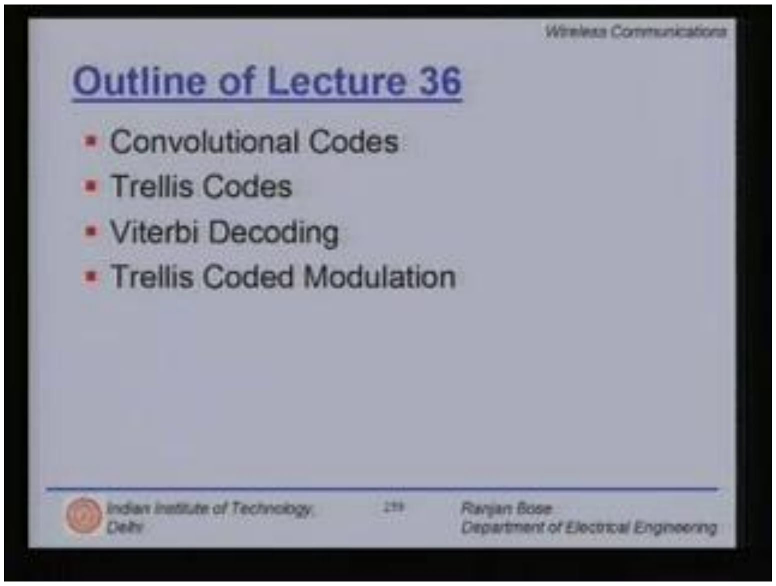
Wireless Communications

Dr. Ranjan Bose
Department of Electrical Engineering
Indian Institute of Technology, Delhi

Lecture No. # 36
Coding Techniques for Mobile Communications

Welcome to the next lecture on wireless communications. Today we will look at yet another class of error control codes called convolutional codes. First the outline for today's talk.

(Refer Slide Time: 00:01:28 min)


Wireless Communications

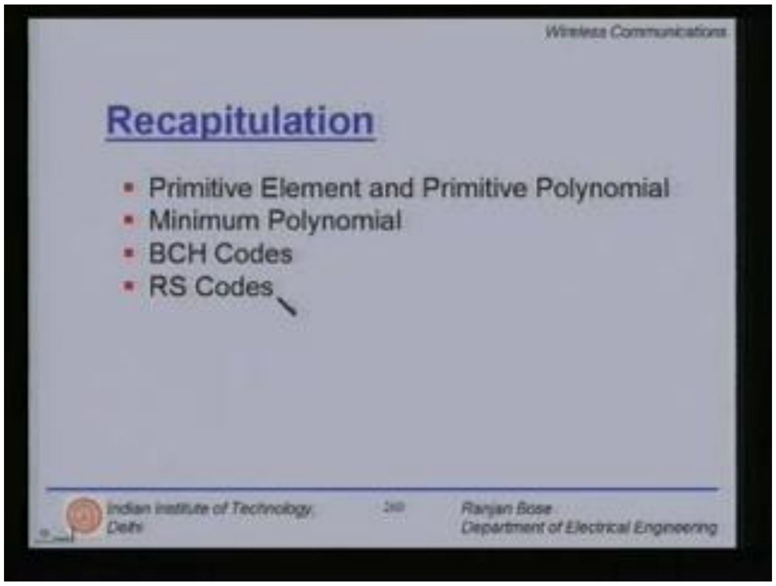
Outline of Lecture 36

- Convolutional Codes
- Trellis Codes
- Viterbi Decoding
- Trellis Coded Modulation

Indian Institute of Technology, Delhi   259   Ranjan Bose, Department of Electrical Engineering

We will start with an introduction to convolutional codes which are essentially codes with memory. We will talk about trellis codes followed by Viterbi decoding which is the standard decoding procedure for trellis codes and then we will have a brief peak at trellis coded modulation which is an attempt to combine error control coding with modulation. So this is the brief outline for today's lecture. So let us see what we have learnt so far, we have talked about cyclic codes in our previous lectures and before that we discussed the linear block codes. We have also realized that cyclic codes form a sub class of linear block codes and BCH codes form a sub class of cyclic codes. We also studied Reed Solomon codes which are again a subclass of BCH codes. We talked about the encoding process for all of them. We learnt how to make the generator matrix and generator polynomials for these codes and also some fast decoding procedures.

(Refer Slide Time: 00:02:07 min)



Wireless Communications

Recapitulation

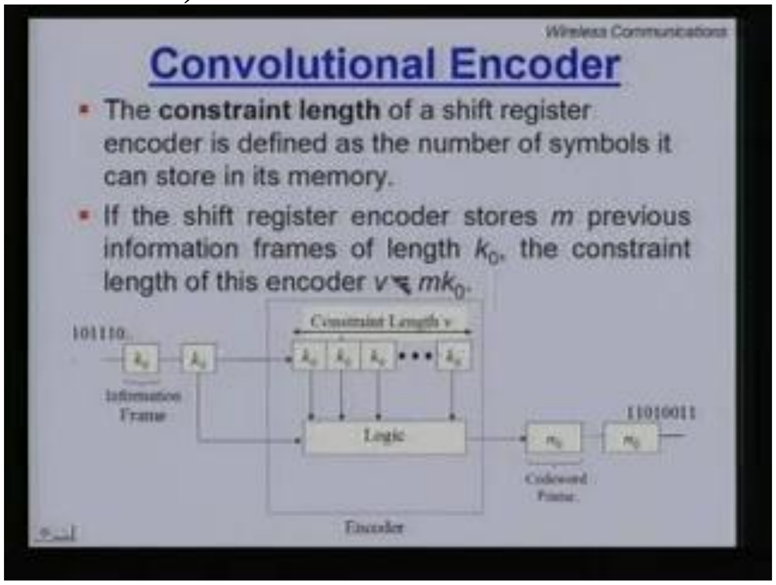
- Primitive Element and Primitive Polynomial
- Minimum Polynomial
- BCH Codes
- RS Codes

Indian Institute of Technology, Delhi 2

Ranjan Bose
Department of Electrical Engineering

Specifically we discussed the primitive element concept, the primitive polynomial and minimum polynomial required to construct the generator polynomials for Bose Chaudhuri Hocquenghem codes called BCH codes. Then we discussed RS codes, please note all of them are linear block codes but these are essentially codes without memory. The whole philosophy of coding is to add redundancies in a known manner which is done using linear block codes. However we can add another dimension to it which is the memory part that means the same input bit stream may not be encoded as the same output bit stream; if you are ensuring that a memory system with memory is used. For linear block codes if you have the same input information word, you will always have the same code word. This is not going to be the case. It should depend on the input stream, the state of the memory and these two will decide what is going to be the output. So let us talk about the convolutional encoders.

(Refer Slide Time: 00:04:03 min)



Wireless Communications

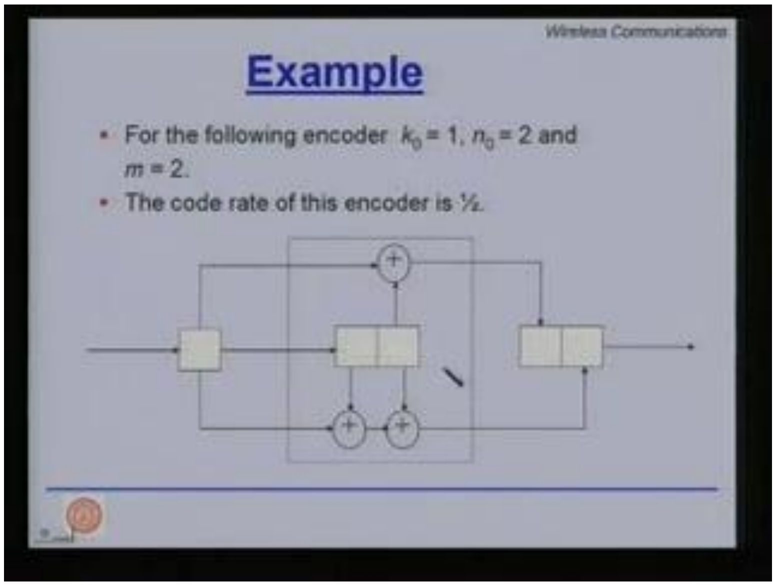
Convolutional Encoder

- The **constraint length** of a shift register encoder is defined as the number of symbols it can store in its memory.
- If the shift register encoder stores *m* previous information frames of length k_0, the constraint length of this encoder $v = mk_0$.

First of all please note that we are going to start with a shift register, the basic memory block. Let us say the number of elements in this memory block is nu where it is measured in terms of k_0 the information frame. What is the information frame? We are processing certain number of bits at a time which are being placed inside the memory and the memory is much larger than the set of bits coming in input to the memory then there is a logic element based on this logic you send out the codeword frames. Please note that codeword frame is larger than the information frame. In fact k_0 over n_0 represents the code rate.

Now the output, the codeword frame depends on two things, the input frame or the information frame and the elements in the memory which have certain bits stored inside them. Together they decide what we are going to output. So the constraint length of a shift register encoder is defined as a number of symbols it can store in its memory. If the shift register encodes m previous information frames of length k_0 as shown here, the constraint length of the encoder nu is equal to m times k_0. You will find several other definitions of the constraint length in the literature this is one of the popular one's.

