

**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**

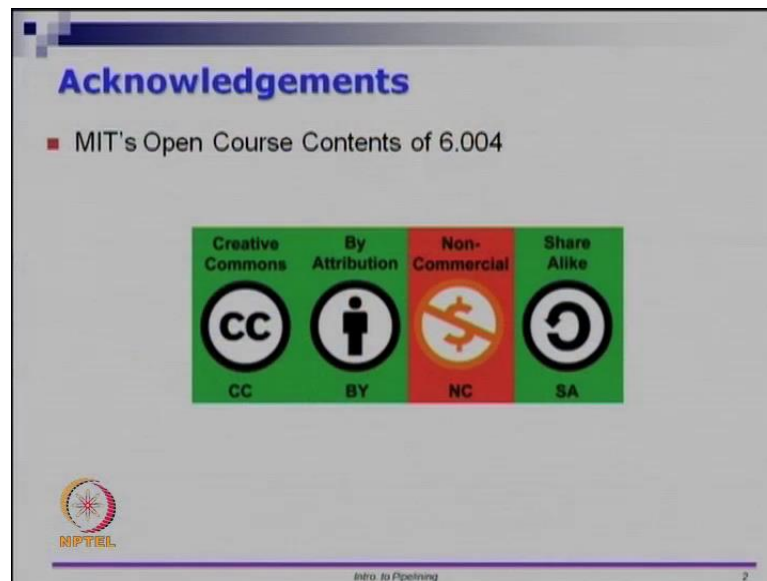
**Module – 43**  
**Pipelining**

Welcome to the 8th week of video lectures. So, in this course Digital Circuits and Systems, we are getting towards the end. So, upfront I want to tell you that, there is some material that is not been covered so far and there were a few E-mails on the forum, things about number representations and about arithmetic circuits. So, I decided to actually take one more week to cover that. So, this week is not the last week in terms of videos.

So, there will be one more week in which I will cover basic arithmetic circuits as well as number representations. So, this is the last, but one week, so we are in the 8th week now and this week, what we will look at is, an interesting topic called Pipelining. So, pipelining has a lot of... So, we do a lot of things in the real world which are actually pipelined, we never call them pipeline though, we will take a real world example of that sort and we will see, what is the inspiration in digital circuits to make them faster.

So, this is a very important topic and this is usually not covered in at least second year under graduate material. This may be covered in either, 4th year under graduates or in the first year graduate level. But, it is a very useful technique to know and if you understand this well, you will be able to understand processor design. If you are going to take any course on CPU's and how they are designed, this is very useful.

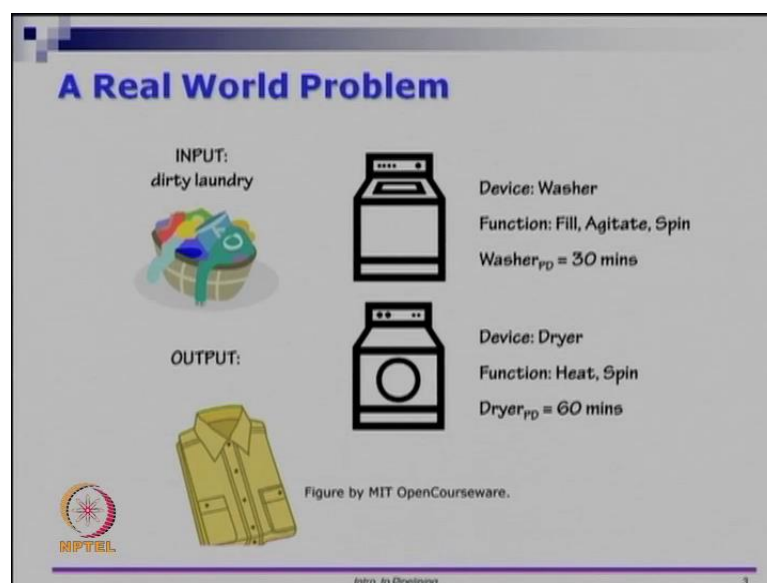
(Refer Slide Time: 01:46)



So, the part of the material for this week's videos are actually taken from this course from MIT and this is called computational structures, the course number is 6.004. So, MIT in the US has been offering a course on computational structures for a long time under what is called Open Course Ware, OCW they call it. And, so they allow professors and others to use this, so it is under what is called creative commons license.

