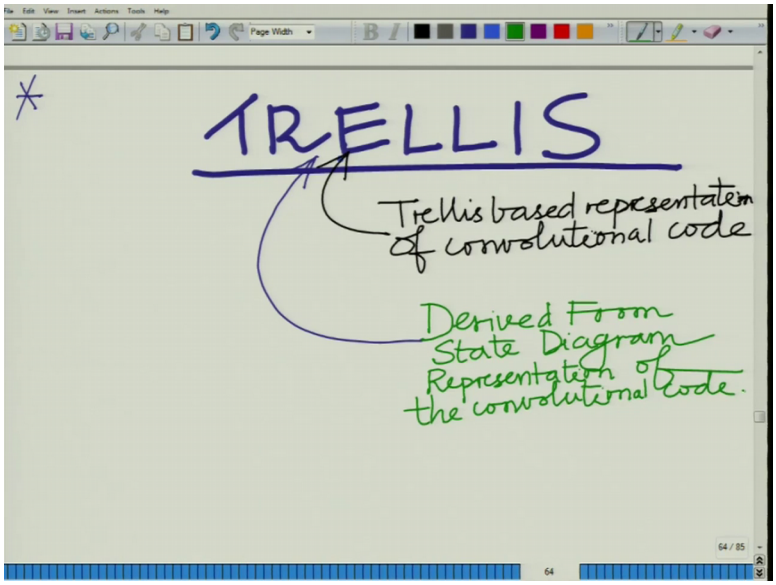
Principles of Communication Systems - Part II
Prof. Aditya K. Jagannatham
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

Lecture – 53
Trellis Representation of Convolutional Code and Valid Code Words

Hello, welcome to another module in this massive open online course. So, we are looking at convolutional codes and today we will introduce one of the most important concepts, the concept of a trellis, which is the one of the most fundamental, I would like to say aid or tools to represent and you understand the convolutional code. And it is derived from the state space representation the convolutional code. It is basically time evolution of the convolutional code. So, if you take the state evolution and unfold it or visualize it as a function of time, then what you get is the trellis based evolutions of the convolutional code all right.

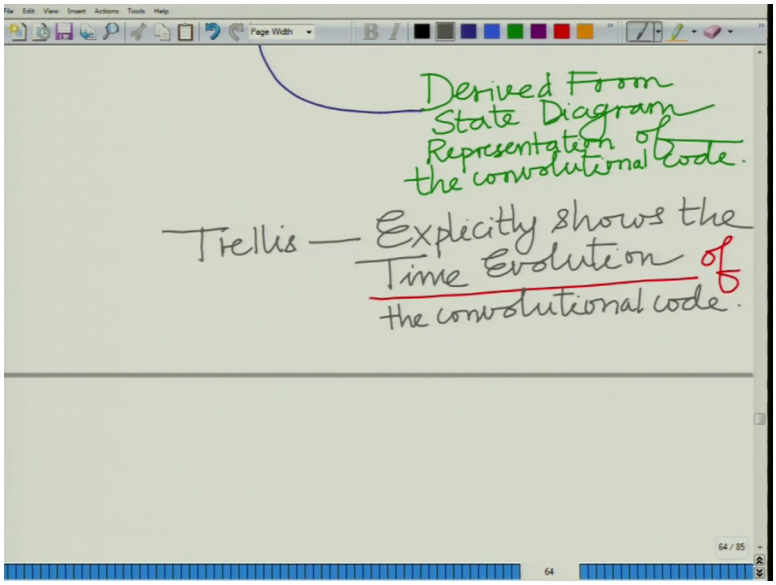
(Refer Slide Time: 00:57)



So, we will describe talk more about this as we go through the module. So, this is trellis a base representation talks about a correct, trellis base representation of trellis based representation of the convalutional code correct, and this is derived from the as I have said it is derived from the state, you have seen the state diagram or state space diagram we have seen the state evolutions diagram correct, we have the state diagram

