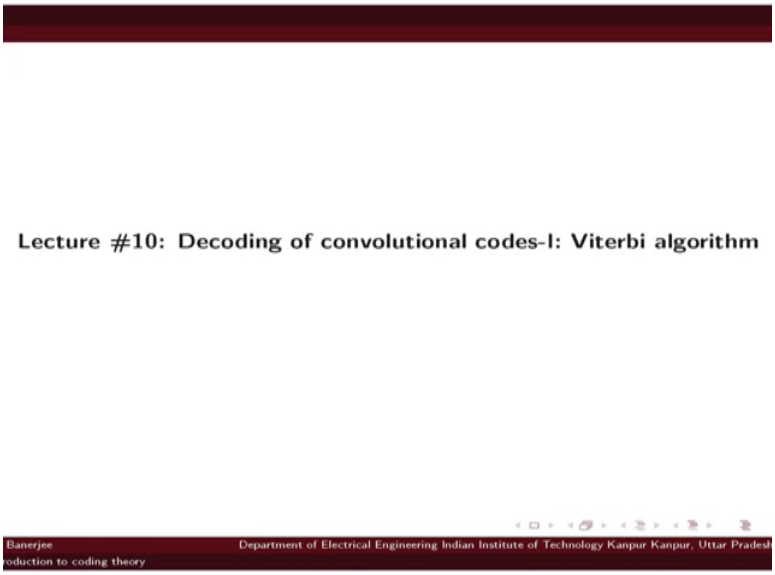
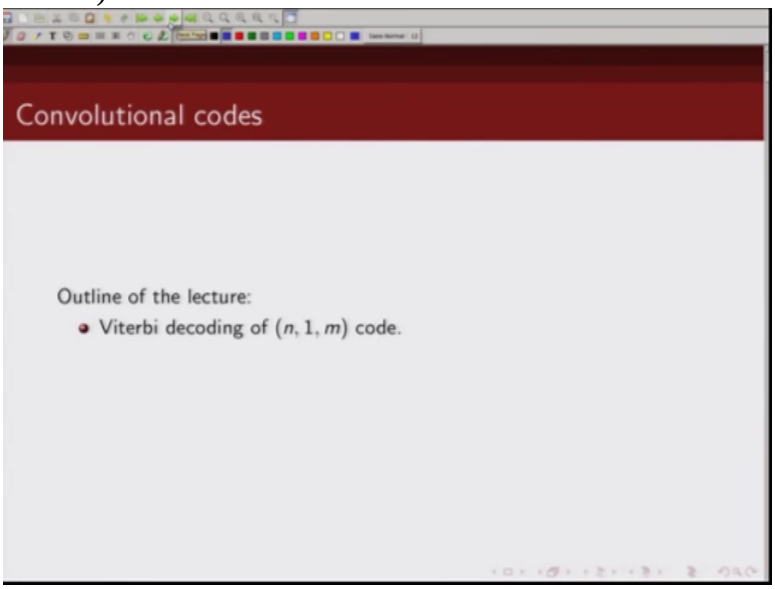
An Introduction to Coding Theory
Professor Adrish Banerji
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
Module 04
Lecture Number 19
Decoding of convolutional codes-I Viterbi algorithm

(Refer Slide Time 00:14)



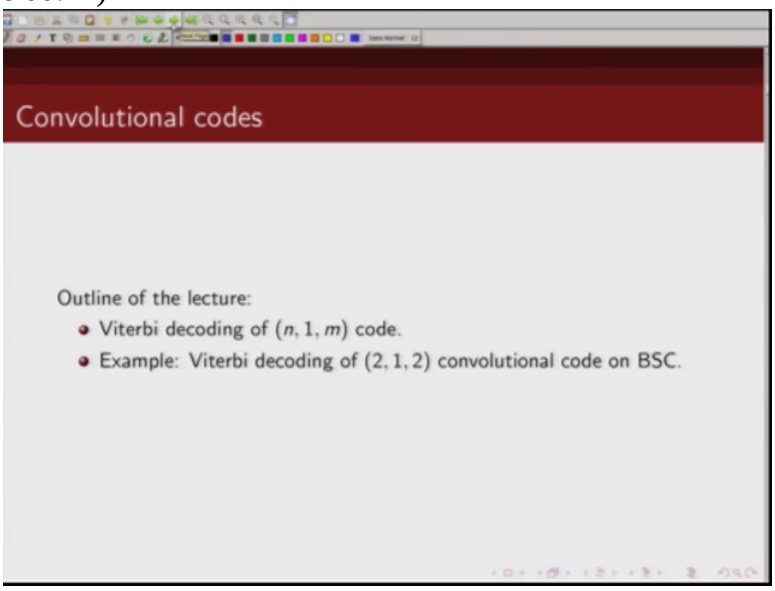
Today we are going to discuss how to decode convolutional code. And in this regard we are going to talk about a maximum likelihood decoding algorithm known as Viterbi algorithm for decoding of convolutional codes.

(Refer Slide Time 00:32)



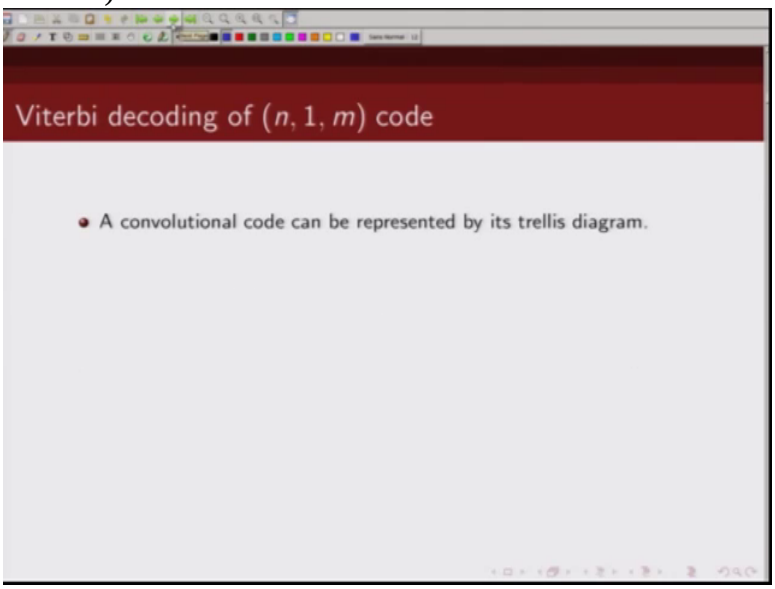
So we are going to illustrate Viterbi decoding of a rate 1 by n convolutional code whose memory order is m

(Refer Slide Time 00:44)



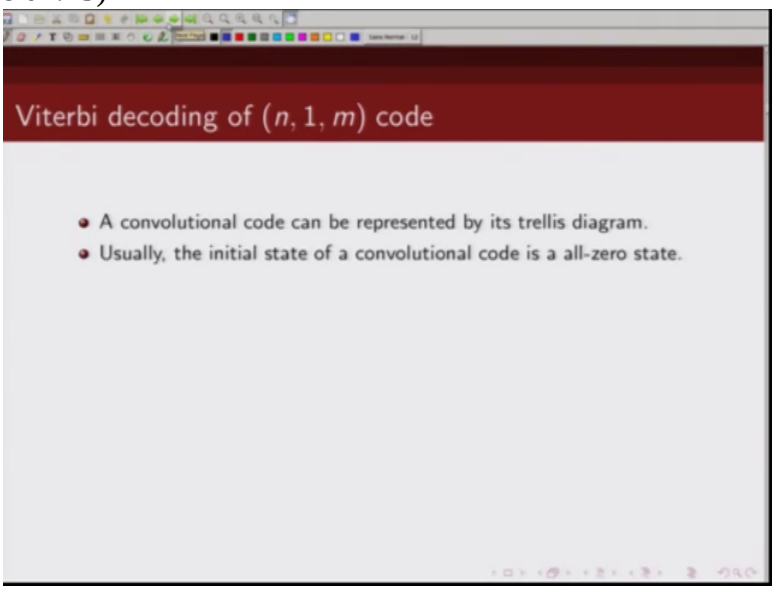
and we are going to illustrate this decoding using a simple example of rate 1 by 2 convolutional code whose memory order is 2. In other words it has 4 states and we are going to assume that our transmission medium channel is a binary symmetric channel.

(Refer Slide Time 01:12)



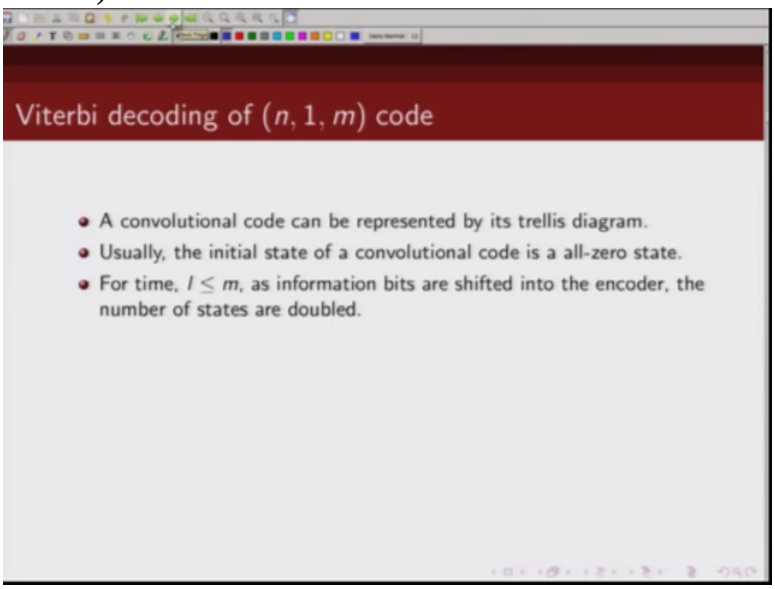
So as we know a convolutional code can be represented by its corresponding state diagram or Trellis diagram. And we are

(Refer Slide Time 01:23)



going to make use of the Trellis representation of the convolutional code to decode it. So initially the convolutional code is assumed to be in all zero state.

(Refer Slide Time 01:39)

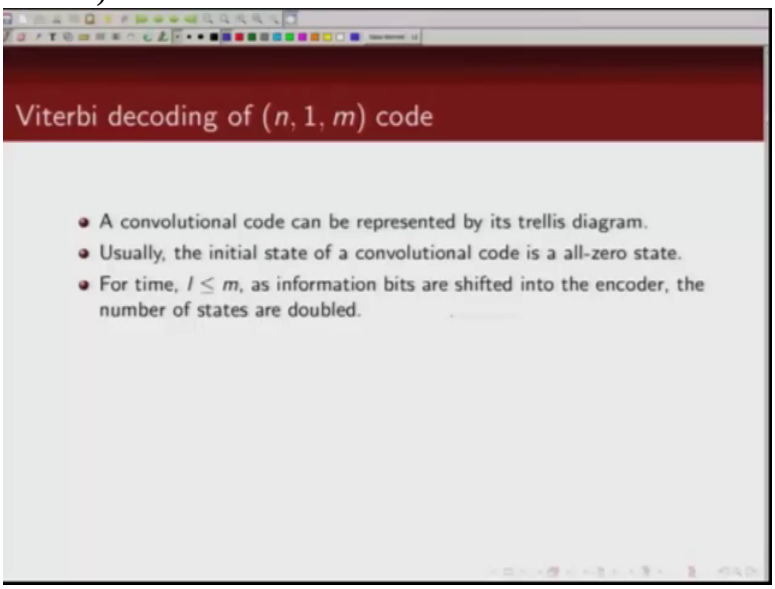


So for time l which is less in the memory order we notice that the information bits are getting shifted in the encoder one bit at a time and the number of states get doubled. So initially it is an all zero state, then once

(Refer Slide Time 02:00)

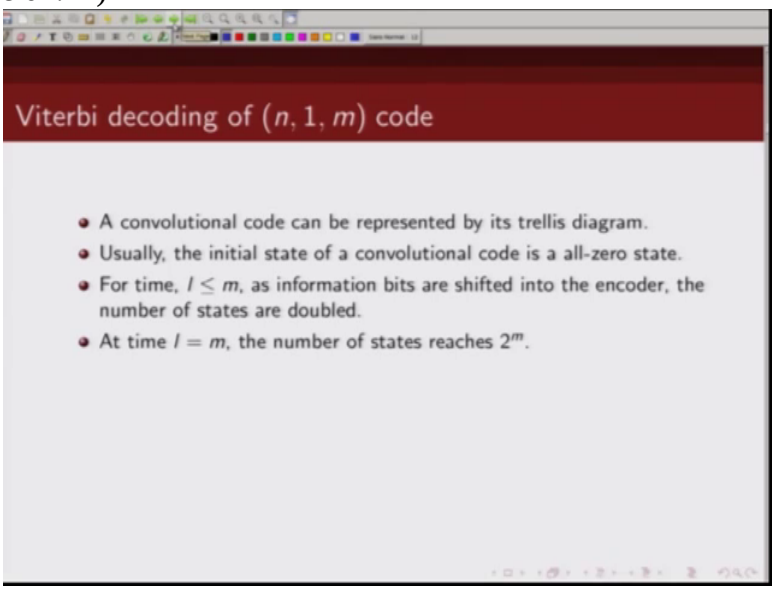
one bit get in, there are 2 possibilities where we can be in. And so as long as

(Refer Slide Time 02:10)



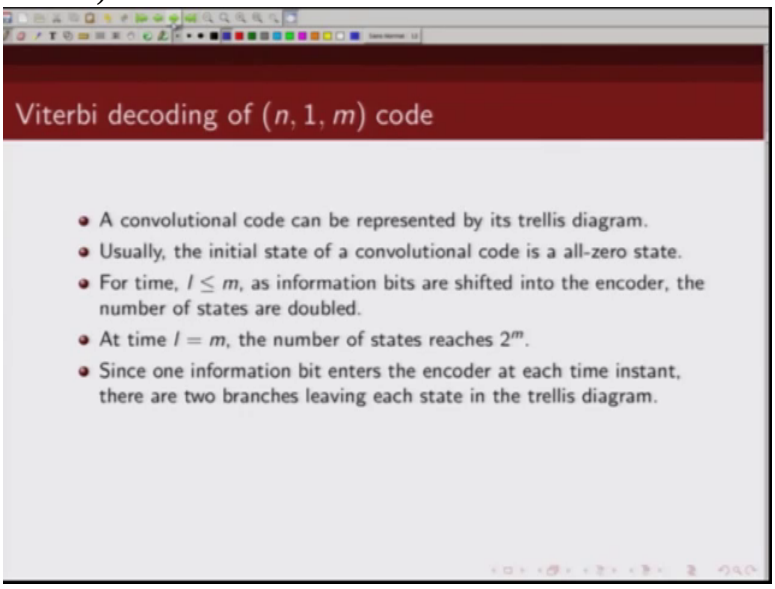
l is less than memory order we see, each time instance the number of possibilities of number of states gets doubled until

(Refer Slide Time 02:21)



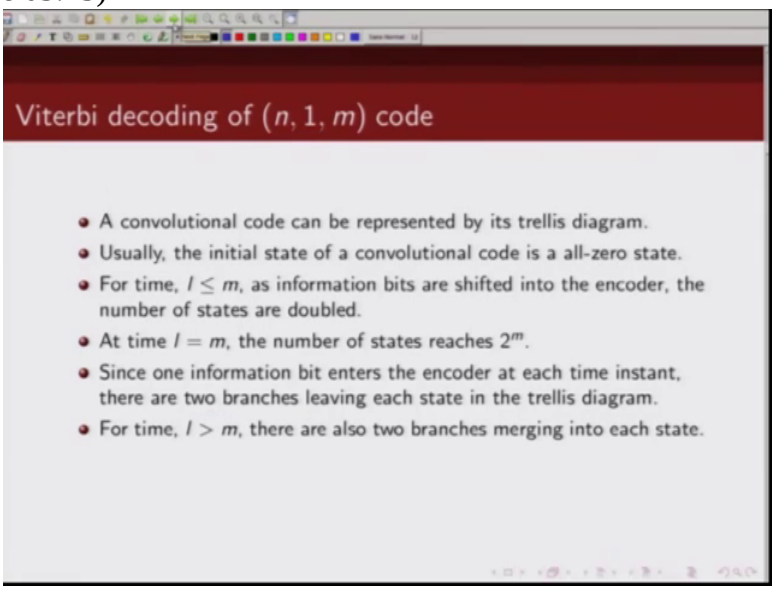
we reach time l equal to m when we reach all possible states which is basically 2 raised to power m because our convolutional code has memory of m. So total it has 2 raised to power m states. So

(Refer Slide Time 02:43)



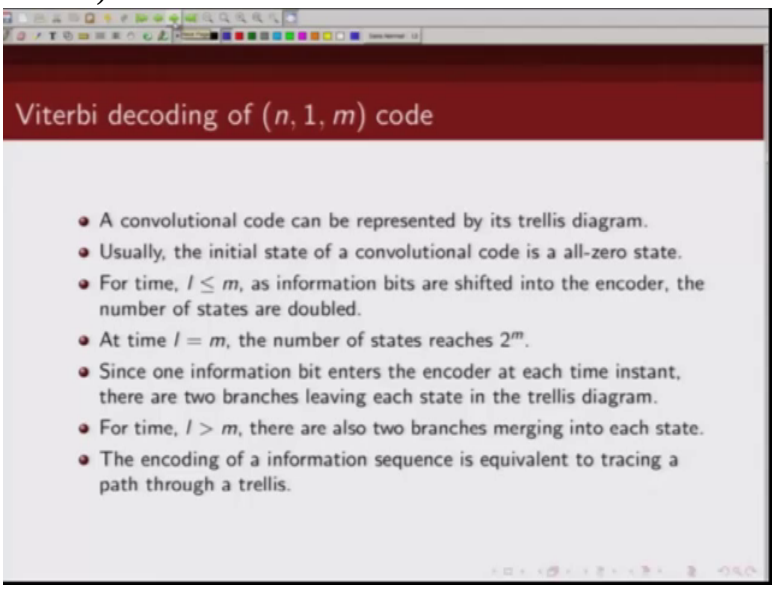
since we are considering a rate 1 by n code so k is 1, so each time instance we have one information bit that enters the encoder at each time. So there are 2 possible branches leaving each state, one corresponding to information bit zero, other corresponding to information bit 1. So in our Trellis diagram we will see that there are 2 branches leaving each state. Now what happens

(Refer Slide Time 03:23)



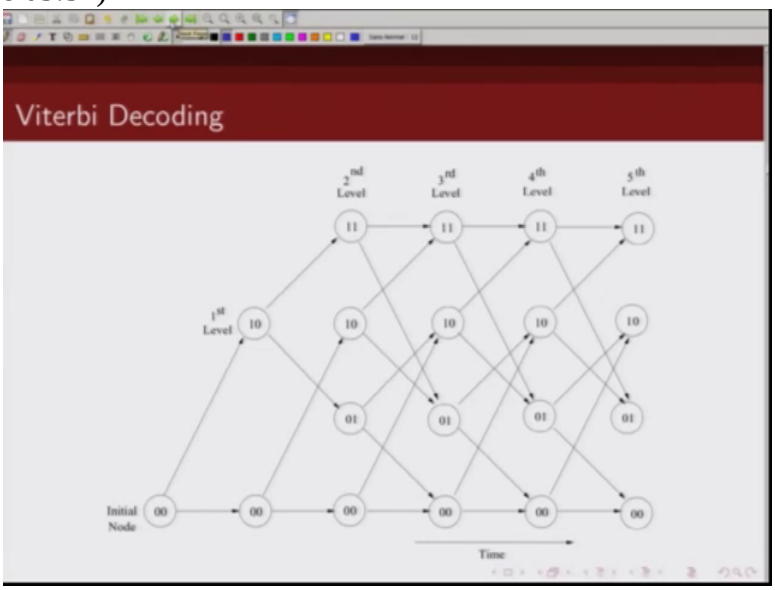
when the time instance is more than the memory order? We will see that two branches are merging into each state. And

(Refer Slide Time 03:35)



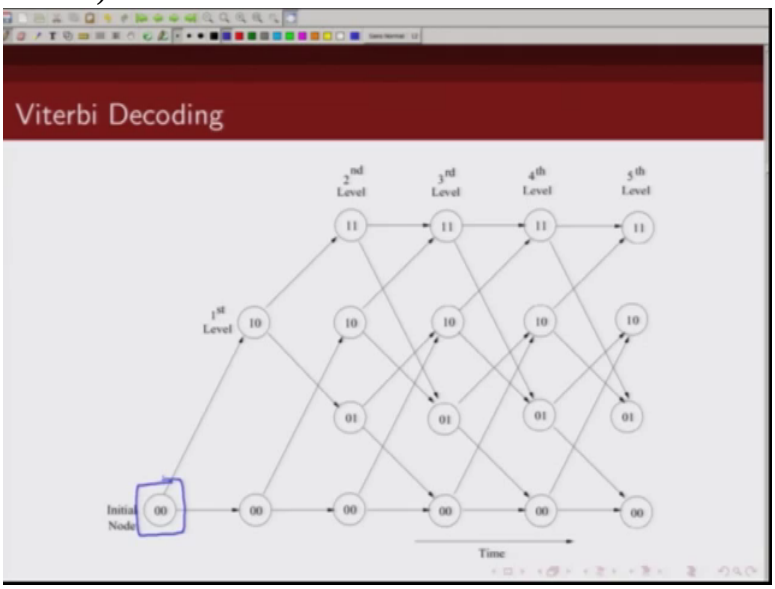
encoding of information sequence can be considered as if we are tracing a path through this Trellis. So any path through this Trellis is a valid codeword.