(Refer Slide Time: 00:05:49 min)

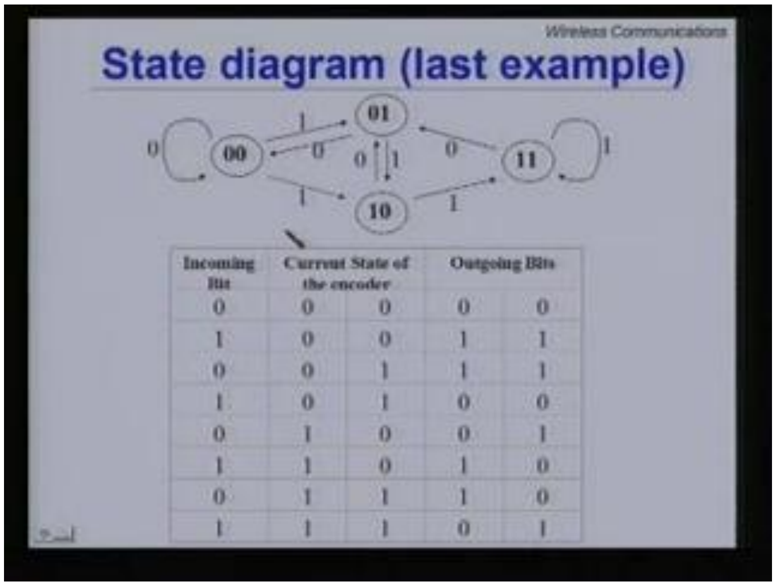

Wireless Communications

Example

- For the following encoder $k_0 = 1$, $n_0 = 2$ and $m = 2$.
- The code rate of this encoder is ½.

Let's look at a simple example of a code with memory. So we have a shift register with two bits and then we take one bit at a time and we do some kind of a basic processing and output two bits. so the number of bits in the codeword frame is 2, the number of bits in the information frame is 1 so k_0 is 1, n_0 is 2 and the rate of this encoder is 1 by 2. This is at a stark contrast to the linear block codes where the n and the k's are usually large hundreds may be thousands. Here we are talking about one bit input and two bit output this is one of the fundamental difference. So here in this encoder as we discussed k_0 is ,1 n_0 is 2 and the code rate is half, it is possible for us to construct a state diagram based on what is the state of the encoder, what input comes in and what is the state transition. Clearly there will be four states in the trellis that will construct using the encoder shown here because the states can be 0 0, 0 1, 1 0 and 1 1 the only 4 possible states.

(Refer Slide Time: 00:07:23 min)

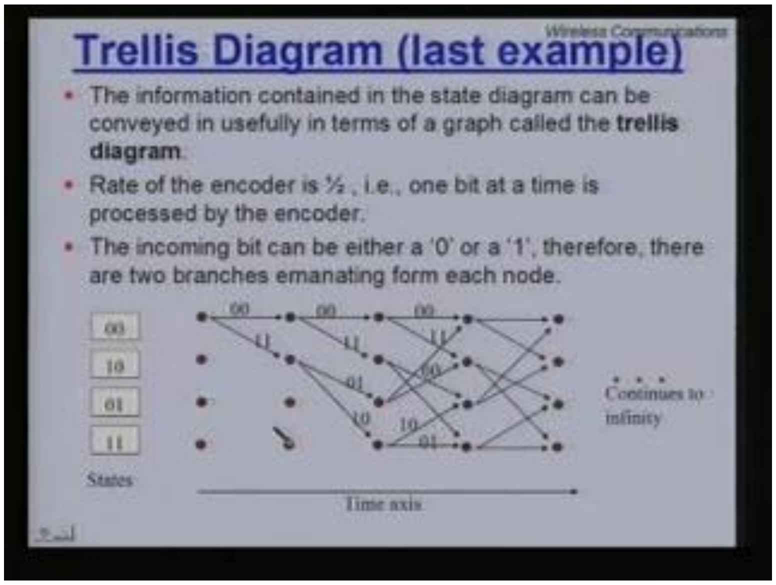


State diagram (last example)

Incoming Bit	Current State of the encoder		Outgoing Bits	
0	0	0	0	0
1	0	0	1	1
0	0	1	1	1
1	0	1	0	0
0	1	0	0	1
1	1	0	1	0
0	1	1	1	0
1	1	1	0	1

Now here I am depicting the state’s 0 0, 0 1, 1 1 and 1 0 and the state transition and it is fairly simple to note down the current state, the incoming bits carry out the logic here and then go to the next state and the outgoing bits. So it’s easy to construct this table and the state diagram but another way to represent the same information of the state diagram is to use a trellis diagram.

(Refer Slide Time: 00:07:51 min)



Trellis Diagram (last example)

- The information contained in the state diagram can be conveyed in usefully in terms of a graph called the **trellis diagram**.
- Rate of the encoder is ½ , i.e., one bit at a time is processed by the encoder.
- The incoming bit can be either a ‘0’ or a ‘1’, therefore, there are two branches emanating form each node.

What is a trellis? Trellis is a semi-infinite structure starts from here but extends up to infinity. It has certain number of states in our example there are four states 0 0, 1 0, 0 1, 1 1 and the arrows show the state transition and the x axis shows the progression in time.

So the philosophy is as follows, each time a bit comes in into your encoder with memory you always are sitting in some state. Now two things can happen either the state remains the same or the state changes. In the process you will send out certain bits which we write on top of the arrow. So it's fairly easy to understand this trellis diagram which exactly has the same information as the state diagram but in a more simplified manner, in a manner in which we want to use the information, in the manner that we have the progression along the time axis. So if I am sitting at 0 0 and one bit input can come in so whether a 0 comes in I move on the upper arm, if a 1 comes in I move along the lower arm. So if a 0 comes in I stay in state 0 0 which is evident because I was sitting in 0 0 and if a 0 comes in it's a shift register so the last 0 goes out, this 0 goes here and a new 0 fits in you still remain in the state 0 0.

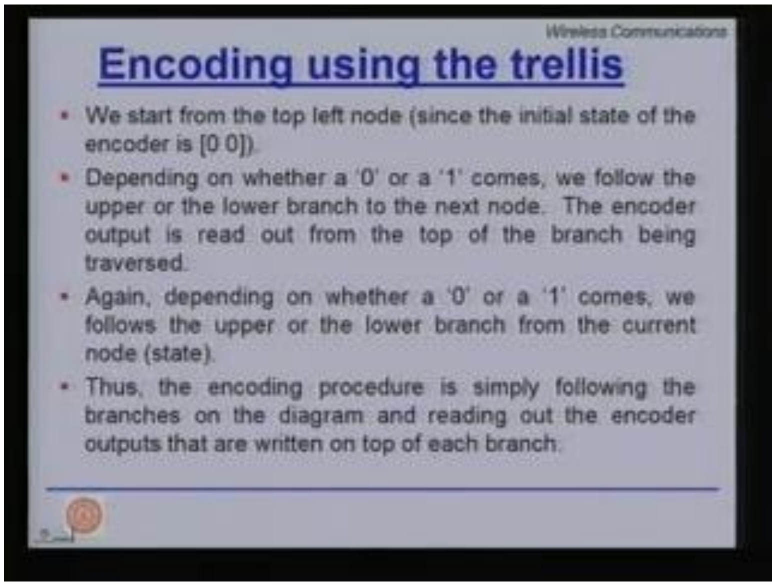
If you look at the state diagram, if a 0 comes in you revert back to the same state and why is it coming like that because state is represented by what is sitting in the memory, this is your memory. If 0 0 was sitting here and it is a shift register another 0 comes in, it will push here this 0 will go here and this 0 will be discarded but for an observer 0 0 state transits back to 0 0. On the other hand if I was sitting in 0 0 and a 1 comes in then this 1 will go here this 0 will go here, the last 0 will be discarded so the next state should be 1 0. So incoming bit is 1, current state is 0 0 the outgoing bits will be 1 1 by logic and the next state will be 1 0. So here if you are sitting in 0 0 and a 1 comes it take the lower path the output is 1 1 and the next state is 1 0 so the dots in the diagram represents the states of the encoder.

If I had a 3 bit shift register I would have 8 states, a 4 bit shift register would give me 16 states. Please note our observations as before coding theory for channel coding deals with introducing structure in the code. So the general philosophy is if you have a lot of structure which is algebraic structure in our case and a noise breaks the structure by changing some one's to zero's vice versa then it is easy to identify where the structure was broken and then substitute back and try to do error correction. So very layman's version an intuitive explanation how coding works but the structure can be only put up at the expense of adding redundant bits. In this case we add another dimension to the structure, we give a geometrical structure on top of an algebraic structure.

This is a very geometrical thing in fact certain transitions are forbidden if you see if I am from 0 0 there is no way I can go directly to 1 1, I cannot have this state transition but the noise doesn't know this noise is random. So many times it will appear at the decoder that the transition from 0 0 goes to 1 1 because of the random bit errors. Clearly you can detect it and you can also locate what is the most probable transition that you should have been through, thereby correcting the error. So added another dimension of structure that's why trellis codes can be stronger than the linear block codes in certain cases. So the information contained in the state diagram can be conveyed usefully in terms of the graph called the trellis diagram. In this case the rate of the encoder is half because one bit comes in and two bit goes out.

