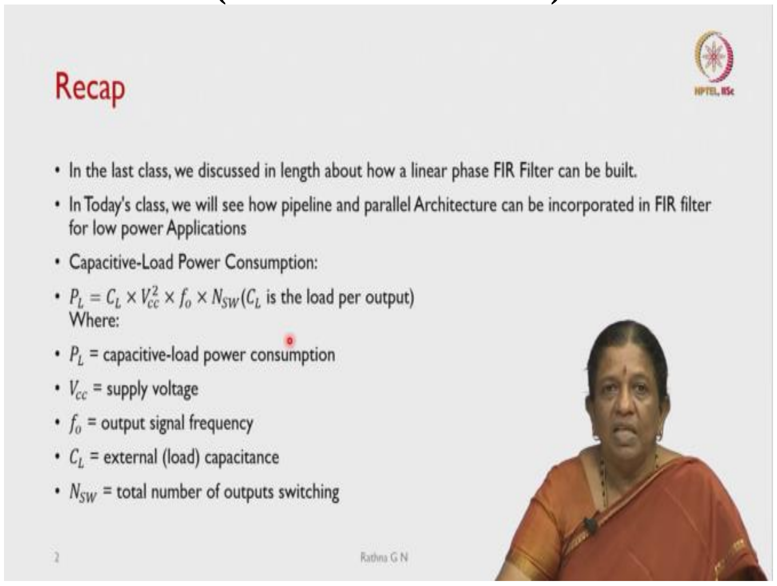
Real-Time Digital Signal Processing

Prof. Rathna G N
Department of Electrical Engineering
Indian Institute of Science - Bengaluru

Lecture - 09
Pipelining and Parallel Processing for Low Power Applications

(Refer Slide Time: 00:28)



Recap

- In the last class, we discussed in length about how a linear phase FIR Filter can be built.
- In Today's class, we will see how pipeline and parallel Architecture can be incorporated in FIR filter for low power Applications
- Capacitive-Load Power Consumption:
- $P_L = C_L \times V_{cc}^2 \times f_o \times N_{SW}$ (C_L is the load per output)
  Where:
- P_L = capacitive-load power consumption
- V_{cc} = supply voltage
- f_o = output signal frequency
- C_L = external (load) capacitance
- N_{SW} = total number of outputs switching

2 Rathna G N

Today we will discuss about pipelining and parallel processing for low power applications. The recap of the last classes we discussed about linear phase FIR filter in length, and then how to build it, that is what, what we have seen in the last class. In today's class, how we can use the architecture that is pipeline and parallel architecture to build a fire filter for low power applications.

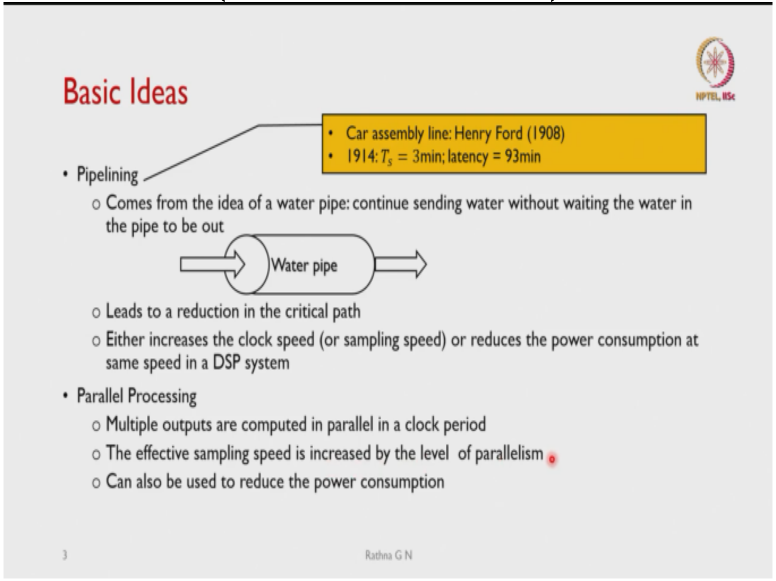
So, when I talk about the low power in this case, we will be considering the capacitive load power consumption, which is given by $P_L = C_L \times V_{cc}^2 \times f_o \times N_{SW}$. So, we say that the power is proportional to, we say the frequency and square of voltage what we call it. So, we can if we reduce the voltage, we know that the power consumption is going to come drastically, but for speed, we have to increase our frequency.

So, we have seen that we can use the architecture for speeding up our computation using pipeline and parallel. So, the frequency is going to be increased. But in this case, we will see that how we can use the thing by reducing our power consumption that use by reducing our operating voltage. So, here we will give it P_L as a capacitive load power consumption and V_{cc}

is our supply voltage and f_o is our output signal frequency and C_L we call it as external load capacitance basically and N_{SW} will be total number of outputs switching.

So, if we assume 1 bit what we are transmitting, so, it becomes $n = 1$ so, switching load will be 1 in this case. So, if we have more number of bits that also will be contributing to our power consumption.

(Refer Slide Time: 02:23)



Basic Ideas

- Pipelining
  - Comes from the idea of a water pipe: continue sending water without waiting the water in the pipe to be out
  - Leads to a reduction in the critical path
  - Either increases the clock speed (or sampling speed) or reduces the power consumption at same speed in a DSP system
- Parallel Processing
  - Multiple outputs are computed in parallel in a clock period
  - The effective sampling speed is increased by the level of parallelism
  - Can also be used to reduce the power consumption

