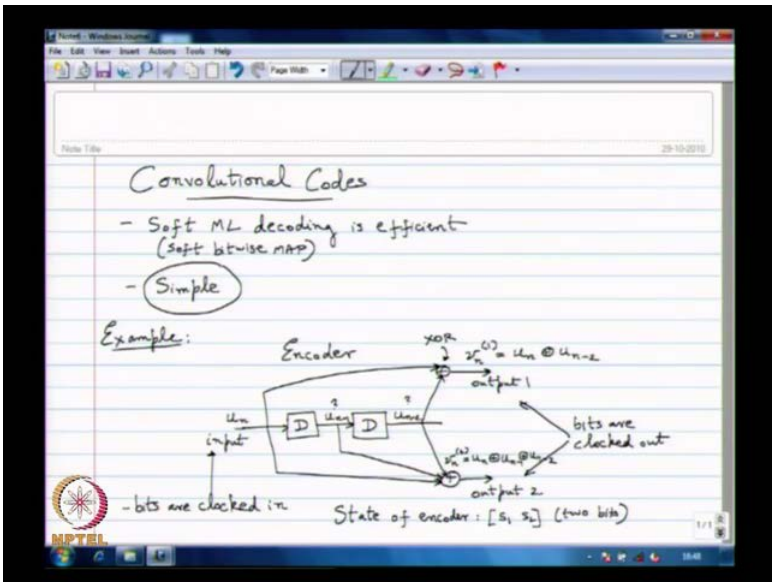
Coding Theory
Prof. Dr. Andrew Thangaraj
Department of Electronics and Communication Engineering
Indian Institute of Technology, Madras

Lecture - 30
Introduction to Convolutional Codes

So, we are starting a brand new topic, and we had a brand new writers exchange shift my position little bit it is better.

(Refer Slide Time: 00:28)



So, we are going to start talking about convolutional codes and towards the end of last lecture I gave you a simple high level reason for why this is interesting it turns out soft M L decoding can be done efficiently. And of course, soft bitwise MAP is also efficient so both of these are efficient for convolutional codes so that is the main selling point for these codes and we will on top of this they also have an amazing advantage that they are really simple. So, they are the simplest possible codes and not simplest possible codes of course, but a very simple to describe and simple to describe the decoding very intuitive and very nice and clearly understand what happens with these things quite easily.

So, we are going to begin with an example so I will give an example of a convolutional code and it will be a very typical example any other convolutional code will be a minor modification of that you will also define it generally later on, but an example is the best place to start. So, let us begin with an example and I will start by drawing an encoder so

instead of just defining the code so if you remember till now we are defining the code directly, code is the list of all code words. Now, I will define the encoder for this code of course that also defines the code in a way right so it is not a bad definition.

So, the encoder that I am going to draw will involve d flip flops all of you must be familiar with what a d flip flop is basically it is a delay element it will have a memory of 1 bit and will hold it for 1 clock cycle. So, it will be like a clocked sequential digital circuit so let me draw that. So, I have 2 d flip flops and they are connected up like this there are 2 outputs and the second one looks like this, so this is kind of an ugly looking diagram of the encoder I am sure you will find better pictures of the encoder. So, this is the input and this is output 1 and output 2.

So, this is the encoder so what happens is bits a clock 10 from the input side at every clock you send 1 input bit in and the clocked out on the output side so that is the that is the high level picture here. So, I think you have seen enough digital systems to know that that what is happening here, but anyway let me emphasize that once again which a clocked in and clocked out if the input side the output side bits are clocked out. Of course, the clock is not shown it is suppose to be there, I am showing it explicitly anywhere it is suppose to be there where is a clock which is running and every tick of the clock or every stage state of the clock it is going to keep shifting a bit inside, and an output bit is two output bits are going to be come out.

So, we need some notation I will call the nth input bit as some u n, so at the nth clock the input is u n so this are all bits. So, what will be the what do I have to write here what will what will happen here u n minus 1, so it would have been the input in the previous clock it would have after a delay it will be there what about this k here u n minus 2. So, once you figure this out it is a simple d flip flop you know exactly what is going to happen the outputs are also easy to write down. So, we have 2 outputs so will have a notation like this for the nth output v n 1 this will be u n xor u n minus 2 by the way this is xor. So, I think it is standard so I have not written it down.

So, the second output is what v n 2 is going to be u n xor, u n minus 1 xor, u n minus 2 so that is the second output bit. So, I am sure you agree with me on this word simple so it is a very simple code, I mean if you look at the LDPC code. For instance its encoder is definitely not going to look anything remotely close to this, it is going to be quite big and

you have to remember a lot of things and you will do a lot of things well things for that, but for the convolutional code the encoder is so simple and this is a main reason.

