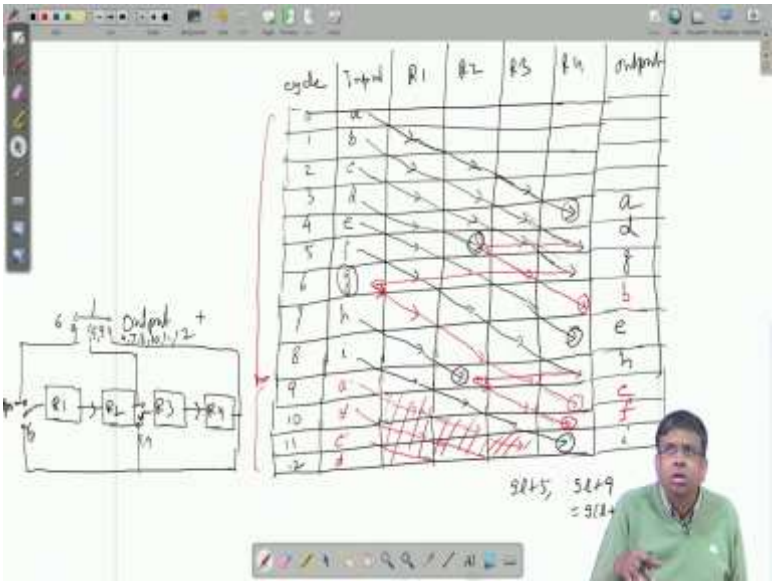
VLSI Signal Processing
Professor Mrityunjoy Chakraborty
Department of Electronics and
Electrical Communication Engineering
Indian Institute of Technology, Kharagpur
Lecture 26
Forward Backward Data Allocation

(Refer Slide Time: 00:22)



So I come back to this figure. So let us consider b, b got halted here, but you see in the very same cycle this fellow d is consumed, okay d is consumed, I do not need to propagate it further down the line. So R2 output was free or R3 input, R3 input is free, from R2 output I take it to the output port, I will not give it to the input of R3.

Because this is a word, right it will go to the output port by a switch, it will not come to the input of R3, which means input of R3 automatically gets free and that is happening because the number of register is chosen to be 4, which we obtained for the lifetime chart.

Which has showed us that 4 is a, 4 is enough to store all live variables in every cycle, which means if I am not reaching the end, somebody else will be consumed, will be over. So, a new fellow new register input will be free and that is R3 here, so R3 input becomes free. So I will give it to the R3 input here and it will continue like this.

So it comes to 6, 6 to 7. But, what is this fellow? This fellow is b, b is from 1 to 7 fine and it is the end. So this is b, similarly comes c, c has reached here, but I am sure somebody will be available, you know some register input will be available, where I can give and it can proceed down the line and look at this g.

G was supposed to go to the input of R1, but g is born and dead in the same cycle. So, g is born the moment it is read, I will send it to the output port by a switch, I will not take it to the input of R1 which means input R1 is free in this chain of registers, so it will be taken like this and given the input of R1. So that it will continue in the same manner.

C is goes up to 10, so, it will continue you see all these boxes are free for it. Okay, that is a again happening because number of register is enough in every cycle, I do not have any storage problem. Okay, so you stored by R1 then up to available then R3, somebody or other will be available, down the line.

The way we carry out it is, you know, following a sequential path from R1 to R2 to R3 to R4, whenever I mean it is requiring that you know that many cycle. So, it goes from R1 to R2 to R3 to R4 and the boxes are available, okay I just give it to the appropriate input and from that onwards like it came to the input of R1 and from that onwards in every cycle some boxes are free, the boxes are free down the line.

Obviously, it goes from R1 goes to R2, R2 to R3 and so on and so forth, okay, because that is a general direction of movement of data from R1 to R2, R2 to R3, R3 to R4 like that, okay it reaches 10 and c is from 2 to 10, so I have found the end, okay. Then one more fellow is left that is f, f is from 5 to 11.

