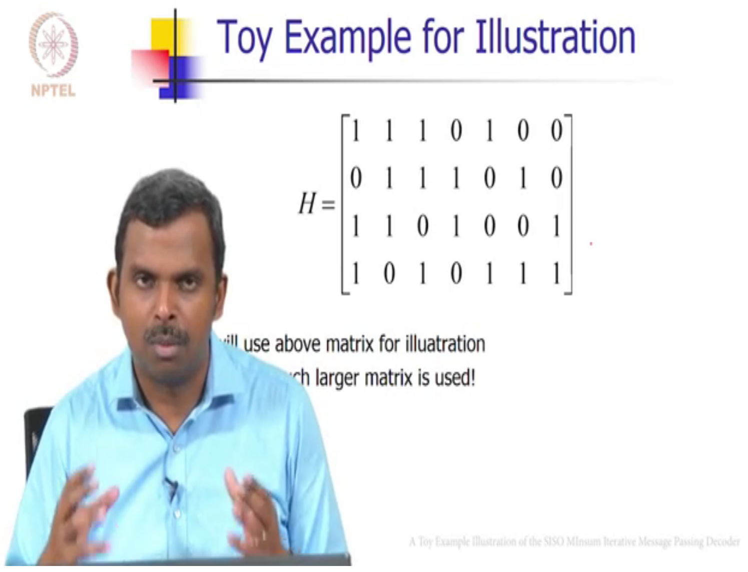
LDPC & Polar codes in 5G Standard
Prof. Andrew Thangaraj
Department of Electrical Engineering
Indian Institute of Technology Madras
A Toy Example Illustration of the SISO Minsum Iterative Message Passing Decoder

(Refer Slide Time: 0:17)



Toy Example for Illustration

$$H=\begin{bmatrix}1&1&1&0&1&0&0\\0&1&1&1&0&1&0\\1&1&0&1&0&0&1\\1&0&1&0&1&1&1\end{bmatrix}$$

'ill use above matrix for illuatration

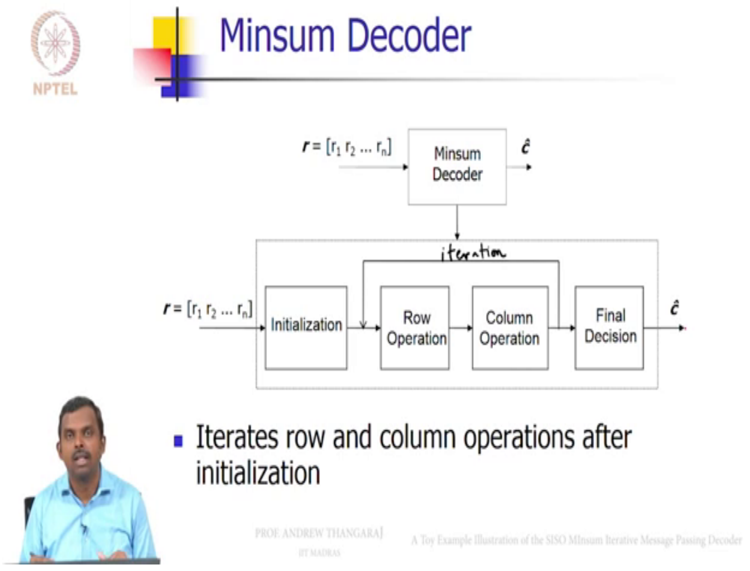
'ch larger matrix is used!

A Toy Example Illustration of the SISO Minsum Iterative Message Passing Decoder

Alright so in the previous lecture saw the definition and description of the message passing decoder sort of a philosophy of return of a lose way, how the believes are improved for bits interactively one after the other okay but while I described the decoder with notation and general way it is always good to see a specific example an exactly what operations happen to make the ideas firm in our heads okay, so that is what we are going to do in this little lecture, will take a very small toy example okays.

So this parity check matrix that I show here is just 4 x 8 parity check matrix, nobody really uses this parity check matrix with message passing decoders and all that, in fact you can implement easily optimal decoder for such codes, you do not need a message passing and all that but it is useful for illustration, just to show you what the operations are? What happens? What are the row operations? What are the column operations? Etc. this is the nice illustration; typically much larger matrix is use but actually extending it to larger matrix is not really difficult, just the basic ideas are important.

(Refer Slide Time: 1:23)



Minsum Decoder

- Iterates row and column operations after initialization

PROF. ANDREW THANGARAJ IIT MADRAS

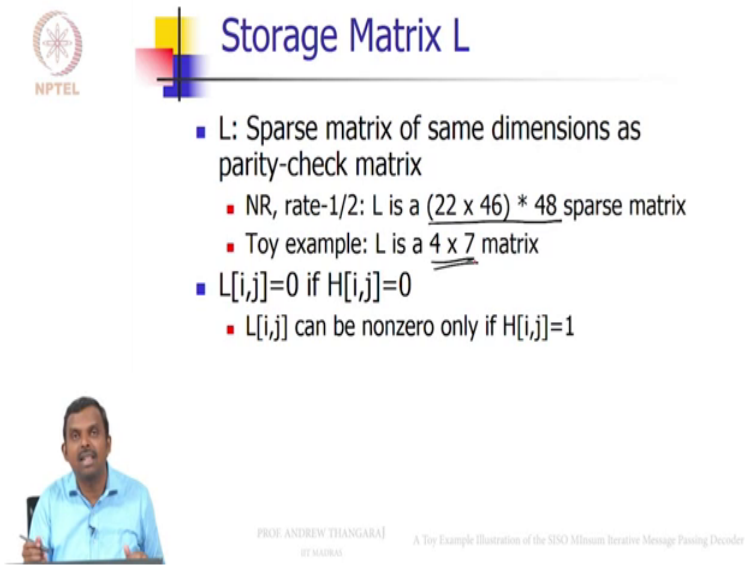
A Toy Example Illustration of the SISO Minsum Iterative Message Passing Decoder

So let us get started, so we are going to focus on a minsum decoder, we will not be doing locked and hyperbolic and all that just minsum and how does that work, you have the received vector R1 to RN entering your minsum decoder and you will put out an estimate for the code word C cap, okay, now, R enters this, there is an initialisation which may involve some quantisation etc whatever you use or it might go into the parity check matrix in some way, okay hope or at the cana graph and some way, then you have a row operation, column operation and then iteration.

Okay so this is the iteration step, okay you can do as many iteration as you want, typically people do ten iteration, eight iteration, how many of iteration you want as you do more and more iterations, the performance will keep slowly and slowly improving okay, so eventually you will see after some eight or ten iterations you may not say too, you may not see too much improvement in error rates okay, so then you choose to stop okay, finally when you stopped you can make a final decision and that will put out your C cap okay, so this is sort of the block diagram of what happens, you iterate between row operation and column operation.

