

LDPC & Polar codes in 5G Standard
Prof. Andrew Thangaraj
Department of Electrical Engineering
Indian Institute of Technology Madras
Soft-input Soft-output Iterative Message Passing Decoder for LDPC codes

(Refer Slide Time: 0:17)



LDPC Codes: Decoding





Andrew Thangaraj
andrew@iitm.ac.in

Soft-Input Soft-Output Iterative Message Passing Decoder for LDPC Codes

Hello and welcome to this lecture on decoding LDPC codes, so far we have seen the definition of the parity check matrix for LDPC codes in the 5G standard, we saw how to encode them, we even wrote up small Matlab program for encoding this codes, so all that is ready now we are ready to jump into decoding okay, so how do we decode, the decoder is the heart of the success of the LDPC codes like I have mentioned before, it is a message passing iterative decoder, it works in multiple rounds of iterations and it actually consist of a reasonably simple operations and the memory structure is what important, so you have to store a lot of things in memory and move it around, access it etc. okay, so you will see as we describe the decoder, what is involved, the principles are interesting and simple.

Now once again we will not to get too much of the theory, we will focus on the implementation aspects. Some of the things I do will be repetitive in the sense that you will be reminded a lot of the single parity check decoding because the single parity check is an important ingredient in the decode for a LDPC codes okay, so let us get started.

(Refer Slide Time: 1:23)





Low-Density Parity-Check Codes

- Linear codes can be specified using either a generator matrix or a parity-check matrix
- LDPC codes are linear codes with a sparse parity-check matrix
 - Sparse: most of the entries in the matrix are 0s
- Proposed in 60s
- Since 1990's, intense research in the area
- Adopted in many standards
 - Wimax, Wifi, DVB etc
 - Now, 5G too

PROF. ANDREW THANGARAJ
IIT MADRAS Soft-Input Soft-Output Iterative Message Passing Decoder for LDPC Codes

As a quick overview of where we are with respect to LDPC codes, LDPC codes are linear codes, specified by sparse parity check matrix, most of the entries are zero or very few ones, it was proposed long back in the 60s, since the 90s this been a revival this codes and now they are part of many standards, earlier standards 2 and now the 5G standard also has them for the data communications part okay.

(Refer Slide Time: 1:49)



Decoding

- Soft-input soft-output
- Iterative
- Message passing
 - Approximation: minsum
- Extensive analysis in literature


PROF. ANDREW THANGARAJ
IIT MADRAS Soft-Input Soft-Output Iterative Message Passing Decoder for LDPC Codes

So what are the main points to note, the decode of LDPC codes is soft-input, soft-output okay, so you remember the soft input, soft output decoders is spoke to you about you deal with these believes or log likely would ratios for the received values and for the bits in the code words and then you improve then using some calculation, it is iterative in the sense that

it does not try to decode the entire code at once okay, so it will use some partial information from the code and then do some decoding and then pass messages around okay.

So that is a sort of the idea in this, there is an approximation involved which is called the minsum approximation, we have spoken about it, this is a very very very well studied decoder today okay, so many papers are there in the literature, we will not look an exhaustive study of all possible variations, just the one type of decoding which is quite popular in practice today okay, so we proceed.

(Refer Slide Time: 2:47)



Soft-decision Decoding

- Process real values received from the channel
- Channel: BPSK over AWGN(σ)
 - $\mathbf{c} = [c_1 c_2 \dots c_n]$; $\mathbf{r} = [r_1 r_2 \dots r_n]$ $c_i \in \{0,1\}$
 - $r_i = s_i + \text{noise} \rightarrow \text{real value}$
 - c_i : codeword bit $\{0,1\}$; $s_i = 1 - 2c_i$ BPSK $0 \rightarrow +1$
 $1 \rightarrow -1$
 - LLR, $l_i = \log [\text{Prob}\{c_i=0|r_i\}/\text{Prob}\{c_i=1|r_i\}]$ c_i : uniform