- Car assembly line: Henry Ford (1908)
- 1914: T_s = 3min; latency = 93min

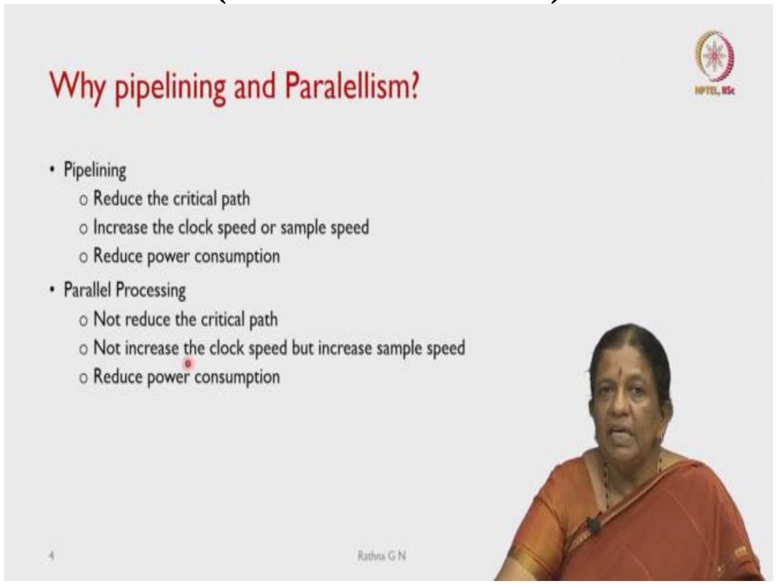
3 Rathna G N

So, we will see what is the basic idea of pipelining, which I have already discussed. That is we took either a car assembly in this case example is given Henry Ford in 1908. In 1914, what they were taking time is 3 minutes, what was the every car was coming out. So, latency in that case, we say was 93 minutes, basically, the maximum time what was taking, although they were able to achieve 3 minutes, but as you can see, 93 minutes was the latency between 2 cars because one of them was taking more time.

So, what we call that example, we took the water pipe in the last class. So, we will be seeing that the length of the water pipe is less than our latency is going to be less, but if it is long, then we know that it is going to have a maximum latency we call it as critical path. So, further increase the clock speed or sampling speed to or reduce the power consumption to for the same speed in a DSP system.

So, for the parallel processing will have multiple outputs are computed in parallel in a clock period. The effective sampling speed is increased by the level of parallelism if we want to increase the speed or we can reduce the power consumption.

(Refer Slide Time: 04:01)



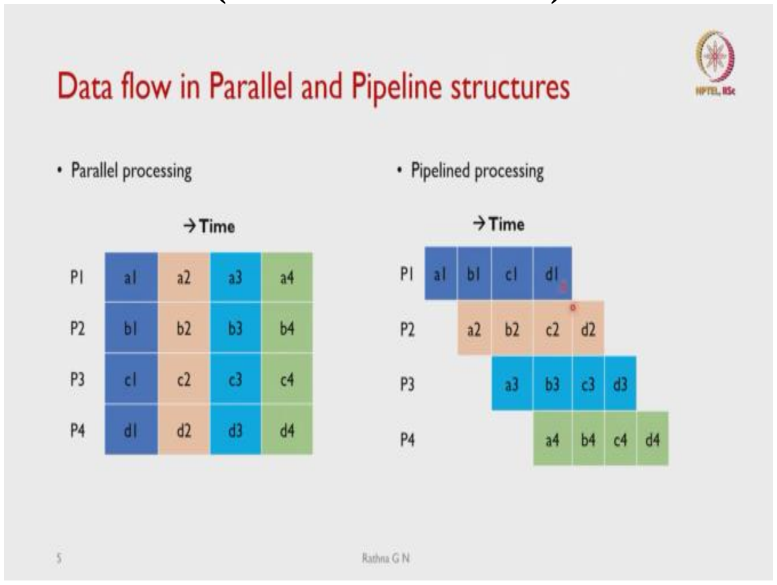
Why pipelining and Paralellism?

- Pipelining
  - Reduce the critical path
  - Increase the clock speed or sample speed
  - Reduce power consumption
- Parallel Processing
  - Not reduce the critical path
  - Not increase the clock speed but increase sample speed
  - Reduce power consumption

4 Rathna G N

So, why do we need pipelining and parallelism? One is to reduce our critical path that is the longest path delay and increase the clock speed or sample speed what we are putting it or achieve a reduced power consumption. Same thing with parallel processing, not reduce the critical path here, not increase the clock speed but increase sample speed and then we can use it for reduce power consumption.

(Refer Slide Time: 04:34)



Data flow in Parallel and Pipeline structures

- Parallel processing

→Time

P1	a1	a2	a3	a4
P2	b1	b2	b3	b4
P3	c1	c2	c3	c4
P4	d1	d2	d3	d4

- Pipelined processing

→Time

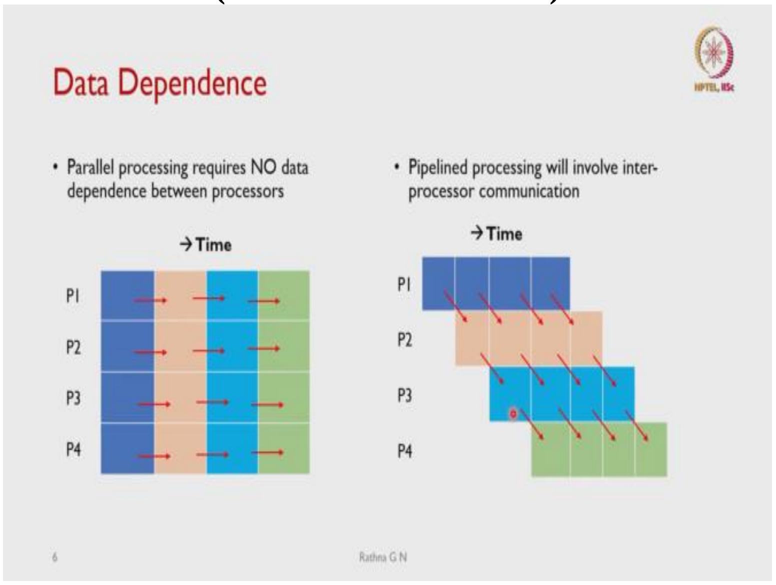
P1	a1	b1	c1	d1			
P2		a2	b2	c2	d2		
P3			a3	b3	c3	d3	
P4				a4	b4	c4	d4

5 Rathna G N

So, we will see the data flow in parallel and pipeline structure how it is going to be implemented. So, we say that we have 4 processors here, P1, P2, P3, P4, if there is no data dependencies, we know that all the 4 processors will be working on different data sets. So, if we have the same number of processors in pipeline processing. So, we say that each processor has to wait for the previous data to be finished, computation to be finished and then it should be coming into the next stage.