So, if I attribute to it and as long as I am not using it for commercial purposes and whatever material I am making as long as I share it with others, MITs license allows me to do that. So, I am going to use some of the material from the computational structures course to teach this.

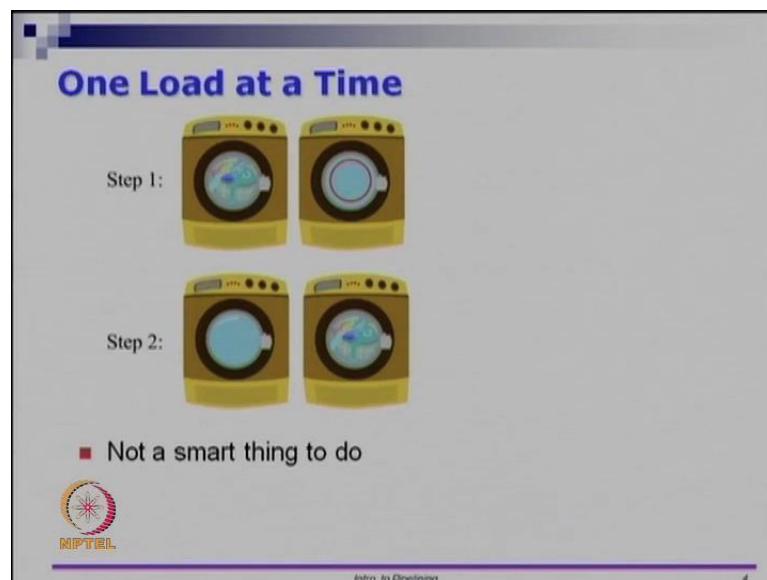
(Refer Slide Time: 02:36)



So, let us now look at a real world problem, so let us say we have a laundry to do and a laundry typically requires you to wash and dry them. So, let us say I have two different machines, I have a washer, the washer needs to fill up, needs to agitate and spin and let us say washers propagational delay is 30 minutes. So, it takes about 30 minutes to wash clothes. Dryer, on the other hand is a separate device is available as a separate appliance, you can buy a dryer and there are dryers which will spin and even apply heat to it. So, that when you get the cloth outside from the dryer, they are already dry, you can fold them and put it aside, you do not even have to hang them for drying. So, these appliances usually take more time, let us assume that the dryer take 60 minutes. So, this is typical of a washer and dryer combination. So, in India when we usually by dryers, they do not dry them completely, so they are still wet and you have to hang them in clothes and so on.

But, there are these appliances available where you can actually dry it, it will come out hot and all you have to do is just fold and keep it aside. So, we have a washer which takes 30 minutes and we have a dryer which takes 60 minutes and let us see how notion of laundry, if you want to wash clothes how it can get done.

(Refer Slide Time: 03:59)



So, if we do one load at a time then what we can do is, we can load the clothes, when we load the clothes in the washer it is going to take 30 minutes and during that time, let us say I have this dryer also, the dryer is not going to do any work. So, when the washer is washing, the dryer is not doing any work and once the wash is over, I have to take the clothes from the washer, put it in the dryer that is in step 2. So, when the clothes are drying, the washer is not really doing anything.

So, clearly this is not a smart thing to do. So, if I have one load of clothes, let us I have only one basket of clothes I have to wash, then for 30 minutes I will wash and for 60 minutes I will dry. But, when I am washing the dryer is not used, when I am drying the washer is not used. This is okay, if I have only one load, but if I have let us say 5 loads of clothes that I want to wash, not using the washer and dryer is not sensible, so it is not a very smart thing to do. If I take one load at a time and I use only the washer and then only the dryer and so on. We will see how this is related to circuits in a little while, but let us take this real world example to it is logical conclusion.

