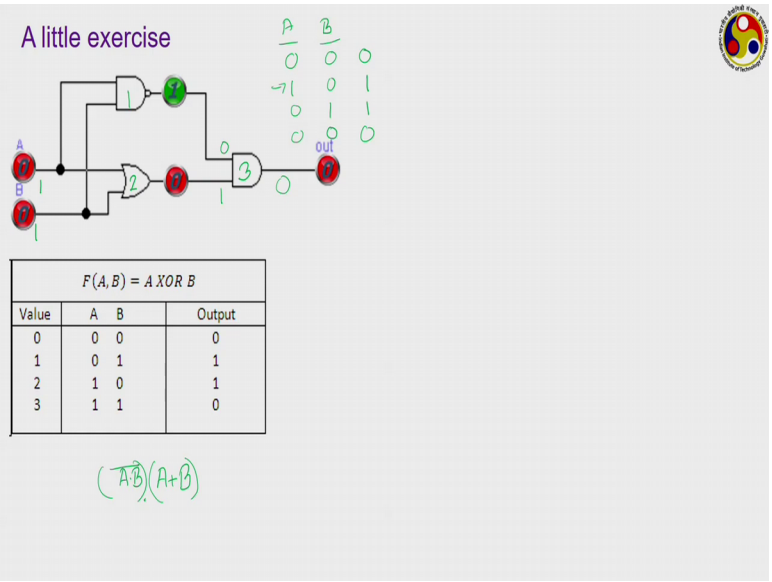
Principles of Mechanical Measurement
Prof. Dipankar N. Basu
Department of Mechanical Engineering
Indian Institute of Technology, Guwahati

Module – 03
Digital Techniques in Measurement
Lecture - 03
Analog-to-digital conversion

Hello friends welcome back to the third lecture of our week number 3 where we are talking about the application of digital techniques in different measuring instruments or in different measuring methodologies. In the first lecture here we have discussed about the different number systems and their mutual conversions; particularly it focus on decimal because that is the one that we keep on using for our day to day applications.

Whereas in the second lecture we have discussed about the fundamental components of any digital circuitry to be more precise the logic gates and their combination and we have seen that how by combining different kind of logic gates we can get any kind of combination of digital outputs from a circuitry. And then we have briefly discussed about or rather in quite detail, we have discuss due to binary codes the BCD codes and the gray codes which are ways of storing some decimal number into their digital format or I should say the binary format.

(Refer Slide Time: 01:39)



Today we shall be continue with the same and also we shall be learning about how to convert a real life analogue input to its digital counter, but before that let us do if it of exercise like we always keep on doing.

Here I have given a simple digital circuitry to you where there are inputs A and B and there are 3 components. Can you identify this gates from their symbol; each of these gates have a proper representation and always you have to identify the nature of the gate from the representation itself.

Now, gate number 1; what is this? This is an NAND gate; this gate number 3 this is an AND gate and whenever we have this dot was whatever this extension in front of an AND gate that represent NAND, which basically opposite or rather reverse the output of an AND gate and what about this number 2? This is an OR gate. So, one particular combination output is given let us try to form the truth table using the other kind of inputs. So, truth table means where let us say A and B are the 2 and finally, output what we are getting. So, our possible combination of input can be 0 0 1 0 0 1 and 0 0; one case is already shown here which is 0 0 let us tried to start with this.

So, our A is 1, B is 0 both are supplied to both the NAND gate and the OR gate. So, what will be the output of the NAND gate? Here 1 and 0 if we supply them to an AND gate, I am sure you remember that when both the inputs when AND gate are 1, we get output as 1; otherwise if it is 0. Now it is NAND which will actually reverse the output; now here we have 1 and the 0 then the output of NAND will be equal to 1. So, we are getting an 1; what about OR? Here one is 1 and 0 are supplied if any one of them is 1; then we are going to get an 1, both the 1s are going to an AND gate. So, your output will be 1, so here your output is 1.

We can write the first one which is already shown that is a 0. What will be the results if we change our output to let us say 0 for A and 1 for B; then again they are going to a NAND gate. So, quite similar to the previous situation the output will be 1; similar for OR gate will be 1 and finally, we are going to get another 1 here.

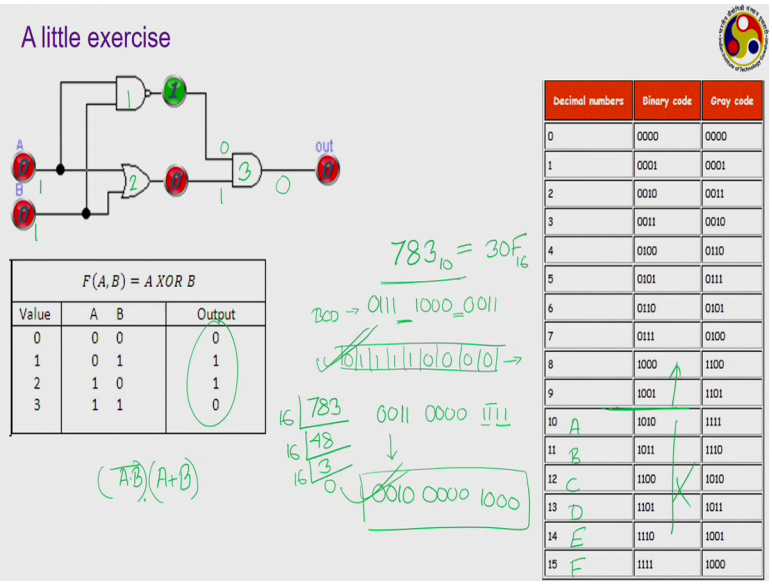
And what about the final situation where both are 0s or rather both are 1s yeah 1 for A and 1 for B, if we supply 1 and 1 to an AND gate; we should get 1, but this is being a NAND we are going to get a 0 here or if I write properly we are going to get 0. What about the OR gate it is equal to 1; actually it does not matter because whenever one of

the inputs to an AND gate equal to 0 final outputs will be equal to 0; so this is equal to 0 and this becomes our final truth table.

Now what we can write in the symbolic math point of view about the nature of the output ? So, the output for our gate number 1 which is a NAND that can be written as we write A dot B that corresponds to an AND; I am putting this over bar represent this NAND. What about the gate number 2 which is an OR gate for this or that we can write A plus B which is how we write or so, this is for gate number 1 this is for gate number 2 and now they are gate both are being supply to an AND gate, so put a dot in between this is what your output that you are going to get.

And also looking at this truth table can clearly see that this is actually what we are we are getting is actually an XOR operation or an XOR gate; that is by combining these 3 we are we can get operation similar to XOR. So, this is the simple exercise that we can have ah; let us do one more exercise which is related to the application of the BCD and gray format of storing binary numbers.

(Refer Slide Time: 05:39)



For that we shall be making use of this tables where we have the decimal equivalent of or rather binary equivalent for decimal this is from 0 to 15 and also their gray equivalent.

Do not need to look at the right hand side of the table the rightmost column of the table because gray we can know how to convert a binary digit to corresponding gray

equivalent. Let us took a random number let us say we have a number 7 8 3; then how in decimal. Then we can always convert this on to binary, but our objective is not to get the binary version of this; rather we want to get the BCD version of this binary coded decimal.

What is idea of BCD? There we have to identify the binary equivalent for each of the digits. So, what is the binary equivalent for 7? Binary equivalent for 7 is 0111, binary equivalent for 8 is 1000 and finally, binary equivalent for 3 is 0011. So, that is how it is going to get stored in the computer memory if you want to stored in BCD format this is what you have your BCD version of this.

By the way the white spaces that we are maintaining like this space here or this space here in practical computer storage, you will not get this it will be more a continuous representation. Like if your bits are looking like this these are the bits looking like this these are the bits in your computer memory then it will write 0 1 1 1. So, this first 4 bit count is the 7 and then immediately the next one will start store 8 and then the next one will continue for the 3; this why it will get stored in a computer memory.

Next is the gray version you want to get a gray version for this. Now we have discussed about the gray and we know that how to get the gray equivalent of decimal digits from 0 to 15, but problem is here all as long as we are dealing with a decimal number which is less than or equal to 15, we can immediately get a gray equivalent which is there in the table itself.

However here we are dealing with 3 digit 3 digit number then how to get the gray equivalent for this? Now here our numbers system that we have gray here in each bit we can store decimal number from 0 to 15. Now if we think about this from hexadecimal point then we can also say that we can store number from 0 to 9 and also A B C D E and F that is basically hexadecimal system. So, whenever we are dealing with decimal number first think about its hexadecimal equivalent and then convert that hexadecimal equivalent to corresponding gray format.

So, here 7 8 3 we have; so, let us first convert this to hexadecimal. So, we divided by 16, so once we divided by 16 we are getting 4 and then 8 leaving us 15, 15 actually in hexadecimal sense is F, so our remainder is F; then we divide 48 by 16; we are going to

get 3 and leading 0 and then leaving 16 or dividing 16 by 3 nothing there we are reminded the 3.

So, this 7 8 3 in base 10 or in base 10 will be having is hexadecimal equivalent is 3 0 F in base 16. So, to get the gray version we have to convert this 3 0 F or each of the digits of this hexadecimal number 3 0 F. We can directly convert that using the table; the other way let us see how we can do this.

First let us see the binary version of each of these digits. So, what is the binary of 3? Binary of 3 is 0 0 1 1 binary of 0 that is all 0s and finally F, F does not have a BCD equivalent I am just taking its binary equivalent of 15 which is 111, but this is not a BCD representation because in your BCD it applies only up to this in this portion, this portion dispersion there is no BCD equivalent BCD is only up to 0 to 9.

Here we are just trying to convert the binary equivalent of each of them to the gray. So, how to get the gray equivalent for this? First remember the rules the most significant bit will remain as it is which is 0; so that remains 0. To identify the second digit in the occupant or the second bit in gray scale we have to take the XOR of the first 2 here both are 0. So, its XOR are also will be equal to 0 then 0 1 XOR will be equal to 1; 11 XOR will be equal to 0.

