

Digital Circuits and Systems
Prof. Shankar Balachandran
Department of Electrical Engineering
Indian Institute of Technology, Bombay
And
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

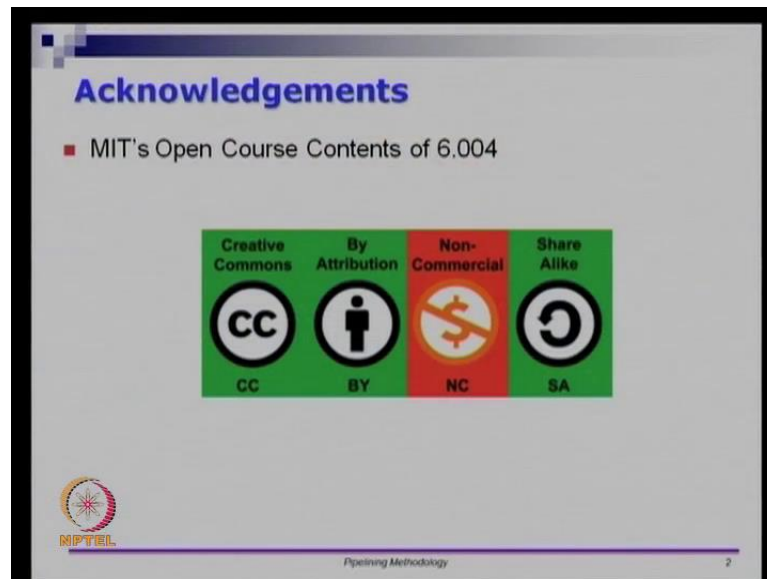
Module - 45
Interleaving plus Parallelism

In this module, we are going to look at what is called Interleaving and we will also see what is called Parallelism. So, in the previous two modules in this week, we looked at pipelining. So, I suggest that you go back and look at the pipelining video once more to understand latency, throughput and what not. But the key requirement is a k stage pipeline should be well formed, in every path you should have a flip flop and you should have k such flip flops going to the output.

The choice k is actually left to you, if you reduce k , typically what happens is the clock period will have to increase, which means your frequency reduces and the throughput reduces. However, you cannot go and blindly increase k , because if you keep increasing k , it is possible that your throughput actually does not improve, but the latency is suffering.

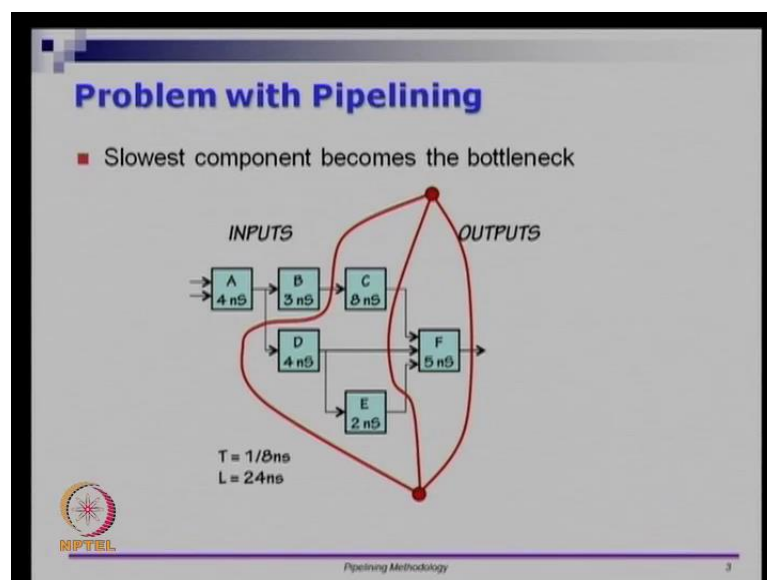
So, we do not have to increase the latency unnecessarily, given an input we want to process it so many time units. So, without increasing throughput if we are one only going to increase the latency that is not a good thing. So, in this video we will look at two different ideas, one called interleaving and another called parallelism, which will help us in improving the latency or the increasing the throughput or in reducing the latency.

(Refer Slide Time: 01:32)



So, as before I acknowledge MIT's open course ware for some of the material here.