So this initialisation and storages sort of important, I will mention that as well, so all these messages that are getting exchanged have to be stored somewhere right and you have to access them and do that, so that is something avail a straight as well okay.

(Refer Slide Time: 2:42)



Storage Matrix L

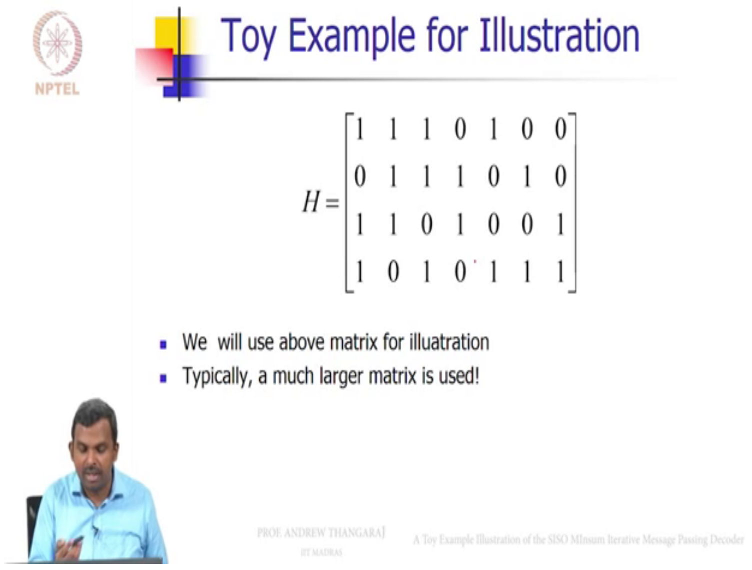
- L: Sparse matrix of same dimensions as parity-check matrix
  - NR, rate-1/2: L is a $\underline{(22 \times 46)} * \underline{48}$ sparse matrix
  - Toy example: L is a $\underline{4 \times 7}$ matrix
- L[i,j]=0 if H[i,j]=0
  - L[i,j] can be nonzero only if H[i,j]=1

PROF. ANDREW THANGARAJ
IIT MADRAS

A Toy Example Illustration of the SISO MInsum Iterative Message Passing Decoder

Alright so the storage matrix is what will be use, this notion of the sparse matrix which has the same dimension as the parity check matrix okay, so if you take for instance the new radio or the 5G standard rate half code L will be 22 x 46 times 48 a sparse matrix, so this is just an example it could be of this size or it can be any other size and the toy example is just 4 x 7 okay, so like I said the real L matrix or the real parity check matrix will be very very large okay but we are taking a toy example okay.

(Refer Slide Time: 3:26)



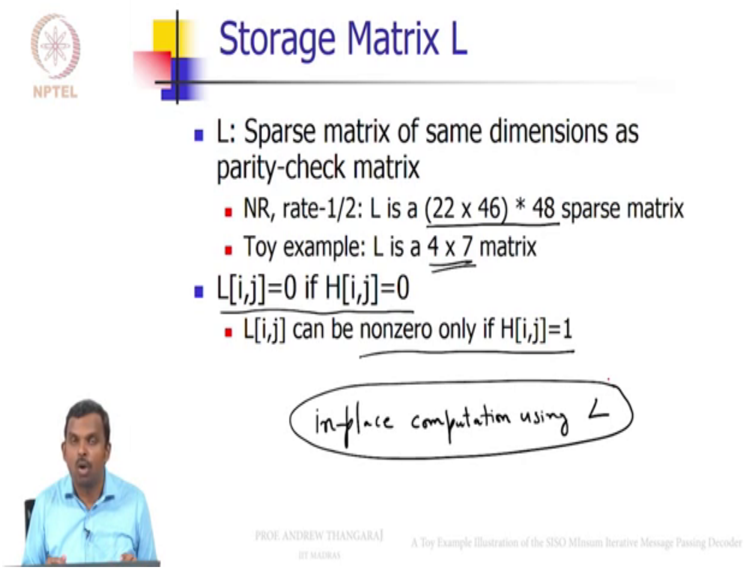
Toy Example for Illustration

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

- We will use above matrix for illuatration
- Typically, a much larger matrix is used!

PROF. ANDREW THANGARAJ
IIT MADRAS

A Toy Example Illustration of the SISO MInsum Iterative Message Passing Decoder



Storage Matrix L

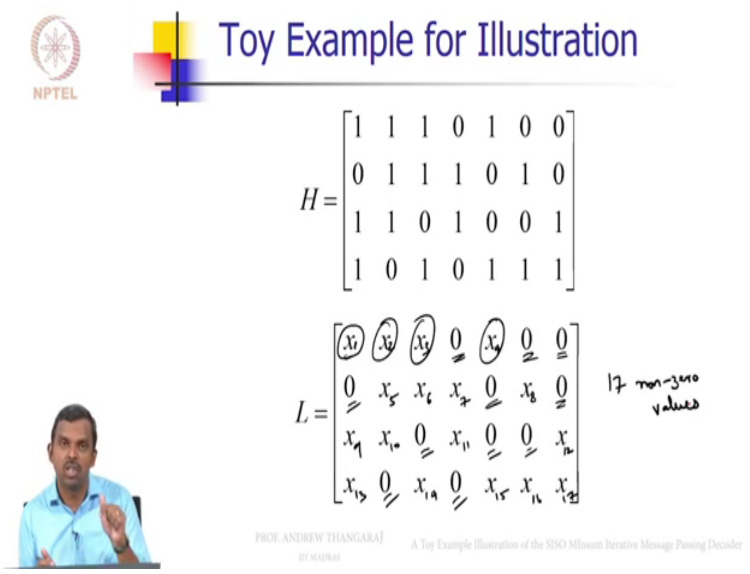
- L: Sparse matrix of same dimensions as parity-check matrix
  - NR, rate-1/2: L is a $\underline{(22 \times 46)} * \underline{48}$ sparse matrix
  - Toy example: L is a $4 \times \underline{7}$ matrix
- $\underline{L[i,j]=0 \text{ if } H[i,j]=0}$
  - L[i,j] can be nonzero only if $\underline{H[i,j]=1}$

in-place computation using L

PROF. ANDREW THANGARAJ IIT MADRAS — A Toy Example Illustration of the SISO MInsum Iterative Message Passing Decoder

So the toy example, okay so this, I am sorry I think I said it 4 x 8, this parity check matrix is 4 x 7 okay, so the toy example is just 4 x 7 matrix, real code will be very very large, okay the same thing is blown up but remember its sparse. Okay the sparsity is quite important okay, like I said it has the same dimensions, it also this L matrix also has the same sparsity structure as your parity check matrix, whenever the parity check matrix is zero, this L will also be zero okay, so it can be nonzero only when the H of IJ is one okay, so this storage matrix is very important, you have to store it and all your operations will be in place on the storage matrix, in-place computation using this L, so that is what we will do, at least for their purpose of implementation and focusing or attention on understanding how the operations happens okay.

