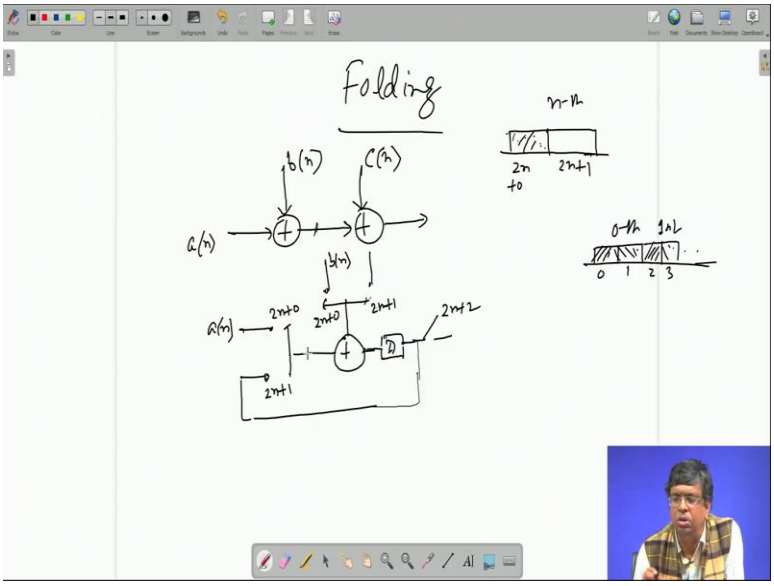
VLSI Signal Processing
Professor Mrityunjoy Chakraborty
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology Kharagpur
Folding of DFG
Lecture 21

(Refer Slide Time: 0:34)



After covering unfolding, now we move to the other technique which is just opposite of unfolding, it is called folding. These names are very interesting. Here in unfolding what we did, we paid in hardware, we parallelized, but system was slow but still we could process input and output at a faster rate. So, input output rate could go up, because we try to retain the circuit fast we gave the fast we could go up to some, some level beyond that may be retaining was not helping then we try to parallelized it. And by that process we showed that if even though system is working at a slow clock input output rate can be faster, that was unfolding.

Here it is opposite we try to gain in hardware at losing speed as an outcome that is instead of using so many hardware units, when you say adder, so, many adders in the circuit why not use one adder or two adders and do the job of all the adders in the circuit by using the minimum number of adder. Similarly, multiplier instead of using so many multipliers in different places in the circuit, for that use only one or two, few multipliers and do the job of all the multipliers in the circuit by them in a time division multiplexed manner.

That is important here as you will see the circuit has to be operate at a faster clock because using say one multiply to do the job of 10. So, using a faster clock in one clock, you will do the job of multiplier one in another clock, you will do the job multiplier 2 like that. That is why system is working will be working at a faster clock, so you lose in speed but getting the hardware and therefore power. To explain, for example, let us consider this that suppose already we have fix like this at nth what clock one data is coming an maybe 16 bit or 8 bit whatever it is getting added with bn in nth clock, result is added with another data cn is your result.

Now, here you see we are using 2 adders. Suppose I want use only one adder, only one adder, then what we will be doing, if suppose a clock period, the nth clock with the nth clock period I will divide it into 2 because there are 2 adders so I will divide it into 2 periods, they are called system clock period system will be faster. So, this is zeroth this is first, so actually this will be 2 nth, this will be, this 2n plus 0, this is 2n plus 1. This I think must be obvious to you, because if you start the timing diagram this way, say zeroth or real what clock this we are dividing into 2 system clocks 0 1 then we have got first, dividing into 2, 2 3 dot dot dot, then 4 5.

So, nth so, first is, zeroth is, 0 and 1 out of first system input clock, we are having 2 system clocks, system clock 2, system clock 3, then if it is second input clock from which we are deriving 2 systems clocks, one is 4 and another 5. So even odd even odd so in the ways of nth 2n plus 0, 2n plus 1. Now what we will do is these will use only one adders. But both the inputs will be switched, this will switch to an, an is coming it is coming on the whole nth clock it is appearing here, but this switch is operating at a clock period twice at a clock period half up the input clock, like this if this total was the input clock period, now it is working system is working at half the period or twice the speed.

