# LDPC & Polar codes in 5G Standard

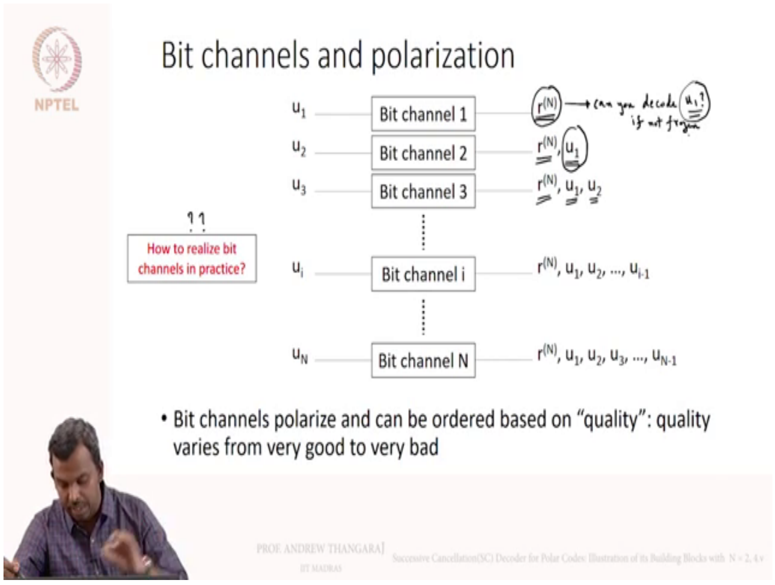
**Prof. Andrew Thangaraj**
**Department of Electrical Engineering**
**Indian Institute of Technology Madras**

## Successive Cancellation (SC) Decoder for Polar Codes Illustration of its Building Blocks with N equals 2, 4

Hello and welcome to this lecture on decoding of polar codes, like mention polar codes are decoded using a sequential decoder, it is also called as successive cancellation decoding, the actual version used in practices, maybe a minor modification of what was originally proposed by the Larrikin but inspiratives very true to that, the sequence of operations and how it works a logic behind it is very similar, once again when I describe the decoder, I will not go too much in to the theoretical aspects of things, the information theoretics, studies is not what we will do for instance, you will hear the system called Bhattacharya parameter which is used quite extensively in the design and study of polar decoders we will not mention that it all, we will focus on implementation as far as these lectures are concern okay.

So without further I do let us get started okay, so once again what is the setting of this polar code, I am repeating this slide from before you have block length which is a power of two messages, length is some K which is less than N and then you form this vector U which has reliability sequence okay and the most reliable position is where you put your message, all the lesser reliable sequence first N minus K lesser reliable sequence you freeze them to 0 and then you do the multiplication U times GN, where GN is its kernel which is 1011 tens a product N times okay, so this is the basic constructions of the polar codes, we saw this before okay.

(Refer Slide Time: 1:42)



## Bit channels and polarization

$u_1$ — Bit channel 1 — $r^{(N)}$ → can you decode $u_1$? if not frozen
$u_2$ — Bit channel 2 — $r^{(N)}, u_1$
$u_3$ — Bit channel 3 — $r^{(N)}, u_1, u_2$
How to realize bit channels in practice?
$u_i$ — Bit channel i — $r^{(N)}, u_1, u_2, \ldots, u_{i-1}$
$u_N$ — Bit channel N — $r^{(N)}, u_1, u_2, u_3, \ldots, u_{N-1}$
NPTEL
PROF. ANDREW THANGARAJ
IIT MADRAS
Successive Cancellation(SC) Decoder for Polar Codes: Illustration of its Building Blocks with N = 2, 4 v

- Bit channels polarize and can be ordered based on "quality": quality varies from very good to very bad

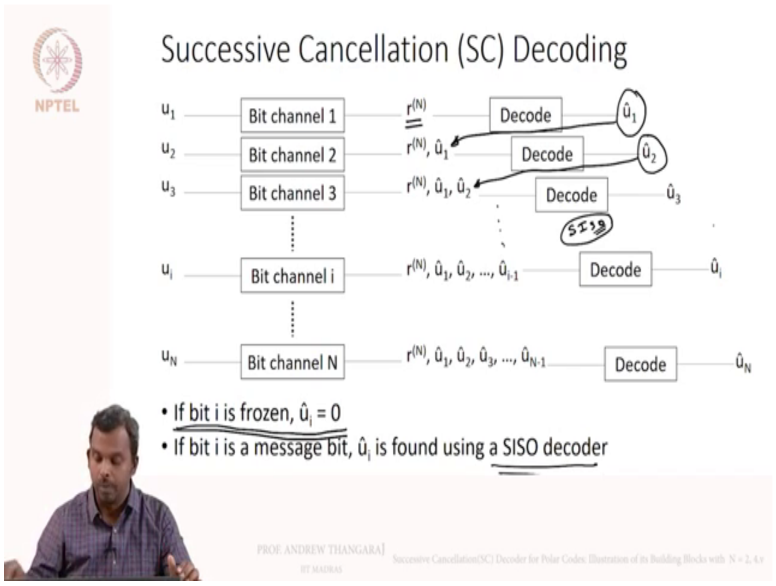
Now this slide is also something we have seen before okay, except for this red part here which says how to realise bit channels in practice? So basic idea in polarization is when you combine these bits using that kernel GN and then you transmit, you create and then you can split what you have into a set of N bit channels okay, each bit going in, each U1 going in and the output is now different okay, the first output for U1 is the entire vector RN, the first output is entire vector RN.