(Refer Slide Time: 4:25)



Toy Example for Illustration

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$L = \begin{bmatrix} x_1 & x_2 & x_3 & \underline{0} & x_4 & \underline{0} & \underline{0} \\ \underline{0} & x_5 & x_6 & x_7 & \underline{0} & x_8 & \underline{0} \\ x_9 & x_{10} & \underline{0} & x_{11} & \underline{0} & \underline{0} & x_{12} \\ x_{13} & \underline{0} & x_{14} & \underline{0} & x_{15} & x_{16} & x_{17} \end{bmatrix}$$

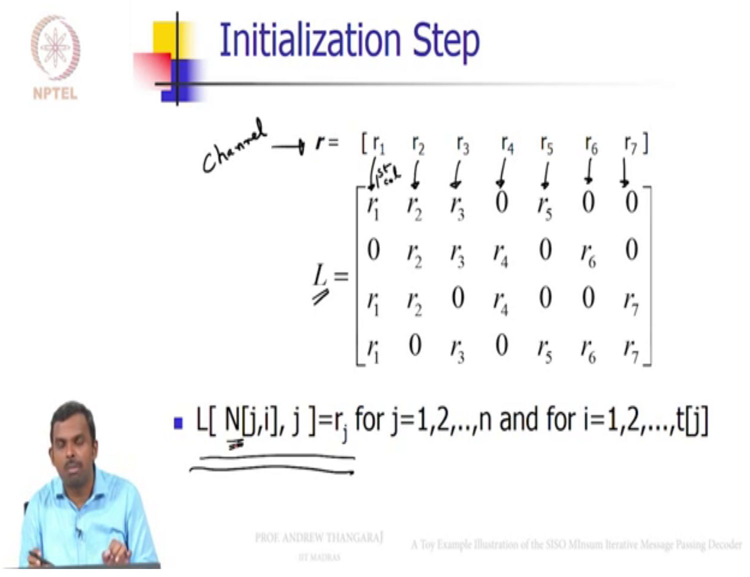
17 non-zero values

PROF. ANDREW THANGARAJ IIT MADRAS — A Toy Example Illustration of the SISO MInsum Iterative Message Passing Decoder

Okay, so here is the toy example, my parity check matrix had once in the first row 110101111 like that okay, so my storage matrix L will have possible nonzero values wherever there is one in the parity check matrix, if there is a zero in the parity check matrix, this L will also be zero okay, so keep that in mind, this is very important, there will become zero otherwise okay, so I have shown the same value X here but just for convenience so I should have actually put X1, X2, X3, X4 okay X5, X6, X7, X8, so this value will keep changing okay, so every value might be something else, it will keep changing with iterations, so just for convenience I shown just one value, there are 17 values, 17 nonzero values in this two-dimensional matrix okay, so every value for instant X4 is in the first row fourth column, at 17 is in the fourth row seventh column okay.

So it should be able to pull the values based on the row and column entries okay, so this is my storage matrix, the storage matrix is quite important, we will use at, in the illustration you will see I will do in-place competition, so I will put values there, I will do some operations, I replace the values, I keep doing this illiterately okay, as I do more and more iterations, the messages will be stored on the storage matrix okay that is very important.

(Refer Slide Time: 5:56)



Initialization Step

$$\text{Channel} \longrightarrow r = [r_1\ r_2\ r_3\ r_4\ r_5\ r_6\ r_7]$$

$$L = \begin{bmatrix} r_1 & r_2 & r_3 & 0 & r_5 & 0 & 0 \\ 0 & r_2 & r_3 & r_4 & 0 & r_6 & 0 \\ r_1 & r_2 & 0 & r_4 & 0 & 0 & r_7 \\ r_1 & 0 & r_3 & 0 & r_5 & r_6 & r_7 \end{bmatrix}$$

- L[N[j,i], j]=r_j for j=1,2,..,n and for i=1,2,...,t[j]

PROF. ANDREW THANGARAJ IIT MADRAS

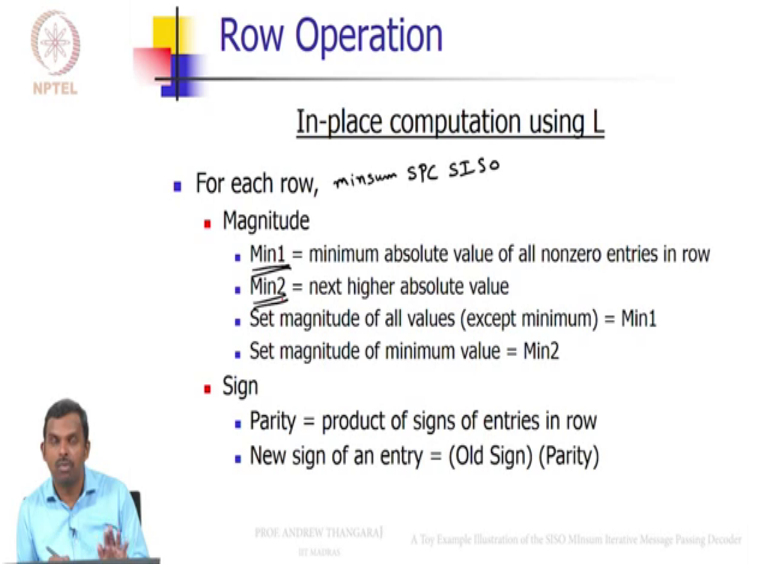
A Toy Example Illustration of the SISO Minsum Iterative Message Passing Decoder

Okay so what is the initialisation step okay, so you have the receive value from the channel, you remember this, Channel gave you this, there will be seven received values because I am looking at length seven code in my toy example okay, I receive the values R1 through R7 okay, it is that okay, so what do I do here, you take R1 and store it in the nonzero positions of in the first column okay, all nonzero positions in the first column of L will have R1 okay, that is the initialisation step okay, so all nonzero positions in the second column will have R2, all nonzero positions in the third column will have R3, all nonzero positions in the fourth column and so on okay.

