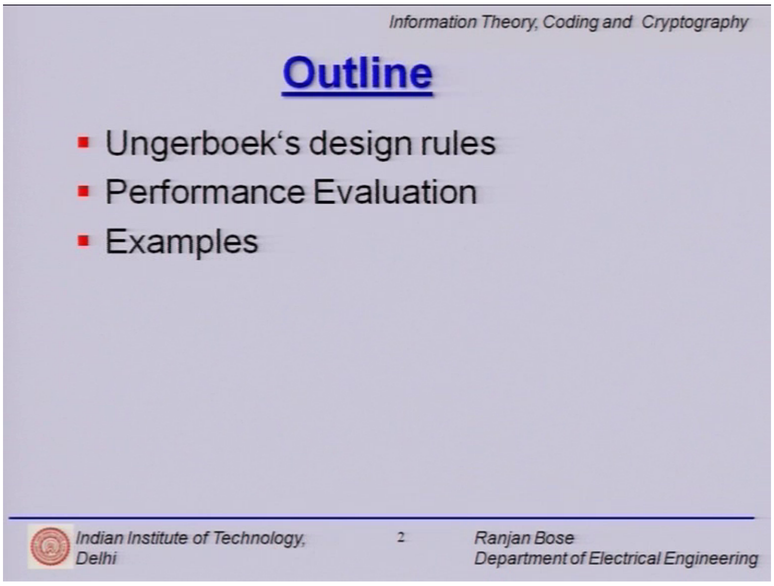
Information Theory, Coding and Cryptography
Dr. Ranjan Bose
Department of Electrical Engineering
Indian Institute of Technology, Delhi

Module - 31
Trellis Coded Modulation
Lecture - 31

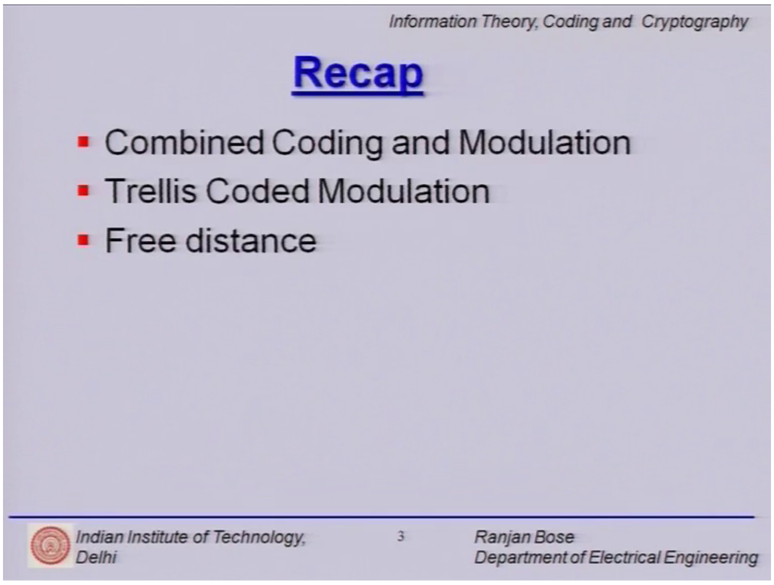
Hello and welcome to our next lecture on Trellis Coded Modulation. Let us start with a brief outline.

(Refer Slide Time: 00:36)



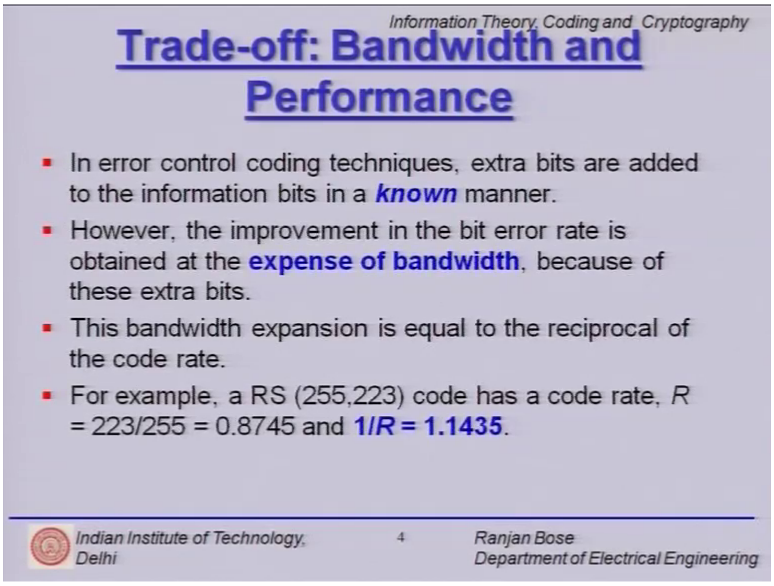
So, we would primarily cover Ungerboek's design rules today and understand how we can design very efficient trellis with good distance properties, and then we would spend some time looking at performance evaluation of TCM schemes over additive white Gaussian noise channels. Of course, we will look at some examples along the way.

(Refer Slide Time: 00:38)



Let us quickly see what we have done so far. We have understood what do we mean by coding and modulation and combining them then we introduced the notion of trellis coded modulation and we introduced the idea of free distance. We will soon see that free distance d free will be the single most important design parameter for trellis coded modulation schemes.

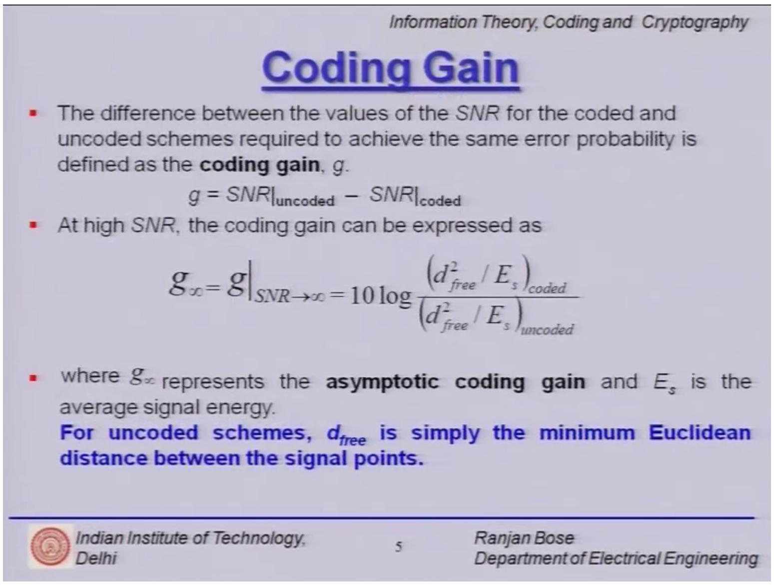
(Refer Slide Time: 01:21)



So, if you remember in error control coding, we introduced extra bits in a known manner to be recovered at the receiver end in order to come back from the errors.

Now, this addition of extra bits came at a cost of additional bandwidth. Therefore, error control schemes always required more bandwidth and we realized that this was inversely proportional to the code rate R.

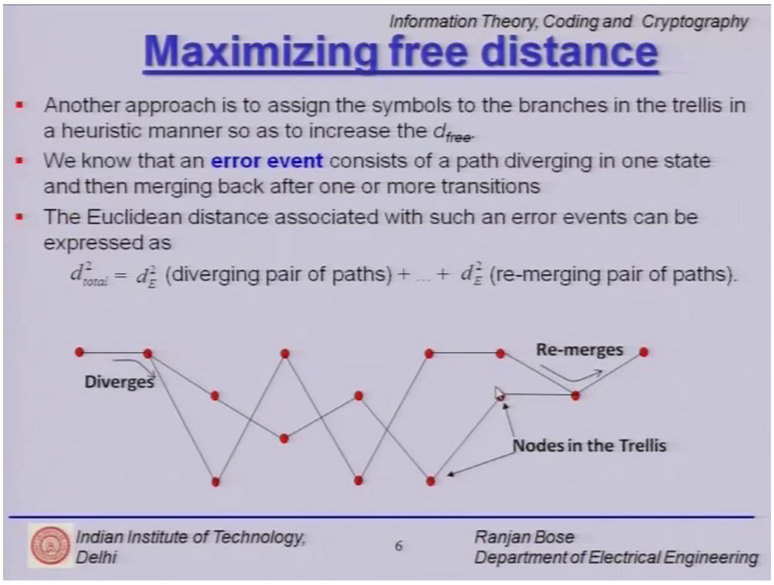
(Refer Slide Time: 01:52)



And, what we decided that in trellis coded modulation scheme we can gain something out of nothing because we can leverage the gain by the error control coding scheme and the modulation scheme together.

So, we defined for ourselves the coding gain where we found out that at a high SNR the coding gain asymptotic coding gain as SNR tends to infinity is defined as 10 log to the base 10 d free over E s normalized coded scheme versus uncoded scheme. For the uncoded scheme deep free is simply the minimum Euclidean distance between the signal points. So, this is this gain that we get because of the trellis d free that as to the advantage and please remember that we have now the trellis labeled by symbols rather than by the bits as in the convolutional encoder case.

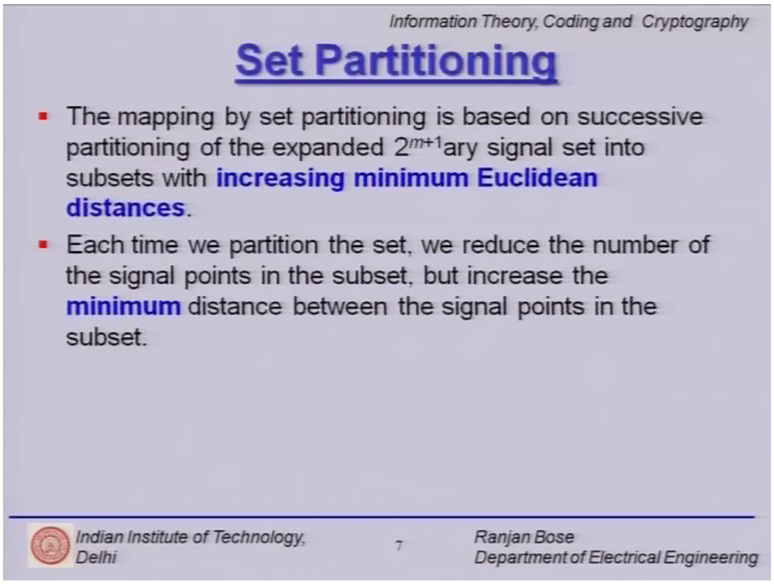
(Refer Slide Time: 02:54)



So, we looked at how to maximize this free distance while designing good trellis coded modulation schemes for that we designed we decided that the error event is when we diverge like this and then we go through the trellis for a couple of hops, and then finally we merge back. So, this constitutes an error event. What happens is we transmit a sequence of bits and it corresponds to a path in the trellis for example, this path and at the decoding end suppose we are using Viterbi we decode another path in the trellis and therefore, this constitutes an error event.