(Refer Slide Time: 05:10)

**One Load at a Time**

Step 1:

Step 2:

Total = Washer<sub>PD</sub> + Dryer<sub>PD</sub>  
= 90 mins

■ Not a smart thing to do

GPTEL

Intro. to Pipelining 4

So, the first thing will note down is, the total time taken is the propagational delay of the washer plus the propagational delay of the dryer. So, that is 30 plus 60 minutes that is 90 minutes.

(Refer Slide Time: 05:23)

### Doing N Loads of Laundry

■ The combinational way




$$\text{Total} = N * (\text{Washer}_{PD} + \text{Dryer}_{PD})$$
$$= \underline{N * 90} \text{ mins}$$



Image by MIT OpenCourseWare.


MPTEL

Intro. to Pipelining 5

Step 1: 

Step 2: 

Step 3: 

Step 4: 

...

Let us say I want to do  $n$  loads of laundry, so I have some number  $n$  and if you want to do it in what is called the combinational way or the non smart way of doing it. I have, so many clothes and what I will do is in step 1, I will take one of these laundry baskets wash it, that will take 30 minutes. Then, I will take those clothes which are washed, put them in the dryer that will take 60 minutes. So, 90 minutes one load is done, then in the 90th minute I will take the second load, put it in the washer, that will take 30 minutes. So, 120 minutes over all plus another 60 minutes to dry the second load, so in 180 minutes I will finish two loads.

So, if I want to finish  $n$  loads, then I will take  $n$  into 90 minutes, so the combinational way of doing it. So, each wash in dry cycle takes 90 minutes and I have  $n$  of these loads, so over all I will require  $n$  into 90 minutes. And you can see the problem here, at any point of time, so I have one single washer and one single dryer, when the washer is in use, the dryer is not being used, when the dryer is in use, the washer is not being in use.

So, in the real world if you had  $n$  loads of clothes, you would really not operate it, in this way. Would you think about using only the washer or only the dryer? So, you would think about, when the washer is washing, you would put clothes. So, when the first load dryer cannot run, so you will wash it and as soon as the wash is over, you put it into the dryer, now the washer becomes free. So, in step 2 the washer is free, you can technically go and put another load inside and this will take 30 minutes to finish, but over all this is going to take 60 minutes.

So, by the time you wash two loads of clothes, you would have done only one load of drying. So, you need to wait for the dryer and then, put it and so on, we can figure out how to do these things. But, clearly taking only one wash and one dry at a time and not using the other one does not make sense.

(Refer Slide Time: 07:35)

### Doing it the Smart Way

- Pipelining
  

$$\text{Total} = N * \text{Max}(\text{Washer}_{\text{PD}}, \text{Dryer}_{\text{PD}})$$


$$= \underline{\quad N*60 \quad} \text{ mins}$$
  
- Technically,

  - $N*60 + 30 \text{ mins}$


- Steady state:

  - Infinite supply of inputs
  - $N*60 \gg 30$ , so ignored


Step 1:



Step 2:




Step 3:



...

Figure by MIT OpenCourseware.


Intro. to Pipelining
6

So, the smart way to do that is doing what is called pipelining. So, the way the pipelining would work is, ideally we would like this. So, initially the dryer is not being used, you take the first load and put it inside for wash and that takes 30 minutes. Whenever it is done, take it and put it in the dryer and take the second load and put it in the washer and whenever the dryer is ready and when if the clothes are washed, you take the wash, put it in the dryer, take a new load and put it in the washer and so on.

So, doing something like this is called pipelining, so this pipelining is something that we do in the real world more often than we can imagine. So, if you are cooking for instance you do the same thing. So, when you are... So, may be you put something on the stove, but then you are cutting some vegetables, you are boiling the water also and so on, it is not that you do all of these steps sequentially. So, when something is cooking, you go and prepare something else and when that is cooked, then you move that to the stove and you start preparing for the next dish and so on. So, that is also actually an example of pipelining.