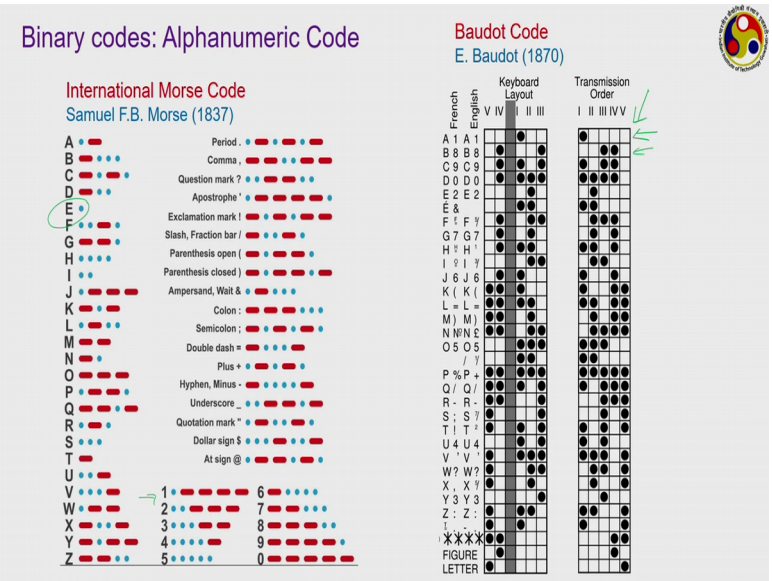
So, next all 0s; so its gray equivalent also will be equal to 0 all ones first 1 is 1, then if we combine these two; 1 1 XOR equivalent is 0 and I combined these two; it is 0 finally, combined these 2 XOR equivalent is 0. And this gives us the gray representation of or original decimal number 783. Again the white spaces should be neglected they will have a continuous representation somewhat like this. So, this one is the BCD representation of 783, this is the gray representation of 783 depending upon in which format your storing the number, the same will go to the register of your computer or in the memory of this.

So, we have seen now or we know now how to convert a decimal number to its gray or BCD equivalent and store that in memory. But numbers are not all rather there is several situations we have to deal with other characters also and that takes us to the next one which is the alphanumeric code. Another type of binary code sometimes in even other number format also we can be stored; these are alphanumeric code which all of you are very much aware about or always keep on using this.

Because just think about a computer or think about your mobile phone; it is not that always we are typing numbers you are always typing texts. So, whenever sending a message to someone, whenever typing an email, whenever typing an Whatsapp message what we do? We keep on typing the characters A B C D this charecters different symbols plus minus etcetera; the even more advanced versions of the at different emojis or smile is that we keep on sending.

Now those are not numbers then we definitely have to convert them to corresponding binary format and then only we can transmit that via the digital networks. So, that is what comes under this alphanumeric code category. There are several alphanumeric codes which has been proposed throughout the historical development of such digital networking.

(Refer Slide Time: 12:41)



The most from the historical point of with first one I should say is the International Morse Code proposed by Samuel Morse in 1837. Now have you heard about this? This is how to how a Morse could look like all the characters. Firstly, it has only the upper case letters A to Z and the digits 1 to 0 and in the international version of this one; several of the symbols are also included like shown here.

If you see properly each of these characters can be represented by a combination of dots and dashes. Like here the colours are given just to distinguish between them, but

practically they all may be in black. So, there are dots and dashes and the combination which is dots and dashers represent different kinds of characters.

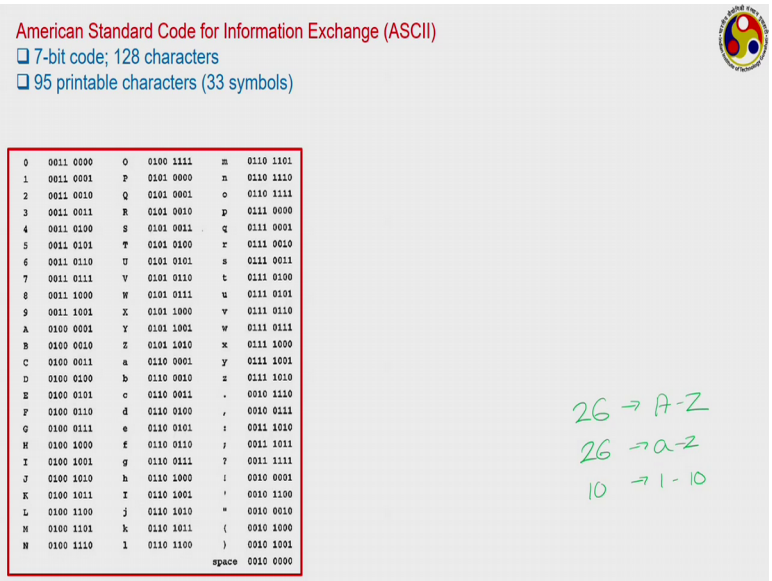
Like A is given as 1 dot followed by a dash and the number of characters to represent any corresponding character number of characters in Morse code required to represent any normal character they are not same. Like thing look at this E; here there is only one dot to represent E whereas, if you take any another one say number 1 here we have a dot and then 4 dashes. So, it they does not follow any particular pattern; this is just something that concerned persons have to remember.

But have you seen or have you heard about any application of the Morse code ? This is what that used to get used in the telegraph system; now in India the telegraph system has been abandoned, it has been withdrawn by the postal department, but is still there in different other places of the world and also used in different other coding or communication systems. And this is that is what is used the, what they use is this Morse code or the international Morse code.

So, the biggest problem of this one is the non uniformity or lack of any particular structure like the number of bits required to represent different characters may be different. That lead to the next one by dot code proposed by E by dot in 1870 here each character is represented by a 5 digit formation. Thus just look at this particular portion here each characters are given 5 bits to be occupied with and some of them will be occupied by black dots, some of them will be by white dots and accordingly we keep on going.

Again there is no particular pattern between successive numbers like look at A and B; A is having a single dot and followed by 4 white spaces where. So, will B 2 white space then 2 dots and then another white space. So, no particular pattern you some have to remember this, but its advantage about the Morse code is that each character occupies equal number of bits which is 5. Then it kept on evolving this is a different alphanumeric codes different countries kept on using different kinds of formats. In fact, they still keep on using different kinds of formats but there was the need of standardising this.

(Refer Slide Time: 15:48)

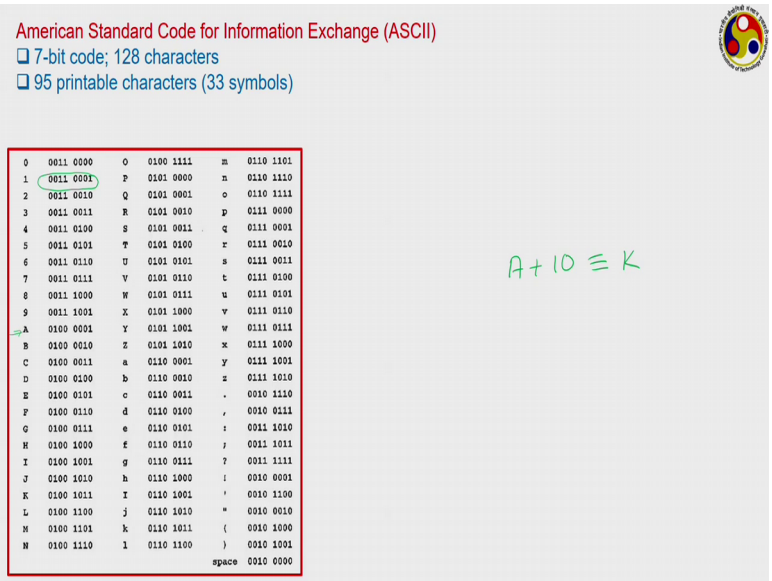


And so it came the ASCII; American Standard Code for Information Exchange which is something which was used from the very early days of computers. It is a 7 bit code there by allowing a total of 2 to the power 7 that is 120 total characters can be represented. Here each character can be represented by a 7 bit formation. So, a total of 127 characters can be considered under this ASCII format.

Ah Out of this 120 characters 95 them are printable; 95 printable we can get? We can have 26 uppers case letters that is A to Z; then 26 lowercase letters small a to small z then we have 10 digits 1 to 10 and there are 33 symbols just one representation like this. These are the printable characters you can see them and corresponding binary equivalent. Each of these printable characters have got a binary equivalent and that follows a particular structure.

Like just try to see at what is 0? First 0 this is your 0 corresponding binary equivalent you see and then when you move to 1 there is only change in one binary digit which is the least significant with that changes. Then the 2 is the next one means once you add 1 with the binary representation of 1; you get 2 and the same way look at the uppercase letters what we have as a if you add 1 to the corresponding binary representation you get B. So, that way it allows some kind of mathematical operation.