$$= \log [p(r_i|c_i=0)/p(r_i|c_i=1)]$$
 - For BPSK over AWGN, $l_i = 2r_i / \sigma^2$ $\frac{2}{\sigma^2} r_i$
- We seek
 - $L_i = \log [\text{Prob}\{c_i=0\} / \text{Prob}\{c_i=1\}]$ ← approximately iteratively

entire received vector

PROF. ANDREW THANGASUBRAMANIAN
BY MADRAS

Soft-Input Soft-Output Iterative Message Passing Decoder for LDPC Codes


Okay, so the setting is sort of similar to how we will be doing all along, you have BPSK over AWGN the noise standard deviation is sigma, the code word is C1 to CN okay, so you have the code word here which is C1 to CN, so each CI, okay so this is a code word and each CI is a bit okay, 0 over 1 okay this is a bit, this is the received value remember this is CI first it converted into SI which is BPSK modulation, okay you can see BPSK modulation here, 0 goes to +1, one goes to -1 okay, it gets converted into symbol and then this is the received value okay, real value okay, so now a lot of you might know if you have some implementation background, the real values are not really used in it is entirety, in the decoder you always quantise it to say 5 bits or 6 bits or 7 bits, I will mention that later on but for now we will keep it as the real value R as far as the description of the decoders is concerned.

Eventually this will all be integers values after suitable scaling and quantisation okay, so that will happen later on okay, so this is the received value, we talked about this LLR which is the channel LLR okay, this log of this ratio of probability of CI given the particular bit okay,

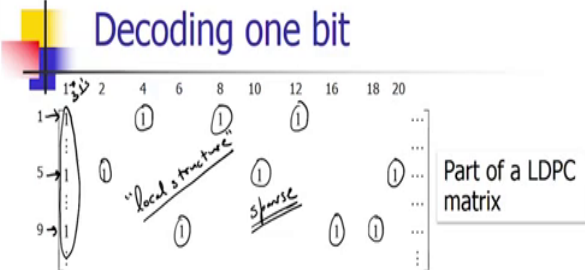
particular received value r_i alone and that becomes this assuming you know, C_i is uniform okay, then you have this and for BPSK over AWGN you have this very simple formula for the LLR right, so we saw this before 2 times r_i by σ^2 is the channel log likelihood ratio for the particular received value okay, given r_i alone the belief on the bit is 2 r_i by σ^2 , so it is proportional to 2 by σ^2 and you can see this is important to know okay, proportional to r_i , I am sorry, L_i is proportional to r_i and the constant is 2 by σ^2 , so this part is either need estimation or usually can be ignored, you can just simply said it as 1 in your decoder okay, so we will do that later on.

So what we seek is the output LLR, where here you have the entire received vector okay, given the entire received vector what can I say about the probability of a particular bit being 0 or 1 okay, now it turns out if you want to exactly write down an expression for this is possible, but the complexity is really huge, your exponential complexity in KN and N and all that and particularly if K is thousand, and N is two thousand you have no hope of doing it okay, so what will happen in this decoder is, we will approximately iteratively compute this okay, not compute exactly of course, approximate computation but we will iteratively go towards it, we will slowly improve the log likelihood ratio, using more and more information from the remaining code words like, so this L_i , small L_i uses only information from r_i , we will try to use other bits of information slowly in a iterative manner and expand the believe that you have, make the better belief that you have on a bit better and better okay, so that is the sort of philosophy, will be little use on describing that going forward okay.


