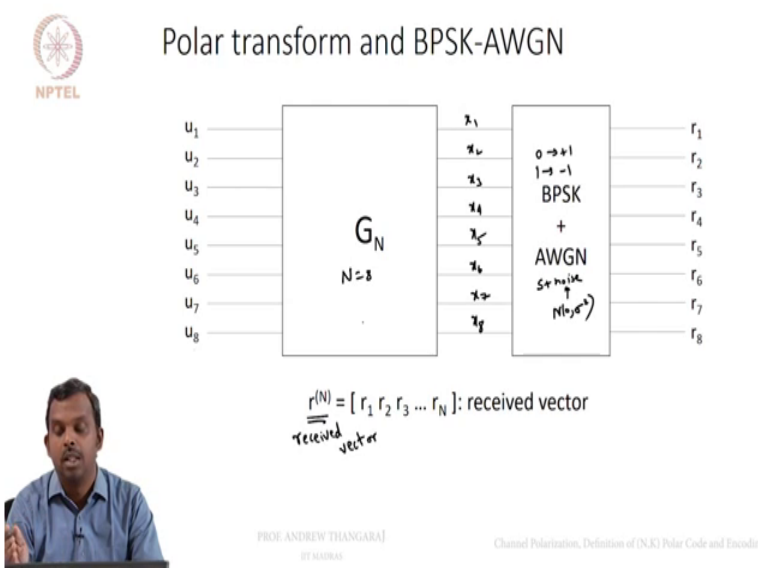
LDPC and Polar Codes in 5G Standard
Professor Andrew Thangaraj
Department of Electrical Engineering
Indian Institute of Technology, Madras
Channel Polarization, Definition of (N,K) Polar Code and Encoding

(Refer Slide Time: 00:17)

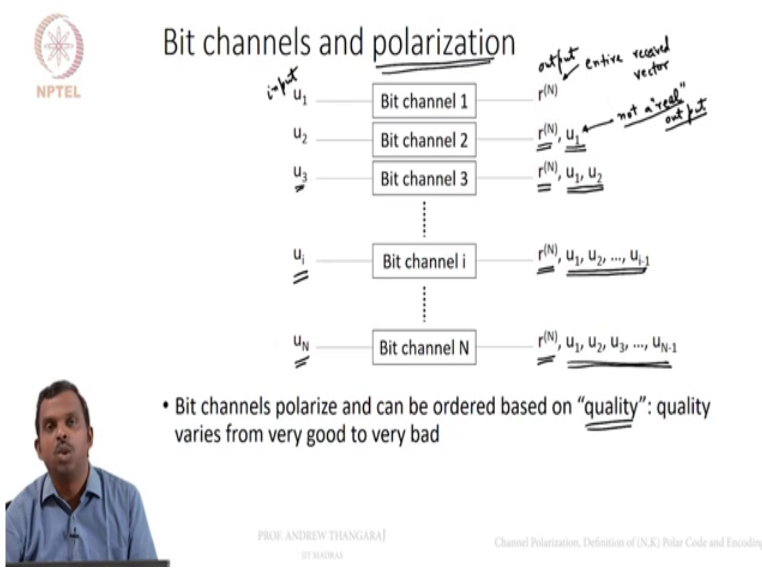


So let's look at what happens if you use the polar transform along with a channel ok. So here is an example I have shown 8 bits here, 8 could be any n I have used n equals 8 for illustration but it could be any other n ok. So you have 8 bits coming in you use the polar transform which is 8 you get 8 outputs ok so this outputs can be anything and usually you can denote it as x1, x2, x3 x4, x5, x6, x7, x8 so this are 8 bits and the you can transmit them through BPSK-AWGN, what is BPSK? Once again 0 goes to plus 1, 1 goes to minus 1 and what is AWGN? It is S plus noise right, so noise is normal with mean zero and variance sigma square, some sigma square.