For the second bit channel it is interesting, you have RN and then you have U1 okay, which is the previous bit okay, so and for the third bit channel you have RN and the two previous bits okay, now how real is it to have the previous bits as included in the output of a channel because you do not really know what these are right, these for transmitted at the transmitter N, you do not know what they were, so is it realistic! That is an important question for us to answer, we will answer that in this lecture okay, but this is the construction but once you do this it turns out the channels becomes polarised okay, so the quality starts varying, there are very good channels, very bad channels and you can use them smartly in your decoder or construction of the code, so that is the crux of the idea behind a polar codes.

Now the crucial question is what I have here okay, how to realise this bit channels in practice? How you do get U1 in practice? Okay, so it is not too difficult to summarise, it is not too difficult to think about how you can come up with U1, you cannot come up with the exact U1, that is obvious but you can come up with an estimate for U1 okay, given RN and given this first bit channel, can you estimate or decode U1 okay.

How a possible you decode? Okay and remember in some cases U1 may be frozen okay, if U1 is a frozen bit then you do not need to decode okay, if not frozen can you decode, if it is already frozen then you know it is 0 okay, so all this positions wherever there are frozen you already know them, this UIs okay, you do not have to do any special decoding but if they are not frozen then you need to decode okay, so that is the idea that is done to realise this bit channels.

(Refer Slide Time: 3:58)


NPTEL
$u_1$
Bit channel 1
$r^{(N)}$
Decode
$\hat{u}_1$
$u_2$
Bit channel 2
$r^{(N)}, \hat{u}_1$
Decode
$\hat{u}_2$
$u_3$
Bit channel 3
$r^{(N)}, \hat{u}_1, \hat{u}_2$
Decode
$\hat{u}_3$
SISO
$u_i$
Bit channel i
$r^{(N)}, \hat{u}_1, \hat{u}_2, \ldots, \hat{u}_{i-1}$
Decode
$\hat{u}_i$
$u_N$
Bit channel N
$r^{(N)}, \hat{u}_1, \hat{u}_2, \hat{u}_3, \ldots, \hat{u}_{N-1}$
Decode
$\hat{u}_N$
PROF. ANDREW THANGARAJ
IIT MADRAS
Successive Cancellation(SC) Decoder for Polar Codes: Illustration of its Building Blocks with N = 2, 4, 8

## Successive Cancellation (SC) Decoding

- If bit i is frozen, $\hat{u}_i = 0$
- If bit i is a message bit, $\hat{u}_i$ is found using a SISO decoder

Let me illustrate that in clearly using a picture like this okay, so this picture you have to read sequentially from the top okay, so you receive RN and you do a decode okay and you get U1 hat, remember what is decode, you have to remember this is very important, in decoding if the bit I is frozen you do not have to do any special decoding okay, if it is frozen you now know the exact UI okay, so you said UI hat to be 0.

If it is not frozen you have to do some decoder operations okay, what decoder operation you do in fact you use a SISO decoder okay, even though you do not need the soft output, the decoder uses a what the internal decoder that is used here for finding out the estimate is actually a SISO decoder okay.

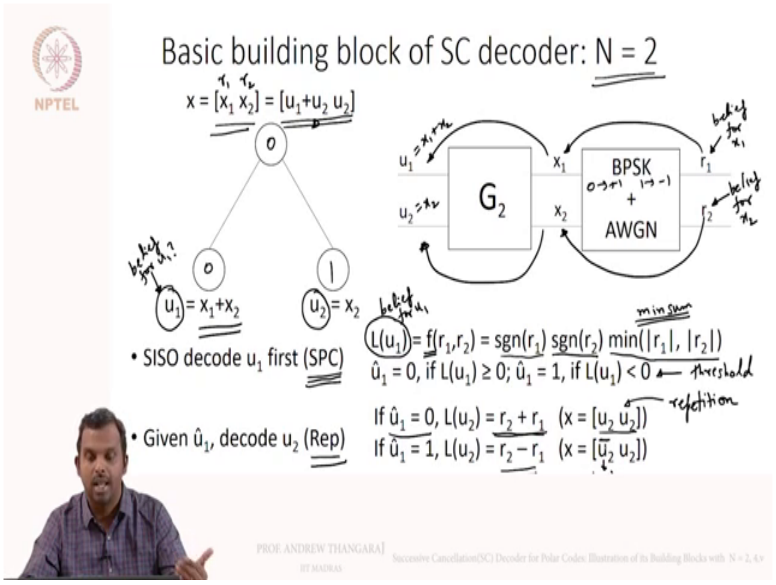
So this part is important because the soft output is used later on okay, later on when you talk about less decoding, you will see the soft output here is used but right now the successive cancellation decoder will ignore the soft output, only take the hard output and use it in the next step okay.