So, pipelining is the smart way of doing activities and what pipelining gives you is, it gives you two things, one these units are going to be busy, we are not going to under

utilize the washer or the dryer. So, in 90 minutes if I use 30 minutes for washing, 60 minutes for drying. So, in 90 minutes the washer is not used for 30 minutes and the dryer is not used for 60 minutes, so it is not an effective use of the appliance. Imagine, I have to rent the washer and dryer, I want to get as many loads as I want, clearly you would not do it this way, so that is one thing that you get from pipelining.

The second thing from pipelining that you get is, if we have  $n$  washes and dries that you have to do run, then it reduces the amount of time that you need for doing this. So, let us see what it will take, the total time it will take is  $n$  times not the summation of washer plus dryer, but the max of washer plus dryer. So, the one that is constraining the load to go is washer any way gets done in 30 minutes, the dryer takes long a time, so for a every load...

So, let us look at this, the very first load will take 30 minutes and once that 30 minutes is over, I take that and put it in the dryer. With this, this is going to take 60 minutes, but mean while I can get another load washed. So, I wash it now, so the maximum of these two tasks is going to take 60 minutes and that is what this max is. So, max of washer and dryer is 60 minutes, it is  $n$  into 60, so over all you will require something which is  $n$  into 60.

So, this is actually an approximation, technically you actually need  $n$  into 60 minutes plus the 30 minutes for the very first washer, when there is no drying that is happening. So, you still need to spend that 30 minutes, so I suggest that you go and work this out, so what we are doing is, at any point of time we keep one wash ready. So, there all lots of loads of cloth that we need to wash, at any point of time, if you want one of them ready for that dryer and whenever the dryer is ready, if you want to put it for wash, put the wash clothes for drying that is the set up that we have assumed here. This will require  $n$  into 60 minutes plus 30 minutes.

So, in the steady state if I have an infinite supply of inputs, then  $n$  into 60 will be much greater than 30. So, saying that the total time taken, if you have a pipeline system is  $n$  into 60 minutes is okay. Imagine,  $n$  being 10, 20, 100 and so on, if I want to wash 100 loads of clothes, this 30 minutes is not a big deal. The very first 30 minutes that we use, when the dryer is not being used is not a big deal. So,  $n$  into 60 is a good enough approximation, we can forget the fact that we took another 30 minutes in the beginning.

(Refer Slide Time: 11:41)

**Performance Measures**

**Latency:**  
The delay from when an input is established until the output associated with that input becomes valid.

(Combinational =  $\frac{90}{1}$  mins)  
(Pipelined =  $\frac{120}{1}$  mins)

**Throughput:**  
The rate at which inputs or outputs are processed.

(Combinational =  $\frac{1}{90}$  outputs/min)  
(Pipelined =  $\frac{1}{120}$  outputs/min)

Assuming that the wash is started as soon as possible and waits (wet) in the washer until dryer is available.

MPTEL

Intro. to Pipelining 7

So, let us look at the so called performance measures that are relevant to pipelining. There are two measures that are relevant to pipelining, one is called latency and the other is called throughput. Latency is defined as the delay from when an input is established, until the output associated with that becomes valid. So, essentially for our cloth washing example, the latency is if I have some unwashed load of clothes, it will need at least 30 minutes for washing and 60 minutes for drying, so it will take 90 minutes.

So, the latency if I do it combinationally it is 90 minutes, whereas if I do it in a pipeline manner, let us go back to the picture ((Refer Time: 12:29)). So, if I take the load then I put it for wash which takes 30 minutes, but if the dryer let us say, I start wash and dry at the same time so, at the time step 2 I have taken the first wash and I have put it here and I have taken a new load and put it for wash. When I have done that, then this wash will actually get over in 30 minutes itself, but we are waiting for another 30 minutes, so that the dryer is ready.

