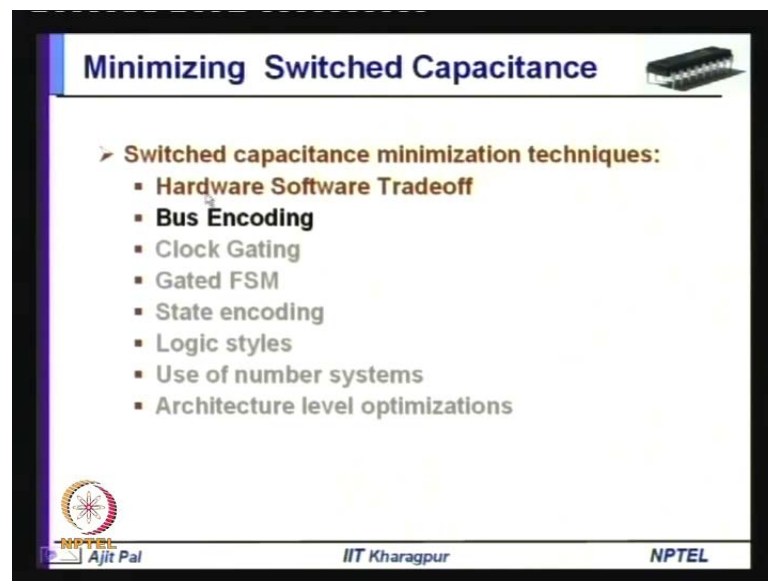


Low Power VLSI Circuits and Systems
Prof. Ajit Pal
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture No. # 28
Minimizing Switched Capacitance-II

(Refer Slide Time: 00:24)



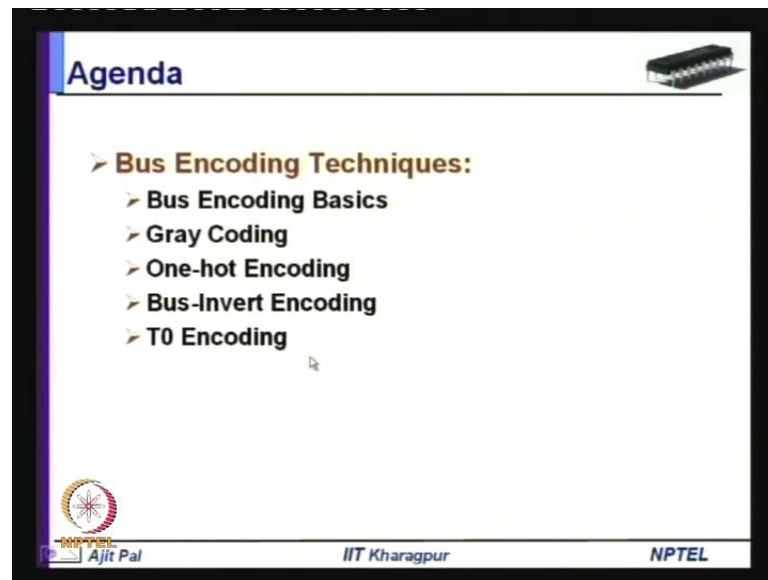
The slide is titled "Minimizing Switched Capacitance" and features a small image of a microchip in the top right corner. The main content is a list of "Switched capacitance minimization techniques" with the following items:

- **Switched capacitance minimization techniques:**
 - **Hardware Software Tradeoff**
 - **Bus Encoding**
 - Clock Gating
 - Gated FSM
 - State encoding
 - Logic styles
 - Use of number systems
 - Architecture level optimizations

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

Hello, and welcome to today's lecture on minimizing switched capacitance. We have started our discussion on various techniques for minimizing switch capacitance, and in the last lecture I have discussed about hardware, software tradeoff. And we have seen some functionalities can be realized by software, and by doing so you can minimize the switch capacitance. And we have illustrated that with the help of two examples AD convertor, and realization of processor architecture by using VLIW instead of super scalar.

(Refer Slide Time: 01:05)



The slide is titled "Agenda" and features a small image of a microchip in the top right corner. The main content is a list of topics under the heading "Bus Encoding Techniques:". The footer includes the NPTEL logo, the name "Ajit Pal", "IIT Kharagpur", and "NPTEL".

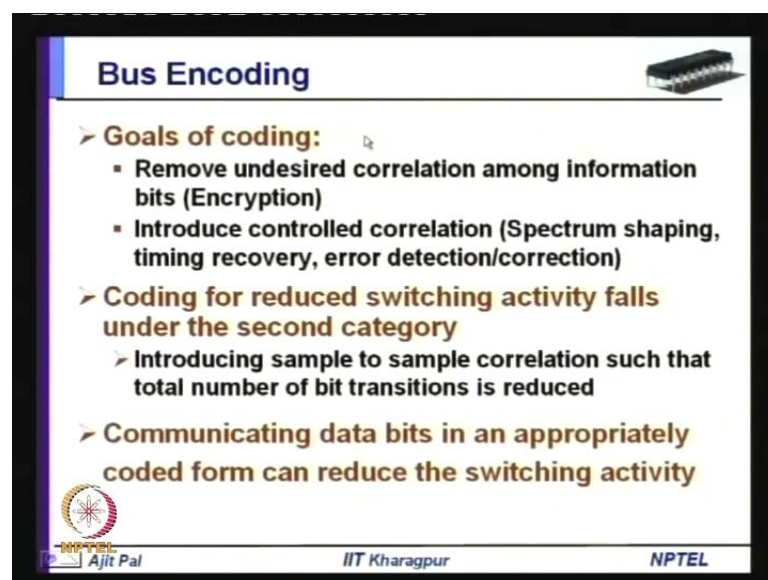
Agenda

- **Bus Encoding Techniques:**
 - Bus Encoding Basics
 - Gray Coding
 - One-hot Encoding
 - Bus-Invert Encoding
 - T0 Encoding

NPTEL
Ajit Pal
IIT Kharagpur
NPTEL

Today, we shall focus on bus encoding, and this is the agenda of today's lecture. First I shall introduce basic concepts of bus encoding, after that I shall discuss several popular encoding techniques like gray encoding, gray coding, one-hot encoding, bus-invert encoding, and T 0 encoding. So, these are some of the popular encoding techniques commonly used for minimizing switched capacitance.

(Refer Slide Time: 01:34)



The slide is titled "Bus Encoding" and features a small image of a microchip in the top right corner. The main content is a list of goals and coding techniques. The footer includes the NPTEL logo, the name "Ajit Pal", "IIT Kharagpur", and "NPTEL".

Bus Encoding

- **Goals of coding:**
 - Remove undesired correlation among information bits (Encryption)
 - Introduce controlled correlation (Spectrum shaping, timing recovery, error detection/correction)
- **Coding for reduced switching activity falls under the second category**
 - Introducing sample to sample correlation such that total number of bit transitions is reduced
- **Communicating data bits in an appropriately coded form can reduce the switching activity**

NPTEL
Ajit Pal
IIT Kharagpur
NPTEL

But before that let me consider very basic concepts of bus encoding; encoding is not a very new concept; it is being used for a long time particularly, in communication. In

communication we have seen it is used for various purposes for example, one approach or one goal of coding is remove undesired correlation among information bits, so you are sending information bits sequentially one after the other. So, whenever you are doing so, if you want to remove correlation then you can do some kind of encoding, and then correlation between or among various adjacent codes which are send one after the other is removed.

