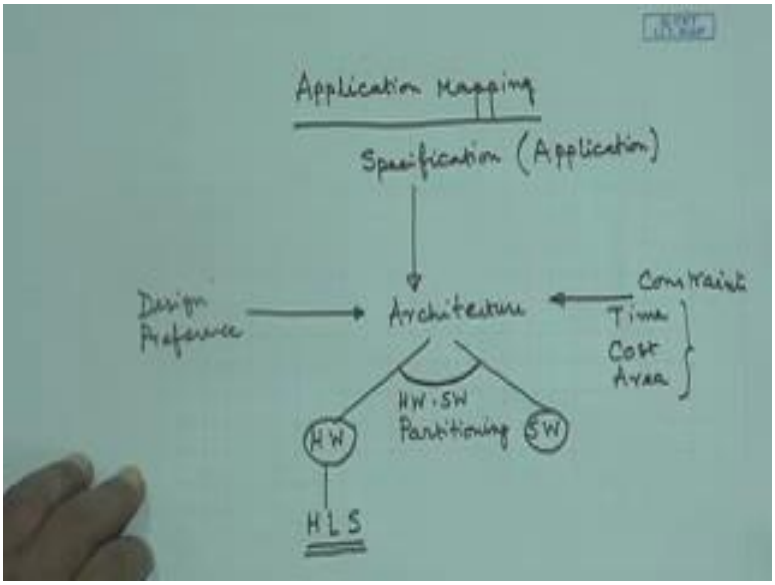
Embedded Systems Design
Prof. Aaupam Basu
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 40
Hardware Synthesis – I

Very good morning, we have till now discussed about the last couple of lectures, we have looked at the different specification algorithm, specification models, and we have seen state charge, we have seen SDL, we have seen synchronies data flow, and Kahn data flow model. Now with all this specification our, if you recall at the very outsight we had shown your diagram, where their different phases of embracing system design were in unsheathed. So, we have complete the specification part, prior to that we have completed the real time operating system of the operating system part, the hardware part also been completed.

Now, with this specification, now we have to come to the real design phase. Now, the real design phase for an embedded system will consist of I can call it design or sometimes we call it application mapping.

(Refer Slide Time: 01:19)



The reason why we call it application mapping, will be evidence soon. What we intend do, we start with the specification, and this specification, is then mapped to some architecture. If you recall we have talked about, what is a module, and how do we

translate a modal a specification module, ultimately we come bring it to an architecture, that is an implementation, implementation architecture. So, this specification will be mapped to a particular architecture. And this specification is that of an application, for an embedded system, we have got an application in mind, and for that particular application we come to architecture. Now, this architecture can be provided by the user, can be specified by the, as a design required maybe, or let me call it a design preference. It maybe a design preference; that I will have two processors, or a single processor whatever.

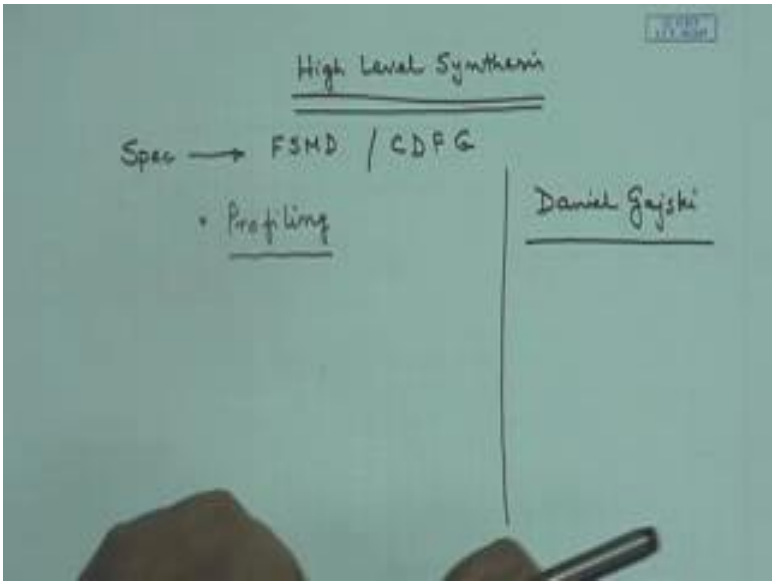
Now, based on that this architecture we actually consist of some hardware paths, and some software paths. Now there can be a situation where the architecture is not specified, you have to find out the architecture. So, then you have to decide on which one will the hardware which one will be the, which functionality will be implemented in software, and which functionalities will be done in hardware, and this part is known as hardware software partitioning. That means, we decide on what, which are the tasks that will be implemented using hardware, and which are the tasks that will be implemented in software. Then we can synthesize the hardware, and synthesize the software.

Now the decision about this hardware software partitioning, is guided by a number of constraints. What are the constrains? The constrains can be; the time constrains, the cost constrains, the area constrains. All these that we had talked about at the very beginning of this course, so depending on these constrains we have to actually try for an optimization solution. So, today will start discussing about, this will suspend this hardware software partitioning approach for a couple of lecture more, first will start with hardware.

