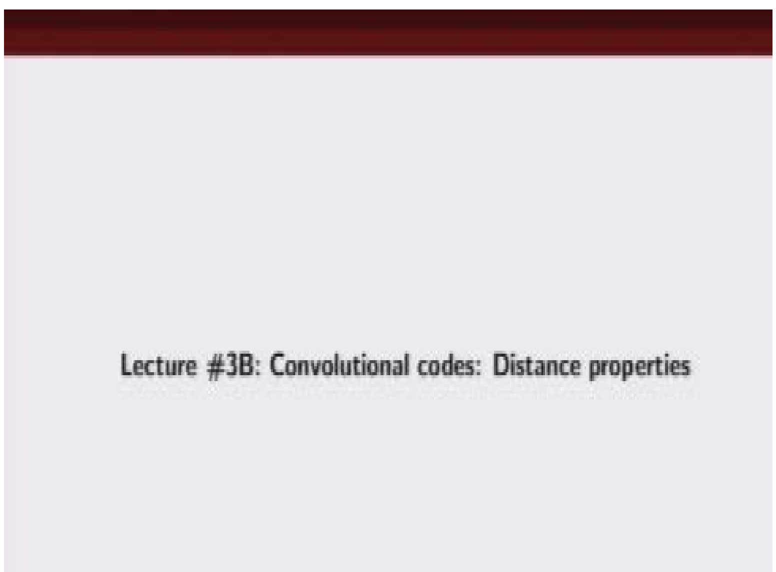
Indian Institute of Technology Kanpur
National Programme on Technology Enhanced Learning (NPTEL)
Course Title
Error Control Coding: An Introduction to Convolutional Codes

Lecture – 3 B
Convolutional Codes: Distance Properties

by
Prof. Adrish Banerjee
Dept. Electrical Engineering, IIT Kanpur

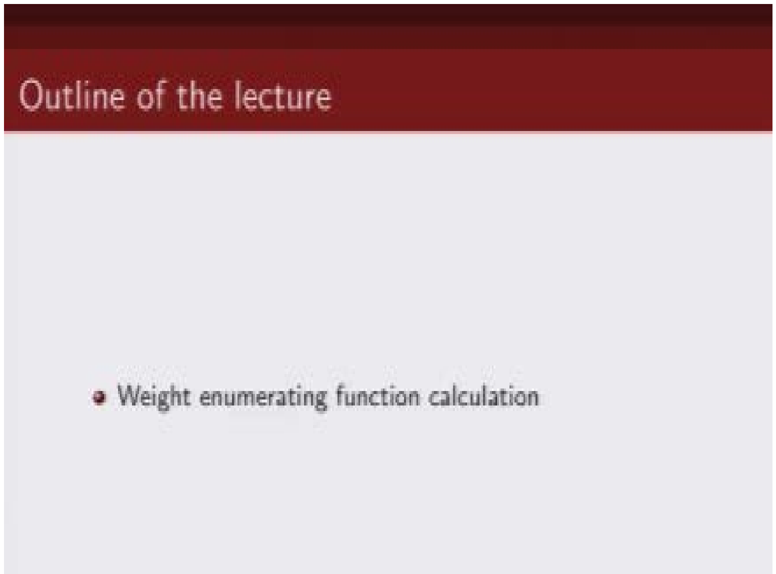
Welcome to the course on Error Control Coding, an introduction to Convolutional Codes.

(Refer Slide Time: 00:20)



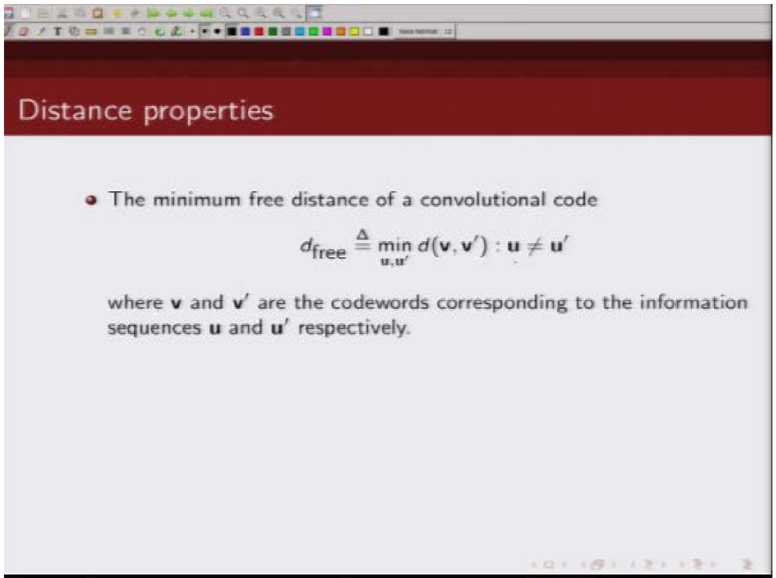
So in this lecture we are going to talk about how we can find out the way distribution of a convolutional codes and we are going to discuss about the distance properties of convolutional code.

(Refer Slide Time: 00:32)



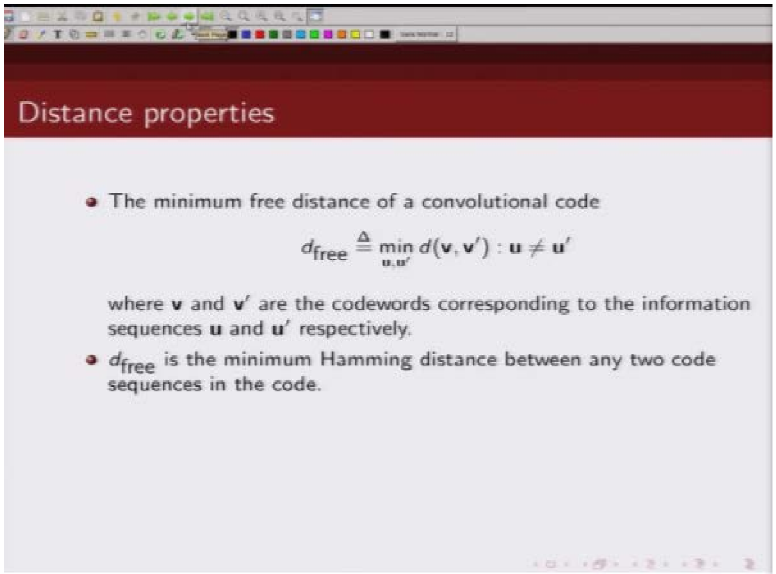
So this lecture deals with how we can enumerate the distance profile or the way distribution of a convolutional code.

(Refer Slide Time: 00:41)



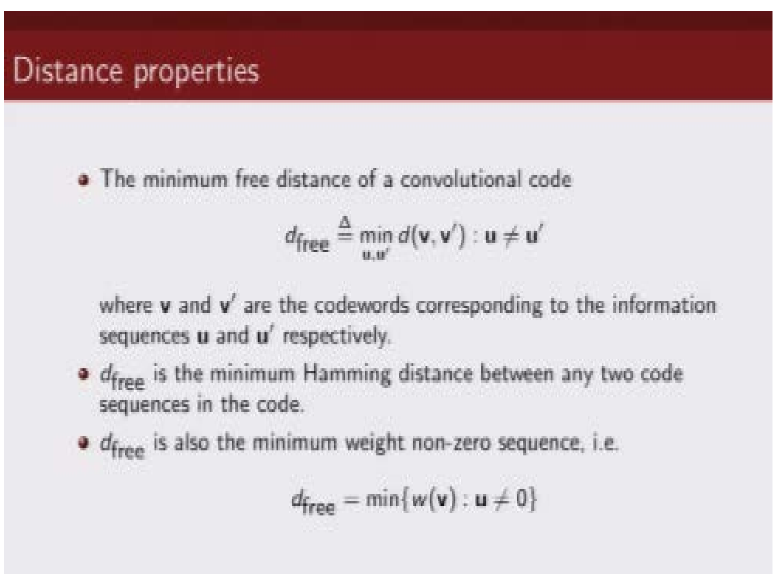
So we can define before we discuss a technique to find out the way distribution let us define what do we mean by a minimum free distance of a convolutional code, so minimum free distance of a convolutional code is defined as, minimum hamming distance between two codes v and v´, where v and v´ are two code words corresponding to information sequence u and u´ where u and u´ are two different information sequences.

(Refer Slide Time: 01:14)



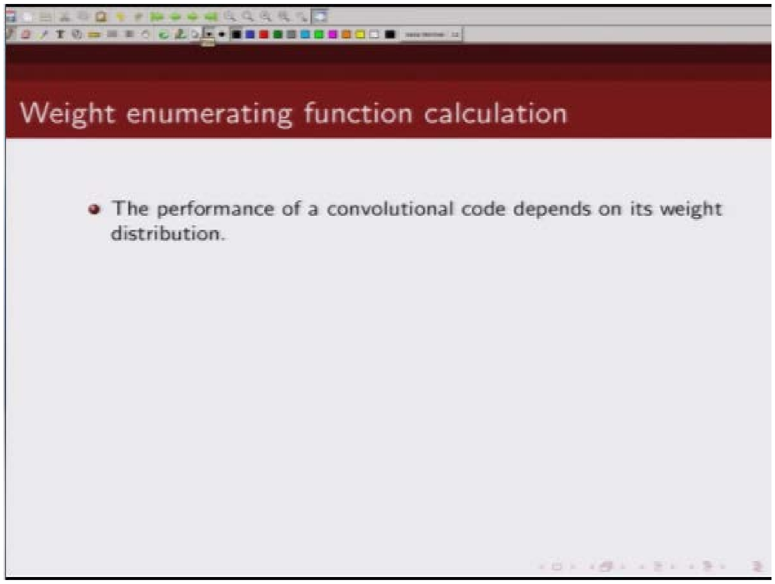
So it is basically free distance is the minimum hamming distance between any two code words in a convolutional code.

(Refer Slide Time: 01:26)



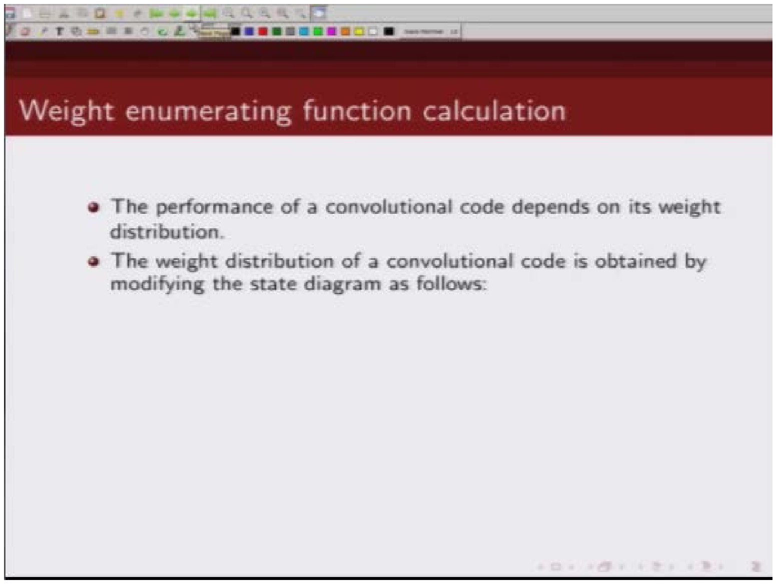
And this is same as minimum weight non-zero, so it is a minimum weight of a non-zero code word.

(Refer Slide Time: 01:37)



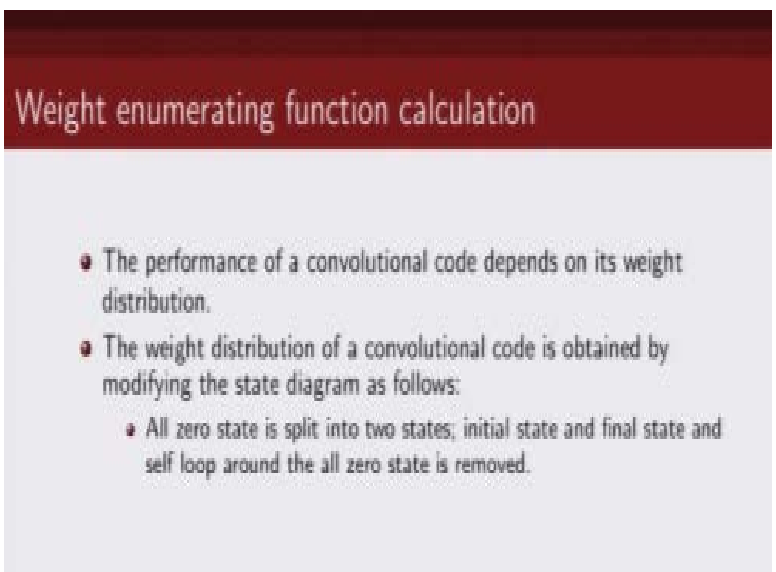
We know that performance of any convolutional code depends on its weight distribution and that is why we are interested to find out what is a weight distribution of a convolutional code. In this lecture we are going to talk about a method based on Mason's gain formula to compute a weight distribution for convolutional code.

(Refer Slide Time: 02:01)



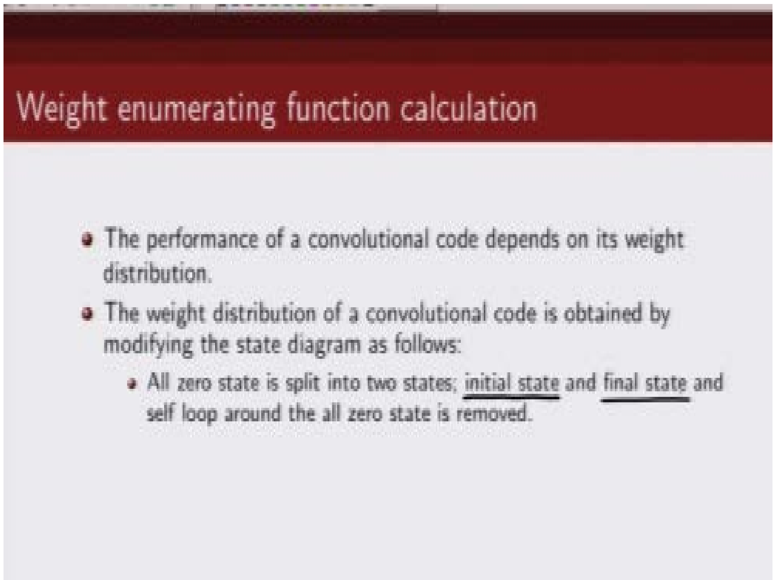
So in this we are first going to modify this state diagram of a convolutional code.

(Refer Slide Time: 02:10)



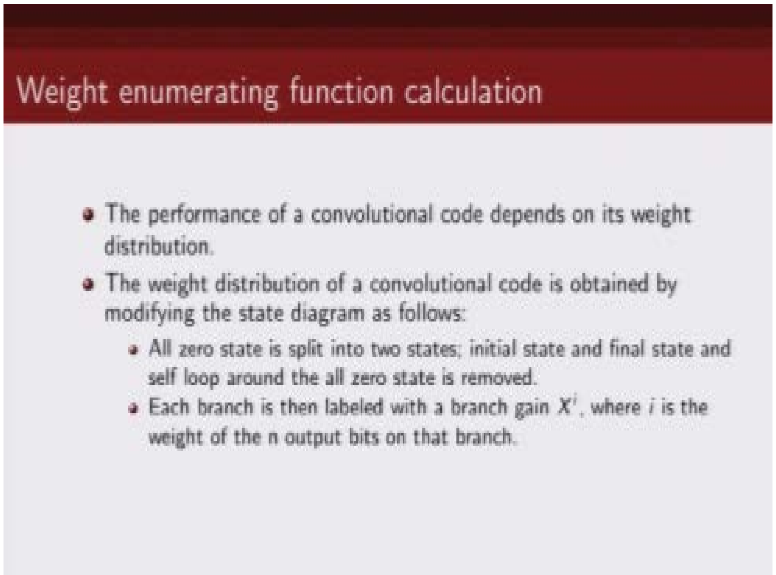
And how are we going to modify this state diagram? We are going to split the all zero state into two state.

(Refer Slide Time: 02:19)



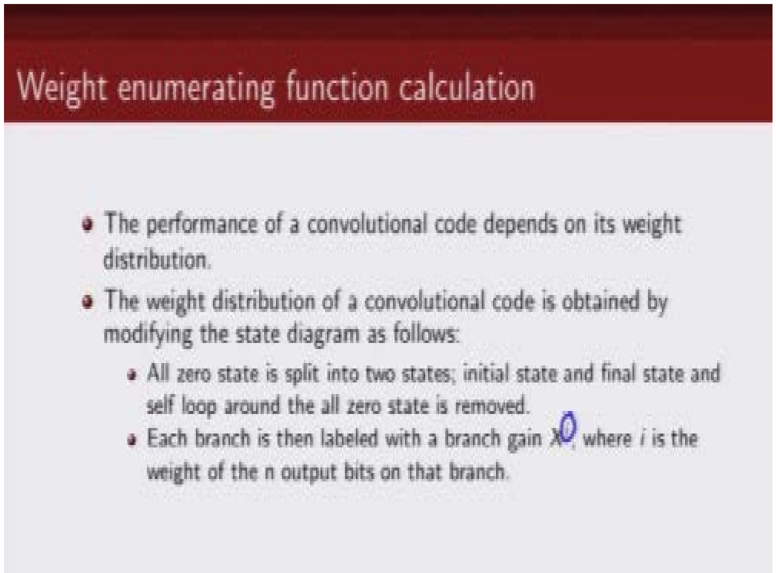
One initial state and second final state and we are going to remove the self loop around the all zero state.

(Refer Slide Time: 02:35)



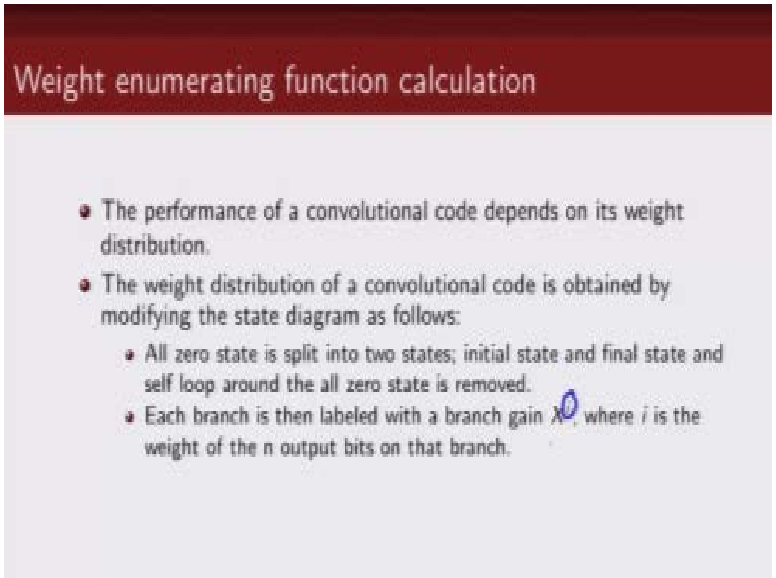
Next each branch which is linking from one state to another state this branch will be labeled by the output weight of the code words so we are going to denote by x^i, where I will be the weight of the coded bits.