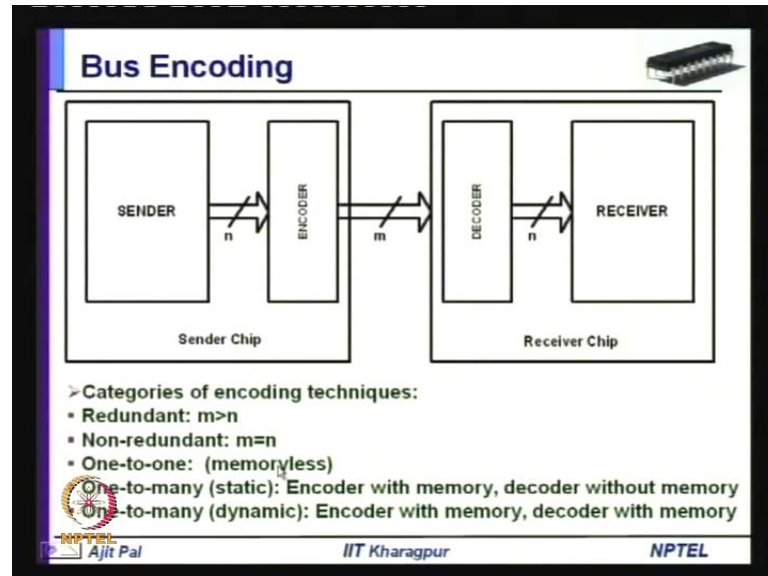
In fact, this is used **while used** in encryption, because whenever you are sending encrypted data the other side, or rather who are monitoring the bus or monitoring the communication medium should not be able to identify what is being sent. So, correlation among the data is removed, and correlation by doing so you cannot identify that this is this can be a particular bit pattern, particular information. So, this is used in encryption, and another type of encoding is used where coding is used to introduce controlled correlation, that means you are sending consecutive data where you are introducing controlled correlation for various purposes, one purpose could be spectrum shaping for example, you are sending some data through a communication media and medium. And whenever, you are sending that then you have to do this spectrum shaping such that the signal can be communicated with minimum attenuation.

So, that means the bandwidth of the medium is matched by spectrum shaping, I mean of the signal, this is one technique. Then timing recovery, sometimes it is used for the purpose of timing recovery for example, you are you are familiar with local area network, **local area networks** In Ethernet Manchester encoding is done, where it is primarily used for the purpose of timing recovery, and then sometimes it is used for error detections you are all familiar with the use of additional bits like parity, then cycle redundancy code, and by doing that you can detect error sometimes correct error.

So, you can see these are being done by introducing controlled correlation among the data that is being sent. And in fact, as we shall see the coding that we shall be doing, coding for reducing switching activity falls under the second category; that means, we shall be introducing controlled correlation whenever we shall be sending data, such that the switch capacitance is minimized that is the basic idea. Now, this will by introducing sample to sample correlation, such that total number of bit transitions is reduced, so this is the basic idea behind this bus encoding for low power. And whenever, you will be

communicating data bits in an appropriately coded form that will reduce the switching activity; that means, the switching activity can be significantly reduced by doing that.

(Refer Slide Time: 05:16)



Now, let us consider what do you really mean by bus encoding? You are having a one particular say sender, let us call me sender chip sender, it can be a chip, it can be a device, it can be a micro processor, it can be a micro controller, and you have got another chip let it be called receiver, and the signal is going out of this chip and that is being send to the receiver.

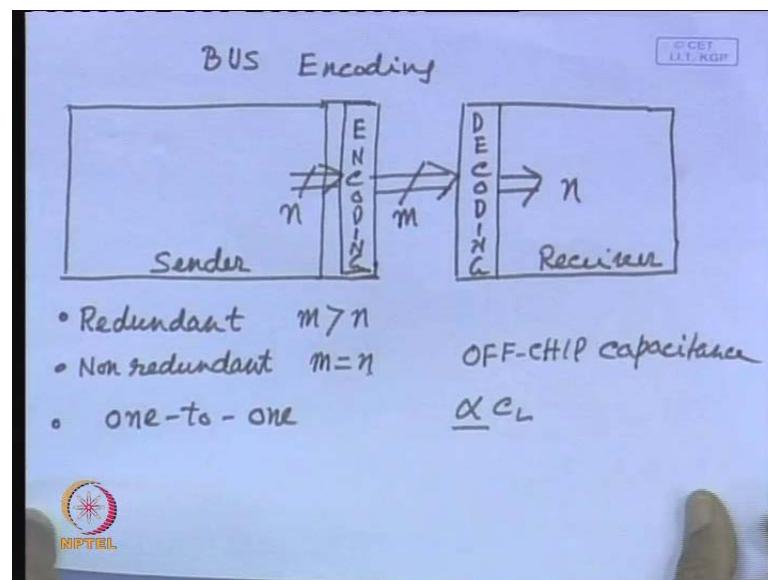
So, whenever you are doing so, it has been found that the capacitance outside the chip that is called at off-chip capacitance **off-chip capacitance** has been found to be very high. Why it is high? We know that internal to the chip various nodes have capacitance of the order of famtofarad, but whenever you take it outside the chip with the help of bonding, bonding varying, padding and so on, take it to the pin then the capacitance becomes of the order of picofarad. So, we can see it increase the significant several order of magnitude higher. And obviously, the dynamic power dissipation is proportional to $c l$, and as the capacitance is very large you have to minimize switching activity, so switching activity has to be minimize, so that the dynamic power dissipation is reduced.

Now, in the particular case how you will be doing it? You will be in the sender, you will be doing the encoding before sending that will become part of it, so here you will do

encoding and then after it is sent, after it is received by the receiver it will do decoding to get back the **to get back the** data.

Now, when the sender is sending, then it may be sending say from this side, n bits, and the decoder may convert it to m bits, and then the encoder may convert it to m bits, and the decoder on the other side again it will convert it to n bits. Now, based on this the encoding technique can be categorized into a number of types for example, there are two possibilities,

(Refer Slide Time: 05:22)



it can be a redundant encoding. In which case m is greater than n , that means the number of bits that is being send over the medium, or which is send outside the chip, or sometimes it can be part of a system on chip, now a days you are familiar with the use of I mean, familiar with the system on chip. In a single chip you can have a number of different subsystems, say this can be one subsystem, this can be one subsystem, this can be one subsystem, this can be one subsystem or it can be a multi-core.

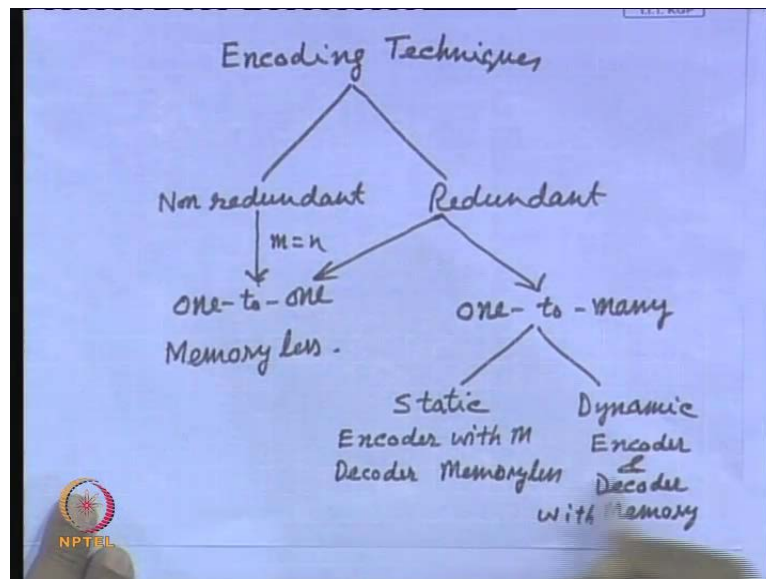
So, in such a case this particular subsystem they may communicate with other subsystem and you can have a common bus through which communication may be done; that means, these are all connected to the bus and that is how communication can be done. This is one alternative, another alternative that you can put a switch through which they can be communicating. Whatever maybe the case, whenever it is going from one

subsystem going out to the another subsystem, again the same problem will arise; that means, capacitance will be large.