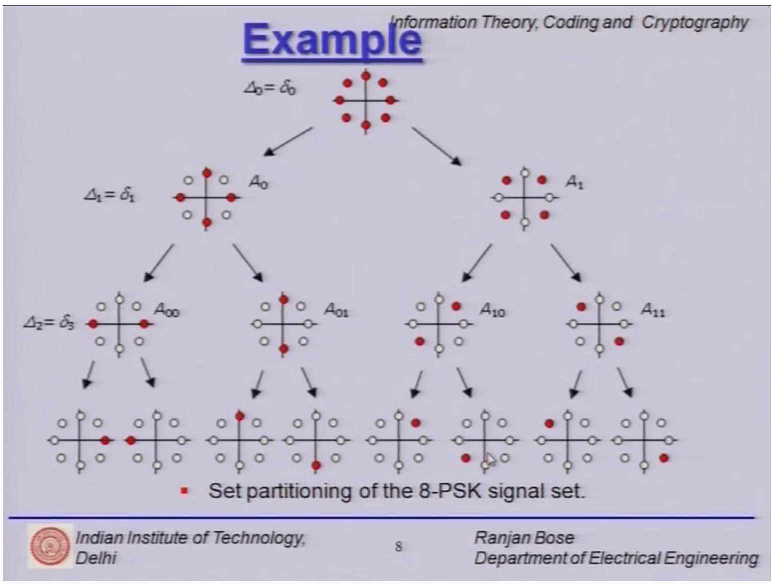
Now, in order to maximize the d free we would like to make sure that the two most closely resembling paths are separated maximally, for that we do not know what happens in between, but at least the diverging and the merging back paths need to be maximally apart in terms of the Euclidean distance. Here we look at the squared Euclidean distance because we take the total of all the branch labels.

(Refer Slide Time: 04:17)



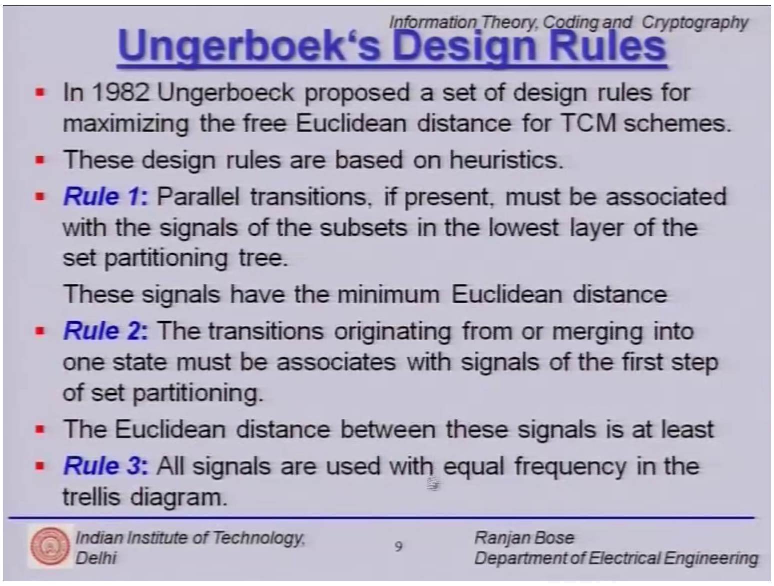
So, we introduce the concept of set partitioning where we consecutively partition a set of the constellation diagram into increasingly minimum Euclidean distances. And what we want to do is to associate the different symbols at different stages of the set partition tree with the branches and label the branches accordingly.

(Refer Slide Time: 04:42)



So, this is a simple example of 8-PSK and how we do set partitioning. So, step one we get into two subsets each one has a larger Euclidean distance and then we continue this further till we get to the maximally separated points. So, this is an example of a set partitioning of 8-PSK signal set.

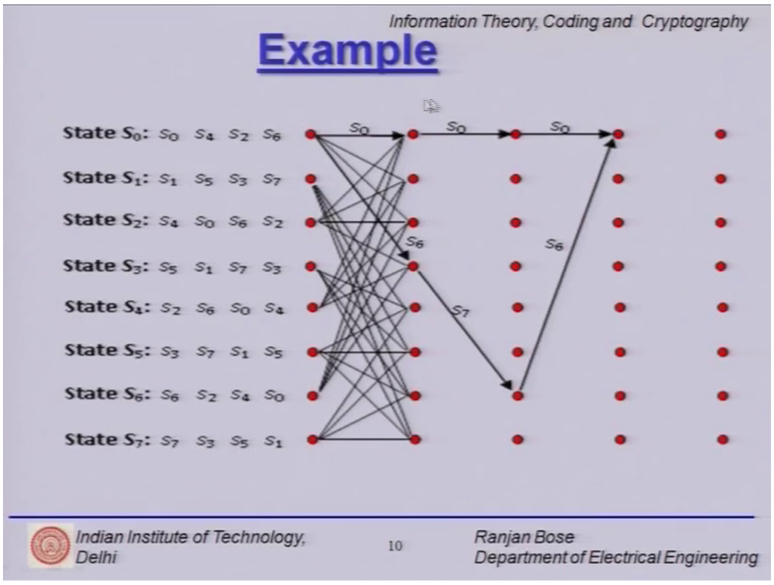
(Refer Slide Time: 05:08)



Now, Ungerboek's design rule tells us in a heuristic manner how to assign symbols to a trellis. This is rule number 1, where the parallel transitions if present must be associated with the signals in the subset of the lowest layer of the set partitioning tree, which means that if you were to have parallel transitions in your trellis then this is the lowest set and if you have parallel transitions then these opposite symbols s 0 and s 4 for example, should be assigned or s 2 and s 6 must be assigned to the parallel transitions and so and so forth.

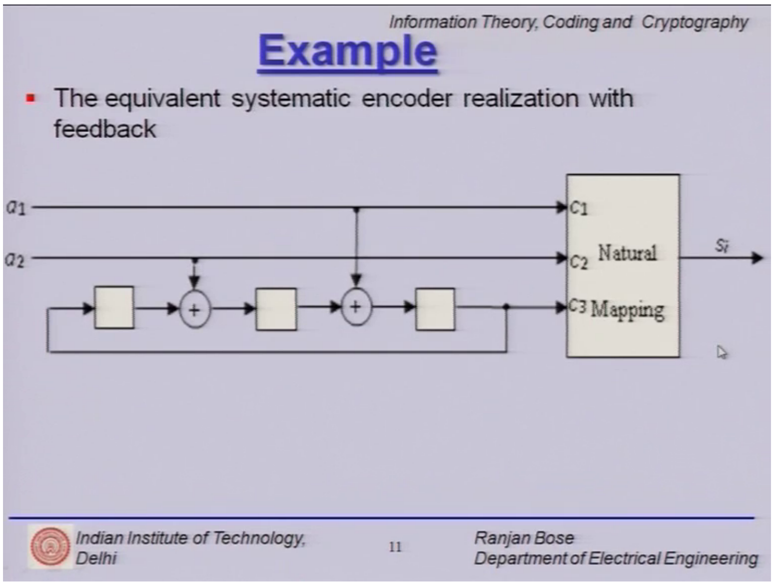
Now, rule number 2 says that the transitions originating from or merging into one-state must be associated with the signals of the first step of the. So, go one step higher first the most damaging ones are the parallel transitions. So, the symbols should be such that they are maximally apart. Then if we do not have parallel transitions then we look at the merging and diverging paths where they should be assigned to the next higher level. And finally, what we must do is try to ensure that all signals are used with equal frequency. This is to our advantage otherwise if we use certain signals too many times then you end up reducing the distance.

(Refer Slide Time: 06:33)



This is an example we did in the last class and how we have been able to assign the diverging paths, and the merging back paths from the sets of symbols which are maximally apart.

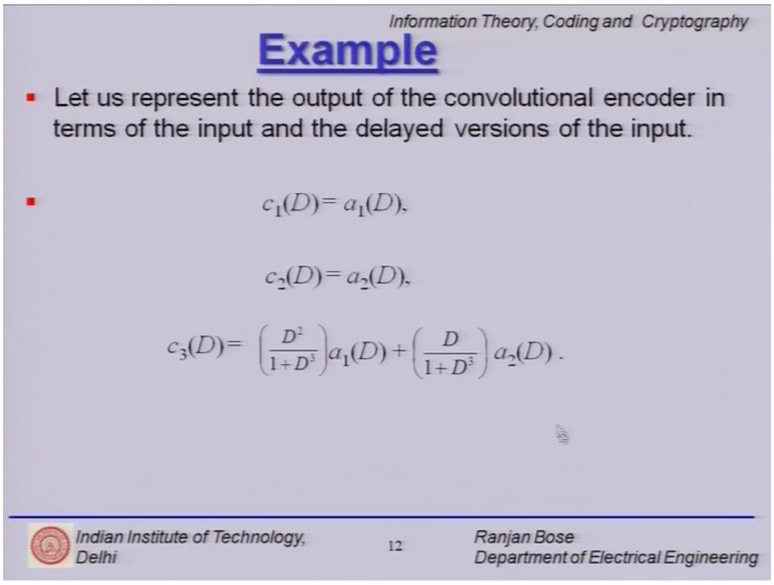
(Refer Slide Time: 06:52)



So, let us look at this example again it is best understood by this example. So, suppose we are looking at a rate 2 by 3 convolutional encoder followed by a natural mapping. If you remember natural mapping means 000 is mapped to symbol s 0, 001 is mapped to symbol s 1 and so on and so forth.

