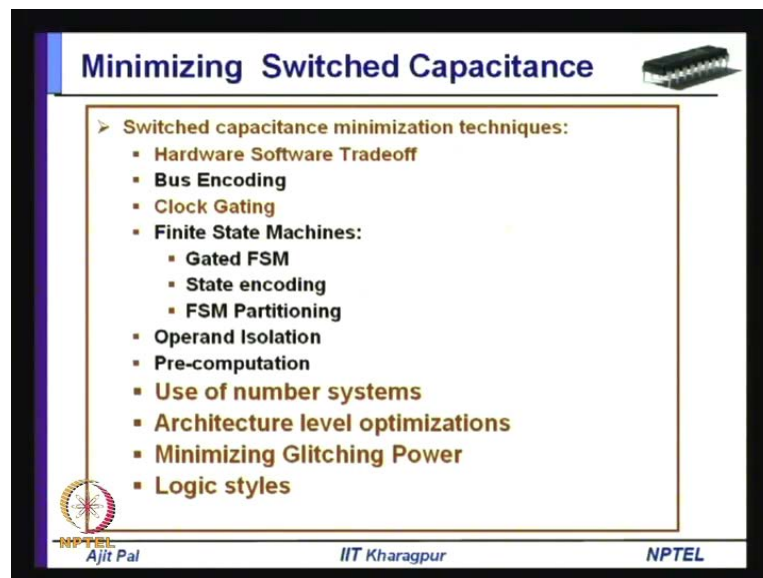


**Low Power VLSI Circuits and Systems**  
**Prof Ajit Pal**  
**Department of Computer Science and Engineering IIT Kharagpur**  
**Indian Institute of Technology, Kharagpur**

**Module No #01**  
**Lecture No #31**  
**Minimizing Switched Capacitance - V**

(Refer Slide Time: 00:27)



The slide is titled "Minimizing Switched Capacitance" and features a small image of a microchip in the top right corner. The main content is a list of techniques for minimizing switched capacitance, presented in a bulleted format. The list includes:

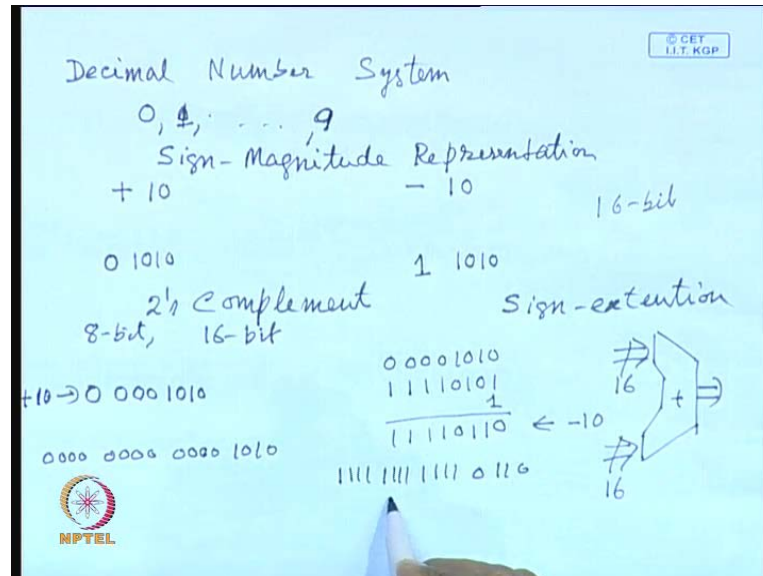
- Switched capacitance minimization techniques:
  - Hardware Software Tradeoff
  - Bus Encoding
  - Clock Gating
  - Finite State Machines:
    - Gated FSM
    - State encoding
    - FSM Partitioning
  - Operand Isolation
  - Pre-computation
  - Use of number systems
  - Architecture level optimizations
  - Minimizing Glitching Power
  - Logic styles

At the bottom of the slide, there are three logos: NPTEL on the left, IIT Kharagpur in the center, and NPTEL on the right.

Hello and welcome to today's lecture on minimizing switched capacitance. We shall continue our discussion on this topic, and in the earlier lectures we have discussed about various techniques; like hardware software tradeoff, bus encoding, clock gating, then use of different techniques like; gated FSM state encoding and FSM partitioning, used in minimizing switched capacitance and finite state machines. We have also discussed operand isolation and pre-computation, which are used to review switching activity in combinational circuits. Now, today we shall first focus on, how we can use a suitable number system in minimizing switched capacitance, before we discuss other techniques like; architecture level, optimizations, minimizing glitching power and various logic styles, that can be used to reduce switched capacitance. All of us are using some number

system in our day to day use, and for arithmetic computation like; addition, subtraction, multiplication division, we also use some number system within the computer.

(Refer Slide Time: 01:37)



Now, let us consider the number system that we use in our day to day system. We use what is known as decimal number system, decimal number system with radix obtained. Then you can represent numbers using 0 1 up to 9, 9 different symbols or numbers, and of course in addition to this, we use positive and negative signs, to represent positive and negative numbers. So, this is typically known as sign-magnitude representation. **Sign-magnitude representation**, where we use sign explicitly by this symbol positive symbol, and then we give the magnitude value say, we would like to represent plus 10, so we write plus 10. Similarly, if it is a negative number, we represent it by a negative sign minus, so this is minus 10. So, this is how we normally represent numbers, in sign-magnitude representation. However, whenever you are using a computer to do computation; obviously, everything has to be in terms of zeros and ones. So, has to represent by a binary values.

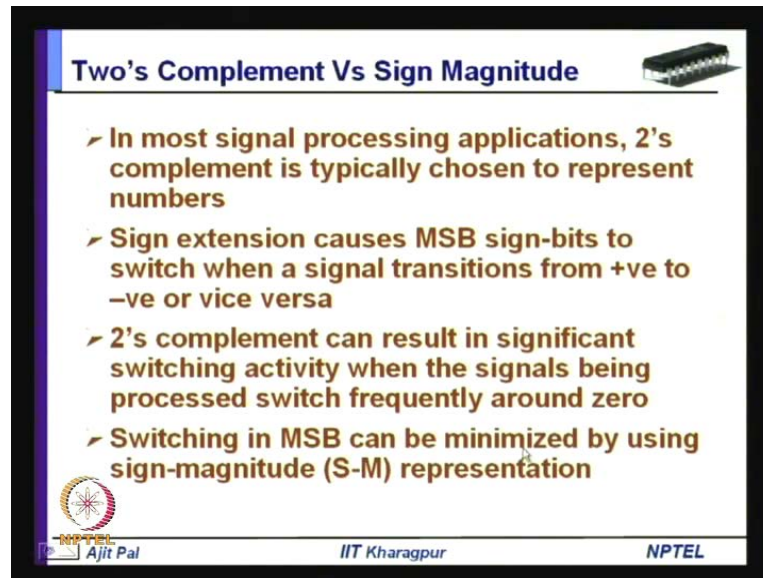
So, sign is also given a binary value, usually 0 is used to represent a positive number, one is used to represent the sign of a negative number. So, in sign-magnitude form, a binary value, binary representation of plus 10 will be 0, then 1010. So, this part is the sign, and this part is the magnitude. Similarly, the sign- magnitude representation of minus 10 can be 1, then of course; the magnitude part will be remaining same, so 1 and

10. However, the sign-magnitude representation is not very convenient in implementing various arithmetic operations within the computer; that means, implementing addition, subtraction and other arithmetic operations, cannot be done efficiently by using the sign-magnitude representation. So, another representation that is being used, that is known as two's complement representation. In two's complement representation, usually we say the number is either 8-bit or it is a 16-bit, and accordingly we represent the various values.

For example, this plus 10 in 8-bit representation of two's complement form will be 0. Then you will add three more 0 and followed by the actual magnitude. Similarly, the negative value of this, has to be obtained in a little different way, what we normally do we first complement the value, say 0 0 0 0 1 0 1 0, so this is being bit complemented, so we make bit by bit complementation 1 1 1 1 0 1 0 1, then we add one to get the two's complement representation of minus 10. So, you get 0 1 1 0 1 1 1 1. So, this is this is minus 10 in two's complement form, and this is plus 10 in two's complement form. So, this is how it is being represented inside the computer, when we perform various arithmetic operations. We use a concept known as sign-extension.