So you create this noise you get received values ok so this is how a polar transform could potentially be used in the BPSK-AWGN channel, AWGN channel and then this r N is the received vector this is just notation for the received vector ok it is fine we used it, it is just an invertible transform of why would you use it, what is the point of using a polar transform before a channel? So turns out if you don't use the polar transform every bit sort of goes through the same channel in some way ok so there is nothing much maybe you can do.

The polar transform combine this bits and gives us one some sort of reordering or changes the channel in some way you can view it in a different way ok. So this notion of looking at individual bit channels after the polar transform is the key innovation polar codes and it turns out those channels vary la lot in terms of quality ok so that is the next picture. So this picture is easy enough there is nothing much going on you have so many bits coming in you do the polar transform you can do it for any n I have shown it for any n equal 8 you can do it for n equals 1024 also, 1024 bits coming in you hit it with a polar transform you get outputs, do the BPSK-AWGN you get received bit ok. Now what is the effect that this has? How do you view this in a polarization point of view? Here is the picture.

(Refer Slide Time: 02:29)



Ok so this picture is the central idea behind polarization. Once again you will not go deep into the theory of this picture except that they will present that from a high level to understand what is going on ok. So it turns out after polarization you can sort of decompose or view what has happened in terms of what is going on to the individual bits in a sequential way ok so you can convert the n channels that you had into n combined channels that you had in the polarization after when you did polarization and split them up sort of into individual bit channels and the first bit channel is defined as follows.

You have u1 going in, this is the input to the first bit channel and the output is the entire received vector r N ok remember once again the entire received vector. This is the first bit channel split bit channel this is just a definition at this point ok I am defining the first bit channel to have u1 as the input and the entire received vector as the output ok. So I am reordering or viewing the previous channels so in the previous picture we had this kind of picture I am seeing this in a different way ok remember I have got all the bits all the received values at the output ok and remember the first bit would have effected all this bits in some way.

So all the received values will have some information about the first bit ok. So it is not wrong to view it this way ok. So we will take input as u1 and the bit channel I will imagine has a huge output has all the outputs together the entire received vector collected together that is the output ok. Now I will define sort of a second bit channel which has input as u2 which is the second bit but look at the output here pay attention particularly the r N is fine ok so you might say the entire received vector is there but to this I will add u1 as one of the outputs of this bit channels ok.

So now at this point this is not a real output ok, why is that? Because the channel is not giving me u1 ok the channel only gives me r N ok I will imagine that somehow I have this u1 ok and we will see how to create this u1 in the actual physical channel it turns out you can create a good estimate for this u1 in some way and cleverly use it and get this effect of channel polarization but at this point I am defining the second bit channel as input u2 and having output as the entire received vector and the previous bit u1 as one of the outputs ok just a definition.

Next comes the third it channel so here I will go little bit further ok so I have the entire r N as the output u3 is the input again ok entire r N is the output along with that I will say both u1 and u2 I already have the output ok once again this are not real outputs at this point but mean just carry along with me, you will see later on how to get some substitutes for this u1, u2 later on. For now u1, u2 will imagine somebody is giving me at the decoder some benevolent person is giving me what u1 and u2 are and imagining this (second) third bit channel.

So likewise you can proceed, what do I do for the ith bit? For the bit channel I, I will imagine that the entire received vector is there that s not very hard to imagine I have the entire received vector then I will also imagine I have the all the previous bits that were transmitted so u1 to u I minus 1 I already have at the output this will be what I will imagine ok so likewise I will go all the way upto the nth bit channel which has all the received values that is ok and then all the previous bits ok. So now what special about this split channel? What is special about this bit channels is? This bit channels are not similar at all ok, they are very-very-very different from each other over (())(06:26) picture they are all very different in quality.