(Refer Slide Time 03:52)



This is just an example of a Trellis of a convolutional code. So initially, as I said, we start off with all zero state. So initially the encoder is assumed to be in all zero

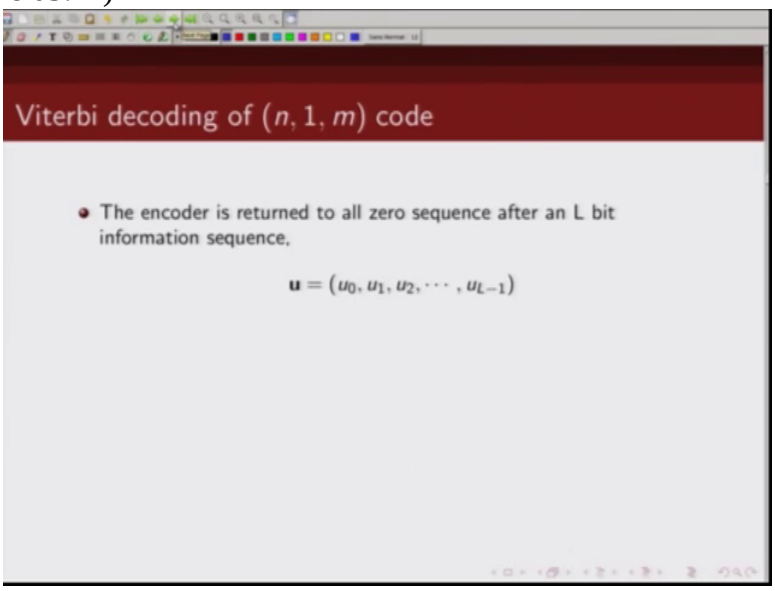
(Refer Slide Time 04:06)



state. So until, say in this particular example, the number of states is 4. You can see 0 0, 0 1, 1 0, 1 1. These are the 4 states. I am denoting them by 0 0, 0 1, 1 0 and 1 1. So until you reach time t equal to 2 which is the memory order of this code we will see that the number of states is getting doubled. Initially there was only 1 state, all zero state. Then you had 2 states and then you had 4 states. And you can see from each state there are 2 branches leaving that particular state. And once your time becomes more than m, then we see that at each state, there are 2 branches which are merging, Ok. I am just denoting, so this is on the x axis is my

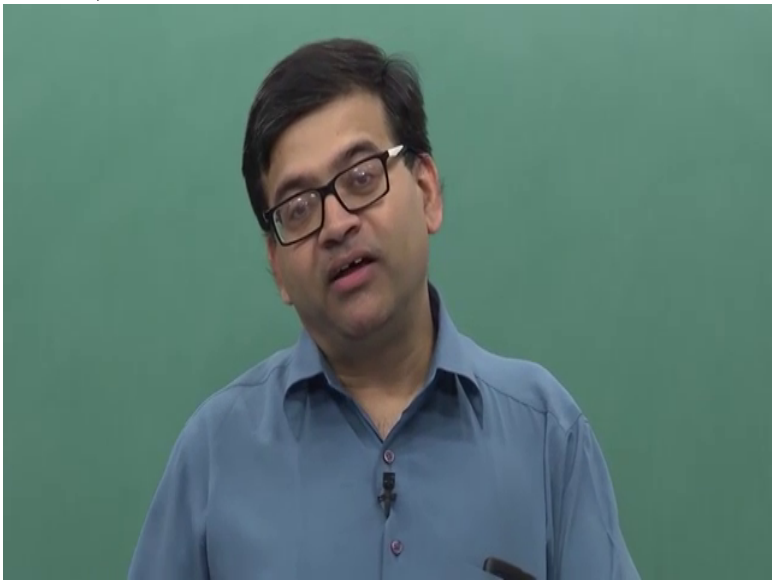
time, I am just denoting them by level, first level, second level, third level, fourth level which is just how the time is progressing.

(Refer Slide Time 05:11)



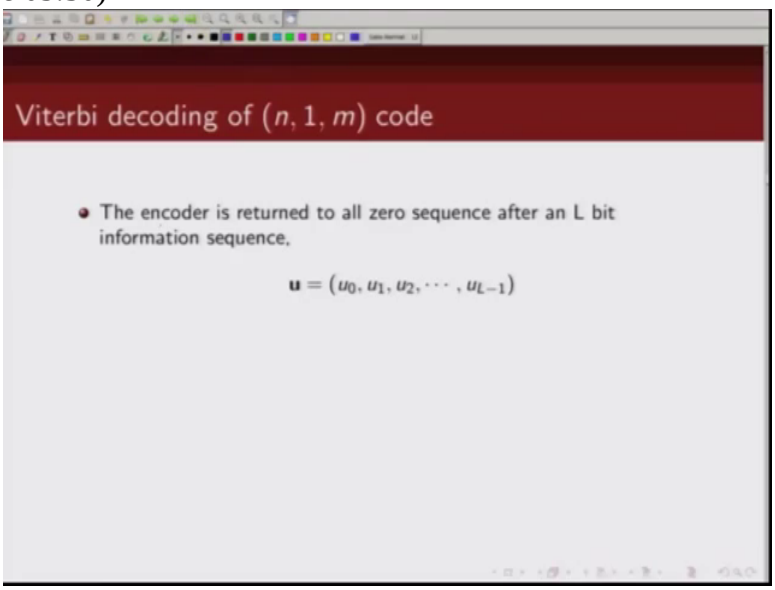
Now usually after we encode the information sequence we bring the, we terminate the convolutional code and what do we mean by terminate the convolutional code? We bring the

(Refer Slide Time 05:25)



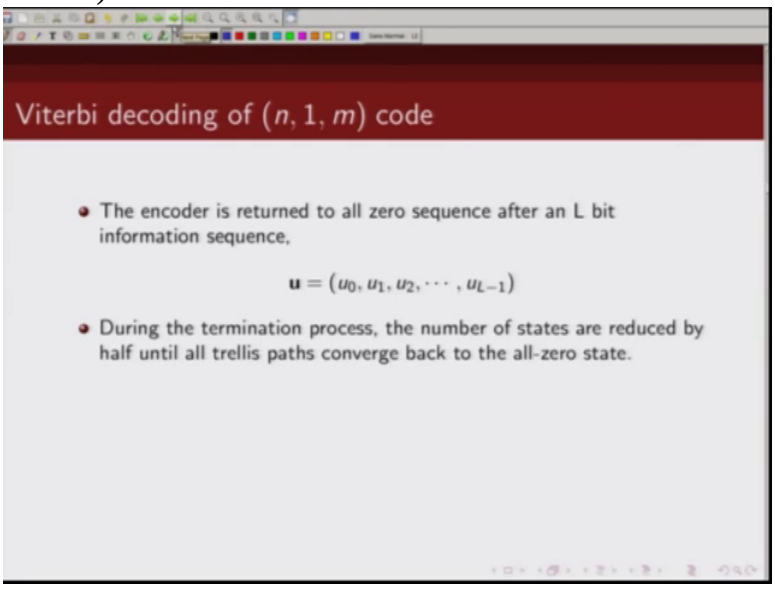
convolutional code back into all zero state. So the encoder is typically

(Refer Slide Time 05:30)



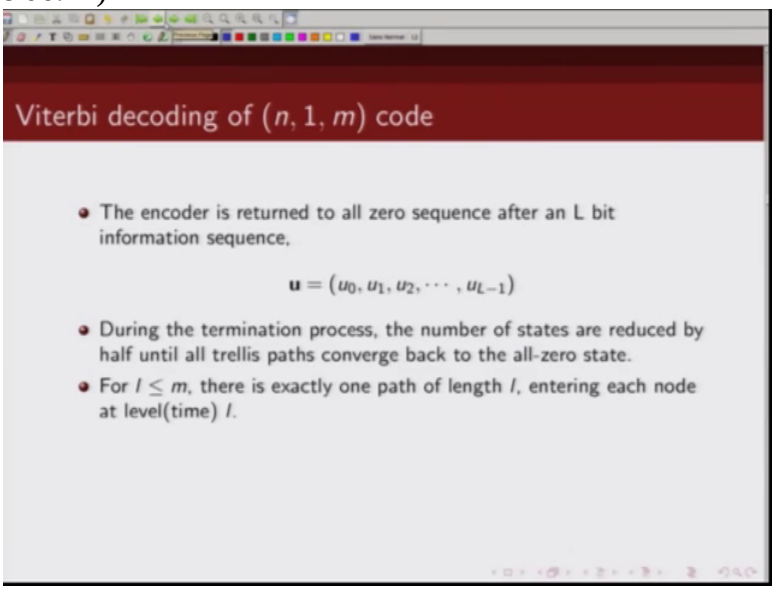
returned to all zero state after we have encoded our l bit sequence. So if you have an information sequence which is l bit denoted by u, so our information sequence is u 0, u 1, u 2, u l minus 1 where these are basically 0s and 1s, now

(Refer Slide Time 05:52)



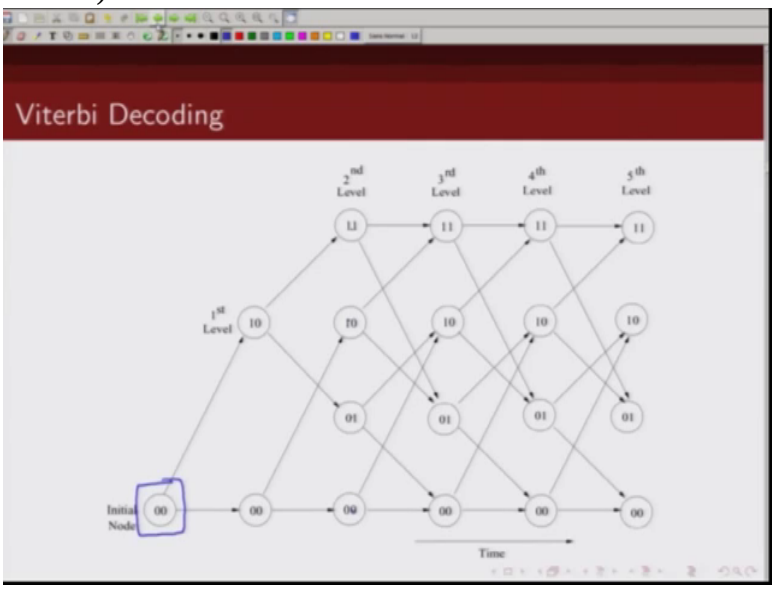
during the termination process, because we are trying to bring it back to all zero state, what happens is number of states get reduced by half until all the Trellis paths will converge to all zero state.

(Refer Slide Time 06:12)



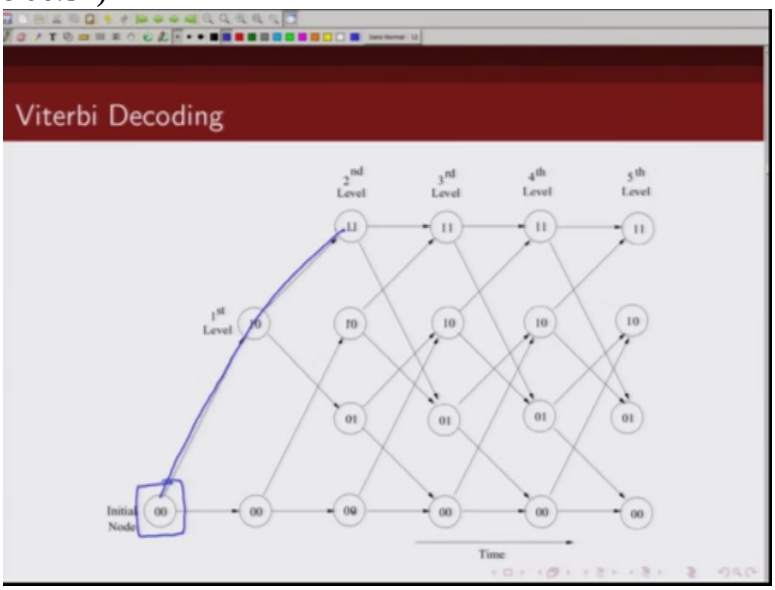
So as we said for time less than memory order, there is only 1 path of length l entering each state, each node you can see, go back to this diagram;

(Refer Slide Time 06:24)



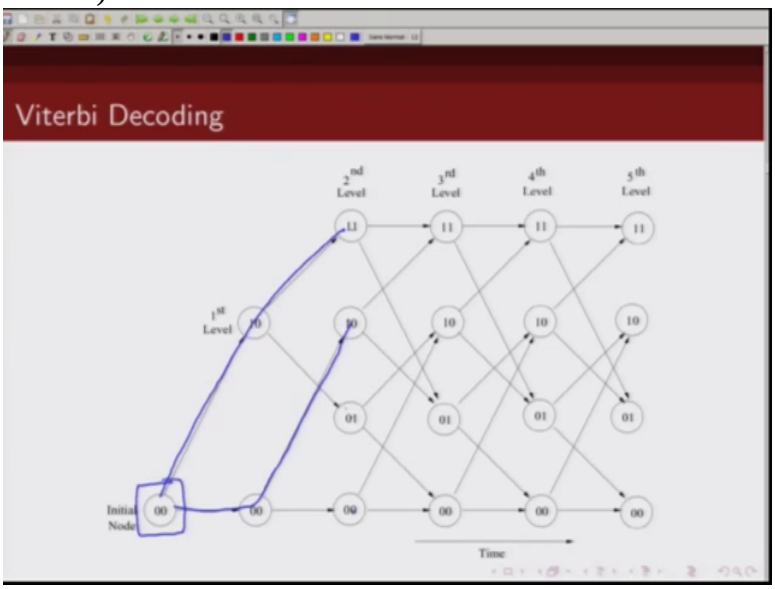
up to this path there is one, let us say I want to reach 1 1 from 0 0, there is only one path through this. I want to

(Refer Slide Time 06:34)



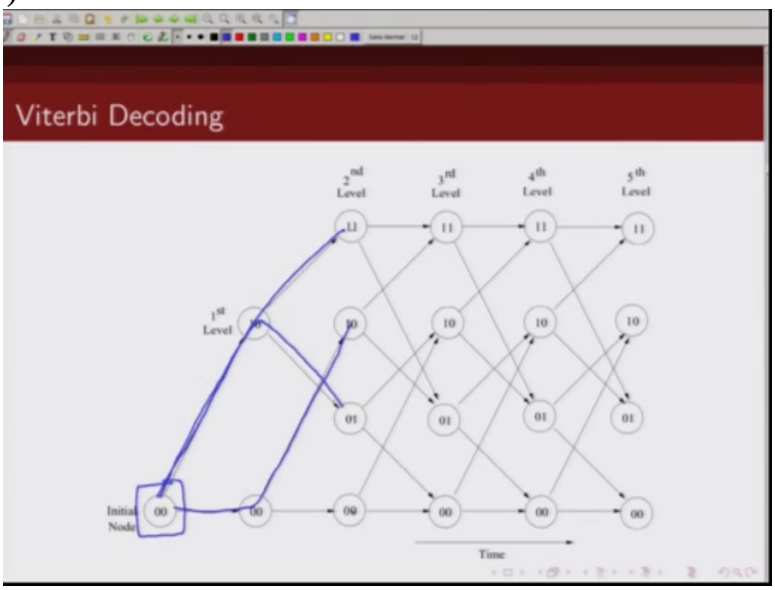
reach here, there is only one path. I want to reach

(Refer Slide Time 06:39)



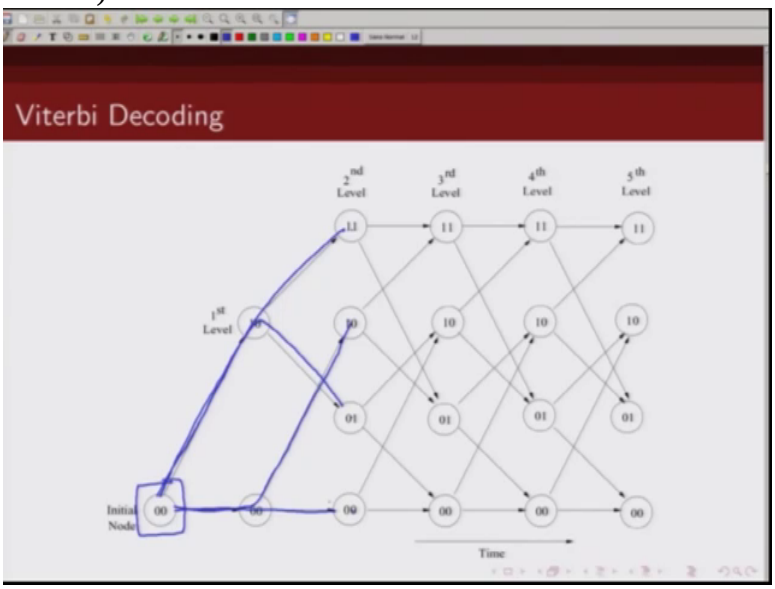
this state. There is only one path.

Slide Time 06:44)



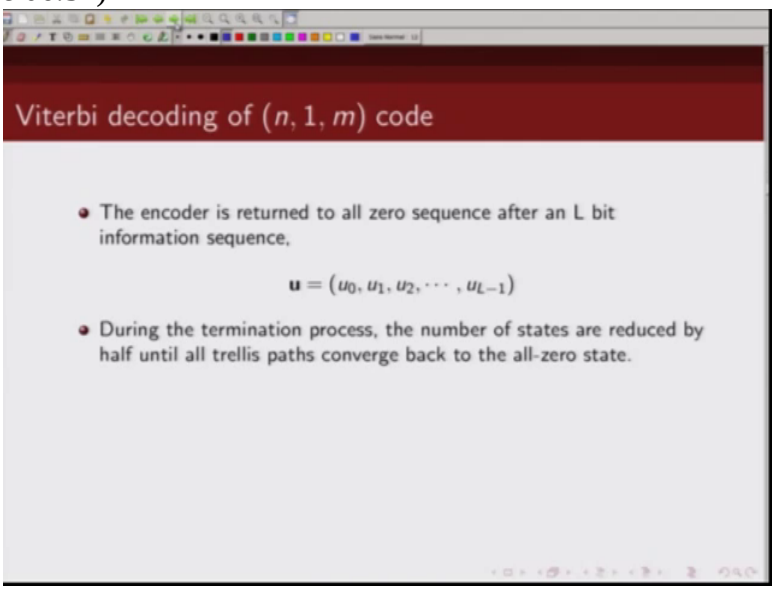
I want to reach this state. There is only one path.

(Refer Slide Time 06:49)

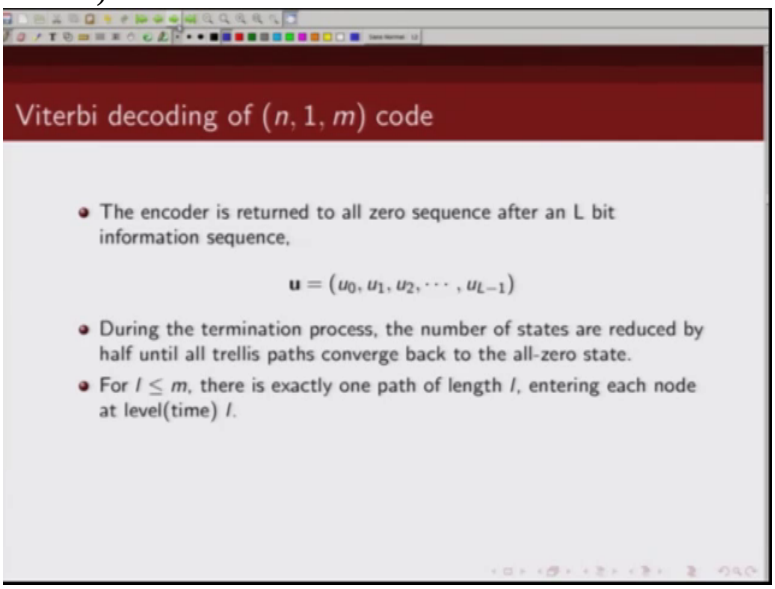


So,

(Refer Slide Time 06:51)

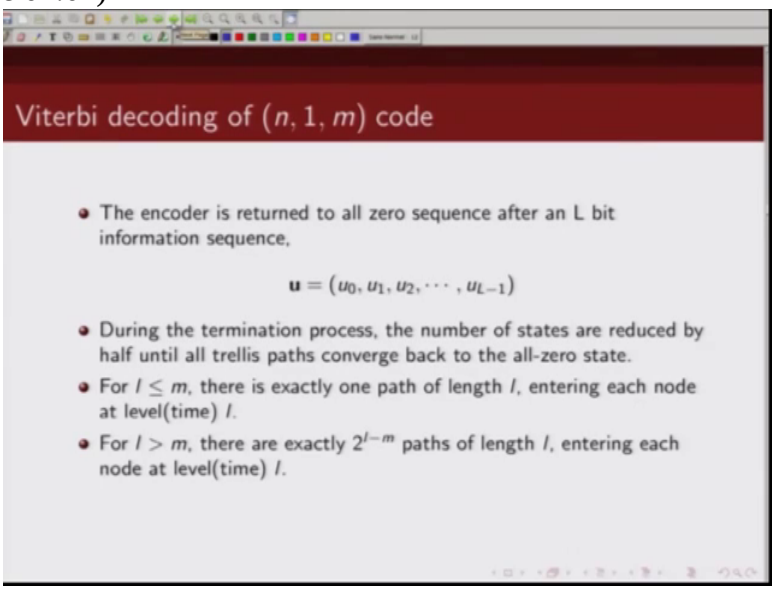


(Refer Slide Time 06:52)



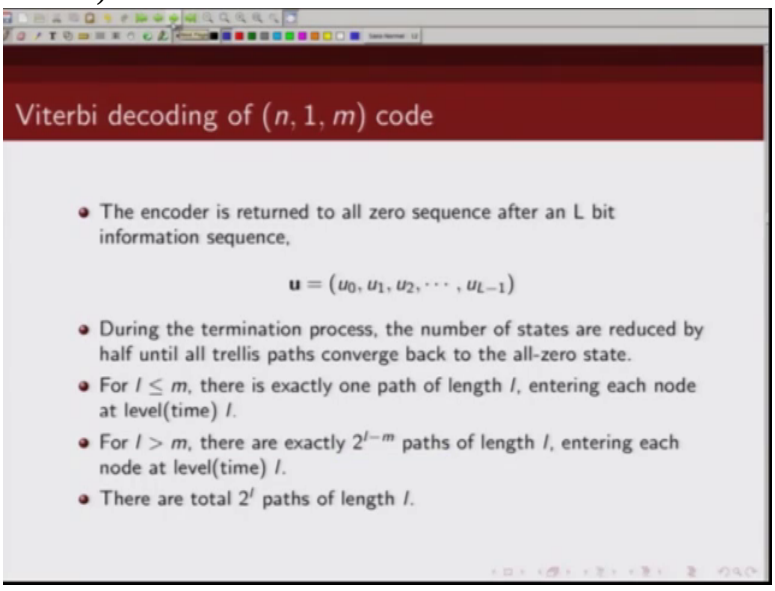
so until time is less than equal memory order there is exactly one path of length l entering each node at each time. And

(Refer Slide Time 07:04)



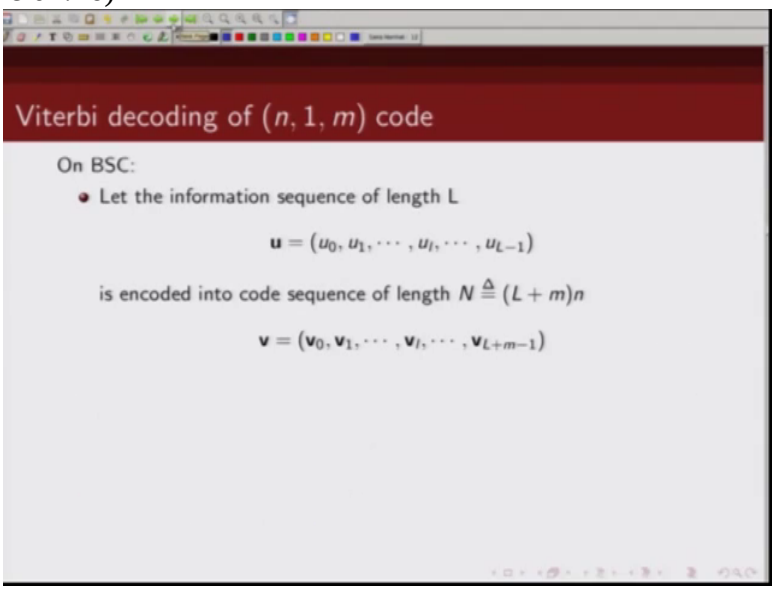
for l greater than m, there are 2 raised to power l minus m paths of length l entering each state at each time instance.