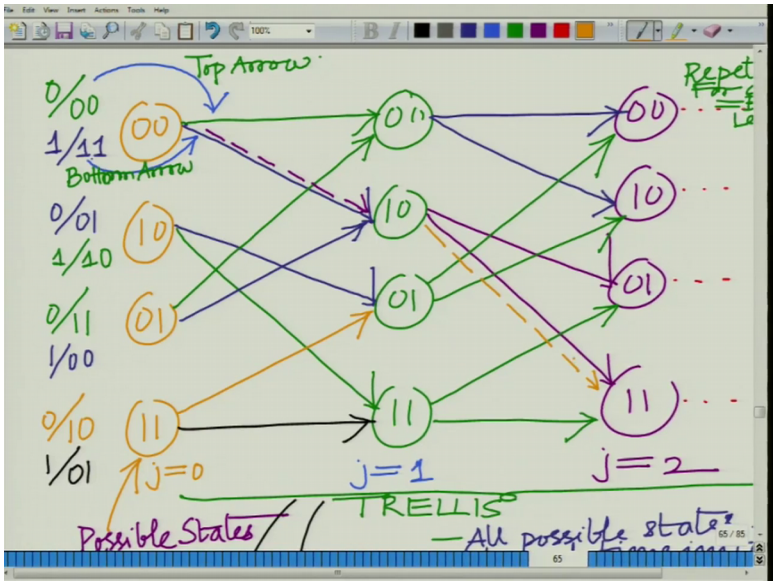
representation derived from the state diagram representation. Of the, it is derived from the state diagram representation of convolutional code. And as we have said a trellis,

(Refer Slide Time: 02:27)



This explicitly shows the time evolution of this shows with the it shows the time evolution of, it shows the time it explicitly shows the time evolution of the convolutional code.

(Refer Slide Time: 03:12)



So, I start again similar to the state space diagram, I start for the distance by drawing the all the possible states, correct? For instance one of the states is 0 0 other state or let me

just make some space here. So, 0 0, 1 0, 0 1, 1 1. These are all the possible states at time j equal to 0. So, these are what are these are possible states at j equal to 0. And then what I am going to do I am going to do separately draw remember in the states space diagram we had only one set of states; however, here in the trellis diagram for each time instance we are going to draw the state separately. So, this is the states at j equal to 1. Similarly one can draw the set of states at j equal to 2; these are all the possible states at j equal to 2. So, what I am doing remember, I am drawing all the possible states at each time instance I am drawing them in a line ok.

So, the first step is to draw all the possible state that is to represent right or to denote all the possible states at a given time instance. As per convenience these are represented in single line. So, that is typically of the trellis representation looks. So, all possible states, all possible states is instance.

Now, represent the time evaluation, from j 2 time instance j 2 time instance j plus. Now how to do this for a instance? Again similar to the state space now, now of cource as we have said this is derived, you can think of this as elaboration or an unfolding of the state. So, it contains no more additional information, but it simply a convenience, because it clearly represents the evolution that is the evolution of the this evolution of the convolutional code as time progresses that is, the state transition the various states variational state and the various states and the transitions between them as time progresses, that is what we mean by when we say the time evolution of the convolutional code ok.

So, now what we have here is basically 0 0. So, if the input is 0 then this goes to as we know from the state diagram, if the input is 0 it goes to 0 0. So, now, what we are saying is time to j is equal to 0 if it is in the state 0 0 on arrival of 0 it gives output that is c 0 0 0 c 1 0 is 0 and it transitions to the. So, this transitions to the state 0. And on the other hand if you have if you have the input 1 then it goes to the state 1 0 and the out puts are 1 comma 1, at time instanceinstance j equal to 1. So, this 0 0 you can see 0 input and output 0 0 this represents the top arrow. And the one written in bottom next to the state is for the bottom arrow. So, this is for the top arrow and this 1 output 1 1 is for the bottom arrow. It just convenient to write it to the left top the now, similarly 1 0 on arrival of 0 it goes to the state that is if input is 0 it goes to the state 0 1. So, that is the top arrow similarly now if I write the bottom arrow if the state is if the input is 1 output is 1 0 and it

goes to the state 1 1. Now for the state 0 1, instead 0 1 if input is 0 1 if input is 0 output is 1 1 and it goes to the state 0 0. And similarly if input is 1 output is 0 0 and it goes to the state 1 0.

