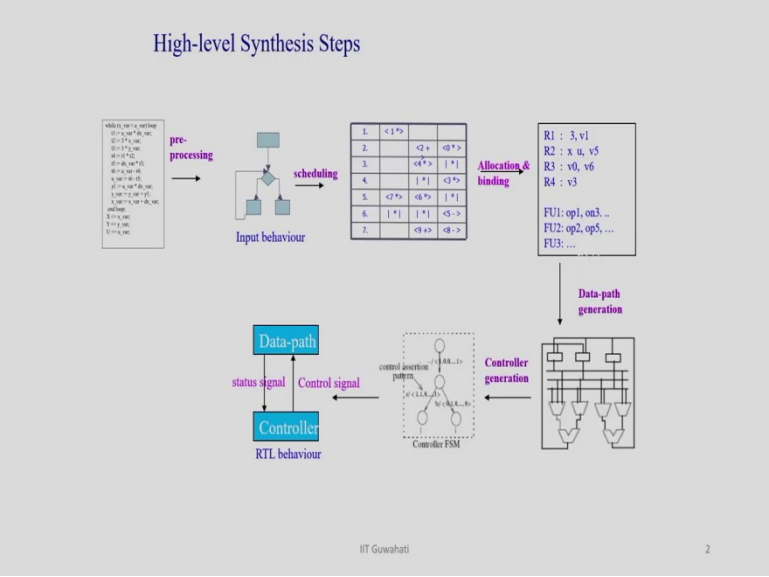
C-Based VLSI Design
Dr. Chandan Karfa
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Module - 06
C-Based VLSI Design: Allocation, Binding, Data-path and Controller Generation
Lecture - 21
Multi-port Memory Binding

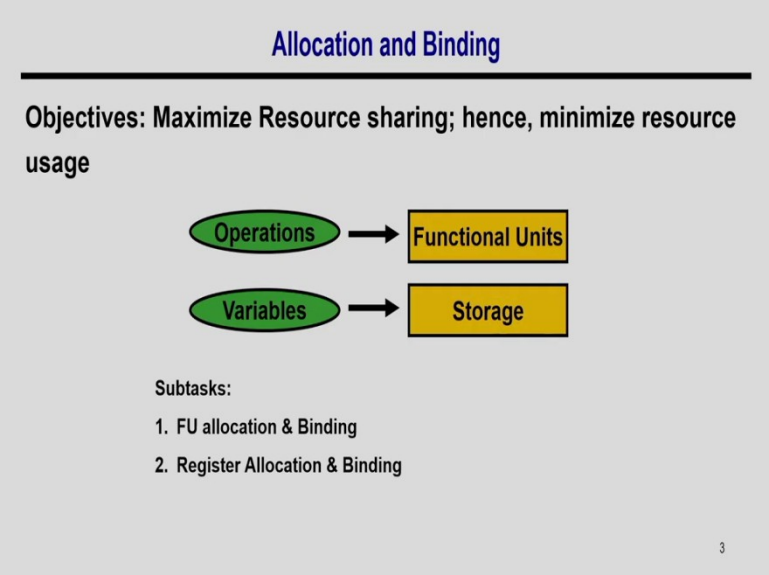
Welcome everyone to this class. In today's class, we are going to discuss the problem of Multi-port Memory Binding.

(Refer Slide Time: 00:57)



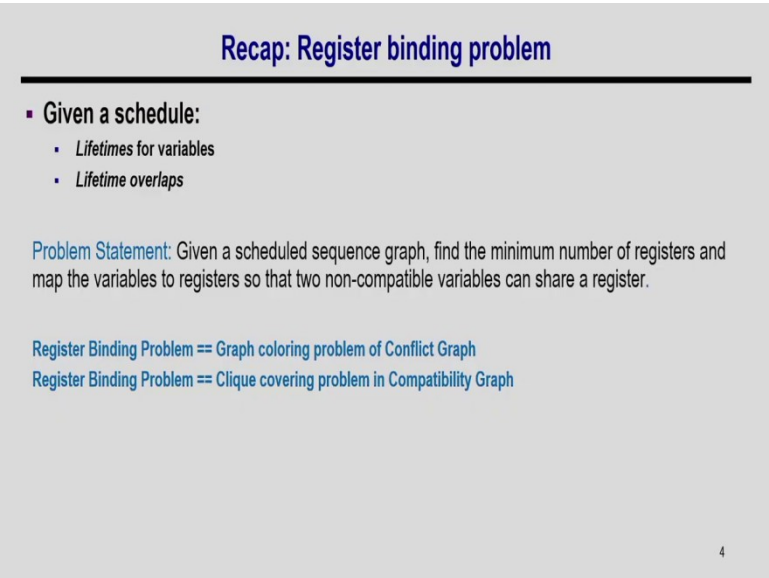
Just to recap the high-level synthesis steps, so we have already discussed that high-level synthesis is a process of converting the high-level way we have written n C into RTL, right. And it consists of several substages like pre-processing, scheduling, allocation binding, and then data path control generation. So, we are now actually discussing this allocation and binding, right.