So you just imagine this, you can visualise that the received values are dropping on the columns and wherever there are nonzero values they go and store themselves their okay, so that is what happens, R1 Store on the first column, R2 on the second column, R3 on the third column likewise okay, so I have, this is just notation, just to show how I am doing it.

I believe, I am using this N of J of I to denote the neighbouring values and all that, it is not too crucial for us, most important is to understand how these initialisation happens, so hopefully this is clear, take the received values every nonzero positions that particular column is occupied by that corresponding received value okay.

(Refer Slide Time: 7:34)



Row Operation

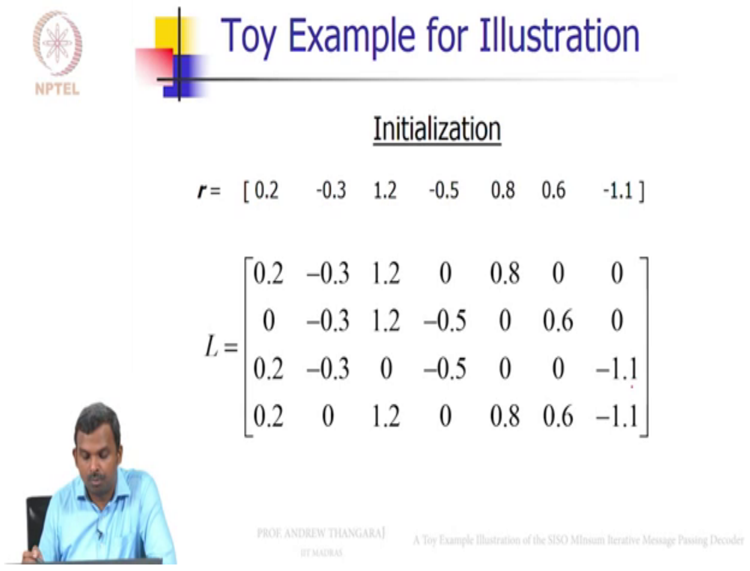
In-place computation using L

- For each row, minsum SPC SISO
  - Magnitude
    - Min1 = minimum absolute value of all nonzero entries in row
    - Min2 = next higher absolute value
    - Set magnitude of all values (except minimum) = Min1
    - Set magnitude of minimum value = Min2
  - Sign
    - Parity = product of signs of entries in row
    - New sign of an entry = (Old Sign) (Parity)

NPTEL
PROF. ANDREW THANGARAJ
IIT MADRAS
A Toy Example Illustration of the SISO Minsum Iterative Message Passing Decoder

Okay, so the row operation is an in-place computation using L, if you remember correctly for each row I have to do our minsum SPC SISO decode, it involves finding the two minimum okay, you find the least absolute value among the nonzero values in that row, the next least absolute value and then use a magnitude of all values except the least value as min1 okay and said the magnitude of the least value as min2 and for the sign you first find the product of all the nonzero values in the row and for every particular entry you either flip or not flip depending upon the overall parity, the overall parity is satisfied, you do not flip, if your overall parity is not satisfied you flipped okay, so this is the SISO minsum SPC decoder we have seen this before okay.

(Refer Slide Time: 8:34)



Toy Example for Illustration

Initialization

$r = [\,0.2 \quad -0.3 \quad 1.2 \quad -0.5 \quad 0.8 \quad 0.6 \quad -1.1\,]$

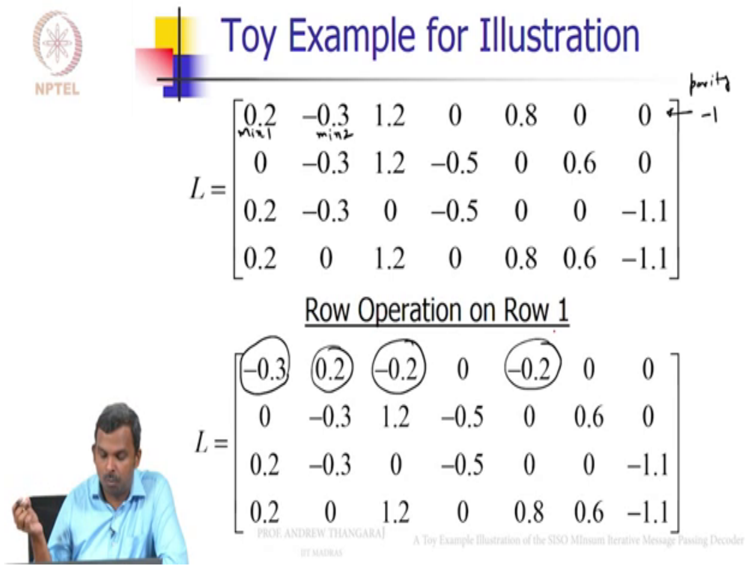
$$L = \begin{bmatrix} 0.2 & -0.3 & 1.2 & 0 & 0.8 & 0 & 0 \\ 0 & -0.3 & 1.2 & -0.5 & 0 & 0.6 & 0 \\ 0.2 & -0.3 & 0 & -0.5 & 0 & 0 & -1.1 \\ 0.2 & 0 & 1.2 & 0 & 0.8 & 0.6 & -1.1 \end{bmatrix}$$

PROF. ANDREW THANGARAJ
IIT MADRAS

A Toy Example Illustration of the SISO Minsum Iterative Message Passing Decoder

So let us see how we do this okay, so here I have done the initialisation, I have taken a simple value for R 0.2, -0.3, 1.2, -0.5, 0.8, 0.6, -1.1 and you can see in the initialisation step the particular receive value occupies all the nonzero positions in the L matrix, what are the nonzero positions in the L matrix once again, wherever you have once in the parity check matrix, you will have the nonzero positions in the storage matrix L okay, so you can see this, what has happened here, it has done okay, hopefully this is clear.

