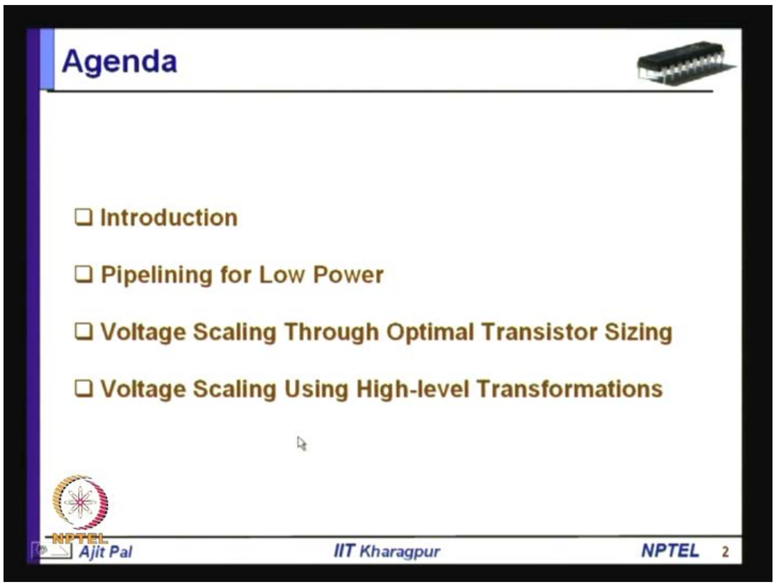
Low Power VLSI Circuits and Systems

Prof. Ajit Pal
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture No. # 23
Supply Voltage Scaling – II

Hello and welcome, today's lecture on supply voltage scaling, this is the second lecture on this topic. In the last lecture, I have introduced the basic concept of supply voltage scaling, then discussed some approaches based on static voltage scaling.

(Refer Slide Time: 00:38)



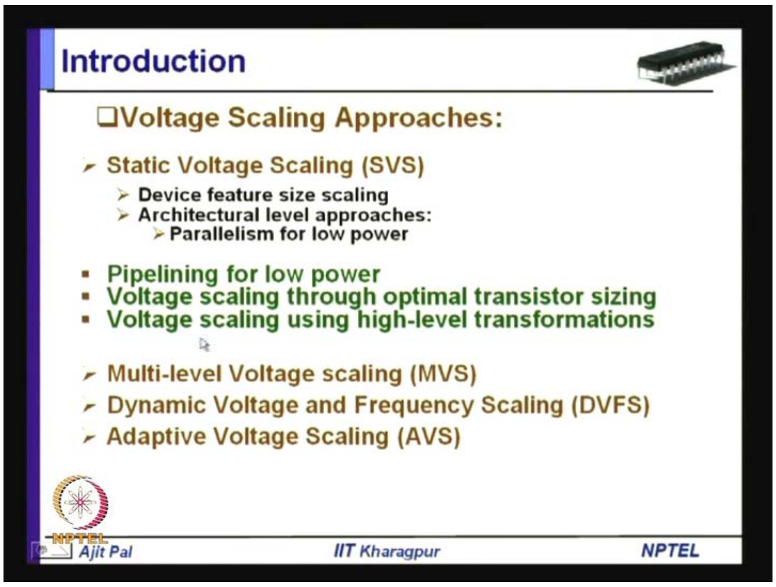
Agenda

- ❑ Introduction
- ❑ Pipelining for Low Power
- ❑ Voltage Scaling Through Optimal Transistor Sizing
- ❑ Voltage Scaling Using High-level Transformations

Ajit Pal IIT Kharagpur NPTEL 2

And here is the agenda of today's lecture after brief introduction, we shall discuss about pipelining for low power, and then we shall discuss voltage scaling through optimal transistor sizing. And finally, we shall discuss about voltage scaling using high level transformations.

(Refer Slide Time: 01:00)



Introduction

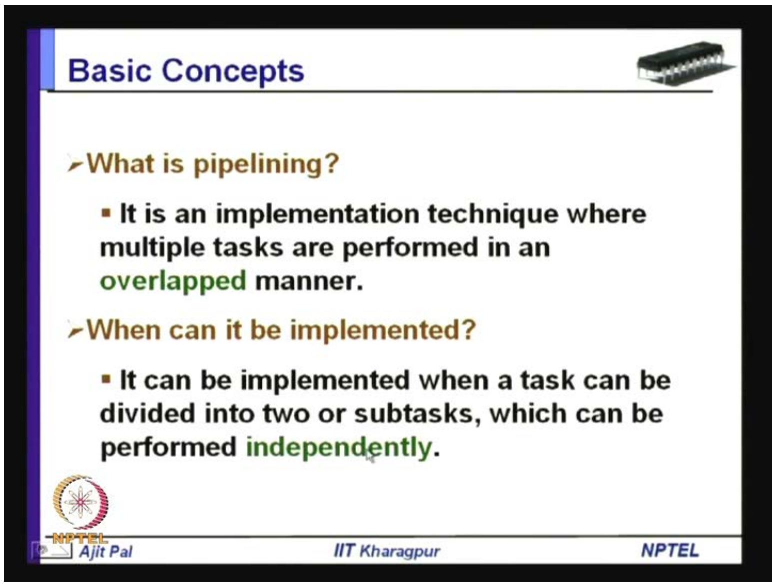
❑Voltage Scaling Approaches:

- Static Voltage Scaling (SVS)
  - Device feature size scaling
  - Architectural level approaches:
    - Parallelism for low power
- Pipelining for low power
- Voltage scaling through optimal transistor sizing
- Voltage scaling using high-level transformations
- Multi-level Voltage scaling (MVS)
- Dynamic Voltage and Frequency Scaling (DVFS)
- Adaptive Voltage Scaling (AVS)

Ajit Pal IIT Kharagpur NPTEL

As I mentioned we have discussed voltage scaling approaches and as you know there are four different categories. We can classify we can categorize them in four different types static voltage scaling multi-level voltage scaling dynamic voltage and frequency scaling and adaptive voltage scaling and in the last lecture. We have discussed about device features size scaling and also one of the important approaches based on architecture level synthesis. That is parallelism for low power and today as I mentioned we shall be discussing pipelining for low power and then voltage scaling through optimal transistor sizing and voltage scaling using high level transformations coming to pipelining for low power.

(Refer Slide Time: 01:50)



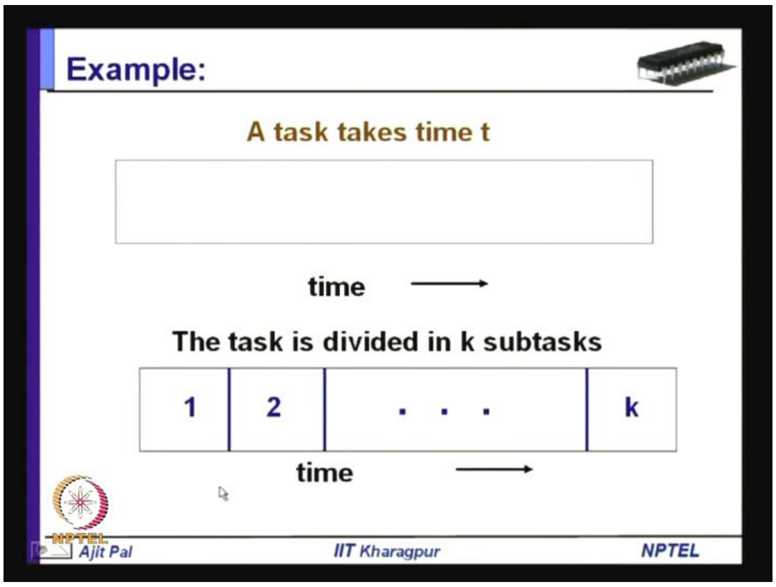
Basic Concepts

- What is pipelining?
  - It is an implementation technique where multiple tasks are performed in an overlapped manner.
- When can it be implemented?
  - It can be implemented when a task can be divided into two or subtasks, which can be performed independently.

Ajit Pal IIT Kharagpur NPTEL

First I shall introduce the basic concept of pipelining what is pipelining pipe lining is an implementation technique. Where multiple asks are performed in an overlap manner. So, this the basic idea or how the pipelining is defined. It is an implementation technique note at note at this particular there implementation technique and also another important there overlap manner question is, when can it be implemented. It can be implemented when ask can be divided into that or more subtasks sub asks which can be performed independently. So, pipelining can be implemented only when a particular ask can be divided into a number of subtasks which can be performed independently. So, this is the fundamental requirement for pipelining.

