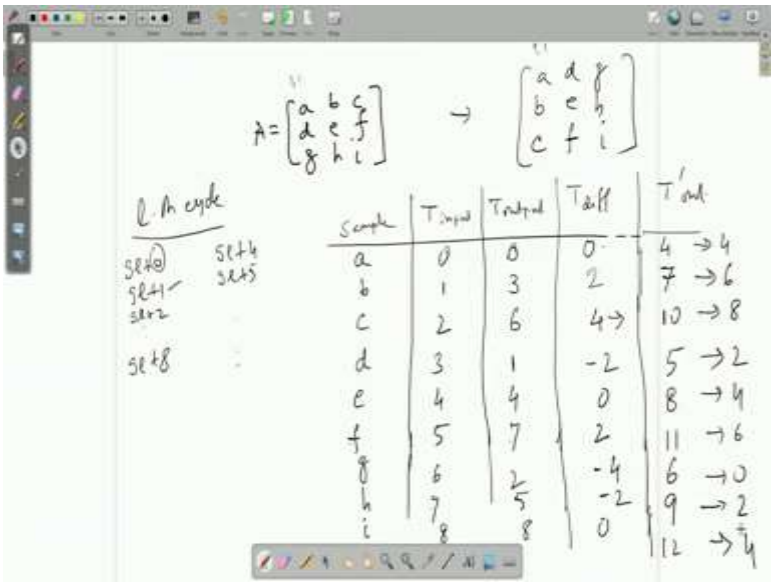
VLSI Signal Processing
Professor Mrityunjoy Chakraborty
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur
Lecture 25
Lifetime Analysis of Storage Variables

(Refer Slide Time: 00:33)



Okay in the last class we are discussing this delay optimization, so for that we have considered what is called lifetime chart, which gives us the minimum number of this thing, I mean, the maximum number of live variables at any stage. Okay because you need to provide that when you are registers. Okay, minimum number of live variables, or number of live variables at any stage.

And you find the maximum of them rather, I am sorry, you find the maximum of them because that many registers are required to solve them, save them. So that we discussed and then we ask this question that who will give you the description on the lifetime of the variables like each variable, if you consider a variable say a, in which cycle it is born, which cycle it is dead and so on and so forth.

And what will be the period of the system, because this will be periodic, right, same phenomenon will occur again and again. So for all this you really have to carry out what is called the lifetime analysis of the system. That is, you have to really investigate what are the variables,

in which cycles they get generated, in which cycle they get consumed that is they become dead, for how many cycles they remain alive, that is they require to restore its updated registers and what is the period of the system and all that.

To explain that, to demonstrate, we consider a very nice example from the book by Keshab Parhi because it is very insightful. So I repeat what we discussed last time briefly that suppose we are given a problem that we have to transpose a matrix, okay you have to transpose a matrix like capital A, is a 3 by 3 matrix you have to transpose it.

So transposition as you see is given here a, b, c, d, e, f, g, h, i that is A, and transposition is a, d, g, b, e, h, c, f, i, this matrix, okay. Now suppose this matrix A is coming cycle after cycle, new and new matrix are given to the system, system will read and then write a new matrix which is a transpose of the particular matrix that is read in that particular cycle.

So suppose in a particular cycle say Lth cycle, the system reads this. So what it does, it takes the whole up Lth input cycle divides it into in 9 parts, 9 sub-cycles that is 9 system cycles which is faster because each input cycle is divided into 9 system cycles. Okay so system clock goes up by 9 in one of them were just 9l plus 0.

It reads reader is positioned at this position, that is 1 comma 1 position of the matrix it reads A, writer also is simultaneously positioned in the matrix, which is to be feed. It is currently it is empty but in the 1 comma 1 location and they are synchronized, they move together. So in 9l plus 0, it reads the sample A.