So I have not told you what this SISO decoder is? I will describe that in the next few slides but this is essentially the idea to try and recreate the bit channels at the receiver okay, how do you recreate the bit channels again? You simply decode okay, so you decode U1 and then use here okay and once you have U1 hat, you use RN and you want to together and then you decode U2 and so on, you proceed like this.

Now the frozen bits are very important okay, when the bit channel is really really not reliable, your decoder will also fail, so that is where the freezing idea comes in, so when you freeze you know the UI hat for sure, you are not propagating the error too much okay, by these, doing this freezings in frequent places, when as you go from the top to the bottom, once in **a** while the bits is frozen right,

So when you freeze you are reinforcing that bit, so the thing does not propagate too much okay, so still there will be propagation a little bit but not too much because of this freezing, so the freezing idea is also very crucial, it helps you avoid some propagations, error propagations and the sequential decoder but nevertheless this is the crux of the idea, so how do you recreate a bit channels at the receiver, you do sequential decoding, you decode one at a time and you go in the sequence 1 to N but wherever you have frozen positions you use 0 okay, that is the essential and crucial idea okay, so it is quite simple idea at some level but this picture is important to keep in mind okay.

(Refer Slide Time: 6:29)



## Basic building block of SC decoder: $\underline{N = 2}$

$x = [x_1\ x_2] = [u_1 + u_2\ u_2]$

$u_1 = x_1 + x_2$ $\qquad$ $u_2 = x_2$

- SISO decode $u_1$ first (SPC)

$L(u_1) = f(r_1, r_2) = \text{sgn}(r_1)\ \text{sgn}(r_2) \min(|r_1|, |r_2|)$

$\hat{u}_1 = 0$, if $L(u_1) \geq 0$; $\hat{u}_1 = 1$, if $L(u_1) < 0$

- Given $\hat{u}_1$, decode $u_2$ (Rep)

If $\hat{u}_1 = 0$, $L(u_2) = r_2 + r_1$ $\quad (x = [u_2\ u_2])$

If $\hat{u}_1 = 1$, $L(u_2) = r_2 - r_1$ $\quad (x = [\bar{u}_2\ u_2])$

Now to describe this SISO decoder, we knew need to use the binary tree representation and we will start with N  equals  2 okay, N equals 2 is the simplest possible case, maybe it is not a practical decoder at all, you will never use code with N equals 2 but it is an important building block in the SISO decoder okay, so I am going to describe what happens in N equals 2, remember N equals 2 is just a depth one binary tree, you have a root and then just two nodes 0 and 1 and U1 is the bit that is input here, U2 is the bit that is input here and then what you put out is U1 plus U2 and U2 okay.

So if you call this is X1, X2, U1, U2 is the input, you run it through kernel G2 you get X1, X2 and X1, X2 is transmitted with BPSK, AWGN, remember BPSK, AWGN is 0 to plus 1, 1 to minus 1 and AWGN is noise added you get R1, R2 okay, the R1 corresponds to X1, R2 corresponds to X2 okay, remember when X1 was transmitted R1 was obtain and X2 was transmitted R2 was obtain, the whole thing about the decoder is how do you recover U1 and U2 right, how do you get a soft information for U1 and how do you get a soft information for U2 and how do you do the hard decision that it, that is the only point and for that, it turns out our old ideas of single parity check and repetition are what is used again okay.

So this only soft ingredients that we used these days is that, is the single parity check code and the repetition code okay, so I tell you how that comes about okay, so if you look at X equals X1 plus X2, you can write U1 in terms of X1, X2 okay, so this is we write X in terms of U1, U2 and the kernel going in this way, you can actually invert the kernel and go back in this direction okay, how do you do this U1 is X1 plus X2 and U2 is X2 right, it is just a simple, so you xor these two guys, you get U1 okay, that is the idea okay.

So it is easy to do and you get U1 as X1 plus X2 okay, so remember this you get U1 as X1 plus X2 and then R1 was received when X1 was transmitted and R2 was received when X2 was transmitted, so R1 and R2 is like a belief for X1, R2 is belief for X2 okay and what do you want, you want belief for U1 okay and what is U1, X1 plus X2 okay, you have a belief for X1, you have a belief for X2, what is the belief for X1 plus X2? Okay this is the classic single parity check problem right.

So this is the extrinsic information for single parity check and for which we can use the minsum method okay, how do I use the minsum method? I will take the sign of R1, multiply with the sign of R2 and then take the minimum of the two absolute values okay, you can also use the Tan hyperbolic and log Tan hyperbolic and get more accurate soft information if you like based on the xor but let us use minsum okay.

So we have seen a minsum is close enough and this minsum function I will called as F okay, so node this notation this minsum function I will call as F, F of R1, R2 is sign of R2, R1 sign of R2 multiplied by sign of R2, multiplied by minimum of the two absolute values okay, so this is something important and note it down okay and you can see that this is just single parity check code right there okay.

So this is your belief for a U1, I will call it L of U1 and it is obtain using the minsum okay, once you have a belief for U1, finding U1 hat is just a simple hard decision estimate, this is just a threshold okay, if the belief is positive you see U1 hat is 0, if the belief is negative you say U1 hat is 1 okay, hopefully this is clear to you so the SISO decoder for N  equals  2 is quite simple, the straightforward, it is a simple single parity check transformation okay.