(Refer Slide Time: 02:46)



Example:

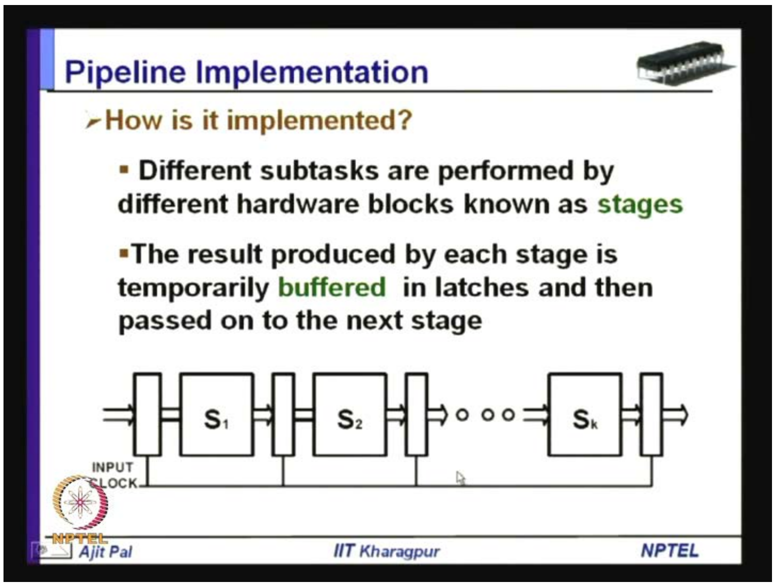
A task takes time t

The task is divided in k subtasks

Ajit Pal IIT Kharagpur NPTEL

For example there we have got a ask which can be performed in time let us assume and if ask can be divided into k subtasks say we have divided into one that up to k of equal size, then this can be pipelined and if they can be performed in an independent manner then you can implement pipelining how.

(Refer Slide Time: 03:11)



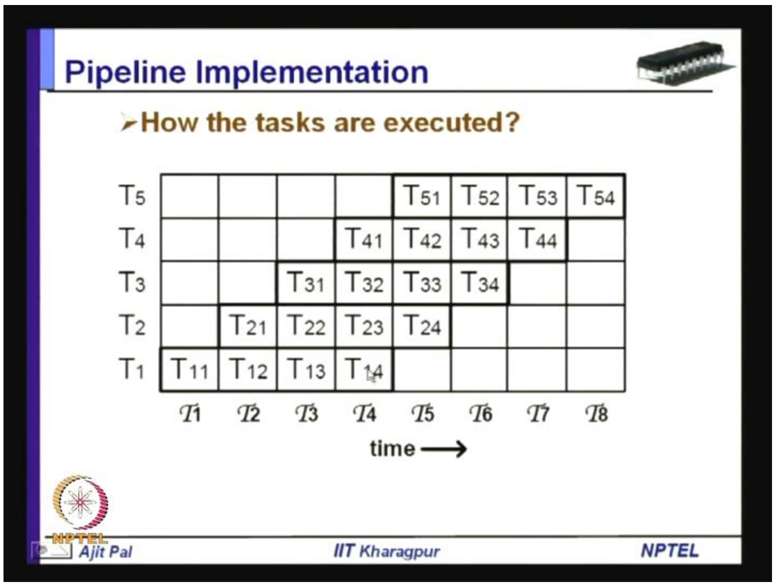
Pipeline Implementation

- How is it implemented?
  - Different subtasks are performed by different hardware blocks known as stages
  - The result produced by each stage is temporarily buffered in latches and then passed on to the next stage

Ajit Pal IIT Kharagpur NPTEL

Let us see different subtasks are performed by different hardware blocks known as stages. So, a stage s one is performing subtask one s. That is performing subtask that in this way stage k is performing subtask k and you can see you are applying clock and you have got latches in between where intermediate results are stored and the clock transfers. The results from stage to other in a synchronous manner so this is how it is implemented and the execution is done in this manner.

(Refer Slide Time: 03:50)



Pipeline Implementation

- How the tasks are executed?

T_5					T_{51}	T_{52}	T_{53}	T_{54}
T_4				T_{41}	T_{42}	T_{43}	T_{44}	
T_3			T_{31}	T_{32}	T_{33}	T_{34}		
T_2		T_{21}	T_{22}	T_{23}	T_{24}			
T_1	T_{11}	T_{12}	T_{13}	T_{14}				
	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7	τ_8

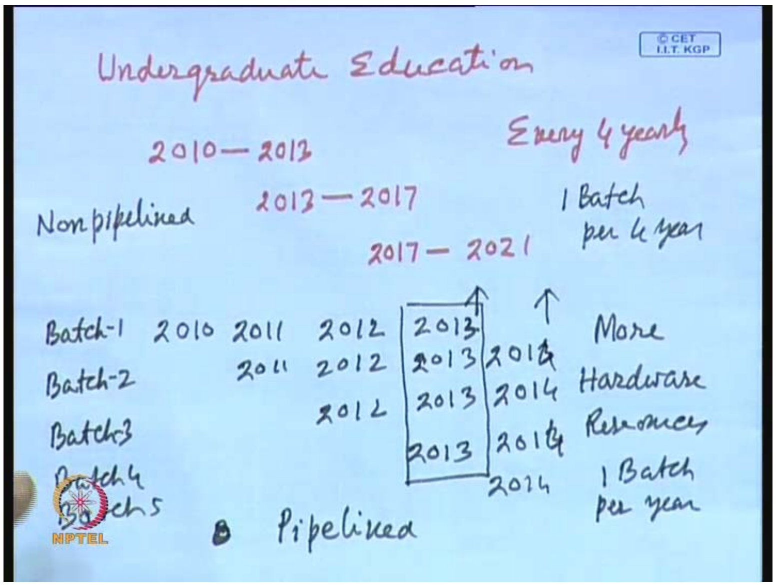
time ⟶

Ajit Pal IIT Kharagpur NPTEL

For example: if we consider five asks ask one ask that ask three and ask four and ask five you can see ask one is divided into four subtasks, which are performed in time au one au that au three and four and then, before the completion of the ask one you can see after the subtask one is completed of ask one in time in slot au, that we are starting the sub task. I mean subtask one of ask that. So, that one is the subtask one of ask to and. So, these that are getting executed in overlapped manner, because we have got different stages, which are able to perform this subtask that means when we reach this time slot au four we find that all the four asks are getting executed in overlapped manner.

However, the the forth part; that means, the subtask four of ask one is getting executed in au four subtask. I mean subtask three of ask, that is getting executed in the same time slot sub subtask that of for ask three is getting executed in the same time slot and subtask one of ask four is getting executed in the same way. So, this is how it is getting executed in an overlapped I believe before we proceed let me illustrate with an with an example very simple example for example, say.

(Refer Slide Time: 05:36)



Undergraduate Education

2010—2013

Every 4 years

Nonpipelined 2013—2017

1 Batch per 4 year

2017—2021

Batch-1 2010 2011 2012 2013

Batch-2 2011 2012 2013 2014

Batch3 2012 2013 2014

Batch4 2013 2014

Batch5 2014

More Hardware Resources

1 Batch per year

Pipelined

Let us consider say undergraduate education undergraduate education b tech or b e. whatever, you call suppose as you know the under graduation takes four years. So, let us assume that we are admitting students in the year that thousand ten. How long they will take they will continue up to four years. So, eleven twelve thirteen fourteen that thousand thirteen, so one batch will pass out from I mean they will get-they will get admitted in that thousand ten and they will get pass out in that thousand thirteen. Let us assume that we do not have enough resources. So, until the students pass out cannot we cannot admit a new batch so; that means, another batch will be admitted in the year that thousand thirteen as they pass out and then fourteen fifteen sixteen seventeen that thousand seventeen another batch will pass out in this way another batch will be admitted in that thousand seventeen and they will continue ill eighteen nineteen twenty and twenty one. So, we can, so three batches are passing out starting with that thousand ten and three batches are passing out in that thousand twenty one, so every four yearly.

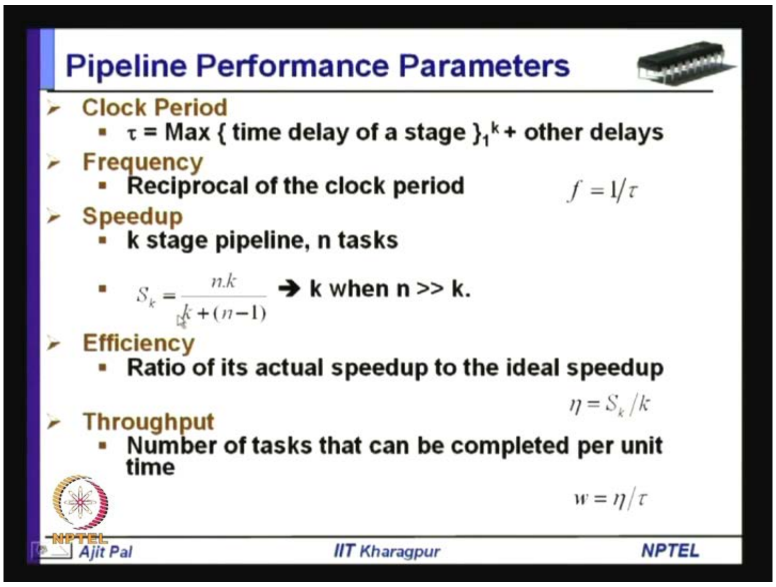
That means as we admit one batch we cannot take another batch and this is, how it is going on. So, in this way we can say, this is non pipelined education now let us say in that thousand ten. We admit first batch and as they go to second year that thousand eleven. We admit another batch say batch one then batch that we admit in that thousand eleven. So, first batch is in that thousand eleven second batch in the first year first batch is in the second year of second year and the second batch is in first year, so then batch three then that thousand thirteen and that thousand.

