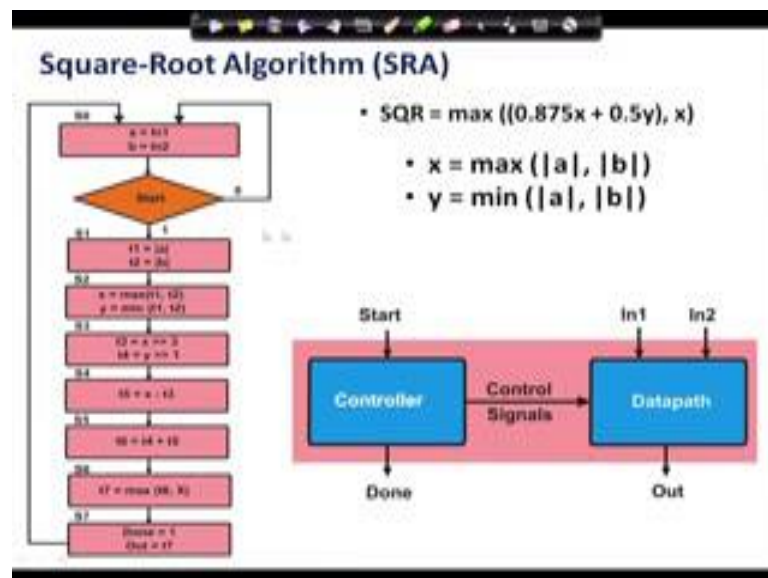


Embedded System Design
Prof. Anupam Basu
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 41
Hardware Synthesis – II

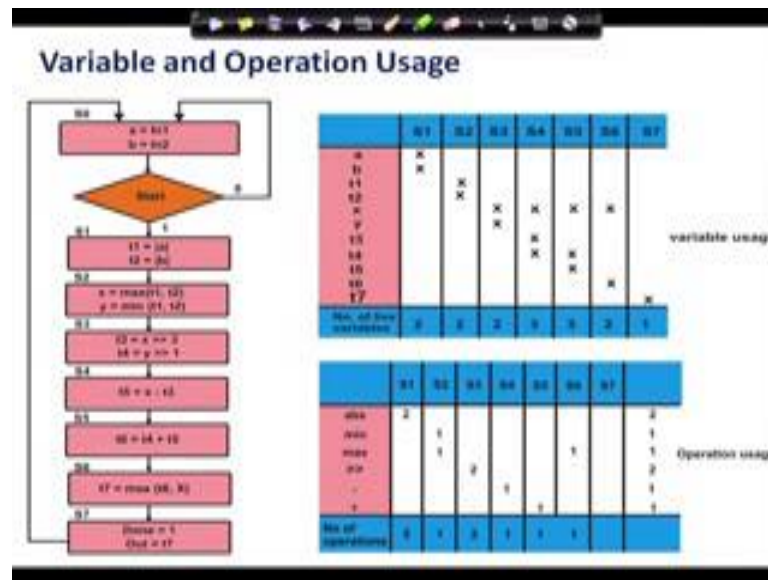
So, we again we will start discussing about high-level synthesis. In the last class, we have talked about profiling and all those things, but some of those things I would like to repeat once again today because I felt that it was not very clearly explained in the last class.

(Refer Slide Time: 00:44)



So, if you just put you in perspective you can see that this was control and I mean this was the floor FSM of the problem that is a square root approximation algorithm using this.

(Refer Slide Time: 01:04)



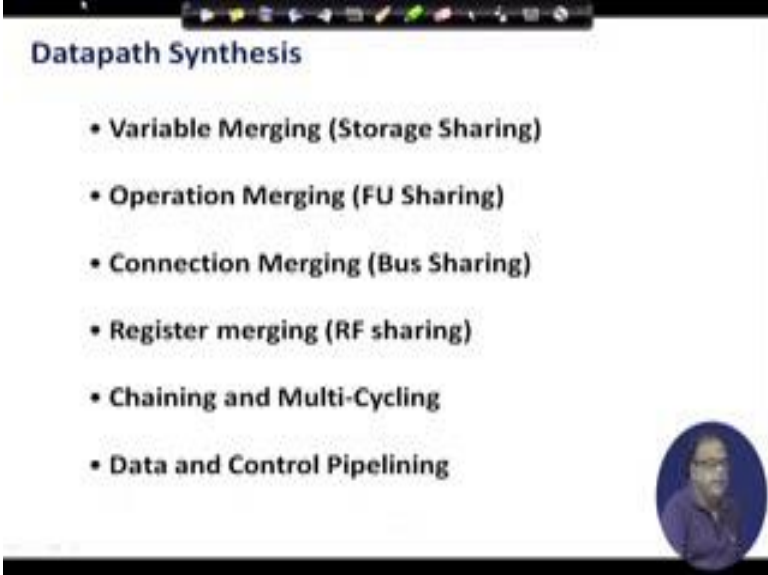
So, when we are trying to do the profiling we came to this. Now, what I would like to mention is that these points I mean I am making a matrix and in the matrix you can see that these are all the variables, these are all the variables. Now, I put a cross on the particular element depending on the liveness of the variables for example, I find that at state s 3, x is live say here, x is live here; before that x was never live right, x becomes live here. And x remains live up to this, x remains live here also in the s 4 and also x is being used in s 6. Therefore all through this x is live.

Similarly, here say s 4, in state s 4, I am having x live I am having t 3 live, why, t 3 started here in from state s 3 and t 3 is being used up to this s 4, so that is also live. What about t 4, t 4 was made live at s 3; and I mean it was written after s 3, and it is live in s 4, right because t 3 is being used here you can see that. Therefore, this one is live this s 4 has got three variables which are live. So, you can see that these three variables are live in state s 4.

