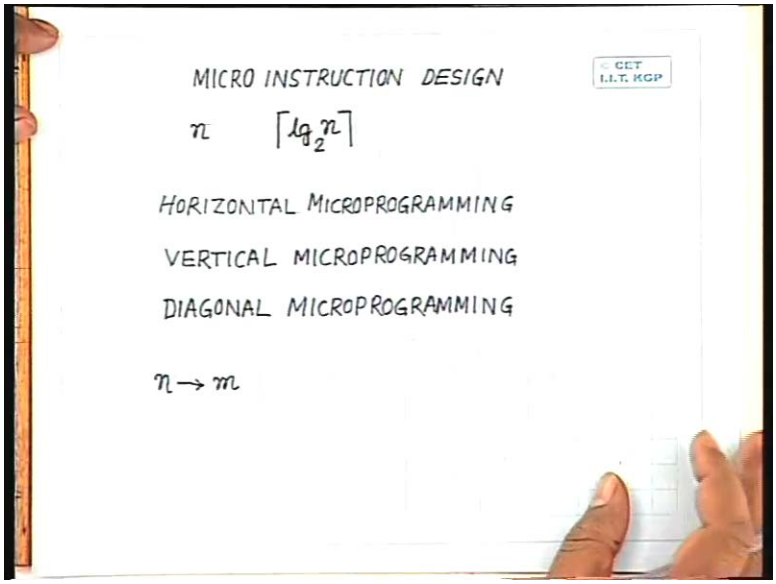
Digital Computer Organization
Prof. P.K. Biswas
Department of Electronic & Electrical Communication Engineering
Indian Institute of Technology, Kharagpur
Lecture No. # 06
Microprogrammed Control -II

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 signals in the control memory. 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.

(Refer Slide Time: 00:02:10 min)



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 through a decoder. In case of a decoder generating more than one decoder output active simultaneously 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

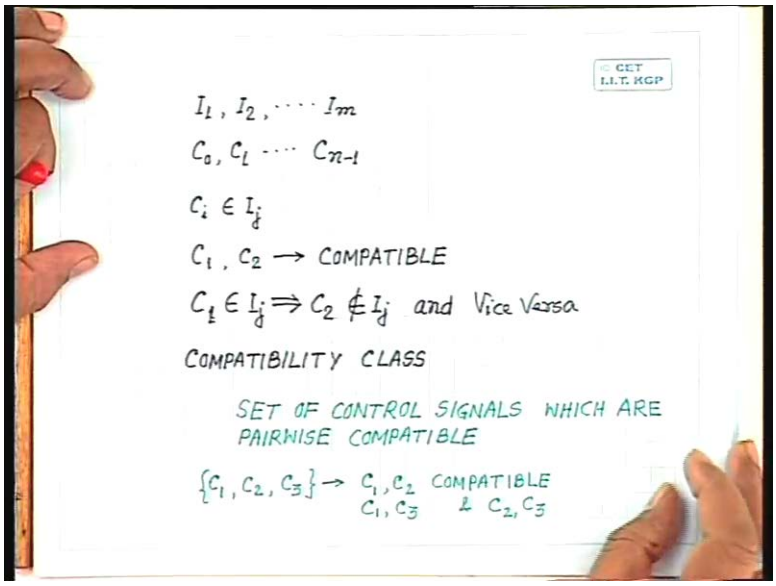
control signal at a time. I cannot generate more than one control signal at a time, so these are two extreme cases.

One case is where every bit is assigned to a control signal which we call as horizontal microprogramming. This is called horizontal microprogramming. In the second case when the control signals are fully encoded and the encoded bit stream is stored in the control memory that is called a vertical microprogramming. So these are the two extreme cases of microprogramming. One is horizontal and the other one is vertical microprogramming. So none of these are suitable for our purpose. So what we need is something in between that is we should try to reduce the number of bits in the control memory.

Simultaneously we should also be able to generate the required number of control signals in parallel. So what we need is something in between which we can call as diagonal microprogramming. So in case of diagonal microprogramming, what we have to do is we have to group the control signals in such a way that in a particular group, no two control signals will be activated simultaneously. Whereas if I need two control signals to be activated at a time then those two control signals must belong to two different groups. Within every group, the control signals are fully encoded. So if this n number of control signals that we put into say m number of groups then I need m number of decoders. Within every group the control signals are fully encoded and because they belong to different groups and every group has a corresponding decoder. So from every decoder I can generate one control signal and because I have more than one decoders, so more than one control signals I can generate at a time.

So here I can reduce the number of bits but not as less as this, as in case of vertical microprogramming. So the number of bits in this case will be more than the number of bits needed in case of vertical programming but it will be less than the number of bits needed in case of horizontal microprogramming. So I compromise on the number of bits to attend parallel activation of the control signals. So how many fields you should have and in every field, how many bits you should incorporate that is the topic of micro instruction design. So let us see how we can design the micro instructions.