Now, whenever you are doing that you can use two types of coding as well as telling, one is your redundant coding where m greater than n , another possibility is non redundant coding. In case of non redundant coding m is equal to n , that means the number of bits that is being sent through the bus, or through the switch whatever it may be, or through the communication medium can be same as the number of bits of the sender. So, in this case the number of bits is not increasing; that means, there is no increase in redundancy there is no redundancy, so these are the two possibilities

Now, in addition to this categorization another possibility is there, it can be have it... you can have different type of mapping, say you can have one to one mapping, say let me do the categorization,

(Refer Slide Time: 10:48)



Now, you may have one too many and there are two say encoding techniques. So, you have two broad categories, number one is non redundant, another is redundant. So, whenever it is non redundant obviously, it will be always be it will always be one to one. What do you really mean by that? That means, a particular code n bit code will always

map to another m bit code, so when m and n are same we call it one to one; that means, a particular code here will map to another code here in a unique way, there is no possibility of one to many, whenever you are having non redundant coding; that means, m is equal to n .

Now, whenever you are having redundant coding then you have two possibilities; one is your one to one, another is your one too many. So, whenever you are having redundant coding, then what you can have a particular code can uniquely map to another code or it may not map to another code uniquely, but depending on the past history the code to which the map can be different. And as a consequence, one to one coding implementation does not require memory; you may say these are all memory less, neither in the encoder or in the decoder. So, one to one encoding does not require any memory for the implementation of the decoder or encoder possibilities, 1 to many can be static or 1 to many can be dynamic, when 1 to many is static then encoder requires memory encoder with m stands for memory, and decoder memory less then we call it static.

On the other hand in case of dynamic encoding both encoder and decoder will require memory, with memory. So, we can see this is how we can categorized the encoding techniques into different types, so we have non redundant then redundant then one to one, one too many; that means, one code will map to more than one codes depending on the past history, and in this case you will require memory as you can see either in the encoder or in case of both. Now, with this with this classification we shall now move to different examples, different types of encoding that I have examples that I have already mentioned.

(Refer Slide Time: 14:19)

Gray Coding Vs Binary Coding

➤ A gray code sequence is a set of numbers in which adjacent numbers have only one bit difference

| Decimal Value | Binary Code | Gray Code |
|---------------|-------------|-----------|
| 0 | 0000 | 0000 |
| 1 | 0001 | 0001 |
| 2 | 0010 | 0011 |
| 3 | 0011 | 0010 |
| 4 | 0100 | 0110 |
| 5 | 0101 | 0111 |
| 6 | 0110 | 0101 |
| 7 | 0111 | 0100 |
| 8 | 1000 | 1100 |
| 9 | 1001 | 1101 |
| 10 | 1010 | 1111 |
| 11 | 1011 | 1110 |
| 12 | 1100 | 1010 |
| 13 | 1101 | 1011 |
| 14 | 1110 | 1001 |
| 15 | 1111 | 1000 |

NPTEL Ajit Pal IIT Kharagpur NPTEL

First we shall consider gray coding; gray coding you may be familiar with this. In gray coding you may start with **say you may** say that, this is let us assume this is your binary code and this is your gray code, you may start with one bit the binary value may be 0 or one, so whenever it is single bit the gray code will also have single bit 0 1, so in this case when it is one bit code there is no change now whenever we go to two bits then you have got 4 possible alternatives that means 0 0 0 1 1 0 and 1 1, so these are the binary codes using two bits.


What will be the gray code? In case of gray code what you have to do, you have to take mirror image that means these bits will be 1 0, and the other bits will be same more significant bits will be same. So, the code will be 0 0 1 0 0 1 1 1 0 that means, 0 0 will map to 0 0, 0 1 will map to 0 1, 1 0 will map to 1 1 and 1 1 will map to 1 0.

Now let us suppose we are interested in increasing another bit, so you will having 0 0 0 0 then you will simply repeat this 0 0 0 0 **sorry** 0 0, now it will be 1 1 0 0 1 0 1 1 1 0 then 1 1 1, that means this part is repeated. Here what you have to do instead of repeating, you have to make again mirror image that means, it will be 1 0 it will be then it will be 1 **1** then it is 0 1 then it is one 1 1; that means, here you have to 0, and here you will be adding one, this is how the mapping will be done correspond to corresponding bits.

So, you see what is the interesting property of this bit of these codes you can see any if you take any adjacent code you will find there hamming distance is only 1; that means,

Gray Coding Vs Binary Coding

- It is useful when the data is sequential and highly correlated, like Instruction addresses
- No of bit transitions is limited to one for sequential data
- For random data, the no of transitions for binary and gray code are approximately equal

 NPTEL
Ajit Pal IIT Kharagpur NPTEL

And as you can see here, it is very useful when the data is sequential and highly correlated like instruction addresses. As you know whenever you are writing a program, a program is nothing but a sequence of instructions, how do you keep the instructions in the memory, you normally keep the instructions in the memory in adjacent memory locations unless it is a Drunch, jam, Subroutine or interrupt.


Normally, the instructions are kept in 1 after the other, I mean are in consecutive memory locations 1 after the other; that means, whenever you are you will be fetching instructions from the memory in other words, when the c p u will fetch instructions from the memory for the purpose of execution on the address bus they will come in consecutive order.

So, in that case instead of coding by using this binary code, if you code by using gray code obviously the number of transitions will be much less because as you go from this code to 1 code only 1 transition from this to this, another 1 transition from this to this, another 1 transition but this is not true whenever it is binary coded as you can see here, it is from here to here it is 1 bit transition but from to here to here it is 2 bit transition from this to this, it is 1 1 bit transition but from this to this 1 2 3 bit transition; that means, the number of transitions can vary from 1 to 4 when it is a 4 bit code but in case of gray code it is always 1 if it is consecutive.

So, this is the task of the matter for gray coding and; that means, it is useful when data is sequential and highly correlated, by highly correlated means we are sending, stored in consecutive memory locations, obviously the addresses will differ by fixed value; that means, depending on the size of the address bus and as know memory is addressed by in terms of bytes, and as a consequence they will differ by say either 4 byte or 8 byte address and that value will be fixed, and whenever you have that kind of correlation then the number of transitions is limited to 1 for sequential data.