And similarly in s 5 you can see that of course, variable x is a live because it was initiated in s 3; it is also available in s 4 it is live; in s 5, it is live because it is being used in s 6. Therefore, it must be live here also. And what about t 4, t 4 was born here; I mean just after this. So, t 4 is live here is being used here, therefore it must be live for all these states in that way we make the liveness matrix here. Similarly, we are making this operation usage matrix in which state they are being used, so that is how in the last class

we are hesitant to explain this. It is not on the variable usage it should be rather I should say it should be variable liveness, it is variable liveness.

(Refer Slide Time: 04:42)



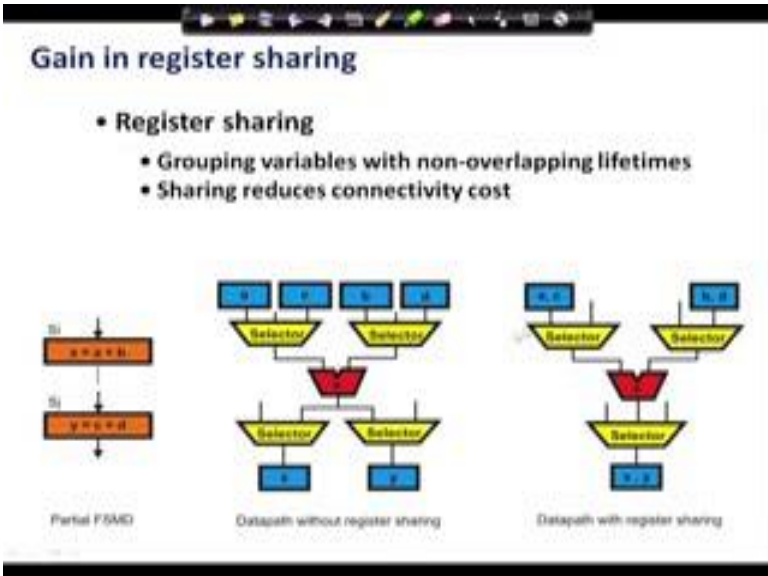
Datapath Synthesis

- Variable Merging (Storage Sharing)
- Operation Merging (FU Sharing)
- Connection Merging (Bus Sharing)
- Register merging (RF sharing)
- Chaining and Multi-Cycling
- Data and Control Pipelining

A small circular portrait of a man with glasses and a purple shirt is located in the bottom right corner of the slide.

So, next we move to, so based on that that was the profiling. And we will be doing different steps like storage sharing, functional units sharing that we discussed in the last class.

(Refer Slide Time: 04:51)



Gain in register sharing

- Register sharing
 - Grouping variables with non-overlapping lifetimes
 - Sharing reduces connectivity cost

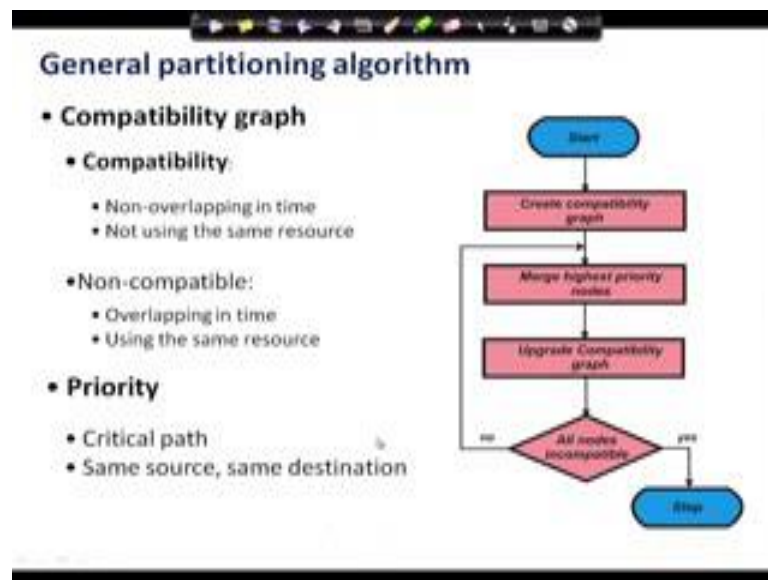
The slide contains three diagrams illustrating the gain in register sharing:

- Partial FSM:** A vertical flow diagram showing two sequential operations, `x = x + 1` and `y = y + 1`, each in an orange box. Arrows indicate the flow from the first operation to the second.
- Datapath without register sharing:** A logic diagram showing four registers (R1, R2, R3, R4) at the top. R1 and R2 feed into a first Selector, which outputs to a Register File (RF). R3 and R4 feed into a second Selector, which also outputs to the RF. The RF outputs feed into two more Selectors, which then feed into two separate Register Files (RFs) at the bottom.
- Datapath with register sharing:** A logic diagram showing two registers (R1, R2) at the top. R1 and R2 feed into a single Selector, which outputs to a single Register File (RF). The RF output feeds into a single Selector, which then feeds into a single Register File (RF) at the bottom.

So, and we also saw this that how we can gain by combining registers depending on the liveness of the registers this was discussed and I think that was clear. Just to repeat you

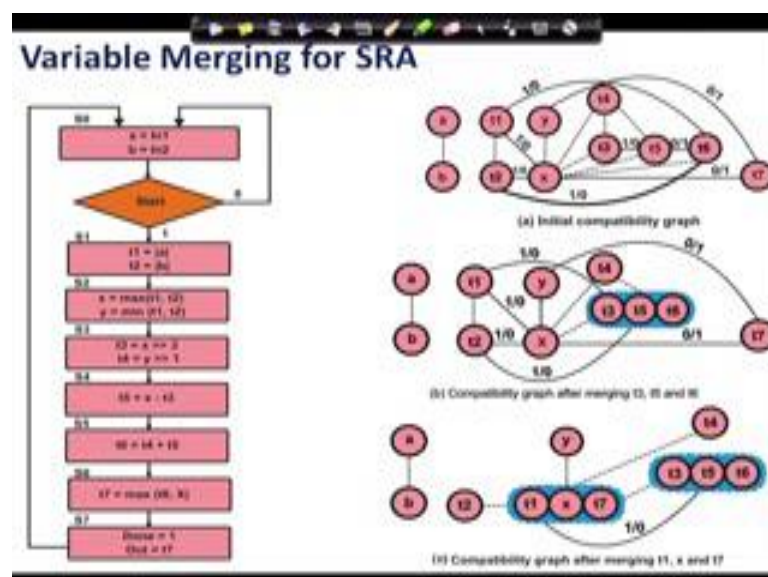
can see that since a and c are not being used in the same step. So, they can be combined to be one register, and thereby I save on I mean number of registers here. And here x and y being combined into one, I can save in the number of selectors here, so that is clear.