(Refer Slide Time: 01:14)



And in the allocation and binding, we have already seen that there are two problems; one problem is binding the operation to the function units and binding the variable to the registers or the storage. And we are actually discussing this problem of mapping this variable to the storage, ok. And in the previous class, we have seen that mapping this variable to the register is a problem, which can be solved either by graph coloring problem of conflict graph or clique partitioning problem of compatibility graph, right.

(Refer Slide Time: 01:43)



So, basically what we have seen is that the objective is if there are variables which we identify the lifetimes and if two variables whose lifetime is nonoverlapping; they can share the same register, they can be stored in the same register, right. So, this is what we

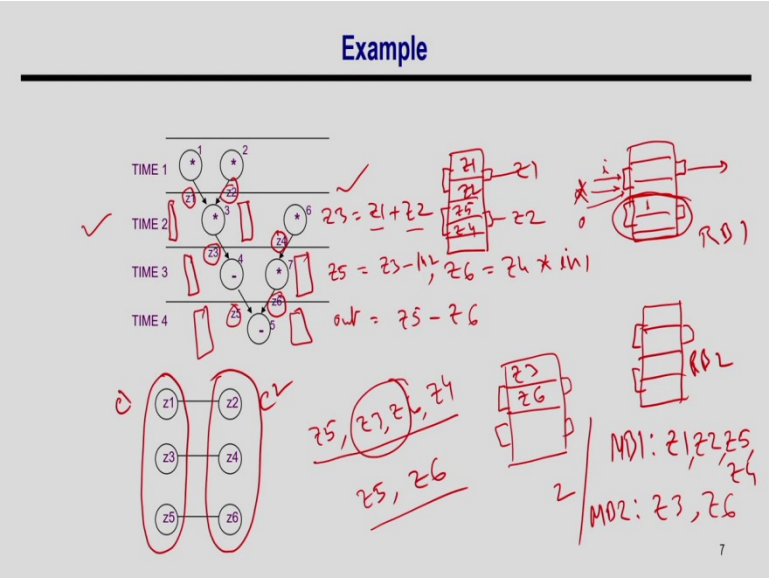
have seen. But in today's class, we try to see this particular problem from a different angle, right. So, earlier we have seen that the registers are an individual entity in the hardware; so, they are actually spread over the hardware.

And I am going to map the variables to the register with an objective to use a minimum number of registers, right. And what we have seen that, if two variable which has non overlapping access say, do not have any lifetime overlapping, we can put them into same register. Because, after the last use of the first variable, there is no requirement of the particular variable in the hardware. So, let us remove that and store the new value into that, right. So, that is the overall idea.

So, now I have we are going to see that same problem, the register binding problem on a different angle, right. So, the angle is something where instead of the register spread over the hardware, usually in a system like these processors, where we have a register bank, ok. So, bank means basically a sequence of registers stored together, ok.

So, we have register banks, instead of individual registers. And the second problem is there that, we have a set of registers and they are not going to be accessed individually, ok. So, what we are going to have? We are going to access through them some port, ok.