This is a huge advantage in practice several times so you might want some system where the transmitter is maybe very weak so to speak it is relying on let us say some very poor energy source it cannot sustain for a long time. So, you want to have an encoder which is really, really, really simple and this will satisfy that constraint. So, if you look at the number of gates in this implementation how many gates are in a d flip flop, any some number which is you can count it one hand right so it is small number. So, you have 2 d flip flops and 2 xor's. So, both ten fifteen gates you will be able to easily do this entire implementation and there will be a clock and it is it is it is dirt chips are given it will work on and on forever. So, that is a big advantage with convolutional codes.

So, the big point is are they any good how well do they do in a bit error rate or block error rates situation. So, we will see when you implement soft M L decoding and in fact implement bitwise MAP or any other decoder it will work really well we will see that also. So, this is the picture hopefully this picture is clear now you can see some generalizations of this picture. So, in general how will a convolutional encoder typically look you will have more than 2 d flip flops you can possibly have some mu d flip flops and then how many outputs can you have as many as you want maybe 3 or 4 or 5 then let us 1 input always.

And then it is just a question of connecting up the different points of the d flip flops outputs to these xor gates and getting your output. So, that is the generalization if I have to present it in the most general form there will be a lot more notation but definitely not more not much inside is lost with these example so you see it to be you get the idea of what is happening here. So, the next important thing to remember is this seems like an encoder which is going to take a stream of input and keep on putting out a 2 streams of output, but in practice with codes it is not going to work this way it is going to be a block of message bits some k bits and they have to be converted into some so many bits before transmission.

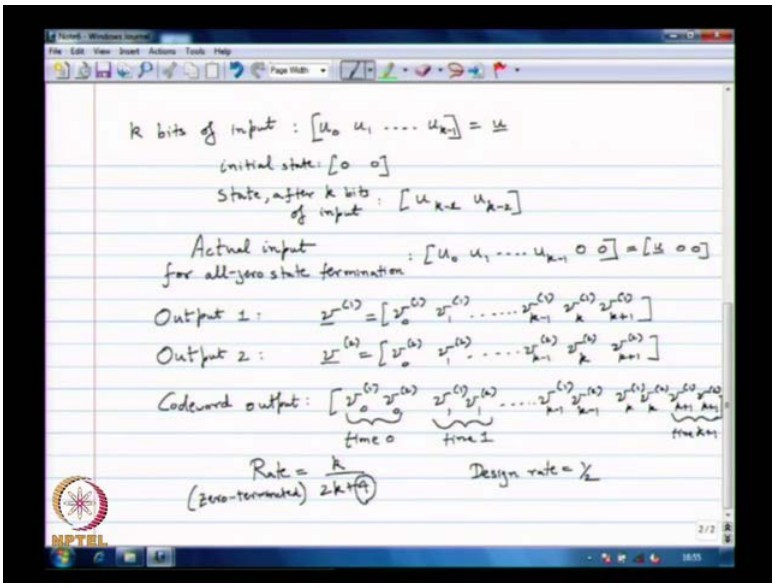
That is one important constraint in practice right so you cannot you cannot keep on, on and on encoding all the time right. So, there are other things that have to happen communications has to started and stopped and so many other things will go wrong if

you have it like a stream so you have to stop and start. So, we have to convert this picture into a block coding picture that is one thing and the other thing is there are 2 streams of output and they have to be interleaved also and sent out all these things have to taken care of in an actual operation. So, let me write down how that works first and then we will see other descriptions of what happens here.

So, the input you can think of as, so first thing I should start out with is the state of the encoder so what is the state of the encoder. So, it is the state of any digital system what is the state of digital system to look at all the outputs of the d flip flops put them together how many ever bits you have that is called the state. State of the encoder is basically 2 bits so you can call it s 1 s 2 if you like so basically it is 2 bits in this example of course, and in general it will be as many bits as the number of d flip flops that you have. So, first question in an actual implementation is what is the initial state so it is typical to take the initial state as 0's, so you are going to put initially 0's in your d flip flop output.

So, we will assume that the initial state is 0 in implementation and then you have so many numbers of bits coming in so let us say k bits of input are going to be clock 10. So, it is very common and very important to terminate your, term terminate your state you terminate your encoder finally, in the all 0 state once again. If you keep clocking in k bits at the end of k bits state is not guaranteed to be in the all 0 state. So, usual in convolutional codes to manually put in 2 further bit inputs to force this state back to all 0 state so that you start at the all 0 state and end at the all 0 state. So, that is good for a lot of descriptions and in practice that is always done so that is called termination so we will do that in practice as well, we will do that when I describe it.

(Refer Slide Time: 11:18)



So, if you have k bits of input so it is a bit tricky how I write it so I am going to write it as u 1 u 2 all the way to u k, but what does this mean which bit will go in first u 1 will go in first. So, maybe it is good to write it as u 0, let me write it as u 0 sorry about that I will start with u 0 and then go all the way to u k minus 1 so that is better, u 0 u 1 all the way to u k minus 1. So, this is my k bits of input my initial state is going to be 0 0 so the final state will be what after k bits, state after k bits will be what.

Student: k minus 2.