(Refer Slide Time: 05:32)



Now, when we talked about this compatibility graph, there were some confusion.

(Refer Slide Time: 05:40)



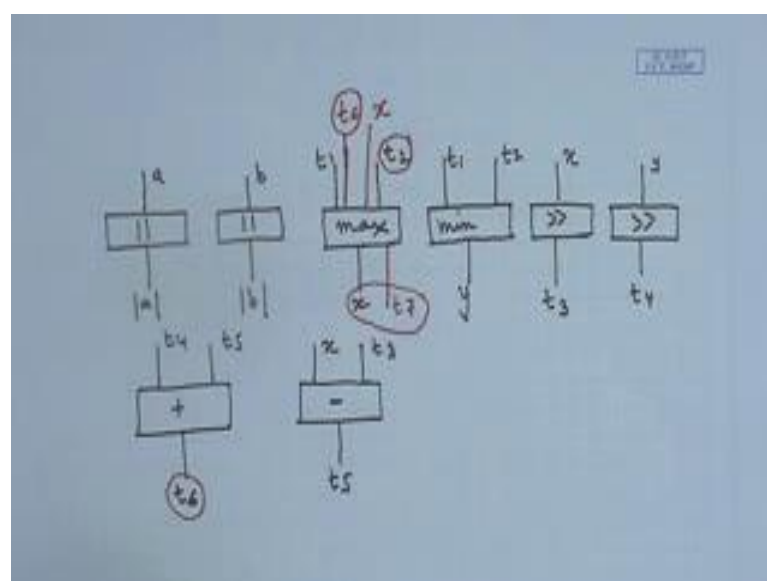
So, let us work it out once again today. So, for the compatibility graph part, first we have to identify the incompatibility. So, I have got my variables, my objective of using the compatibility graph now is for looking at the variable margin, how I can combine the

variable, and once that is clear the same thing will apply for functional unit merging and others. So, I can see I have got variables a, b, t 1, t 2, x, y, t 4, t 3, t 5, t 6, t 7.

Now, first I connect the incompatibility edges, just the incompatibility edges of the edges which are occurring at the same time or sharing the same resource. So, you see here a and b are occurring at the same time, so I add this no edge to it. Similarly, for t 1, t 2, they are being used at the same step, so they are incompatible. What about x and y, x and y are being used in the same step they are incompatible. X and t 4, why are they incompatible, x and t 4, here x is being used and I need to keep the value of x here x is live from this point, and when x is being shifted t 4 is also being written because the shifting is being done in the same cloth. Similarly, x and t 3 are incompatible. So, I first mark the incompatibility here. So, I get only the dotted edges I have not drawn any compatibility edges till now.

Now, anything the complement of the incompatibility graph should be the comp complement of the incompatibility graph would be the compatibility graph. But here I am connecting the nodes which are somehow I just want to put in some sort of a precedence of margin among these I am using connecting these based on which are the nodes that are using the same resource. For a second, let us look at concentrate on this part only. And let us look at what are the functional units that I am using.

(Refer Slide Time: 08:55)



Let us have look at the functional units. So, I have an absolute computation; absolute computation coming at this stage. And its input is say actually it is coming here. So, its input is a and output is absolute value of a. I am using another one which is b. Now, here I have got a max operation. And what are the inputs of max t 1 and t 2, and output of max is x. I have got the min operation here that min operation also has got t 1 and t 2 and I am getting y. So, here you see that max and min are sharing the same inputs. Now, here I have got a shifter and that shifter has got input x and output is t 3. I have got another shifter whose input is y and output is t 4.

What else is left? Plus and minus; now, this minus has got x and t 3; and output is t 5 plus has got t 4 t 5 and the output is t 6. Again I use the max here right, here again the max is there. So, I come back to this max. Now, at this max, again I am needing the inputs are t 6 and x. So, here I have got t 6, I am drawing it is a different color or make it a little elongated t 6 and x. Therefore, you see t 2 and x, t 1 and t 6 are using the same resource here.

Now, now looking at this usage pattern, let us try to see mark these edges where the resources are being used. For example, you see t 1 and t 6. What is happening to t 1 and t 6, you look at this t 1 and t 6 - these two variables t 1 and t 6 these two variables are being used in the same max; t 2 and x are using the using the same max. So, here you see and what are they sharing, they are sharing as the input they are sharing this component their inputs to the same component. Therefore, t 1 and t 6 they have got one input common, no output common this operator, so because when next time this max is coming then the output is t 7, so that is a different output.

Now, let us have another look x and t 7. What is our take on that x and t 7 x and t 7, let us come here x is using the shifter where is t 7 yeah sorry this max the output was t 7, the last max output was t 7. So, x and t 7 are using the same operator as the output. Therefore, if I can merge them that will be the same they say I can since both of them are coming from the same operator as the output in two different control steps; I can try to merge them. There is a possibility of merging. So, this margin possibilities are being shown here, so that is 0 1, I hope it is clear.

Now, let us once again try t 6 and t 2, what happened t 6 and t 2. Let us look at this that again on the paper. t 6, so what is t 6, t 6 is here and where else is t 6 max t 6 and t 2 yes,

t 6 and t 2 are sharing the input of the same max. Therefore, they are one sharing I can because see this max it is commutative evaporation. So, I can very well make the left input the right input that is not an issue here. So, therefore, that is also 1 0, so that is how we make this compatibility graph where we mark the incompatibilities and then give the precedence of the other possibilities of margin with 1 0 this sort of thing.