However, when for random data the numbers of transitions for binary and gray codes are approximately equal. So, if the code that receive the data is not correlated that they are random in nature, so in such a case you do not get this kind of benefit; that means, the number of transitions of only one in case of gray code, so this has been illustrated with the help of a simulation,

(Refer Slide Time: 20:46)

Comparison of the temporal activity 

| Benchmark Program | Instruction Address | | Data Address | |
|-------------------|---------------------|------------|--------------|------------|
| | Binary Coded | Gray Coded | Binary Coded | Gray Coded |
| Fastqueens | 2.46 | 1.03 | 0.85 | 0.91 |
| Qsort | 2.64 | 1.33 | 1.32 | 1.25 |
| Reducer | 2.57 | 1.71 | 1.47 | 1.40 |
| Circuit | 2.33 | 1.47 | 1.33 | 1.18 |
| Semigroup | 2.68 | 1.99 | 1.38 | 1.34 |
| Nand | 2.42 | 1.57 | 1.25 | 1.16 |
| Boyer | 2.76 | 2.09 | 1.76 | 1.72 |
| Browse | 2.51 | 1.64 | 1.32 | 1.40 |
| Chat | 2.43 | 1.54 | 1.32 | 1.20 |

Bit transitions per instruction executed
 Ref: Chandrakasan & Broderon, Low Power Digital CMOS Design, KAP


Ajit Pal
IIT Kharagpur
NPTEL

I shall show about that as you can see here, there is a comparison of the temporal activity. So, these are some of the here you have got a number of Benchmark Programs, Fast Queens, Quick sort, Qsort, Reducer and so on. These are some Benchmark Programs which have been run on a processor, and as you can see on the instruction bus, when the addresses are coded in binary then the number of bit transitions per instruction executed, suppose a program requires 1000 instructions and total number of transitions on the address bus is counted and that is divided by the number of instructions to get the

number of bit transitions per instruction, and that is being compared you can see in case of address bus.

So, that is called instruction bus or address bus. So, you can see in case of binary coded form of the address we find the number of bit transitions per instruction is 2.46, 2.64 and so on. On the other hand when it is gray coded it is reduced to 1.03, 1.33 almost half, so there is a significant reduction in the number of transitions and there will be consequent saving of dynamic power, and this has been taken from Chandrakasan and Broderon's book on low power digital CMOS design published by Kluwer Academic Publisher.

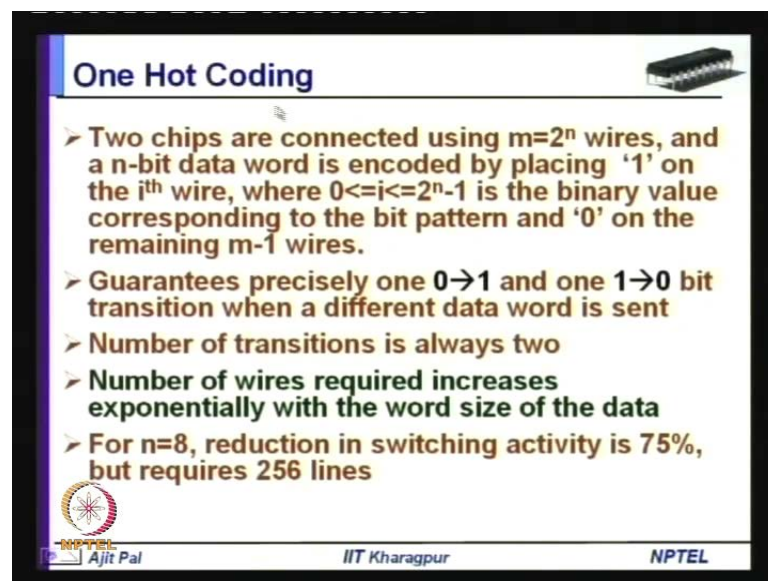
However, this kind of advantage you do not get on the data bus, because data is not correlated, data is can be random you can store some random values in your memory as data and so there is no correlation in the data and you can see the number of bit transitions per address, per instruction is not reduced, sometimes rather it is increased for example, for First Quench number of bit transitions per instruction is increasing slightly compared to binary coded data. So, you can see this particular approach gray coding can be used only when they are the data is highly correlated as it is true in case of instruction bus, instruction fetching or other instruction address bus where you fetch the instructions one after the other.

Now, how do you really implement them? Implementation is rather easy, you can see here you can code in a very simple way, I mean do not require very complex hardware, you will require only $n - 1$ $n - 1$ to 1 exclusive or gates for encoding as well as $n - 1$ exclusive or gates for decoding. So, in both cases you require $n - 1$ exclusive or gates, and because the b_n is g_{n-1} is equal to b_{n-1} , and g_{n-2} is equal to b_{n-1} exclusive or b_{n-2} and so on, so g_0 is equal to b_1 exclusive or b_0 , this is how you can get the various Gray Codes and similarly, you can get back the binary codes in this way, g_{n-1} is equal to b_{n-1} and b_{n-2} is equal to g_{n-1} exclusive or g_{n-2} and so on, so in this way you can get back the binary codes.

So, from this you can make this conclusion that Gray Coding is a non redundant code, because the number of bits is not changing, here you have got n bits here also you have got encoded code also has got n bits, so the number of bits is same so it is a non redundant code. Moreover, here you are doing one to one mapping as you have seen you

have got a 1 to 1 correspondence between Binary Coding and Gray Coding, so it is a 1 to 1 coding, 1, to 1 coding, and as a consequence you do not require and memory for the implementation of either the encoder or the decoder. So, as it is clear from this particular diagram they can be implemented simply by combinational circuit in this particular case exclusive or gates. So, this is Gray Coding, now we shall consider another very interesting code that is One Hot Coding

(Refer Slide Time: 25:25)



One Hot Coding

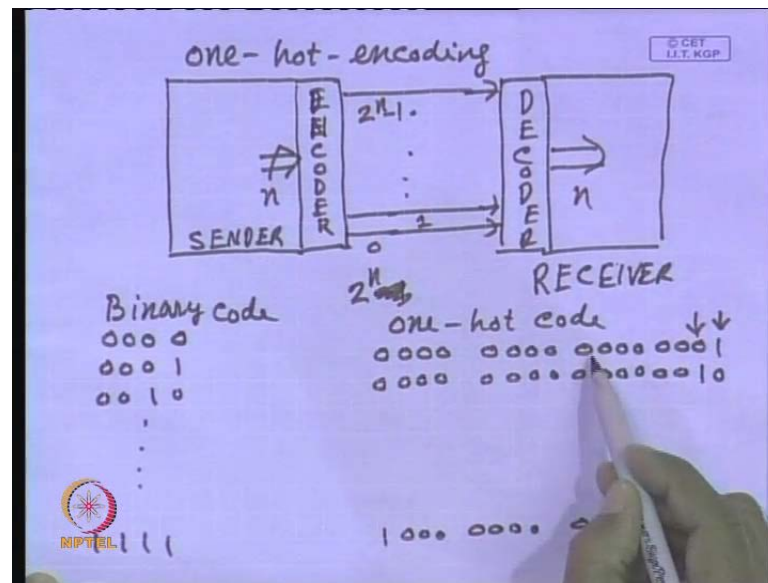
- Two chips are connected using $m=2^n$ wires, and a n -bit data word is encoded by placing '1' on the i^{th} wire, where $0 \leq i \leq 2^n - 1$ is the binary value corresponding to the bit pattern and '0' on the remaining $m-1$ wires.
- Guarantees precisely one $0 \rightarrow 1$ and one $1 \rightarrow 0$ bit transition when a different data word is sent
- Number of transitions is always two
- Number of wires required increases exponentially with the word size of the data
- For $n=8$, reduction in switching activity is 75%, but requires 256 lines

NPTEL Ajit Pai IIT Kharagpur NPTEL

So, in this particular case what you are doing, say you have got one particular device maybe sender, then you are doing encoding and here you have got decoder and you have got your receiver, so this is you can say this is your receiver. In this case the n bit code that is coming from the sender is converted into 2 to the power n minus 1 bits on after the decoding; that means, decoder is converting to 0 1 1 in this way finally, it is 2 to the power n minus 1 .