(Refer Slide Time: 00:07:40 min)

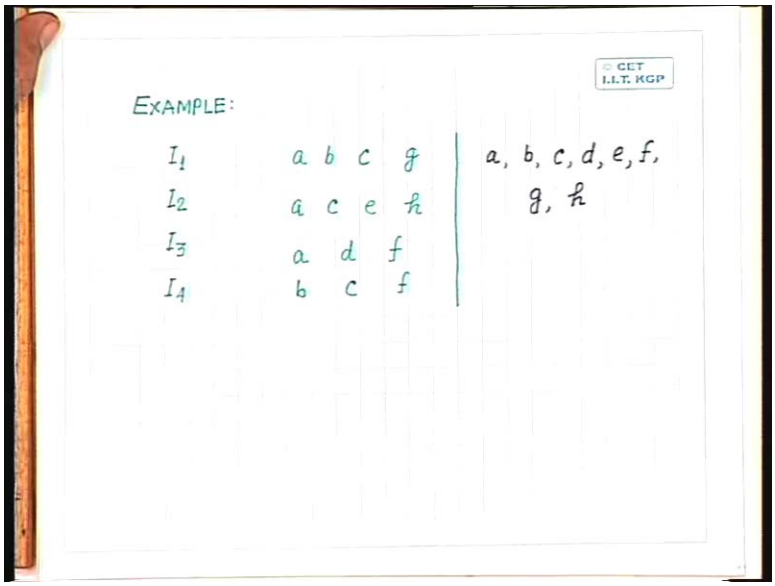


So let us assume that in a particular system, we have m number of micro instructions which are designated as I_1, I_2 to I_m. So these are the number of micro instructions. Micro instructions will be identified by analyzing the instruction that you have in the CPU. Let me assume that I have n number of control signals say C_0, C_1 up to C_{n-1} or I can also rename as C_1 to C_m that does not matter. So I have m number of micro instructions and I have n number of control signals. So whenever any micro instruction is executed, every micro instruction will activate one or more of this control signals. In some cases it may be needed that a micro instruction generates only one control signal. Some of the micro instructions may generate more than one control signal. So if it generates more than one control signal, all those control signals are to be activated simultaneously and that we have to consider while designing the micro instructions. We say that a control signal C_i belongs to a micro instructions I_j, if the micro instructions I_j activates control signal C_i. So that way more than one control signals can belong to a micro instruction I_j because I_j may activate more than one control signals.

We also say that given two control signals say C_1 and C_2, we say that these control signals C_1 and C_2 to be compatible. We say these two control signals C_1 and C_2 to be compatible if C_1 belongs to I_j implies C_2 does not belong to I_j and vice versa. That is C_1 belongs to I_j implies C_2 does not belong to I_j and C_2 belongs to I_j implies C_1 does not belong to I_j. If this is true then we say that this control signal C_1 and C_2 they are compatible. So once you define the compatible control signals, we can define what is called a compatibility class. So compatibility class is defined as a set of control signals such that the control signals within that set are pair wise compatible. So compatibility class will define as a set of control signals which are that is if say control signals C_1, C_2 and C_3, this forms a compatibility class. This indicates that C_1, C_2 are compatible, C_1 C_3 is also compatible and C_2 C_3 they are also compatible. So is that compatible that the definition that you have given for the compatibility C_1 C_2? So compatible for only one instruction I_j or another instruction? With respect to one instruction they are compatible. May be for an instruction I_j they are compatible.

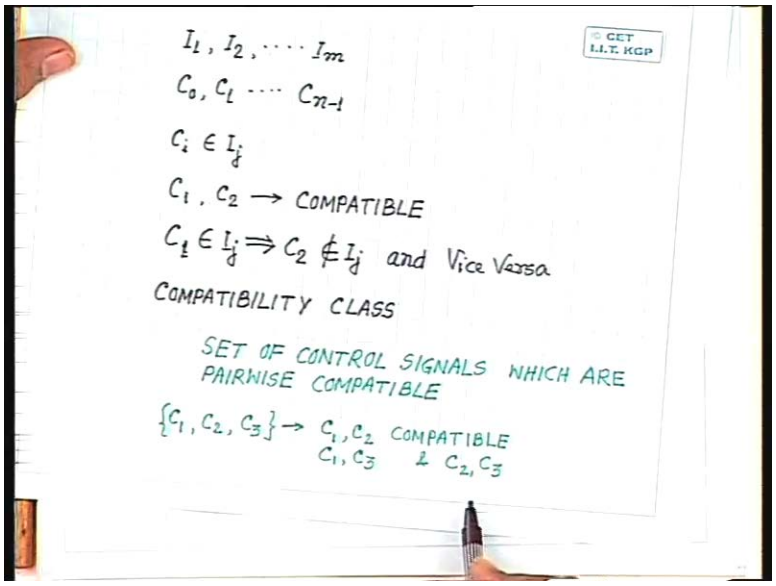
At first or any other instruction, for any other instruction they may not be compatible. So what we have to do during the design is we have to identify the set of control signals which are compatible with respect to every instruction. So when I say that a set of control signals forms a compatibility class that means the control signals within that set are pair wise compatible. So if I take any pair of control signals within that set, they must be compatible. That is what is a compatibility class. Here we have to identify the control signals which are compatible with respect to every micro instruction. So that is what we have to do while designing. So in this case this design can be performed in an analytical way. 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: 00:15:18 min)