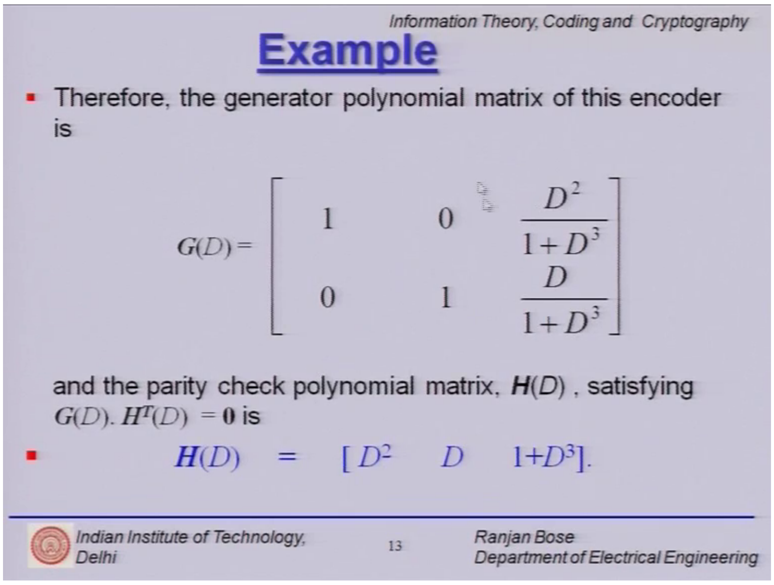
So, clearly there are 3 bits at the output of this convolutional encoder and therefore, we must have 8-PSK in order to convert it into a symbol Si. So, 2 bits come in goes through the convolutional encoder 3 bits come out they are mapped using the natural mapping and one symbol comes out. So, the equivalent trellis will have 8 states because there are 3 memory elements and there will be 2 paths emanating sorry 4 paths because they are 2 bits which are input. So, corresponding to 00, 011, 0 and 1 1 you will have 4 branches coming out and each branch will be labeled by a symbol.

(Refer Slide Time: 07:59)



So, the other way to visualize this is in terms of the delay versions. So, if you look at just the first C 1 it is directly a 1 and C 2 is directly a 2, but C 3 is a 1 with one delay and then a 2 goes through two delays and the sum goes through three delays. So, if you solve this then you can label C 1 as a 1, C 2 as a 2, because they were directly connected, but C 3 if you solve those intermediate equations can be represented in this delayed version.

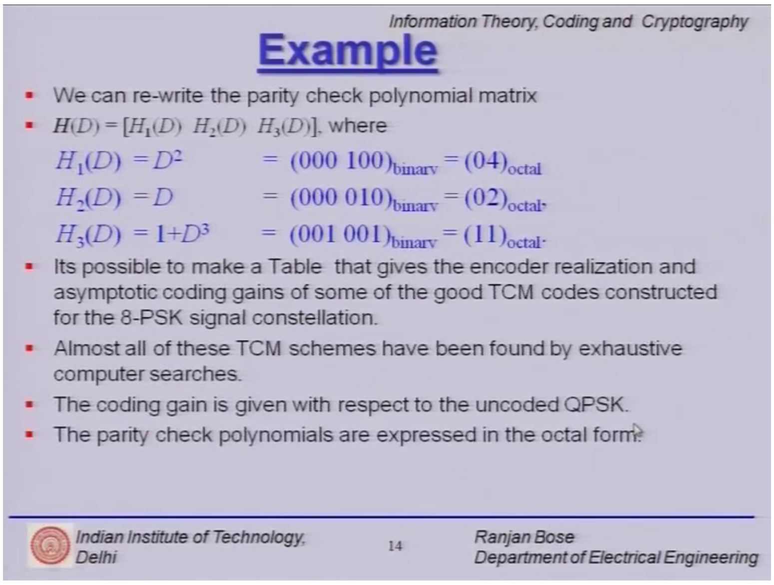
(Refer Slide Time: 08:50)



So, we have an equivalent generator polynomial matrix for this encoder which we studied earlier in convolutional encoder as follows. So, you have a simple representation of this generator polynomial matrix, ok. This unity this one and one here and identity matrix in the beginning shows that it is kind of a systematic encoder.

So, clearly if we have G D we can write out the H D matrix the poly the parity check polynomial matrix H D such that G D into H transpose D should be equal to the 0 matrix. So, you can quickly make an observation and write as follows. So, if you multiply G with H transpose then this D squared is specter by this one D specter by this one so, numerator becomes D squared plus D divided by 1 plus D cubed and you quickly realize that you have GD H transpose D equal to 0. So, it is easy to make H D from G D and G D is equally easy to realize from the visual observation.

(Refer Slide Time: 10:13)

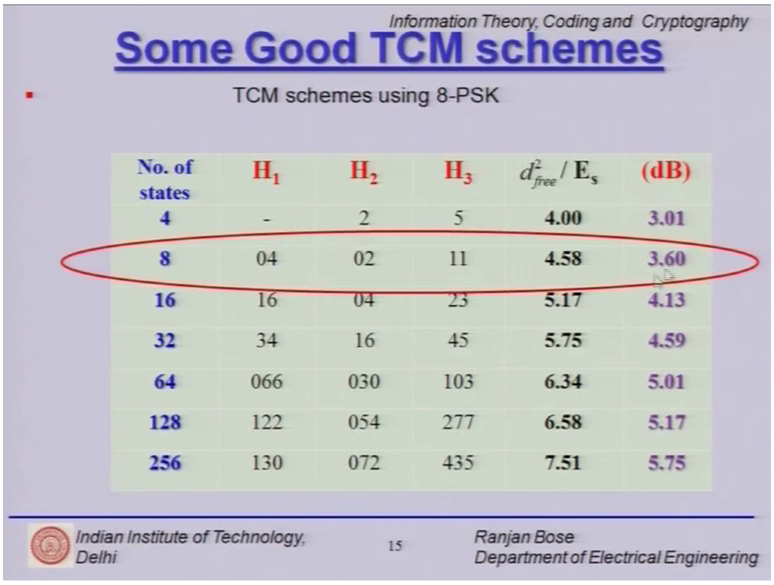


So, now you have this H D matrix in terms of the H 1 D, H 2 D and H 3 D and you can write it in terms of binary or octal as follows. So, D squared. So, this stands for the coefficients of D raised to the power 0, if it is missing. So, it is 0 this is the coefficient for D raised to the power 1, which is missing. So, it is 0 and this is the coefficient for D raised to the power 2 which is present. So, it is 1. So, that is the binary representation and this is the octal representation.

Similarly, H 2 D it is the coefficient for D is there, coefficient for D raised to the power 0 is missing, coefficient is for D squared is missing and therefore, it is 010 and binary and 02 in octal. The first three bits stands for the first digit secondary bits stands for the second digit and H 3 D is 1 plus D squared if you see 1 plus D cubed. So, you have 1 here and this is the coefficients of D cubed; so 1 1 in octal. So, I can represent this simply using the octal notations which is found in the literature, ok.

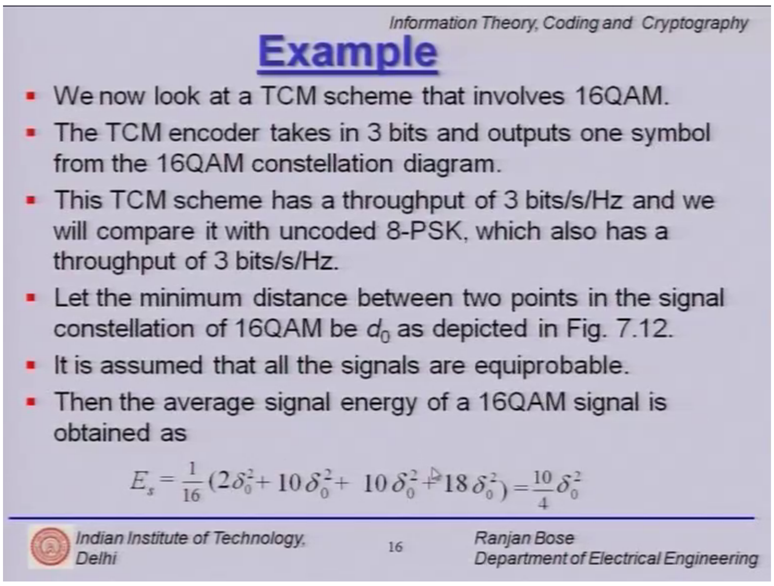
So, it is now possible to form a table which goes the encoder realization and asymptotic gains of some good TCM codes usually constructed from binary searches because there is no hard and fast design rules. So, we can always come up with a table, they have been found by exhaustive computer searches.

(Refer Slide Time: 11:58)



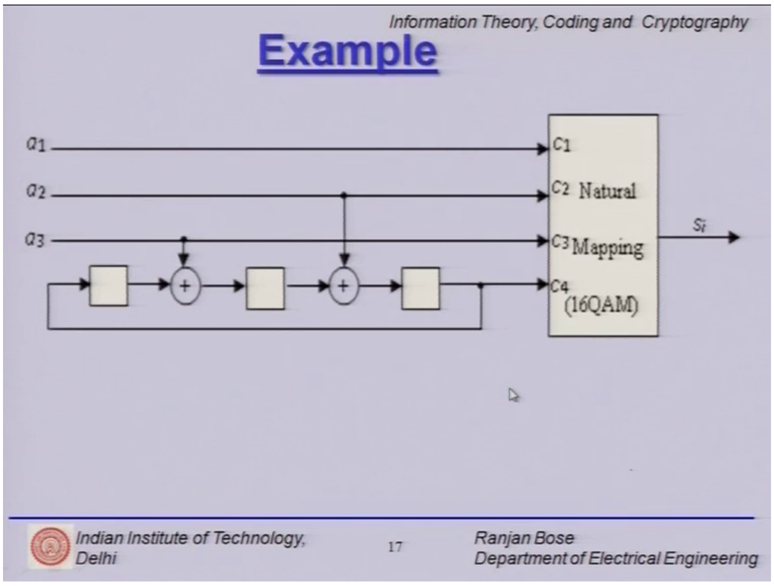
So, we can write them out and document their gain asymptotic gain with respect to QPSK as follows. Here is a table of good TCM schemes of which we have already looked at this number of state 8 H 1 – 4, 2 and 11 in octal notation, where we calculated that this normalized free distance squared is 4.58 leading to a gain of 3.6 dB asymptotic coding gain, but that is not the only one we have, so many other possibilities and by now we understand how to write the encoder using the octal notation as follows. So, this is a set of good TCM schemes using 8-PSK.