So, you have got 2 to the power n minus 1 lines after decoding and decoder **sorry** here it is encoder this is decoder. So, decoder again will convert it back to n bits, how it is working. Let us consider the basic idea behind this One Hot Encoding, let us consider 4 bits say 0 0 0 0 0 0 1 0 0 1 0 , in this way you can have 1 1 1 1 4 bits this is the binary code. The 1 hot code will have 16 bits, 2 to the power 4 minus 1 ; that means, the number of it will be two to the power n 0 2 to the power n minus one.

(Refer Slide Time: 25:36)



So, the number of codes is equal to number of lines will be equal to 2 to the power n number of bits, not 2 to the power n minus 1, 0 to 2 to the 2 to the power n minus 1 that is 2 to the power n; that means, in this case it will be 16 so in here it will be 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 and then 0 0 0 1 because it is the 0 0 0 0 in this case the zeroth bit is Hot or 1, for 0 0 0 1 the corresponding code will be 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 so the next bit is Hot or one.

So, in this way you go you can go on generating the code and for 1 1 1 1 it will be 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 and 0 0 0 0, so in this particular case one important point that you have to notice is that, only one bit is one and the hamming distance between any pair of code is two, always two. So, that is the important characteristics of this One Hot Encoding and as a consequence what is happening, two chips are connected using m is equal to 2 to the power n wires, so you will require two to the power n wires and an n bit data wire is encoded by placing one on each of the ith wire as you have already seen, and where 0 ith less than 2 to the power n minus 1 is the binary value corresponding to the ith bit pattern, and 0 is the remaining number of 1 wires as we have already explained illustrated with the help of this; that means, 1 of the bits will be 1 and remaining bits will be 0 and value will be here in this case, it is here it is 2 the next one will be 4 8 and so on.

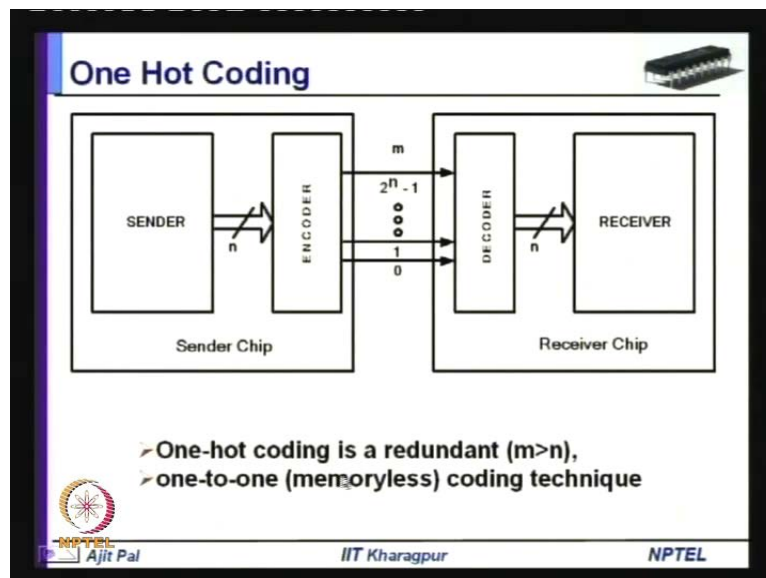
Now, this guarantees precisely 1 0 to 1 and 1 1 to 0 bit transition when two different data word is sent; that means, whenever you are sending any pair, so here you see if **the** this

was the earlier 1 then you are sending this 1 to 0 1 transition 0 to 1 another transition, this is applicable to any pair of code. So, only two transitions will occur number of transition will always two, the number of wires required increases exponentially with the word size of the data.

So, this is a very discouraging situation because the number of wires as you can see is increasing exponentially, so although this reduces the switching activity significantly, the number of wires increases exponentially as the number of bits increases; that means, it can be used for small values of n, not for large values of n

For n is equal to 8, the reduction in switching activity is seventy-five percent so you can see you can have 75 percent reduction in switching activity but unfortunately it requires 256 lines, so n is equal to 8 and m is equal to 256.

(Refer Slide Time: 31:10)



So, you see this is how the n number of lines increases and this is how you can implement one hot encoding, there is an encoder which generates two to the power n lines and decoder converts it back to n lines. So, obviously One Hot Encoding is redundant and also it is a 1 to 1 encoding. So, this is an example of I have already explained that 1 hot encoding, that redundant coding can also be can also have 1 to 1 1 to 1 coding, so this is the example of redundant coding with one to one mapping; that means, this one 1 hot encoding is an example of this, and here it do not require any

memory for its implementation because it is a 1 to 1 coding. So, this is one hot encoding that you have discussed in detail.

(Refer Slide Time: 32:05)

Bus Inversion Coding

- It is a redundant coding scheme where $m=n+1$
- If the i^{th} data word is S_i , then either S_i or $\sim S_i$ is transmitted depending on which would result in fewer no of bit transitions
- An extra bit P encodes the polarity of the data word

NPTEL Ajit Pal IIT Kharagpur NPTEL

Now, let us consider another encoding which is very popular Bus Inversion Coding, so bus in case of bus inversion coding what is being done, the i^{th} data word let us assume it is S_i , then you send either S_i or \bar{S}_i , depending on which would result in fewer number of transitions.

(Refer Slide Time: 32:34)

© CET I.I.T. KGP

$n\text{-bit } S_{i-1} \implies$ Hamming Distance between S_{i-1} & S_i

$n\text{-bit } S_i$
 \bar{S}_i

| | | |
|-------------|----------|---|
| S_{i-1} | 10110011 | |
| S_i | 11001101 | 6 |
| \bar{S}_i | 00110010 | 2 |

IF HD is greater than $n/2$, then send \bar{S}_i

IF HD is less than or equal to $n/2$, then send S_i

NPTEL

So, in this case say S_i minus one was the code that was the data bits that was sent, now you are getting S_i , so what you will do? you will compare the hamming distance between these two codes; that means, the this is the data bits obviously it will have n bits this will also have n bits, and this n bit data is compared with n bit data, and hamming distance is found out. And now this was sent earlier over the bus, let us assume now you have to send you will you have to send S_i , and you have got now two choices, you may send either S_i or you will send $\overline{S_i}$ when, what you will do you will find out find out the hamming distance between S_i minus 1 and S_i .

So, if hamming distance H_D is greater than $n/2$, that means if the hamming distance is greater than $n/2$ what you will do; that means, the if the if you send this then if you send it, the number of bit transition will increase; that means, the hamming distance is greater than $n/2$, in that case then send $\overline{S_i}$. On the other hand if hamming distance is less than or equal to $n/2$ then send S_i . Let me illustrate with the help of a very simple example, let us consider you are sending 1 0 1 1 0 0 1 1, so this is the, this is S_i minus one.