Sorry; this is the batch of one twelve thirteen and. So, there in the year of the as, we go to year that thousand fourteen. We find that batch one will pass out batch that will go to second first fourth year batch three will go to third year and batch four will go to second year and batch one batch five will go to first year. So, you can see if we consider the year that thousand thirteen.

We can see batch one is in the fourth year batch that in that is in the third year batch three in that in the second year batch four in the first year though this is nothing but pipeline implementation of the education. This is a very simple example of pipelining and this kind of example. We will find in our day to day life in many places in automobile industry and many other places you will find this how it is being done. So, only requirement for this pipeline implementation this is this, we can say as pipeline implementation; obviously, you will require more hardware resources, so more hardware more hardware resources because you will require.

Classroom more laboratories more hostels and. So, on to accommodate students of four batches but the advantage is every year one batch will pass out that thousand three that thousand four that thousand five that means after the after the initial entry. All the students will I mean every year one batch will pass out; that means, there throughput is one batch per year earlier it was one batch per four year and there one batch per year. So, throughput is four times by using pipelining. So, this is a very simple example of pipelining and exactly what I have shown there in the same way the education is taking place.

(Refer Slide Time: 11:04)



Pipeline Performance Parameters

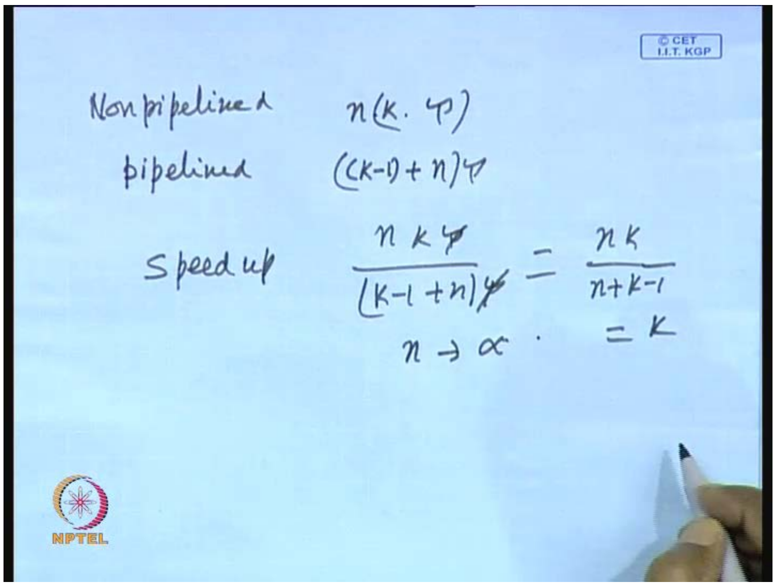
- Clock Period
  - τ = Max { time delay of a stage $\}_1^k$ + other delays
- Frequency
  - Reciprocal of the clock period $f = 1/\tau$
- Speedup
  - k stage pipeline, n tasks
  - $S_k = \frac{n.k}{k+(n-1)}$ → k when n >> k.
- Efficiency
  - Ratio of its actual speedup to the ideal speedup $\eta = S_k/k$
- Throughput
  - Number of tasks that can be completed per unit time $w = \eta/\tau$

Ajit Pal IIT Kharagpur NPTEL

Now, in the context of pipelining, there are some performance parameter we use number one is clock period clock period. You know as we have seen when we go for implementation. We have to use some latches we have seen there and we have to apply clock. So, what is the clock frequency clock frequency is decided by the maximum time delay of the stage plus other delays due to latches and other things.

So, it will be slightly more than the a time required to perform a subtask in a particular stage and frequency will be decided by this maximum delay that is one by au is the clock frequency and you can see you can measure the speedup for a case stage pipeline say whenever you are performing asks. You will require how much time will require in a non-pipelined manner in a non-pipeline manner you will require n k au on the other hand you will require k plus n minus one into au.

(Refer Slide Time: 12:24)



Non pipelined $n(k \cdot \tau)$

pipelined $((k-1)+n)\tau$

Speed up $\frac{nk\cancel{\tau}}{(k-1+n)\cancel{\tau}} = \frac{nk}{n+k-1}$

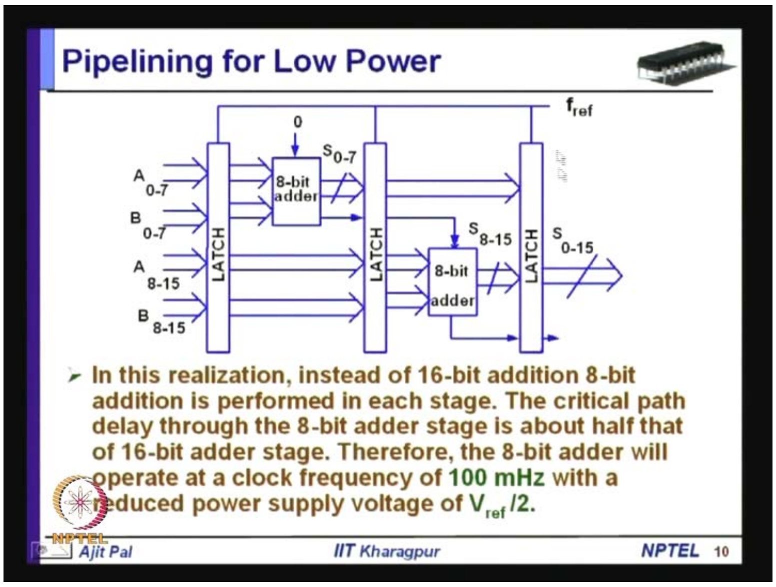
$n \to \infty \quad = k$

I mean non-pipeline manner the time required is non-pipelined time required is to execute n asks is n into k into au. You know this is the time required for each ask I mean k into au and for n ask it will require n k au on the other hand for pipeline implementation. We have seen that after the k minus one stage we will require only k minus one plus n. So, first batch will come out there and n f one will come out there. So, this into au will be time required for non-pipelined. So, speedup that you achieve by using pipeline is n k au by k minus one plus n into au will cancel out. So, this is equal to n k by n plus k minus one now when n is very large n is very large say it is infinity that means you have to execute a very large number of ask.

In such a case what will happen you can ignore, you can ignore this will -this will become n k by n this one can be ignored. So, this will become equal to this will approach. So, for a large value of n what will happen this will become equal to k n into k and n plus k that means this part will become equal that means speedup will be a factor of will be equal to k when n is large you can you can make it will become equal. (Refer Slide Time: 11:04). So, we find that speedup can be that means the ideal speedup can be k that k is the number of stages and efficiency is essentially. The ratio of the actual speedup to, the ideal speedup and ideal value will be equal to one, because s k as we have seen ideal value is k. So, s k by k will be equal to one on the other hand throughput that is the number of ask that can be completed per unit time will be equal to v equal to eta by au and that will be equal to again. It will be equal to f that means f one by au that is equal to epsilon. It will be equal to one and. So, it will be one by au.

So, number of ask that can be completed per unit time will be because per cycle one ask can be will be complicated in completed in pipeline system. So, this is the fundamental principle of pipelining and pipelining is widely used not only in our day to day life in many situations, but it is used in computers. So, inside computer the pipelining in all modern processors pipelining is used noise particularly to implement execution of instructions but primarily what is being done in modern processor pipelining is used to improve performance to have speedup to have higher throughput and we have seen how the throughput is increased. How this speedup is achieved but what we shall discuss is how pipelining can be used to achieve low power not pipelining for performance but pipelining for low power.

(Refer Slide Time: 15:57)



Pipelining for Low Power

- In this realization, instead of 16-bit addition 8-bit addition is performed in each stage. The critical path delay through the 8-bit adder stage is about half that of 16-bit adder stage. Therefore, the 8-bit adder will operate at a clock frequency of 100 mHz with a reduced power supply voltage of $V_{ref}/2$.

Ajit Pal IIT Kharagpur NPTEL 10

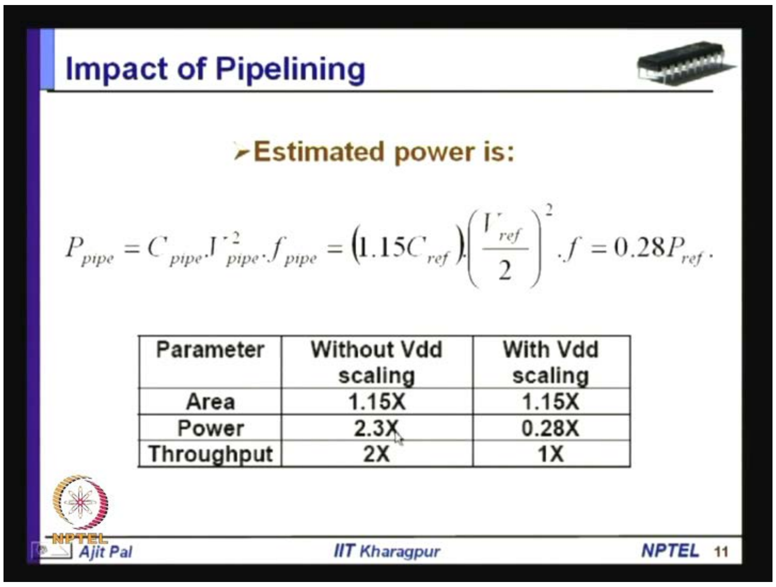
Let us see how it can be done let us go back our original example; that means, performing sixteen bit addition and there is the pipeline implementation of six sixteen bit addition. We have done it by using that stages. So, we have got that stages each stage is of eight bit that means the sixteen bit addition has been divided into that eight bit additions. So, there the sixteen bit addition is asked which has been divided into that subtasks each of eight bit addition. So, you will require that eight bit adders in each stage, and you can see in this stage in a particular stage.

