So I also said C_4 equal to 1. The remaining memory locations are, the remaining gates are set to zero. Then during time state T_2, the control signals that is required are output enable of the instruction register which is associated with bit number C_5. So I set C_5 is equal to 1 and I also need load of the memory address register to be activated which is associated with bit number C_0. So I also set C_0 equal to 1. These are the two control signals which will be generated when you read this particular memory location and the remaining bits in that particular location are set to zero.

So you find that initially when you power on the machine, the first operation that is to be performed is an opcode fetch. So if I ensure that whenever the machine is powered on or whenever the machine is reset, the control memory address register will also be reset to zero. So that ensures that just after switching on the machine or just after resetting the machine, the micro programmed controlled unit will start generating control signals from the zeroth location in the control memory. That means the first control signals it will generate are C_0 control signals associated with C_0 and C_1 which are nothing but load memory address register and output enable of the program counter. From the program counter the address goes to memory address register.

Now in addition to this, we also have said that every location in the control memory will have two more fields. One is the next address field or from the next control signals are to be read and it also have a field, one bit field which is the address select field. So I will put along with this, in the same locations the next address fields within the control memory address register. So here we find that after generating these control signals, the next control signals are to be generated from the next memory location within the memory which is memory location one. So I put the next memory location in few more bits. I put it as 0 1. Right now I am putting it in decimal form but this has to be coded into binary and the number of bits required in binary form that has to be used. For simplicity I am putting this as decimal number.

So after execution of this, the next address is one that is this particular location in the control memory. So after generation of this control signals, next time the control signals will be generated are C_4, C_3 and C_2 and C_4, C_3, C_2 means increment program counter, read memory and load instruction register. So after these control signals are generated, the next control signals are to be read register from the third location in the control memory. So I again put in the next address field as 0 2, so you will find that I am putting in the decimal point.

Now after this third memory location is read and these control signals are generated that means I have already read in the instruction and the instruction opcode is there in the instruction register. After that the instructions are to be decoded and depending upon the decoder output, the control signals have to be generated and that is the actual execution phase of the instruction. So during execution phase as we have said, the address has to come from the external address generator. So this address cannot be specified within the control memory itself. So whatever I put in this branch address location that is immaterial because the address now is to be generated by the external source but for that what you have to do is we have to set the mode field accordingly. So as we have said that whenever mode field is set to one, the next address is taken from the next address field of the control memory. If the mode field is set to zero then the next address is taken from the external address generator.

So I have to set the mode fields accordingly. So what I will do is I will put the mode field here as one because after execution of this, the next address will be this branch address field. Here also I set mode field equal to one because after generating these control signals, the next control memory which is to be read is this one and these control signals are to be generated but after this, the address has to come from the external address generator, so I set mode field to zero. So this is the branch address field and this is the mode field in the control register. So after doing this, if I assume so that instruction that has been fetched is nothing but it's a move R_1, R_2 instruction. So if it is move R_1, R_2 instruction then during time state T_3 what are the control signals that you have to generate? One is output enable of register R_2 and the other control signal that is to be generated is load of R_1.

So I can have two more bits for to be associated with these control signals. So let me say C_6 is the bit which is associated with output enable of R_2 and say C_7 is the bit which is associated with load of R_1. So accordingly I will have bits C_6, I will also have bits C_7. Suppose when this is decoded, the external address generator that generates an address. Let us give a logical level say address L, L can have any value. So the logical address that is generated is address L. So the control signals that will be required during T_3 for execution of this instruction move R_1, R_2 must start from location L in the control memory.