Now this hardware we have to synthesis. Sometimes it maybe that, you are given some assets and you have to implement within those assets, but sometimes you will be requiring to design an assets. For example, you have come to an architecture and then you decide that this architecture will be broken down into, say task a b c will be done on a hardware, and d e will be done on software, you decide that. Then this a b c, the d e which is being done in software will go to the processor. Processor will implement the software, but for the hardware part, you will have to synthesis the hardware; that is what we called the high level synthesis, where we synthesis the hardware.

Now, when we do high level, so today will first start discussing about this high level synthesis, how we do that. I mean whenever you do a course on electronic design automation, then you will have more details of this high level synthesis, in the embedded system course it is only a part of this course. So, will not go to that detail, but I will try to expose you, to some of nuances of this high level synthesis.

(Refer Slide Time: 06:22)

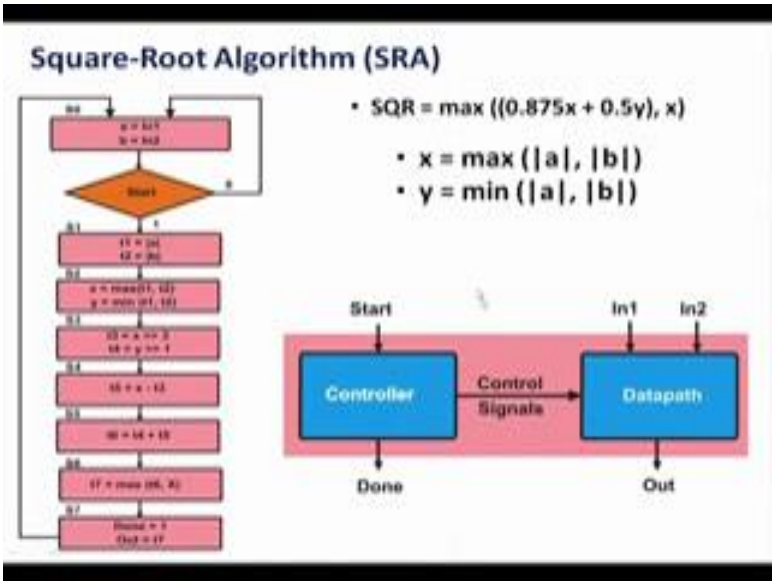


So, in high level synthesis, we first start with specification, and that specification can be. As we have seen the specification was given in state chart or VHDL whatever. Ultimately that can be flatten down, into either an FSMD finite segmentation with data, or a control and data flow graph. So, essentially will have some sort of graph. This graph will show, the control and data flow graph shows, the tasks, the data tasks, as well as their dependences. So, we will start with this.

The first thing that we do is, some sort of profiling. We try to do some sort of profiling, by profiling what we mean, is we try to see what are the; say I look at the particular task, and try to find out how many steps of operation, how many operational steps will it take, how many. Just rough cut idea about how many operators I will need. How many variables I will need how many register I will need, all those things I try to do through a profile. So, let us first try to see how profiling works, lets taken example, the example that I am taken, I am using all through in this course, has been taken from, I have given

you the reference of the book by Daniel Gaiski; the embedded system design. So, I am taking that example from professor Gaiskis book and the example in this lecture.

(Refer Slide Time: 08:28)



So, here is an example of square root algorithm. Square root algorithm means, it is actually this square root algorithm which are calling SRA, is actually finding out the square root of a square plus b square, we are calling that SRA. So, just as an example, because this example is good enough, to show different features, that comes up in this task. Now, here you can see that this a square plus b square we are doing not in a straightforward manner, but through little convoluted manner. This is not the best way of doing it, but it will help us to illustrate many points, where we take x to be the maximum of absolute value of a and b, y is the minimum of absolute a and b, and the SQR is a max of 0.875 of x, plus 0.5 of y or x whichever is the higher, x or 0.875 of x 0.5 of a plus 0.5 of y. So, this is how we come will compute SQR there is a square root of a squared plus b square will be computed in this way.

So, initially we will need a design. So, let us look at this part, we will have a controller and the data path. Now please recall that we have done a similar exercise, when we discussed that greatest common desire divisor gcd right. Now here we will do a similar exercise, but we will go into a little more detail of the steps. So, the controller will have a start and done, and the data path will have input 1 and input 2, and the output will be the square root, and the controller and the data path will be communicating. Actually the

control and data path communicates using control signals, as well as the controller receives the data path the status signals. In this case we are not using any status signal.