(Refer Slide Time: 12:45)



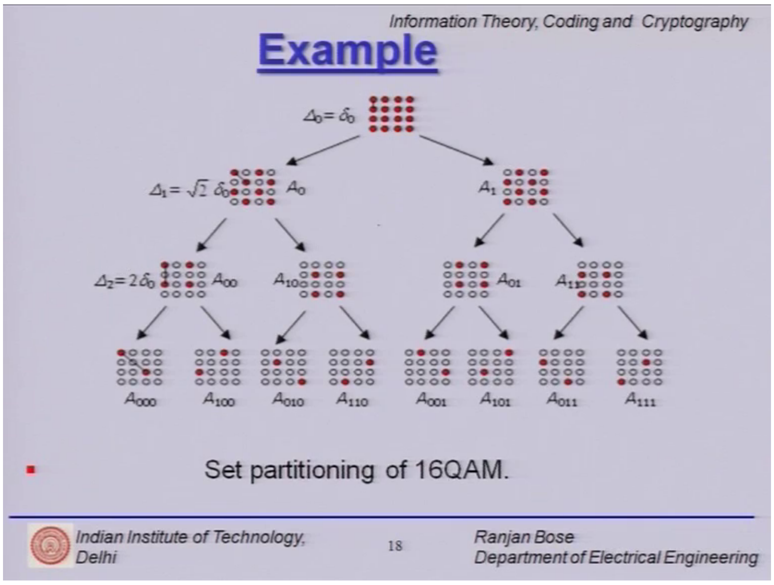
So, now let us look at another example using 16 QAM and this encoder takes in 3 bits and adds one more bits to gives it is a 4 bit output. So, the convolution encoder is a 3 by 4 consequently we cannot use 8-PSK we must use 16 QAM and what we do is that you can first find out what is the average signal energy for 16 QAM. If you remember all the symbols do not carry equal energy because they are not equally spaced from the center or the origin.

(Refer Slide Time: 13:25)



So, this is the example of the rate 3 by 4 convolutional encoder. It takes 3 bits and gives out 4 bits followed by a natural mapping, but this time I need to have a 16 QAM and consequently we get mapped symbol S i out. Again, there are 3 memory elements leading to 8 states in the trellis. So, rate 3 by 4 convolution encoder natural mapping eight-state trellis.

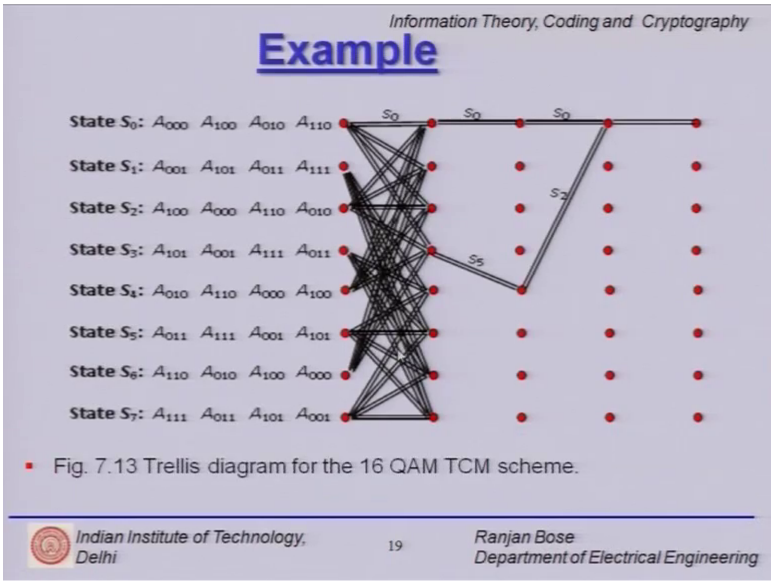
(Refer Slide Time: 13:57)



So, first we have to do set partitioning, but this time we start with the 16 QAM and if you see as a first step I divided into two subsets each one. We, throughout the alternate symbols leading to an increased Euclidean spacing between the symbols as follows and then we do not stop we continue further. So, we get even larger distances between the neighboring symbols. So, first two subsets and then two more for each so, four, total number of subsets and then eight subsets we have continuously increase the distance. So, this is the example for set partitioning of 16 QAM.

Now, we will use Ungerboek's rule to assign the symbols from the correct level of the set partitioning tree to the diverging paths and the merging back paths.

(Refer Slide Time: 14:58)



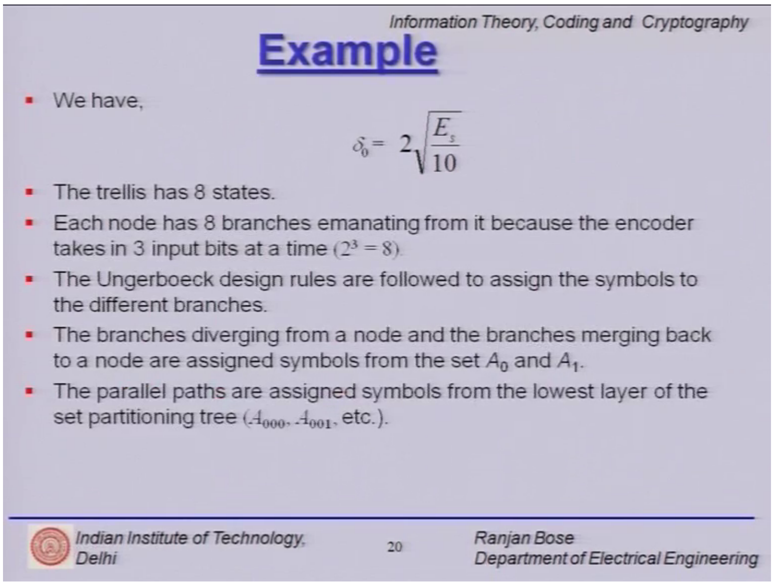
So, this is an example of that convolutional encoder if you see there are parallel transitions here so. In fact, every node has eight outgoing branches of which there are four para pairs of parallel transitions. So, we must apply Ungerboek's rule for this as you can see there were three memory elements leading to eight states in the trellis, right and there were three input bits coming in leading to eight branches emanating from each node from 000 up to 111.

Here is assignment. So, here is the assignment of the symbols from the set partitioning tree that we just now saw. This is a tree and this is what we mean by A 0 and A 1 as the subsets this is A 00, A 10, A 01 and A 11. Similarly, these eight subsets A 000, A 100 up to A 111. So, we are going to assign the parallel transition from this lowest rug in the ladder,

So, these two must be assigned to the first parallel transition, these two symbols must be assigned to the second and then subsequently we can look at the different transitions. So, A 000 so, the two symbols from A 000 are assigned here A 100 are assigned to the next set of parallel transitions. Two symbols from A 010 to this one. And finally, two symbols from A 110 to this one alright, but we have also made sure that the diverging branches must be such that they get assigned from the next higher level.

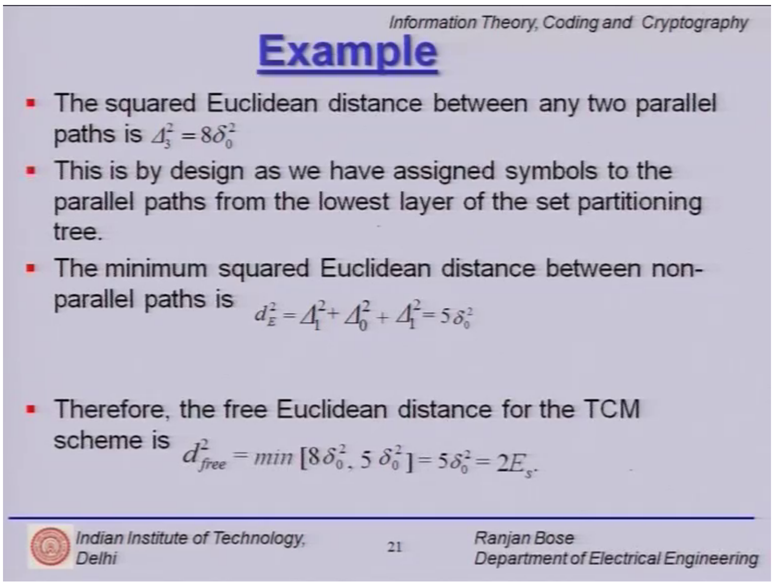
So, the diverging branch should be such that the symbols are assigned from these parts. So, that is the case because there more than one ways to assign these two symbols to the parallel branches. So, similarly the merging back branches merging back branches must be assigned to the next higher level. So, this is an example how Ungerboek's design rule is used to assign symbols to the trellis with parallel transitions.

(Refer Slide Time: 17:38)



So, if you calculate now the d free based on this then you can find out what is asymptotic coding gain.

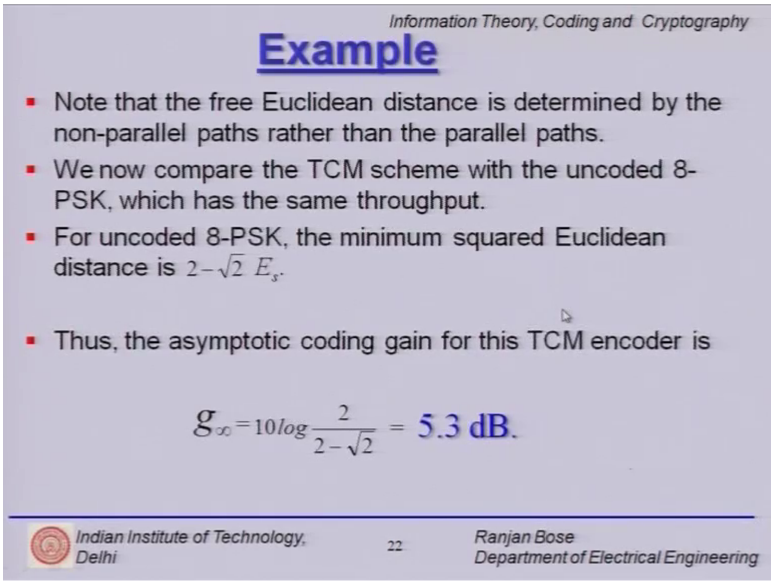
(Refer Slide Time: 17:51)



This if you do the calculations about the squared Euclidean distance then the minimum squared Euclidean distance between non parallel paths is 5 delta naught squared and you have if you look at the squared Euclidean distance between two parallel transitions is 8 delta naught squared. So, we have been able to ensure that it is not the parallel transitions that are leading to d square d free squared.