(Refer Slide Time: 17:35)

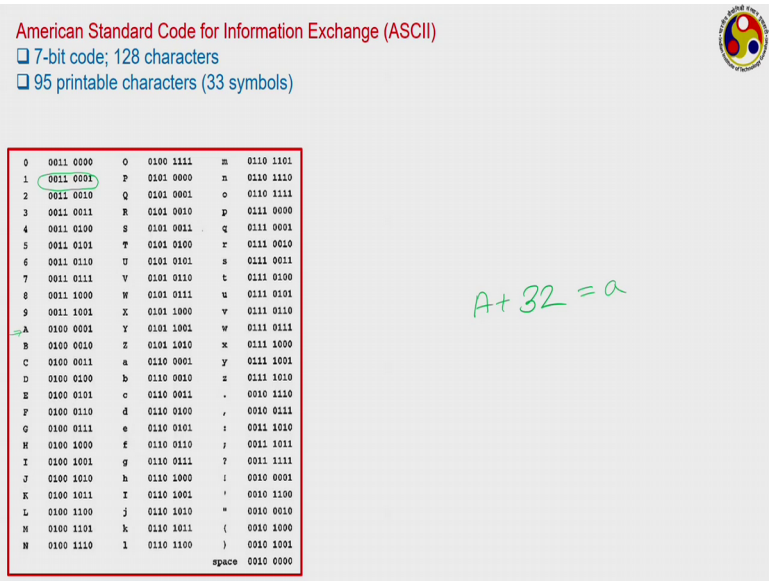


0	0011 0000	O	0100 1111	m	0110 1101
1	0011 0001	P	0101 0000	n	0110 1110
2	0011 0010	Q	0101 0001	o	0110 1111
3	0011 0011	R	0101 0010	p	0111 0000
4	0011 0100	S	0101 0011	q	0111 0001
5	0011 0101	T	0101 0100	r	0111 0010
6	0011 0110	U	0101 0101	s	0111 0011
7	0011 0111	V	0101 0110	t	0111 0100
8	0011 1000	W	0101 0111	u	0111 0101
9	0011 1001	X	0101 1000	v	0111 0110
A	0100 0001	Y	0101 1001	w	0111 0111
B	0100 0010	Z	0101 1010	x	0111 1000
C	0100 0011	a	0110 0001	y	0111 1001
D	0100 0100	b	0110 0010	z	0111 1010
E	0100 0101	c	0110 0011	.	0010 1110
F	0100 0110	d	0110 0100	,	0010 0111
G	0100 0111	e	0110 0101	:	0011 1010
H	0100 1000	f	0110 0110	;	0011 1011
I	0100 1001	g	0110 0111	?	0011 1111
J	0100 1010	h	0110 1000	!	0010 0001
K	0100 1011	I	0110 1001	'	0010 1100
L	0100 1100	j	0110 1010	"	0010 0010
M	0100 1101	k	0110 1011	(	0010 1000
N	0100 1110	l	0110 1100	)	0010 1001
				space	0010 0000

That means sometimes we also write in this way like we write A plus say 10; then we should get which character? A is this then 1 2 3 4 5 6 7 8 9 10; we are at K. So, this is this way we can represent, but here actually A and K represents and also 10 represent there binary equivalent and this is some sometimes we write in ASCII sense.

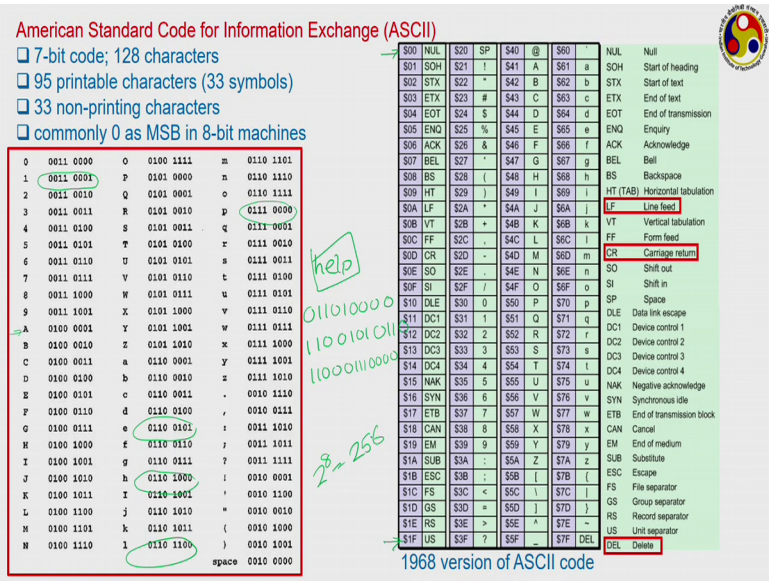
Another interestingly that we can observe the difference between lowercase and uppercase letters; whatever the position of the lower case letters low each of the lower case letters are equally positions are equally positions or at an equal distance compared to its uppercase counterpart; actually there 32 characters apart.

(Refer Slide Time: 18:26)



So, if we write A plus 32; means I am talking about the binary equivalent of A and if we add the binary equivalent 32; you should get small a. So, this way we can easily shift keep the easily can shift from one letter to; jump from one letter to other or other one character to another one and we can sometimes perform quite a bit of mathematical operations on them.

(Refer Slide Time: 18:56)



So, this is what we have in ASCII character, but here 95 of them are printable; there are 33 non printable characters as well like one none shown in this particular table. Here

generally on machines are 8 bit machines or rather they can be much higher bits also, but in 8 bit machines it is occupying 7 bit and so 0 generally is the one that is used in the as the most significant bit.

Like shown in this table if you are all of them the most significant bit is acquired by 0. This is a more comprehensively sometimes called the 1968 version of the ASCII code. Here if you check from starting from the first one up to this particular one, this entire first column they are all non printable characters and you have another one here; the dell these are all non printable characters.

All this non printable characters were used in the early days of the computers are (Refer Time: 19:46) digital instrument some for performing different kind of control operations like the printers etcetera. Most of them are absolute apart from just a few handful of them all others are obsolete and non relevant. These are the details of all these non printable characters only say a few of them.

Like the line feed, carriage return, the delete these are the one that are still under use now the how to use this ASCII character set? So, let us take an example say we want to write help. So, using your using the keypad of your computer mobile you can always type this help, but it is not stored in the memory of the device on it does not get transmitted as this h e l p; rather it gets transmitted in their corresponding ASCII binary format.

So, what will be the ASCII representative representation for each of them or ASCII counterpart? So, h where is h in this? This is h. So, in your in the memory it will be stored as 0 1 1 0 1 triple not then comes e. So, this is your e; just 3 stage back from there. So, it continues like this with this a e; rather 0 I am writing in the next line, but actually it will continue just 0 1 0 1; then we have l this is your l. So, it is 0 1 1 0 1 1 0 0.

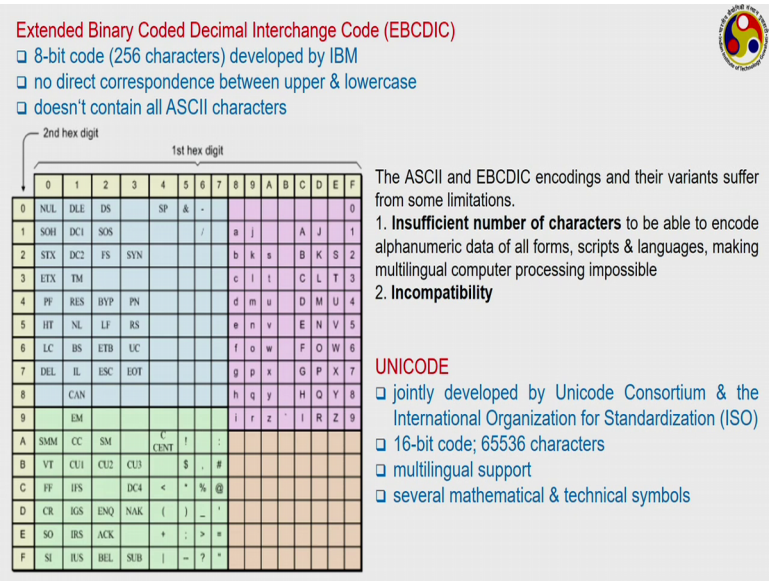
And finally, p just find p from there. So, this is p 0 1 1 1 0 0 0 it will be continuous change just because of the lack of space I have written it comprising of 3 lines, but it is a series of digits 0s and 1s like this; 8 digits or 8 bits corresponding to each of your character. So, total 8 into 4; 32 bits, 32 numbers of 0s and 1s; combining together represents your original word help. And this way we can form the ASCII I counterpart for any of the printable characters appear in this table.

But one issue is that all the digits, that are appearing all the characters that are considered in this ASCII set; they are just common characters to English. The upper and lowercase English letters and or English alphabets and the digits, but there are several other languages and infinite number of languages and also characters that appears in the word and therefore, ASCII I character set is not sufficient to represent even several of the English language sorry even several of the European languages like French or German ah; they have several other characters also which are very frequently used.

And if you want to include the Asian languages like Mandarin or Japanese; they have huge character set to be considered or even the Indian languages. There are so many languages in India each of them having a different script or many of them having a different script. The script of the North Indian languages does not match with the script of the South Indian languages. So, each can provide certain new characters.

In that case ASCII fails very much. So, there are lots of efforts to improve that to consider a more number of characters in the same data set within the scope of the same data set by expanding the number of bits involved in this. And extended ASCII which is an 8 bit character set that was proposed by certain companies it is an 8 bit. So, it considers 2 to the power 8 that is a total of 256 characters, but that even that is also not sufficient.

(Refer Slide Time: 23:34)



Another character set that is quite commonly used that is EBCDIC or called Extended Binary Coded Decimal Interchange Code; it is an 8 bit code. So, it can represent 8 characters using its binary format and it is developed by IBM and generally it is found application in IBM specifically in IBM machines and by certain other companies.

But strictly speaking its application is more restricted to IBM machines only. This is kind of representation I would not suggest you to try to go through this because it is not a very of course, it is standard for IBM, but it is which does not following regular pattern like ASCII. There is no direct correspondence between the upper and lowercase letters and also all the characters that appears in ASCII when they do not find appear; they do not appear in this character set and therefore, it has only limited application.

And despite having double number of characters then ASCII it still suffers because it cannot represents all those other characters that are possible. So, effort was on to expand this alphanumeric code centre by using higher bit; higher bit system. This ASCII and EBCDIC encoding and their variants suffer from the similar limitations like insufficient number of characters, which I have just mentioned and there are compatibility between each other.

