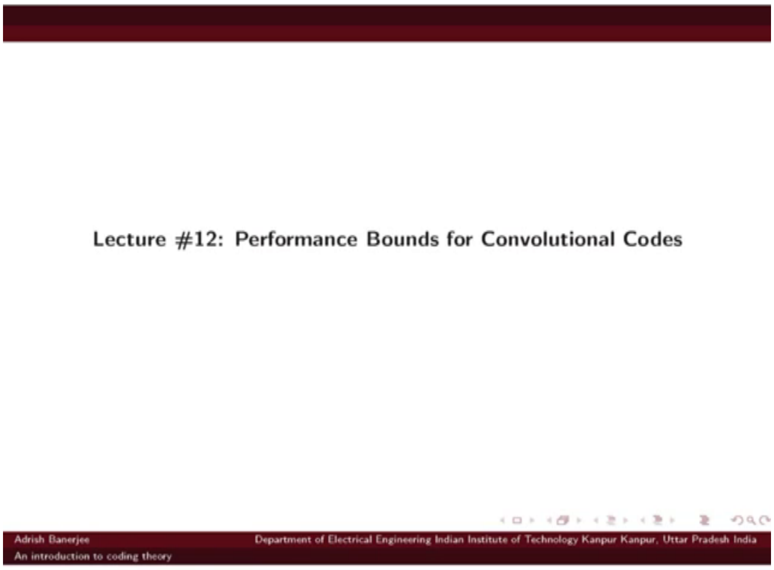
An Introduction to Coding Theory
Professor Adrish Banerji
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
Module 05
Lecture Number 23
Performamce Bounds for Convolutional Codes

(Refer Slide Time 00:13)

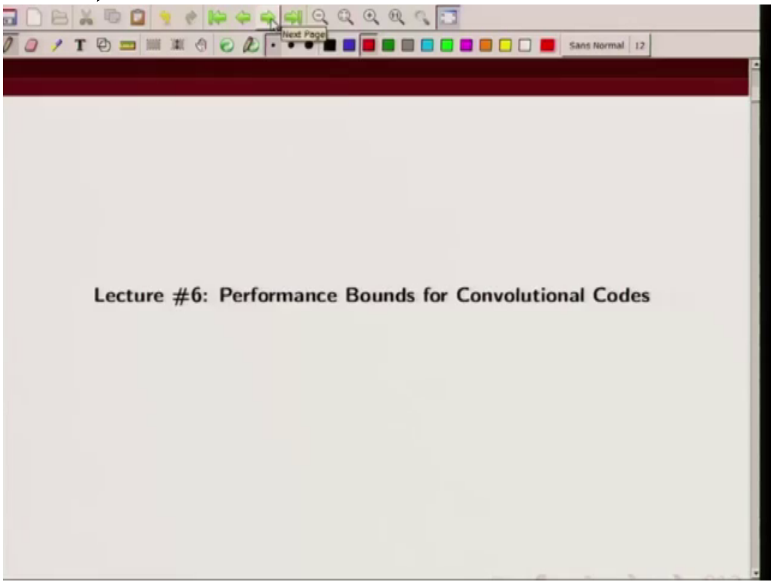


So in this lecture we are going to talk about

(Refer Slide Time 00:16)

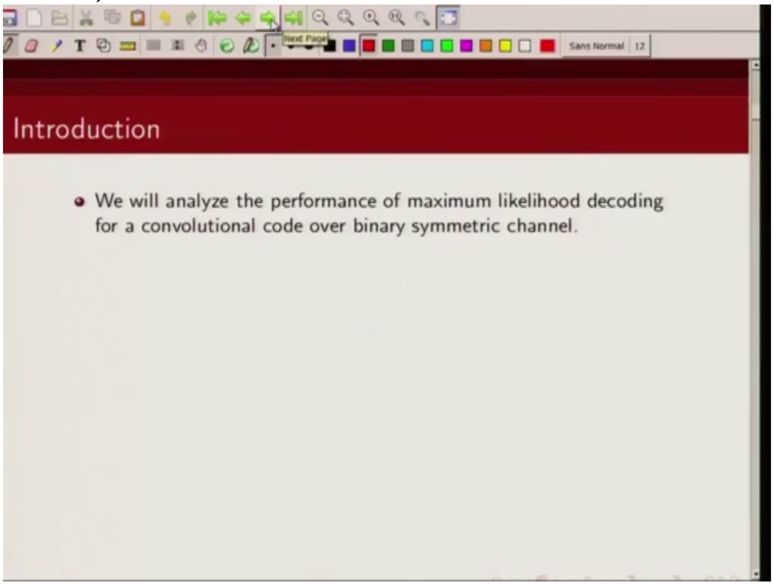
performance bounds for convolutional codes.

(Refer Slide Time 00:22)



In particular

(Refer Slide Time 00:23)

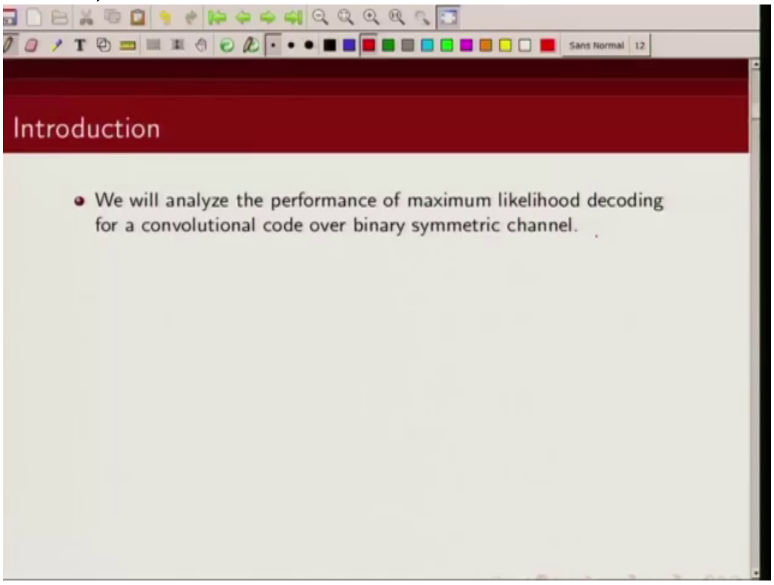


we are going to analyze the performance of

(Refer Slide Time 00:26)

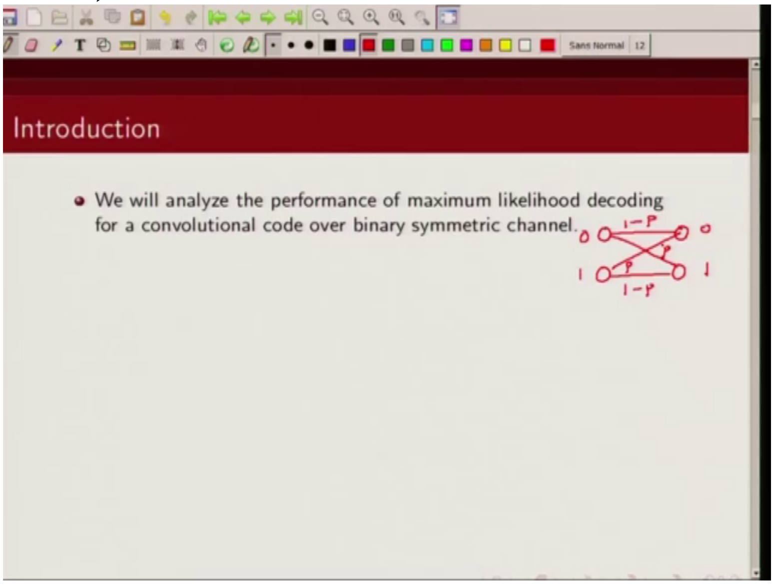
Viterbi Decoding which is maximum likelihood decoding for convolutional code and as an example we will consider a simple binary symmetric channel. Now if you recall what is a binary symmetric channel,

(Refer Slide Time 00:39)



so there are 2 inputs, 0 and 1, 0 and 1 and with some probability 1 minus p we receive the bits correctly and with a crossover probability of p, the bits get flipped. That's our binary symmetric channel.

(Refer Slide Time 01:01)

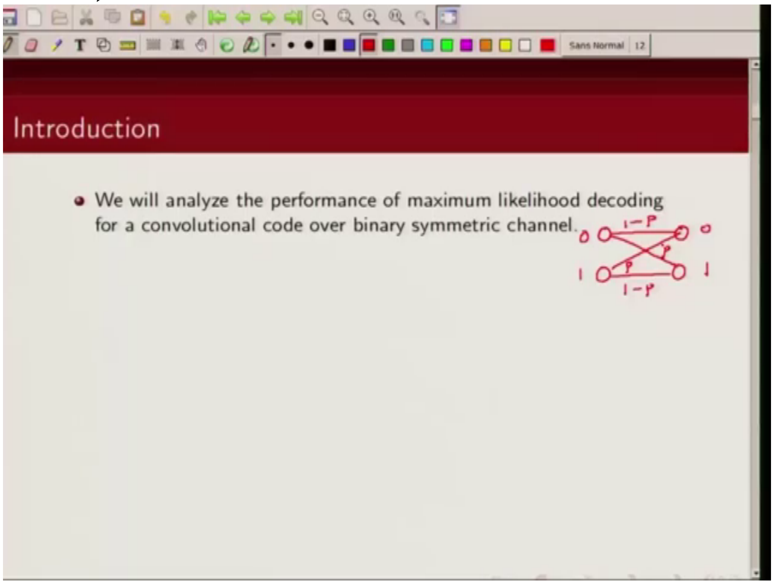


So we are going to analyze

(Refer Slide Time 01:03)

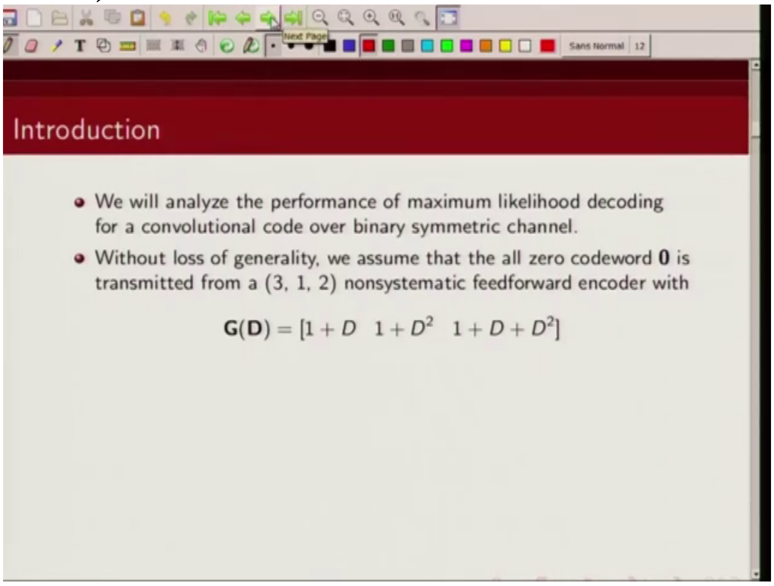
the probability of error, probability of error when we are doing Viterbi Decoding

(Refer Slide Time 01:10)



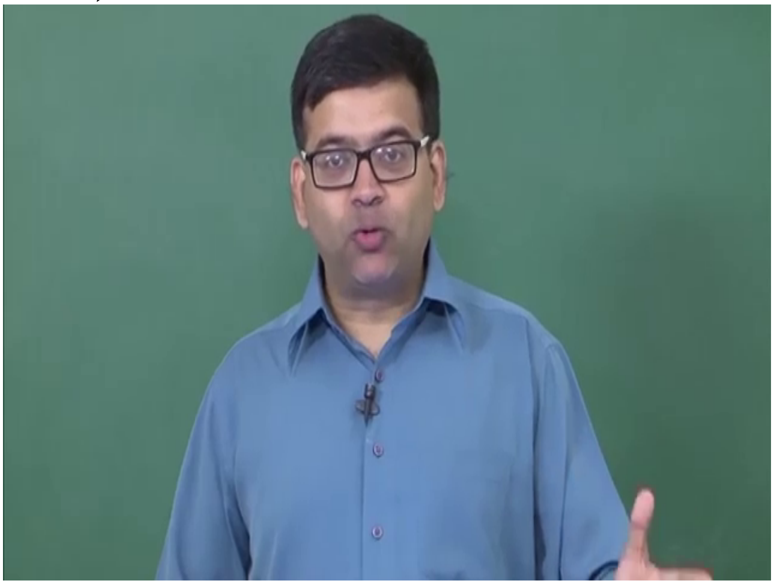
of convolutional code over a binary symmetric channel.

(Refer Slide Time 01:17)



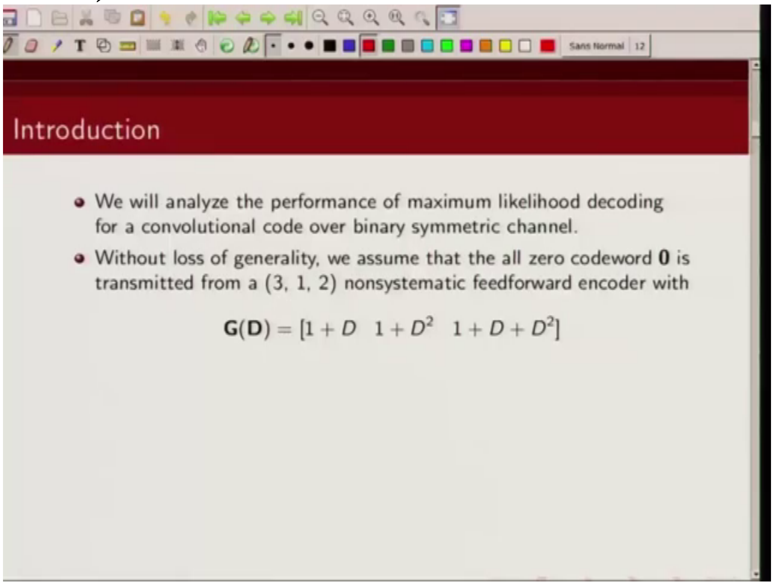
So as an example we will consider one particular

(Refer Slide Time 01:20)



convolutional code. In this case, we are considering a

(Refer Slide Time 01:25)

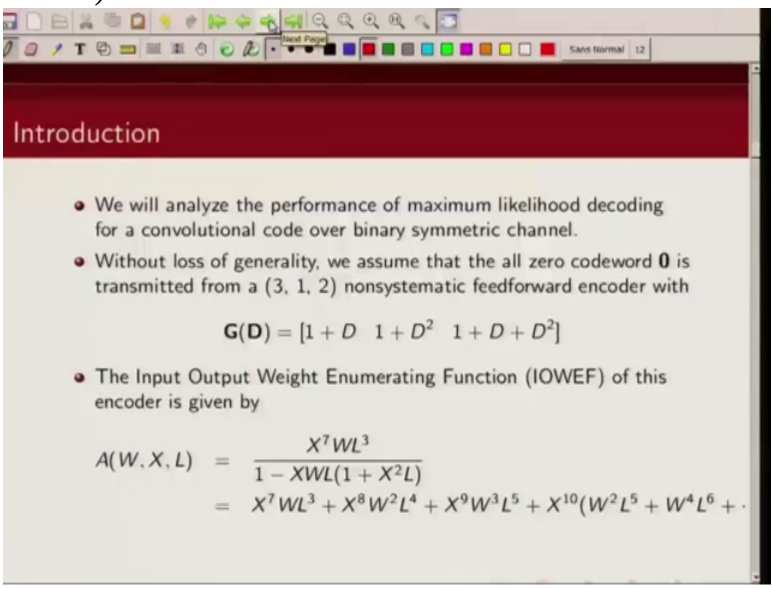


rate 1 by 3 convolutional code which has memory 2, so it is a four state convolutional code whose generator matrix is basically given by this. So these are my generators, 1 plus D, 1 plus D square, 1 plus D plus D square and without loss of generality we will assume that all zero codes was transmitted. So these are codewords transmitted over a binary symmetrical channel. And

(Refer Slide Time 01:58)

we received the bit, we applied maximum likelihood decoding Viterbi Decoding and now we are interested to find, characterize the performance of the Viterbi Decoding algorithm.

(Refer Slide Time 02:11)

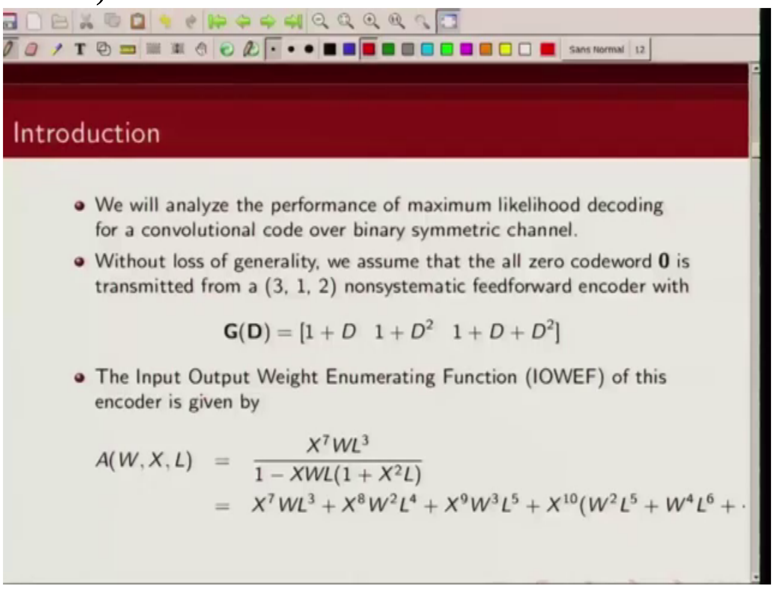


Now we would require one more concept which we discussed in lecture 2 c which is input output weight enumerating function. So what does input output weight enumerating function tells us? It tells us about

(Refer Slide Time 02:27)

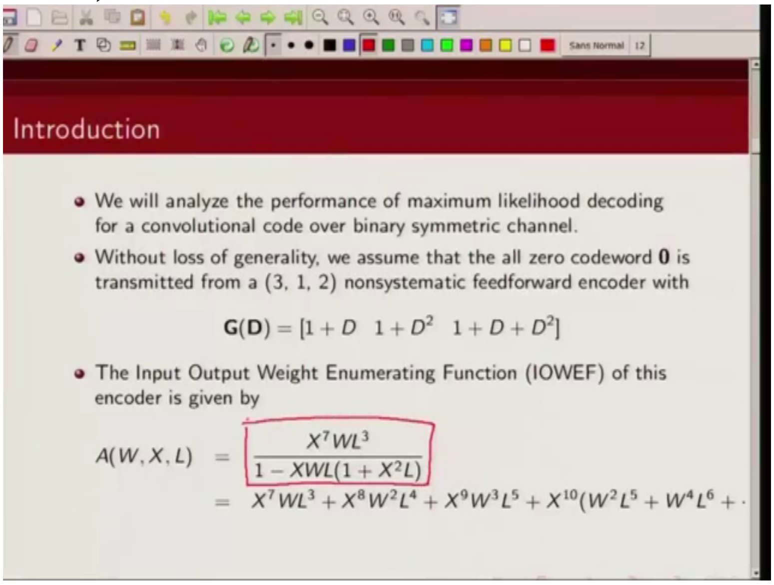
what is the input that will cause or what is the corresponding output that it will cause and what's the length of that particular sequence. So as you can see, all zero paths to the Trellis are all valid codewords. So this weight enumerating function essentially enumerates all non zero codewords.

(Refer Slide Time 02:52)



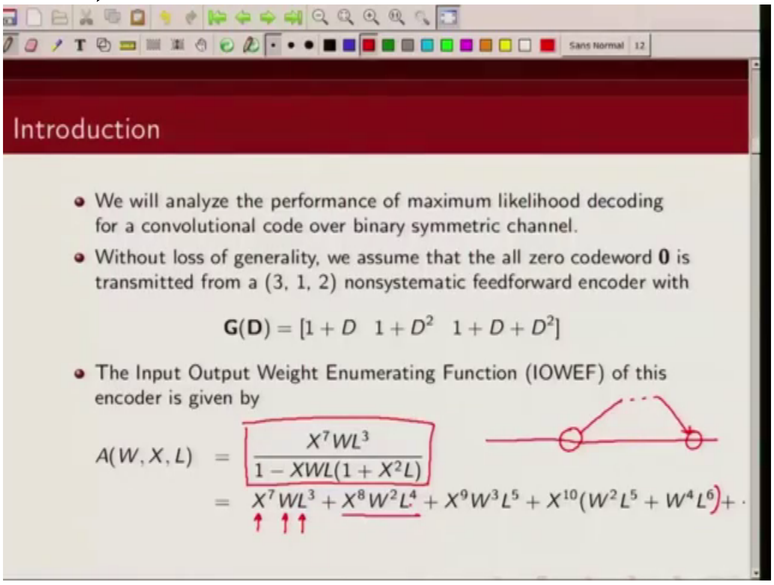
So for this particular example the input output weight enumerating function is given by this expression

(Refer Slide Time 03:04)



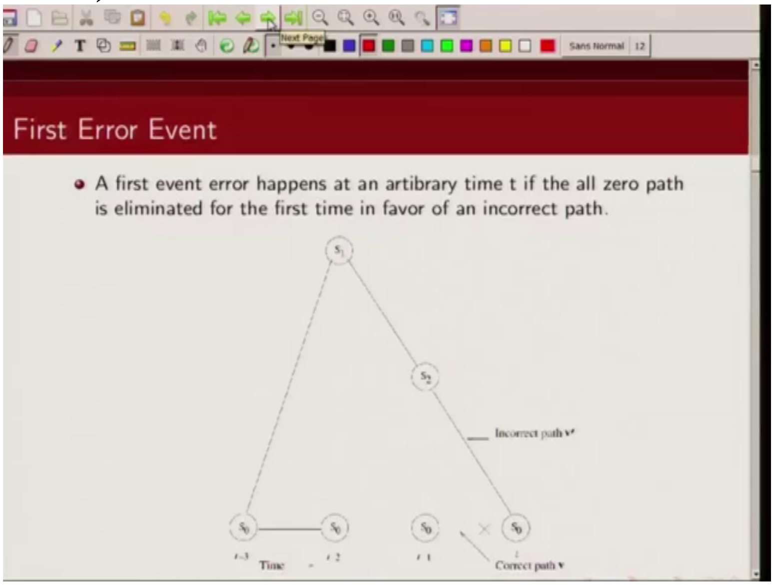
and which we can expand, we can divide this by this. So this we will get our terms. Now here x is my, the exponent of x will give me the overall weight, the exponent of w will give me the information sequence weight and this length will give me the length of this path like of this so the time it diverges to all zero state to the time it remerges to all zero state. So x 7 w l 3 means there exists a codeword of distance 7 which has information weight 1 and the time it diverges from all zero state to the time it merges back to all zero state, that is 3. And there is one such codeword. This can be interpreted as there is a codeword of weight 8 and corresponding input weight is 2 and it has length 4. Length 4 meaning, again this l denotes, so you have your codeword which diverging from all zero state and then after staying in some all zero state it is merging back into all zero state. So this l denotes the time, the time from which

(Refer Slide Time 04:43)



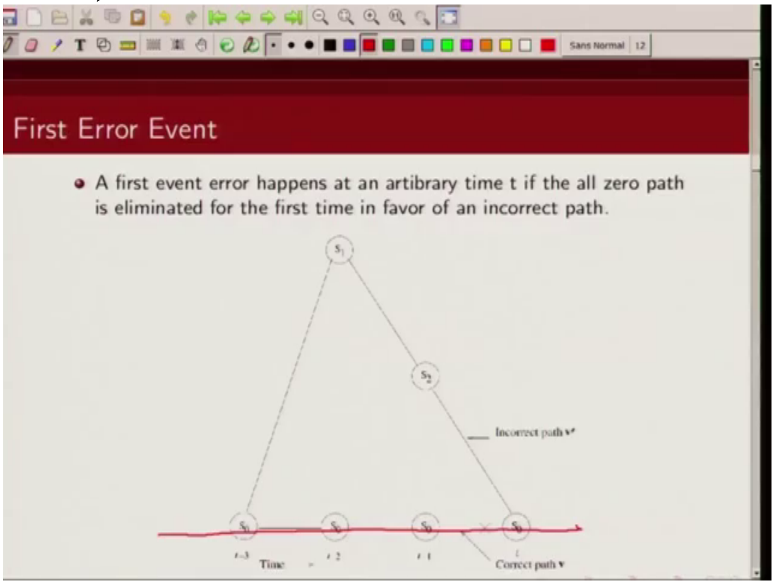
it moves away from the all zero state to the time it takes to come back to all zero state.