(Refer Slide Time 07:18)



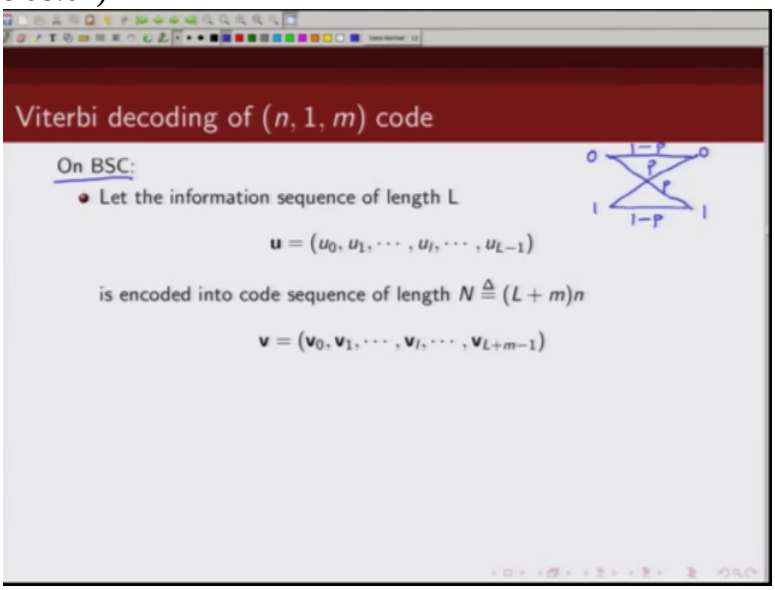
And total there are 2 raised to l paths of length l in this Trellis diagram.

(Refer Slide Time 07:26)



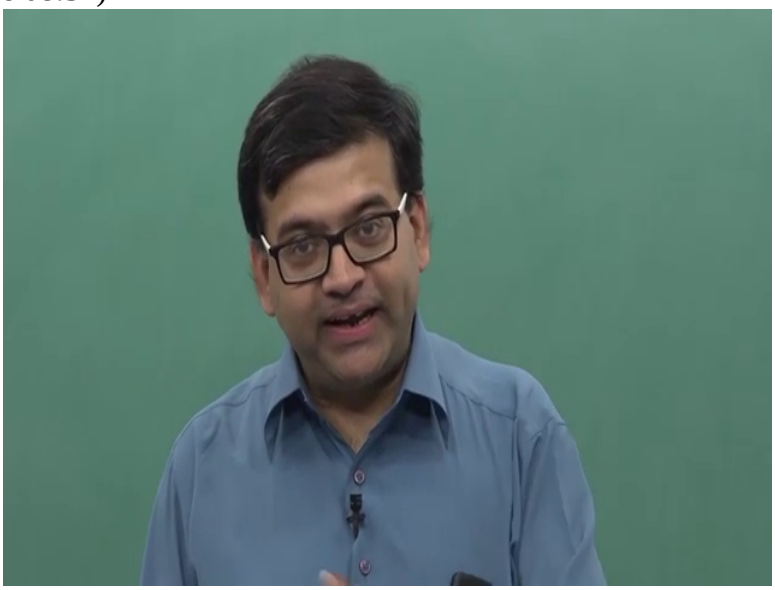
So let us consider how we are going to decode this convolutional code which can be represented by its, by its Trellis diagram. So we are considering a binary symmetrical channel. Now recall what is a binary symmetrical channel. So we have 2 inputs, 0s and 1, similarly we have 2 outputs, 0 and 1. So there is a 1 minus p probability of getting the bits correctly and then there is a cross over probability p of bits getting flipped.

(Refer Slide Time 08:02)



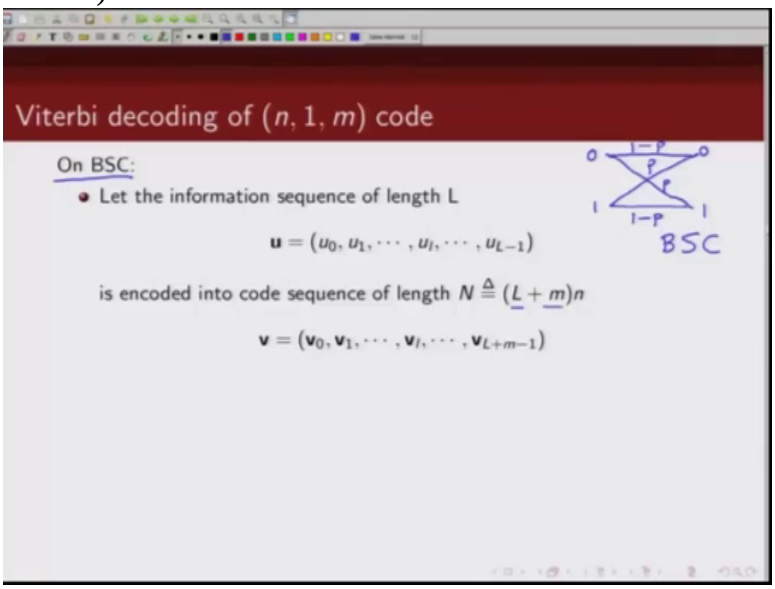
So this is our binary symmetric channel. So let us consider that we have information sequence of length l denoted by u and it has been encoded into a sequence length n. It is coded using a rate 1 by n code. Now l corresponds to number of information bits. Now remember we are terminating the

(Refer Slide Time 08:34)



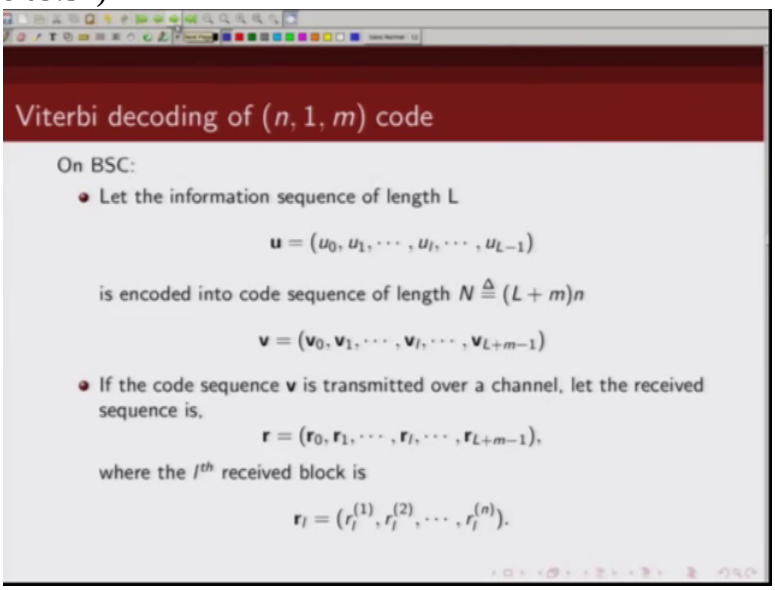
convolutional encoders so we require

(Refer Slide Time 08:38)



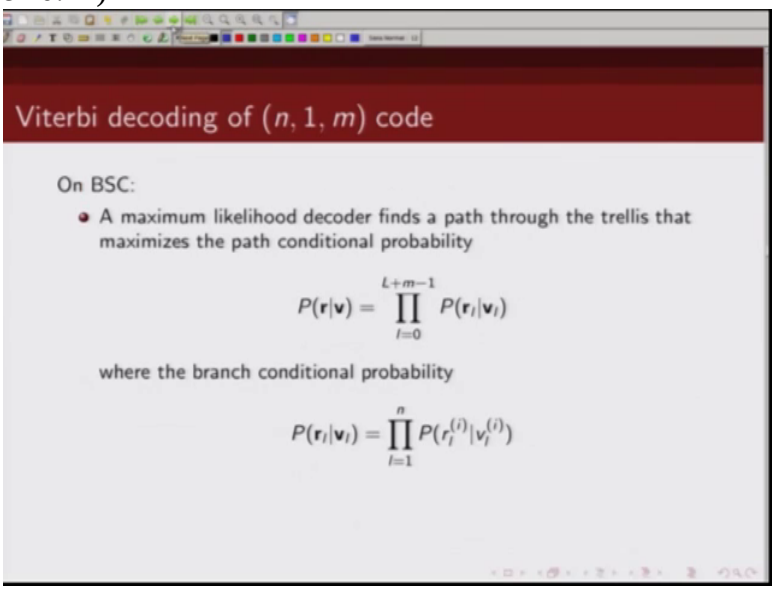
m number of bits to terminate it, to bring it back to all zero state. So total basically number of bits including the tail bits or termination bits will be l plus m and since it is a rate 1 by n code the codeword, code sequence length will be given by l plus m into n. So our code sequence is of the length l plus m denoted by v 0, v 1, v 2 , v l plus m minus n. And where each of them v 0, v 1, v l are basically n bits. So if you

(Refer Slide Time 09:31)



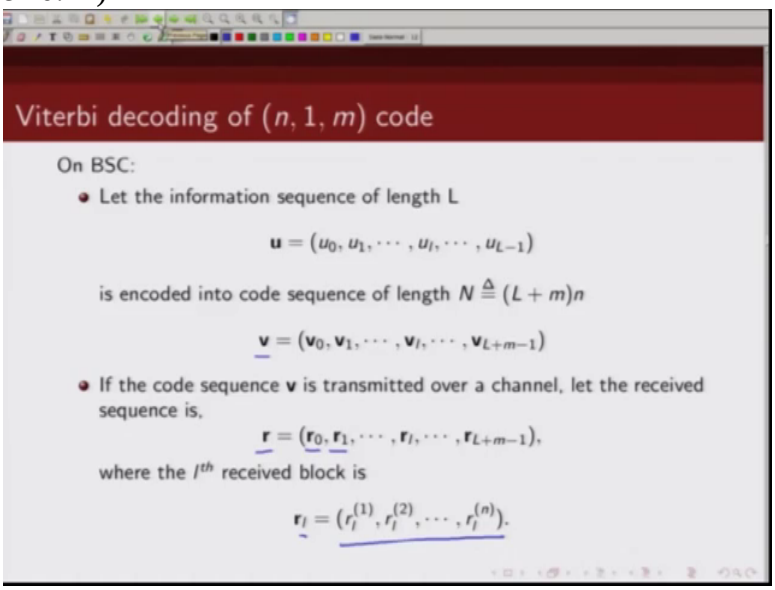
transmit this code sequence over this binary symmetrical channel what we receive is denoted by, received sequence is denoted by r. So we receive r 0, r 1, r 2, r l plus m minus 1 where each of these r i's are consisting of n bits, Ok. So each of these r i's are basically this n bit vector. This is corresponding to the transmitted code sequence. So this r is the received code sequence corresponding to the codeword or the code sequence that we have transmitted.

(Refer Slide Time 10:12)



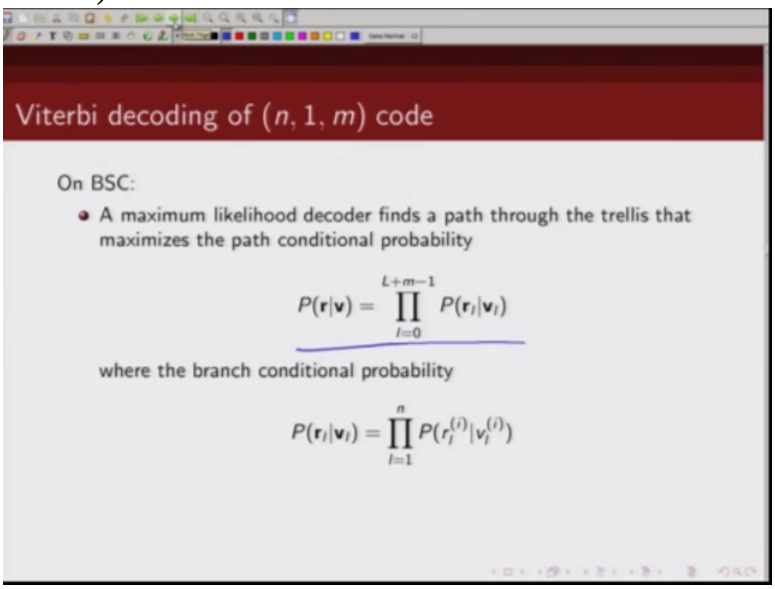
Now we know that a maximum likelihood decoder will try to maximize the likelihood function. So a maximum likelihood decoder is going to find a path through this Trellis that will maximize the path conditional probability so it will maximize the likelihood function. Now we can define the probability of r given v. Now what is this r? r consists of

(Refer Slide Time 10:44)



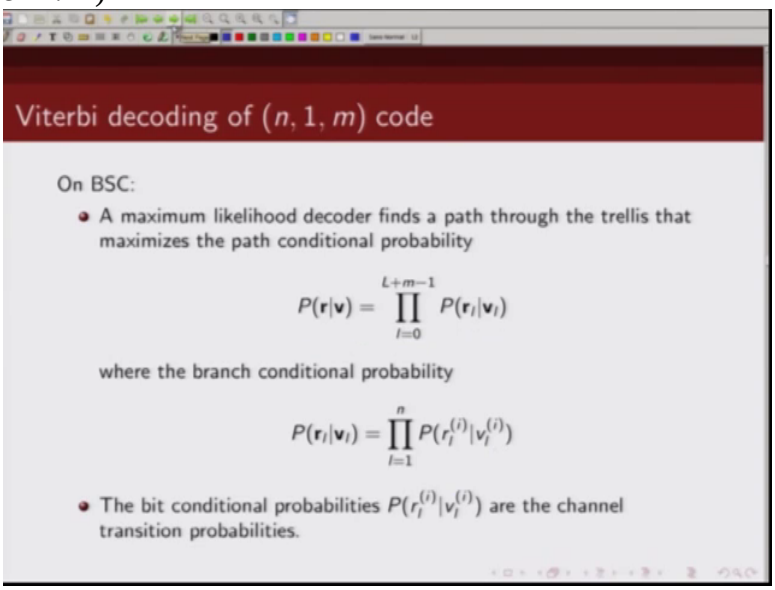
these l plus m elements and each of these r i's are for the n bit vector. So we, since it is a memoryless

(Refer Slide Time 10:58)



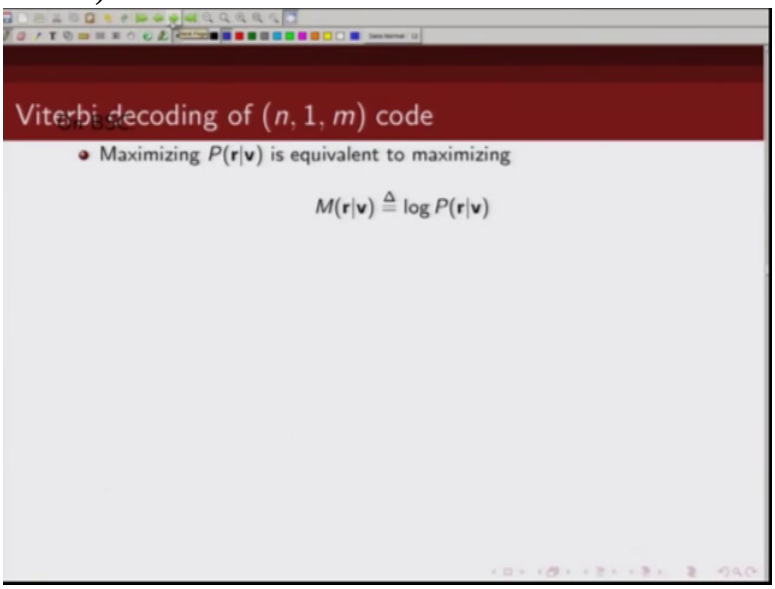
channel, we can write this probability of r given v in terms of these r i's and v i's like this and this can be written in terms of the bit likelihood function. So we can write this in terms of r i's and v i's. And this can be written further in terms of the bit matrix which is given by this.

(Refer Slide Time 11:27)



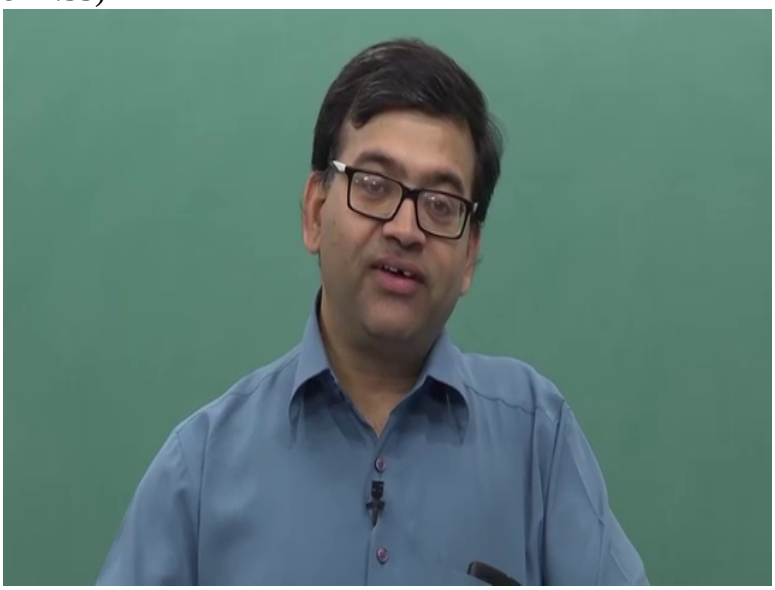
And what are these probabilities? These probabilities are nothing but these are corresponding to the channel transition probabilities.

(Refer Slide Time 11:40)



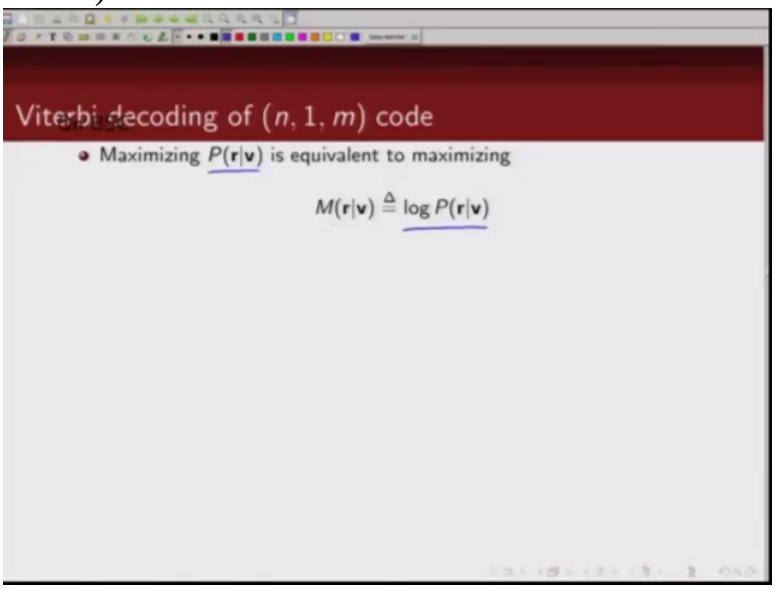
Now maximizing this likelihood function is equivalent to maximizing the log of the likelihood function. Why, because log is a

(Refer Slide Time 11:53)



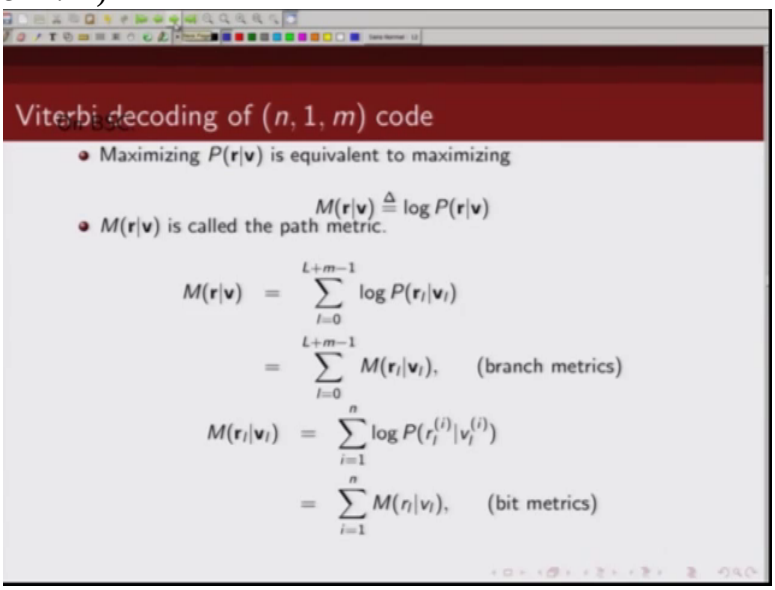
non decreasing function. So

(Refer Slide Time 11:59)



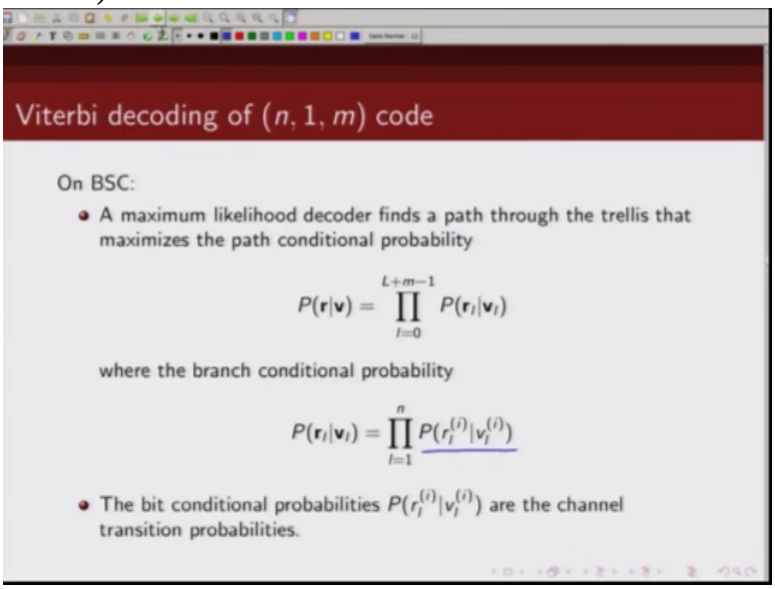
we can equivalently write maximizing this as, our objective as maximizing the log of probability of r given v.

