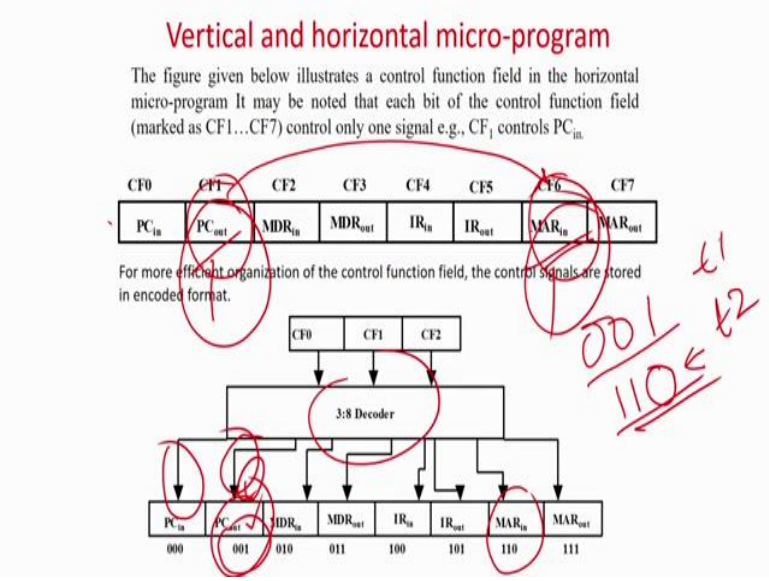
So, whenever we want to optimize based on encoding or compressing of the signals in each of these cells or each of the words in the program control memory, we call it as a vertical micro-program. In horizontal it is very flat and no optimization is there. So now, what way you can do the only way you can do is you have to encode encoded by mean many way xyz. So, whenever we want to optimize on sparse values of a matrix there is only one way to do it we have to encode it.

Now, encoding can be very flat like you if there are 16 bits as output, you go for a 4:16 decoder that is the one way of compressing them there can be several others as we look here. So, basically first the most formal or the preliminary way of doing it is basically encode the signals in the control memory. So, what do you do? So, whatever signals has to be applied you just encode them and use a decoder, I will show you a figure and then we will come back.

(Refer Slide Time: 52:27)



So, for example, this is one word of the control memory or this is the format of the control memory word. So, $PC_{in}, PC_{out}, MDR_{in}, MDR_{out}$ all these values are there.

Now, I encode it. So, how many are there? There are 8 bits which has to be controlled or 8 locations which has to control. So, I just put a 3:8 decoder then, what will be the memory size? The memory size will be 3 bits. So, if you give 000 the corresponding row of the decoder will be 1 and you generate $PC_{in} = 1$.

Next if you if you give the signal 001 this is the memory. So, memory is now 3 bit. So, if you give the value of 001 001 in the memory. So, you are going to make this second line high of the decoder, and you are going to get the value of 001 which actually makes this line high. So, in fact PC_{out} will be equal to 1.

So, now, you have to observe that this is one of the drastic way of compressing. So, instead of 2^n the size will just become n or if it is n you will go for log n upper ceiling. So, it is 8 you are going to get 3 highly compressed value we will have, but the problem is that at any point of time only one of these signals can be made 1. So now, the things will start becoming slower. Say for example generally we know that what happens basically we always have program counter out and memory address register in. So, this actually always happens simultaneously.

So, here what will be happen first you will have to give the value called 001. So, if you are giving the value 001 PC_{out} will be equal to 1. But now, simultaneously you cannot also make $MAR_{in} = 1$. So, what we have to do it will be in 2 steps in first step the memory control memory will give the value 001. Which will make $PC_{in}\ PC_{out} = 1$, then the program counter there will be some stable register or 1-bit flip flop which will hold the value, but it has to be a 1.