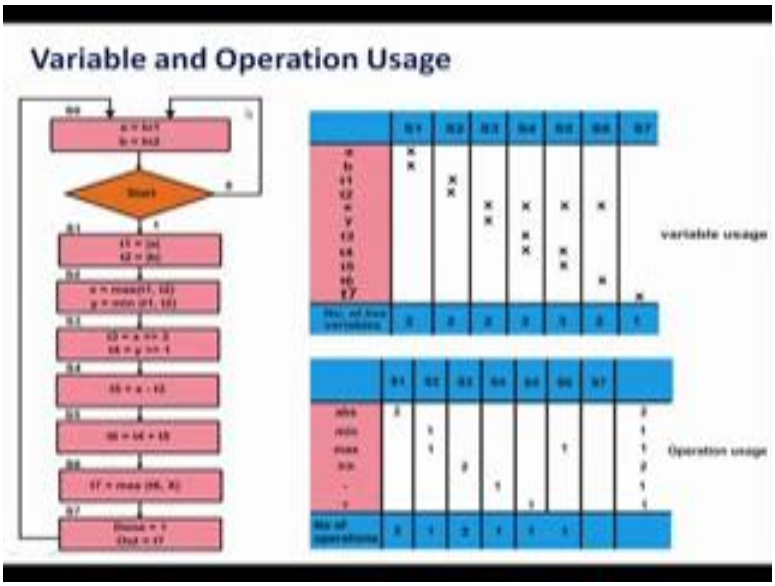
Now, let us look at this FSMD. This is a FSMD, where we have got s 0 as the initial state, where a is accepting the input 1, which is this one input 1, and b is accepting input 2 then we start; that is s 0. Now if the data comes in, we does not come in we repeat, and then when the start is given; that means, you see the controller is asking to compute the square root we come here.

In s 1, we are doing, we are computing the modulus of a and modulus of b and keeping them into two variables t 1 and t 2; that is this part then we are computing into x and y; that is here x and y as the max and mean. So, there are two operators max and mean. Now, we are shifting max of, we are shifting x with three positions to the right. So, what does it mean? If we move three positions to the right, then we are dividing, dividing by eight, so it is becoming 0.125 right, 0.125 of max of a b; that is x is being divided by 0.125 and t 4 is being shifted one place. So, it becomes 0.5 y, we wanted to have 0.5 y. So, I am doing to shift operations here, and then t 5 is x minus 0.1 to 5. So, that is giving me 1 minus 0.25 is giving me 0.875.

So, I am completing that using a subtractor, and then this one is t 4 plus t 5. So, I am getting this value of t 6 , so t 4 plus t 5 is this 1 0.875 x plus 0.5 y, and then I am taking the max of that in statistics, and I am taking the max of this one t 6 and x, and that is my output and I am sending the damn signal.

So, here I can see that I am moving through eight states. Each of these states are computing some doing some computation. Right now we can see the computations; one is computing the absolute value twice, from on two variables, finding the max, finding the mean, shift operations addition and subtraction, max is again repeated here. So, I need max and min. So, now I can see that in order to carry out this operation, what are the operands I need, and how many variables I need. So, I can see that without attempting to do any optimization, I am just writing it down, and I am using variables freely at my will and so I am getting so many variables.

(Refer Slide Time: 14:40)



Now, I start doing a little trick here. This table well I look at the variable usage. So, I have got s 0, I have left out, but a and b are coming from the input, input is coming from the outside, and here I am only considering the elements that are being a red, because if you think of a clock cycle. Then at the positive edge of the clock cycle the data will be red, and it will be transferred to assignment at the next clock cycle; that is why here I am considering only the read variables, the variables which are red. So, in state s 1 I can see the variables which are being used, are a and b. In s 2 the variables which have been used are t 1 and t 2. Now what are this max and min? These are the operators, the functional unit on which t 1 and t 2 will be served as the input, and max and min will do the computation.

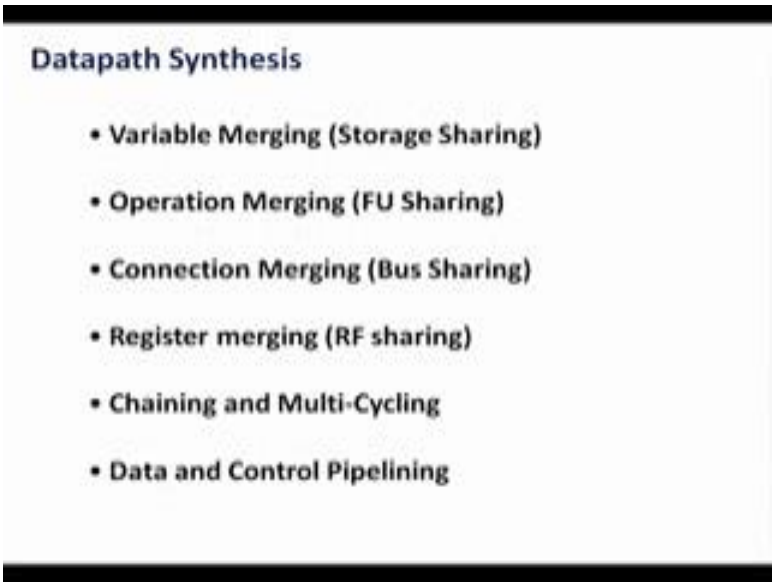
So, here we can see that t 1 and t 2 in state s 2. Similarly if you work out you see, in state s 3, we have got just a variable; which are the variables x and y. In s 4 we are using x, in state s 4 we are using s. I am sorry this x t 3 right. In s 4 x and t 3, t 4 should not have come here. So, there is a mistake here, s 5 again; in that way you put on the all the variables. Now, here I can see that the numbers of variables are, I can see how many variables I need. Looking at this if I assume this, I can find out from here what is the maximum number of registers I need, from this table. Similarly we can look at the operations. The operations in all the states s 1 to s 7, s 1 to s 7 we can find out that in state s 1. In state s 1 we need, what are my operations, absolute max mean shift minus plus. So, I need two units of absolute operations. Here I need one unit of mean operation

and one unit of max operation. Here s 3 I need two shifts. Now I can have two shifters; one shifting three places, one shifting one place if I do them concurrently.