So this 0 bits t input 0 bits 9l plus 0. It is not required to be consumed because writer will write it in the same place 1 comma 1, so no need to store it, that is why T output was 0 and the difference between them is 0 cycle, that is why I need only 0 cycles to store them. That is, I do not need any cycles. Okay, let us go up to this then reader moves at 9l plus 1, moves to b. It reads b, writer also moves here at 1 comma 2 position.

But then you see here it does not write b. It does not have to write b, it will write b only when it goes here that is 9l plus 0, 9l plus 1, 9l plus 2, 9l plus 3. So b will be written when the writer is in the cycle 9l plus 3. I am writing 3, the 9l is there that I am dropping just for the sake of, you

know clarity, so it is read in cycle 1 that is 9l plus 1 on the either side it is stored that is in consumed or you start with 9l plus 3.

Okay, so difference actually it is wrong. The difference between them is 2 not 3, 3 minus 1 is 2. Okay, then reader moves further at 9l plus 2 it goes to c but here writer also moves here, but it does not write c, when does it write c? When it is coming to this position that means 9l plus 0, 9l plus 1, 9l plus 2, 9l plus 3, 9l plus 4, 9l plus 5,9l plus 6. So 9l plus 6, writer is at this position that is 3 comma 1 position that is the time when he writes c.

So c is read in cycle 2 but written in cycle 6, 9l plus 6 so difference is 4, alright. Then reader moves further it goes to this position 2 comma 1, it is 9l plus 3, 0, 1, 2, 3, 9l plus 3 but it has to write b in this position where the writer was in which cycle? 9l plus 1, so input is 3, output is 1, 9l plus 3, 9l plus 1. So difference is negative. There is a problem it becomes non-casual.

Anyway, let us still carry on then reader moves further it reads e in 9l plus 4 writer also moves here at 9l plus 4. Here e is read and immediately written, no need to store, so 4 and 4. So I mean difference in cycle is 0. Then reader moves further at 9l plus 5 goes to read f. But f is required here at which cycle? 9l plus 0, 1, 2, 3, 4, 5, 6, 7 which cycle? 7. So it has to be stored from 9l plus 5 to 9l plus 7, so how many cycles? 2 cycles.

Then it moves to g, this position 3 comma 1, it reads g. But g is to be read written when writer was in this position 1 comma 3. That is when the writer was in which cycle? 9l plus 0, 1, 2. So that is why 9l plus 2 upgrade is returning 6, so again negative 2 minus 6 which is minus 4. We further carry on in 9l plus 7, reader comes to this position h it reads h but h is to be written here at this position which is 2 comma 3, so that was at 9l plus 0, 1, 2, 3, 4, 5 so 9l plus 5.

So here again, there is a casual in the problem read in cycle 7 but we written in cycle 5, so minus 2 And lastly i, this is read by this reader at 9l plus 8 and also written in the same place so 9l plus 8 so difference is 0. To take up the casuality problem what do we do now, we see the maximally negative value minus 4. So we say that let the reader start at 9l plus 0 and go ahead as it is but let there be a delay.

A latency, a delay in the writer, writer should write, start writing at 9l plus 4. Okay so only at 9l plus 4 it will be in this position in this empty matrix, okay which position? 1 comma 1 but reader

will be at 1 comma 1 in cycle 9l plus 0. If that be that means earlier output, earlier a was vague in 9l plus 0 which is same now but earlier a was written in 9l plus 0. Now I have to add 4 to that. So it becomes 4 alright.

And the difference is now 4 cycles. This arrow and the subsequent mean, after the arrow whatever I write, there is a new difference. What do the difference is 4, because this is delayed by 4, then b this fellow reader reads b at 1 as before but writer will write b while writer is here. What is that cycle? Writer is here 9l plus 4, 5, 6 so writer will have to write c sorry, I mean, yeah, let us consider b, so at 9l plus 1 reader reads b, no problem.