(Refer Slide Time: 6:38)



Decoding one bit



- Use the received value corresponding to Bit 1
 - 1st estimate, $l_{10} = \text{LLR} | r_1 = 2r_1 / \sigma^2$ channel
- Use parity checks that involve the bit
 - Row 1: $c_1 + c_4 + c_8 + c_{12} = 0$; 2nd estimate, $l_{11} = \text{LLR} | r_4, r_8, r_{12}$
 - Row 5: $c_1 + c_2 + c_{10} + c_{20} = 0$; 3rd estimate, $l_{15} = \text{LLR} | r_2, r_{10}, r_{20}$
 - Row 9: $c_1 + c_6 + c_{16} + c_{18} = 0$; 4th estimate, $l_{19} = \text{LLR} | r_6, r_{16}, r_{18}$
- Same can be done for all bits



DR. ANANDH THANGARAJ
IIT MADRAS

Soft-Input Soft-Output Iterative Message Passing Decoder for LDPC Codes

So let me describe what happens little bit more clearly, so let us focus on the low-density parity check matrix, so this is the matrix okay, this is only a part of the matrix, the top left if you will okay, so the first column has three ones okay, so the first column has three ones, everything else is zero and the ones are in the first row, fifth row and the ninth row okay, every other row has zero in the first column okay, there is one only in the first row, fifth row and ninth row okay and if you focus on the first row, the first has once in the fourth column, eighth column, twelfth column, of course the first column is also one, fourth, eighth and twelfth column, likewise the fifth row has once in the second column and tenth column and twentieth column okay and the ninth row has once in the sixth column, sixteen column and eighteenth column okay, remember this is all sparse matrix, so you expect a lot of zeros okay, there will be a lot of zeros, very few ones okay, so these is the one, this is the sort of the local structure okay.

So this notion of a local structures very important in this, approximate iterative message passing decoders, if you just focus on the first bit alone and look at the parity check matrix, this is how everything looks okay, you do not see anything beyond this, of course the first bit is also affected by other bits, these are not the only bits which affect the first bit, right because through these bits, it is further connected to other bits, right the twelfth column may have more ones okay and then that made have more connections okay all of that is there, but for now we can ignore all of that and just focused on the immediate structure in the first bit okay.

Now if we are given only R_1 okay, we know what the belief of the first bit is, this is like the first estimate okay, so this is only from channel, okay you are only given R_1 , what can you say about the belief of the first bit, $2 R_1$ by sigma square, so that we know, there is no problem okay, so what we do now is, we try to exploit the local structure okay, the connection in row one, row five and row nine okay, now what does row one tell you okay, we saw this before, every row of the parity check matrix actually defines a single parity check code, it gives you a parity check constraint that is satisfied by one subset of the bits of the code word, so row one is telling you that C_1 , C_4 , C_8 and C_{12} belong to the single parity check code okay, so that is something that know okay.

So using just the SPC decoder okay, so the second estimate is formed by the single parity check decoder, you can compute another estimate for the first bit okay, the log likelihood ratio for that bit given R_4 , R_8 and R_{12} okay, so how will I do this, I can do the minsum

approximation or even the exact computation with the log tan hyperbolic function etc, you can compute the extrinsic estimate from this okay, so this should again be extrinsic okay, so this is easy enough to do, I am using only the single parity check code defined by the first row.

The same way row five also defines a single parity check code involving C1, C2, C10 and C20, so now I can use the same single parity check code again but with R2 and R10 and R20 okay, now I will get another estimate okay, that is again extrinsic, again uses single parity check decoder, the same structure again but with different received values okay, now you get another estimate, third estimate okay, now likewise you can also form a fourth estimate okay, which comes from row line which is C1, C6, C16 and C18.


So the local structure has given you already for estimates okay, the first estimate came from the channel itself, second estimate came from the first parity check it was not, the bit was involved in, third estimate from the second parity check, fourth estimate from the third parity check, so every parity check in the local structure gives you more estimates for that particular bit and how were those estimates calculated, using the single parity check decoder the soft-in, soft-out decoder okay and you compute the extrinsic information okay.

So the same thing can be done for the every other bit as well is that okay, so I showed you for the first bit, now even if you go to the two hundredth bit, the two hundredth bit will have a local structure, it will have certain ones in the columns which are connected to certain ones on the rows, you can go and do the same computation okay, except that you should know which bits the two hundredth bit is connected to, which are the rows, where are the ones, once you know where the ones are, you can go pull out these received values, send them through the same SPC decoders, so you just need this single parity check decoder which will keep doing seesaw calculations for you okay, so that is the trick to this okay, so there is the essential ingredients in an LDPC codes, so you use the local structure and get more estimates for a particular bit okay.