Next step what you do? Next step you give the value of 6 that is MAR_{in} you have to give the value 110 in the control memory; that means, you are going to this value will be fed from the control memory to the decoder. So now, it will make MAR_{in} equal to 1. This is say t1 and this is say t2 then, what it going to happen in two times that this configuration will be there, we see program counter output we will go to the memory address in.

Now, what happens now, but still now PC_{out} is also equal to 1 this is also equal to 1. Now, we have to give a time step where you reset actually both, in case of a horizontal micro-program you need not worry about resetting it or holding the value. Simultaneously you give this as 1 and this as 1 and your job is done, but here we are compressing it we are calling as a vertical micro-program because, we are compressing the structure of the memory. So, you can only make any one point one at a time. So, I make $PC_{in} = 1$ and at a time 1, then in the next time go if I make $MAR = 1$ in that case there should be some guy to remember that, PC_{out} was equal to 1 at that point of time.

Because, otherwise what is going to happen when I am making this as 1, this line will become a 0. Because whenever I am giving the signal 110 basically PC_{out} will become 1.

So, here there are 2 disadvantages one is that you have to you can you will have more number of steps required to give simultaneous 1s to the different control positions. So here two 1s are required so, you require 2. If multiple 1s are required you have that many number of time steps. Not only that whatever was one simultaneously required in a horizontal micro-program those values you have to remember like, $PC_{out} = 1$ and $MAR = 1$ together they should be 1, but here you are doing in 2 steps.

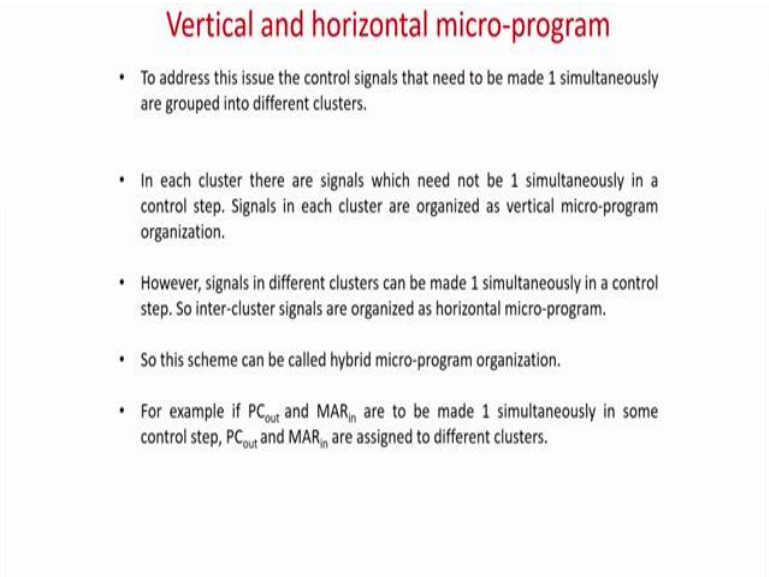
So, somehow the program counter out has to be 1 it has to be remembered till you go to the second step, and after the second step is done then again you have to reset these both and then again start from the third location and so forth. So, that is slight these are these are the 2 disadvantages longer time, you have to remember and again you have to reset. But, the advantage is that the memory size actually gets compressed in the value of in the terms of log from 2^n, it will go to the order of n. So, that is the big advantage.

So, if we have lot of 0s then such problems will become in such some kinds of number of stages will be less, but if we have more number of 1s then making them simultaneously 1 is not possible. So, the time steps will increase, but this vertical micro-program advantage comes because, they are all lots of 0s and therefore, only we go for the vertical technique. So, it is also very it is also slower compared to a horizontal micro-program, right?

So, now, we go to the theory. So, what it says that it encodes the signal as I have shown you. The output the control signals are stored in an encoded format in the control function field. The outputs of the control function fields are first get into a decoder and then they are applied to the ports. As we are having a decoder so, only one bit can be 1 at a time of the output of the decoder. So, only one control point can be a 1. And the control signals are divided into more number of control steps; obviously, because at each point of time only 1 bit can be a 1. So, more number of control steps will be required.