(Refer Slide Time: 01:37)

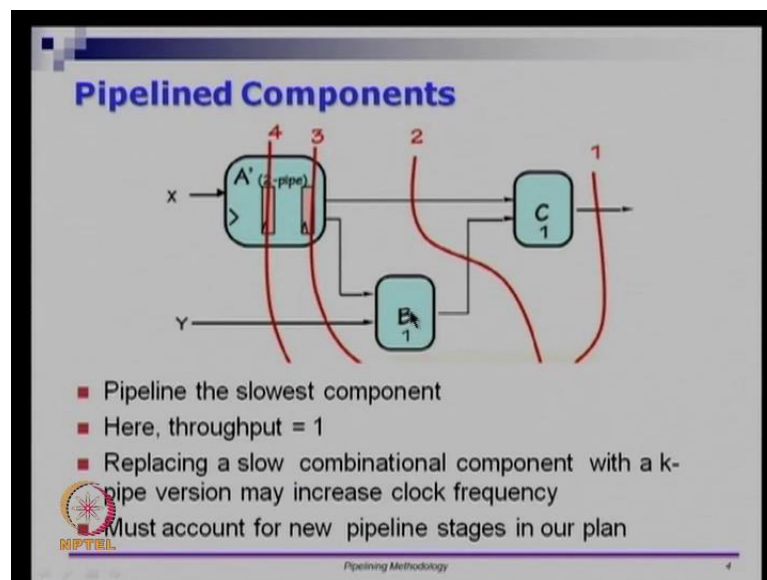


So, let us see the pipelining, the slowest component here is C and C is the bottleneck, I call this the weakest link in the previous video. So, C is a battle neck and we want to see whether we can somehow get around this lowest component. So, the problem is, because

C is something that we cannot divide, so as I said C is a module and we cannot divide the module C any further, if we cannot do that, then what can we do.

So, if C was indeed breakable, let us say C was breakable and C is a circuit which has some logic, let us say at some level I can break it into two different blobs. Let us say one with 5 time units, one with 3 time units and so on, then it may not be as bad as this case here, the throughput may actually be better than 1 by 8 nanoseconds. However, the first assumption is that, this block C itself is not breakable.

(Refer Slide Time: 02:39)



So, let us see the first case, so the pipeline components for some reason, if we take the slowest component. So, the other example that I had was, I had this one which had 2 time units, A had 2 time units delay, B had 1 time unit delay and C had 1 time unit delay that is another circuit that I showed in the last video, let us say somehow I was able to pipeline A. So, A was not a single combinational block, somehow I was able to pipeline A itself, which means I was able to insert a flip flop inside A and I also place a flip flop at the output of A.

So, I am going to call this pipeline circuit A dash and if I made that a 2 stage pipeline. So, this time unit two time period, let us say I broke it down as 1 and 1. If that were

possible, so then there is this path A dash which takes input from X. This circuit A dash produces something which is stored in this flip flop, then there is A double dash which is here, which takes this input and places the output here. If that happens, then from X through A dash within one time unit I can reach this flip flop.

Within another time unit, I can reach this flip flop and going from here to here will take two time units. So, that is still not reducing the latency, so the latency is not only in this single blob here, it is also in this path B and C. However, if I place flip flops in all these locations, let us see what is done. We start with the output, we place the flip flop register here, we place flip flop here, we go to the input, we place 2 flip flops one on this wire and one on this wire.

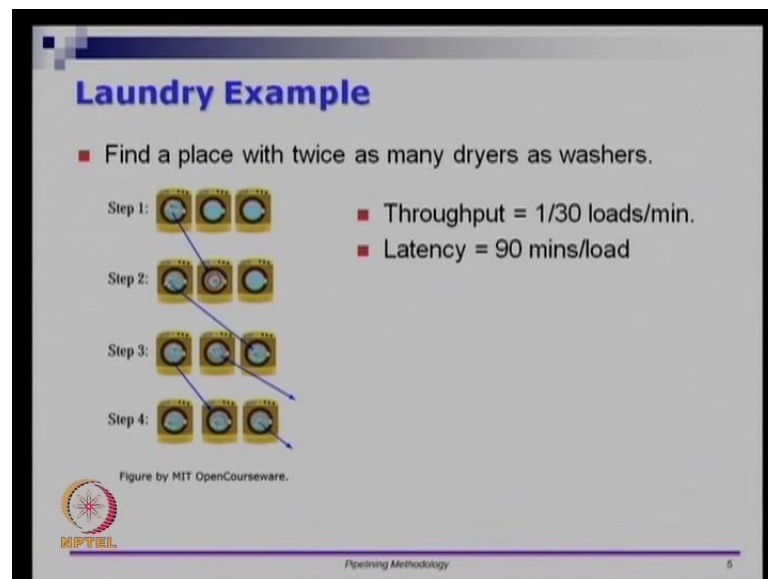
We go back one more stage, we place a flip flop here from this path on X, as well as one for Y and we place one more for the path on X and correspondingly to make it a k stage pipeline, we place 1 on y. So, we have 3 sets of paths, all these paths have 4 stages of pipeline. So, these are the 4 pipeline stages, one pipeline stage within A and this is a dummy flip flop it is just copying Y, another pipeline stage within A itself and another dummy flip flop on Y, the third stage is the output of A and the output of B and the fourth stage is output of C.