So, I have already spent 60 minutes in some sense, for 30 minutes for washing and 30 minutes for waiting for the dryer and then, another 60 minutes for drying. So, which means the latency in the pipeline set up requires 120 minutes. So, the assumption is that wash is started as soon as possible and waits in the washer, until the dryer is available. If you are allowed to take the wet cloth out, you can actually load it with another set of clothes.

So, we are assuming that the wet clothes remain in the washer itself till the dryer is



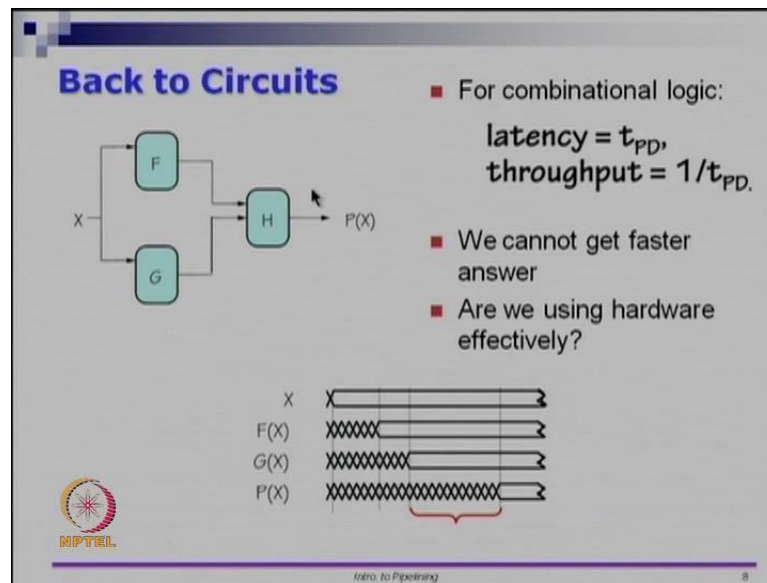
ready. So, if you make that assumption, then the pipeline system has a latency of 120 minutes. So, latency is from the time when you started the work, how long did it take you to finish the work, so that is called latency. So, the pipeline systems may add to the latency of the work, so instead of 90 minutes we have taken 120 minutes.

So, a single load of clothes may take more time to wash, but what you get from a pipeline system is higher throughput. So, throughput is defined as the rate at which the inputs or outputs is processed. So, let us look at this, for the combinational set up once every 90 minutes you get an output. So, you wash and you dry and once every 90 minutes you get a load. So, the combinational throughput is  $1 \text{ over } 90$  outputs per minute.

So, on a permanent basis if you ask, then I can say that 1 by 90 loads are getting done in every minute. So, even though it takes 90 minutes to produce the first load, it just like measurement, it just a rate. So, the rate at which the loads are process this  $1 \text{ over } 90$  outputs per minute, whereas in a pipeline system every 60 minutes you are ready to take the next load because... So, you will move it is from the washer to the dryer, the washer becomes free, we can take the next load and put it.

You know that in the next 60 minutes, the dryer will become free, you can take the wet clothes, put it in the dryer, the washer again becomes free and so on. So, for every 60 minutes I can go and put a new load of clothes in, so the throughput on the pipeline system is more than that of the unpipeline system or the non pipeline system. So, the pipeline system is able to process  $1 \text{ over } 60$  outputs per minute. So, it is going to give... So,  $1 \text{ over } 60$  is larger than  $1 \text{ over } 90$ , so the throughput of the pipeline system is better than the throughput of the combinational system.

(Refer Slide Time: 15:33)



So, let us get back to circuits, let us say we have a circuit like this. I have  $X$  which is an output and I have 2 inputs  $F$  and  $G$  which they go to  $H$ . So, assume that  $F$ ,  $G$  and  $H$  are all combinational to begin with. So, what we are getting out of the circuit  $P$  of  $X$  is some composition  $H$  of  $F$  of  $X$  and  $G$  of  $X$ , so  $H$  is some function. So, you can think of it as  $F$  is some AND gate, this is some OR gate that is some XOR gate or something like that. Let us assume that all these are combinational blocks. Let us see how this whole thing works.