So, each processor you will be seeing that the latency of this P4 processor is, it has to wait for 4 clock cycles till it can operate. So, once all the data have been filled then they can work in parallel. So, that is what the latency of the pipeline whereas in the parallel processing all of them, there should not be any dependency between the data in this case.

(Refer Slide Time: 05:44)



Data Dependence

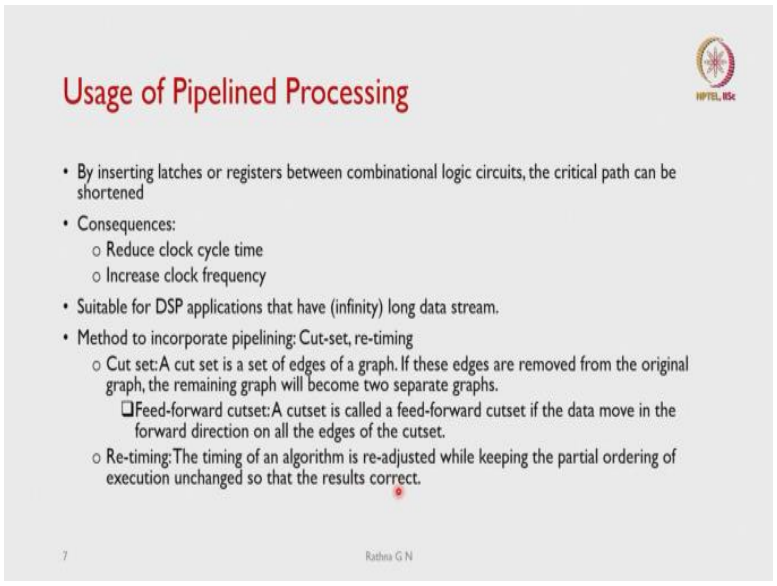
- Parallel processing requires NO data dependence between processors
- Pipelined processing will involve inter-processor communication

6 Rathna G N

So, when we say that how it is going to have this thing data dependence, we say that in parallel processing requires no data dependence between the processors then only they can run in parallel. So, as you are seeing it, the data flow in P1 processor is going to go in horizontal way, same thing with respect to the rest of the processor. Whereas, in the case of pipeline, we said there is a data dependency between the 2 processors.

So, P2 is waiting the data from P1 the same way P3 will wait from P2 and then P4 will be waiting from P3. So, this is how the data will be if they are dependent then the flow is going to be in this fashion.

(Refer Slide Time: 06:29)



Usage of Pipelined Processing

- By inserting latches or registers between combinational logic circuits, the critical path can be shortened
- Consequences:
  - Reduce clock cycle time
  - Increase clock frequency
- Suitable for DSP applications that have (infinity) long data stream.
- Method to incorporate pipelining: Cut-set, re-timing
  - Cut set: A cut set is a set of edges of a graph. If these edges are removed from the original graph, the remaining graph will become two separate graphs.
    - ❑Feed-forward cutset: A cutset is called a feed-forward cutset if the data move in the forward direction on all the edges of the cutset.
  - Re-timing: The timing of an algorithm is re-adjusted while keeping the partial ordering of execution unchanged so that the results correct.

7 Rathna G N

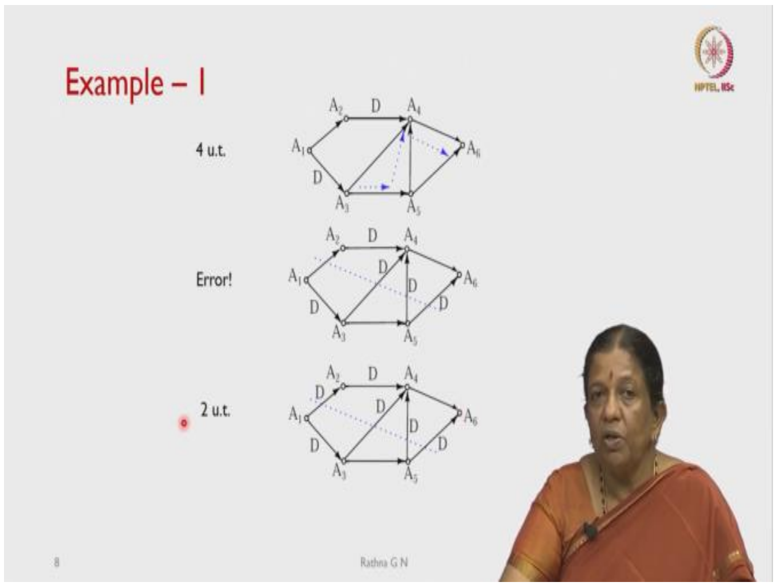
So, coming to the usage of pipeline processing, how we can do the thing, we can do by inserting latches or registers between combinational logic circuits. So, that way we will be reducing the critical path or it can be shortened. So, the consequence of introducing this will be reducing the clock cycle time and increase the clock frequency. So, it is suitable for DSP applications that have we call, it as infinity long data stream.

So, we are assuming that input data is coming continuously, so, we have to operate on that. So, method to incorporate this pipelining how we are going to do it, we will see it in a while using the cut set, and then do the retiming of the circuits. So, we will define the cut set, we call that as a set of edges of a graph, if these edges are removed from the original graph, the remaining graph will become 2 separate graphs.

That is there are no dependence between the 2 graphs that have to become independent, then only we call that as a cut set, usually, we apply feed forward cut set in our filter. So, in this case, we call feed forward because data move in the forward direction on all the edges of the cut set. So there will not be any feed backward data flow. So only we consider the if there is a feed forward data, then we call, that cut set as a feed forward cut set.

The other one is re-timing so how we are going to see the re-timing algorithm is re-adjusted while keeping the partial ordering of execution unchanged so that the results are correct. So, we will see that how we will be doing that.