Suppose we have to add a 8-bit number, I mean suppose this is 8-bit value and another number is of 16-bit, so how do you add this 8-bit number with a 16-bit value, that can be done by sign-extension of this number and convert it into 16-bit number, then perform the addition; that means, your adder will take both the values 16-bit, here also it will take 16-bit, but if it is a 8-bit number then it has to be converted into 16-bit by using the concept of sign-extension. For example, if we do this sign-extension of this plus 10, it will become 0 0 0 0 then 0 0 0 0 then 0 0 0 0, and then 1 0 1 0, this is the plus 10 in 16-bit; that means, 16-bit representation of plus 10, so you have got. So, this part you can see nothing, but sign-extension, so this is the positive number. The sign bits have been extended to fill the remaining bit. Similarly minus 10 can have say 1 1 1 1, 1 1 1 1, 1 1 1 1 then 0 1 1 0. So, this is the minus 10 in sign-extended form. So, this way you can perform a numbers, perform addition subtraction and other things of binary numbers in two's complement form, and if there the magnitudes are not same, then we do sign-extension to convert them of the same number. So, this sign-extension of binary numbers, two's complement numbers creates problem, what kind of problem it creates, let us see.

(Refer Slide Time: 07:17)



**Two's Complement Vs Sign Magnitude**

- In most signal processing applications, 2's complement is typically chosen to represent numbers
- Sign extension causes MSB sign-bits to switch when a signal transitions from +ve to -ve or vice versa
- 2's complement can result in significant switching activity when the signals being processed switch frequently around zero
- Switching in MSB can be minimized by using sign-magnitude (S-M) representation

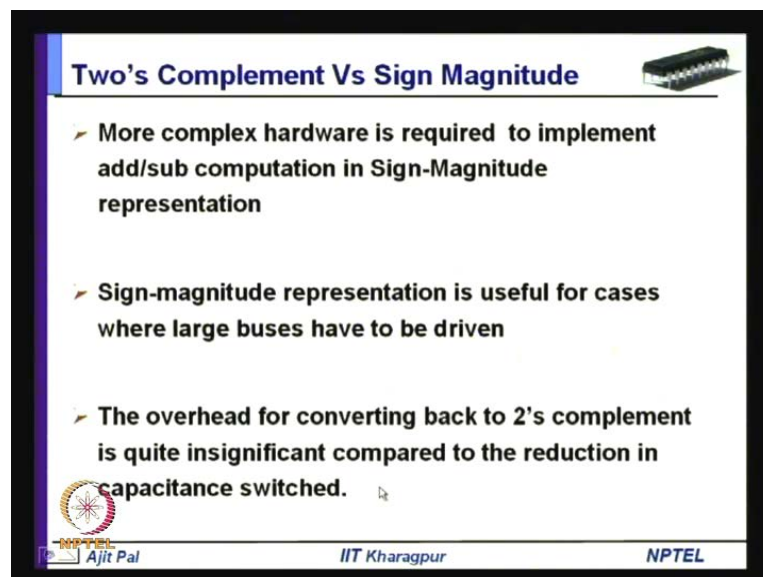
 NPTTEL  
Ajit Pal IIT Kharagpur NPTTEL

So, as I have told in MOS signal processing application, two's complement is typically chosen, to represent numbers, primarily because the two's complement number system allows you to realize adder and subtract (0) a hardware efficiently. Then sign-extension causes MSB sign bits to switch, when a signal transitions from positive to negative or vice-versa take place. For example, let us consider a situation, where the numbering changing from plus 1 to minus 1 alternately, I mean may be in between 0, so that means, the number is switching between plus 1, minus 1 and 0, value is small and may be, because of noise, because of other reasons it is changing, over 0 and it is becoming minus 1 or plus 1 or 0. You can see how the bits will be changing, here plus 1 in 8-bit will be equal to 0 1 1 1 sorry 0 0 0 0 this is a positive number. So, it will be 0 0 0 1, and 0 obviously will be 0 0 0 0 in two's complement, then 0 0 0 0, and minus 1 will be, you see that you have to perform bit complementation of all these bits, and then you have to add 1. So, that will make it 1, so 1 1 1 1 all 1. So, you see if the value changes from 0 to minus 1, you see the number of bits that will switch is 8. Similarly, now if it changes from say minus 1 to plus 1 or plus 1 to minus 1, number of bits that will switch is 7.

So, you see large number of bits are switching, whenever we are representing in two's complement form, 2 c form to represent numbers. On the other hand, instead of representing numbers in two's complement form, if we representing sign-magnitude form, then you can see how the number of bits will switch. So, plus 0 plus 1 is 0 0 0 0 1 and 0 is 0 0 0 0 0 0 0 1 and minus 1 is essentially 0 0 0 1 0 0 this is the sign and

magnitude part is same as the positive number. So, in this particular case, so whenever you switch from 0 to 1, number of bits that switches is only 2, on the other hand whenever you switch from say minus 1 to plus 1 the number of bits that switches is only one. So, this is a remarkable change in the number of bits that switches; that means, if the dynamic range of a number is small, and if it switches sign very quickly, then it may lead to large switching activity in case of two's complement form. On the other hand in the sign-magnitude form, the number of transitions will be much less. This is the basic idea of this particular technique, as I have told two's complement can result in significant switching activity, when signals being processed switched frequently around 0, and switching in MSB can be minimized by using sign-magnitude represents as I have already explained.

(Refer Slide Time: 10:52)



**Two's Complement Vs Sign Magnitude**

- More complex hardware is required to implement add/sub computation in Sign-Magnitude representation
- Sign-magnitude representation is useful for cases where large buses have to be driven
- The overhead for converting back to 2's complement is quite insignificant compared to the reduction in capacitance switched.

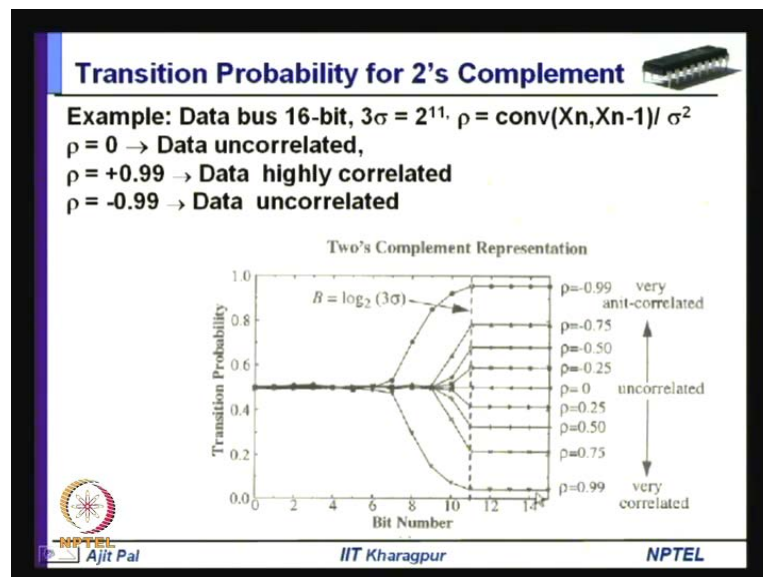
NPTEL Ajit Pal IIT Kharagpur NPTEL

Now let us see how you can really use it or in what situation we can use it. You see a sign-magnitude representation is useful for cases, where large buses have to be driven. For example, you are using, let us assume a microcontroller or microprocessor, and this microprocessor is sending data to, may be a digital to analog converter, which is receiving digital data and producing analog corresponding value, at the output of the digital to analog converter. Now, here if the values, I mean the dynamic range is small and if the change is taking place over 0, and if it takes place frequently, then you can see large switching activity result over the bus. So, you are sending through bus, and as you know the capacitance of this bus is quite large. So, this switching activity will lead to lot

of change in switched capacitance, leading to high dynamic power dissipation. This however, can be minimized by using sign-magnitude representation; in that case what you have to do, you have to add one converter. So, here you will see, say two's complement to sign-magnitude form, and here at this end you have to add another converter which will perform sign-magnitude to two's complement form.

So, if you add these two hardware, then before sending the data you can convert it into sign-magnitude form, then the sign-magnitude representation of data, can be send over this bus, and on the other hand, on the other side, you can convert it back with the help of the sign-magnitude to two's complement converter, and here there is a **(O)** what is that that **(O)**. This can be done, only when the addition of this converters, two's complement to sign-magnitude and sign-magnitude to two's complement converter, is essential and overhead. So, this overhead will add capacitance to the systems. So, by adding this capacitance, if the switch switched capacitance is less than, I mean is more. Then I mean by adding this, if the switch capacitance is less than without adding this, only then this is beneficial; that means, what you have to do, the overhead of converting back to two's complement is quite insignificant compared to the reduction and switched capacitance. Only in such situation you can use this type of technique.

(Refer Slide Time: 13:57)



And let us have some little more analysis of this approach, here data bus is of 16 bit, 3 sigma is 2 to the power 11. Here essentially this represents the dynamic range; that

means, the most of the bits are changing over 11 bits, remaining three bits does not change much, and rho that is your, although known as, this is essentially known as correlation, convolution  $x[n] * x[n-1]$  by rho square. So, again that will be that you will achieve will be dependent on two things; number one is the dynamic range of the number and another parameter is correlation of the data bits. So, this correlation and the dynamic range will affect the performance of this introduction of two's complement to sign-magnitude converter, and sign-magnitude to two's complement converter. So, let us do some analysis and see.