Yes, after k bits of input will be u k minus 2, u k minus 1 u k minus 2 and I do not want this so this is not this is you cannot end in a random state it turns out it is bad for the decoder. So, you want to go back to the all 0 state so you have to add along with the k bits of input 2 additional terminating bits which in this case the way I have drawn it will always be 0 0 so that you can take it back to the initial state. So, the actual input will be actual input for all 0 state termination will actually be u 0 u 1 all the way to u k minus 1 and then I will add two 0's.

In the general case, when you have let us say mu d flip flops how many 0's will you have to add mu 0's that is all that is no problem so you need do that, that is what will happen. Now, let us look at the output this is being clocked 10 so the output, output 1 so maybe we can call this as some vector u and this will be u 0 0 so you can do that. The output 1 will be so we not write n the vector v 1 it will be v 1 0 v 2 0 I am sorry did I get that right

no, v 1 1 so until v 1 k minus 1 can I stop here no. So, when I put in the 0's also I will get some output and I have to retain that also if I want a complete picture I have to retain that also.

So, in fact you will have a v 1 k and a v 1 k plus 1 and the output 2 will be v 2 and that will again once again be v 2 0 v 2 1 v 2 k minus 1 v 2 k and v 2 k plus 1. So, the actual code word output that is actually going to be transmitted will be an interleaved form of these 2 things it will be basically v 1 0 v 2 0 v 1 1 v 2 1 I will write one more so let me not write one more it is good enough all the way to v 1 k minus 1 v 2 k minus 1 v 1 k v 2 k and then v 1 k plus 1 v 2 k plus 1 that is my actual output.

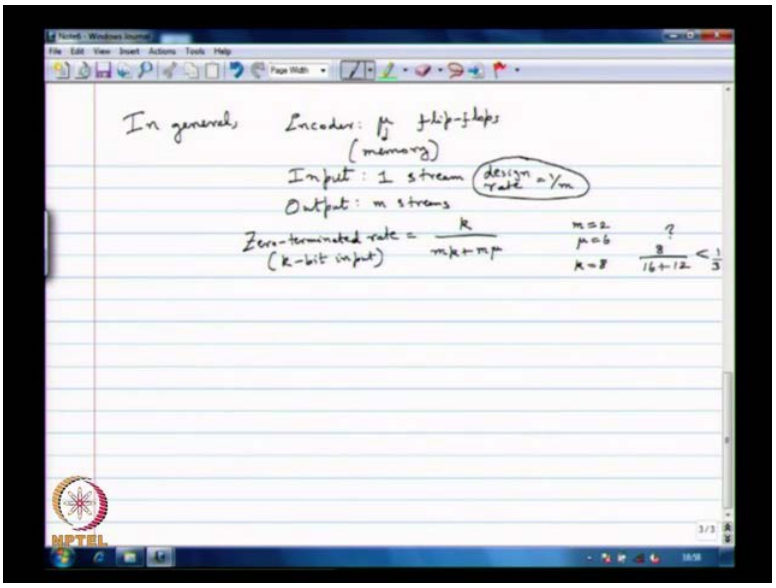
So, these two are the outputs corresponding to the first time instance 0th time instance, these two are the outputs corresponding to the second time instant time 0 so to speak, if your index time this will come out at time 0 time 1 and so on and the last one will be time k plus 1. So, your message was actually k bits long that is the message that you have you cannot say the 0 0 is the message so it is not any random message it is know at the receiver so it is only k bits long. How many bits of code word do you have.

Student: 2 k plus 2.

2 k plus 2 so the actual rate of this code in this 0 termination operation a 0 terminated rate is actually k by 2 k plus 4. So, you have this 4 additional bits that have to be sent for k bits. So, this is the 0 terminated rate while the design rate is what design rate is half so if you look at the encoder, the encoder rate is half for every t every time instance it seems to be doing half, but when you actually operated you will get this penalty so there will be this penalty because of the 0 termination that is ok.

Why is not why is 4 not a big deal it is like it is it is obvious to many people if k becomes 1000 then clearly the 4 is not going to hurt you at all, but remember, but keep in mind mean you may not know this, but convolutional codes are even used with very low block lines k could be 8 or 10 or something and with that case things like 4 will matter a lot. So, in general so this is the picture for this specific rate half code.

(Refer Slide Time: 18:11)



In general, if you have let us say so in general if you want to generalize it so it is quite easy to that you might have an encoder with, encoder might have mu flip flops if it does so then it is called since mu is called memory so for obvious reasons it is called memory of the encoder. So, the encoder might have mu flip flops and input we will keep as 1 for the purposes of this course we will keep as 1 of course, there is other theory for more than 1 input also it is possible to have 2 or 3 inputs and then you need a slightly more complicated picture, but for this course we will keep input as 1 just 1 stream. In output in general can be some m streams right so if this is true the design rate is 1 over m so that is the design rate what will be the 0 terminated rate.