Now, last class we are talking about that if I merge say this one, this one, I could have merged something else also this t 3, t 4, t 5, t 3, t 5, t 6 they are compatible. Similarly, by merging them what do I gain, I gain in one source point and one destination point, you see. So, I can merge two variables here. Here if I had merged only these two t 2 and x, I would have gained in 1. If I had gained in if I have merged t 2, x and t 7 that would have yielded me the same result that 1 1, so I had a choice of merging these three and this one or these three.

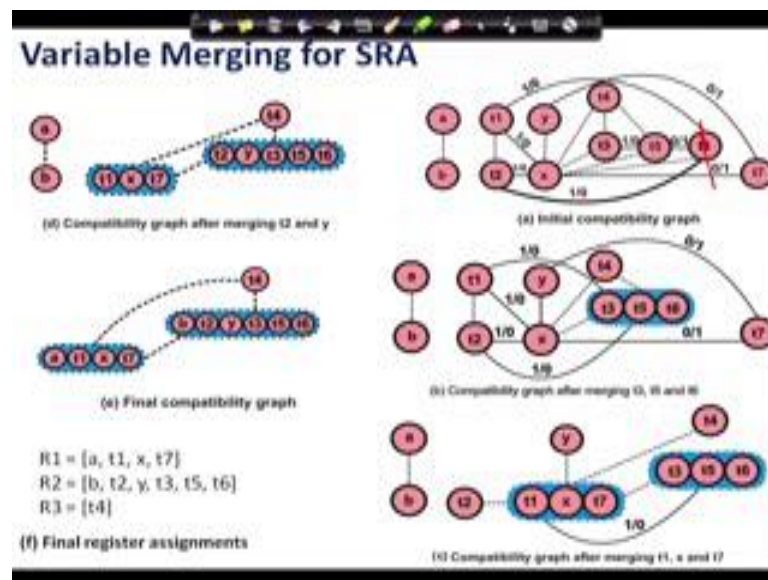
Now, this sort of decision making will be coming up at every point of our design algorithm. And there are different ways, different heuristics that are adopted different heuristic measures that are adopted to decide on which one to be adopted which one I should go ahead. Those of you are aware of the algorithms now of course, I can go by some greedy method or I can go by some branch and bound or I can take some linear programming approach, different way approaches can be done to find out what will be optimum point.

So, here we have taken this t 3, t 5, t 6 to be merged. Now, once that is merged what else can come with this we saw this in the last class, this one can come t 1 x t 7, t 1 x and t 7 that is that can also be emerged because that will give me a benefit of again 1 1, 1 1 the same thing. So, I merge them as soon as I merge them then t 4 became incompatible with this cluster why which was otherwise compatible with x has now become incompatible with this cluster right can you see that. And t 2 which was otherwise compatible with x has now become incompatible this cluster, because t 2 was incompatible with t 1 and I have brought t 1 in this cluster. So, now I am left with this. Now, what can I do I cannot bring it in he here, but a and b I can bring in; t 2 was not incompatible with this, so I can take t 2. So, I can take t 2 in the, yes, what did you say what.

Student: (Refer Time: 19:11).

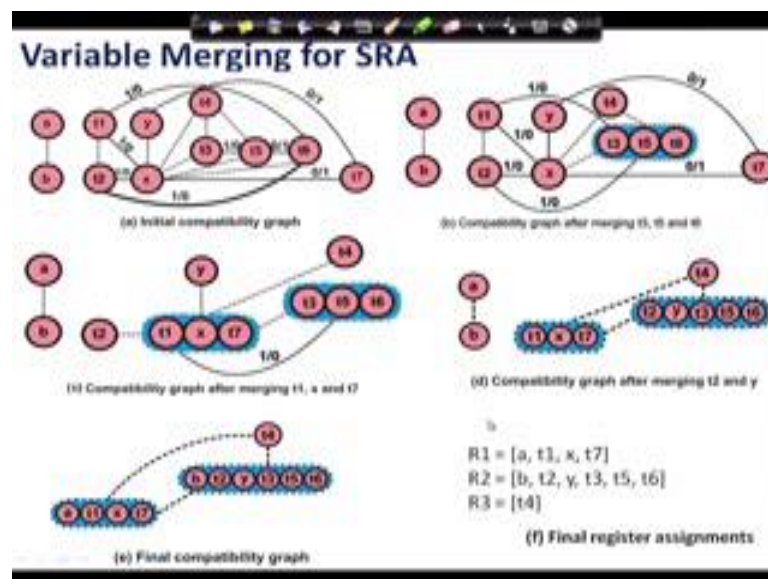
Professor: That was a mistake, here this should be a dotted line this yeah that was that was. If x and y , let us see x and y are incompatible or compatible, they were incompatible, because they were in the same state. So, the here is a mistake this should be a dotted line as it being shown here.

(Refer Slide Time: 19:42)



So, now, let us look at this therefore, now what is happening with this.

(Refer Slide Time: 20:06)

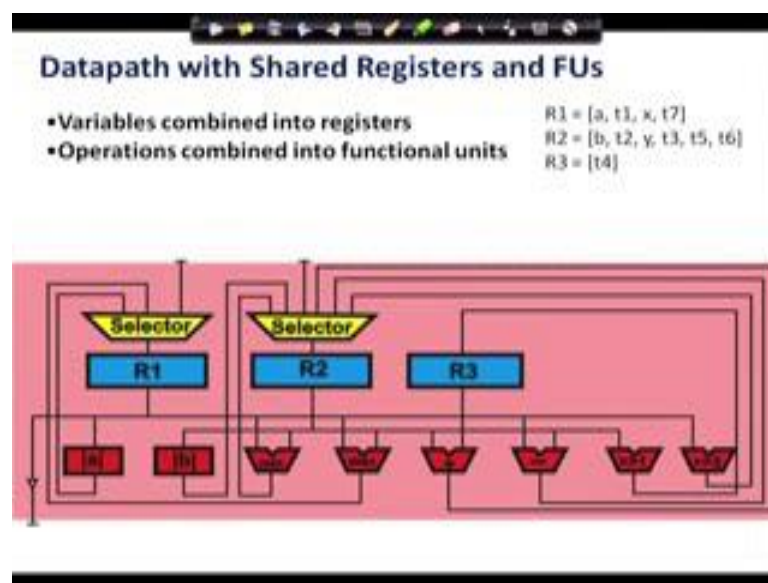


So, now I will further try to I mean continue with this merging. So, now I have got this. So, from there, I came to this, now the third one is once these are done then I can merge