Now, you can very well say that this absolute. For example, I could have, look here is a interesting point, I need two absolute computations right. Now you could have said that I will do in with one absolute computation. So, I will in that case, I could have saved on the cost of the absolute computation, but this state had to be broken down into two states. Thereby I would have increased the time that would be spent for computation. Therefore, I am not doing any optimization right now. So, there are two. In that way this chart, will tell me how many operations are in use, and whether I can. Suppose I select an ALU, in that case all these I can do with an ALU straight way. So, in that case the number of operations will, in a decrease.

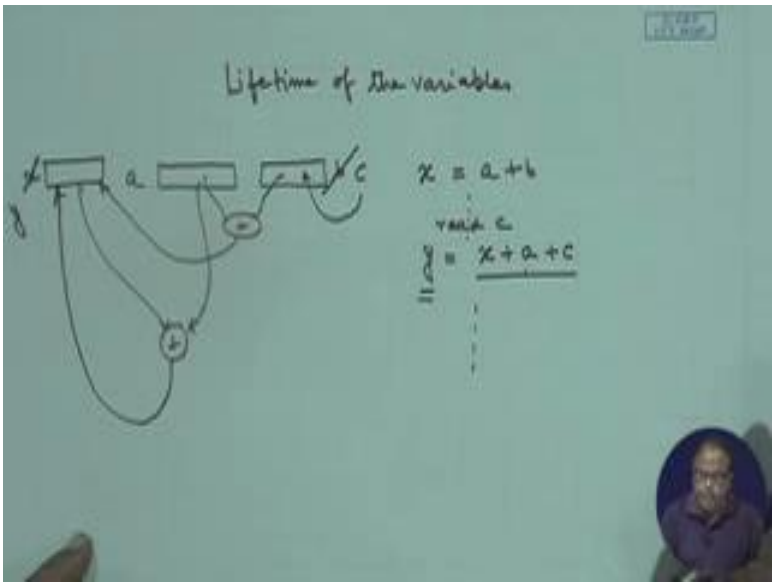
So, here I do the preliminary profiling that I was talking about. Once I do the profiling. So, that gives me an idea of what are the operators I need, and what are the variables I may need.

(Refer Slide Time: 19:06)



Next I proceed; I will do some data path synthesis now from this point. For the data path synthesis I will do a couple of things; one is I have seen what are the variables used.

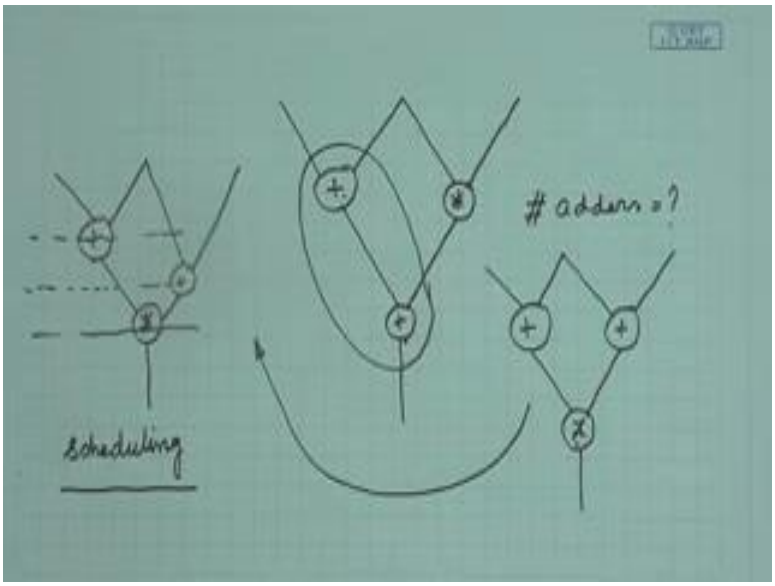
(Refer Slide Time: 19:29)



Now, I may not need so many variables, there is one thing known as the lifetime of the variable. For example, suppose I am doing something here, and then later down here I am doing y assigned, say x plus a plus c. Suppose I want to do. now at this point I have got a variable a, and a variable b, and the variable x. all these are needed at this point; a and b are being added, and that is being stored in x. Now when I come here a is still alive. Suppose b is not needed anywhere here.

At this point I have got my x. So, I can take this x straightway. This a is still live, because I am needing this a here. So, I can take this a here, but b is no longer needed, because the computation of b is over. So, I can very well in between store, suppose I read c here, read c. Then I can very well read c at this point. therefore, instead of a b c x, I can very well managed with a b c and, this y suppose later on excise not needed anywhere, then this y could have gone here and after reading. So, this could be y. So, therefore, I want to see which are the variables, which are live at a particular point of time. So, looking at the liveness of the variables, I can merge some variable, so that is one technique. In a similar way we can merge to functional units.