Student: (( ))

So, these are very easy things to figure out, but anyway let us do that once and then I will do that. It is going to be k or any for k bits of input so 0 terminated rate actually depends on the lengths of the message for k bit input k by m k plus

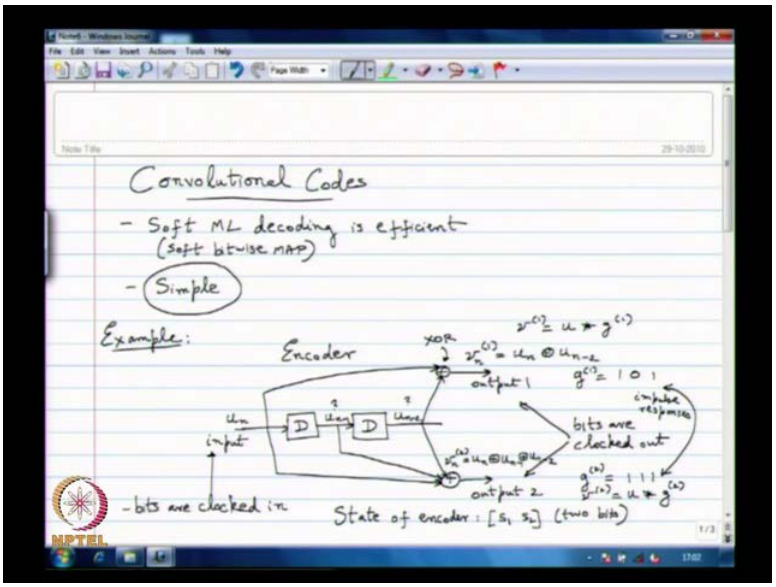
Student: m times k minus, m times mu

m times mu, so if you put numbers like for instance m equals 2 mu equals 6, mu equals 6 is a popular number in practice, in many standards mu equals 6 is a popular number, m equals 2 is also a popular number. If you put k equals 8 what happens to the design rate were might be 1 half, but the actual rate is going to be 8 by 16 plus 14, 14 no 12 my

multiplication tables are getting rusty. So, even though you are saying the rate is 1 half because of this extra 12 bits what does the rate actually becomes, become it is become less than a third it is become less than a third so this is less than 1 by 3 oh my god less than 1 over 3.

So, this is a problem in practice people deal with it differently, but anyways so we will just assume 0 termination all the time so I things like 8 are dangerous, 8 bits of input are dangerous when you want to do 0 termination you may not have real control over your rate. So, hopefully the operation is clear so let us go back and look at the encoder circuit once again.

(Refer Slide Time: 21:13)



So, in general for a convolutional encoder if you have rate 1 over m and memory mu there going to be d flip flops, 1 input stream and if you have m outputs m different xor gates at which some connections are made so that is the idea. So, the question is why is this called convolutional

Student: convolving with some streams of 1's and 0's of 1 stream.

All of you must seen maybe hopefully enough linear system theory. So, this is basically some kind of a very linear system and you can think of an impulse response it is a discrete time linear system except that you might be used to systems where the impulse response is real numbers. So, you use to real numbers here the difference is it is all bits

everything is modular 2 other than that everything else is the same, is the same standard linear function theory will apply. So, if you send an impulse as input what does an impulse as input, 1 followed by a bunch of 0's you will get a certain response at output 1 and output 2 that will be impulse response 1 and impulse response 2.

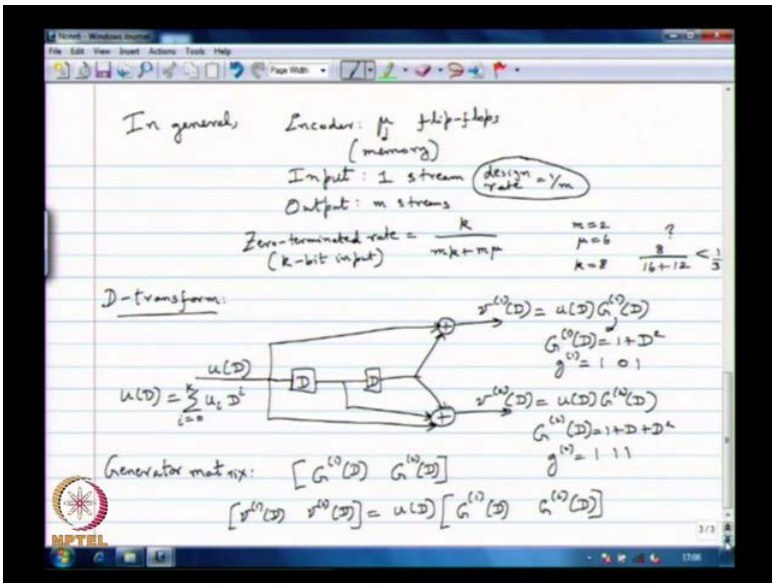
If you now do an arbitrary input how can I compute the output simply take that input and convolve with the impulse response you will get the output. So, it is all linear same thing will hold same thing you can do at output 2 also you have an impulse response corresponding to output 2 and then you do that. So, what are these impulse responses let us try to find them maybe I call it g 1 what is g 1.