which one can I put in over here. I can merge t 2 and y, these two can be merged together and t 2 and y can vary well come to this cluster, because they are compatible with this. So, I can have this sort of a clustering. Now, a and b, a and b are I mean they are not at all in any conflict with this, so it can go in any cluster. So, I could have merge b here and a here, I could have done it in the other way also.

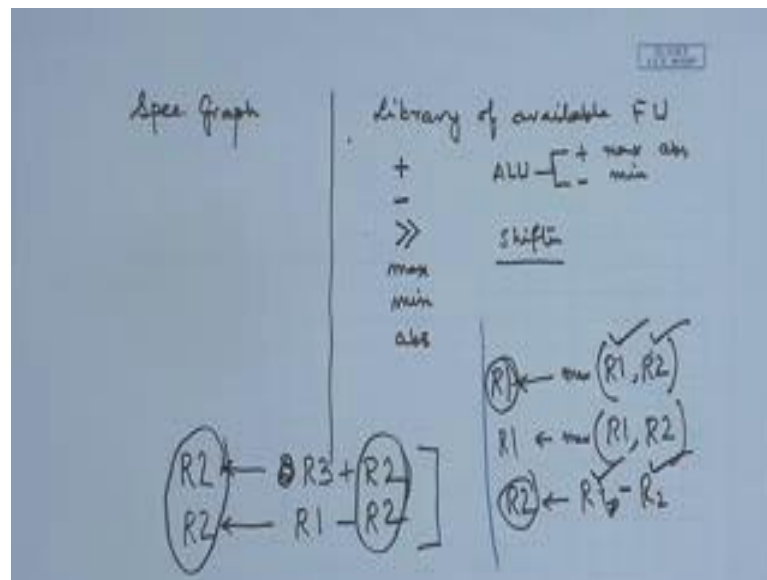
Therefore, I have now so given all those so how many variables I had, I had a, b and x, y 4 and t 1, t 2, t 7, so many 11 variables. Now I can I have been able to merge them into three registers; t 4 does not go with anything else therefore, t 4 will be in one register R 3; and a, t 1, x, t 7 goes in R 1; and b, t 2, y 3, t 5, t 6 goes in R 2. So, now, I have got three registers, so that is how we have been able to merge the variables and we get three registers, so that is how we carry out. We are now trying to do the optimization at every step.

(Refer Slide Time: 21:47)



So, now how would my data path look like, my data path will look like this that now I have got three registers and I have got all the explicit units. Now, one thing I should have mentioned here that when I created this residence graph, here I also looked at I actually looked at the library of functional units that are at my disposal.

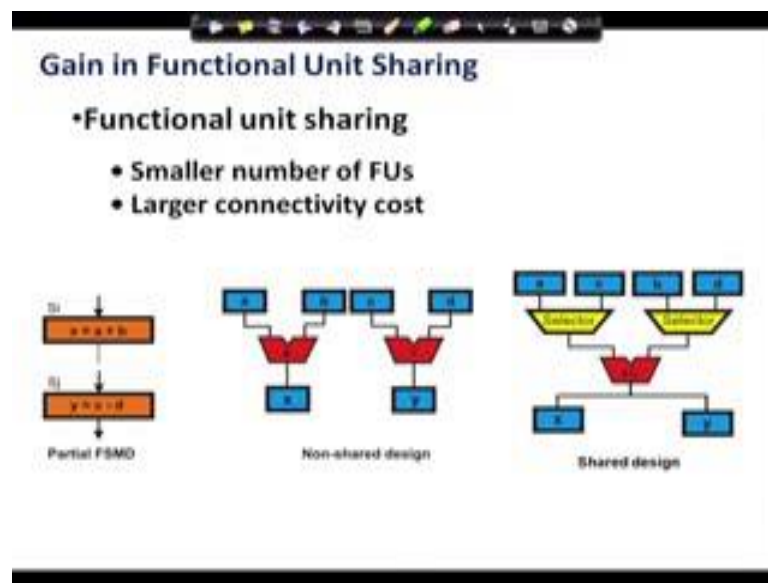
(Refer Slide Time: 22:27)



See, whenever I am doing it I am given a specification graph, I have got a specification graph. And on the other hand, I have got a library of available functional units. Here, when I made this graph I had assumed that there is an added, there is an subtractor, there are shifters, there are max, min and absolute comparatives absolute computations. Also I have assumed that there might be some ALU, which actually takes care of the plus, minus, max, min can be put in here that was another possibility and abs and shifter was something else separate.