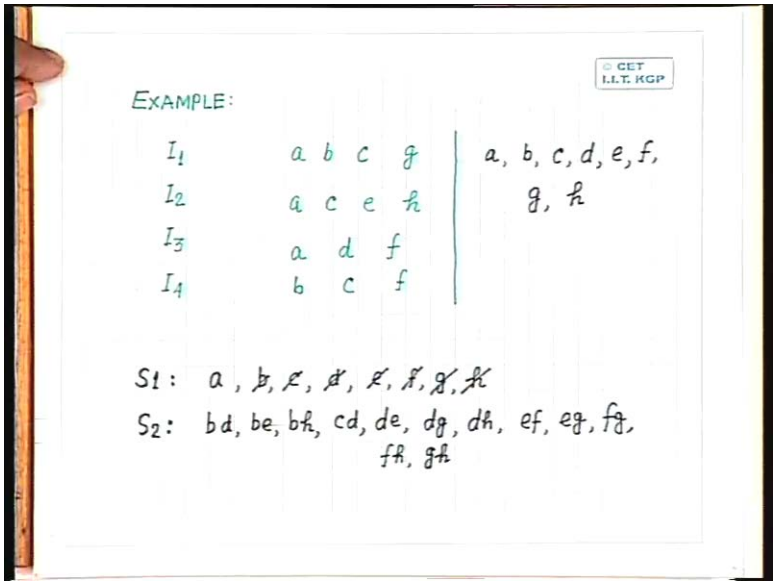
So here I consider 4 micro instructions say I_1 I_2 I_3 and I_4. So suppose we have these 4 micro instructions and the control signals which are activated by I_1 are let us say a b c and g. The control signals which are activated by I_2 let us say those are a c e and h. Control signals which are activated 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 b c and f. So we 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 I_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 here we have defined what is the compatibility class.

(Refer Slide Time: 00:17:44 min)



I define another term which is maximal compatibility class. So we will say that maximal compatibility class is a compatibility class to which no other control signals can be included without introducing incompatibility. So it is a compatibility class. We have said a compatibility class as the set of control signals which are pair wise compatible. So if I have a compatibility class such that I cannot insert any other control signal to the same class without introducing an incompatibility then that particular compatibility class is called a maximal compatibility class that means that compatibility class cannot be expanded further by introducing new control signals.

(Refer Slide Time: 00:19:20 min)



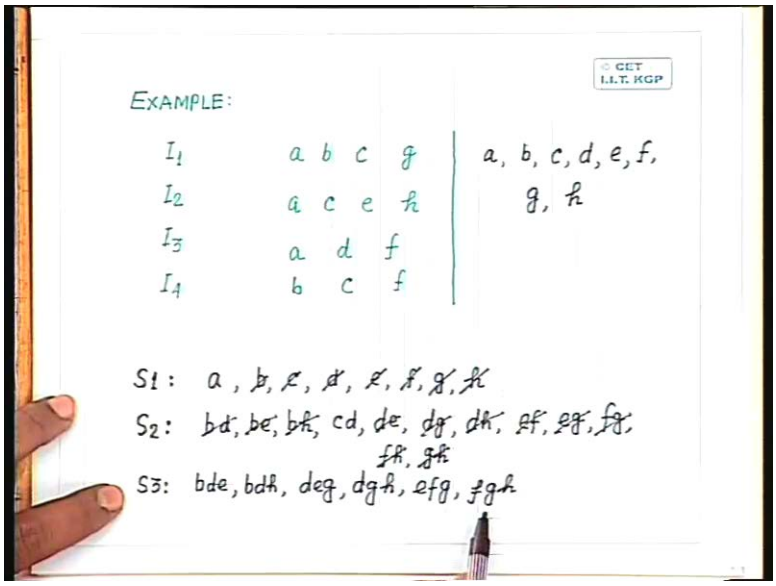
So it should be quite obvious that our design should try to find out maximal compatibility classes because only when I have a maximal compatibility class then only I can guarantee that I can minimize the number of bits. If the classes are not maximally compatible then minimization of the number of bits is not guaranteed. So we will try to design the micro instructions keeping in mind that we have to find out the maximal compatibility classes. Now how do you do it? We do it hierarchically. We follow different steps say in step number 1 or S_1 I assume that I have defined compatibility classes but every compatibility class consists of only one control signal. So if a class is having only one control signal that does not valid our compatibility class definition because there is no other control signal. So there is no question of incompatibility. So in step number one, I consider that all these control signals as independent classes. So I have a b c d e f g and h, so each of them forms a compatibility class by itself.