So, if you look at d free square it is the minimum of the diverging and the merging back branches. So, we calculated d free between this node and this node either through this branch or it diverges and then merges back. So, it is not the parallel transition, but the others which are causing it to have an error event and consequently we have the d free dictated by the non parallel path which is 5 delta naught squared equal to 2 under root 2 E s.

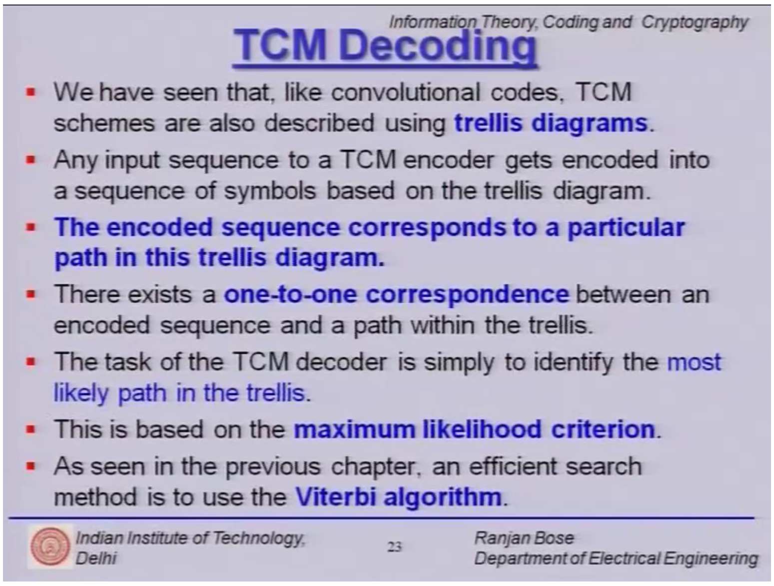
(Refer Slide Time: 18:56)



So, now that we have the minimum Euclidean distance corresponding to the d free for the TCM scheme we want to find out the asymptotic coding gain. So, if we had not used rate 3 by 4 encoder, if we had this 3 bits coming and we had to transmit them by modulating in them we would use 8-PSK. So, we find out the squared Euclidean distance from 8-PSK which is 2 minus under root 2 E of s and now, this asymptotic coding gain is simply the ratio 2 over 2 minus root 2 and this is log to the base 10, 10 time. So, it gives you a whopping 5.3 dB coding gain.

So, that trellis the complicated looking trellis also gave us a pretty good asymptotic coding gain. If you remember in electrical engineering even a coding gain of 3 dB is worth looking into. Now, here we have 5 point 3 dB asymptotic coding gain. So, this is really a very good design, it is a good example of a rate 3 by 4 convolutional encoder coupled with the natural mapper.

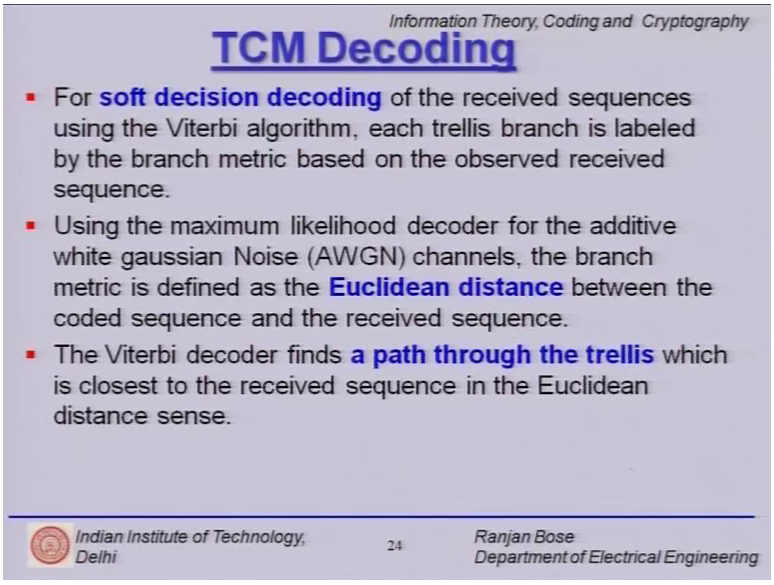
(Refer Slide Time: 20:12)



Now, let us quickly spend some time looking at the decoding part. So, we have already made this observation that there is a one to one correspondence between the bit stream coming in and a path in the trellis. Only in this case the trellis paths are labeled by symbols and not by bits. So, the decoding problem is finding the most likely path of the trellis with respect to that which is received and most likely comes in terms of the minimum Euclidean distance.

So, we use the maximum likelihood criteria to do it and Viterbi algorithm is commonly used for this decoding technique.

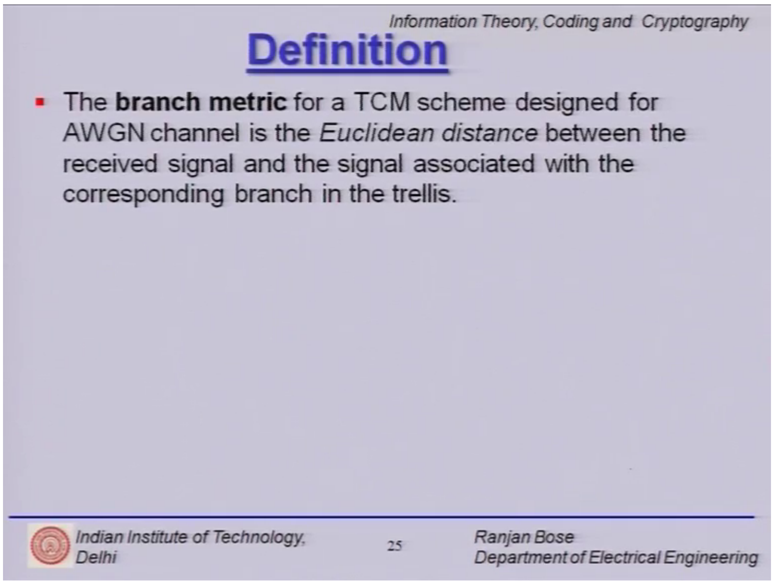
(Refer Slide Time: 20:54)



So, for soft decision decoding of the received sequence is using Viterbi algorithm each trellis branch is labeled by the branch metric based on the observed received sequence. The only difference is in convolution decoding we used hard we used the bits and therefore, the hamming distance here we will use the Euclidean distance.

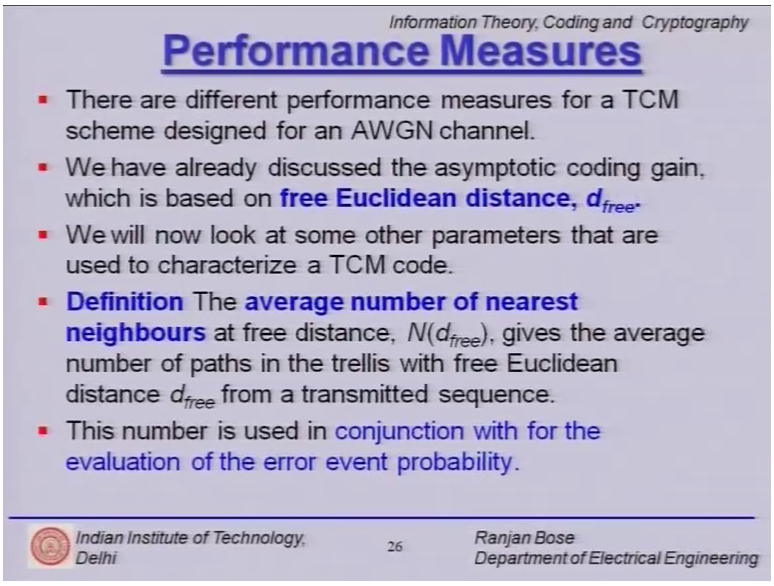
So, using the maximum likelihood decoder for the additive white Gaussian noise channels the branch metric is defined as the Euclidean distance, and the Viterbi decoder tries to find out a path in the trellis which is most closely resembling that is closest in terms of the Euclidean distance with respect to the received path.

(Refer Slide Time: 21:43)



So, a branch metric for a TCM scheme is now in terms of the Euclidean distance and we now try to find out the performance of TCM schemes in additive white Gaussian noise channel.

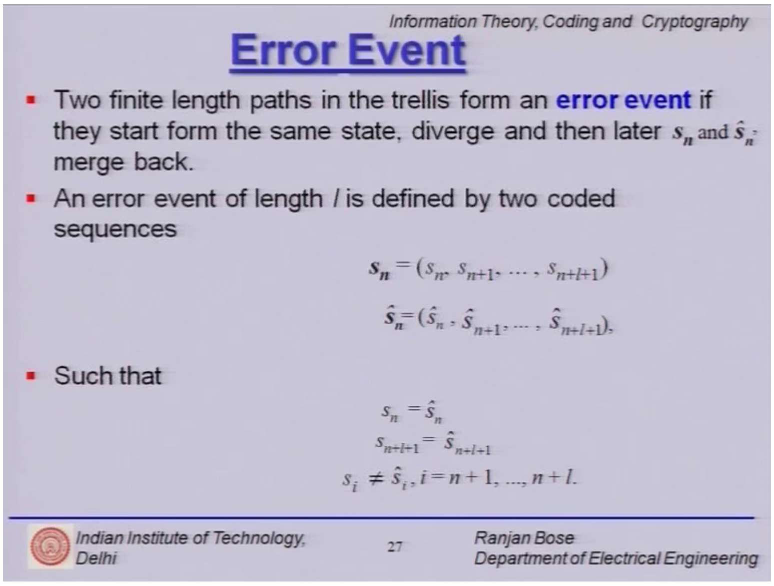
(Refer Slide Time: 21:56)



So, in the next few slides we will develop a mathematical tool. It will be slightly involved, but what we will do is try to get the message across in terms of what we are trying to do. So, we are now going to work with this free Euclidean distance.