(Refer Slide Time: 08:39)



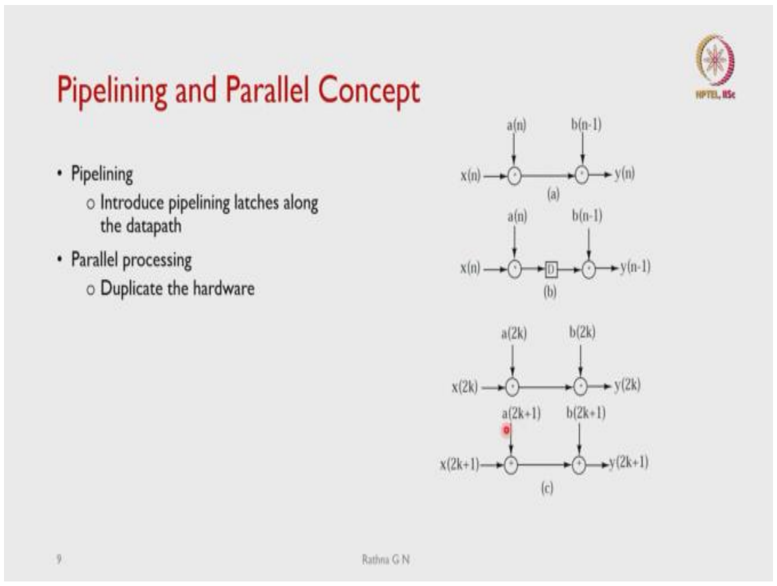
So as an example, we will take that this is the data flow graph what we have it so data is moving from A_1, A_2, A_4 and then A_6 is the last thing or it can flow in this direction. So we say that D means there is a delay in this data path. So, what will be the longest delay that is what, what we see, we have to see in this graph, or we call it as the critical path, since there is a delay.

So we will assume that it will be going into this register, it can be a register or so that it is going to hold the data and then in the next clock cycle it will be going into this point. So here we will be seeing that from A_3 the data has to flow to A_5 and then to A_4 and then to A_6, this is the longest path in this. So, which we call it as it is going to take 4 clock cycles or 4 unit time to get the data from one node to the last node in this case.

So we said that we will put a cut said basically, so I can cut the graph into 2 pieces in this way. And then we said, we are going to introduce delay units, wherever we have put the cut set. So, or a shift register, one of the thing, what we will be using it, as we can see that we have cut this into 2 parts this way, but we have forgotten to put a delay unit here. So, that is how it says this is an error, basically.

So how if we cut this properly, and then put all the delay units in the cut set, then what happens to our critical path what will see it, so we say that from A_3 to A_4, I have a delay path, and then from A_4 to A_6, it is direct. And then what we have is from the delay unit, if I take that the data is moving, it has to move to A_4 and then A_6. So that way it takes 2 clock units to reach the maximum delay in this circuit or in the data flow graph.

(Refer Slide Time: 11:12)



Pipelining and Parallel Concept

- Pipelining
  - Introduce pipelining latches along the datapath
- Parallel processing
  - Duplicate the hardware

Rathna G N

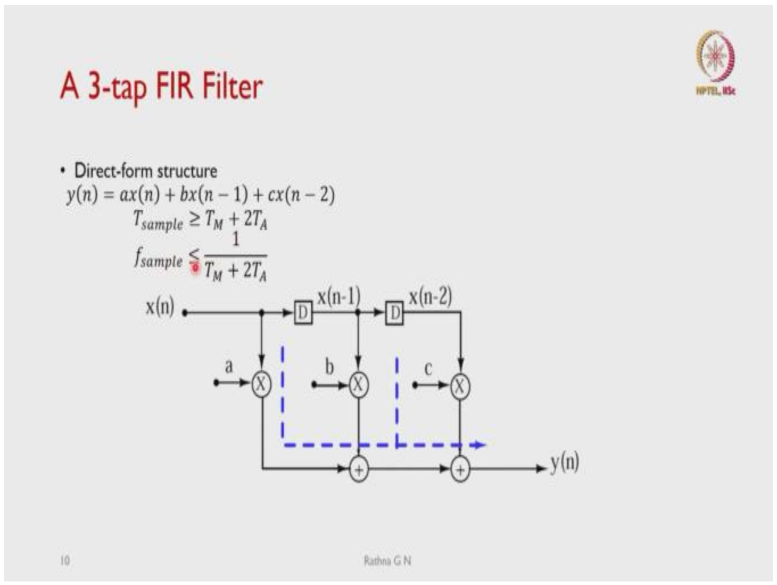
So, what is it? So some of the concept, what we will see how we will be representing our pipelining and parallel concept is represented. In the pipelining, we introduced pipelining latches along the data path. So this is the original circuits, $x(n)$ is the input. And then this is one of the $a(n)$ is the coefficient $b(n-1)$ can be the other coefficient. So $y(n)$ is going to have it as $a(n) \cdot x(n) + b(n-1) \cdot x(n)$ basically.

So, this is my $y(n)$, when I say that the critical path is going in this case is 2 adders. So, it has to pass through them to get my $y(n)$. So I want in a single clock cycle my $y(n)$ has to come out, then what I do is I cut this, line into 2 parts and introduced a delay line. So I am putting increasing the hardware. So $x(n)$is this, it will be added in 1 clock cycle. And this is going to be latched here.

And in the next clock cycle, I will be adding with whatever data coming from here, and the output in the current state is going to be $y(n-1)$. And then in the next clock cycle, I will be getting the $y(n)$. So I will have compared to previous structure, I will be getting 1 delay output, initial output after that it is going to come continuously when we talk about the parallel processing, so we will be duplicating the hardware.

So, here we have duplicated 2 units basically, then what happens to the input? It has to be $x(2k)$ to the one of the stage for the other one $x(2k+1)$. So here, even the coefficients as you will be seeing it, it becomes $a(2k)$ and $b(2k)$ k. So, k will be varying from 0 to n - 1 in this case only 2 what we have it, so, the other input to the second structure is $a(2k+1)$ and then $b(2k+1)$.