But writer will write it only when it is here, at what cycle then? Earlier it was at cycle 3 now it will be added by 4, so 4 actually 9l plus 4, 9l plus 5, 9l plus 6, 9l plus 7. So earlier it was 3 now 3 plus 4, 7. So difference is 7 minus 3, sorry 7 minus 1, 6 earlier difference was 2 so 2 plus 4 either way so 6. Then reader goes to c in cycle 2 but writer will write c at this position right.

So that is earlier c was written at 6 so 9l plus 0, now it will be written as 6 plus 4, 10 that is 9l plus 4, 5, 6, 7, 8, 9, 10 so 10. So read in cycle 2 for remaining 10. So difference is 8 so it has to be store in longer duration, okay store for longer duration, earlier it was only for 4 cycle now it is 8 cycle. Everybody is getting added by 4.

Then d, d is read as before at cycle 3 that is 9l plus 3. But writer will write it at when the writer finds it in this position that is 9l plus 5, okay 9l plus 4, 9l plus 5. So d is 5, that is 1 plus 4. You can see that is also 5. So cycle read in 3, written in 5, so difference is 2 earlier differences was minus 2.

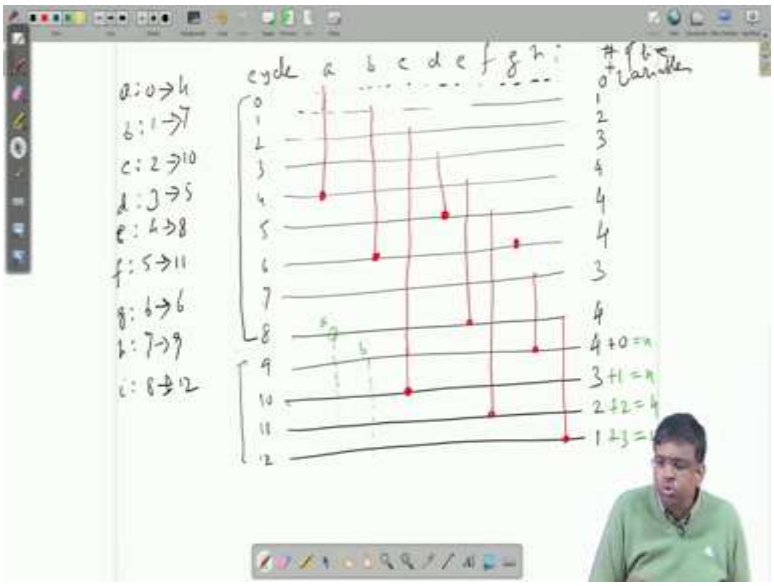
Now I have got rid of the minus c. They come to e, e was read in cycle 4 written in cycle 4 earlier now it will be at 4 plus 4, 8 so difference is 4. F, f was read at 5 cycle 5, read at cycle 5 and written at cycle 7, 9l plus 7 when I am here, so 7 will simply get delayed by 4 so it will become 11 so difference earlier was only 2 now 2 plus 4, 6 a very simple arithmetic.

Then comes g, g was read by reader at cycle 6, 9l plus 6 as before and g was to be written here, so now 9l plus 4, 9l plus 5, 9l plus 6 so 6 and again 6, so difference now is 0. Earlier it was maximally negative minus 4 now it has becomes 0. So I got rid of the negative thing.

Similarly, the other ones h was written at cycle 7 and was to be was read at cycle 7 was to be written at cycle 5 here that will get added with 4. So it becomes 9, so 7 to 9, new difference is 2. At 8 i was written at, read at, i was read at or i is read at 8, earlier it was written at 8, now 8 plus 4, (14) 12. So difference earlier was 0 now 4. So by introducing a delay between the two, I have got rid of the negativity here.

Okay I always got some positive or non-zero, some non-zero positive but zero there is some non-zero number of cycles by which each variable used to be stored in the system alright.