Student: 1 0 1

So, it is just going to be 1 0 1 so that is it, what about g 2 1 1 1 so these 2 things are the impulse responses. In general if I want to find the input for u n if you take u n and then convolve it g n 1 so this sequence u, so if you think of the sequence itself as u the sequence v 1 is simply going to be the sequence u convolved with g 1. So, the sequence v 2 is going to be u convolved with g 2 so these are standard ideas not too difficult. So, the convolutional is also very easy to do basically this convolutional will mean u n plus u n minus 2 this convolutional will mean u n plus u n minus 1 plus u n minus 2 the exactly the same thing we just saying it in multiple different ways.

So, the nice thing about convolution is you can also think of it as polynomial multiplication, so you go to this transform domain usually you do the z inverse. So, z inverse is common when you do real numbers when you have binary and all that instead of z inverse people usually use d capital d so you can go to the d notation which is the d transform so to speak of this impulse response and get capital g 1 of d and capital g 2 of d. And then you do a g transform, d transform on u to get a u of d how will I find v 1 of d simply do u d times v of d so that is the idea. So, let us let us do that a little bit more with one more examples.

(Refer Slide Time: 24:40)



So, let us do this this is, this is we are trying to do d transform notation so to speak I mean I have just say it is notation it is nothing more than that at this point. So, we get 2 impulse responses so you think of the sequence u n as u of d what is u of d now, u of d is basically summation u i d power I, so i will go from 0 to wherever so 0 to let us say capital, I do not know I will just put capital k here, just to me keep something definite. So, I have 2 d flip flops 1 output is these 2 things and the other output is all 3.

So, I can write v 1 of d the d transform of the output here as u of d times capital g 1 of t what will be this guy g 1 of d will simply be 1 plus d squared so remember g 1 was simply 1 0 1 so the d transform will be 1 plus d square. So, it is all a same thing I mean just saying the exact same thing the basic thing I am saying is v 1 n is u n plus u n minus 2. You can write in so many different ways another way of doing it is to write it as u of d times g 1 of d where g 1 of d is 1 plus d square it is a d transform notation. When you learn it in DSP I am sure they told you that this is the greatest revolution ever in signal processing, but that is all very simple stuff no means it is not very complicated.

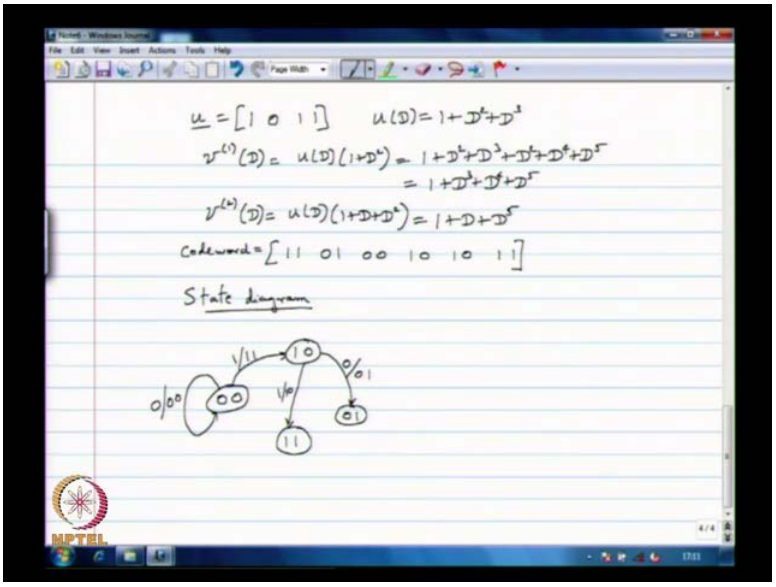
So, u of d u of d times g 2 of d and this g 2 of d will now be 1 plus d plus d squared which corresponds to the impulse response 1 1 1. So, really no big deal here and you get the same equation back I mean if you just plug it in you get the exact same case back is there a question

Student: (( )) k minus 1 or…

I have just put a capital k there it is not it is not the small k it is not related to the previous k. So, this is some maybe if you want I will give some other notation there I just want it something just to define what the d transform. So, how many ever you have their you have to do it because you 0 terminate you have to put the 0's also, but the 0's do not show up in the d transform anyway so it does not matter, alright. So, this is the transform notation so there is again once again a short hand notation for this so what you do is you can do d transform generator matrix so this is this is pretty important because you will get some nice tricks that you can play once you do this.

You have a generator matrix and d transform that is basically g 1 of d, g 2 of d. What does this generator matrix do for me if I want to find v 1 of d and v 2 of d what can I do, I can simply take u of d and multiply it with the generator matrix g 1 of d g 2 of d. So, this is for this simple and nice reason is called the generator matrix. So, let us maybe do one quick and dirty example.