So, finally, in the state 1 1 if input is 0 it goes output is 1 0 and it goes to the state 0 1 that is given by the top arrow and if input is 1, the output is 0 1 output is 0 1 and it goes to the state 1 1 that is given by the bottom arrow. That is given by the So, for each state at time instance j to each time instance j here j equal to 0 j equal to 0 j equal to 0 j equal to 1. All right from each state in j equal to 0 to each state So, at state j equal to So, at time j equal to 0 we have listed all the state at time j equal to 1 we have listed all the states and we are showing the transition all the possible transitions from j equal to 0 j equal to 1. And for each state at j equal to 0 there are 2 possible transitions that is one that is indicated by the top arrow, correct? And one that is indicated by the bottom arrow for instance.

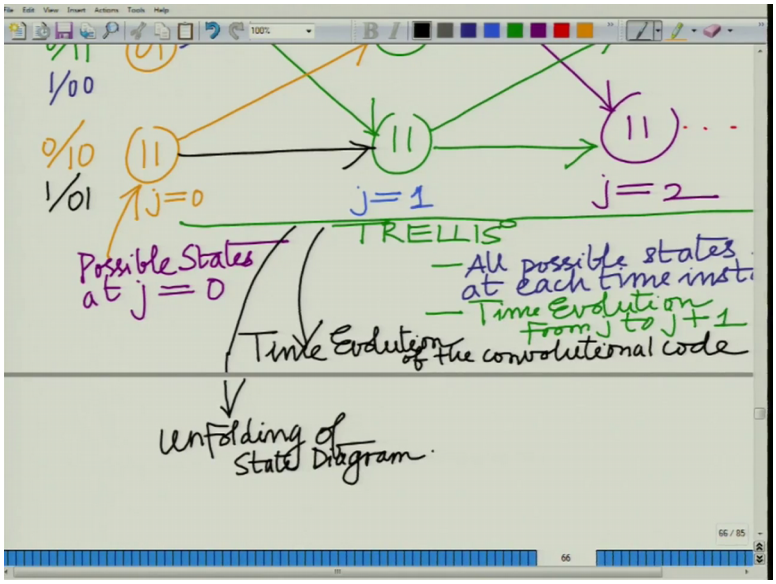
If you look at 1 0 the top arrow denotes that on arrival of input bits 0 the output bits are 0 1 and it goes to the state. For instance, you take 1 0 it indicates that on the arrival of input bit 0 the output bits are 0 1 and it transitions to the state 0 1 at time instance j equal to 1. And now this convolutional code remember this same state diagram holds true at every time instance. So, what holds for this first stage or you can say state 0 also holds true for stage 1, that is from time instance j plus 1 time in instant j equal to 1 to time j equal to 2. And similarly from time since j equal to 2 to j equal to 3, because state diagram representation at every time instance is the same, correct? And therefore, the evolution, correct? All the possible that transitions right are the same. So, this stage is repeated at every time instance.

So, what we have is again the same thing top arrow bottom arrow. I again no need to rewrite this because this is the same as what we have at j equal to 0 that is top arrow represents 0 output 0 0 bottom arrow represents 1 output 1 1 and transition to state 1 0 0 ok.

So, it is the same thing again 1 0 top arrow bottom arrow corresponds to input 1, correct? At 1 0, correct? At again 0 1 top arrow to 0 0 corresponding to input 0 outputs 1 1. Bottom arrow corresponding to input 1 output 0 0 and finally, again 1 1 top arrow corresponding to input 0 output 1 0 transitions to 0 1. Bottom arrow corresponds to input 1 output bits 0 1 and transitions to 1 1 and this can subsequently be repeated for all times

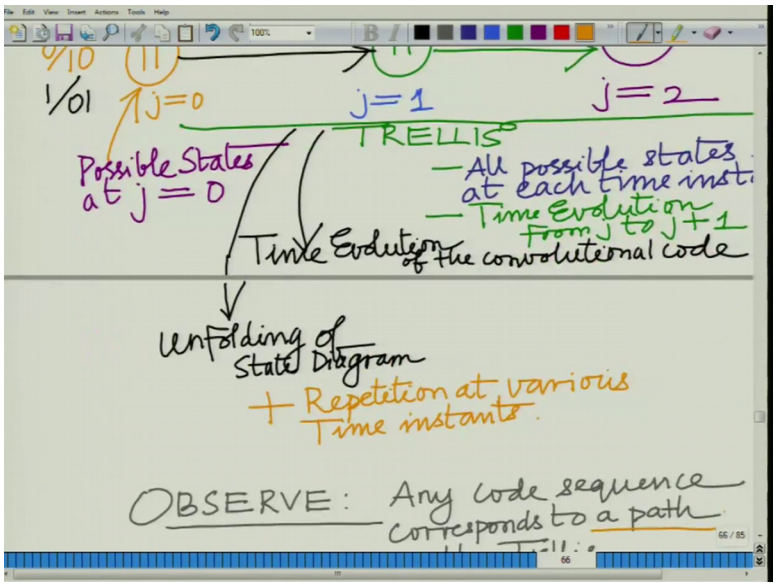
instance. So, again it is a repetition 4 time instance 4 various for all type equal to block length. You will have as many of course, you will have as many repetition as there are as there are input because for each input for each input bit you will add one layer of the states, to denote a transition from at time instance j that is a transition from corresponding to the input bit at time instance j transition from time instance j from the state at time instance j to the corresponding to the appropriate state at time instance j plus 1. So, you repeat this and this diagram is basically this, which is you can also think as the a unfolded state diagram and this is basically your trellis.