So, this is the well formed pipeline and replacing a slower component which is A, which was two time units of delay with the k pipeline version. So, we move, we took that combinational circuit, somehow broke it down, put flip flops inside and we made that a two stage pipeline. The moment we made this two stage pipeline, the other inputs should also go through two pipelines stages, and then finally we made a 4 pipeline stage here to get this circuit running.

However, just putting this two pipeline stage is not enough and if we did not pipeline here and here, the latency would still have been, the worst case would have been the B, C path that would have been two time units. So, we needed to break that down also to get a throughput of one per every time unit. So, the throughput of this system is 1, in every time unit I can get 1 input process. So, if I place an input here, it will take 4 cycles and then it will come as output.

But, meanwhile the second input could be given here, it will take 4 cycles. In the fifth cycle I will get the second input process, in the sixth cycle I will get the third input process and so on. So, after 4 cycles, if I have inputs which are continuously coming, every cycle after four cycle I will have an output.

(Refer Slide Time: 06:53)



So, let us look at the laundry example. So, one way to look at the laundry problem is, I have the washer and I have the dryer, I assume that there is only one washer and there is only one dryer. What if, I go to a different laundry, where there is a washer and there are two dryers, let us look at this. There is a washer and there are two dryers, if this is possible, then we can do something very interesting.

So, I take the clothes here, wash it, so step 1 is I wash, there is only one washer, the wash that I have I take it and put it dryer one. So, dryer one will start getting used at 30 time units, at 90 it will produce the output, dryer two is still not busy, because you do not have two sets of clothes that are washed. What I can do is, this is 30 time units, step 2 can start at 30 time units and I can put another load of clothes. So, that will get done at 60, the step 2 can get over at 60.

Because this other dryer is available I do not have to make this wet clothes wait inside

the washer, instead what I will do is, I will take those wet clothes and put it in the second dryer. So, let us see what happens now, so I start at time 0, at time 30 the first wash is ready, at time 60 the second wash is ready, I will give it to the second dryer. And at time 60, I can put a third load of clothes in the washer, because washer is free, at time unit 90 the third unit, the third wash load will get washed.

But, now let us see what happen to the first dryer. The first dryer started at 30, it will take 60 time units. So, at 90 time units the first set of clothes are dry, so this blue line that you are seeing here is the first set of clothes that got dry. Once, that got dried, the step 4 can start at 90 time units I can put a new wash in the washer, but the load that I got from here can go to the dryer, the third load that I washed can go to the first dryer.

But, when I am doing that, the second set of clothes is still getting dried in the second dryer. So, let us see this thing, what we have done is, the very first time step in the very first 30 time units, both the washer and dryer. So, the washer is used, but the dryer cannot be used, because there is nothing read yet. In the second cycle, one of the dryers is not used, but one of the wash was ready, so I use one washer and one dryer. In the third time step this washer, this dryer and this dryer all are being used.

So, from here on, if I have an infinite supply of clothes, all the dryers and the washer will be kept busy. So, I am making an effective use of the resources that is given to me. The washer as well as the dryers are getting used and every 30 time units, I will get one load washed, which means the throughput is now going to become 1 by 30 and the latency is still only 90 minutes per load. Because, a single set of clothes will require one wash and there will always be a dryer that will be ready.

So, one wash plus one dry is 90 time units, there is no wet clothes sitting inside the washer any more, we do not need that any more. As soon as the clothes are washed, I will know that there will be a dryer that is ready, I can take that and I put it in that dryer. So, essentially you, every 30 time units you can take a set of wet clothes out, you can put a new set of dirty clothes in, the wet clothes that you took out there will be one dryer that is ready to take that and you can supply it to that. So, that is what is going to happen every 30 time units.

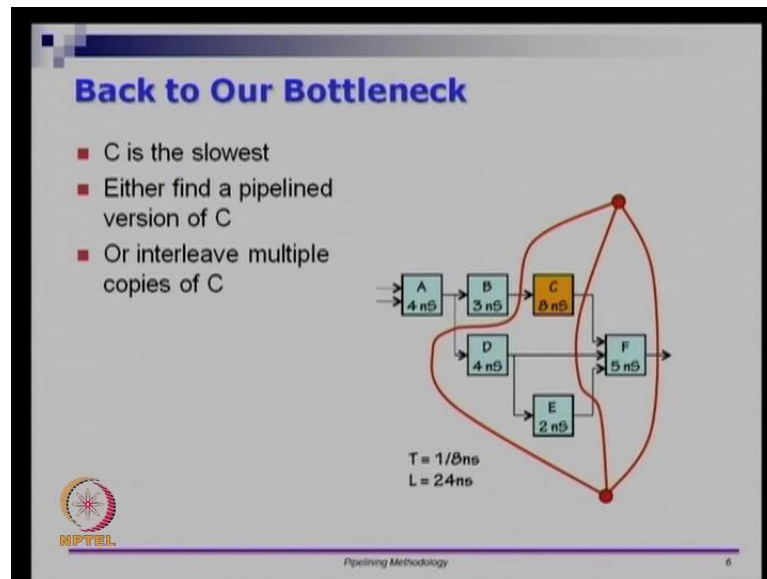
So, let us see this, as bunch of things that is there in every 30 time units, once you hit study state. Every 30 time units, there will be one set of clothes that we will get washed, one set of clothes that get dried and another one may still be in the process. So, I will be able to put one new set of dirty clothes in, I will be able to take one set of dry clothes out and because of that, I can transfer the one set of wet clothes to the dryer.