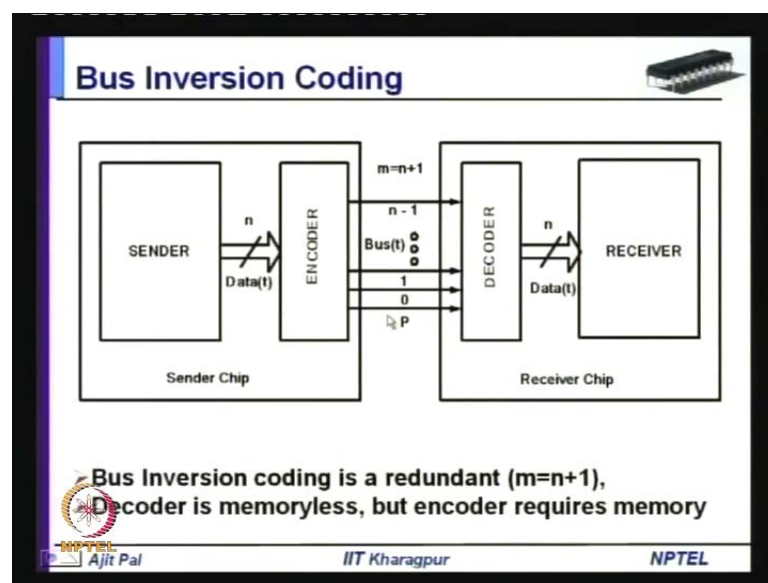
Now, suppose you are you are let us consider an example where hamming distance is greater than $n/2$, let us assume the next data S_i is 1 0 1 0 0 1 1 0 0, so in this case what is the hamming distance 1 2 3 4 5 and 6 so hamming distance is 6; that means, if you send this one after this what will happen, number of bit transitions on the bus will be 1 2 3 4 5 and 6.

So, it is profitable, more profitable to send $\overline{S_i}$ and in the $\overline{S_i}$ is 1 0 1 1 0 0 1 1, so in such a case hamming distance between this 1 and $\overline{S_i}$ is you can see it is only, it is becoming same, no 0 1 0 0 unfortunately, I have taken the compliment let me take a let, let me make it 1

So, in this case 1 1 0 0 1 1 0 0 so the hamming distance is 1 2 3 4 5, 1 2 3 4 5 so let us make it 1, so 1 2 2 3 4 5 6 in this case it is 6, 1 2 3 4 5 and 6. So in this case if you take the compliment of it, it will be 0 1 0 1 1 1 0 then it is 1 0, so now you compare the hamming distance between this two you will find 1 then 2 3; that means, if you send this the number of transitions is 6, on the other hand if you send this but it is 3, it should be 2 1 0 is 1 1 0 1 1 so 1 1 so it will be 0 0, so that means here it is 2.

So, we find that if you send S_i bar then the number of transition is becoming two, so it is profitable to send S_i bar two that that is how by comparing the hamming distance between the previous data, and the present data you can find out whether you have you have to send S_i or S_i bar, that is the basic idea of this Bus Inversion Coding. **So, it is a redundant**, Now the question arises how the other side will be knowing that inverted value of data has been sent not the original data, so you have to transmit one additional line to inform the receiver that the data that is being sent is inverted, and that is the reason why an additional bit known as polarity bit is needed added with the data.

(Refer Slide Time: 37:57)

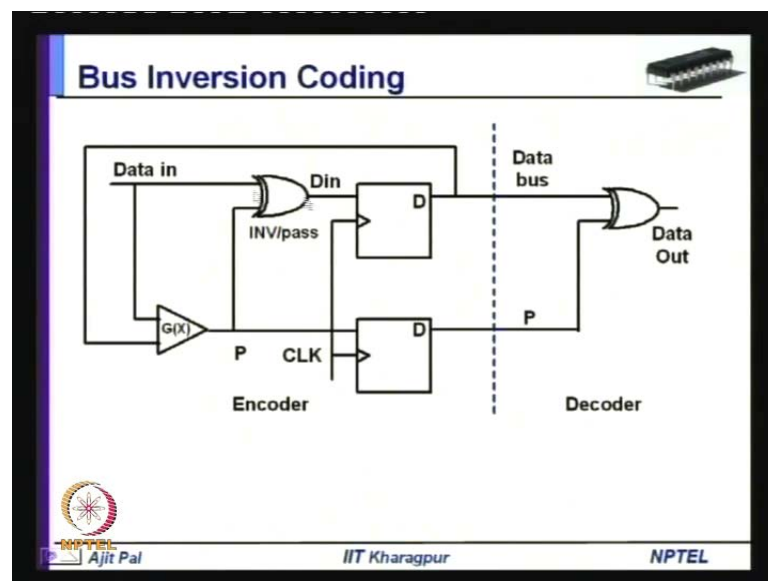


So, this is the example this is how it is being implemented you have the sender, and you have the encoder here which converts the n bit data into n plus one bit including this polarity bit. So, including this polarity bit the number of bits is n plus one so there is redundancy but redundancy is not much only one additional bit is added for transition over the Bus and decoder will convert it to n bit.

Now, in this particular case how do you implement the encoder and decoder? In this case decoder is memory less but encoder will require memory. Why encoder will require memory? the reason that encoder will require memory because you can see when you are transmitting S_i or S_i bar you have to preserve S_i minus 1, and compare with S_i so the code that you will be sending is dependent on the past history what was sent earlier, and you have to compare with respect to that whether the hamming distance is more or I

mean is increasing or decreasing by sending S_i and accordingly you will be taking decision, and as a consequence the encoder will require memory. So, as it is mentioned here encoder will require memory however, your decoder will be memory less, the reason for that is in the decoding side with the help of the polarity you can easily find know that you have to invert the data bits or you have to take it as it is, so that can be very easily found out from the parity bit.

(Refer Slide Time: 39:41)



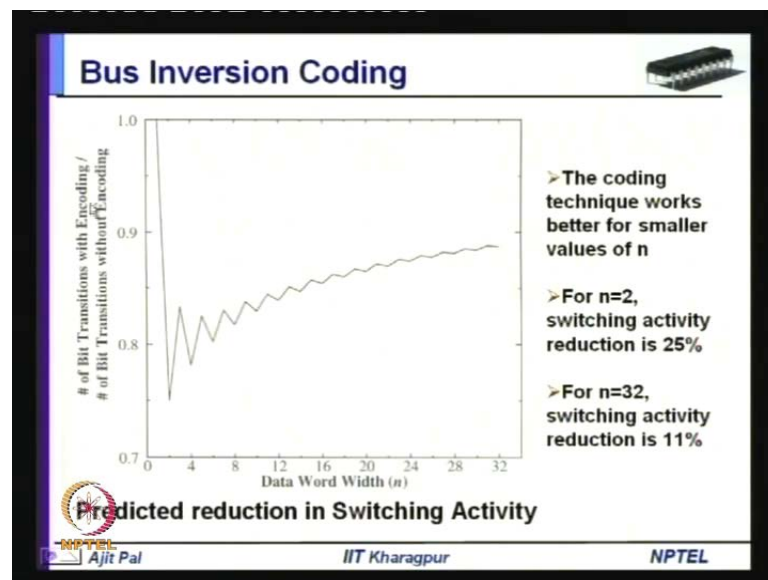
So, that the way it can be implemented is shown here so you can see this is the data that is coming, and you have the memory elements as I have told where you are preserving the previous value, which is being compared with the present data bits and after comparing these $G(X)$ which is computing the hamming distance or other the polarity that is being sense depending on the hamming distance n is greater than n , by two or less it is finding out the value of p , and that p is being used to transmit the inverted data or the original data, and so you are performing exclusive or to decide whether you are complementing it or not similarly, the polarity bit is also sent through this memory element and it goes to the other side, and you can see this is the decoder side and the decoder side does not require any memory element because it can simply implement it can be simply implemented with the help of n exclusive or gates.

So, here only one bit is shown obviously, if your data is n bit you will require n exclusive or gates to implement the decoder, and encoder will not only require n exclusive or gates,

n class one, flip flops, but a complex combinational circuit to compare the present data and the previous data that was sent over the Bus and find out **the**, that whether the hamming distance is greater than n by 2 or less.