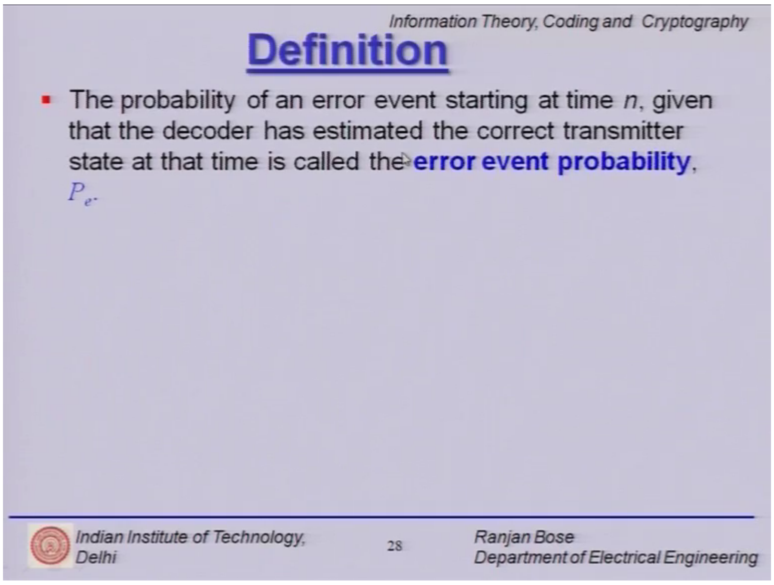
Now, what we would define is the average number of nearest neighbors N as a function of d free gives the average number of paths in the trellis with the free Euclidean distance d free. So, d free is actually the weakest link in the chain and we would like to find out how many weak links are there in the chain. So, this N as a function of d free is used in conjunction with d free to evaluate how good a trellis coded modulation scheme is.

(Refer Slide Time: 22:50)



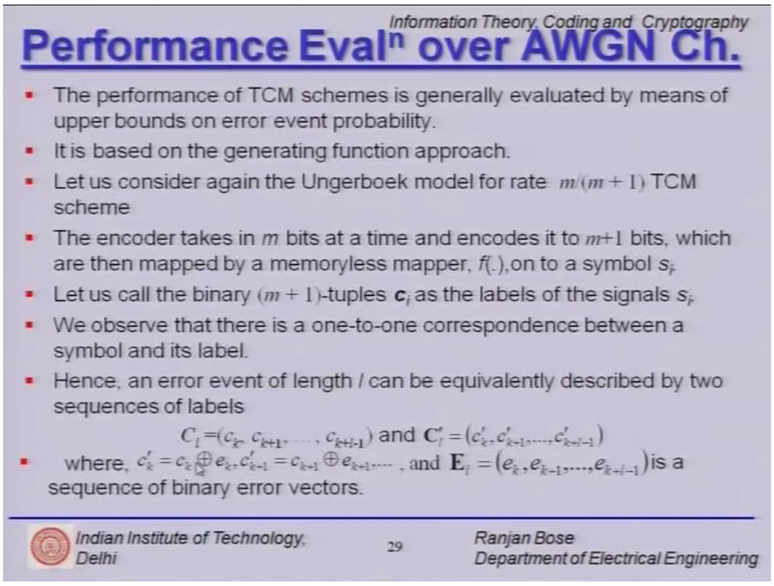
So, let us look at an error event definition. So, we have already defined that we have a sequence S n, this is a vector of S n. S n plus 1 and so on so forth, but we have l branches. So, S n plus l plus 1 and we have a received sequence or the estimate that we try to get in terms of the guessed sequence that we sent, right. And we would like to get S n equal to S hat n.

(Refer Slide Time: 23:28)



So, the probability of an error event starting at time n, given that the decoder has estimated the correct transmitter state. And the time is called the error event probability P e and we will try to get an upper bound on this P e.

(Refer Slide Time: 23:44)



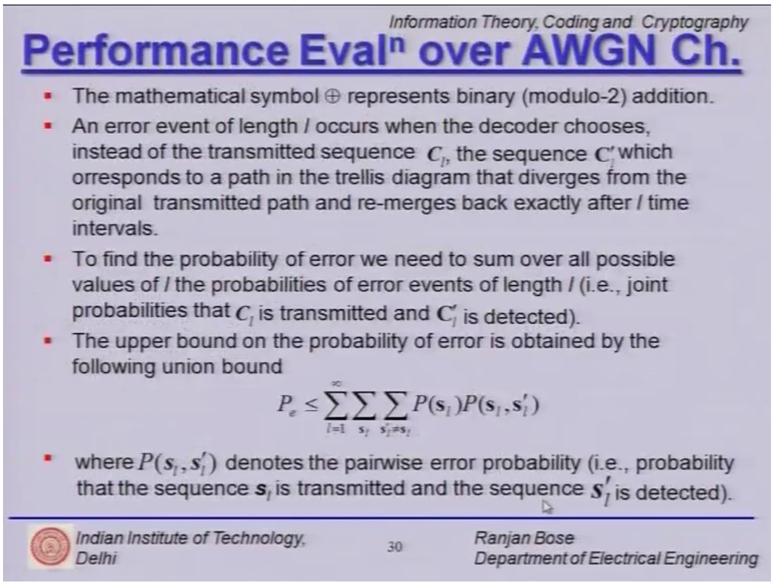
So, the performance of a TCM scheme is generally evaluated by means of the upper bound on the error event probability. We will soon realize why an upper bound is used simply because it is impossible to keep track of all the possible path that may result in an error event. So, we look at the whole group of possible paths and we come up with and some kind of a union bound.

Now, the performance evaluation is based on the generating function approach that we have already studied in our earlier portion of trellis coded modulation and the convolutional encoder. So, here please note we are going to look at m over m plus one TCM scheme in general and this will be coupled with a mapper. So, encoder takes m bits and converts into m plus 1 bits and what we have is we have a binary m plus 1 tuple c i as the label for signal s i. So, we have these labels that we are going to work within the next few slides and the idea is to recover these labels and there is of course, a one to one correspondence with this c i to s i. We observe s i and then we try to recover the c i.

So, an error event of length l can be equivalently described by two sequences of labels C l and C l prime, ok. So, we are we are going to work with this and error event is when they are not the same, ok. So, how do we describe it? Well, if they are not the same then an error has happened and we have this c k binary addition e k where this is the error event. Again, this is a binary label c prime k plus 1 this is nothing but the original one that was transmitted plus an error binary event.

So, just like that we have a sequence of c k, c k plus 1 and so and so forth we have a sequence of this error binary error vectors, ok.

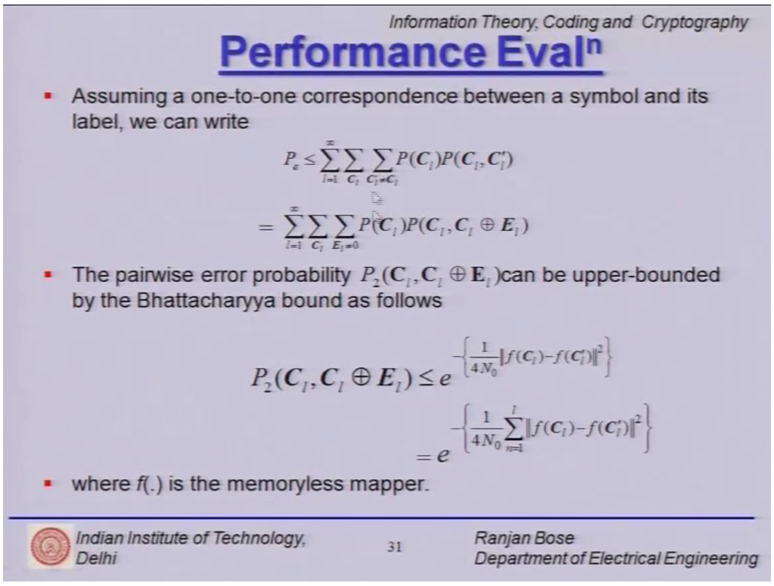
(Refer Slide Time: 26:07)



So, basically in the performance evaluation over additive white Gaussian noise channel, we look at the upper bound on the probability of error simply by looking at this union bound, the probability of s l and the pair wise error probability P s l comma s l prime where s l prime is not equal to s l. So, we look at all such cases therefore, it is summation over all the cases where s l prime is not equal to s l.

And, then we have all the possible symbols, right. So, we have a summation over all the possible symbols and then we have all possible path lengths of the error event. So, error event can be of one length. So, in one hop it diverges and merges back or two hops or three hops up to infinity, because the trellis is a semi infinite geometric structure. So, we have these three summations here. And therefore, we have a union bound on all the cases, where s l is not the same as s l prime.

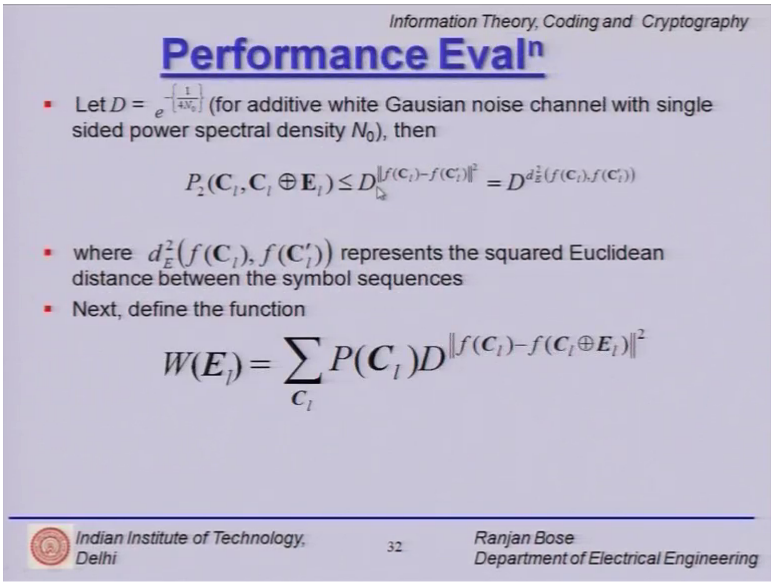
(Refer Slide Time: 27:21)



So, now we can do a quick set of algebraic steps and, we look at this upper bound. Now, since there is a one to one correspondence between a symbol and it is label we have replaced the symbols by this labels. So, we have now the sequence of labels and the probability that C l is not the same as C l prime. So, we do over all those possible cases and we have all the possible symbols and we have the C l labels and again all the possible length of the path, but we have just now put together that this C l prime is nothing but the original C l plus the error. So, it is simply written as follows.