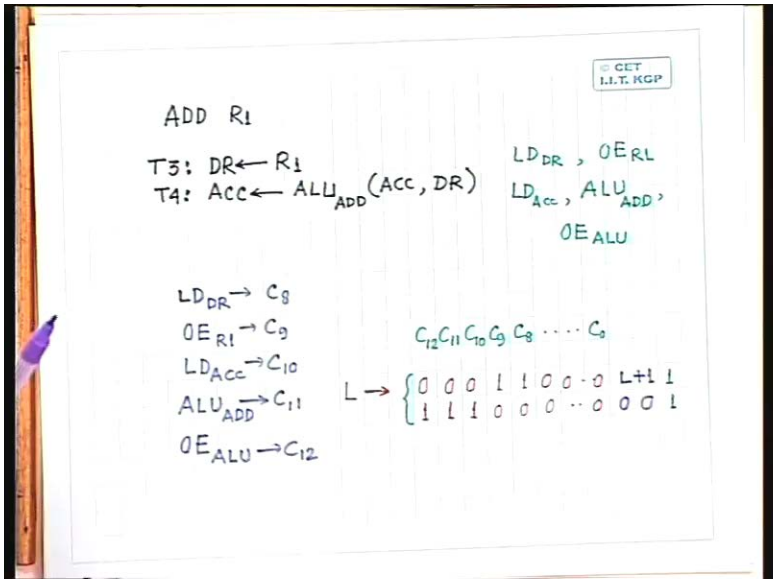
So suppose this is location L in the control memory, so in this location what I have to put because I have to generate C_6 and C_7 so these two bits have to be made equal to one and the rest of the bits will be equal to zero. So I will set in location L, C_7 equal to 1. I will also set C_6 equal to one. The rest of the bits will be equal to zero because these control signals will not be required during T_3 while execution of this move R_1, R_2 instruction. What will be the status of the other bits? Once this instruction is executed then I have to go back to instruction fetch cycle that means time state T_0 and for time state T_0 onwards, the opcodes are stored starting from location 0. So that means in the branch address location I have to put the address 0 0 and the mode field has to be made equal to 1. So I set the mode field equal to 1 and in these cases, these bits will also be 0 because these control signals will not be generated during these machine states.

So find that after generating these control signals, this execution of the instruction move R_1, R_2 is complete. So after completing execution of this instruction, the control memory address register will be loaded with value 0 0 because the mode field I have set to 1 and as the control memory address register is loaded with value 0 0, the next address in the control memory that will be read is this one. Here I generate the control signals C_1 and C_0 which are meant for load memory address register and output enable of the program counter. That means I am going back to next instruction fetch cycle and these things will repeat.

Next time this L, the value of L will depend upon which instruction has been fetched and put into instruction register and the decoder, instruction decoder will decide the value of L. So this is how I can write a software so I can put it this way that this is nothing but a software for execution of this instruction itself. So I have a set of assembly language instructions for execution of a program and every instruction itself is executed by a program which I call as a micro program. So you will find that for this simple situation move R_1, R_2 the micro program was a single micro instruction. So this I can call as a micro instruction, so it is a single micro instruction.

Whereas for execution of some other programs, maybe I need more than one micro instructions. I think we had taken some such example where unit more than one. The example that we have taken is add.

(Refer Slide Time: 00:35:25 min)



ADD R_1

T3: $DR \leftarrow R_1$ — LD_{DR}, OE_{R1}

T4: $ACC \leftarrow ALU_{ADD}(ACC, DR)$ — LD_{Acc}, ALU_{ADD}, OE_{ALU}

$LD_{DR} \rightarrow C_8$

$OE_{R1} \rightarrow C_9$

$LD_{Acc} \rightarrow C_{10}$

$ALU_{ADD} \rightarrow C_{11}$

$OE_{ALU} \rightarrow C_{12}$

$$\begin{array}{ll} & C_{12}\,C_{11}\,C_{10}\,C_9\,C_8 \cdots\cdots C_0 \\ L \rightarrow & \left\{\begin{array}{l} 0\;0\;0\;1\;1\;0\;0\cdot 0\quad L+1\;\;1 \\ 1\;1\;1\;0\;0\;0\cdots 0\;\;0\;0\;1 \end{array}\right. \end{array}$$

One of the instructions that we had is add R_1. In add R_1 the micro operations that had to be done is during machine state T_3, the content of R_1 has to be loaded to data register then during machine state T_4 accumulator is to be loaded with sum of accumulator and data register. So that we had put in this way, ALU performing add operation on accumulator and data register. So for this operation, I need a micro instruction and the control signals that are needed are load data register and output enable of R_1. For this micro operation to be executed during machine state T_4, the control signals which are required are load accumulator. Then we need ALU add control signal because when this control signal is made equal to 1 then only ALU will perform add operation.

Simultaneously we also need output enable of ALU because from the ALU, output of the ALU is going to the common data path. So I also have to activate output enable of ALU. So if I say that this load data register within this assignment, I put assign load data register to control signal say C_8, output enable of R_1 to C_9, load accumulator to C_{10}, ALU add because I am not used any of these control signals before. In the control signals that we have considered so far these control signals are not considered. So I have to have new bits for generation of this control signals. So for ALU add I need C_{11} then output enable of ALU is C_{12}. So I put it this way, I have to have bits C_8, C_9, C_{10}, C_{11}, C_{12}. Then during machine state T_3, the bits which are needed are C_8 and C_9. So I put these two bits equal to one. The remaining bits will be zero.