(Refer Slide Time: 02:57)



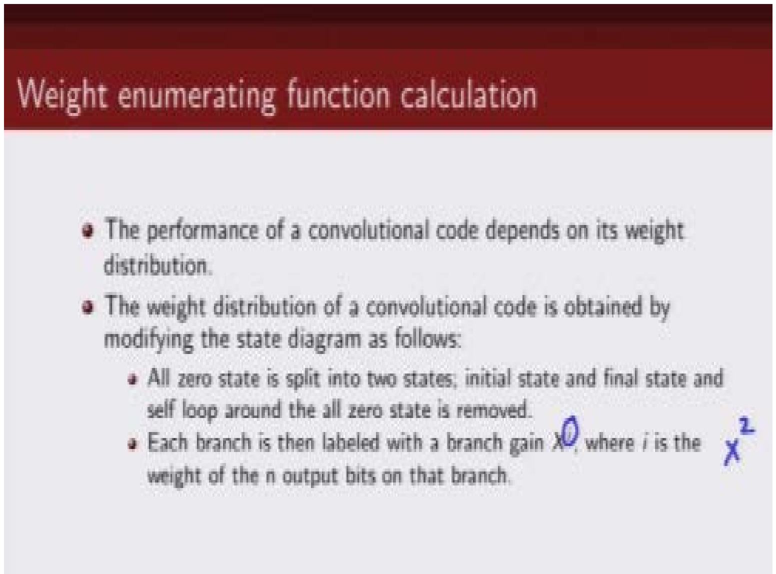
So for example if we make a transition from state 0 to state 01 when an input 1 comes and output is 11 so in that case.

(Refer Slide Time: 03:14)

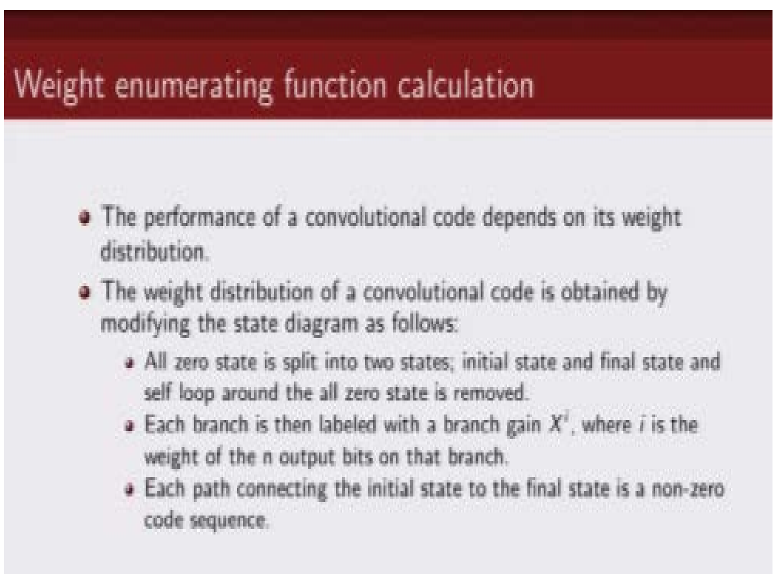


Since the output is 11 we will label that branch by x^2.

(Refer Slide Time: 03:22)

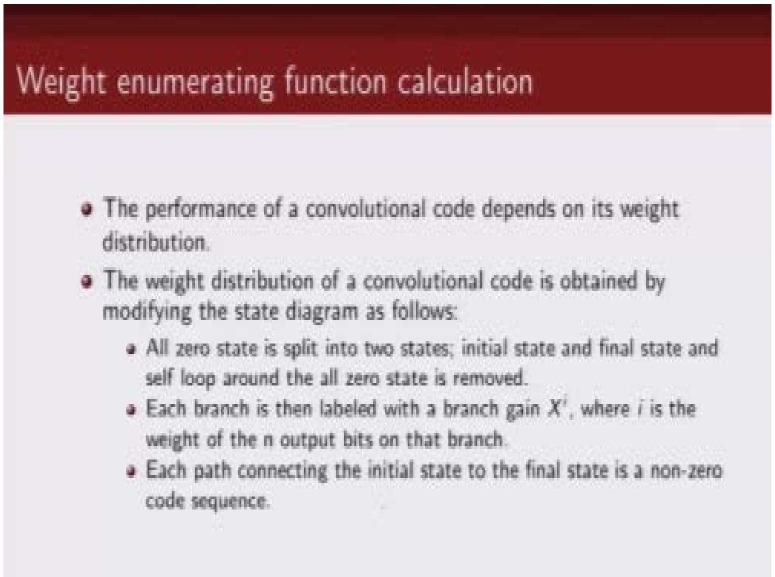


(Refer Slide Time: 03:14)



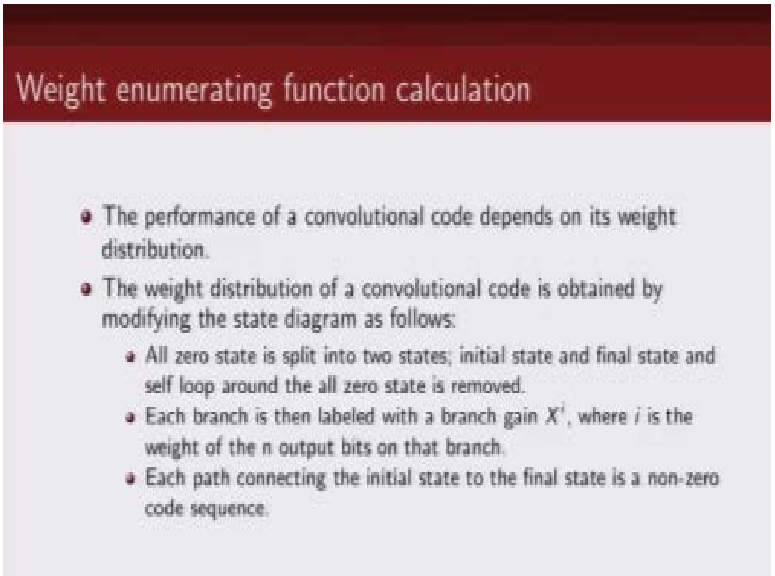
So now after we split this all zero sequence, all zero state into two states initial state and final state what we will have is each path starting from this initial state to the final state that will be our valid code word.

(Refer Slide Time: 03:46)



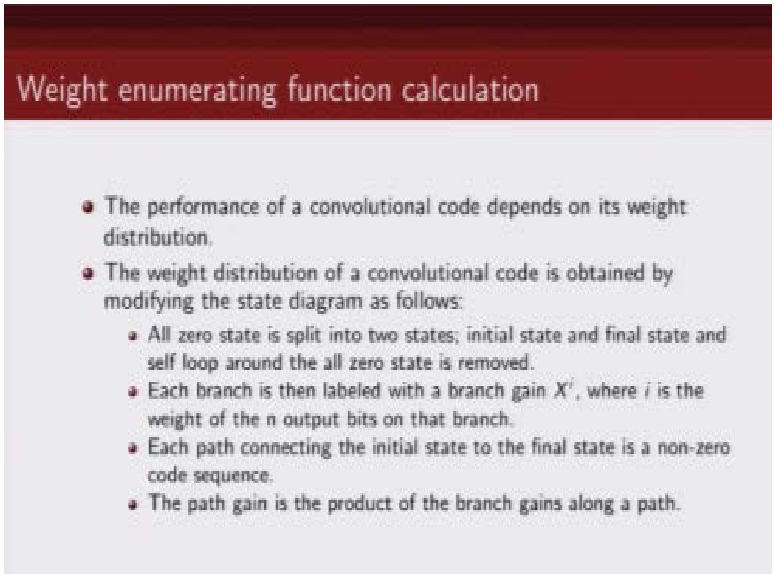
So each path connecting the initial state to the final state is a valid non-zero code word because we have removed the self loop around the all zero state.

(Refer Slide Time: 04:00)



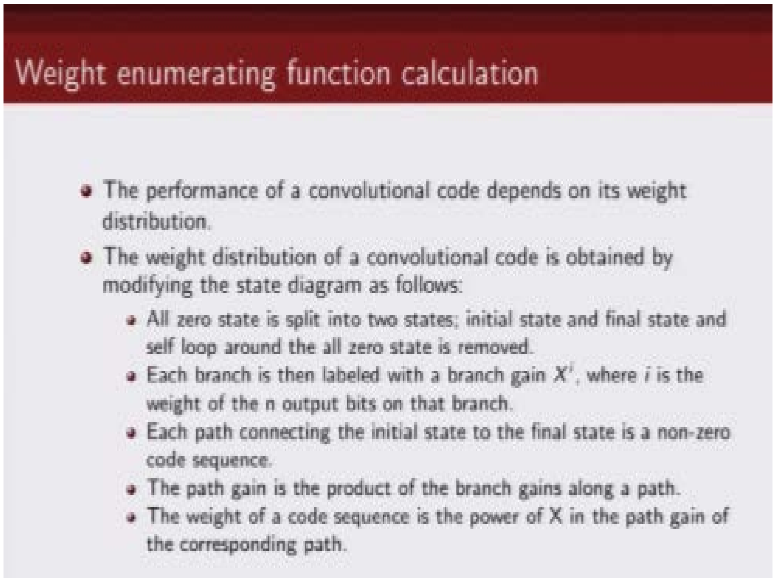
So this modified state diagram is now going to show us all possible non-zero code words.

(Refer Slide Time: 04:13)



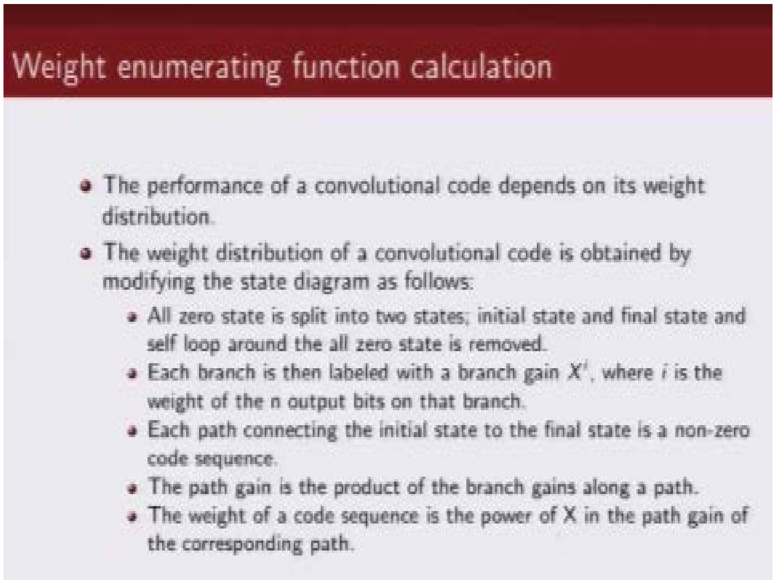
We define a path gain as product of branch gains along a path.

(Refer Slide Time: 04:20)



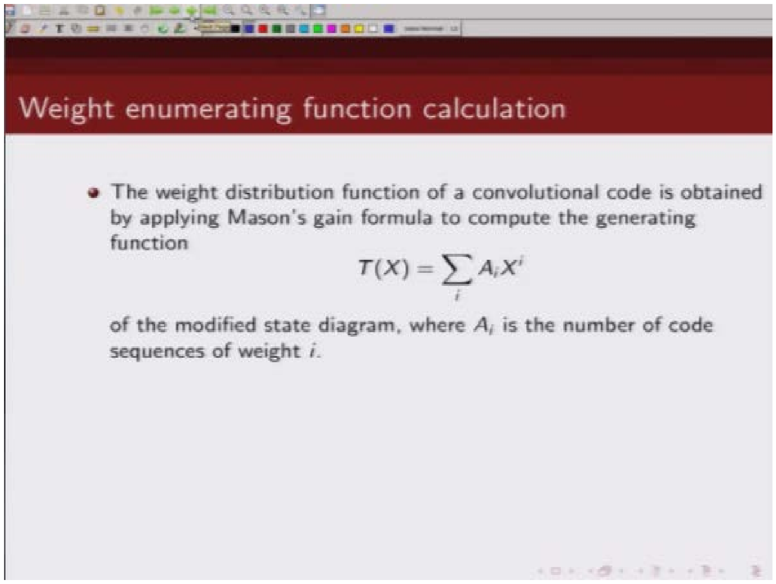
And the weight of a code sequence will be nothing but the power of x, in the path gain of the corresponding path because what we are doing is, so each branch is labeled by its corresponding output weight. So if we look at each path going from the initial state to the final state and we look at the power of x.

(Refer Slide Time: 04:54)

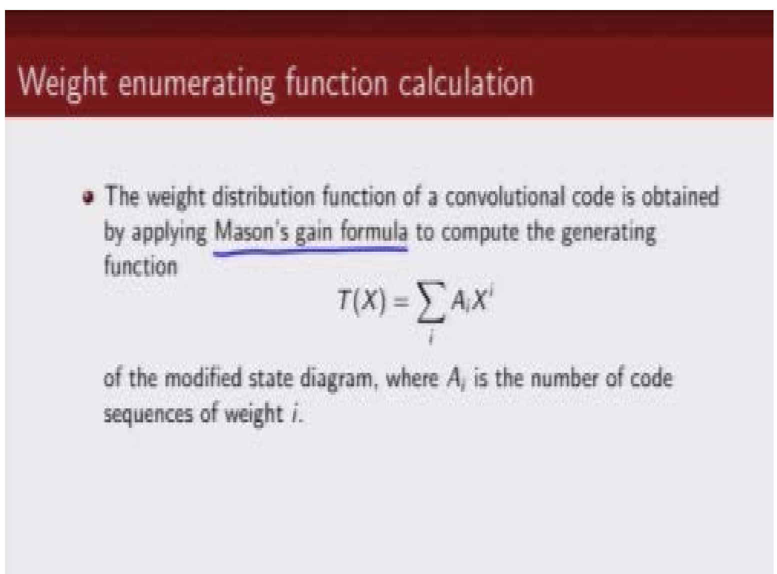


That will give us the overall weight of that particular non-zero code sequence.

(Refer Slide Time: 05:02)

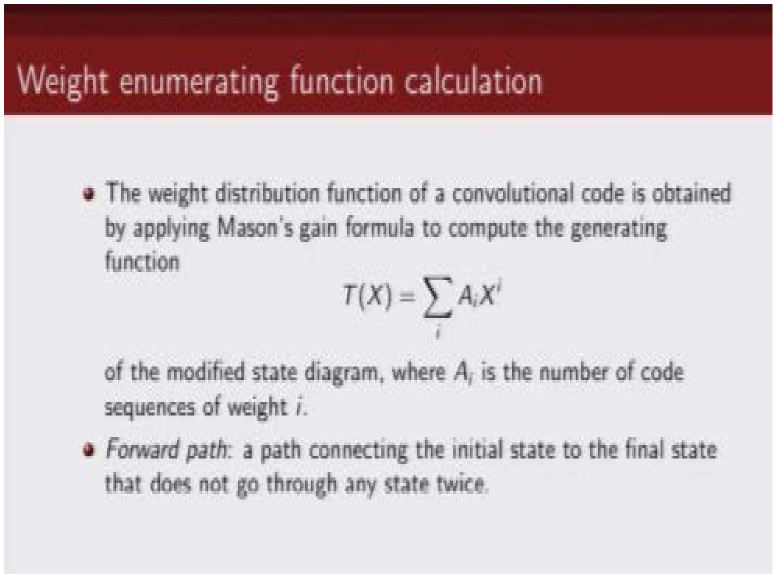


(Refer Slide Time: 05:02)



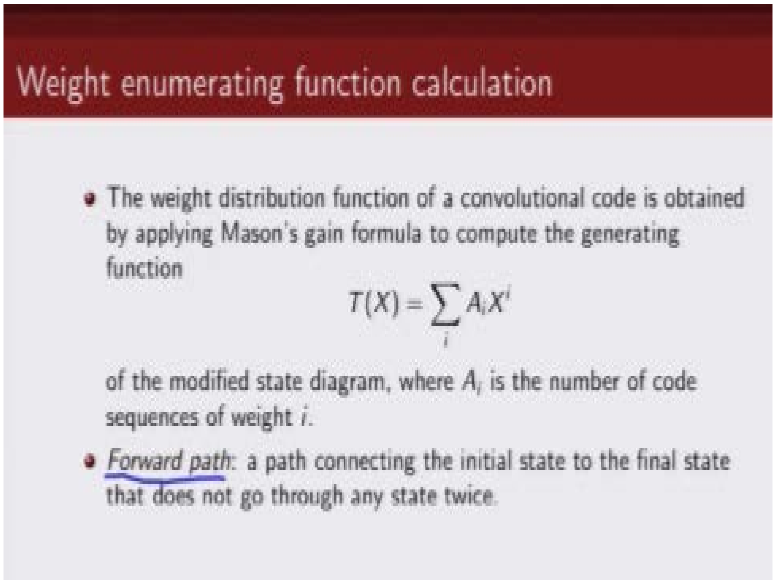
As we said we are going to use Mason's gain formula to compute the weight enumerating function for the convolutional code. So we are going to describe how we are going to use the Mason gain formula. So we are representing by T(X) the generating function which will basically enumerate all code words of weight i.

(Refer Slide Time: 05:38)



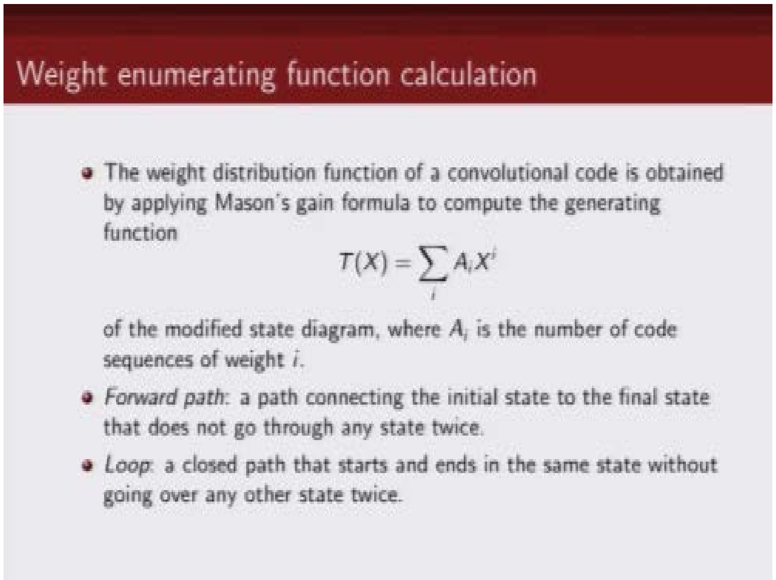
Now let us define few terms that we are going to use in Mason's gain formula, the first term that we are going to define is what is known as forward path.

(Refer Slide Time: 05:48)



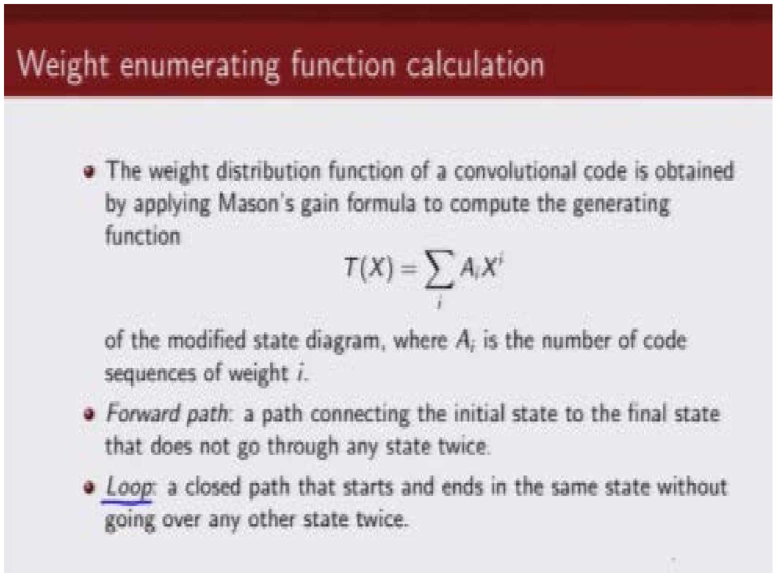
So a forward path is a path from the initial all zero state to the final state and the condition is this path should not go over any state twice that is our forward path.