We have got one eight bit order adder and another eight bit adder in the next stage and intermediate results are stored in these latches. You can see that this value that s zero to seven will be stored there and this carry will pass on to the next stage and that will be added in the next clock period and you can see their finally, we s zero fifteen sum you are getting and carry you are getting from this particular second stage.

So, in this implementation as you can see that we are performing eight bit addition instead of sixteen bit additions. So, therefore the time required will be half that of sixteen bit addition that means the delay of the sixteen bit adder is double that of the delay of the eight bit adder. So, what we can do if we want toper form the computation in the during the same period; that means, the time required to compute sixteen bit addition if we want to you perform eight bit addition in the same time then what you can do you can reduce supply voltage it to half.

So, it will perform eight bit addition in the same time and that is what is being done here. So, we apply a supply voltage v reference by that to this adder as well as to this adder of course, the clock frequency will be reference because the through put we are not we are not compromising on throughput it will keep on producing result at every clock cycle of f reference. So, using this implementation what kind of performance we achieve we can see the or what is the power dissipation. So, power dissipation p pipe will be equal to c pipe that is the capacitance of this circuit this part of the circuit and there you can see this is an eight bit adder this is an eight bit adder, but these together makes sixteen bit adder only additional things are latches. So, the increase in capacitance will not be as high as the parallel implementation. We have seen in parallel implementation using that parallel adders the increase in capacitance was that point that c references.

(Refer Slide Time: 19:10)



Impact of Pipelining

- Estimated power is:

$$P_{pipe} = C_{pipe}.V^2_{pipe}.f_{pipe} = (1.15C_{ref}).\left(\frac{V_{ref}}{2}\right)^2.f = 0.28P_{ref}.$$

Parameter	Without Vdd scaling	With Vdd scaling
Area	1.15X	1.15X
Power	2.3X	0.28X
Throughput	2X	1X

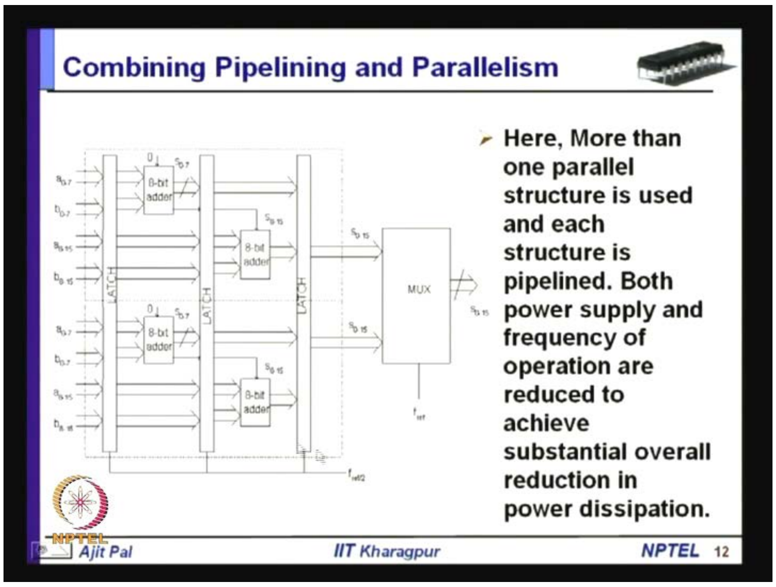
Ajit Pal IIT Kharagpur NPTEL 11

But there it will be much less and that is the reason why it is one point one five c reference fifteen percent increase in capacitance because of the latches; however, you will be applying a supply voltage of v reference by that. So, the power dissipation is one point one five c reference into v reference by that square and; however, this will be f reference this reference is missing, but this is the f reference

So, this value if you compute you will get zero point e that eight p references that means the power dissipation is twenty eight percent of the that original sixteen bit addition. So, we find that we are able to reduce the power dissipation by voltage scaling; however, we are maintaining the throughput same that means this is essentially pipelining for low (refer time: 20:00) power and not pipelining for performance. So, there we have compared these that situations whenever we do pipelining for performance in such a situation there is an increase in area by fifteen percent there is a increase in power dissipation by I mean that point three x that means that point three times of the original adder but we get a throughput of that x that is that times. So, instead of that there if we are if we are doing pipelining for low power we find that there is an increase in area that means we are trading area for lower power. So, lower power is point that eight x and throughput is remaining same.

So, this is how you can achieve pipelining for lower power without compromising performance but you will require a (refer time: 21:00) little more area as we have seen.

(Refer Slide Time: 21:05)



Combining Pipelining and Parallelism

- Here, More than one parallel structure is used and each structure is pipelined. Both power supply and frequency of operation are reduced to achieve substantial overall reduction in power dissipation.

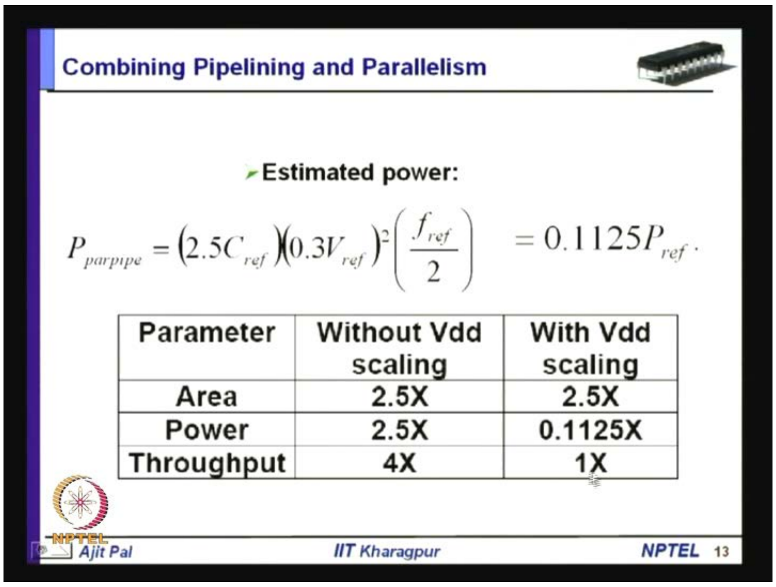
Ajit Pal IIT Kharagpur NPTEL 12

Now, we can combine pipelining with parallelism. So, earlier we have discussed parallel implementation today we have discussed pipeline implementation we can combine parallelism with pipelining that means we can have that parallel implementation and each of them is pipelined.

So, there you can see we have got a sixteen bit adder implemented in pipelined way another sixteen bit adder pipeline sixteen bit adder. So, we have got that sixteen bit adders each of them is pipelined and in such a case; obviously, you will require that eight bit adders in first the stage and that eight bit adders in the second stage and you will require a multiplexor as you require in case of parallel implementation to take the output there and (refer time: 22:00) these adders these adders will be operating at a lower voltage much lower voltage even less than v reference by that and the clock frequency there is f reference by that here. The clock frequency will be f reference by that, because there you are generating at half the rate and the output at the output of the multiplexor you will be getting at the full rate. So, throughput there will be same as the original circuit.

However this part of the circuit will be will be operating at a lower voltage, and with a lesser speed, because clock the rate at which data will be input at will be f reference by that.