(Refer Slide Time: 11:34)



So therefore, let us see the history a, I am re-writing from here a is born in 0 and goes up to born in 0 and look at this T prime out 0 to 4. B born in 1 goes up to 7, c born in 2 goes up to 10, d born in 3 goes up to 5, e born in sorry e born in 4 goes up to 8, f born in 5 goes up 11, g born in 6 goes up to 6 born and dead is the same cycle, h born in 7 goes up to 9 and i, 8 to 12 alright.

But this is only one period lth. In the lth input cycle, we read certain elements a, b, c, d up to i, of the incoming metrics and had to store them by delay registers in the system over these cycles. But in the very next input cycle that is L plus 1th I will have another matrix maybe it will have elements a prime, b prime, c prime up to I prime. So I have to store them also using the same delay register.

So there will be overlap, okay as you have seen in the previous examples, and now I have to find out minimum number of registers required. That is, I have to find out the maximum number of live variables in the system at any clock cycle. I will need that many registers. If I have that many registers then number of live variables at any cycle will be covered by them, will be, I mean I will be able to store them, okay, so that is done by lifetime chart.

You all know how to carry out lifetime chart. So will simply obtain that from this figure. So lifetime chart we have to find out the minimum number of registers required, there is a magic figure with that you can store all the live variables whether from the current cycle or coming from the previous cycle, previous input cycle or belonging to current one.

You can always store that is whether there is overlap or not, you can only store all live variables in any particular system cycle by those registers that comes because of out of this lifetime chart. So remember how you did the lifetime chart? There is a cycle plays a rigor 1, 2, 3, 4, 5, 6, 7, 8, 9 no point in going beyond because I told you how to calculate by going back.

We will do that again so you will see a long list is a, b, c, d, e, f, g, h, I, okay. And number of, this is number, number of live variables alright. Just for alignment let me draw a dotted line cycle 1, cycle 2, cycle 3, cycle 4, okay. Let us start with a, a is born in cycle 0, just a minute. A is born in cycle 0 and goes up to 4, so 0 to 4.

Then b, b is born in cycle 1 goes up to 7. C, c is born in 2 goes up to 10, so c born here goes up to 10. Oh that means I need more lines, because I need few more lines (sorry). Maximum here is 12 so I need 10 of, 10, 11, 12. You see this column 4, 7, 10 maximum is 12. So I have to go up to cycle 12, okay. So I am dealing with c, c from 2 to 10. So it goes down up to here. D, d is from 3 to 5. E, e from 4 to 8, f 5 to 11. G 6 to 6, so it is born and dead. H 7 to 9, 7 here goes to 9. I 8 to 12, okay.

Let us see here. People have just born, just a minute, a should start from here, so in this cycle I need 0 here, I need 0 here 1 because b is brought but a is carrying on, so is to be stored. Then here 1, 1, 2 this is just brought so 2. Here 1, 1, 1, 3 this one is just born. Here 1, 1 even in the cycle when it is consumed it has to be stored in a flip flop, it is taken from the output of a flip flop.

So storage is required in that cycle. There is a 1 plus 1 plus 1 plus 1, 4 not this because this is just born. So question of storing it does not come in this cycle that comes from the next cycle onwards, so just 4. Here 1, 2, 3, 4 this is just born, so same question again I will count it just 4, 1, 2, 3, 4, just born and dead so just 4. Here again 1, 2, 3, 4 this not to be counted this is just born so it is 3.

1, 2, 3, 4 this is 4. 1, 2, 3, 4, 1, 2, 3, 1 but period is 9, so 0 to 8 and then this. So as I told you here 4 you have to go back by sorry, 9 if 9 you have to go back by the period 9 and pick up the figure from here which is 0. That is, if I start actually say another one, another cycle, another there is a reader start reading another matrix now in the L plus 1th cycle input cycle, which begins from here.

Then there would been another a, right, like this. So that will get connected, counted, that will get counted, that figure comes from this 0. Likewise, we I mean we would have come from here and so on and so forth. So I am considering that second input cycle L plus 1th but I do not really have to do all that phrase again. I can go back and pick up that figure from here.