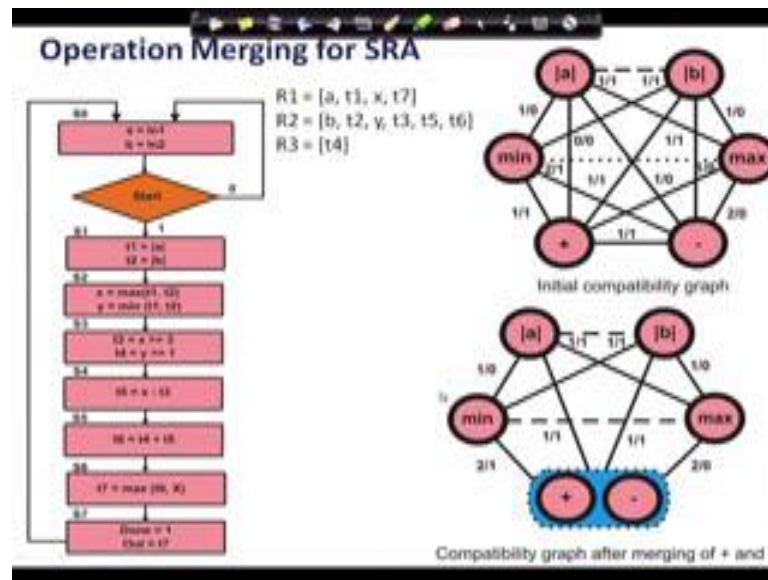
Now, this shifter we actually handles in a different way because I can actually have a shifter or I can do shifting on the fly right by just doing some adding some control step and I do not have to pay any I may avoid paying any hardware cost for that. So, given this, here we are showing that we are not economizing on my functional units or the library, I am not economizing on them I am using them explicitly and. So, if I do that, I have got a, mod of a, mod of b, min, max all these things two shifters. And this is the data path and these are the registers which holds this variable fine

(Refer Slide Time: 24:18)



Next, what do I gain by functional unit sharing. I again come with an example here the same example. Say I have got a plus b going to x and c minus d going to y, and here I find that a and b, c and d are coming to two different functional units, but very well I can merge them into one ALU which can do both addition and subtraction. And assume let us assume that the cost of the adder plus the cost of the subtractor is less than the cost of the ALU. If that be so then it is prudent to have this and consequently what are we gaining. So, not only this, we are adding two selectors, because now I need to include the selected. So, the cost of this plus the cost of the selectors should be less than these two addition, and here I can send it to x and y here is only thing, so that is what how I can gain by sharing functional unit.

(Refer Slide Time: 25:29)



Now, this is a simple case. If I come to the operation margin for our square root algorithm now, what are my operators, my operator were as was shown here; all these were the operators 1, 2, 3, 4, 5, 6, 7, 8. Now I have kept out the shifters, the reason I have already told. So, I have got six keeping out the shifters, because the shifters I can handle separately. So, here you can see again in the same way, you confirm that a and b are incompatible I am sorry, but not the variable a and b. I am saying this operation mod a and operation mod b are incompatible. Why? Because they are taking place in the same step mod a and max are not taking place at the same step.

Now, let us look at I have marked it 1 1, why? Let us see what happens to this mod a mod a goes to t 1, where is t 1 located, t 1 is located in R 1. So, t 1 can be synonymously attributed the value R 1 the name R 1, t 1 has been put in R 1. And what happened to max, max is taking t 1 and t 2. So, max is taking t 1 and taking x. Now you see, so t 1 max is sharing t 1 that is the R 1 with mod of a. And also as an output it is sharing x with mod of a with sorry with R 1. So, R 1 is being shared both of the destination also the source. So, I marked it 1 1.

Let us take another example plus and minus what happened to them plus, where is plus, plus is here. So, this plus t 6 assign t 4 plus t 5, I can now write as if I just go by RTL, register transfer logic what would it be, t 6 means for which one can I write this t 6, t 6, where is t 6, R 2. So, R 2 is getting t 4 plus t 5. What is t 4? t 4 is R 3 and t 5 is R 2, so R

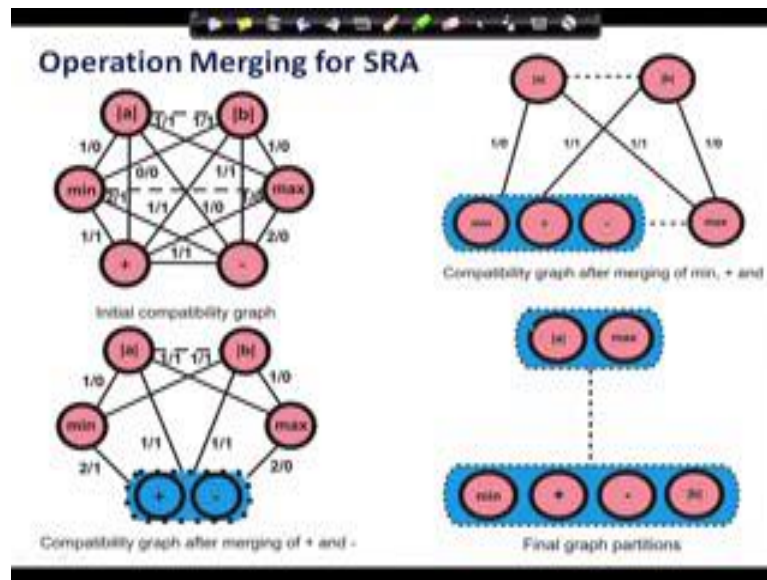
3 plus R 2. And what happened to the minus, what is minus here? Minus is t 5, t 5 is R 2 and what has been subtracted x which is R 1 minus t 3, what is t 3 R 2.

Now, looking here we can see that one destination is shared one source is shared therefore, it is 1 1. In that way we can make this graph clear. Now, once we make this graph now we go ahead by combining and there also we look at the gain, I could have merged these two, I could have merged these two, because they are giving the benefit of two here I am getting benefit of two. Here max and minus let us work it out again just to make the thing clear.

What is happening to max, max is taking t 1 t 2 and going to x t 1 is where R 1, and t 2 is where R 2, and these two are being maxed and it is going to x, x means R 1. Max is again here there is again max here t 6 and x, t 6 is R 2 it could be R 2 I have already merged that. So, x is R 1. So, you see for both the operations that is the beauty of the merging the variable merging that I have done earlier that now these two variable these two registers are containing all the variables for the two instances. So, I do not need to do anything else. And where is that going to t 7 and t even is R 1. So, now, you see what I am I gaining here are both of them going to R 1 t 7 and what is t 5 sorry if I take this t 5 sorry max and max and max and difference sorry. I have to take the difference I am sorry the differences are x and t 3, x that means, R 1 and t 3 is R 2. So, I take a subtraction of them and store it in t 5 is R 2.