(Refer Slide Time: 9:15)



Toy Example for Illustration

$$L = \left[\begin{array}{ccccccc|c} \underset{\text{min1}}{0.2} & \underset{\text{min2}}{-0.3} & 1.2 & 0 & 0.8 & 0 & 0 & \xleftarrow{} \overset{\text{parity}}{-1} \\ 0 & -0.3 & 1.2 & -0.5 & 0 & 0.6 & 0 & \\ 0.2 & -0.3 & 0 & -0.5 & 0 & 0 & -1.1 & \\ 0.2 & 0 & 1.2 & 0 & 0.8 & 0.6 & -1.1 & \end{array}\right]$$

Row Operation on Row 1

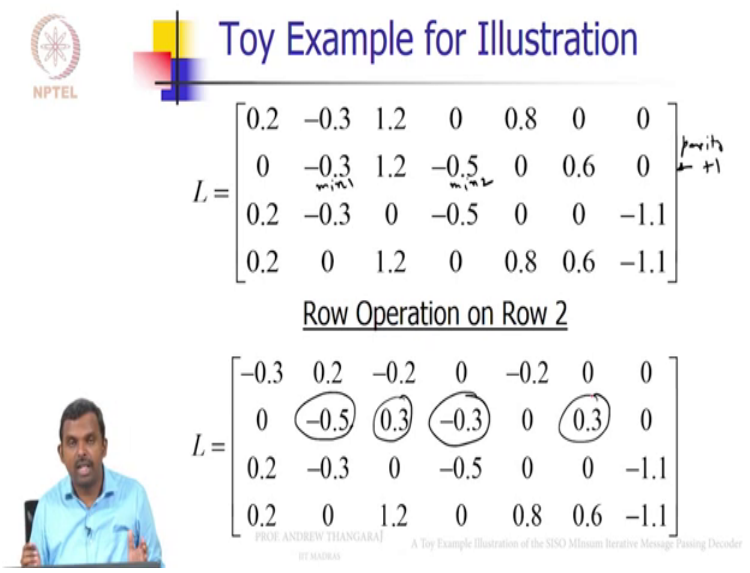
$$L = \begin{bmatrix} -0.3 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & -0.3 & 1.2 & -0.5 & 0 & 0.6 & 0 \\ 0.2 & -0.3 & 0 & -0.5 & 0 & 0 & -1.1 \\ 0.2 & 0 & 1.2 & 0 & 0.8 & 0.6 & -1.1 \end{bmatrix}$$

PROF. ANDREW THANGARAJ
IIT MADRAS

A Toy Example Illustration of the SISO Minsum Iterative Message Passing Decoder

So now how do you do row operation, if you look at the first row, you have absolute value 0.2, 0.3, 1.2, 0.8, so this is min1, this is min2 and what is the overall parity, overall parity is -1 okay, if you take the product of the signs, this is positive, first one is positive, negative, positive, positive, so if you multiply the signs you gets -1, which means you have to flip the signs okay, so in the least position you put the next min2 okay, but you put minus, why because you have to flip the signs, so you have 0.3 here okay, so you will see 0.3 here, -0.3 here, then everything else is 0.2 in absolute value and you have flipped the signs okay, so the original sign here was negative, so you made it positive, the original sign here was positive, so you made it negative here but you kept all the values point okay, so that is the row operation on row1, is that okay as simple as that okay, just find the two least minimums and then do the substitution of the values and adjustment of the sign okay, so now you can repeat this for every row.

(Refer Slide Time: 10:28)



Toy Example for Illustration

$$L=\left[\begin{array}{ccccccc}0.2 & -0.3 & 1.2 & 0 & 0.8 & 0 & 0\\ 0 & \underset{\text{min1}}{-0.3} & 1.2 & \underset{\text{min2}}{-0.5} & 0 & 0.6 & 0\\ 0.2 & -0.3 & 0 & -0.5 & 0 & 0 & -1.1\\ 0.2 & 0 & 1.2 & 0 & 0.8 & 0.6 & -1.1\end{array}\right]\begin{array}{l}\\ \text{parity} \leftarrow +1\\ \\ \\ \end{array}$$

Row Operation on Row 2

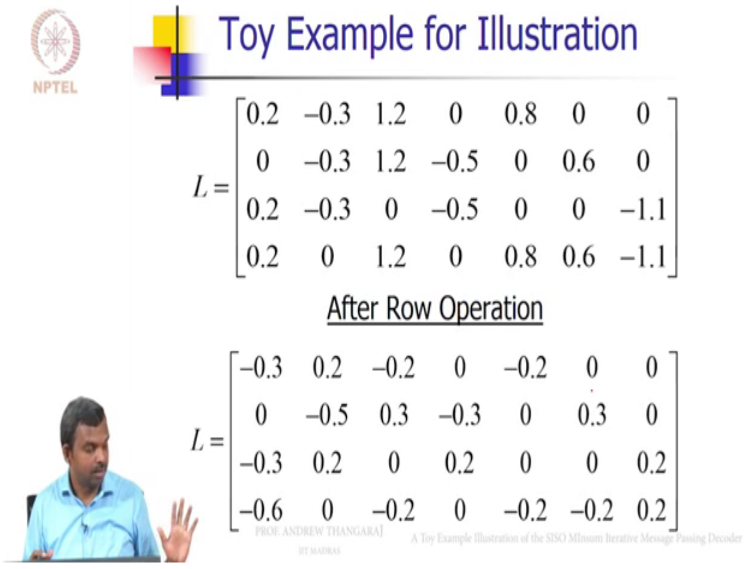
$$L=\left[\begin{array}{ccccccc}-0.3 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0\\ 0 & \boxed{-0.5} & \boxed{0.3} & \boxed{-0.3} & 0 & \boxed{0.3} & 0\\ 0.2 & -0.3 & 0 & -0.5 & 0 & 0 & -1.1\\ 0.2 & 0 & 1.2 & 0 & 0.8 & 0.6 & -1.1\end{array}\right]$$

PROF. ANDREW THANGARAJ IIT MADRAS

A Toy Example Illustration of the SISO Minsum Iterative Message Passing Decoder

Okay, so you do row2, so you see here this is min1, this is min2 okay and the overall parity is +1 Right this -1 into +1 into -1 into +1 it became +1 okay, so now what do I do, the min1 I replace with the min2 okay -0.5, everything else is 0.3 and the sign is retained as the original sign okay, wherever is positive, is positive okay, so I do not want illustrate any further in this, hopefully you can see, in every row, you do row operations like this.