(Refer Slide Time: 13:15)



So, this basically shows the trellis diagram all right. Which is basically time evolution of convoloutional code. And as we said the trellis so, let us note that also trellis represents a time evolution of the convoloutional code. Let represent the time evolution convoloutional code, now further or you can also say this is the unfolded state diagram. It is an unfolding of the state diagram, you can say it is an unfolding, it is an unfolding of the state diagram ok.

(Refer Slide Time: 14:32)



That is basically unfolding of the state diagram followed by the reputation at various time instance, plus reputation at various time instance.

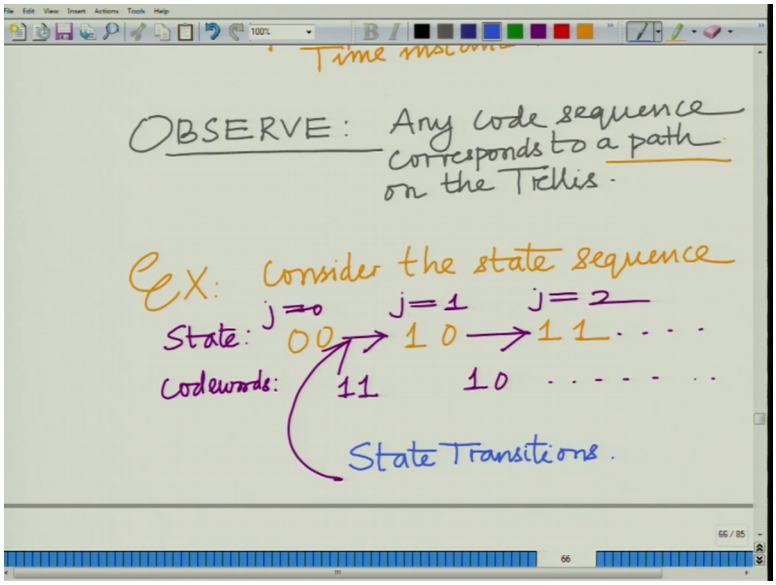
So, if you will look at the trellis it is a nothing but a block reputation of a state diagram that is where as the state diagram, it simply representation of the states and the transition. If you take the state diagram and represent it and repeat it sort of as a block at every time instance. So, basically sort of unfold this state diagram what you get is basically a trellis re presentation. And as you can see it is derived I mean from the state diagram representation one can derive the trellis representation from the trellis representation. One can go back to the state diagram representation all right. So, both are the equivalent except that trellis is the much more convenient way to visualize the time evolution of convolutional code ok.

Now, other important point is observe that in now observe and this is important, this is a very important point on which everything that we are going to discuss further. In convolutional code is going to observe that any code sequence observe, that any code sequence corresponds to a path on this trellis. So, any code sequence and this is important any code sequence corresponds to a path, what we mean by this is, for instance if he look at 0 0 1 0 1. So, let us say we start from 0 0. So, 0 0 and then we proceed to 0 0, a let say we proceed to first 1 0 and 1 0 we proceed to 1 0, we proceed to let say, 1 0

we proceed to 1 1 at time instance j equal to 1 we are at state 1 0 and then we proceed to 1 1.

So, now you can see this state sequence corresponds to a path in the trailers. And that path is basically nothing but what is denoted by, if you can look at the path that I am talking is from 0 0 to 1 0 to 1 1.