Now this is not the whole story okay, so only the local structure, only conveys partial information right, so you are not still connected to the rest of the code words, rest of the bits in the code word or the received values, only the first twenty or so received values are being used here okay and in fact not even twenty, 3, 6, 9, 9 are the received values are being used here, these codes might be two thousand bits long okay.


So then you to use the other received values as well but you want to do it efficiently, for that you use message passing okay, so each code will use the local structure and then pass messages to each other and improve each other in some iterative nice fashion okay, so that is the essential idea but this mission of how the code works through the local structure row wise is very important in implementing, now also understanding how the decoder works okay, so we will see some more concrete examples going forward but this is the setting okay.

(Refer Slide Time: 12:50)




Minimum approximation

- $|z| = \text{abs}(f [f(l_1) + f(l_2) + \dots + f(l_w)])$
 - $f(x) = \log \tanh (|x|/2)$
- $|z| \approx \min\{\text{abs}(l_1), \text{abs}(l_2), \dots, \text{abs}(l_w)\}$
- Saves computation of nonlinear function
- Used in a slightly modified format
- Almost no loss in the approximation in practice




PROF. ANDREW THANGARAJ
IIT MADRAS

Soft-Input Soft-Output Iterative Message Passing Decoder for LDPC Codes



x	y	$z = x \oplus y$	$1-2x$	$1-2y$	$1-2z = (1-2x)(1-2y)$
0	0	0	1	1	1
0	1	1	1	-1	-1
1	0	1	-1	1	-1
1	1	0	-1	-1	1



PROF. ANDREW THANGARAJ
IIT MADRAS

Soft-Input Soft-Output Iterative Message Passing Decoder for LDPC Codes

NPTEL

x	y	$z = x \oplus y$	$1-2x$	$1-2y$	$1-2z = (1-2x)(1-2y)$
0	0	0	1	1	1
0	1	1	1	-1	-1
1	0	1	-1	1	-1
1	1	0	-1	-1	1

$x \in \{0, 1\}$
 $y \in \{0, 1\}$

PROF. ANDREW THANGARAJ
Soft-Input Soft-Output Iterative Message Passing Decoder for LDPC Codes

NPTEL

x	y	$z = x \oplus y$	$1-2x$	$1-2y$	$1-2z = (1-2x)(1-2y)$
0	0	0	1	1	1
0	1	1	1	-1	-1
1	0	1	-1	1	-1
1	1	0	-1	-1	1

$x \in \{0, 1\}$
 $y \in \{0, 1\}$
 $E[z] = 0 \cdot P(0) + 1 \cdot P(1) = P(1)$

PROF. ANDREW THANGARAJ
Soft-Input Soft-Output Iterative Message Passing Decoder for LDPC Codes

NPTEL

Efficient Calculation of Conditional LLR

- Suppose $z = x + y$; x, y, z : binary RVs (indep)
 - $(1 - 2z) = (1 - 2x)(1 - 2y)$ ← same as XOR
 - $(1 - 2p_z(1)) = (1 - 2p_x(1))(1 - 2p_y(1))$ Take expectations
 - $\tanh(l_z/2) = \tanh(l_x/2) \tanh(l_y/2)$ $l_z = \log \frac{P_z(0)}{P_z(1)}$
 - $\text{sgn}(l_z) = \text{sgn}(l_x) \text{sgn}(l_y)$ l_x, l_y
 - $\text{abs}(\tanh(|l_z|/2)) = \text{abs}(\tanh(|l_x|/2)) \text{abs}(\tanh(|l_y|/2))$
- Suppose $z = x_1 + x_2 + x_3 + \dots + x_w$
 - $\text{sgn}(l_z) = \text{sgn}(l_1) \text{sgn}(l_2) \dots \text{sgn}(l_w)$
 - $|l_z| = \text{abs}(f[f(l_1) + f(l_2) + \dots + f(l_w)])$
 - $f(x) = \log \tanh(|x|/2)$