So, the throughput is 1 by 30 loads per minute and the latency is 90 minutes per load. So, we have not increase the latency, but by installing additional appliance or we have put more hardware and we got better throughput. So, by putting in more hardware, in this case we did not take the dryer and pipeline it. What we have done is, we have taken two dryers, instead of taking a dryer and I could have done something like this.

So, instead of buying one appliance for dryer, I can buy two appliances, one appliance which takes the wet clothes and maybe semi dries it and the second appliance which takes the semi dried clothes and dries it completely. If I have done that, then each of these appliances if they can do it in 30 time units, then that is called pipelining the dryer itself. Instead what I have done is, I have actually made copies of the dryer, I put a complete dryer, I did not expect the dryer to take any less than 60 time units.

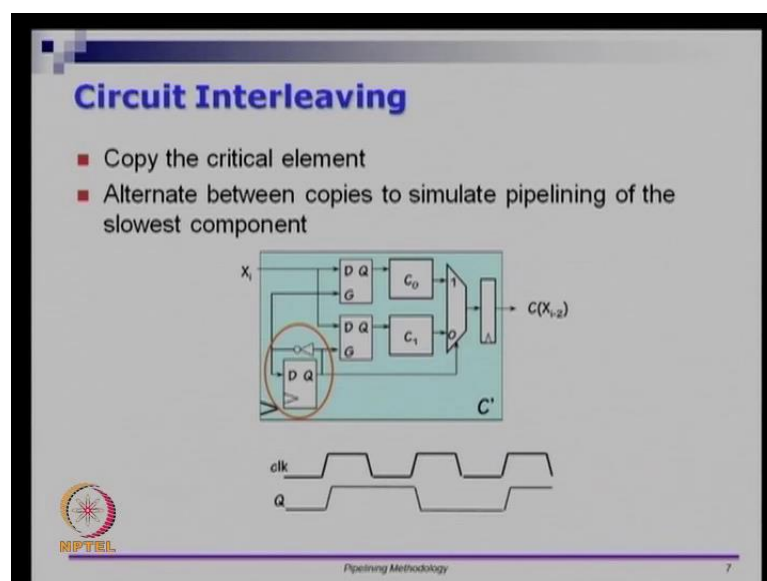
So, this dryer as well as this dryer are both going to take 60 time units, I did not reduce the delay of the dryer itself. But by putting an extra dryer, what I have done is I have made something really nice. So, by putting extra hardware I have done what is called interleaving.

(Refer Slide Time: 12:51)



So, going back to the bottleneck if... So, in this circuit C is the slowest, if I want to improve I can either go and take the C as a circuit and somehow find a pipeline version of the circuit itself. A pipeline version is one in which there will be flip flop stages within C itself, which means the combinational delay is 8 nanoseconds, but it is something less than 8 nanoseconds. Either you do that or if C cannot be pipelined, I can do interleaving.

(Refer Slide Time: 13:24)



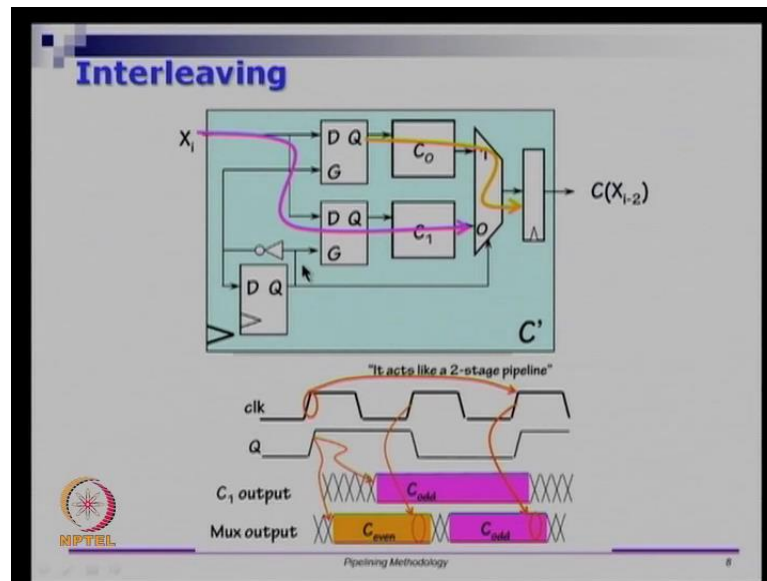
So, let us see this picture and how to interleave. What I am going to do is, I am going to have two copies of C. So, the critical element is C, I take two copies of C, so this C has no direct relationship to this circuit. So, let us say that C is the critical component, C is some critical component, I put two copies of C. So, C naught and C 1 essentially I have replicated the hardware, in this case I made two copies of dryer and what I will do is, I alternate between copies to simulate the pipelining.

