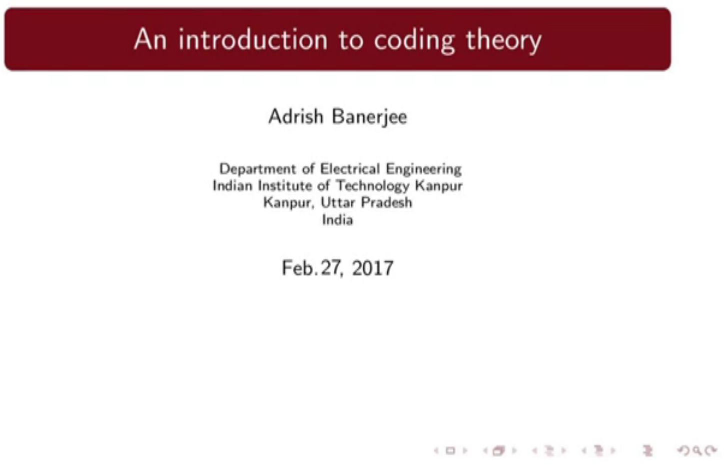
An Introduction to Coding Theory
Professor Adrish Banerji
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
Module 06
Lecture Number 25
Decoding of low density parity check codes-I

(Refer Slide Time 00:14)`



An introduction to coding theory

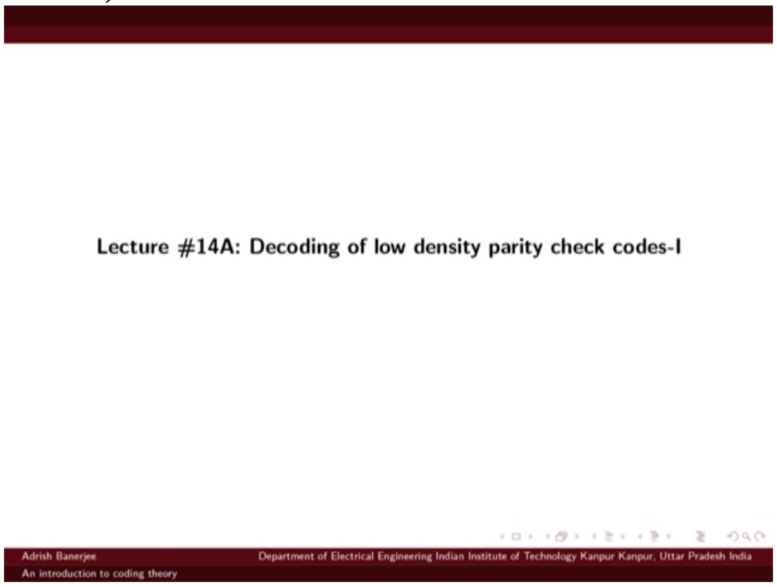
Adrish Banerjee

Department of Electrical Engineering
Indian Institute of Technology Kanpur
Kanpur, Uttar Pradesh
India

Feb. 27, 2017

Today we are going to discuss

(Refer Slide Time 00:16)

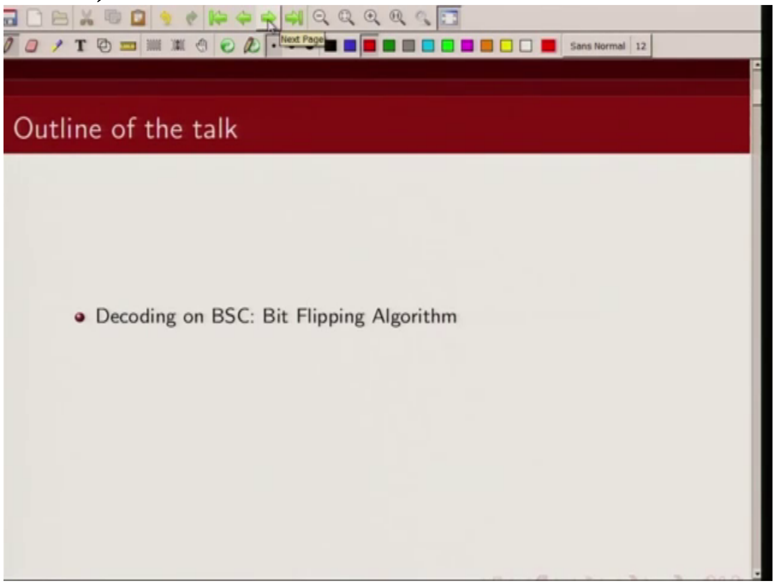


Lecture #14A: Decoding of low density parity check codes-I

Adrish Banerjee   Department of Electrical Engineering Indian Institute of Technology Kanpur Kanpur, Uttar Pradesh India
An introduction to coding theory

decoding of L D P C codes. So to start

(Refer Slide Time 00:21)

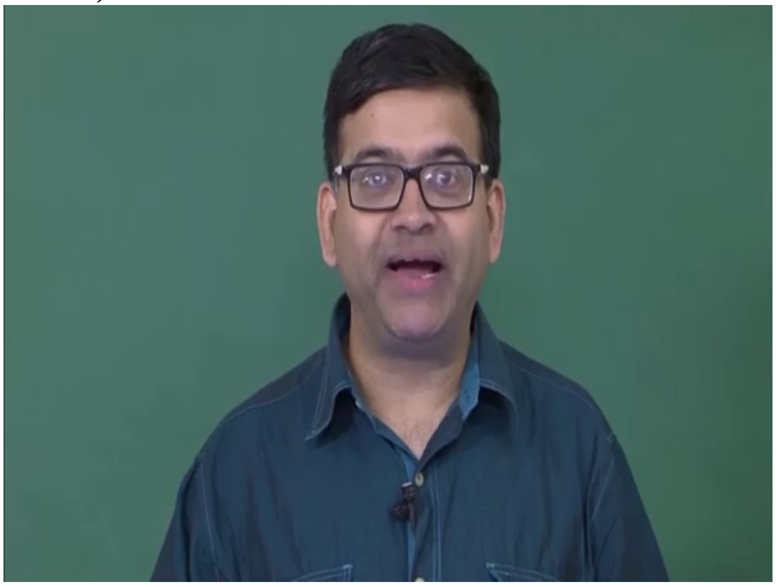


Outline of the talk

- Decoding on BSC: Bit Flipping Algorithm

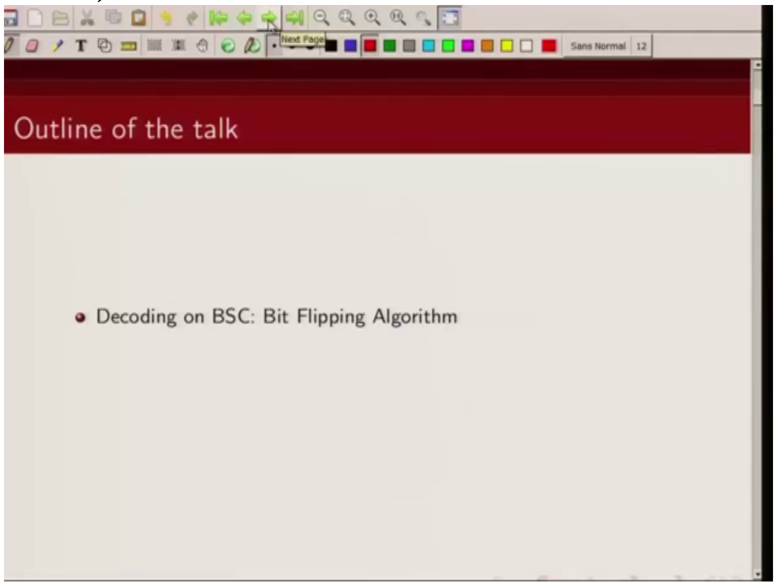
with we will first take a simple example of transmission over a binary