(Refer Slide Time 12:14)



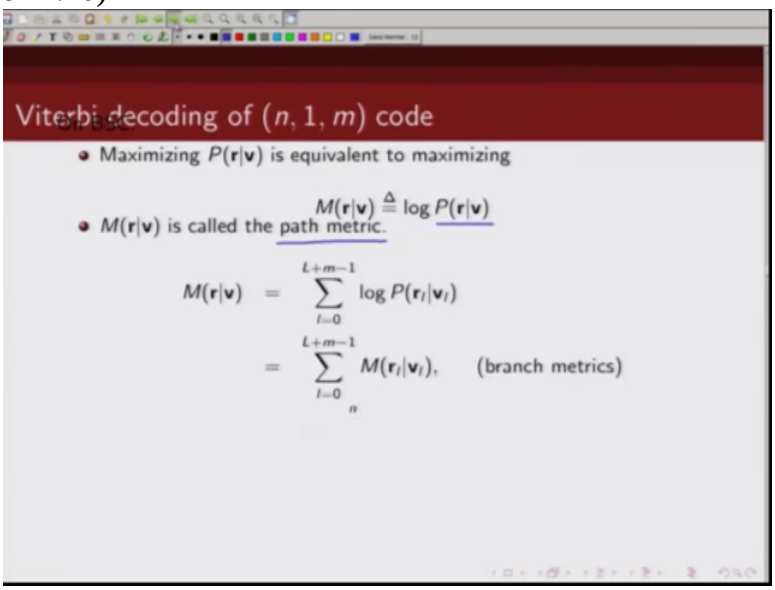
And this is, we call this as path metric. How do we compute the path metric? Go back and look into the expression of probability of r given v.

(Refer Slide Time 12:28)



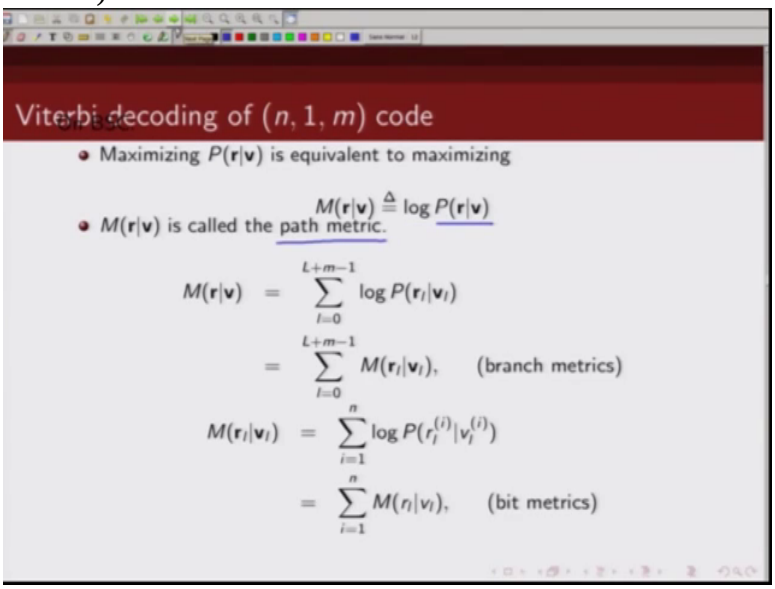
So if we look at probability of r given v it is given by this. So if we take a log of this, this product term becomes summation.

(Refer Slide Time 12:40)



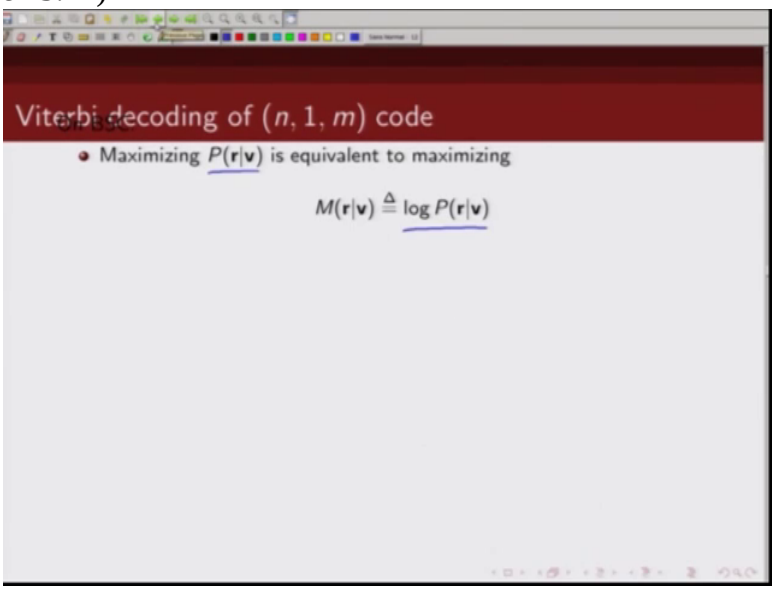
So we can write

(Refer Slide Time 12:41)



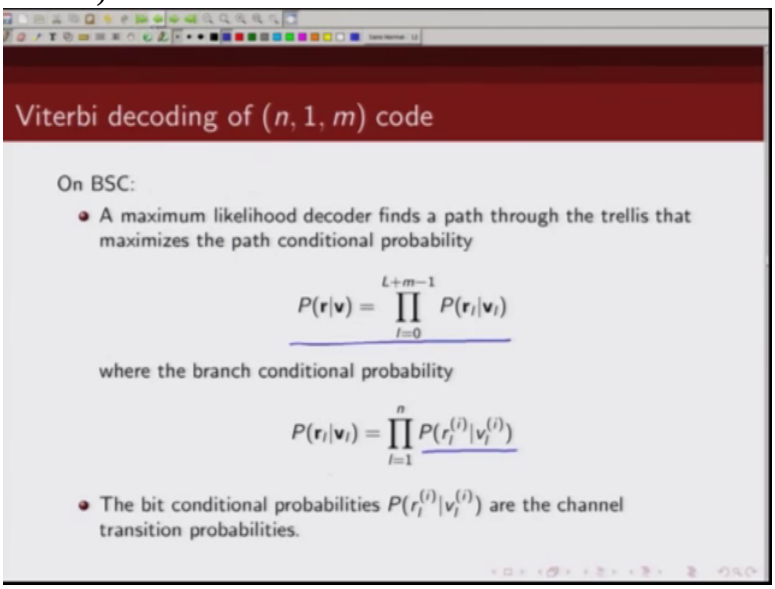
basically them as, so we can write our path metric as summation from 0 to l plus m minus 1 log of probability of r l given v l where l goes from 0 to capital L plus m minus 1. And we are calling this as our branch metric. So this is our branch metric. Further the branch metric can be written in terms of bit metric. Now we know what is the probability of r l given v l. This

(Refer Slide Time 13:22)



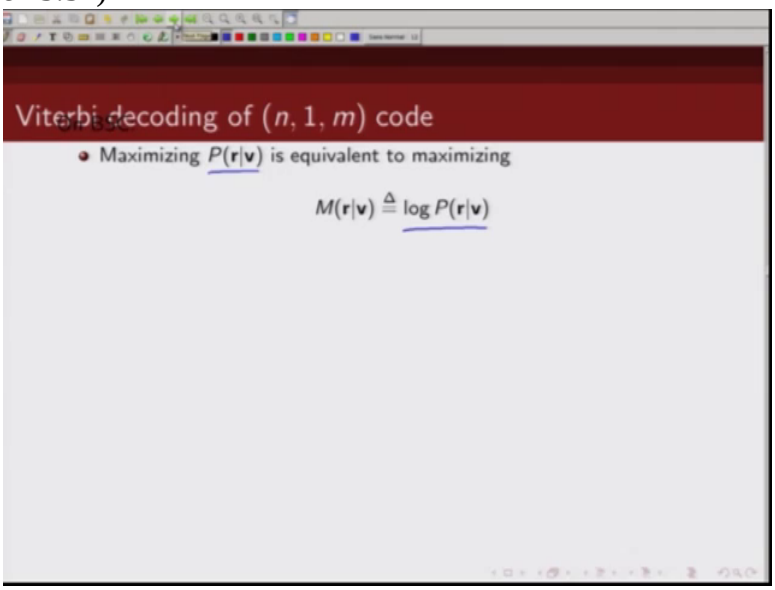
is given by

(Refer Slide Time 13:24)



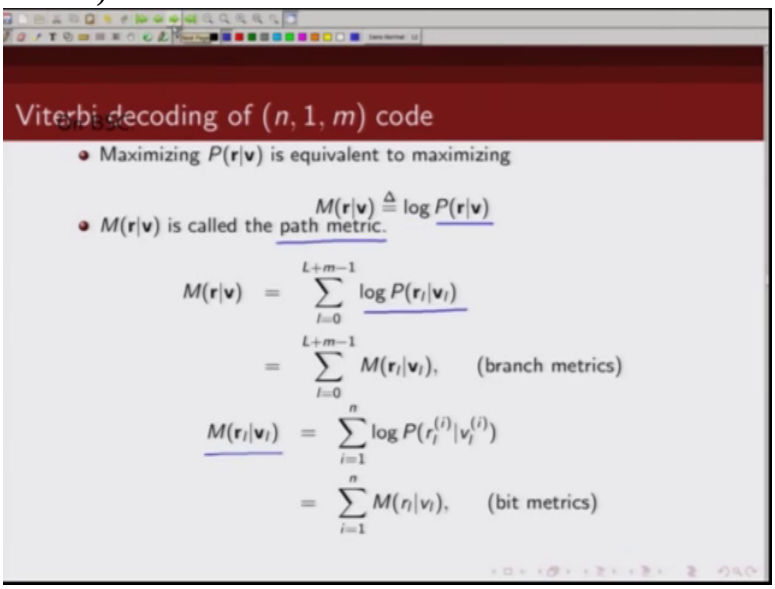
this expression. So if we take log of this, again this product term will become summation, so what we would get is

(Refer Slide Time 13:34)



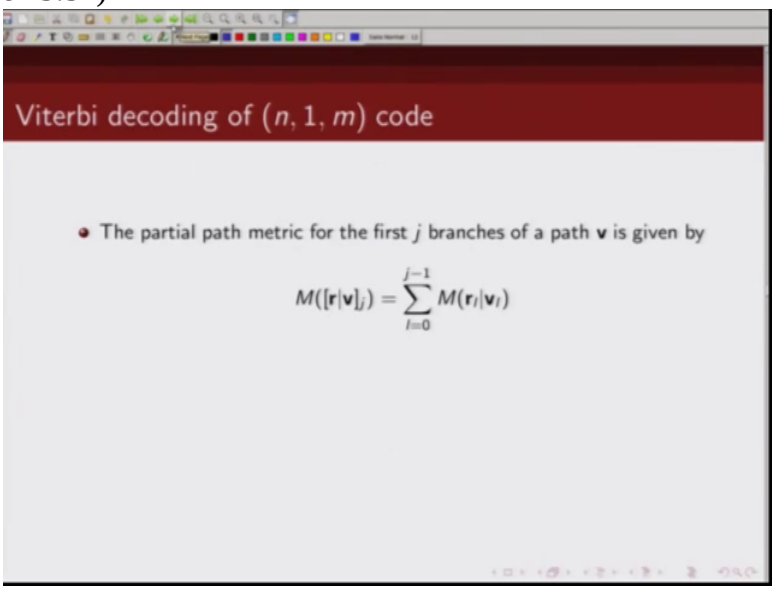
this.

(Refer Slide Time 13:36)



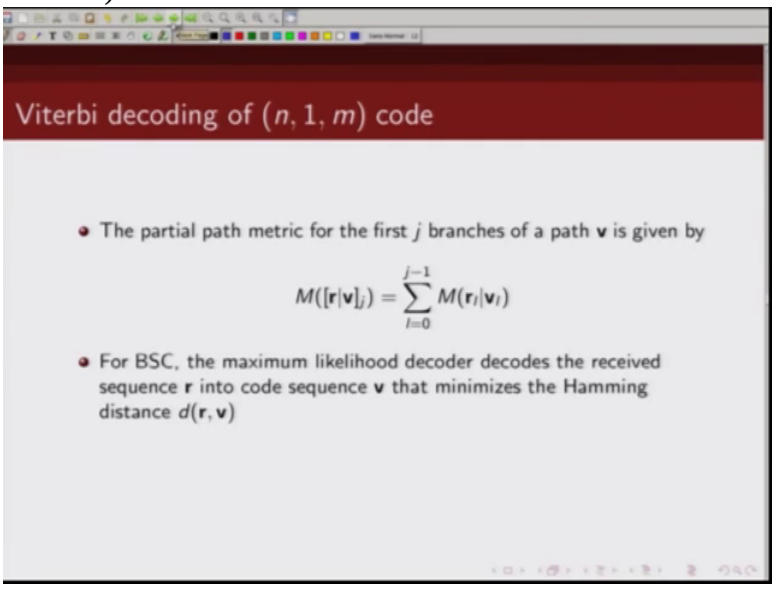
So you can see we can conveniently write path metric as sum of branch metric and branch metric we can write this as sum of these bit metric.

(Refer Slide Time 13:51)



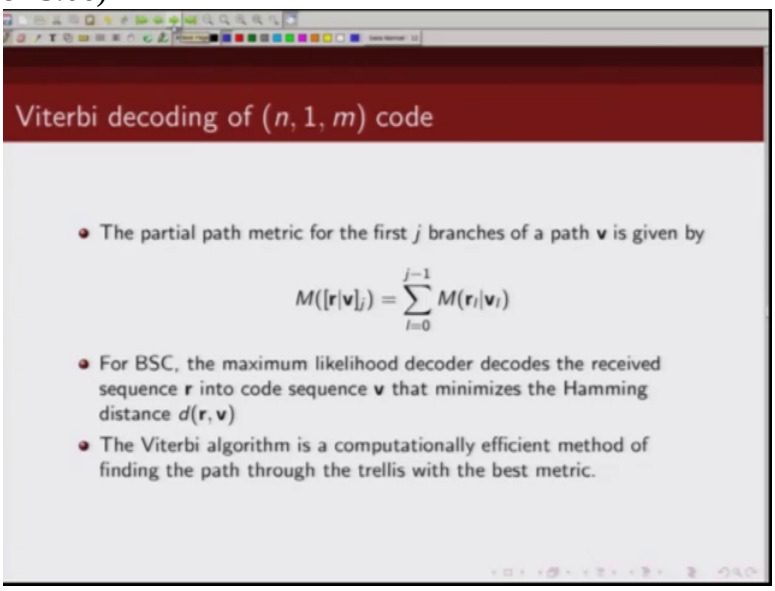
Similarly we can define a partial path metric for the, up to j th branch of a path v as summation of path metrics up to jth branch. So this is path metric computed from zero to j minus 1. So this is what we are defining as partial path metric.

(Refer Slide Time 14:17)



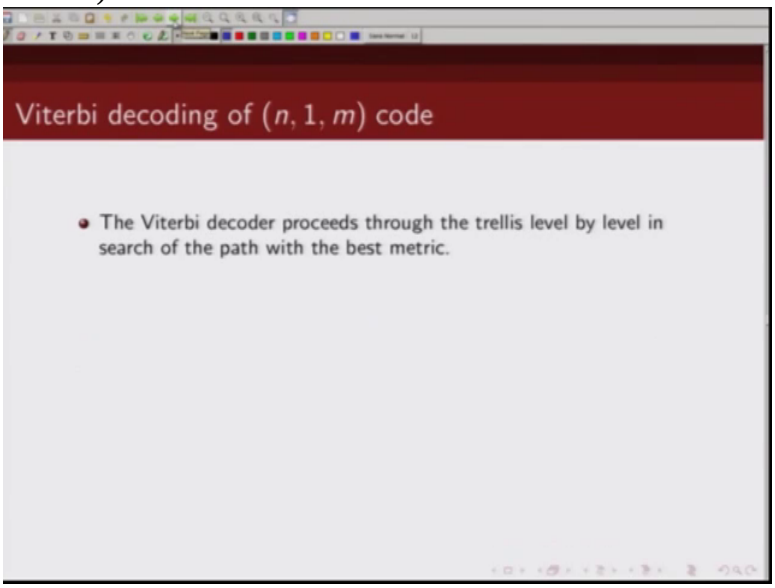
Now what is the maximum likelihood decoding rule for a binary symmetrical channel? For a binary symmetrical channel, the maximum likelihood decoder is one that decodes the received sequence r into code sequence v such that the Hamming distance between the received code sequence and the transmitted code sequence is minimized. This we have seen in the initial lecture on Introduction to Convolutional code that maximum likelihood decoder for binary symmetrical channel will try to minimize the Hamming distance between the received code sequence and transmitted code sequence.

(Refer Slide Time 15:06)



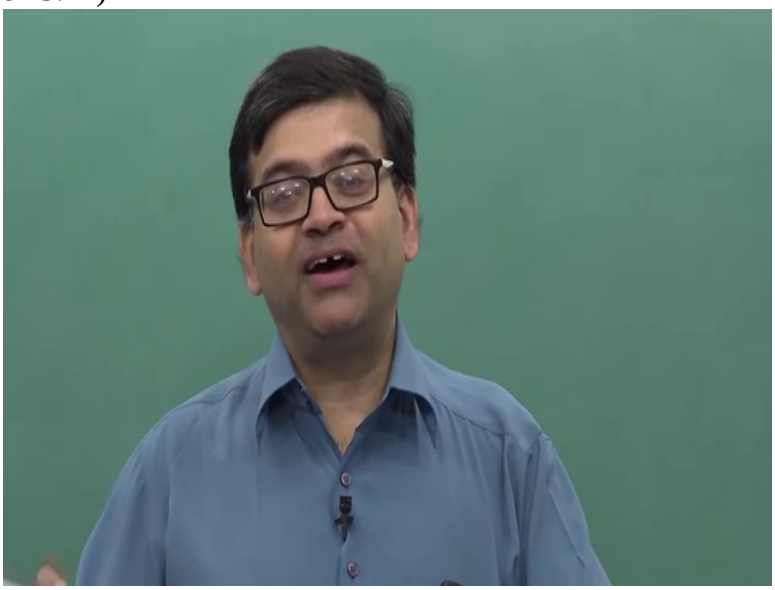
And Viterbi algorithm provides a computationally efficient method of basically just doing that. And it tells us a way to efficiently find a path through this Trellis which will maximize the likelihood function. In other words it will choose the path that will basically maximize the path metric. It will choose the path

(Refer Slide Time 15:33)



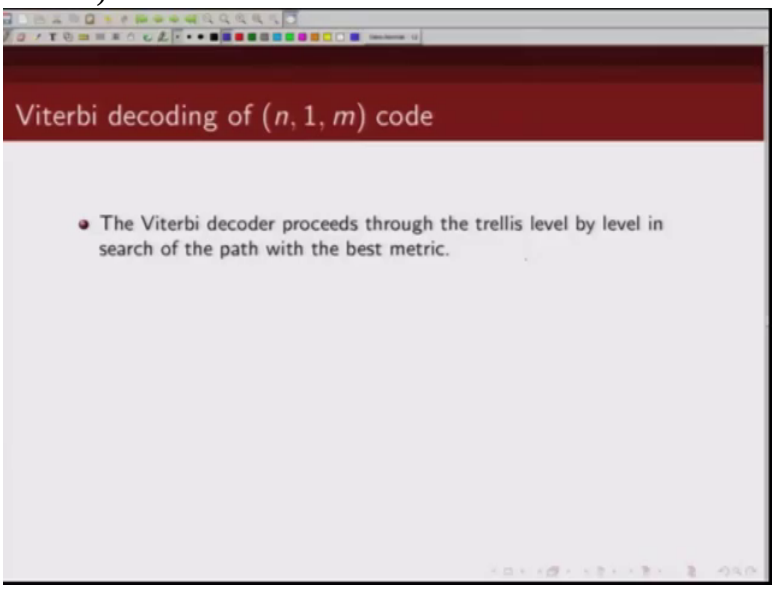
which will maximize the path metric. So how does Viterbi decoder works? So it proceeds through the Trellis one time instance at a time in search of the

(Refer Slide Time 15:47)



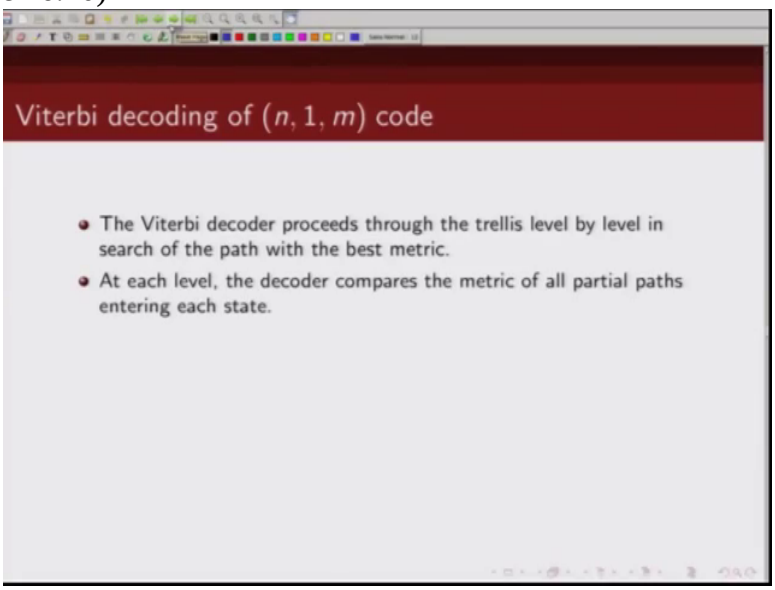
best partial path metric up to that particular time instance. So you start with time t equal to 1. At that time instance you will try to compute the best path metric at each of the nodes, at each of these states. And this you repeat for next time instance. You do the same thing. You try to find out what is the best path metric for that particular time at that

(Refer Slide Time 16:18)



particular state. So

(Refer Slide Time 16:20)

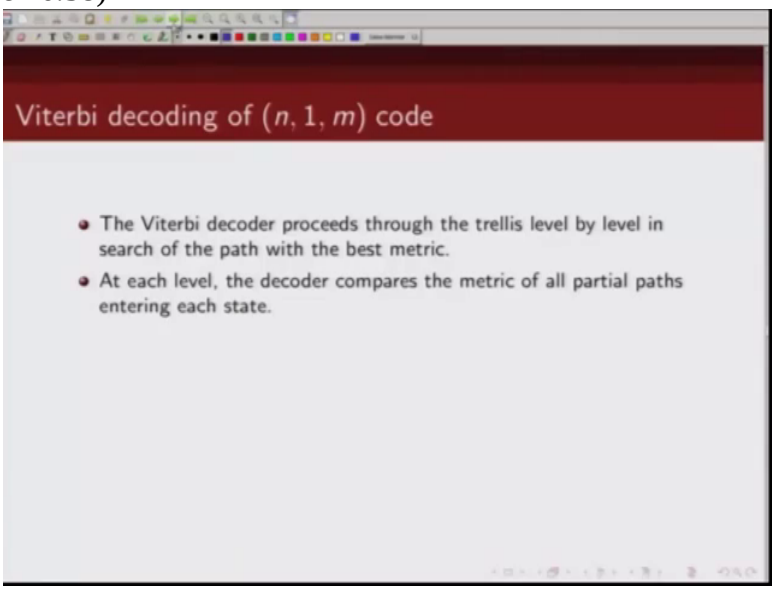


at each level the decoder is going to compute and compare the metrics of all partial paths entering that state and it is going to choose the one