So, you can see encoder is with memory and decoder is without memory, so this is the example of that; that means, I have already told whenever you are having one too many, in this case it can map to 1 of the two values, complemented value or un-complemented value, and accordingly encoder is with memory and decoder memory less, this is the example of this particular situation where you have got 1 too many encoding and using dynamic situation, static situation, so this is your Bus Inversion Coding.

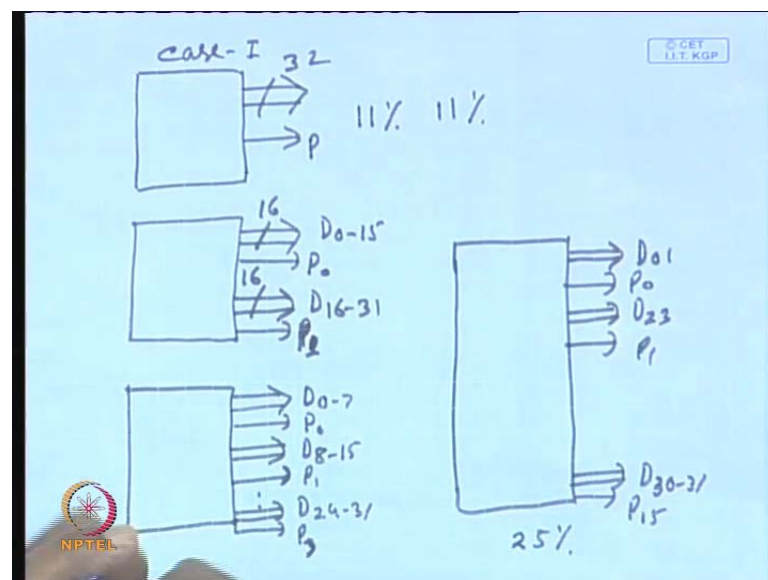
(Refer Slide Time: 42:03)



Now, we shall consider another very interesting example, before I move to another example there are some important features of Bus Inversion Coding, the efficiency is dependent on the value of n , and efficiency is maximum as we can see the number of bit transitions by the number of bit transitions without encoding; that means, number of bit transitions with encoding by number of bit transition without encoding that has been computed for different value of n . And here as you can see the these value is minimum, this ratio is minimum for n is equal to 2, and as the value of n increases this particular value increases, to state in simple terms you can see here when the value of n is equal to two the switching activity reduction is 25 percent.

So, this is 25 percent reduction in switching activity, so it is 0.75 so there is twenty-five percent reduction in switching activity, on the other hand when n is equal to 32 some are here, we can see the switching activity reduction is only 11 percent because here it is eighty-nine, so it is eighty-nine so 11 percent. So, what does it really mean? That means, the whenever you are doing Bus Inversion Coding the efficiency is dependent on the number of bits, and it is maximum when n is equal to 2; that means, you get maximum reduction in switching activity and, you can have and the as the number of bits increases the switch, the efficiency degrades. In such a situation you have got a tradeoff, what is the tradeoff.

(Refer Slide Time: 43:54)



Suppose you are sending 32 bit, so case one, say you can send 32 bit including one parity bit p . So, in this case you will have a reduction of about 11 percent in switching activity, now you have got several alternatives you group the bits into 2 parts, so here you have got 16 and here you have got another 16 and obviously, you will require two polarity bits as p_0 and p_1 , say it is d_0 to 15 and it is d_{16} to 31, so here you have got 2 additional polarity bits so you are increasing the redundancy by grouping into two types.

So, in this way you can have third alternative where you can group into 4 different components, so here you have got d_0 to 7 p_0 then you can have d_8 to 15 then p_1 and in this way last 1 will be having d_{24} to 31 and polarity p_3 , p_0 p_1 p_2 p_3 . So, an extreme case could be, say you can have only two bits say d_0 1 1 polarity bit then d_2 3

another polarity bit, in this way you can have d 31 that means you have divided into 16 groups where you have got fifteen polarity bits

So here, what is the tradeoff? Tradeoff is that you are increasing the redundancy and as you increase the redundancy obviously, your area will increase of the chip but the switching activity will reduce, so here is a tradeoff between increases in area, versus reduction in the switching activity.

So, as you can see here for n is equal to 2 switching activity is 20 percent, reduction is twenty percent, for n is equal to 32 switching activity reduction is 11 percent. So, here you will be having 25 percent reduction, and here you will be having 11 percent and for these cases you will be having in between reductions.

So, for large value of n the n bit data Bus can be divided into smaller groups, and each group is coded independent by associating polarity bit with each of the bit. So, when a 32 bit bus is divided into 4 groups of 8 bits it gives 18.3 percent reduction; that means, you get 18.3 percent reduction in this particular case, 25 percent 18.3 percent and 11 percent.

So, this is how the tradeoff ends. So, depending on your requirement, depending on the, which one is more important, area is important or reduction in switching activity is important you can choose one of the encoding technique.

(Refer Slide Time: 47:50)

T0 Encoding

- The Gray coding provides an asymptotic best performance of a single transitions for each address generated when infinite streams of consecutive addresses are considered.
- However, the code is optimum only in the class of irredundant codes, i.e. codes that employ exactly n-bit patterns to encode a maximum of 2^n words.
- By adding some redundancy to the code, better performance can be achieved by adapting the T0 encoding scheme, which requires a redundant line INC.
- The T0 code provides, zero transition property for infinite streams of consecutive addresses.

Encoding $(B(t), INC(t)) = \begin{cases} B(t-1), 1 & \text{if } t > 0, b(t) = b(t-1) + S \\ b(t), 0 & \text{otherwise} \end{cases}$

Decoding $b(t) = \begin{cases} b(t-1) + S & \text{if } INC = 1 \text{ and } t > 0 \\ B(t) & \text{if } INC = 0 \end{cases}$

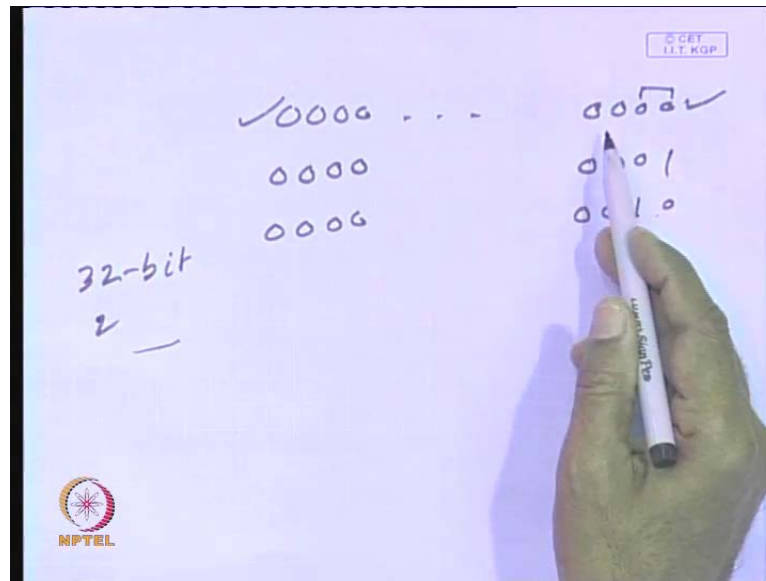
Ajit Pal
IIT Kharagpur
NPTEL

Coming to the last technique that is your T 0, T 0 encoding. In case of we have already seen, in case of Gray Coding, the Gray Coding provides an asymptotic best performance of a single transition for each address generated when infinite streams of consecutive addresses are considered; that means, whenever you are running a program, where the instructions are coming only from consecutive addresses and that number is infinite, in such a case the number of bit transitions asymptotically approaches to one; that means, you do not have any jam, you do not have any ub rooting call, you do not have any interrupt but in reality that will not be so.