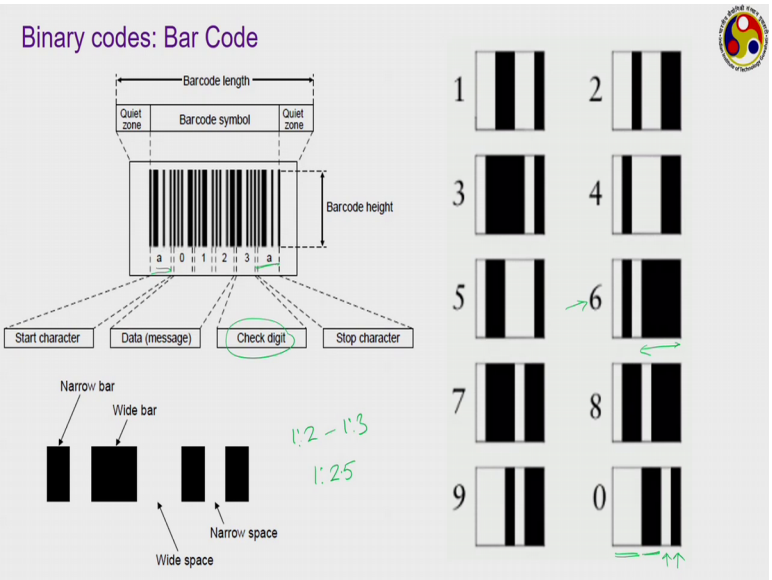
All characters in ASCII does not appear in EBCDIC and visa versa. So, we have this unicode now a days; the International Organisation for Standardization or ISO and Unicode Consortium that together the developed the Unicode; it is a 16 bit code system; that means, it can represent 65536 number of characters total; which generally involves almost all major languages that major all the characters that appears in major scripts throughout the world.

And at the moment you will find this unicode in most of the modern operating systems android or Microsoft on Linux based systems;;they all generally use unicodes now a days, but that it is; that is even not sufficient also for certain languages for certain characters appearing certain scripts steel are absent in this unicode and there is always scope for further improvement.

. So, presently unicode provides multilingual support and also it incorporates several mathematical and technical symbols. Thereby for all the technical writings we can easily use this unicode at least for the moment now. But the 16 bit itself may also may not be

sufficient in future and in next 5 years time, you may find a modified version of the unicode appearing.

(Refer Slide Time: 26:31)



There is another kind of course, I would like to mention very briefly that is the barcode; I am I am I am mentioning this because I know all of you are very much aware about this. Bar code is a combination of several black and white line simultaneously or side by side.

So, in all bar code you have a particular kind of pattern there is a couple of buffer zone or quiet zone or either side. And then you have a start character this portion is a start character which will indicate the indicate to the scanning device that exactly in which position from which position its start scanning.

Similarly the stop character here which indicates the scanner exactly where to stop the scanning and then you have the data message in between portion and we have check digit we generally checks whether the reading is correct or not. Barcodes you will find there are 4 kinds of characters; the there is a narrow bar wide bar and similarly narrow space and wide space or you can think this way; the black lines there can be thin or there can be thick similarly white lines can be thin or thick.

Generally the ratio of their thickness varies in the range of 1 is to 2 to 1 is to 3; quite often you will find the ratio as 1 is to 2.5 to be an optimum one. I will not going to the

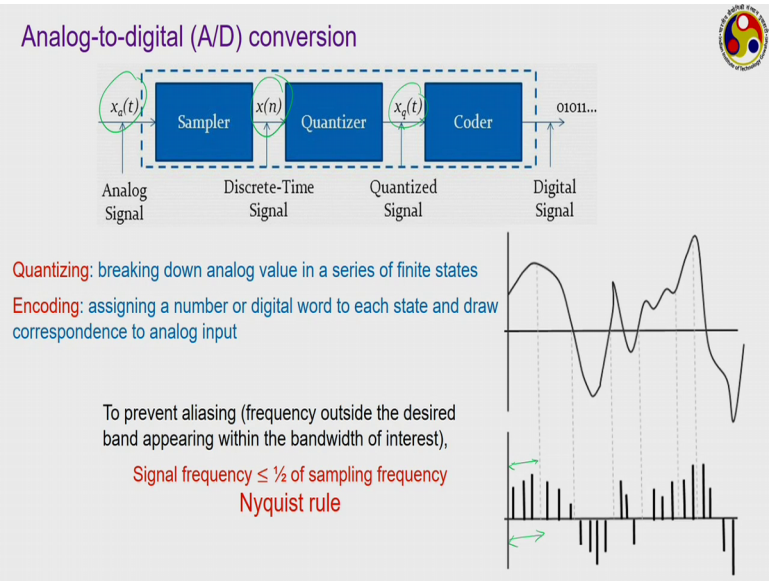
detailed of how barcode came in exactly what are the farmers and because there are several kinds of standard for barcode; this is just one of the formats.

Generally in each bar code you will find there are 7 parallel lines to represent each character. This is one representation just look at one; there are 2 white lines then 1 black lines the 2 white lines again and finally 1 black line that is 1. Look at 6 there is 1 white line, 1 black line then another white line and then 4 black lines together; this portion there are basically 4 black lines together or look at 0 what you can find? There are 3 white lies in which zone then 2 black lines then 1 white and then another black.

So, this way the different characters can be represent the barcode this has just digits in one of the standard systems ah; similarly any kind of alphabet can also be represented here. And that is something that you are very much aware about you can see the application of this one, whenever you go to some kind of shopping where they keep on using.

So, this is about different kinds of binary codes that we keep on using; we had learned about BCD and gray codes which are helped which are used for storing numbers or digits in the binary format. And we have discussed now about the alphanumeric and also the barcodes, but now we have to check exactly how we can convert a real life decimal signal or I should say an analogue signal to corresponding binary format.

(Refer Slide Time: 29:31)



So, let us move to the analogue to digital conversion. In any analogue to digital converting instrument you will find there are generally 2 steps. The analogue signal that you have I should say there are 3 steps; first there is a sampling step that is analogue signal from their certain specific number of data points are collected which is called the sampling.

There by the continuous signal this continuous analogue signal gets converted to discrete signal whereas, particular number of generally a specified number of data points are available over a period of time. And then using that quantize data this sample data; we get a quantize sample quantized signal which perform with this quantizer, the digital instrument generally perform this quantization operation and then we have the encoding.

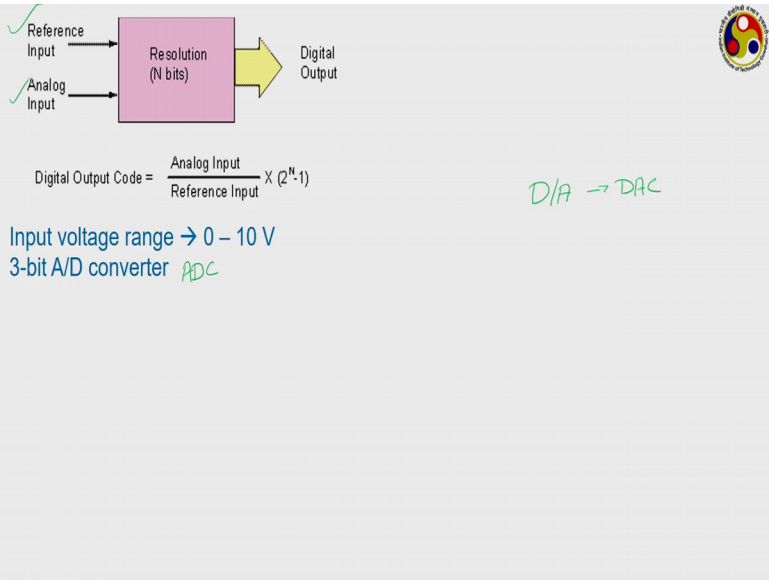
So, in the quantization step the analogue signal is broken down the series of finite states and then during encoding in number of digital words; generally a number is assigned to each of the states to get the final corresponding digital output; like the example shown here. You have the analogue signal in the top and during the sampling step we have collected a particular number of data points, depending upon the resolution of your instrument, depending upon the sampling speed of the instrument we can collected a particular number of samples which are then quantized and encoded to the concerned format that we want.

Here one important you when you are going to detail of the electronics the frequency of collection of the signal or the sampling speed itself is very important to ensure the correctness of the data; to prevent aliasing which is actually the frequency of outside the desired band appearing within the bandwidth of interest. We need to ensure that the signal frequencies less than half of the sampling frequency. Preferably we want the sampling frequency to be much much higher than the signal frequency. Sampling frequency is given as like over a particular time interval how many samples you have collected.

So, like in this example there are 3. So, if this time interval is 1 second, then a sampling frequency is 3; higher the sampling frequency it is always better your instrument is always be able to give much beta representation of the actual analogue input. So, we always want the sampling frequency to be high and it the; to the lowest limit it should be

at least twice of the actual signal frequency. So, this particular rule is called the Nyquist rule.

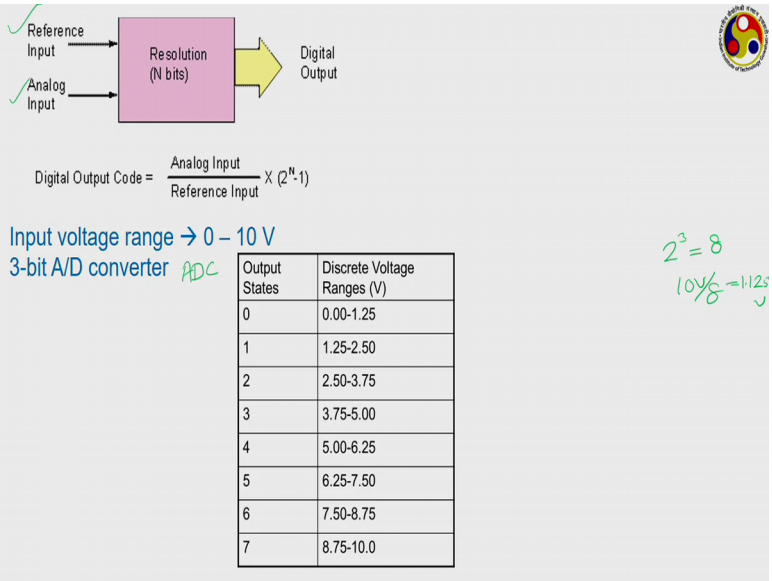
(Refer Slide Time: 31:57)



So, now in any and each of the instrument we have the analogue input coming in and quite often we generally have a reference input also. This reference input is compared with analogue input to provide some kind of modulated signal which is sampled, then quantized depending on the number of bits that may instrument is using and then we get the digital output of the end.