Remember once again we are not doing U1 and U2 together okay, so this SISO decoder for the polar code with N  equals  2 is slightly different from, maybe doing the overall decoding together okay, you decode U1 first and once you find U1, you use the estimate of U1 to decode U2 okay, you are not doing it jointly, you are not finding U1 and U2 jointly okay, you could find U1 and U2 jointly that will be another decoder but the polar decoder does not work like that, it is works in a successive cancellation always okay.

So this is more like a successive cancellation SISO decoder, it is not really the sort of optimal SISO decoder from other areas okay, this is a successive cancellation SISO decode okay, so once you know U1 hat notice what happens okay, if U1 hat is 0 okay, your X is U2, U2, this is actually repetition okay, alright think about it, so if you are U1 hat is 0 then your X simply became repetition okay right, so what is the meaning of reputations? So what you sent was actually U2, U2,

So R1 is also a belief for U2, R2 is also a belief for U2 okay because U1 hat it identified as 0, you assume once you find U1 hat you assume that U1 is 0 okay, that is the nature of successive cancellation decoding, you figure out U1 and then use it, assume it is true and find the next one okay, once U1 you know is 0 it is actually a repetition code, X is just U2, U2, so R1 is belief for U1, R1 is belief for U2, R2 is another belief for the same U2.

Now you have repetition code you simply add the beliefs okay, now if U1 is 1 actually the situation is very similar, what is X, it is U2 bar which is the complement of U2 right, 1 plus U2 and U2 okay, so R1 is a belief for U2 bar, the complement of U2, so if you have a belief for the complement, what is the belief for the actual bit U2 minus R1 okay, so if you think about it, see BPSK 0 going to plus1, 1 going to minus1, if for a particular bit you have a belief it is complement and the belief is negative right minus of that okay.

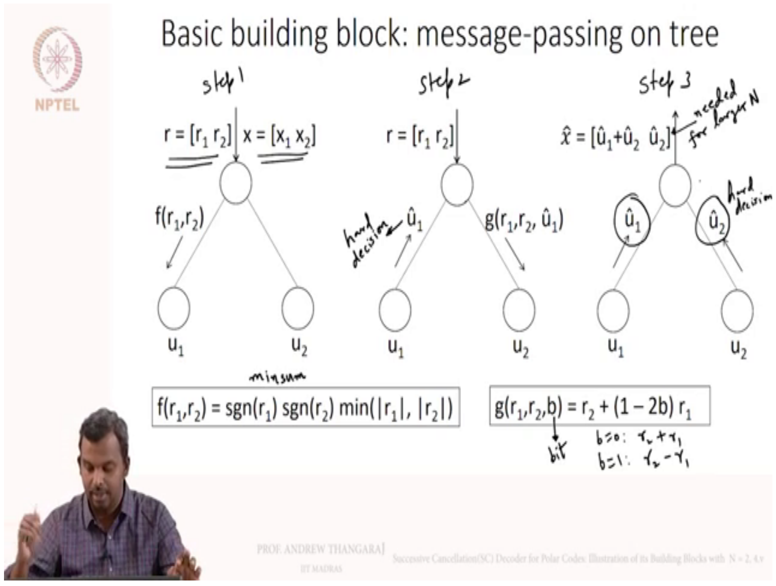
So when U1 hat is 1? What you transmitted is U2 complement and U2, R1 is a belief of U2 complement, so minus R1 is another belief for U2 okay, similarly R2 is a belief for U2 okay, so what is a total beliefs? R2 minus R1 okay, which is actually again an addition except that because of this complement you have to do R2 minus R1 okay, so once again I want to emphasise was going on here, let me repeat from the top what is going on?

The basic building block for the polar decoder is a successive cancellation SISO decoder for this simple N equals 2 situation and for that the essential components are once again the single parity check and the repetition okay, the single parity check is sort of obvious, only the extrinsic matters and it comes to, from just U1 being X1 plus X2 okay, so you use the minsum to find its belief and for the repetition there is this little twist because of the successive cancellation.

So if the previous bit was 0 it is a pure repetition, so you just add R2 plus R1 because both received values are beliefs for the same bit U2, on the other hand if U1 hat is 1 then there is a complement involved, so you have to multiply one of the beliefs where minus1 and add okay, so you do that, you get the belief for the next one okay, so this is a crux of the polar decoder, the simple block that you have and this is how you decode the tree okay.

So I am going to in the next slide interpretive this steps as message passing decoding on the tree okay, so once we see it as message passing, you can start extending it to larger block length and see how the message passing can be extended to larger blockings okay, so let me show this as message passing on a tree.