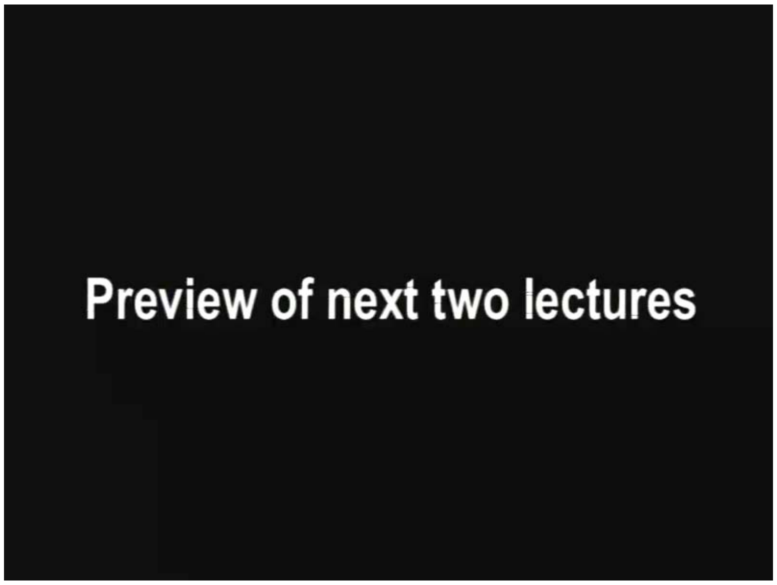
Similarly on this side, other bits will be also zero. So it starts from bit number C_0, so all these bits will be equal to zero. During machine state T_4, I need the control signals load accumulator, so that is C_{10} which will be equal to 1. I need ALU add control signals C_{11} that will also be equal to one. Output enable of ALU that is also equal to one. The remaining bits will be equal to zero.

So you will find that for execution of this instruction add R_1, I need two micro instructions. So it is a program micro program consisting of 2 micro instructions and this will be the value of L. So L will now be this value which has been generated by the external address generator. Similarly we can also fill up these values. Here what will be the next address field in this particular case? Next address field will be L plus 1 and the mode field will be equal to 1 and after that next address field will be 0 0, mode field will also be equal to... Why zero? Mode field will be equal to 1 because the next address field will come from here only. So this is a micro program which executes the instruction add R_1. So I can also put it this way that a task is executed by a program which is a set of instructions. Every instruction itself is executed by a micro program which consists of a number of micro instructions.

So what is the advantage that we gain if I go for micro programming? You find that because this whole thing is loaded in a control memory and the control memory can be programmed as per requirement. So if I find that today I decide that the control signals are to be generated in some sequence but next day I find that the sequence in which I have generated the control signals they are not proper. I should change the sequence of control signals. So if I want to change the sequence of control signals, when I have the hardware control circuit generator then the entire hardware has to be changed. Whereas if I do it using the micro programming where the sequence in which the control signals are to be generated and how the control signals are to be generated because that is a program loaded into a control memory keeping every circuit same. If I simply change the control memory, the sequence of generation of the control signals will be different. So I have more flexibility.

Of course you can question that when it is a design of CPU, I cannot change the control memory within the CPU any time but still in that case, it is the design flexibility that you have. But does it have only a advantage? Speed wise this will be slow because for generation of any control signal, I have to read the memory content. So because it is software in nature, it will be slower than hardware control unit but the advantage is it is more flexible, gives flexibility while designing and the second advantage is it is more compact because the whole lot of hardware control circuit is now put in just the control memory. So it is compact and flexible but speed wise it will be slower. It will be more accurate in that. That accuracy does not matter much. So with this we take a break.

(Refer Slide Time: 00:44:34 min)


Preview of next two lectures

(Refer Slide Time: 00:44:38 min)