(Refer Slide Time 04:50)



So let us talk about what do we mean by error here. So since we consider an all zero sequence we expect that our correct path should be the one which goes through all zero state

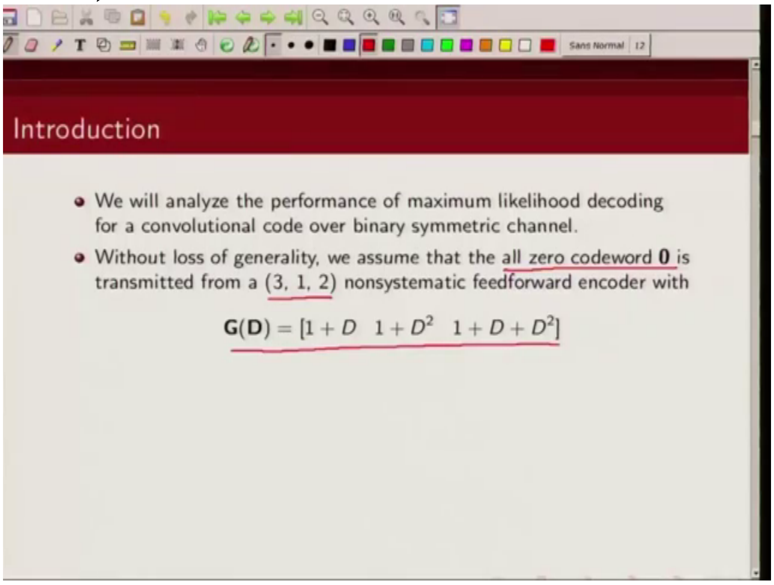
(Refer Slide Time 05:07)



(Refer Slide Time 05:08)

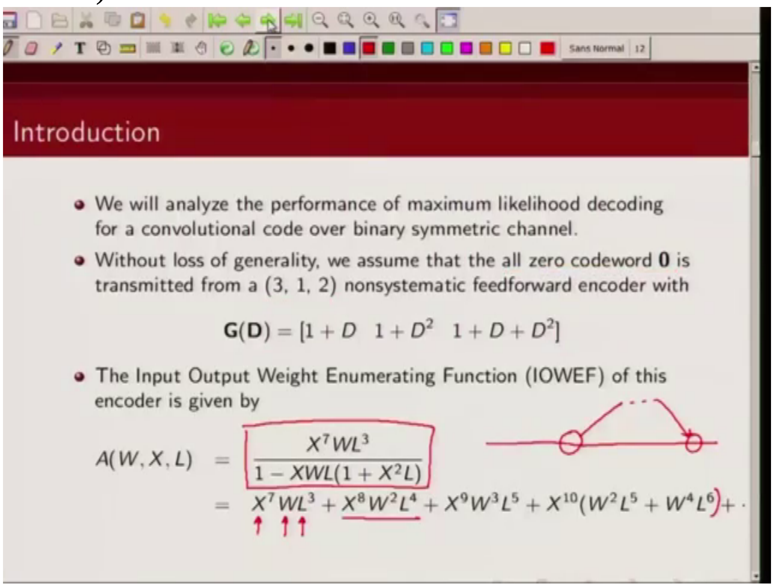
because our transmitted code sequence that we assume, you recall,

(Refer Slide Time 05:12)



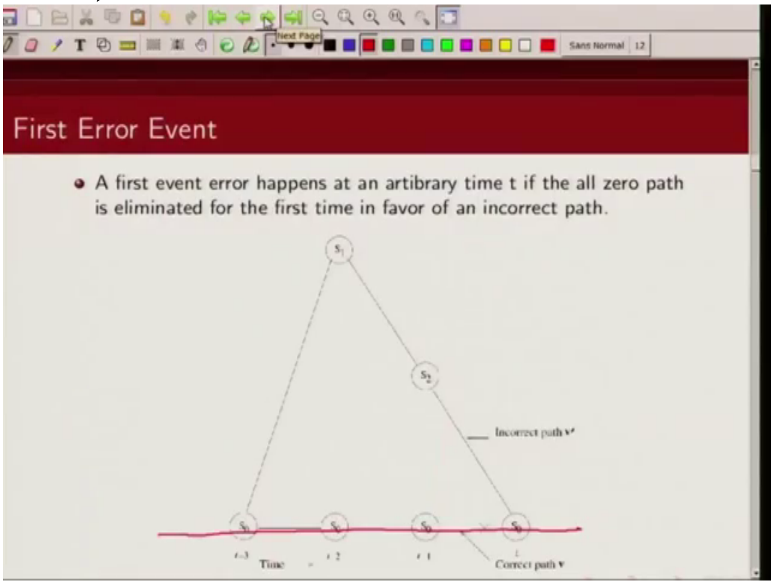
we assume that an all zero codeword was transmitted. So what is the correct

(Refer Slide Time 05:19)



path?

(Refer Slide Time 05:21)

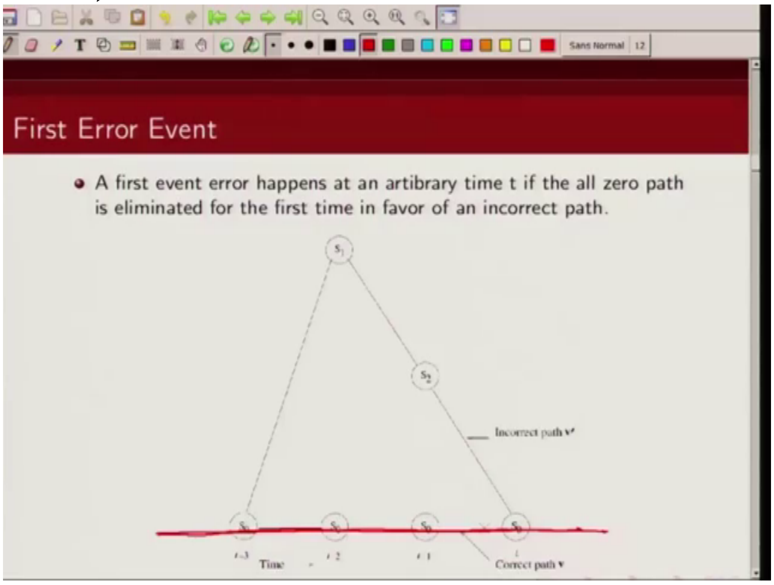


Correct path is one that goes through the all zero states. So this is the correct path. Now when will error happen?

(Refer Slide Time 05:30)

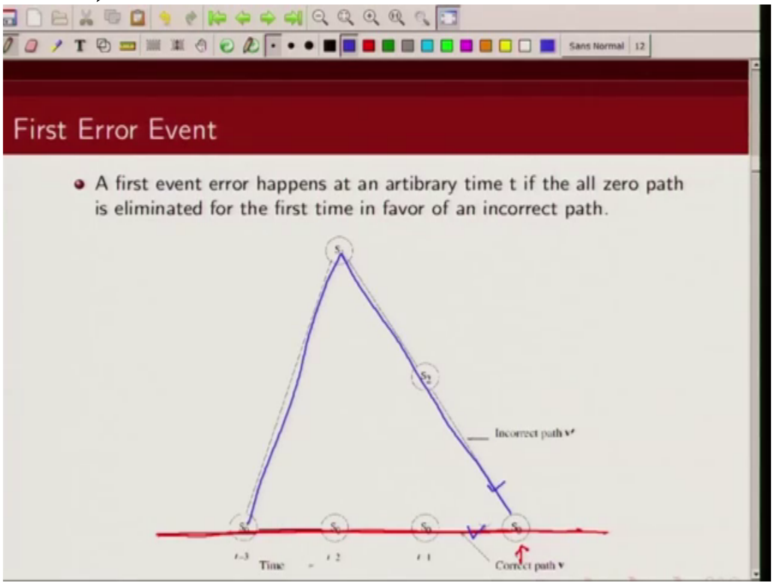
An error will happen if

(Refer Slide Time 05:33)



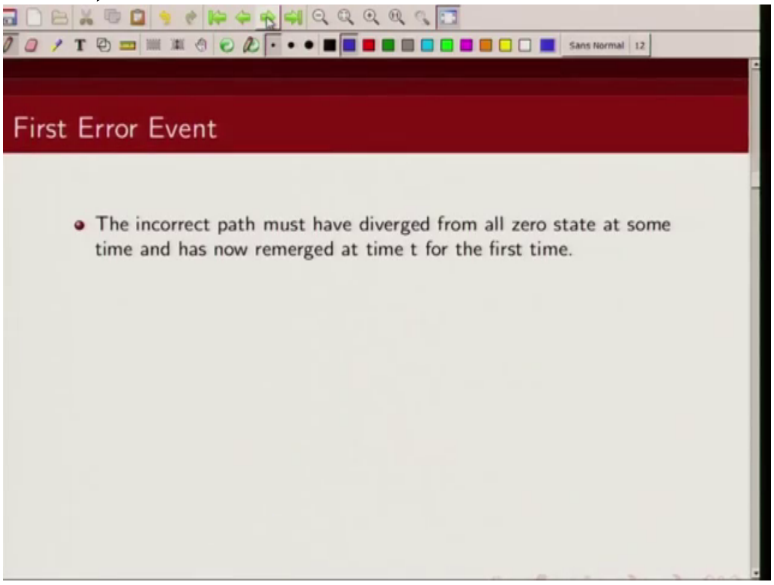
at a particular time instance let's just say this is the time instance I am looking at. If at this time instance, instead of deciding for this path, instead of deciding for this path, if I go for this path then error will happen. Why? Why is this an incorrect path? Why is this an incorrect path? This is an

(Refer Slide Time 05:55)



incorrect path because my transmitted codeword was all zero sequence. So if I decide in favor of this path instead of this path then I make an error. So I say a first event error happens at some arbitrary time t if the all zero path is eliminated in favor of a non zero path which is our incorrect path. So the first instance when I do that, so the instance when I do that, that's basically my event has error because I should have chosen all zero path. But at this instance what happened was the metric corresponding to this path was better so I chose this and I eliminated this path. But this is a wrong path because my transmitted codeword was all zero sequence.

(Refer Slide Time 06:55)

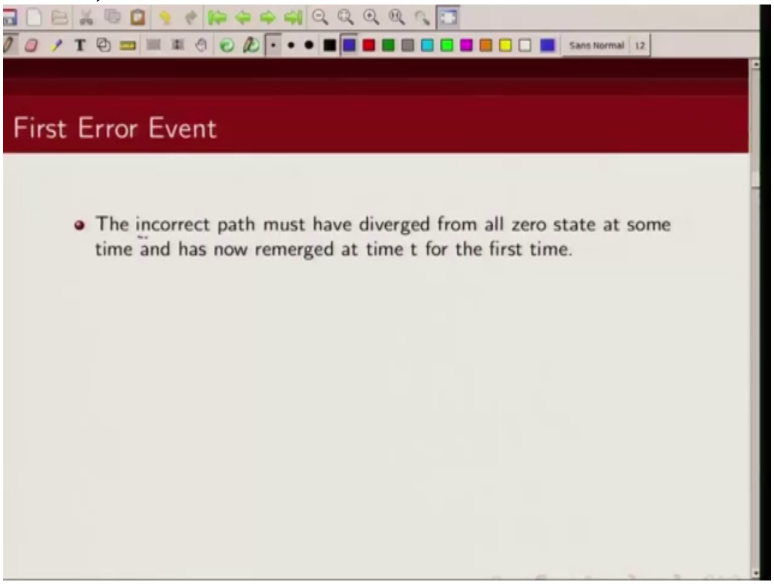


Now what does,

(Refer Slide Time 06:58)

we know that all paths to Trellis are essentially valid codeword. Now what does

(Refer Slide Time 07:06)

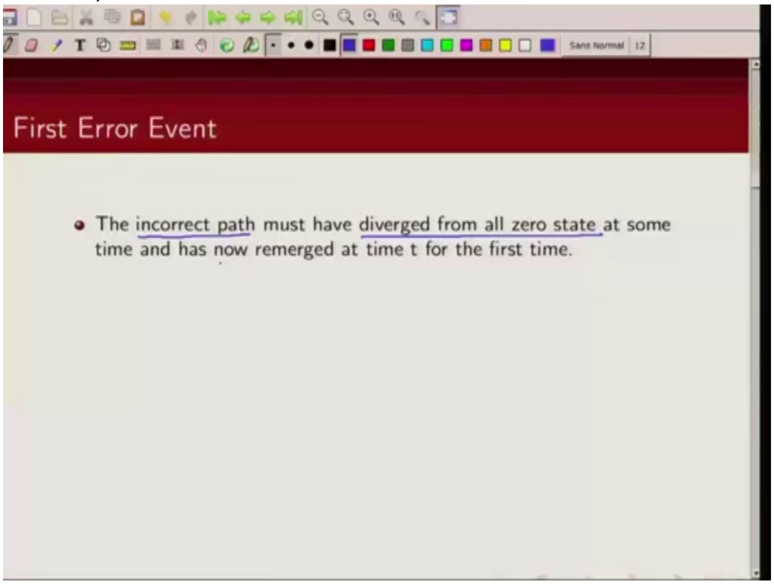


this incorrect path does? It diverges from all zero state at some other time before this time t

(Refer Slide Time 07:13)

and has merged back

(Refer Slide Time 07:18)

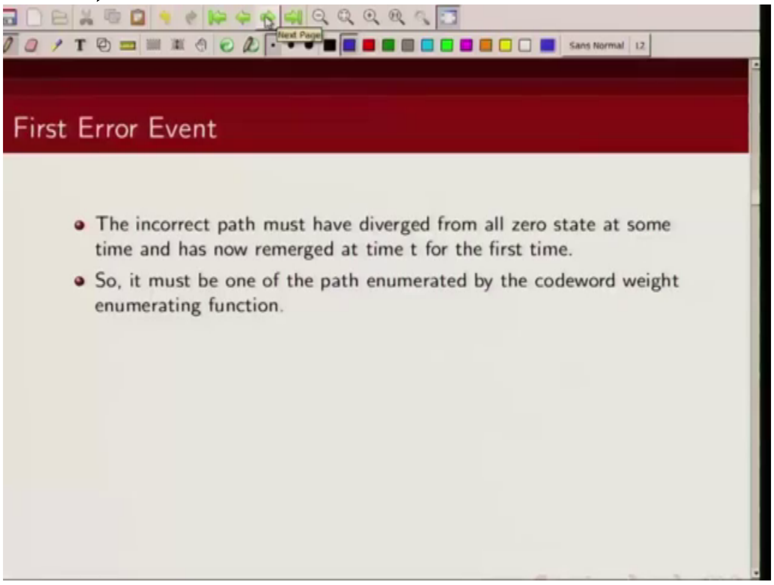


into all zero state at time t for the first time. So that's what we call first error event. So at a time t when we are deciding in favor of some other path other than the all zero path so that means this is the time

(Refer Slide Time 07:37)

when this other path, incorrect path has merged into all zero state. So at some time in the past it would have diverged from all zero state and now at time t it is merging back into

(Refer Slide Time 07:51)

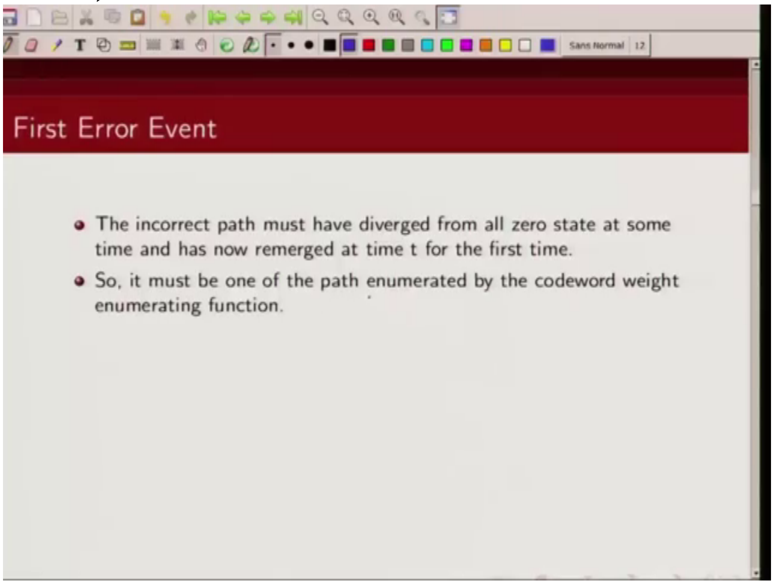


all zero state. Now can we find out what is the weight corresponding to this incorrect path? Yes we can because

(Refer Slide Time 08:03)

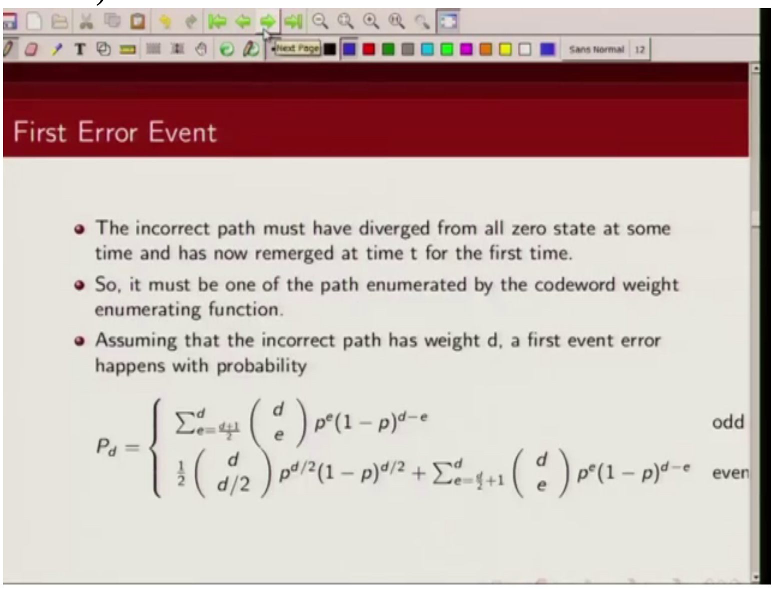
when we write the weight enumerating function of the convolutional code it enumerates weight distribution of all valid codewords. Now any path through this Trellis diagram is a valid codeword. So we can easily

(Refer Slide Time 08:22)



enumerate what is a code weight corresponding to any path through this Trellis.

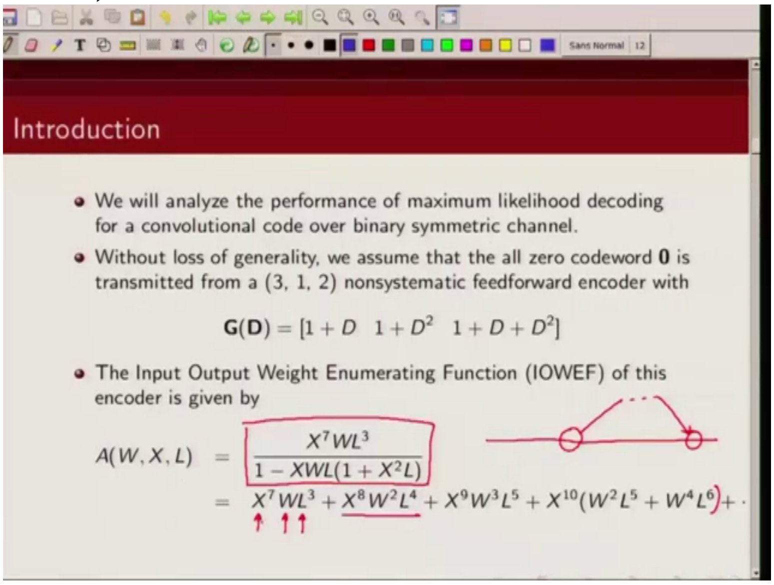
(Refer Slide Time 08:30)



Now when will we decide in favor of this incorrect path and not the all zero path?

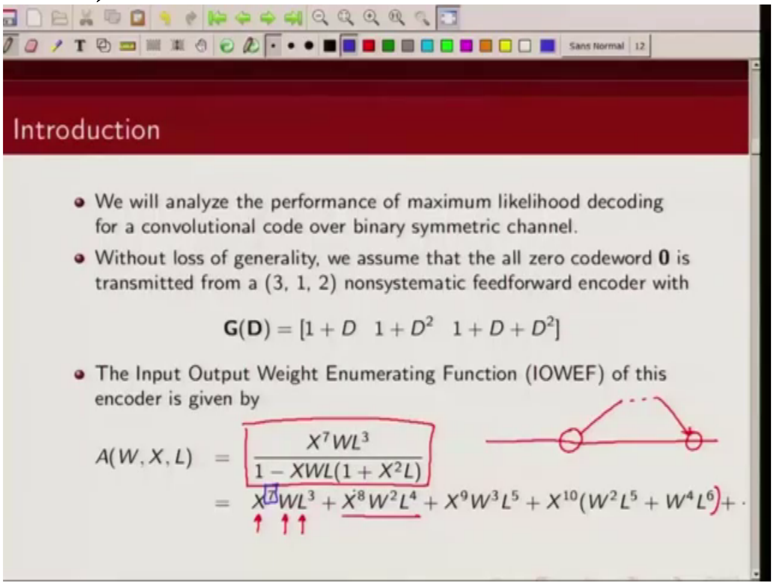
We can take the example of the same code that we are considering. Let's

(Refer Slide Time 08:45)



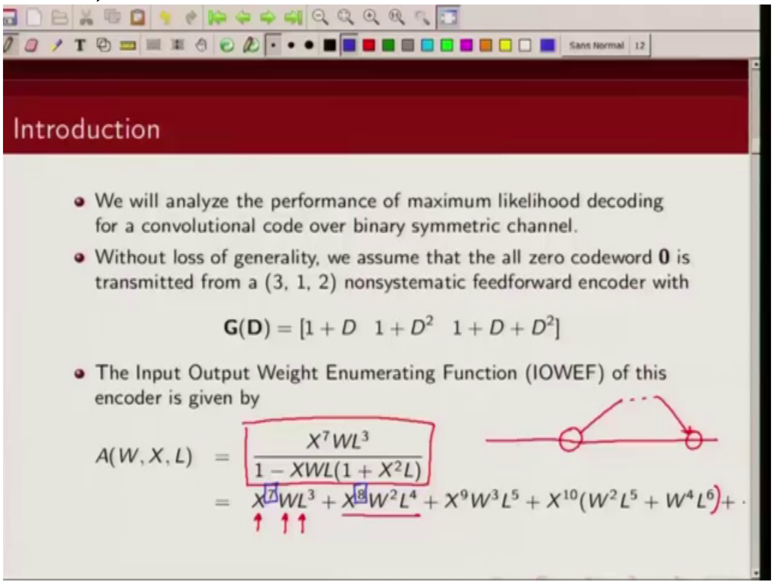
just this code as d free 7 then it has one

(Refer Slide Time 08:50)



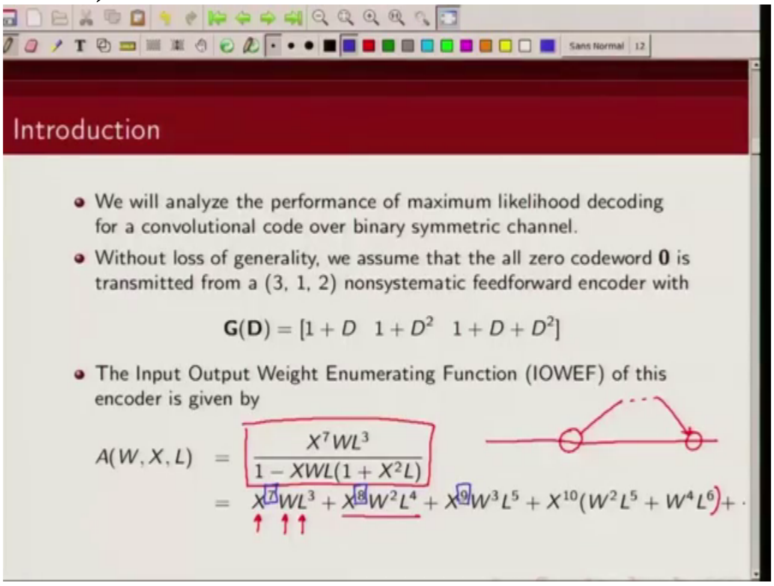
codeword of weight

(Refer Slide Time 08:51)



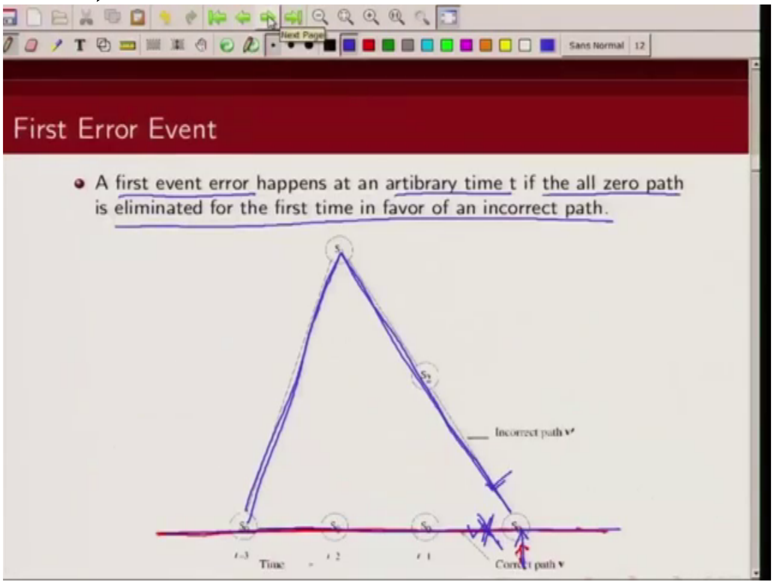
8, one codeword of weight

(Refer Slide Time 08:53)