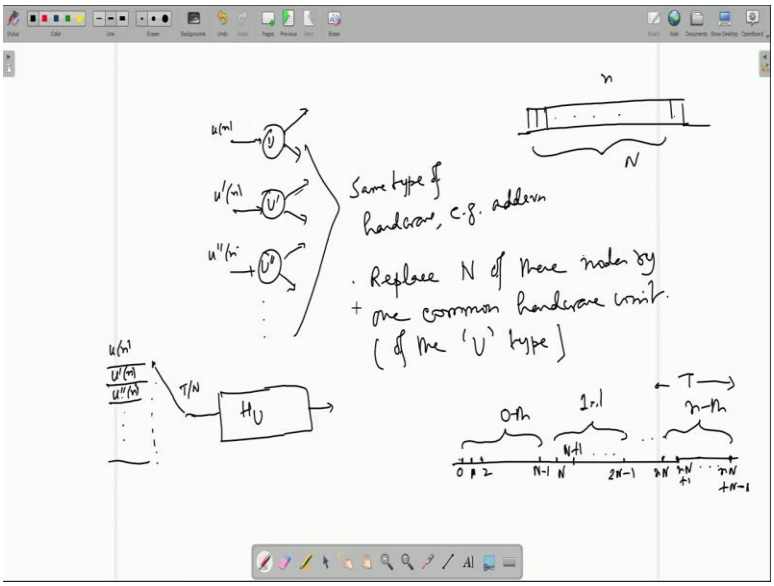
So, I will be touching this at 2 n plus 0 though n is available all throughout the nth input cycle from here to here. I will be touching it during these half., and during that time I will be touching bn, so bn also I touch at 2n plus 0. So during this half 2n plus 0 I am doing addition of an and bn. So I am taking if t is the original clock duration now it is t by 2, t by 2 so, clock period is half, speed is twice or your speed is 1 by t now this 2 into 1 by t. So, in one of the system cycles of shorter period, period t by 2 I add an and bn then the result comes this result has to be added with Cn, Cn and then this result what I do I move it here to other end of the switch, the switch comes back, sorry, the switch comes back here at 2n plus 1.

That is what I moved to the new next system cycle within the same nth Watt input Watt cycle 2n plus 1 that time it comes. So what I have to do I have to hold this result in a delay. So in 2n plus 0, in 2n plus 0, I am adding an and bn, result is coming out or storing here, it is not a single flip flop, it is an array of flip flops with a 16 bit line is the bass of 16 bit there will be 16 flip flops in parallels or resistors say. So in 2n plus 1th is cycle this data comes up here that means this result, this output is available here at 2n plus 1, switch moves here at 2n plus 1, why 2n plus 1 because at 2n plus 0 I added then there is a delay. So this result comes up at 2 n plus 1 switch this switch moves here at 2 n plus 1, this moves here at 2n plus 1.

Hold on, so in 2n plus 1th cycle, you (()) (7:16) get result and Cn they get added and the result comes out here. So this result you can take at next cycle 2n plus 2. So see I used only one adder so adder publicity has gone down. Of course I am using switches which are box but here only two adders were replaced by one adder I could have had 100 adders still I could have used only one adder to replace all of the clock would have gone up by 100 times. But number of adders would have gone down by 100. And I would have used a single set time division multiplexed manner

So, this is an idea of folding one example of folding I am folding 2 adder into 1 adder. So, folding factor is 2, 2 hardware units brought down to 1 hardware unit and there by the clock speed also went up by 2, in one of the clock period, which is the system clock period which is 2n plus, its 0 system clock period, I do one job, I do one job of the adder or I do the job of one of the adder, in the other system clock, I do the job of the other adder. So, this is basically doing the job of both the adder in a time division multiplexed manner first at2n plus 0 then at to 2n plus 1.

(Refer Slide Time: 8:45)



Let us try to generalize this to a DFG. Suppose in a DFG there are various nodes, you know U, maybe U prime, maybe U double prime, dot dot dot. They are all of the same type. Same type, same type of hardware, for example, they all could be adders, this has his separate input line, this has his separate input line, this has separate input and like that, this can go to various other nodes, this can go to various other nodes are and that, all right. I want to replace n, n is an integer, n of these nodes by one common hardware unit of the we can say U type, U means U category, U, U prime, U double prime, they are all coming under the same type of hardware if it is a adder then u category will be adder, if it is a multiplier u category will be multiplier.