(Refer Slide Time: 03:32)



So, if you look into the idea is like this say; so we have a set of register banks. So, this is my register bank 1, say I have it say 5, 6 registers. There is another bank which is say has

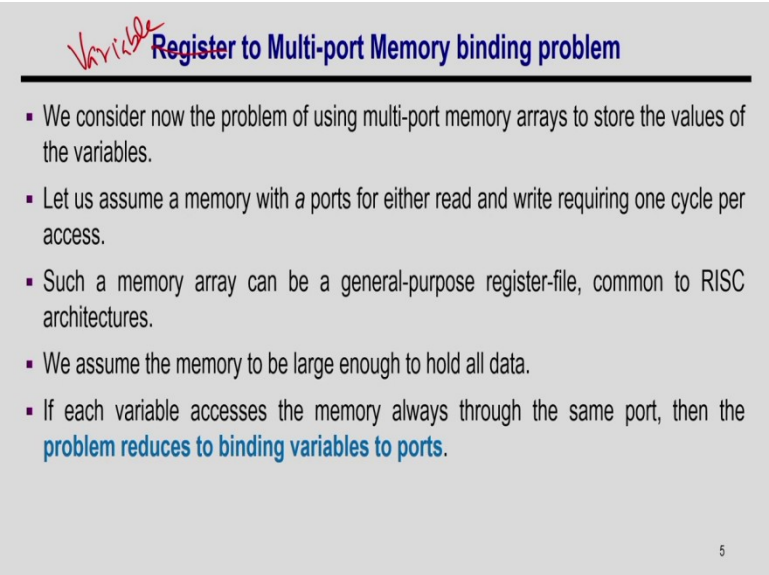
4 registers and I am not going to access this register individually ok, rather I will have a port here, right. So, this is the read combination port. So, if I just put an address here and then if I give a data here and I give a right enable here, then the corresponding address where. suppose this is I have given say register i and this is my register i; so this data whatever d value and will be stored in this particular location, ok. And similarly, if this enable signal is 0; that means, I am going to read it. So, this data does not matter here.

So, I am going to access this register and that particular output will come here, right. So, this is what is the port and similarly, this is the port combination. So, port combination 2, right. So, now, we what is the problem in the hardware, we have a set of such register bank ok, and I want to map.

So, the number of ports may be flexible, but usually, it is one or two in the hardware, right. So, this is my register bank 1, this is my register bank 2, and so on. So, there are such combinations, so there may be say 10 register banks, each consisting of 10; registers there say 100 registers available, but it is not that I am going to have them available individually.

Here the problem is, that I have a set of variables, and I have to map them to registers, but here their access is restricted by the register bank and their port. So, we have to map this variable to the register banks, ok. So, that is the problem.

(Refer Slide Time: 05:28)



So, that is what is the memory binding problem, right. So, basically it has multiple port and it is a memory, right. So, say (Refer Time: 5:34) sequence of register is memory (Refer Time: 5:36), sorry basically a memory. So, now, we are mapping this registers sorry this variable, so this is actually a variable.

So, I am mapping these variables to the memory, which has multiple ports right, the number of ports is also limited. But, what is the problem here? Why the solution that we have discussed in the previous class that you do a graph coloring or clique covering; why that particular type of solution would not work here, ok? Let us try to understand.

So, let us take the same example that I have taken earlier. So, this is my schedule graph right, and I am actually bothered or trying to map this variable z_1, z_2, z_3, z_4, z_5 and z_6 right, these are the six registers I try to map. So, earlier when my registers were actually spread over the hardware; what we have seen that we actually identify the lifetime, right. So, while we identify the z_1 lifetime is this, z_2 lifetime is this, z_3 lifetime is this, z_4 lifetime is this, and this is the z_5 and z_6 lifetime, right.

And then what we have seen is that, since this z_1 and z_2 are overlapping, z_3 and z_4 are overlapping, and z_5 and z_6 is overlapping, this is the conflict graph, right. And the conflict graph if we try to put the colour; so I can put colour 1 to this combination and I can put the colour 2 to this combination. So, this is my register 1 or colour 1, this is colour 2 or register 2 and this is perfect; because this z_1, z_3 and z_5 they are actually having a non overlapping lifetime. So, the single register can store this, ok.

Now, come back to this multi-bank register bank, right. So, let us just say I have one register bank, which has said five registers right and it has two ports, ok. Let us have two port. So, what we have seen that, this z_1 and z_2 cannot share a single register right; that is what we have seen, because they are actually accessing the same time step, right. But now if you think about the register bank; since it has two ports, I can put both z_1 and z_2 into the same register bank, not into the same register, but in the same register bank, right.