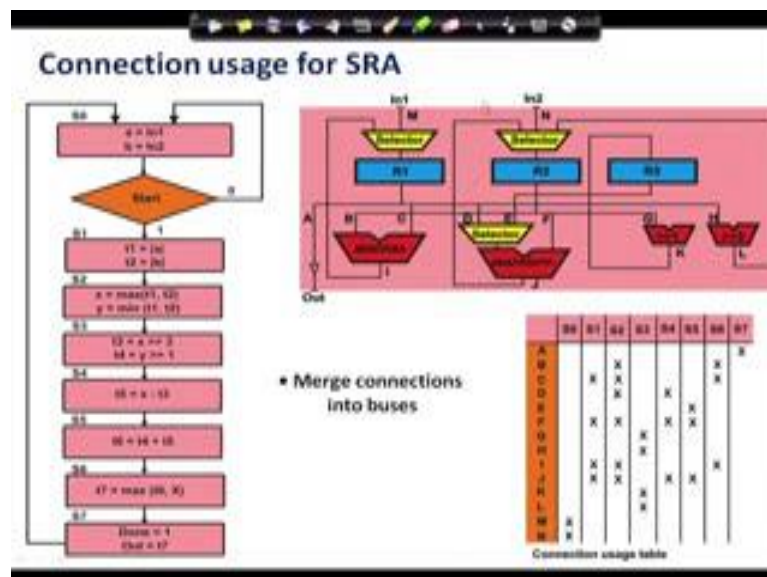
So, here we are sharing two sources, the destinations are different, destinations are different, but the sources are the same. Therefore, it has been marked as 2 0 clear all right now. So, we go ahead with the margins. So, I first merge these two, I could have merged these two as well. Next, I go on doing this. So, these two were merged then I merge mean with this, max cannot come with this because of incompatibility. Now, other things can come in a and max has been put in because they are compatible. So, I can merge these two then this and a and max has been merged and all these, b is we can easily come to this cluster we can come therefore, I have got this as a final graph partition clear. So, I have got now, therefore, besides the shifters what else do I need, I need two operators two operators two functional units, so that is how I have reduced it

(Refer Slide Time: 32:24)



And these two are incompatible might be they are active at the same step or they are other usage patterns.

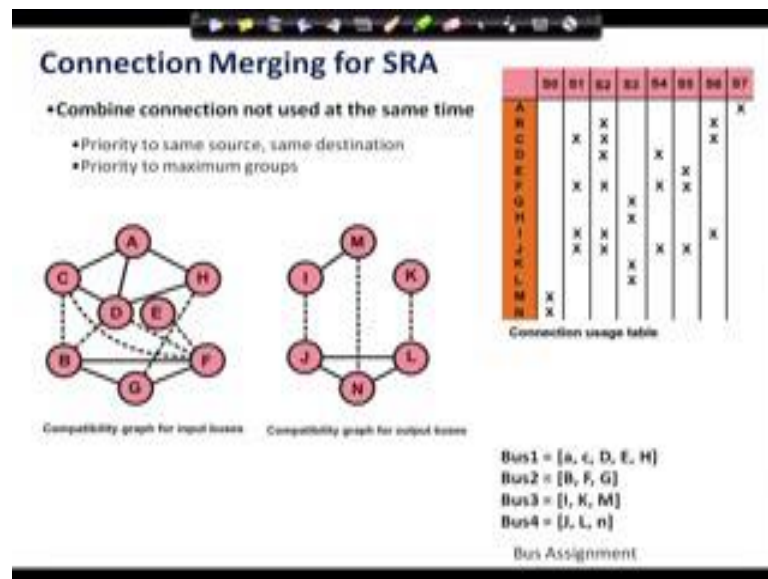
(Refer Slide Time: 32:35)



So, ultimately I come to this abs and max one abs and abs mean plus minus that is another one. So, I have put in some selectors and shifts have kept separately. Next quickly let us look at till another thing that we need to optimize in high level synthesis are interconnects. Those of you who have worked with such design you see how couples are the bunch of where I mean and when you got the p c b s and all those the

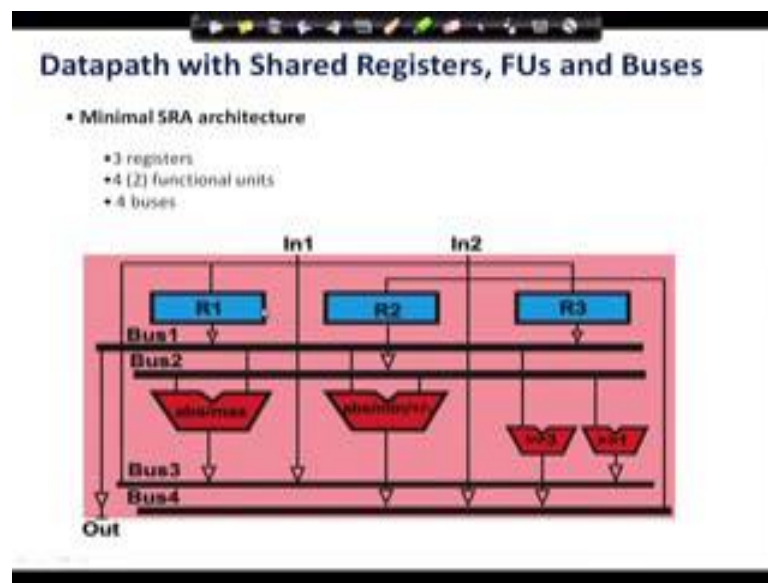
interconnects play a very important role in term of realistic as well as for other purposes other benefits. So, here we are showing the interconnect points a, b, c, d, e, f, g, h each other in points and the output points are i, j, k, l. So, I will write down i, j, k, l and here these m, n, so I write them down. And again I look at the connection usage table just as we are done earlier. Now, in the same way, we can merge the connections into buses.