(Refer Slide Time: 22:43)



Combining Pipelining and Parallelism

- Estimated power:

$$P_{parpipe} = (2.5C_{ref})(0.3V_{ref})^2\left(\frac{f_{ref}}{2}\right) \quad = 0.1125P_{ref}.$$

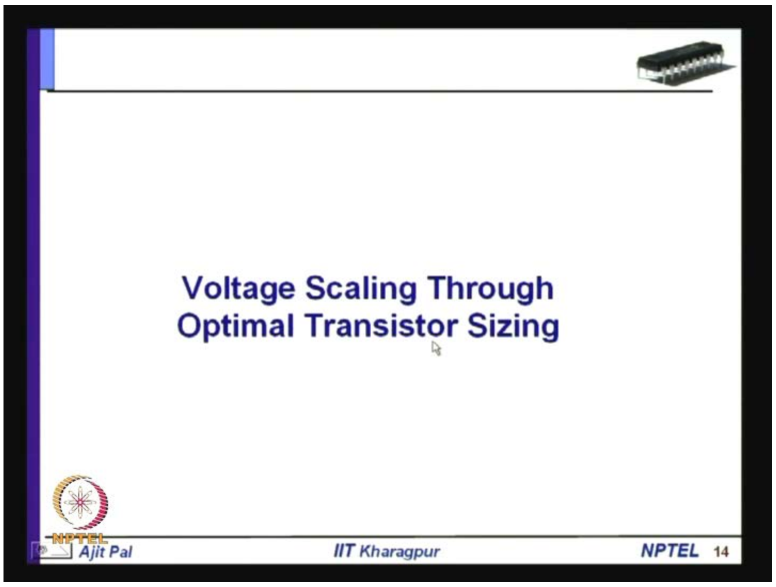
Parameter	Without Vdd scaling	With Vdd scaling
Area	2.5X	2.5X
Power	2.5X	0.1125X
Throughput	4X	1X

Ajit Pal IIT Kharagpur NPTEL 13

Now, let us see what is the estimated power. So, this parallel so parpipe-parpipe that means parallel pipeline implementation the capacitance is increasing by factor of the point five supply voltage is thirty percent of (refer time: 23:00) the original circuit and frequency is f reference by the. So, if we multiply we get point one the five references. So, we find that the power dissipation is roughly equal to eleven percent of the original circuit that means if we go for parallel and if we combine parallelism with pipelining. We get a get still I mean larger reduction in the power dissipation and of course, in this case you are maintaining the same performance. So, a there you can see the comparison in this. This particular column gives you whenever we do not do voltage scaling. So, without voltage scaling that means we get higher throughput at the cost of larger area and larger power dissipation.

So, with an increase in area of the point five x (refer time: 24:00) and power dissipation of the point five x we get a throughput of four x, because in this particular case it is possible to have four times throughput compared to our original sixteen bit addition but instead of that if we go for parallelism and combine parallelism and pipelining for low power and we do voltage scaling in such a case we can see although the area increases by the point five times the power dissipation is point one the five the five times of the original circuit keeping the throughput same as the original one. So, this is how we can we can we can use parallelism, we can use pipelining we can combine parallelism with pipelining to achieve lower power.

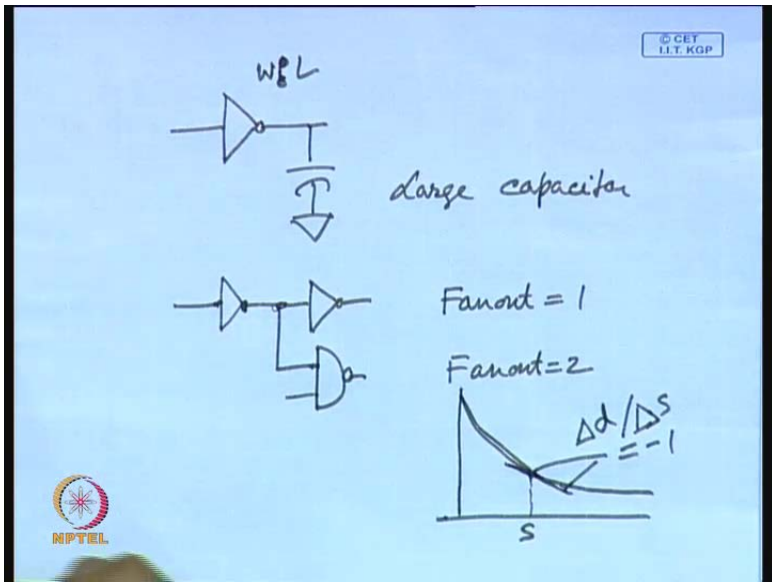
(Refer Slide Time: 24:55)


Voltage Scaling Through Optimal Transistor Sizing

Ajit Pal IIT Kharagpur NPTEL 14

Now, we change here. And shift to another topic voltage scaling through optimal transistor sizing (refer time: 25:00) what do you really mean by transistor sizing. We know that a transistor can be realized with different length and width. So, whenever we increase the width what happen the resistance, decreases that means, if the width of a transistor is increased the resistance reduces.

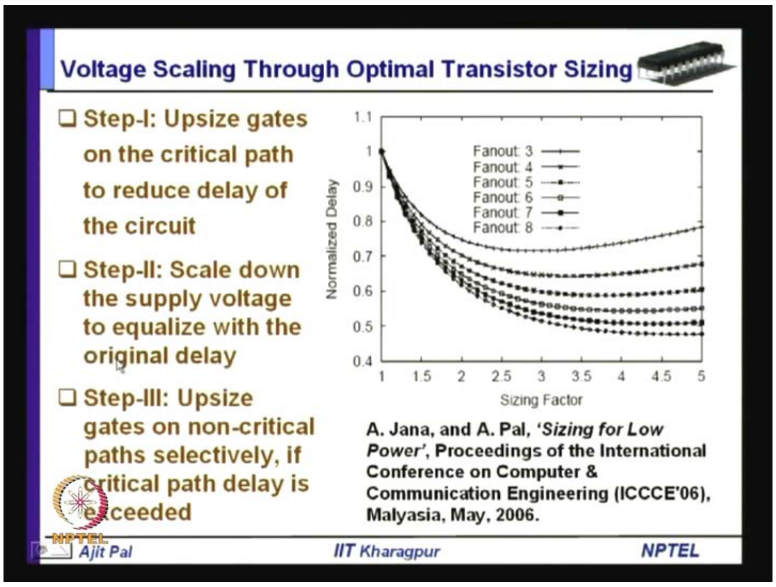
(Refer Slide Time: 25:39)



So, one very common technique to reduce delay is transistor sizing; that means suppose this particular stage one inverter stage is driving a large capacitor earlier we have discussed in detail how we can this drive a large capacitor.

We have seen that the you can use inverters (refer time: 26:00) with a higher w by l that means w by l ratio is increased, and instead of single stage multiple stages can be used to achieve smaller delay. So, sizing transistor sizing is very common in reducing the delay now instead of reducing delay if we are interested in lowering power dissipation can we use transistor sizing.

(Refer Slide Time: 26:37)



Voltage Scaling Through Optimal Transistor Sizing

- ❑ Step-I: Upsize gates on the critical path to reduce delay of the circuit
- ❑ Step-II: Scale down the supply voltage to equalize with the original delay
- ❑ Step-III: Upsize gates on non-critical paths selectively, if critical path delay is exceeded

A. Jana, and A. Pal, *'Sizing for Low Power'*, Proceedings of the International Conference on Computer & Communication Engineering (ICCCE'06), Malyasia, May, 2006.

Ajit Pal IIT Kharagpur NPTEL

So, I shall explain how it can be done. So, voltage scaling through optimal transistor sizing, so there we have done some experimentation you can see as the fan out increases. Fan out means you can see say suppose you have got an inverter. If it (refer time: 27:00) is driving only one inverter the fan out is one fan out is equal to one if it is driving say another gate then say fan out is that. So, as the fan out increases as you know the output capacitance increases, because the gate capacitance of the next stage is coming and also parasitic capacitances are there for interconnection.

So, that means as the number of stages as the as the fan out is increasing the delay increases, now the you can do this sizing of this transistor such that this delay is reduced. Question is what is-what is the optimal sizing how much I mean the width has to be increased by what ratio. So, that is that you can find out from this (refer time: 28:00) particular curve you can see. This this particular curve corresponds to fan out is equal to three then when fan out is equal to four. You can see delay is larger but reduction is more as we go for further increase in fan out and the lower curve corresponds to fan out four we can see the we can the delay is reduced more whenever the fan out is more that means the sizing has more impact is more effective whenever the fan out is more on the other hand fan out is less the reduction in delay is less as you can see. If we consider say sizing factor the point five. So, in the first case when the fans out is equal to three-there is a small reduction. So, there it is it is reduced from one to it is normalized delay. So, it is reduced to point (refer time: 29:00) eight five on the other hand if we consider this curve. So, with fan out is equal to eight you can see it is reduced to point five that means the reduction is more. So, what can be done let us consider a one particular curve.

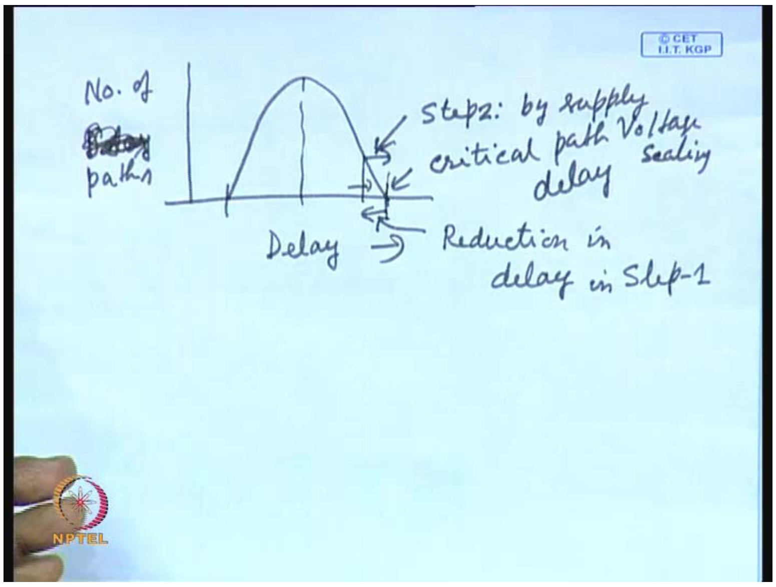
Say for a particular fan out this is the curve now, what can be done; we can draw a agent that means this particular line that corresponds to delta delay by delta sizing factor size which is equal to minus one we find out this particular point. Where it is minus one and this is the optimal sizing factor. So, this sizing factor is considered to be one, because up to this point it reduces and reduces more softly and after that reduction is not much as you can see it (refer time: 30:00) more or less saturates.