Now, here you see this fellow is over, which means at cycle 9 R2 output goes to the output port by a switch, it does not go to the R3 input. So from R4, I bring it to the input of R3 and move it forward again you see these boxes are open, f is from 5 to 11, so it will go from here to 10, 10 to 11 and done, so f, alright.

But one thing you see from 0 to 8, this was say Lth cycle. Next cycle start from here, so it will take I require mu, I will be reading, system will be reading a mu a that is maybe a prime mu b, b prime mu c, c prime, mu d. But, will they give rise to the overlap? Answer is no, because you can easily see a, a prime will be born here. It will go like this. I am just following this route dot, dot, dot, dot, b prime is born here it will go like this, c prime born here, it will go like this and so on and so forth.

So these fellows, I am just passing them actually they are not overlapping with this I mean they are I mean they are not coming in conflict, okay, that is 4 registers are good enough to store variables of this Lth cycle and live variables from L plus 1th cycle. Okay in any system

cycle, like system cycle number 9, I have got h coming here, I have got g coming here, I have got f, but I have got also a prime, 4, 4 are a required.

Similarly in cycle 10, I have got a prime coming here, b prime and 2 fellows from the previous one, this guy, which is i, this guy which is c, okay, so 2 fellows from the previous one, previous input and 2 fellows from the current input but total is 4.

So you see 4 is that magic figure, since I have chosen that I am never in short supply of any registers even knowing I am considering the overlapping zone, where if any horizontal lines, I mean this you know, boxes, horizontal row or any row I have got some boxes filled up by previous variables like a to i, part of them some of them and some boxes filled up by the current variables that is from a prime to i prime okay.

But there is always no conflict, some boxes are available for the first category some boxes are available for the second category, whereas 4 is good enough. We obtained it from the lifetime chart, okay. If this be, then how do have a circuit for it? Okay, let me see I mean need some space.

I have R1, R2, R3, R4 an input is coming here input. You can see one thing and output is here, you can see one thing input always except for this line number 6 input is always going to R1 input of R1, only in cycle 6, the input data directly goes to the output, right. So from input one line goes to the output and output has a switch, output would touch it at 9l plus 6.

I am just writing 6 okay but actually 9l plus 6, so output port here the switch it will touch this at 6. So at 9l plus 6 whatever be the input which is g that will go to the output. R1, at this time at cycle 6, input of R1 will come from this here that is R4 output, look at the red line. So R4 output will be fed back to R1 input at cycle 6.

So it will be switching, it will touch it at 6 and touch it at the other cycles. You know, I am not writing them if I write 6, and I do not write anything or maybe for your sake I write 0, 1, 2 I mean there is no space so, so I am orally telling you at cycle 6 R1 touches, R1 input touches this line at all other cycles that is 0, 1, 2, 3, 4, 5 and then 7, 8, it touches this one, this is the meaning of it.

Okay, may be the switch I re-draw switches here. It will toggle between either this or this It will touch it at 6, 9l plus 6 in all other cycles it will touch here, when it touches at 9l plus 6

input takes this path it is born and dead it goes to the output, but in all other cycle it touches input line, so input goes to R1 input.

R1 input you can see is always going to R2 input look at these, these, these, these, these, these, these, these here, here, here okay, that means R1 to R2 there is a permanent correction whenever whatever be there in R1 output that will always go to R2 input,

R2 output, now look at this R2 column, R2 output, here it is going to its cycle 5 fellow d, this is consumed means from R2 output, it will be sent back to output port and also here, okay, from R2 output in cycle 9, it will be send back to output right. So, R2 output this will go here at cycles 5 and 9.

Now, I tell you one thing, actually it is 9l plus 5 and 9l plus 9. Now I prefer this notation, but the notation followed by KK Parhi, in his book and also in papers is these whenever it is 9l plus 9, it is becoming say 9 into l plus 1. So as though it is 9 into l plus 1 plus 0 that in the 0^{th} cycle, 0^{th} system cycle for the next input cycle. That is basically 9, you take modulo 9 that is 0.