In step two I too try to find out the compatibility classes containing two control signals. For that what I do is I take every control signal one after another and try to find, try to pair that with some other control signal in such a manner that compatibility is not violated. So first I will take the control signal a and try to see whether along with a, I can put some other control signal or not. So that still that pair will remain compatible. So here you find that if I take a, I have to see whether b can be paired with a. It cannot be paired because I_1 activates both a and b. Can c be paired? No. Can d be paired? No, because I_3 needs both a and b simultaneously. The e cannot be paired because I_2 activates both a and e simultaneously, f cannot be paired, I_3 activates both a and f simultaneously, g cannot be paired because a and g they are activated simultaneously by I_1. And h again it cannot be paired because I_2 activates both a and h simultaneously.

So with a I cannot pair any of the control signals. Let us see whether with b I can pair any of the control signals or not? bd, be, bh. What else? I think with b, no other control signal can be paired. Then I can have a pair of cd, I can have a pair of de, I can have pair dg, I can have pair dh, I can have pair ef, I can have pair eg, I can have pair fg, I can have pair fh and I can have pair gh. So these are the compatibility classes containing two control signals each. So once I form the compatibility classes containing two control signals each, from the previous step I removed all the control signals which are subset or the compatibility classes which are subset of some compatibility class at the next level. So here you find that except a, all others are subsets of some compatibility class in step 2. So I will retain only a and remove all other control signals from here.

Then I go to next step 3. In step 3 my objective is to try to find out compatibility classes containing three control signals. 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 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.

(Refer Slide Time: 00:23:46 min)



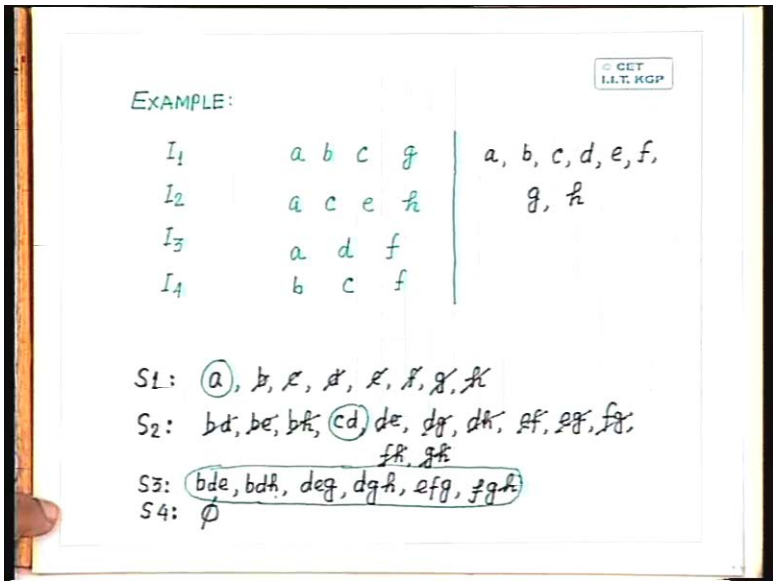
So again once I find out the compatibility classes containing three control signals each, again I remove from the previous step all the compatibility classes which are subset of some compatibility class in step S_3. So here you find that except cd, all other compatibility classes can be removed because 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 four control signals each and here you will find that I cannot insert any other control signal with any of the classes in S_3. Say for an 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, d is already there. What about f? d and f? They are activated simultaneously by I_3. So I cannot include f in this because in that case df will become incompatible. Can I include g? No, because b and g they are activated simultaneously. Can I include h? h, I can include so that comes at the next one. So I have these two bde and sorry, with bde here, I am trying to find out compatibility classes with four. So can I include h with this? e and h they are activated simultaneously by I two, so I cannot include any other control signal with bde.

So similarly if you analyze all of this, you will find that I cannot generate any compatibility class having four control signals. So at this step four, this will be phi because I cannot generate any compatibility class with four control signals. So now we find that after forming this, these are the maximal compatibility classes that I have. a is a maximal compatibility class because we have seen that I cannot pair any other control signal with a, still maintaining compatibility. So it is a maximal compatibility class. cd is again a maximal compatibility class because I cannot pair any other control signal with cd still maintaining the compatibility property. Then all these also become a compatibility class. each of them is a compatibility class and they are maximal. So I have 1, 2, 3, 4, 5, 6, 7 and 8; 8 maximal compatibility classes.