In the previous case all the channels here all the channels where which you received r1 are the same. Here this n channels will have very different behavior ok. So you can imagine why, right so first of all this channels are very different from the previous channels ok they have one bit input but the outputs are huge ok and also you are assuming the previous bits are available so they can be very different in quality and that is true ok so it turns out you can study this bit channels very carefully and you can sort of order them based on quality ok and the quality you can show where is a lot from very good to very bad.

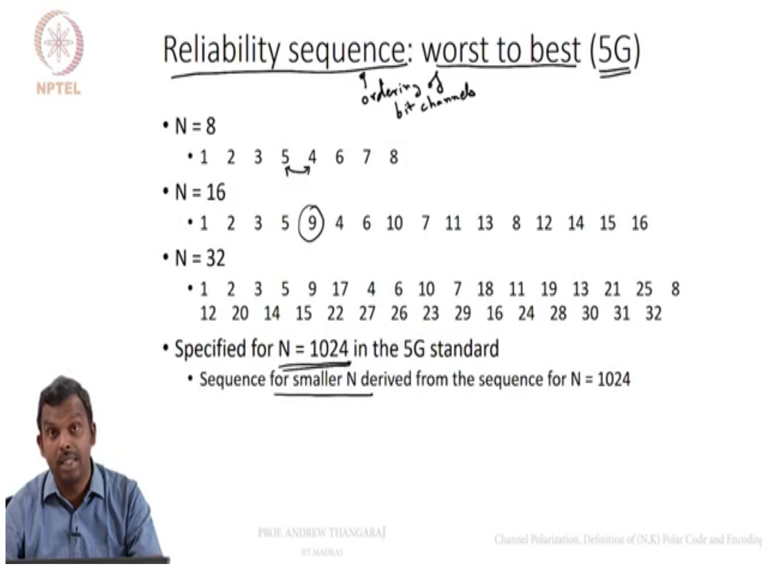
Ok and it turns out as you increase N they either become very-very-very good or very-very-very bad there are literally no channels which are sort of mediocre neither very good nor very bad ok nothing in the middle no middle path all bit channels either become very-very good or very-very bad so this is the essence of polarization. What does polarization mean? Go either go to the north pole or the south pole right polarization (polar) one of two things nothing in the middle ok.

So that way this bit channels get polarized ok they either became very-very-very good or very-very-very bad ok, so this are called this is called polarization.

There is a wonderful information theoretic proof for this polarization for what type of channels it happens and all that and once again we will now study that at least in this course will be later on will add some additional content for this but for now we will just look at how to use this polarization to define polar codes and how to do use this polarization to design decode as for polar codes ok and achieve good performance that will be the focus of the lectures in this course but this idea is very central but keep note of this not having the real output and what do you do for that, we will postponed this discussion when you come to the decoder.

So now we will use this idea of polarization and start talking about how to define polar codes ok you come to this motion of how to recreate this u1 at the output side later on ok.

(Refer Slide Time: 08:37)



Ok so now the next important idea is that of a reliability sequence ok so I mentioned how after polarization if you take the bits you polarize them and transmit them one after the other end over the BPSK-AWGN channel there is a lot of polarization ok. Some bits become some bit channels and then you can define this clever bit channels which have previous inputs.

If you do that then some channels become very-very good some channels become very-very bad. Now which channels are good which channels are bad? Ok that is given by the reliability sequence this tells you the ordering of channels, ordering of bit channels from worst to best ok. Which is the worst among this N bit channels!? And which is the best among this first bit channels? I am giving you the reliability sequence as per the 5G standard ok. Now there are multiple ways to create this reliability sequence and a lots of research going on and how accurately you can find the reliability sequence, how realistic it is etc-etc-etc all of that we are not going to go into detail but we will simply provide the final answer ok.

So it turns out for N equals 8 according to the 5G standard where reliability sequence is this ok. One the bit channel 1 is the worst channel, next comes 2 ok after that is 3, after that is 5, after that is 4 then 6, 7, 8 ok. So it is not the same order as 1, 2, 3, 4, 5 so you see this two got switched for N equals 8 ok. So the 4th bit channel is better and the 5th bit channel according to this reliability sequence ok. So like I said the way in which you do this is not very precise it is sort of noise and simulations and all that.