So, now we can use the Bhattacharyya bound to limit upper bound this error event. What is this pairwise error event? Probability between C l being transmitted and C l plus some error being received is now less than this is a function of C l minus C l prime. It is the Euclidean distance square, where f is a memoryless mapper. So, this is how we are using the Bhattacharyya bound to get into this Euclidean distance concept.

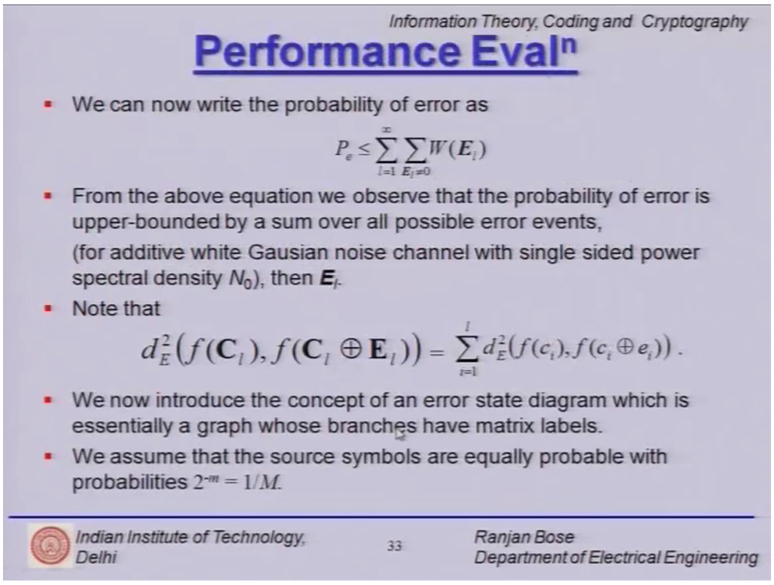
(Refer Slide Time: 27:28)



So, now define capital D as e raised to the power minus 1 over 4 N naught. So, this will help us write this in a compact form. So, now, this pairwise error probability is upper bounded by this D raised to power this norm of f C l minus f C l prime, and this is nothing but the squared Euclidean distance between these two.

So, we now define this n function W of E l. So, please note it is a function only of E l and this is great, because we do not want this W to be a function of any particular path in the trellis it is only a function of the error event. Therefore, we are defining it as this. This is simply defined as follows and now we will plug in this W into the term for pairwise error probability.

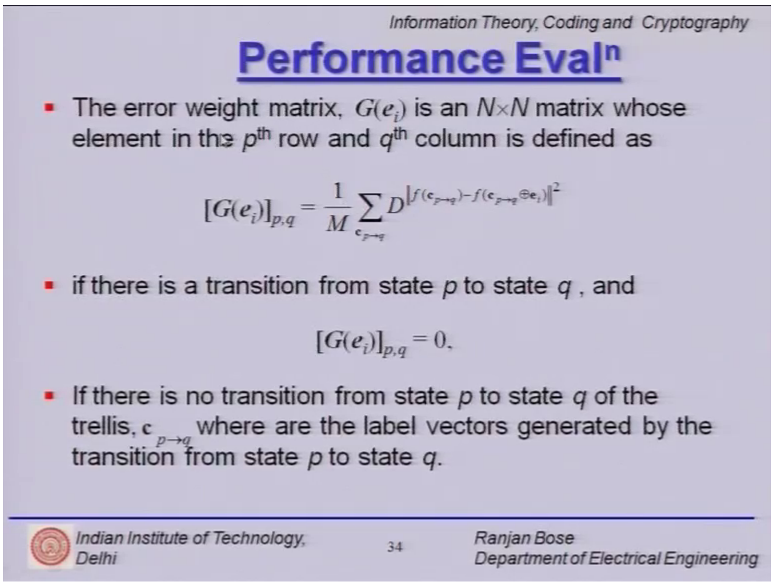
(Refer Slide Time: 29:46)



So, now pairwise error probability is now simply for all possible path lengths from 1 to infinity and error not being equal to 0. Earlier it was at C l is not equal to C l prime and for all this cases where error is not equal to 0 it looks at all the possible W's here, fine.

So, we now look at finally, how to get a handle on this upper bound on the right hand side.

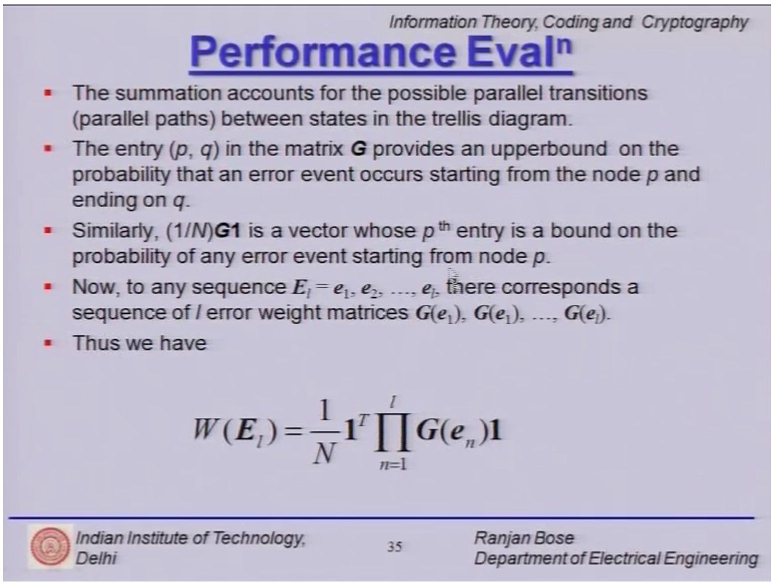
(Refer Slide Time: 30:22)



So, we now have this error weight matrix G e i is an N cross N matrix whose element in the p-th row and q-th column is defined as follows, ok. So, we define this error weight matrix and see how clearly we are defining it we have this D which we have defined earlier, this is the mapper of f c p to q, right and minus c p to q plus this error e i. So, if you do this is talking about from transition from state p to q.

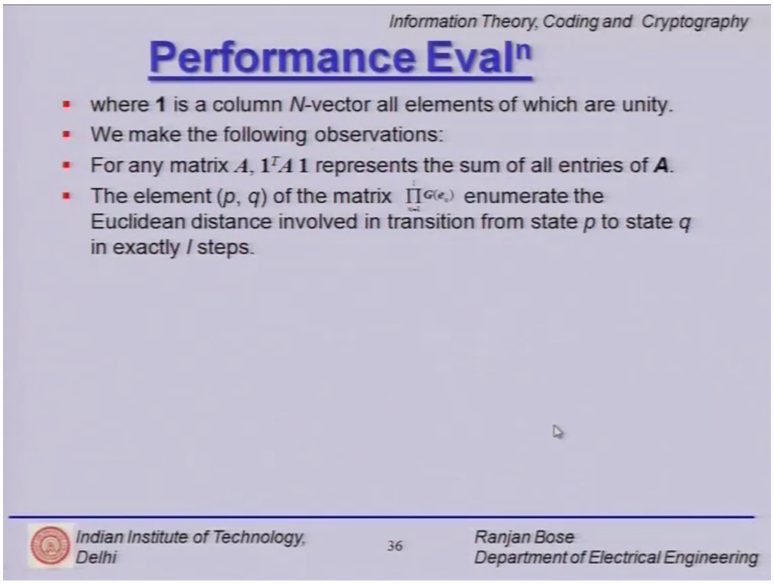
So, we are now looking at all the possible transitions. So, we are looking at all the possible transitions and how the error is incorporated when we go from one transition to other, right.

(Refer Slide Time: 31:25)



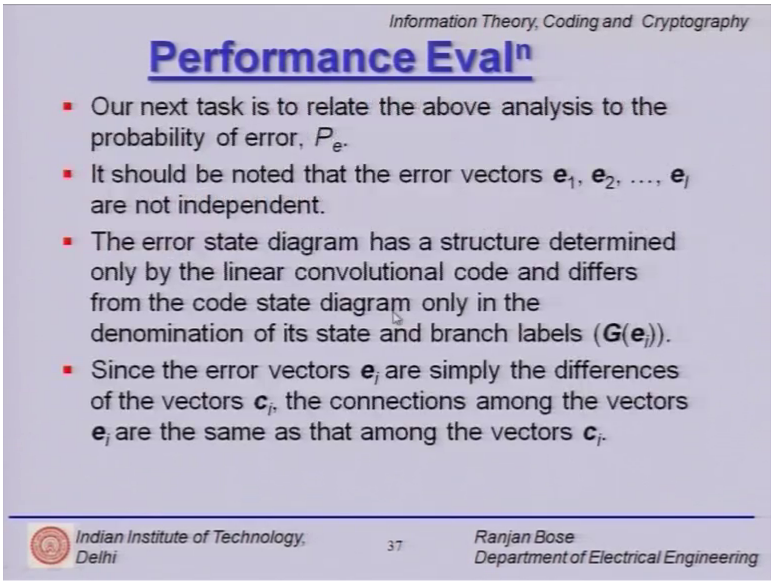
So, if you do this math completely you can see that this W E l which is only a function of the error can be written as 1 over N then 1 this is a vector transpose product of G e n 1. So, this is what we can do some basic mathematics to come to this one.

(Refer Slide Time: 31:49)



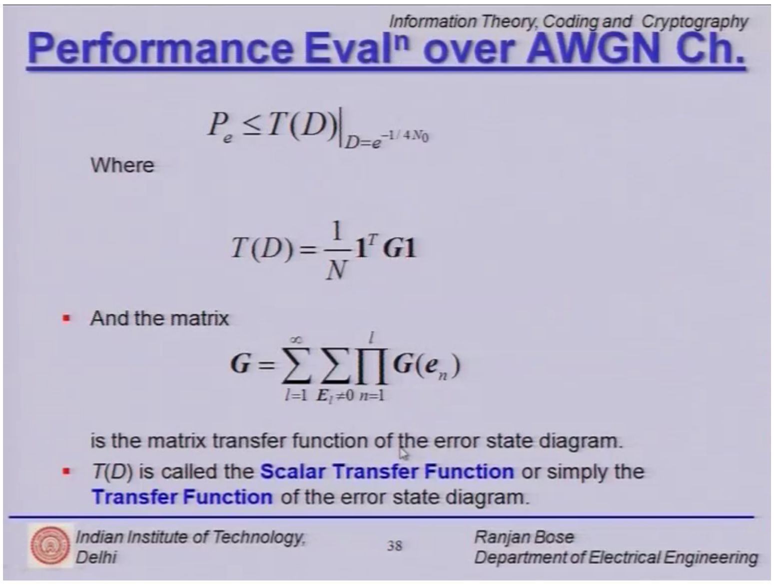
One I have said is a column vector of N length vector.