So instead of 9 l plus 9, he writes is 0, he writes is 0 and 5, I prefer 9l plus 9, but since it is 9, 9 modulo 9 is 0, if it is 10, 10 modulo 9 is 1, if it is 10 it would become 1. So, for l plus 1th cycle, input cycle it will be the 1, system cycle 1. But input cycle goes to l plus 1 that is 10. If it is 10, 10 modulo 9, that is 1, 1 would have come here, okay, and I would have written 1 here, instead of 0, it is a modulo notation.

So if I follow up on the modulo notation, it is actually 9l plus 9, but which you can also write this way 9 means 0 to 8. Those are the system cycles for 1 input clock 9 minutes, I am entering the 0^{th} system cycle of the next input clock l plus 1th it that is why we write 0 here. That is the style followed by ())(12:58).

5 remains 5 because 5 modulo 9 is 5, the moment we exceed 8, it become 9, 9 is effectively 0 for the next input cycle that is 9 modulo 9 is 0 or it is 10, 10 modulo 9 is 1, 1 is 11, 11 modulo 9 is 2, like that. So for this diagram, I am following this notation. Okay, or I mean, I am doing it so that you are not in any confusion when you read KK Parhis's book.

But if you are clear about what I said, just now then I can, if you permit, I can stick to my notation where I do not replace 9 by 0. I just write 9 here. Okay, maybe I do that because I have already told you the meaning. So that if you read the book, you will not find 9 here you

will find 0, but you will understand what it means, so when you see this lecture I am writing it as 5 and 9, in KK Parhi's book 9 will be 0, okay.

Well, because he is following a modulo notations, so let me stick to my notation 5 comma 9, alright. So that is from R2 output, and when R2 sets them, that time who comes to R3 input, R4 output goes to R3 input. Again here R4 output goes to R3 input, what are the cycles 5 and 9. So R4 output comes here and 5 and 9.

This guy touches this otherwise it touches this, alright, it touches their 5 and 9, otherwise it touches this in all other cycles and R3, R4 you can see there is a direct path, whenever there is R3 output, R3 output that has always been given to R4 input the line continues, continues in all of them, it continues like this.

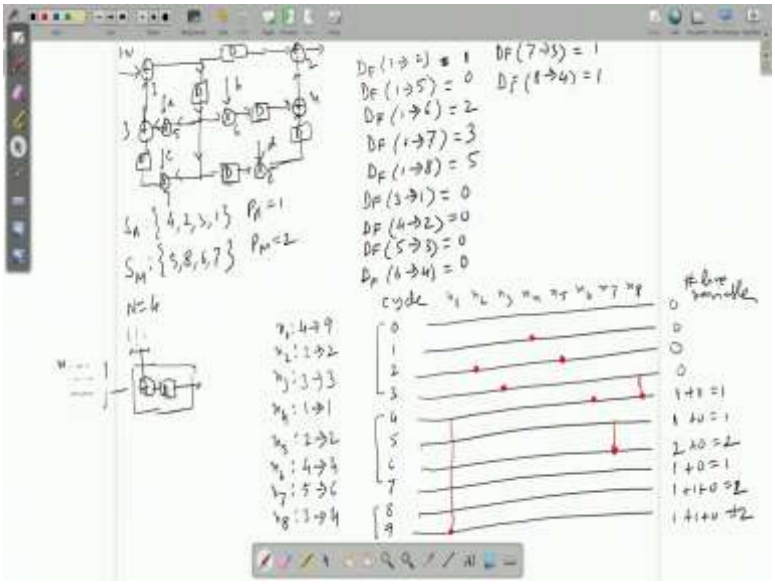
Okay, that means R3 to R4, there is a direct output and R4 output here, here, here, here, here, here they go to output port. So, in which cycles 4, 5, just a minute, so here 4, then here 7, 4, 7 4, 7, then 8, up to 8 is fine. We are here, 10, 10 in Parhi's book will be 10 modulo 9, which is 1, so he will write 1, but I am writing 10. Then 11, 11 in Parhi's book or in some other literature will be 11 modulo 9 of these 2.