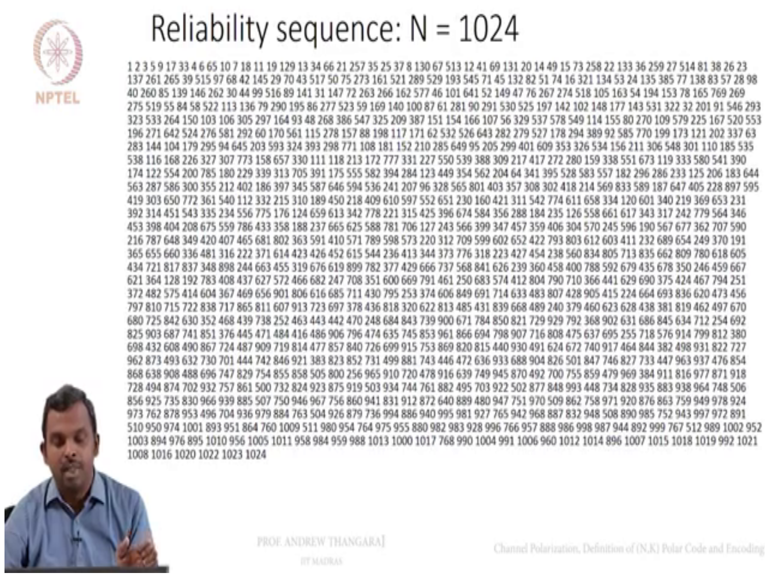
Maybe I will point you to some other references and if you are interested in reading about how to find this reliability sequence? But in this lecture we won't focus on how to find this reliability sequence? We will just think about how to use them for designing for encoding and decoding polar codes ok. So here is the reliability sequence for where n equals 16 so you see this 1, 2, 3, 5 comes again and then suddenly 9 comes ok, 9 becomes bad channel then you have 4, 6, 10 this is the sequence ok. So for instance 14^{th} channel, 14^{th} bit channel is better than the 12^{th} bit channel.

The 8^{th} bit channel is worst than the 12^{th} bit channel like that, that is how you read it, this is the sequence in which the quality of the channel is (())(11:02) 16 bit channels they are ordered like this. This is for N equals 32, remember all this numbers like I said I am pulling from the 5G standard, the 5G standards specifies this for N equals 1 0 2 4 ok and sequence for smaller n is derived from the sequence for m0, m equals 1024 ok.

So I am just picking it up from there ok so you can also do this if you have access to the 5G standard you can go pick up their reliability sequence for 1024 and then from that find the subsequences which are 5n equals 8, 16, 32 and all that ok so that is the final summary. Once you do the polar transform and transmit on the BPSK-AWGN channel for instance or any other channel for that matter you can take a global so the reliability sequence the bit and then create this bit channels. The bit channels become polarized and they can be ordered from the worst to the best and the sequencing a sort of given here.

Ok you can take one sequence, how do I find the sequencing? When I am going into details that but the final answer is given to you ok so this is the sequencing alright, so this is the notion of reliability sequence it is very crucial for designing and defining polar codes very important ok.