(Refer Slide Time 16:34)

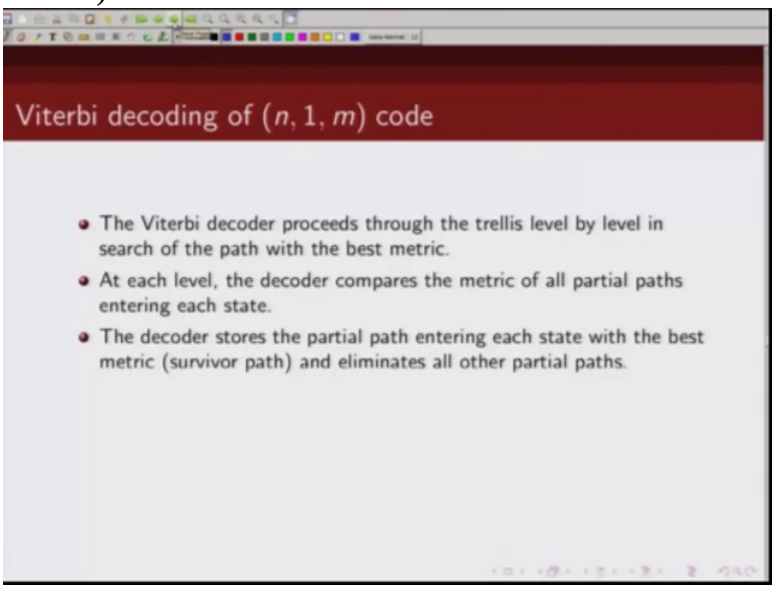
which has the best partial path

(Refer Slide Time 16:38)



metric at that particular

(Refer Slide Time 16:39)

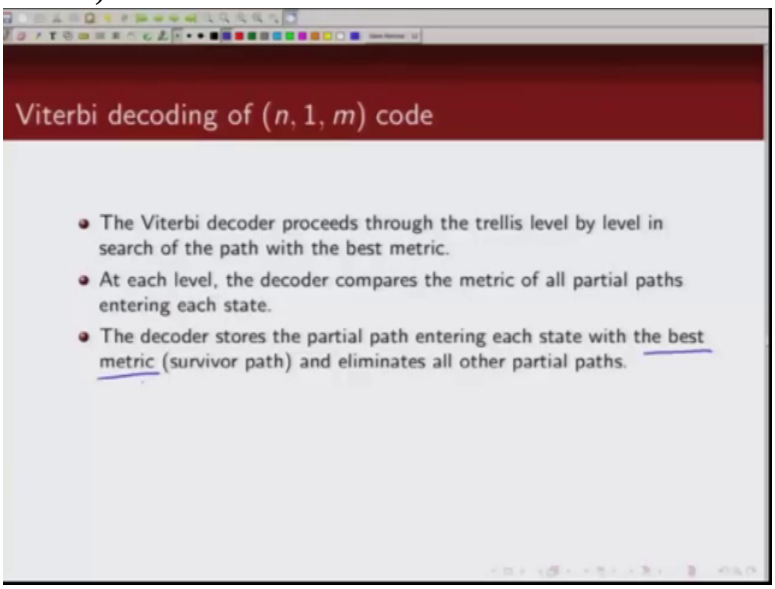


time instance. So the decoder stores the partial path entering each state with the best metric. Note that there are 2 branches which are converging for time t greater than m, there are 2 branches converging at each state. So from those

(Refer Slide Time 17:00)

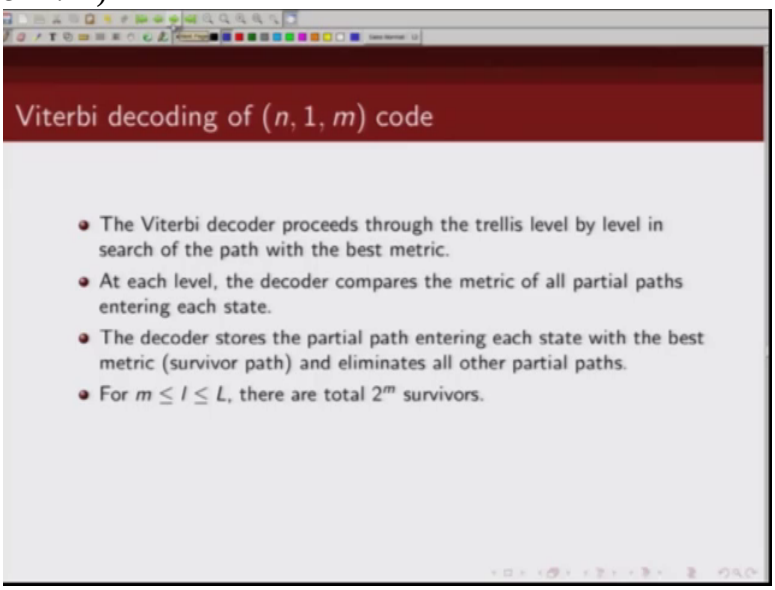
2 states, from those 2 branches it is going to pick up the one branch which gives us the best path metric. So it will store that best path metric

(Refer Slide Time 17:11)



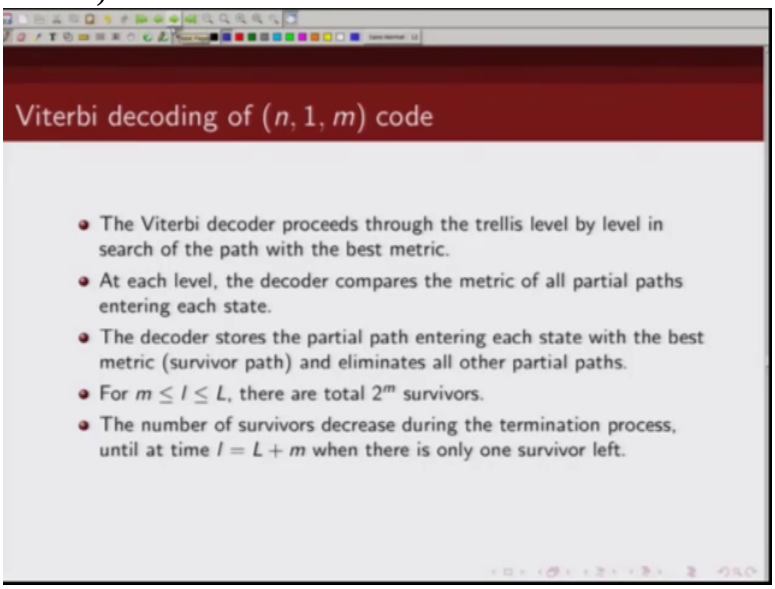
and remove all other path metrics which do not give us the best path metric.

(Refer Slide Time 17:22)



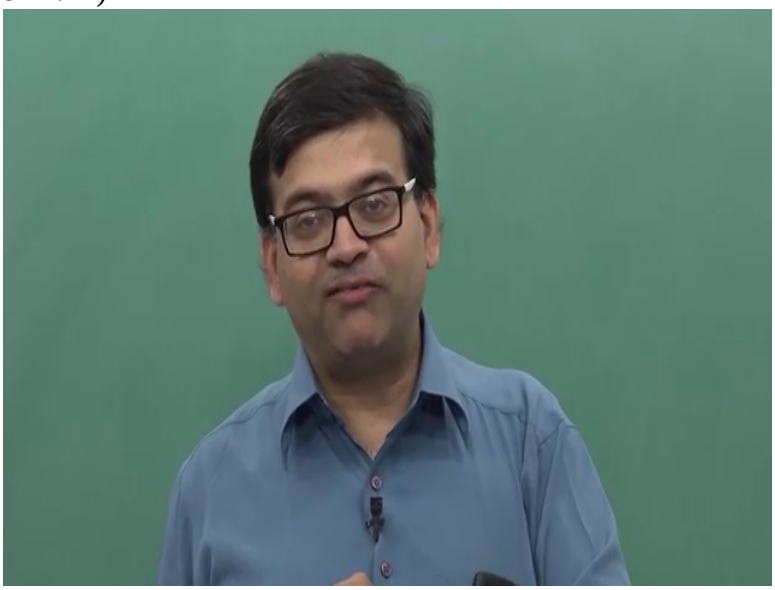
So for time instance l between m and l there will be total 2 m survivors and

(Refer Slide Time 17:32)



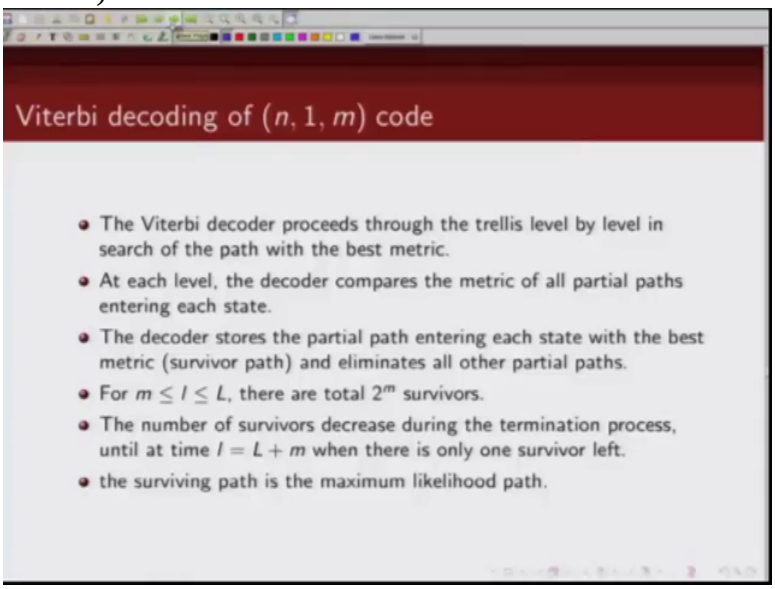
during this termination process number of survivors will diminish until we reach the final state, the all zero state where we will be left with only one survivor and then we

(Refer Slide Time 17:44)



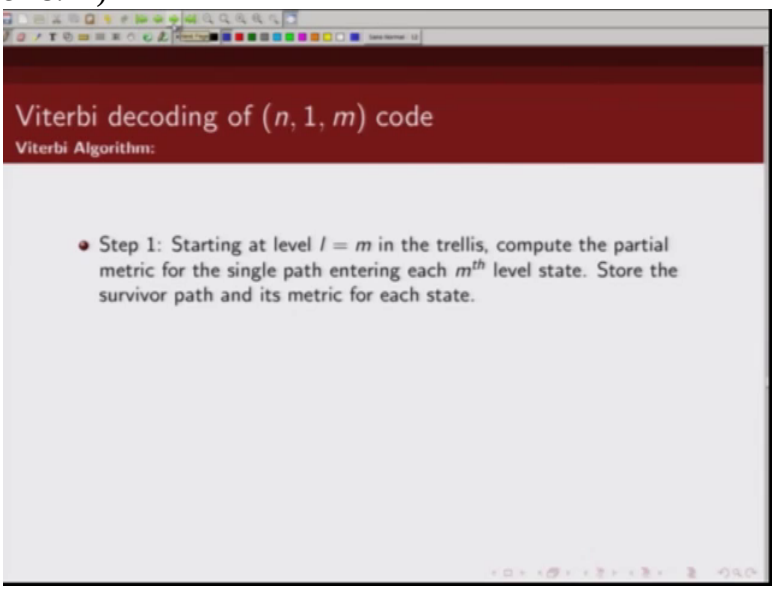
backtrack to get back our decoded

(Refer Slide Time 17:50)



path or decoded codeword. So the, when we reach the final time instance which is final state all zero state, we will be left with only one path and that is the surviving path is our maximum likelihood decoded path and then we backtrack to get back what were the code bits and information bits corresponding

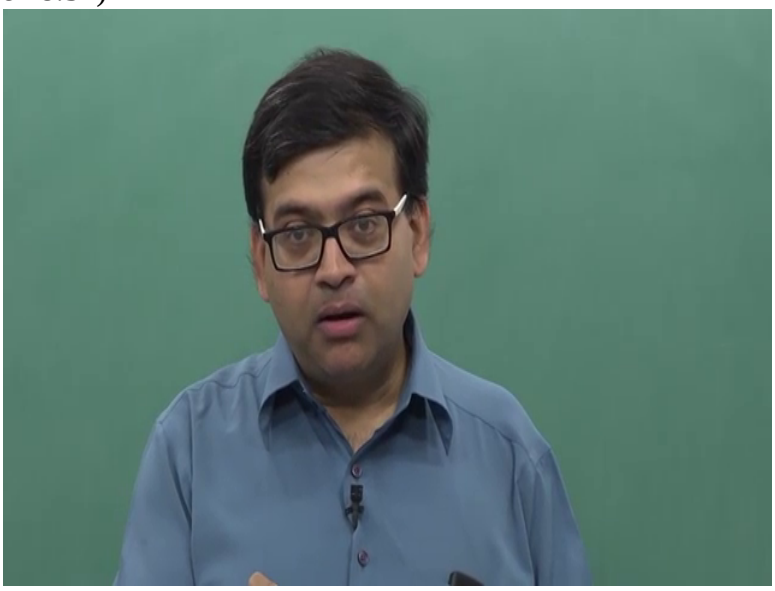
(Refer Slide Time 18:12)



to that particular path.

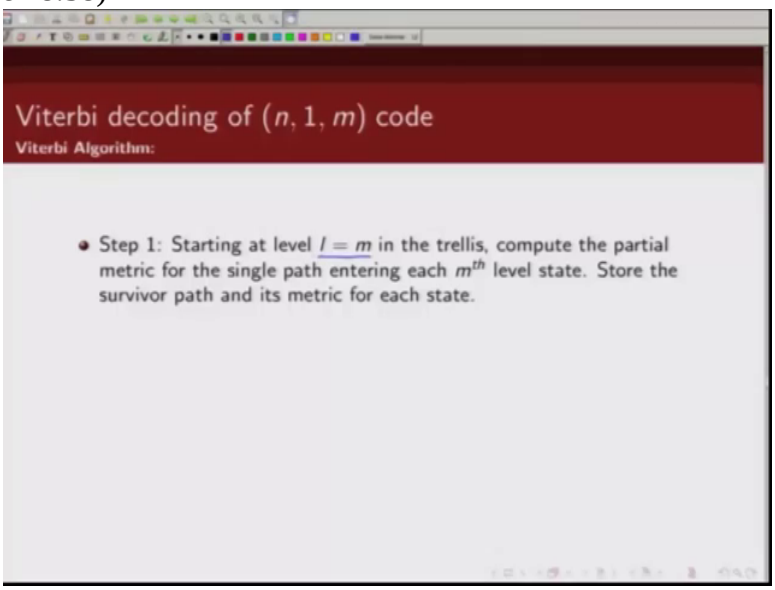
So how does, so to summarize basically how does this Viterbi algorithm works? So let us say starting from t or l up to m, note up to the memory order there is only one path

(Refer Slide Time 18:31)



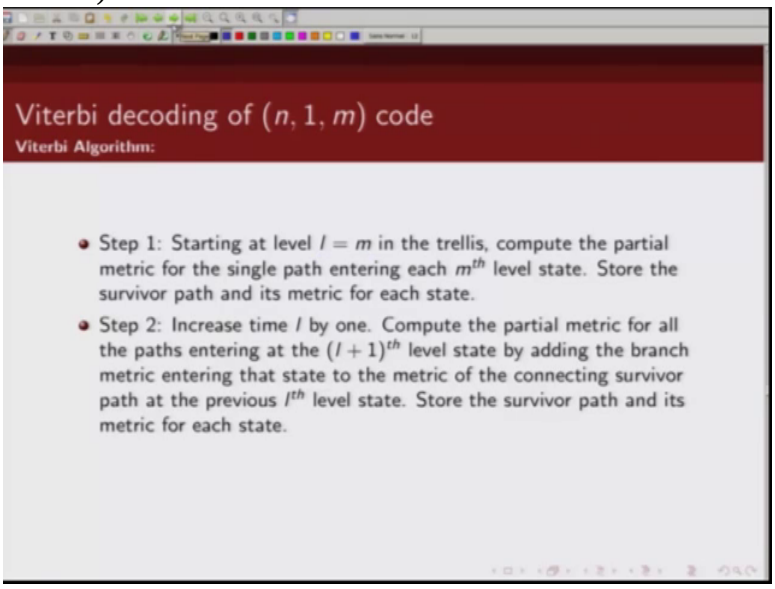
from the initial state, all zero state to any other state. So starting at time

(Refer Slide Time 18:38)



up to l equal to m in the Trellis, we compute the partial path metric for the single path from all zero state to any particular m th state and we store the survivor path as well as the metric value for that particular state. Next

(Refer Slide Time 18:58)



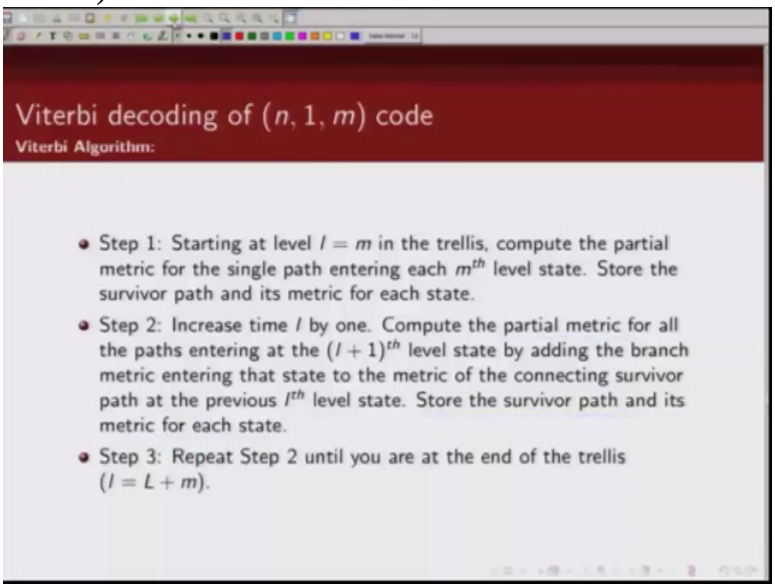
we increase the time by one instance and now we compute the partial path metric for all the paths at that time instance which are entering that particular state and we compare the partial path metric and choose the one which gives us the best path metric and we ignore other paths. So at next time instance we are going to store the survivor path which is the one which has one best metric and we will also store what its metric value is. And how do you compute the partial metric at time l plus 1 given that you know the path metric at time l? So what we do is we add to the path metric at time l, we add the branch metric value to whatever is the partial

path metric at that particular state and that's how we compute the path metric at time l plus 1. Now this will be clear when we illustrate

(Refer Slide Time 20:11)

this with another, with an example immediately after this.

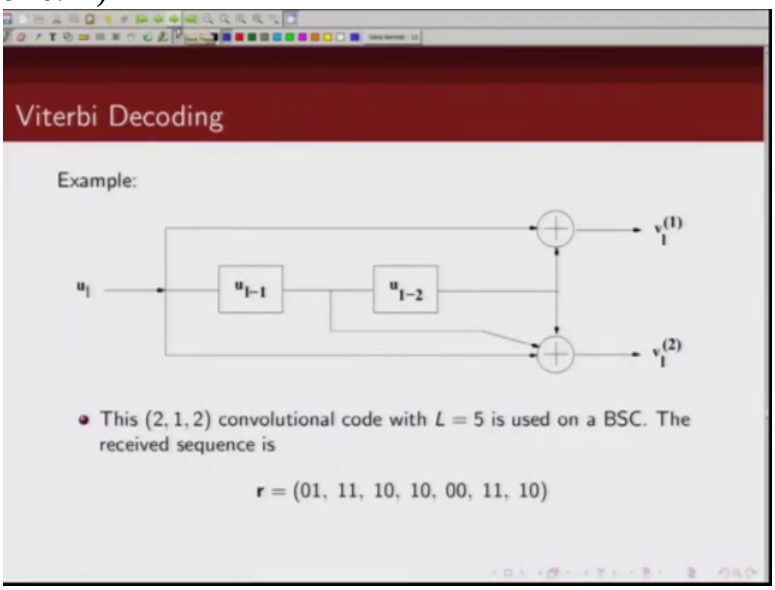
(Refer Slide Time 20:16)



And we repeat this process until we reach the end of the Trellis diagram.