So, this is equivalent of, if there is a set of clothes that have to be washed, either that will be dried by dryer 1 or dryer 2 and I will keep alternating between dryer 1 and dryer 2. Whatever comes out of the washer, I will either put it in dryer 1 or in dryer 2 and I will keep alternative between dryer 1 and dryer 2. So, that is the picture here, so what we have is, we have designed a circuit which is interleave.

So, let us look at the mechanics slowly, so we have some kind of a latch here. So, this is the D flip flop and... So, this is the gated D latch and this is a gated D latch and this is a flip flop. So, the output of this always has a flip flop, so because it is a pipeline circuit, a k stage pipeline should always have a flip flop. So, I put a pipeline register here and since the output quick could be coming from either this logic or this logic, I need a multiplexer.

So, this is we get two sets of clothes that are coming out of the dryer, it could come either from C naught or C 1, at a point of time only one is come. But it could come either from C naught or C 1. So, we put a multiplexer there and there is another logic which controls, how the timing is going to work. So, we will see this timing in the next picture.

(Refer Slide Time: 15:17)



So, we have this circuit here and let us look at this picture here, I have something called Q, the Q bar is given as the enable for this latch and Q is given as enable for this latch. So, at any point of time, either this latch is enabled or this latch is enabled, so if this latches enable this will be disable. So, if Q is 0, this is disabled and this is enabled, what that would mean is X_i can go into input of C naught.

However, if Q is 1, this latch is enabled and this is disabled, which means C naught will have the old input, but C 1 can take the new input. So, you can see how cleverly this Q controls the new input, whether it should go to C naught or whether it should go to C 1. If the new input has to go to C 1, then Q should be on, if the new input has to go to C naught, then C should be off.

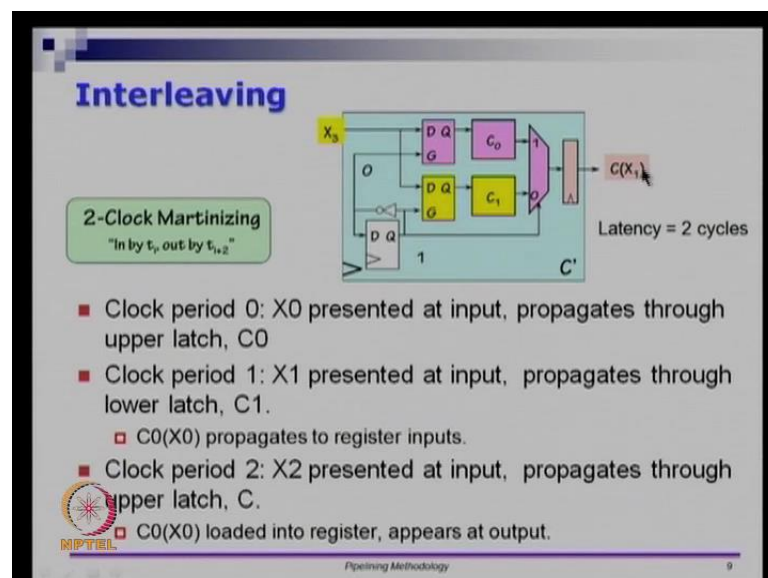
And when the new input goes to one of them, the other input is supposed to be still processing the previous output, previous thing. So, if I am going to put a new set of washed clothes into the dryer into some dryer, the other dryer is still processing the one of the previous loads, it is not ready yet, so that is the meaning of this. So, what we have is, there is a clock that goes every time period and Q, if you notice what is happening into Q, there is a clock pulse that is coming in and Q is going to toggle at the rate which is one half of C.

So, this is like a single flip flop, so Q is going to toggle whenever the positive edge trigger comes in. So, when the clock pulse comes in a Q was 0 changes to 1 when clock come here it goes back to 0 and so on. So, Q is toggling at half the rate, because you have double the time period the clock frequency of Q is half and this Q is going as latch enable for this as well as here.