So the incoming bit can either be 0 or 1 therefore there are only two branches emanating from each node. If 2 bits were coming in, had k_0 been 2 then you will should have 4 branches coming from any node corresponding to 0 0 bit coming in, 0 1 bit coming in 1 0 or 1 1 bit coming in.

(Refer Slide Time: 00:13:37 min)


Wireless Communications

Encoding using the trellis

- We start from the top left node (since the initial state of the encoder is [0 0]).
- Depending on whether a '0' or a '1' comes, we follow the upper or the lower branch to the next node. The encoder output is read out from the top of the branch being traversed.
- Again, depending on whether a '0' or a '1' comes, we follows the upper or the lower branch from the current node (state).
- Thus, the encoding procedure is simply following the branches on the diagram and reading out the encoder outputs that are written on top of each branch.

So why do we use this trellis structure? It seems both encoding or at least the visualization of the encoding and the decoding is extremely easy using this trellis diagram. First of all the idea is that you start with the top left node which is the state 0 0 depending upon which bit comes in, whether a 0 or a 1 comes in you choose whether to go on the upper branch or the lower branch, 0 if you want to traverse the upper branch, 1 implies go for the lower branch and then you go to the next node and then you can carry forth this kind of a mechanical process.

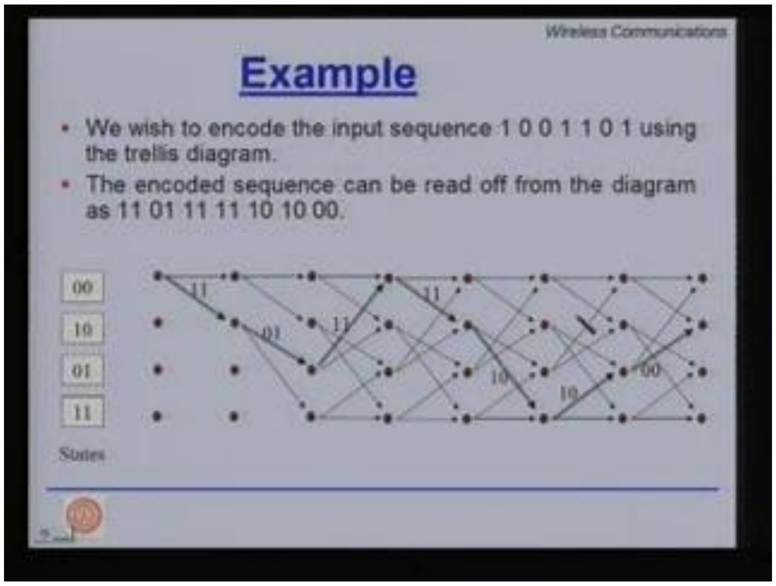
So if you want to look at a trellis and see that you start with a 0 0 and if a 1 comes in you go here and then a 0 comes in you take the upper branch and if the 1 comes in take the lower branch and if a 0 comes in take the upper branch. So what is the input bit stream doing? Input bit stream is telling you which branch to choose. At every bifurcation you have to take a decision and the input bit stream tells you which direction you can move that means we make a very important observation. Any input bit stream corresponds to a unique path in the trellis because the bit stream is telling me choose upper, lower, lower, upper, lower and so and so forth.

A different bit stream will let me choose upper, upper, lower, upper, another one. So there is a unique correspondence between an input bit stream and a path in the trellis which means if you want to jump ahead and look ahead we'll see the decoding process is nothing but identifying the most likely path in the trellis. So it's a search of the paths question. [Conversation between student and the professor: (Refer Slide Time: 16:00)] Sir in this case the third laws in the last row, yeah. There is a third element, yeah. So from that one if the 1 0 is there it should be the lower one zero one the upper one, you are talking about this node, no below that node, this one. So from that one zero is there I mean we are going in the one zero.

So let us look at this node. The question being asked is how does the transition happen if you are sitting at this node? What do we mean by sitting at this node? The encoder has the memory elements containing 1 1 that's why the state 1.

Now two things can happen either a 0 can come in or a 1 can come in. If a 0 comes in here, this is in a shift register, the right most one will be discarded the left one will occupy the place of the right one and a 0 will come in. So you will clearly go to state 0 1 so you will go to state 0 1. what is written on the branch tells you after the logic what is output. So in the process if I ever sitting on 1 1 and a 0 comes in I would go up to 0 1 but what will be thrown out of the encoder is 1 0. On the other hand if I am sitting at 1 1 another 1 comes in, my state transition will not happen but I will shoot off something different. So coming to the encoding we have realized that encoding process is nothing but following a particular path within the trellis.

(Refer Slide Time: 00:17:40 min)


Wireless Communications

Example

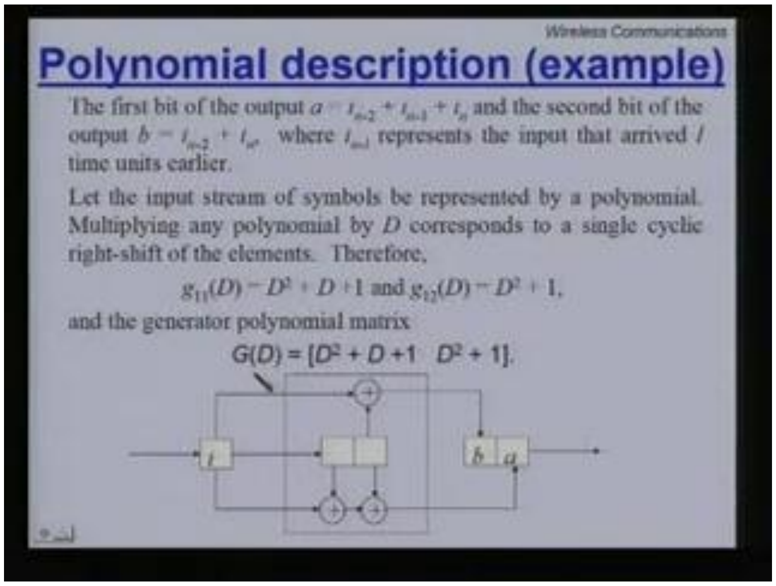
- We wish to encode the input sequence 1 0 0 1 1 0 1 using the trellis diagram.
- The encoded sequence can be read off from the diagram as 11 01 11 11 10 10 00.

Let's look at an example. So here is a four state trellis, just by looking at the trellis we can say a couple of things. The first thing we can say is that there are only two elements in the memory hence the four states. The other thing we can say is that from every node there are only two branches emanating that means the input frame size is 1 but we cannot say anything about how many bits are being thrown out. Of course we have written it down that over every branch we have labeled two bits. So n_0 is 2, by depiction we have shown that two bits are thrown out for every bit that comes in. So we have been able to identify a lot just by looking at the trellis diagram. So if you are asked to design a convolution encoder, it suffices to give the trellis diagram and vice versa.

Suppose we wish to encode the sequence 1 0 0 1 1 0 1 using the following trellis diagram. We start here that is the starting point initialization 0 0 and then the moment we see a 1, we take the lower path we are in this node then the next bit comes in, its 0 we take the upper branch and then 0 again we take the upper branch and then we get a 1 we take the lower branch. Another 1 we take another lower branch and upper branch and lower branch. So this sequence of bits at the input corresponds to the darkened lines in the trellis diagram, not only that I didn't have to go and read out or do the calculations and again and again using the logic to see what will be the output bit stream. I read it out 1 1 0 1 1 1 1 1 1 0 1 0 0 0 that's my encoded output for this bit stream.

Question: Sir in that previous diagram can we change the logic, internal logic. Yes, so the question being asked is for any trellis, this trellis or the previous trellis can we change this logic? The answer is yes, in fact that decides what is a good encoder and not so good encoder. In fact what is written out on the branches, what is output depends upon what is the logic we use. So we have used binary additions here, exhorts if we combine not these two but let alone just this one and this one then I will get something else on the output. We will find out or we'll find a way to categorize which is a good trellis and which is not so good trellis but yes if you change the logic what you write on the branches will change. So now let us go back to our encoding process here and we have observed that any input bit streams corresponds to a unique path in the trellis.

(Refer Slide Time: 00:21:13 min)



Wireless Communications

Polynomial description (example)

The first bit of the output $a = i_{n-2} + i_{n-1} + i_n$ and the second bit of the output $b = i_{n-2} + i_n$, where i_{n-l} represents the input that arrived l time units earlier.