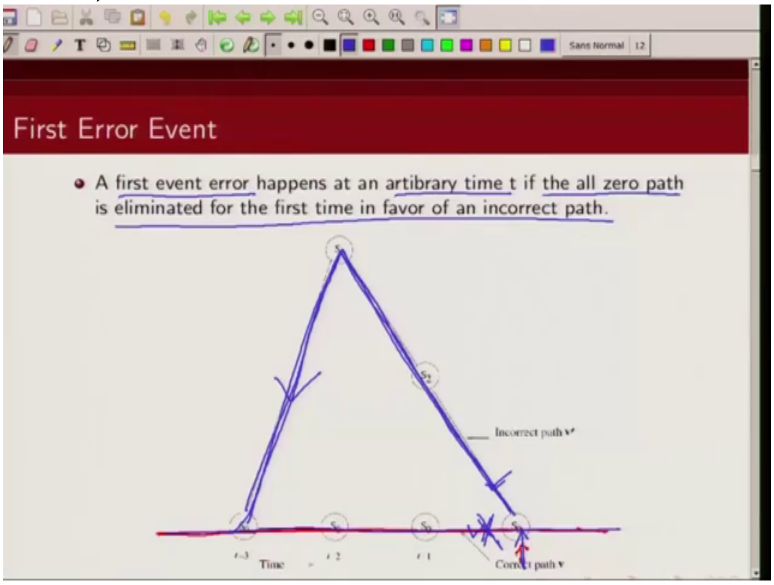
9, there are 2 codewords of weight 10 like that, so minimum distance d free is 7 here. So let's say, let's

(Refer Slide Time 09:02)



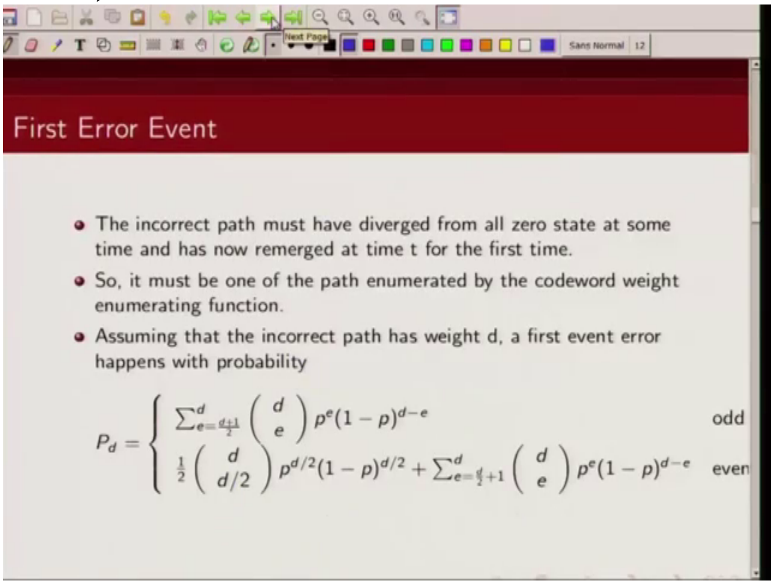
assume that this incorrect path has weight 7. Now when will you decide in favor of this path rather than this path? When the number of errors are such that when there are more than 4 errors such that your received sequence is closer to

(Refer Slide Time 09:20)



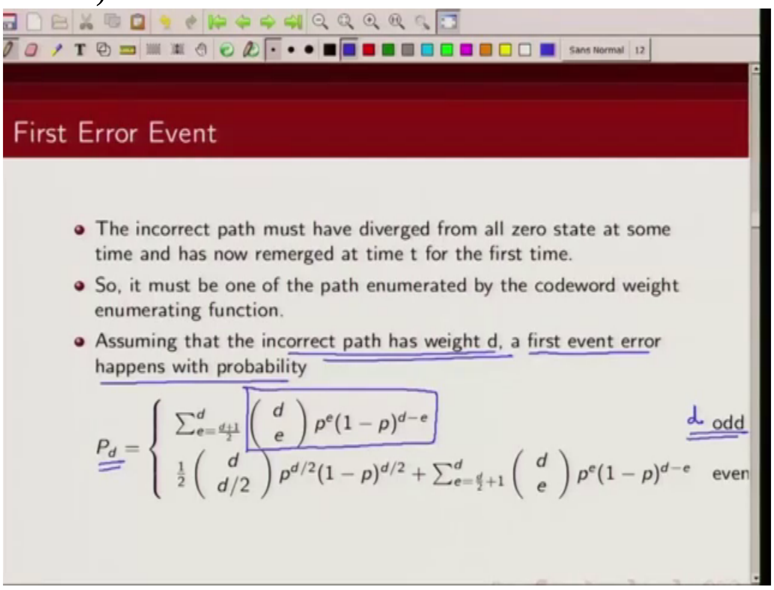
this sequence rather than this, right then you will make an error. Or let's say if this path was weight 8 then of course when your error is more than 4 bits you will decide in favor of this path and whenever error has happened in 4 bit location there is equal probability, you can either choose this path or you can choose this path.

(Refer Slide Time 09:47)



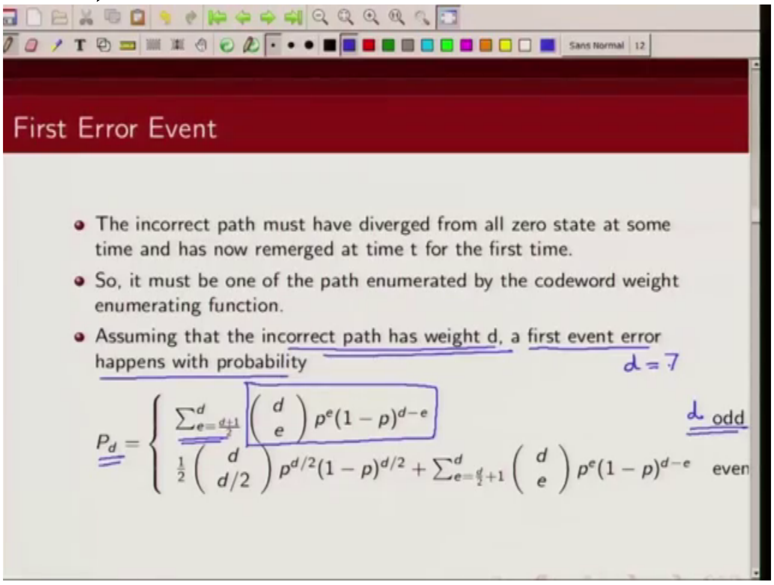
So we can calculate the probability of this first event error. We can find out its probability. What is its probability? Let's say this incorrect path has weight d. So d is the weight of this incorrect path which has diverged from all zero state at some other time and has merged back at all zero state at time t. Now if d is odd then this probability is given by sum of this probability and

(Refer Slide Time 10:31)



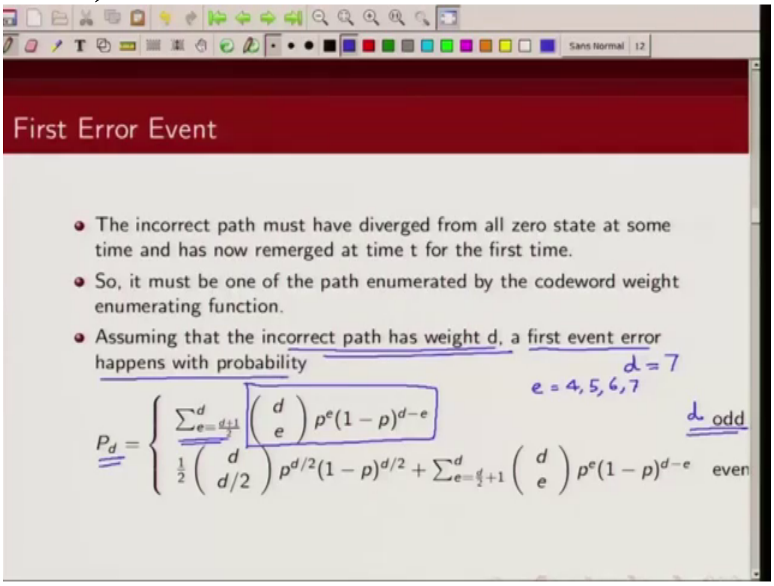
where summation takes place over all error pattern which are greater than d plus 1 by 2 up to d. So in the example we just considered, let us say if we had incorrect path of weight 7 then

(Refer Slide Time 10:51)



we should look at error pattern of 4, 5, 6 and 7. If 4

(Refer Slide Time 11:01)

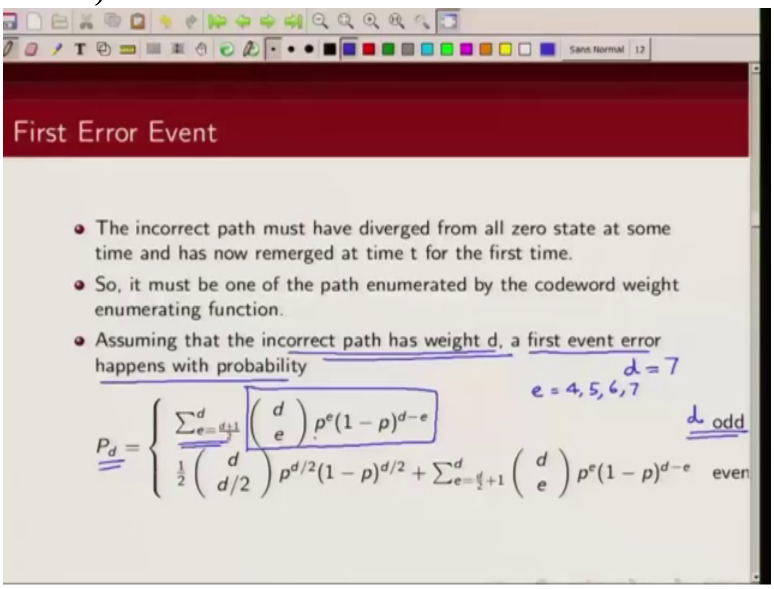


errors happened, may be it will bring us closer to this incorrect path rather than all zero path.

(Refer Slide Time 11:09)

Now p

(Refer Slide Time 11:12)

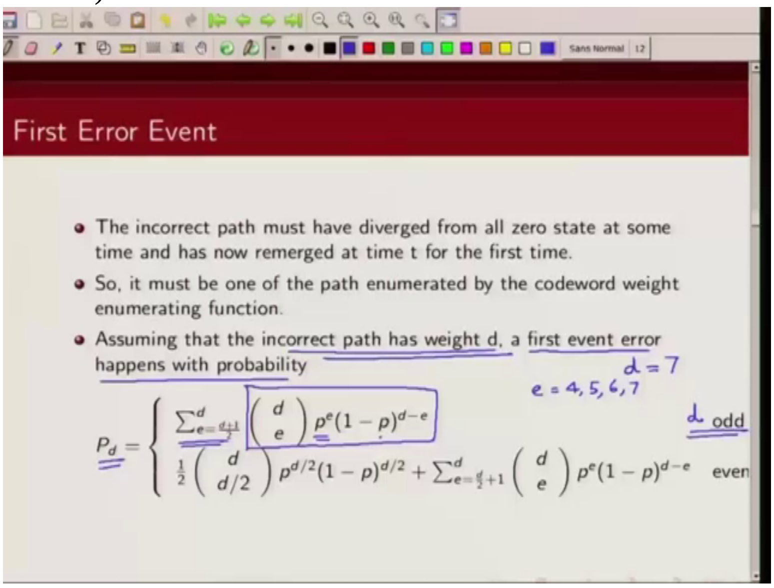


is the cross over probability of changing or flipping of the bits, so p times e will give us the probability that e number of bits have

(Refer Slide Time 11:24)

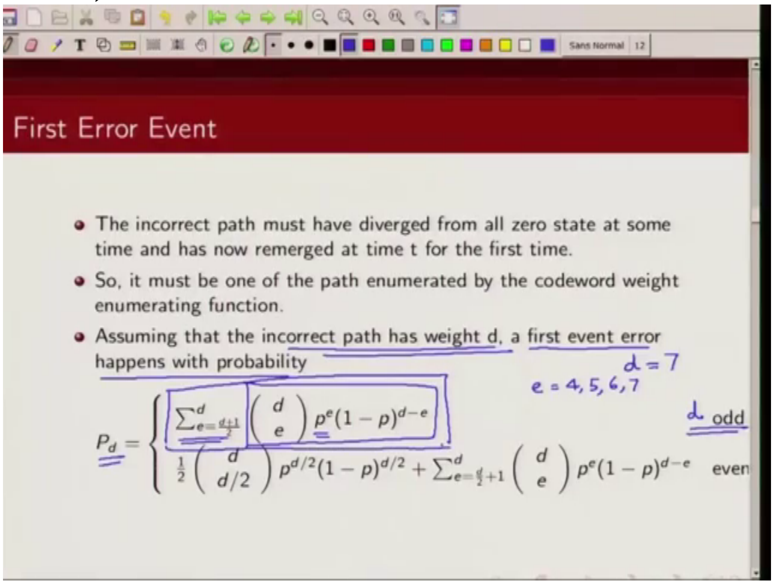
flipped and 1minus p

(Refer Slide Time 11:26)



raised to power c minus d will tell us the probability that d minus e bits have not flipped and d C d choose e is number of possible combinations of, in which we can have these error patterns, e error patterns among these set of d weight so this will give us the probability of first event error.

(Refer Slide Time 11:56)

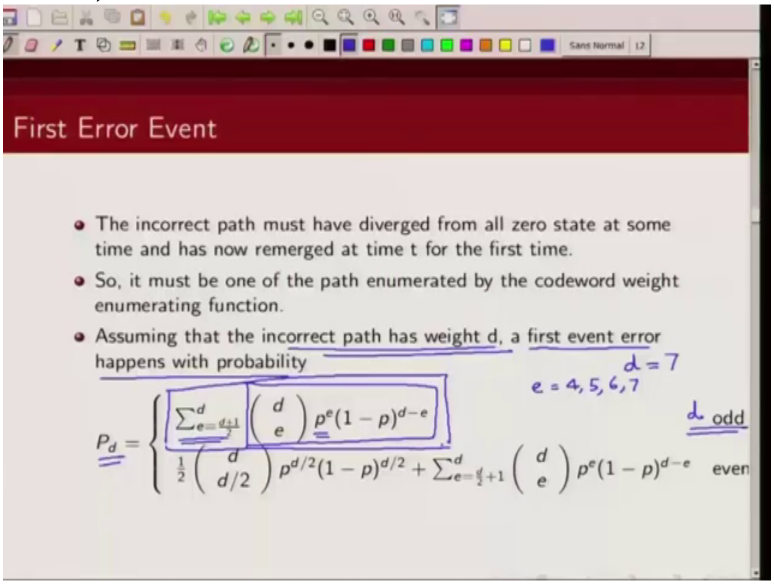


Again we will make a decision in favor of incorrect path if our received sequence is closer to

(Refer Slide Time 12:05)

incorrect path. And we know what is the maximum likelihood rule for binary symmetrical channel. It is we choose the codeword such that Hamming distance between the received codeword and this chosen codeword is minimized. So if there

(Refer Slide Time 12:20)

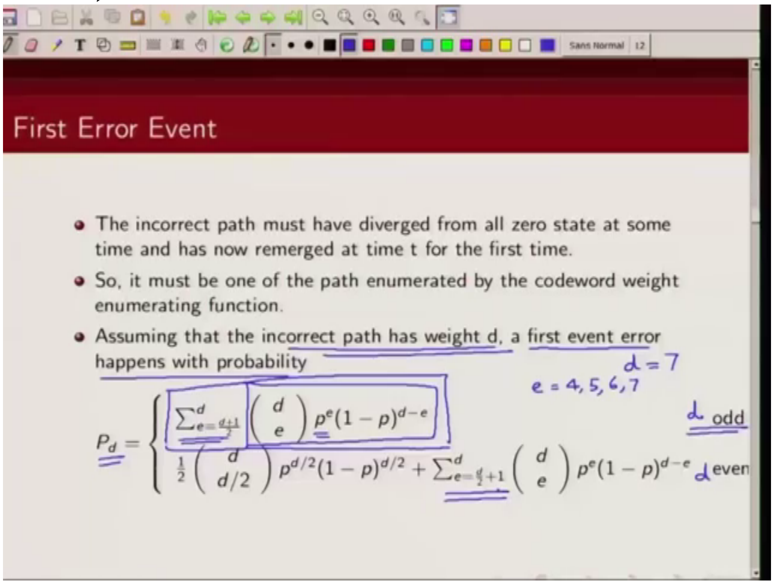


are more than, there are four or more errors for the case when incorrect path has weight 7, then you are going to choose, basically go for this incorrect path. What about when d is even? When d is even, for all error patterns from d by 2 plus 1 to d,

(Refer Slide Time 12:43)

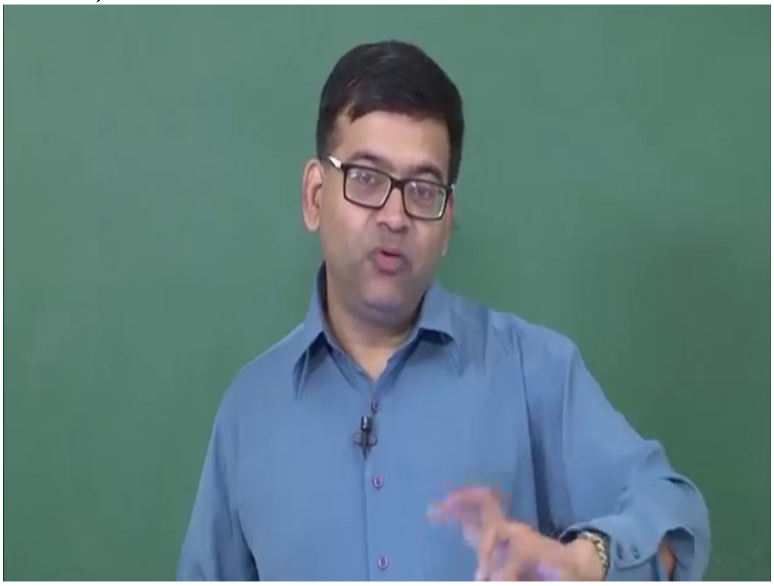
you will decide in favor of incorrect path and whenever the error is

(Refer Slide Time 12:49)



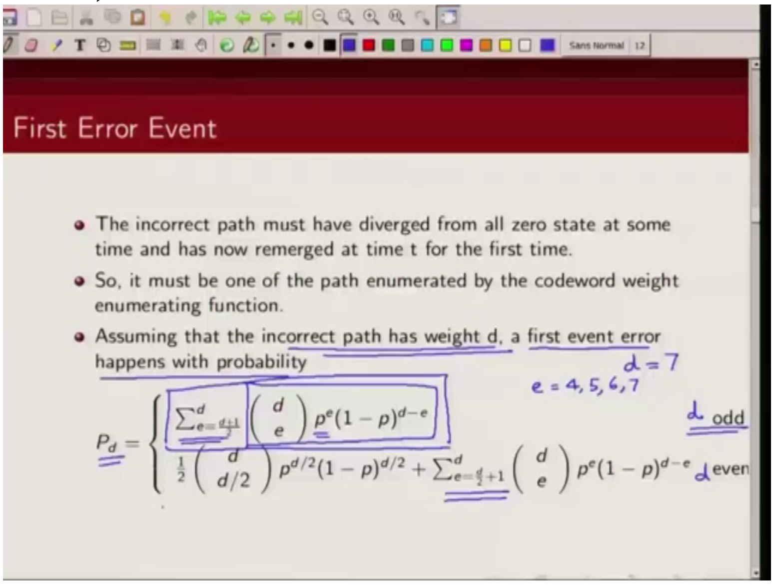
d by 2, then there is a 50:50 chance because you can just

(Refer Slide Time 12:56)



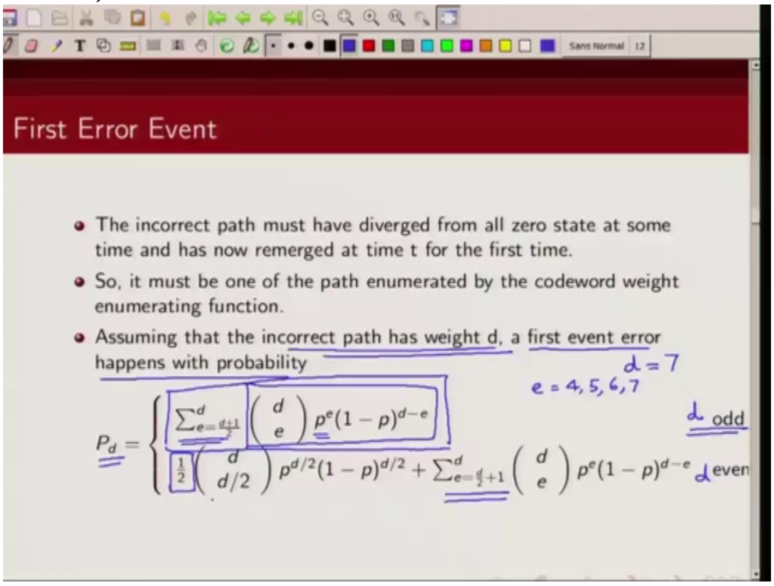
then, then there is a tie in the matrix, so you can just flip and choose either of the path. And that's why for

(Refer Slide Time 13:04)



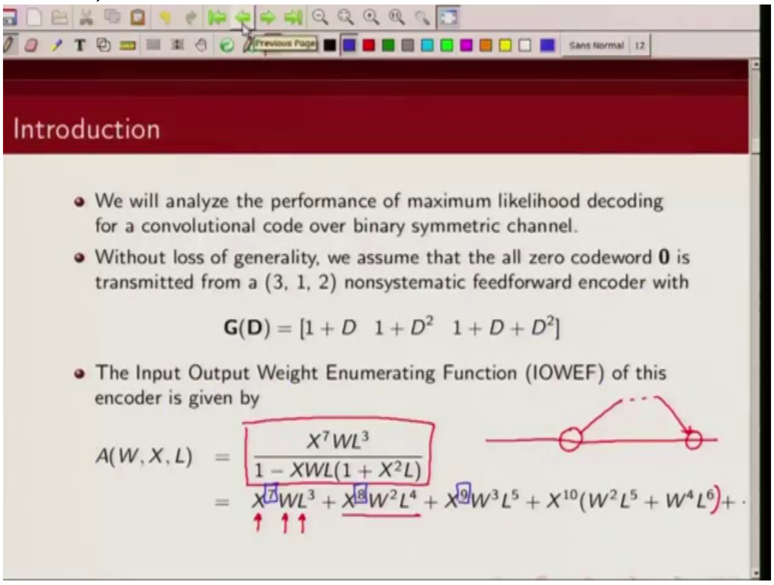
d by 2; I have written here it is half times

(Refer Slide Time 13:08)



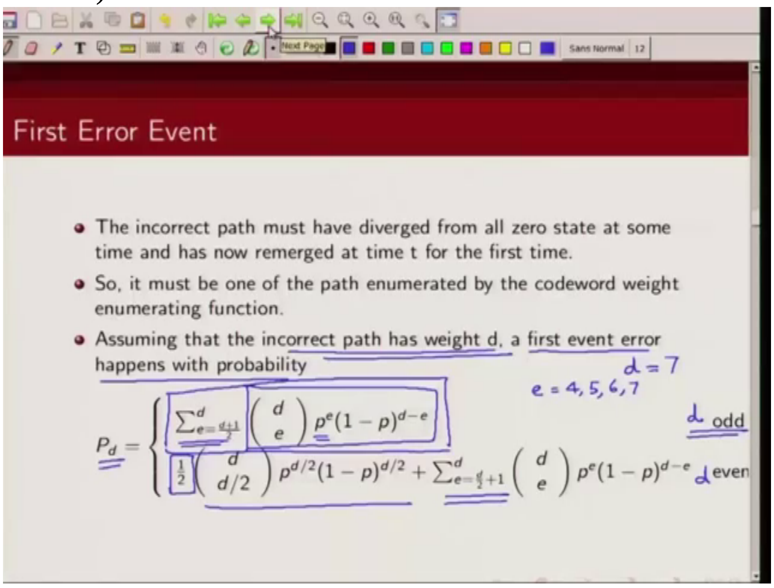
this probability plus all error patterns of weight more than d by 2. So for example in this case we had one path of

(Refer Slide Time 13:22)



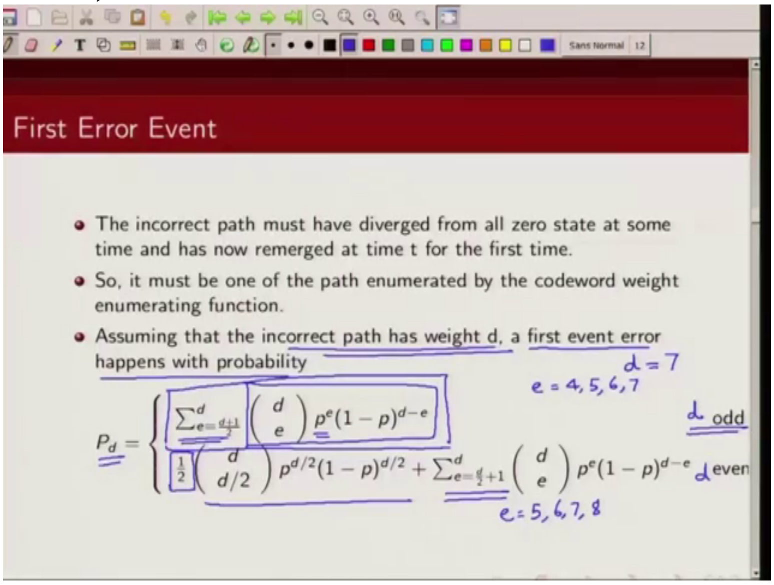
weight 8, right? So if our incorrect path was of weight 8, then

(Refer Slide Time 13:31)