(Refer Slide Time: 13:37)



A 3-tap FIR Filter

- Direct-form structure

$y(n) = ax(n) + bx(n-1) + cx(n-2)$

$T_{sample} \geq T_M + 2T_A$

$f_{sample} \leq \frac{1}{T_M + 2T_A}$

10 Rathna G N

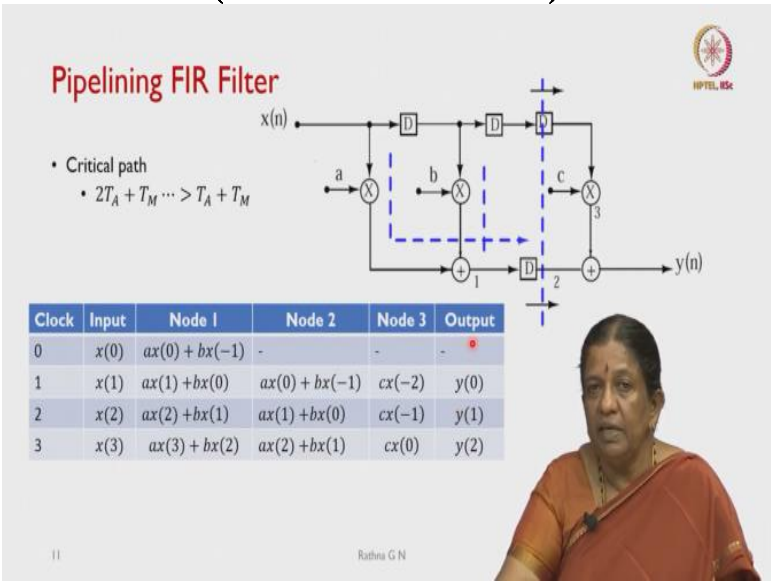
So, coming to the thing, when we use the FIR filter will consider as a 3 tap FIR filter. So, if we use the direct form structure, as you are seeing that I am talking about the structure, even the structure is going to matter, when we are drawing or designing our circuit. So, here we have used the direct form in that case, what is the equation for my FIR filter $y(n) = ax(n) + bx(n-1) + cx(n-2)$.

So, we say that the sample period T sample should be greater than or equal to T_M that is the multiplier plus the longest path in this is going to be from here to here, which is going to incorporate 2 adders. So, my sample period has to be $T_M + 2T_A$ when I talk about the sampling frequency should be less than or equal to $\frac{1}{T_M+2T_A}$. So, depending on the time it is going to take to multiply and then add will decide my sampling frequency for this 3 tap FIR filter.

So, the coefficients what we have assumed is a, b and then c. So, how will I get my $y(n)$? What is shown here? $x(n)$ is the input, and these are the delay lines $x(n-1)$, and then $x(n-2)$. So as the equation suggests here, $y(n) = ax(n) + bx(n-1) + cx(n-2)$. So, that is what my 3 tap of FIR filter, when I consider the sampling frequency, what should be for this structure? So if I talk about the sampling period T_{sample} should be greater than or equal to the time taken to do multiplication.

And because the longest or the critical path delay, what we call it, here is 2 adders, what it is included 1 multiplier and then 2 adder, so it becomes T_M + 2 times time for addition, so the sampling rate is going to be less than or equal to $\frac{1}{T_M+2T_A}$.

(Refer Slide Time: 16:03)



Pipelining FIR Filter

- Critical path
  - $2T_A + T_M \cdots > T_A + T_M$

Clock	Input	Node I	Node 2	Node 3	Output
0	$x(0)$	$ax(0) + bx(-1)$	-	-	-
1	$x(1)$	$ax(1) + bx(0)$	$ax(0) + bx(-1)$	$cx(-2)$	$y(0)$
2	$x(2)$	$ax(2) + bx(1)$	$ax(1) + bx(0)$	$cx(-1)$	$y(1)$
3	$x(3)$	$ax(3) + bx(2)$	$ax(2) + bx(1)$	$cx(0)$	$y(2)$

11 Rathna G N

So we will see how we will avoid this. One of the thing is, as we said, we can have a cut set in the because as you are seeing in the structure, all of them in the feed forward path, the flow what it is going, so I can have a cut set. So we said that it is taking 2 clock cycles to add, whether I can reduce it to 1 clock cycle, then what I am going to do is I am going to put a cut set here as you are seeing in the blue line in the forward path, that is what, what it is shown here, then I have to introduce 2 delays in the path, because I have cut 2 lines here.

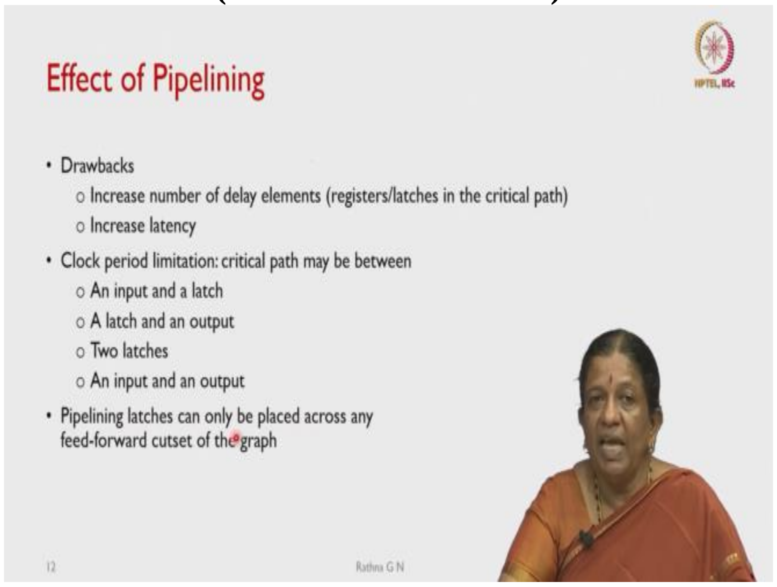
So there will be addition of delays in this case, now, you will be seeing the longest path in this case is going to be, so 1 multiplication and 1 addition. So, we will come back what we are going to define what is my longest path. And from here to here also it is going to take 1 clock cycle and from this delay to my output is going to have 1 clock cycle. So, we can make this structure as first unit and the second unit what we have it so, what happens to the output that is what, what it says is critical path from $2T_A + T_M$ which is changed to $T_A + T_M$.

So, the clock will see that what will be the output at 0 if the input to this is $x(0)$, then node 1 what we call it what is going to happen in this clock cycle is it will be multiplying $ax(0) + bx(-1)$ in this case. So, in the next clock cycle, so, you will be seeing that $x(1)$ will be the input to this unit and then the thing is going to be node 1 it is going to compute $ax(1) + bx(0)$.