They are all named like in terms of U. U, U prime, U double prime, U triple prime, dot dot dot, just various nodes, but they employ the same hardware. I am just identifying them, out of them I am taking capital N of them. And replacing by 1, 1 means what one hardware unit of the U type, see original U was a adder. So capital N adders, all of them are adder capital N of them I replaced by one adder. So, basically I will have one hardware unit, each of the U type and what it will do, there are data lines, one maybe for, if you can call it data line, if you call it say data coming here, you can give it a name small un. This is u prime n this u double prime n like that. So here also same un, u prime n, next will be u double prime n, the same data lines.

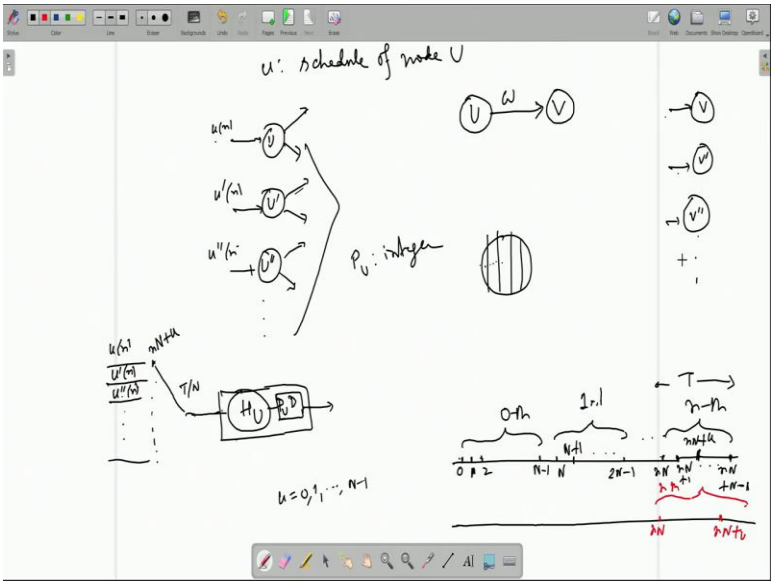
So they are coming at a original input clock, that is original input clock maybe like this nth. But what I am doing I am using a switch like a commutator it will move. It is not necessary that its

ordering will be like this that un first, then u prime, u double prime, ordering will be up to you. That is called scheduling, who first, who next, who next to next, like that. I am just showing an example, that if you start at un, going to u prime, u double prime but then ordering can be changed. So what will happen the original clock will be not divided into small sub clocks.

It is a system clock. So this will be how many total N, capital N if you see the timing diagram timing diagram wise zeroth input clock that will have capital N number of system clocks faster, so I can index them the zeroth, first just there will be time period you know clock period around that 0, around 1, I am just indexing them zeroth, 1 you know, 2 dot dot dot, n minus 1, then first it will be from n, n plus 1 dot dot dot 2n minus 1. So, in general nth small nth, small nth original clock then is input clock, that will start at what n into N, n into N plus 1 dot dot dot, n into N plus N minus 1.

So in one of these clocks, the switch will touch one of the input lines in next clock it will touch another of the input lines and like that. So, data here we will enter at a faster rate. So, in original clock period was capital T it will be now T by n at a faster rate, it will give the output. So, in one of the clocks it will take up the job of say u. So, that time the output will be giving going along these lines. So, this should go to those directions. Next time if it takes some u prime N next clock, that time this output line should walk like these two lines, these lines it will give the data to those doors. So, those things have to be incorporated.

(Refer Slide Time: 14:13)



So, now with this what I do, suppose in the DFG out of all the various u type of doors, u, u prime, u double prime dot dot dot, I pick a particular u, a particular u. And suppose there is an H from u to another V, V also has, it is, V can be an adder again, that can be multiplier, that can be divider, that can be some. So, V also has got various nodes you know, like V,V prime, V double prime, dot dot dot dot, they may not be same in number, but V also has various copies of the same hardware unit with multiplier, one multiplier, another multiplier, another multiplier like that. They are all taking input from different lines, different nodes.