SPC SIS is easy to compute

PROF. ANDREW THANGARAJ
Soft-Input Soft-Output Iterative Message Passing Decoder for LDPC Codes

Okay, so how do you calculate this, we have seen before, I am going to go through this really really fast without spending too much time on it, we saw that if we have two binary random variables X and Y , this is the modular 2 addition okay and Z is $X+Y$ then this is true okay, $1-2Z$ is $1-2X$ into $1-2Y$, this is the same as Xor okay, so if you do not believe me, you can make little table here X $1-2X$, Y $1-2Y$, if you look at it, maybe I should write it bit more with some more space, so if you make a little table here or maybe I can go to the yes.

So if you make a little table here XY Z equals X xor Y is 01, 00 I am sorry, 0001, 1011, you have 0110, if you look at $1-2X$, $1-2Y$ and $1-2Z$ do like the BPSK modulation or a speak, you have 11-1-1, 1-11-1 and you if you look at $1-2Z$ you get 1-1-11 and which is actually to $1-2X$ times $1-2Y$ okay you can multiply this two and you get this okay, so this is sort of after BPSK the xor becomes multiplication okay, in terms of +1-1, it is a multiplication.

Okay, so that is what is being used here $1-2Z$ is $1-2X$ into $1-2Y$, now you can take expectations okay, you can take expectations these binary random variables will assume are independent or at least uncorrelated for this proposed, so if you take expectation, expected value of binary random variable being equal to 1 is the same as probability that is equal to 1 okay, so that is another little quick, little result that one can derive.

So remember if X is 0 or 1, this is P of 0, P of 1, expected value of X is 0 times P of 0 + 1 times P of 1 and that just P of 1, so for binary random variable expected value is equal to the probability, it takes the value 1.

So if you take the expectations it just becomes this product okay, now you can do this little bit of work here and defined this LX to be $\log_2 P_X$ of 0 by P_X of 1 okay and likewise for LY and LZ also and you can show that this is true, so this is a result I showed earlier on with a tan hyperbolic okay, this is the crucial result, the tan hyperbolic of LZ by 2 is tan hyperbolic of LX by 2 times tan hyperbolic of LY by 2 and then you use the fact that is an odd function and separated into sign and absolute value and when you extend Z to $X_1+X_2+ X_3+XW$ the same thing applies okay the sign multiply for LZ , absolute value works through this function F okay.

So we have seen all this before I am just doing a quick recap to tell you why the SPC, SISO decoder is easy to compute, that is the main thing here okay, so one can do a good implementation of this, there are which possible to do it in hard where quite efficiently.

(Refer Slide Time: 16:45)



Minimum approximation

- $|z| = \text{abs}(f [f(l_1) + f(l_2) + \dots + f(l_w)])$
 - $f(x) = \log \tanh (|x|/2)$
- $|z| \approx \min\{\text{abs}(l_1), \text{abs}(l_2), \dots, \text{abs}(l_w)\}$
- Saves computation of nonlinear function
- Used in a slightly modified format
- Almost no loss in the approximation in practice

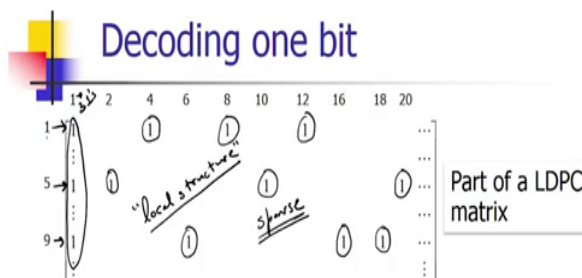


PROF. ANDREW THANGARAJ
IIT MADRAS

Soft-Input Soft-Output Iterative Message Passing Decoder for LDPC Codes

Okay, so nevertheless people tend to approximate this, so there is an, there is a useful reason why you want to approximate it, it is good to have further lesser computations and if it is good enough, if it is close enough to actual the exact completion why do all that, so most people do this minsum approximation you called it minimum approximation, so let us just say minsum approximation with just the best thing to do here, so we saw this before so the absolute value is dominated by the minimum of these values okay, so let us say if the computation of non-linear function, now this really almost no loss in practice, so usually one can do a little offset as well okay, so this is also something you seen before and we will use it in the decoder.