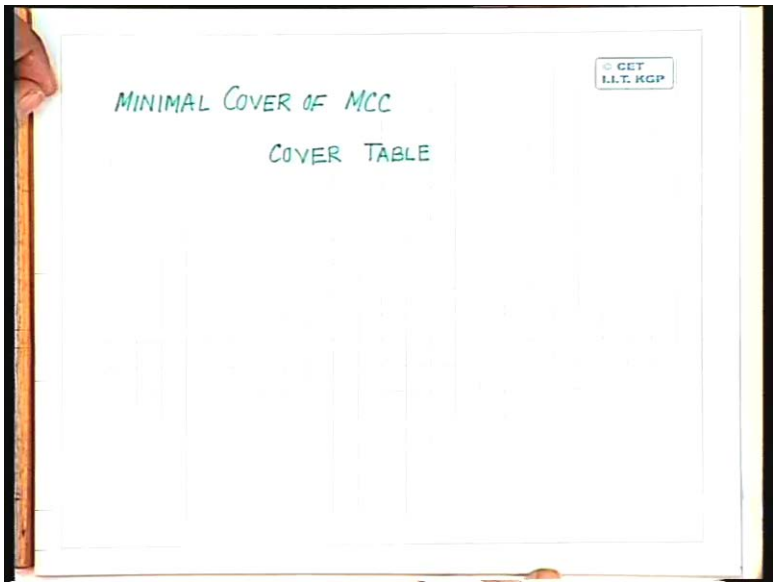
So once I find out this compatibility classes, I complete one stage of my design because now I have to encode these compatibility classes as different fields. The next stage is I have to find out whether all these maximal compatibility classes or needed or not because you find that many of the control signals are common in different compatibility classes.

(Refer Slide Time: 00:28:58 min)



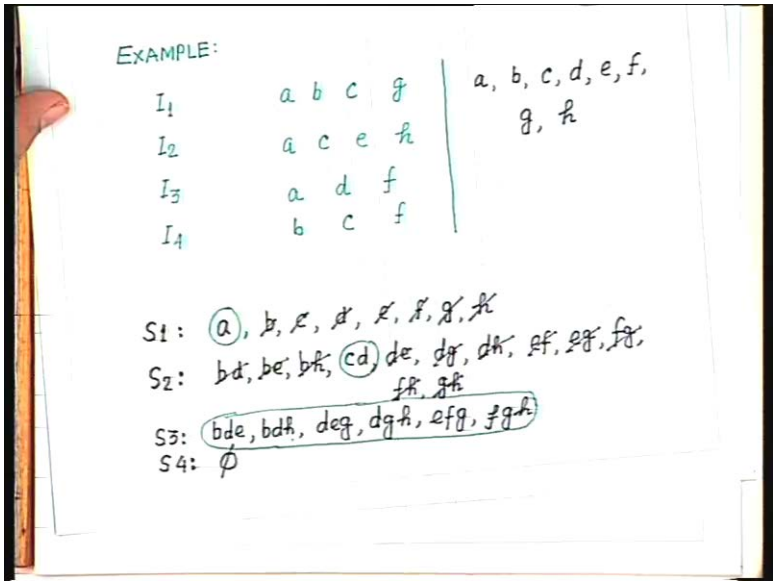
B appears here, b also appears here, d appears here d also appears here, h appears here h also appears here. So though I have obtained the maximal compatibility classes but it is quite natural that may be all these maximal compatibility classes are not needed to be coded. If I can find out a subset of all these set of maximal compatibility classes which will include all the control signals.

(Refer Slide Time: 00:30:00 min)