General input voltage let us take an example that our instrument is able to operate within the input voltage range of 0 to 10 volt. And we are having a 3 bit AD converter AD converter is this particular way we represent; analogue to digital converter. Similarly DA converter D slash a represents digital to analogue converter. Another way of writing this is we shall be using this form and very soon ADC that is the Analogue to Digital Converter, similar this is also written as DAC; Digital to Analogue Converter.

(Refer Slide Time: 3:05)

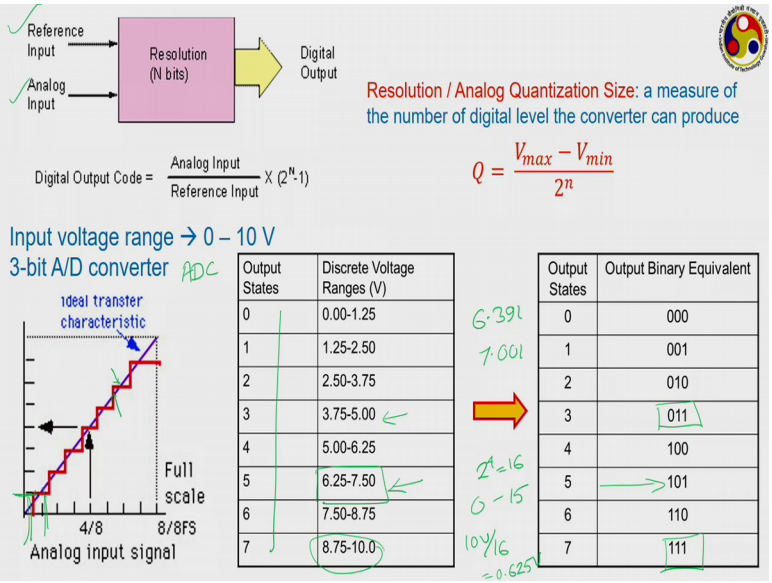


Output States	Discrete Voltage Ranges (V)
0	0.00-1.25
1	1.25-2.50
2	2.50-3.75
3	3.75-5.00
4	5.00-6.25
5	6.25-7.50
6	7.50-8.75
7	8.75-10.0

So, we are using an 3 bit AD converter. So, 3 bit means how many levels it can allow? 3 bit means how many correspondingly how many decimal values that we can allow? We can have 2 to the power 3 that is 8; that means, 8 levels can be allowed within this range of 0 to 10 volt.

So, if we divide this 10 volt by 8 what we get? We get 1.125 volt; so that gives you the resolution of the instrument and this is what we have. So, we can divide this entire voltage range of 0 to 10 volt into 8 distinct levels; each of them having a span of this 1.125 volt. So, anything appearing between 1 to; 0 to 1.125 will be identified 0 or the state number 0 then 1.25 to 2.5 will be identified as 1. This way finally, anything appearing between 8.75 and 10 volt will be identified as a level number 7 or state number 7.

(Refer Slide Time: 33:57)



And then convert all these 0 to 7 into the digital equivalence equivalent. So, we know how to convert the first one will become in 3 bit representation; it will become 0 0 0 and the final one that is 7 in digital or state number 7 will become triple one; this is how we get the digital equivalence of the original input signal.

So, you the input signal that you have provided in 0 to 10 volt as our instrument is 3 bit. So, we are converting that to 8 different levels and we are getting the equivalents; for anything appearing between say in the range of 3.75 to 5 we are getting this the digital equivalent is 0 1 1. Let us say your input signal is 6.391. So, exactly which level it appears? 6.391 appears in this range of 5. So, it's usually equivalence will be 101.

So, another way if you think about if you are digital input is 7.001 that will also appear in the same range that is any value your analogue value appearing within range of 6.25 to 7.5; your digital instrument will not be able to classify between them, they will represent all of them by this state 5 or 1 0 1 is the output.

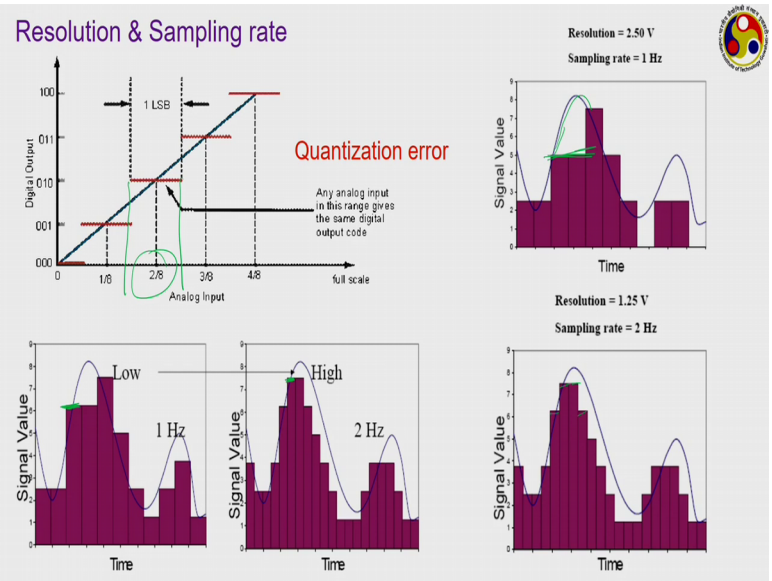
So, that is one way of drawback of your system and to rectify that we need to go for higher grade system. Like think about if your instrument is a 4 bit one then what is the small with of each of the states are how many states it can measure? So, it will be able to measure 2 to the power 4 that is 16 number of states that is starting from 0 to 15 states; it will be able to measure and what will be the which of these states that will be 10 volt divided by the 16. So, I think it is 0.625 volt.

So, within a span of any signal appearing in this span of 0.625 volt will be identified as 1. And then accordingly it will be dividing the entire span of 0 to 10 volt within 16 levels. So, this is what we call as resolution or sometimes also called analogue quantization size; it gives a measure of the number of digital level the converter can provide produce.

So, it is just I have mentioned V max minus V min by 2 to the power n is the resolution of your instrument and finally, we get a representation like this. Like for the 8 bit one that we are talking about anything appearing within this particular zone; any signal appearing within this particular zone there be represented by this particular digit. Similarly anything appearing within this particular zone they will all getting or respondents to this.

So, we have to be careful like if we want very precise measurement a difference between say this particular. Let me clarify this means here say this is one analogue value this is another analogue value and we want to make an distinction between them; your digital instruments may not be able to do this because they are falling within the same state level.

(Refer Slide Time: 37:02)



So, this is again just repeating the same; the resolution any value appearing within this signal range will be given the same digital level and that will continue for each of the digital levels and the only option is to increase the resolution of the instrument. And that

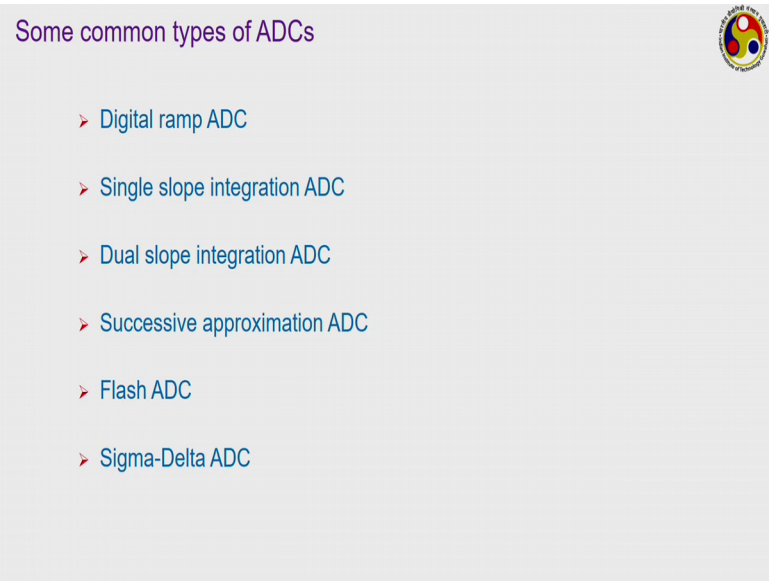
is to allow more number of states within the range of voltage that you are dealing with; this is called the quantization error.

If you remember in our first lecture in this particular module; we mentioned about the quantization error. That is analogue signal is independent of quantization error because you are not being any sampling there, but is in digital signal we are doing the sampling leading to this quantization error. Another point that comes into picture is the sampling speed how many samples you are collecting over a particular interval?

Like in the there are 2 situation shown there in the first case you have collecting less number of points, in the second case you have collecting the double number of samples. And definitely your resolution is much better because you are able to measure like here over anything appearing this span is given the same value whereas, it is smaller the second case in the second case.

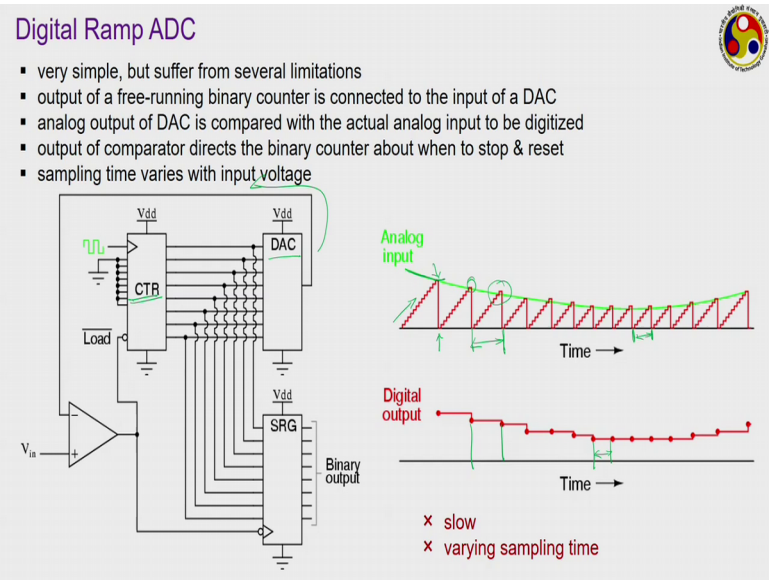
There are 2 examples shown first one is having high low resolution or rather quite big resolution and low sampling rate, second one is having more precise resolution and higher sampling space. So, and definitely second one will give you much better digital representation because you can see in the first one this entire portion while the curve is changing like this; this digital signal is stuck at the same point whereas, here it is again broken into 3 4 different levels. So, we can have much better representation when both resolution and sampling speed are better or favourable.