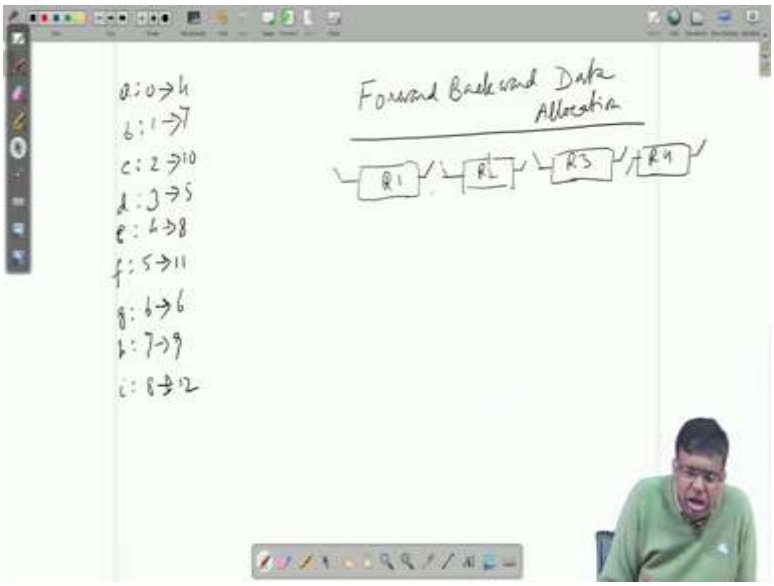
So 4 plus 0, 0 is corresponding to this actually for a prime, okay this maybe b prime. So 4 plus 0, all these were explained in the previous class then 3, you go back by 9. So it was 10, if go back by 9 you have 1. So 3 plus 1, this is 4, this is 4, 2 if you go back by 9 this 11, so you have to go to 2, line 2 and here figure also 2, so 2 plus 2,4 at 1 which is 12. If you back by 9 you go to line 3 and this is 3.

So 1 plus 3, 4, so 4, 4, 4, 4, 3, 4, 4, 4 so you see maximum value is 4 here that is any cycle, system cycle number of maximum number of variables that are live. Live means which are required to be stored which are not yet consumed that is 4. Therefore, if I am giving 4 delay registers okay, I can save them. Only thing is how to save them, how to or for which process, which technique to use that is later.

There is allocation that is what we are coming to later. You got theoretically, I know that with 4 registers in any cycle I can store for the next cycle, okay. At any second you give me some live variables and you give me 4 registers, I can happily store them for use in the next cycle and so on and so forth alright.

If that we then how to use this 4 registers now to optimally carry out this saving that is storage business okay not optimally may be efficiently carry out the storage business. A very popular technique is called forward-backward data allocation that I am coming to now. We have got the figure 4, so now I am erasing it.

(Refer Slide Time: 22:37)



Here in the last example we found out that minimum number of registers required is 4. In general it can be say l, small l or maybe say some other figure, maybe small m then what we do is this we employ m number of m. If the minimum number of registers that comes out of that lifetime charge is m, a must 4 in the previous example, but in general, suppose it is m then in this method we employ m number of registers in cascade.

You could call them R1 for register 1, R2 for register 2 and all that. Free of these is data will continuously move through them. It will not remain static in any of them. But then input port or import terminal of every register is switched that is it can come from the previous register or it can come from elsewhere maybe the output of some other register which is a successive bit, okay.

And output of every register also is switched. It can deliver to the next register down the line or it can give it to some other place. So both input and output will have switch switches, okay, they are called switched inputs, switch output registers and we have R1 for first one then R2 the first two, then R3 it will go up to Rn in that manner.