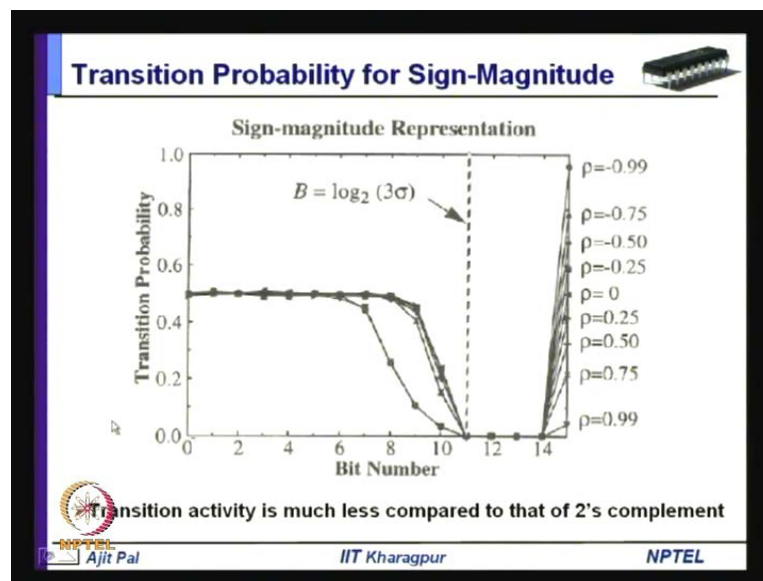
Here is the, you know transition probability that has been obtained for this particular situation, where three sigma is equal to two to the power eleven, and rho is zero, or it can be, plus it can vary. I mean depending on the correlation of the data, and you can see when rho is 0 data is uncorrelated, uncorrelated means it can become positive, it can become negative it can, it is a random so; that means, any bit can change probability of change of any bit is 50 percent. So, that is the reason why transition probability is 0.5. So, when it is uncorrelated all the bits can change, the probability of any bit to become 0 or 1 is same, in such a case we get transition probability of 0, and the bandwidth as I told is represented by  $\log_2 3$  sigma, that is your, and here you can see this is essentially in the dynamic ranges over 11 bits, remaining three bits do not change much now, but of course it depends on the correlation factor. So, if the data is correlated, what you really mean by correlated. Correlated means, what can happen if the data may change likely slowly.

So, in such a case over time if it changes slowly, then we can say data is correlated; that means, the present value and next value are correlated, means you know they are very close and it is slowly going becoming positive, then it is slowly becoming negative, so the data is correlated. So, when the data is correlated, the correlation then; obviously, the switching activity will be less as you can see, and whenever rho is equal to 0.99 it is very much correlated. So, you can see. Of course, the lower order bits will change, because dynamic range is higher, and you can see depending on the correlation, say it is point 0 to 5 then you can see only the higher order bits, and these bits are not changing much, but lower order bits are changing; that means, probability is 0.5 and as the correlation factor increases, then you can see the even lower order bits are not changing much. So, the bits are changed from say bit 7 to 14, there could there switching activity is reducing

7 to 11, these switching activity of opposite bits are changing, as the correlation is increasing. On the other hand, if the data is anti-correlated; that means, they are changing fast.

So, it may so happen that, data is changing fast like this. So, data is changing quickly. So, it is anti-correlated you can see, in this particular case it is anti-correlated. So, in such a situation, depending on what kind of anti-correlation is there, the switching activity is increasing from four bits starting from 7 to 11 as you can see, it is increasing and also higher order bits it is increasing; that means, including sign bit, it is increasing. So, of course, if the dynamic range is small then; obviously, it will move towards this for this particular dynamic range,  $3\sigma$  is equal to  $2$  to the power  $11$ , this is the situation. So, you can see, that transition probability is dependent on two parameters; one is the bandwidth or the dynamic range of the signal, and second is the correlation factor, and depending on that the number of transitions on different bits will vary, which is represented by these plots.

(Refer Slide Time: 19:13)

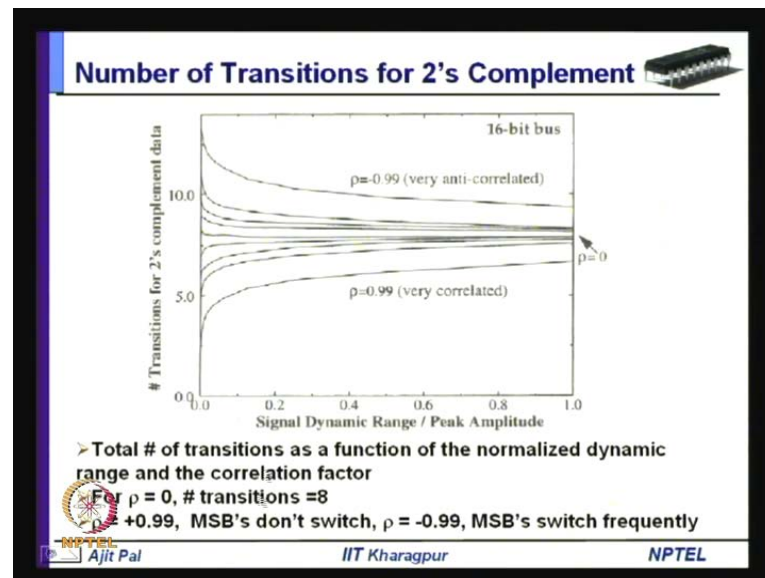


Now, let us consider the same situation for sign-magnitude representation, one was for two's complement representation. Now, you have come to two's complement sign-magnitude representation. Here as you can see, except the sign bit for all other bits the transition probability is always less than point five, and not only that, as the correlation increases, the transition probability of the sign bit as well as the other bits reduces. So,



transition probability of order bits starting from bit 6 reduces, for sign-magnitude representation, and it never what becomes more than 0.5, because as you have seen in case of sign-magnitude representation, you have seen that the number of transitions will be always less, and even with anti-correlated data, the number of transitions will never exceed 0.5. And as a consequence the for sign-magnitude representation, the transition activity is much less compared to that of two's complement representation. And obviously, this reduction will be more and more if the bandwidth is less, and also the data is highly correlated, and as the data becomes anti-correlated; that means, switching occurs more frequently, between positive and negative, and of course the sign which will change more frequently if the data is uncorrelated.

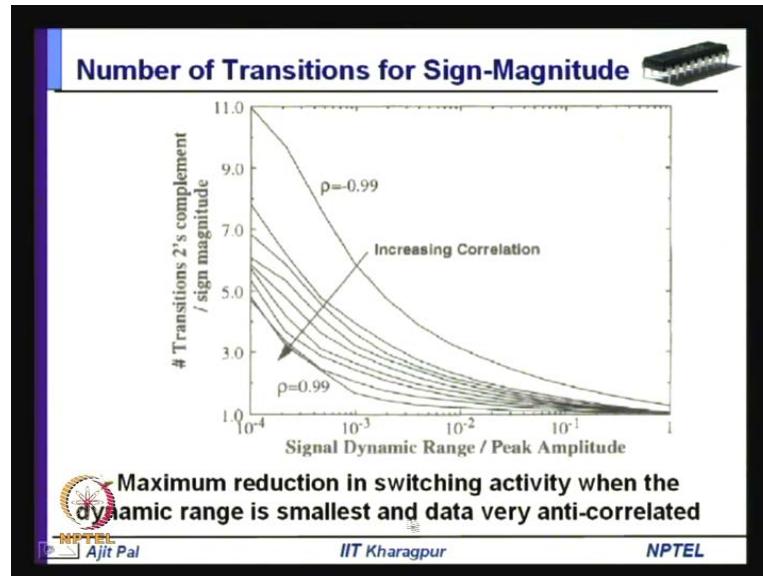
(Refer Slide Time: 21:02)



Now, there is another plot, where the transition for two's complement data is plotted on the y axis, again this signal dynamic range, but peak amplitudes, as I told it depends on two parameters; the correlation factor and the dynamic range. As you can see as the signal dynamic range is increasing, the probably transition activity is increasing, and whenever it is highly anti-correlated, the transition activity is much less compared to when it is highly correlated, and rho is equal to 0.99 and for point rho is equal to 0, you can see, it is roughly point 8, because it is, I mean it will be 8 because it is 16-bit number. So, the transition number of transitions for two's complement will be 8, for rho is equal to 0, for 16-bit number and number of transitions will be 8. And for 0.99 MSB do not switch at all, and for minus 0.99 MSB switches frequently. And as a consequence

the variation of the transitions for two's complement representation take place in this way, depending on the values of dynamic range and the correlation along different data.