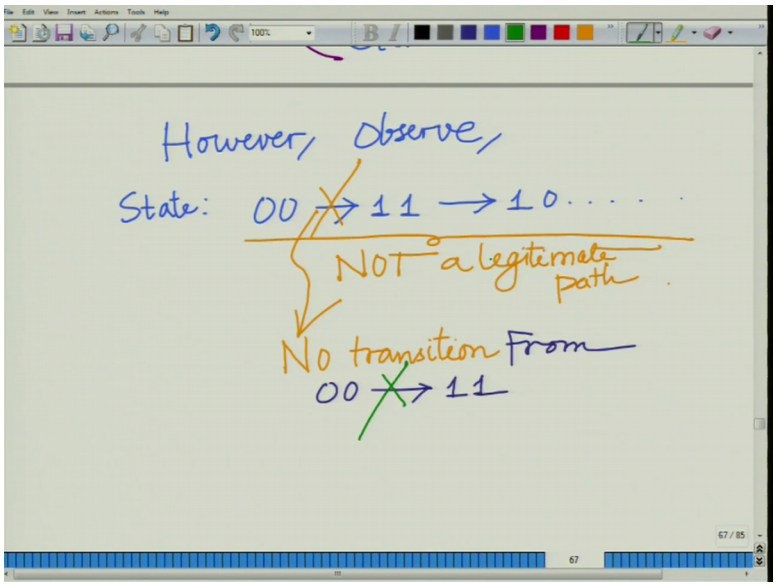
(Refer Slide Time: 17:43)



So, what you see is any code sequence corresponds to a path in the trellis, I want to emphasize this again. So, basically you can visualize it as for instance consider the state. For instance example consider a simple example, considered the state sequence 0 0 1 0 1 1 this is at time j equal to 0 this is at time j equal to 1.

This is a time j equal to 2 this corresponds to a path that is 0 0 to 1 0 to 1 1. And the transition 0 0 to 1 0 that generates the outputs. So, these are the state transition and these are the codeword generators. So, 0 0 to 1 0 you can see 0 0 to 1 0 that corresponds to the bottom arrow that generates, 1 1 0 to 1 1 that again corresponds to the bottom arrow at 1 0. So, that generates the output 1 0. So, this generates output 1 0, and as the states transition this continues. So, these are the code words that a generation and these are the state transitions, these are your state transitions, these are your state transition ok.

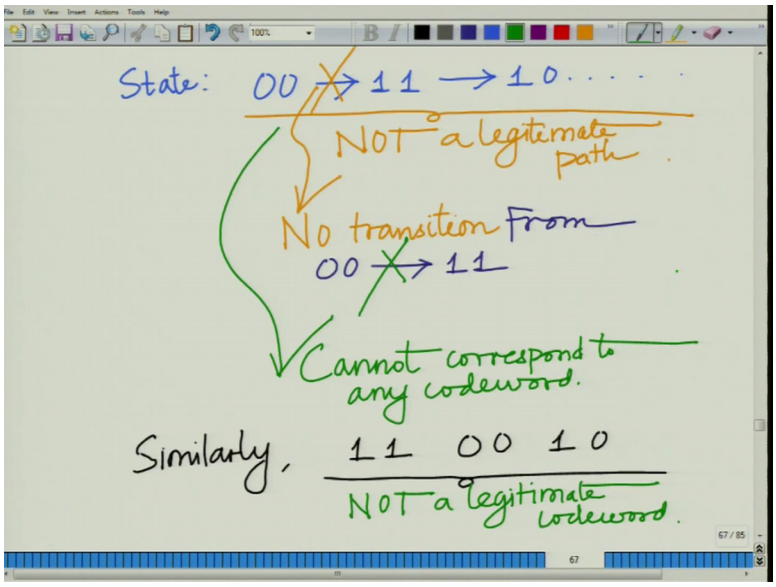
(Refer Slide Time: 19:24)



Now however observe that; however, observe 0 0 to 1 1 to 1 0. Now state 0 0 to 1 1, now observe this is not a legitimate path. This is a not a legitimate path, because if you start with 0 0 there is no path no transition that can take you to 1 1, correct? 0 0 there is no path to 1 1.

So, this is not a path this is not. So, there is no transition from 0 0 this does not exist hence.