(Refer Slide Time: 06:11)

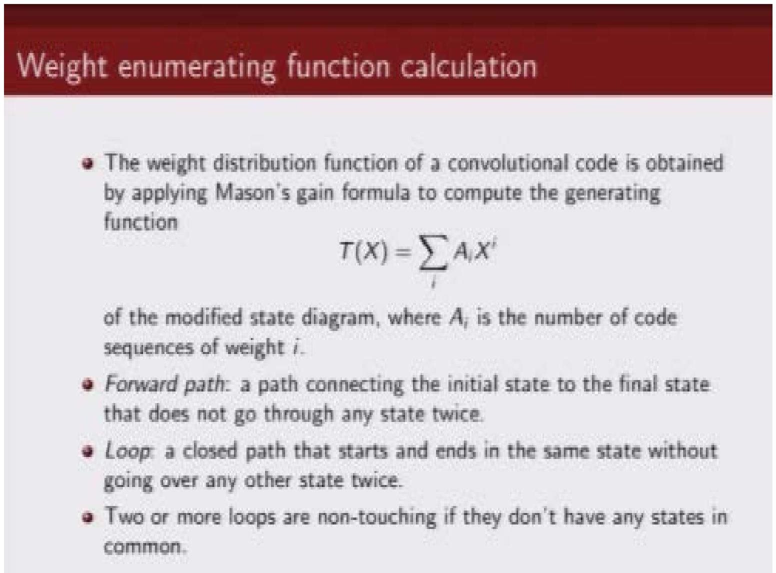


Next term that we define is basically a loop what is a loop? A loop is a close path that starts and ends in the same state without going over any state twice, that is a loop.

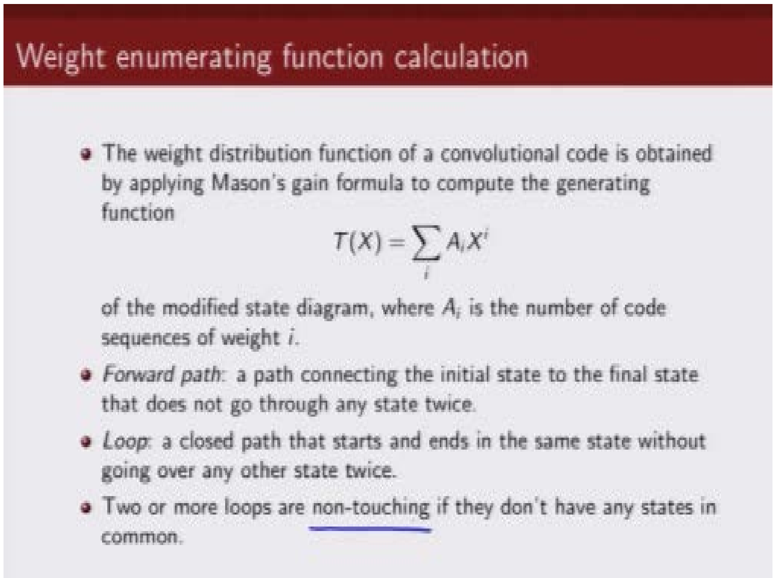
(Refer Slide Time: 06:16)



(Refer Slide Time: 06:31)

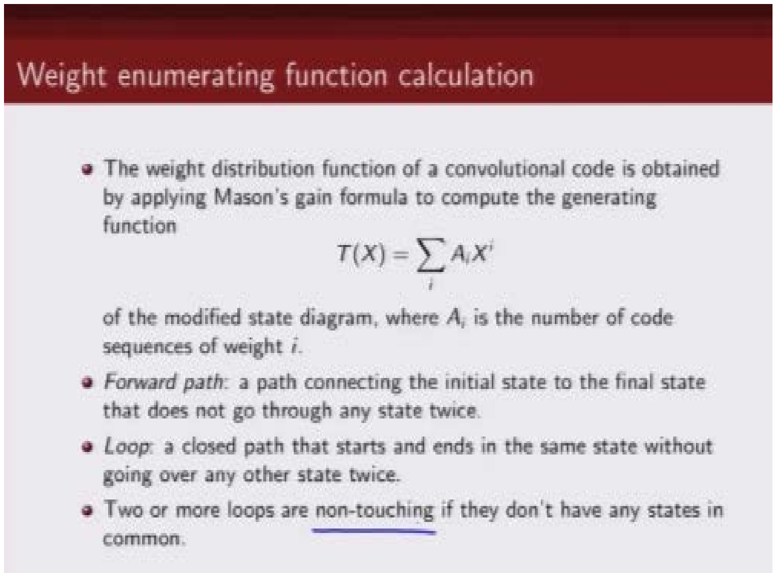


(Refer Slide Time: 06:34)



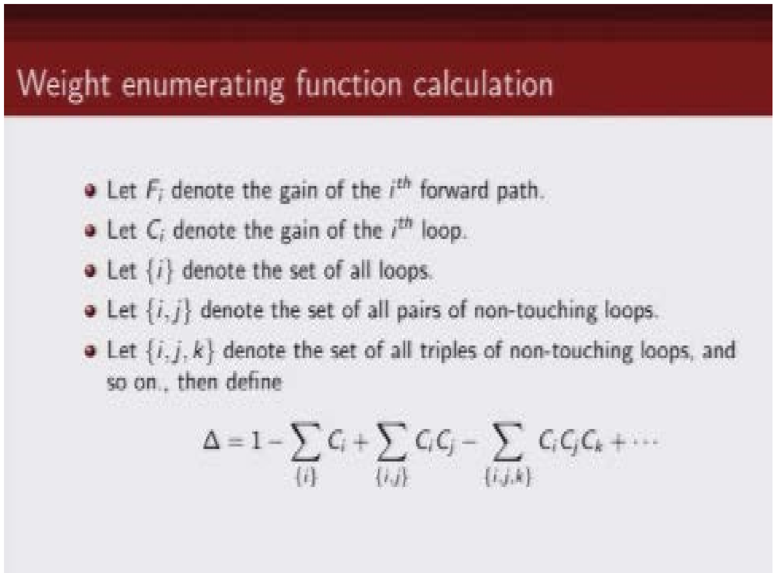
When do we say two loops are non touching, we say two loops are non touching if they do not have any state in common so again I repeat these three definitions.

(Refer Slide Time: 06:48)



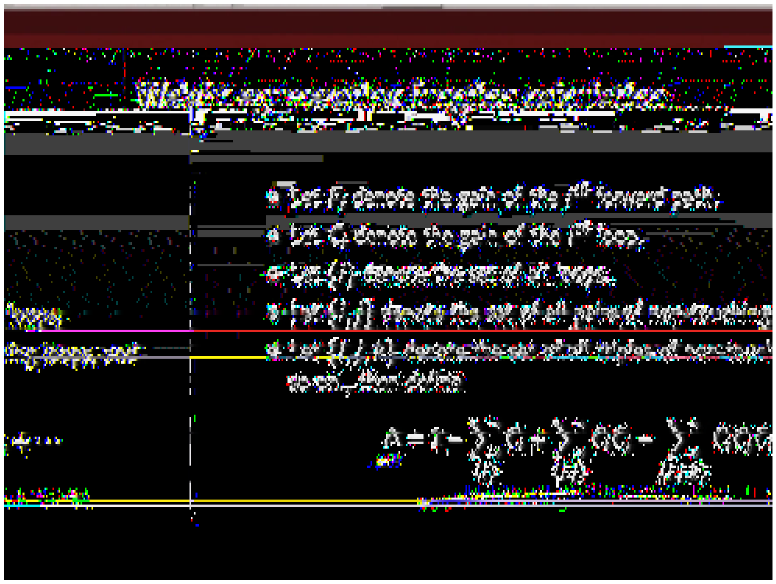
Forward path is a path from initial state to the final state without visiting any state twice, a loop is the close path starting and ending at the same state without going over the same state twice and two or more loops are non-touching if they do not have any state in common.

(Refer Slide Time: 07:13)



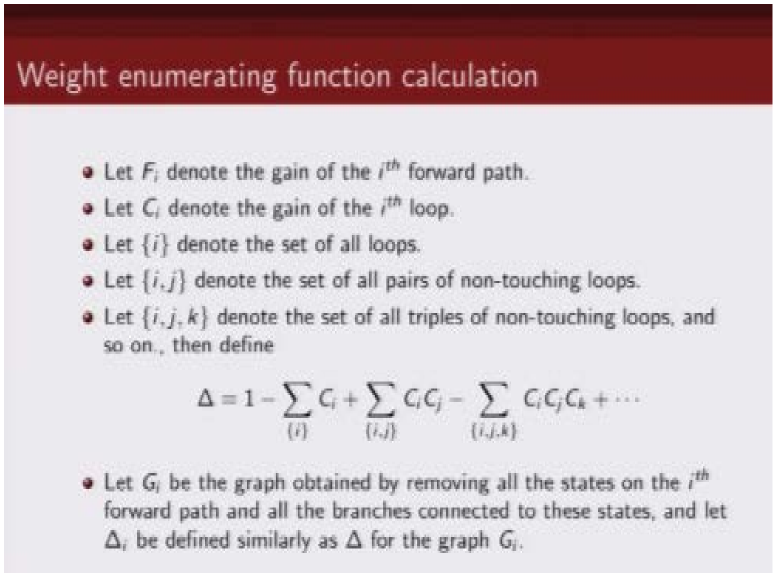
Now let us denote by F_i the gain of the i^{th} forward path and let C_i denote the gain for the i^{th} loop, we denote by this the set of all loops similarly this set of i and j will denote set of all pairs of non – touching loops. This triplet will define set of all triplets of non-touching loops. So if we use this we define a term Δ which is defined as follows.

(Refer Slide Time: 07:59)



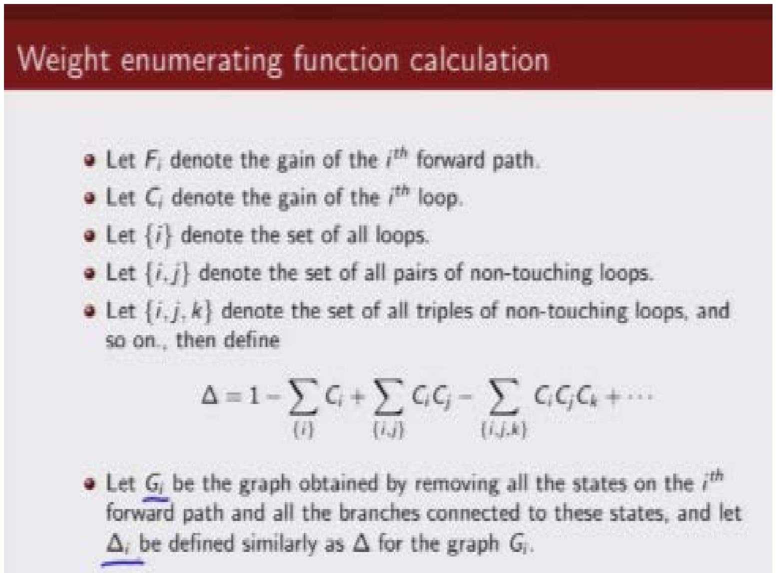
This 1-Σ of all the gains of the loops plus product of gains of all those non-touching loops minus this is product of set of all triplets of non-touching loop and it goes on like this, so that is our Δ.

(Refer Slide Time: 08:27)



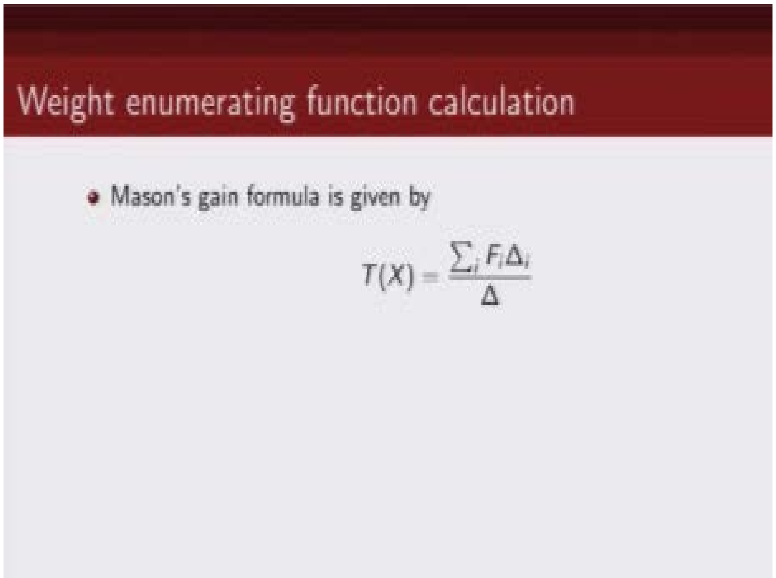
We define our graph that is obtained after we remove all states belonging to an i^{th} forward path by G_i .

(Refer Slide Time: 08:39)



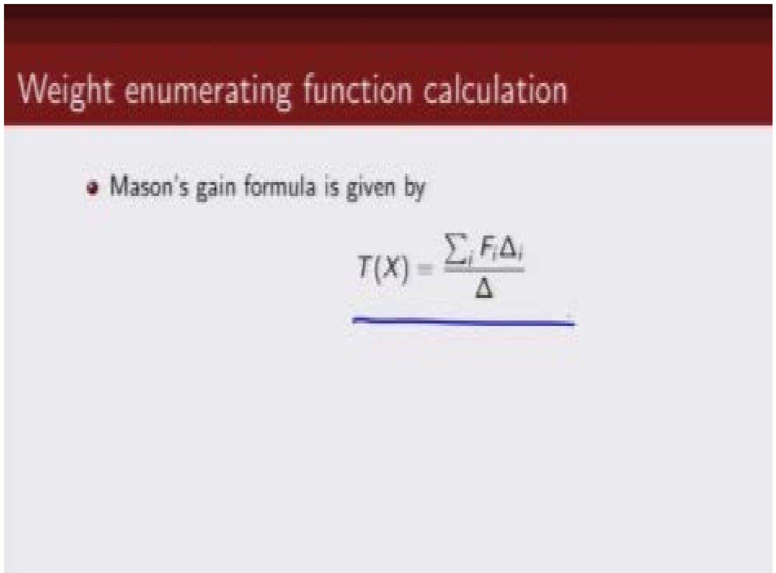
So G_i is basically the graph remaining after we remove the i^{th} forward path. And the Δ corresponding to this modified graph will be denoted by delta Δ_i.

(Refer Slide Time: 08:58)

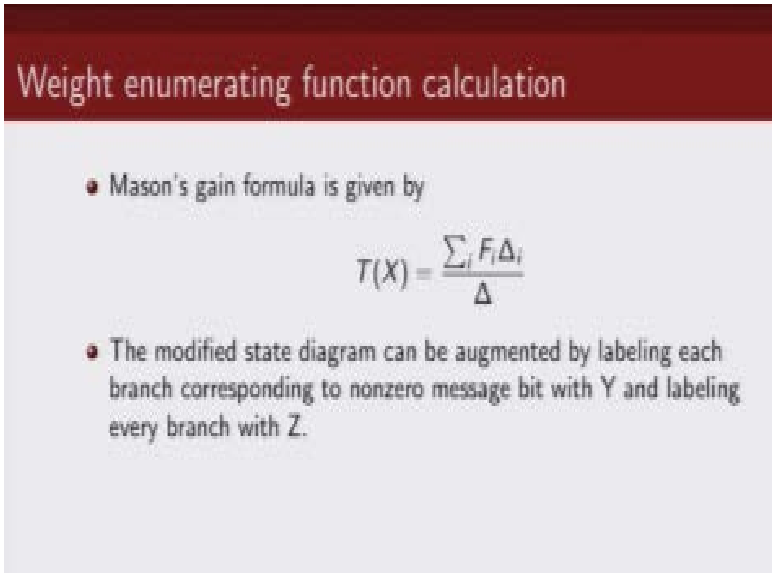


So the mason gains formula then says that the generator function for this convolutional encoder can then be given by this expression.

(Refer Slide Time: 09:07)

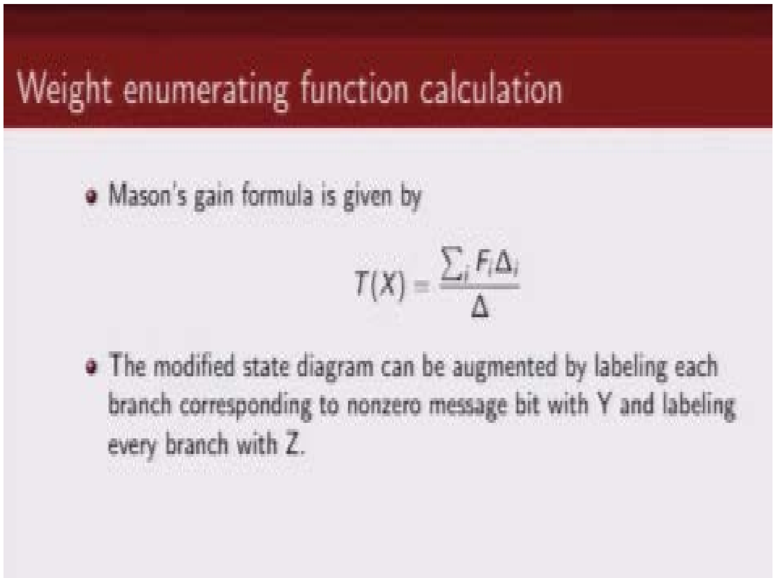


(Refer Slide Time: 09:07)



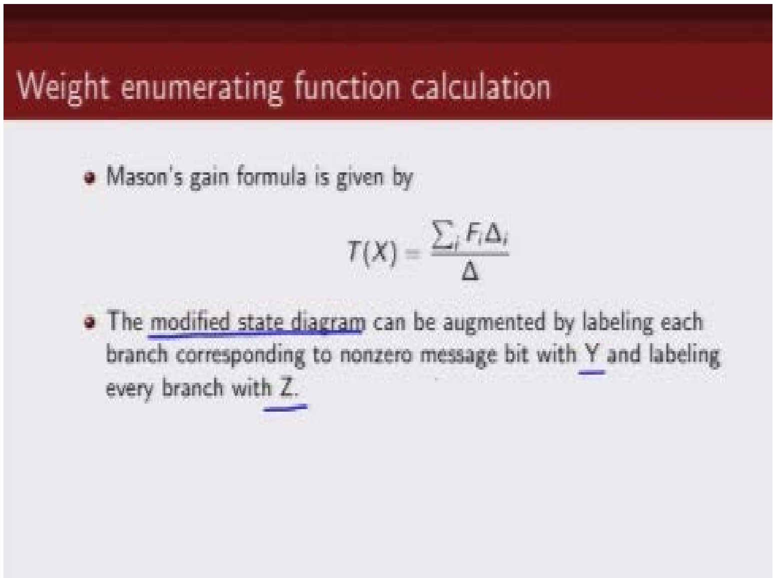
So this modified state diagram can be augmented to include more information, now what we had done so far was we labeled the branches of this state diagram by the overall total weight.

(Refer Slide Time: 09:33)



Now we can also augment this by mentioning what is the input that results in that output weight so we can label the weight of the input by Y.