So, if you look at muxes output, the C 1s output is here, so C 1 get some input and it is going to take let say this is our washer, it is going to take 60 time units to process mean while the even block which is on the top will get ready, if this is going to take 60 time units half of that in 30 time units, the C naught would be ready with it is data. And after this 60 is over, this is also ready and so on.

So, what we have is we will get even followed by odd, followed by even, followed by odd and so on. So, again if you go back to the laundry example, all the even number loads will come out of one dryer and the odd number loads will come out of one dryer and this is the 2 stage pipeline. Because, the inputs were given at some stage after two clock cycles the outputs are getting out, so this is the 2 stage pipeline.

(Refer Slide Time: 18:32)



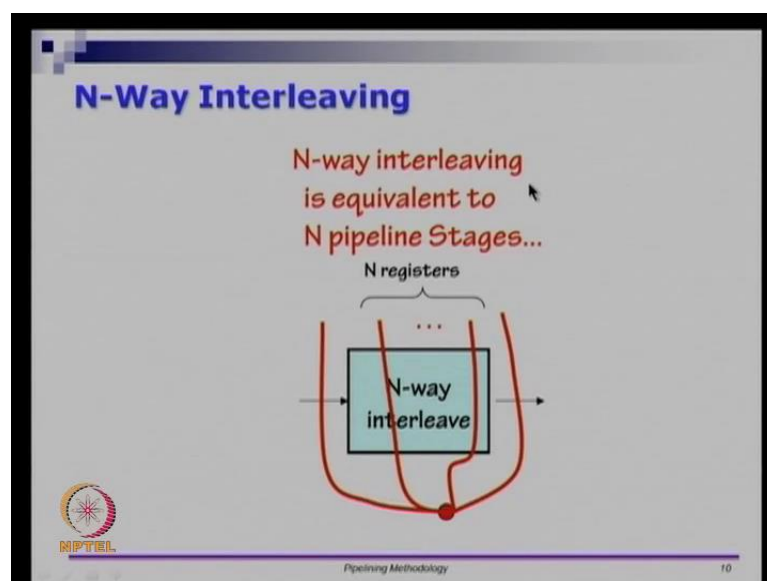
So, this is also called 2 clock martinizing, so if then input is in by t_i , the output is ready

by t_i plus 2. So, the latency of this system is 2 cycles. So, let see how this will work at clock period 0 if I put X_0 at the input it will propagate through latch C_0 . So, it will propagate through that and will be ready as input to C_1 at clock period 1 what I am going to do is, I am going to give X_1 a new input that is presented at the input it propagates through this latch and goes as input to C_1 in clock and C_0 of X_0 is propagating through the register inputs to the register input.

Clock period 2 X_2 will be presented here and it will propagate to the latch C_1 again. So, at 0 it goes to C_0 at two again will come to C_1 , but C_0 of X_0 will get loaded into this register. So, C_1 of X_1 will come out of this register, so if you put a time period is 0 there is X_0 here that will be made available to combinational logic here and at the end of time period 2 it will be made available to register here. So, when the clock pulse comes at clock period 2 it will make that and put it outside.

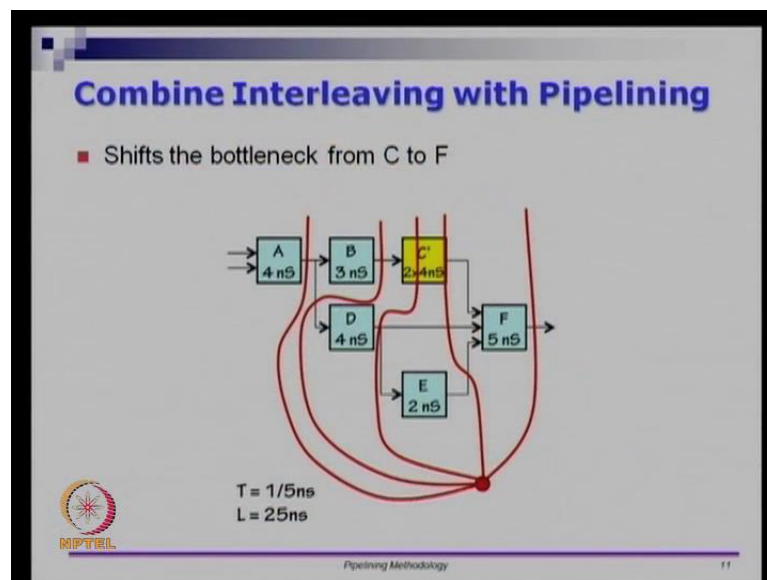
And in the next clock pulse whatever is coming out of this mux will be ready here that will come out that will be C_1 of X_1 and so on. So, if the X_3 is the input X_1 is the output that I can get, if X_4 is there, C_1 of X_2 is the output and so on. So, this is a interleave circuit.