So, the delta d by delta s points we find out depending on the fan out and the transistors are the transistors can be sized by that ratio. So, then we follow these steps. So, we have a complex circuit where there are some critical path and non-critical path soupsize gets on the critical part to reduce delay of the circuit; obviously, as we increase that do the sizing for on gates of the critical part there will be reduction in the delay and then second step is scale down the supply voltage to equalize the original delay that means whatever the reduction in delay occurs there.

Now, we are we are scaling down the supply voltage such that whatever delay we achieved we use that (refer time: 31:00) slack as a consequence after this time the delay will be equal to the original circuit; however, the supply voltage will be less and as a consequence. There will be significant reduction in the power dissipation; however, whenever you reduce the supply voltage many non-critical parts will become critical. So, again you have to selectively identify transistors and the non-critical path and then introduce sizing in a in a selective way. So, that the delay is not more than the critical path.

This is how by in by using these three steps we can do voltage scaling by optimal sizing of the transistors. So, what essentially we are doing normally in a complex circuit.

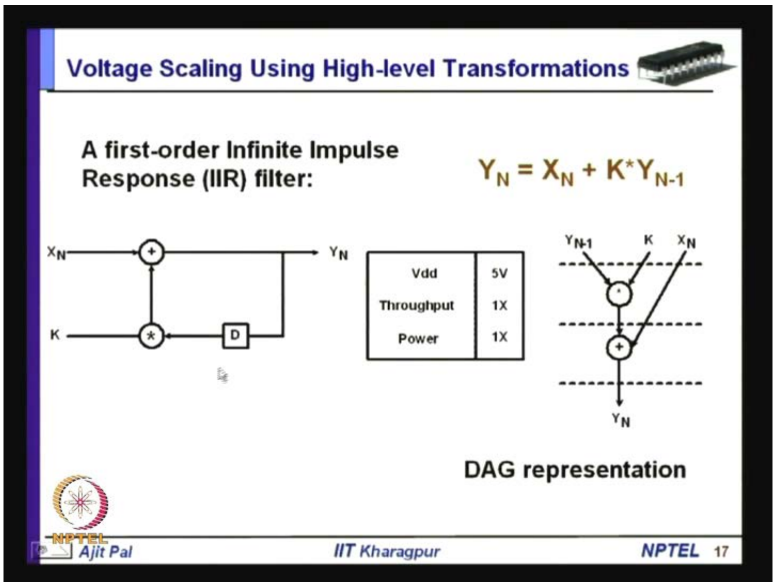
(Refer Slide Time: 31:53)



If we plot the delay distribution sorry delay distribution (refer time: 32:00) will be there this is the delay distribution delay is on this line and there number of number of paths in the circuit. You will find that it will have a plot like, this kind of Gaussian distribution that means the number of paths is large in the middle and the delay is more this is the critical. This is the critical path delay this is the critical path delay and there are many other paths which will have smaller small delay and delay will vary over a range. So, this is the normal distribution of delays on different parts in a complex circuit now what we are doing initially we are we are doing the we are increasing that, I mean size by sizing. We are reducing the delay (refer time: 33:00) and pushing it may be to this point.

Then we are reducing increasing that. So, this is the first stage reduction in delay in first in step one then what we are doing whatever slack. We achieved we are you are reducing the supply voltage and as a consequence delay is increasing again and it becomes same as the original. So, this is the step the step the by supply voltage scaling supply voltage scaling after that what will happen as we do the supply voltage scaling some of the delays of the non-critical part will go to this, this side and makes it makes it this part in such a case (refer time: 34:00) again you have to do sizing. So, that all the for delay for all the path remains within this critical path what does it mean as the delay is remaining within the critical path there is no loss in performance that means performance is not compromised but because of larger supply voltage we are achieving in the second step there will be significant reduction in power dissipation. So, this is the technique in which we have done voltage scaling through optimal transistor sizing. So, this is how we can you can utilize sizing; obviously, whenever we do this here'll be increase in area, because wherever we are upsizing here'll be increase in area that means, there again we are trading area for lower power by this technique. So, this is the second technique that we can do (refer time: 35:00)

(Refer Slide Time: 35:01)



Voltage Scaling Using High-level Transformations

A first-order Infinite Impulse Response (IIR) filter:

$Y_N = X_N + K^*Y_{N-1}$

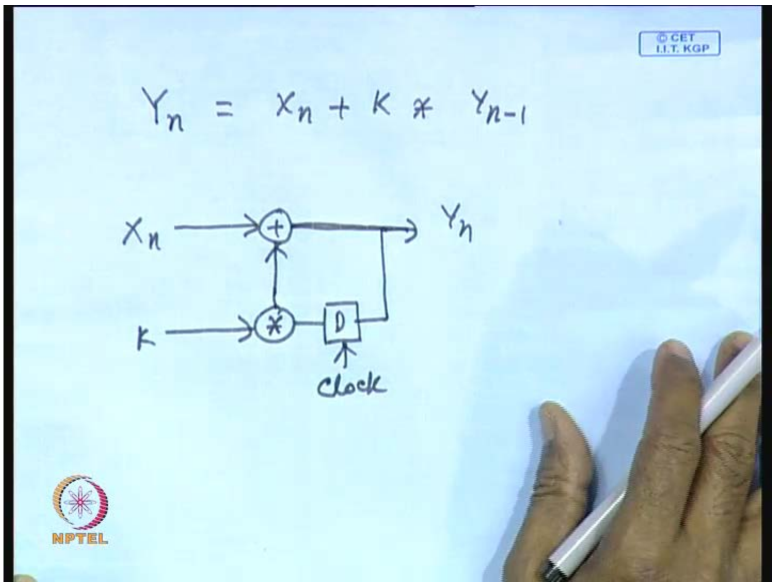
Vdd	5V
Throughput	1X
Power	1X

DAG representation

Ajit Pal IIT Kharagpur NPTEL 17

Now, let us focus on third technique voltage scaling using high level transformation whenever we are realizing some circuit can do high level transformation what kind of transformation. We shall do let me explain with the help of this example suppose, we have to realize a first order i i r filter infinite impulses response filter.

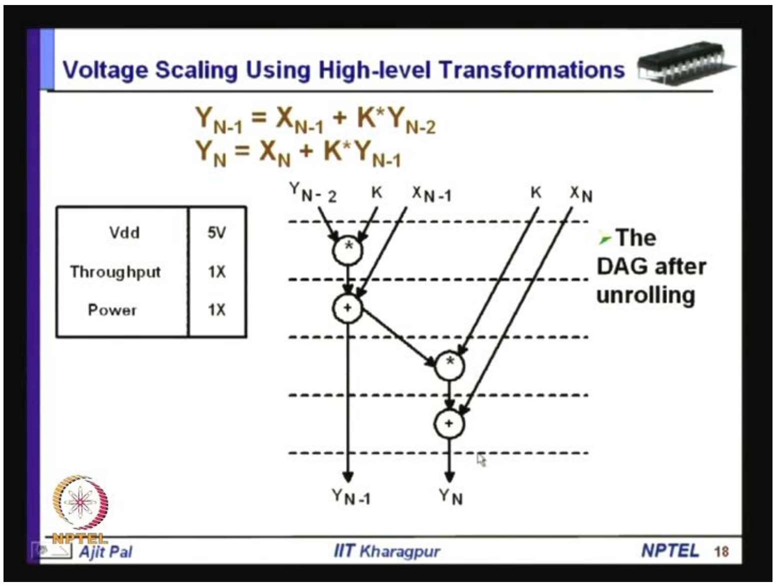
(Refer Slide Time: 35:34)


$$Y_n = X_n + K * Y_{n-1}$$

As you know the expression for that filter is y n is equal to x n plus k into y n minus one. So, this is how repeatedly-repeatedly this is completed in a alternative manner. You can realize a circuit in this way. You will apply x n this is the input x n will (refer time: 36:00) be added with y n minus one and you will be have to multiply the factor you have to multiply with k and how do you generate this is this is where you generate y n and y n minus can be generated by delay element or latch. So, this is where you will apply clock. So, this is how the circuit can be realized with the help of a adder and a multiplier.

So, this is this computation is going go on as you apply different values of x n and k is a constant. That is fed there and this is how you can realize a i r a i i r filter (Refer Slide Time: 35:01) and the corresponding dag is shown there is a corresponding dag that means you are doing multiplication in one step in one times one clock cycle (refer time: 37:00) and addition you are doing in a next clock cycle. That means this is the directed acyclic graph. You have you have done the scheduling you know whenever you do high level synthesis you have to do the scheduling. So, scheduling is done in this way at the multiplication in the first time slot and addition in the second time slot. So, this is how you can do the computation. So, this we shall use as reference. So, let us assume the supply voltage is five volt and throughput is one x and power that is required for this computation is one x.