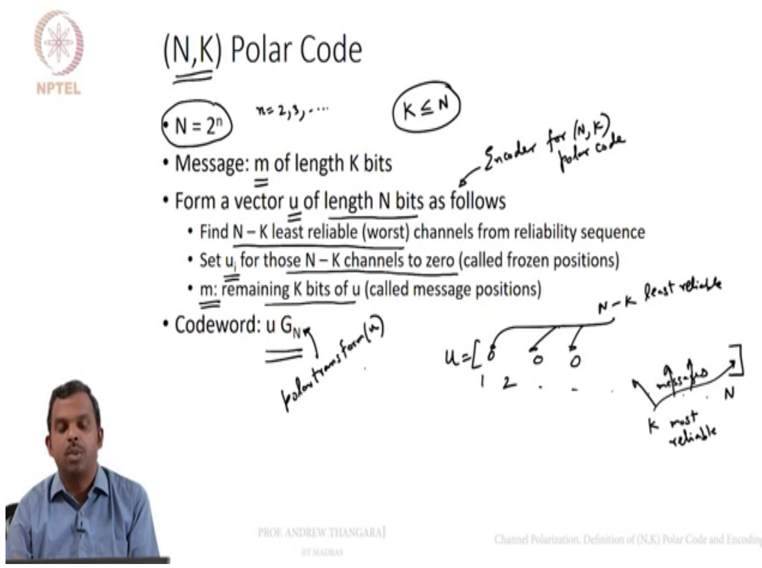
(Refer Slide Time: 12:15)



So here is the full reliability sequence for N equals 1024 I will not expecting you to learn this by heart you don't have to do this you don't need to mug up this sequence but this is the sequence. It is given in the standard and you can see how it goes there is no pattern here right. So it is sort of random in some sense but not all that random sort of increases from 1 to 1024 in some generic way but suddenly you will have some big numbers showing up in early on right so it sort of difficult to predict there is no real pattern here as far as at least I know there is no pattern here it is hard to come up with this estimates and also you may if you read more will see if the noise variance changes in the AWGN channel this reliability sequence actually changes. Ok so it actually changes so it is hard to pick up one reliability sequence for a wide range of noise variances but people use some close simulation studies and carefully study it and come up with something like this.

Ok so the reliability sequence is a difficult thing to find ok so not very easy to find. There are good methods available today in the literature but like I said even if you find it strictly it will vary from noise variance to noise variance and it is not sort of contain so many cases it is not contained some cases it is contain, so you can use some approximations and that is the structure but it is a very hotly research topic to, so reliability sequence is important but for us, we are not going to focus too much on the difficulties around reliability sequence. We will simply take the one that is given in the standard and be happy about it ok so if you are doing that there is no problem.

(Refer Slide Time: 13:56)



Ok so having done all this work I will study the polarization having studied this bit channels having understood what the definition or this reliability sequence is we are ready to define the polar code ok. So N K polar code is defined for N equals 2 power n ok and the small n could be when usually you want to start at 4 (sorry) 2 at least 2-3 so on ok. The message m of length k bits and this k is going to be a less than or equal to n ok. Message m is of length k bits that is ok what do you do to define the polar code I will define the encoder for polar code, ok this is the encoder for the N K polar code.

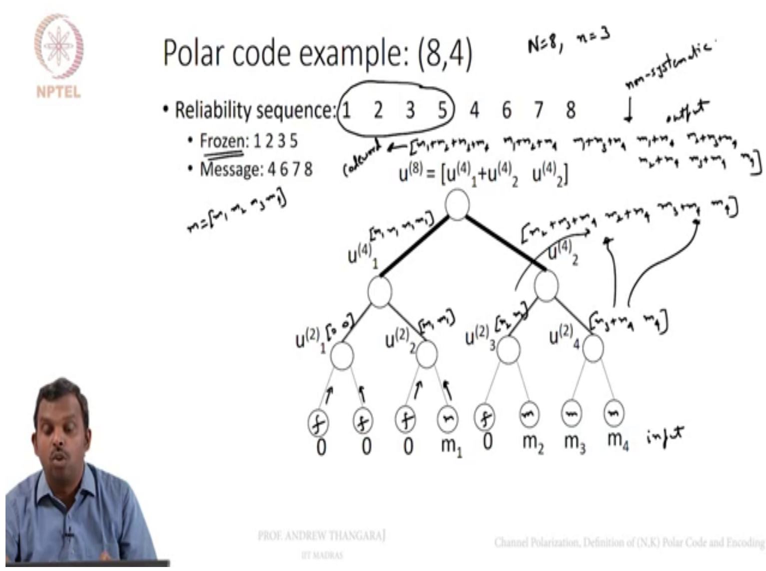
I will form a vector u of length n bits ok, how do I form that? I will first find the n minus k least reliable or worst channels from the reliability sequence ok there is a reliability sequence. What is the first n minus k least reliable, the first n minus k numbers in the sequence is the least reliable right so you take the n minus k least reliable and I will set the u i for this N minus K channel to zero ok so I am saying channels here remember this are like N minus K positions ok. The first N minus K least reliable positions as per the least reliability sequence use set those u i to be equal to 0, remember u i has length n ok the first N minus K positions whatever they are you set them to 0 ok, the u i corresponding to the first N minus K positions in the reliability sequence remember it is not in the same order, the order sometimes changes.