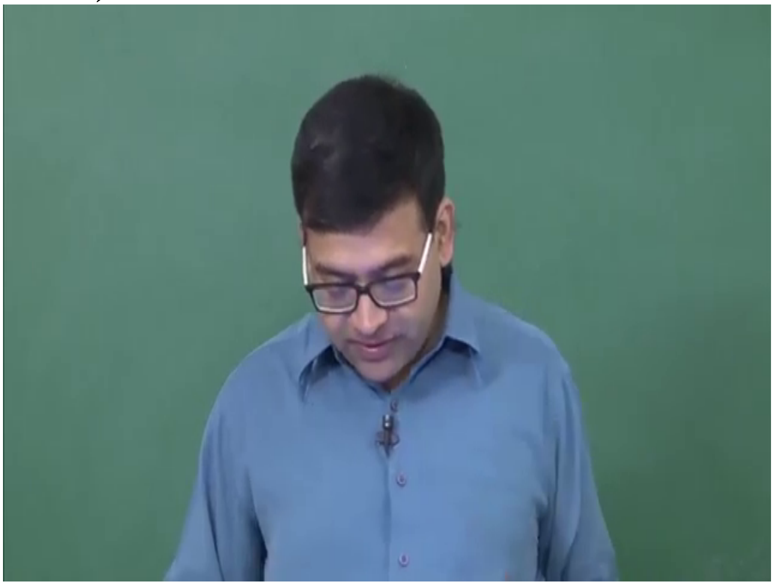
all error patterns of 5, 6, 7, 8 they

(Refer Slide Time 13:38)



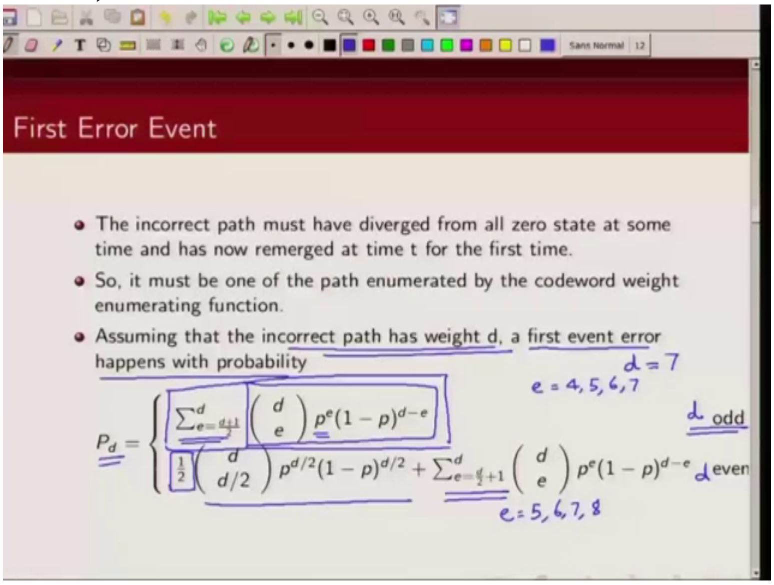
would have caused decision in favor of this incorrect path

(Refer Slide Time 13:43)



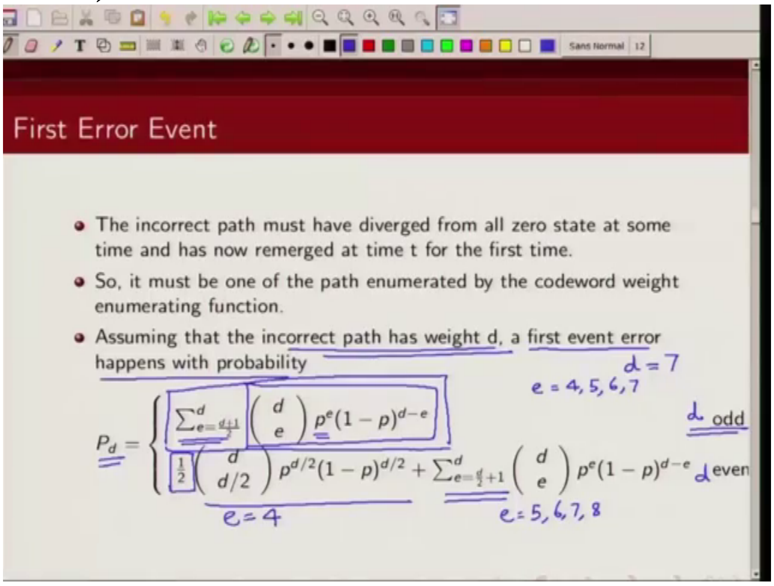
and whenever there is

(Refer Slide Time 13:45)



error pattern of

(Refer Slide Time 13:47)

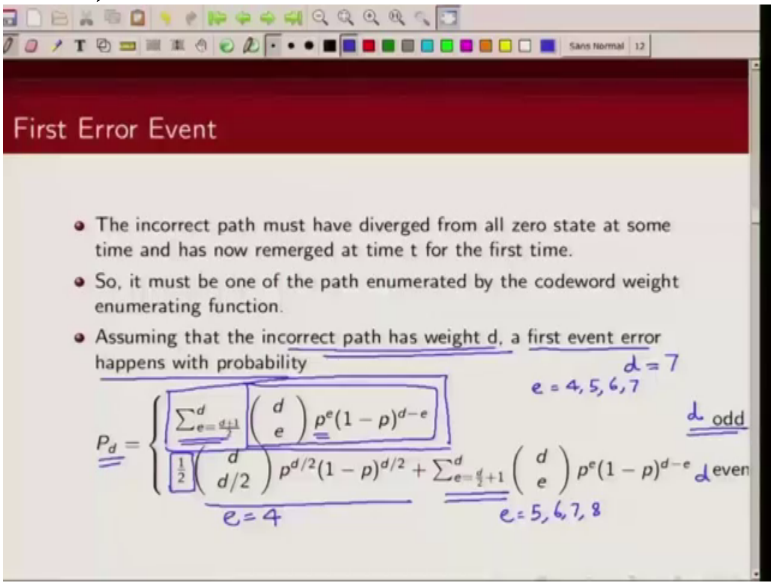


weight 4, there is a 50:50 probability that I may choose

(Refer Slide Time 13:53)

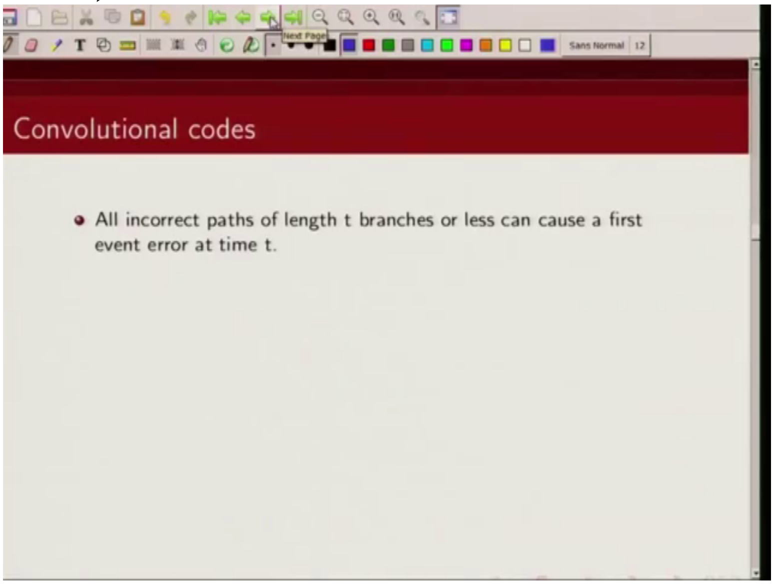
an incorrect path or I may choose an all zero path because the Hamming distance is same from all zero path or this incorrect path. So that's why

(Refer Slide Time 14:02)



I have a half here. So this is a probability of first event error. Now

(Refer Slide Time 14:12)

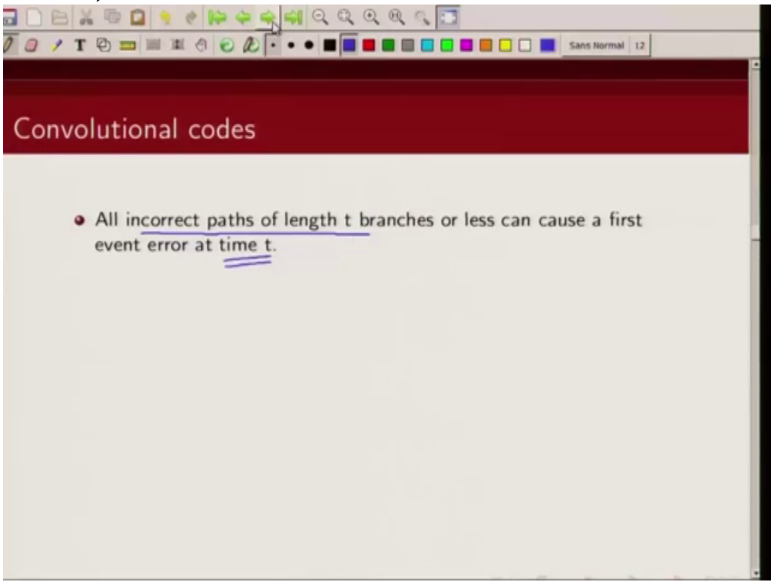


if we are looking at time t then all incorrect path lengths of length t or

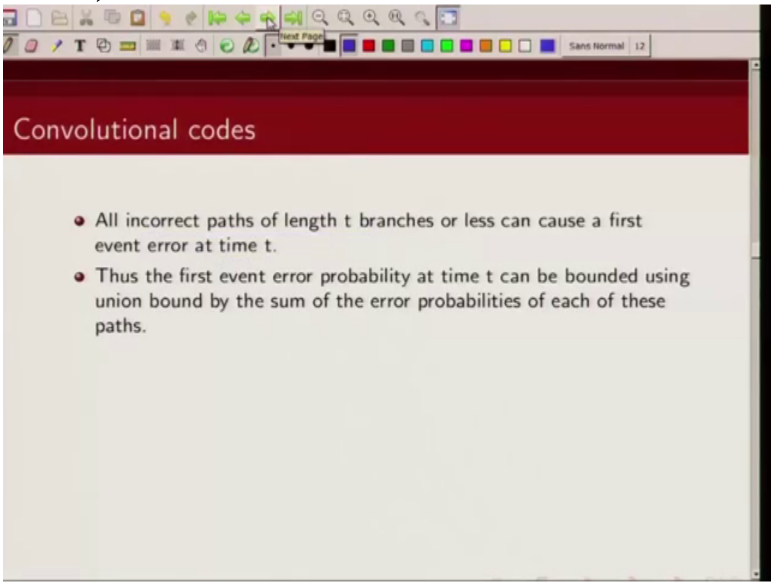
(Refer Slide Time 14:20)

less can cause first event error.

(Refer Slide Time 14:24)

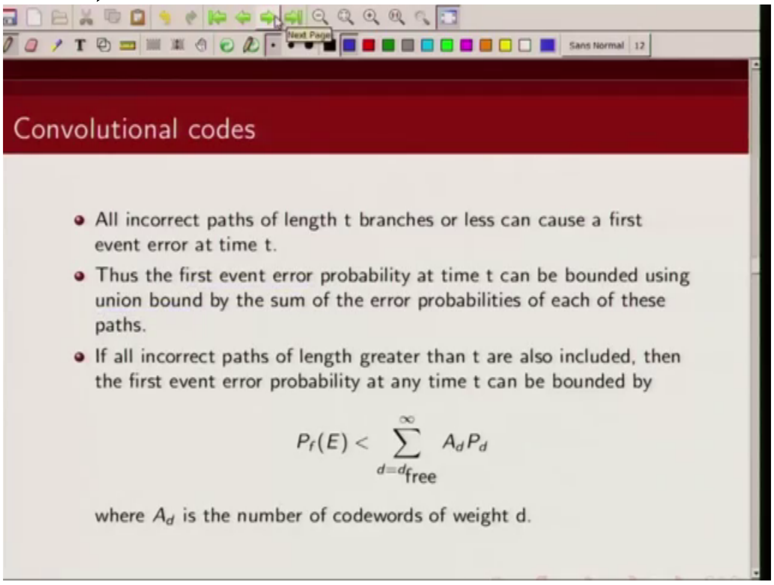


(Refer Slide Time 14:25)



So we would now use union bound to, essentially upper bound this probability of first event error probability. So what does union bound says? Probability of union of errors is basically upper bounded by sum of the probabilities of error. So we are going to upper bound this first event error probability by sum of error probabilities of all these incorrect paths. Now

(Refer Slide Time 15:01)

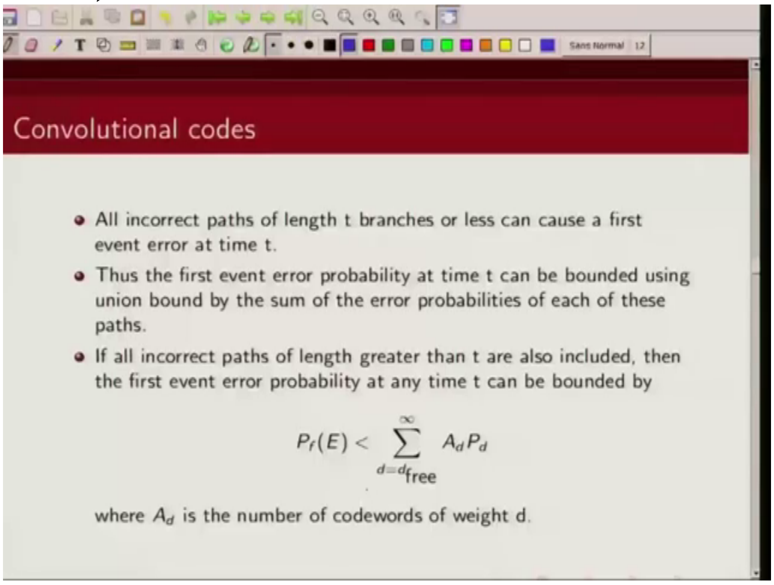


if we also allow

(Refer Slide Time 15:04)

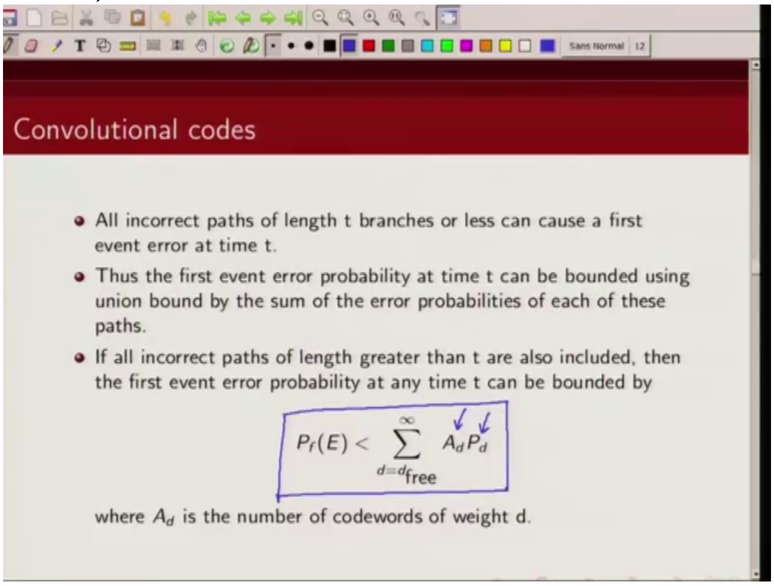
incorrect path of any length even greater than t, then in that

(Refer Slide Time 15:08)



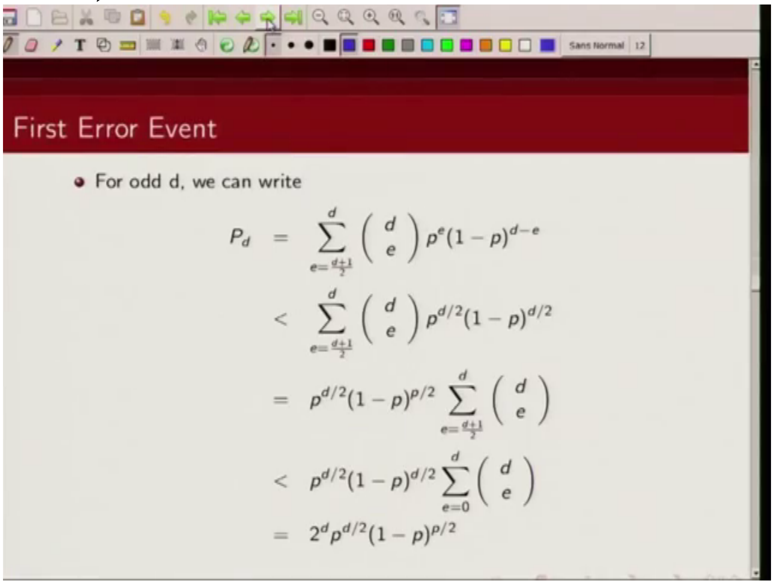
case the first event error probability can be upper bounded by this. So this is the probability of first event error of weight d and how many such weight d paths exist, that is given by this and you sum over your d which goes from free distance of convolutional code to infinity. So this will give you an upper bound

(Refer Slide Time 15:44)



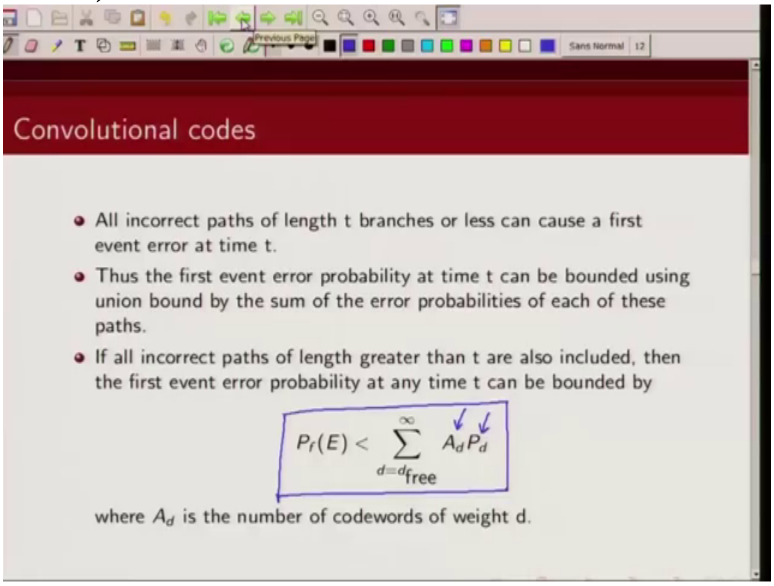
on probability of first error event probability. Now can we make use of the weight enumerating function of the convolutional code to calculate this? And this is what we are going to show.

(Refer Slide Time 16:05)



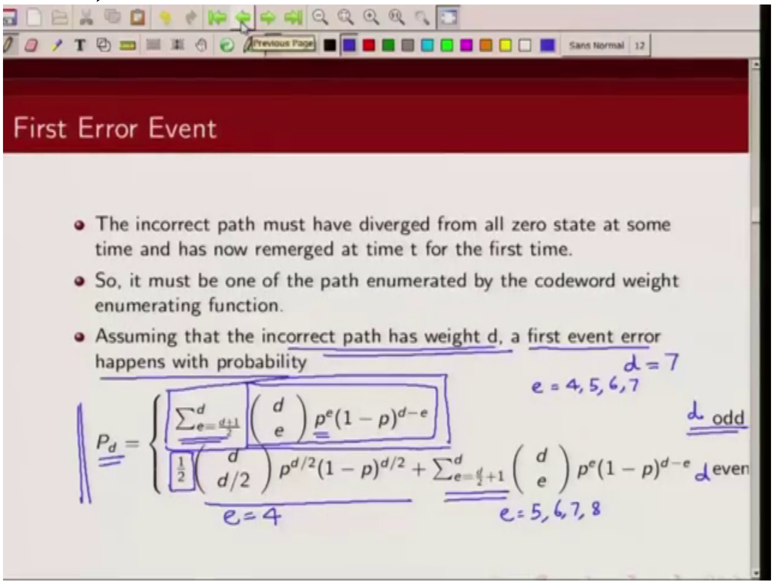
So let's first try to simplify the expression for probability of first event

(Refer Slide Time 16:15)



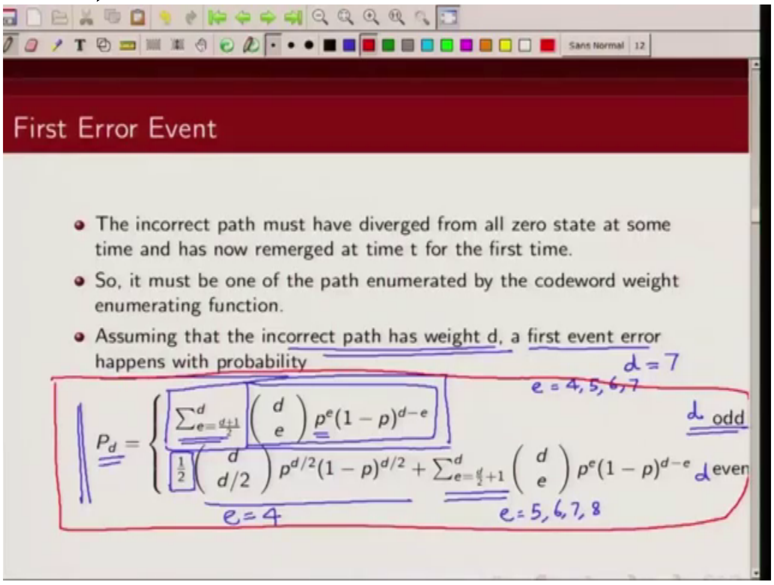
error. So

(Refer Slide Time 16:16)



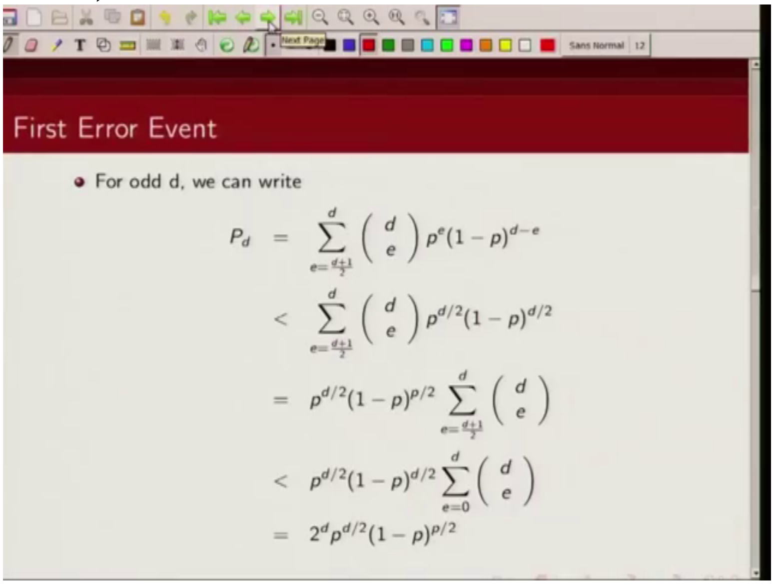
this was the expression for, this was the expression for first event error probability. So let us try to simplify this expression. So we will first do it for when d is odd and then we will do it for when d is even.