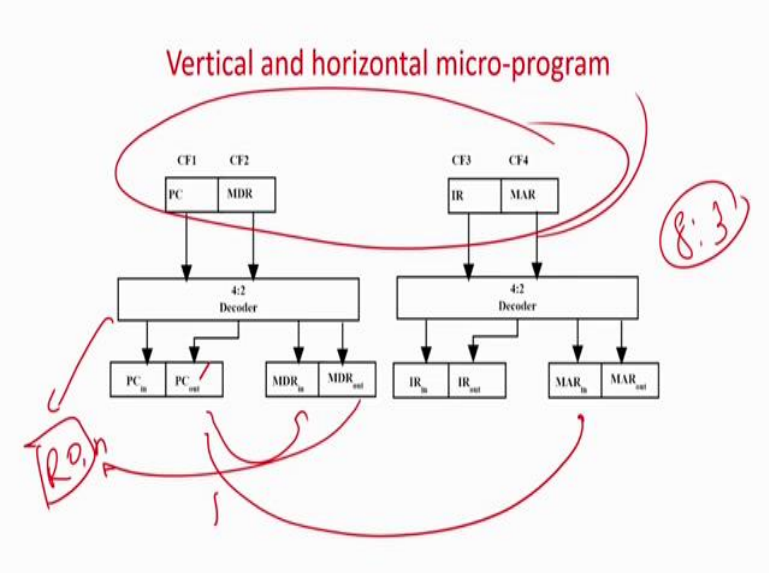
Vertical optimization require several micro instruction executions to do than one horizontal instruction can do it; that means, in vertical approach you require more number of micro instructions, in horizontal a very less number of field to because horizontally is parallel approach, and vertically is a sequential approach of solving the problem.

(Refer Slide Time: 57:30)



Now, that is one way of solving that is just optimize just full optimization, full vertical just encode it by a technique which we have shown you require a decoder to solve it, but now, there is another way as I told there are 2 techniques another way is very interesting that can I make a cluster? So, what is the cluster idea I will take a figure and then I will come back to the theory.

(Refer Slide Time: 57:53)



So, these are cluster so we have clustering. Now, we will not make fully means full compression. What we will know say it will try to find out which signals are simultaneously

required to be 1, like as I told you PC_{in} and MAR sorry PC_{out}, PC_{out} basically always requires to feed memory address register in that is generally we have observed.

So, what I do? I actually make a cluster 2 clusters. In 1 cluster I will put the PC_{out}, in another cluster actually I will put memory address register in. So, in if there are 2 different clusters basically. So, interestingly what is going to happen I can make this one 1 and this one also simultaneously, but as there are 2 different clusters means there is this is 1 control memory and this is another control memory now there are 2 cluster. I means we should not call it 2 different memory basically, logically we have partitioned into 2 different memory clusters. So, in this cluster only 1 bit can be a made 1 and, in this cluster also 1 bit can be made 1, but simultaneously 1 bit can be 1 here and 1 bit can be 1 here. So, that parallelism I have given. So, this is actually called a hybrid kind of a multi programing approach, neither it is fully horizontal neither it is fully vertical.

So, here what they have done they have made a cluster in which case if you look. They have made a cluster of $PC_{in}, PC_{out}, MAR_{in}, MAR_{out}$ one. Generally, if you see we always try to put in and out of a port in 1 cluster because, generally PC_{in} and PC_{out} you do not want to make it 1 simultaneously. Similarly, a memory address register memory address register in memory address register out together you do not require to be in.

But, generally we can have memory data register out and instruction register in simultaneously. So, this one I can make 1 and this one I can make together. Similarly, program counter out generally goes over here. So, that will way can make a cluster thinking out that which of the 2 bits can be simultaneously 1, that you put in different clusters and your job is done. So, by if I do it like this. So, you can see the hardware requirement is slightly higher. In the previous case we required a 3 one 8: 3 decoder, but here you require two 4:2 decoder. So, 2: 4 decoder, see higher side the memory number of bits were 3 in this case, but now, it will be 4 bits in this case.