Why? Because it has two port and in say one this in the time step 2; since I have to access this z_1 and z_2 and it has two ports, I will read z_1 through this port and z_2 through this port. So, where, so obviously they are not mapped to the same register; you try to keep in mind that, they are not mapping to the same register, but register bank. And bank

is set of registers, so they will be placed into the different register; but these two variables can be mapped to the same register bank.

Now, if you assume that I have already mapped this z_1 and z_2; z_3 cannot be mapped to this. Why? Because in this time step what operation I am doing, $z_3 = z_1 + z_2$. In this time step what operation I am doing? I am doing $z_5 = z_3 -$ in2 is happening say, so forget about that part. So, then $z_6 = z_4 *$ in1 something, some input say, right. So, this is input 2; because input I am not mapping to register, we are only bothered about the temporary.

And here it is happening say out = $z_5 - z_6$; this is what is happening in this operation. So, now what is happening here, z_3, z_1 and z_2 these are the three variables I have going to access in time step 2.

And since I am assuming that I have a memory bank that has two access ports; so only two of them can be mapped to the same bank, ok. So, once we say these two registers are mapped to the same bank; they will store somewhere, that is automatically decided, right. So, there are a series of registers, we you mean you can just put in some places, right.

But what is stopping me here is that, here z_1 and z_3 were mapping to the same register, because their lifetime was non-overlapping; but here since I have already decided that z_1 and z_2 go to this register bank, I cannot put z_3 here, because then I have to access three values, right. So, there are two reads and one right, but you have to access that location.

So, at the same time since it has two ports, I cannot access three values; hence I cannot map both z_3, z_1 and z_2 into the same bank. So, I have to put z_3 in this bank. So, let us say as I have another bank of two ports, right. So, this is how I solved the first step. In the second step what I am doing is, I have actually z; I am accessing z_5, z_3, z_6 and z_4 right, because there are four access happening.

So, I would let us say I assume that in 1, in 2 is inputs; I am not considering them now. So, there are four access, right. So, I cannot map all these four values to same registers right; because then I cannot access them at the same time, ok. So, what I have to do? I have to take any two and put into one's bank; say I am just putting say. So, here actually

z_3 and z_4, so z_3 is already here. So, I cannot just move it to there; because z_3 already decided to be put here and z_4 and say let ill let us say I have decided to put z_6 here.

So, since z_3 and z_6 is already put into this memory bank 2, I cannot put z_5 or z_4 right; because in this time I have to access all four registers, four values and only two-port are available. So z_3 and z_6 here and I can put z_5 and z_4 here, right. z_5 and z_4, which is perfectly fine, because if this bank store z_1, z_2, z_5, z_4; but I am only accessing z_5 and z_4, so two port will sufficient.

And what is happening in the last step? It is z_5 and z_6. And since z_5 is here and z_6 here, it is actually satisfying the port in common; I have to access only one value from this memory bank and one value from this register bank, which is supported by two ports, ok.

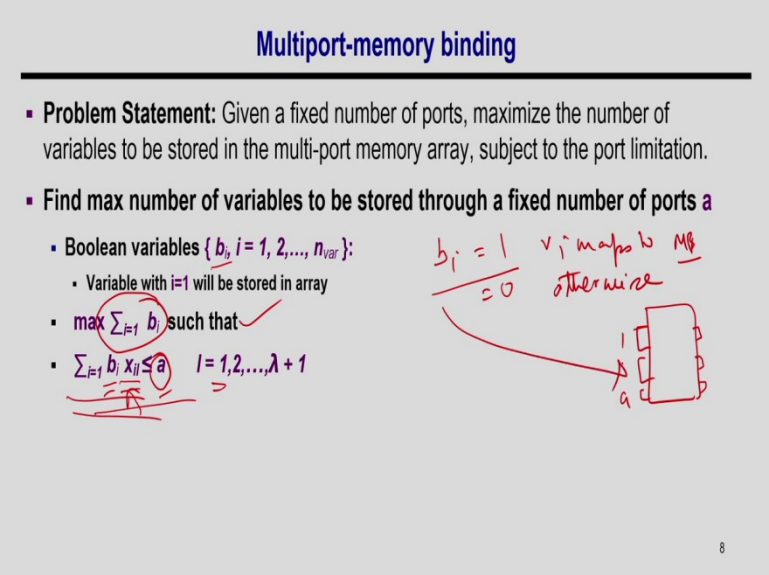
So, this is how we solved the problem. So, you can actually understand that my memory bank 1 will store z 1; you can have multiple solutions, I just talked about one possible solution z_2, z_5 and z_4. And my memory bank 2 will store z_3 and z_6, this is the solution I got.