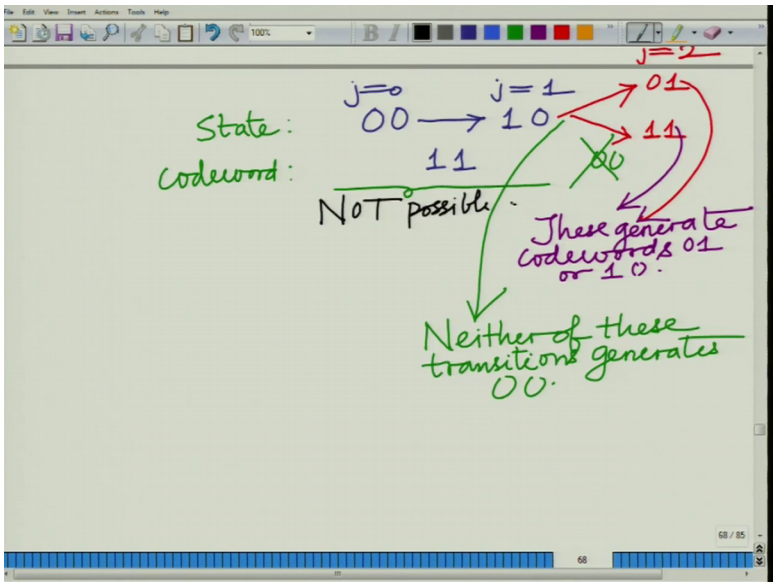
(Refer Slide Time: 20:53)



This is not to not a legitimate part. And there for cannot generate any code, cannot generate and therefore, cannot correspond to any codeword. Can not correspond means, by which mean that it cannot correspond to any legitimate codeword that is generated by this trellis. That is a point. Now similarly let us look at a codeword word see we have looked at a state sequence. Now let us look at code sequence. So, I will look we will look at it similarly from the perspective of a codeword sequence ok.

So, similarly 1 1 0 0 1 0 this is not we will show that this is not a legitimate codeword, this is not a legitimate code. For instance you can see 1 1 0 0 1 0. So, 1 1 so, let say we start at state 0 0 we generate codeword 1 1, which means the transition to 1 0. Now in 1 0 there is no input which generates the codeword 0 0. You can either generate only 0 1 or 1 0 correct. So, basically let us say you start with the state.

(Refer Slide Time: 22:42)

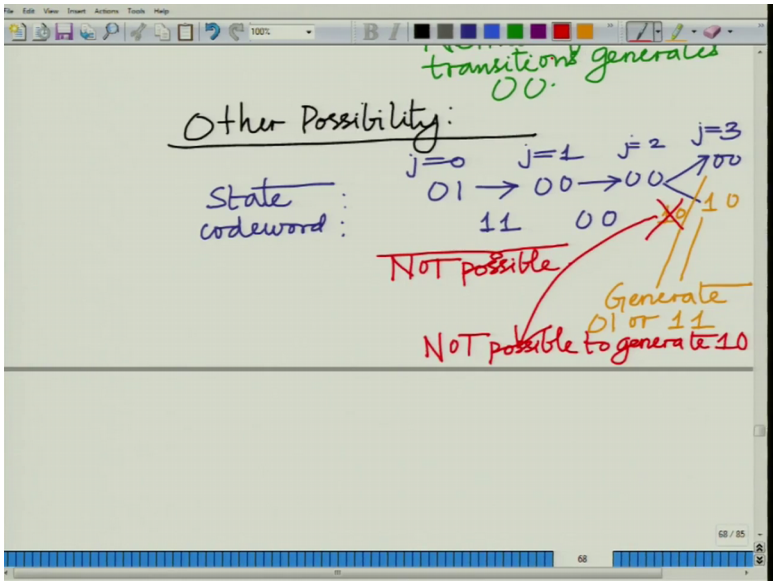


So, we are looking at codeword 1 1. So, let say you start at the state codeword. So, let say you start at the state 0 0 you proceed to the state 1 0 . So, this is at j equal to 0 this is at j equal to 1. You are generating the codeword 1 comma 1, this is fine this is but; however, from 1 0 you can only go to states 0 1 or 1 1, at j equal to 2 this corresponds to j equal to 2 you can either got to 0 1 and 0 1 and these generate code words, if you go to 0 1 or the from 1 0 if you go to 0 1 or 1 1 de generate code words 0 1 or 0. So, it is not possible to generate it is not. So, these generate code words and these generates code words either 0 1 or 1 0. It not possible to generate the code word and these generate .