(Refer Slide Time: 14:53)



Okay, so this is what happens in for the basic building block in message passing for a tree okay, U1 and U2 are the message bits, this is the code word bit and this is the belief or channel received value, once again I keep saying belief, people use a log likelihood ratios what we are going to be slightly more generic there, usually we use the received values directly as beliefs okay, now obviously that happens in BPSK, AWGN, if the modulation is different you have to compute the belief per bit using some method and then use this algorithm okay,

So what is the first step? This is step 1, this is the step 2, this is step 3, what is the first step? You have R1, R2 the belief for X1 and X2, you send F of R1, R2 from the root node to the leaf node on the left okay, so this is very important, this direction whether it is on the left or right is very important, left comes first okay, you always go left and below first okay, so that is the rule remember okay.

And this is the minsum function F of R1, R2 okay, so you compute that and send it out okay, in the second step the leaf node figures about what U1 hat is and sends it back to the root node okay, so that is just a hard decision okay, so this is hard decision on F of R2, so notice the belief we had for U1 at this point we are not using, later on when we study the less decoder we will use that belief okay, so right now the belief for U1 is not used here, it is soft output, it is available, it is just merely done, merely do a hard decision decoding okay.

So now once you have U1 hat okay, what goes the next step, in the step after you have U1 hat from the root node, you will send a message to the right child okay, so in the first step you send a message to the left child and then you got something back from the child okay, so this is very important, you send a message to the left child okay and then you get, you wait for something to come back, the decision to come back from the child okay.

Once you get that decision you send a message to the right child okay, the child on the right okay, so once you have that you send a message to the child on the right and what is that message? That is where the repetition code comes in, I am going to use this function G okay, so this is a bit okay, 0 or 1, this function G is succinctly captures the complement situation okay, if B is 0 you have R2 plus R1 okay, B is 0 you have R2 plus R1 and if B is 1 you have R2 minus R1 okay, so this is just a shorthand notation to capture this, the complement the situation that you have here.

So what you send here is G of R1, R2, U1 hat, notice the little switch here I am calling it R1, R2 but they do R2 plus 1 minus 2B R1 okay, something to pay attention to okay, so once you have this, this is the second step, you sent a message to your right child okay, these G function was involved in that and then you do a hard decision okay, so once again at this point we ignore the belief for U2, we simply do the hard decision, we do not ignore it and we do the hard decision through it out but later on in the less decoder will make explicit use of the belief for U2 as well okay.

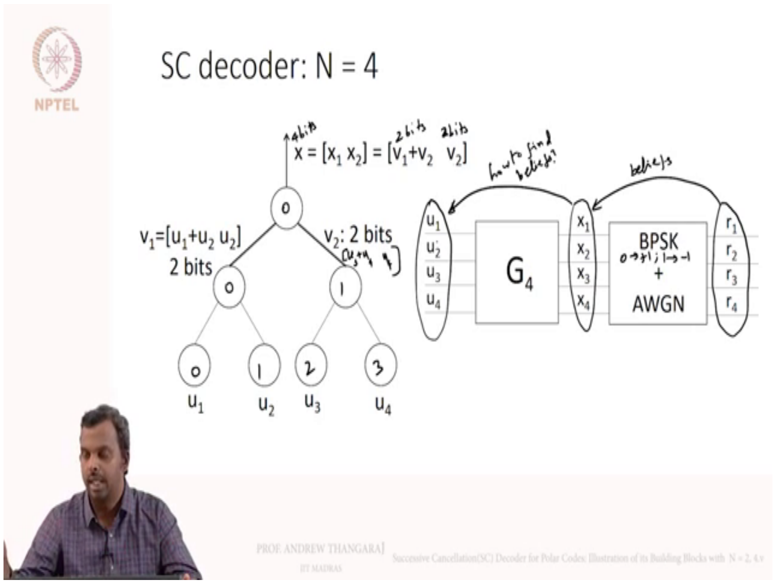
So now once the root node has heard decisions from both the children okay, so here is a decision from the first child, here is the decision from the second child both the leaf node, so both the children have given a decision back, the root node can now decide okay, what will it decide? U1 hat plus U2 hat, U2 hat this is the code word which goes up from the estimated code word, which goes up from the root node okay.

I want to point out a couple of things here, my messages is in U1 hat and U2 hat, I really do not need the code word right, what do I do with the code word? Messages is U1 hat and U2 hat, once I find U1 hat and U2 hat I am done okay, I really, maybe do not need this, X hat is this okay, so this as a turns out is needed for a larger N okay, so this kind of flows is needed, you will see the next slide I will start describing N equals 4, find N equals 4 sending a message from this node is also important okay.

So it sends up will also play a role and that we will see when we look at N equals 4 okay, so this this part is strictly if you just decoding N equals 2, you do not need to go back and construct this X hat, you can just happy with, be happy with U1 hat and U2 hat, you are done with that but you need this X hat for when you uses as a building block, when you use this as a building block for larger decoding, larger block length you need this X hat also okay.

So this is the N equals 2 building block, hopefully it was clear the next step is, how to use this as a building block in N equals 4 okay, we want to go to that in the next slide okay.