(Refer Slide Time: 20:26)



In general N way interleaving is you take a single circuit and you make N copies of that. So, imagine if the washer was taking 30 time units and the dryer was actually taking 90 time units, not 60 if it were taking 90 time units. So, this washer can produce a load of clothes in every 30 time units, then I can actually go and triplicated I can have three dryers and the first wash clothes will go to dryer 0, the second washer clothes will go to dryer 1 the third one will go to dryer 2 and every 30 time units I can still get one set of cloth set are also dried out. So, but what that would mean is have bought one more dryer, so interleaving comes at the expense of copying hardware I need more and more hardware and N way interleave is equivalent N pipeline registers.

(Refer Slide Time: 21:26)



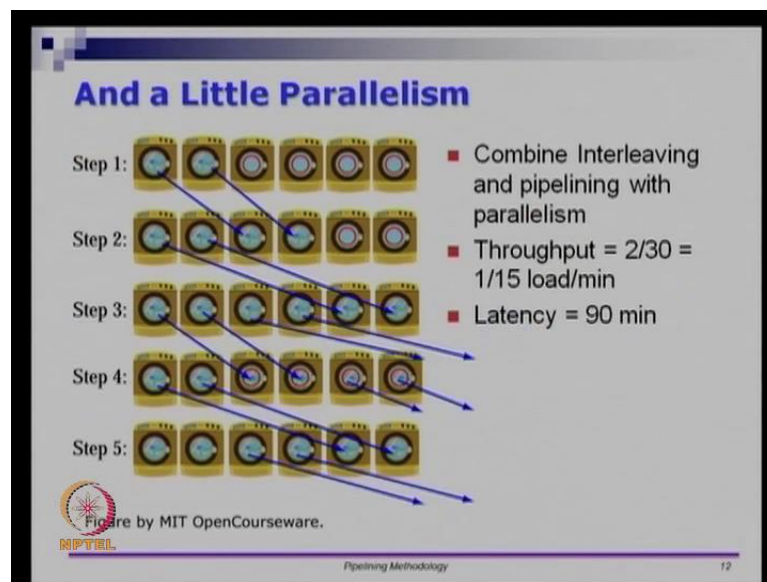
You can also combine interleaving with pipelining, so in this example what is happened is we took C and we replace that with the two stage pipeline. So, in this example it is a two stage pipeline, let us assume that the C was broken down into 4 and 4 time units. So, this is 8 time units, now is broken down as 2 to 4 in 4 and 4 time units and if that is there you may actually be able to improve the clock period.

So, the bottleneck is not seen any more, because C is now C dash and C double dash it has two blocks, the block bottleneck will then become F, F is now 5 time units. So, the throughput will become 1 by 5 nanoseconds and because you are going to put pipeline

registers which are matching in this example there are 5 of this red lines which means the pipeline stages is 5. So, 5 into 5 nanoseconds is 25 nanoseconds.

So, from 20 nanoseconds the latency in the earlier one was 24 nanoseconds, now it is 25 nanoseconds. But the throughput is 1 by 5 nanoseconds, so every 5 time units I am getting one thing out the combinational circuit every 20 time units I was getting 1 out, the 3 stage pipeline that we had earlier every 8 time units I will get 1 input processed. But now every 5 time units I can get 1 input processed, so this is the nice thing to have.

(Refer Slide Time: 22:52)



You can also add a little bit of parallelism, so if I have 2 washers and 2 dryers 4 dryers. So, earlier I showed this example where I had 1 washer and 2 dryers, this 1 washer for plus 2 dryer could sustain one load in 30 minutes. But if I have 2 washers and 4 dryers then I can sustain 2 loads every 30 minutes, because I have duplicated the whole setup, 1 washer plus 2 dryer can give me 1 load in every 30 minutes. So, 2 washers and 4 dryers can give me 2 loads in every 30 minutes, if I have 3 washers and 9 dryers I can get it done as 3 loads in every 30 minutes.