So, in node 2, because this has moved to the next clock cycle, so it will be $ax(0)$ the previous one $+bx(-1)$ and in the node 3 what we call it is here, which is going to be $cx(-2)$ and this will be my $y(0)$ and same thing you will be seeing that in the third clock cycle. So, the all the units become full, then $y(2)$ will be my output which is completely correct. So, compared to the previous one, you can draw this sketch for the one which did not have the cut set.

So output will be start coming from here $y(0)$, $y(1)$, $y(2)$, $y(3)$, but in the this thing pipeline case, in this case since we have put only 1 cut set the delay is going to be 1 clock cycle. So, we will not be getting anything here it is irrelevant or it is garbage what we will have it so, output starts coming from here.

(Refer Slide Time: 19:12)



Effect of Pipelining

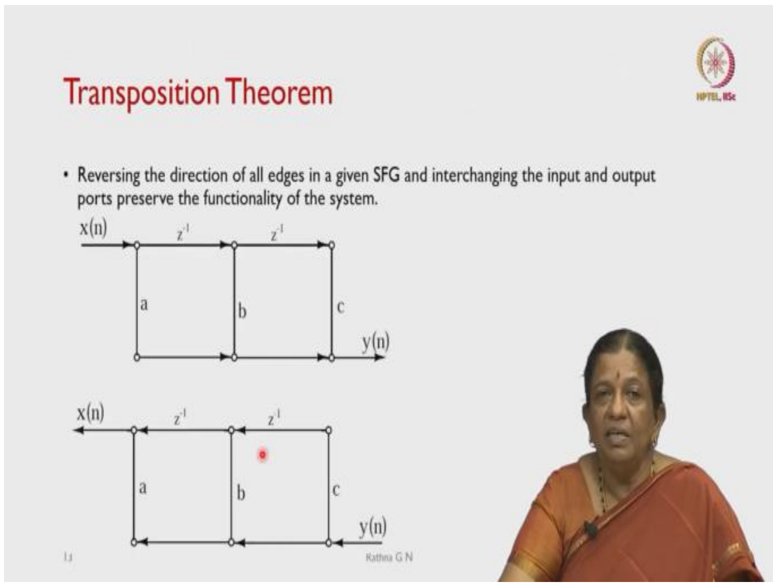
- Drawbacks
  - Increase number of delay elements (registers/latches in the critical path)
  - Increase latency
- Clock period limitation: critical path may be between
  - An input and a latch
  - A latch and an output
  - Two latches
  - An input and an output
- Pipelining latches can only be placed across any feed-forward cutset of the graph

12 Rathna G N

We will see effect of pipelining what are the drawbacks we will be increasing the delay elements, number of delay elements, what we call it as registers or latches in the critical path, this is a hardware addition what we have to provide, this causes increase in latency, then what we say is a clock period limitation we will see it, so the critical path may be between an input and a large we call it a latch and an output or between 2 latches, an input and then output.

So this we call it as the critical path. So, this is how we measure how many multipliers or accumulators or how many hardware is in between these 2 things that will be the maximum critical path what I have to consider. So, we say pipelining latches can only be placed across any feed forward cut set of the graph, this we will be specifying every time, and then we had to go through the thing.

(Refer Slide Time: 20:22)



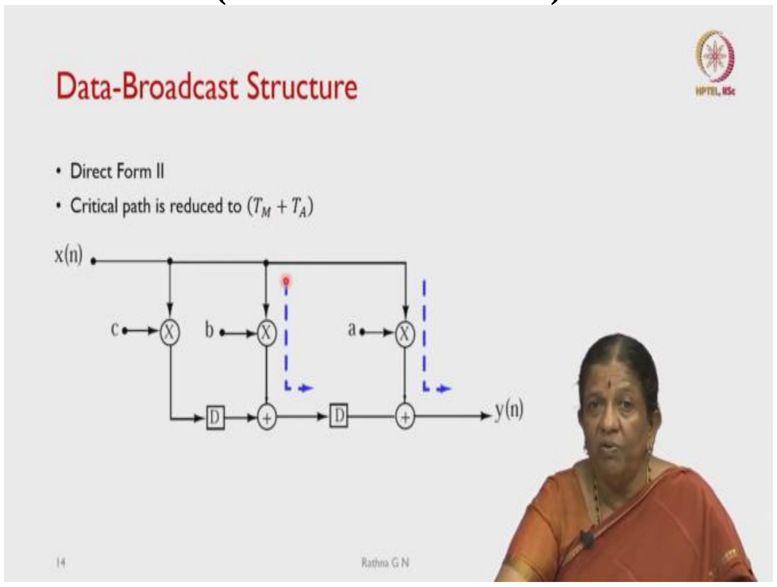
Transposition Theorem

- Reversing the direction of all edges in a given SFG and interchanging the input and output ports preserve the functionality of the system.

Next one is, as I was talking about the structure, can we have a transposition theorem incorporated in the signal flow graph, that is, what we are going to do is reversing the direction of all edges in a SFG signal flow graph and interchanging the input and then output ports, preserve the functionality of the system. So, we say that $x(n)$ is the input $y(n)$ is the output this is in the feed forward what we have it.

So this delay can be represented as a D or in the zee domain, we will call it as z - 1 is going to be that delay. So, each node will be representing my this as an adder and this is a multiplier. So, now, what we are going to do is my output I will be changing it as an input and then input as an output and all the direction are going to be reversed basically, that is what the transposition theorem says. So, as you will see the thing $y(n)$ and then we will be keeping a, b, c as it is so, this has got changed.