So, you understand that here the problem is not identifying the lifetime, so lifetime would not help here. So, the way we solved the problem for, the generic raised to allocation problem, where the registers are access spread over the hardware, that particular way of identifying the lifetimes.

And then finding the conflict graph and then trying to put colors into the conflict graph and finding the minimum colour is equivalent to the minimum number of registers is not the solution; that is not the approach to be taken for this particular problem, right.

So, here what is more important? Identify the access in a end in each step right; what are the variables are getting access is in time step and based on their access, you make sure that you map this variable to the registers in such a way that, your access is not getting violated, right. So, that is the problem here, right. So, if I just define the problem statement now.

(Refer Slide Time: 13:54)



So, this variable to the multi-port binding problem can be defined as follows; given a fixed number of ports, maximize the number of variables to be stored in a multi port memory, such that the port limitation or is not violated, right. So, I want to map maximum number of variables to a memory, such that their access is getting satisfied by the number of port available, ok. So, that is what is the problem.

So, now you have many n number of variables right; I want to map those variables to ports. So, I want to see how many of them can be mapped to memory bank 1, right. And then once that is solved, I can remove those access; because that is done and then for the rest of one, I again have to try to the try to solve same problem, how many of them the variables can be mapped to the memory bank 2 and so on.

So, this is how I am going to, solve the problem, ok. So, now, the question is, I understand the problem and how do we want to solve it? So, what I am going to do here is, I am going to give an ILP-based solution, integer linear programming-based solution for this problem, ok. And you remember the problem formulation that has given is solving the max.

It will give you the maximum number of variables that can be mapped to a single register bank. So, as I mentioned, once you get that; you can remove those accesses and you can actually reformulate the problem and you can actually solve it the same thing again to find out what is the set of variables can be mapped to register bank 2 and so on.

So, basically, the whole formulation can be run iteratively to identify the total number of memory banks needed to store all the variables, ok. So, this is what is the approach I am going to store, I am going to take, right.

So, let us now try to solve the problem. So, what I have told you is that, my objective is to identify the maximum number of variables that can be mapped to a single memory bank, which has a fixed number of ports, say ‘a’ number of ports are there, ok. So, let us try to solve the, I mean formulate the problem. So, I have the Boolean variables b_i, which is basically for each variable, ok.

So, I have a Boolean variable b_i, which is basically designate a variable in ‘n’, ok. So, I have ‘n’ variables, right. Now, what is my objective? I want to, so $b_i = 1$ if that particular variable v_i maps to the memory bank, right. So, I am actually solving for one bank only, how many of the variables can be stored to the memory bank. So, I am not solving the complete problem, I am just trying to find out the number of variables that can be mapped to a single memory bank, ok.

And if it is 0 otherwise, right. So, I have to find out the maximum number variable for which $b_i = 1$. So, that is my objective, this is an unknown variable. So, what is my objective? My obviously, I want to maximize the number of variables mapped to this particular memory bank, ok. And what is the constant? It is already known to us that, the number of accesses to this variable should be limited by the number of port available, right.

So, I assume that I have a memory bank that has a number of ports, right. So, a number of port and then I want to map a maximum number of variables, such that each time step the number of variables getting access from this particular memory bank is less than equal to a, right.

So, how can I write that? So, this b_i is basically for the variable, if that variable is mapped to this memory bank. And this X_{il} gives those corresponding operations, whether it is getting access in the time step or not, right. So, if that particular variable is getting access in the time step or not.