(Refer Slide Time: 22:33)




Now, let us consider this in the situation for sign-magnitude form. So, number of transition for two's complement by sign- magnitude. So, it has been normalized with respect to two's complement representation. So, here you find the, here as you here for highly anti-correlated form, sorry highly anti-correlated form, the reduction is less, but whenever it is highly correlated, it reduces significantly, this ratio is quite large. So, number of transitions for two's complement by sign-magnitude, here the reduction is of course, you can see it also changes with the dynamic range, but and as the dynamic range increases, the ratio is less and as the dynamic range is small, then the ratio is more. In other words maximum reduction occurs in switching activity, when the dynamic range is small and data is very anti-correlated. So, reduction is achieved whenever it is anti-correlated. And in other words this suggests, when you will use, when you will chose sign-magnitude representation. You will chose sign-magnitude representation for sending above the bus, when the dynamic range is small and data is ant-correlated. If the dynamic range is large and data is correlated, you may not get the benefit of using sign-magnitude representation.

(Refer Slide Time: 24:16)

### Architectural Level Optimizations

- Architectural optimizations include optimizing
  - Number representation for arithmetic computations
  - The ordering of operations
  - Resource utilization
  - Minimizing glitching activity

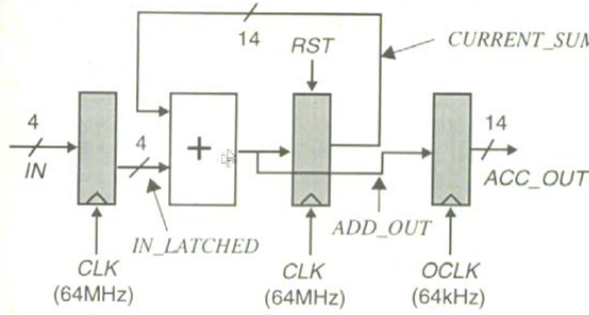


Ajit Pal IIT Kharagpur NPTEL


Now, let us focus on architectural optimizations, including number of presentation for arithmetic computation. The ordering or operations, resource utilization and we shall discuss techniques for minimizing glitching activity.

(Refer Slide Time: 24:31)

### Optimizing Data Representation



- 2's Complement implementation of an accumulator
- Bit-3 tied to bits 4-13 for sign extension

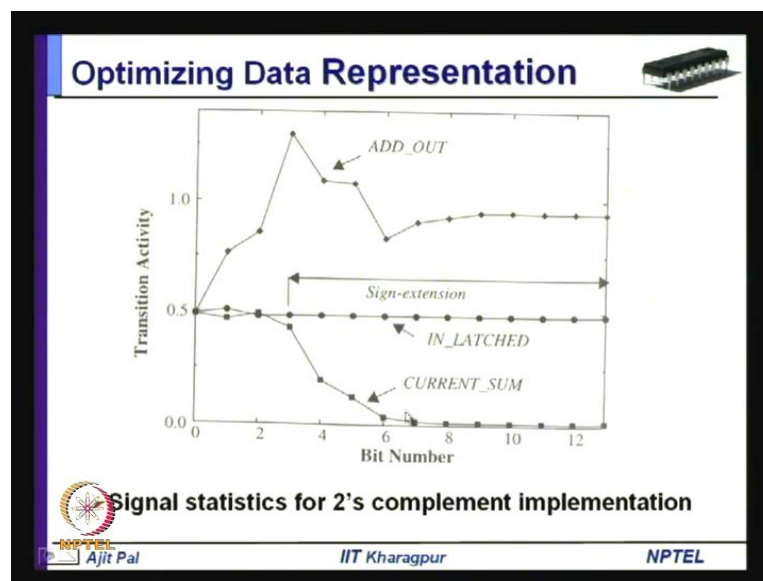


Ajit Pal IIT Kharagpur NPTEL

Coming to again two's complement representation of an accumulator, this is a operation you know a large number of sample values, are added or accumulated in many DSP applications, say one thousand sample values are accumulated, and that is being used after accumulating thousand sample values. So, this is what is being done, by this

hardware. So, here you can see or applying clock at the rate of 64 megahertz, here the data is only 4 bit, and that is being added, that is being kept on adding for thousand different values, and after the accumulation has taken place for thousand different values. The final result is taken out from the output of the adder, with the help of this latch, which is operating at 64 kilohertz. So, you can see, this one thing which is not shown here, here you are doing sign-extension, because you have to add or you can see after accumulating for thousand times, the size of the, number of bits that been required is 14 bit, that is the reason why the adder is of 14 bit, but here the data is a 4 bit. So, you have to do sign-extension before you perform addition, with the 14 bit number, and as a consequence as you do sign-extension, and then you know if the sign changes frequently, then lot of switching activity will occur, and that will lead to lot of switching activity within this adder, and this happens, because of sign-extension of this number, because dynamic range is small only 4 bit, but you are adding with 14 bit number.

(Refer Slide Time: 26:29)

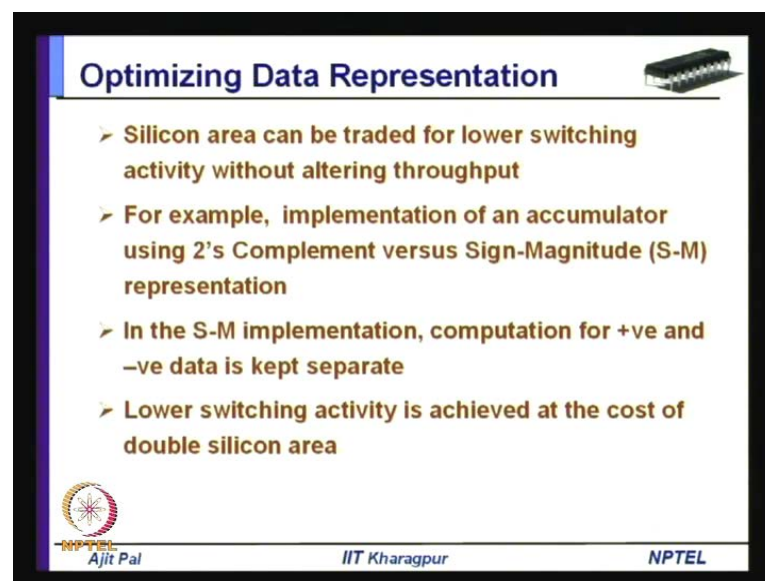


So, whenever you do this, if you plot the switching activity at different point, you can see at the adder output the switching activity is very high, the reason for that as I told this adder output, what is happening you are doing sign-extended value of data you are adding with the 14 bit number. And as a consequence at this point adder output will have lot of intermediate transitions, leading to high switching activity. On the other hand, after the addition is performed, you know that in latched data will not have that much switching activity, here you know here you can see the current sum will not have that

much activity, so here there is effect of kind of low pass filtering. So, as you as you keep on adding data the switching activity gradually reduce.

There is a kind of gradual you know smoothness on the card, and as a result you get lesser and lesser switching activity, in higher bit numbers, lower bit, I mean higher bit numbers not changing at all, but you can see the switching activity is much less in the current sum, but in this the in latched data the sign extension part will have point 5 switching activity, and this the lower part 4 bit will have three bit which is essentially magnitude. Again we will have switching activity plus 2.5. So, on this three lines, we have seen, here, here and here, how the switching activity changes, and of course the switching activity is maximum at this point, how can we reduce, the power dissipation due to high switching activity, by changing the architecture of implementing this accumulator.

(Refer Slide Time: 28:31)



**Optimizing Data Representation**

- Silicon area can be traded for lower switching activity without altering throughput
- For example, implementation of an accumulator using 2's Complement versus Sign-Magnitude (S-M) representation
- In the S-M implementation, computation for +ve and -ve data is kept separate
- Lower switching activity is achieved at the cost of double silicon area

NPTEL  
Ajit Pal

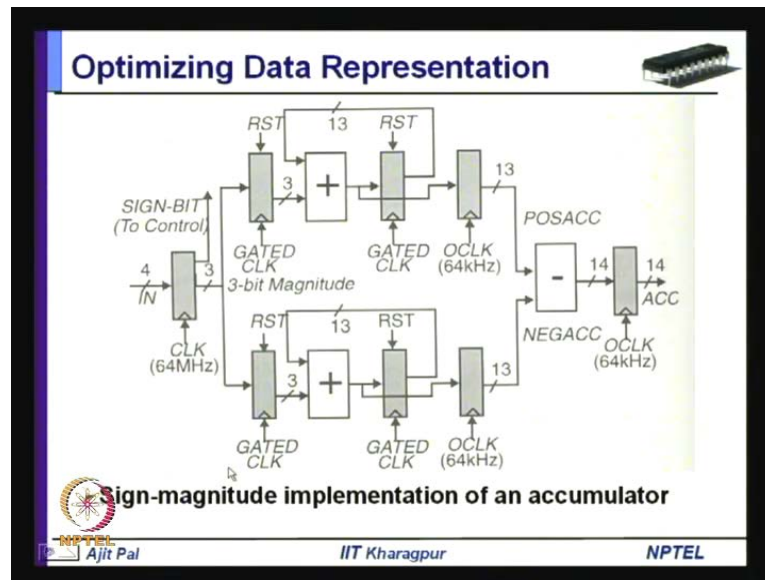
IIT Kharagpur

NPTEL

What we can do, we can trade silicon area for lower switching activity; that means, we can use some additional hardware, to achieve lower switching activity, and for example implementation of an accumulator using two's complement, can be done in a different way. We can do accumulator implementation, using sign-magnitude representation, and in sign-magnitude representation, you know computation is for positive and negative data is kept separate, how it is being done, what you can do, so depending on whether the data is positive or negative, you separate out the data, and do the additions of positive

data in one part and negative data by using another part. Then finally when the accumulation has taken place, then you perform subtraction between the two accumulated data to get the final result. So, you will get the lower switching activity, at the cost of double silicon area.