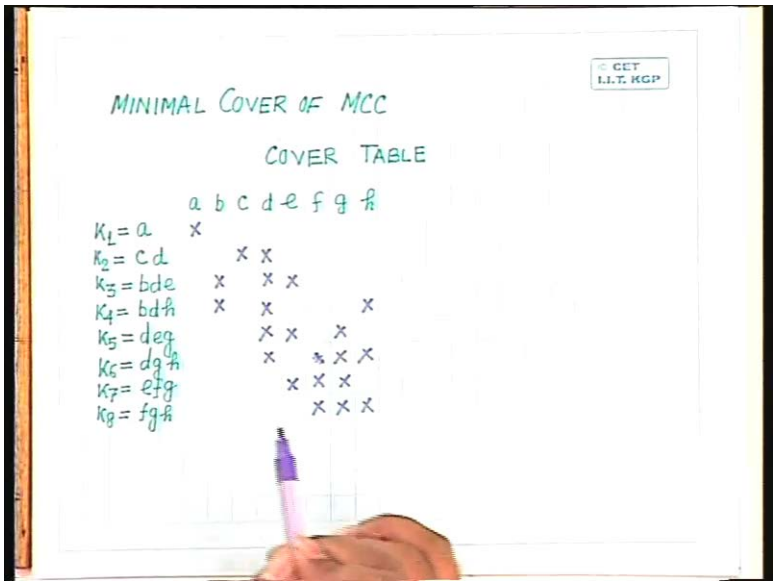
Then that subset I can encode to generate my desired control signals. So this is a subset which is called a minimal cover. So what we have to find out is a minimal cover of this set of maximal compatibility classes. So what we have to find out is minimal cover of the maximal set of compatibility classes. So I have to find out of maximal compatibility classes that let us put as MCC, MCC stands for maximal compatibility classes. So we have to find out minimal cover of maximal compatibility classes. So once I find out the minimal cover that is a minimal subset of the set of maximal compatibility classes which includes all the control signals and we will try to include only those control signals with every field corresponding to every maximal compatibility class in the minimal cover. Now this can be done by using a tabular method. We will use a table which is called a cover table. What is this cover table? In this cover table for every maximal compatibility class we will have a row and for every control signal we will have a column.

(Refer Slide Time: 00:31:42 min)



So with the help of this maximal compatibility classes that we have generated, I find that because there are 8 maximal compatibility classes, so in the cover table I have to have 8 rows. Those rows we designate as say k_1, k_1 means it is a maximal compatibility class containing the control signal a only.

(Refer Slide Time: 00:32:00 min)



Then we will have k_2, it is a maximal compatibility class containing the control signals c and d. We have a row k_3 containing the control signals b d and e. We have to have k_4 containing the control signals b d and h. We have k_5 containing the control signals d e and g. We have k_6 containing the control signals e f g or dgh, the next one d g and h. We have to have k_7 containing the control signals efg. The next one is efg and we have another row for compatibility class k_8 containing the control signals f g and h. So these are the number of rows that we will have. Again we will have a column for every control signal.

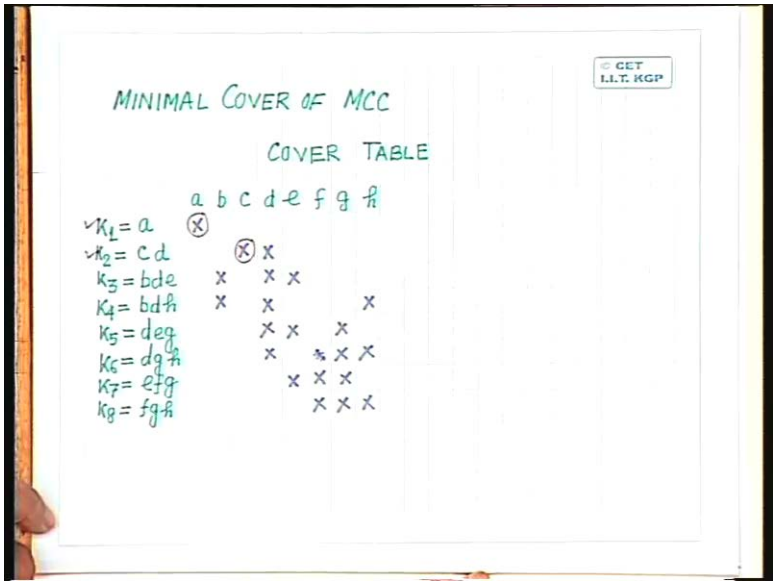
So for a we will have one column, for b we will have another one, c d e f g and h. So I have columns corresponding to every control signal and I have rows corresponding to every compatibility class. Now this table has to be filled up like this. For every row wherever a control signal corresponding to a given column is present, I will put a cross in the corresponding location. So for k_1, row k_1 which contains only the control signal a I will put a cross in this location. So this row corresponding to k_1 and column corresponding to control signal a. For k_2 which contains control signals c and d, I will put crosses in both these column c and d. k_3 which contains b d and e, so b d and e. For k_4, it contains b d and h, so I will put b d and h. For k_5 I will put d e and g, so I will have crosses in d e and g. For k_6 it is d g and h, so I will have crosses d g and h. k_7, e f and g so I will have crosses e f and g, k_8 f g and h so I will have crosses f g and h. So I complete the cover table.

Now from this cover table I have to find out those maximal compatibility classes which forms a minimal set of maximal compatibility class. So how do I do it? I study this cover table, find out the columns. I study the columns, if there is any column which contains only one cross that means that is a control signal which is included only in the maximal compatibility class corresponding to the row where the cross is present. So I get a cover table. In this cover table I will try to find out the columns containing only one cross. If I get any column which contains only one cross that indicates that the row where the cross 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?