But, this will be slightly faster compared to the full vertical case because, simultaneously I can take one signal here and one signal here. And we can make one at a time. So, if you make full horizontal then if you want to make only 1 bit in 1 cluster that is 1 extreme which is called the horizontal, and here if you think everything is in 1 cluster that is another extreme that is called the fully vertical micro program.

So, in this case there's something called a hybrid. So, the basic guideline is you choose the signals which very frequently has to be 1's. So, that you put in separate cluster. So, here I have given example of 2 clusters you can make multiple clusters depending on the needs, the motivation is in each cluster you should not put signals which requires to be 1 simultaneously. But, it may not be easily solvable because, if you go that go to that level that we are keeping individual bits in a cluster which can never be one together then we will go to a fully horizontal micro-program where each bit is independent.

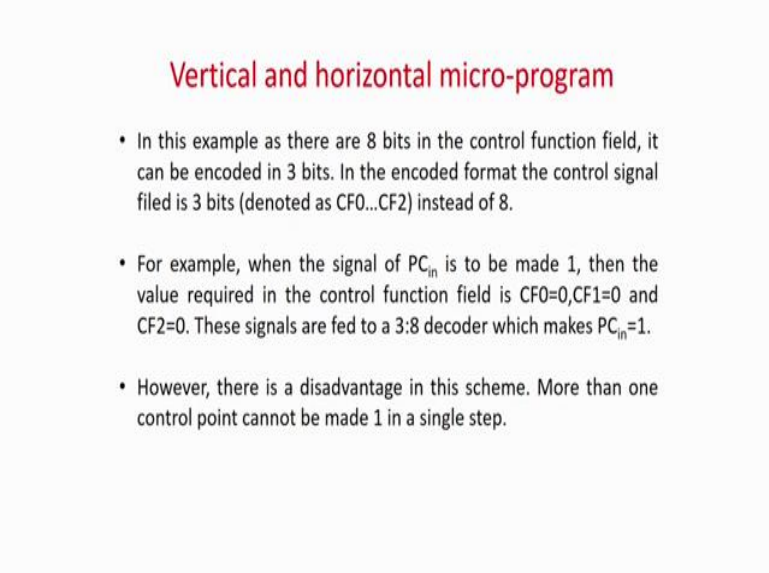
So, we can try to think that in very high number of micro instructions which signals are to be one at in one point of time you give to give it to a different clusters, like PC_{out} and memory address register in. This will always happen in the micro-program instructions corresponding to fetch of an instruction. So, you put it to a different clusters so, that is basically idea.

But, sometimes if you say that the program counter in maybe you have to go to for example, I can say that somehow for some reason may be say that program counter out has to go to memory data register in that's a stray example then of course, you require 2 bits. For example, if you also I am saying that here there is another cluster which is having say $R0_{in}$ there is some $R0$ register is also in this cluster. So, just an example, so in this case basically the memory data out and R data register in will not happen for that you will require 2 steps. So, that the cluster ; that means, whatever is in one cluster if you want 2 signals to be one simultaneously it has to be done in 2 steps, that cannot be done in a single stage.

In that case the time step will increase, but in between large number of clusters are there for inter cluster multiple bits can be 1 simultaneously. So, this way of compressing is actually called a hybrid approach, and sometimes we call hybrid kind of a vertical organization. So, basically that is what is the idea.

So, another technique basically is to be made clusters. So, in each cluster signals which need not be 1 simultaneously in one step you put in 1 cluster. And whichever need to be 1 simultaneously you put in different clusters. For example, like PC_{out} and memory in should be made simultaneously, similarly what I was telling in the example is written in the slide you can go through the slides basically.