So, the throughput will increasing if I go and increase the parallelism in the setup. So, the more washers and dryer combinations I have the better the higher the parallelism in the

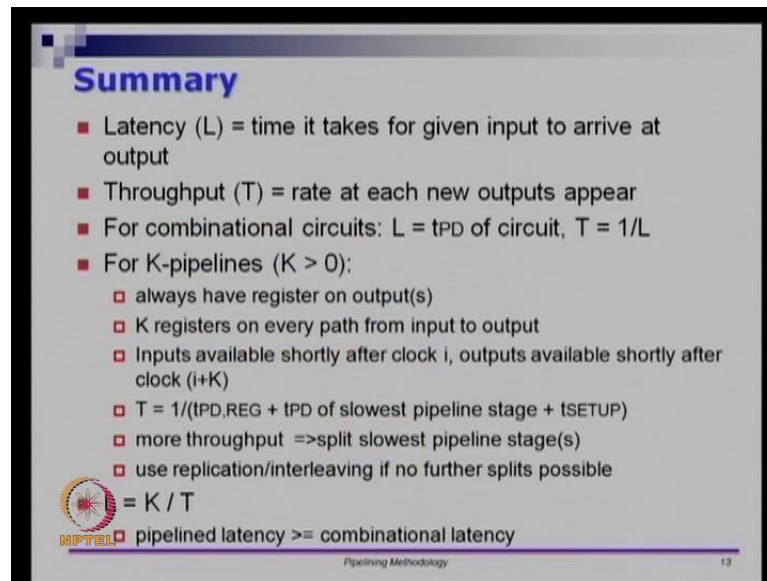
circuit will be which means better the throughput will be, but higher parallelism comes at the expense of lot of hardware. So, if I want two loads every 30 minutes then I need to by 2 washers and 4 dryers.

If I want 3 loads every 30 minutes then I need to spend on 3 washers and 6 dryers and so on. So, it is not free latch it is not free of cost, better throughput always comes at the expense of higher amount of hardware. So, it depends on whether your optimizing for throughput or your optimizing for hardware, if your optimization is for throughput you should be ready to spend more in terms of logical units, we have to have more and more gates and that will help you and getting more throughput.

However, there can be some diminishing return even in throughput, even if you through more and more hardware we saw that unless thus a fundamental way of changing the circuit itself, higher amount of hardware does not nesses or higher level of pipelining does not necessarily mean that you will get better throughput. So, keep a lot of these things in mind, so in this picture we have pipelining, we have interleaving and we have parallelism, the parallelism comes from the...


So, we will start a pipelining as soon as the washer is over we take it into a dryer and put it into a dryer that is pipelining. And we had 1 washer plus 2 dryer combination that is equivalent to interleaving, because we do not want the wet clothes to weight inside the washer if you had 2 dryers which is a copy of... So, we had 2 dryers, one is copy of the other that took care of interleaving. So, that is just the left part of this, the right half is just duplicating that work completely, so that is parallelism. So, we have pipelining, we have interleaving and we have parallelism all in the same picture in this example.


(Refer Slide Time: 25:54)



Summary

- Latency (L) = time it takes for given input to arrive at output
- Throughput (T) = rate at each new outputs appear
- For combinational circuits: $L = \text{tpd of circuit}$, $T = 1/L$
- For K-pipelines ($K > 0$):
 - always have register on output(s)
 - K registers on every path from input to output
 - Inputs available shortly after clock i, outputs available shortly after clock (i+K)
 - $T = 1/(\text{tpd}_{\text{REG}} + \text{tpd of slowest pipeline stage} + t_{\text{SETUP}})$
 - more throughput => split slowest pipeline stage(s)
 - use replication/interleaving if no further splits possible

 $= K / T$

 pipelined latency \geq combinational latency

Pipelining Methodology 13

So, in summary latency is the time taken for a given input to arrive at the output, even in this picture the latency is still 90 time units, it is not getting any better. So, 30 minutes for washing and 60 minutes for drying, we have not broken the latency into anything less, until we come up with the different dryer technology itself which will take the less than 90 minutes, the latency is going to be definitely 90 minutes we cannot make it any better.

Throughput is the rate at which the new outputs are appearing, this also means at the rate at which you can give new inputs. So, for combinational circuits the latency is the propagational delay, the worst case propagational delay and the throughput is one over the latency for K pipelines, where K is greater than 0 the summary is always have it on the outputs, you should have K registers on every path from input to output, input is available shortly after clock i and outputs are available shortly after clock i plus k.

So, that is how you should have take care of things, the time period that you can use is 1 over the propagational delay plus the propagational delay of the slowest pipeline stage plus the setup time. More throughput means, we have to split slowest pipeline stages, unless you split the slowest pipeline stage we will not be able to get more throughput. So, you can do that by either replicating or by interleaving if no further splits of possible.

Finally, latency is K divided by T , so T is the time period, so T is not throughput here T is time period. So, T is actually throughput, so K stages divided by T will give you the latency. So, pipeline latency is usually greater than combinational latency, but pipeline throughput is usually much better than combinational throughput. So, that brings me to the end of this module, so this is the end of module 45.

In module 46 and 47 what we are going to see is, we are going to see how to write some of these things in Verilog. So, in the 3 modules in this week we have seen what is pipelining, what is interleaving, what is parallelism and how to combine all the three.

So, thank you and I will see you at module 46.