(Refer Slide Time: 22:02)

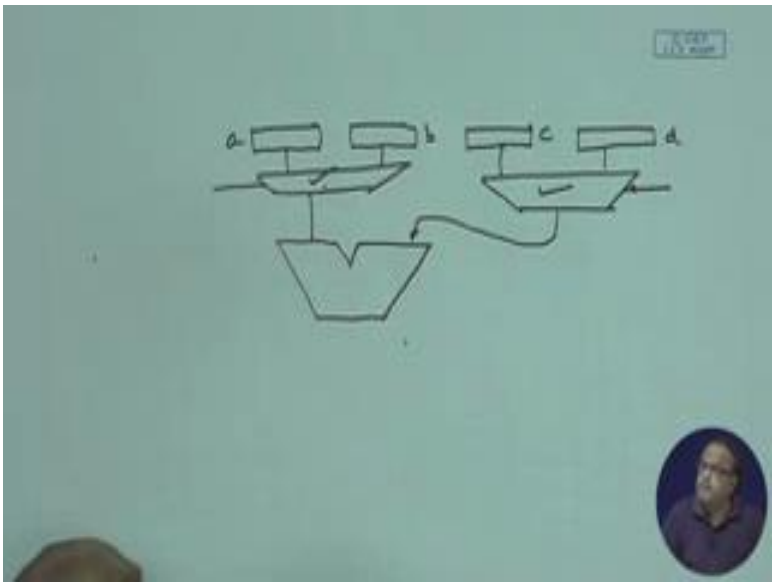


For example suppose I have got in a data flow graph, I have got some graph like this. Suppose these an operation. How many adders do I need, how many adders? Only one it suffice, because this one and this one these additions, are not being done at the same time; therefore, I can very well mark this function unit, but that would not have been possible, if my operation was something like this right.

In that case I would have needed two adders, but if you think a little bit, you can say that if time is not that big a constraint, then I could have very well transformed this one, to be something like this, that first I do this with these two. Then I reuse this adder, the output of this adder, will be coming to this multiplier, waiting there, and this adder, if I had schedule later in between here. So, this is step one, step two I do another addition, using the same adder, and then do the multiplication in the third step. Here I am doing the multiplication and second step if I had delayed I could have reused this address, but that is I am saving the functional unit at the cost of computation time.

This task will encounter, this phenomenon, this is known as scheduling. Our people call it scheduling also; that is I am allocating the operation at different control step. So, here the purpose was to show that we can merge the operators. Similarly we can merge the connection.

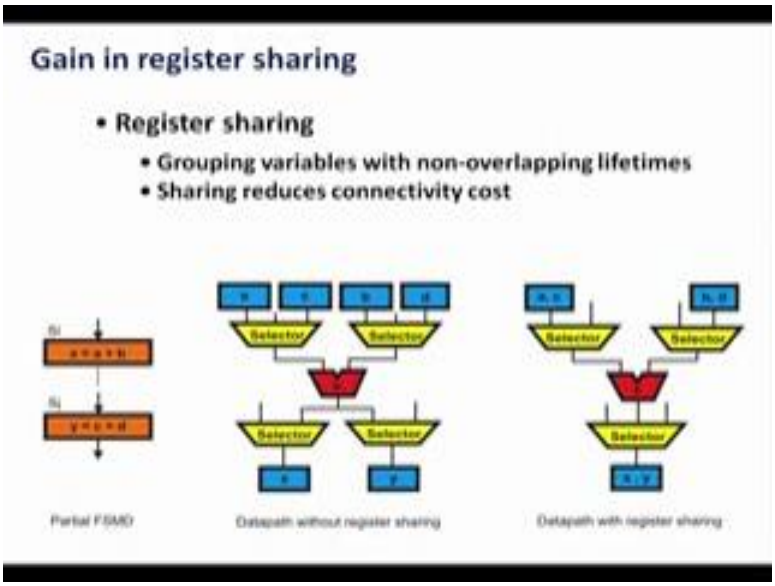
(Refer Slide Time: 24:39)



For example if I have, say some operator is here, some ALU is here, and the ALU is getting data from variable a, it can take the left input, can take from variable a or b or c or d, say the right input comes from. So, c or d comes to multiplexer, here I put a mask. And from this multiplexer I take this data, and this one also comes to a multiplexer with some control, and I can take data from here. So, a or b comes here and c or d comes here.

Now, in this case I can merge, if they do not come at the same control step, I can very well mark this and can reduce the number of multiplexers. If for example, a and c could be put in one, register and b and d could be put in one, then I would have saved in one multiplexer. Thus we can try to do some connection or bus sharing or multiplexer minimization. We can do register merging, we will show all these things gradually to this example.