Let the input stream of symbols be represented by a polynomial. Multiplying any polynomial by D corresponds to a single cyclic right-shift of the elements. Therefore,

$$g_{11}(D) = D^2 + D + 1 \text{ and } g_{12}(D) = D^2 + 1,$$

and the generator polynomial matrix

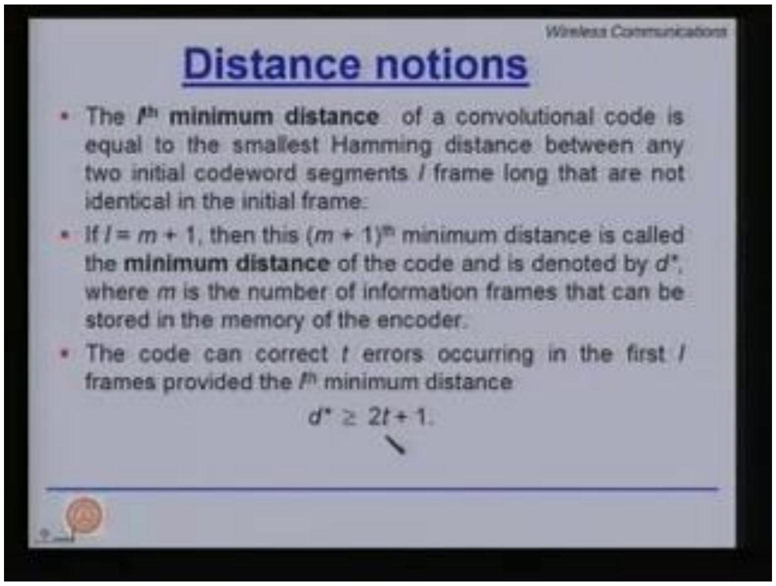
$$G(D) = [D^2 + D + 1 \quad D^2 + 1].$$

So you can also have a polynomial description of the convolutional codes and what you can do is represent it in terms of a delay. So in this simple diagram that we have been using as an example a and b let these two be the 2 output bits. So a can simply be represented in terms of i and the delayed version of that. So if you look at the logic a is nothing but i + a delayed version of i plus two delayed versions of i. So $i_{n-2} + i_{n+1} + i_n$ whereas the second bit b is nothing but $i_{n-2} + i_n$ for any. So let the input stream of symbols be represented by a polynomial, multiplying polynomial by D corresponds to a single cyclic right shift of the element. Therefore what we can write is g_{11}, what does g_{11} imply? Yes it is the generator polynomial which corresponds how the first bit one is associated to the first output bit, g_{12} represents the relationship between the first input bit and we have only one input bit to the second output bit.

Had there been two input bits and three output bits you will have g_{12}, g_{13}, g_{21}, g_{22}, g_{23}. So this is an efficient way to represent how the input and output are related. The power of D tells you how much is the delay if you multiply it with by D, you increase the delay by one, you increase the right shift by one. So you can define the generator polynomial matrix. Please observe the words we have talked about generator polynomial before for cyclic codes, we have talked about generator matrix for linear block codes.

We are now defining something called as generator polynomial matrix which basically has these two components g_{11} and g_{12} here in GD. I can specify my entire convolutional code using the generator polynomial matrix.

(Refer Slide Time: 00:23:52 min)



Wireless Communications

Distance notions

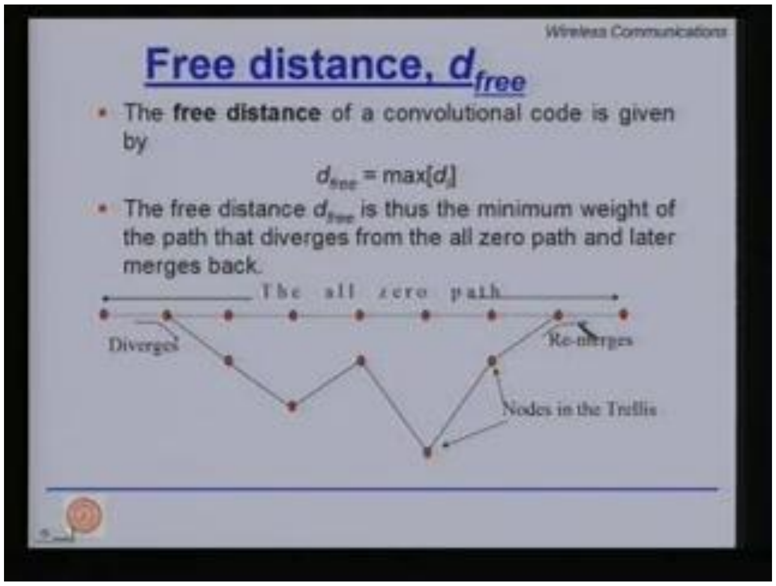
- The l^{th} **minimum distance** of a convolutional code is equal to the smallest Hamming distance between any two initial codeword segments l frame long that are not identical in the initial frame.
- If $l = m + 1$, then this $(m + 1)^{th}$ minimum distance is called the **minimum distance** of the code and is denoted by d^*, where m is the number of information frames that can be stored in the memory of the encoder.
- The code can correct t errors occurring in the first l frames provided the l^{th} minimum distance

$$d^* \geq 2t + 1.$$

Now let us have the distance notion and these are a matter of definition, the lth minimum distance of a convolutional code is equal to the smallest hamming distance between any two initial codeword segments l frame long that are not identical in the initial frame, so it's a definition. So what are we trying to do? Why do we need these definitions? We need to know how good or bad is our convolutional encoder. The first thing to observe is that any sequence of bits corresponds to a path in the trellis. An error will be made when you mistake one path for the other path because decoding process requires you to pick and choose the most likely path that means we must have a notion to have difference between the two paths in terms of a hamming distance. How do you show two paths are different?

A convenient measure is the hamming distance between the two paths. If all the paths are very different they do not resemble each other then a few errors introduced by the noise will not make one path look like another path in the trellis. That is you will not mistake one bit sequence for another bit sequence. So we need a distance notion essentially to tell you how different two paths in the trellis are. The code can correct t errors occurring in the first l frames provided the lth minimum distance d* is greater than 2t +1 this notion comes directly from our observation in the case of linear block codes. That is if the two paths are separated by d* distance then t errors can be corrected in that segment if d* is greater than or equal to 2t +1 same logic.

(Refer Slide Time: 00:25:57 min)


Wireless Communications

Free distance, d_{free}

- The **free distance** of a convolutional code is given by

$$d_{free} = \max[d_l]$$

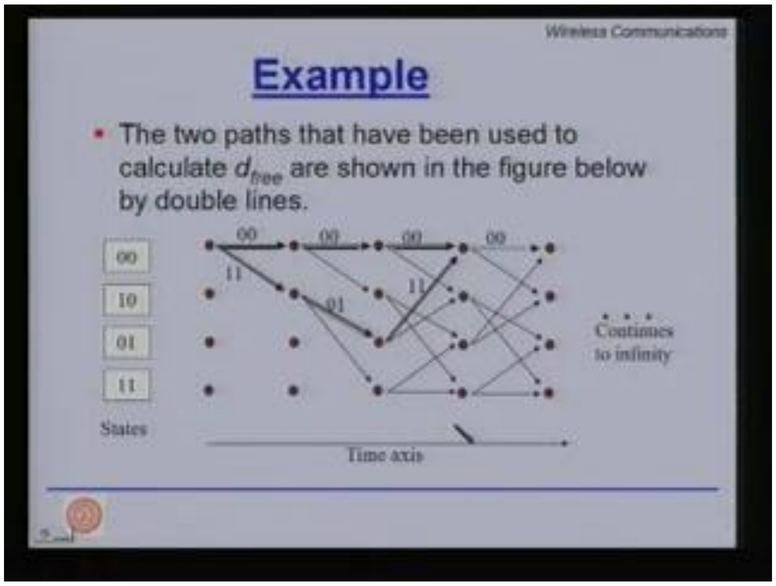
- The free distance d_{free} is thus the minimum weight of the path that diverges from the all zero path and later merges back.

The all zero path
Diverges
Re-merges
Nodes in the Trellis

Now let us define a very interesting concept which is called d_{free} or the free distance of a convolutional code. The free distance of a convolutional code is given by max d_l, the free distance is thus the minimum weight of the path that diverges from the all zero path and later merges back, for a linear code this holds true. That is if you are talking about a linear convolutional code then all distances from the all zero path are representative distances. So all you have to do is take the first path which is the all zero path, you will see zero zero's written on the top of it it's called the all zero path. It's possible that one of the paths diverges goes through ups and downs and then finally remerges, segment by segment for these two segments there will be some distance hamming distance.

What is hamming distance? The number of places the vectors are different. What are the vectors we are talking about? Whatever numbers you write the vectors, you write on the branch whatever is the output bits is what you write on the branch. We are talking about the number of places these vectors differ. So we go segment by segment, how far are these in terms of the hamming distance, how far this segment is with this segment, how far is this segment with this segment and so and so forth. That will give me the distance between the two nodes. Now you measure all possible distances between the pairs of paths, for a linear convolutional code it's good enough to compare all possible distances from the all zero path, whichever is the minimum is called the minimum free distance of the convolutional code.

(Refer Slide Time: 00:28:09 min)

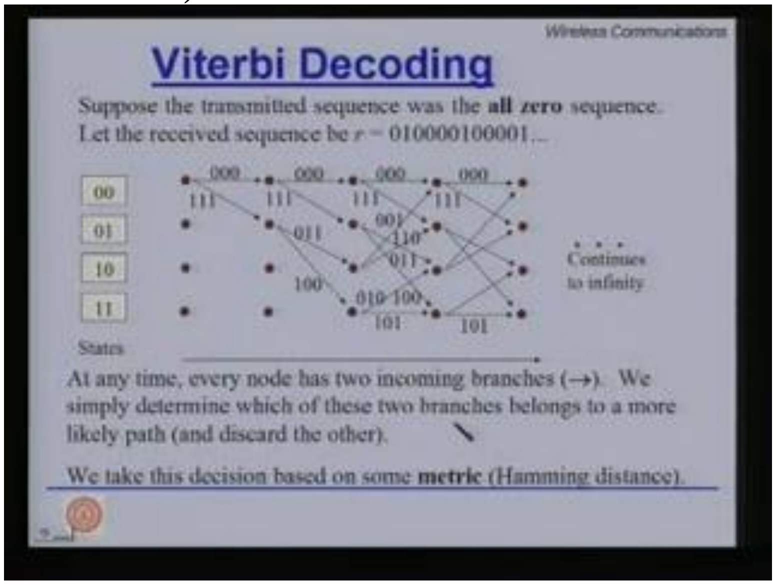


Example

- The two paths that have been used to calculate d_{free} are shown in the figure below by double lines.