So, you provide  $X$  which is an input to the system,  $F$  and  $G$  are going to take their own time to process  $X$ . So,  $F$  of  $X$  takes some time and in this example let us assume that this combinational block  $G$  is going to take more time. So, you have  $G$ ,  $H$  you can process only after  $F$  and  $G$  are both in a steady state, so you wait for both of those. So, this is the time taken by  $H$  once  $F$  gets settled down and  $G$  gets settled down then you have to wait and only then you can process  $H$ .

So, for combinational logic the latency that we have is the propagation delay itself, so let us again look at this picture here. So, once I have placed  $X$ ,  $F$  of  $X$  is going to get ready after some time and since  $G$  is taking a little more time. So, it looks like  $G$  is getting ready only at this time point and at this time point  $F$  and  $G$  are both ready. So,  $H$  has some finite time and only then you process, only then you get output. So, in terms of delay what we have is maximum of  $F$  and  $G$  plus  $H$  that is the time that it will take.

So, if this is 2 units delay and if this is 3 units delay and let us say  $H$  is 5 units delay. So,  $F$  will place the... So, if  $X$  will get transformed it two time units will be available here  $G$ ;

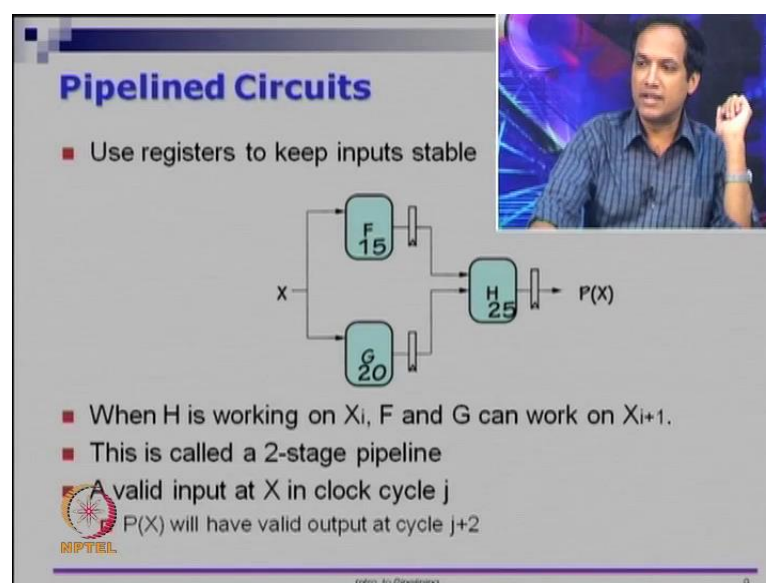
however, will take three time units. So, I have to wait for the latest are the one which is coming late. So, F of X is coming earlier then G of X I have to wait for G of X and then H will take F of X and G of X and take some amount of time and process it and send it out as P of X.

So, the latency in combinational logic is whatever the propagation delays and you know how to calculate the critical path, if I give you two blocks like this and if give you another block this, you know how to calculate the propagational delay, we did this in week 6. So, if you do not remember it go back and review the videos in week 6, we did this. So, the latency is the time at which X became study to the time at which P of X became available. So, it is the time difference between this pre from here, all the way up till here that is the propagation delay.

The through put is I can once I give this inputs, I have to wait and keep the input study till the output becomes study which means I can only give one input every latency whatever the latency is. So, one by the propagational delay is latency for combinational circuits. We cannot get a faster answer, then the latency that is given and the number of inputs I can give to this whole chip is constrain. So, I can I can only process as many as the latency will allow me to.

So, the question is are we using this hardware effectively or not, so let see if you want to do it as a pipeline circuit, we are going to use registers and keep the input stable.

(Refer Slide Time: 19:22)



So, in this example what is done is we are assuming that F takes 15 units of time and G

takes 20 units of time, H takes 25 units of time. So, in the non pipeline version the H will have to wait for the one whichever is delayed F for G. So, G it is 20 a time units the G takes plus 25 time units that H takes. So, you have to wait for 45 time units, if I place X the TPD in the combinational setup is 45 time units.