So let us come

(Refer Slide Time 20:24)

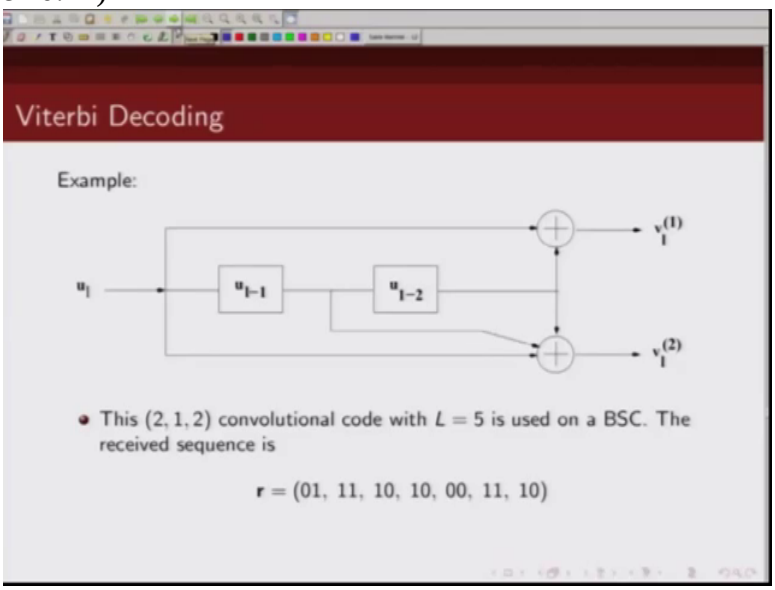


to an example. To show how we compute the partial path metric, how do we choose the

(Refer Slide Time 20:32)

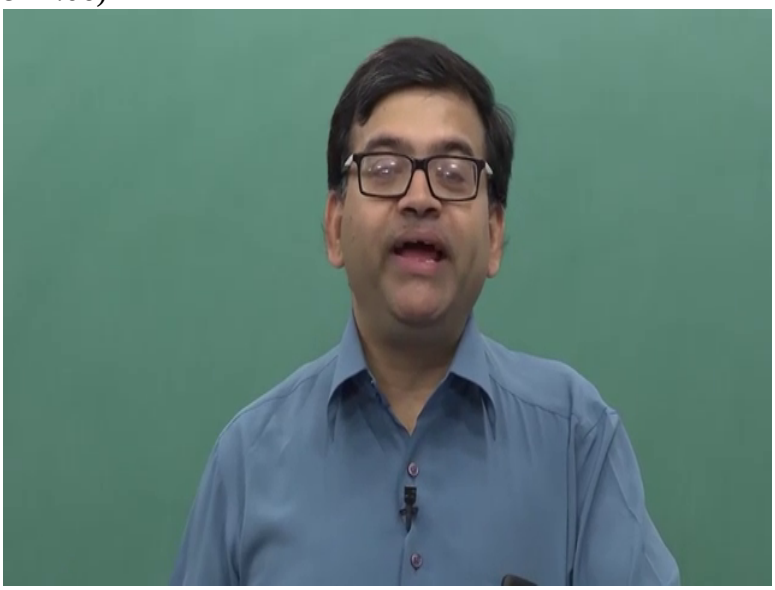
best surviving path, how do we eliminate the path which does not give us the best path metric, Ok. So let us take an example

(Refer Slide Time 20:44)



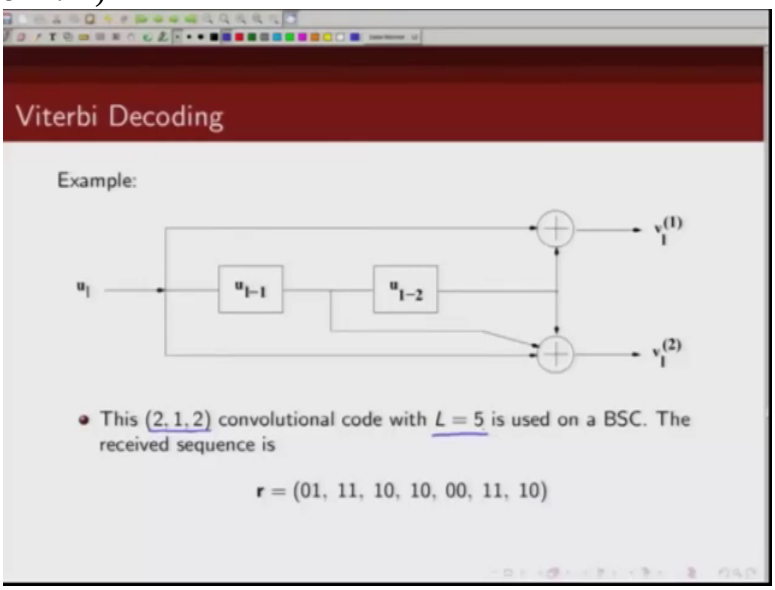
f a rate 1 by 2 code which has memory 2. This is the encoder of a rate 1 by 2 code. We are given the number of information bits were 5. Now since its memory is 2, we require 2 bits to bring it back

(Refer Slide Time 21:06)



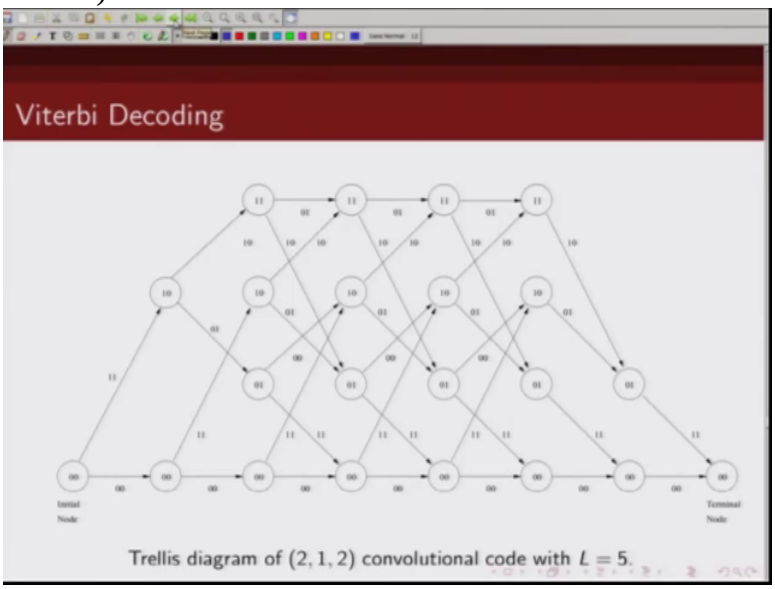
to all zero state. So total basically we are transmitting

(Refer Slide Time 21:11)



5 information bits plus 2 tail bits. So total 7 bits and since it is a rate 1 by 2 code, so 7 bits will result in 14 coded bits. So this is basically the received sequence corresponding to those 14 bits that we transmitted. This is the received sequence over the binary symmetric channel. And the question that we have to answer is, given this received sequence we need to find out what is the transmitted codeword or what is the transmitted information sequence. So how do we proceed? The first step is we will draw the state diagram of this convolution encoder and we will draw the Trellis diagram of this convolution encoder. So

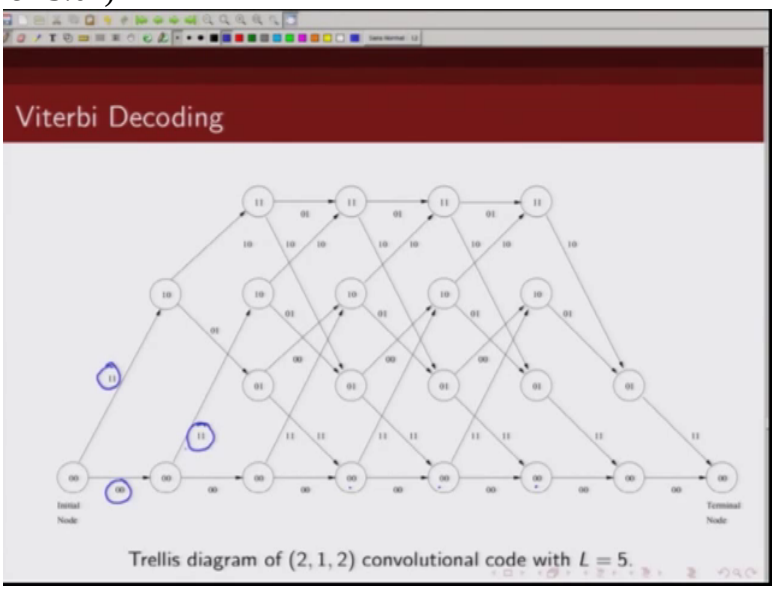
(Refer Slide Time 22:03)



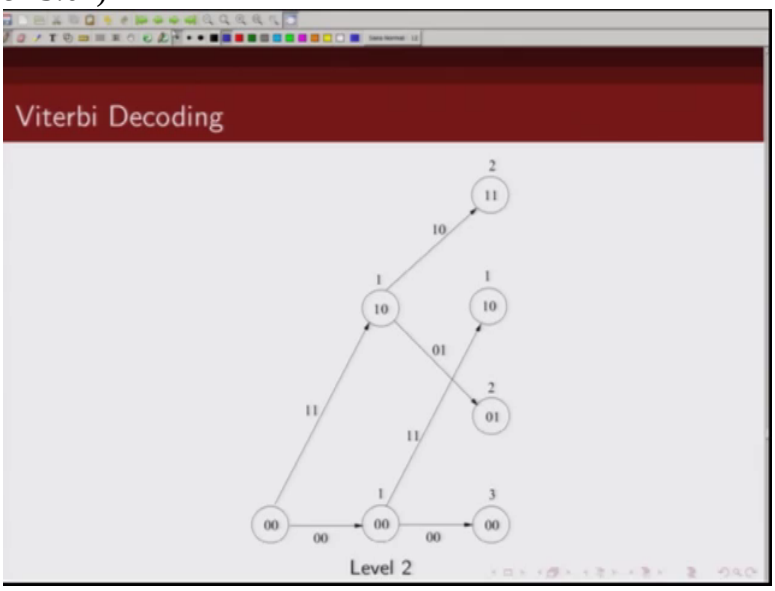
what I have here is I have the Trellis diagram of this rate 1 by 2 convolutional encoder. Please note initially we are starting with all zero state, so this is time instance 1, 2, 3, 4, 5. These were our information bits. Now what happened after this, we are terminating the convolution

code to all zero state so number of states reduces by half at each time instance so from here from 4 it becomes 2 and from here it becomes 1, 1 state. So you can see we are basically transmitting 5 information bits and 2 tail bits and what I have labeled here is I have. at each branch I have labeled what is the codeword corresponding to this transition.

(Refer Slide Time 23:01)



(Refer Slide Time 23:02)

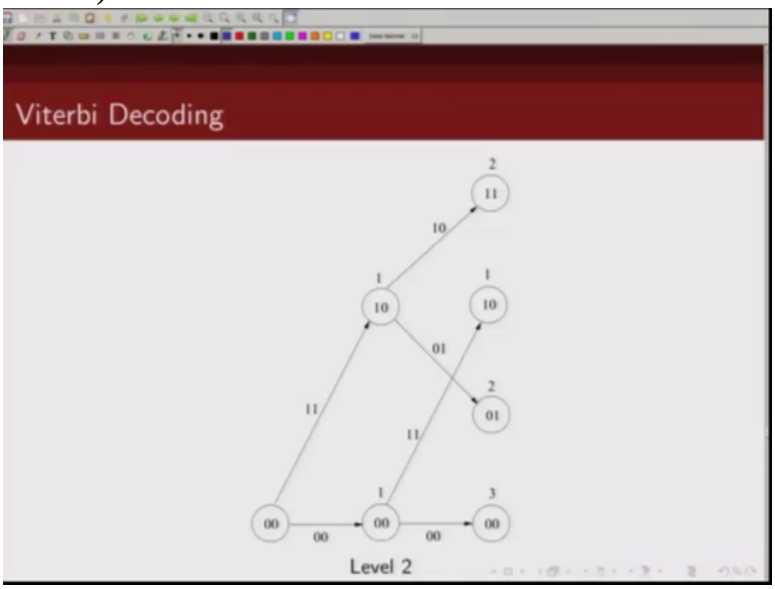


So what is the first step? We will first try to compute

(Refer Slide Time 23:06)

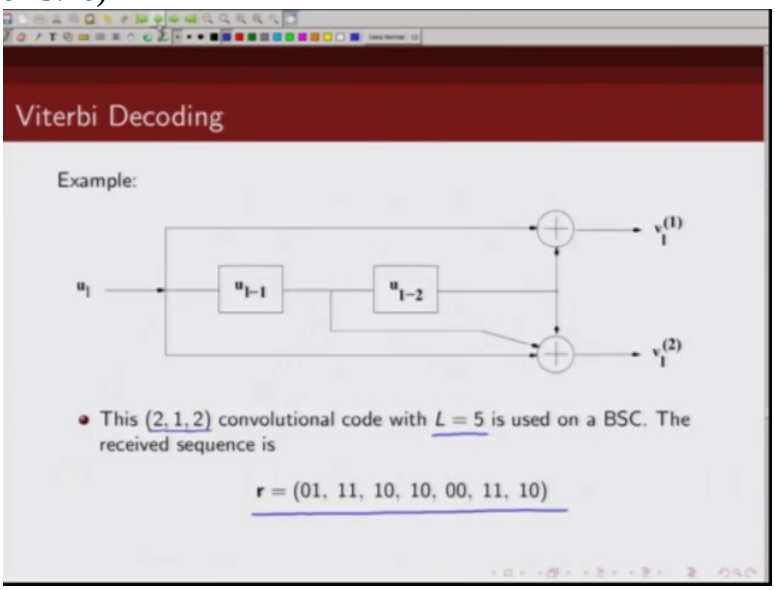
the partial path metric up to time t equal to 2. Why time t up to 2, because the memory order of this convolution code was 2, so from all zero state

(Refer Slide Time 23:18)



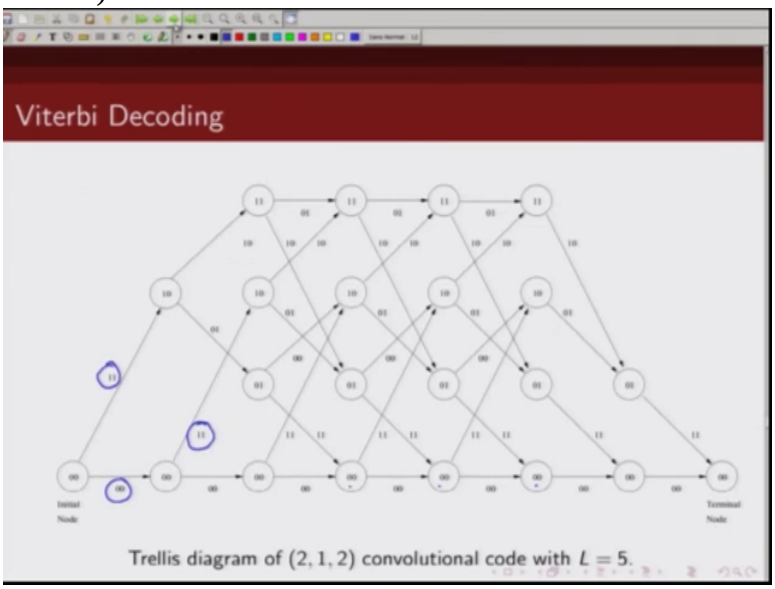
up to time instance 2, there is a single path from all zero state to any particular state. So for example if you want to reach state 0 1, we can reach it. From first, we will go from 0 0 state to 1 0 state and from 1 0 we will go to 0 1 state, Ok. Now how do we compute these path metric? So let's go back and see

(Refer Slide Time 23:46)



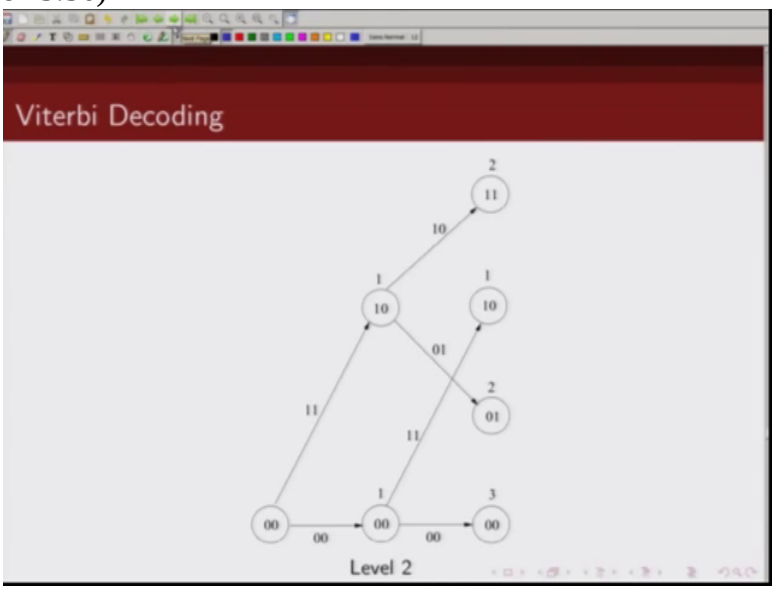
what our first 2 received sequences were. It was 0 1 and 1 1.

(Refer Slide Time 23:54)



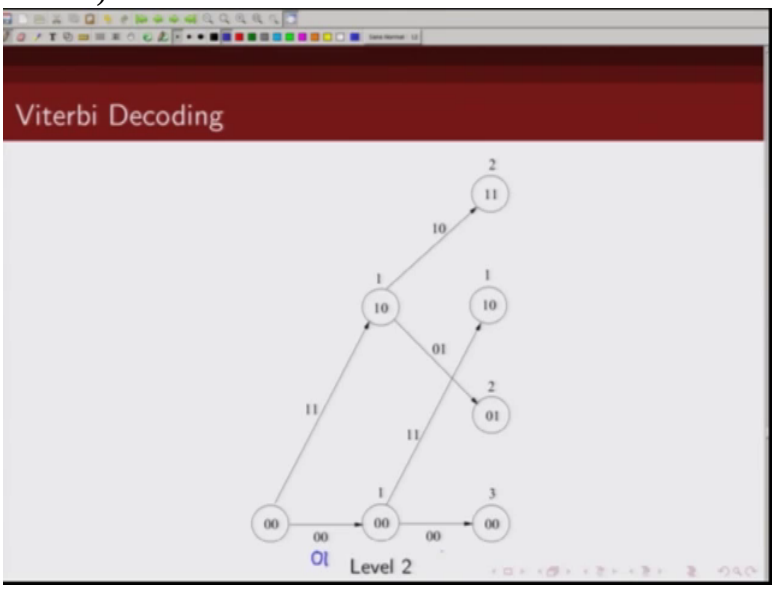
So we had received

(Refer Slide Time 23:56)



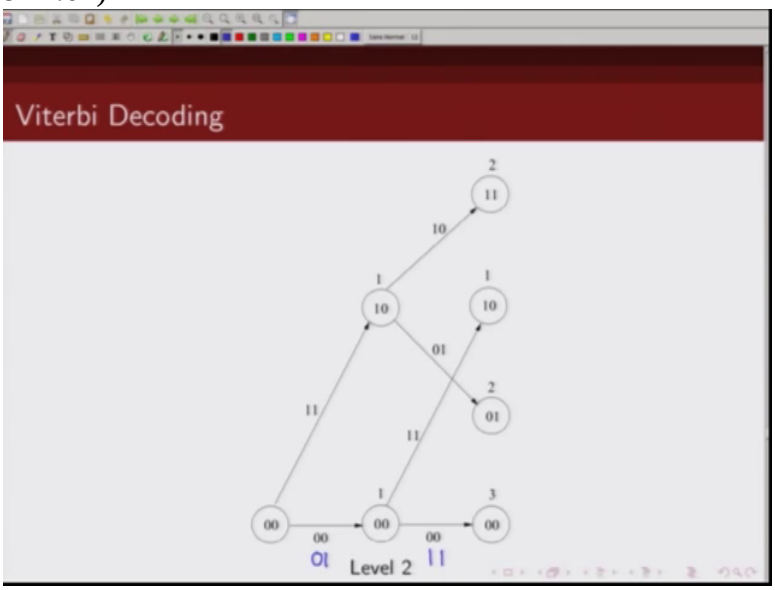
here 0 1 and

(Refer Slide Time 24:01)



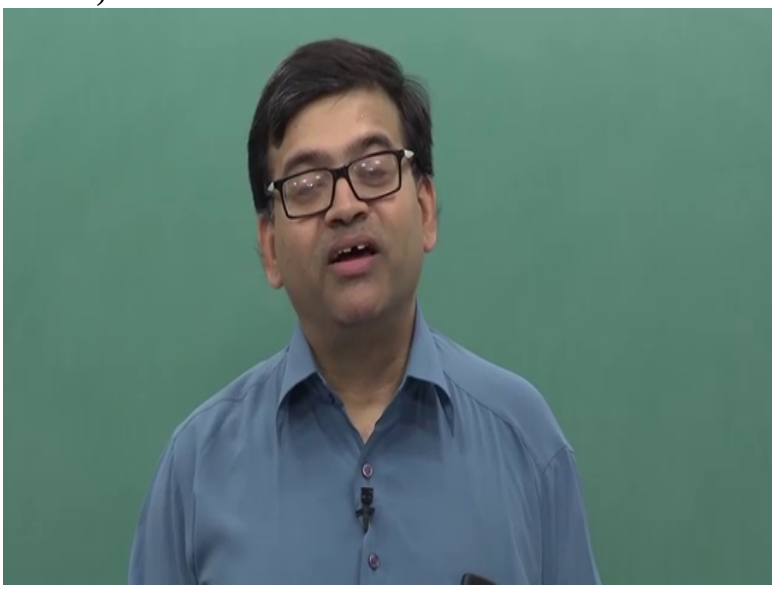
1 1.

(Refer Slide Time 24:02)



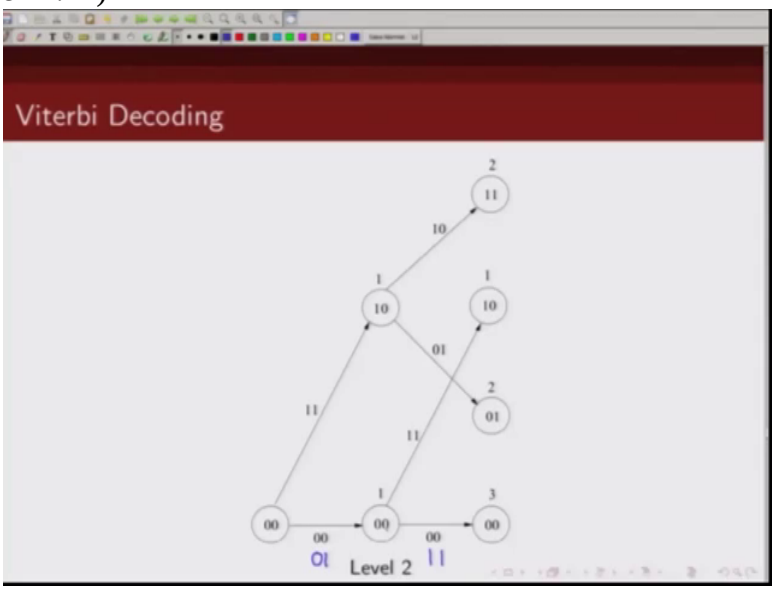
Now remember for binary symmetric channel the maximum likelihood decoder is

(Refer Slide Time 24:10)



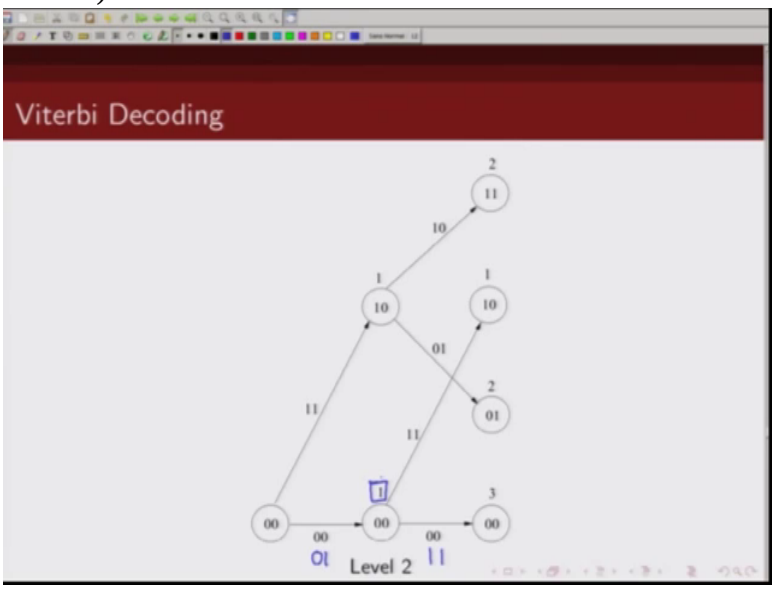
trying to find the Hamming distance between the received sequence and the code distance and it will try to choose a path which will minimize the Hamming distance between the r received sequence and codeword v. So let us look at

(Refer Slide Time 24:27)



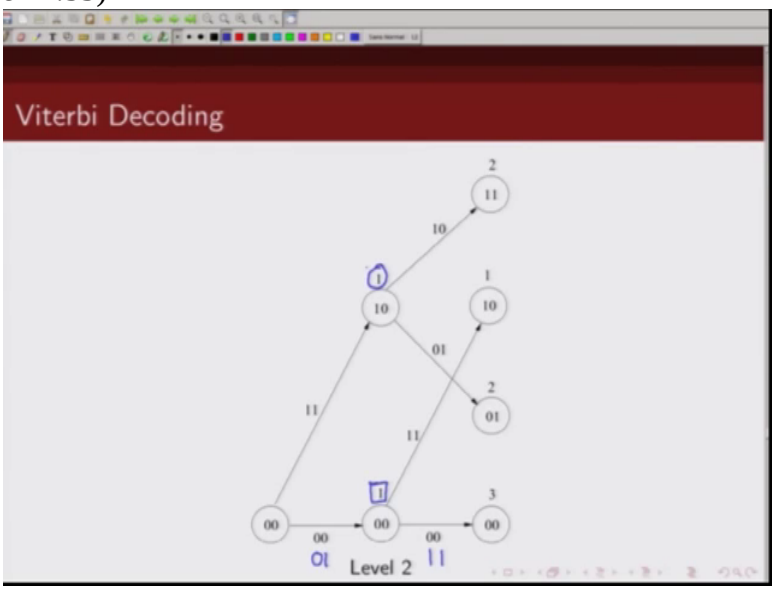
this path from 0 0 to 0 0. What is, what should have been the codeword 0 0, we have received 0 1 so what is the path metric corresponding to this? That's 1. So this is the path metric.