(Refer Slide Time: 17:30)



- Use the received value corresponding to Bit 1
 - 1st estimate, $l_{10} = \text{LLR} | r_1 = 2r_1 / \sigma^2$ channel
- Use parity checks that involve the bit *SPC decoder (siso) extrinsic*
 - Row 1: $c_1 + c_4 + c_8 + c_{12} = 0$; 2nd estimate, $l_{11} = \text{LLR} | r_4, r_8, r_{12}$
 - Row 5: $c_1 + c_2 + c_{10} + c_{20} = 0$; 3rd estimate, $l_{15} = \text{LLR} | r_2, r_{10}, r_{20}$
 - Row 9: $c_1 + c_6 + c_{16} + c_{18} = 0$; 4th estimate, $l_{19} = \text{LLR} | r_6, r_{16}, r_{18}$
- Same can be done for all bits

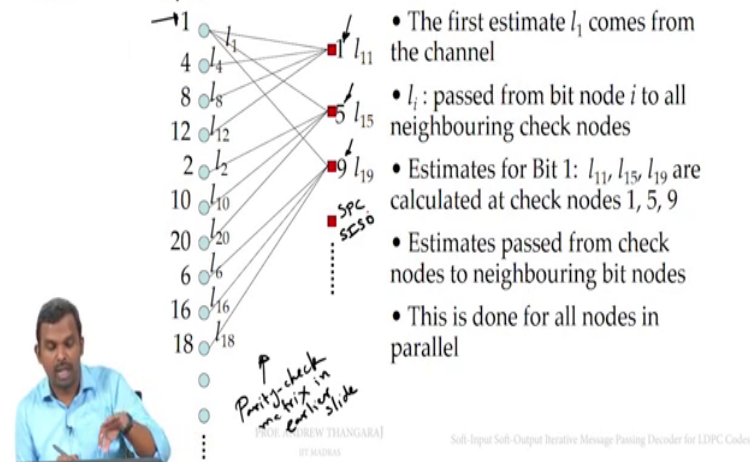


PROF. ANDREW THANGARAJ
IIT MADRAS

Soft-Input Soft-Output Iterative Message Passing Decoder for LDPC Codes



First Iteration in Tanner Graph




Okay, so like I said this is the, this is actually the tanner graph corresponding to the parity check matrix we saw before okay, in earlier slide okay, so let me show you that earlier slide, a real quick okay, so this is the parity check matrix you have row 1 connected to column 1, column 4, column 8 and column 12 okay, so if you go back to the tanner graph you have bit node 1 connected to check node 1, check node 5 and check node 8 okay.

So this computation, the local structure computation is very very conveniently captured in the tanner graph okay and what does the tanner graph further capture? It further captures the message passing through which these bits exchange the local structure information without leading to any problems; it captures all of that very very nicely.

Okay, so if you look at the first iteration okay, where you obtain this estimates okay, this LI is passed from bit node i to check node for all the bits okay, so instead of just talking about the first bit I can now talk about all the bit nodes, now all the bit nodes will be passing the LLR to all the check nodes that they are connected to and all the check nodes will do the SPDC code completion okay, so the check nodes will do SPC, SISO computation okay.