You set those to zero now there will be K remaining positions for u ok, the K remaining positions you set them to be equal to the message ok alright so you have this vector u ok it has n positions

1, 2, so on till n ok the N minus K least reliable positions you put 0 ok so you put 0 here, you put 0 here wherever they are least reliable you put 0 ok and then the remaining guys you put message K most reliable so least and most with respect to the reliability sequence ok, so there you put m messages, messages go there.

Ok so you form a vector u like this and then what is the code word? Code word is simply the polar transform of u ok u times G N polar transform of u is it ok, that is the definition for the N K polar code for nay n you can do this once you have reliability sequence you have a polar code right so as simple as that right. So hopefully this is clear we started with a 2 by 2 polar panel we created G N polar panel then we looked up that polarization property and then we define the reliability sequence. Once you define the reliability sequence you have a polar code ok, so that is it.

(Refer Slide Time: 17:26)



Here are a few examples I am starting with N equals 8 small n equals 3 ok and defining the 8, 4 polar code this is the reliability sequence so the frozen positions the once for which you choose them to be zero this are called the frozen positions ok so once again I think I missed defining that the least reliable N minus K least reliable channels which you set to zero are called the frozen positions and then the remaining are message positions ok. So if you do 8, 4 code reliability sequence is 1 2 3 5 4 6 7 8 so the frozen positions are 1 2 3 5 ok, the message is 4 6 7 8 ok.

So how do you visualize the polar code construction? You can go back to the binary tree representation of the polar transform ok and 1 2 3 and 5 you set as 0, so this is all frozen ok I will write f here to indicate that this are frozen and then write m here to indicate this are message ok. So the first, second, third and fifth leaf nodes are frozen to 0, the remaining are set to m1, m2, m3, m4. Now will this encoding happen? You will do 0 here, 0 here and you will get a 0, 0 here right 0 0 goes to 0 0 you get 0 here m1 here and you will get m1 (m1) here right.

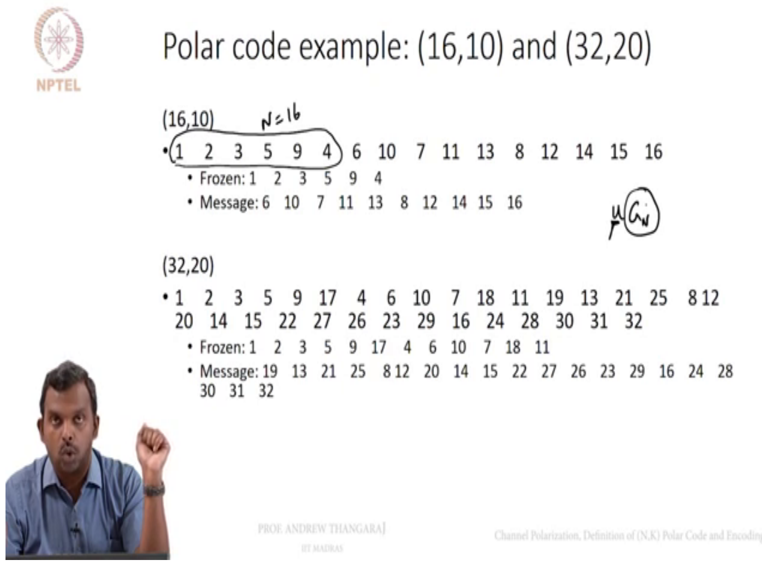
So 0 is coming here m1 is coming here so 0 plus m1 is m1 then m1 itself is retain here likewise you have 0, m2 so this will be m2 m2 ok this is m3, m4 so will have m3 plus m4, (m4) ok then what will, happen in the next step? This will this is 0 0 m1 m1 so this 4 bit vector will be m1, m1, m1, m1 this will be m2 plus m3 plus m4, m2 plus m4, m3 plus m4 and then m4 right so that is what happens at this node you combine m2 m2 and m3 plus m4 m4 you take the ex-or of this two keep it here and then this is retained as it is ok so remember this is retain as it is. The ex-or of this two goes here.