(Refer Slide Time: 29:41)

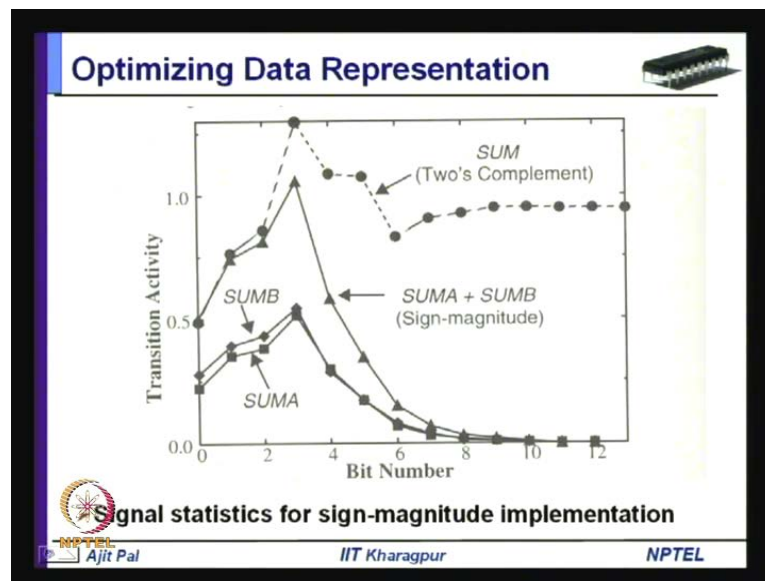


So, as you can see how this is done in this particular implementation. So, in this case the sign bit has been used to channelize the data; either to this part or to this part. So, you this gated clock is realized, or gated clock is implemented with the help of the sign bit, and if the data is positive, then this latch is enabled and accumulation is done by this part. Similarly, the data is negative, then this latch is enabled and the accumulation is done by this part. So, you can see although your input data is in two's complement form, but you are doing the additions with the help of two separate hardware; one for positive numbers another for negative numbers. And as a consequence the switching activity in this hardware or in this hardware will be much less, because you are only passing positive data. So, this sign extended part will remain same, because all are positive.

Similarly, if it is negative data sign extended part will be all 1 in this case. And in this case sign extended part will be all 0 in this case, and as a consequence the switching activity at the output of this particular adder 1 will be much less compared to two's complement form, and after performing this at high speed you know, after you have done this additions, you will of course, perform final subtraction at lower rate 64 kilohertz;

that means, subtract thousand accumulation of thousand data has taken place. You will now shift it to in a 1 adder, I mean subtract or it is doing subtraction positive accumulated and negative accumulated data, and then you are finally, getting the accumulated data at the output 14 bit data. So, here we are not sacrificing throughput, only thing that we are doing, we are trading area for lower switching activity, by duplicating the hardware.

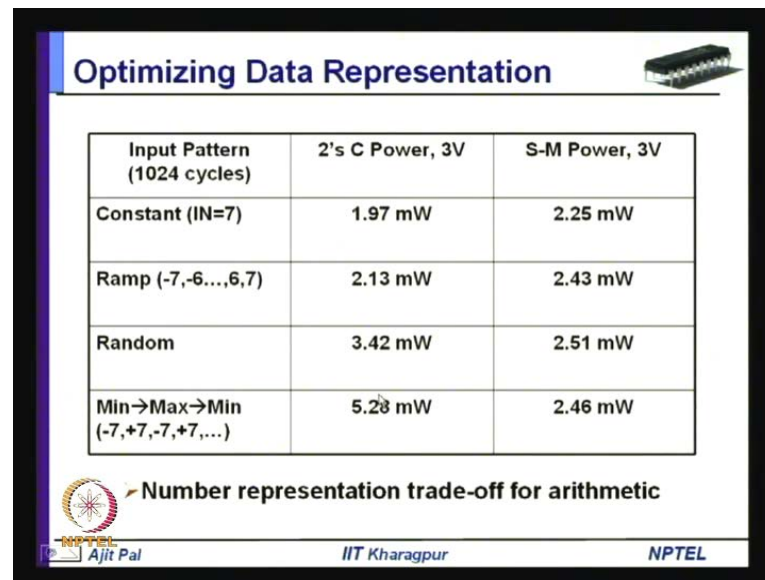
(Refer Slide Time: 31:55)



And let us see the transition activity for two different situations; the upper curve is, this particularly is dotted lines, there that correspond to two's complement implantation. So, here, that the sum output, you can see there is lot of switching activity. On the other hand these thick lines correspond to this sign-magnitude implementation, you can see either for sum a or sum b; sum a or sum b means, you can see sum a or sum b means, here you are doing sum a and you are doing sum b. So, sum a and sum b the switching activity is much less, and of course whenever you are doing that sign-magnitude, that sum a plus sum b in sign-magnitude form, when you are doing addition then of course, you have little more switching activity, but you can see that switching activity in this part, here you are doing at a lower frequency 64 kilohertz, as a consequence, since it is multiplied by factor of f, you know and the switching activity at the output of this adder will not increase the switched capacitance much, because you are you will be multiplying by factor f. So, here you are doing at the rate of megahertz, and here you will be doing at the rate of kilohertz.




(Refer Slide Time: 33:29)



**Optimizing Data Representation**

Input Pattern (1024 cycles)	2's C Power, 3V	S-M Power, 3V
Constant (IN=7)	1.97 mW	2.25 mW
Ramp (-7,-6...,6,7)	2.13 mW	2.43 mW
Random	3.42 mW	2.51 mW
Min→Max→Min (-7,+7,-7,+7,...)	5.28 mW	2.46 mW

Number representation trade-off for arithmetic

 NPTEL  
Ajit Pal

IIT Kharagpur

NPTEL

So, as a consequence the overall reduction in the power dissipation will be much more in this implementation, and you can see here, some experiment has been done, for data with different correlation factor. In the first case you are feeding constant, constant means you are not changing the input at all. So, when you are feeding the constant value, data is highly correlated, and whenever data is highly correlated as we know, it will not make much change, much difference for sign-magnitude or two's complement, because the switching activity is dependent on the correlation factor. So, when it is highly correlated it will not make much difference; however, since the sign-magnitude implementation has got larger area capacitance, that will lead to larger power dissipation. So, when you are feeding constant value  $n$  is equal to seven, then you can see input pattern, your constant input pattern you are feeding for 10024 cycles, then the power dissipations in two's complement, implementation is 1.97 mille watt, where as for sign-magnitude representation power is 2.25 volt, using the same supply voltage of three volt in both the cases.

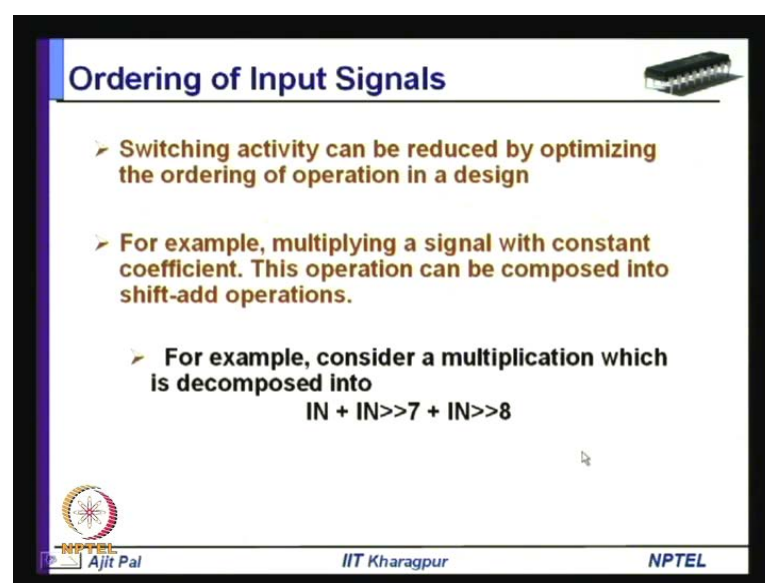
However, if you use a ramp, in case of ramp you are essentially, it is also correlated, you are increasing and decreasing. So, you are increasing from minus 7, here it is 0, then you are making it. So, here not like this, here minus 7 to 0, then plus 7. So, you are essentially ramp, so that means plus 7, then in this way you are doing. So, minus 7 to 0 plus 7 and you are changing in this way, you are using a ramp. Then again you are putting as well, minus 7, 0 plus 7, so this is how you are applying your data. So, in this



particular case, data is also correlated, it is not that it is not correlated, it is also correlated. So, when there is correlation as you can see, you do not get much benefit in sign-magnitude implementation, because of larger capacitance. So, here also there is no reduction in power dissipation, but there is increasing power dissipation in sign-magnitude realization.