(Refer Slide Time 16:36)



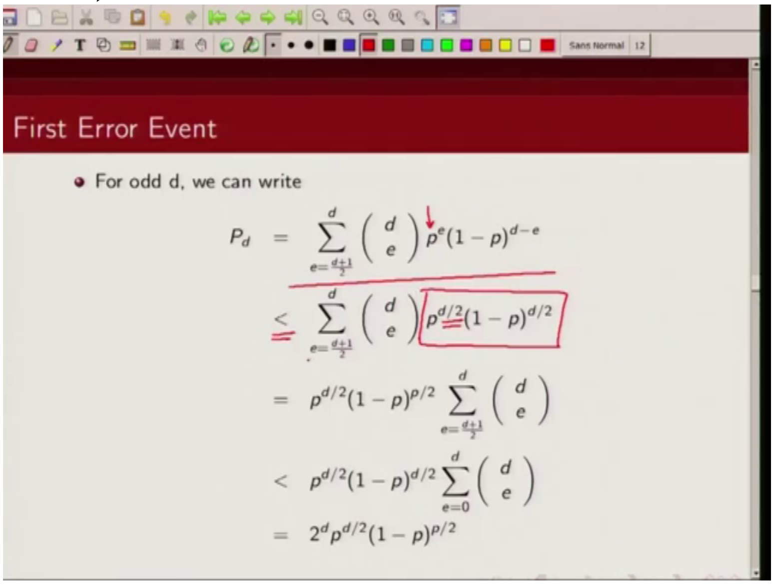
So when d is odd, the expression is given by this. Now note that this crossover probability is typically small. So and this again it is a number between 0 and 1, so if you raise it by a higher number you get smaller quantity. So the first upper bounding that we are doing is instead of raising it to e, we are raising it to d by 2 and again because it is a number between 0 and 1, see the actual expression would have been p raised to power d plus 1 by 2, then d plus 3 by 2 by 2, and like that, basically this number will go on decreasing because you are raising it to a higher value. Now I can just fix, instead of varying it as fix, I keep it as fixed and I kept it as d by 2 and that's the first upper bound that we are getting. Next this term does not depend on

(Refer Slide Time 17:47)



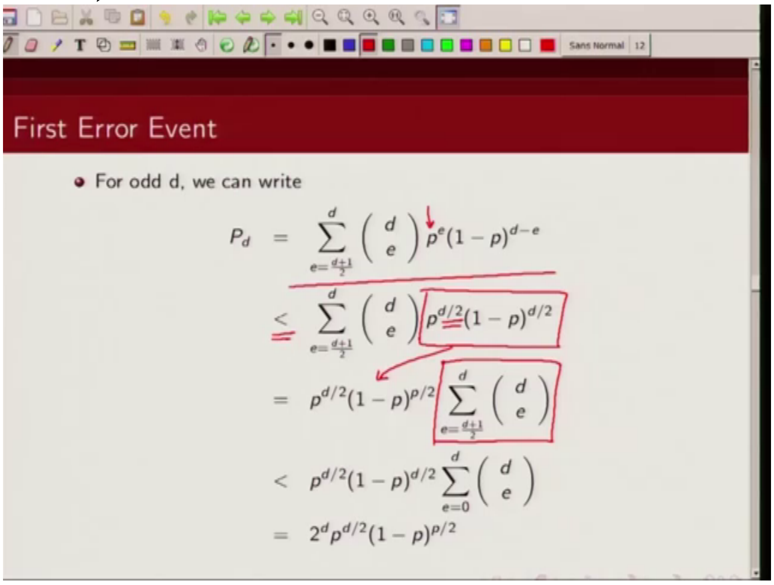
e. So I take this out. So what I am left with is this,

(Refer Slide Time 17:55)



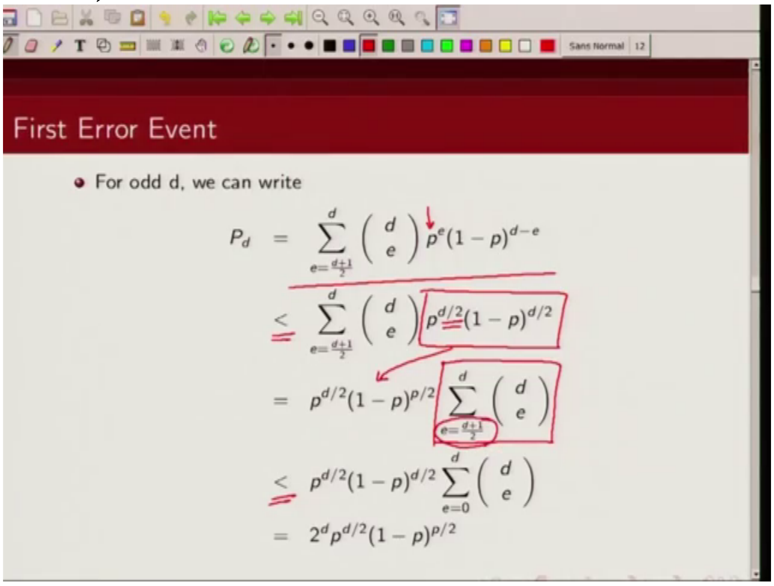
d choose e where e goes from d plus 1 by 2 to d. Now next upper bounding what I did was I replaced it by

(Refer Slide Time 18:06)



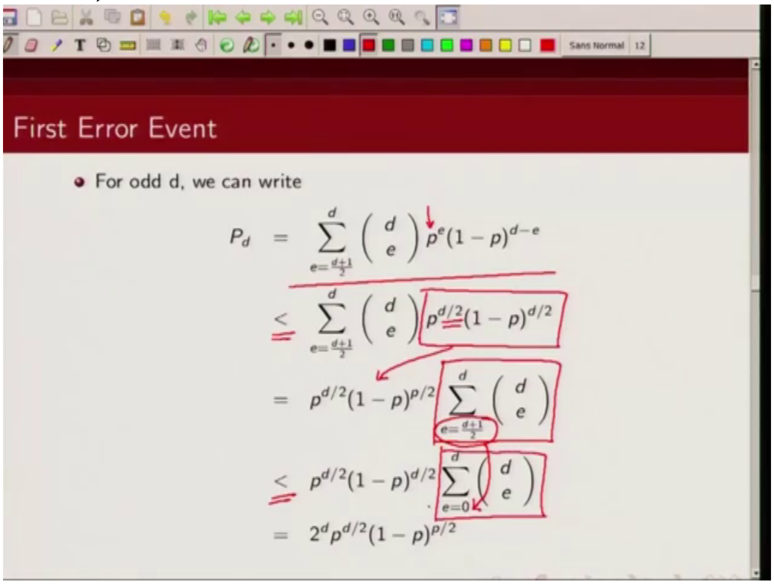
e equal to zero. So instead of e equal to d plus 1 by 2, I am adding more terms. I added from e equal to 0. So that's why this upper bound came and what is this,

(Refer Slide Time 18:23)



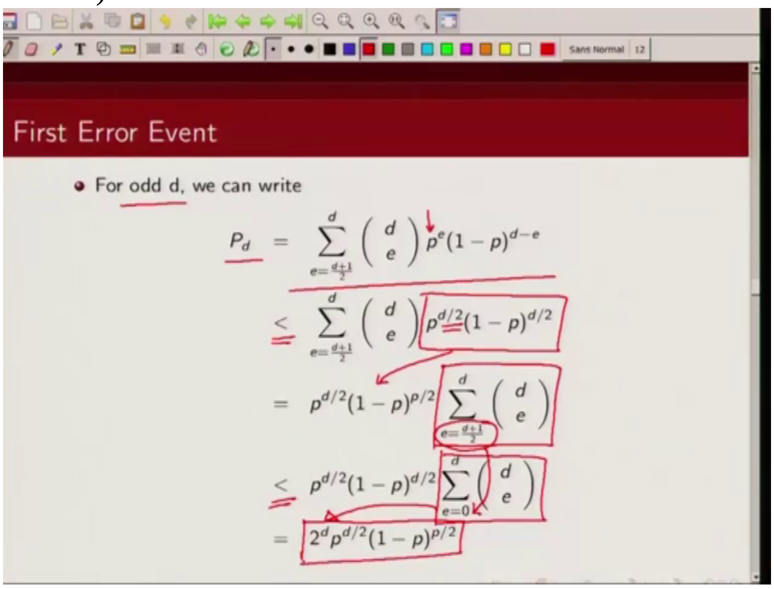
this is nothing but 2 raised to power d. So then for odd d I can write this first event error probability as upper bounded by

(Refer Slide Time 18:38)



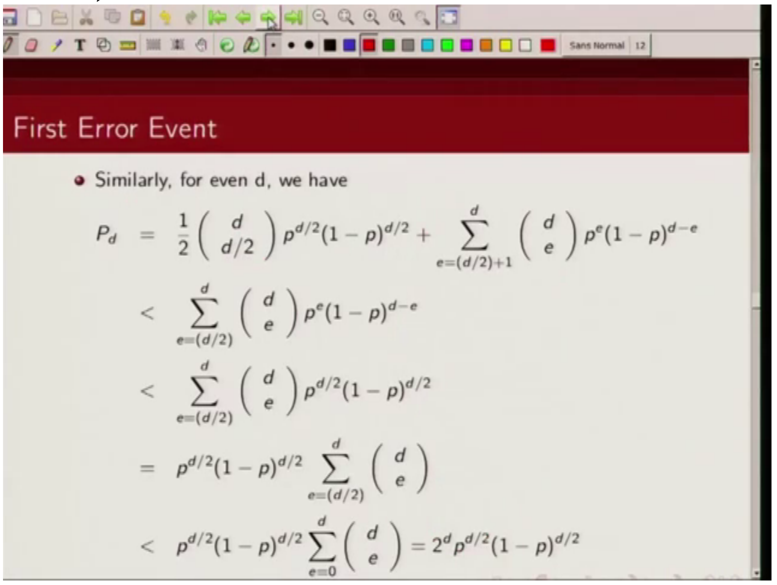
this quantity. Next

(Refer Slide Time 18:42)



let's do the same thing for even d. So for an even d, the expression for first event error is given by this expression. So first thing that I do was I upper bound this by changing this exponent from e by 2 plus 1 to e by 2. Note that in this expression when the error pattern is d by 2, I considered a 50% probability of

(Refer Slide Time 19:15)

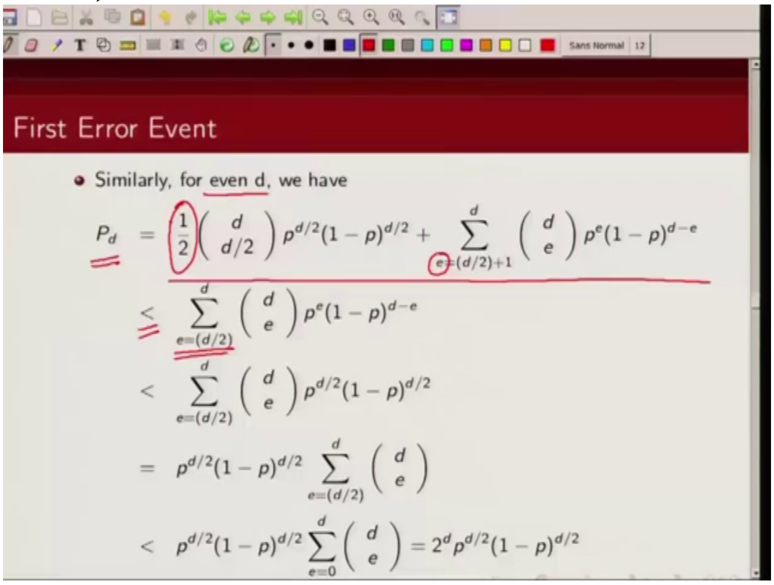


making an error. Now I just removed that and I added that here. I removed that

(Refer Slide Time 19:21)

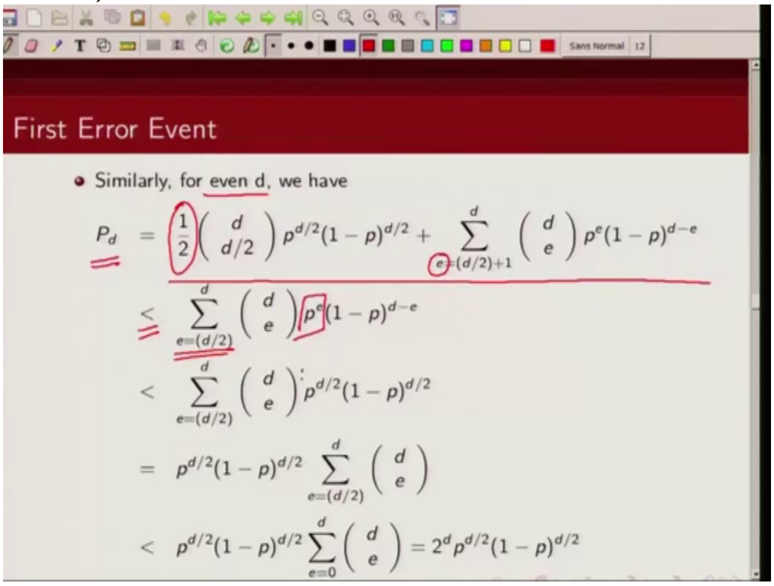
50% thing here, so that's why I am getting an upper bound here. Next same as in previous expression I replace p raised to power e by p

(Refer Slide Time 19:31)



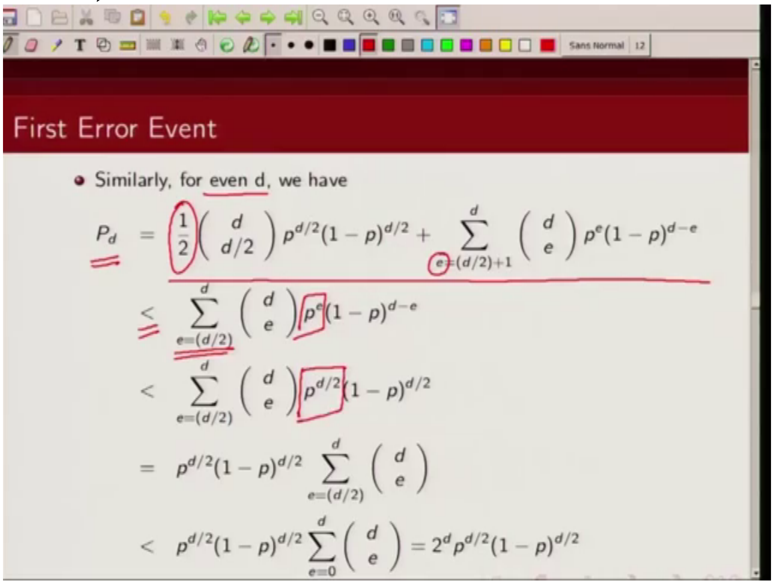
raised to power d by 2. And again

(Refer Slide Time 19:33)



because p is a number between 0 and 1, if you raise

(Refer Slide Time 19:37)

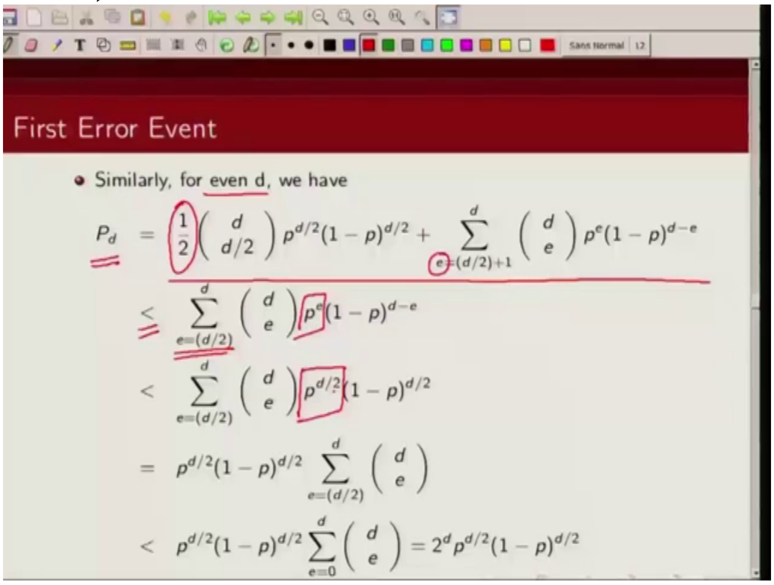


it to a higher number it decreases. So I just

(Refer Slide Time 19:41)

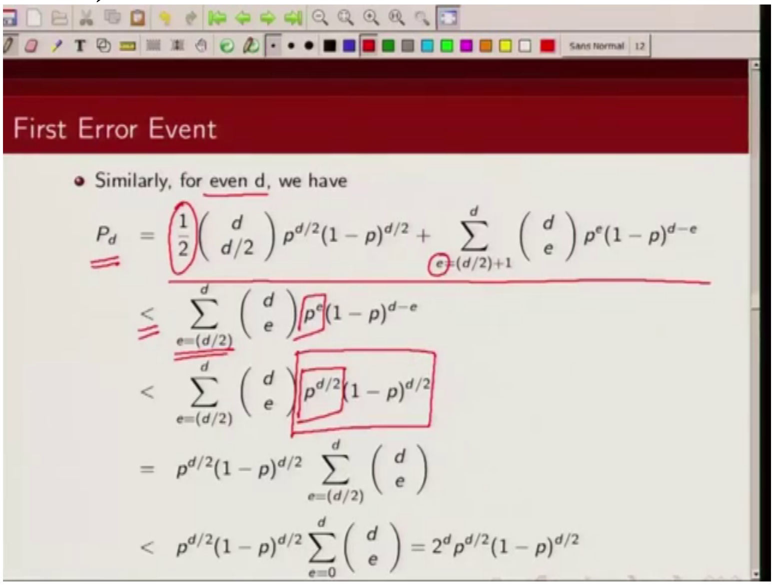
kept it at d by 2 which is a smallest value of e. Third thing which I do is again like for the odd case this term

(Refer Slide Time 19:51)



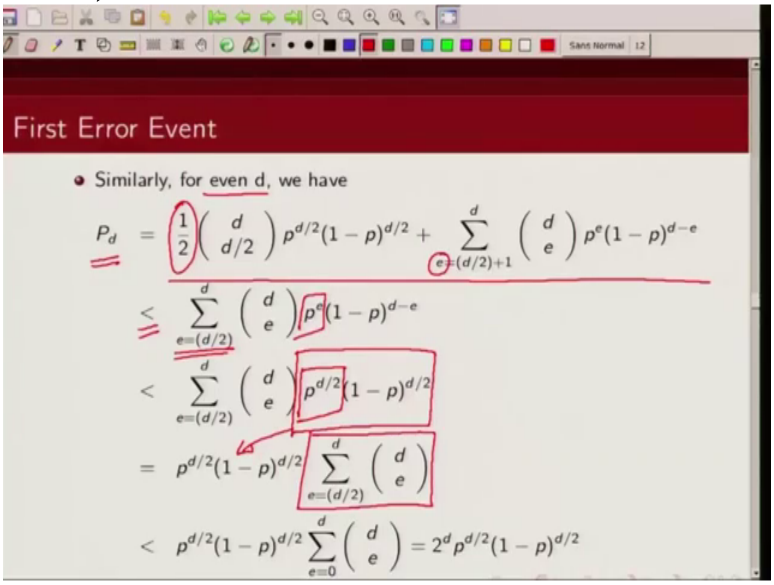
does not depend on e, so I bring it out. So I brought it out. And what I am left is with this,

(Refer Slide Time 20:00)



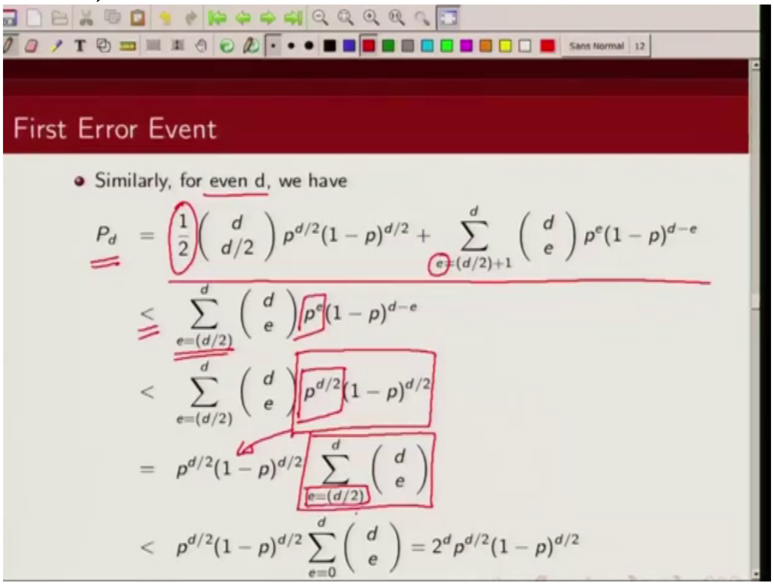
right and then I add terms from e 0 to up to d by 2 so I replace this

(Refer Slide Time 20:10)



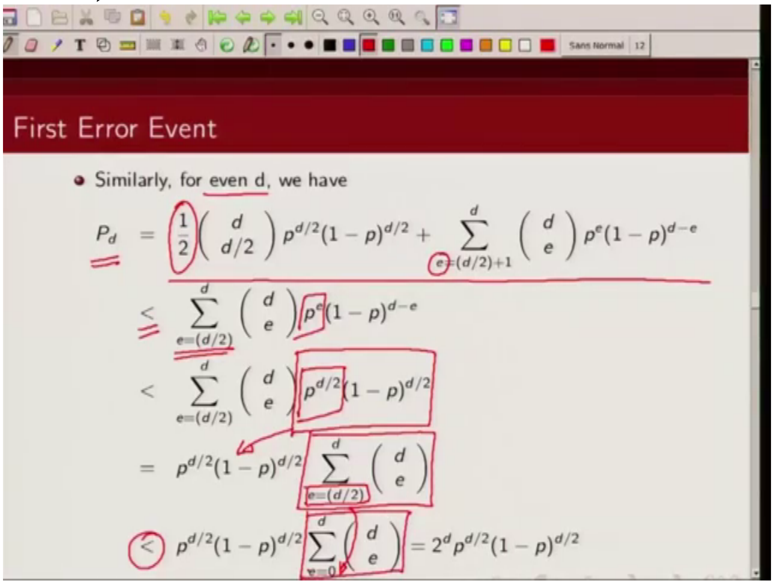
by this. So again I am upper bounding it, because I am adding additional terms here. So this quantity will be less than this quantity. And finally this is 2 raised to power d.

(Refer Slide Time 20:24)



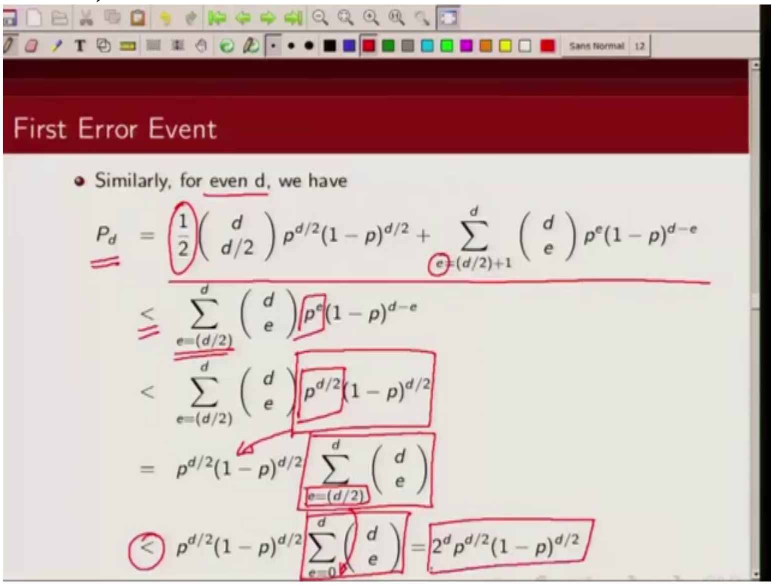
So I get my expression for

(Refer Slide Time 20:28)



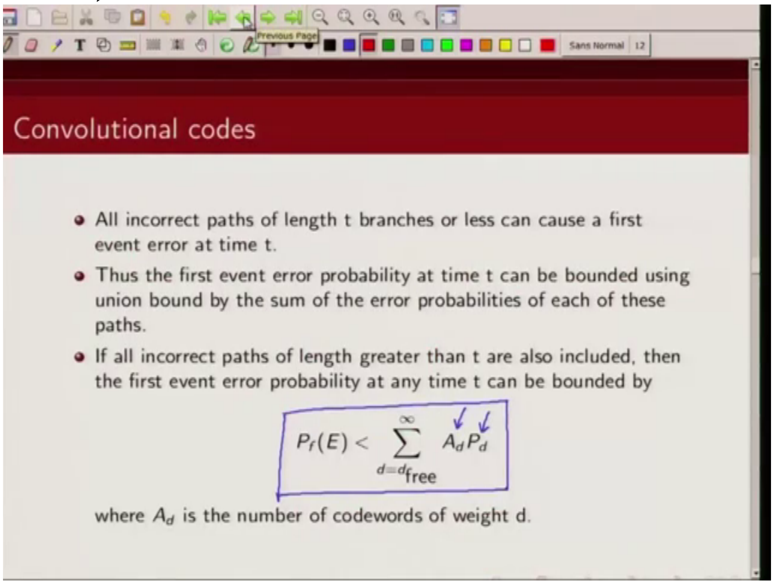
first event error probability for d being odd given by this expression. So p d is upper bounded by this for when d is even and we showed even

(Refer Slide Time 20:45)



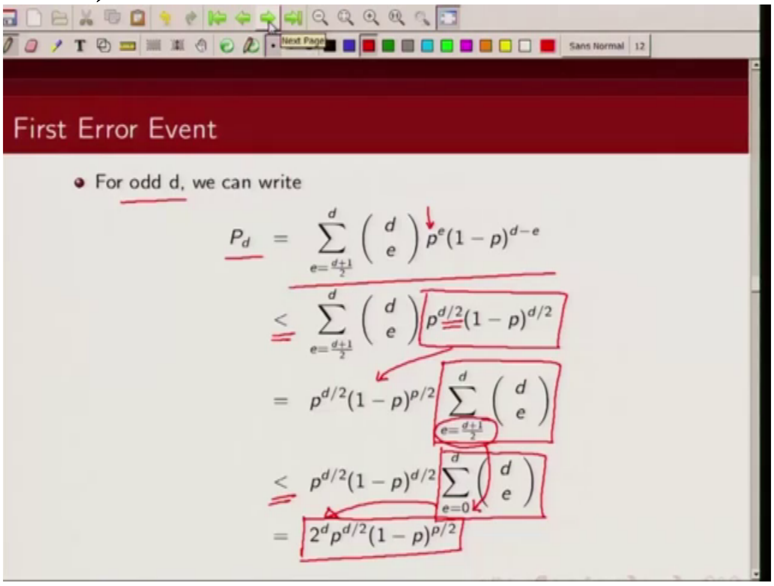
for

(Refer Slide Time 20:46)



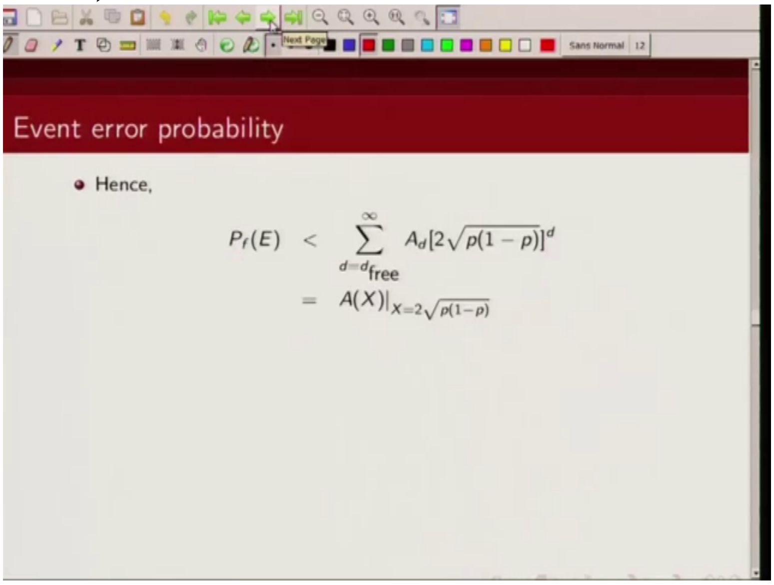
when d is odd it is upper bounded by the same quantity. So then

(Refer Slide Time 20:55)



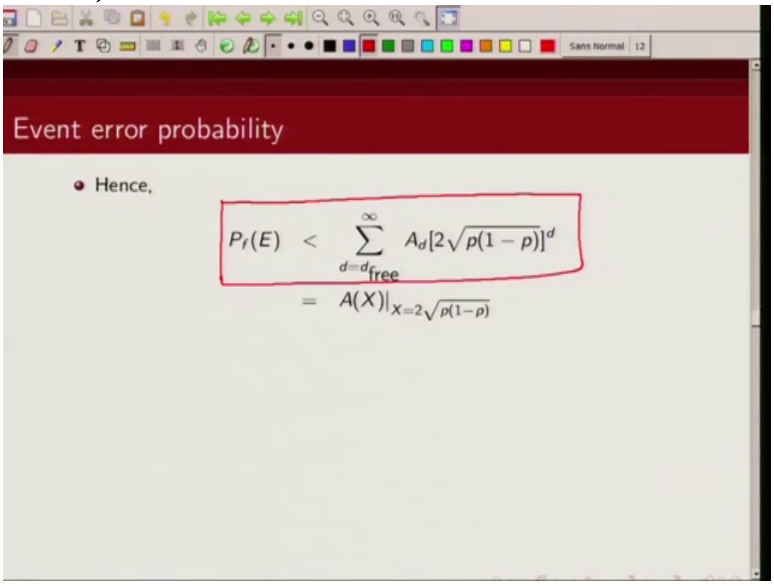
what we can do is we can write that our first event error probability is upper bounded by this.