(Refer Slide Time: 21:37)



Data-Broadcast Structure

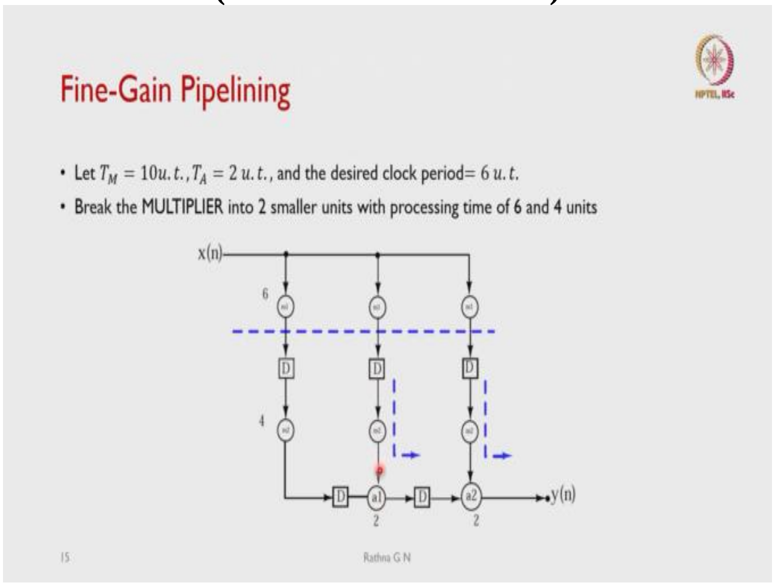
- Direct Form II
- Critical path is reduced to $(T_M + T_A)$

Now, what is the thing is going to happen, so, we will incorporate in our data flow graph here from the signal flow graph, we call that as a direct form 2 (form II) structure we call that as a data broadcast structure. And in this case, you will be seeing that the critical path without adding any delay elements, it has become $T_M + T_A$ we will see it in a while how it is going to be. So, to have the understanding correctly, what we have done from the previous thing, we have reverse the implementation of this graph actually.

So that we are not comfortable going from right to left. So, we will put it back as a forward path itself. So, you will be seeing that $x(n)$ is going to be broadcast to all the 3 multipliers here it is c, here it is b and then a and then we have a delay element already present in the original one, this is the original. Now, you see the thing the critical path in this case is going to be 1 multiplier and 1 adder this is from input to output and then from input to latch is also 1 unit.

And then here also it is going to be input 2 latch is here only have a 1 multiplier delay. So, the maximum that is critical path is 1 multiplier and 1 adder. So, you can see that the structure if you choose it properly, I need not have to have any cut set or any pipelining introduced in some of the graphs that one has to consider.

(Refer Slide Time: 23:24)



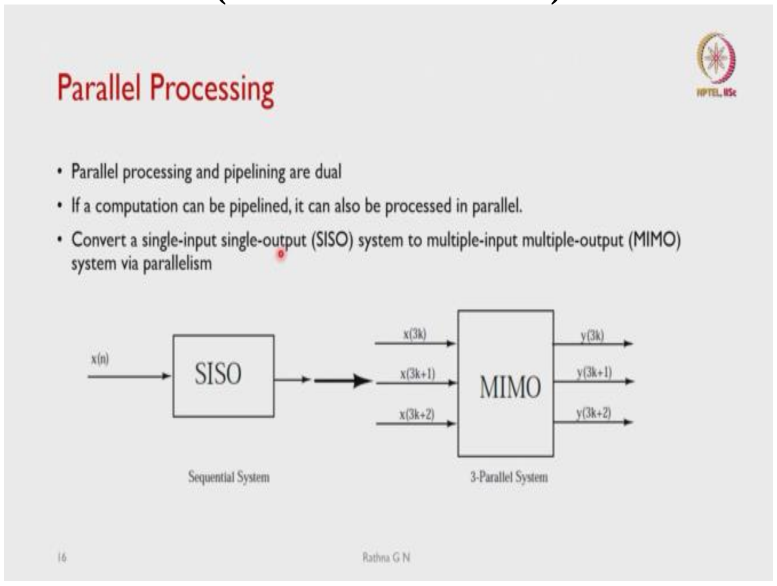
Fine-Gain Pipelining

- Let $T_M = 10u.t., T_A = 2\,u.t.$, and the desired clock period= $6\,u.t.$
- Break the MULTIPLIER into 2 smaller units with processing time of 6 and 4 units

Now, we will say that all of us know that multiplier is going to take longer time compared to my adder. So, as an example, we call this as fine gain pipelining. So, let $T_M = 10$ unit time and adder takes 2 unit time and the desired clock's period what we want is 6 unit of time. So, how we can achieve because multiplier is going to 10 units + 2 units will be my longest path in the broadcast section, if I am computing it, but my clock period has to be 6 units, then what we do is we can bifurcate our multiplier into 2 smaller units with processing time of 6 and 4 units. So, you will be seeing that 1 multiplier m1 is going to take 6 units and then m2 multiplier will take 4 units. So, we are doing a cut set here in the forward path and then put a delay element in all the 3 legs of my filter.

And then we will be seeing that from anywhere to anywhere that is input to my this latch is 6 units in all the cases and from here you will be seeing this latch to this latch is only 4 units and the whatever shown in the blue will be the my critical paths basically from this latch to this latch it has 1 multiplier and 1 adder. So, which will be taking 6 units of time and from here to here also. So, this we call it as fine gained parallelism or sorry pipelining.

(Refer Slide Time: 25:07)



Parallel Processing

- Parallel processing and pipelining are dual
- If a computation can be pipelined, it can also be processed in parallel.
- Convert a single-input single-output (SISO) system to multiple-input multiple-output (MIMO) system via parallelism

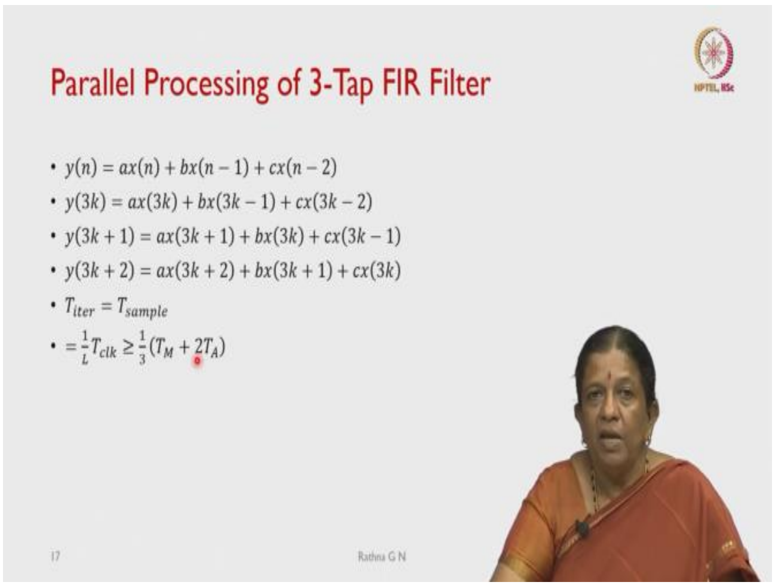
x(n)
SISO
Sequential System
x(3k)
x(3k+1)
x(3k+2)
MIMO
y(3k)
y(3k+1)
y(3k+2)
3-Parallel System

16 Rathna G N

So, the next one will see that parallel processing how we are going to work on, we say they are dual basically, and if a computation can be pipelined, it can also be processed in parallel that is what, what we say. So, how we are going to do this, we say convert a single input single output. So, SISO is my single input single output that is what $x(n)$ to this, system into your multiple input multiple output, we call it as MIMO system via parallelism.