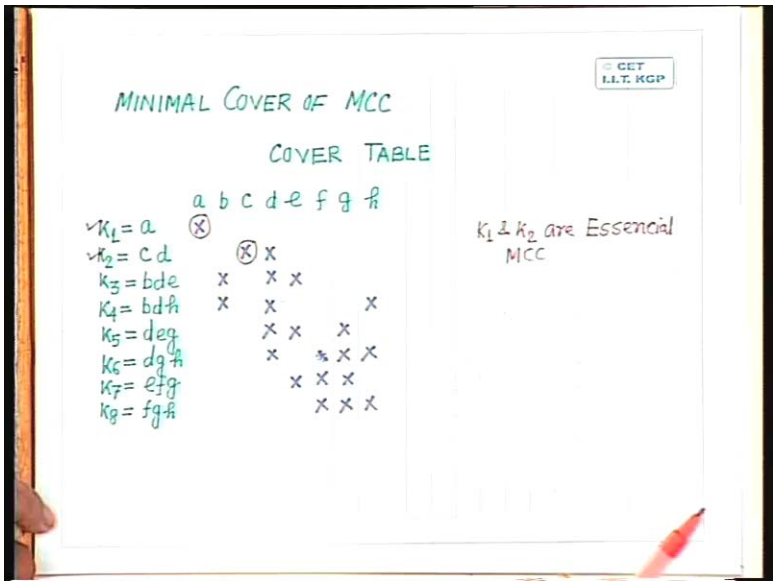
Our aim is I will try to remove some of the maximal compatibility classes to get the minimal cover such that the minimal cover will contain all the control signals. Now while trying to eliminate some of this maximal compatibility classes, these are the compatibility classes such that corresponding to that in a column I have only one cross. 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 the minimal cover, those compatibility classes must be retained. So these are compatibility classes which are called essential compatibility classes and in our minimal cover we must retain the essential compatibility classes.

(Refer Slide Time: 00:38:30 min)



So you 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 cross that means the corresponding compatibility classes k_1 and k_2, these are essential compatibility classes. So because these are essential compatibility classes they must be retained in our minimal cover.

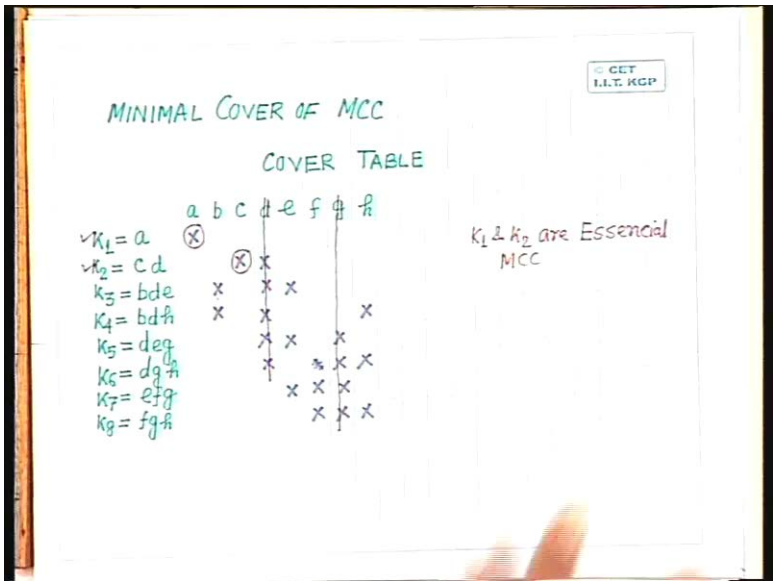
(Refer Slide Time: 00:39:07 min)



So k_1 and k_2 are essential MCC's. So these two k_1 and k_2 must be retained in the minimal cover. Other members of the minimal cover now we will try to find out. For that you have to follow few steps. Again by studying this compatibility class, I mean by studying this cover table, first you have to identify if there are any columns which are identical. If there are more than one columns that are identical that means the corresponding control signals are also identical. So in such cases I will remove all such columns except one which are identical. So I will retain only one column and the rest of the columns which are identical to that will remove because all those control signals are identical. So by studying this cover table, you find that there is no such situation. There are no columns more than one columns which are identical. So I cannot remove any of the columns like that.

Next what we have to do is we have to again by studying the columns, we have to find out if there is any column which is a subset of other column. So if there is any column which is a subset of some other column then the column which is subset that is called a dominated column and the column of which it is subset that is called a dominating column. So if there is any dominating column in the cover table, I remove that column.