(Refer Slide Time 00:27)



symmetrical channel and we are going to talk about bit flipping algorithm to decode L D P C codes. And then in the next lecture we will talk about probabilistic decoding algorithm based on

(Refer Slide Time 00:40)

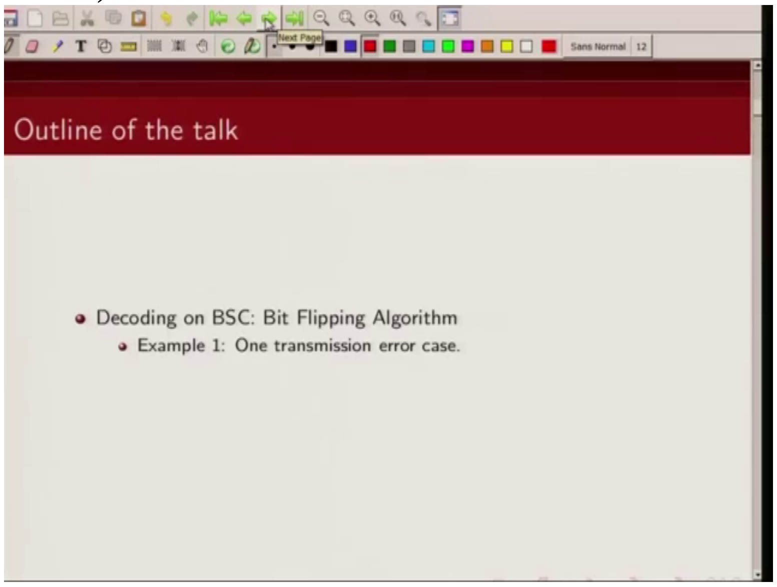


Outline of the talk

- Decoding on BSC: Bit Flipping Algorithm

belief propagation.

(Refer Slide Time 00:42)

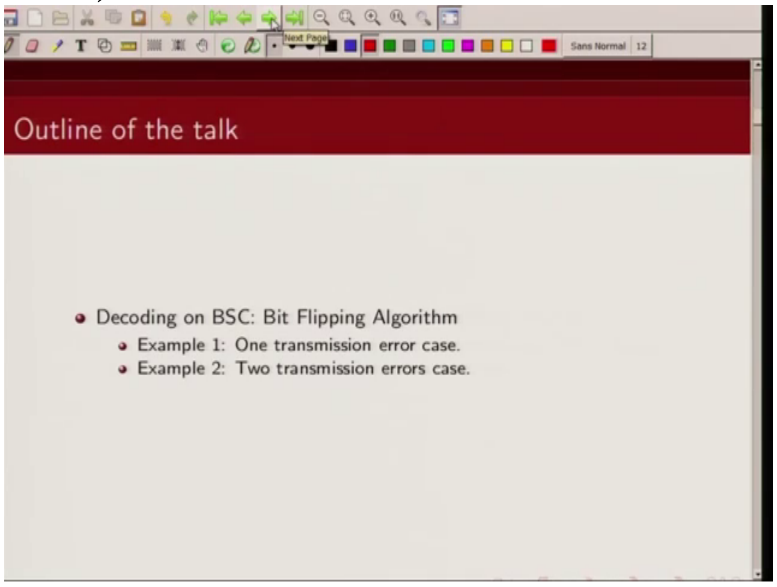


Outline of the talk

- Decoding on BSC: Bit Flipping Algorithm
  - Example 1: One transmission error case.

So we will consider two cases today. First where there is only one error that has happened and second

(Refer Slide Time 00:49)

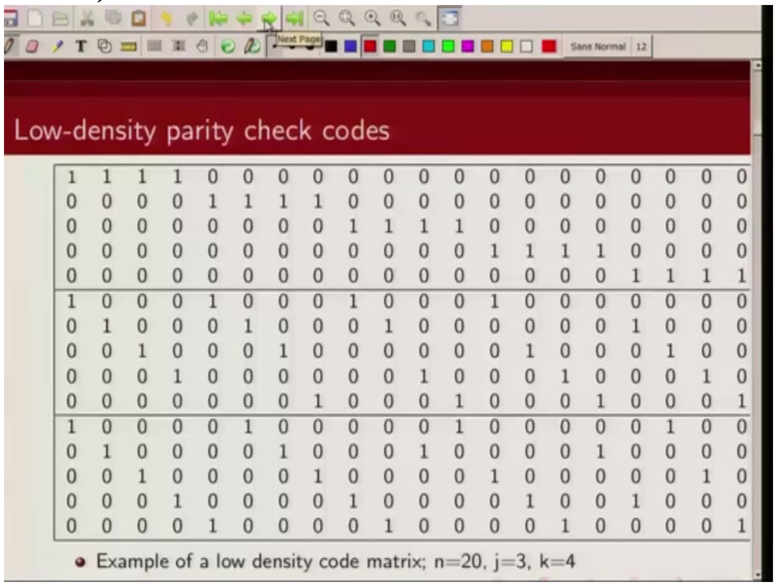


Outline of the talk

- Decoding on BSC: Bit Flipping Algorithm
    - Example 1: One transmission error case.
    - Example 2: Two transmission errors case.

where there are 2 errors have happened and we will show how to correct these errors using L D P C codes.

(Refer Slide Time 00:57)



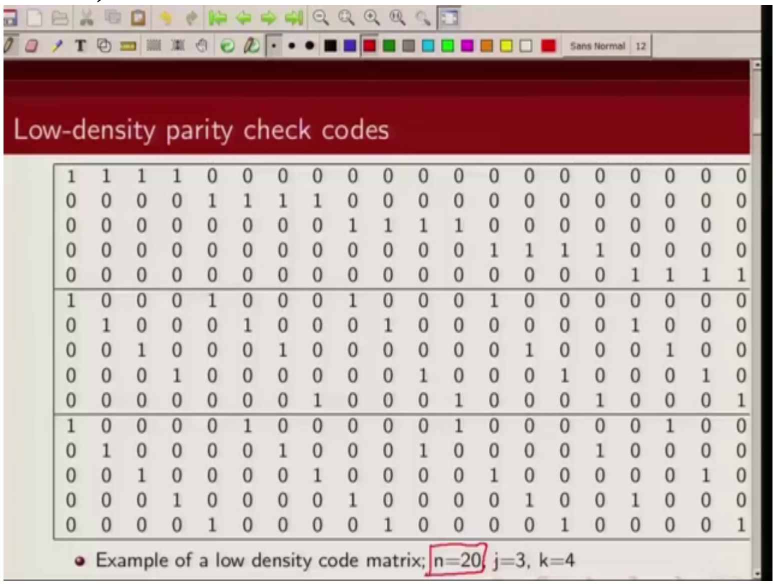
Low-density parity check codes

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

- Example of a low density code matrix; n=20, j=3, k=4

So recall this is an example of low density parity check code of block length 20, the column weight