(Refer Slide Time 21:04)



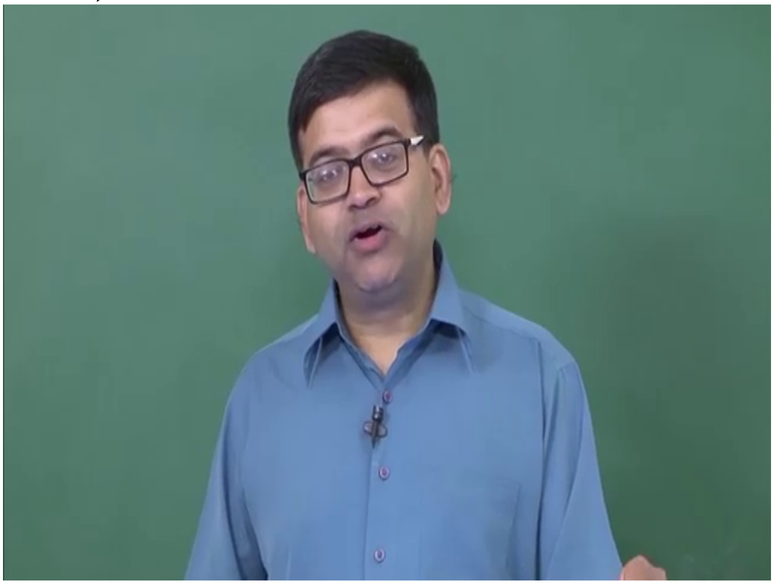
We just showed for d equal to

(Refer Slide Time 21:08)



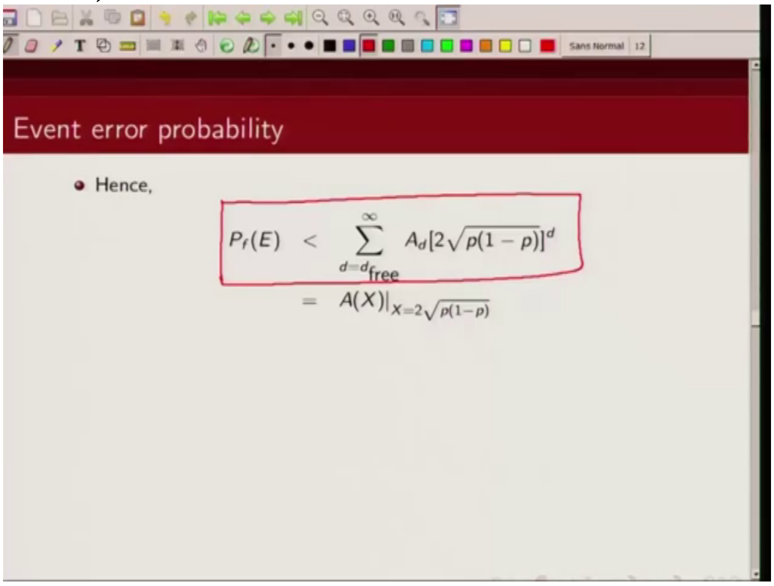
odd and d equal to even separately. So we can upper bound this

(Refer Slide Time 21:13)



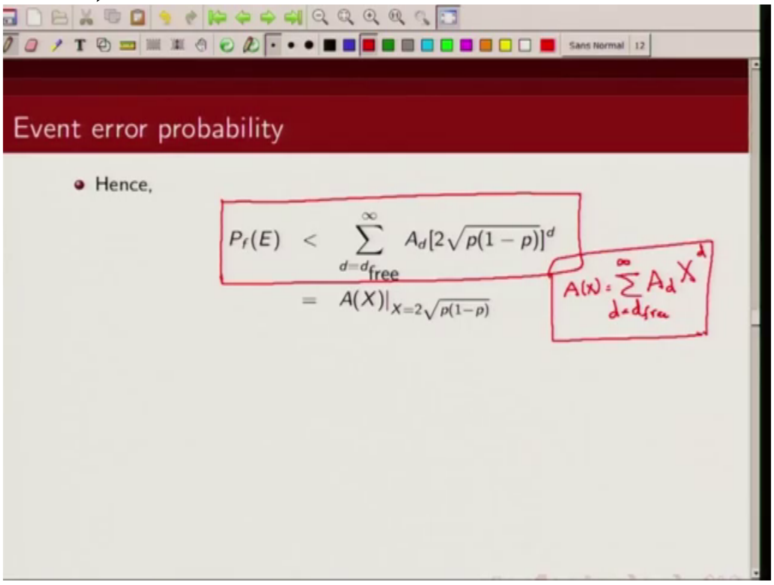
probability like this. Now what we are going to do is we know our weight enumerating function so will give us the distribution, so if we have some weight enumerating function, it essentially will give us from d to d free to infinity it will give us how many codewords of weight d are there and x raised to power d, Ok. So we are essentially making use of this weight enumerating

(Refer Slide Time 21:54)



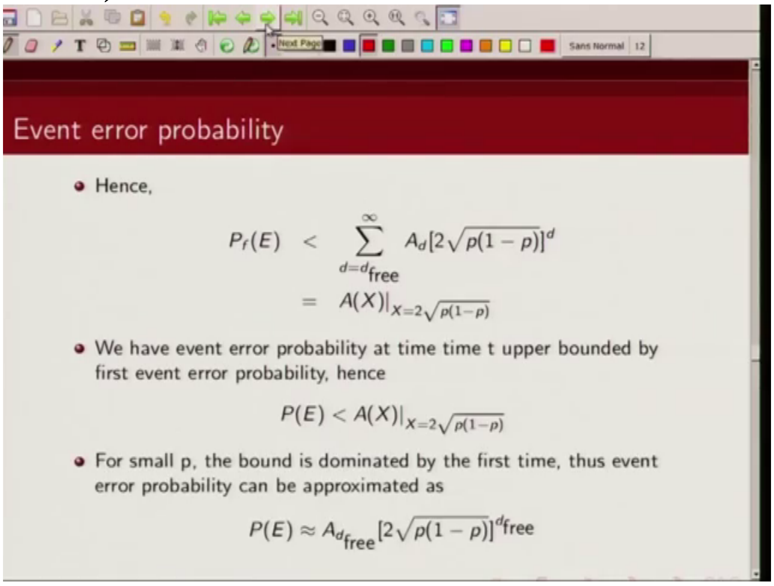
function and we look at the form of our first event error probability, so looking at these two then we can write then that first event error probability can be computed from the weight enumerating function by replacing the weight, that x by this quantity. So this we obtain by comparing this equation with this equation, Ok. Now

(Refer Slide Time 22:32)



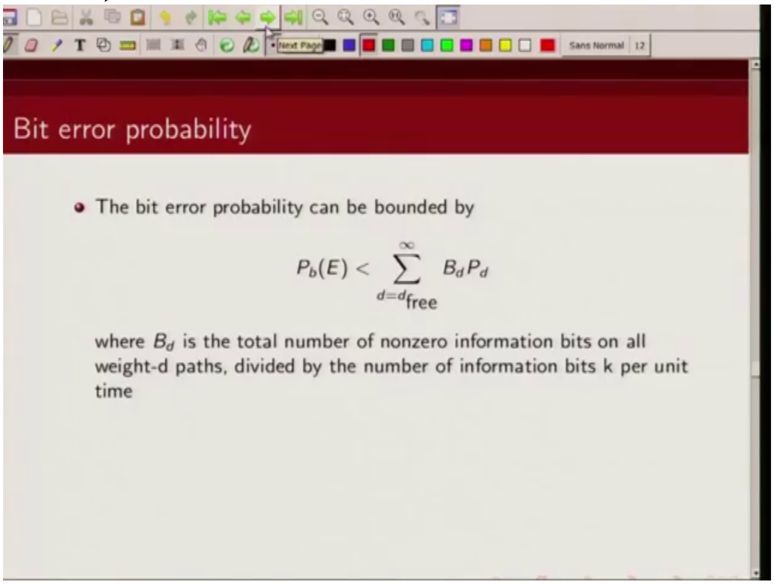
what sorts of error events can happen? So let's spend on that,

(Refer Slide Time 22:38)

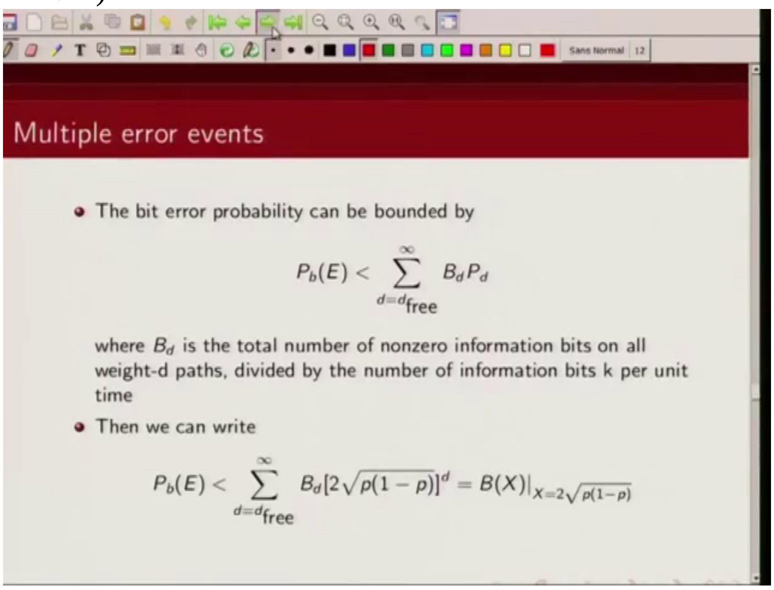


what sorts of events can happen.

(Refer Slide Time 22:40)

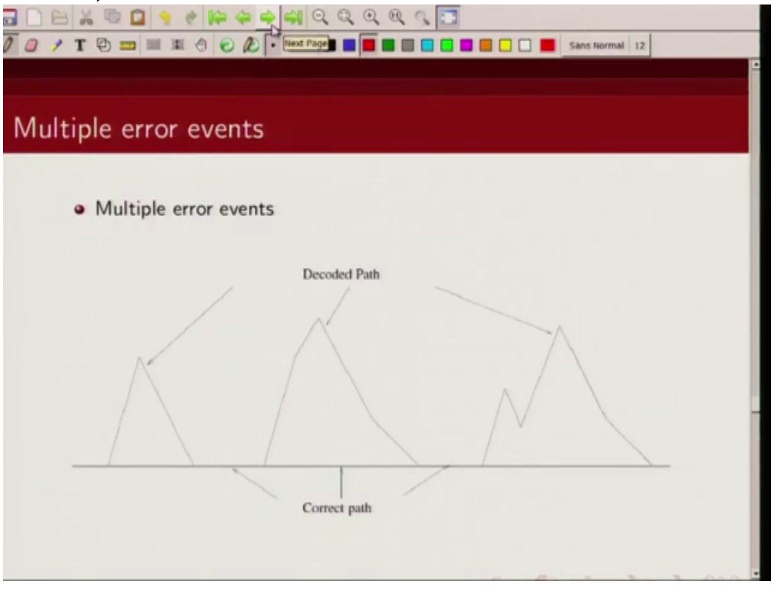


(Refer Slide Time 22:40)



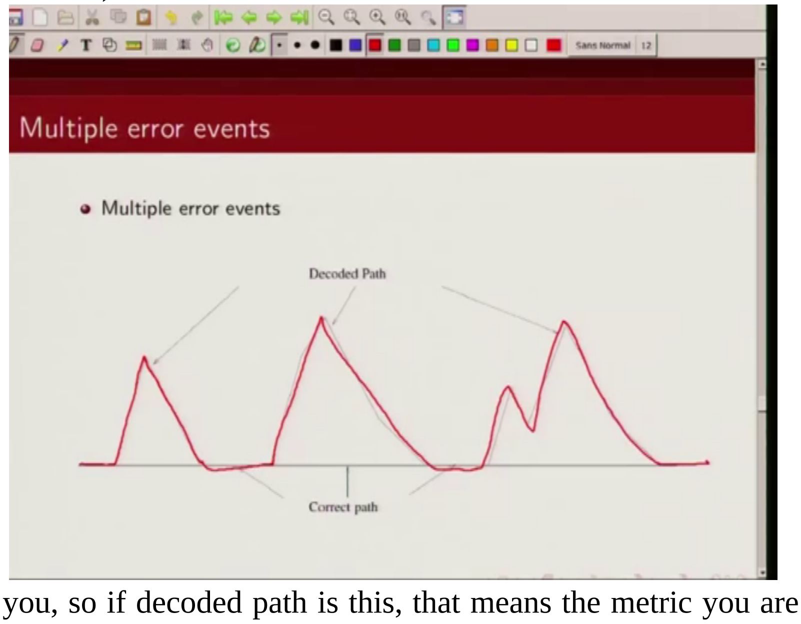
So

(Refer Slide Time 22:42)



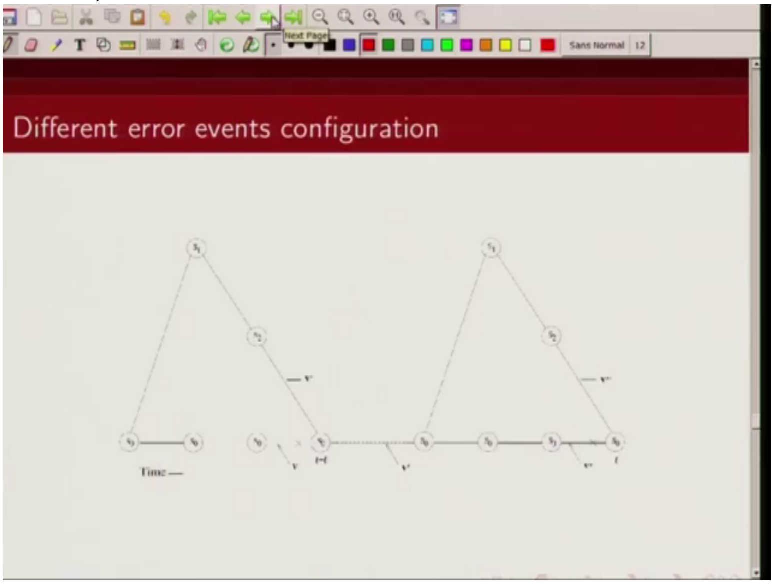
of course you can have a scenario where you have multiple error events, so you had, you diverged from all zero state, then you merged back, then you again diverged, you merged back, you diverged, you merged back, this can happen.

(Refer Slide Time 22:55)



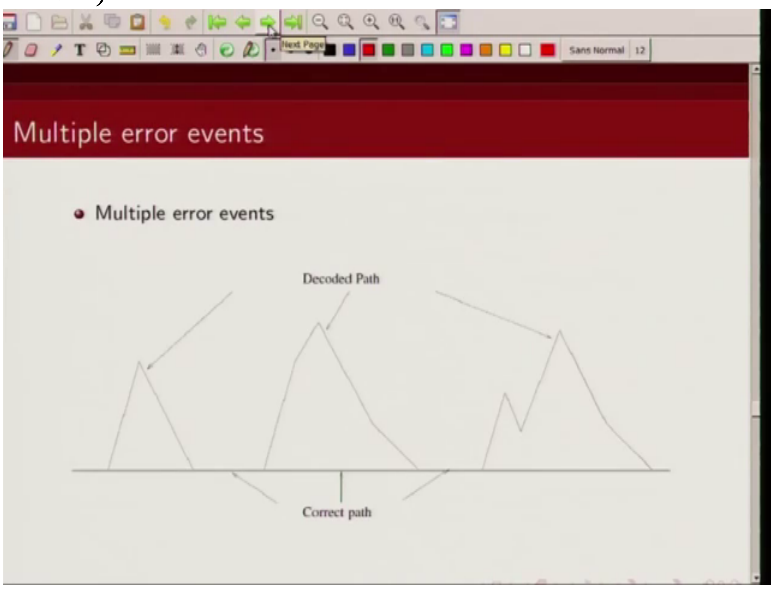
Now when will you, so if decoded path is this, that means the metric you are getting here is better than the metric value for the all zero state and that's why you are deciding in favor of this. Now there are multiple ways

(Refer Slide Time 23:13)



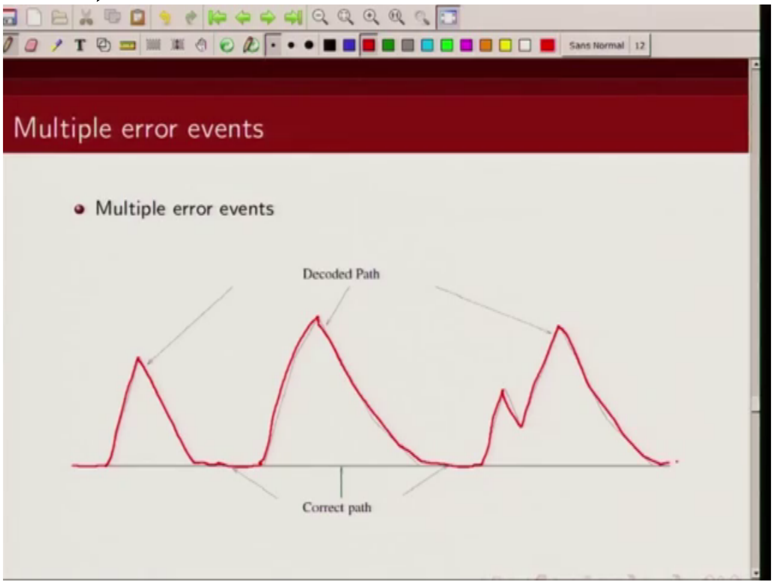
in which these errors can happen.

(Refer Slide Time 23:18)



So error can happen, error events can happen in multiple ways. So let's say there might be situation when you might have diverged from all zero state and then merged back into all zero state. And then after some time you again diverged from all zero state, stay away from all zero state and then you would merge back and then again you stay away from all zero state, diverge and then

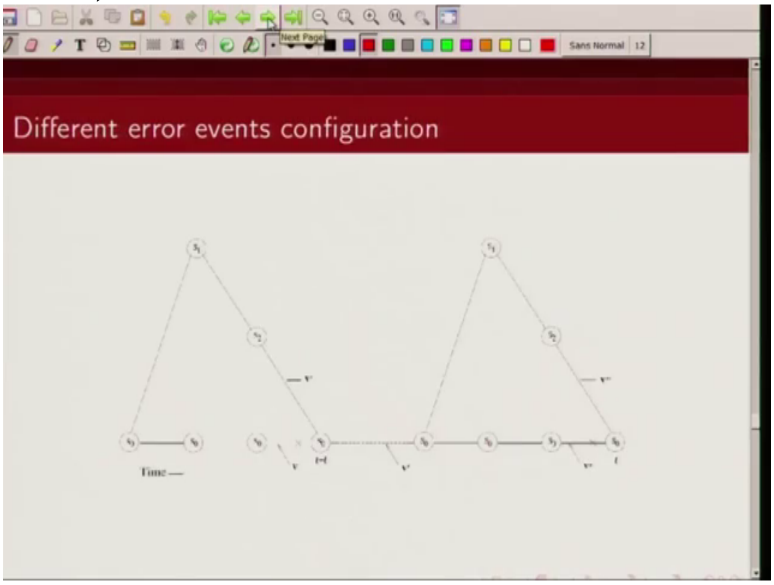
(Refer Slide Time 23:46)



then again merge back. So you can have multiple error events happening.

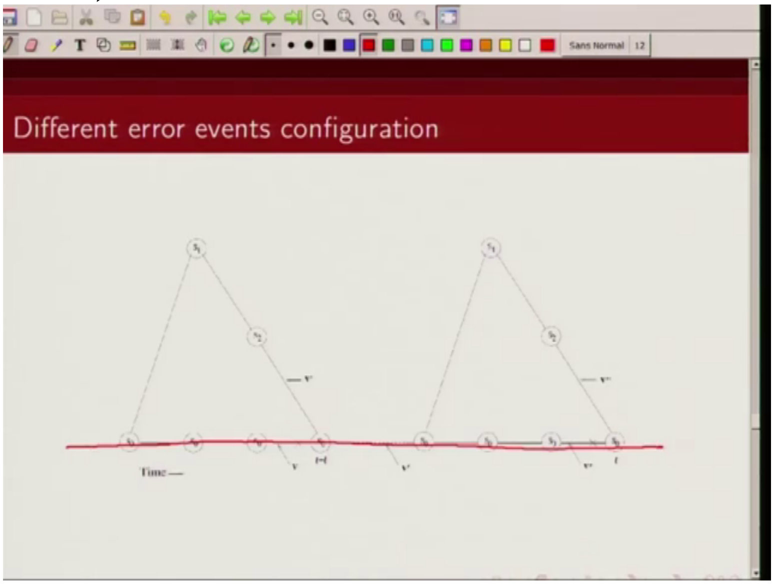
Now

(Refer Slide Time 23:54)



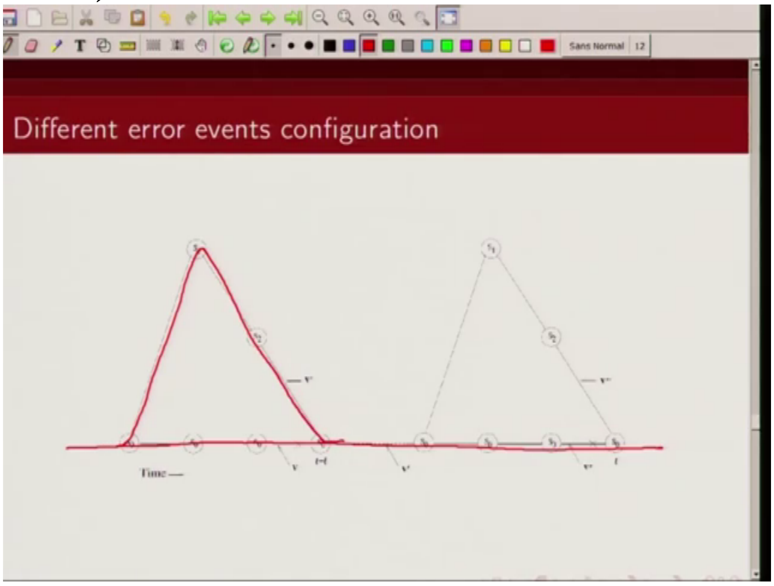
let's look at what are the various types of error configurations you can have. So for example you could have a scenario like this. So

(Refer Slide Time 24:04)



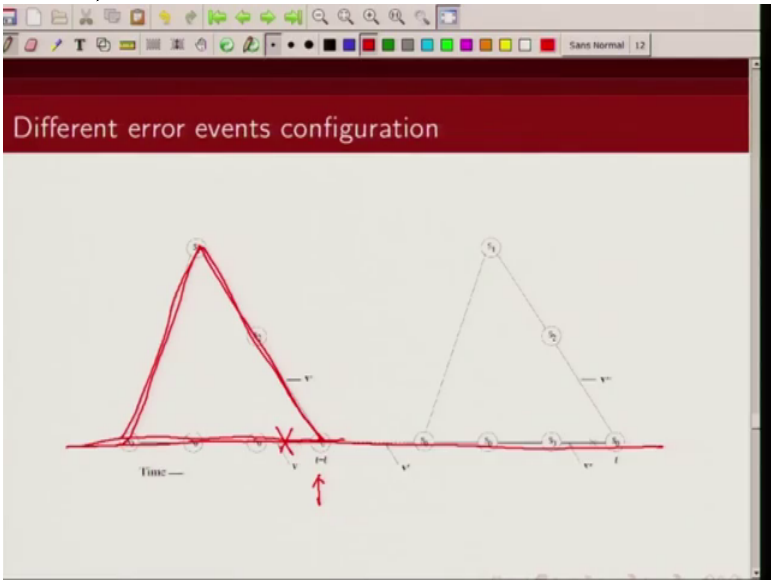
you diverge from all zero state and then you merge back