Now, when it comes to U particular U and particular V from here, I find that in the DFG, there is an H between them with some delay W, this delay is with respect to the original clock, original clock is slower, system is faster in one original clock period there are capital N number of system clocks, one that we have seen already. In the original clock period, I have got capital N number of system clocks. So, W number of input clocks means W into capital N that many system clocks. Now, what happens after how to unfold this, how to, sorry, how to fold this or in a folding situation where I am taking U, U prime, U double prime dot dot dot all of them, capital N all of them replacing them by one hardware unit of the U type and working out the time division multiplexing like this.

So in particular clock this will take up the job of U then it will do the processing, then outputs would, that outputs would go to V, not V prime, not V double prime after some amount of delay. How about delay? Originally W input cycle now what that is what we will find out. So suppose in the nth cycle, in the nth cycle at N capital N maybe plus small u that time, that this time at that clock HU accepts this data un that is at, that means small u can be either 0, one of the points or 1 or dot dot dot up to this. So if the figure it is general, in the general case put here so at nn plus U that time the switch touches this, means that time this hardware unit will take up the job of U because then it is taking, because the input un is touch then by this hardware unit.

So it will then, that means it will take up the job of U, not of U prime, U double prime at that system clock. By the way, since this hardware unit now is working at a faster clock. It is often necessary to minimize the critical path of the device here, maybe adder multiplier whatever, by doing some pipelining, pipelining means I will forward path pipelining say I will have some pipelining latches. Like I will do cuts and retiming, forward path I will introduce delays that is in general. So that critical part reduces here this device can walk at a faster rate. So, if that be, that

there will be delays to, you know I mean, just a minute. This is suppose the hardware unit say adder or some multiplier.

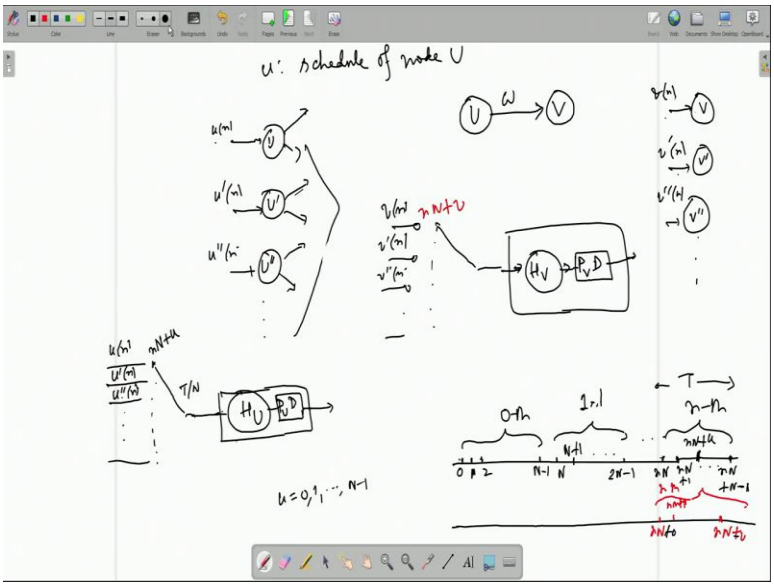
I will, I should follow, do forward path cuts and retiming and introduce delays between them so, that I mean critical path reduces because why it is necessary, because here system is working at a faster rate. So, the original critical part of the hardware unit may not be short enough for processing of data at that rate. So, it may be necessary to do some pipelining by forward path cuts and retiming. So there will be some delays coming in which means input will be delayed, I mean output here will be delayed version of, original output because I will be introducing delays in these hardware.

What for, what for, the delays will be doing forward path retiming or forward path pipelining. That is this stage will be divided into like as I told you, if this is U it will be divided into various stages and between every two stage there will be a delay, between next two stage there will be a delay like that pipelining. So, data coming here will not appear here in the same cycle it will go through various delays. So, it will appear after some, some number of cycles. So, just for counting purpose I showed that number of delays separately.