(Refer Slide Time: 33:50)



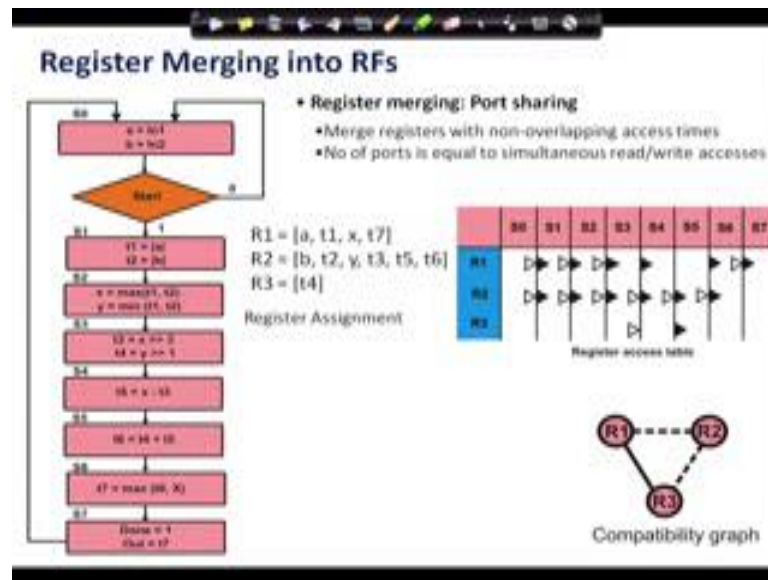
Now, what will be our compatibility now, the compatibility is that whenever now here I have not put any compatibility means whenever two connections are active at the same time I cannot merge them. Two distinct connections say for example, a is active here look at this table, a is active here, and c is active in other states which are disjoint right. So, a and c are compatible. What about a and b, a and b should also be have been compatible, but I mean c and d, c and d are they coming at the same control step, no, in two different state. So, based on that, I can draw this graph, and in the same way I can merge the graph and get busses. All interconnects I connect. Suppose I do this and I therefore, I get interconnects A, C, D, E, H this should be capital I am sorry. So, I will get four buses. So, in the same way, I am not spending too much I time on that you can yourself do the exercise and see that.

(Refer Slide Time: 35:06)



Ultimately, the design which was gradually was being defined and came to this scenario. Now, after merging the buses we could come to this, where I have got only four buses, four functional units. Now, it is now your decision you could have merged these two also, it could have merged these two also, but now I have got a much neater design not only neater design, but much more cost effective and cheaper design both in terms of realistic as well as cost. So, similarly last thing that we would like to do is merging the registers into register files. Now, I have got these registers R 1, R 2, R 3. Now, you know register file, register file is the box registers. So, when I can access the register file I access the register file at a time.

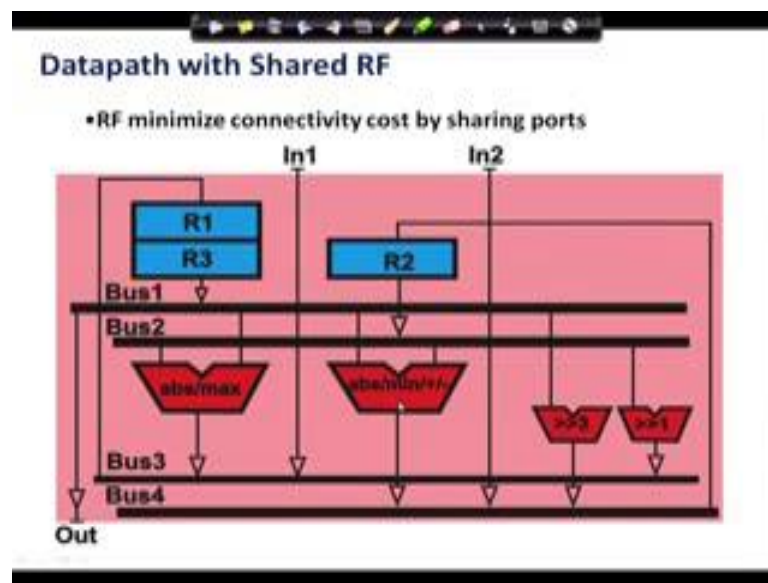
(Refer Slide Time: 36:12)



So, can we merge this further? You see my R 1 is a, t 1, x, t 7, all these variables are there. Now, if you just apply your mind you will see that I actually do not need three registers, because they are now clearly now I have not looked at, at which clock cycle they are reading or writing. Here what happens is at the first clock cycle and state s 0, this one is being written R 1 is being written here, because a is there. So, R 1 is being written here. And at the same clock I mean here s one in state s one I am reading R 1 I am reading R 1 here.

Now, in that way, I just plot this that this is being let us take R 2, at state s 0 R 2 has got b, so that is being read R 2, b is being read and b is in R 2 written sorry written, b is being written. And here the next clock b is being read all right. Similarly, for the state s 1 at this point R 1 state s 1 here, R 1 has been read and all these are being done. As I plot in this way I find that R 1 and R 3 are compatible, because no where R 1 is being accessed R 3 is being accessed, but that is not the case with R 2 R 3 is being accessed with R 2, here you see R 3 is being accessed to the R 2. But R 1 wherever R 3 is being accessed R 1 is not being accessed right (Refer Time: 38:36) it is the same clock. S 3, in s 3 I am taking x and x is in R 1, and shifting it. And y is in R 2, and y is being shifted therefore, both of them are being active, but R 1 and R 3 has got no conflict, they are being accessed at mutually exclusive times. Therefore, they can be merged and we can get a better result.

(Refer Slide Time: 39:04)



So, I merge them in R 1, R 3 in a register file, and I have got R 2 here. So, my design becomes much more effective. So, I stopped here this part. So, what have we done here we have seen that we start with the FSM or some specification and make an FSM out of it. And from there how we can do profiling and step-by-step we can go for optimization. Actually for larger circuits these things are done by software, but such optimizations, but you need to know what is the basic how say for example, you have to design an embedded system in which you have to make a small (Refer Time: 40:03) very simple (Refer Time: 40:05). And you do not have any access to any tool then this is the approach by which you should go about.