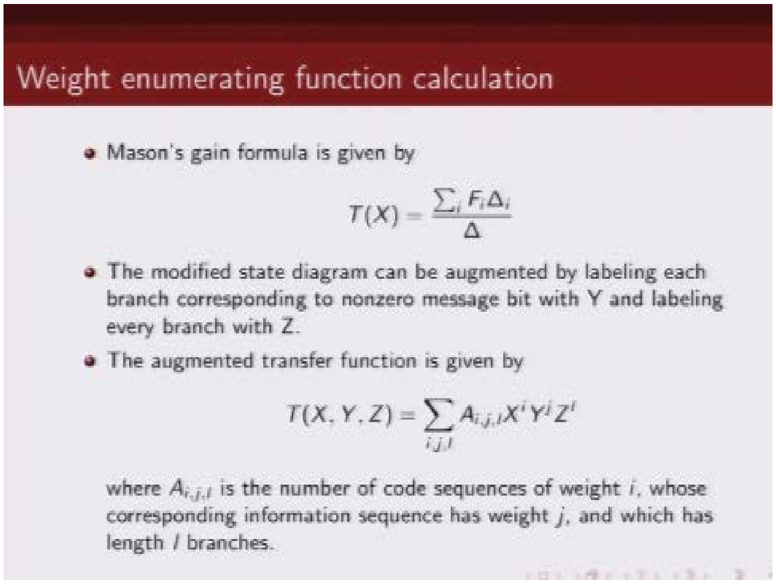
(Refer Slide Time: 09:43)



So the power of Y will denote what is the input weight and similarly we can label each branch by Z. So in a path gain formula, the degree of Z will tell us, like what is the length of nonzero path. So degree of Z will tell us like once it diverge from all zero state after how much time it comes back into all zero state.

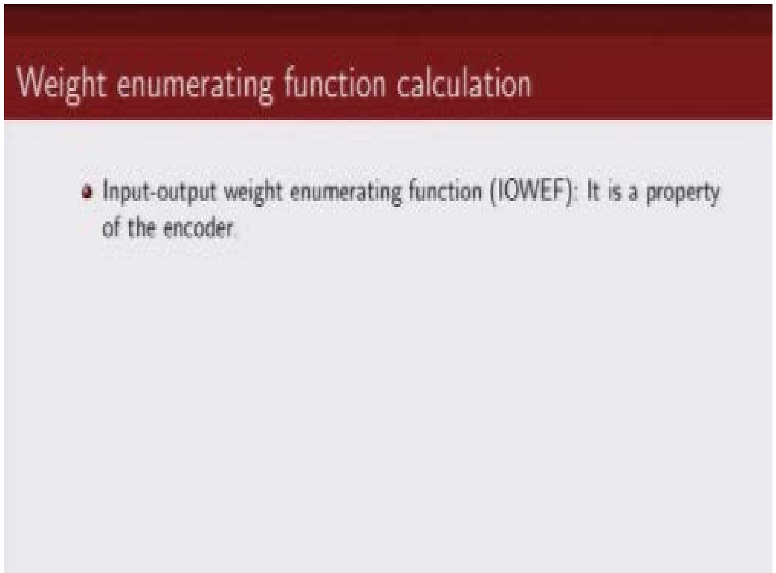
So we can augment our state diagram by what I call a modified state diagram by adding two additional information, one is the weight of the message bit which will be denoted by power of Y^i and other is branch which should be denoted by Z.

(Refer Slide Time: 10:04)



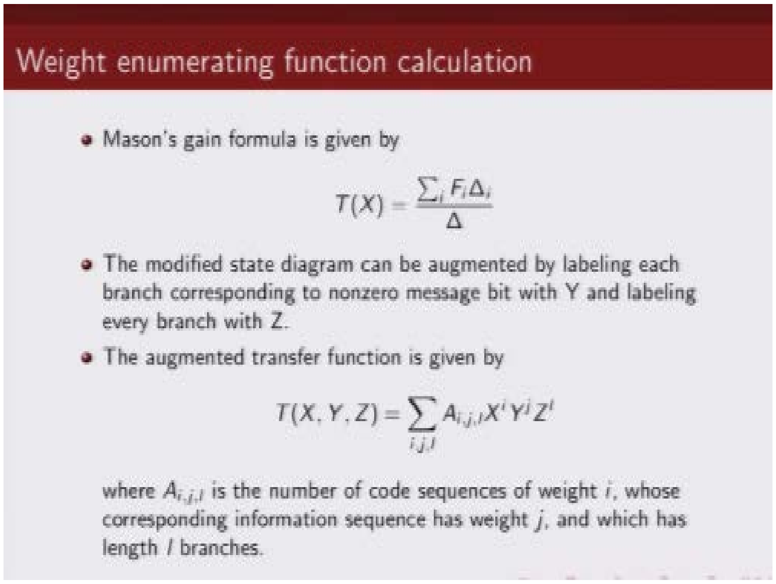
So we can then similarly define an augmented transfer function which will not only tell us the code word weight but it also tell us what is the input weight that results in that particular output weight and it also tell us length of that particular code word, by length I mean the time it diverges from all zero state until it cause back to all zero state.

(Refer Slide Time: 11:21)



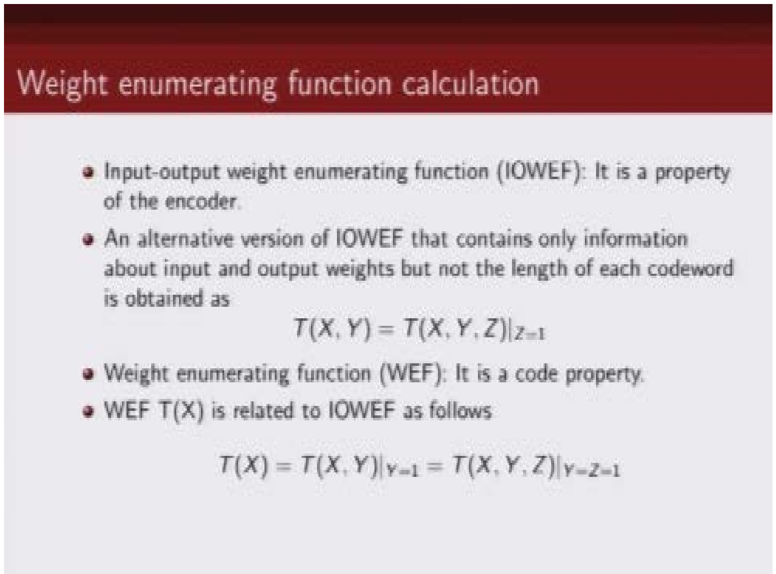
So this is basically what we call input-output weight enumerating function.

(Refer Slide Time: 11:29)



Because this function is enumerating for what input you get what output, okay so this will give us weight input-output weight enumerating function.

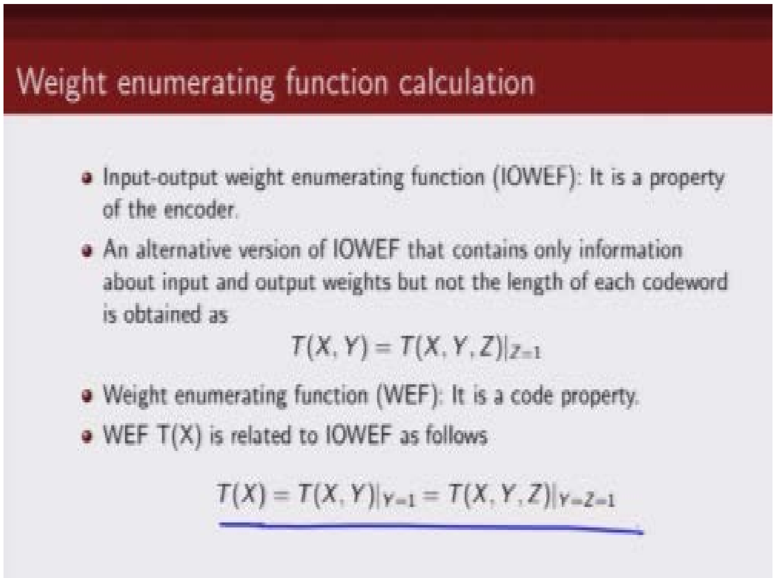
(Refer Slide Time: 11:39)



And it is the property of the encoder an alternative version of this input-output weight enmerating function is one that contains only information about the input and output weight and not the length of each code word, so if we put z = 1 basically this is going to give us [indiscernible][00:12:05] in diversion of input-output weight enumerating function.

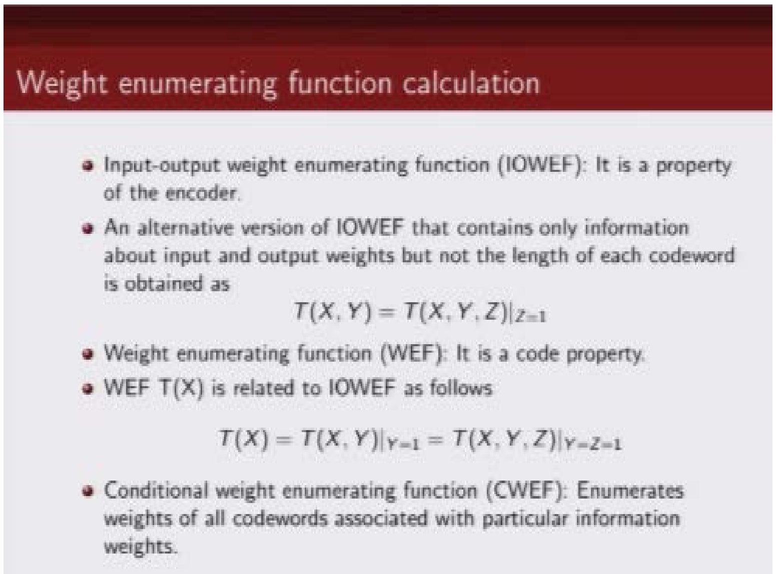
And what is weight enumerating function? The weight enumerating function will only tell us what is the overall code word weight and this is a property of the convolutional code.

(Refer Slide Time: 12:24)



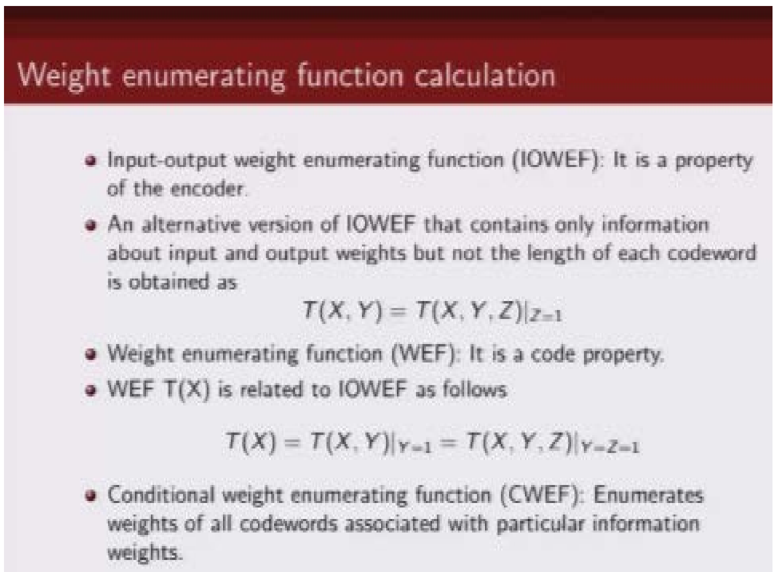
So weight enumerating function is related to input output weight enumerating function in this particular way so if you put z and y as 1 in the input output weight enumerating function we will get back our weight enumerating function.

(Refer Slide Time: 12:44)



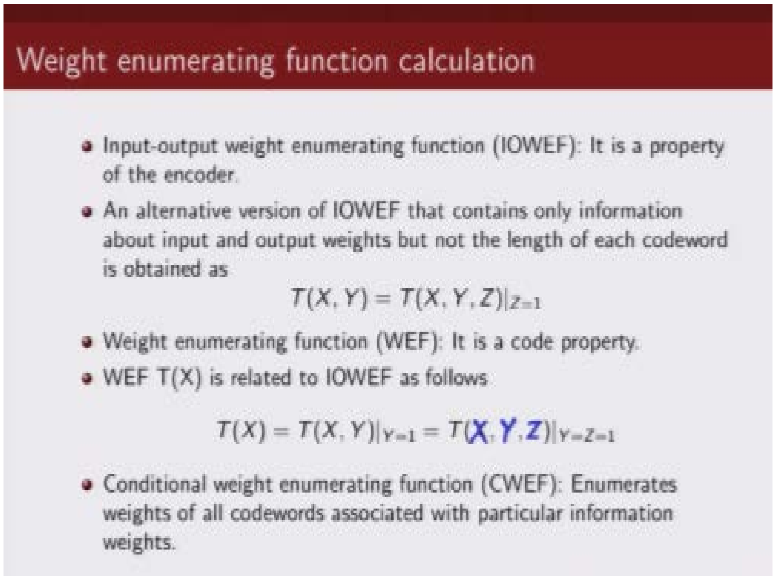
Similarly we can define what is known as conditional weight enumerating function so what is conditional weight enumerating function? The conditional weight enumerating function it enumerates weights of all code words associated with a particular information weight, so if you are interested in knowing what is the output weight correspond to weight four input sequence, so from the input output weight enumerating function by collecting all terms which will have w_4.

(Refer Slide Time: 13:21)



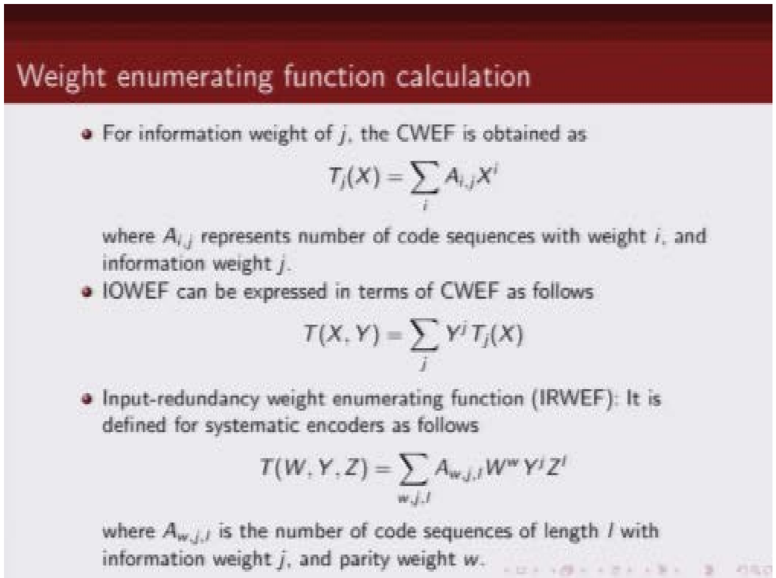
We can find out what is the weight of all code words corresponding to a output weight input weight of four and again we are using Y to denote the input weights so if you are interested in input weight four we should look for terms containing Y^4.

(Refer Slide Time: 13:51)



So this denotes the input weight this denotes the output coded weight and this denotes the length.

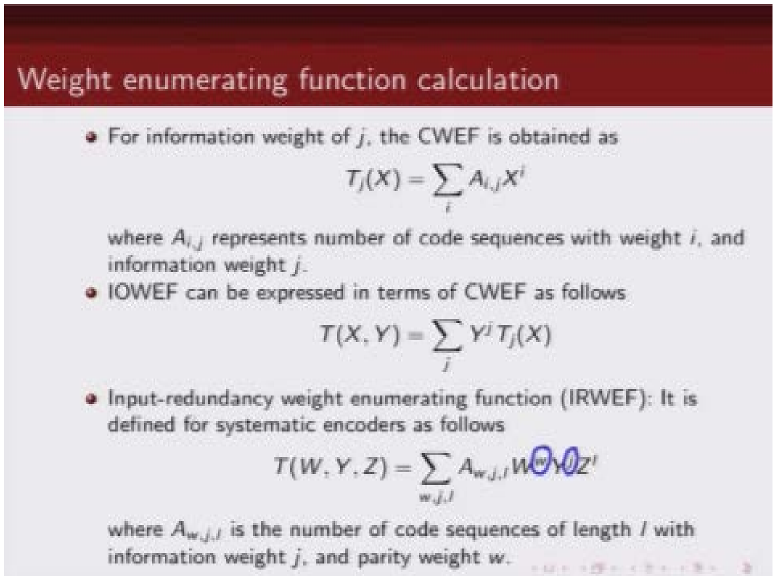
(Refer Slide Time: 14:02)



So as I said for a input weight of j conditional enumerating function will give us what is the output code weight that you can achieve for a input weight of j. And we write our input output weight enumerating function in terms of conditional weight enumerating function so this is basically input of weight j will result in conditional weight enumerating function and we show it for all j that will be our input output weight enumerating function.

There is another property which is define for systematic encoders which is called input redundancy weight enumerating function. So here because the output weight of a systematic encoder consist of weight of the information bits and weight of the parity bits.

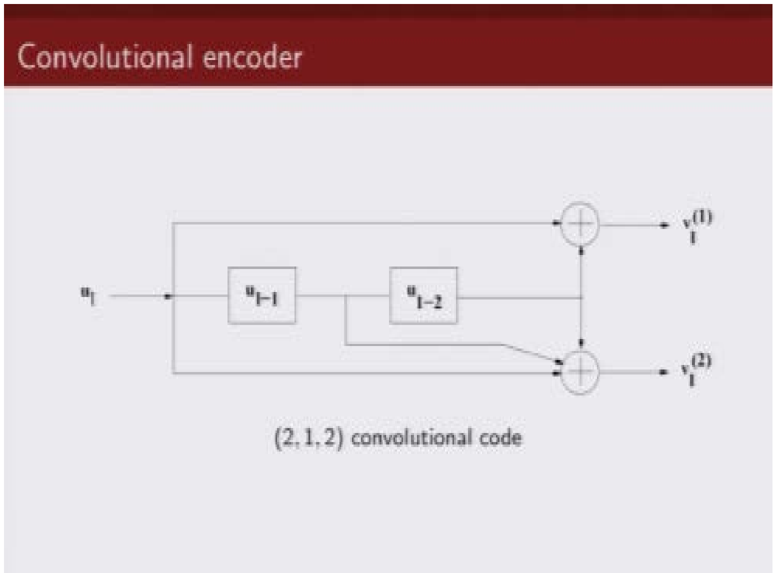
(Refer Slide Time: 15:10)



Now since the power of Y already is denoting the weight of the information bits so when you are asked to show the output weight you can just instead of saying the output weight you can just specify the weight of the parity bits, because it is systematic encoder the overall weight will be weight of the parity bits and weight of the information bits.

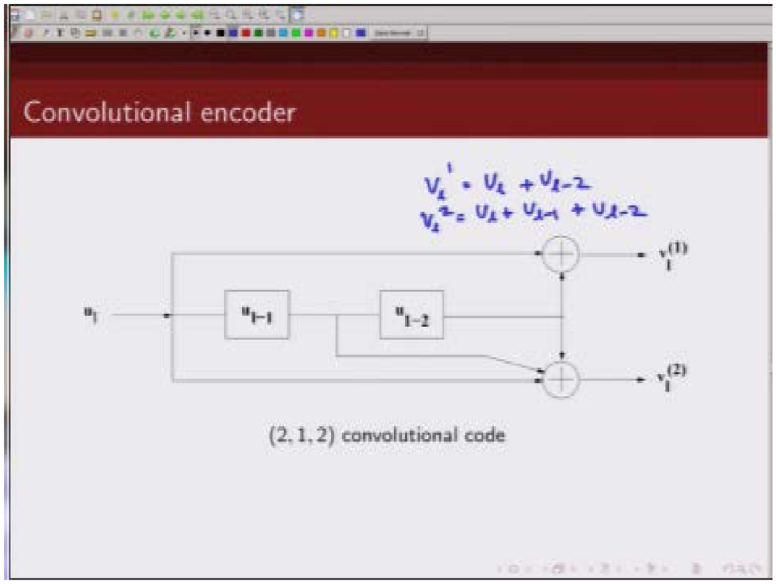
So overall weight would be w+j so input redundancy weight enumerating function is defined for systematic encoders. So where instead of specifying the overall coded weight here you only specify the weight of the parity bits.