(Refer Slide Time 01:08)



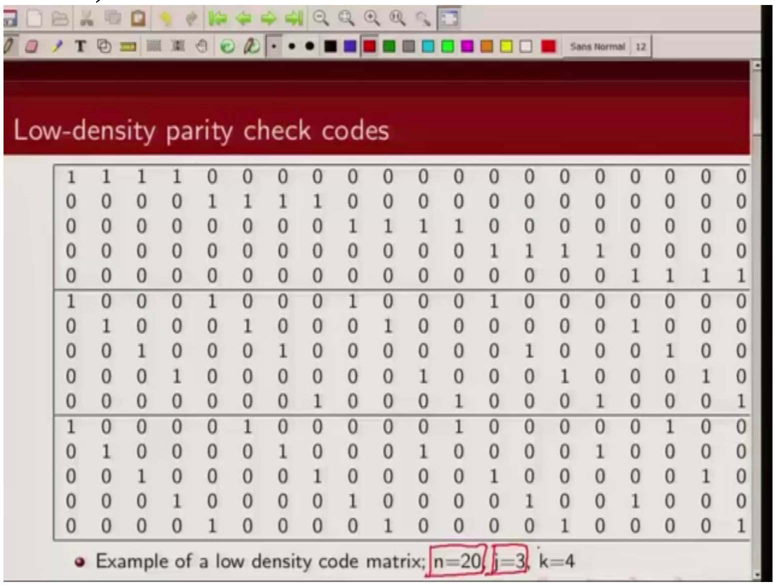
Low-density parity check codes

$$\left[\begin{array}{cccccccccccccccccccc}
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
\hline
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0\\
0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\
\hline
1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\
0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0\\
0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&1&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1
\end{array}\right]$$

- Example of a low density code matrix; $\boxed{n=20}$ j=3, k=4

is 3

(Refer Slide Time 01:11)



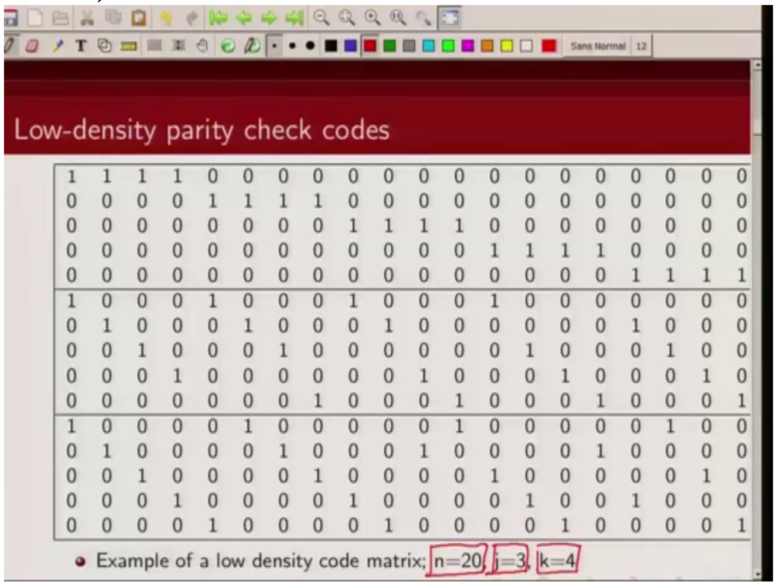
Low-density parity check codes

$$\left[\begin{array}{cccccccccccccccccccc}
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
\hline
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0\\
0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\
\hline
1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\
0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0\\
0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&1&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1
\end{array}\right]$$

- Example of a low density code matrix; $\boxed{n=20}$ $\boxed{j=3}$ k=4

and

(Refer Slide Time 01:12)



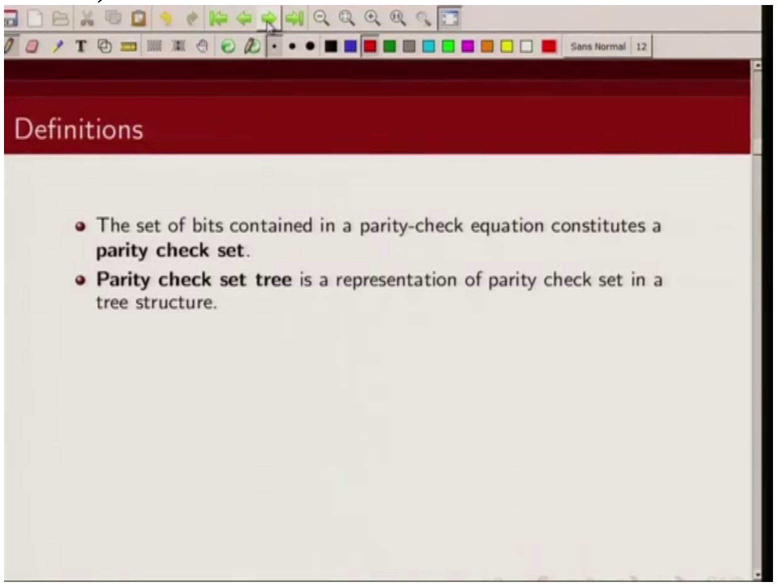
Low-density parity check codes

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

- Example of a low density code matrix; $\boxed{n=20}$, $\boxed{j=3}$, $\boxed{k=4}$

row weight is 4.

(Refer Slide Time 01:17)



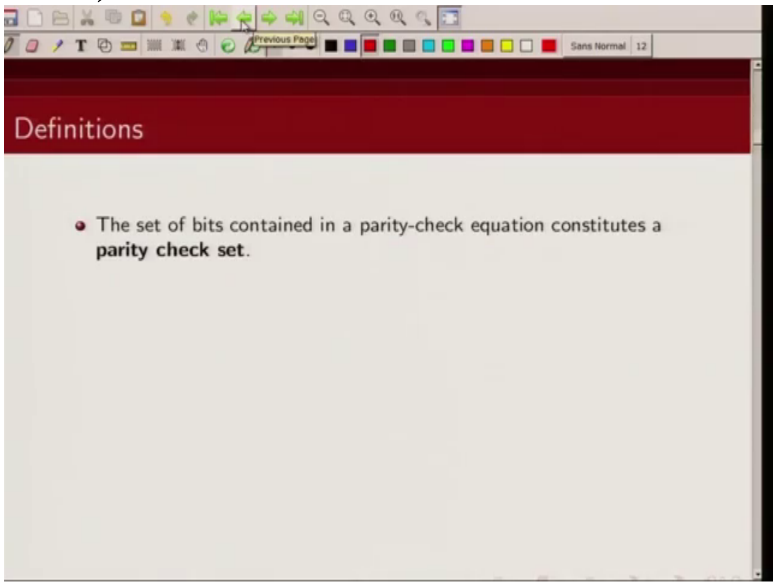
Definitions

- The set of bits contained in a parity-check equation constitutes a **parity check set**.
- **Parity check set tree** is a representation of parity check set in a tree structure.