(Refer Slide Time: 31:57)



And so, we would like to finally, get to this P e, the probability of error.

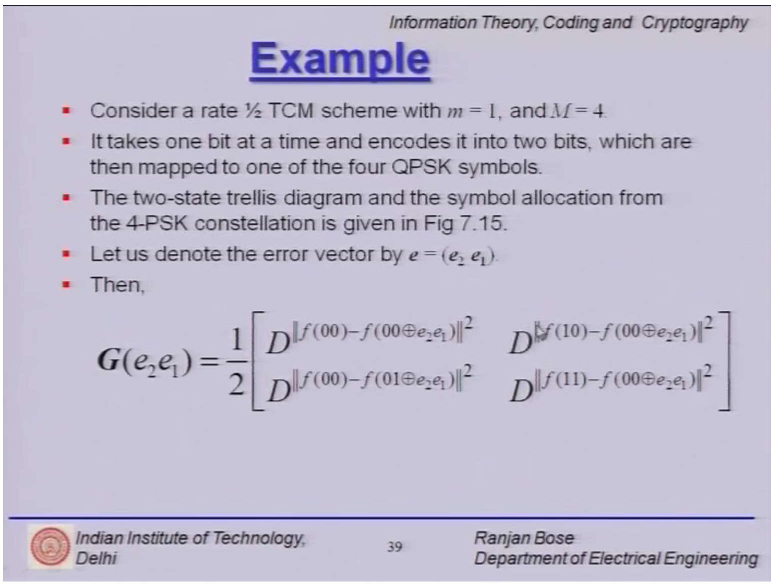
(Refer Slide Time: 32:08)



And, if you do this you can write out in terms of that modified state diagram T of D, where D is as earlier e raise to the power minus 1 over 4 N naught.

So, here T D is very easily written as 1 over N 1 vector transpose G 1 vector and this in some sense you have seen earlier in the modified state diagram, and G matrix is defined as follows. T D is called a scalar transfer function or simply the transfer function of the error state diagram. So, once we learn how to calculate, and we will try to see an example, then it is very easy to get an upper bound on that.

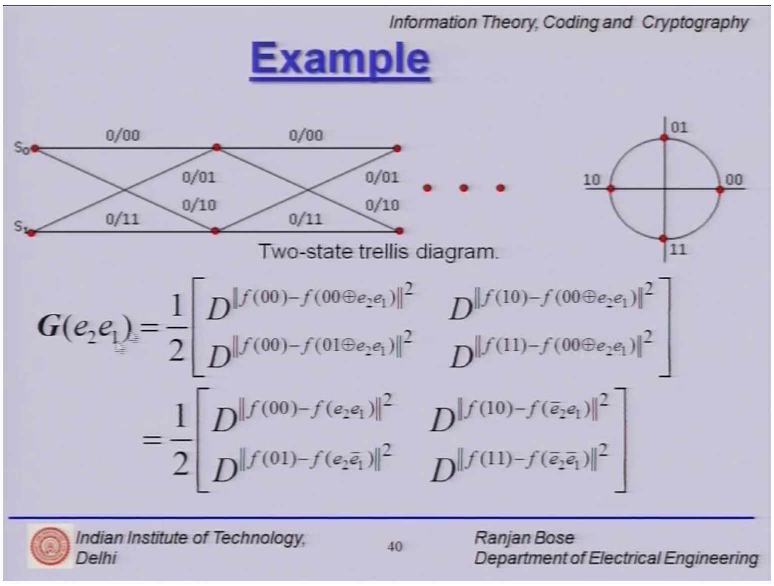
(Refer Slide Time: 32:59)



So, let us quickly go through an example to see how this works. We have a rate 1 by 2. So, m is equal to one m over M plus 1, TCM scheme and M is equal to 4 capital M. So, it takes one bit converted into two bits and two bits required QPSK to be used.

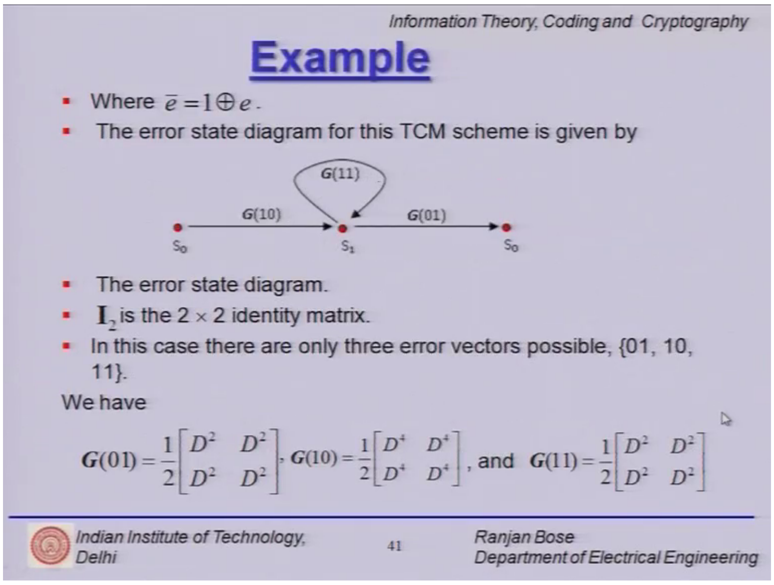
Now, the two-state trellis diagram is we will just show it and the error vector e will be as follows. So, this is an example how G e 2 e 1 can be written, ok. So, we have this 00, 10, 01 and 11.

(Refer Slide Time: 33:45)



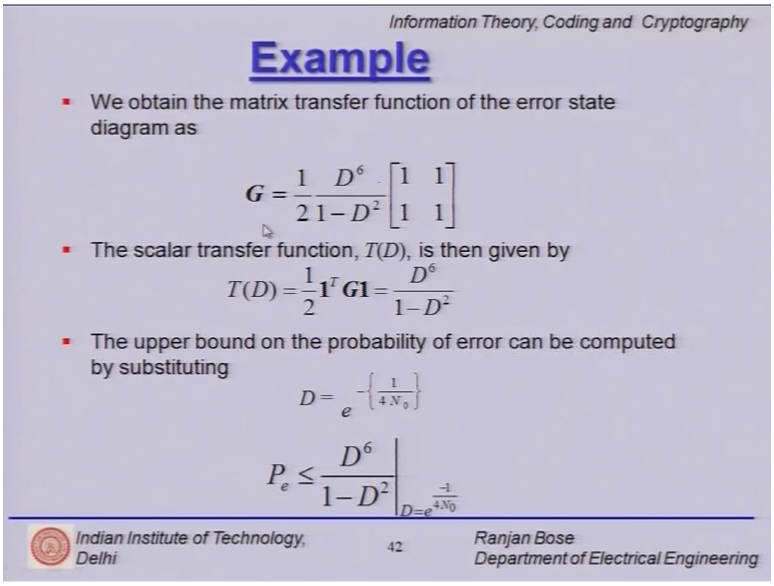
And, here are the two-state trellis it is a simple example, where we have this QPSK and the four symbols are being used and you can easily write G e 2 e 1 as follows.

(Refer Slide Time: 34:05)



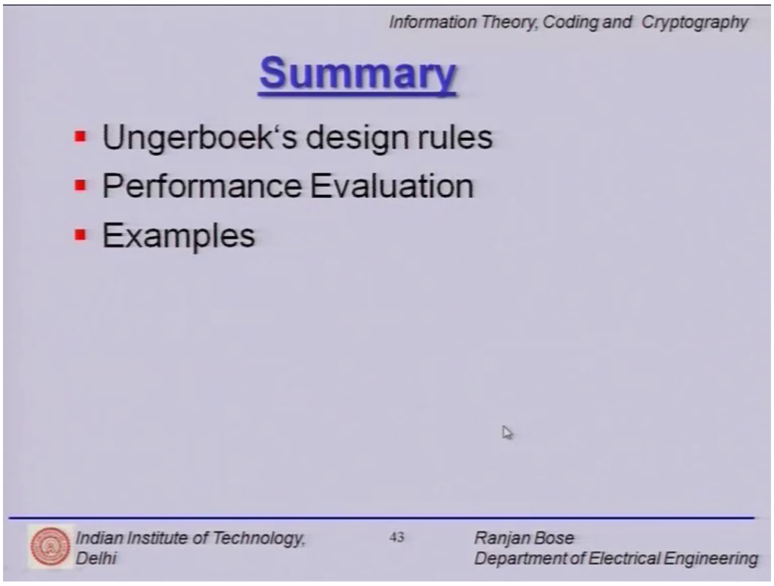
So, finally, if you follow the steps we have these three matrices G 01, G 10, G 11 in terms of this D.

(Refer Slide Time: 34:25)



And, very quickly we can calculate this scalar transfer function T D and it comes out as D square D raised to the power 6 over 1 minus D squared we could have solved it using a traditional method also using dummy variables in the middle we have learnt how to solve this. So, T D comes out to be this. Once we have the T D in any way you would like to calculate then you substitute this D equal to e raised to the power minus 1 over 4 N naught and you have the upper bound on the probability of error.

(Refer Slide Time: 34:58)



So, this gives a basic idea. So now, we would like to kind of summarize what we have done today. We have looked revisited actually Ungerboek's design rules, where we would like to understand how the parallel transitions and diverging and merging back paths are assigned. Then, we looked at how to evaluate TCM schemes over additive white Gaussian noise channel. We made the observation that d free is the single most important parameter for TCM schemes that we will use and it comes out that the probability of error upper bound is strictly dependent on this d free notion. We also looked at certain examples to see how we can calculate this probability of error.

With that we come to this end of this lecture.

Thank you.