(Refer Slide Time: 26:04)

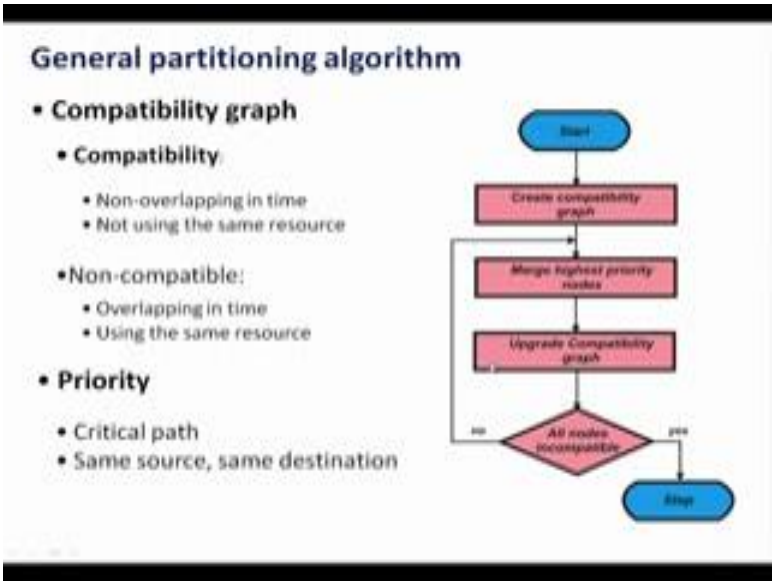


Now, let us see some concepts like how we can gain by register sharing. Here again look at this I had mentioned this example here you see, x assigned a plus b. So, there is an addition operation, and y assign c plus d there is an addition operation, but these are (Refer Time: 26:31) to disjoint control steps, in state s i and state s j clear. So, now I have got four variables; a b c d and therefore, registers, just the example that I was drawing here that is being shown here, the example that I was showing here, the same thing is being shown here, that there are two multiplexers, and this one is taking either a, because either at this point this adder. Here I am using please note I am using one adder. Why am I using one adder? Because this edition and this edition are at two disjoint control steps. So, I can merge these two additions to one, at the cost of additional selectors, at the cost of additional multiplexes, and I connect in this way a or c will come here, b or d will come to the right input of the adder, and accordingly the output will go to either x or y.

Now you see since they are coming at disjoint control steps. I can very well merge a and c, because a and c are never live together. There is no point in this simple computation, that both a and c are being red simultaneously; that is not the scenario. Therefore since they are being done at disjoint times, let me take a and c and merge them into one register. Similarly b and d I merge two in one register. Therefore, I still have this selector there are two selectors at this level. And similarly x and y, x is not being used here. So, x and y you are being margin kept in here. So, this is the data path that we get by little bit

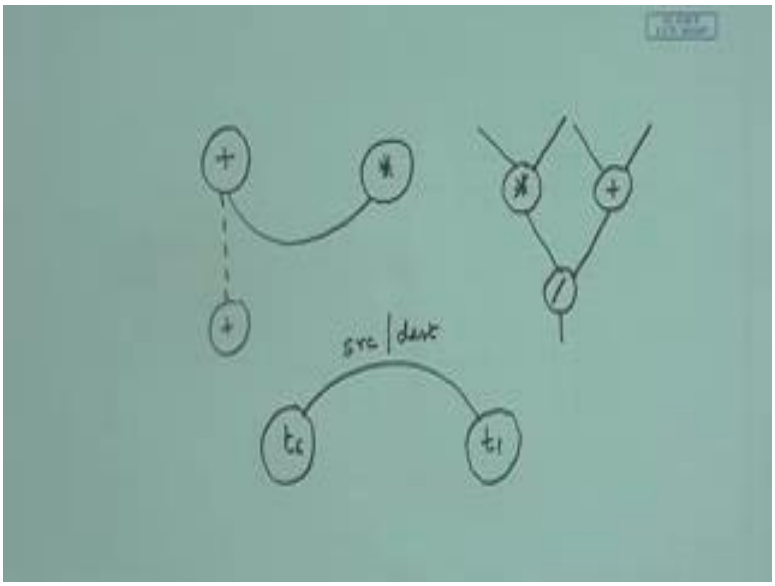
of optimization, by merging the registers. Still I am needing, in this process I have merge one selector, but two selectors, these two selectors are still there.

(Refer Slide Time: 28:45)



So, how do you do this sort of partitioning, I mean how do we do this sort of partition. I mean which one can I merge, which one rather. Let me call it I mean, how do you decide which two can be merged together. For that the general algorithm is here. We form a compatibility graph. What is the compatibility graph? A compatibility graph as you all knows that the nodes, a graph consists of nodes in ages, and say two operations; say 1 is plus 1 is multiplication, they are compatible. If they are compatible I will connect them, with a normal edge. If they are incompatible; suppose this and another, say this edition are incompatible then I will mark them with a dotted edge.