We will first define a few terms and then we will come to the decoding of that.

So first thing we will define what is a parity check set. So what is a parity check set? It is the set of bits that are participating in the parity check equation. So set of bits that participate in a parity check equation, they constitute a parity check set. So for example if you

(Refer Slide Time 01:48)

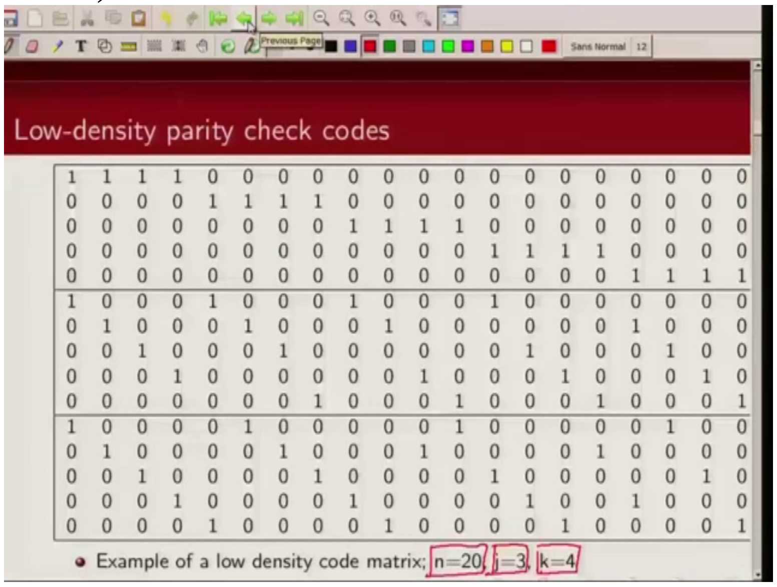


Definitions

- The set of bits contained in a parity-check equation constitutes a **parity check set**.

look at

(Refer Slide Time 01:49)



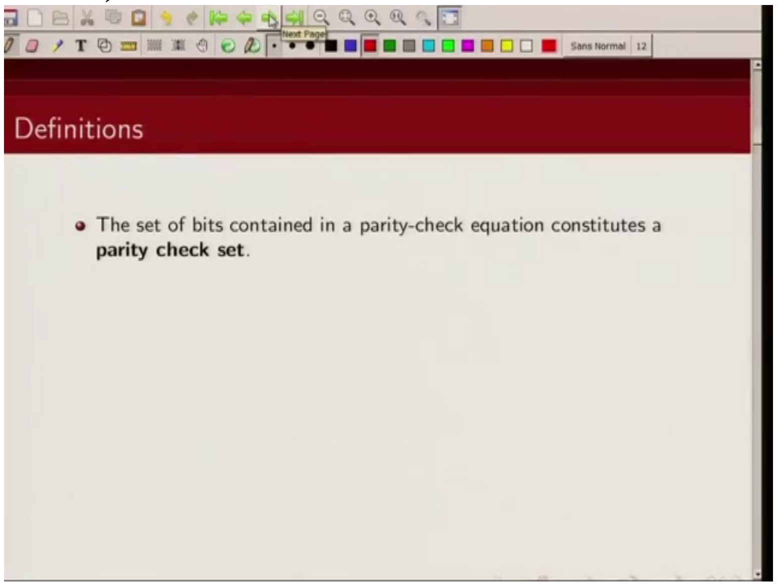
Low-density parity check codes

$$\begin{array}{|cccccccccccccccccccc|}
\hline
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
\hline
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0\\
0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\
\hline
1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\
0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0\\
0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&1&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1\\
\hline
\end{array}$$

- Example of a low density code matrix; $\boxed{n=20}$, $\boxed{j=3}$, $\boxed{k=4}$

this particular parity check equation, now these are the bits that are participating in this parity check equation. So these bits will form a parity check set. If we look for example at this particular row, now this bit, this bit, this bit and this bit these are the 4 bits that are participating in the parity check equation. So these bits will form a parity check set.

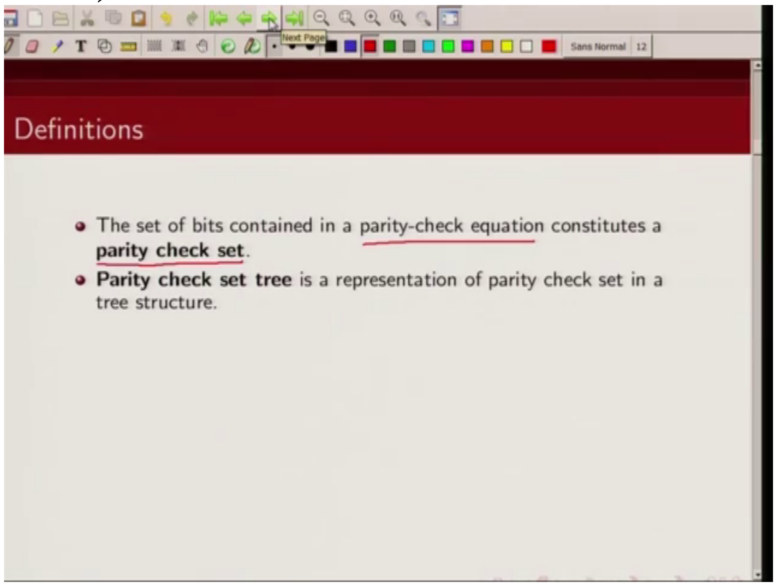
(Refer Slide Time 02:26)



Definitions

- The set of bits contained in a parity-check equation constitutes a **parity check set**.

(Refer Slide Time 02:28)

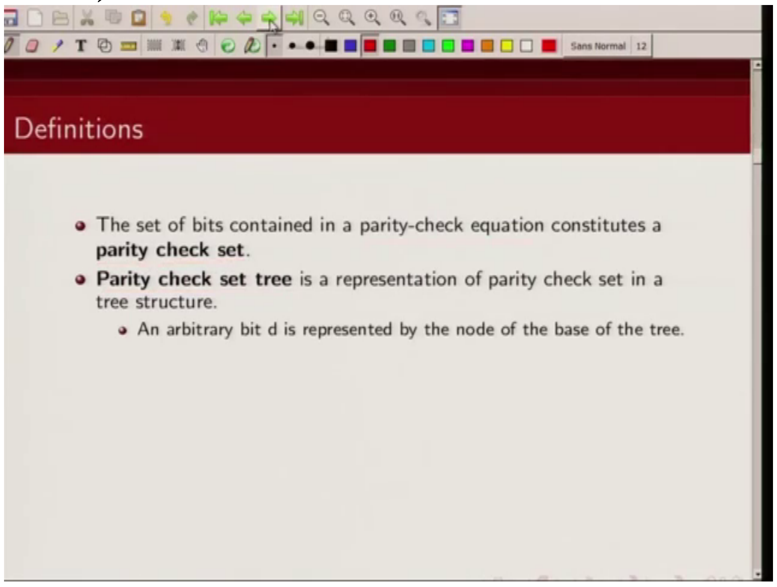


Definitions

- The set of bits contained in a parity-check equation constitutes a **parity check set**.
- **Parity check set tree** is a representation of parity check set in a tree structure.