But, when you do pipelining what you do is, you place what are called registers, you put registers here, here and here. So, the moment you do that if I go and look at the rate at which the inputs can be given, if I go and look at the flop to flop delay, the flop to flop delay is 25 time units, because in this flop this flop is 25, this flop this flop is 25 both of them are 25 so, the maximum is 25 and if I assume... So, if I go and look at the input to flop delay that is 15 here that is 20 here, so the maximum of this is 20.

So, the input to flop delay is 20 time units and flop to flop delay is 25 time units. Because, this flop to flop delay is greater than the input to flop delay, I cannot give any inputs any faster than the flop to flop delay itself. So, I can keep giving inputs every 25 time units. So, if I bring in an input at time unit 0, at time unit 15 it will be available here at time 20 it will be available here, but to the system I am going to give 25 time units. So, this is equivalent to the wet clothes that are waiting in the washer itself.

So, they wait here at the 25th time unit they get registered, because the clock pulse comes in. Then, when the clock pulse comes in here, it will take 25 time units to go to this register, meanwhile you can place the next set of inputs here which will take 50 not 20 time units to process they will wait here till the next clock pulse comes in and takes these in pulse. So, what we are doing is, we are essentially when you pipeline a circuit we are making effective use of the hardware.

So, when H is working on  $X_i$ , so if I give X at time unit  $i$  let me call that input as  $X_i$ , this the 5th input or the 7th input or the 9th input and so on. If H is working on  $X_i$ , F and G are actually going to work on  $X_i$  plus 1. So, if H is processing the 5th input, then F and G will be processing the 6th input and P of X will be the 4th inputs output and so on. So, when H and G are working on  $X_i$  F and G I will work on  $x_i$  plus 1 this allow...

So, if we go back to the laundry example when I am drying the third load I can wash the 4th load and the second load is ready outside. So, that is the analogy here, so here you will have  $X_i$  minus 1, this is  $X_i$ , this is will be processing  $X_i$  and F and G will be processing  $X_i$  plus 1, you cannot give  $X_i$  plus 2 at this point of time. So, this setup is called 2 stage pipeline, it is called 2 stage pipeline, because there are two stages to the

moment I place the input, it goes through two stages, one stage which is here and another stage which is here. So, this is called a 2 stage pipeline.

If I go and look at the number of flip flops that X has to encounter from going from X to P of X through this path it requires one flip flop stage and another flip flop stage here. So, it is 2 stages if I come from here it takes one flip flop stage and another flip flop stage, this also requires 2 flip flop stages. So, this is called a 2 stage pipeline, a valid input at X in clock cycle j will get P of X in cycle j plus 2 I already mentioned that.

(Refer Slide Time: 23:29)

**Pipelined Circuits (contd)**

- Consider the delays inside the block the flipflops have zero delays

Diagram: Input X splits into two paths. The top path goes through block F15, and the bottom path goes through block G20. Both paths then feed into block H25, which produces output P(X).