(Refer Slide Time: 11:05)



Toy Example for Illustration

$$L = \begin{bmatrix} 0.2 & -0.3 & 1.2 & 0 & 0.8 & 0 & 0 \\ 0 & -0.3 & 1.2 & -0.5 & 0 & 0.6 & 0 \\ 0.2 & -0.3 & 0 & -0.5 & 0 & 0 & -1.1 \\ 0.2 & 0 & 1.2 & 0 & 0.8 & 0.6 & -1.1 \end{bmatrix}$$

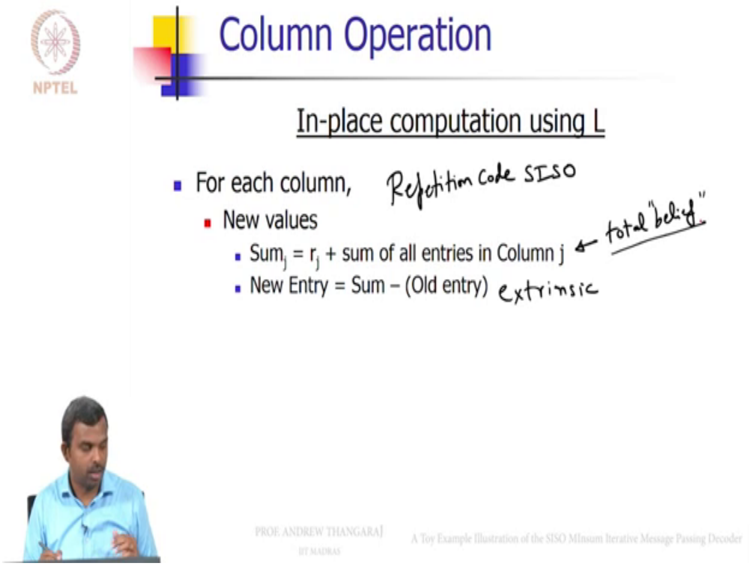
After Row Operation

$$L = \begin{bmatrix} -0.3 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & -0.5 & 0.3 & -0.3 & 0 & 0.3 & 0 \\ -0.3 & 0.2 & 0 & 0.2 & 0 & 0 & 0.2 \\ -0.6 & 0 & -0.2 & 0 & -0.2 & -0.2 & 0.2 \end{bmatrix}$$

PROF. ANDREW THANGARAJ IIT MADRAS A Toy Example Illustration of the SISO Minsum Iterative Message Passing Decoder

Okay after you done with the row operation, this system matrix you will have okay, so in every row you have the processed value for the entries there okay, how did we do the processing, we did minsum SSPC, SISO decoding, the simple min1, min2 method okay, that is it, now the row operation is done.

(Refer Slide Time: 11:27)



Column Operation

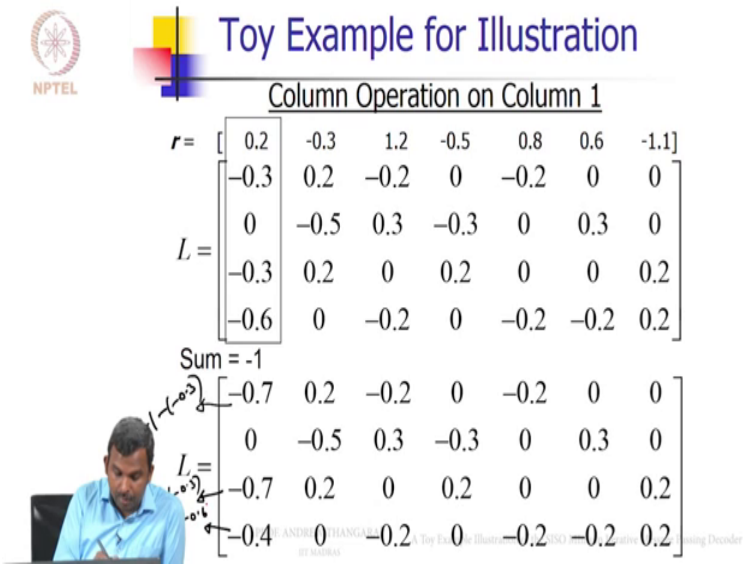
In-place computation using L

- For each column, Repetition Code SISO
  - New values
    - $\text{Sum}_j = r_j$ + sum of all entries in Column j ← total "belief"
    - New Entry = Sum – (Old entry) extrinsic

PROF. ANDREW THANGARAJ IIT MADRAS A Toy Example Illustration of the SISO Minsum Iterative Message Passing Decoder

Now you can do the column operation, for column operation once again we will do in-place competition using L, for each column we will find the sum of all the entries okay in the column plus the received value okay remember the received value came from the channel, you have to do reputation code SISO decoding, you take the channel value, take all the values, added all up okay, so you will get sum J okay, then the new entry will be the sum minus the old entry okay, so this is the way in which we do the extensive competitions for repetition code okay, once again what is this? This is repetition code SISO decoding okay, so this is actually the total belief or LLR, I call it belief, so that is simplified word here okay, so this is the total belief, any decision you make will be based on the total belief okay and particular iteration.

(Refer Slide Time: 12:28)



Toy Example for Illustration

Column Operation on Column 1

$$r = [\,\boxed{0.2}\ \ -0.3\ \ 1.2\ \ -0.5\ \ 0.8\ \ 0.6\ \ -1.1\,]$$

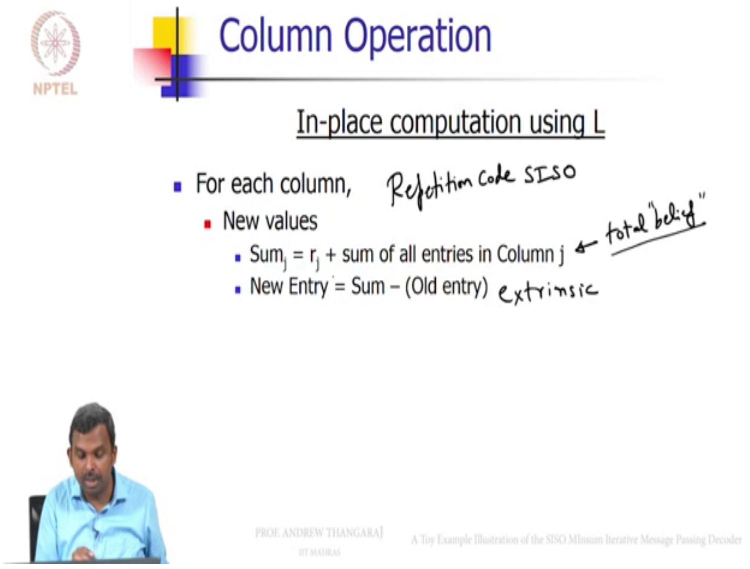
$$L = \begin{bmatrix} -0.3 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & -0.5 & 0.3 & -0.3 & 0 & 0.3 & 0 \\ -0.3 & 0.2 & 0 & 0.2 & 0 & 0 & 0.2 \\ -0.6 & 0 & -0.2 & 0 & -0.2 & -0.2 & 0.2 \end{bmatrix}$$

Sum = -1

$$L = \begin{bmatrix} -0.7 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & -0.5 & 0.3 & -0.3 & 0 & 0.3 & 0 \\ -0.7 & 0.2 & 0 & 0.2 & 0 & 0 & 0.2 \\ -0.4 & 0 & -0.2 & 0 & -0.2 & -0.2 & 0.2 \end{bmatrix}$$

So this is a, let me once again illustrate how this happens, now for the column operation in the first column, you look at the first column, entries in the first column are -0.3, -0.3 and -0.6 and the value received from the channel is 0.2, so the sum is going to be -1 here is it correct, so if you add up everything, you will get -1.2 +0.2 which is -1 is that okay, so you see that, alright and then what do we do, how do you find -0.7 here, so you take -0.1 and then you subtract -0.3 from it you get -7 is that okay, -1, minus -0.3 okay that became 0.7, so what about here, again the same thing -1, minus -0.3, some minus the old entry okay, -1, minus -0.6 okay, once again I remind you.