So in the first integration you are going to imagine there channel received value comes into the bit nodes, you do the LLR, channel LLR calculation $2 \ln \frac{1 + \sigma^2 R_A}{1 - \sigma^2 R_A}$ and then that LLR gets passed to all the check nodes on the tanner graph okay, so the tanner graph nicely captures this passing of messages and then what does the check node do? Check node does single parity check SISO computation okay and then it passes back the extrinsic on each of the at just back to the bit nodes okay, so that is how you get this L_{11}, L_{15}, L_{19} okay,

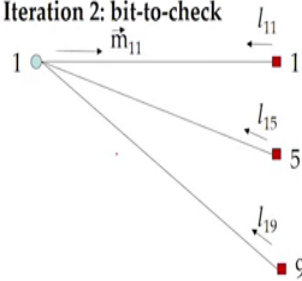
(Refer Slide Time: 19:42)



Next Iteration (i)


Iterations are done to involve more bits and checks in the decoding

Iteration 2: bit-to-check



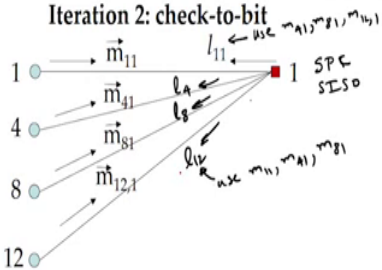
- $\vec{m}_{11} = l_1 + l_{15} + l_{19}$; $\vec{m}_{15} = l_1 + l_{11} + l_{19}$; $\vec{m}_{19} = l_1 + l_{11} + l_{15}$
- Similar rule for other bit nodes

PROF. ANDREW THANGARAJ
 IIT MADRAS
 Soft-Input Soft-Output Iterative Message Passing Decoder for LDPC Codes



Next Iteration (ii)

Iteration 2: check-to-bit



- Repeat computation of l_{11} , l_{15} and l_{19} using \vec{m}_{41} , \vec{m}_{81} and $\vec{m}_{12,1}$
- Similar for all check nodes

PROF. ANDREW THANGARAJ
 IIT MADRAS
 Soft-Input Soft-Output Iterative Message Passing Decoder for LDPC Codes

So how does, how exactly does this happen, so let us look at it very very closely here okay, so let me show how that happens very very slowly, so you have a the check node 1 which is connected to bit node 1, bit node 4, bit node 8 and bit node 12 okay, so the message or the LLR comes from bit node 1, I will denote as M_{11} , bit node 4 is M_{41} , bit node 8 is M_{81} , bit node 12 is $M_{12,1}$, these are the four LLR that are coming in okay, now this check node 1 does like I said SPC, SISO okay, it does calculation, it uses the four guides and computes L_{11} okay and so now you have to just repeat the computation throughout, now this will also send an L_4 okay in this and act to this in L_8 and L_{12} also okay, so it will send back information to all the bit nodes it is connected to.

So now L11 will involve, will use M4 1, M8 1 and M12 1, on L12 on the other hand will use M11, M41 and M81 okay, so this is what is critical in the message passing and now it can be very nicely succinctly captured okay, so L11 uses M41, M81 and M12 1 to compute the extrinsic LLR for bit 1, what is L12 on the other hand, it uses M11, M41 and M81 and computes the extrinsic information for bit 12 okay, so likewise in one go you can do all the computation okay.

(Refer Slide Time: 21:29)

Next Iteration (i)

- Iterations are done to involve more bits and checks in the decoding

Iteration 2: bit-to-check

Channel $l_1 \rightarrow 1$

repetition code SISO

send Extrinsic

\hat{m}_{11} l_{11} 1

\hat{m}_{15} l_{15} 5

\hat{m}_{19} l_{19} 9

- $\hat{m}_{11} = l_1 + l_{15} + l_{19}$; $\hat{m}_{15} = l_1 + l_{11} + l_{19}$; $\hat{m}_{19} = l_1 + l_{11} + l_{15}$
- Similar rule for other bit nodes

PROF. ANDREW THIANGARAJ
BY MADRAS

Soft-Input Soft-Output Iterative Message Passing Decoder for LDPC Codes

So now that you have done the check node SISO computation, how do you do the next iteration? How do you keep moving to the next iteration? What you do in the next iteration is very very interesting okay, so you have on the bit node you have what comes from the channel okay, L1 is what comes from the channel right okay and then in the first iteration you have L11 coming from this check node, L15 coming from this check node, fifth check node, L19 coming from this check node okay, so what you have here is a repetition code okay, the same code is sort of repeated forth time, the same bit I am sorry, the same bit is repeated four times, it came through the channel and you got one LLR L1, it seems to be coming through the check node 1 okay and you have LLR for it L11 likewise you, it is coming through from check node 5 it has an LLR L15 and the same bit it is coming through this sort of this check node 9 with the belief L19 okay, you are getting four different beliefs, they are all for the same bit okay, so you actually have a repetition code.