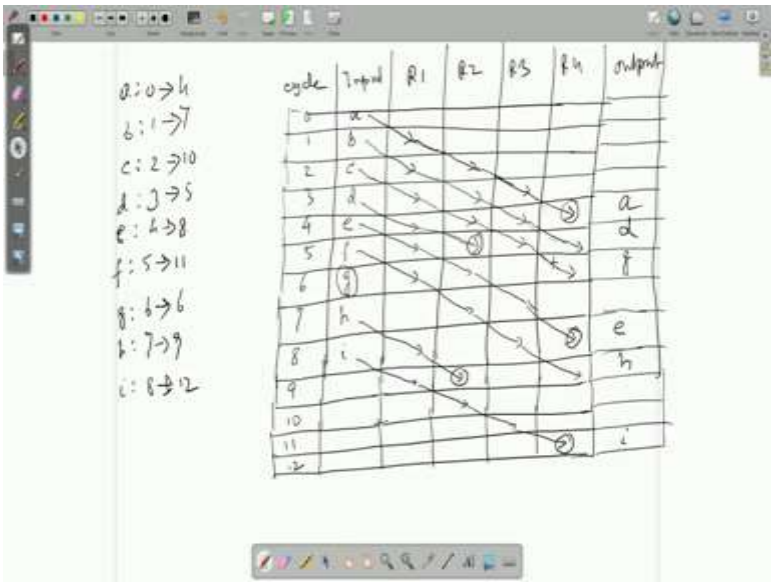
So it is a kind of cascade not necessary that cascade connection has to be followed like data from the output of any register may not have to pass to the very next one down the line, but can just use a switch and go elsewhere. Look another, further a register which is further down the line.

Okay, so let us take example let us explain all this through this previous example where m was equal to 4. So you will have 4 registers combination is like this, R1 this is the cascade, R2, R3, R4 we will arrange them like this.

This output is switched, it can move in various places, input is switched, here input is switched, output is switched like this. Particularly I would have had a cascade combination but depending on the lifetime chart, I may have to move output of R1 to R3 rather than to R2 or to R4 or R4 output we have to be feedback to R2 input like that, that is how this correction.

But will be a formed but remember never data will be stored permanently in a particular register that will continue to move cycle after cycle from one register to another register. Data will not be static. These are (())(26:10). To do this what we do is this we form a table, very nice table. The table looks like this.

(Refer Slide Time: 26:26)



This is cycle input, cycle bits I am considering that Lth input cycle which means system cycles are 9l plus 0, 9l plus 1dot, dot, dot up to 9l plus 8. I write R1 for register 1, R2 for register 2, R3 for register 3, R4 for register 4, and then output. Here also I will consider up to 12, so cycle 0, 1,

2, 3, 4, okay. We write it cycle 0, whoever is born. Here in every input, every system cycle, only one fellow is born.

So I will carry out this example is that but then there may be some cases where in a particular system cycle more than one fellow are born, then how to, what to do that I will consider little later after I complete this. So cycle 0, a is born. I write a, what we do we give I write a here means I give a to the input of R1.

If I write a here and then you see it goes from 0 to 4 right. It goes from 0 to 4. So you remember that lifetime chart, it started from 0 then it goes to cycle 1, cycle 2, cycle 3, cycle 4 then it gets consumed. So a I give to R1 its cycle 0 when it is born and it is stored in R1 for that cycle and comes to R1 output these R1 is stored is coming to R1 output at cycle 1. So in this box means the output of the register is available.

Then I give it to the input, I mean, I do not disturb anything because it has to be stored in cycle 4. So I give it to the input of R2, so the line continues. If it continues, it is given to the input of R2. It comes to the output of R2 at cycle 2. Then again it has to be stored in the next cycle so cycle 3, it continues, the line continues, it is stored in cycle 3 means R3 bits. It will be available at the end of the cycle here at the output of R3.