So there is write 2, I am still writing 11 here, that is really 12 in their book in his book will be 12 modulo 9 it is 3, but I am writing 12, all right, so this is a circuit. So this gives you a fairly good idea about how to carry out lifetime chart after doing lifetime analysis and what to do the lifetime chart you know, the minimum number of registers required, then you follow these forward backward data allocation table.

This method of moving the data and still storing them in the system and develop a circuit for it, this is the whole journey for minimizing the number of delay registers, okay, because I am always hitting on minimum number of delay registers required, there is why delay is optimized.

Now I consider a more relevant example that is the digital filter we started with the very beginning of this folding section, folding you know we started with digital filter. I go back to that example. Remember quickly I draw that figure again.

(Refer Slide Time: 17:24)



Adder 1, adder 2, adder 3, the same will stop 4, multiplier 5, multiplier 6, multiplier 7, multiplier 8, okay. The schedules were this, that is the folding sets SA was 4, 2, 3 1, SB sorry SM multiplier that was 5, 8, 6, 7 pipeline latches, PA was 1, PA was 2, okay, and period was 4 that as you read the input cycle system is working at 4 times faster than the input because folded by 4.

Then after 4 system cycles, another input comes at new set of variables will get we will have to be stored also. So there will be overlap between 2 or more than 2 cycles and we still have to carry out storage of all the live variables in this filter.

Now remember, we obtained those folded delay figures, I mean, the delay figures after folding, DF I am just writing from there copying from there, DF 1 to 2 we had 1, 1 to 5 we had 0, 1 to 6 we had 2, then DF 3 to 1 that was 0, DF 4 to 2, that was to 0, DF 5 to 3 also 0, DF 6 to 4, 0, DF 1. Okay, this were the thing.

Now let us convert first lifetime analysis identify the variables. Adder 1, what is this 4 adder? 4 adders are nothing but I mean they are replaced by 1 hardware adder unit of this type. That is there is an adder actual adder, it will pipeline by 1 latch that I am showing separately by 1 delay the output, this side is switched, this side is switched.

There are lines, input lines, there are lines a particular cycle, say let us consider adder 1, adder 1 has cycle I mean schedule 3, 0, 1, 2, 3 so at schedule 3, it will touch if it is adder 1, I mean if it is input okay at schedule 3, let me erase this at schedule 3 it will touch that particular line, what line, in at schedule 3, because IN is entering adder 1 at schedule 3, this

fellow will work for adder 1 it will touch in and it will touch and some other appropriate line here.

Then it will carry out a submission and it will get delayed by 1 cycle. So, adder output the output is born but internally also delayed by 1 cycle, so it was born in what schedule at cycle at cycle 3 because you know 0, 1, 2, 3, 3 is the schedule of adder 1, cycle 3 this fellow is touching the appropriate inputs like in from here, carrying out the job of the adder 1, okay.

But there is an internal delay there a pipelining so at schedule 4, there is clock 4 the adder output is coming in the system that means the adder output, adder 1 output is born inside not cycle 3 but cycle 4 because of 1 more delay inside. So n1, I call it n1 for the n for node, n1 there is a node output 1, it is born in cycle 4.

And then you see 1 is required to go up to 2, I need 1 delay, 1 to 5, 0 delay same 1 goes to 6 needs 2 delay, 1 to 7, 3, 1 to 8, 5 that means node 1 output used to be delayed maximally by 5 which will take care up 2, 5, 6, 7 all, okay, so I need to store n1 in the system for 5 cycles, so if it is born in 4 it should go up to 9, 4 to 9, okay.

So that 1 to 2 done it requires only 1 cycle of storage 1 to 5 done no cycle of storage required 1 to 6, 2 cycle of storage is required, 1 to 7, 3 required 1 to 8, 5 so I will take because the variable is common n1, 1 is common for all of them. So I take the maximum that is fine, so is born in cycle 4, so delayed by 5. So 4 plus 5 is 9, from 4 to 9, it has to be 4 is born and 4 to 9 is the journey in the system.