(Refer Slide Time: 38:48)



Now, there are different types of analogue to digital converters; we shall be discussing about some of them, each of them has it is own shortcoming, each of them has its own merit. Some of them are very fast, some of them are more precise than the others some of them can be very accurate, but are very costly or complicated in their operation. So, let us just quickly check the way some of them are operate.

(Refer Slide Time: 39:12)



First one is the digital ramp ADC in this is the circuit diagram for this ah; at the moment there is no need to go for the circuit diagram. The digital ramp ADC is a very simple one, but it has several limitations it is probably the first in this line.

Here we have the free running binary counter be done about binary counter. So, we have a; this is the counter like the diagram that is shown here is of an 8 bit thing. So, you have a free running binary counter free running means like initially the counter status all 0s. Then it changes the first digit to 1 and then after particular time interval it take a changes to 1 0, then again the same time interval apart it changes to 1 1 and this way it keeps on changing 0 0; this way it will keep on changing without any control. So, this is what we referred by free running binary counter.

And output from this free running binary counter is supplied to a digital to analogue converter. How it is done? That we shall be learning in the next lecture that is the digital to analogue conversion. But the output of this free running binary counter is supplied to a digital a DAC or Digital to Analogue Converter and the output of this DAC this

particular one is directed to a comparator, where it is compared to the actual analogue input.

So, as long as your analogue input is higher than their counter will be allowed to run. But whenever the output of your the output of your digital to analogue conversion that becomes higher then of course, at that instant you have cross the analogue value; so we should stop.

Just look at here your analogue value is this over this range and the your counter is increasing, the voltage corresponding to digital to analogue conversion that keeps on increasing. Just at this particular point it is touching the analogue signal or maybe just crossing it. So, thus counter this comparator will activate a control mechanism which will stop the counter thereby giving this particular digital output value and the counter will be set back to 0.

So, the objective or rather the operation of this counter this comparator rather is to ask the binary counter to stop as soon as the output of this DAC becomes greater than the actual analogue input; there by you get this. Immediately the counter comes back to 0 it comes back to 0 and then again continues to operate that is 1 bit at a time; it keeps on increasing. And it continuous to increase still again it encounters the analogue signal like here. During that entire period your output is stock here then here you are getting another output for this.

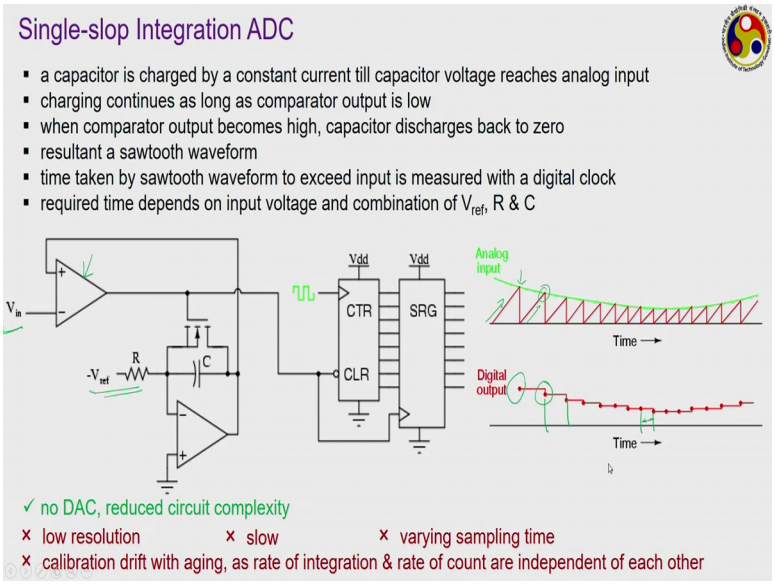
So, this is a very simple mechanism of operation, but what do you feel? What are the shortcomings it can have? Definitely it is quite slow because every time for you are starting from 0, you are continuity will meet the analogue signal and as soon as you meet analogue signal you fall back to 0; so it is very very slow in it operation.

If the sampling time keeps on varying with the input voltage like you can see in this case the sample one sample is to collected here the next one will be collected here. So, this is the span that depends upon the input voltage value which is high; like in this case one sample is collected here the next one is collected here the next one is collected here it is a much smaller span because the input voltage is like that in this particular time.

So, the sampling spirit or sampling time keeps on varying with the input voltage itself which I also may not be a very favourable one. Like we can see here this is the gap due

to successive samples whereas, here the gap is much smaller in several applications this can be quiet disastrous. So, we do not want this to happen in many practical applications. So, one big problem for this digital ramp ADCs it slow and second is this varying sampling time; it is really slow one of the slowest among all the ADC that we are discussing, but advantages it is very simple it is cheap and easily operable.

(Refer Slide Time: 43:24)



Next we move to the single slope integrating ADC; in the single slope integrating ADC we have a capacitor, this is the one. This capacitor is charged by a constant current till the capacitor voltage reaches the analogue input. That is the output of this capacitor is supplied to this comparator, but it is compared with the actual analogue input which is there.

And as long as the analogue input is smaller the capacitor is allowed to be charged till the they reach the output voltage. When the comparator output becomes high the capacitor discharge is back to 0 that results in a sawtooth waveform just like this. Over this the capacitor output is increasing capacitor is continuous become in charged. So, voltage of the capacitor also increasing just here its meets the analogue signal; so, that provides you first digital output and capacitor immediately discharge is to 0.

And again it continues till it meets another the analogue signal again that may providing the second one. So, we get a sawtooth kind of waveform as the output of this capacitor voltage it in the previous case if you remember you had a staircase kind of configuration

of or staircase kind of output going to the comparator whereas, here we have a ramp or we have continuously increasing one.

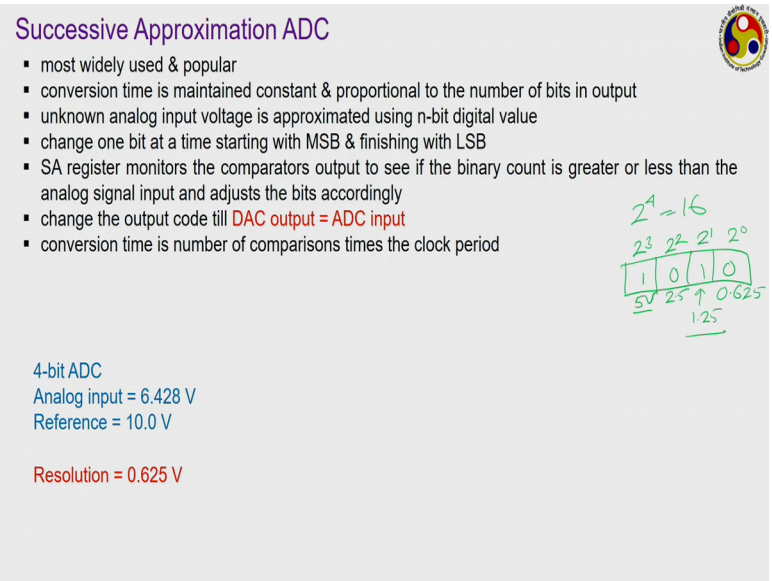
So, the time taken by the sawtooth waveform to exceed input is measured with the digital clock. And the time required depends again on the input voltage and also it depends on the combination of the corresponding circuitry that the value of the capacitance, corresponding resistance, the reference voltage also which is there with the capacitor itself; it depends on all of them.

So, it does not involve any kind of digital to analogue conversion. So, the circuitry is much simpler it is cheaper also in that sense, but it suffers from all the disadvantages that a digital ramp one have. Like low resolution slow and also varying sampling speed you can clearly see here; this is the sampling interval where the sampling interval will much slower in this case.

So, it has varying sampling time and there are another problem can be the calibration drift with aging. As the rate of integration and rate of counter independent of each other that is as it continuous continuously use; the capacitance where is extremely change there by similarly effecting the output on this.

One improvement on the single slope integration ADC on single slope integrating ADC is the double slope one; where instead of being charged by a constant current, the capacitor is charged by a current proportional to the input voltage thereby reducing the operation time quite a bit rest of the operating principle remains the same. If someone is interested you can search over the internet for double slope integrating ADC.

(Refer Slide Time: 46:13)



But we shall be talking about now the successive approximation ADC which is the probably the most widely used and popular one. It has a conversion time constant maintain constant and proportional to the number of bits in the output.

Like in the previous 2 examples or previous 2 convertors we have seen that the conversion time depends on the input signal itself. Depending on the, that is in both cases whether you are going for a staircase approach in the digital ramp or whether you are going for a constant current approach in case of a linearly increasing voltage in case of the integrating ADC; we have a varying sampling time.

Whereas, the biggest advantage for successive approximation here your conversion time is constant in this proportional to the number of bits in the output which should be the ideal one. Now, unknown analogue input voltage is approximated using an n bit digital device, one bit is changed at a time starting from the most significant one and finishing with the least significant one.