(Refer Slide Time: 37:38)



Voltage Scaling Using High-level Transformations

$Y_{N-1} = X_{N-1} + K^*Y_{N-2}$
$Y_N = X_N + K^*Y_{N-1}$

Vdd	5V
Throughput	1X
Power	1X

- The DAG after unrolling

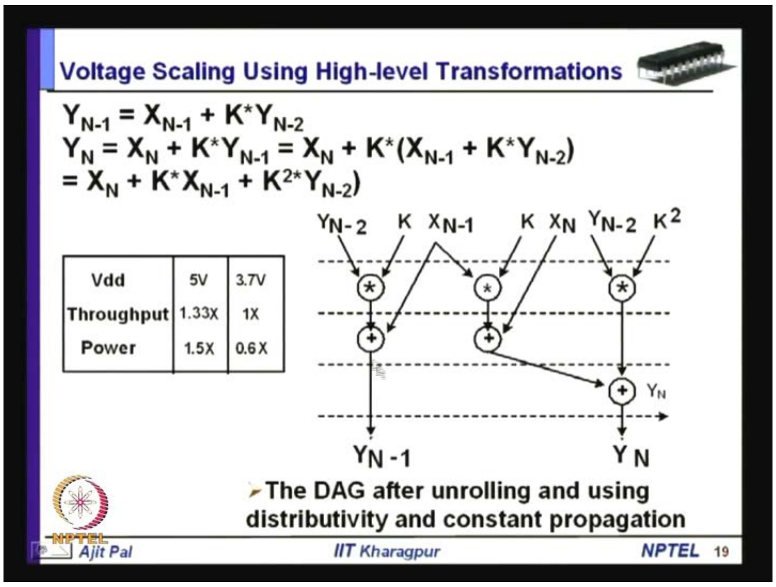
Ajit Pal IIT Kharagpur NPTEL 18

Now, let us go to a some kind of transformation what we have done in this particular case we are computing the outputs simultaneously and by using a technique known as loop unrolling. So, loop unrolling is a very useful technique (refer time: 38:00) that is used in many situations for example, whenever we go for optimizing compiler that compiler. You know does loop unrolling to improve instruction level parallelism which help in having more instructions which can be executed in parallel that is important in the context of minimizing hazards in pipeline circuits. So, that is done in optimizing compiler there also we have done loop unrolling.

So, loop unrolling means is what is being performed in one alteration we are performing in a single alteration we are performing such operations we can unroll it three times then we will what is performed in three alterations now. It is done in single alteration. So, whatever happens whenever we do loop unrolling the size of the cord cords increases, but in this particular case we are not doing (refer time: 39:00) by using by executing instructions but using hardware. So, there if we want to generate y n minus one and y n together; obviously, we have to duplicate the hardware and this is the corresponding dag.

So, there you will be applying y n minus the k x n minus one k and x n and he[re]- you will be generating y n minus one and y n together. So, you can see the outputs are generated simultaneously now you may be asking is there any benefit by doing this. You are requiring four clock cycles and you are producing the outputs earlier you were requiring the clock cycles you were producing one output. So, there is no benefit. So, simply by doing loop unrolling it does not give you much benefit unless you exploit the some features of loop unrolling what can be done you can do some transformation (refer time: 40:00)

(Refer Slide Time: 39:59)



Voltage Scaling Using High-level Transformations

$Y_{N-1} = X_{N-1} + K^*Y_{N-2}$
$Y_N = X_N + K^*Y_{N-1} = X_N + K^*(X_{N-1} + K^*Y_{N-2})$
$= X_N + K^*X_{N-1} + K^{2*}Y_{N-2})$

Vdd	5V	3.7V
Throughput	1.33X	1X
Power	1.5X	0.6X

- The DAG after unrolling and using distributivity and constant propagation

Ajit Pal IIT Kharagpur NPTEL 19

Y_{N-2}
K
X_{N-1}
K
X_N
Y_{N-2}
K^2
Y_N
Y_{N-1}
Y_N

So, there what we are doing we are using kind of constant using technique like constant propagation and using the property of distributive-distributive and by doing that you can see the computation can be done in three steps in three clock cycles. So, there we are performing we are computing k square and keeping it in a resistor that is available as a constant. So, you can see y n minus one is produced in this way there you require y n minus one is equal to x n minus one plus k star y n minus the. So, this requires the clock cycles.

So, this multiplication is done there k into y n minus one and then addition with x n minus one is done there. So, you get there y n (refer time: 41:00) minus one then in this to produce y n we are we are doing some kind of simplification. We are substituting y n minus in this expression and then we are getting this expression x n plus k star into x n minus one plus k star y n minus the. So, there y n minus one has been substituted then we are simplifying it to get this expression.

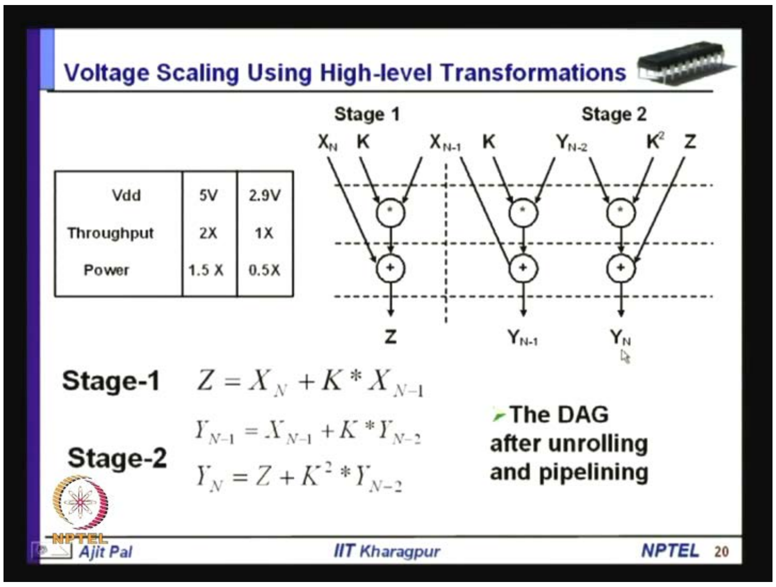
So, x n plus k k into x n minus one plus k square into i n minus that, do this is being implemented in this part of the circuit. So, this; obviously, will require three clock cycles. So, you can see there whenever you do this kind of simplification this kind of transformation we can compute in three clock cycles. Now we have got the options we can compute in three clock cycles and; obviously, (refer time: 42:00) we are able to compute faster than the original circuit where you have done (( )) unrolling. So, there you'll it requires four clock cycles there. It requires three clock cycles instead of that what can be done if time required for three clock cycles can be made same as that of time required for four clock cycles by reducing. How can it be done by reducing the frequency and simultaneously reducing the supply voltage?

So, we reduce the supply voltage because you know we are increasing the slack. Whatever was required in three clock cycles now you are doing in four clock cycles? So, we can we can afford to use lower supply voltage and that is what is being done in this particular case. So, you can see we have done supply voltage scaling the voltage of the (refer time: 43:00) for this implementation is reduced to three point seven volt and throughput is one x and power dissipation is point six x. So, you can see there is significant reduction in power dissipation of course; we could have as mentioned earlier these techniques can be used to improve performance as well.

So, if we do not do supply voltage scaling if we operate at five volt then throughput increases by thirty three percent, because we are completing our execution in three clock cycles instead of four clock cycles; however, at the cost of higher power dissipation larger power dissipations that means we are we are dissipating one point five times that of the original circuits, I mean of course, power computation.

So, you require tree multiplier three adders. So, in the original circuits but you are producing the outputs in the original circuit it was one multiplier and one adder to output use one output but there you are producing (refer time: 44:00) the output and it is three times. So, hard power dissipation will be one point five times that of the original circuit. So, you see there also we can do transformation and then supply voltage scaling to achieve lower power we can scale reduce power dissipation by using a technique by using pipelining.