(Refer Slide Time: 00:41:23 min)



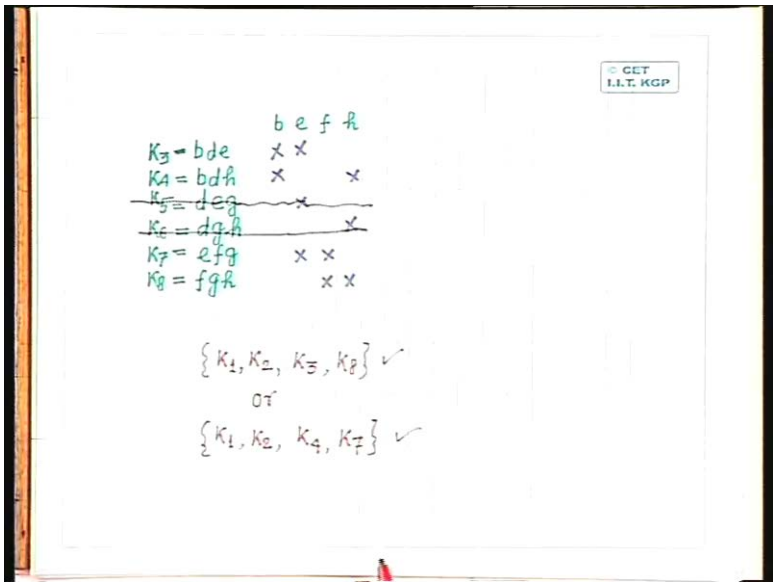
So you find that by studying this cover table, I have these two columns b and d where b is dominated column and d is the dominating column because for every cross in column b, there is a cross in the same row in column d. So for this I have one cross here, for this also I have one cross here. So what I try to do is I remove the dominating column. So this is a column d which dominates column b. Is there any other such column? f and g. g is the dominating column which dominates over the column f. So I also remove this column g from the cover table because g is the dominating column which dominates over f. I also removed d which is a dominating column which dominates over b.

So once I remove this, after that what I do is I form a reduced cover table because I have removed some of the entries. So what will be the reduced cover table now? We have seen that the MCC's k_1 k_2 they are essential. So I have to retain them in the minimal cover. So in the reduced cover table I will have rows corresponding to the other MCC's. I will have row corresponding to k_3, I will have row corresponding to k_4. I will have row corresponding to k_5, k_6, k_7 and k_8 where k_3 is equal to bde, k_4 is bdh, k_5 is deg, k_6 is dgh, k_7 is efg and k_8 is fg and h. The columns that I will retain are b e f and h because a and c, they being corresponding to the essential MCC's, I have to any way incorporate that. So in the reduced cover table I removed that.

Similarly d and g we have removed them because those are the dominating columns. So the remaining columns I retain and the MCC's which are not essential that I retain in the reduced cover table. So after doing this, the reduced cover table looks like this one. So this becomes my reduced cover table. So once I have this reduced cover table, now as we have done in case of columns that the dominating columns we have removed. In the reduced cover table, I try to find out whether I have rows, some of the rows may be dominating some other rows. So in this case our approach will be that a row which is dominated that will be removed. In case of column, the column which is dominating that we have removed. In case of rows our approach will be reverse that is a row which is dominated that will be removed. So if you do that, you find that k_5 and k_6,

these two are dominated rows because in case of k_5 I have only one cross in column e which is also present in case of k_3. For k_6 I have only one cross in column h which is also present in k_8. So these two rows k_5 and k_6 they are dominated rows so I remove k_5 and k_6, because these are the control signals which are present in other MCC's. So I can as well remove that.

(Refer Slide Time: 00:43:12 min)



Now we have to select from the remaining k_3, k_4, k_7 and k_8 what are the MCC's that we should get in the minimal cover? So now if you study this, you find that I have two options. I can have k_3 and k_8 along with k_1 k_2 because then I cover all the control signals. If I just incorporate k_3 and k_8, I have b e f g because f and h are present in k_8 and b e are present in k_3. Similarly I can also have k_4 and k_7 instead of k_3 and k_8. Then also I cover all the control signals. So by minimal cover can be k_1 k_2, anyway I have to retain them because they are essential.

In addition to this I can have k_3 and k_8, so this can be one minimal cover or I can have k_1 and k_2 those being essential have to be retained in any of the minimal covers and the remaining two can be k_4 and k_7. So these are the two minimal covers that I can have. So now the micro instructions that we have to design can follow either this or this. So this indicates that in micro instructions, I have to have four fields. One field corresponding to k_1, the other field corresponding to k_2, other one corresponding to k_3 and k_8 so there are 4 fields. Here also I will have 4 fields corresponding to k_1, k_2, k_4 and k_7. So I can use any of this minimal cover to design my micro instructions and within every field, the bits or the control signals can be fully encoded. So this is the formal design approach which can be used for designing the micro instructions.