(Refer Slide Time 24:41)



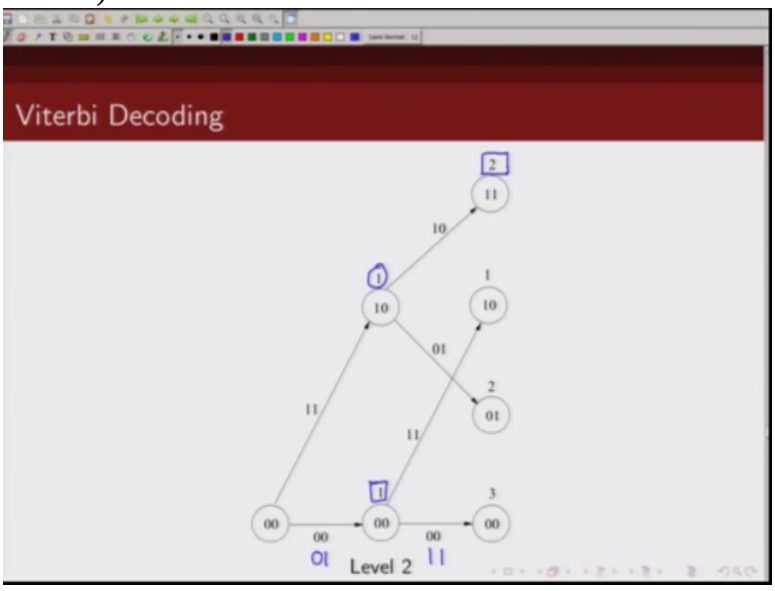
Now look at here. Code sequence is 1 1. This is 0 1. So path metric is differing in Hamming distance 1. So that's my

(Refer Slide Time 24:53)



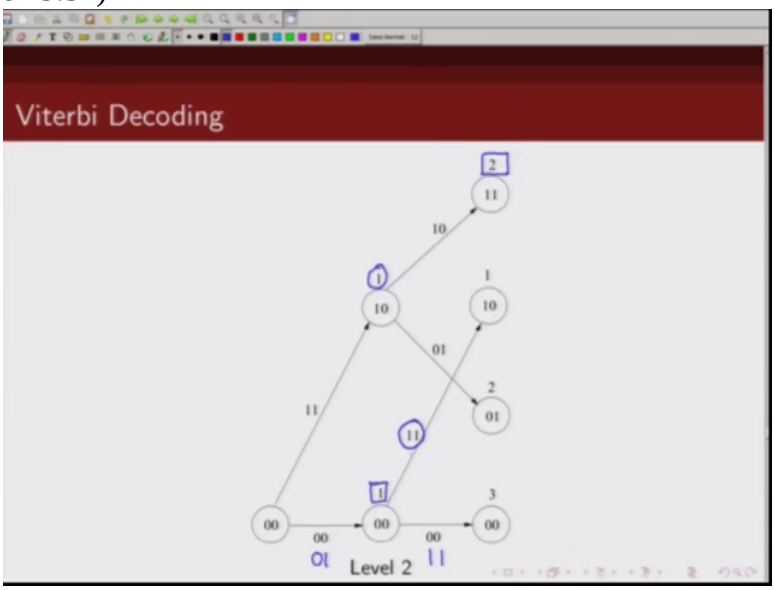
path metric here. Now what is the path metric here? This is equal to the path metric, partial path metric at this time plus branch metric corresponding to this. So this is 1 0 and what we received was 1 1 so what is the Hamming distance between 1 1 and 1 0? That's 1, so 1 plus 1, so 2. So this will have a path metric of 2.

(Refer Slide Time 25:22)



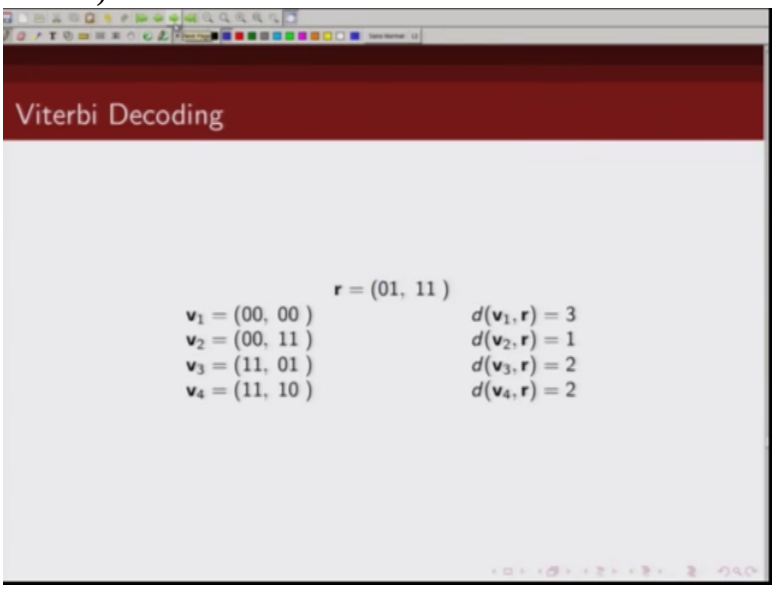
Similarly this, what is the path metric at this instance? So note the branch metric for this branch is zero because code sequence

(Refer Slide Time 25:34)



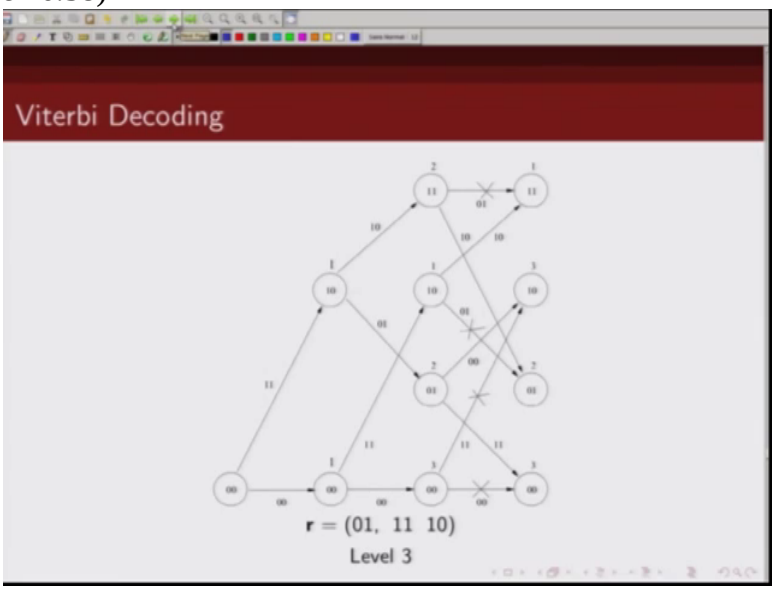
is 1 1 and received sequence is 1 1. So Hamming distance is 0. What was the partial path metric? The partial path metric here was 1 plus 1 and the branch metric here is 0. So 1 plus 0 that is 1. Similarly here we can find out what is the partial path metric here? This is going to be partial path metric here plus the branch metric of this. So what is the branch metric of this? This is 0 1 and this sequence is 1 1 so it is differing in the first bit location. So 1 plus 1 that's 2. And similarly here the partial path metric is 1, the branch metric in this case is 2, so 1 plus 2, the partial path metric is 3.

(Refer Slide Time 26:26)



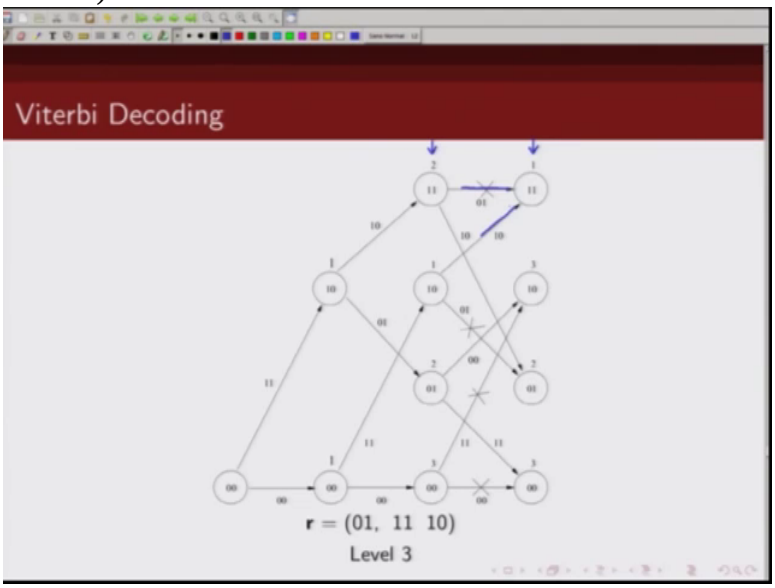
And this is what I have also illustrated here. Our received sequence was 0 1 and basically these are the Hamming distance between code sequence and the received sequence.

(Refer Slide Time 26:38)



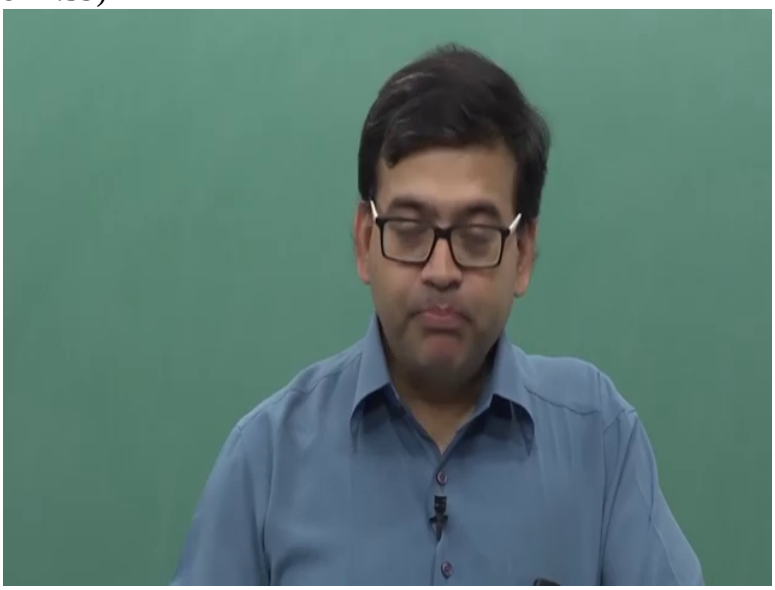
Now once we know what are the path metric here at this time instance then next step, what do we do, we need to find the path metric here. Now how do we find the path metric here? Now note here we have 2 branches which are entering the state. One is this, other is this, right. So

(Refer Slide Time 27:06)



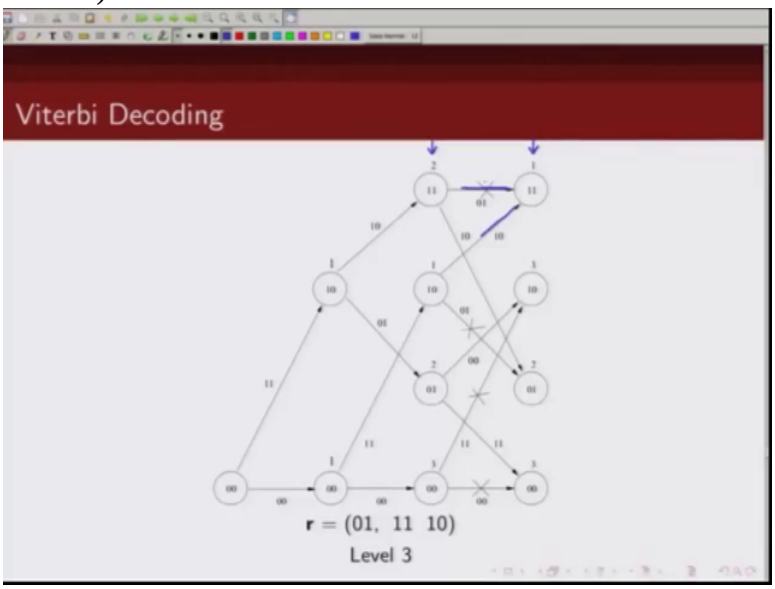
how do we compute the partial path metric here? This we compute by adding to the partial path metric of this state plus the branch metric of this branch or the path metric at this state plus the branch metric corresponding to this transition. So we are going to choose which one of these two is a better, gives us a

(Refer Slide Time 27:39)



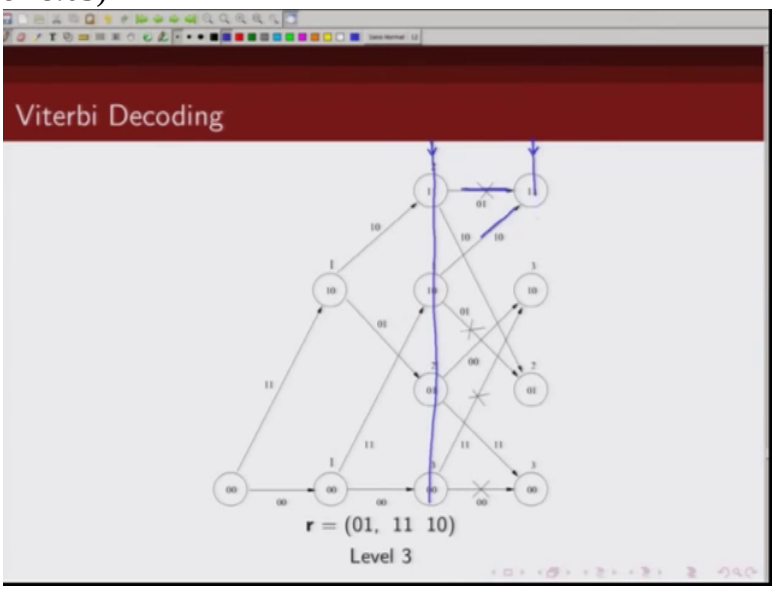
best, better path metric and then we are going to choose that path as our surviving path and we are

(Refer Slide Time 27:47)



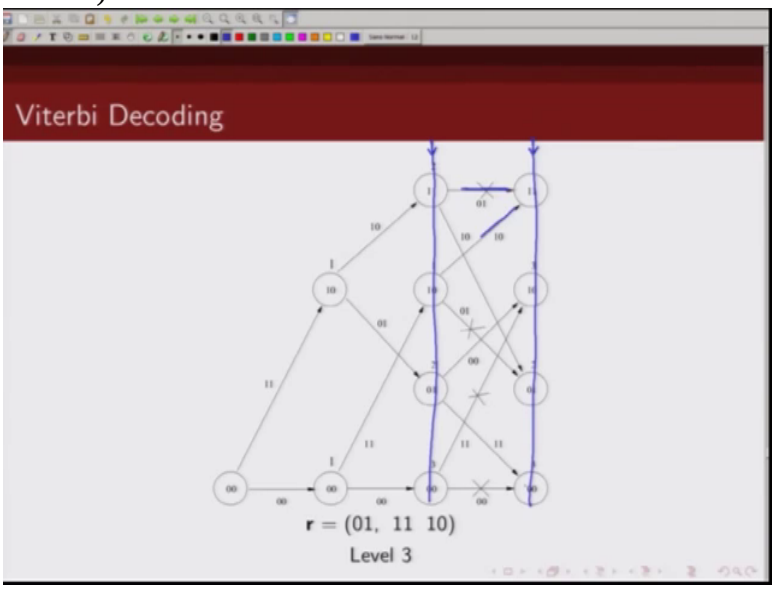
going to delete the path which does not give us a best metric and we will also store what is the best path metric corresponding to that particular transition. So we are looking at this time instance, this Trellis

(Refer Slide Time 28:03)



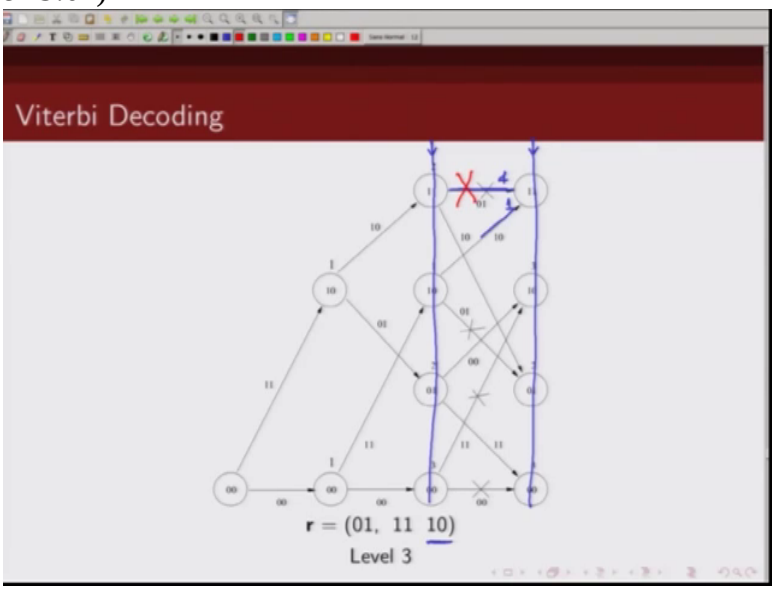
section. We are looking

(Refer Slide Time 28:04)



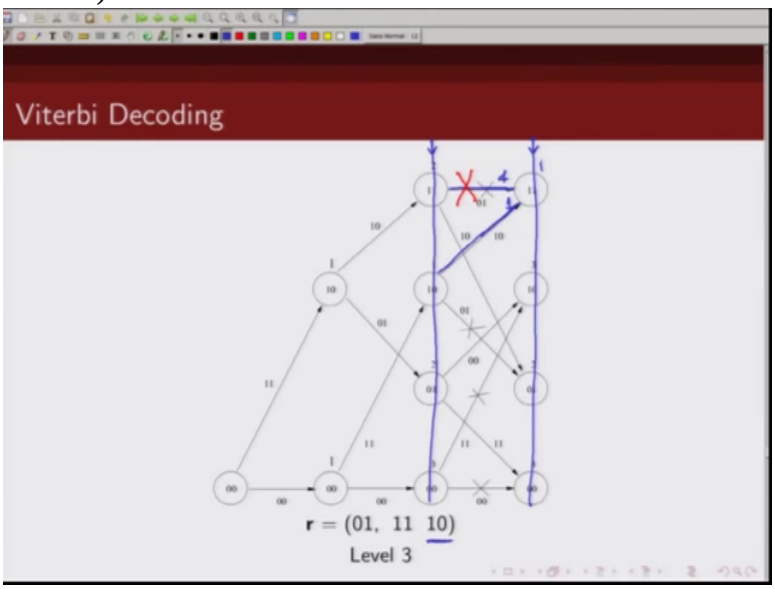
here. So what was the received sequence? Here it was 1 0. So what was the branch matrix corresponding to this transition, this transition? This is 0 1, this is 1 0, so Hamming distance in this case is 2 and what was the Hamming distance here? Path metric was 2, so 2 plus 2 this would give me partial path metric of 4. Now let's compute path metric of this. Here the partial path metric was 1. And this was 1 0.1 0 and 1 0, the Hamming distance is 0, so corresponding to this, the path metric is 1. Which one is a better path metric, 1 or a 4? This one is a better path metric. Why? Because Hamming distance here is less. It is 1 where as corresponding to this path, it is 4.So what we are going to do, we are going to delete this path

(Refer Slide Time 29:07)



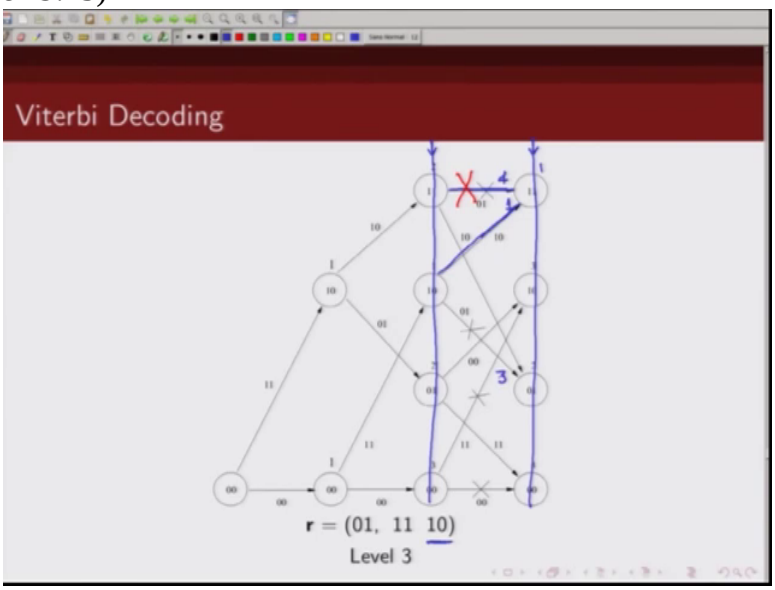
and we are going to keep this path. Not only we are going to keep this path, we are also going to keep the corresponding

(Refer Slide Time 29:18)



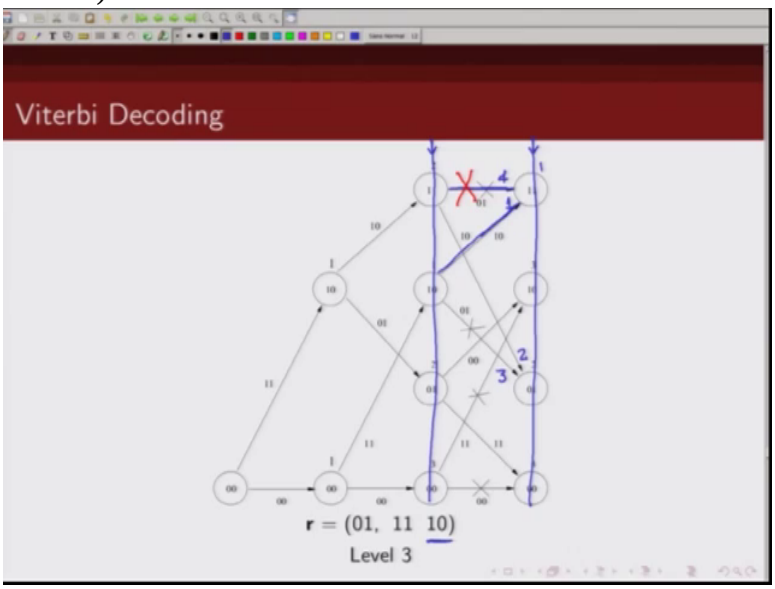
path matrix which is 1. Similarly we will do the same step for other nodes. Let's just take this example. This node 0 1, so one path metric we will get by adding to the path metric of this, partial path metric of this which is 1, and 0 1 and 1 0 which is Hamming distance of 2, so 1 plus 2, this path is giving me partial path metric of 3.