(Refer Slide Time: 20:00)



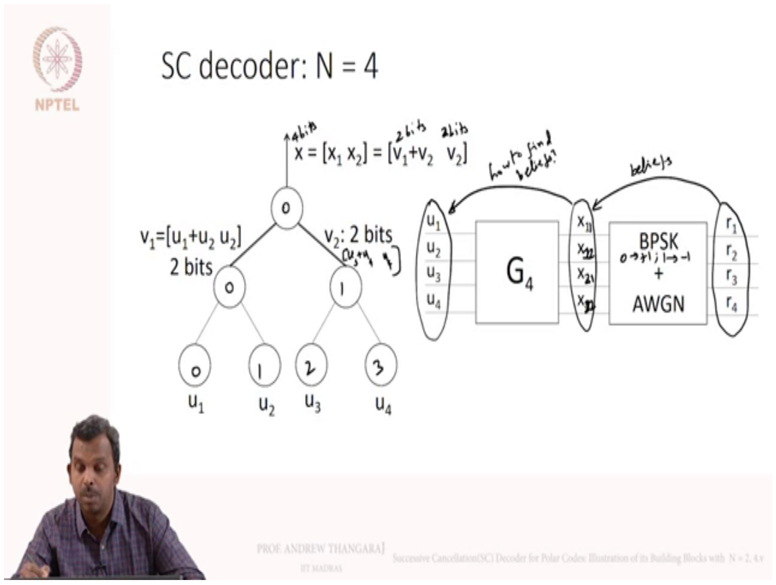
So now how is the situation find N equals 4, if you remember the binary tree is slightly larger, you have depth two in the binary tree okay, so the root node is here and then you have depth one node and then you have depth two nodes right, so that is the nodes in the binary tree and you have U1, U2, U3, U4, you find then intermediate vector V1, I am going to call it V1, this is U1 plus U2, U2 and V2 is what? V2 is again U3 plus U4, U4 right, this is V2, so notice this is two bits okay.

Now this V1 and V2 are remember two bits long, these are vectors of two bits and these are combined again to find a code word X, X1, X2 is V1 plus V2, V2 okay, you can write back in terms of U but V1 plus V2, V2 is pretty nice, it is just two bits each though, so this is two bits, two bits, so it becomes four bits okay, overall you have four bits, so this is the idea we also saw the program for writing this encoder, so this is how the whole thing works okay.

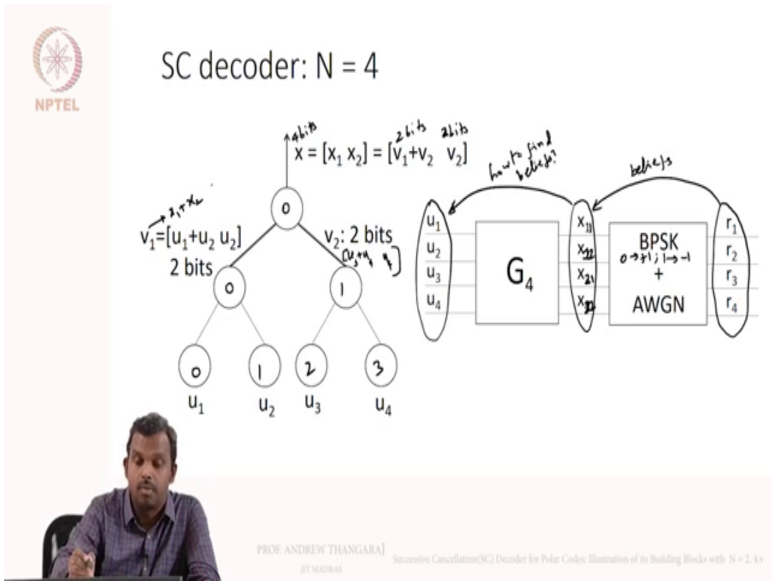
This is the code, U1, U2, U3, U4 is what you originally have and then you go to X1, X2, X3, X4 after this operation with G4 and then BPSK AWGN as before 0 going to plus1, one going to minus1 the noise getting added you get R1, R2, R3, R4 remember again R1, R2, R3, R4 are beliefs for X1, X2, X3, X4 how you find beliefs for U1, U2, U3, U4 is the central question that we answer in the success of cancellation decoder, the polar decoder that we have, we do it in a successive fashion, you first find the belief for U1 then use it, use the decision on U1 want to find the belief, for U2 and then use the decision on U1, U2 to find the belief on U3 and then you find the decision use on U1, U2, U3 to find the belief for U4 okay.

So that is will proceed as will go along the tree, you will see will go along the tree in a certain way, these are much like before and I will describe that the next slide okay, so the tree plays a very crucial role.

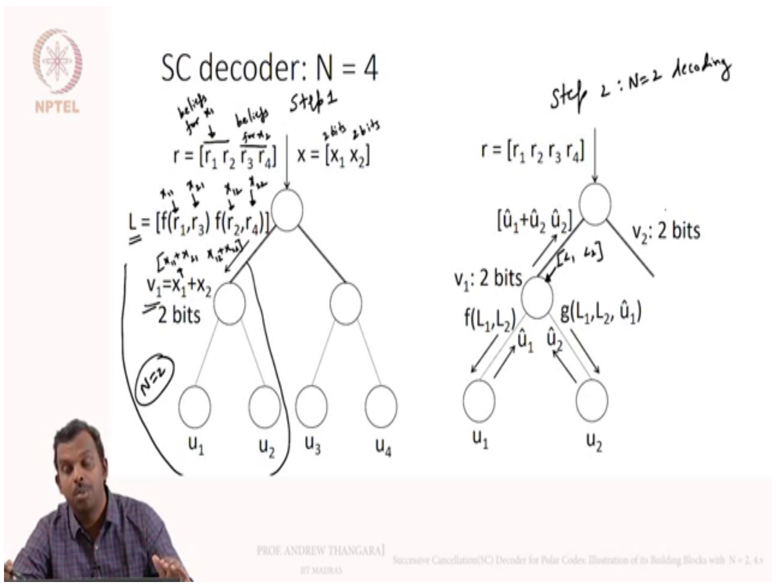
(Refer Slide Time: 22:15)