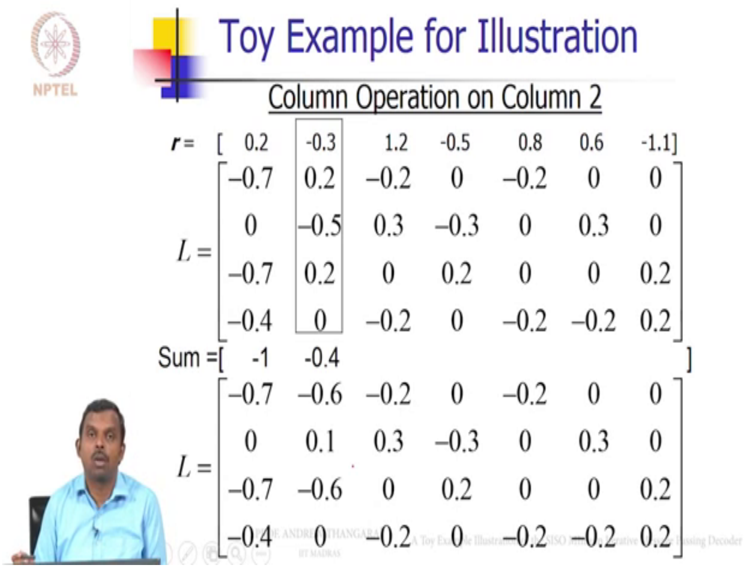
(Refer Slide Time: 13:31)



Column Operation

In-place computation using L

- For each column, Repetition Code SISO
  - New values
    - $\text{Sum}_j = r_j$ + sum of all entries in Column j ← total "belief"
    - New Entry = Sum – (Old entry) extrinsic



Toy Example for Illustration

Column Operation on Column 2

$$\boldsymbol{r} = [\ 0.2\quad \boxed{-0.3}\quad 1.2\quad -0.5\quad 0.8\quad 0.6\quad -1.1\ ]$$

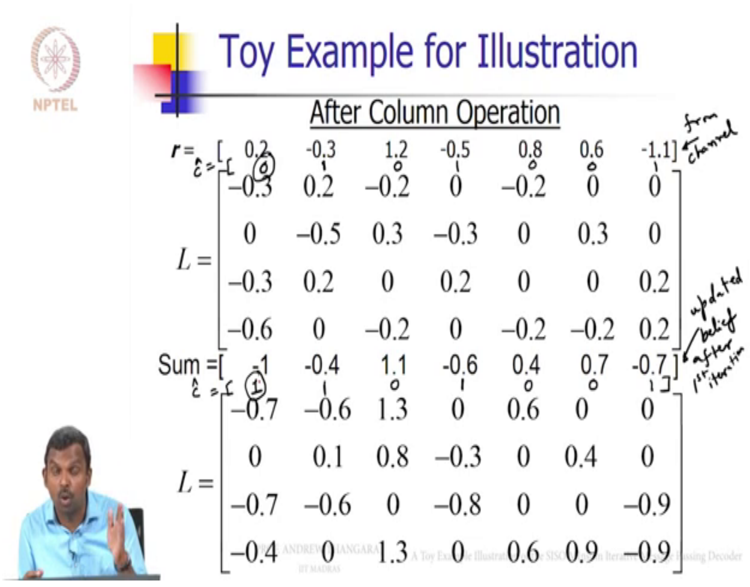
$$L = \begin{bmatrix} -0.7 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & -0.5 & 0.3 & -0.3 & 0 & 0.3 & 0 \\ -0.7 & 0.2 & 0 & 0.2 & 0 & 0 & 0.2 \\ -0.4 & 0 & -0.2 & 0 & -0.2 & -0.2 & 0.2 \end{bmatrix}$$

$$\text{Sum} = [\ -1\quad -0.4\quad \ ]$$

$$L = \begin{bmatrix} -0.7 & -0.6 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & 0.1 & 0.3 & -0.3 & 0 & 0.3 & 0 \\ -0.7 & -0.6 & 0 & 0.2 & 0 & 0 & 0.2 \\ -0.4 & 0 & -0.2 & 0 & -0.2 & -0.2 & 0.2 \end{bmatrix}$$

If you add up all the entries of the column and the corresponding received value and the new entry is going to be the sum minus the old entry okay that is the extensive competition. So you know repeat this for every column okay what we will do in the second column, you add up everything, you get -0.4 okay and then how do you replace the corresponding value -0.4,-0.2 is -0.6, -0.4 minus of -0.5 is +0.1, then -0.4, -0.2 is -0.6, you add it up that way will be done with the column operation.

(Refer Slide Time: 14:00)



Toy Example for Illustration

After Column Operation

$$r = [\;0.2 \quad -0.3 \quad 1.2 \quad -0.5 \quad 0.8 \quad 0.6 \quad -1.1\;] \leftarrow \text{from channel}$$

$$\hat{c} = [\;\boxed{0} \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1\;]$$

$$L = \begin{bmatrix} -0.3 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & -0.5 & 0.3 & -0.3 & 0 & 0.3 & 0 \\ -0.3 & 0.2 & 0 & 0.2 & 0 & 0 & 0.2 \\ -0.6 & 0 & -0.2 & 0 & -0.2 & -0.2 & 0.2 \end{bmatrix}$$

$$\text{Sum} = [\;-1 \quad -0.4 \quad 1.1 \quad -0.6 \quad 0.4 \quad 0.7 \quad -0.7\;] \leftarrow \text{updated belief after 1st iteration}$$