(Refer Slide Time 24:10)



and at this time instance let's say your metric corresponding to this path was better than metric corresponding to all zero state path. So you have discarded all zero

(Refer Slide Time 24:26)

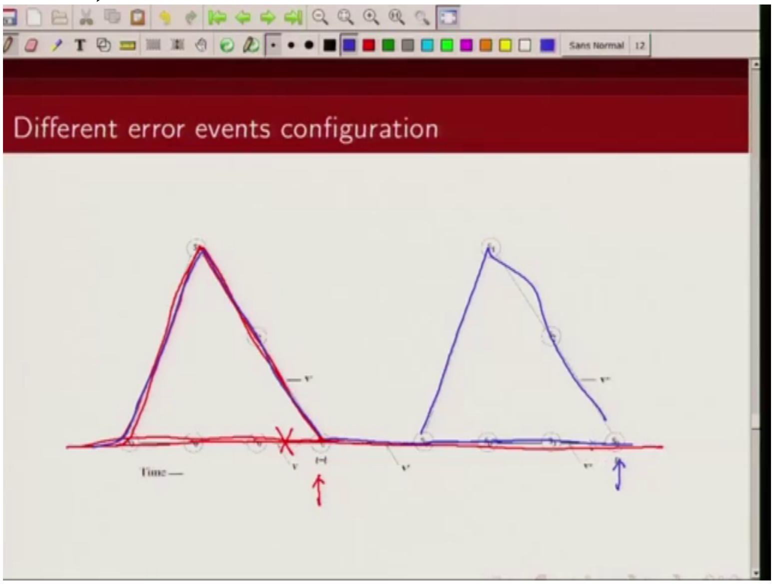


state path and

(Refer Slide Time 24:28)

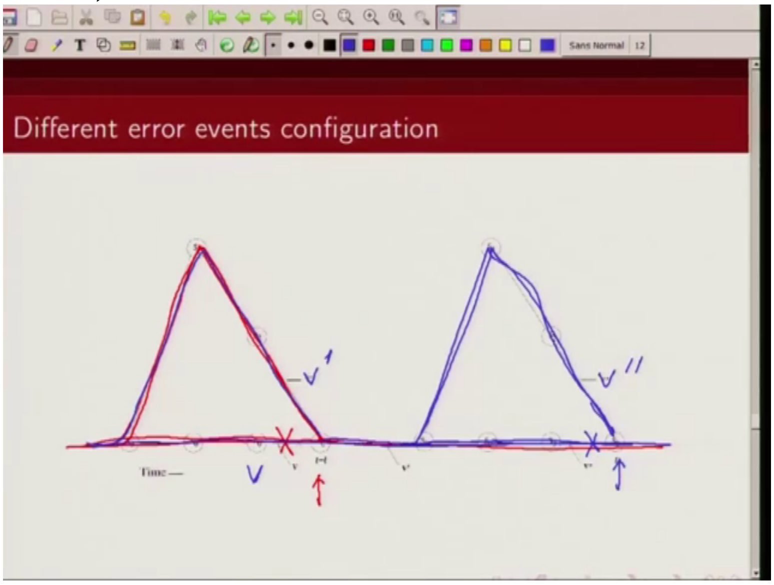
decided in favor of this path. Now further up when you came here you are here and there was one path which was going like this and when you came to this time instance t then again

(Refer Slide Time 24:45)



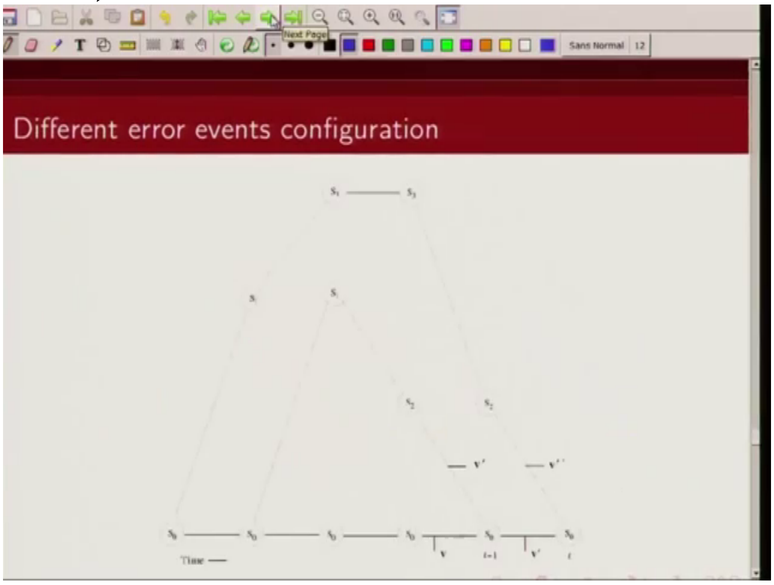
the metric corresponding to this was worse than the metric value corresponding to this path, so you decided to go for this particular codeword just calling this all zero state as vector v, I mean codeword v, this is codeword v dash and this is codeword v double dash.

(Refer Slide Time 25:07)



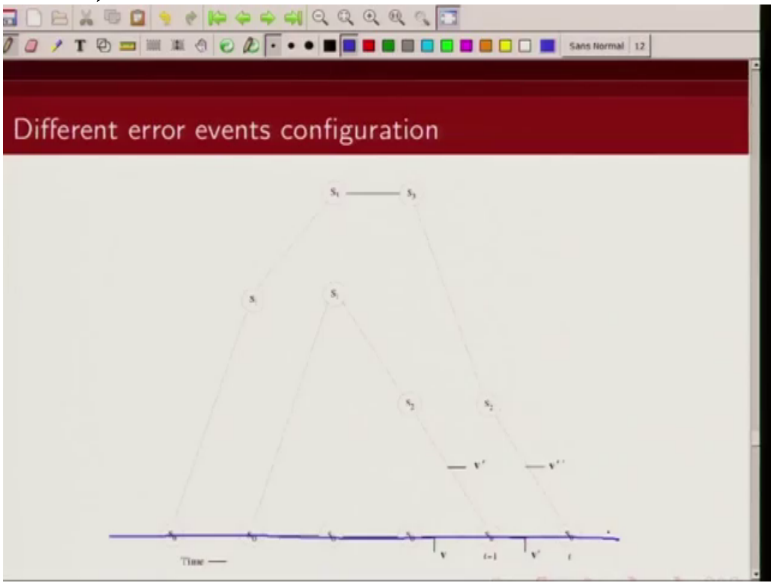
So the point which I am trying to make is whenever you have these kinds of multiple error events you can always upper bound your event probability by error probability of the first event error. So when you decide in favor of this particular codeword; that means this has better metric corresponding to our v, right. So we can upper bound the probability of event error by first event error probability and

(Refer Slide Time 25:45)



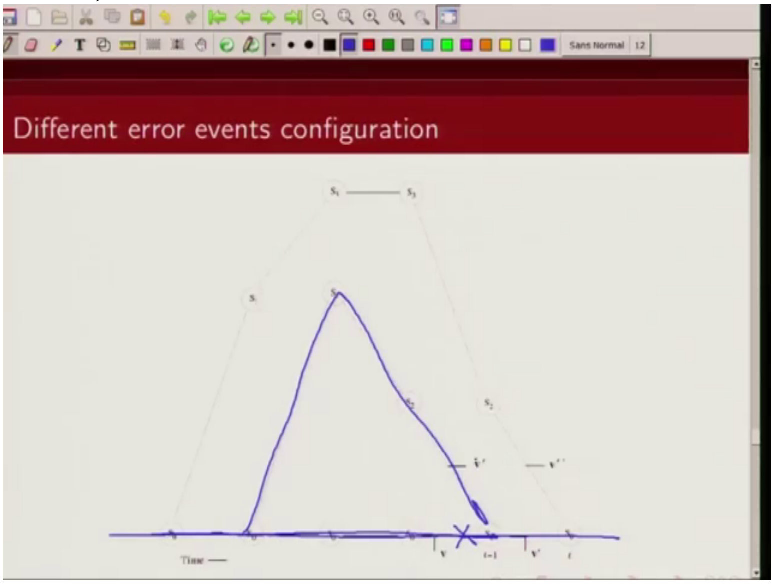
this could have multiple configuration. For example this is your all zero state. So you are here,

(Refer Slide Time 25:51)



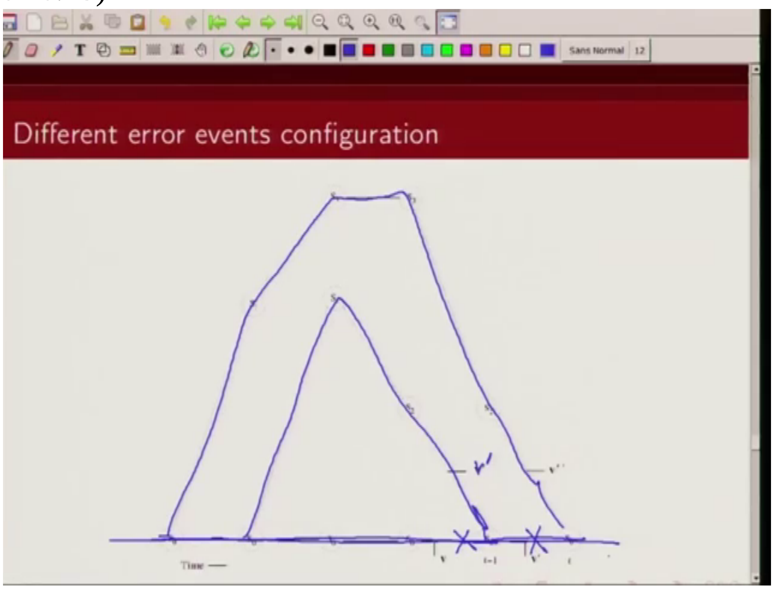
all zero state. When you came to this point you noticed that this metric was better than this metric so you discarded this and you decided in favor of this particular codeword which is your

(Refer Slide Time 26:04)



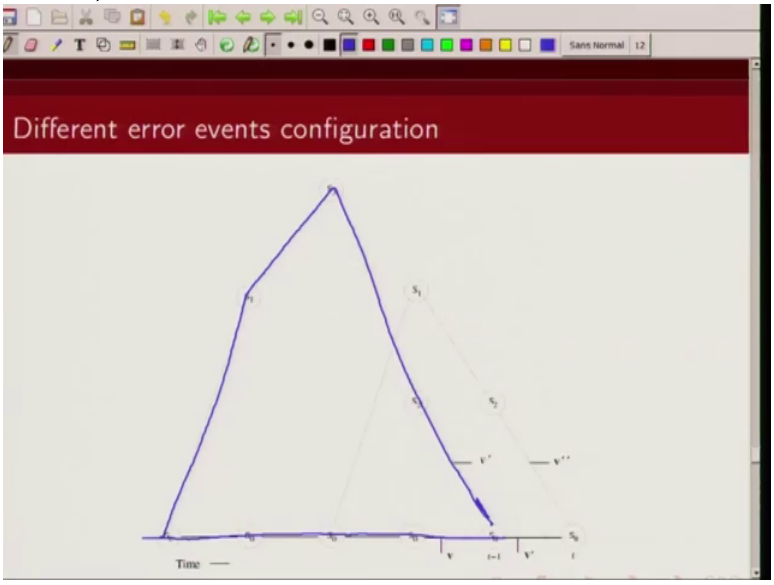
codeword v dash but when you came here again this metric was better than this, so you discarded this and you decided in favor of this. So this is another way

(Refer Slide Time 26:15)



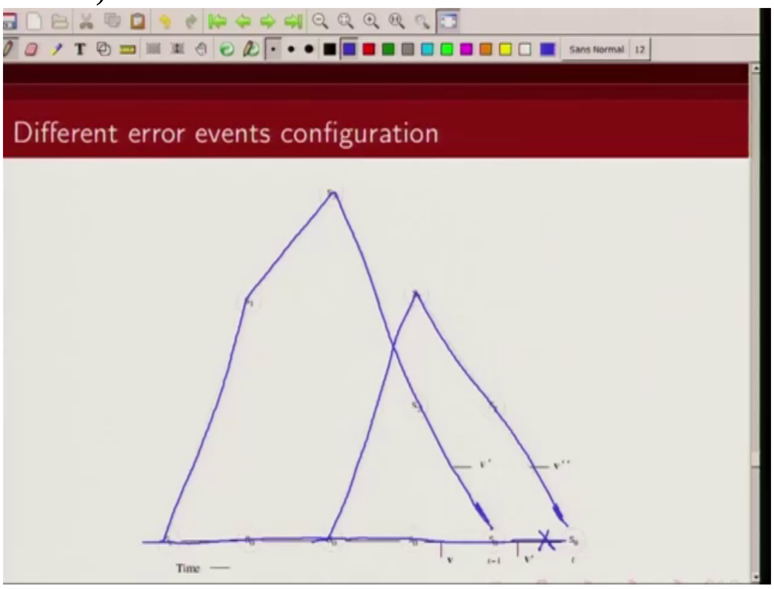
multiple error events can happen. Or you could have a situation like this. You are at all zero state. When you are at this particular instance you notice this has a better metric. So you decide in favor of this incorrect path

(Refer Slide Time 26:31)



but then when you reached here, you noticed this path has a better metric so you discarded this and then you decided in favor of this. Now for each

(Refer Slide Time 26:42)

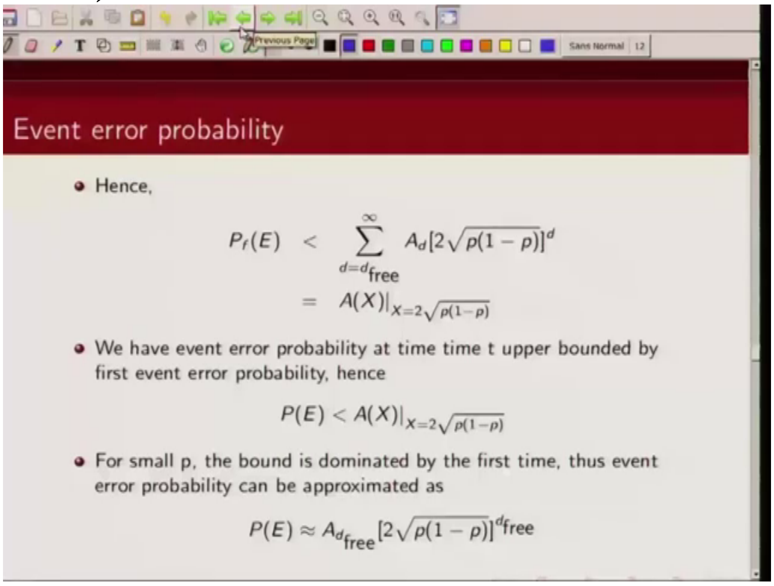


of these cases

(Refer Slide Time 26:44)

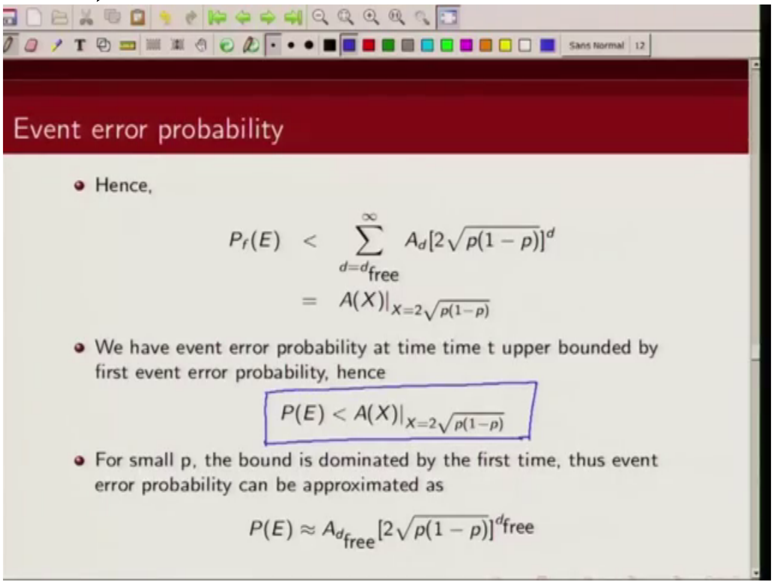
we can upper bound the event probability by the first

(Refer Slide Time 26:48)



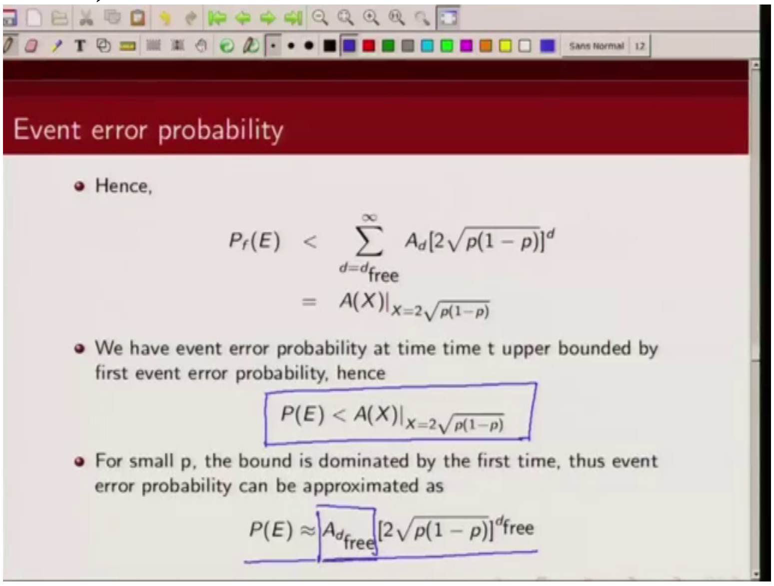
event error probability which we have already computed. So this event error probability can be upper bounded by the first event error probability so we can write a probability of error

(Refer Slide Time 27:06)



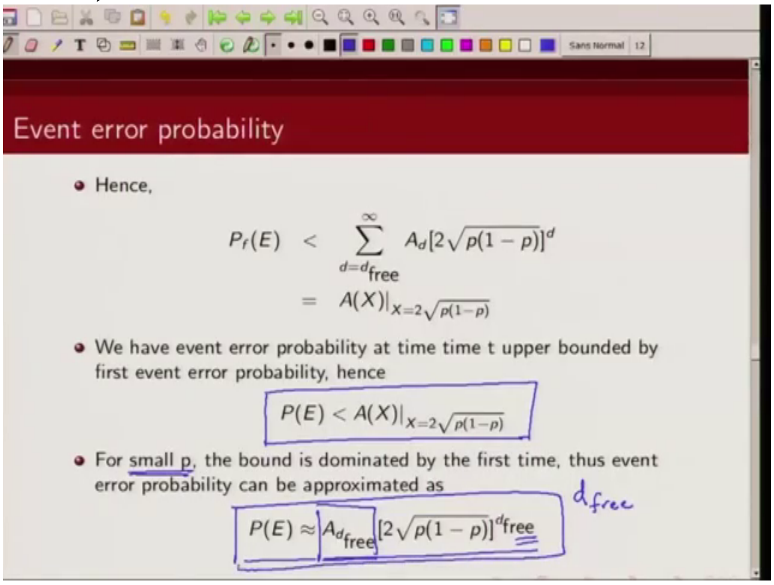
in this particular fashion. Now note here you had terms corresponding to various these, so you had terms corresponding to d free, d free plus 1, d free plus 2, so since the value of p is typically very small, the most dominating term in this error probability expression is your first term which is the d free term. So you can also approximate your probability of event error by how many such events are there which have

(Refer Slide Time 27:51)



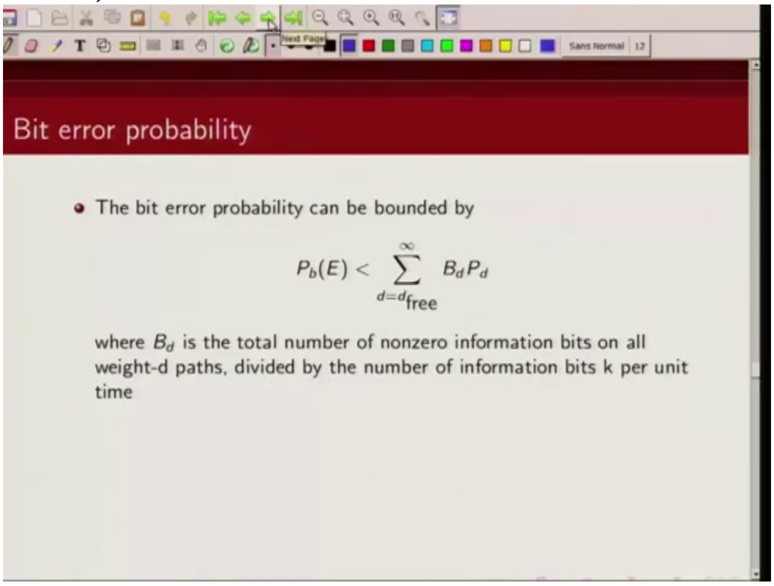
paths which have weight d free and this raised to power d free. So case when p is very small, this bound is dominated by the first term which is the term corresponding to d free. And in that case we can write down an approximate expression for

(Refer Slide Time 28:16)



probability of error like this.

(Refer Slide Time 28:22)

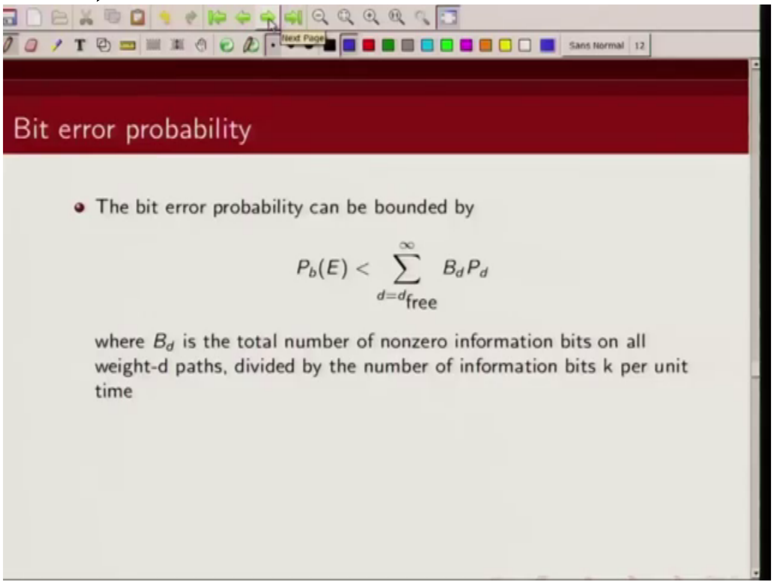


Now can we modify this expression

(Refer Slide Time 28:26)

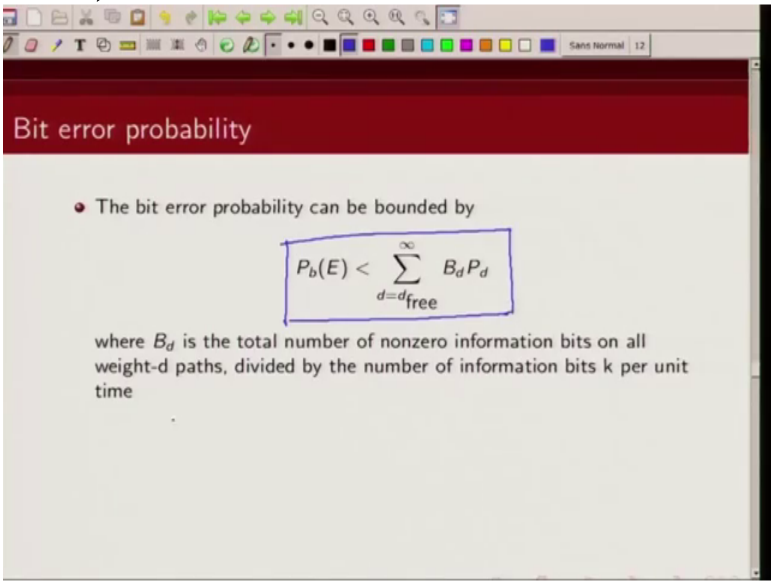
to calculate bit error rate probability? The answer is yes. So you know what is the weight corresponding to an incorrect path. Now if you can find out what is the information weight corresponding to these incorrect path and then

(Refer Slide Time 28:48)



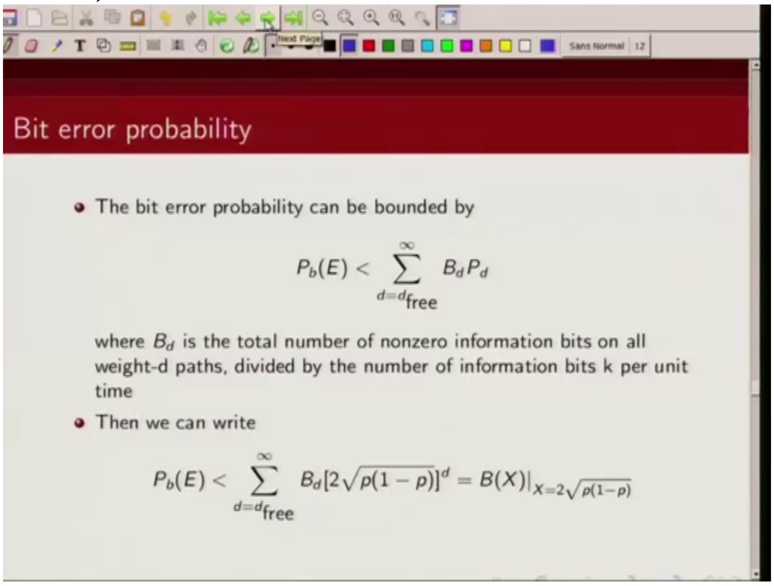
we divide it by total number of information bits, we can find out what is the bit error rate probability. So bit error rate probability can be computed from this first event error probability. This is given by this expression where

(Refer Slide Time 29:06)



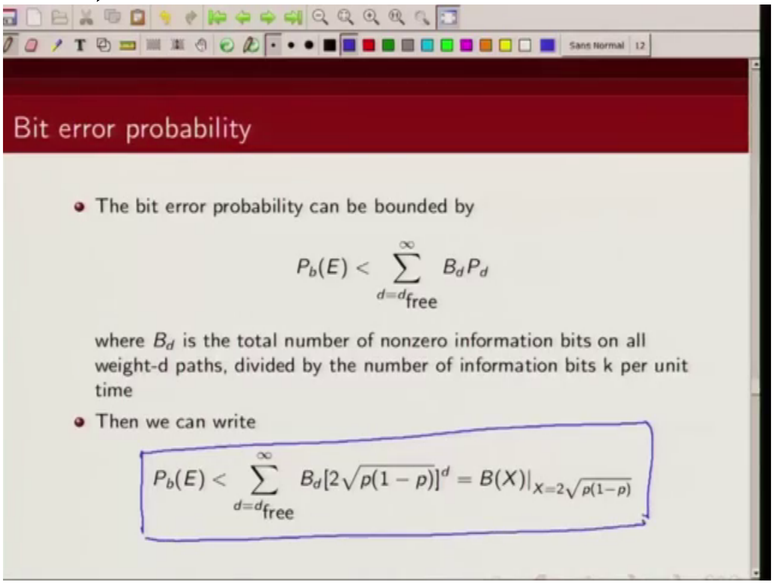
B d is the total number of non zero information bits on all these weight d incorrect paths divided by k information bits per unit time.