Therefore, it is not possible to generate both neither of this transitions generates 0 0 correct. So, neither of these transition generates 0 0.

Now, similarly on the other hand if you can start at 0 1, the other possibilities you can is you can start at state 0 1. So, if you start at state 0 1 you are generating code word 1 1 which means you are going to the top brands 0 0. So, 0 1 if you start so, this is not possible. So, this combination not possible, they we have eliminate this. On the other hand other possibility.

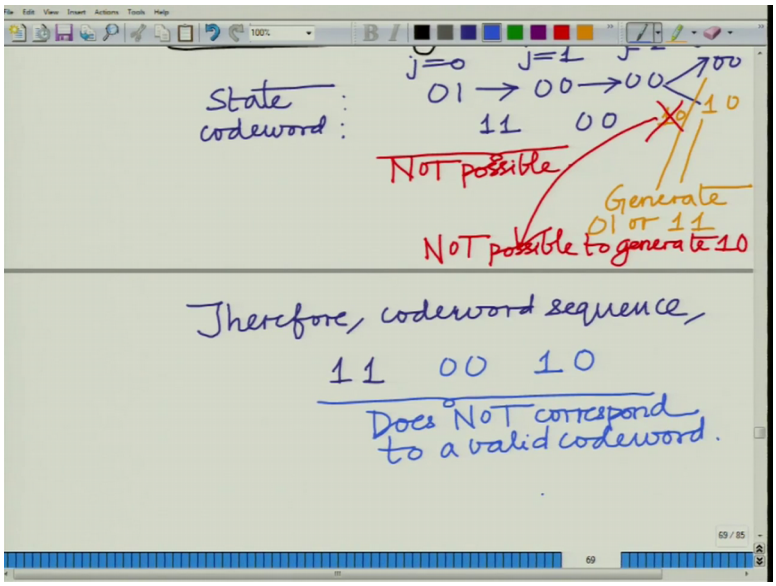
(Refer Slide Time: 25:24)



Now, let look at the other possibility. Other possibility is we look at the state and codeword state is let see 0 1 at j equal to 0, transition to well 0 0 and j equal to 1 that generates 1 1, sub 0 0, you transition to 0 0 at j equal to 2. Your transition to 0 0 that generates 0 0, now from 0 0 you can either transition to 0 0 or this is at j equal to 3. You either transition to 0 0 or 1 0 and both these generate. And these generate either 0 0 or 1 1. So, it is not possible to generate 1 0. So, none of these transitions generates not possible to generate 1 0 .

So, it is not possible to generate 1 0 therefore, once again you see that this is not possible not possible, again not possible.

(Refer Slide Time: 27:29)