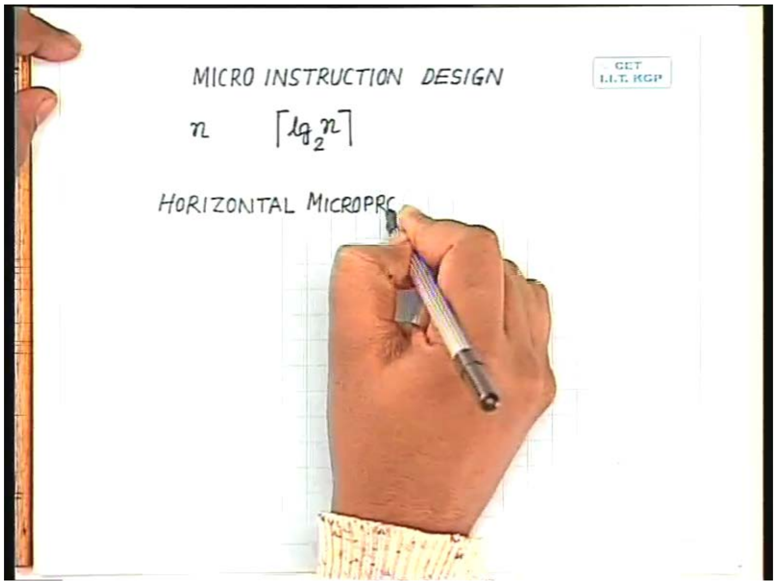
Lecture # 06

Microprogrammed Control -II

Now let us see another aspect that is designing of micro instructions. Till now we have seen that in the control memory, every location will have the number of bits which are same as the number of control signals that you have within the system. Now in any system I can have hundreds of control signals that means our control memory every location must have hundreds of bits which makes the control memory very big. So we have to think, is there any way by which the number of bits in every location in the control memory can be reduced.

One of the simplest way is instead of directly putting the control signals in the control memory, you encode the control signals and put the encoded control signal in the control memory.

(Refer Slide Time: 00:45:51 min)


MICRO INSTRUCTION DESIGN

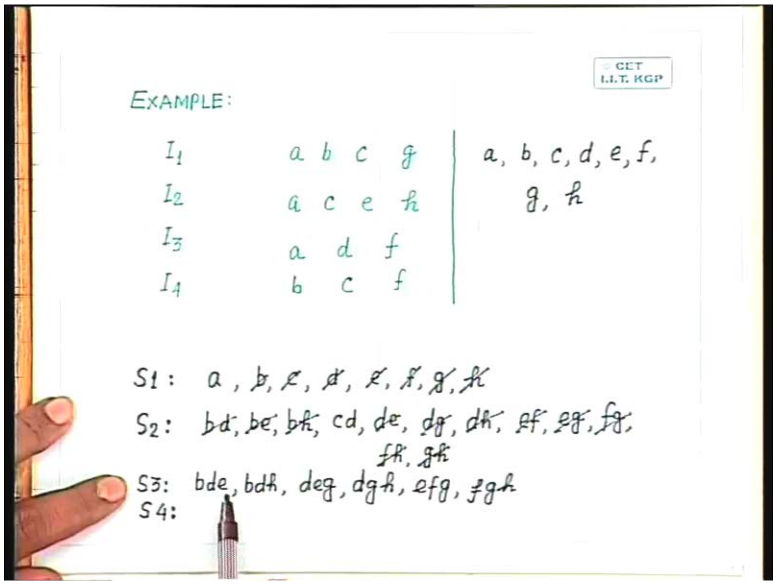
$n \quad \lceil \lg_2 n \rceil$

HORIZONTAL MICROPR

So that way if I have n number of control signals present in a system, the number of bits in every location in the memory that we will need is log of n to the base 2 and then ceiling function of this. So that will be the number of bits needed in every location in the control memory but this has a disadvantage. That is while analyzing the instructions that we have seen that in any cases, we need more than one control signals to be activated simultaneously. Now when we are encoding the control signals and putting them as encoded bit stream in the control memory that means after reading a location from the control memory to generate the control signals, I have to decode that encoded bit stream, so that has to pass to a decoder.

In case of a decoder generating more than one decoder output active, simultaneous case is not possible. So that puts a restriction that if we have fully encoded bit stream to represent the control signals. I can generate only one control signal at a time. I cannot generate more than one control signal at a time. So these are two extreme cases. One case is where every bit is assigned to a control signal which we call as horizontal microprogramming. This is called horizontal microprogramming. So let us take an example to see how the design can be done. I take a very simple example with only say 4 micro instructions, so let us take an example.