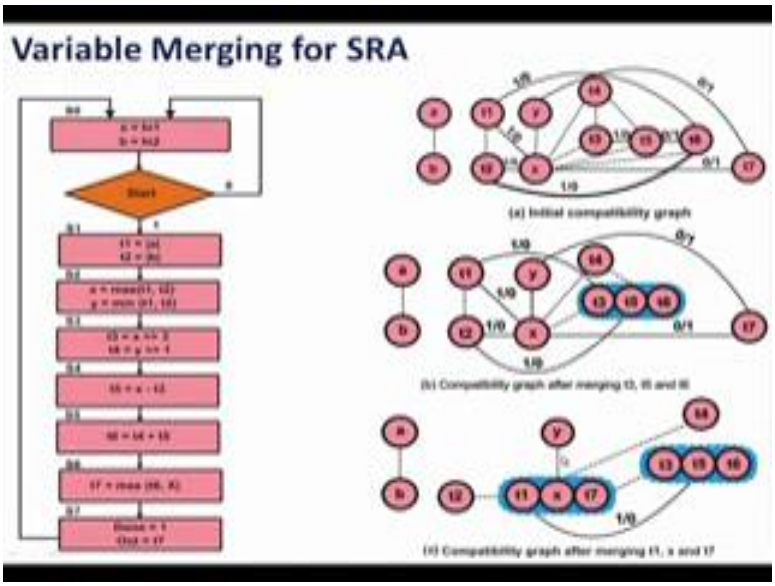
(Refer Slide Time: 29:24)



When do I say that two operators are compatible or two nodes are compatible? Two nodes are said to be compatible if, they are not using the same resource. If I have got only one adder, then and two variables are accessing them, then both these variables are coming on the left and right of that adder. So, they cannot be merged.

And if these two are non-overlapping in time, if these two operators are not active at the same time in that case is a possibility that they can be merged. So, they are compatible. Non compatible means that they are overlapping in time, at the same time the addition and multiplication are being done. Suppose I am I have decided in my schedule that I will do the multiplication here, and the addition here. So, in that case I cannot have them done through a same ALU; say here I am doing subdivision. So, I cannot do these two, will not be compatible if I fear the same ALU for them right they can they are overlapped in time. And besides there will be some priority which will see in the algorithm. We will mark the highest priority nodes will show that, and update the comparability graph, and see if the compatibility if all nodes are now incompatible. I have already covered all the compatible nodes in that case we stop. So, let us show it with an example.

(Refer Slide Time: 31:46)



It will be difficult to complete this in today's lecture, but let me proceed. So, here for this our example, this is the compatibility graph. Do not bother about these numbers on the edges right now, do not bother about them. Just think of the edges dotted edges and voltages, just let us look at that. So, here you see a and b, then; obviously, incompatible; why, a and b are being used at the same step, they are being used at the same step; therefore, they cannot be merged together. Similarly t 1 and t 2 are being written onto in the same step here therefore, they are incompatible.

But what about t 2 and x? t 2 is being written into here and x, I mean x is being written in the next positive edge of the clock. So, t 2 is being red here, and nowhere I find that t 2 and x are simultaneously active. On the right side; t 2 and t 1 are being used. So, t 1 and t 2 are incompatible, but t 2 and x are not being used right, t 2 is being used here, x is being used here, and t 2 is not being used anywhere else.

So, let us look at what will happen with t 1 and x, t 1 and x are should also be compatible, t 1 and x are compatible. Let us look at t 3 and x t 3 and x are not compatible, because it is a shift operation, and the shift operation is being done at the same cycle. So, now x and t 7 let us see, t 7, t 7 is on the after the computation of max, after the computation of max t 7 will be (Refer Time: 34:20) to. So, they can be put in the same register if possible. t 5 and t 6 let us see, t 5 and t 6. So, they have got nothing in common right. t 5, x and t 3 are incompatible, t 4 and t 5 are incompatible, where is t

4, and t 5 are incompatible, t 6 and x are incompatible, t 6 and x are incompatible here you see . So, in that way we can draw the compatibility graph of the entire. I mean here we are only looking at the variables, we are this compatibility graph is being used for with the purpose of merging the variables. Right now we are not merging the operands here yes.

Student: (Refer Time: 35:22) variables has discrete all the variables (Refer Time: 35:30).

The question is that a or b, a and b are not being checked with other variables, why is the question is that, why is where I am checking all these cross product actually t 1 with x t 2 with x and all those, why am I not doing it for a and b. The reason is obvious, because a and b are not being used anywhere inside, after once I come here a and b are not being used anywhere, a and b are compatible with all other nodes here. Nowhere else a and b has been used further. Once I have shifted this a and we have taken the modulus of a I have come to this t 1.

Student: (Refer Time: 36:15) incompatible some other (Refer Time: 36:19).

Not exactly, I mean you could have very well done a and t 2, you will get a compatibility always, a and t 2 you would get a compatibility. So, all everywhere else is, it is very obvious, because this a, the writing into a has taken place here itself and after that we are starting that is why, it is an initial allocation otherwise, you could have drawn. You are right, you could have drawn it also there is no harm, in drawing the entire compatibility graph. Here we have just used some intuition that a and is compatible with everything else, so we are not dealing with that, there is no other special reason. So, once we do this we have got several options. We can see t 3, t 5, t 6 are compatible.