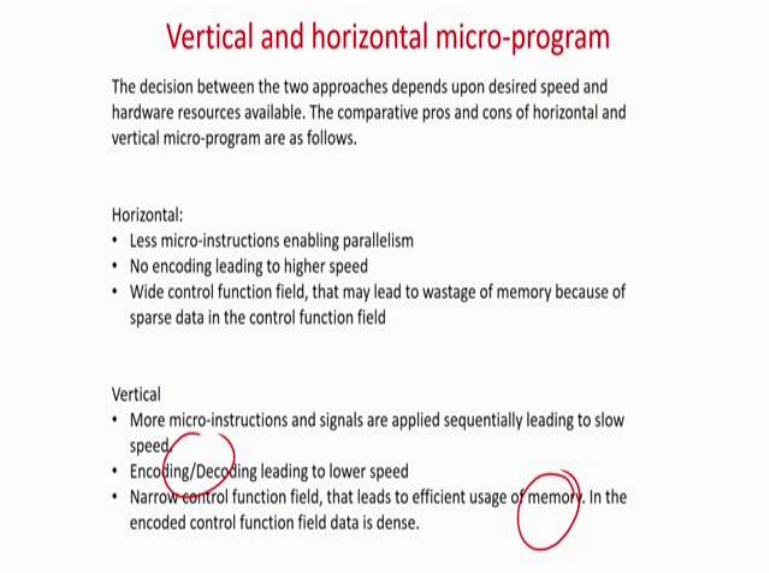
(Refer Slide Time: 62:38)



So, now, if you look at a comparison, so in this case basically these are a basically what idea in this example there are 8 bits that if you look at this slide. So, means how to make a control etcetera all the comparisons are set. So, in this example there are 8 bits in the control field. So, you require a decoder to solve the problem. For example, if PC_{in} is to be made 1 then the values of the control field 000 and decoder will make it as a 1. So, it actually inter this slide actually the interpretation of the figure.

This interpretation of this figure is given in this slide, and I have already explained you this is just for you to read and get what was I was actually discussing. But, disadvantage is that more than one control point cannot be made in a step, if it is a full vertical arrangement. It is actually this slide actually explains this field. This is what I have described that how do an inter cluster organization.

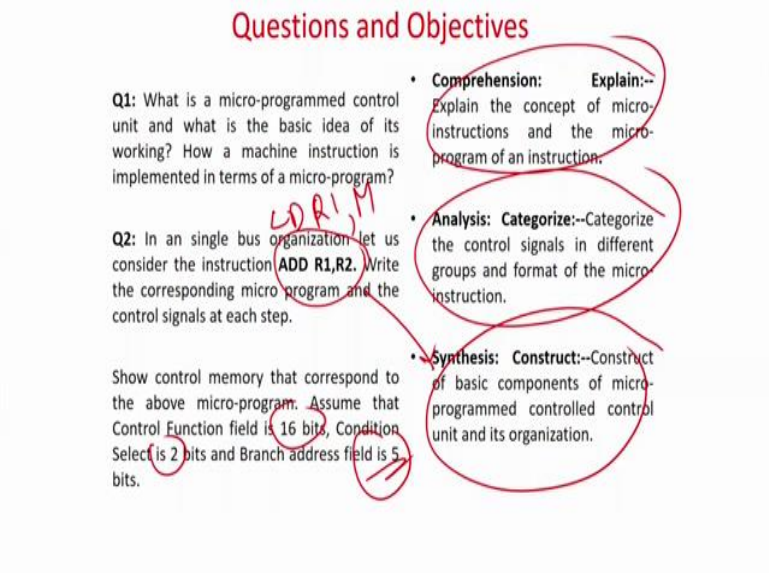
(Refer Slide Time: 63:31)



So, now, consolidating what we have discussed, in horizontal micro-program less number of micro instructions are required. It's a fully parallel architecture no encodings, extremely fast, but actually wide control field is required. So, the memory size will be higher, and there will be because there are lots of 0s involved generally because, only 1 or 2 points it will be controlled simultaneously that is made 1. So, there is lots of memory wastage.