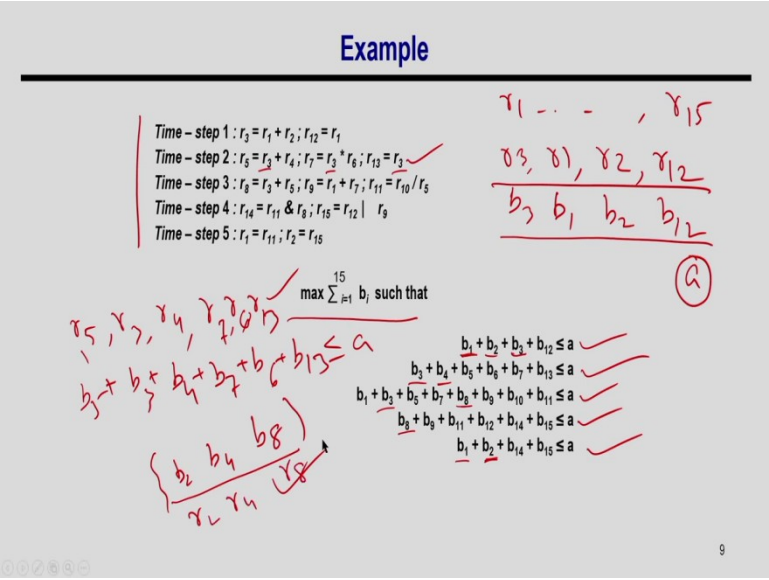
So, this actually says for each time step l, the variables that are getting access for that. So, among those registers only, a number of variables can be mapped to this memory

bank, ok. So, this is the constant. So, the ILP formulation is very simple, I want to maximize this, that are a maximum number of variables for those are mapped to this memory bank; subject to the access limitation that, every time step the number of accesses to those variables that are mapped to that memory bank should be limited by a, should be less than equal to a, ok.

So, this is what is these constant talks about. And if you give this particular to the ILP server; what it return? It will return you the b_i value. So, for the variable for which $b_i = 1$, those variables can be mapped to this register bank, right. So, I just solve the problem for one register bank and then I can iteratively solve this thing for register bank 2, memory bank 3 and so on ok. So, this is what is the approach.

So, let us take an example, say here the lifetime does not matter right; but the schedule is very much important. Because we need to know what are the operation getting executed in time step, because from there only I can understand what are the variable is getting going to access, right.

(Refer Slide Time: 19:04)



So, let us say I have given these particular schedules, right. So, in time step 1, $r_3 = r_1 + r_2$ is happening and $r_{12} = r\ 1$ is happening. Time step 2, $r_5 = r_3 + r_4$ is happening; $r_7 = r_3 + r_6$ is happening and so on right, this is how these operations are getting scheduled, right. So, I want to solve this, I want to identify how many of this.

So, there are how many variables are there? r_1 to r_{15}, there are fifteen variables are there, r_1 to r_{15}; I want to identify the variables which can be mapped to this register bank, given register bank, ok.

So, obviously, I want to maximize this, I want to identify a maximum number of variables that can be mapped to a particular bank. So, my objective is this and these are the constraints. So, how many constraints will be there? I will have a constant for each time step. So, how many constraints will be there? Since there are 5-time steps, I have I will have five constraints, ok.

So, what are the constraints for time step 1? So, I can see here that r_3, r_1, r_2 and r_{12} is getting accessed in time step 1, ok. So, that means the corresponding variable is b_3, b_1, b_2 and b_{12}. So, and since the number of port available is a; so only a number of this four will be accessed, I mean can be mapped to the same bank. So, that is given by this constraint; that $b_1 + b_2 + b_3 + b_{12} <= a$.

So, suppose a =2, so that means a maximum two of them can be mapped to this particular memory bank; among r_3, r_1, r_2 and r_{12}, only two of them can be mapped to this particular register bank, if the number of port is 2, right. So, this is the for constraint time step 1. So, similarly, I can identify constant on time step 2. So, I will identify the variable getting access; there are too many, so I can just note for one more time step.

So, r_5, r_3, r_4, r_7, r_3, r_6, and r_{13}; you see here there this r_3 is getting access three times, but if I read it once, I can use it anywhere. So, I do not have to put, I do not have to read it three times. So, I can consider only r_3 equal to 1, one time; because it is something if I read it from bank, I can use it I mean multiple operations, right. So, that is why I just put r_3 equal to one time. So, the corresponding variable is b_5, b_3, b_4, b_7, b_6, and b_{13}. So, what will happen?