	latency	throughput
unpipelined	45	1/45
2-stage pipeline	50	1/25
	worse	better

MPTEL

Intro. to Pipelining 10

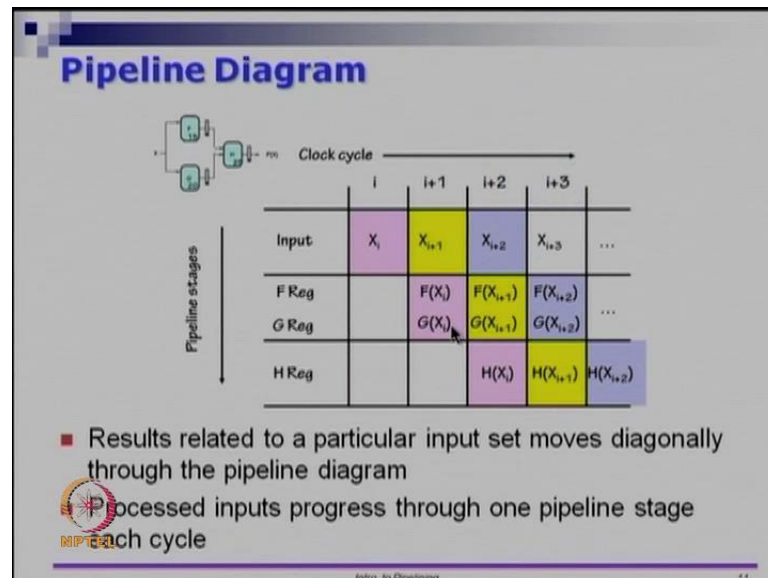
So, if you assume that all the flip flops are 0 delays, then the unpipeline latency will be 45 time units. So, it is maximum of 15 comma 20 plus 25, so it is 45 time units. So, the through put is 1 by 45 a 2 stage pipeline; however, will have a latency of 50 time units, because even though this gets ready in 55 or 20 you wait for this extra time period to the clock pulse to come and register here.

So, here you will have 25 time units of processing, here you will have 25 time units of processing. So, over all the latency is 50, if I give an input here it will take 50 time units for P of X to be ready. So, the latency has increased; however, the through put is every 25 time units I can give a input. So, it is 1 by 25, so earlier we had 1 over 60 loads per minute, here I can give 1 by 25 inputs per minute or input inputs per time unit, I can give 1 by 25 inputs per time units.

So, the latency even though it increases the through put of a pipeline system reduces. So, one thing that we have assumed is, we have assume that the flip flop itself has 0 clock

queued delay and the internal delays are 0, we assume that this 50, 20 and 25 are enough to satisfy the setup time constraints on these flip flop, we have assumed that the flip flop is essentially a 0 delay structure.

(Refer Slide Time: 25:05)



So, when you do this we can construct what is called a pipeline diagram, what you have in the x axis is time units or the inputs. So, this is input i, input i plus 1, input i plus 2 input i plus 3 and so on. So, the X is the input, so that is what you to see here, so  $X_i$  is input given at time period i, the F register and G register can only get F of  $X_i$  and G of  $X_i$  at time i plus 1 and this will be placed as input to the combinational logic H. So, H will take at the end of time period i plus 1, H will take F of  $X_i$  and G of  $X_i$  and it will produce H of  $X_i$ .

So, if you notice this pink diagonal here, this  $X_i$  is input given a time i, a time i plus 1 F of  $X_i$  and G of  $X_i$  will be ready and a time i plus 2 H of  $X_i$  or P of  $X_i$  is ready. However, at i plus 1 I can place the inputs  $X_{i+1}$  they will get ready at i plus 2 for H to take in and H will produce the output at i plus 3. So, this yellow one is the second set of input and the violet one corresponds to the third set of inputs, you can think of this as, this is the unwashed clothes, this is the washer and this is the dryer.

So, the washer and dryer example we did not have F and G there was only one washer there was one F and one H. Whereas, here we have F and G there are two combinational blocks. So, the results related to a particular input moves diagonally across the pipeline diagram and the process inputs progress through one pipeline stage at a time. So, from

the external input it goes to stage 1 and then goes to stage 2, it gets out of stage 2 at time step 2 away and this process is this goes through one step at a time. So, this kind of pipeline diagram is useful to imagine what is happening in the circuit.

So, this brings me to the end of this module, in this module what we saw is the notion of pipelining. So, the term pipelining is used to specify this fact that we do not have to waste our time, waiting for some other stage to finish the work, when the next stage is doing some work you take the previous stage and give it some more work and this keeps happening in lock step fashion, eventually what you have is the latency of a single input to go through the pipeline stages may increase the amount of time may increase. However, the number of inputs that you process per time unit improves, the through put improves in a pipeline system. We will see the ramifications of this latency and through put in the next video. So, this brings me to the end of this module.