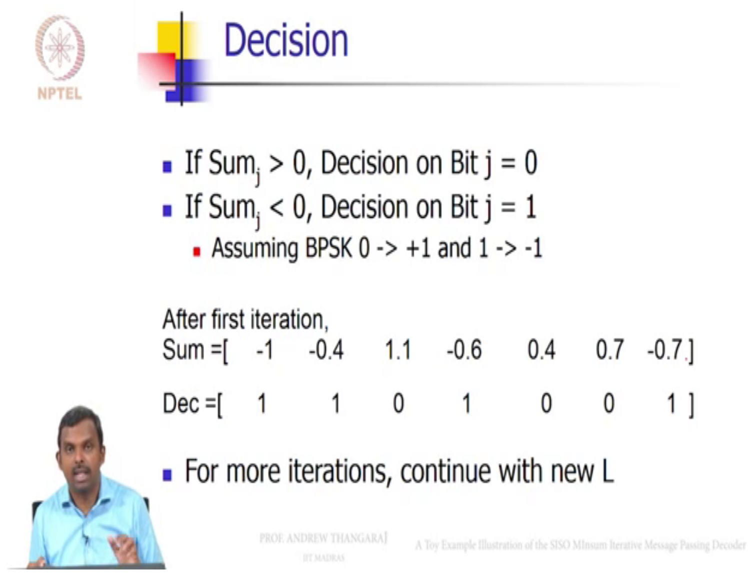
$$\hat{c} = [\;\boxed{1} \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1\;]$$

$$L = \begin{bmatrix} -0.7 & -0.6 & 1.3 & 0 & 0.6 & 0 & 0 \\ 0 & 0.1 & 0.8 & -0.3 & 0 & 0.4 & 0 \\ -0.7 & -0.6 & 0 & -0.8 & 0 & 0 & -0.9 \\ -0.4 & 0 & 1.3 & 0 & 0.6 & 0.9 & -0.9 \end{bmatrix}$$

Now you can finish of all the operations and you get this okay, so this is what is very important, this is the received value from channel okay, so this is your belief, if you do not do any iteration okay, so what do you got from the channel, you are not using anything in the code that is your belief, now this is your updated belief after first iteration, the sum of everything is your updated belief after first iteration, so this is the end of the first iteration when you have computed the total belief at the end of the first iteration and you have also completed the new L matrix, now this is the message from bit node to check node in the second iteration okay, -0.7, -0.7, -0.4 you can do for the processing but this is your belief okay, so if you notice this was 0.2 right, so if you have to decide based on 0.2, you would have decided that the bit C cap is 0 here but notice what happened to C cap here after your iteration okay it became -1 , so this is actually 1 is not it right, so here you have conducted something okay, so in the next bit C cap is just 0, so this is also 0 here, maybe I should write the C cap right below, right so this was 0, 1, I am sorry it should be also 1, 0, 1,0, 0, 1 the C cap after the first iteration became 1, 1, 0, 1, 0, 0, 1 okay that was one bit which got flipped okay, this bit got flipped okay, you can see why that has happened, okay 0.2 became -1, okay so because of your decoding this is happened okay.

So what you did here, you can keep repeating okay, whatever you the first timed you can repeat, do you have the L matrix you can do the row operations, you have the R also, right the channel received value okay and you can repeat the column operation also, okay with once you have L, once you know how to keep updating it and row operation, column operation you can repeat the same thing again and again and every time you remember, the sum is the total belief the end of the iteration, okay you can make decisions based on the sum, you can threshold the some to make decisions to get C cap, okay that is the important thing to remember.

(Refer Slide Time: 16:32)



Decision

- If $\text{Sum}_j > 0$, Decision on Bit j = 0
- If $\text{Sum}_j < 0$, Decision on Bit j = 1
  - Assuming BPSK 0 -> +1 and 1 -> -1

After first iteration,

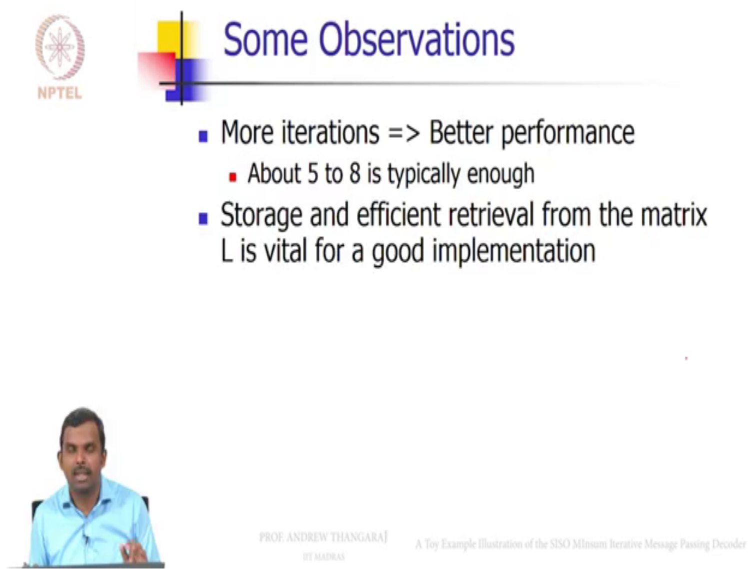
Sum =[	-1	-0.4	1.1	-0.6	0.4	0.7	-0.7]
Dec =[	1	1	0	1	0	0	1]

- For more iterations, continue with new L

PROF. ANDREW THANGARAJ IIT MADRAS A Toy Example Illustration of the SISO MInsum Iterative Message Passing Decoder

Okay, so how do you make decision, if some J is greater than 0, decision on bit J is 0, if it is less than 0, decision on bit J is 1 okay, so after the first iteration this was your some and your decoded values these okay, for more iterations you keep continuing with the new L okay, I illustrated this in the previous slide as well okay.

(Refer Slide Time: 16:52)



Some Observations

- More iterations => Better performance
  - About 5 to 8 is typically enough
- Storage and efficient retrieval from the matrix L is vital for a good implementation

PROF. ANDREW THANGARAJ IIT MADRAS A Toy Example Illustration of the SISO MInsum Iterative Message Passing Decoder

So here are observations, if you implement this you will see, if you do more iterations and if you are parity check matrix is well-designed, if it is a low-density parity check matrix, you will get a better performance with more and more iterations, typically about 5 to 8 are enough for moderate block lengths, if you go to very large block lengths you have to explore the entire the parity check matrix more, so you will need more and more iterations okay, storage and efficient retrieval is the biggest challenge, okay the computations are not the biggest power consuming, area consuming, implementation constraints for LDPC decoder, the storing the entire L matrix and efficiently reading and writing into it parallely, this is the an important constraint but today people have very good implementations working at very fast rates okay.

So that closes the example and the discussion on the decoder for LDPC codes, this is the end of this lecture, will come back in the next lecture and look at how to do a slightly simpler implementation, what are the various modifications for this decoder okay, so actually most people implement what is called a layer decoder in practice, so we will discuss that in the specifically in the next lecture okay, thank you.