(Refer Slide Time 29:25)



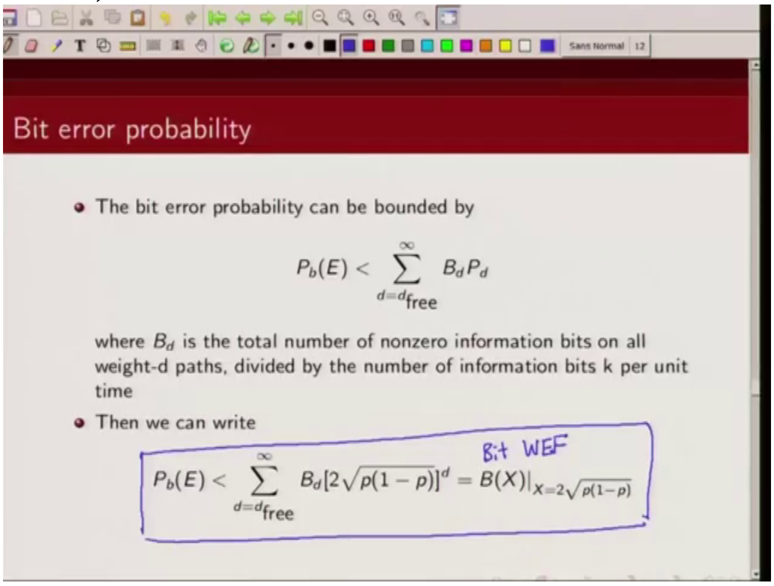
So if we plug that value of bit error rate probability we can see that,

(Refer Slide Time 29:35)



we can similarly write the expression for bit error rate probability; it is the bit, this is bit weight enumerating function and in a minute we will talk about

(Refer Slide Time 29:50)

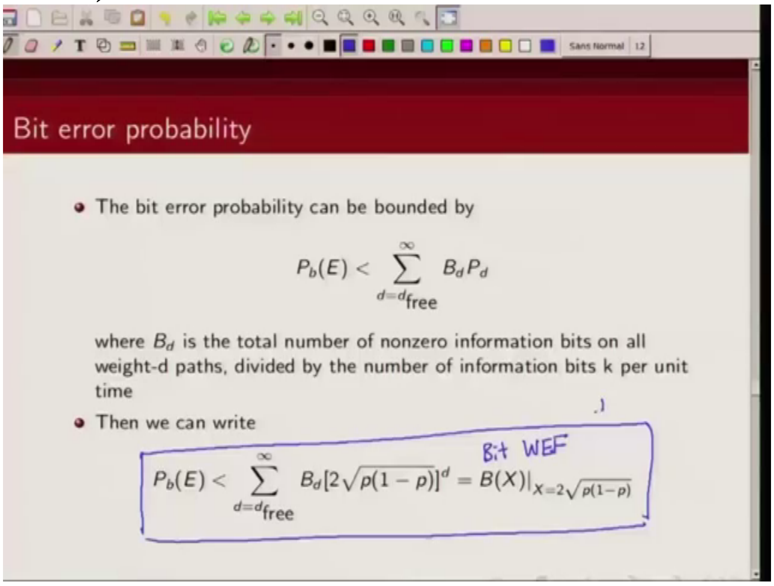


how to

(Refer Slide Time 29:51)

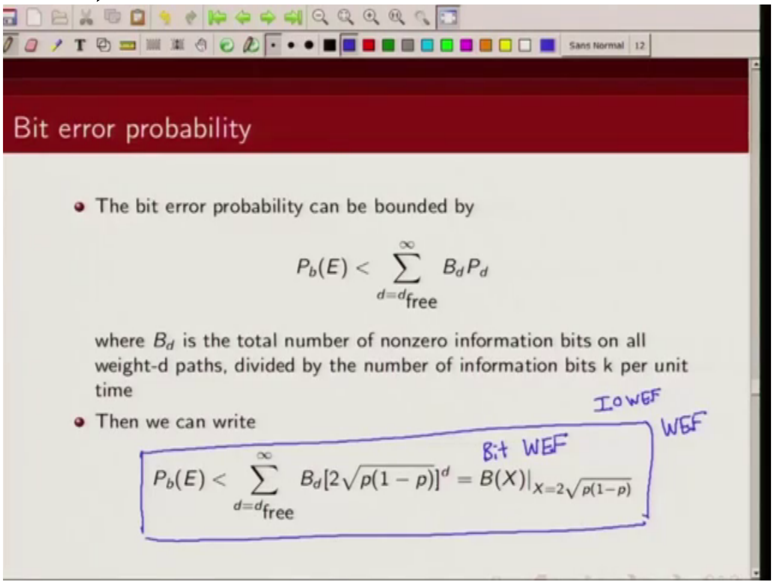
generate bit weight enumerating function from

(Refer Slide Time 29:55)



input output weight enumerating function or from weight enumerating function. We will talk about this, so

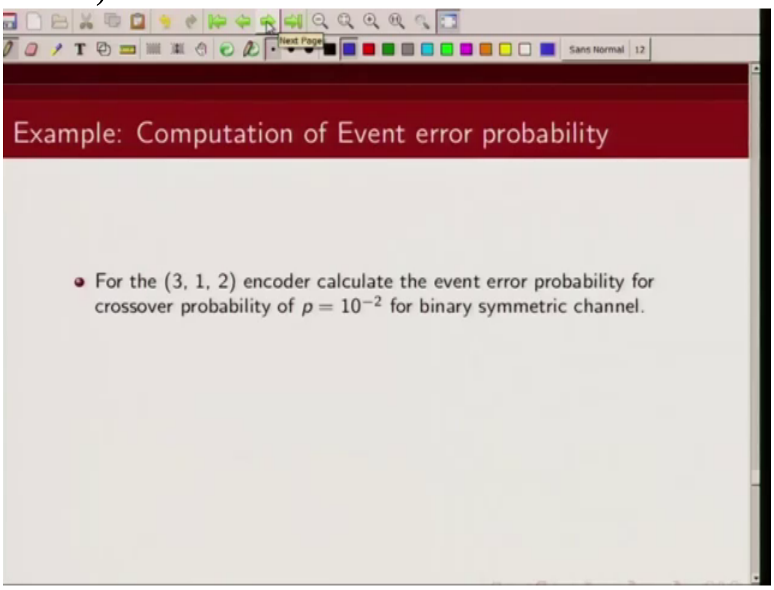
(Refer Slide Time 30:01)



this can be computed from bit weight enumerating function by substituting x equal to this quantity, Ok.

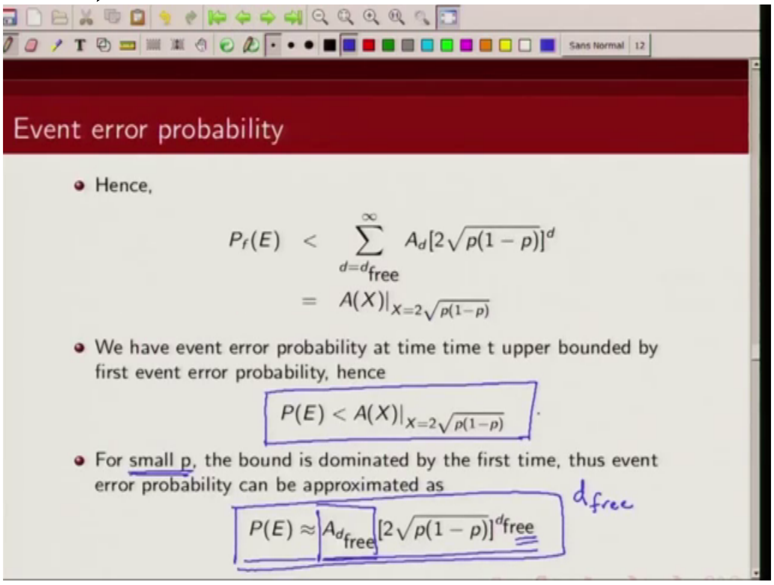
Let's just take an example to illustrate this. So let us

(Refer Slide Time 30:17)



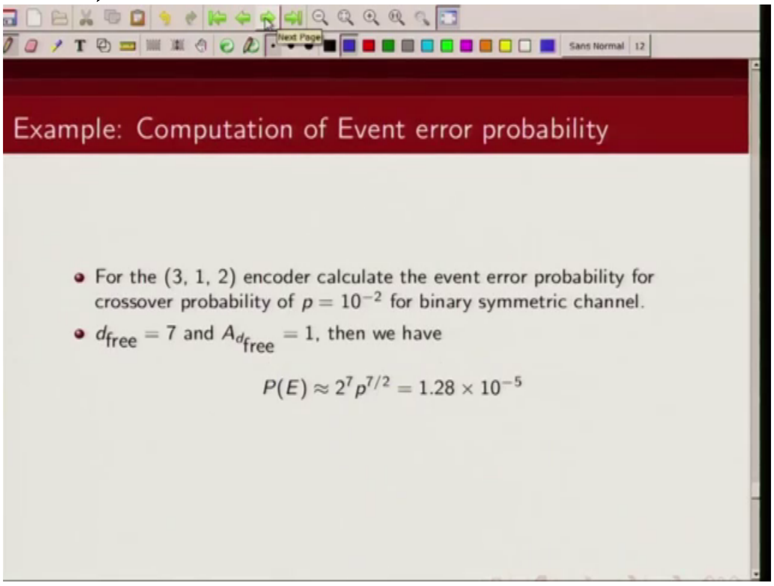
consider the same encoder that we have considered. It's a 3 1 2 convolutional encoder, it's a feed forward encoder and what is given to us is the crossover probability is point 0 1. You are asked to compute what is the event error probability. Now we know the expression of event error probability, the bound on event error probability that's given by this

(Refer Slide Time 30:46)



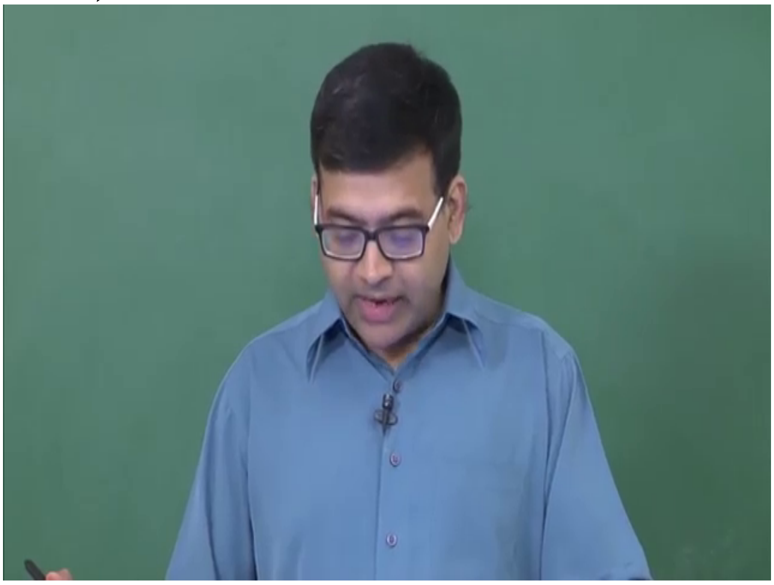
expression. And in this particular example p is very small. p is point 0 1. So we can approximate, we can get the approximate expression of event error probability from this. Now what is the free distance of this convolutional code? Again if you go back, the minimum weight sequence was 7. So d free for this particular example was 7.

(Refer Slide Time 31:15)



So free distance was 7. And there was only one such sequence of weight 7. So we plugged those values of d free and a d free in the expression for probability of event error and we get probability of event error to be roughly 1 point 2

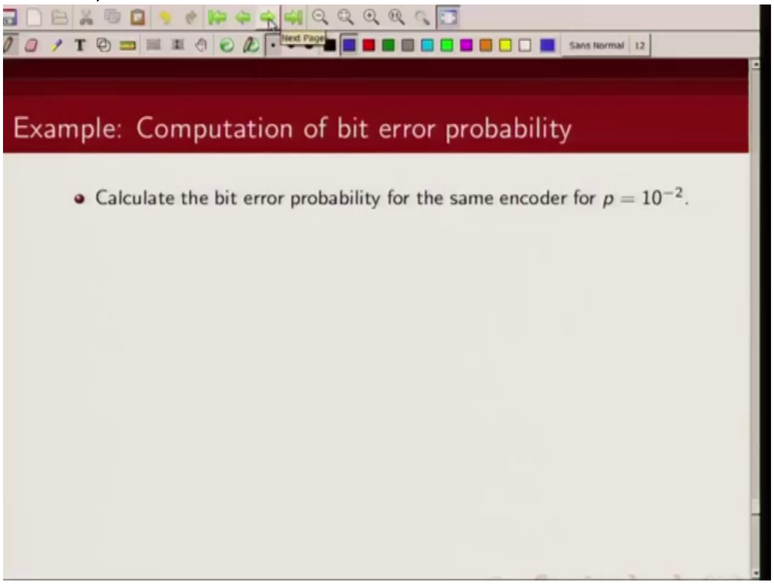
(Refer Slide Time 31:35)



2 times into 10 to power minus 5.

Now how do we

(Refer Slide Time 31:39)

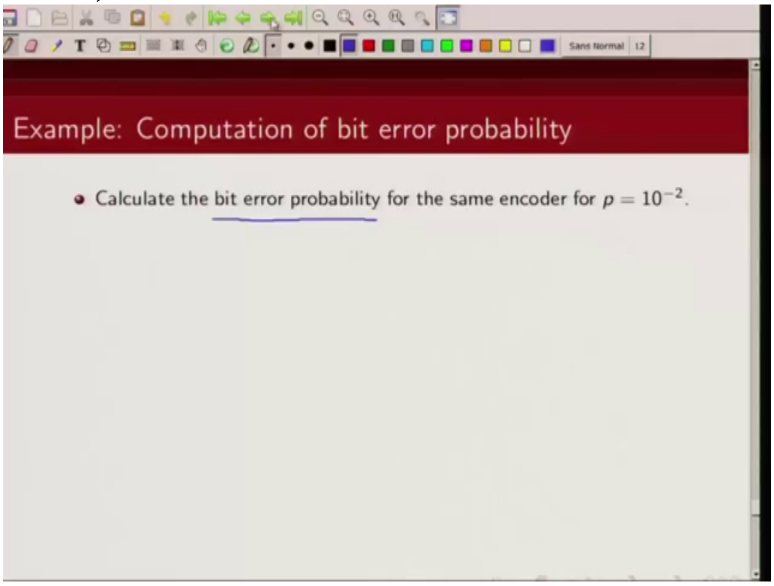


compute the bit error probability? So first we have to generate bit weight

(Refer Slide Time 31:45)

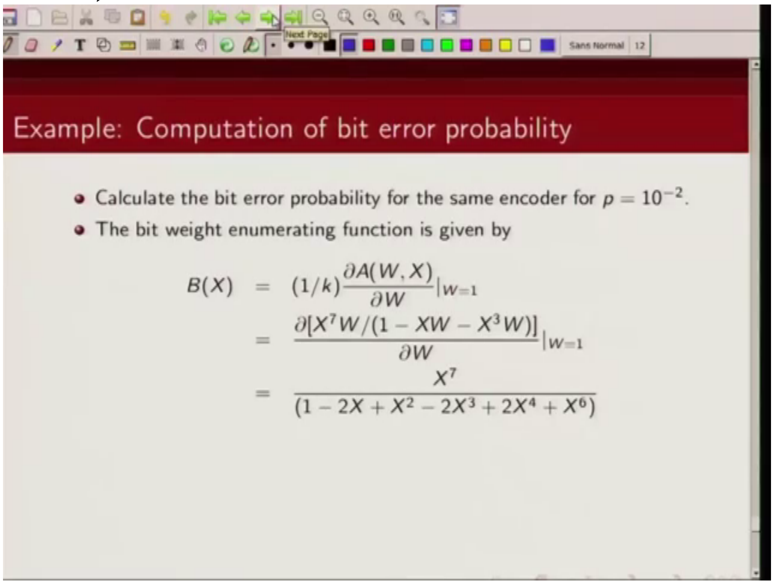
enumerating function. We know how to compute weight enumerating function. It has been explained in lecture 2 c.

(Refer Slide Time 31:54)



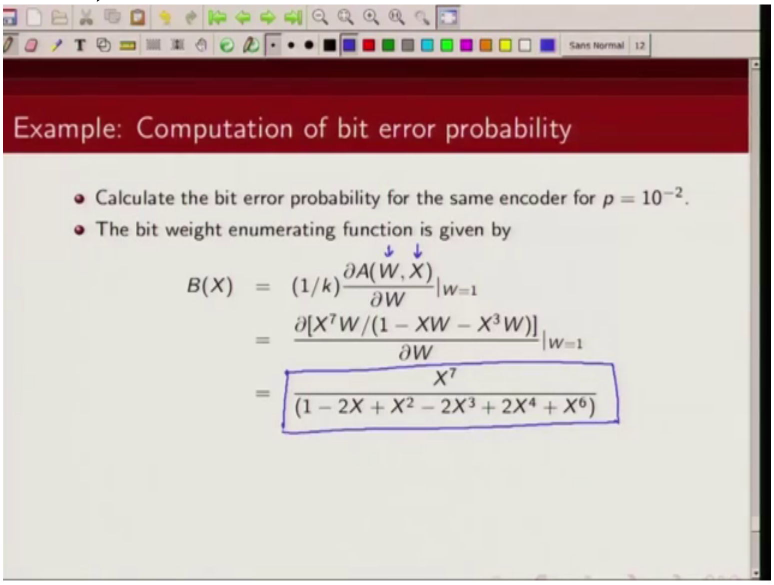
So assuming

(Refer Slide Time 31:55)



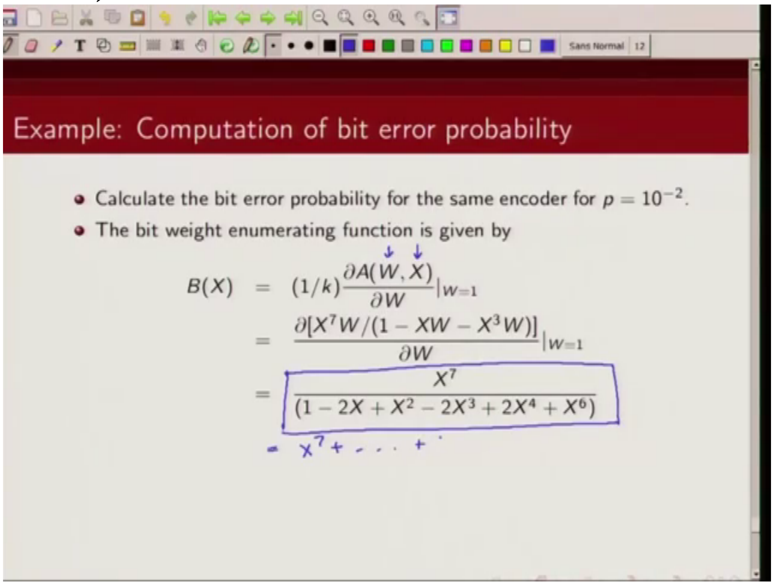
you have the weight enumerating function which tells you and you have the input output weight enumerating function which will tell you what information bit causes or what is the corresponding output weight so you could compute bit weight enumerating function by partial derivative of your input output weight enumerating function with respect to w and putting w to be equal to be 1 and 1 by k times that. If you do that you will get the expression for bit weight enumerating function. So for the example that we have considered we already had the expression for input output weight enumerating function. So if we plug that in we get the expression for bit weight enumerating

(Refer Slide Time 32:49)



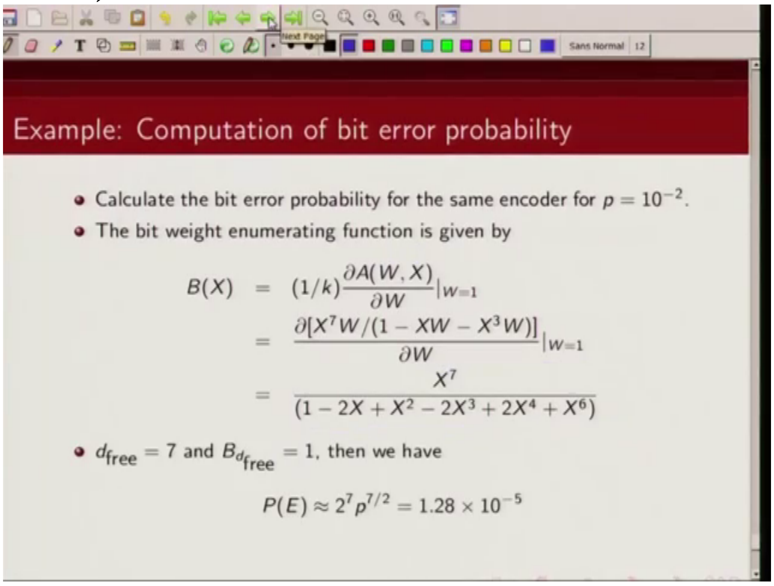
function like this and then we can divide this by this. So we will get x 7 plus some some terms like that.

(Refer Slide Time 32:57)



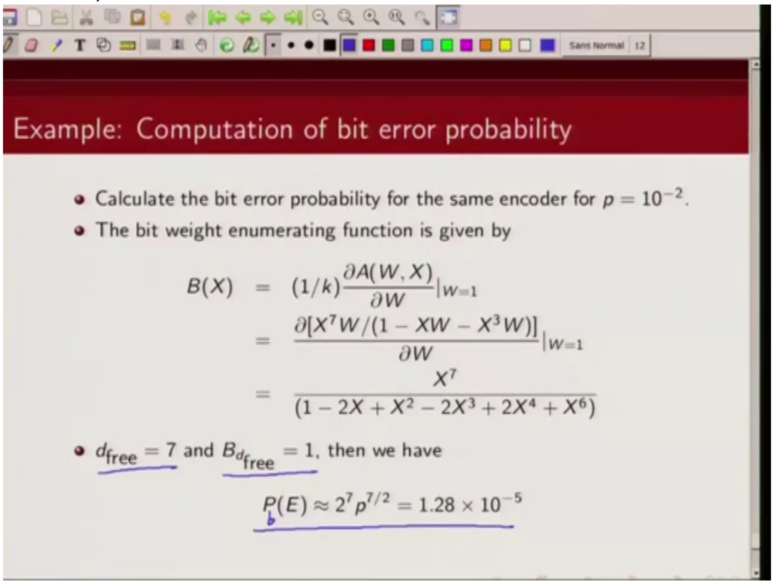
So in this case also, what is d free? d free is 7 and B d free is also 1.

(Refer Slide Time 33:04)



So this quantity is 1, d free is 7 so we can approximate the expression because the first term corresponding to the free distance will be the dominating term. So we can write the probability of bit error

(Refer Slide Time 33:20)



to be this, Ok. So with this I will conclude this discussion on

(Refer Slide Time 33:29)

performance analysis of convolutional code over binary symmetric channel. Now similar analysis can be done for other channels as well, thank you