(Refer Slide Time: 29:02)



Let us say u the input code word so I am going to use all these notations interchangeably so if a put u with a bar underlined it means thinking of it as a vector if I u of d I am thinking of it as the polynomial in d transform notation etcetera. So, let us say u is 1 0 1 1 go ahead and tell me what u of d will be what is u of d 1 plus d squared plus d power 3. So, what is v 1 of d going to be basically u of d times 1 plus d squared so I have to multiply u of d with 1 plus d squared so it is quite simple if I do that I am going to get 1

plus d squared plus d power 3 plus d squared plus d power 4 plus d power 5 and that will be 1 plus d power 3 plus d power 4 plus d power 5 it is not too difficult to figure out.

And v 2 of d you can find very similarly, it is going to be u of d times 1 plus d plus d squared so it is basically this guy plus d times that. So, it is going to be 1 plus d plus power 5 so if you do the multiplication you will get that. So, from here if you want you can figure out the code word output what will be the code word output.

Student: 1 1 0 1

1 1 0 1 0 0 1 0 1 0 1 1 what will happen after this how many should I put out that is it that will be the end seems a little surprising, but you see now why the 0 termination is nice you know you have to do the 0 termination because this polynomial formula will work very nicely we did not do it will be a little bit more complicated. So, we have made the polynomial formula can work with a 0 termination that is one way of one fancy way of thinking about very simple idea.

So, the next thing that is done usually in digital systems is some kind of a state diagram right and state diagram here will be quite simple so you know there are only 2 bits states. So, let us do a state diagram there are only 4 possible states 0 0 from 0 0 what states can you go to remember I have a sequence of d flip flops and I am flipping I am bringing in bits from the left so how do I do a state diagram, I have to now think about what next state I can go to, but basically what input will come in I am in state 0 0 what input will come in, it can either be 0 or 1 if a 0 comes in what will be my next state I will continue to be in 0 0.

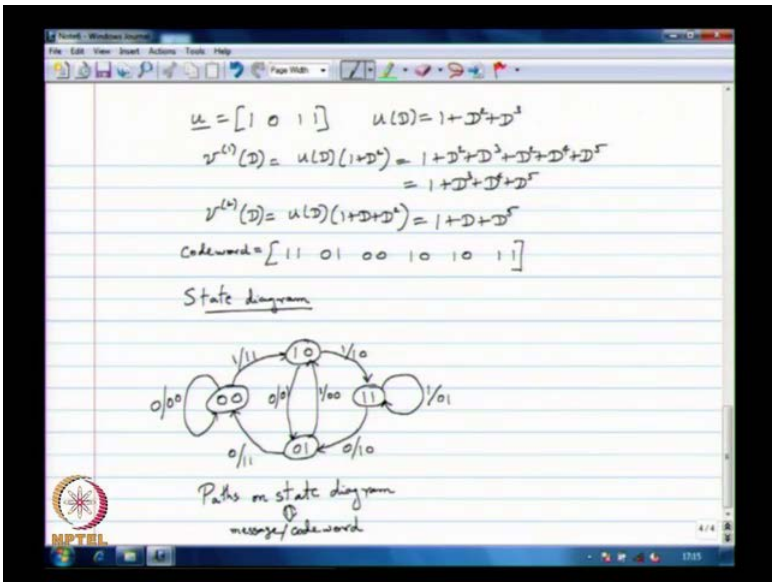
What will be my output I have state 0 0 and then 0's coming in my output will be both 0's so this is a very standard way of denoting a state diagram. In case I am in 0 0 and an input 1 comes in what state will I go to 1 0 so of course, the convention is important I am going to write always s 1 s 2 the first d flip flop first you can also do it [FL] then everything will change so but I am going to write it as s 1 s 2. So, the next state can also be 1 0 if the input is 1 in which case the output will be 1 1 just a question of looking at that state machine and figuring out what is happening it is a simple d flip flop so it is not too hard.

Now, from 1 0 what can happen input can be 0 which means I will go to…

Student: 0 1.

0 1 with an output of 0 1 or input could be 1 in which case I will go to 1 1 and the output will be 1 0 maybe my picture is going to look really ugly, but anyway it will be accurate so the only thing I can guarantee. Now, from 0 1 what can happen if my input is 0, 0 0 so I am going to make my picture little bit more beautiful.

(Refer Slide Time: 33:54)



So, let us say we do this differently so we will say if my input is 1, if my input is 0 I go to 0 1 with an output 0 1 and if my input is 1 I go to 1 1 with an output 1 0 so this will make the picture a little bit better. Now, from 0 1 if the input is 0 I am going to go to 0 0 with some output and if the input is 1 I am going to go back to 1 0 with some output and at 1 1 if the input is 1 I am going to stay here, if 1 1 the input is 0 I am going to stay I am going to go here and fill out the corresponding things I think this will be 0 0 and the 0 1 so this will be 1 0 and if this will be 1 1.

So, this is a state diagram that you can fill out so I did it rather quickly, but I think this is quite basic and you might be able to do it if you put give it some thought. So, what is the utility of the state diagram what you do with the state diagram I learnt it long back.