But then I have to go up to 4. So again, the line will continue. It will be given to the input of R4. R4 will store at the end of cycle 4, it will come out here at the end output of R4. So this is the beginning. A is given to the input of R1 because the line is hitting here, R1 stores it in cycle 1 and gives it at its output at cycle 1. Then again get R1 output is fed to R2 input and R2 stores it and gives it at its output at cycle 2 and then R3 takes as its input and gives it as its output at cycle 3.

Then R4 again takes it and gives it at its output at cycle 4 and then it is over. So we put a circle and we write here and this cycle I get an output a. Then comes b, b is born in cycle 1 so I give b here. So you see there is no conflict, a is given to the input of R1 in cycle 0 then a moves to the output of R1 in cycle 1, that time I give b at the input of R1 then it cycle 2, R2 stores a, R1 stores b.

Then b is coming up at the output of R1 and at the end of cycle 2 a is coming up and the output of R2. So the output of R1 here, output of R2 here, there is no conflict. So b continues in the same manner then goes to the input of R2 because b is a long journey 1 to 7. So b follows a, okay b is given the input of R2, R2 shows it as the output here say at the end of cycle 3. At the end of cycle 3, R3 output shows a. So again no conflict with R3 as I told you earlier, this output goes to R4 input.

R4 stores it in cycle 4 and cycle 4, R3 stores it. It goes to the output of R3. Then it goes to R4, R4 had stored a, in cycle 4. In cycle 5 it is free so it is just happily taking it from R3, storing it but I have to go beyond. Okay 5, cycle 5 I have already reached the end of the journey there is R4 output, but b has a longer path I have to go to 7. What to do? Do not worry, I will for the time being stop here but it is not the end of the journey.

I know one thing that this number of register which is 4 is the magic figure. There is a minimum number of register I worked out from the lifetime chart that gives me an assurance that, that 4 is enough in any cycle to store the live variable.

So these variables still live, but using this 4, I should be able to store them there is some register if it is available at R4 output. Some that is one of them, at least from R1, R, R3, R4 will still be available to store it. I simply have to detect it okay one or more than one. I would have to detect it or detect them and give it to the input of that.

It will be available that as soon as it comes from the figure that the number of registers which is 4 is a magic figure obtained from lifetime chart alright. So I stop here. I do not end that journey, I will come back and take it to some register input for further storage.

But quickly come to c, c is 2, from 2 to 10 so it goes like this. Again, there is no conflict, as I told you, c is given to the input of R1 in cycle 2 but R1 output only in cycle showing b in cycle 2 so no conflict. Then R1 stores c in cycle 1 in cycle 3 and gives you that is output. And that goes to R2 input, R2 stores it in cycle 4 when it is free.

The given to R3 input in cycle 4, R3 has began the output in cycle 4, so input is free so it is given it to the input. So in cycle 5 it is free it is storing further. But c has a very long journey 2 to 10, so

again it is giving to R4 input in cycle 5 that time R4 output was b, so no conflict. Then in the next cycle 5, R4 stores it then again I have to halt here. I will come back.

I will definitely find some register to store in the subsequent cycles because the number is the magic figure, which is enough for storing all the live variables in a particular cycle, okay. Then comes d, d is 3 to 5. Okay so d is born. It goes like this. This is free, R1 input it gave R1 output I have it at 4, then it comes here is giving to R2 input in cycle 4, cycle 5 R2 is free, R2 stores it gives you the output and this journey is from 3 to 5 so no need to proceed further.

I put it down here and this is my b, alright. Then comes e, cycle 4 to 8. So I can proceed in the same manner. Now I think you can understand 4 to 8. So (sorry) this is e, f 5 to 11 again I use the end of journey at cycle 9 but I will come back to this. Then G 6 to 6, so G I do not have to store, it is born and dead, so G here, okay.

After g I have got h 7 to 9, so h born in 7 it goes like this, this is h and i 8 to 12, 8 you will understand the meaning of this journey by arrows it goes up to here and here 12 I how to take care of this ones which got halted and have not been competed this I will do it the next half of the lecture. Thank you very much.