Ok that is what happen in this and then finally the code word 8 bit code word is going to be again ex-or of this two and then this retained as it is right so let me write it down laboriously for you m1 plus m2 plus m3 plus m4, m1 plus m2 plus m4, m1 plus m3 plus m4, m1 plus m4 and the remaining 4 are the exact same things as this m2 plus m3 plus m4, m2 plus m4, m3 plus m4, m4 this is the encoded version so this is the input to the transform and this is the output of the transform and that is the code word so this is the code word ok.

So you have a message m which is m1 m2 m 3 m4, m is m1, m2, m3, m4 and the code word is this ok remember plus again as ex-or or modulo 2 addition you do all this things ok.

So if you want you can create or generate a matrix out of this you will get when this easily specifies the generator matrix ok. You can construct that if you like ok you will get a 4 by 8 generator matrix and you notice this is not a systematic encoding, isn't it? m1, m2, m3, m4 does not appear by itself in the code word it is not systematic but this is the encoding that you have ok. So once you fix length and the number of message bits and once you have reliability sequence you can always do polar encoding and this tree representation gives a very simple way in which you can implement the encoding aspect.

(Refer Slide Time: 21:40)

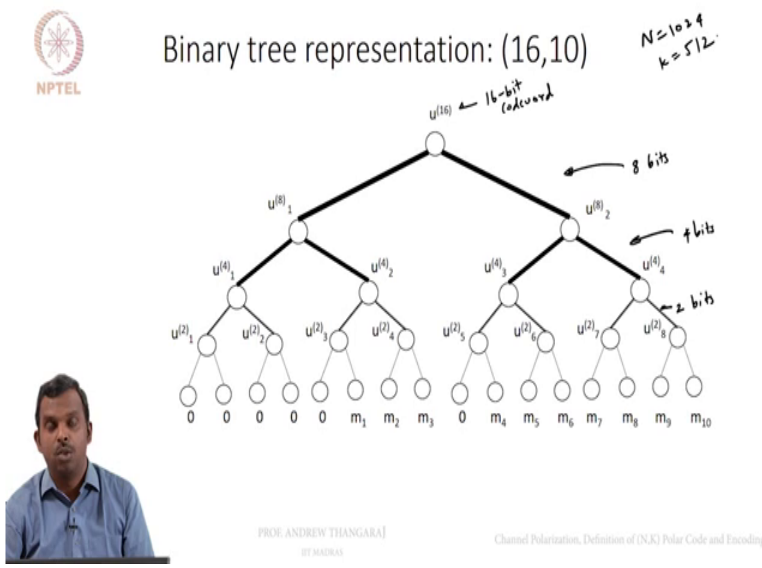


I am going to give a few more examples here is a 16, 10 example ok so for n equal 16 the reliability sequence is this and you have to freeze the first 6 positions right, so first 1, 2, 3, 5, 9 and 4 will get frozen in your tree you have 16 leaf nodes the 1st nodes, 2nd node, 3rd node, 5th, 4th and then 9th are going to be set as zero the message will go into the remaining place and then will start encoding as before you will get the code word at the output ok a code word each bit of the code word will be some ex-or of some m1, m2, m3 like that.