So what is a parity check set tree? It is a graphical represent of a parity check set in a tree like structure. How?

(Refer Slide Time 02:39)

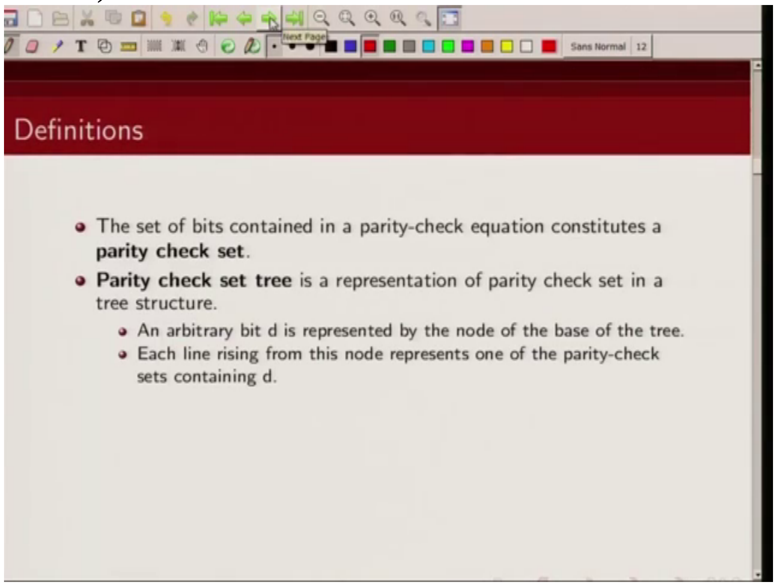


Definitions

- The set of bits contained in a parity-check equation constitutes a **parity check set**.
- **Parity check set tree** is a representation of parity check set in a tree structure.
  - An arbitrary bit d is represented by the node of the base of the tree.

We will explain. So any arbitrary bit is represented as node of the base of the tree.

(Refer Slide Time 02:50)



Definitions

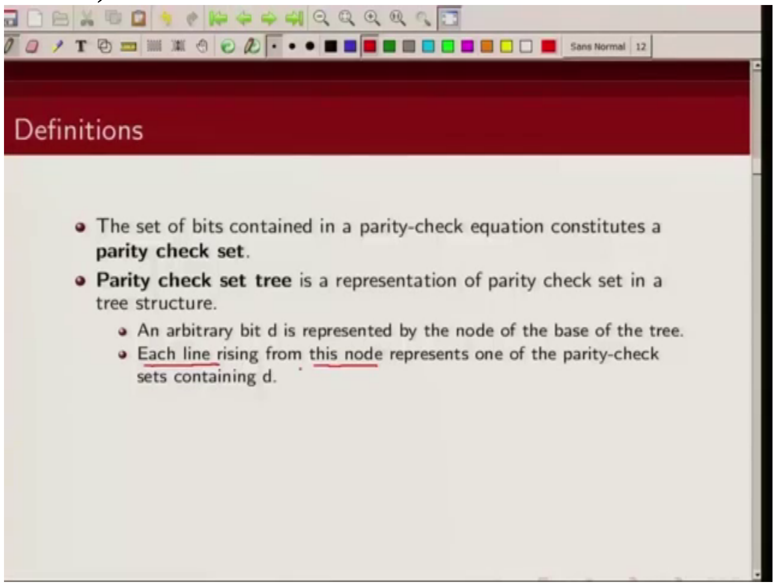
- The set of bits contained in a parity-check equation constitutes a **parity check set**.
- **Parity check set tree** is a representation of parity check set in a tree structure.
  - An arbitrary bit d is represented by the node of the base of the tree.
  - Each line rising from this node represents one of the parity-check sets containing d.

There is a line arising from this node and each of these line represent one parity check equation where

(Refer Slide Time 03:01)

this particular bit is participating. So each line

(Refer Slide Time 03:05)

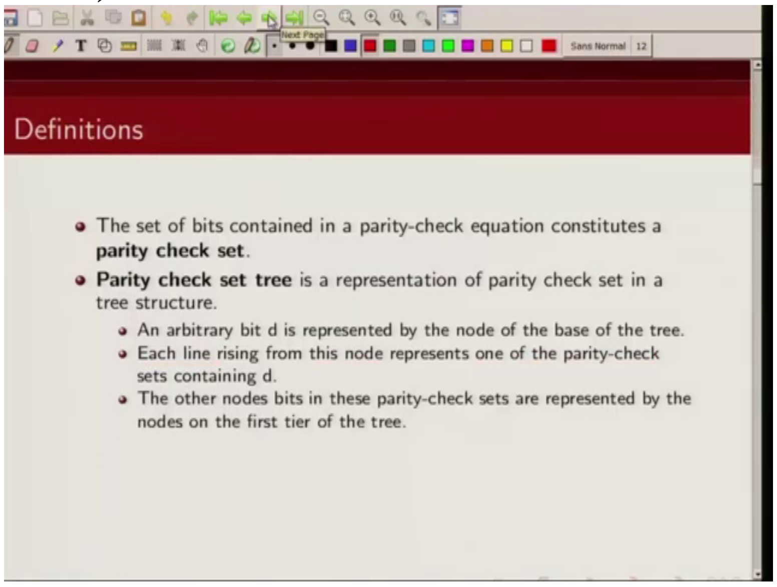


Definitions

- The set of bits contained in a parity-check equation constitutes a **parity check set**.
- **Parity check set tree** is a representation of parity check set in a tree structure.
  - An arbitrary bit d is represented by the node of the base of the tree.
  - Each line rising from this node represents one of the parity-check sets containing d.

arises from the node and it represents one of the parity check equations or one of the parity check sets where this particular node is participating.

(Refer Slide Time 03:18)

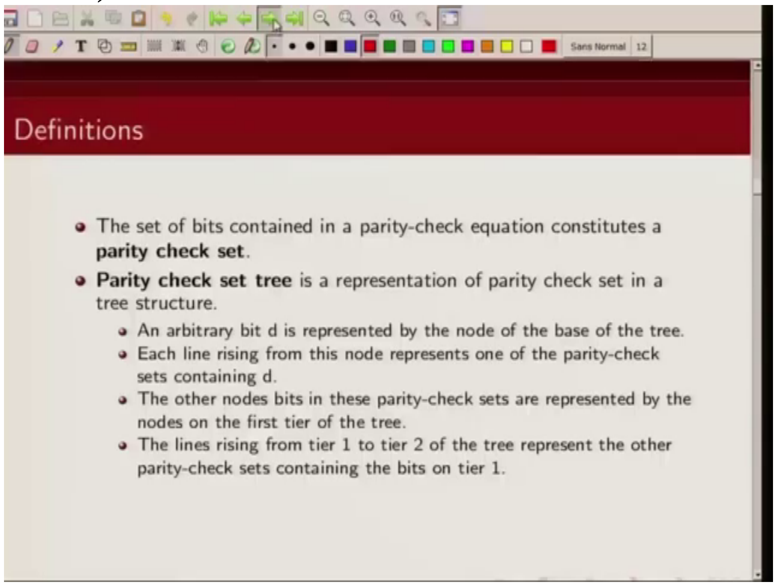


Definitions

- The set of bits contained in a parity-check equation constitutes a **parity check set**.
- **Parity check set tree** is a representation of parity check set in a tree structure.
  - An arbitrary bit d is represented by the node of the base of the tree.
  - Each line rising from this node represents one of the parity-check sets containing d.
  - The other nodes bits in these parity-check sets are represented by the nodes on the first tier of the tree.