On the other hand, when you have got random data, data is changing I mean it can have any value; any bit has the probability of transition of 0.5. In such a case it is a random data, and as a consequence you can see two's complement representation, realization based on two's complementation gives you 3.42 mille watt compared to 2.5 watt 51 mille watt, that you can achieve while using realization based on sign-magnitude form. So, this gives you reduction in the power dissipation, because of lesser switching activity, or lesser switched capacitance. Now here is a extreme case, here you are changing between say minus 7 to plus 7. So, you are changing minus 7 to plus 7, not this way minus 7, then plus 7, then minus 7, then plus 7 this way you are changing. So, in this case data you can say extremely anti-correlated. So, whenever it is highly anti-correlated, then you know we get good benefit using sign-magnitude representation as it is evident from this particular result. So, you find that for two's complement realization based on two's complement we get 5.28 mille watt compared to 2.46 mille watt, whenever you do sign-magnitude based realization.

(Refer Slide Time: 37:36)



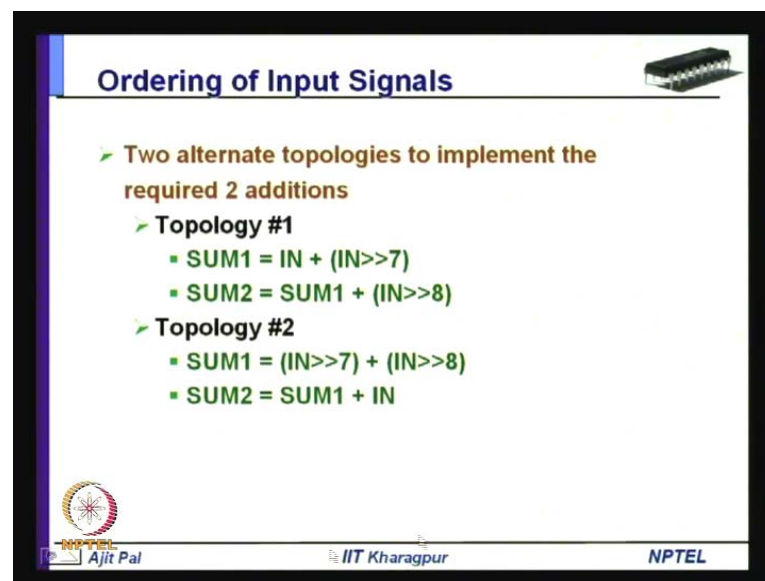
**Ordering of Input Signals**

- Switching activity can be reduced by optimizing the ordering of operation in a design
- For example, multiplying a signal with constant coefficient. This operation can be composed into shift-add operations.
- For example, consider a multiplication which is decomposed into  
$$IN + IN \gg 7 + IN \gg 8$$

 NPTEL  
Ajit Pal IIT Kharagpur NPTEL

Now, we switch gear we consider another technique; ordering of input signals, you know switching activity can be reduced by optimizing the ordering of operation. So, here what you are doing, for example multiplying a signal with constant coefficient, this operation can be composed into shift-add operations. So, sometimes you can reduce switching activity, by changing the ordering of operation. Let us consider a simpler example, where you have to perform, consider a multiplication in which it is decomposing to three additions. So, what you have done, you have to ultimately add three numbers; one is IN another is IN into 2 to the power minus 7, that can be achieved by shifting the data by 7 times towards right, that is that is effectively dividing by 2 to the power 7 and another is IN into shifting the data by 8 times towards right, essentially it is division by 2 to the power 8. So, these three data have to be added. Obviously, the magnitude of this number, which has been shifted 7 times towards right, or the number which has been shifted 8 times towards right, their magnitude will be much less.

(Refer Slide Time: 39:09)



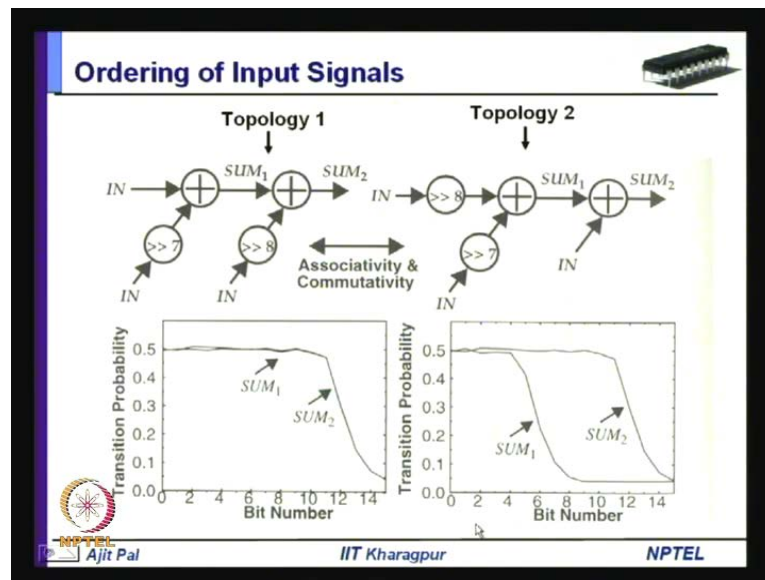
**Ordering of Input Signals**

- Two alternate topologies to implement the required 2 additions
  - Topology #1
    - $SUM1 = IN + (IN \gg 7)$
    - $SUM2 = SUM1 + (IN \gg 8)$
  - Topology #2
    - $SUM1 = (IN \gg 7) + (IN \gg 8)$
    - $SUM2 = SUM1 + IN$

NPTEL  
Ajit Pal  
IIT Kharagpur  
NPTEL

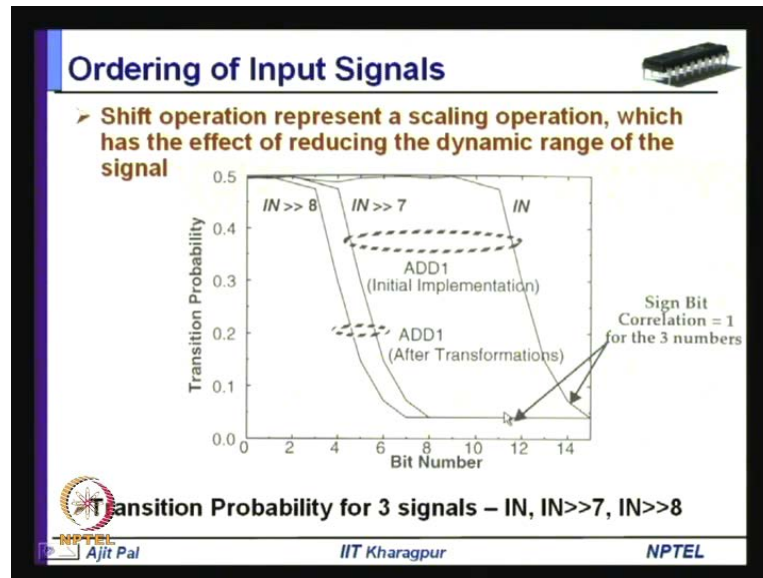
So, whenever you have to add these three data, you have got two possible topologies, in the first case, you will perform addition of the IN with IN into 2 to the power minus 7, and then you add with that partial result, you add with IN into 2 to the power minus 8. So, you can do in this way, in this order or you can do the addition, first you add the smaller numbers IN into 2 to the power minus 7, plus IN into 2 to the power minus 8, and then you add this intermediate result, with the in the initial value IN. So, we can do it in these three ways, and let us see the result.