(Refer Slide Time: 15:59)



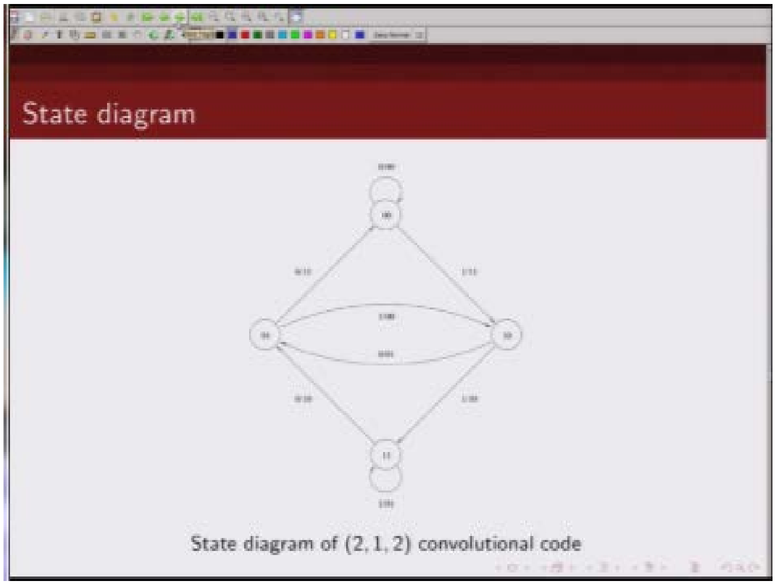
Now let us take an example to illustrate how we can find the weight enumerating function of a convolutional code.

(Refer Slide Time: 16:10)



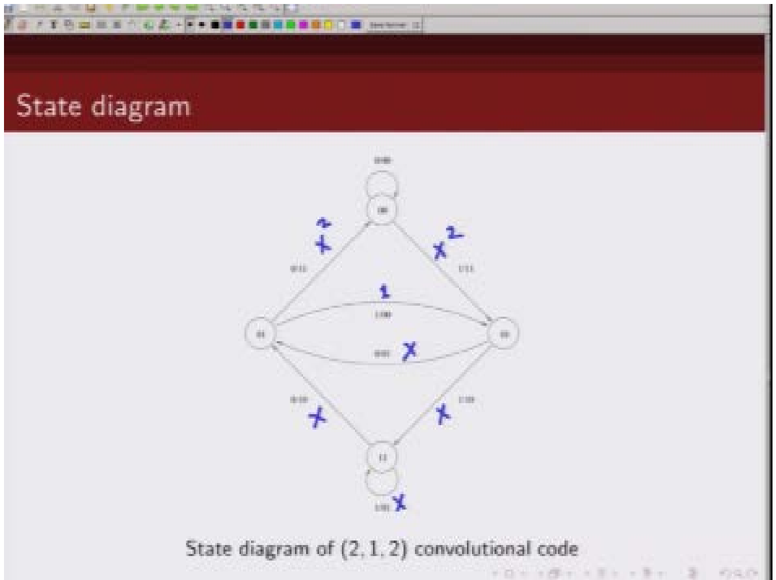
So we are going to consider a rate 1/2 convolutional code whose memory is 2 to this is the convolutional code we can see basically v_l^1 is $u_l + u_{l-2}$ and v_l^2 is nothing but $u_l + u_{l-1} + u_{l-2}$ the convolutional code.

(Refer Slide Time: 16:45)



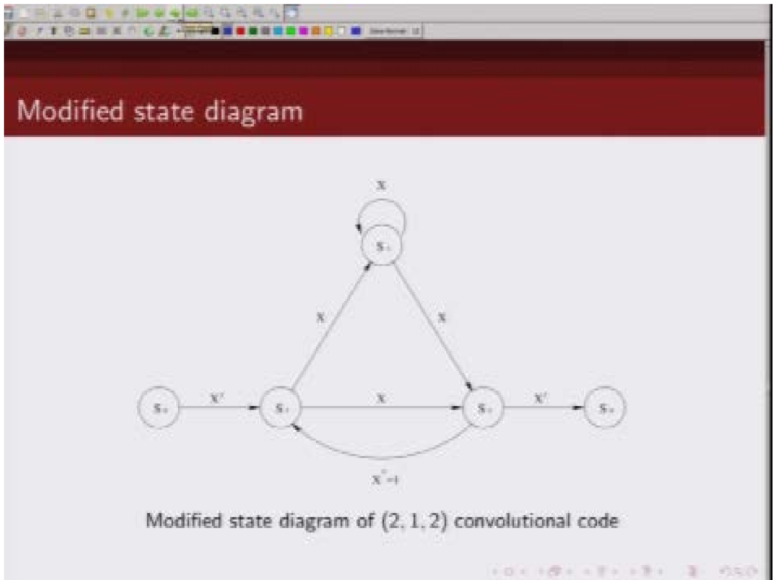
Now the state diagram for this convolutional encoder is given by this. Now we are going to modify this state diagram for the purpose of calculating the weight enumerating function. So recall what are the modifications we have to do.

(Refer Slide Time: 17:04)



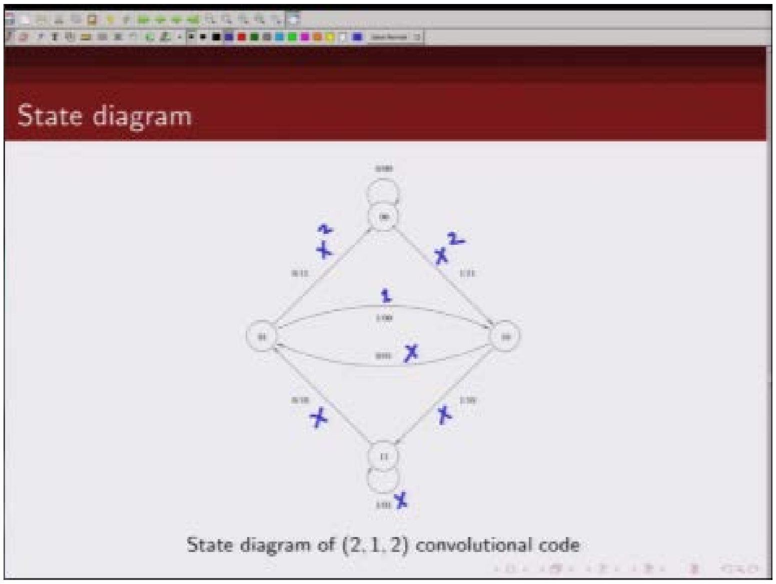
We have to remove this self loop around all 0 state and we have to split this all 0 state into 2 state, initial state and final state. Next we have to label all these branches by weight of the output bit. So this will be x^2. This will be x^0 which is 1, this will be x, this will be x^2, this will be x, this will be x, this will be x. if you go back.

(Refer Slide Time: 17:48)



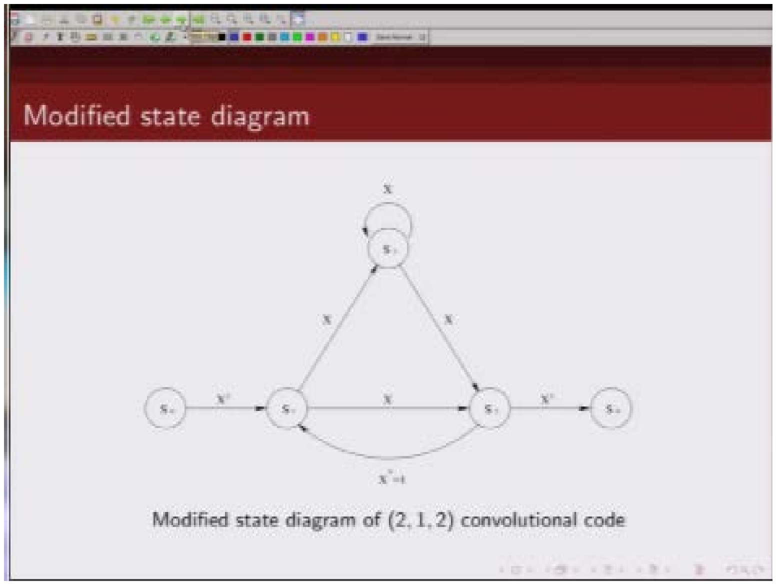
This is how my augmented will modified state diagram will look like. So what I did was.

(Refer Slide Time: 17:58)



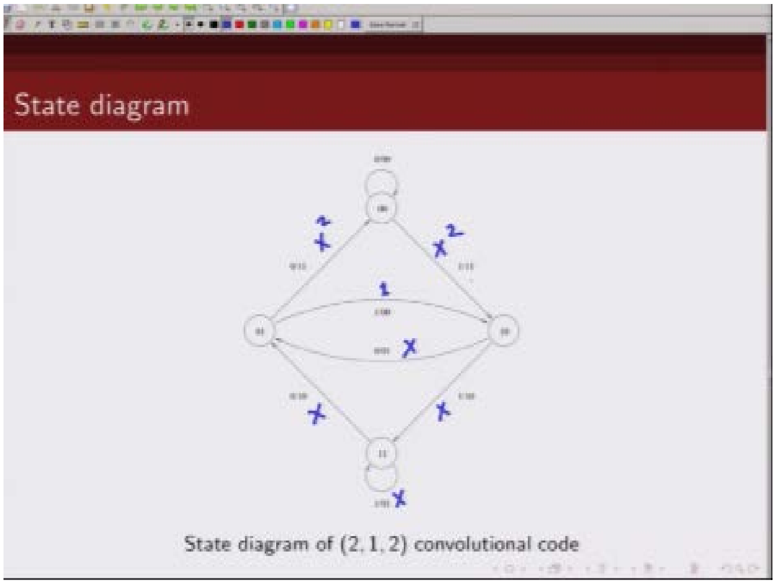
I have these states here, I split this all 0 state into 2 state.

(Refer Slide Time: 18:08)



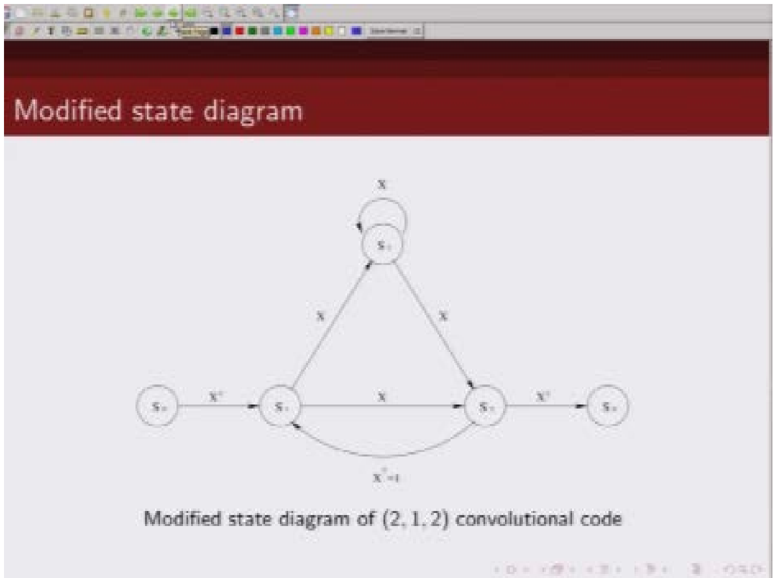
So this state was split into initial state and final state okay. Next what did I do?

(Refer Slide Time: 18:14)



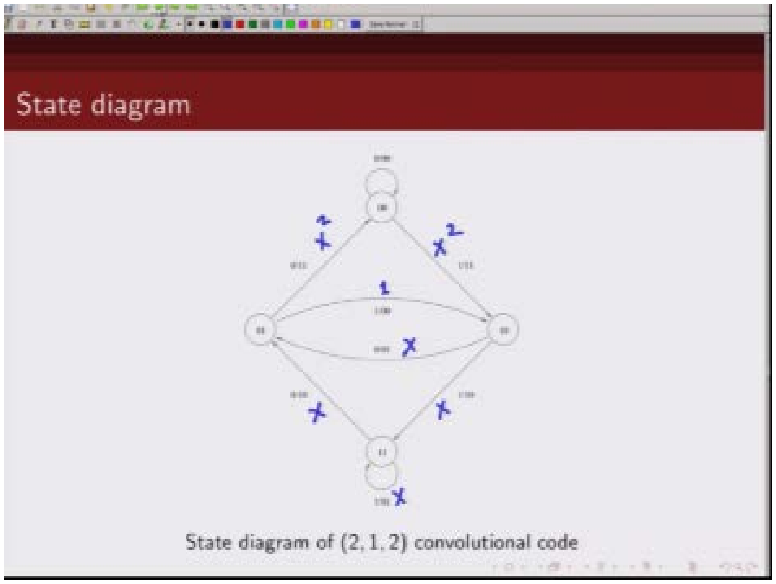
I redrew the same diagram but I labeled each transition by the weight of the outputs so you can see from 00 I am going to 10 and its output weight is 11, which is x^2. So let us go back here.

(Refer Slide Time: 18:32)



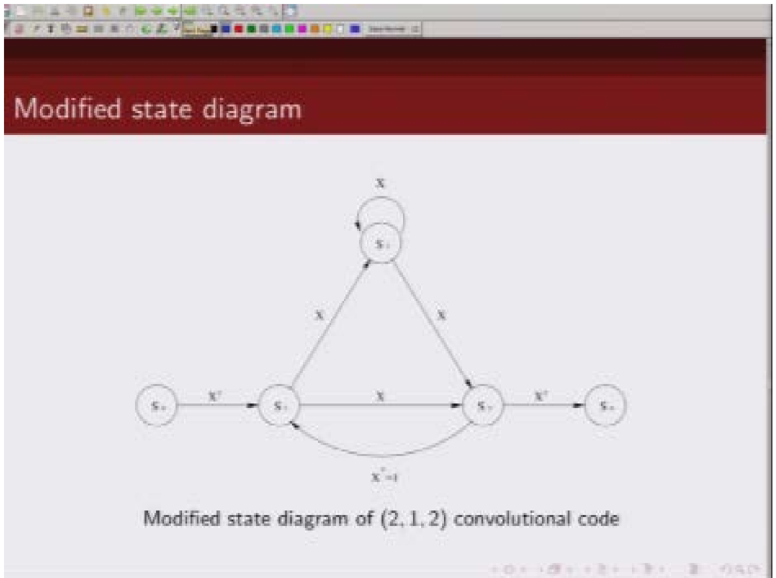
From 00 I am going to this state and output is x^2 this branch is labeled by x^2.

(Refer Slide Time: 18:42)



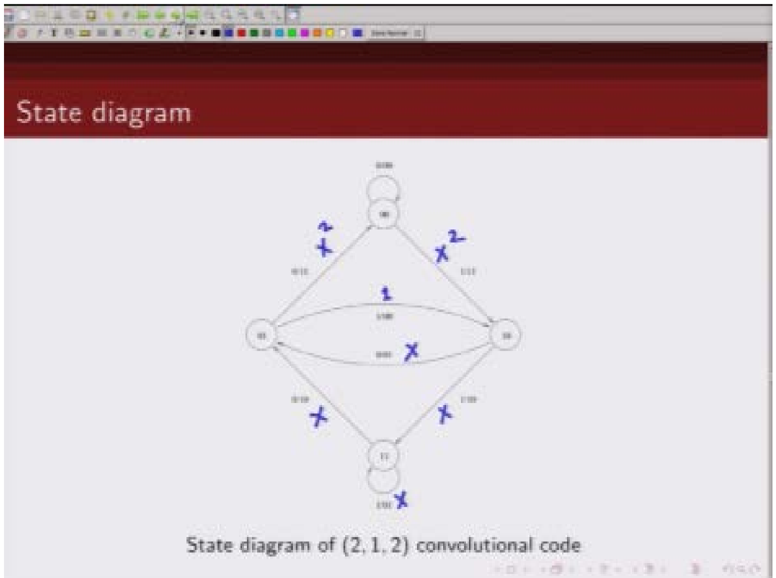
Similarly, you can see from let us say from this state you are going to this state and the weight is x, you can see.

(Refer Slide Time: 18:54)



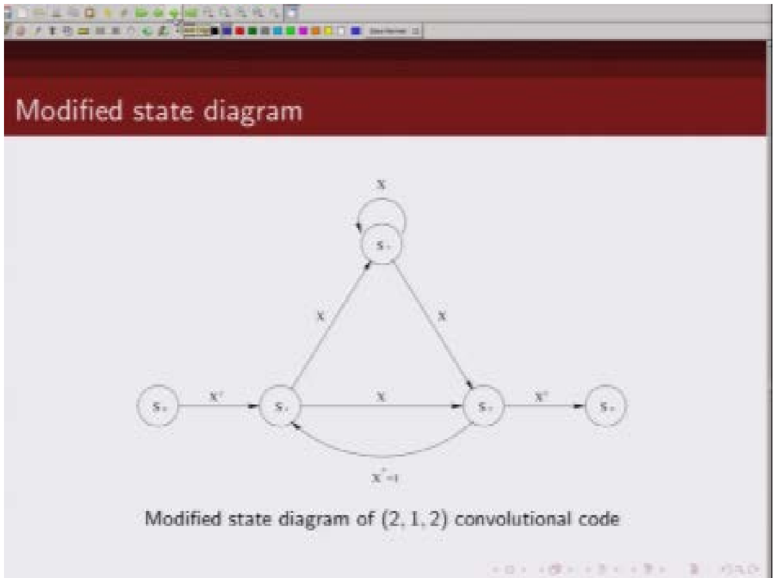
From this I am going to this state and the branch is labeled by x okay.

(Refer Slide Time: 19:02)



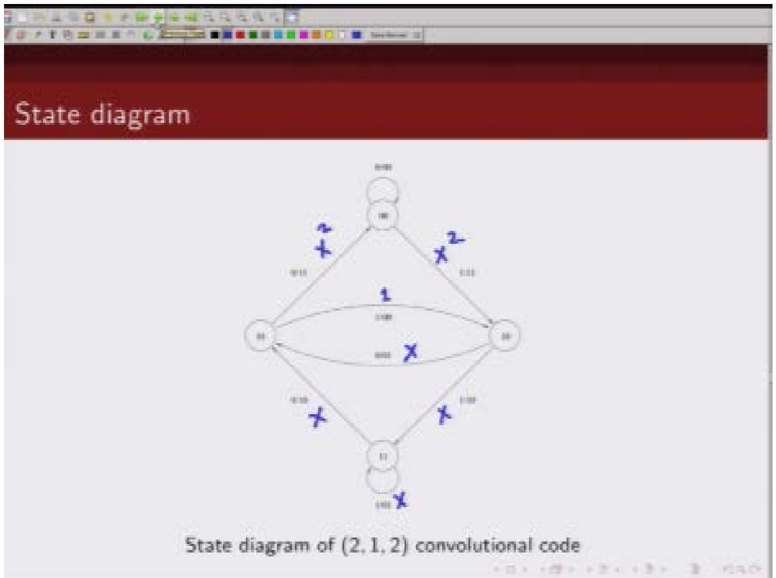
From this state you are going to this 11 state, and the branch is labeled by x.

(Refer Slide Time: 19:10)



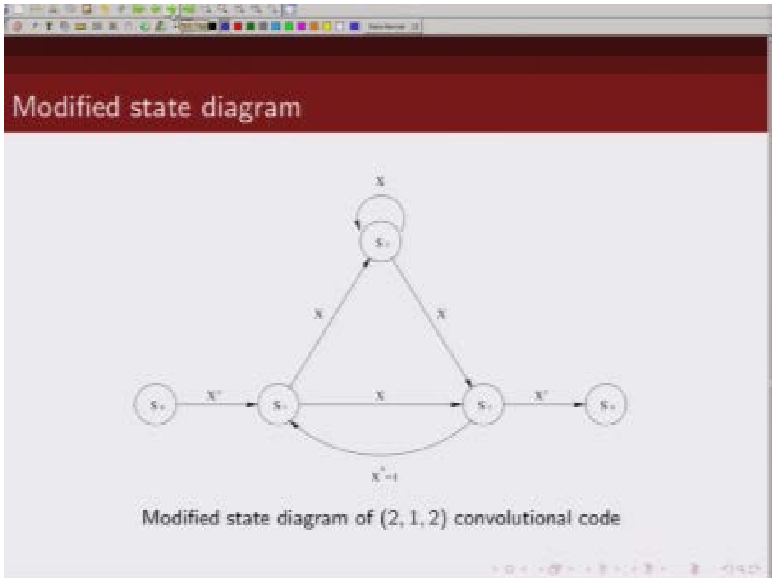
From this you are going to this 11 state and it is labeled by x.