So, this summation should be less than a; so that means $b_3 + b_4 + b_5 + b_6 + b_7 + b_{13} <= a$. So, this make sure that, if a = 2, only two of this particular variable among these six variables can be mapped to this particular, to this memory register bank. So, similarly, I can identify time step 3, time step 4, and time step 5, ok.

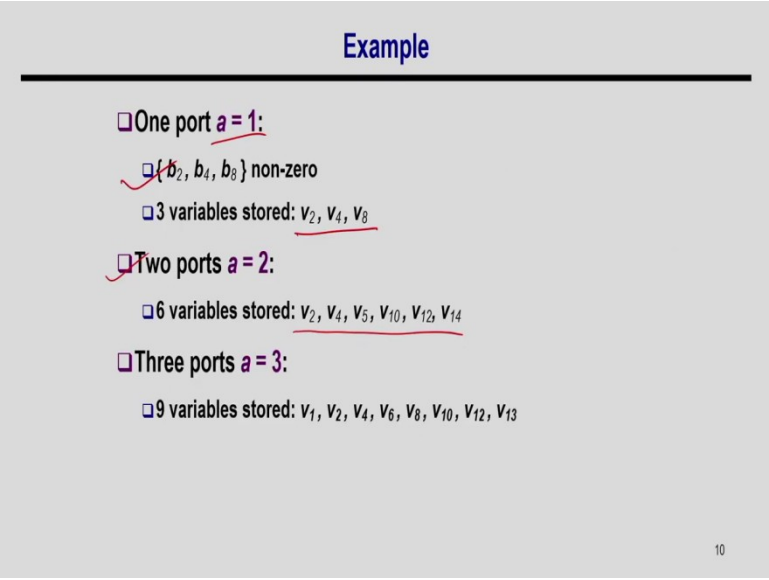
So, these are the constraints. See if I give this to the solver and if I mention this a value; a constant say, 1 or 2 or 3 4 whatever it is, it will tell me the variables that we are going

to map to this register. You can see and understand that there are actually variables, which is actually this constant is on; it is not this constant are not independent right, they are interdependent.

For example, you see here b 1 occurs here as well as here; so that means the way I have to assign the value of b_1, even b_2 also true right, such that both the constant gets satisfied. It is not that I just individually take a constant and satisfy that will solve the problem; but it is not, because you might put b 1 and b 2 on the bank and then you see and say b_1, b_2 and say b_1, b_2.

And then you try to such a that and then in maybe in that same next constant; you try to map such a way that, if the particular constraint is not getting satisfied, right. For example, you see here this b_3 occurs here, b_3 occurs here, b_3 occurs here. So, the way I am going to assign this b_3; if b_3 is equal to 1, it will actually impact all these constraints. So, the way I am going to assign the value of this 1 or 0 to this b_1 or b_{15}; so others all the constraints get satisfied right, that is the ILP will do what do it for us, right.

(Refer Slide Time: 23:33)



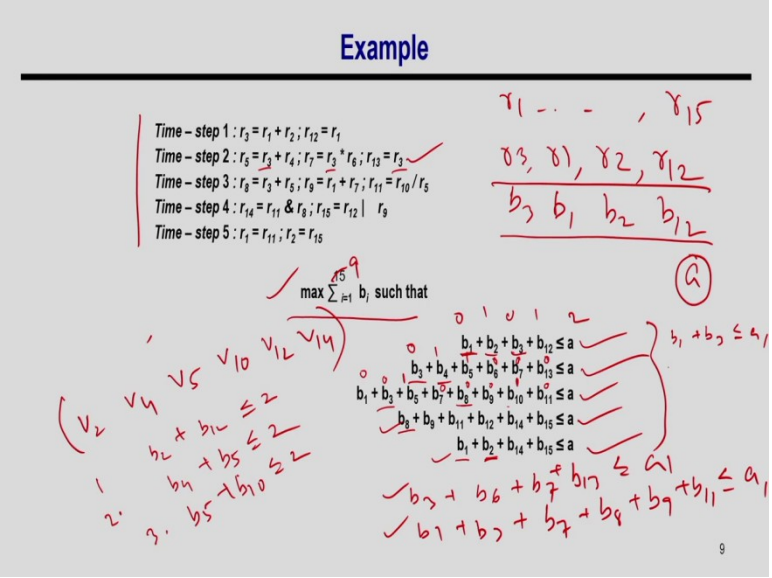
So, let us say I have one port, ok. So, if I have one port for this what are the variable; maximum how many variables can be mapped to this, right? So, the solution that you will give you this v_2, v_4 and v_8 right, so basically b_2, b_4 and b_8, so let us see. So, b_2, b_4, and b_8; I have to make sure that since is a single port, not any time step this a maximum