## SC decoder: N = 4



## SC decoder: N = 4



Okay, so this is the first step I am showing here, this is step one okay, so this is step two, remember again we got R equals R1, R2, X1, X2, remember X1 is two bits, X2 is also two bits okay, so that is so I have written up, X1 is two bits, X2 is two bits, so this R1, R2 is belief for, so let me be careful here, so that some ambiguity in the notation, notice this is, I have also written X1, X2, X3, X4 here remember this is not X1, X2, so I should write this as a X11, X12 and X21, X22 okay, so this is better notation, go back and correct it.

So I have X11, X12, X21, X22, this is the first bit of X1, second of bit X1, first bit of X2, second bit of X2 okay, so X1, X2 are two bits each, these are beliefs for X1 okay, the R1 is the belief for the first bit of X1, R2 is a belief for the second bit of X1, likewise this guy is belief for X2 okay remember that, okay let me write this step, 1 here of this side and then I can say this is beliefs for X2 okay.

So the vector together R1 and R2 is a belief for X1, the entire X1, X11 and X12 okay, so now remember once you have X what is V1? V1 is X1 plus X2 okay; you can go back to the previous thing and see X is X1, X2, V1 plus V2, V2 and V1 is actually X1 plus X2 okay, so that something important to know okay, so V1 is X1 plus X2 okay, so you have beliefs for X1, you have beliefs for X2, what is the belief for X1 plus X2? You use minsum again okay, how do you use minsum? You use minsum on R1, R3 okay.

So because why do use minsum on R1, R3, this is belief for X11, this is belief for X21, this is belief for X12, this is belief for X22 okay and V11, the first bit of V1 is actually X11 xor X21 right is not it, this is what? This is X11 plus X21 and then X12 plus X22 okay, so X11 plus X22, X21 all you find the belief that, you do the minsum F of R1, R3 okay, R1, R3, then X12 plus X22 how do you do the belief for that F of R2, R4 okay, so that is it for, so simple single parity check code soft output decoding and that is the beliefs you send down below okay.

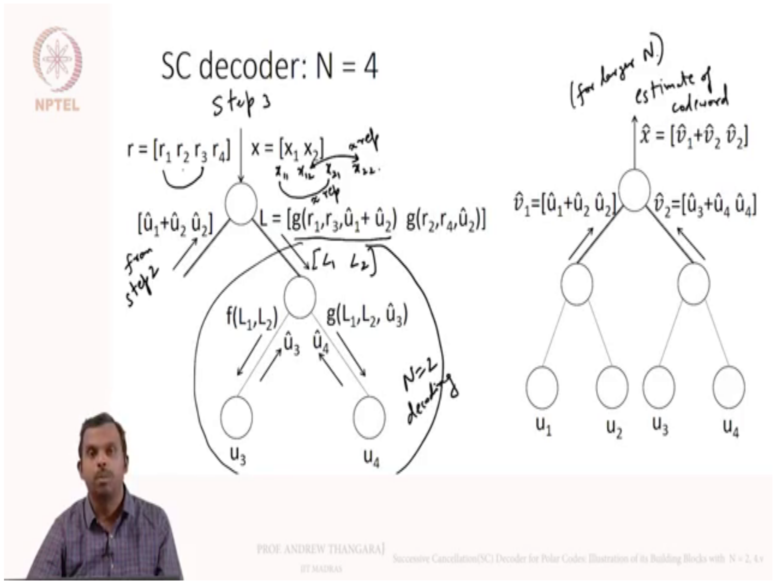
So you have two bits here for V1 and I got beliefs for both of them okay alright, so once again you had four bits of the code word, you go out four received values, four beliefs you group them two at a time and then computed the two beliefs for V1 okay and now once you have beliefs V1, this is just a vector of length two, you are back to the N equals 2 situation right, at this point you are at the N equals 2 situation okay, what happened at N equals 2? You had root node and you had to beliefs coming in, same thing is happening here, you have a root node and you have two beliefs coming in and you have two leaves below that okay.

In step two is just N equals 2 decoding okay, so that is the basic building block okay, what you do? You have two beliefs coming in, remember you have the two beliefs L equals L1, L2, maybe I should write that down somewhere here, L1, L2 are coming in here okay, so that is the L that you had here F of R1, R3, F of R2, R4, L1, L2 is coming here and what you do in the N equals 2 decoding, we know what to do? You do the minsum function F of L1, L2 goes down, U1 hat comes back in, G of L1, L2 U1 hat goes down to the right side and then U2 hat comes back in and what goes back up? U1 hat plus U2 hat and then U2 hat okay.