(Refer Slide Time 29:45)



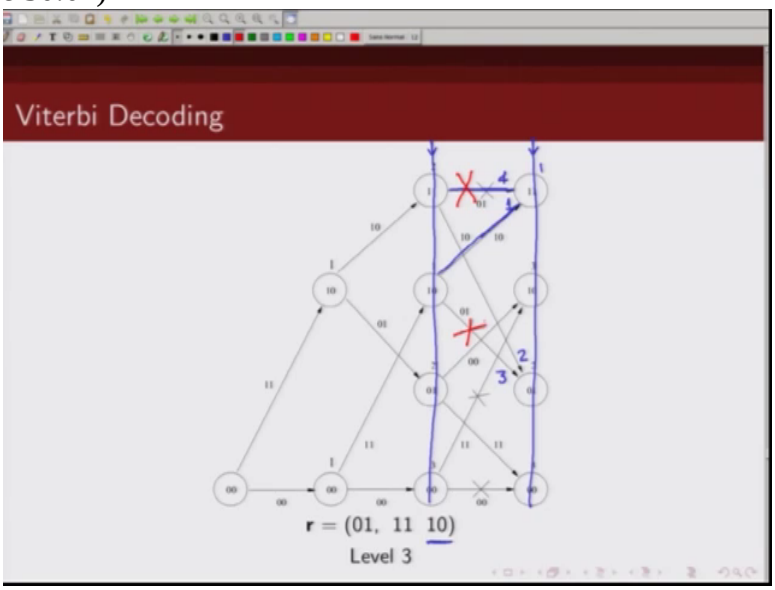
What about this path? This is 2, and 1 0. 1 0, 1 0 Hamming distance is 0, so 2 plus 0 that will give me partial path metric of 2.

(Refer Slide Time 29:57)



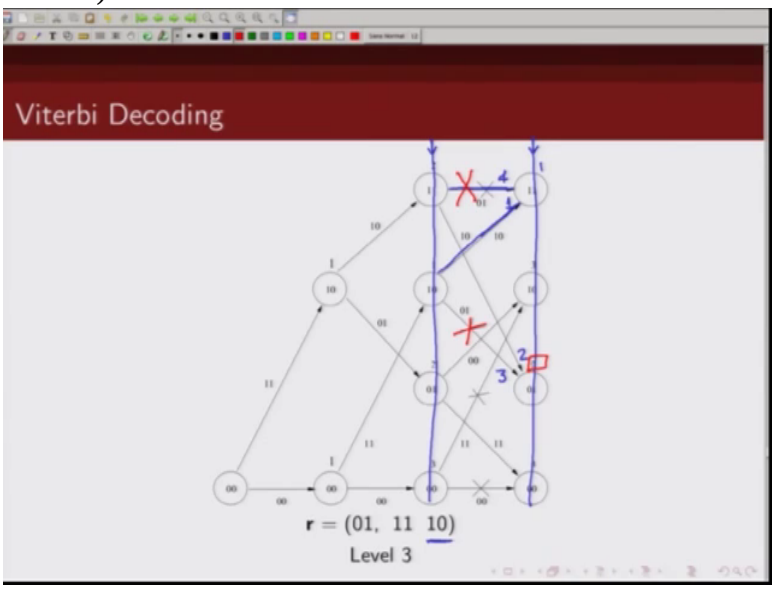
Now clearly this is not the

(Refer Slide Time 30:02)



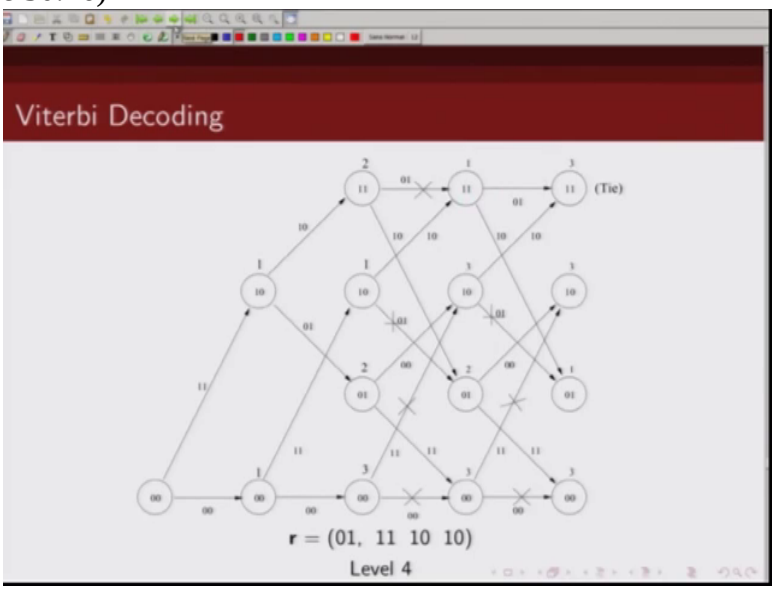
best path metric so I will keep this surviving branch and corresponding path metric which is basically given by 2. So at each

(Refer Slide Time 30:14)



node I am going to find out partial path metric corresponding to all the branches that terminate at that particular state and I am going to keep the same, I am going to keep the one which gives me the best path metric. I am going to keep the track of what is the path metric value and also keep track of the surviving path. So this step will be repeated. The same

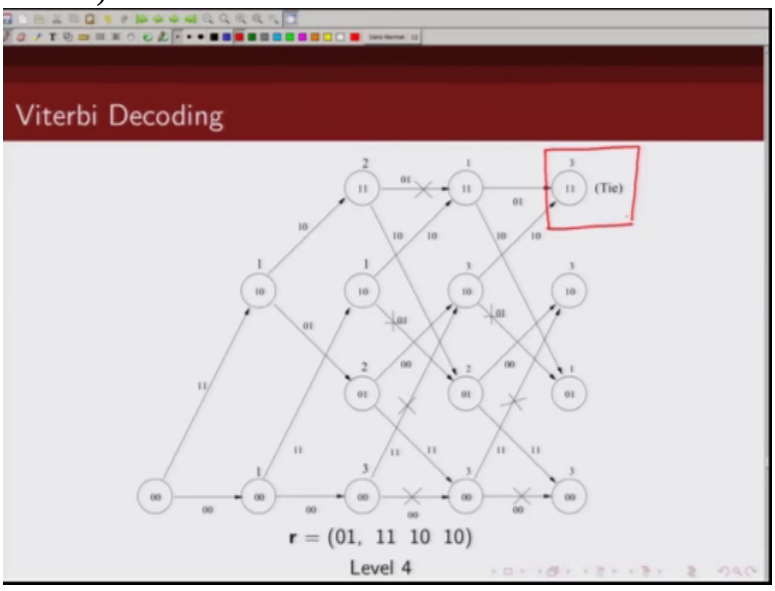
(Refer Slide Time 30:46)



thing I do here.

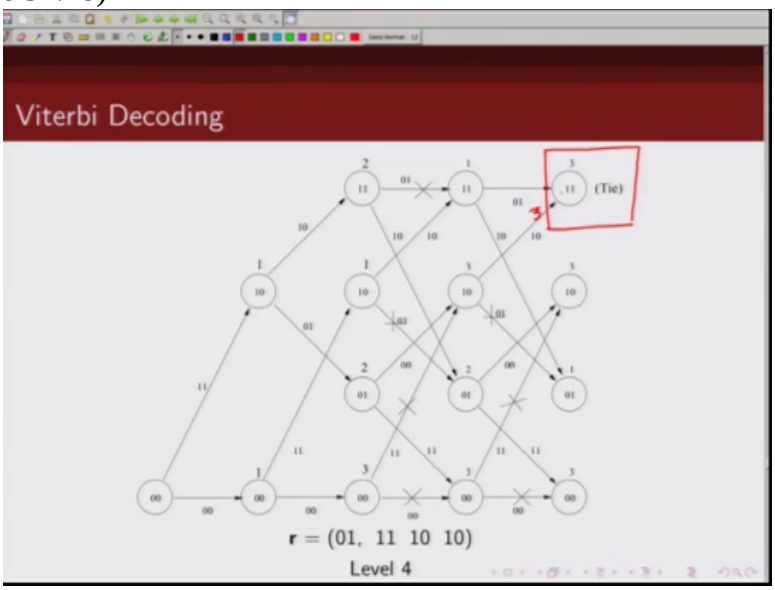
Now let's take an example here of this particular node. How do we

(Refer Slide Time 30:56)



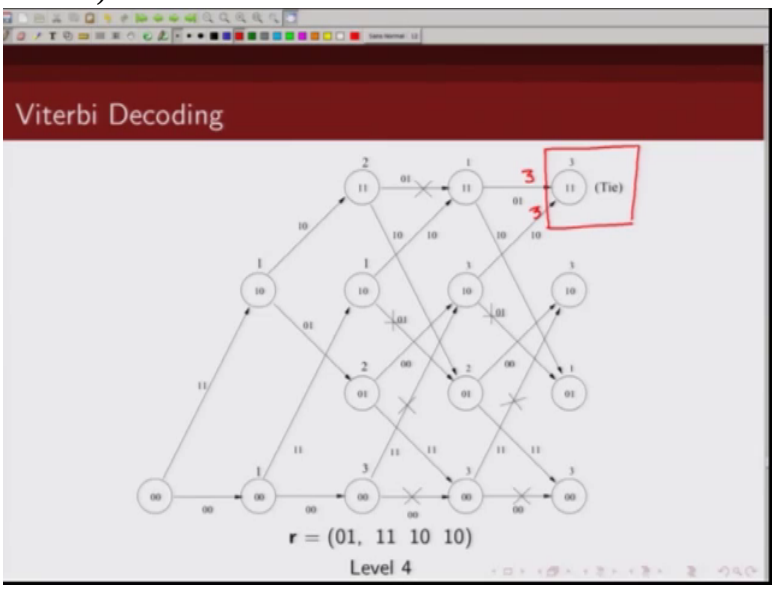
compute the path metric here? The partial path metric here, this is the partial metric here will be either, partial path metric here plus this branch metric which is 1 0 and 1 0 so this is branch metric here is 0, so 3 plus 0 is 3. Now look,

(Refer Slide Time 31:18)



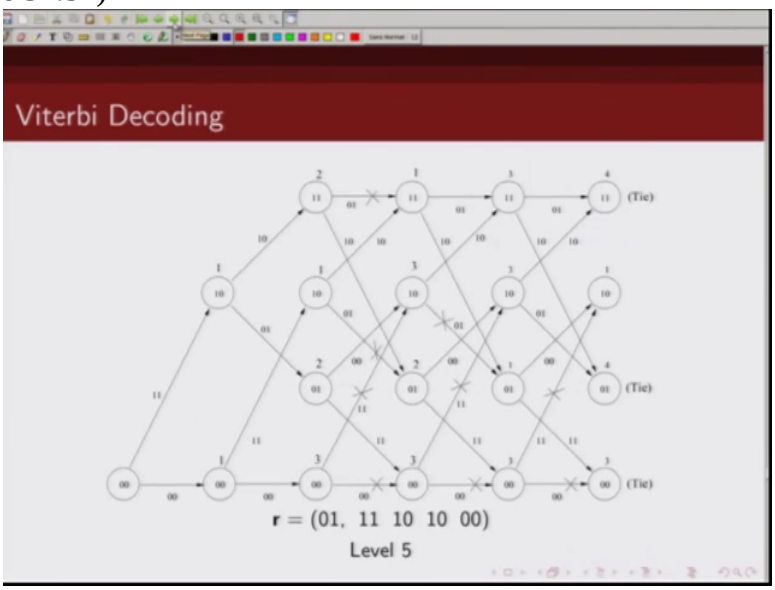
let's look at this. The partial path metric here is 1, and this is 0 1, so 0 1 and 1 0, so this Hamming distance is 2. 1 plus 2 is 3.

(Refer Slide Time 31:30)



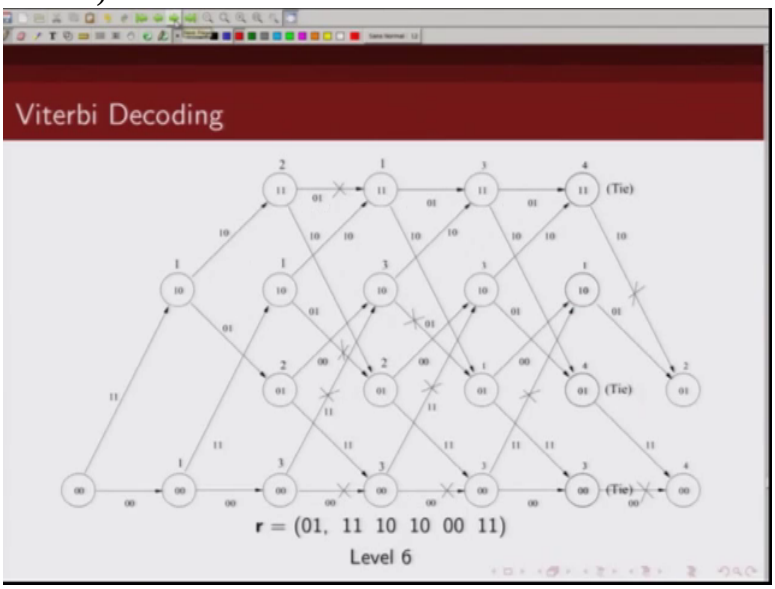
So note here, both the paths here have the same path metric. So there is a tie. So what do I do when there is a tie? So in this case, both the paths are equally likely so I can pick any one of them. So this process is going to be repeated. So I will continue with this process.

(Refer Slide Time 31:54)



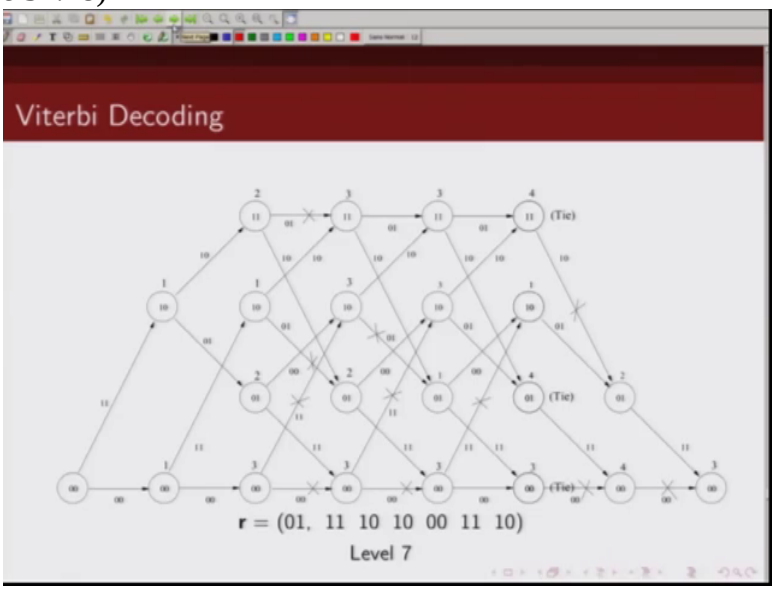
So again I repeat, at each time instance what I am doing is I am trying to compute the partial path metric at each of these states. And how do I compute the partial path metric at each of these states? This is the partial path metric at previous time instance plus the branch metric corresponding to the transition which will bring you to that particular state. And when you have multiple branches converging at particular state, you need to pick the branch which will give you the best path metric. That would be your surviving path and you need to keep track of the metric value as well. So we continue this step until

(Refer Slide Time 32:44)



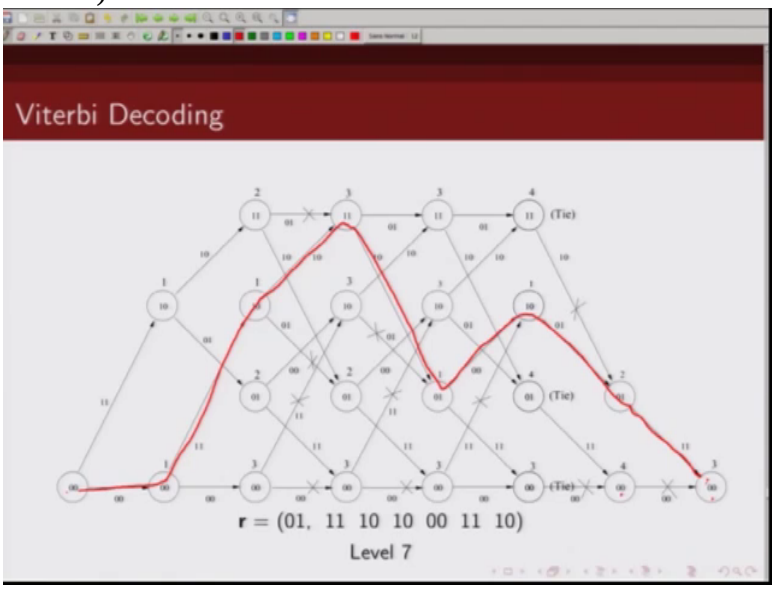
we come to the end.

(Refer Slide Time 32:48)



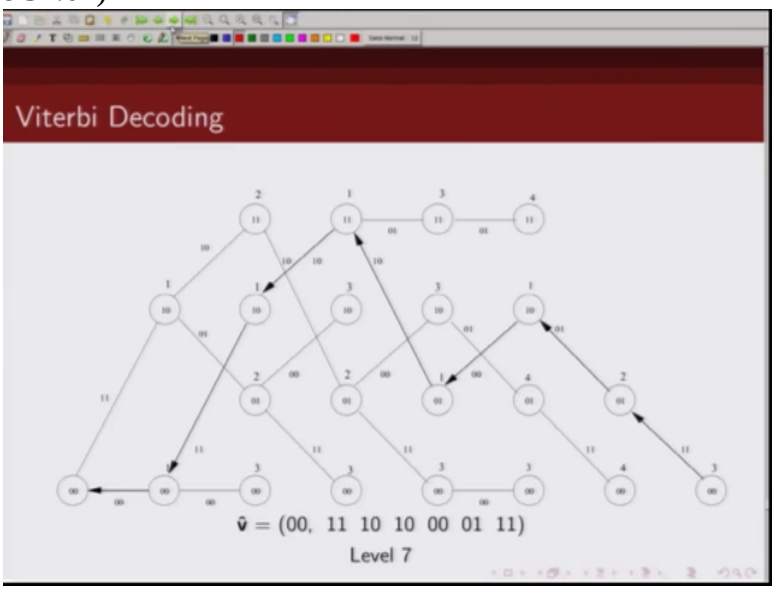
So as we know after 5 bits, because tail bits are coming, so number of states are getting reduced so from 4, the number of states became 2 here and then you have number of states basically 1. So once you come to this point then there would be only one surviving path, Ok. So once you have only one surviving path, then what you need to do is you just need to backtrack to find out what is the transmitted codeword. So you can see here, when we reach this all zero final state there is only one surviving path and that path if we track back, this path, this path, this path, this path, this path, this path, this path. So you can see there is

(Refer Slide Time 33:42)



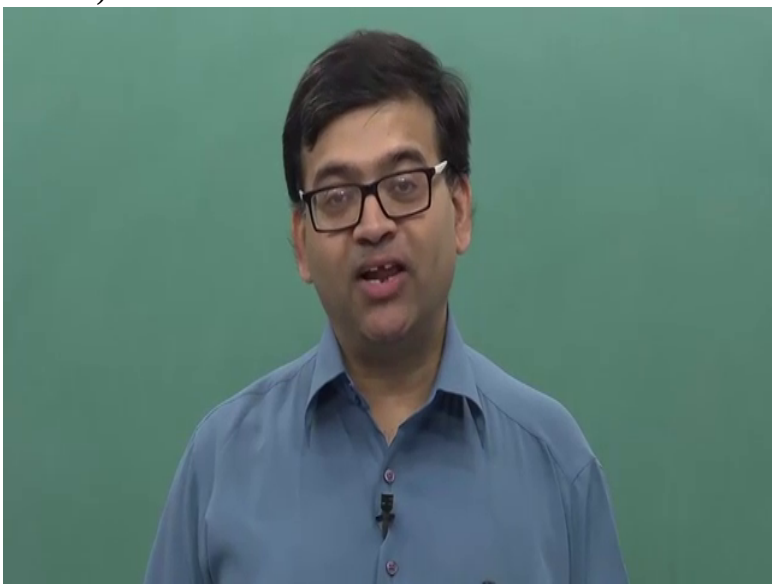
one survival path basically that you will get and this is my decoded code sequence. So corresponding to this received sequence, my decoded sequence is 0 0, 1 1, 1 0, 1 0, 0 0, 0 1, 1 1, Ok and this is basically shown here.

(Refer Slide Time 34:07)



So this is my backtrack path and this is my estimated code sequence. So this is how basically we decode a convolutional code using Viterbi algorithm,

(Refer Slide Time 34:23)



thank you