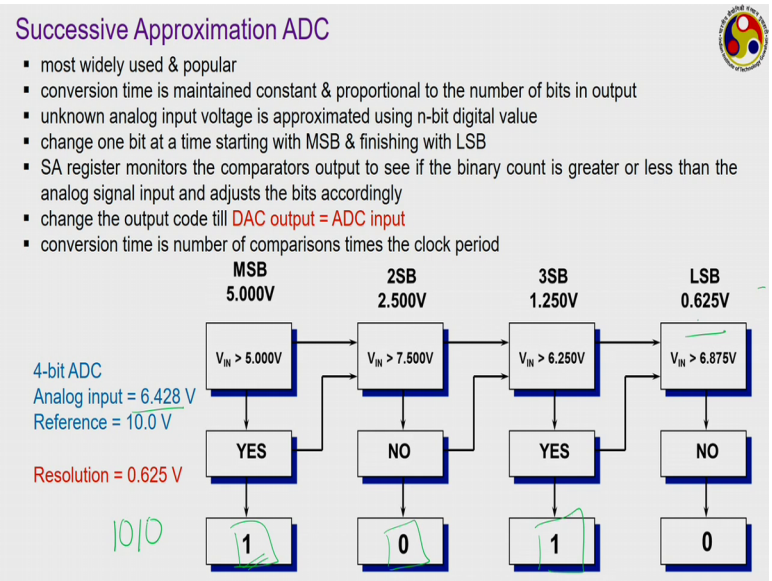
And there is register which monitors the competitors output to see if the binary count is greater than or less than the analogue voltage and accordingly it adjust the bits the; its stops when the output of your DAC becomes equal to ADC. So, the final conversion time is equal to the number of comparison time is the clock period and number of comparison is equal to the number of bits you know since quite of scared let us check with an example.

We have a 4 bit ADC which is receiving an a sum analogue input of was 6 point volt and its is a reference of 10 volt. So, what will be its resolution? It is a 4 bit; that means, it can allow a total of 2 to the power 4 that is 16 levels and there are 4 bits. So, so each of the bits the first one will correspond 2 to the power 0, next one 2 to the power 1; this is 2 square and finally, 2 cube total 16 level it can provide. And what is the resolution? That should be equal to 10 volt divided by 16; so 0.625 volt; so resolution is 0.625 volt.

So, what is the voltage corresponding to each of them? First one is 0.625 into 2 to the power 0 that is 0.625 volt itself. Second one 0.625 into 2 to the power 1; that is 1.25, third one 0.625 into 2 square that is 2.5 and the most significant bit corresponds to 0.625 volt into 2 cube that is 5 volt. So, each of these volt each of these have some input.

What I mean here like suppose you are having a 1 here and all 0s; then corresponding digital to analogue conversion will return a 5 volt. Whereas, if you are having suppose 1 0 1 0 then corresponding digital to analogue conversion will return a 5 volt plus 1.25 that is 6.25 volt and precisely that factor is using successive approximation ADC.

(Refer Slide Time: 49:17)



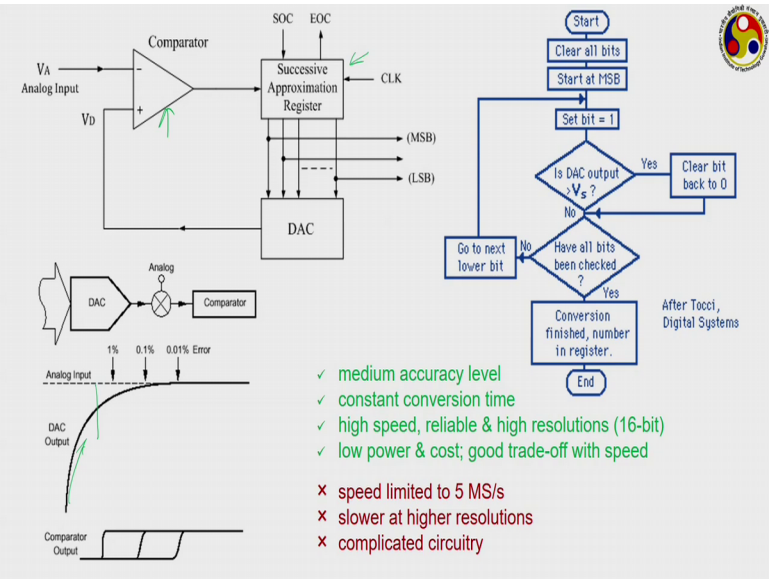
This is an example; here the output that or so analogue input that we are convert is this. So, we start from the most significant bit we set all these bits to be equal to 1 or rather the register of the instrument sets all of them to be equal to 1. And we start with MSB when it I set MSB or I should say initially MSB set to 1, others are set to 0 by the resister.

So, when it is 1; we are getting 5 volt and the comparator checks whether the input is greater than or 5 volt; greater than 5 volt or not? In this case it is greater; so this one is retained. Now, it moves to the second one which was initially 0 now it is set to 1. So, how much is are output 5 plus 2.5 that is 7.5. 7.5 volt is the output of your DAC and whether it is greater or not? It is lesser, so it goes back to 0.

Now the third bit that goes to 1. So, how much is your output form from the DAC? Remember MSB has already been set to 1, second one has been set to 0; so you have a 5 plus 6 5 plus 1.25 volt; so 6.25 volt; whether it is greater comparator checks and the analogue input is greater than that. So, this 1 is retained here; so 3 bits are final 1 0 1.

Finally the least significant bit is set to 1 and comparator checks whether the input voltage is greater than 6.875? No it is less than that. So, you set to 0 and final output becomes 1 0 1 0; this is the digital output that you are going to get from this. If we can increase the number of bits then we can get even more number of binary digits to represent the same signal.

(Refer Slide Time: 51:00)



This is kind of circuit diagram where this is your successive approximation register which keeps on changing 1 bit at a time most significant bit first like when it is one other set to 0; it checks the output of this DAC to this in this comparator. And depending on the result of that it keep this one either as 1 or goes back to 0. Once this is set then it

moves to the next one which is set to 1 and this process is repeated till we finalize what is the digit in the LSB.

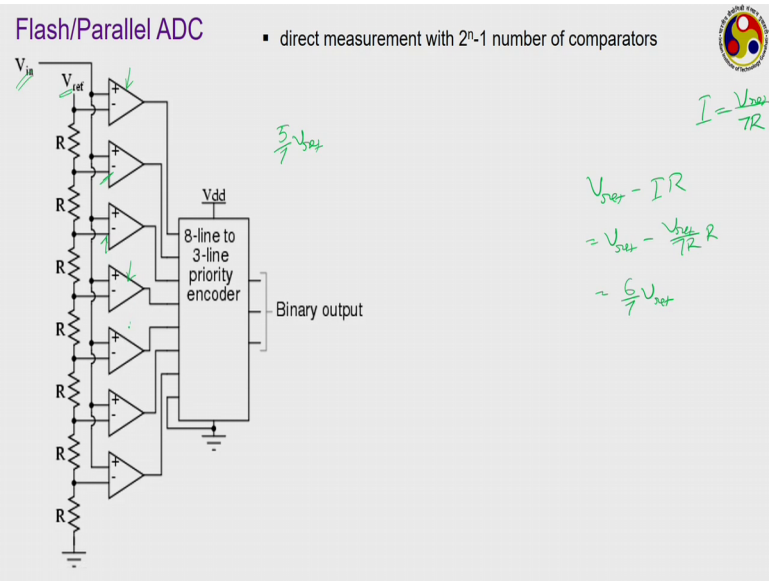
You can think the DAC output keeps on changing like this. Your analogue input is known which is a constant and then DAC output when you set the most significant bit; you have set of the big one then there is a smaller much smallest change finally, we are going back to the analogue input. If we think an algorithm point of view just what I mentioned first all bits are cleared and we start with most significant bit; MSB like here first MSB set to 1, others are all 0.

So, the DAC output is greater than this then if DAC output is lesser then it will remain as 1; otherwise it will go back to 0 and then it moves to the next significant bit. This way we keep on moving till the least significant bit depending on how many number of bits you have; you have you can we know the number of times we have to compare and accordingly the total time require for conversion is fixed.

It has medium accuracy level depends on its own resolution, but the constant conversion time is the biggest advantage; it is a high speed reliable and high resolution device can go up to 16 bit which is probably the highest among all the ADCs. Then it consumes low power and cost it is good trade off with the speed; the speed is low rather I have mentioned that the conversion time is constant I have mentioned high speed, but compare to if you other ADCs speed is slow.

It is slower as a relation keeps on increasing the slower because like 8 bit cases we have to compare 8 times or as in 16 bit you have to compare 16 times. So, the conversion time keeps on increasing and the circuitry is a bit complicated because of all this comparators involves or I should say all this comparisons involved; you have to modify each of the bit once at a time.

(Refer Slide Time: 53:25)



So, the next one comes is the flash or parallel ADC which is the probably the simplest one to understand from operation point of view and it is very very fast. It gives direct measurement with 2 to the power n minus 1 number of comparators. How that is done? Look at the circuitry here we are dealing with n a 3 bit output system. So, your total number of levels it can measure is 8 which is 2 cube; that is eight. So, we shall be having it there are ah; 8 minus 1, that is 7 number of comparators will be there like shown here.