Now other nodes in this parity check constraints are represented as nodes in the first tier of the tree. Now what do I mean by this?

(Refer Slide Time 03:29)

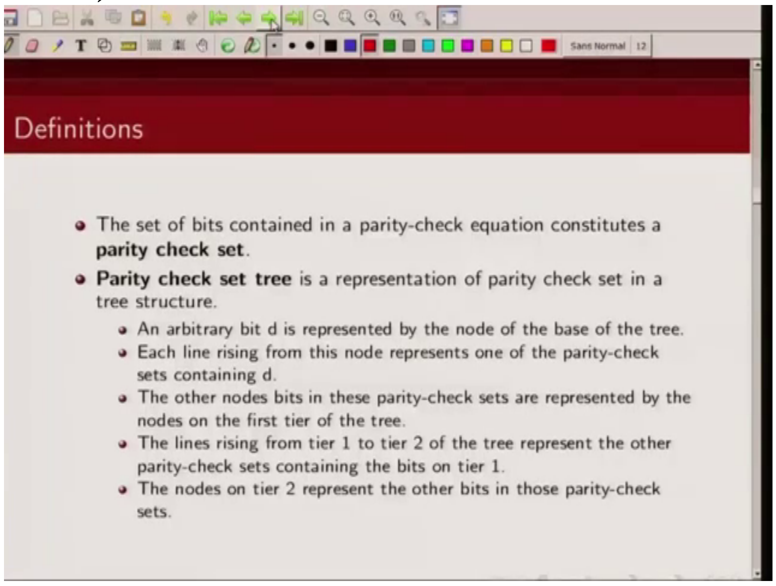


Definitions

- The set of bits contained in a parity-check equation constitutes a **parity check set**.
- **Parity check set tree** is a representation of parity check set in a tree structure.
  - An arbitrary bit d is represented by the node of the base of the tree.
  - Each line rising from this node represents one of the parity-check sets containing d.
  - The other nodes bits in these parity-check sets are represented by the nodes on the first tier of the tree.
  - The lines rising from tier 1 to tier 2 of the tree represent the other parity-check sets containing the bits on tier 1.

Let's just

(Refer Slide Time 03:30)

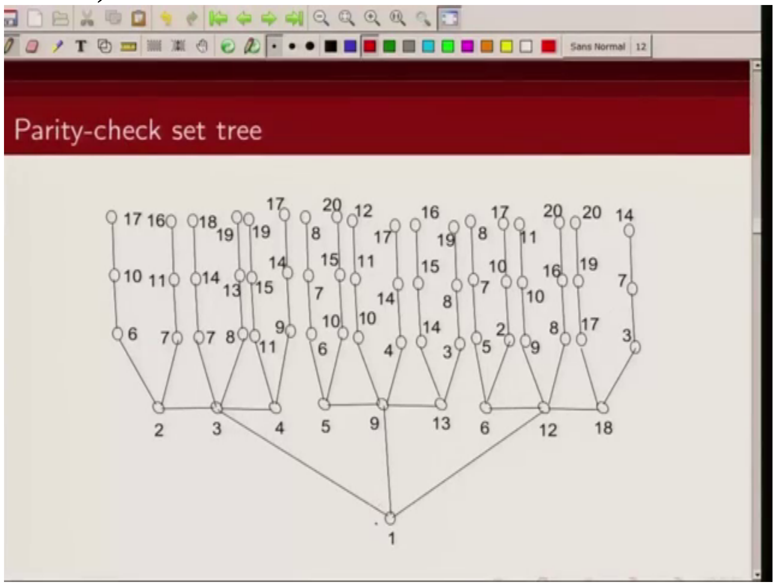


Definitions

- The set of bits contained in a parity-check equation constitutes a **parity check set**.
- **Parity check set tree** is a representation of parity check set in a tree structure.
  - An arbitrary bit d is represented by the node of the base of the tree.
  - Each line rising from this node represents one of the parity-check sets containing d.
  - The other nodes bits in these parity-check sets are represented by the nodes on the first tier of the tree.
  - The lines rising from tier 1 to tier 2 of the tree represent the other parity-check sets containing the bits on tier 1.
  - The nodes on tier 2 represent the other bits in those parity-check sets.

look at, so let's say I

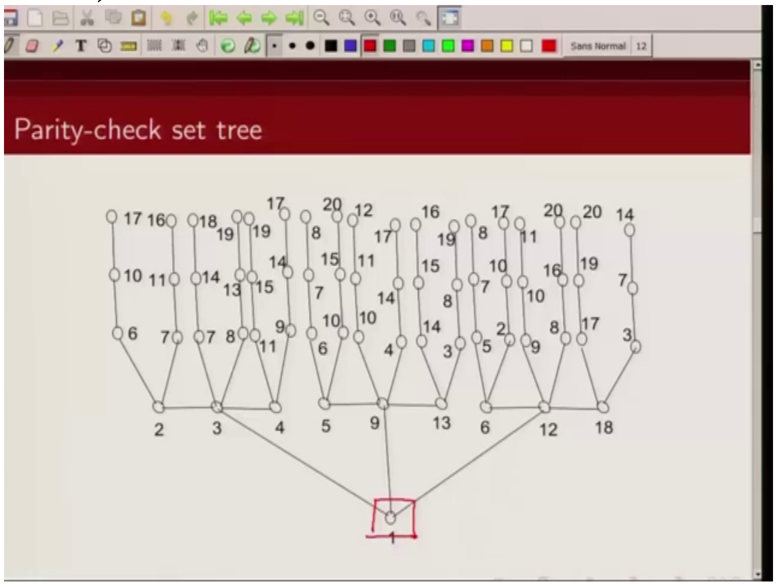
(Refer Slide Time 03:33)