Let’s look at an example. It’s a four state trellis two inputs coming, two bits come in and so for every node you have two branches emanating out and you also have written two bits on every branch which depicts that two bits are thrown out for every input bit that comes in. Now from the all zero path if you can see zeros are written on all the branches, is the all zero path. We have been able to demarcate one particular branch which diverges and after some steps merges back. So if you calculate the distance, the hamming distance between these two paths 0 0, 1 1 differ at two places plus 0 0, 1 0 1 differ at one places. So 2+1, 3 + 0 0, 1 1 differ at two places plus 2 so 2+1+2 is 5 so 5 is the distance between all zero path and this path that diverges. You can carry out this exercise for all other pairs of paths, whichever is the minimum and incidentally whatever has been depicted forms a pair of paths which gives you the d_{free} so the d_{free} for this convolutional encoder is 5.

(Refer Slide Time: 00:29:39 min)



Viterbi Decoding

Suppose the transmitted sequence was the **all zero** sequence.
Let the received sequence be r = 010000100001...

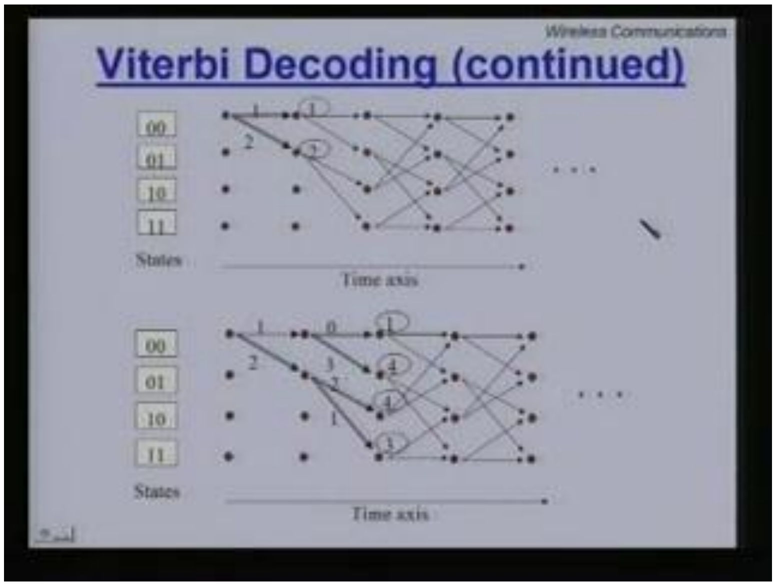
At any time, every node has two incoming branches (→). We simply determine which of these two branches belongs to a more likely path (and discard the other).

We take this decision based on some **metric** (Hamming distance).

Now let us talk about something called as Viterbi decoding. It was proposed by Andrew Viterbi, he was one of the founding members of Colcom. This is one of the basic and popular techniques to do decoding for trellis codes. The objective is very simple, the objective is to find out the most likely path in the trellis but clearly if it keep track of all the paths, the number of path grows exponentially as we move along the time axis. So there has to be a smart way to overcome this difficulty of retaining all possible paths. Their philosophies as follows and input bit stream comes in and we have to keep matching it branch by branch with the inputs that are written on top of the branches, so as we proceed and we will see it through this example. We keep on defining a metric and keep track of so far what is the most likely paths possible.

The other thing to remember is that the correct path that we are searching for must pass through a set of nodes, all paths pass through a set of nodes. So identifying a certain path tantamounts to identifying the set of correct nodes that means we don't have to keep track of what are the correct paths, if we can see what are the correct nodes. So at each node we take a decision; for example look at this node that two branches coming into this node. Clearly only one will be correct if we have a metric, if we have a measure to find out which is the more likely candidate we only retain that and discard the other one. Thereby we have proven the research logically and optimally. This is based on the philosophy of dynamic programming where you divide the problem into sub parts and each sub part is optimally solved so that the overall solution is optimal.

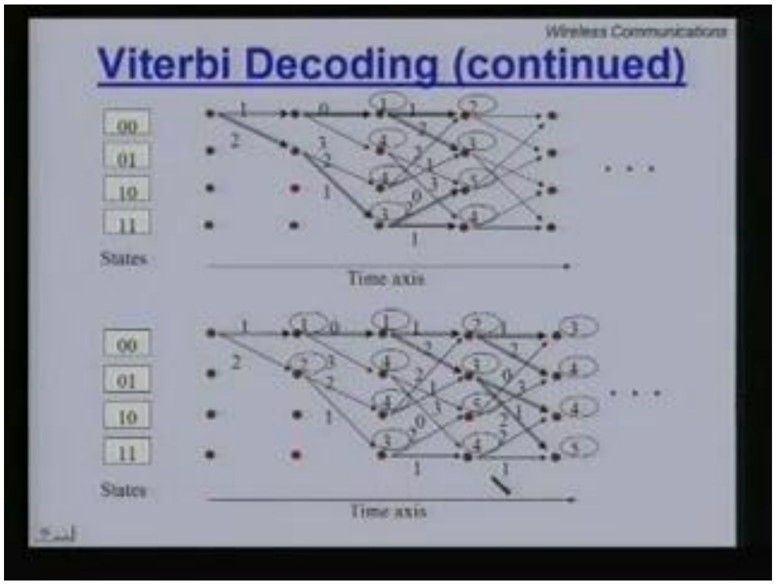
(Refer Slide Time: 00:32:17 min)



So let's look at the decoding algorithm. Suppose in this first step the metric in our case it is the hamming distance between the received bit stream and the vector written on the top of the branch. So if you go back, suppose the received vector is the following 0 1 0 0 0 0 so and so forth 1 and so and so forth and for the sake of illustration only let actually the all zero vector be sent and there a few errors being present, the ones represent the error. The question is when we pass it through the Viterbi decoder how does it get back to the original stream that we send.

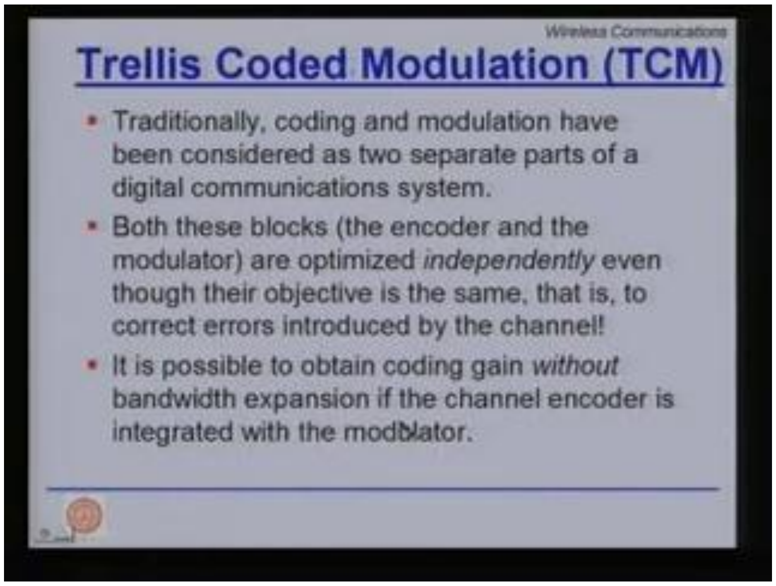
So if you go back you see 0 1 0, since there are three bits written on every branch it implies that n_0 is 3, two branch is coming out from every node so k_0 is 1 so it is 1 by 3 encoder that we are talking about. For this one if you look at the metric between the first three bits because we'll take three bits at a time because three bits are written on every branch. The distance between the first branch and this first three bits received is one whereas between these three and the first three bits it's two, so my metric for the first two cases will be 1 and 2. If you proceed from here onwards, the next three bits are zeros so if you add to one zero metric because no more addition took place you still get one but here along this one you have 1 1 1 but here the next three bits were 0 0 0. So the metric goes up by three so 3+1, 4 is what we'll write here, so this can be done mechanically for all the branches that has been shown in this example, here 1 and 2 is what we got. Further metric was 0 so 1+0 is 1 but one on this branch we calculated it was 3 so 1+3 is 4 and so and so forth so you can write the metrics for all but we clearly not stop here.

(Refer Slide Time: 00:34:51 min)



We continue this process and very soon we encounter the third stage where two or more branches, in these example only two branches are converging at this place. We go back to our philosophy that correct path should pass through correct nodes. So we must take decisions at a node, at this node I have to choose between this one or this one and here we can safely retain that branch which has the lower branch metric and discard the other one because passing through that node only that branch would be optimal which is so far optimal and we continue this exercise beyond and beyond. So we only retain those many paths that corresponds to those many nodes in a state, here only 4. The size of my memory which retains the number of paths does not increase exponentially it is limited to the number of states.