(Refer Slide Time: 39:47)



So, in this case we are adding a large number with a small number, and again a large number with a small number, as a consequence you know you have to do, here you have to do sign-extension, here you have to do sign-extension to do the additions, and as a result the switching activity is quite high, as you can see, both at the output of sum one and sum two, but you can perform the same thing without losing the, without sacrificing correctness of data, you can use the property of associability and commutability, to perform the same operation by adding the two smaller number first, then adding it with the larger number. And as you do that, as you can see, since you are adding to smaller numbers, obviously the switching activity at the output of sum one will be much less. As you can see it is much less switching activity, particularly in the higher order bits, there will be no change at all, switching activity will be much less. On the other hand, of course here we are adding a small number with a large number, in that case switching activity will be comparable to the sum two of the previous situation. So, we find that at least at the output of one adder you are able to reduce the, switching activity significantly, by using this particular topology of implementation.

(Refer Slide Time: 41:15)



And here you know same thing, shown in a different way, transition probability and how the bit numbers are how the changing from different bit numbers. Here it is IN 7, IN 8, and you are doing addition of these two numbers, and you can see how the adder output reduces the switching activity that you have already seen, and whenever you perform addition of these two numbers, obviously the initial implementation, when you do addition, we using these two numbers switching activity was larger, because you are adding with a small number with a large number. So, sign bit correlation is one for different numbers, that correlation is shown for different values, obviously for smaller numbers there is larger correlation among the higher order bits, but whenever you are using, you know that smaller number and a larger number that correlation is different as you can see. So, transition probability for three different signals, I have shown here and based on that we got this result, this transition probability of this sum 1, sum two and for the output of these adders.

(Refer Slide Time: 42:38)

**Optimizing Resource Utilization**

- There are 2 choices for implementation
  - Time multiplexed architectures
  - Fully parallel architectures
- The degree of resource sharing should be optimized because
  - Resource sharing can destroy signal correlations and increase switching activity
  - For example, time sharing busses (output of 2 counters)

NPTEL Ajit Pal IIT Kharagpur NPTEL

Now, we shall consider another interesting situation, this is regarding optimizing the source utilization. You know there are two choices time multiplexed architectures, versus fully parallel architecture. So, you can have time multiplexed architecture, or you can have fully parallel architecture. This I can explain with an example of our microprocessors.

(Refer Slide Time: 43:14)

8-bit 8085 Intel

40-pin

MM

A<sub>8-15</sub>

A<sub>0-7</sub>

ALE

8088

Bus Multiplexed

Few MHz

8086 80286

80386, 80486

Pentium

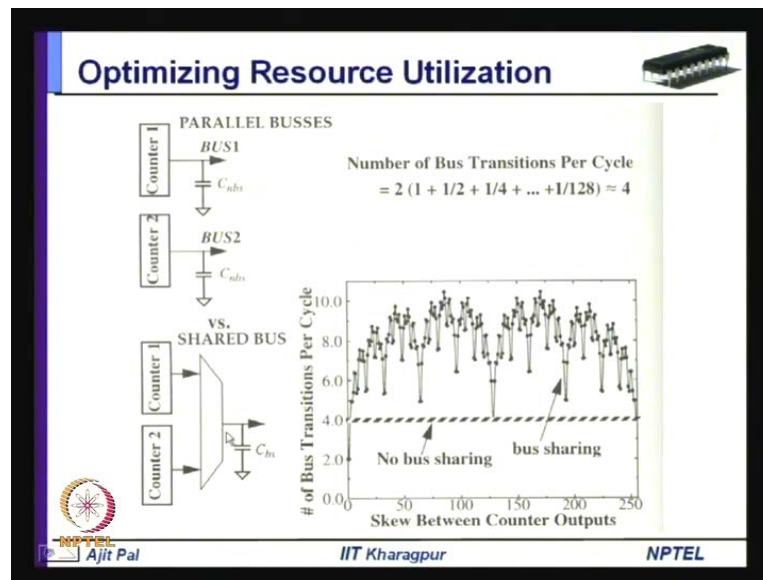
NPTEL

I do not know whether you have studied, that 8 bit puller microprocessor that is your 8085 by Intel. In this processor you may remember that, it had a 8 to 15 was coming out,

this is the higher order at this at this bus, but the data bus and lower order at this bus, so ADO to seven; that means, what does it mean. It means that same bus you are using for sending data as well as address, so that was being done in your 8085. So, we used to call it bus multiplexed. So, you are multiplexing the bus, so initially you are sending the data, which is being latched in the memory, normally you have got a memory here, main memory, so this lower order at. So, ALE signal was generated by the microprocessor that was latching the address fast in the main memory. Then the data you are reading using the same lines; that means, the lower order at this lines were sent using these lines first, then data were read or sent, using the same lines in the in subsequent cycles. So, a bus was time multiplexed.

Obviously, this had large number of pins, so number of pins was reduced by eight, and as a consequence you are able to realize the 8 bit microprocessor by using a 40-pin chip, and as you know the cost of a chip is dependent on the number of pins. So, at that point of time, when 8 bit micro, that 8085 was implemented, it was wiser to implement using this time multiplexed bus, but you know it has one drawback. Whenever you do that the switching activity is very large, on these lines, first you are sending address, then you are sending data, lot of switching activity will take place, and that will lead to large power dissipation. So, this time multiplex implementation mainly may have good resource utilization. You are using the same bus to communicate two different data in two different time instants or different cycles. Obviously, resource utilization is more, but the switching activity is much higher. The degree of resource sharing should be optimized, because resource sharing can destroy signal correlations and increase switching activity. For example, time sharing busses out of, I have already considered.

(Refer Slide Time: 46:03)



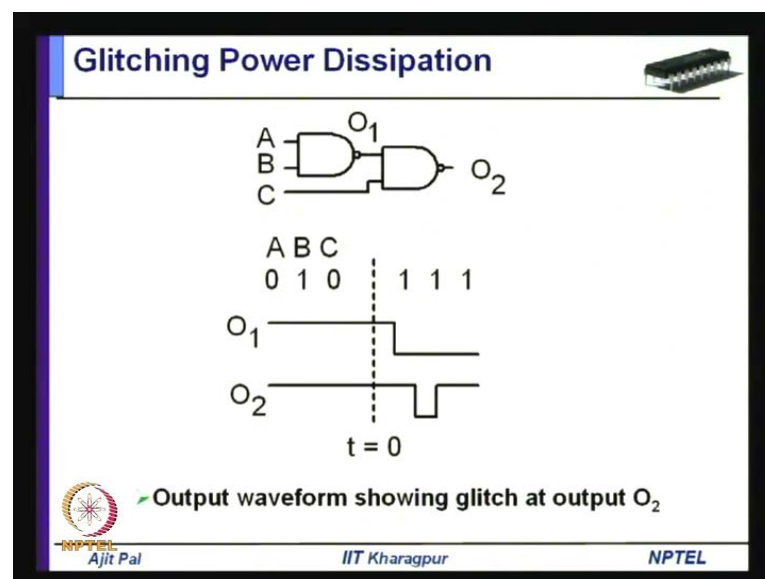
Let us consider this example; here you are taking output from two counters. These two counters may be, you know their counter counters may not be correlated; one is counting something, other is counting something, that count value may be in the module of the counter may be different, or the rate at which counting is take place can be different. Now you can use either shared bus, as it is shown here. In this case instead of using two busses, you are using a single bus; obviously, the capacitance will be less in this particular case,  $C_{bus}$  that is your, that is the bus over which you are doing, and first you are sending this, then you are sending this or you can have two separate busses. So, the number of bus transition per cycle is eight, the reason for that is you have got 8 bit data.

And since it is coming out from a counter, the number of bus transitions plus average value of the number of this transitions plus per cycling will be four, as you know count value will change from all zero to all one, and the average value will be four. Now, with this situation, with this background, let us see how the number of bus transitions per cycle changes, in these two different situations. So, whenever we use parallel busses then you can see, the number of bus transition per cycle remains fixed four, it does not change, because here you are taking from one counter, and here you are taking from another counter, and obviously the number of bus transition per cycles remains constant. I mean average number of bus transition per cycle remains constant that is four. However, whenever you do multiplexing then you can see, the number of transitions is little random in nature, and it can attend very high peak value, with minimum value of

four. So, you can see the switching activity on the shared bus or common bus is very large, and as a consequence in this particular case, we will have large switching activity.

So, you have to decide which particular system will use. For example, I was giving the example of this Intel microprocessor. So, in the earlier processors like 8085 and 8088 the bus was multiplexed, but subsequently as the frequency was increased. You know both 8085 and 8088 were operating in the range of few megahertz, as the frequency increased in subsequent processor, starting from 8086 to 80286, 80386, 80486 or in Pentium in subsequent processors no bus multiplexing was done, separate busses were provided, essentially to reduce power dissipation, switching activity, switch high switching activity, and also you know that increase the throughput, because you are sending over different busses. So, this particular technique, I mean this shows that why the bus were not multiplexed in subsequent processors.

(Refer Slide Time: 49:52)



Then, another technique that is your glitching power dissipation. Glitching power dissipation you have already discussed, that occurs because of delay of the gates, and I have already explained this particular, you know that the occurrence of this glitch, because of the delay of this gate that take place, and this glitch how can you reduce the glitching power dissipation.