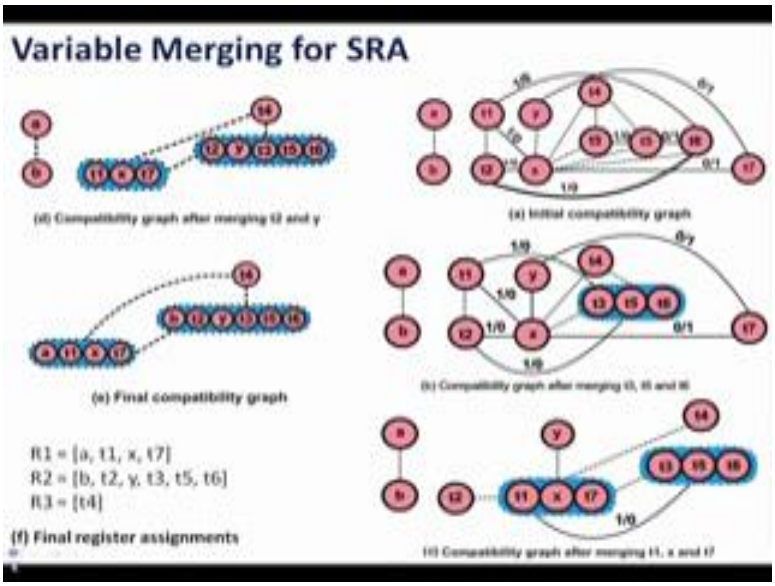
So, I can merge them, I can see that t 1 t 2 x and t 7 are compatible. There is no incompatibility among them. So, I could have merged them in the same register, I could have merge t 1 and x, so which one to select. For that I may like to adopt some optimization I might like to take some cost functions into consideration, but right now I am not doing it right now I am arbitrarily say I am making a choice. It is also not very arbitrary I will tell you why. Suppose I am merging these t 3, t 5, t 6. Now at this point if I merge them, so I get this. Now once I merge them, you see the compatible relation of this cluster is a complete cluster. Now t 6 is compatible with to t 1. So, where is t 6? t 6, t

6 and t 1. So, this entire cluster is now compatible with t 1 you see, t 6 was compatibility with t 1, so this is entirely.

Now, I just come to this, come to this, weights on this edges what are these weights on the edges. When I show that between t 1 and t 6, this edge t 1 and t 6, is 1 slash 0; that means, how many sources and destinations are they sharing; that means, I have got one node here; say t 6 and there is another node here maybe t 1, and I am putting a compatibility edge. I have done that only for the compatibility edges, not for the incompatibility edges. I say how many shared sources, how many shared destinations. Let us have a look at this. t 6 and t 1, where is t 6, t 6 has an input t 4 , and t 5, t 4 and t 5 are inputs to t 6, and t 1, where is t 1. t 1 and t 6, t 1 is taking input from mod of a, and t 1 is, I am sorry, t 1 is sharing is serving as a source to this. So, based on that I will discuss it in detail in the next class, this will take a little more time. So, based on these allocations we can take the decision, whenever I have got a choice, that whether I can select this one or this one, I can use this information, this information I will do well on that again tomorrow in the next class.

Based on that I can do the clustering, so suppose I made the clustering here. Let us have will start from this again. So, quickly let us see, suppose I have clustered this. Now I find that with t 5 t 6 this cluster who else is compatible. I can see that, I have not merged t 1 I could have done here, but t 1 x and t 7, this one is also a path right compatible paths t 1 x t 7, suppose I have cluster that, I have clustered them. So, in that way I can go on clustering, and I can go on further clustering here. So, I find t 1 x t 7. Now, who else? Now these two are, these two are compatible, these two are compatible right. So, this class; that means, all nodes, when is a cluster compatible to another clusters, when all nodes of this cluster are compatible to all nodes in this cluster. Therefore, straight away I can merge, these two these two I can simply merge right.

(Refer Slide Time: 41:51)



So, I can merge this further. So, t 1 x t 7, but here I have done.

Student: (Refer Time: 42:05) not compatible in these two (Refer Time: 42:09).

In these two plus, no here, oh here I am sorry, there is a dotted line I am sorry you are absolutely right. So, I cannot merge there is a dotted line, there are some edges, there are some edges which are incompatible; therefore, I could not merged them. So, this one is one, this one is another now these are all incompatible. Now a and b are independent, they are compatible with everything. So, I can randomly take a here and can take b here, I could have done it the other way, I could have taken b here, b here and a here. So, ultimately when I do this clustering, then I am left with three clusters t 4 this cluster and this cluster. Accordingly I can allocate three registers only instead of. So, many registers r 1 will be a, r 1 will how will take care of the variables a t 1 x and t 7 up to will take care of these variables, b, t 2, y, t 3, t 5 t 6, and r 3 will be t 4, because that is not compatible with anything.

There is not compatible with anything means, there must be some time sharing among these variables are resource sharing. So, based on that will come back to this in the next class and we will see how we can utilize this compatible notes further, and I will also explain in detail this sharing, the values are (Refer Time: 43:40) edges in the next class.