Here P for pipelining, U for U category, PU is an integer, maybe 2, maybe 3, maybe 4, that times D, if it is four 4 D that is four delays will be used to pipelining this, which means even if an input comes corresponding output will not be the original output, but it will get delayed here by 4 cycles then only it will be coming here. Why we need it as I told you, system is working at a faster clock. So, we may have often need to pipeline it so that it can process data at that rate that is why some delays for forward path pipelining will come up, those delays we are collecting separately for counting purpose only.

PU is an integer that many number of delay is used to forward pipeline it, these delays we are showing separately. So, this is a situation so, the small u, N capital N plus small u a particular system clock within the nth input clock. It is divided by n into capital N plus small u if u is 0 we are here, if u is 1 we are here, in general u is here, u is called, the small u is called schedule. Of whom, that node u, whose input is un and it is touching there at n N plus u. Similarly, in another timing diagram under the same, same nth, sorry, we will have n N and dot dot dot, maybe that n N plus some small v here. One hardware unit I will be using to take capital N of these V category and replace them by that.

(Refer Slide Time: 22:23)



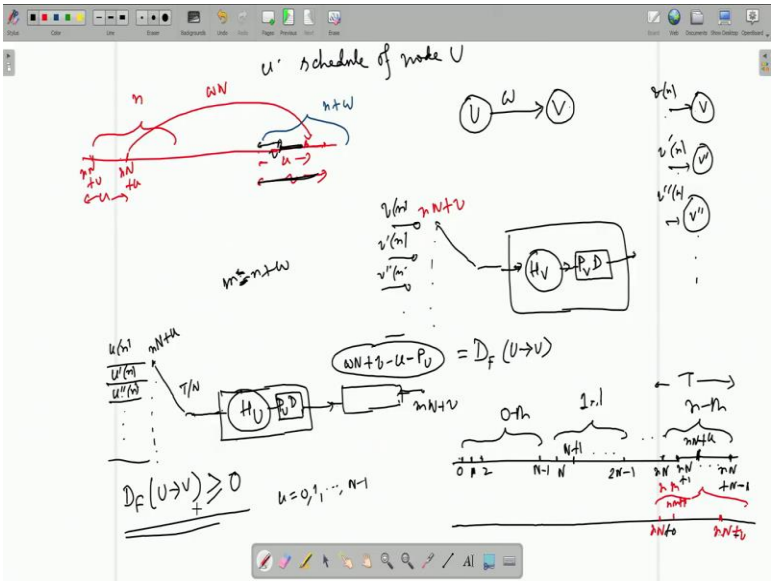
That is, let me explain, this is for U, similarly for V also, say hardware unit is of the type HV maybe this is an adder, this is a multiplier, maybe both can be adder you know maybe there are 10 adders, 5 I am taking here, another 5 here all that possible. So, but then this is HV and again this is pipeline by PV, number of delays. I am showing one input here, there can be multiple inputs, taking data from multiple directions. The HV is the thing and again if this is small vn and this is v prime n, this is v double prime n, here also may be a say vn, v prime n.

So, capital N number of such nodes I replace by one hardware unit of the V category, if it is a multiplier, this is a multiplier with a delay coming due to pipelining. And it will touch this line once, it will touch this line once and like that. I am beginning the schedule of node v to be small v that is n N plus 0, n N plus 1, but at n N plus v, small v that time I am saying this will touch this line that this input will be v n. So, this will work for this v, we have scheduled dx, small v could have been 0s, could have been 1, could have been 2, I am writing in general case.

So, we have done this for vu we have done this for v. Why u and v are important because there is a particular H coming from node U specifically node U, not from U prime, not from U double prime specifically from U. And specifically to V, not to V prime, not to V double prime but specifically to V, but in my folded setup, I do not have separate U, I have only one hardware unit which does the job of many of the U type of processor, one of them is capital U of course and it is doing that job at n into capital N plus small u somewhere here. Similarly, I do not have I mean,

if it is there is a node v here I do not have separate node for V here but out of I take capital N of the V type of nodes replaced by one unit. It is touching various input lines at n N plus v it is doing the job of that.