(Refer Slide Time: 00:47:53 min)



CET I.I.T. KGP

EXAMPLE:

I_1	a b c g	a, b, c, d, e, f, g, h
I_2	a c e h	
I_3	a d f	
I_4	b c f	

S1: a, ~~b~~, ~~c~~, ~~d~~, ~~e~~, ~~f~~, ~~g~~, ~~h~~

S2: ~~bd~~, ~~be~~, ~~bh~~, cd, ~~de~~, ~~dg~~, ~~dh~~, ~~ef~~, ~~eg~~, ~~fg~~, ~~fh~~, ~~gh~~

S3: bde, bdh, deg, dgh, efg, fgh

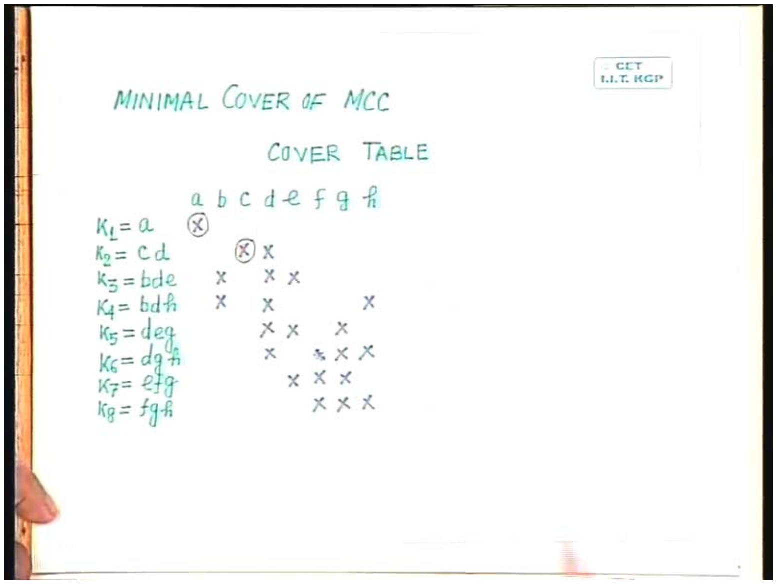
S4:

So here I consider 4 micro instructions say I_1, I_2, I_3 and I_4. So suppose we have these 4 micro instructions and the control signals which are activated by I_1 or let us say a b c and g. The control signals which are activated by I_2 let us say those are a c e and h. Control signals which are activated by I_3 let us say those are a d and f and the control signals which are activated by I_4 let us say those are c b c and f. So you will find that for this example, the total number of control signals that we have are a, b, c, d, e, f, g and h. These are the total control signals that we have for this example, out of which a_1 activates a b c and g, I_2 activates a c e and h, I_3 activates a d and f, I_4 activates b c and f.

Now before designing the micro instructions for this particular example let me define one more term that is maximal compatibility class. So for that what I will do? I will take every compatibility classes from step 2 or we have compatibility classes containing, all the compatibility classes containing two control signals each and try to insert another control signal in that still maintaining the compatibility property. So if I do that in step s_3, you will find that I will have a number of compatibility classes like bde, bdh, deg, dgh, efg and fgh. So by analyzing this, I can find out that. So again once I find out the compatibility classes containing three control signals each, again I remove from the previous step all the compatibility classes which are subset of some compatibility class in step s_3.

So here you find that except cd, all other compatibility classes can be removed because they are subsets of some compatibility class in step number s_3. So once I have this s_3 then I should go for next step s_4 where we will try to find out compatibility classes containing 4 control signals each and here you will find that I cannot insert any other control signal with any of the classes in s_3. So for example bde, with bde I cannot include a, with bde I cannot include, b is already there so I don't have to consider that. I cannot include c because b and c they become incompatible. I will try to find out the columns containing only one class. If I get any column which contains only one class that indicates that the row where the class is present that is the only maximal compatibility class which contains that control signal. There is no other maximal compatibility class containing the same control signal. Now what is our aim?

(Refer Slide Time: 00:53:14 min)



MINIMAL COVER OF MCC

COVER TABLE

	a	b	c	d	e	f	g	h
$K_1 = a$	⊗							
$K_2 = cd$			⊗	x				
$K_3 = bde$		x		x	x			
$K_4 = bdh$		x		x				x
$K_5 = deg$				x	x		x	
$K_6 = dg^{f}h$				x		x	x	x
$K_7 = efg$					x	x	x	
$K_8 = fgh$						x	x	x

Our aim is I will try to remove some of the maximal compatibility classes to get the minimal cover such that the minimal cover will contain all the control signals. Now while trying to eliminate some of this maximum compatibility classes, these are the compatibility classes such that corresponding to that in a column, I have only one class. Those compatibility classes cannot be removed because if I remove that compatibility class in that case those control signals will also be removed. So in a minimal cover, those compatibility classes must be retained. So these are the compatibility classes which are called essential compatibility classes and in our minimal cover, we must retain the essential compatibility classes.

So you will find that by studying this, I have two columns. The column corresponding to a and the column corresponding to c. These are the two columns which contain a single class.