If you want you can write out and generate a matrix for that and you will have a definition for 16, 10 polar code ok. the same thing happens for 32 ok there is nothing different about it n equals 32 so and 20 is the K so you have to freeze the first 12 positions you go all the way up to 11, so what are the frozen positions? 1, 2, 3, 4, 5, 6, 7 then you have 9, 10, 11, 17 and 18 ok some sort of numbers that they get frozen to zero remaining at the message bits once again you can combine and create a generate a matrix if you like ok. So in fact viewing the freezing as (()) (22:54) times G N is also quite easy ok.

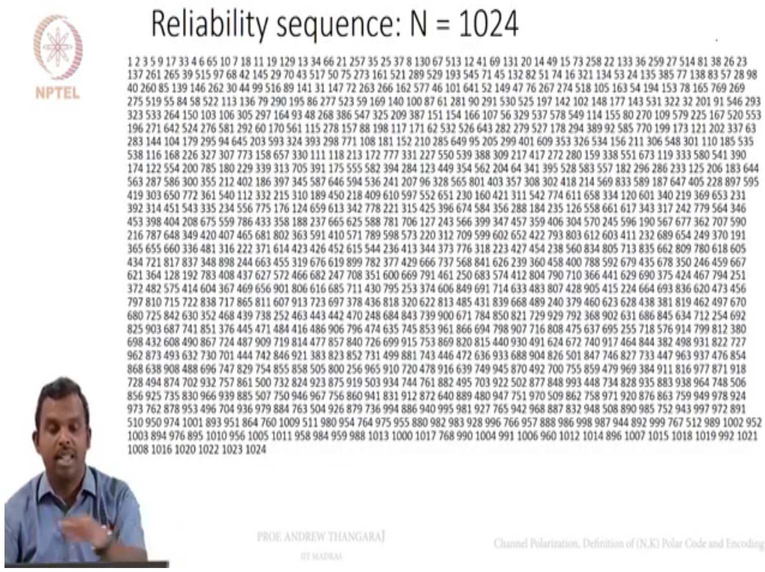
So this is the polar transform you look at the non-frozen positions in U and the rows corresponding to G N in that, that will give you the generator matrix ok so that is a quick way to generate come up with a generator matrix for the polar codes ok. So this is example.

(Refer Slide Time: 23:15)



And here is for 16,10 the full blown binary tree representation I am not going go through and show you the computation but this is what happen here frozen bits zero and then the message bits 10 of them and they combine and you create the 16 bit code word. Remember once again this are all 1 bit here, this are all will be 2 bits right and then you have 4 bits, then you have 8 bits here ok and then finally you have 16 bit code word ok. So that's it that is the polar code and even if you have so you can imagine if you have N equals 1024 and K equals 512 what do you do?

(Refer Slide Time: 24:07)



Ok you go back to the big slide that I showed you ok this is N equals 1024 and K equals 512, what will you do? You take this big vector and then mark out the first 512 positions and make them frozen and imagine this tree and on the leaf you freeze all of them set them to zero put the message bits in the remaining places and then do the encoding ok I mean it is hard to do it by hand but you can write a quick program and that will work quite fast ok.

So that is the end of description of the polar codes hopefully this was clear to you once again the important things to focus on at the polar transform is notion of bit channels, how we define them and the reliability sequence is defined that is very-very crucial and once you have a reliability sequence you have a polar code ok once you have it for any N it is the polar code ok so N, K polar code your frozen positions and message positions you freeze the first N minus K and you put your messages in the remaining K ok, that is the end of this lecture on polar codes. In the next lecture we will see MATLAB programming for writing a quick encoder once again we will write a simple encoder that works for doing the polar encoder ok, thank you very much.