(Refer Slide Time: 25:55)



Then how much delay I should have originally I had w number of input delays, input cycle whether that is what is given with respect to input cycle. Folding is my creation, but this node, this DFG is given in terms of original input, I have got w number of delays. That means in the timing diagram, I can erase this timing diagram now or maybe I draw here separately. From nth this is n N plus 0, I am interested that n N plus u I am starting at u first. So at that system clock the data enters sorry data enters goes through this process, after being processed by the u category it has to be delayed by w number of input cycle.

Now one input cycle has got capital N number of system cycles. So w number of input cycle means w into capital N, that many, w into capital N that many system cycles. So, it will come under some other node, other umbrella if it is small nth, it is small n plus with. So gap between them is w number of input cycle which means w into capital N that be number of system cycle. So, so you are starting at small uth, if you go by capital N you enter the next but it is the same small uth point because the speed (()) (27:24) capital N when you jump to the right by capital N, if you start at small u here also go to small u.

If you are jumping by wn, so here again you are coming at this much is wu sorry u, but, so this that this is coming, this is coming after this w number of input cycle, this is coming here., but schedule of v that is, it is not taking up the job of v category at this specific point it is doing maybe further to the right. This must may be v that means if an if after getting processed here and then getting delayed by wn it has come here you understand it will be again if this is small u if this is small u, this also small u because you are jumping by an integral multiple of capital N to jump by capital N, 2N, 3N whatever you will hit the uth point only, so u.

But then that is not the point where in general, this hardware unit is taking up the job of v because I am interested specifically in v. Maybe it is schedule n into capital N plus v, there is v is further to the right. That means I have to have further amount of delay of v minus u, if v is greater than u. That means delay this output should have a delay how much wn plus v minus u the out of this delay already a PU amount of delay has gone in. So, net delay required will be minus PU. Then here if you tap, that is at tap at that point any small n into capital N plus v if it tap it here you will get what you are getting here then you can happily give it to this.

That means, if this u has, u category, u node, u node has got schedule small u, v node has schedule small v and then chosen by u fixed then after this processing by u, you should have a delay of this much amount. So that when you touch this at the schedule small v that is at n into capital N plus v, it may not be n N, small n, it may be some other unit just a minute which is coming delayed is coming after some delay. So, if you touch it at some other point maybe some m, for some integer n, m into N plus v, m is actually very simple a small n plus w.

Because u plus nth, so it become n into capital N this is n plus wth because after this it has to delayed by w cycle, w input cycle so n plus w. So, m is n plus w, that is what I wrote n plus w that times n plus v as I tap, I will get back what I was getting here. So, this is the delay that is required and this we denote by this P U to V or maybe there is another notation. Let me switch over to, this is that old notation better will be D, this is what (()) (31:58) D for delay, f for folding after folding with respect to the faster system clock delay between u and v will be this much.

Because after folding you do not have separate u nodes, separate v node there is an hardware unit, it is sometimes working for u node u there is a hardware unit which sometimes working for node v. So, I find out why it works for node u and reason is schedule. So n into capital N plus small u there is a time data comes from small un which means this hardware unit is taking up the

job of u node only. So it does the processing if all are adder, this an adder but goes through some delay because of pipelining. After that I have to hold it, I have to pass it through w number of input delay which means w into capital N number of system cycle delay.

w number of input delay means from nth, it will go to n plus wth which is, which is mth. So that means in terms of input cycle is W into capital N that many times system cycle. So from uth point here again under another umbrella I go to the uth point, but schedule of capital V may not be u, maybe V. If V is larger than u then another additional time V minus u times I have to wait further, so additional (()) (33:25) require, out of which PU who goes because PU already has been implemented, because small v, small v could have been here this much, I mean, instead of this it could have been less.

So, v minus u is negative which means, I do not need so much of delay I did less than that, how much less v minus u. So, from wn something will be subtracted, so that I am here, what here is v minus u or you can write v minus u equal to minus of u minus v. So, u minus v, this much will be subtracted from wn so you are here. Out of which PU will go. You only will need to make sure that this is not negative, because there are minus signs. So, you have to make sure that your schedules are design such that it is therefore negative, we built on this in the next class. Thank you.