So it is mentioning how this code word going up is important okay, so this goes up to the root node okay, so now we have to figure out what the root node is going to do okay, so the root node send some belief down to its left child okay and now it is getting decisions back on the left child okay, so remember this operation once again, the root node got some beliefs from the top okay, it did some minor processing with this F and sent beliefs down to its left child okay and it has now got back decision from its left child okay.

Now the root node will have to send something to its right child okay, which is? What is going to happen in the next step, how do we do that at? Just like we use the function F, will use function G now but when you use the function G remember there are two bits now, you have to send the vector of two, so what do you do that is the next step.

(Refer Slide Time: 27:27)


SC decoder: N = 4
step 3
$r = [r_1\, r_2\, r_3\, r_4]$
$x = [x_1\, x_2]$
$[\hat{u}_1+\hat{u}_2\; \hat{u}_2]$
from step 2
$L = [g(r_1, r_3, \hat{u}_1+\hat{u}_2)\;\; g(r_2, r_4, \hat{u}_2)]$
$[L_1\; L_2]$
$f(L_1, L_2)$
$g(L_1, L_2, \hat{u}_3)$
$\hat{u}_3$ $\hat{u}_4$
N=2 decoding
$u_3$ $u_4$
(for larger N)
estimate of codeword
$\hat{x} = [\hat{v}_1+\hat{v}_2\; \hat{v}_2]$
$\hat{v}_1 = [\hat{u}_1+\hat{u}_2\; \hat{u}_2]$
$\hat{v}_2 = [\hat{u}_3+\hat{u}_4\; \hat{u}_4]$
$u_1$ $u_2$ $u_3$ $u_4$
PROF. ANDREW THANGARAJ
IIT MADRAS

Okay, so this is step three okay, so you got this decisions from the previous step right, so from previous step, from step two we got this two, the decisions from the left child are coming back up okay, now what are the beliefs you compute here, you use G okay just like before except that you are going to use G on two bits right, so you have X1, X2 here, you look at what happens here, so once again X11, X12, X21, X22 and then you have R1, R2, R3, R4, you look at R1 and R3 and X11 and X21 okay.

So if you look at just those two bits, they make a repetition code okay just like before and everything depends on the decisions, U1 hat plus U2 hat, if U1 hat plus U2 hat is 0, it is a pure repetition code, you can simply add the beliefs R1 plus R3 but if U1 hat plus U2 hat is 1, you have do the complement so the belief will become R3 minus R1, so that is why you do G of R1, R3, U1 hat plus U2 hat, that is a repetition okay, so it is like repetition.

So likewise these two guys are also like repetition okay, so you do G of R2, R4 U2 hat and U2 hat is your decisions to light that repetition okay, so you use once again this function G except you know group your bits as repeating blocks two at a time and then use the same function G and the decision that you got from left child to guide what to compute as beliefs for the right child okay, so this is L equals L1, L2 believes going down to the right child and this part is now N equals 2 decoding and that is what I am doing here okay, F of L1, L2 goes down below to the left child, U3 hat as the decisions that comes back up, G of L1, L2, U3 hat goes to the right child, U4 hat is a decision that comes up okay, that is L2 decoding and once U3, U3 hat, U4 hat have comeback back up to the this node, it sends back V2 hat which is U3 hat plus U4 hat, U4 hat and then V1 hat was anyway there U1 hat plus U2 hat, U2 hat and then you can find the X hat which is V1 hat plus V2 hat, V2 hat.

Okay so now once again, once you find the U1 hat, U2 hat, U3 had, U4 hat you are done as far as N  equals  4 decoding is concerned okay, you do not need anything more okay but we are going to proceed and look at this calculation of the code word also, this is the estimate of the code word and you do not really need estimate of the code word for decoding N  equals  4 but why will I need estimate of the code word here, when I want to go for larger end okay, so this is useful for larger end once again okay.

So this is the essential working of the polar successive cancellation decoder soft input, soft output actually but we do hard decisions as we saw, it is sequential, it decodes one bit and uses that decision to go through and decode another and it uses very basic simple building blocks just two bits single parity check code xor of two bits together okay, just minsum with two inputs and two bits repetition okay, just repeating twice and if you going for larger block length, a group the bits and identify the two bit  repetition and you are using it  okay, nothing more happens on the polar decoder, it is very simple steps and the sequence in which they are combined together is the beauty in it and that is where the performance comes in okay.

So I am going to stop here for this lecture to explain the philosophy and the way in which the decoder works and in the next lecture I will start generalising, so you see what the node does? The node receives beliefs from the top send some beliefs to its right, left child gets back decisions, send some beliefs to the right child gets back decisions combines the decision and send its back up okay.

So this is what every node does on the tree okay, so once we describe what every node does, we can describe the entire decoding algorithm for any N okay, so far I showed you only N equals 4, we will generalise from here and talk about how the decoder works for a general and that I will do in the next lecture, once we are done with that we can write a simple Matlab code to stimulate it and we were done with the single successive cancellation decoder for polar codes. Thank you very much.