(Refer Slide Time: 19:17)



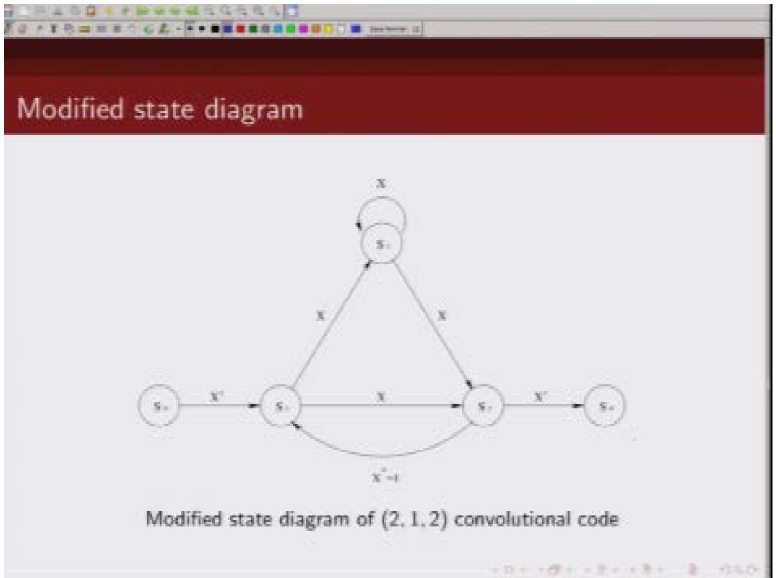
Around the state 11, there is a loop which has weight 1 x.

(Refer Slide Time: 19:27)



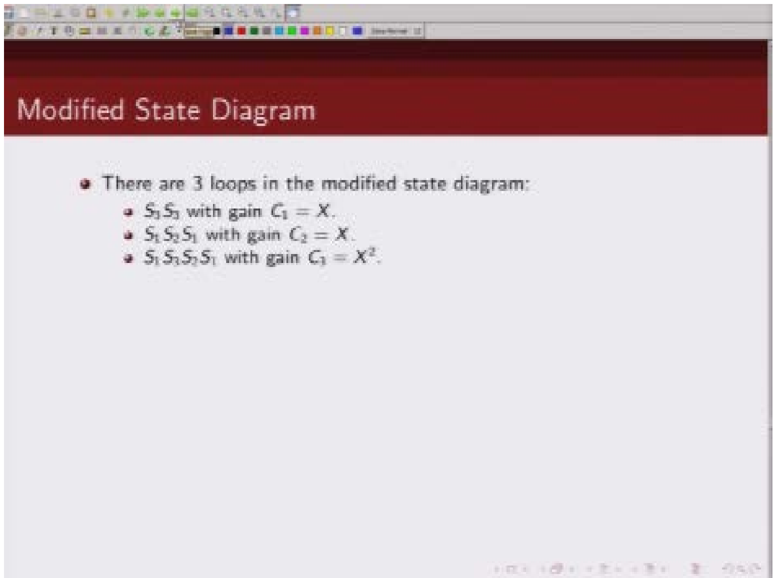
So this is my loop around 11, which has weight x, so like that basically we modify the state diagram and this is how our modified state diagram of a rate ½ convolutional code that we just showed looks like okay.

(Refer Slide Time: 19:53)



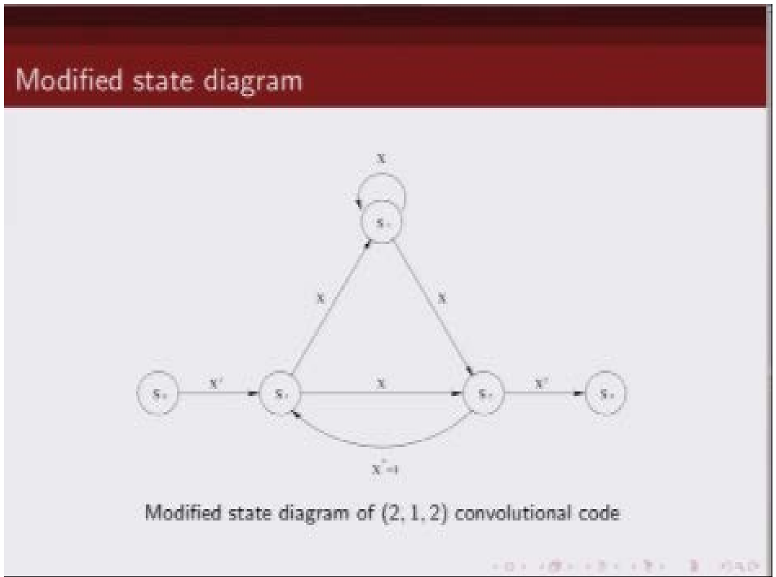
Now the next step is, we need to find out what are the forward parts, what are loops, what are the non-touching loops, and then we need to find out the path gains along those forward paths. We need to find out delta is corresponding to this forward paths, and then we need to apply Mason's gain formula to get the weight enumerating function.

(Refer Slide Time: 20:24)

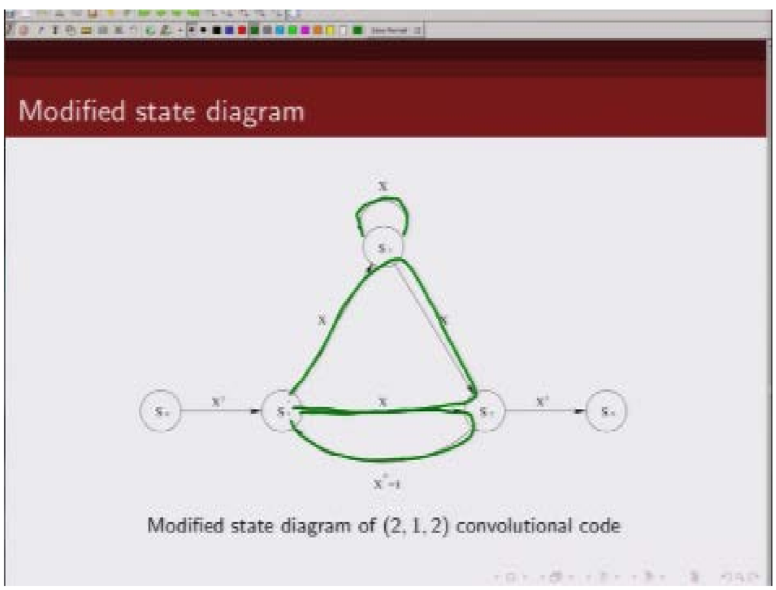


So first we find out what are the loops, so there are three loops in this, 1 is a self loop around the state S_3 you can see.

(Refer Slide Time: 20:36)

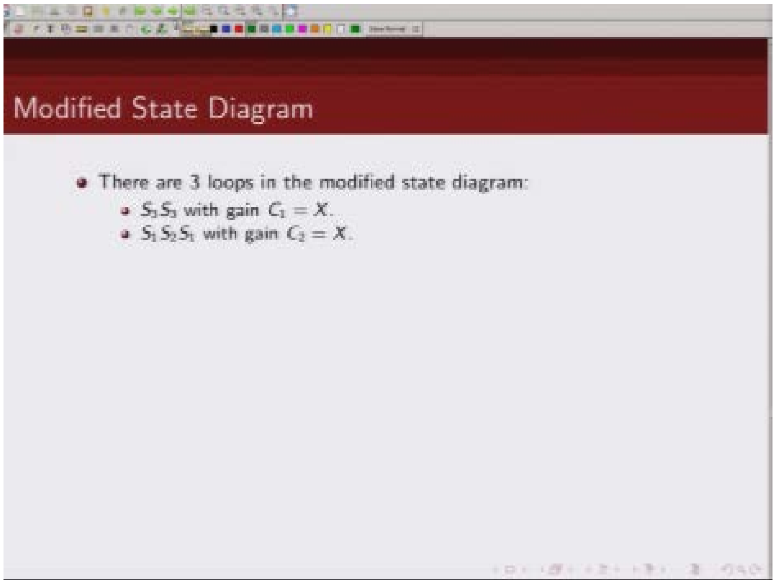


(Refer Slide Time: 20:38)



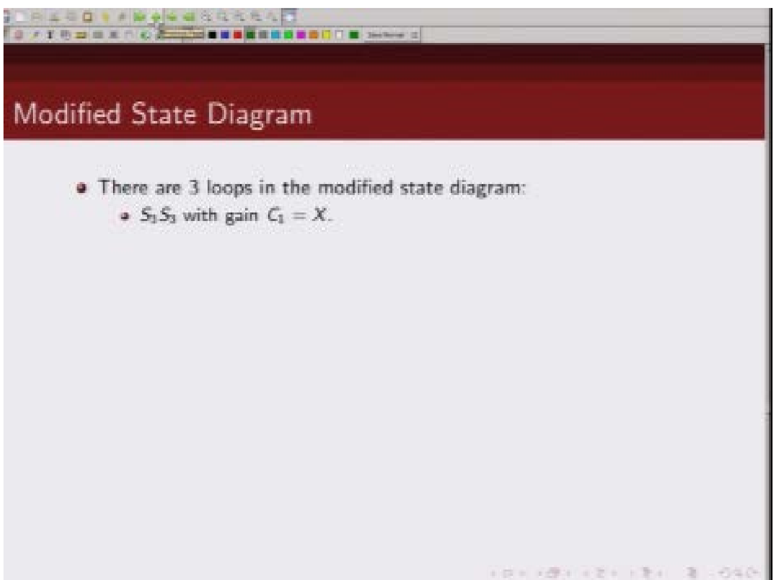
This 1 loop right, there is another loop right, and then there is another loop. So there are three loops here and that is what I am denoting.

(Refer Slide Time: 20:56)

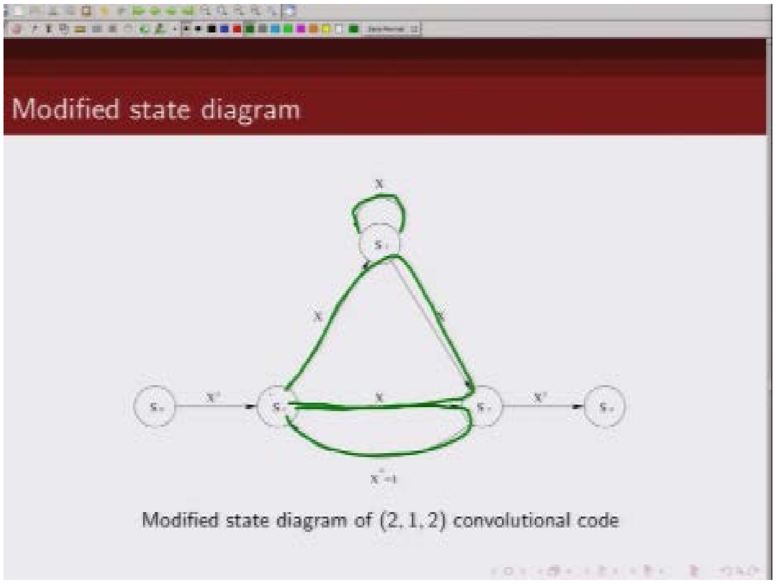


It by S_3S_3 is gain X, next one is $S_1S_2S_1$.

(Refer Slide Time: 21:02)

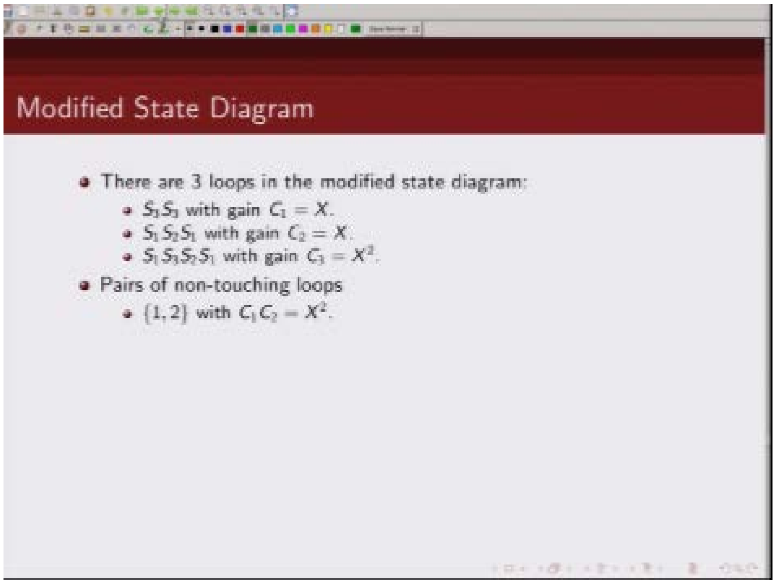


(Refer Slide Time: 21:05)



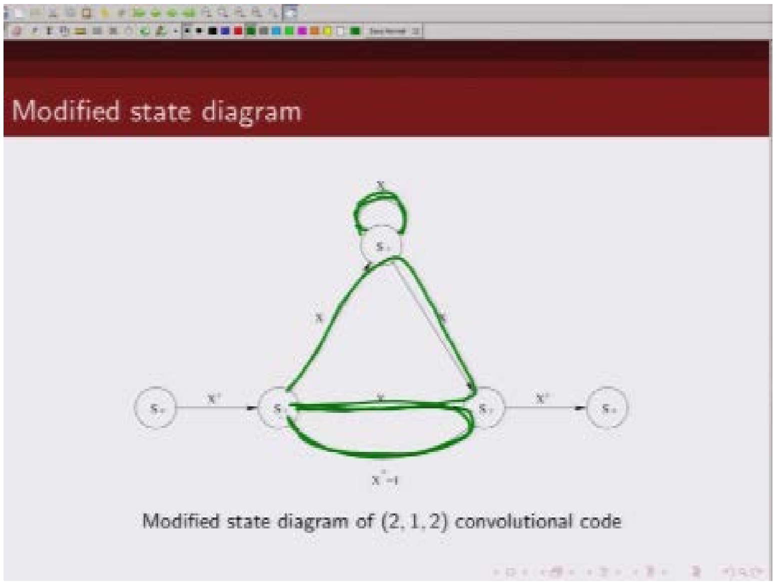
$S_1S_2S_1$ is this one okay. And the third loop is given by this.

(Refer Slide Time: 21:12)



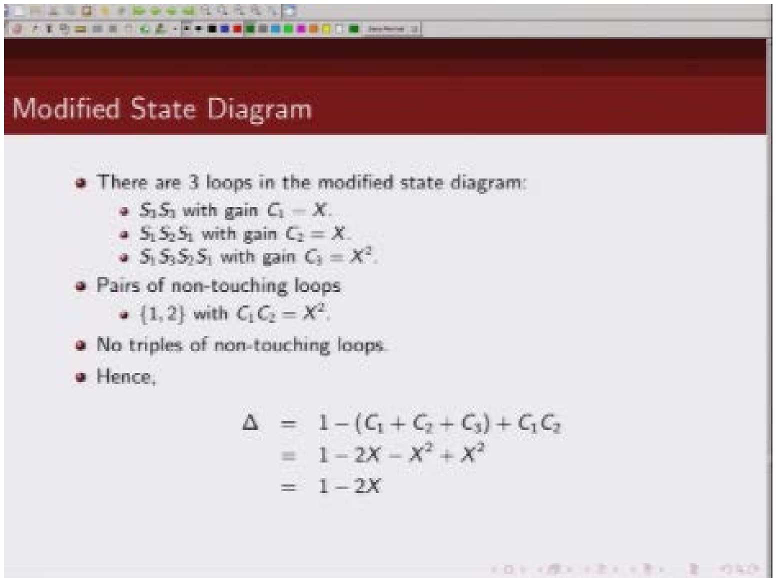
So these are the three loops and corresponding to these three loops these are the gains. Next, what are the pair of non-touching loops? Now, only these two C_1 and C_2 are non touching loops you can see and go back to this example.

(Refer Slide Time: 21:37)



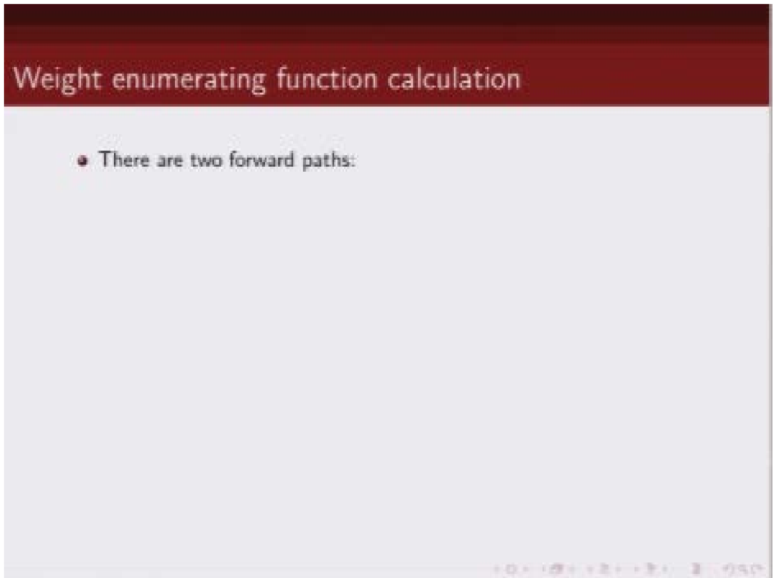
This loop and this loop are non-touching why this loop contains S_3 and this loop contains S_1 and S_2. So they do not have any state common between these two loops.

(Refer Slide Time: 21:59)



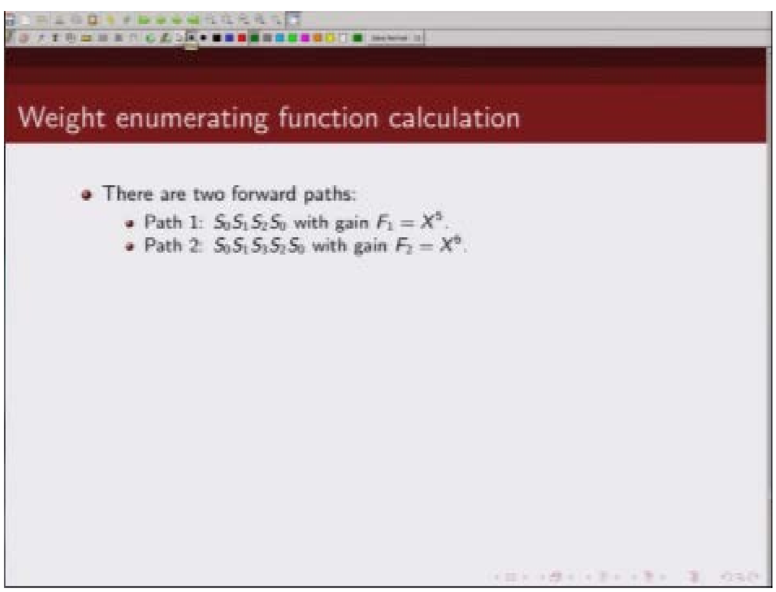
So the set of non-touching loops is basically the C_1 and C_2 and the gain corresponding to them is basically X^2. And there is no set of three loops which are non-touching. So now, we can then find out the value of delta which is 1-Σ of these loop gains and plus set of non-touching loops so this comes out to be 1-2X.