In vertical what we do actually we are encoding the control units control signals. So, there is a less number of memory is there is required. So, wastage is less, but you require decoding and encoding. So, leading it to slower speed, and not only because of decoding you require multiple number of states to solve the same control problem. So, it's a long time taking approach, and what we have also seen that there is some kind of an hybrid approach in which you can make nice clustering based on the fact that 2 signals which has to be 1 should be given 2 different clusters then, you get something good trade off that without wasting much memory, you get a almost near speed of as a fully horizontal micro-program.

(Refer Slide Time: 64:32)



So, basically with this we come to the end of this unit. In the next unit basically, we will look in more depth basically the type depending on different type of instructions or macro instruction. How exactly the full instruction will basically go through? Ah the micro instructions and how they will be executed? Because mostly in this unit we have tried to focus on the fetch part, but after fetching actually there is that instruction decoding and depending on the type of instruction whether it is an ADD or subtract or something you have to go to different micro different location from where the corresponding micro instructions are there, that is how a full instruction can be executed.

So, all those components we will be looking in the next unit this unit actually give you an idea, but what is a basic micro-program architecture how you can optimize? What are the basic micro-program in memory looks like? What is the micro-program counter? The basic idea we have given. In the next unit actually, we will try to go in slightly more depth with the concepts and try to see basically main idea will be how we can basically take a full code, and try to see the full execution in terms of micro instructions. That is, we will try to go in elaborate more on some of the points, which we have the basics which we have discussed in this unit.

So, before we end as we always discuss some questions and try to see the objectives are made. So, the first question is what is a micro-program control unit? What is the basic idea of working? How a machine instruction is implemented in terms of micro program? So, after discussing this unit I think you will be able to answer this and once it is answered basically this

comprehension objective that explain the concept of micro instruction, and basically construct the basic components because, how a machine instruction is implemented in terms of a micro program? So, the basic architecture etcetera, so this comprehension and this construct synthesis objective will be met because, we will be describing a basic architecture you can you should be able to basically describe the basic architecture of a micro-programmed control unit which we have shown and then, just you take how a code actually gets broken down into some terms of some signals the basic theories? So, this actually fulfills this objective.

The second question is I asked you that you take an instruction $ADD\ R1\ and\ R2$. And you take a control memory 16 bits signals, condition bit is 2 bits and address bit is 5 bit, 5 bit; that means, the address space is of 2^{32}. I ask you to that you please try to see how the full instruction can be executed because, mainly in this unit we have discussed the fetch part of it. After fetch part if it is an ADD some other set of micro instructions will be addressed. If it is $LOAD$ some other set will be executed that is fetch part is common, but after that it gets divert means bifurcated depending on the type of instruction.

So, now, please try to your because already we have discussed how to do for the fetch. Please try to extend it for ADD and basically see how the micro-program memory looks like. And of course, we show the 3 bits 3 memory locations are required for fetch, but as I have asked you to go for the other steps also that is executing the instruction. So, we require a higher memory size. So, I have kept the address bit field as 5. So that you have the places to put the other signals which will be required for exact execution of the, $ADD\ R1\ and\ R2$ which basically comes after the instruction is decoded. So, that part you can try.

So, this actually questions will directly help you in meeting this objective that construct the basic components of a micro-programmed control unit and it is organization. Because here, you will have the fetch stage as well as you have the decoder decoding and the execution stage also. In fact, again you will be able to categorize the control signals in terms of groups and format of the micro instruction. Because, in this case we will have basically 2 basic parts will be there, one is for the fetch and the other part is for the execution. So, you will be able to categorize also if you want to try with ADD then, also you can try with $LOAD\ R1$, memory if you try these two basically then we will be able to find out how I can categorize the control signals based on the different format of micro instructions for different type of codes etcetera. So, by this 2 questions actually will be easily able to you should be able to solve these questions

based on the discussion in this unit, and actually you meet this objectives which was the target of this unit, with this we come to the end of this unit.

Thank you.