(Refer Slide Time: 50:20)

### Minimizing Glitching Power Dissipation

- “Extra” transitions can be minimized by
  - Balancing all signal paths
  - Reducing logic depth

Realization of A.B.C.D in cascaded form

Balanced realization

NPTEL Ajit Pal IIT Kharagpur NPTEL


Glitching power dissipation can be reduced, by using balanced implementation instead of cascaded implementation. So, if you use cascaded balance implementation in this way, you know the delay here and delay here is more or less same, as a consequence at the output of this gate the switching activity will be much less. So, in this particular case there will be switching activity here and here O2 and O3, which will not be present here. So, extra transitions can be minimized. In other words the glitching power dissipation can be reduced by balancing all signal paths, and reducing logic depth, and also this reduces the logic depth. Reducing logic depth also reduces the delay of the critic that we know, that delay of that particular network. So, whenever you are realizing multilevel implementation of bullion functions, it is advisable to use balance circuit, and as much reduction in logic depth as possible.



(Refer Slide Time: 51:29)

### Logic Styles for High Performance and Low Power

- **Potential Logic Styles**
  - Static CMOS Logic
  - Dynamic CMOS Logic
  - Pass-Transistor Logic (PTL)

Ref: D. Samanta, Ajit Pal, *Logic Styles for High Performance and Low Power*, Proceedings of the 12th International Workshop on Logic and Synthesis, 2003 (IWLS-2003), pp. 355-362, May 2003




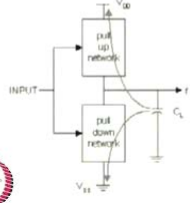
 NPTEL  Ajit Pal  IIT Kharagpur  NPTEL





Now, coming to the last topic, you know I have already discussed about the use of different logic styles; static CMOS circuit, dynamic CMOS logic style and pass-transistor logic. Nowadays you know most of the VLSI implementations are done by using static CMOS. Primarily because CAD tools are available, for matured CAD tools are available for based on static CMOS, unfortunately matured CAD tools are not available for logic realization, using dynamic CMOS or pass-transistor logic. So, one experimentation was done, by developing suitable CAD tool, by realizing dynamic CMOS and pass-transistor logic.

(Refer Slide Time: 52:16)

### Static CMOS Logic

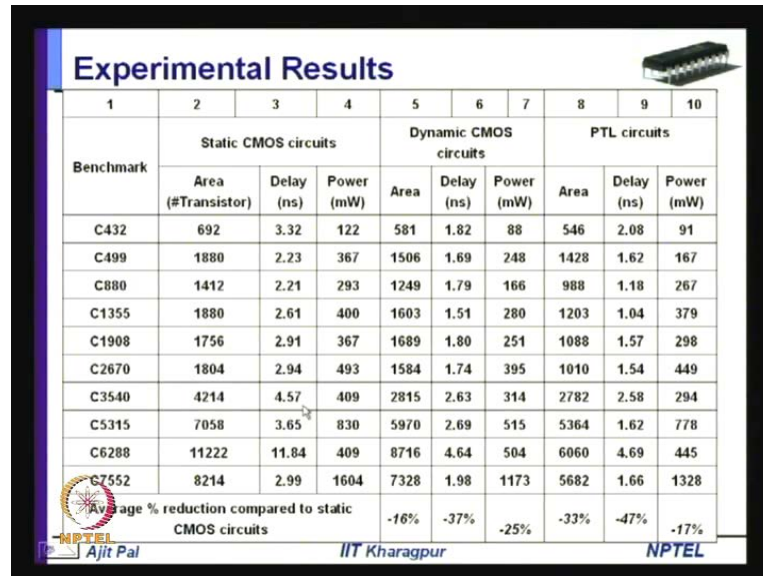
- **Advantages**
  - Ease of fabrication
  - Good noise margin
  - Robust
  - Lower switching activity
  - Good input/output decoupling
  - No charge sharing problem
  - Availability of matured logic synthesis tools and techniques
- **Disadvantages**
  - Larger number of transistors (larger chip area and delay)
  - Spurious transitions (glitch) due to finite propagation delays leading to extra power dissipation and incorrect operation
  - Short circuit power dissipation
  - Weak output driving capability
  - Large number of standard cells requiring substantial engineering effort for technology mapping



 NPTEL  Ajit Pal  IIT Kharagpur  NPTEL

I shall show you, I have already discussed the advantages and disadvantages of static CMOS dynamic CMOS. So, I am not going into the details at this moment.

(Refer Slide Time: 52:30)



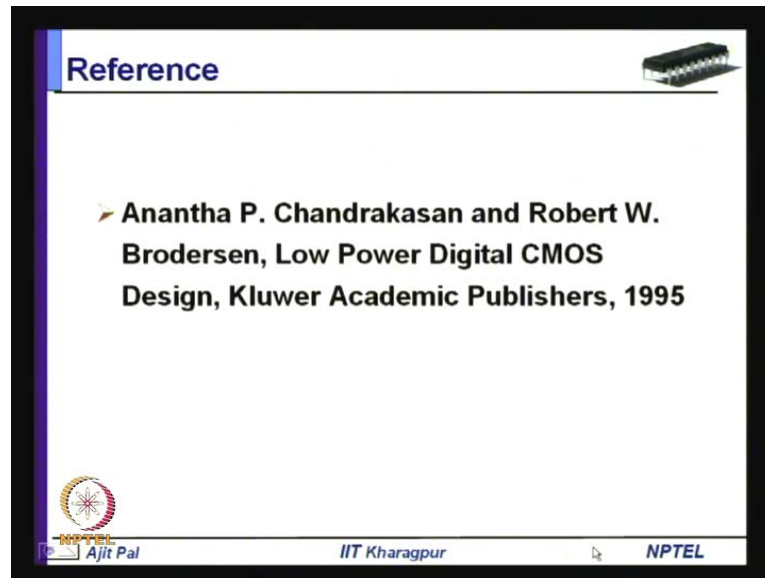
**Experimental Results**

Benchmark	Static CMOS circuits			Dynamic CMOS circuits			PTL circuits		
	Area (#Transistor)	Delay (ns)	Power (mW)	Area	Delay (ns)	Power (mW)	Area	Delay (ns)	Power (mW)
	C432	692	3.32	122	581	1.82	88	546	2.08
C499	1880	2.23	367	1506	1.69	248	1428	1.62	167
C880	1412	2.21	293	1249	1.79	166	988	1.18	267
C1355	1880	2.61	400	1603	1.51	280	1203	1.04	379
C1908	1756	2.91	367	1689	1.80	251	1088	1.57	298
C2670	1804	2.94	493	1584	1.74	395	1010	1.54	449
C3540	4214	4.57	409	2815	2.63	314	2782	2.58	294
C5315	7058	3.65	830	5970	2.69	515	5364	1.62	778
C6288	11222	11.84	409	8716	4.64	504	6060	4.69	445
C7552	8214	2.99	1604	7328	1.98	1173	5682	1.66	1328
Average % reduction compared to static CMOS circuits				-16%	-37%	-25%	-33%	-47%	-17%

NPTEL  
Ajit Pal  
IIT Kharagpur  
NPTEL

So, let us come to the final result, where we have implemented the same circuit same benchmark circuits, c 4 36 I mean these are discussed benchmark circuits, with you can see the number of transistors required in different cases, and you have implemented by using static c MOS dynamic c MOS and PTL pass-transistor logic. So, the tools CAD tools were developed at that automated implementation were done, to realize using static c MOS or dynamic c MOS or pass-transistor logic, and we find that the reduction in the area delaying power for dynamic c MOS, and here is the reduction in area delay and power in pass-transistor logic. So, you find that if you realize the same circuit, using dynamic c MOS there can be 16 percent reduction in area, 37 percent reduction in delay, and 25 reductions in power dissipation. So, reduction in energy which is power delay product is quite significant, may be more than 50 percent. Similarly, whenever you realize using pass-transistor logic, which I have already discussed in detail, you can see on the average 33 percent reduction in area take place, because pass-transistor logic realization requires a lesser number of transistors, as you can see the number of transistors required has been shown here, which is representative of the area and then the reduction in delay is 47 percent, using pass transistor logic and reduction in power dissipation is 17 percent.

(Refer Slide Time: 54: 16)



So, you find there is significant reduction in area delay and power. So, with this we have come to the end of today's lecture. And here is a reference the various techniques that you have discussed today, has been taken from a book by Anantha P. Chandrakasan and Robert W. Brodersen, the title of the book is low power digital c MOS design, published by Kluwer Academic Publishers; it was published some time in 1995. So, with this we have come to the end of on the various lectures on minimizing switched capacitance. In the next lecture we shall start our discussion on minimizing leakage power dissipation.

Thank you