Obviously, in in reality the number of bit transitions will be little more than one, depending on how many transitions occurred, say every fifteen consecutive instructions there may be one transition, so that in that case it will be little more than one but asymptotically it gives the value of one. However the code is optimum only in the class of irredundant codes that is codes that employ exactly n-bit patterns to encode a maximum of 2 to 2 to the power n words.

Now, by adding some redundancy to decode better performance can be achieved by adopting two T 0, T 0 stands for transition 0 encoding scheme which requires a redundant line one. So, here the idea is that you will be having some redundancy but the number of transitions will asymptotically approach 0, how can it be done, and that is being achieved in T 0 encoding, so the t 0 code provides 0 transition property for infinite streams of consecutive addresses, how the encoding is done is explained here you can see you have added a bit known as increment inc but before I explain the encoding and decoding let me explain the basic idea.

(Refer Slide Time: 50:21)



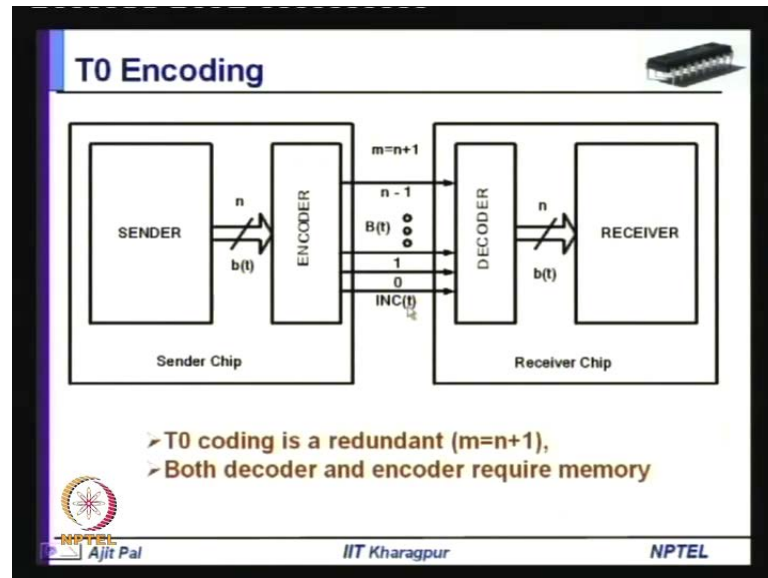
Suppose initially the address was say 0 0 0 0 and if you know the next address is 0 0 0 0 and 0 0 0 1, what you can do instead of sending this one you can send this one. And in form the other side that you increment by one, so in such a case there is no change on the Bus, and next time also suppose increases by 1 value, again **you do not** you keep on sending this and inform the other side increment it by 1 from the previous value. So, this is the basic idea of this T 0 encoding, how long you can continue with this, as long as it is only increase increment, as long as it increases by a fixed value known as astride, if the it is a 32 bit co address then the first 2 bits will be 32 bits, means you will require 4; that means, 2 bits will be required, and then 2 bits differences will be there between two adjacent addresses.

So, I mean depending on the address size it can be 2 bit it can be 4 bit, because the address is usually byte encoded, it is specified in terms of bytes adjust Bus. So, this is the basic idea and now we will be able to understand this encoding scheme. So, b t is the encoded transition which will be sending an INC, and you will be sending and whenever b t is equal to b t minus 1 plus astride then what you will do, you will keep on sending the previous value and one indicating that intimate bit one informing the receiver side that increase by one.

On the other hand, if b t is not equal to b t and b t minus 1 then of course, you have to you have to send b t, new value you have to send and then increment bit will be 0. So,

this is how the encoding is done and other side again we will do the decoding if whenever increment bit is 1 then b t it will, be b t minus 1 plus s that it has to be increased by the stride value otherwise, an new value will come that is b t.

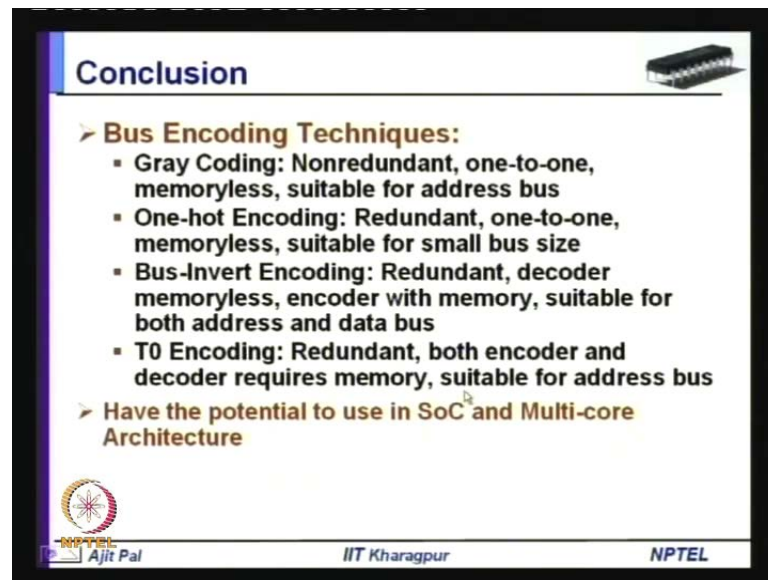
(Refer Slide Time: 52:59)



And this is how it can be implemented and can see the encoder will be generating those b t bits and INC increment bit, and on the decoder side it will generate the b t.

So, here it is capital B t, here it is small b t, and this particular code is redundant in nature but in this particular case you can see both the encoder and decoder will require memory, why it will require memory? The encoder has to preserve what was sent, what is the next value whether it will increase or it will change, so previous value has to be stored. Similar, the receiver also preserve the earlier value, because it has to keep on adding the new value s, I mean to to get the new value you have to keep on adding with s. So, that is the reason why both encoder and decoder will require memory.

(Refer Slide Time: 54:08)




Conclusion

➤ **Bus Encoding Techniques:**

- **Gray Coding:** Nonredundant, one-to-one, memoryless, suitable for address bus
- **One-hot Encoding:** Redundant, one-to-one, memoryless, suitable for small bus size
- **Bus-Invert Encoding:** Redundant, decoder memoryless, encoder with memory, suitable for both address and data bus
- **T0 Encoding:** Redundant, both encoder and decoder requires memory, suitable for address bus

➤ **Have the potential to use in SoC and Multi-core Architecture**

 NPTEL
Ajit Pal

IIT Kharagpur

NPTEL

Now we can conclude the Bus encoding techniques, we have discussed basic concepts of Bus encoding and considered 4 different encoding techniques Gray Coding, one Hot Encoding, Bus-invert encoding, and T0 encoding. And these encoding techniques have potential to use in system on chip and multi-core architecture. Nowadays, lot of embedded systems are being designed those are known as SoC system on chip, where you have scope for using this type of encoding similarly, multi-core is becoming increasingly popular, and the number of cores on a chip increasing from 2 to 4 to 8 and so on, and there is prediction the number of cores may exceed to fifty-six. So in such a case also there is scope for large scale use of encoding within the chip. So, with this we have come to the end of today's lecture, **thank you**.