So, you will have a seen here draw input is $x(3k)$, $x(3k+1)$ and $x(3k+2)$, you will be wondering what is it initially we took example parallel for 2 this thing parallel systems, here we are assumed 3 parallel systems. So, you will be seeing that your $x(n)$ will be changed as $x(3k)$ in this case, and y will also be $y(3k)$, $y(3k+1)$ and $y(3k+2)$ so, this is 3 parallel system.

(Refer Slide Time: 26:19)



Parallel Processing of 3-Tap FIR Filter

- $y(n) = ax(n) + bx(n-1) + cx(n-2)$
- $y(3k) = ax(3k) + bx(3k-1) + cx(3k-2)$
- $y(3k+1) = ax(3k+1) + bx(3k) + cx(3k-1)$
- $y(3k+2) = ax(3k+2) + bx(3k+1) + cx(3k)$
- $T_{iter} = T_{sample}$
- $= \frac{1}{L} T_{clk} \geq \frac{1}{3}(T_M + 2T_A)$

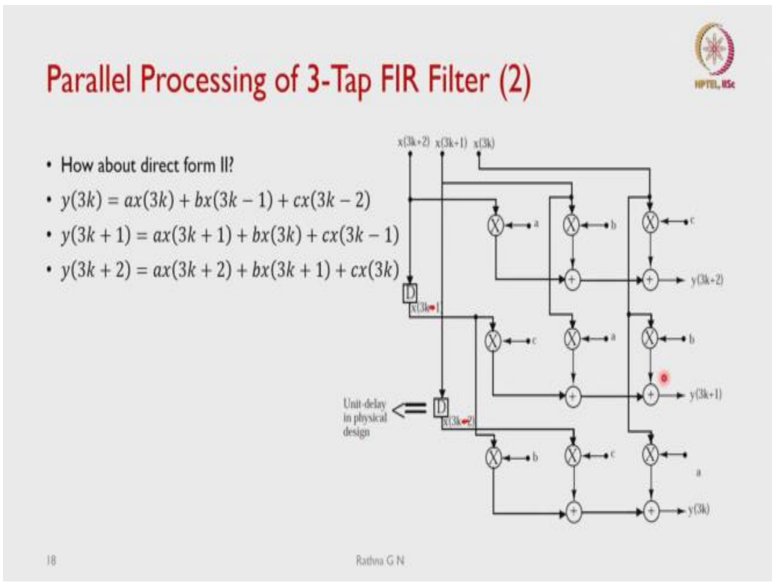
17 Rathna G N

Then, what happens to my input and output representation? We have original what we have is $y(n) = ax(n) + bx(n-1) + cx(n-2)$, when I pass this in 3 parallel section, what will be my output is $y(3k)$, $y(3k+1)$ and $y(3k+2)$, how the inputs have to be you will be seeing it. So, I will be replacing n with 3k, $ax(3k) + bx(3k-1) + cx(3k-2)$. So, in the next case, 3k + 1, what we will be adding with respect to n is 3k + 1.

So, if you substitute that $3k + 1$ and then $3k + 1 - 1$ will become 3k, $x(3k)$ in this case, and then in the next case $cx(3k-1)$, so, you will be seeing even $y(3k+2)$ will be represented in this way, then what happens, we call time taken for iteration is equivalent to T_{sample} what we call it here in this case, it becomes $\frac{1}{L} T_{clk}$. So, which is greater than or equal to because number of parallel section I have assumed is in 3 here.

So, which becomes $\frac{1}{3}(T_M + 2T_A)$, because I have not put any pipelining here, so, adder delay is going to be 2 units in this case.

(Refer Slide Time: 27:57)



Parallel Processing of 3-Tap FIR Filter (2)

- How about direct form II?
- $y(3k) = ax(3k) + bx(3k-1) + cx(3k-2)$
- $y(3k+1) = ax(3k+1) + bx(3k) + cx(3k-1)$
- $y(3k+2) = ax(3k+2) + bx(3k+1) + cx(3k)$

18 Rathna G N

So, what happens to 3 tap FIR filter, however, I will be representing it in the direct form 2, as you will be seeing it. So, if we use the direct form 2 what is the thing, so, this is the output what I want, $y(3k)$, $y(3k+1)$ and $y(3k+2)$, and we will consider $y(3k+1)$rest of it, you can work it out. So, we will see that how we are going to get $ax(3k+1)$ what I wanted the first output, so you will be seeing that $x(3k+1)$ is coming here.

And then which is fed to my this multiplier, which is $ax(3k+1)$. So which goes into the adder. So what the next one, what I want is $bx(3k)$. So you will be seeing b is here, and then $x(3k)$ is the thing which is coming here, which is going to be multiplied with your $bx(3k)$ so which gets added. So what is the last one what I want it, $cx(3k-1)$ what I want in this case, so how we are going to get the thing.

So what we do is I can take $x(3k+2)$, because when I introduce a delay line in this, because this is the 3 parallel structure what we have taken the thing, so delay is going to be 3 units. So this becomes $x(3k+2)-3$. So which is with 1 delay, so it is going to be $x(3k-1)$ this input, which is going to be multiplied with c. So all of them get added and then we will be getting $y(3k+1)$. So, you can check the rest of them whether you are going to get this equation correctly with the structure or not. So thank you for hearing this lecture.