(Refer Slide Time: 00:36:07 min)


Wireless Communications

Trellis Coded Modulation (TCM)

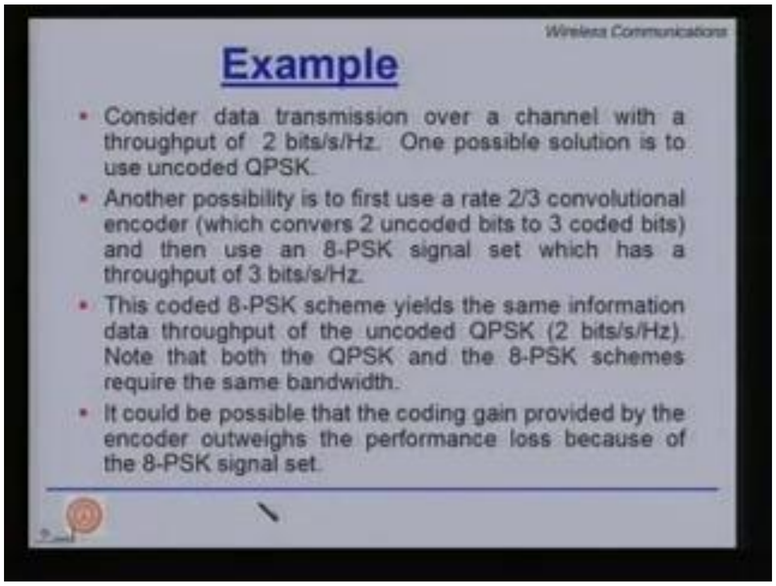
- Traditionally, coding and modulation have been considered as two separate parts of a digital communications system.
- Both these blocks (the encoder and the modulator) are optimized *independently* even though their objective is the same, that is, to correct errors introduced by the channel!
- It is possible to obtain coding gain *without* bandwidth expansion if the channel encoder is integrated with the modulator.

Now we move one step forward and try to combine coding with modulation. As a specific example we'll talk about trellis coded modulation however you also have block coded modulation which combines linear block codes with modulation. What is the reason for doing this? So far we have seen that the encoder block, the channel coding block and the modulation block have been optimized separately however one of the objectives is the same for both error correction or immunity to noise. A different modulation schemes have been there to overcome the effects of noise as well, so BPSK performs better in additive white Gaussian noise than QPSK for example. There also the different modulation schemes have been designed that for a given power what is the best noise performance you can get.

Surprisingly the objective of error control coding is also to overcome the effects of noise. So in 1982 Ungerboeck proposed combing the channel coding block and the modulation block together and optimize them together to come up with coded modulation and it has been found that the performance of the coded modulation is better than separately optimizing coding and modulation. The other interesting philosophy is as follows when you do error control coding you add redundant bits. A redundant bits would require extra bandwidth. Question is can I combine modulation where I can play with the bandwidth requirement and error control coding together.

So that I can have an improved performance without spending more on bandwidth, it's like asking something for nothing and TCM promises exactly that, you can get additional noise performance without asking for more bandwidth. So traditionally coding and modulation have been considered as two separate blocks, both these blocks the encoder and the modulator are optimized independently even though their objective is the same that is to correct errors introduced by the channel. Of course modulation also has another purpose to send the message across but to do so without errors. It is possible to obtain coding gain without bandwidth expansion if the channel encoder is integrated with the modulator and optimized together.

(Refer Slide Time: 00:39:07 min)


Wireless Communications

Example

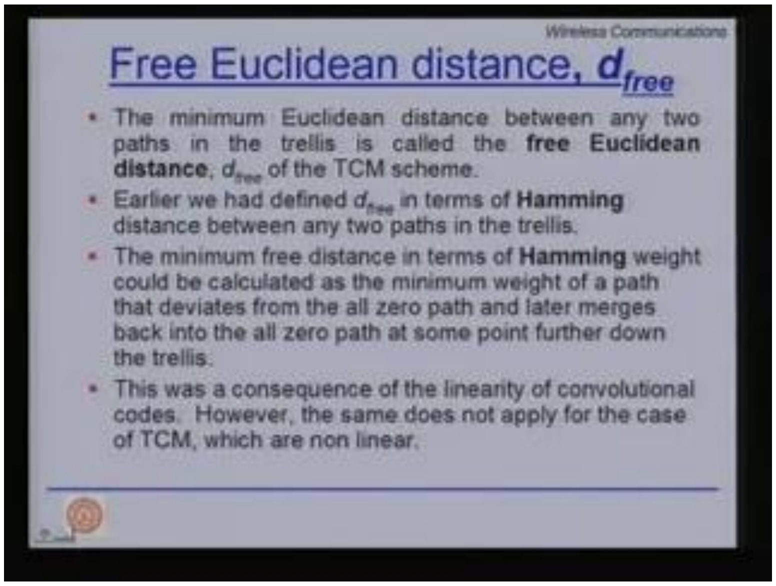
- Consider data transmission over a channel with a throughput of 2 bits/s/Hz. One possible solution is to use uncoded QPSK.
- Another possibility is to first use a rate 2/3 convolutional encoder (which convers 2 uncoded bits to 3 coded bits) and then use an 8-PSK signal set which has a throughput of 3 bits/s/Hz.
- This coded 8-PSK scheme yields the same information data throughput of the uncoded QPSK (2 bits/s/Hz). Note that both the QPSK and the 8-PSK schemes require the same bandwidth.
- It could be possible that the coding gain provided by the encoder outweighs the performance loss because of the 8-PSK signal set.

Consider data transmission over a channel with a through put of two bits per second per hertz. So basically we have to have two bits per symbol. The general idea is to take QPSK because QPSK will take two bits per symbol and send it across. given the signal to noise ratio you will get a certain probability of error. The other possibility is to take two bits at a time and pass it through a 2 by 3 encoder my convolutional encoder. So now you already have some redundancy because now you have one extra bit, so for two bits you get three bits out but we now do not use QPSK but we use 8 PSK which can take 3 bits per second. So I did not ask for any extra bandwidth, approximately QPSK and 8 PSK would consume the same amount of bandwidth approximately. So what did we do? In the constellation diagram if you are not going to expand more power we have brought the constellation points closer together.

So by going from QPSK to 8 PSK we have actually increase the probability of error. We know that however before doing that we have already added some redundancy by using a convolutional encoder. Now convolutional encoder will try to do error correction thereby reducing the probability of error, moving from QPSK to 8 PSK has brought the constellation points closer thereby increasing the probability of error. If I have done my math correctly then the advantage gained by coding will more than over compensate the loss that I am encountered by moving my consolation closer. If that is true then I can get performance improvement without increasing the bandwidth. If however I do not optimize properly, I might end up losing.

So the 8 PSK scheme yields the same information data throughput of the uncoded QPSK but note that both QPSK and 8 PSK schemes require approximately the same bandwidth. It could be possible that the coding gain provided by the encoder outweighs the performance loss because of the 8 PSK signal set and that is exactly done it is doable and trellis coded modulation can really outperform without asking for extra bandwidth.

(Refer Slide Time: 00:42:04 min)



Wireless Communications

Free Euclidean distance, d_{free}

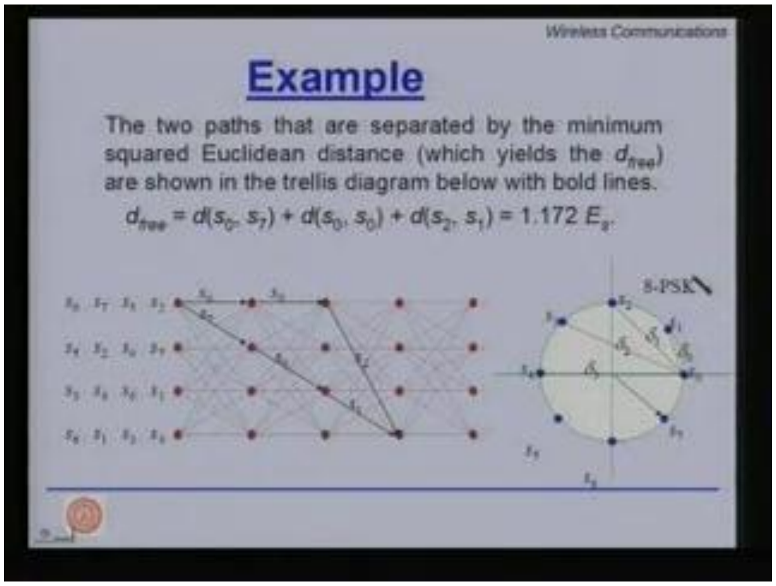
- The minimum Euclidean distance between any two paths in the trellis is called the **free Euclidean distance**, d_{free} of the TCM scheme.
- Earlier we had defined d_{free} in terms of **Hamming** distance between any two paths in the trellis.
- The minimum free distance in terms of **Hamming** weight could be calculated as the minimum weight of a path that deviates from the all zero path and later merges back into the all zero path at some point further down the trellis.
- This was a consequence of the linearity of convolutional codes. However, the same does not apply for the case of TCM, which are non linear.

Now you have just like the free distance which is actually the free hamming distance in case of trellis codes, you have the free Euclidian distance. The definition is the same but this time instead of hamming distance you are talking about Euclidian distance between two paths. So you go for the various pairs of path in the trellis and that pair which gives you the minimum Euclidian distance is the d_{free} for trellis coding modulation. Just like free hamming distance is used to tell how good is your trellis code, similarly free Euclidian distance tells you how good is your trellis coded modulation but one word of caution convolutional codes are generally linear by definition.