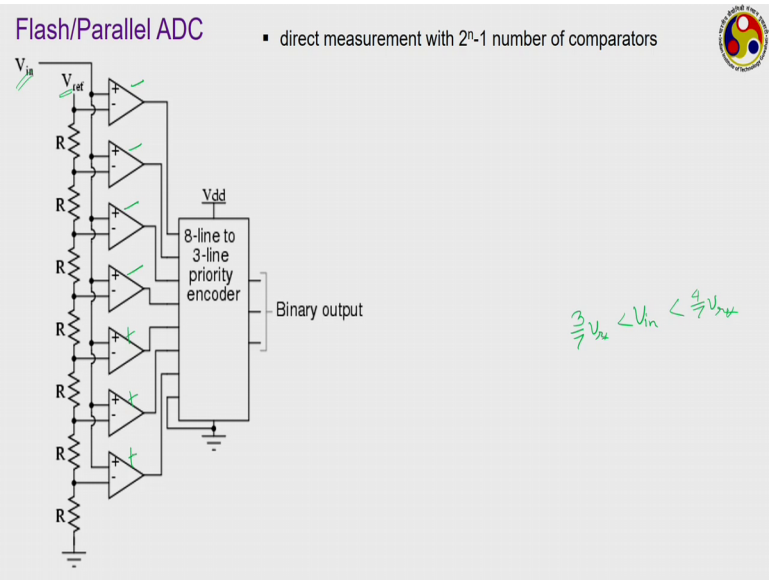
Now, there will be a reference voltage this is the reference voltage and then there are 7 number of registers; equal value of this resister you can see all these R there are 7 number of registers and 7 number of comparators. This chain of resistors are connected to the ground; so, what is current that is flowing through this? Can you calculate the V ref is the voltage that you are applying on one end other end is grounded there are 7 number of R resistance. So, the current that is flowing through this I is equal to this V ref divided by 7 R that is the current that is flowing through this.

Now each of the comparator in one side that is connected to the input voltage on the other end it is connected to either the V ref or a modified version of that. Like the first one the top one is connected to the V ref; second one is connected when the V ref connected to the V ref through one resistor. So, what is the voltage at this particular point?

It is V ref minus I into R I into R is the corresponding laws that is V ref minus V ref by 7 R into R that is 6 by 7 V ref. So, the second one is subjected to 6 by 7 V ref similarly the third one will be subjected to I should say this particular line; this particular portion it will be subjected to 5 by 7 V ref and this way it will keep on increasing or decreasing rather this will be subject to 4 by 7 V ref; 3 by 7 V ref for this 2 by 7 for this and this one just one seventh of the V ref that you are providing.

So, the reference voltage for each of the comparator that keeps on changing and that is compared with the input voltage. Now depending on the value of your input voltage one or more number of comparators will give you positive output and some of them give you negative output.

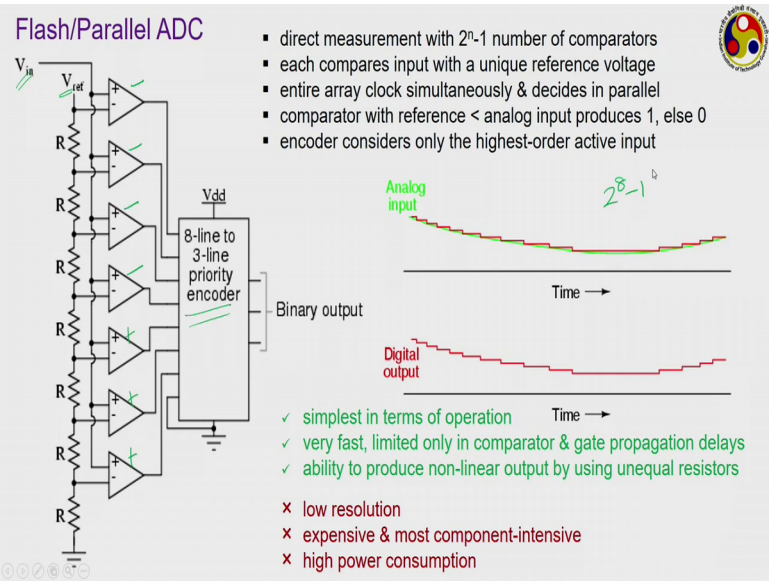
(Refer Slide Time: 56:09)



Like suppose if your V in is in between 3 7 or I should say it is greater than 3 seventh of V ref, but it is less than of 4 seventh V ref. Then it is greater than 3 7th V ref, but more than 4 7th of V ref; then the what output you will get even to get from that?

The first one will give a positive output because here V in is greater, second one who you are. Sorry in the first one you are supplying V in and V ref; so it will give a negative output, second one the reference is 6 by 7 of V ref; so it is negative; it is 5 by 7; so it is negative. This is 4 by 7 this also negative, but here the reference is 3 by 7. So, this will give you a positive one this is to positive one this also positive one.

(Refer Slide Time: 57:14)



So, for this particular input voltage it is giving you positive output from 3 of the; 7 comparators whereas, others are negative. So, the entire arrays clock, simultaneously and decides in parallel and if the comparator comparative with the reference is less than the input; then we get 1 otherwise 0.

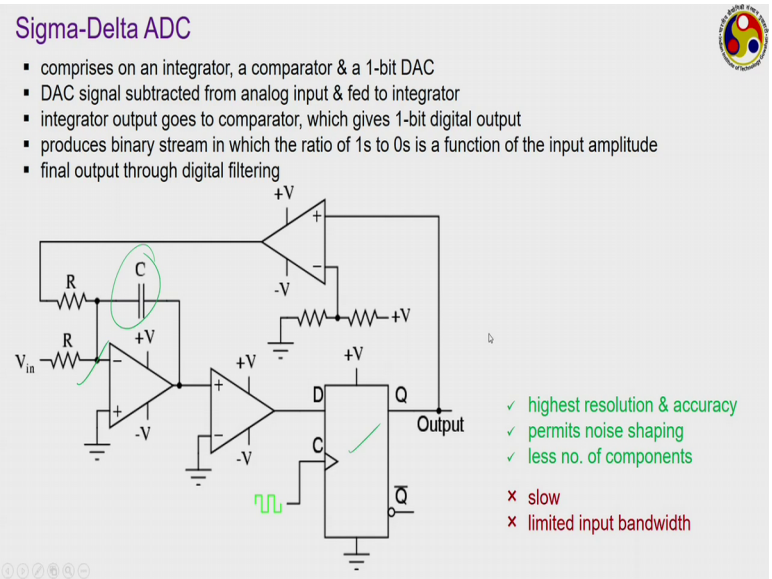
And now only the one that is having the highest order that input is considered like here 3 are giving positive. So, only this one will be considered because here the reference is the highest and then according your encoder decides which should be the binary output. This is the way the final signal looks like; it is simplest in terms of operation very fast that is why we call it parallel because all encoder rather all these 7 components are working parallelly. It depends only on the delay that is necessary with the comparators; otherwise it is very very fast.

It is able to produce non-linear output as well that is one very important application like here all this 7 registers are equal in magnitude. But if something we you want non equal reference or non we want to have a non-linear conversion; then we can change the value of this resistors itself. Therefore, we can get non-linear output as well which none of the other variation of this ADC converters or ADCs can give.

But its resolution is quite low expensive and most component intensive because like in this case for 3 bit output; we are needing 7 number of comparators. Whereas, think about if you are suppose if you are talking about an 8 bit conversion; then you will be needing

2 to the power 8 minus 1 number of comparators. So, number of comparators keeps on increasing rapidly with the number of bits output bits. Accordingly it becomes expensive and very very difficult to design and the power consumption also will keep on increasing yes.

(Refer Slide Time: 59:06)



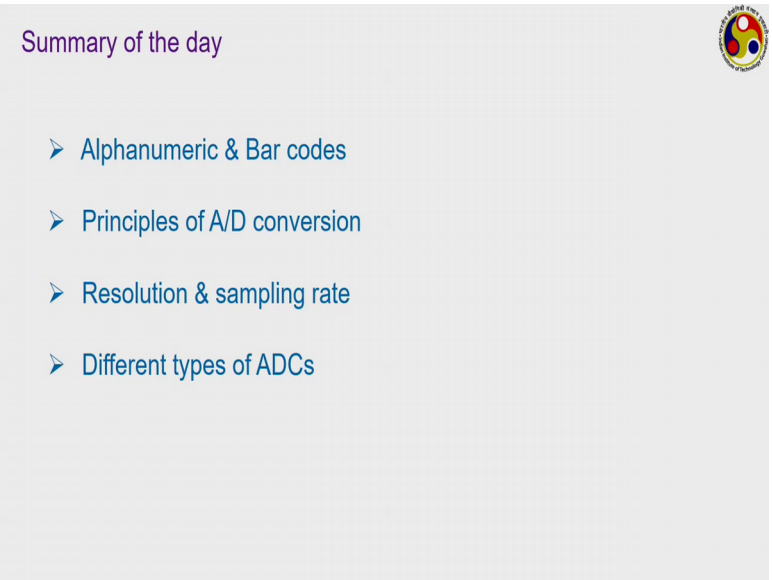
The final one I shall be measuring very very briefly; it is called the sigma delta ADC or sometimes the opposite the delta sigma ADC. Delta in Greek letter represent the difference, sigma represents summation that is what this one does. It has one integrator which does the summation, one comparator which take the difference and then 1 bit DAC.

The DAC single is subtracted from the analogue signal and fed to the integrator here this is the integrator, this your difference. And depending on the output of this it is supplied to this DAC which converts 1 bit at a time and then we get a stream of 0s and 1s. From where through the filtering procedure the final output I am not going to the detailed procedure of this because this is a bit complicated and not frequent also from the per view of this course.

But its advantages; is it gives the highest possible resolution and also very high accuracy it permits noise shaping and also much less number of components; particularly when you compare with the successive approximation one. But it is again a bit slow because we get just 1 bit at a time; it has a limited input bandwidth. Sigma delta ADC probably is

a the most accurate one among the all that we have discussed, but the successive approximation one is the one that is most commonly used in general purpose.

(Refer Slide Time: 60:29)



So, that takes us to the end of the day where we have discussed about the alphanumeric and barcodes and we have discussed the principles of AD conversion, the role of the resolution and sampling rate and finally, we have discussed about different types of ADCs.

We have discussed primarily about 5 different t types of ADCs, the digital ramp, then the integrating ADCs, successive approximation ADCs, flash ADCs and also the sigma delta ADCs. Flash or parallel ADCs and successive approximate these 2 are the most popular one whereas; sigma delta is the most accurate one. So, that is it for the day in the next lecture which I hope to be a smaller one, where we shall be discussing about the other process that is the digital to analogue conversion to finish of this particular weeks form.

So thanks for your attention; we shall be meeting very very quickly.