one of them will be accessed, right. So, a maximum one of them will access every time step, let us see.

So, in time step 1, only b_1 is getting access; not b_4, b_8, so then it is fine. Here b_4 is getting access, no b_4, b_2 or b_8; here b_8 is getting access, no b_2 or b_8; here b_8 is getting access, there is no b_2 or b_4; here b_2 is getting access, no b_8 or b_4, right. So, that mean, this actually this mapping satisfies all the constraint. And so, if there is a single port, I can map these 3 three variables that r_2, r_4 and r_8 to this register bank.

Similarly, if say number of ports is two; then it says that you can actually map this v_2 to these are the variables. So, let me just so, let me just check, whether that particular mapping is correct or not.

(Refer Slide Time: 25:15)



So, let me just, so v_2, v_4, v_2, v_4, v_5, v_{10}, v_5, v_{10} and then v_{12}, v_{14}. So, there are six registers, six variables can be mapped to the same register bank, ok. And what I have to do? I have to make sure that each time step, a maximum two of them is getting access; if not, then this is not satisfying the constant. So, let us see. So, in time step 1, I am getting access b_2 and only b_2, right. So, the and b_{12} right; so that means it is getting satisfied that, this plus this is greater than equal to 2.

And others are 0, this is 0, this is 1, this is 0, this is 1. So, is getting less than equal to 2, which is getting satisfied. In time step 2, I have accessed b_4. So, this is 0, b_3 is 0, this is 1;

I am accessing this one also $b_5 = 1$, b_6 is 0, b_7 is 0, and b_{13} is also 0. We can see only b_4 and b_5, rest are 0.

So, then it is getting satisfied in time step 2. In time step 3,so I have accessing b_5, this is 1 b 5; then this is 1, b_{10} is 1 and this is 0, this is 0, this is 0, this is 0, this is 0, this is 0. So, I have actually accessed $b_5 + b_{10} <= 2$.

So, I can understand that this mapping, this six-variable mapping to the same register bank satisfying time step 3 also. And so, on I can, so that it is actually satisfying constraint 4 and constraint 5, ok. So, this is how the ILP gives me the solution that with two bank, maximum six such variables can be mapped to the same bank. So, once I have this is done what I just told you that, I can rewrite this constraint by removing this variable, right.

So, I can rewrite this expression by just, because this is already mapped, right. So, I can rewrite this expression by $b_1 + b_3 <= a_1$, say this is my a1 for the next register bank, right. So, then $b_3 + b_6 + b_7 + b_{13} <= a_1$ and this is

$b_1 + b_3 + b_7 + b_8 + b_9 + b_{11} <= a_1$, right. So, this is how I can actually rewrite the expression. So, just you can see here that, I am removing the variable which is actually 1; because these are the variable already mapped to memory bank 1.

So, I have to now solve the problem for the rest of the variable. So, this is how I can actually the value which is 1, I can remove them and the constant will be reduced and now I want to solve the same problem. So, here I can I will just put it to 9, because 6 of them is already done; then I am trying to solve the problem for memory bank 2. And then I will identify the variables, which can be mapped to memory bank 2 subject to the number of ports, it may be 1, 2 or 3, ok.

So, this is how I am going to solve the problem and then once the mapping I got it from; then I will remove those variables again from this register constant and I am going to solve it for register bank 3 and so on. This is how I will identify how many registers bank is needed and for each bank what are the variables we going to map, ok. I hope you understand this and with this I conclude this today's discussion.

Thank you.