(Refer Slide Time: 22:34)



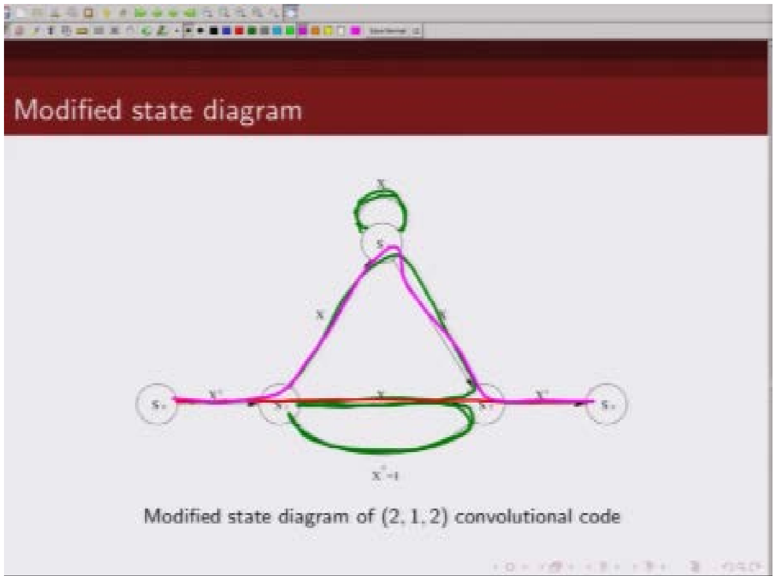
Next we are going to find out what are the forward paths.

(Refer Slide Time: 22:41)



So there are two forward paths in this, and we are going to show you.

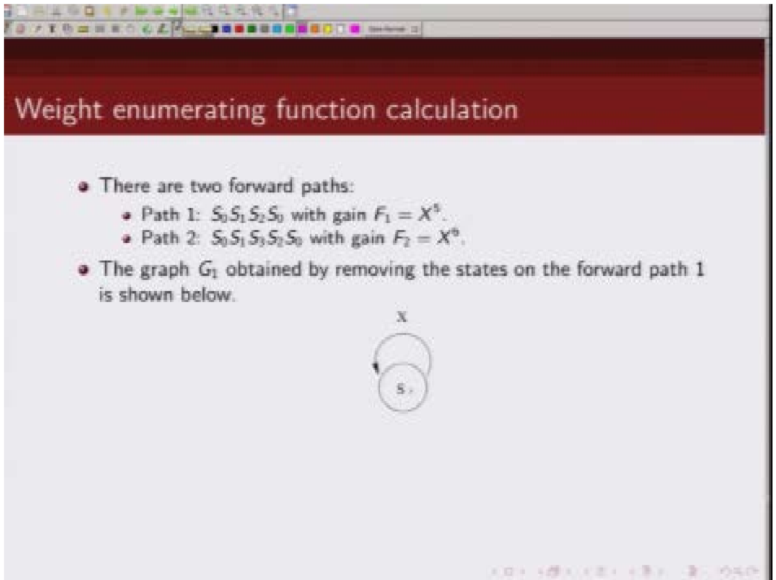
(Refer Slide Time: 22:47)



So let us use a different color pen let us use a red color pen, remember what is the forward path, a path from the initial state to the final state without going over any state twice. So one forward path is this, find what about another forward path, the another forward path is this. Both the cases you can see I am not going over any state twice.

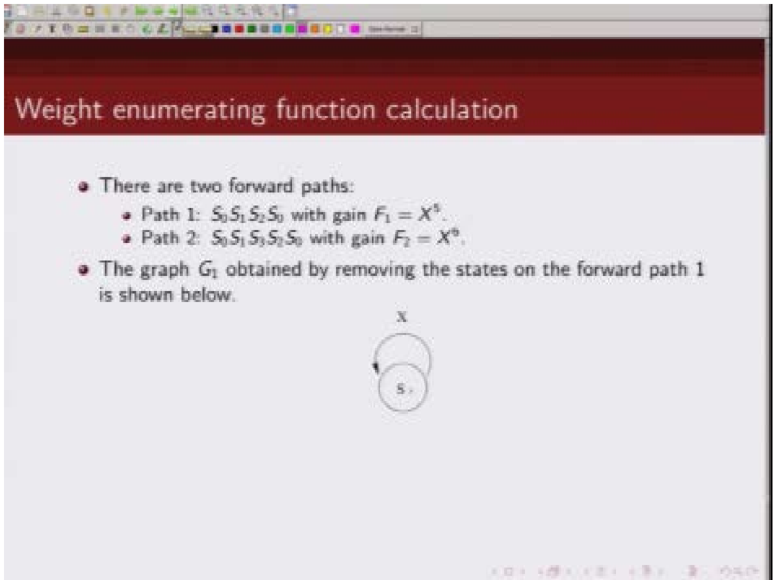
And there are only two forward path in this case. And what are the corresponding path gain for the one which I marked with red, this is x^2 x and x^2. So this will be $X^5$and this will be x^2 xx, x^2. So this will be X^6.

(Refer Slide Time: 23:53)

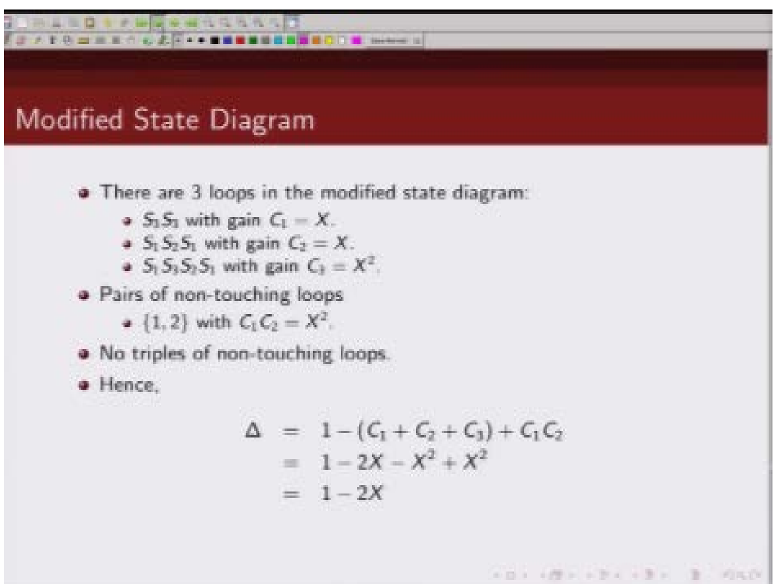


So then we have to forward path the 1 with gain X^5 another with gain X^6. Now what is the next step, we need to remove the forward path and see what is the graph remaining. And we need to compute the delta corresponding to that.

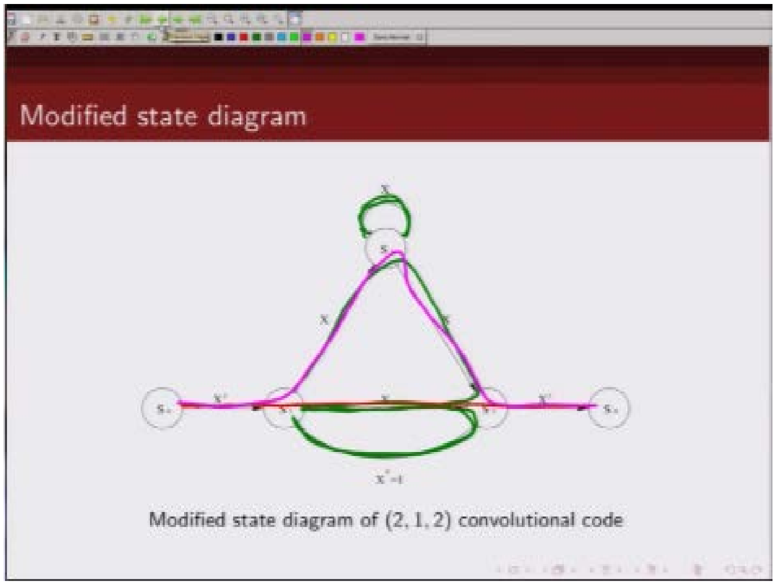
(Refer Slide Time: 24:16)



(Refer Slide Time: 24:17)

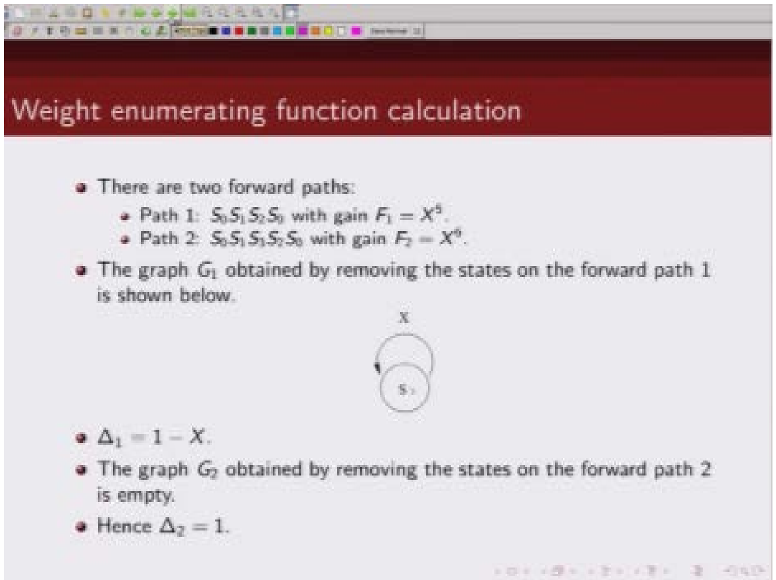


(Refer Slide Time: 24:20)



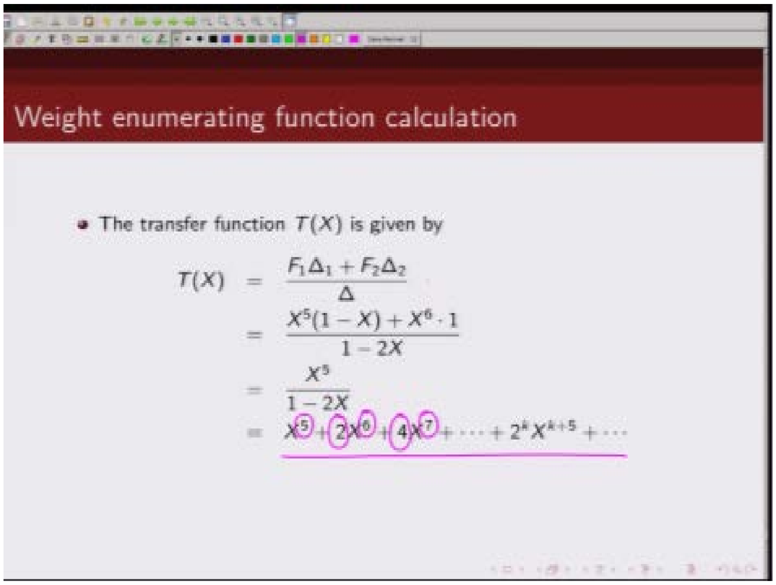
Now again let us go back to the same diagram, if I remove this forward path what is left in the graph only this, this node only this is remaining. And what about if I remove this forward path, if I remove this forward path everything is gone there is nothing left in the graph.

(Refer Slide Time: 24:38)



So that is what I am seeing here, if I remove the forward path 1, the only graph remaining is this, and the delta corresponding to this is basically there is only one loop with gain X so this is 1 – X. and for the second case, there is no graph left so Δ2 will be 1 okay. So now I have F1 Δ1, F2 Δ2 and I also have the value of Δ.

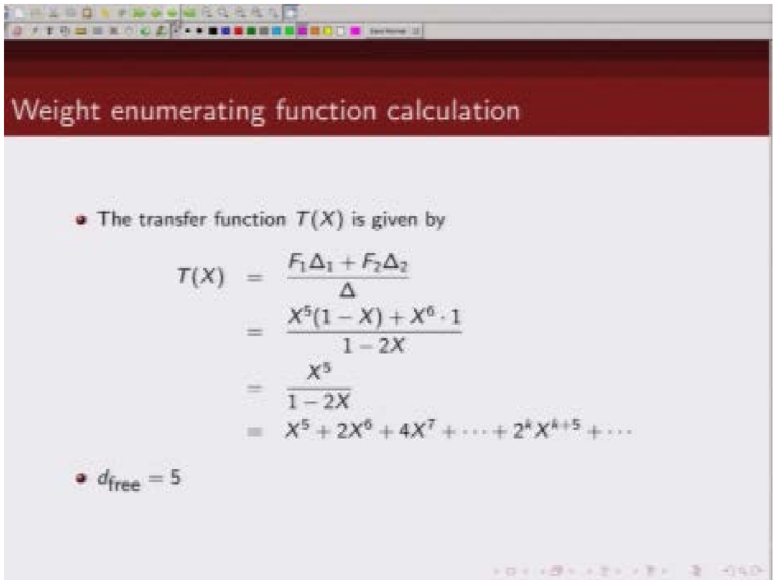
(Refer Slide Time: 25:17)



So I can then apply Mason's gain formula to get the weight enumerating function. So the weight enumerating function is given by this expression so I plug-in the value of F1 Δ1, F2 Δ2 and Δ and what I get is this expression which I can write like this.

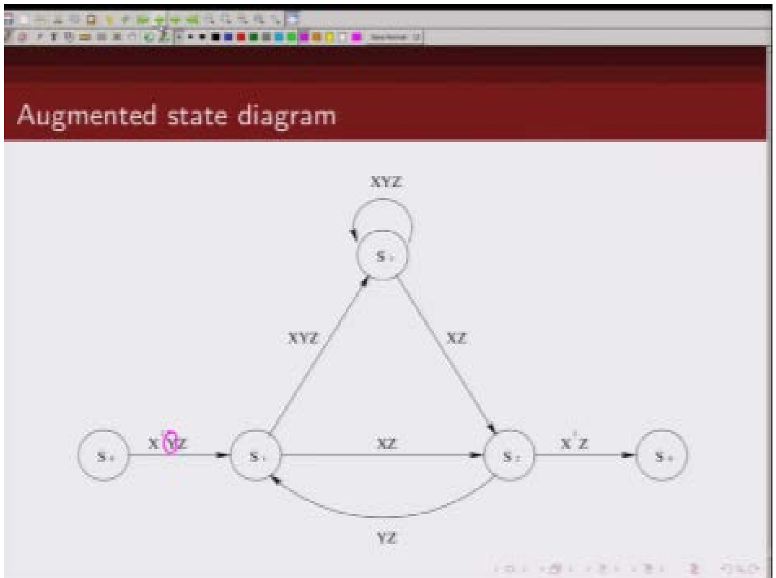
So you can see basically my output consist of one code word of weight 5, two code words of weight 6, four code words of weight 7. So you can see this transfer function is completely enumerating the weight distribution of my convolutional code. In the same thing I can do with augmented transfer function.

(Refer Slide Time: 26:04)



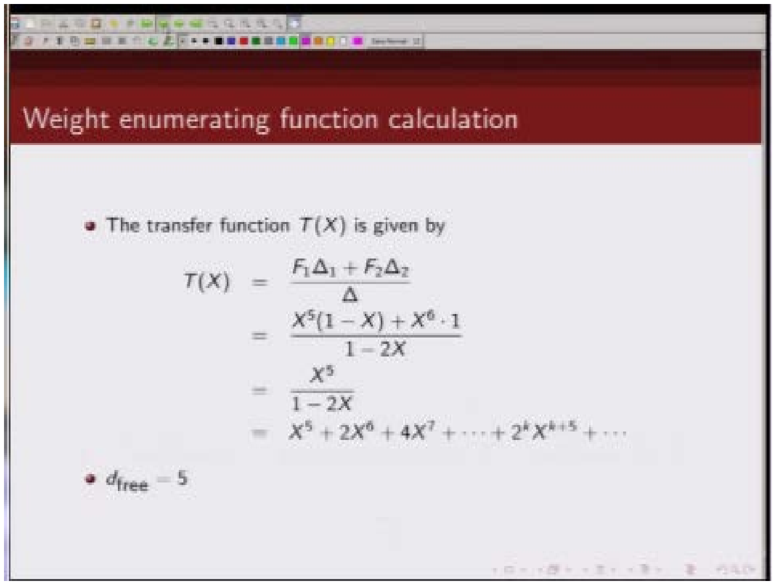
And again because the minimum weight is 5 so free distance of this convolutional code is 5.

(Refer Slide Time: 26:16)

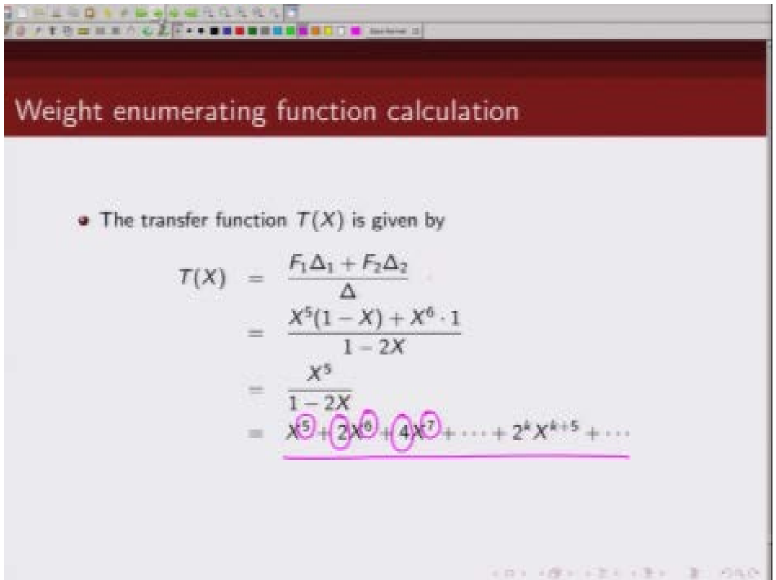


Now we repeat the same exercise with augmented state diagram. Now what was augmented state diagram each valid branch we added a, z to denote this you can reach from one state to another in one step. And we also added in each of these branches the weight corresponding to the information bits. So the information bit weight was 0, so with y0 so that was 1. So you can see in some cases the information sequence weight is 0.