It has to be stored, it has to remain stored, it has to remain alive in the system from 4 to 9, okay. Then comes n2, n2 there is no problem, n2 is a node output, okay n2 adder 2 schedule is 1 that time is carrying out the job of addition 1 more delay, so at cycle 2 the output is formed and it is not required to stored.

So it is 2 to 2, okay, remember schedule is 1 where there is node 2, add one more delay inside, so n2 output is formed at cycle 2 and consume that time only because it is not entering the filter again so it is consumed that time, no need to store n3, n3 schedule is 2 I had add one more delay.

So born in cycle 3 and look at here, 3 has only 1 line, 3 to 1, 0, so born and consumed in the same cycle, n4, n4 is adder output 4, 4 schedule is 0 and 1 delay inside, so it will born in 1,

but it will go up to what? 4 to 2, 0 delays born in cycle 1 and consumed also cycle 1 sorry, okay.

Then n5 multipliers, n5, 5 schedule is 0, but there are 2 delays inside, so born in 2, again 5 to 3, 0 delay. So 2 to 2, n6, 6 schedule is 2 and 2 more delay inside so born in 4 and 6 to 4 0, there is only thing 6 to 4, 6 goes only to 4. So 4 to 4 because 0 delay, no need to store, n7, n7, 7 here schedule is 3 and 2 more inside, 2 more delays inside so 3 plus 2, 5.

In cycle 5 only it comes up okay, and 7 to 3, 1, so 5 to 6, cycle 5 born delayed by 1 cycle to be consumed at node 3, certainly 5 to 6 and n8, 8 schedule is 1, 2 more delays inside, so multiplier output comes out at cycle 1 plus 2, 3 and 8 to 4, 1, so 3 it is born 1 cycle of delay, so 3 to 4 okay.

Using these I can carry out the lifetime chart, sorry. So I go up to 9, because in this column highest figure is 9, okay 9, so and I write here n1, n2, n3, n4, n5, n6, n7, n8, n9 and number of live variables, alright.

Okay, start with n1, n1 born in 4 goes up to 9. So n1 goes born in 4, goes up to 9, then n2, 2 n2, n2 born in 2 consumed that time, no need to store n3, 3 to 3, n4 1 to 1, n5 2 to 2, n6 4 to 4, n7 5 to 6, n7 5 to 6, n8, 3 to 4 there is no n9 I wrote wrongly, there is no n9, 3 to 4 alright. And what is the period? Period is 4, so this is one period 0 to 3, 4 to 7, 8 to 9.

First let us what is here, 1 cycle only, so here 0, here 0, here 0, here 0 because they are just born, here 1, but this is to be stored. And this is born, this is born and dead, so they do not count, 1 is required here, none is required here. 1, 1 is required 1, 2, 1 required here, 1, 1 required here 1, 1.

Now what I do is from this 1, you go back by 4, so 1 plus 0, 1 from 1 you go back by 0 by 4 so 1. From this 2, you go back by 4. So, okay there is 0 again, and lastly from this 1, you go back by 4, so again 0, 1.

Now this 1, you go first go back by 4, here, 1 and again go back by 4 0, so 1 plus 1 plus 0, because here 2 overlaps would come, something which is getting started here. Something which is started here and one more, okay. And now again 1 here, so you go back by 4, there is 1, 2, 3, 4 this 1 and again another 4, 1, 2, 3, 4.

So 1 plus 1 plus 0, so this is sorry, this is 2, this is 2, alright. So you get the figure 2 where there is a maximum. So, the maximum number of live variables is 2, which means I need 2

registers R1 and R2 to carry out the storage, these are lifetime analysis and then lifetime chart then we will use 2 registers R1, R2 and follow that forward backward data allocation methodology. Okay, and carry out the storage and then develop a circuit which will be done in the next class. Thank you very much.