Parity-check set tree

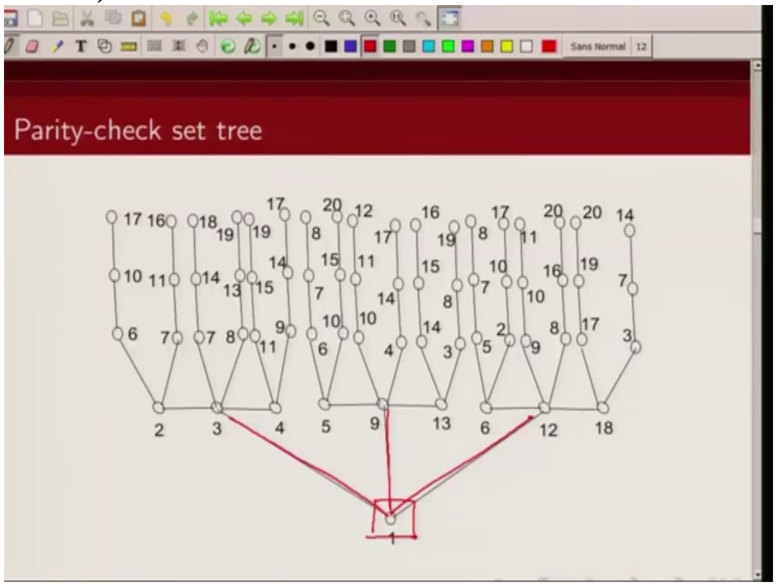
have this node, first node

(Refer Slide Time 03:35)


Parity-check set tree

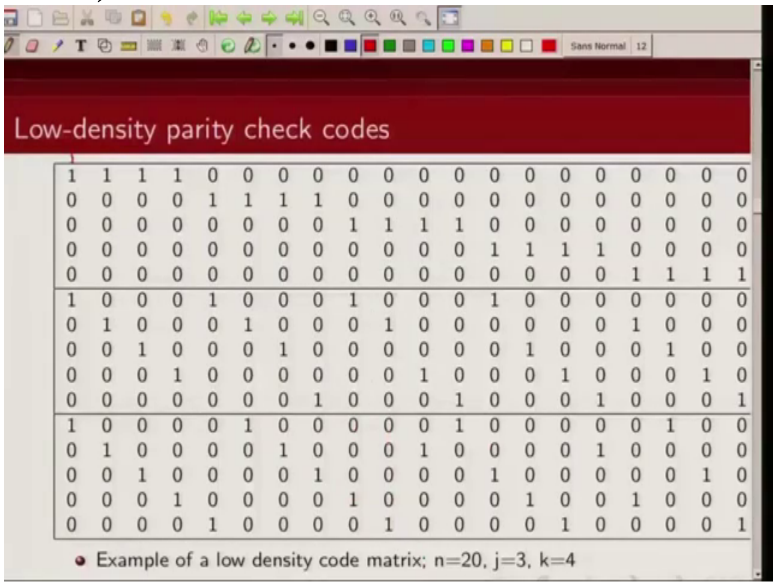
calling it node 1. Now this node participates in 3 parity check equations. You can see 1, 2, 3

(Refer Slide Time 03:48)


Parity-check set tree

go back to our, so we are looking at

(Refer Slide Time 03:53)



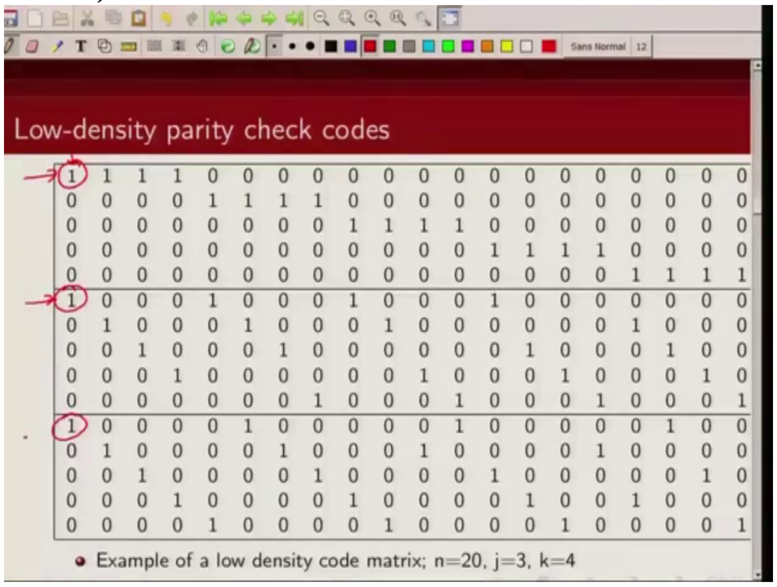
Low-density parity check codes

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

- Example of a low density code matrix; n=20, j=3, k=4

first bit. It participates in this parity check equation, this parity check equation and

(Refer Slide Time 04:01)



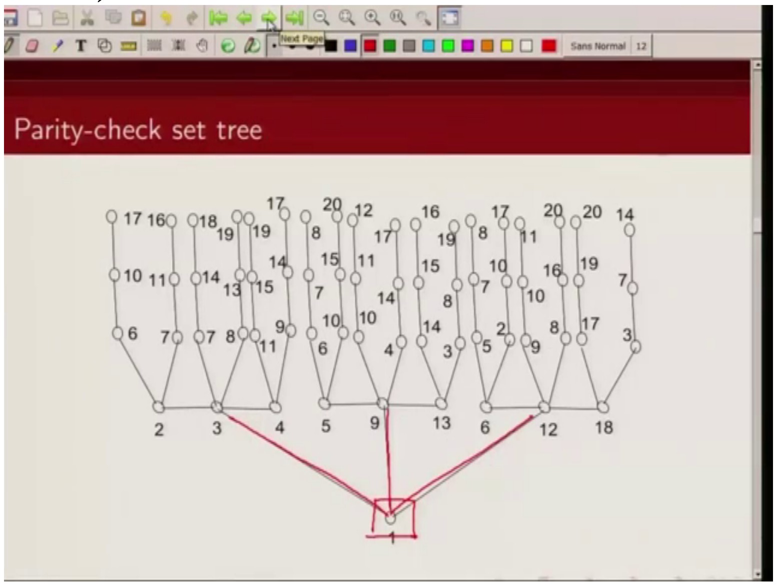
Low-density parity check codes

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

- Example of a low density code matrix; n=20, j=3, k=4

this parity check equation. So there is one line corresponding

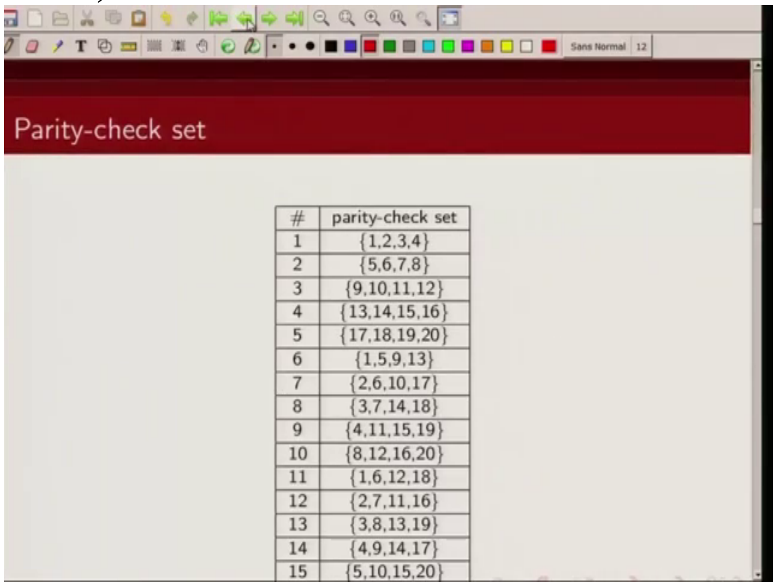
(Refer Slide Time 04:07)