The construction that we do we use linear adders, a shift registers. So you end up making a linear code. on the other hand the nonlinear it is introduced in the mapping process because you are mapping the three bits for example to 8 PSK symbols that is a nonlinear process so just going with the all zero path may not give you the free Euclidean distance for TCM but for linear convolutional codes having free distance is simply taken from the all zero path. Here for TCM you must consider all possible pairs of paths. Why is it called the convolution encoder? If you look at the convolutional encoder it looks like a filter which performs convolution to the input bit stream hence the name convolutional filter it’s like an LFSR.

Let’s look at an example of a TCM. Here we have 4 states but for every state you have 4 branches coming out. Clearly four states indicate that the two bits are present in the memory of the TCM encoder. However two input bits are coming in contributing to 4 branches coming out of every node. Now for the output bits we don’t send them we map it back to one of the eight possible 8 PSK. So in this example two bits come in into the convolutional encoder, 3 bits come out but each of the 3 bits are mapped to one of the possible 8 PSK symbols. So instead of writing three bit outputs on top of the branches we write symbols, so for any 2 bits that come in a symbol goes out, we have combined the encoder and modulator.

(Refer Slide Time: 00:44:06 min)


Wireless Communications

Example

The two paths that are separated by the minimum squared Euclidean distance (which yields the d_{free}) are shown in the trellis diagram below with bold lines.

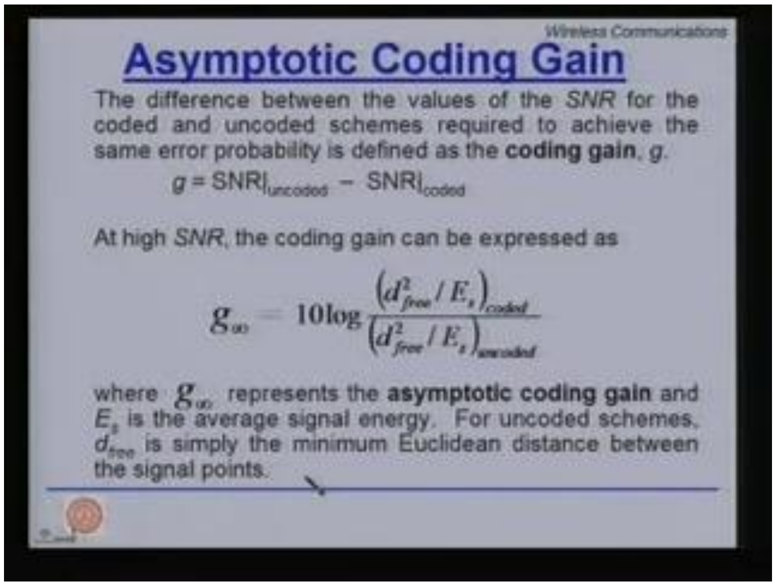
$d_{free} = d(s_0, s_7) + d(s_0, s_0) + d(s_2, s_1) = 1.172\ E_s.$

Now if you want to find the free Euclidian distance if you have done the exercise and these two pairs of path represent the minimum Euclidian distance. Here you can see that the all zero path is not there, this is clearly not an all zero path this is not an all zero path. So it happens that the minimum Euclidian distance is does not involve the all zero path because it is a nonlinear code. All you have to do to find the minimum Euclidian distances, find the Euclidian distance between s_0 and s_7, s_0 and s_0, s_2 and s_1 from here and sum total is the minimum Euclidian distance. It comes out to be 1.17 two E_s.

Now let's define something called as an asymptotic coding gain, it is a difference between the values of the SNR for the coded and uncoded scheme required to achieve the same probability of error that's called the coding gain usually defined for high SNR values. So how much will I have to increase my signal to noise ratio in dB, if I have to achieve the same bit error rate if I have no coding performance.

So clearly uncoded will require a larger SNR, with coding you require a less SNR so you can trade off SNR for bandwidth but here in TCM we have not even asked for extra bandwidth. So if we define $g_{infinity}$ the asymptotic coding gain as 10 log d_{free} squared over E_s normalized coded divided by d_{free} squared E_s uncoded. Here $g_{infinity}$ represents the asymptotic coding gain and E_s is the average signal energy. For uncoded schemes d_{free} is simply the minimum Euclidian distance between the signal points. Now let us talk very briefly about mapping by set partitioning. So the question is how do you map, what is the best way to assign the 3 bits to the different symbols, how do you write symbols on the branches?

(Refer Slide Time: 00:46:04 min)



Wireless Communications

Asymptotic Coding Gain

The difference between the values of the *SNR* for the coded and uncoded schemes required to achieve the same error probability is defined as the **coding gain**, *g*.

$$g = \mathrm{SNR}|_{uncoded} - \mathrm{SNR}|_{coded}$$

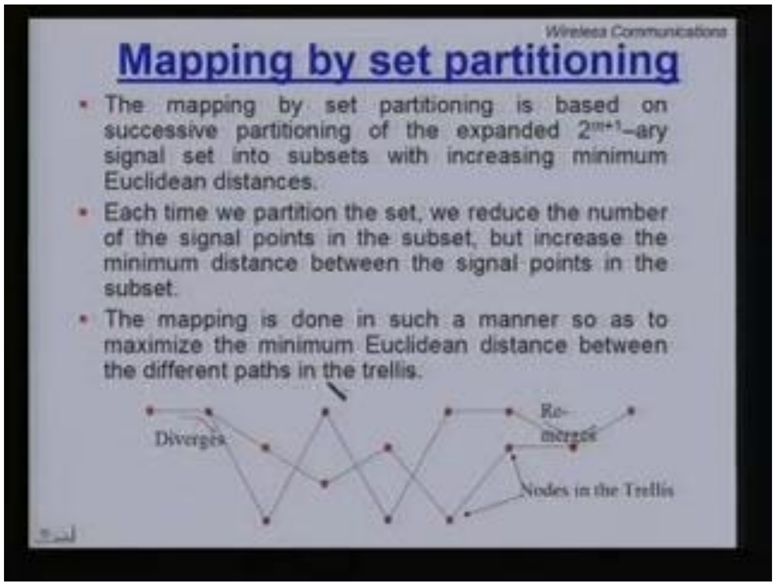
At high *SNR*, the coding gain can be expressed as

$$g_\infty = 10\log\frac{\left(d_{free}^2/E_s\right)_{coded}}{\left(d_{free}^2/E_s\right)_{uncoded}}$$

where g_∞ represents the **asymptotic coding gain** and E_s is the average signal energy. For uncoded schemes, d_{free} is simply the minimum Euclidean distance between the signal points.

If you go back and look at the trellis diagram, here part of coming up with a good code is to put certain symbols on top of the branches. So question we are addressing is how do we put this symbols, why this s_0 and s_7, why not s_4 or s_3, who decides the symbols? So Ungerboeck who had proposed TCM had come up with one of his ad hoc techniques called the set partitioning. So let's look at it, the mapping by set partitioning is based on the successive partitioning of an expanded 2^{m+1} -ary signal set into subsets with increasing minimum Euclidian distance. We'll take an example, each time we partition the set we reduce a number of signal points in the subset but increase the minimum distance between the elements. Mapping is done in such a manner so as to maximize the minimum Euclidian distance between different paths basically get the best possible d_{free} Euclidian.

(Refer Slide Time: 00:47:33 min)

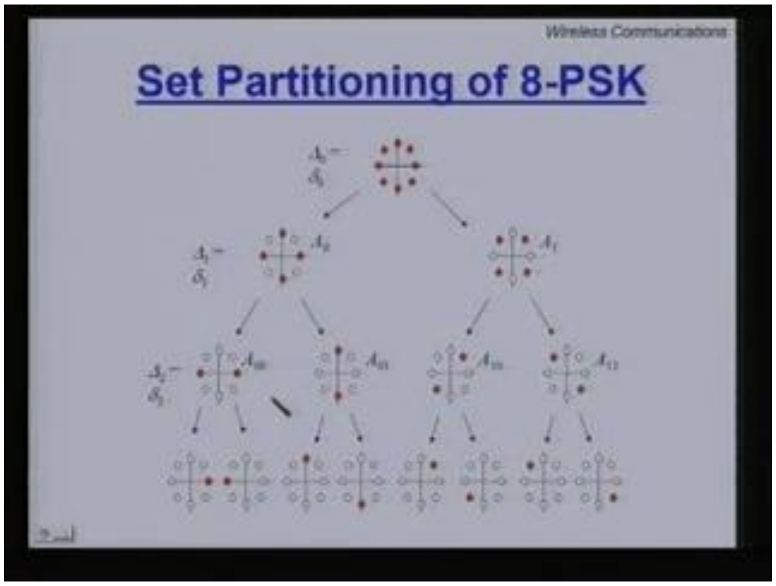


Wireless Communications

Mapping by set partitioning

- The mapping by set partitioning is based on successive partitioning of the expanded 2^{m+1}–ary signal set into subsets with increasing minimum Euclidean distances.
- Each time we partition the set, we reduce the number of the signal points in the subset, but increase the minimum distance between the signal points in the subset.
- The mapping is done in such a manner so as to maximize the minimum Euclidean distance between the different paths in the trellis.