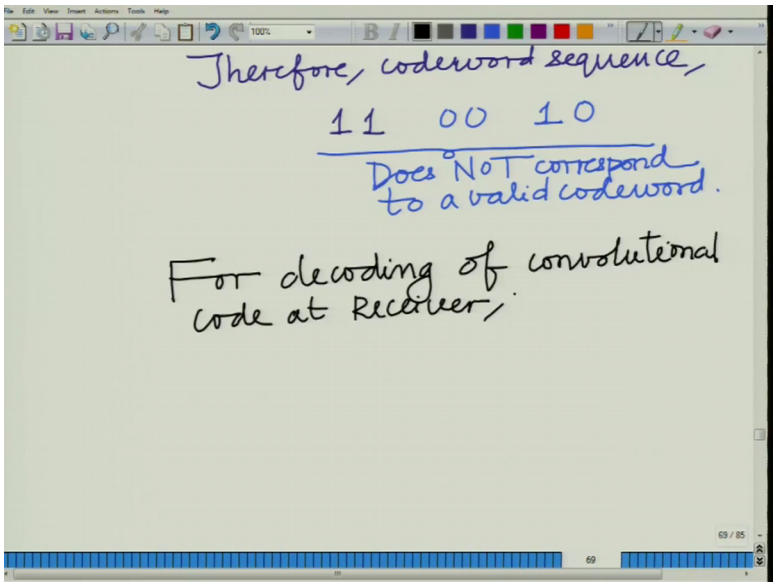


This is not possible. So, therefore, codeword sequence therefore, the codeword sequence 1 1 0 0 1 0 this does not correspond to a legitimate code word. Or you can say does not correspond to valid codeword. This does not correspond to a valid codeword. And therefore, the most important thing that you have to realize is that if you are given a codeword right. If someone tells you that this is a code word that is a corresponds to this particular convolutional code then, that has to be generated by a state sequence that has to correspond to a path. By a path means that has to correspond to a certain state sequence in the trellis right. That has to correspond a certain evolution of the of the convolutional code characterized by a sequence of states at the various time instance in the trellis. So, I have to map any so, it is a so, given so, given a state sequence or a path to the trellis that corresponds to codeword valid codeword and more important.

So, this forward direction is fine, that is how we are generating, that is how we are generating the convolutional code. Now the revere is also important that is if you have a codeword that has to if it is a valid codeword that has to correspond to a legitimate path or a legitimate sequence of states through the trellis. Now that is important this inverse mapping from codeword to the path sequence to the trellis is important, when you want to de code remember at the transmitter we coding. So, all are going is basically corresponding to a information bit sequence, I am generating the code words or the code word ok.

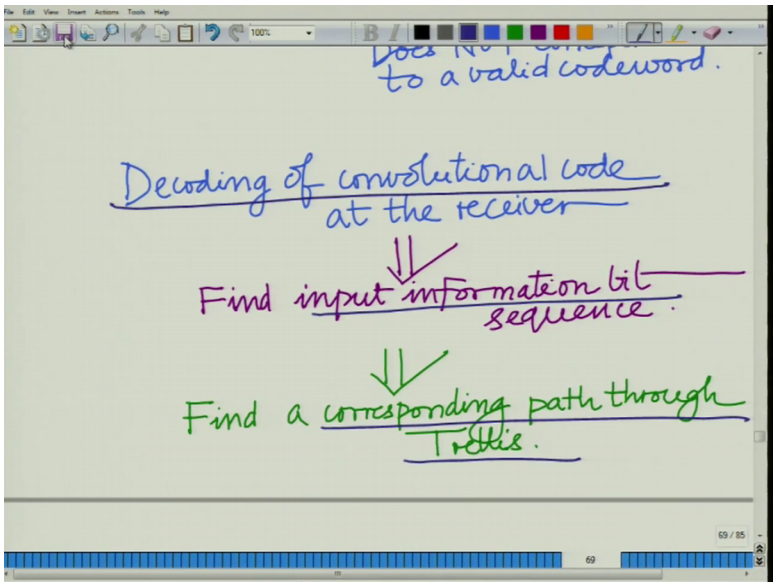
Now, at the receiver I have to precisely do the opposite that is given the code word sequence I have to map it back to an information or an input bit sequence. For that it is important to realize that basically you have to map it back to a corresponding path to the trellis that is the importance of this point.

(Refer Slide Time: 30:13)



So, for decoding of convalutional code at the receiver, for decoding of convolutional code at the receiver right.

(Refer Slide Time: 30:46)



For decoding of convolutional code at the receiver or let us put is this way decoding of convolutional code at the receiver, this implies that I have to basically find in full input find the input information sequence, which basically intern implies that I have to find a path or map it to a find the corresponding path through the trellis.

So, for decoding of the convolution code I have to find the input information bit sequence, which means have to find corresponding path through the trellis. And that is what convolutional code that is what the decoding of a convolutional code is all about. And this is what we are going to spend time trying to discover or trying to infer trying to infer that is, basically how can one do this inverse mapping that is of course, we have looked So far we have looked at a the generational for convolutional code the representation states way representation from that the trellis base representation and the generation of the convolutional code word correct.

Now, the other thing that we are going to look at is given a received code word how to map it back, how to map it back to an information bit sequence. Or to a to a path in the trellis and from the path of course, one can once you have the state transitions that is the we sates at the various time instances and a valid state transitions, that is you have a valid path the trellis, and you can derive the information bit sequence correct. So, like the path right, the path can be used to automatically derive the information bit sequence. Because every path every transition corresponds to an input bit and the generated output code bits. So, that is basically what we are concerned with. And which is very important in to completely analyze convolutional code. And to understand the decoding aspect of convolutional codes all right. So, we will stop here and look at other aspects in the subsequent module.

Thank you very much.