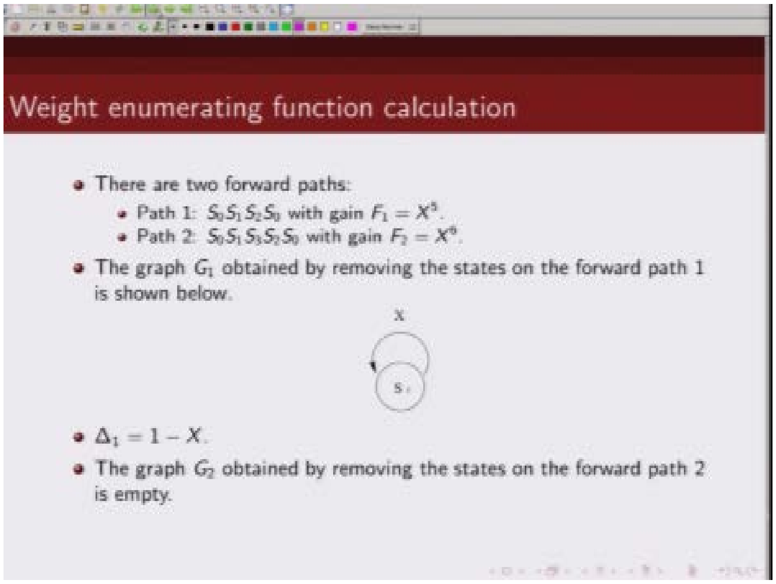
(Refer Slide Time: 26:56)



(Refer Slide Time: 26:57)

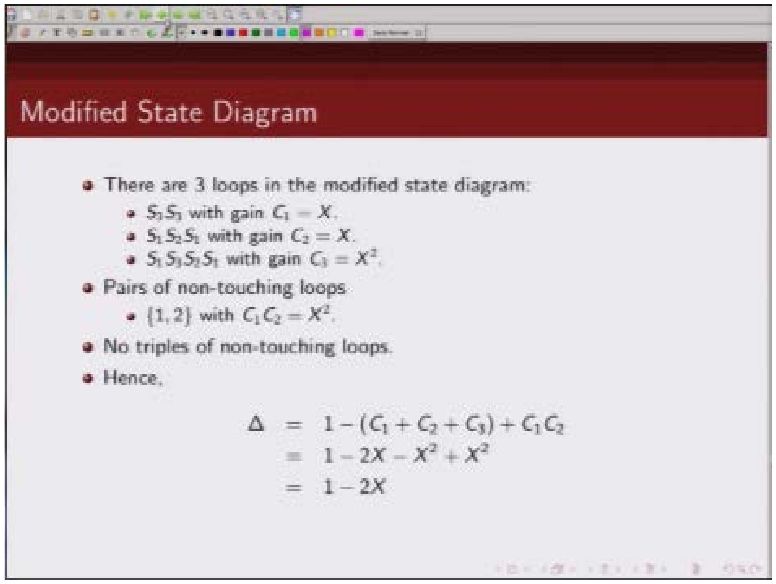


(Refer Slide Time: 26:58)

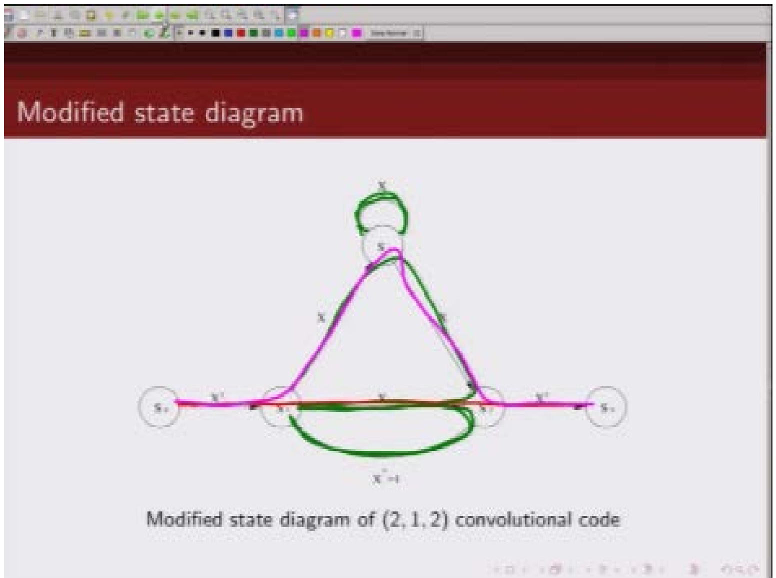


So let us just go back to the original state diagram.

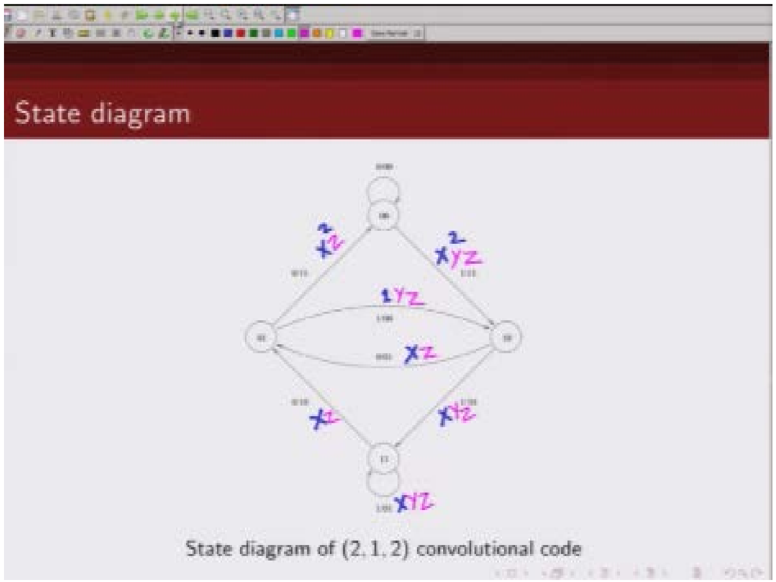
(Refer Slide Time: 27:01)



(Refer Slide Time: 27:05)



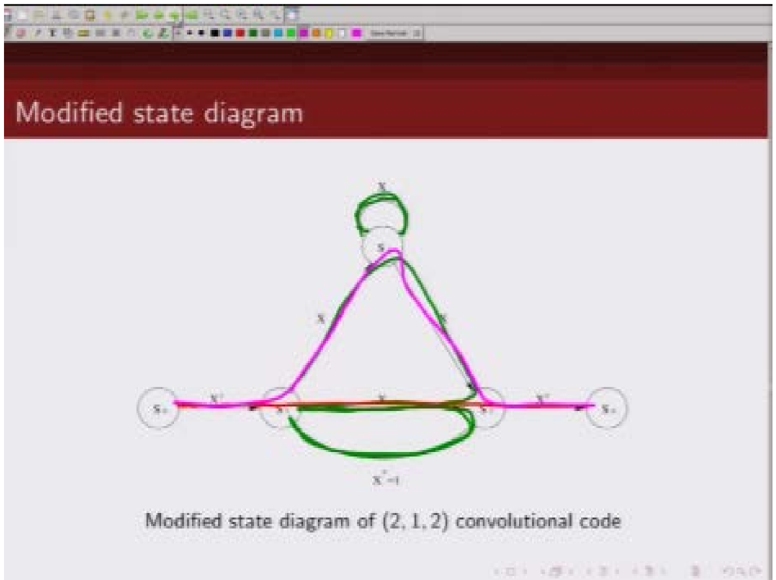
(Refer Slide Time: 27:06)



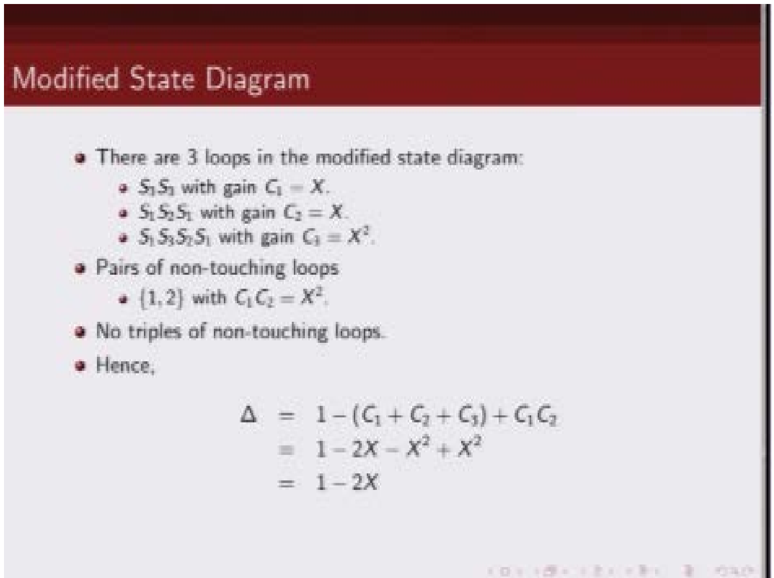
Yeah, let us go back to this, so you can see for this transition from 01 to 00, what is the weight of the information sequence that is 0. So y0 that is basically 1, what about this the weight of the information sequence here the input is 1, so this will be y. What is the weight of information sequence that is 1 so this will be y, this is weight information sequence is 0 so y0 is 1.

So wherever you had 1 here you are adding basically y. This is 1 and similarly at each of these transitions where there will be a z added to denote the length okay. So that is your augmented state diagram.

(Refer Slide Time: 28:11)

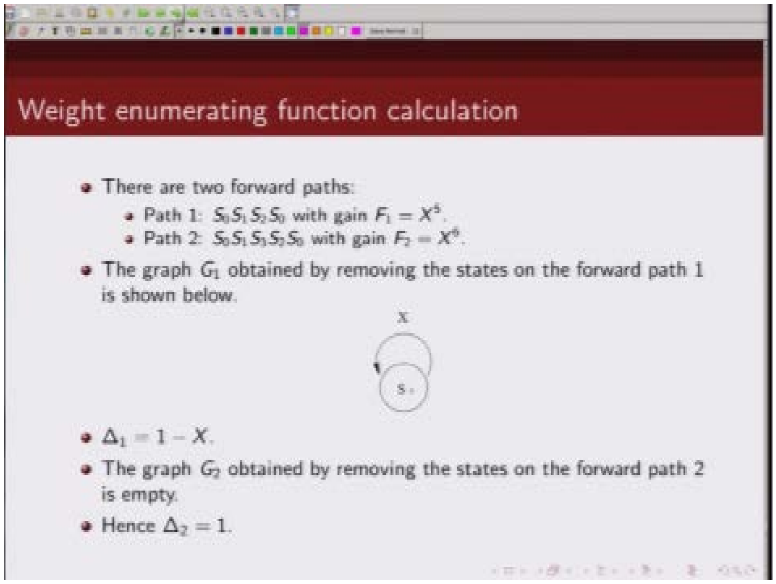


(Refer Slide Time: 28:11)



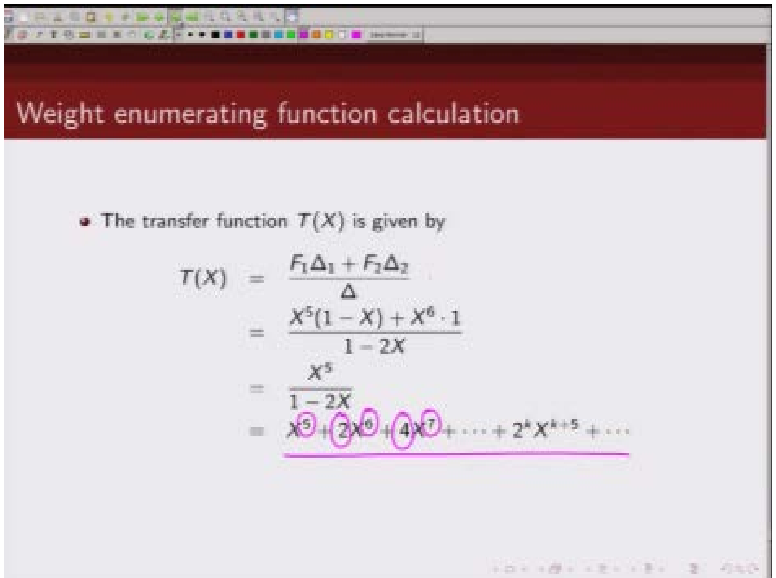
And that is what I mean, the completed augmented state diagram is what.

(Refer Slide Time: 28:15)

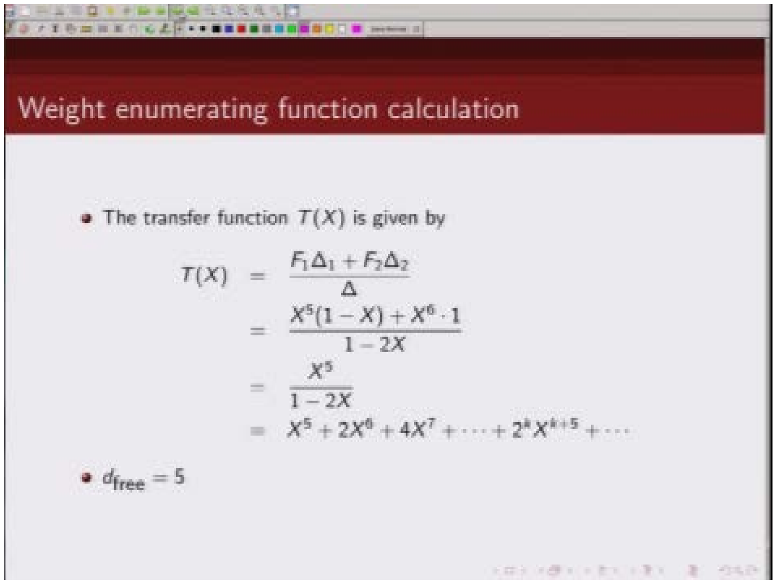


I am showing you here.

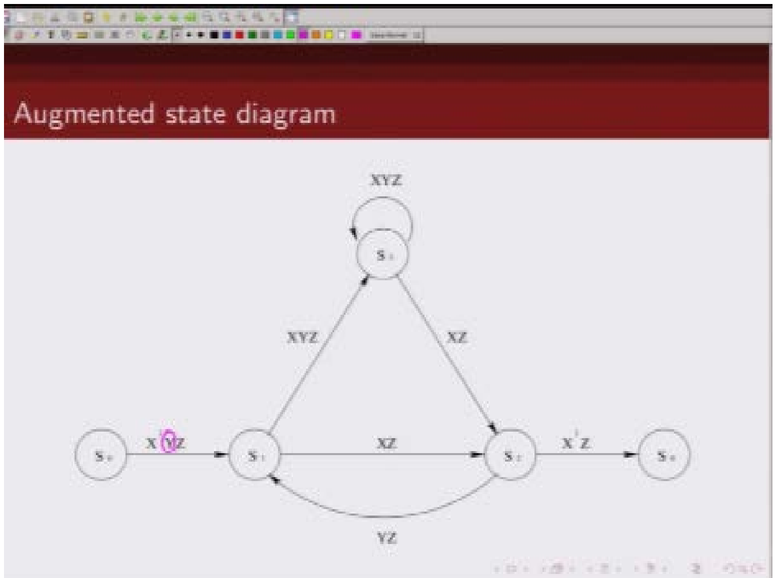
(Refer Slide Time: 28:16)



(Refer Slide Time: 28:17)

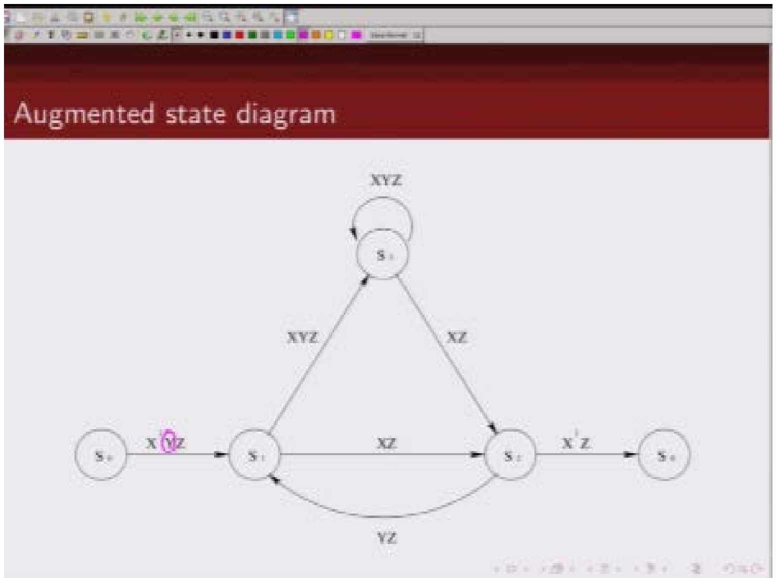


(Refer Slide Time: 28:17)



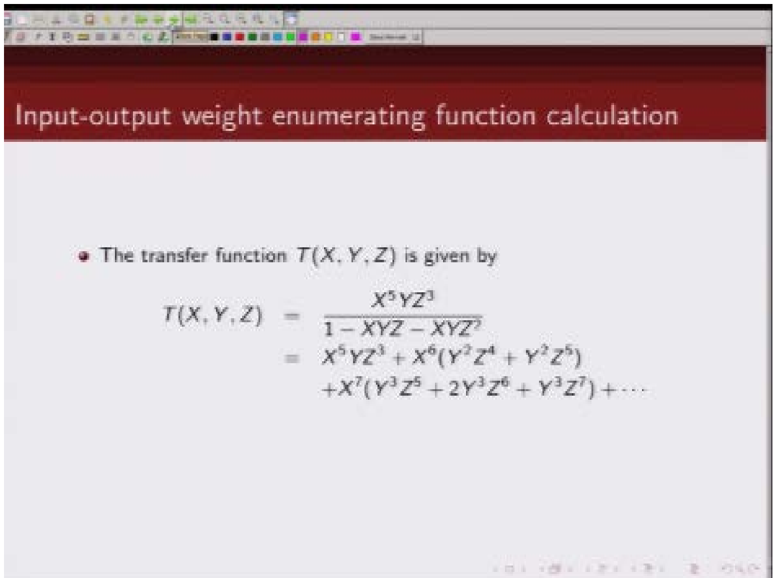
This is basically my augmented state diagram where I am not only specifying the coded weight but I am also specifying what input causes that output bit and z to denote the length. And I follow the same procedure using Mason's gain formula to compute.

(Refer Slide Time: 28:41)



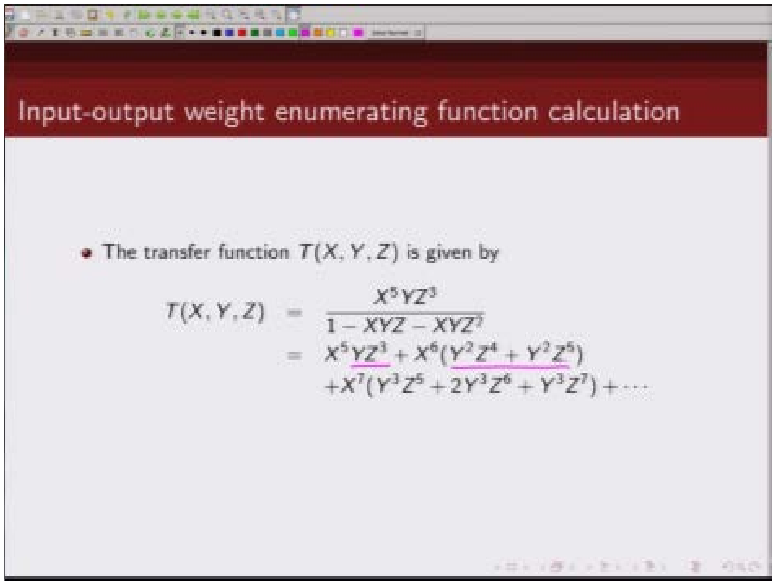
The weight enumerating function.

(Refer Slide Time: 28:42)



So I get this information I am skipping the steps is exactly the same procedure I just laid out for computing the weight enumerating function, and you can see it gives us lot more information.

(Refer Slide Time: 28:58)



The weight enumerating function said we had one code word of weight 5. Now it says that code word of weight 5 basically was caused by message information bit 1 and the length of the [indiscernible][00:29:18] from all 0 state before it merge with all 0 state was 3. Similarly there we have shown that there were two code words of weight 6; this completely specifies what those two code words was.

One which was generated by message bit v2 of length 4 this was message bit to length 5. So you can see at the augmented state diagram, if we use it to generate the transfer function it gives us lot more information. So with this I will conclude this lecture. Thank you.

<u>Acknowledgement</u>

Ministry of Human Resource & Development

Prof. Satyaki Roy

Co-ordinator, NPTEL IIT Kanpur

NPTEL Team

Sanjay Pal

Ashish Singh

Badal Pradhan

Tapobrata Das

Ram Chandra

Dilip Tripathi

Manoj Shrivastava

Padam Shukla

Sanjay Mishra

Shubham Rawat

Shikha Gupta

K. K. Mishra

Aradhana Singh

Sweta

Ashutosh Gairola

Dilip Katiyar

Sharwan

Hari Ram

Bhadra Rao

Puneet Kumar Bajpai

Lalty Dutta

Ajay Kanaujia

Shivendra Kumar Tiwari

an IIT Kanpur Production