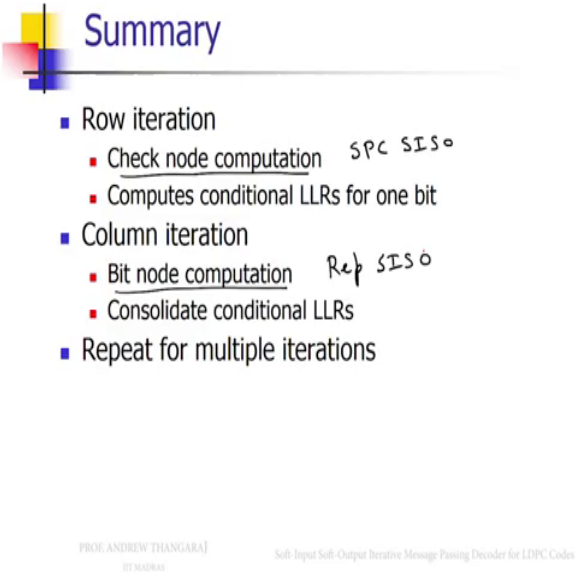

So what you need to implement here is a SISO for a repetition code okay, so that the very neat way in which the simple SISO decoders for the repetition code and the SPC code sort of work to get it, so how do you compute M11, M11 is L1+L15+L19, remember the repetition

code you just add the LLRs, how do you compute M15 and M19 in this direction, it is given here M15 is a L1+L11+L19, the other 2 guys add up okay.

So these are all remember you have to send only extrinsic, this is an important rule in message passing, you cannot send total beliefs, if you send total beliefs things will go wrong, there will be too much dependency in these LLRs and you will not be doing correct operations okay, so you have to only send extrinsic information.


So if you get something, so if you get L11 from this check node 1, you do not send anything with L11 back to that guy okay you only send L1+L15+L19 back to that guy okay, so that is what happens here, it is a very neat way in which iterations proceed okay.

(Refer Slide Time: 23:48)



Summary

- Row iteration
 - Check node computation *SPC SISO*
 - Computes conditional LLRs for one bit
- Column iteration
 - Bit node computation *Rep SISO*
 - Consolidate conditional LLRs
- Repeat for multiple iterations



PROF. ANDREW THANGARAJ
© 2014 NPTEL
Soft-Input Soft-Output Iterative Message Passing Decoder for LDPC Codes

So once again if you want to summarise okay, there is a row iteration okay where all the LLRs are processed in every row okay, this is the same as the check node computation happens here, so likewise this column iteration which is a bit node computation okay, so this is a SPC, SISO, this is a repetition SISO okay, both of them with some approximations can be made to involve only just linear operations, either additions or minimum or something like that which is a linear in the sense, no major non-linear function going on, it is a very easy thing to implement okay.

So now what you did in one iteration, you can keep repeating again and again and again okay and when you repeat again and again and again it turns out if you imagine little bit, for every bit you are using information from all other received values okay, so this message passing

method okay just by using a SISO decoder for SPC codes and for repetition codes one can keep iterating and repeating and repeating and improve the believes okay.

So what we are going to see next is a couple of simple examples where this is illustrated okay, one example where you will see the parity check matrix, you will see the received vector and you will see how it goes through the iterations and how the believes change and how they improve okay, so this will give us a focus on an implementation and will help us write Matlab programs for these decoders okay, so that will be the next lecture.