(Refer Slide Time: 44:42)



Voltage Scaling Using High-level Transformations

Vdd	5V	2.9V
Throughput	2X	1X
Power	1.5 X	0.5X

Stage-1 $Z = X_N + K * X_{N-1}$

Stage-2 $Y_{N-1} = X_{N-1} + K * Y_{N-2}$

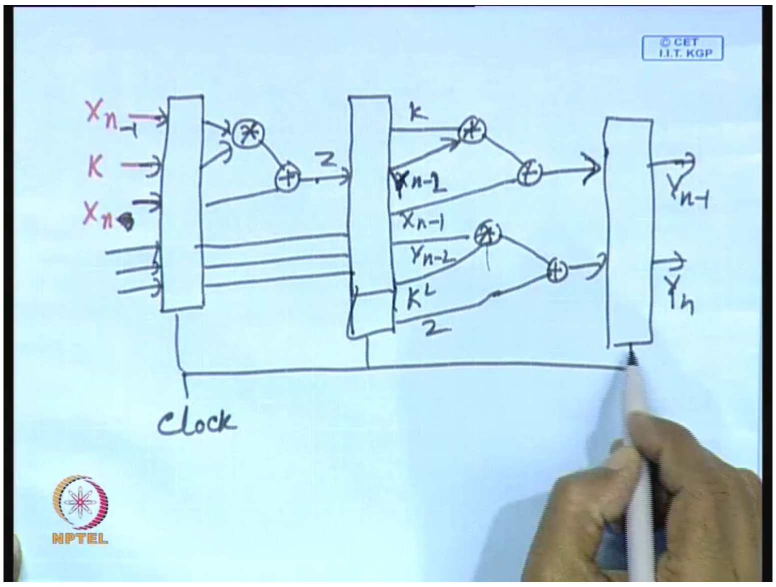
$Y_N = Z + K^2 * Y_{N-2}$

- The DAG after unrolling and pipelining

Ajit Pal IIT Kharagpur NPTEL 20

So, this particular stage can be pipelined how can it be pipelined you can see in the stage one we shall compute z some intermediate result then z will be used in the second stage to produce the y n minus one and y n. So, there the implementation will be somewhat like this say.

(Refer Slide Time: 45:05)



You will be applying x n (refer time: 45:00) k x n minus one these three will go to a latch. These will go to a latch is the first stage latch and then you will be requiring a multiplier. So, k into x n minus one this, so x n minus one and let it be k. So, x n minus one and k these are applied there and then we do addition. So, this output is coming there and this produces z.

Now, you'll require another stage and there you'll be requiring multiplier adder multiplier adder (refer time: 46:00). So, what we will be applying there you'll apply multiplication k this factor will go there k and you are multiplying it with x n minus one x n minus one. So, x n minus one k into x n minus one and you are adding with sorry this will be sorry. This is y n menus y n minus this is y n minus the y n minus the and there it will be x n minus one and in this particular case. We will be applying again y n minus the and k square and there from this stage we will go addition will go z this z is going there (refer time: 47:00).

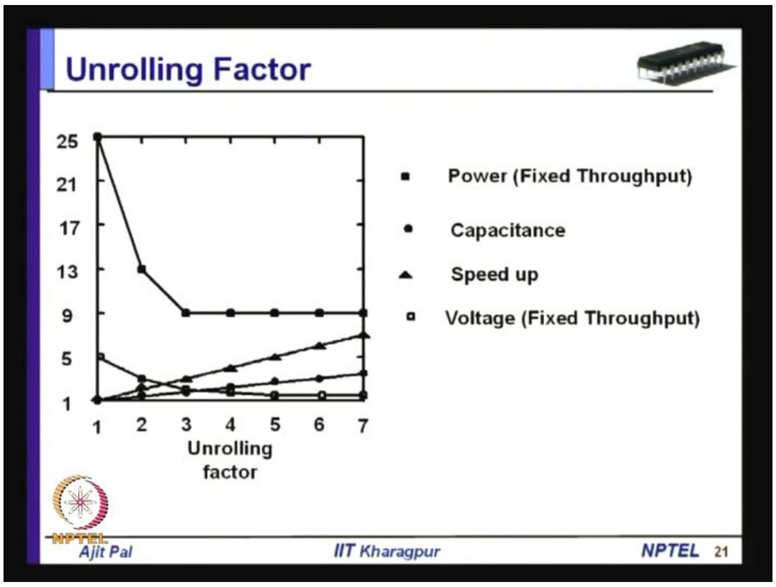
Obviously all those other inputs will pass through the first stage and there it will produce at the output of this. You will get the outputs one is your y n minus one and y n. So, this is the pipeline implementation and there you will apply clock. So, you can see there each of this stage will require the clock cycles because you have got the stages. So, this stage will require the clock cycles that is delay for each stage is equivalent to the clock cycles that means after tf

the pipeline is filled up it will produce output at the interval of path the clock cycle of the original circuit. That means the clock frequency is twice that of the of the original circuit that means in this case if you look at this particular diagram we find that (refer time: 48:00) in the time slots all the computation can be done. So, the first stage will require the clock cycles. The second stage also will require the clock cycles though the time the clock frequency of the pipeline stage is decided by the clock cycles that means you have got the option again there either you can do the computation and produce result quickly or you can reduce power dissipation by scaling down the supply voltage (Refer Slide time: 44:42). So, there is the result for that whenever you do voltage scaling and the time required time to produce these outputs will be same as that is required in this circuit in this circuit.

So, whatever time is required for clock cycles same time is required there in that case. Throughput is remaining same that means the (refer time: 49:00) the rate at which outputs are generate is identical; however, there is a significant reduction in power dissipation fifty percent on the other hand if you want to speedup have higher throughput the x then of course, it can be done by using higher power larger power dissipation. So, this is achieved by using loop unrolling and pipelining. So, you can see we can do transformation high level transformation and also we can make use of pipelining to reduce power dissipation

Now, the question arises how many stages you will use in this pipeline stage

(Refer Slide Time: 49:46)



Unrolling Factor

Ajit Pal IIT Kharagpur NPTEL 21

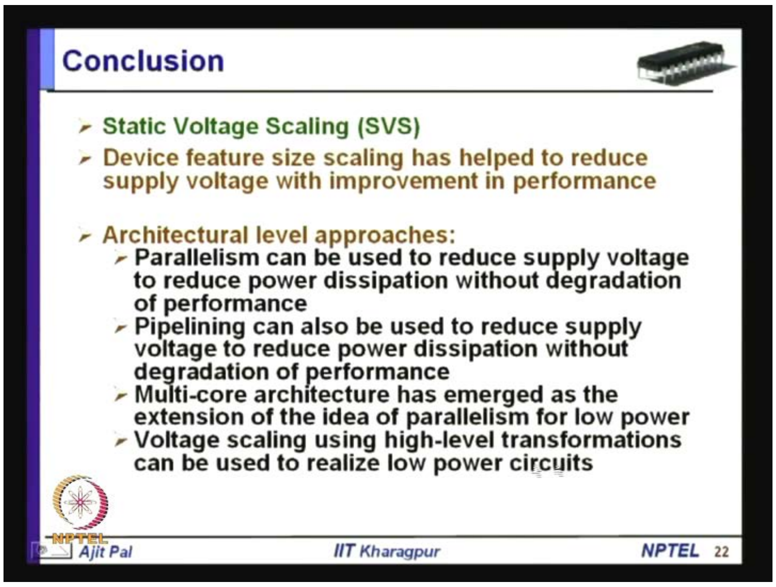
So, there is the curve for different situations as you can see the unrolling factor can be increased we have considered where the i a r filter operations has been unrolled by the times you can unroll it by three times four times (refer time: 50:00) five times and so on. So, what it is the benefit that we achieve by unrolling by more number of times is shown in this particular diagram

First of all if you let us focus on the power dissipation we can see as we as whenever we do the times there is significant reduction in power dissipation. We have seen fifty percent reduction in power dissipation keeping the throughput same if we unroll by three times still there is a reduction it comes to about that means nine instead of twenty five. So, there is significant reduction but further increase in unrolling does not really reduce the power dissipation because overhead increases. So, overhead it is of the benefits of more unrolling soup to unrolling by a factor of three gives you should benefit.

However as you do the unrolling you can see there (refer time: 51:00) other parameters like capacitance keeps on increasing as it is shown by this linearly it increases, because number of additional as you do the unrolling they will be requiring more and more hardware and. So, the capacitance will keep on increasing speedup factor will increase speedup will also increase linearly but voltage that you can use can be reduced only initially as you go from single-single stage to I mean unroll by the or unroll by three beyond that it cannot really reduce the supply voltage much and that is the reason why we were not getting reduction in power dissipation.

That means there is an optimum unrolling factor beyond which it becomes a point of diminishing return and at some point there is no return at all. So, that means you will have to choose a (refer time: 52:00) unrolling factor which gives you maximum benefit and beyond that it does not give you much result.

(Refer Slide Time: 52:11)



Conclusion

- Static Voltage Scaling (SVS)
- Device feature size scaling has helped to reduce supply voltage with improvement in performance
- Architectural level approaches:
  - Parallelism can be used to reduce supply voltage to reduce power dissipation without degradation of performance
  - Pipelining can also be used to reduce supply voltage to reduce power dissipation without degradation of performance
  - Multi-core architecture has emerged as the extension of the idea of parallelism for low power
  - Voltage scaling using high-level transformations can be used to realize low power circuits

Ajit Pal IIT Kharagpur NPTEL 22

So, this with this we conclude our discussion on this topic now we can summarize what we have discussed, so far. We have discussed device feature size scaling in the previous lecture and we have seen how it has helped to reduce supply voltage with improvement in performance and we can we have discussed several architectural level approaches like parallelism pipelining and we have also discussed multiple implementation and how they can be used to achieve lower power by exploiting parallelism and also we have discussed voltage scaling using high level transformation that can be used to realize lower power and also and we have discussed. How sizing can be done to reduce power. So, with this we have come to the end of today's lecture in the next lecture we shall discuss (refer time: 53:00) about other techniques like multi-level voltage scaling. Thank you.