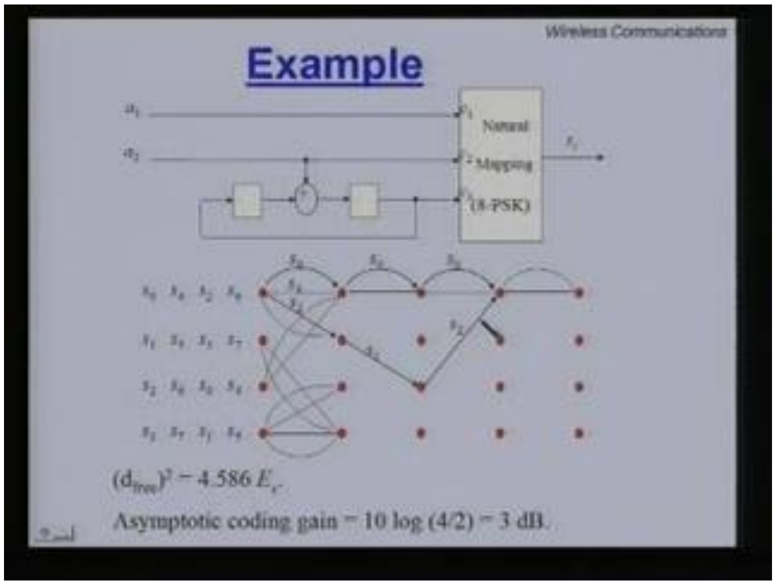
(Refer Slide Time: 00:49:04 min)



Set Partitioning of 8-PSK

Let’s look at the set partitioning of 8 PSK so the basic idea is as follows keep subdividing it into smaller and smaller sets of points but each time you subdivide you increase the minimum Euclidian distance between the points. So here I take the 8 PSK divided into 4 and 4 and then 2 and 2, for each step I increase the Euclidian distance. Now if you look at a simple example, here is an example of a trellis with parallel paths and Ungerboeck gave a set of three rules which says that you assign the diverging paths from the lower sets so that the Euclidian distance is a maximum. So this is s_0 and s_4 then you assign these two to the pair of diverging branches so here s_0 and s_4 for one diverging s_2 and s_6 for the next one and same for the merging band branches.

(Refer Slide Time: 00:49:37 min)

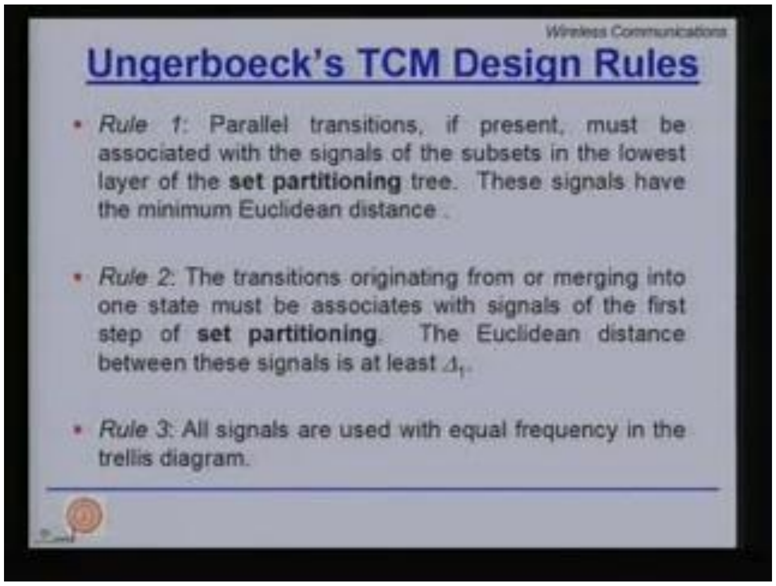


Example

$(d_{free})^2 = 4.586\, E_s$

Asymptotic coding gain = 10 log (4/2) = 3 dB.

(Refer Slide Time: 00:50:27 min)

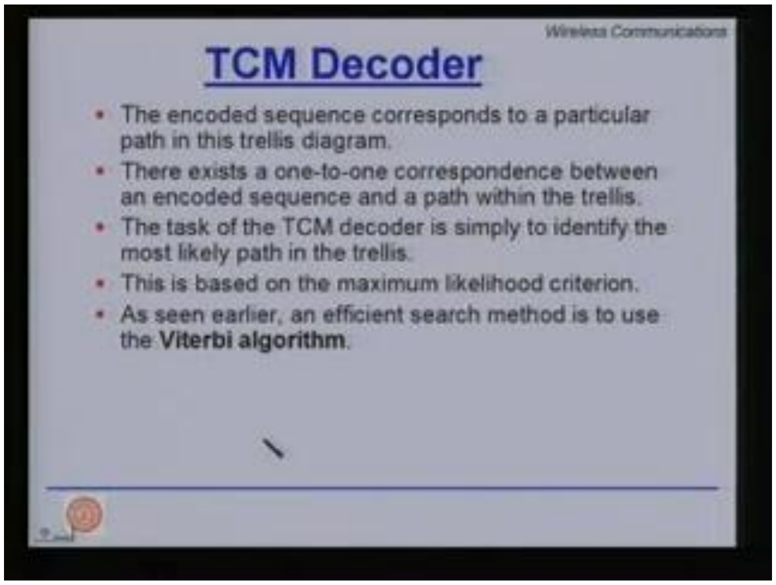

Wireless Communications

Ungerboeck's TCM Design Rules

- *Rule 1:* Parallel transitions, if present, must be associated with the signals of the subsets in the lowest layer of the **set partitioning** tree. These signals have the minimum Euclidean distance .
- *Rule 2:* The transitions originating from or merging into one state must be associates with signals of the first step of **set partitioning**. The Euclidean distance between these signals is at least Δ_1.
- *Rule 3:* All signals are used with equal frequency in the trellis diagram.

So what are the design rules? Rule one for parallel transition if present must be associated with signals of the subsets in the lowest layer of the set partitioning tree, you just saw how to do set partitioning because these signals have the maximum minimum free Euclidian distance. Rule two, transitions originating from or merging into one state must be associated with signals of the first step of set partitioning. All signals are used with equal frequency in the trellis diagram, these are ad hoc roots but work well because this no way we can know what is the best way to assign optimally, all the symbols to all the branches in the trellis. So it talks about the diverging branches, it talks about the remerging branches, it talks about the parallel parts and tries to maximize based on only these three the maximum Euclidian distance.

(Refer Slide Time: 00:51:32 min)

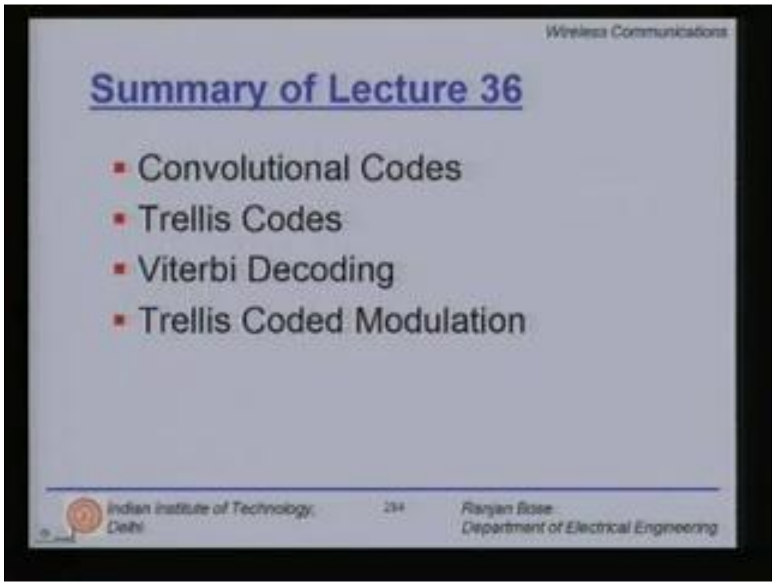

Wireless Communications

TCM Decoder

- The encoded sequence corresponds to a particular path in this trellis diagram.
- There exists a one-to-one correspondence between an encoded sequence and a path within the trellis.
- The task of the TCM decoder is simply to identify the most likely path in the trellis.
- This is based on the maximum likelihood criterion.
- As seen earlier, an efficient search method is to use the **Viterbi algorithm**.

The TCM decoder is similar to the Viterbi decoder but here the metric will not be based on Euclidian, it will not be based on hamming distance, it will be based on Euclidian distance. The task for the TCM decoder is simply to identify the most likely path in the trellis and again we can use the Viterbi decoder to do the decoding part.

(Refer Slide Time: 00:52:00 min)



Wireless Communications

Summary of Lecture 36

- Convolutional Codes
- Trellis Codes
- Viterbi Decoding
- Trellis Coded Modulation

Indian Institute of Technology, Delhi 254 Ranjan Bose Department of Electrical Engineering

Let us now summarize today's lecture. We started off with convolutional codes which are codes with memory, we decided how to represent convolutional codes instead in the form of state diagrams and then finally in the form of trellis diagrams. We learnt how to encode using a trellis diagram as well as decode using a trellis diagram we looked at the Viterbi decoding algorithm and did an example then we moved over to trellis coded modulation which promises to combine channel coding with modulation to give me enhanced performance without asking for additional bandwidth. So we will try to conclude our lecture here and look at certain other aspects of mobile communication in the subsequent lectures.