Student: Given input I know how the I mean I can think with time I can say interval is…

Actually time is missing in this picture so that is a bit of a problem so people will we will change this diagram soon enough, but the idea is basically as the state evolves you know

you can see how the state is going to evolve from time to time that is the idea in the state diagram and there are general system theoretic ways of saying which state diagrams are good which state diagrams are bad etcetera. These things are used in convolutional codes also, but we would not go into all the detail in this course so we will simply look at it and say it is one more thing that we can do with the convolutional codes so that is nice. But, let me ask you some questions where are you to begin with

Student: 0 0.

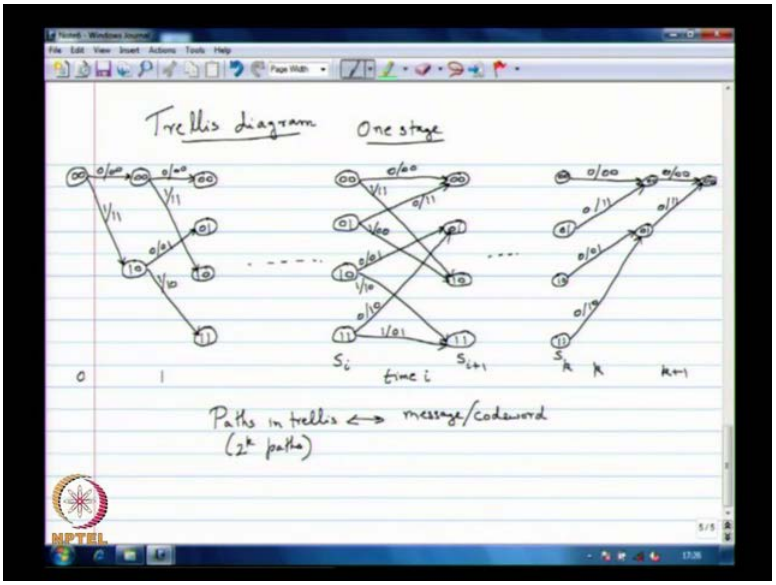
0 0 from 0 0 you can possibly go only to 2 other states 1 0 and 0 1, 1 0 and 0 0 I am sorry only 2 other states that is correct, but not 0 1 0 1, but 1 0 and 0 0. So, the path you take gets defined on the state so any input sequence you have can be mapped into a path on the state diagram right starting at the all 0 you will go to something and you can map that path, but along with the 0 termination what else can I do I can come back to the all 0 state so in fact from any other state you can come back to the all 0 state in 2 steps you do that and along that path both the input and the output are clearly representive so the input you know what it is and the message is there and the code word is also there.

So, paths on the state diagram are important correspond to message slash code word that is one thing. The other idea which is also very important that the state diagram conveys is no matter how long my input bit sequence is my input bit sequence can be 8 or 8000 my state diagram still has only 4 states, because my input bit input bit goes up my state complexity is not going to increase the number of states is not going to go up. That is the reassuring thing when you see the state diagram, so that is something that is exploited in the decoding how no matter how long the decoder is at a code number of code word bits is anyway it comes only from a 4 state the system. So, there is some structure which can be exploited on the decoding.

But, coming back to the point that was about the time axis, the time is kind of explicitly not present in the state diagram, so it is implicit you should know where you are should have a count of how many hops you did from state to state to know time. So, that problem is rectified when you go from state diagram to what is called the trellis diagram. So, you move to a trellis diagram and there you explicitly used time, a time axis and draw different state diagrams for different times that is what you do. So, you do not have the same state for instance is 0 0 you have this 1 state for any time so it is just that you

split it up and draw it for different times and then the picture becomes little bit cleaner and you can think about your decoding and encoding very nicely in the trellis diagram. The trellis diagram is very standard and that is what used most of the time.

(Refer Slide Time: 39:00)



So, let us do the trellis, is there a comment or some question no. So, what you do in the trellis diagram is yes so you going to draw so let me draw an arbitrary state instead of starting from 0 0 etcetera let me draw an arbitrary states somewhere in the middle. So, you can be in anyone of the 4 states 0 0, 0 1, 1 0, 1 1 and the next state it is going to be again one of four possibilities, but instead of drawing them instead of assuming that the same 0 0 states represents the state at the next instant I will draw one more set of states. So, this is the state before some time i so this is time i, before the ith input is clocked in these are the states, after the ith input is clocked in these are states.

So, maybe I can call these state as s i and this will be s i plus 1 so if s i is 1 0 what does it mean I was at state time state 1 0 after i minus 1 bits have been clocked in and before the ith bit is clocked. Now, I can do links here so what will be now I can join these two things that was something fancy I did not do it I did on its own, it is nice. So, if 1 comes in it is going to go all the way down here from 0 1 also same thing will happen 1 0 and 1 1 same thing happens and now I can write my input and output on this 0 0 0, oh my god is this correct, might let me know if I making a mistake. Let me go back to the state diagram and make sure out of 0 1 out of 1 0 with 0 you get 0 1.

So, the same thing is the state diagram so it is exactly the same there is nothing that has changed except that now my branches go from 1 state to some other state so to speak at the next time that is all, this is called the trellis diagram the ith general stage of a state diagram. So, this is called 1 stage of a trellis so at each stage you get 1 input bit and 2 output bits so once again I am talking about this example, but if the example were if the encoder were something else it is easy to see that you can reconstruct a very similar diagram so you have a state diagram and then you convert it definitely into a trellis diagram there is no problem. So, the ideas are quite clear.

Now, what do I do if I have k input bits so I have to start at time 0, but at time 0 there is something slightly special going on because I know already that there is only state 0 0. So, at time 0 I can either get a 0 which means I will continue to be in state 0 0 or I can get a 1 which means I can go to state 1 0. So, what happens at time 1 at time 1 the trellis becomes, after time 1 the trellis becomes full, all the states can be occupied. After this you have the complete stage so this is the time axis you have 0 1 so on you have to go all the way up to k minus 1. So, I am going to put dot, dot, dot here so you can fill it out and then after the sorry let me draw the k minus 1 little bit more better.

This is s k minus 1 so this is after the k minus 1 thing has been clocked in so this is how the trellis will look the entire trellis if you want to do it for the entire input. So, you have k minus 1 inputs at the end of the k minus 1 bit or it was a kth bit which is you will be s k minus 1 am I right or is it s k, so I think I should do s k here so let me do the let me changed this out let it just be s k after the k minus 1 after well, after the kth bit has been clocked in you will be in s k is that, right? When I say kth bit you should remember it is numbered from 0 to k minus 1. So, let me say k minus 1th bit instead of saying so we will start at 0th bit, 0th bit first bit so until k minus 1th bit.

So, after the k minus 1th bit is clocked in after u k minus 1 is clocked in you will be in s k what will happen next I am going to be clock in clocking in only 0 so when if you do 0 then the states you can have is only 0 0 and 0 1 you cannot have anything else. And then finally, 1 more 0 will take you to 0 so this is time k and then k plus 1. So, we have the 0th stage first stage so on till a general ith stage and then a k minus 1th stage which is the last message bit coming in. Then you have the kth stage where you input a 0 k plus 1 stage where you again input a 0 take you take it back to the all 0 stage

So, this is how an entire trellis diagram will look for the convolutional encoder that we had for the simple example. If you have some other example it is easy to see that the similar diagram can be re redone if you have mu bits in the state memory mu then how many states will I have 2 power mu, then how many stages will it take how many 0's do I have to clock in here, mu 0's. So, I will go out to k plus mu minus 1 so you start at 0 and go all the way to k plus mu minus 1 in general.

And it will take mu bits to be clocked in before reaching all the 2 power mu so we will not reach it immediately you will have only 2 states after the 0th stage, you will have 4 after the first 8 so until 2 power mu after the mu bits have been clocked in. So, that will be the general picture there is no point in drawing the general picture because I think the intuition is clear in this. So, you see this is a directed trellis so we have direction this is such because we are going from left to right that is the idea here we are starting at the all 0 state and ending at the all 0 state.

And the analogy that I had before for both parts is also true here every set of inputs that I have will correspond to a path on the trellis, if you have 2 different inputs there will be 2 different paths they will never be the same so that is the idea let me fill it out and then so let me just do this. So, every message sequence you have or every code word sequence you have corresponds to a path in the trellis, so path in a trellis, paths in trellis corresponds to message and code word pair, alright?

So, in this diagram how many different paths are there 2 path k, so has to exactly 2 path k paths so you do not have to worry about counting the paths once again, why should I not have to worry counting the paths once again, every paths corresponds to a unique bit message. So, how many possible messages are there 2 path k and there will be exactly 2 path k paths no problem there so there are 2 path k paths. There are various other ways of counting that also, but anyway you get 2 path k paths there is no problem there.

So, this intuition is crucial in the decoding the fact that the every code word every message corresponds to a path remember what do you, what do you have to do when we do M L decoding, we have to do correlations or minimize the Euclidian distance. So, you have to some kind of a distance for every with of the received word with every possible code word and pick that code word which gave you the least distance that is the idea in M L decoding.

What people will do in this trellis diagram is you do not compute the distance in one shot for all the code words, for all the code word positions start from the left and keep accumulating distance as you go along. You will see as you go along you can eliminate a lot of paths very early you do not have to wait I mean in every state you will see you can eliminate a lot of paths in fact it is enough if you retain only 4 paths in this, you do not have to keep doing that computation for all the 2 path k paths.

Initially, M L is scary because you have to do 2 path k correlations, 2 path k computation of distance it turns out in the trellis it is enough if you do for 4 roughly 4, 4 not exactly 4 you have to do some more calculations, but you have to do only 4. You have to only keep 4 in mind all the time you do not have to worry about otherwise you can keep eliminating as you go along the starting at all 0 and ending at all 0 and that is the key behind the decoder. So, we will pick up from here in the next class.