Parity-check set tree

to each of these parity check equations. Ok, now in this parity check equation, you can see

(Refer Slide Time 04:17)

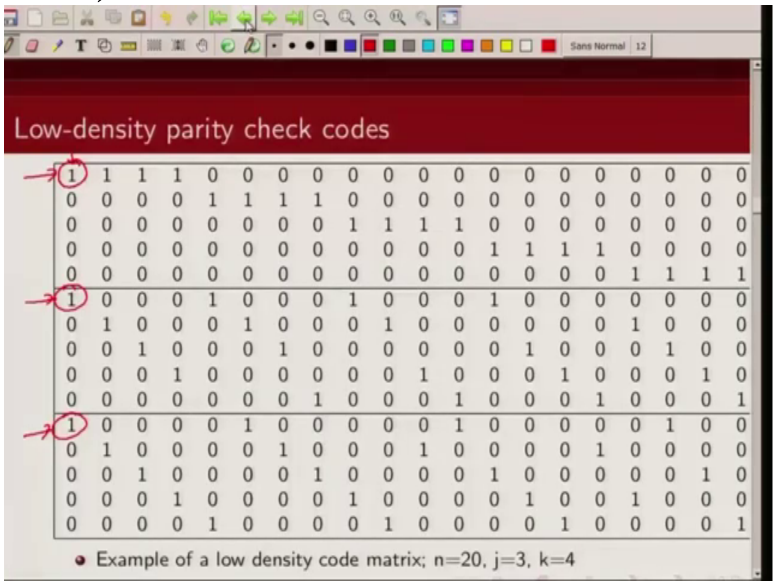


Parity-check set

#	parity-check set
1	{1,2,3,4}
2	{5,6,7,8}
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{1,5,9,13}
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{1,6,12,18}
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

which are the other bits participating?

(Refer Slide Time 04:19)



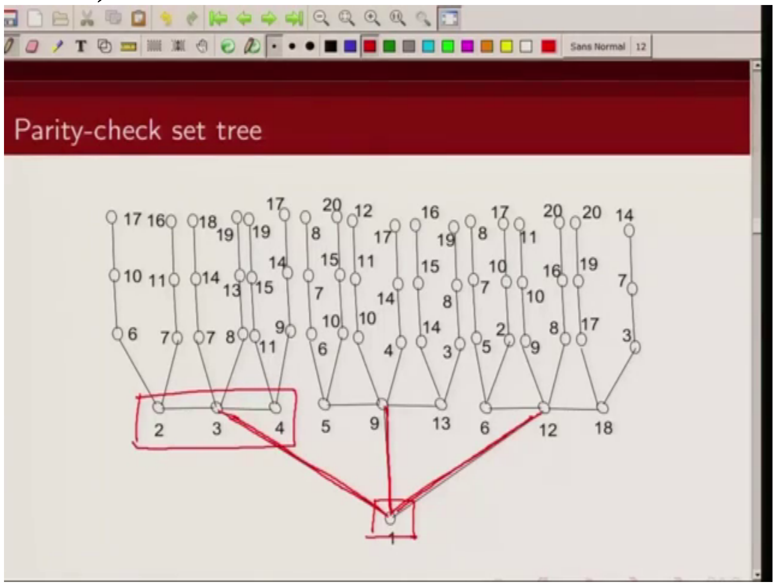
Low-density parity check codes

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

- Example of a low density code matrix; n=20, j=3, k=4

So bit number 2, bit number 3, bit number 4, so how did we write that? So the other bits that are participating in the parity check constraint, they are written like this.

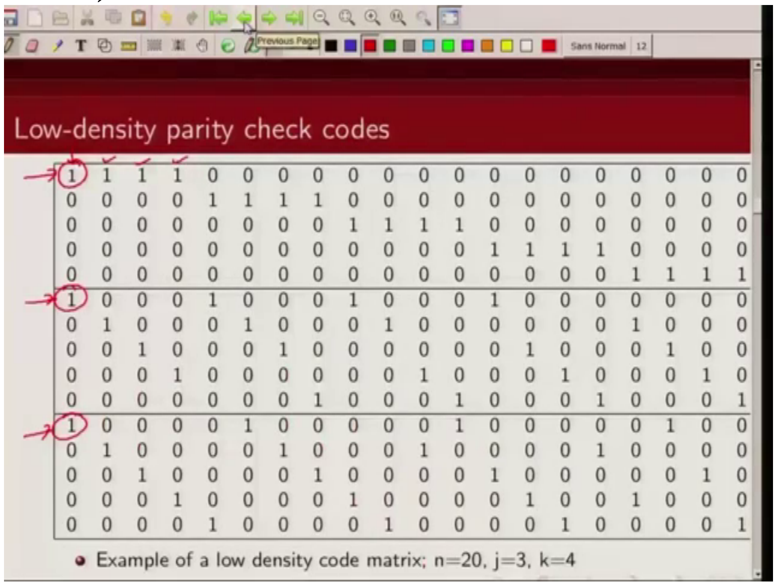
(Refer Slide Time 04:32)



Parity-check set tree

So 1, this is one parity check constraint, and 2, 3, 4 bits are participating. Similarly if you look at here,

(Refer Slide Time 04:43)



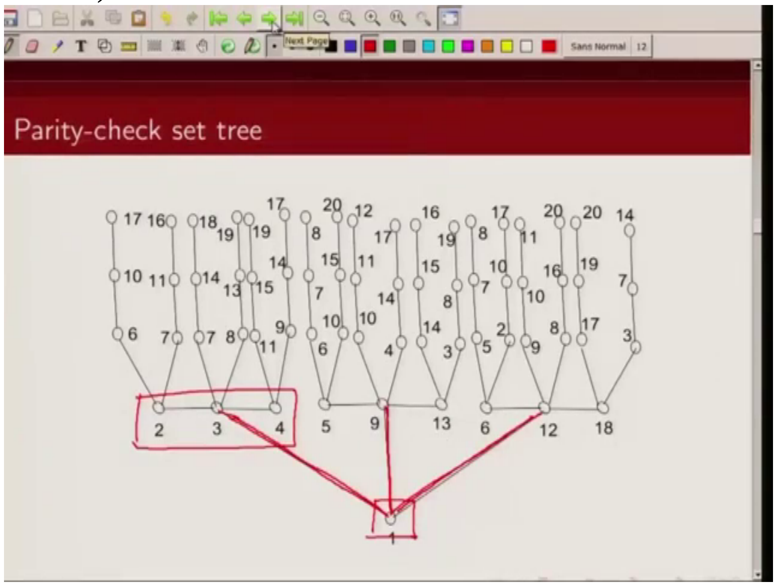
Low-density parity check codes

$$
\begin{array}{|cccccccccccccccccccc|}
\hline
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\hline
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\hline
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
\hline
\end{array}
$$

- Example of a low density code matrix; n=20, j=3, k=4

this is bit number 5, 9 and 13 are participating in this particular

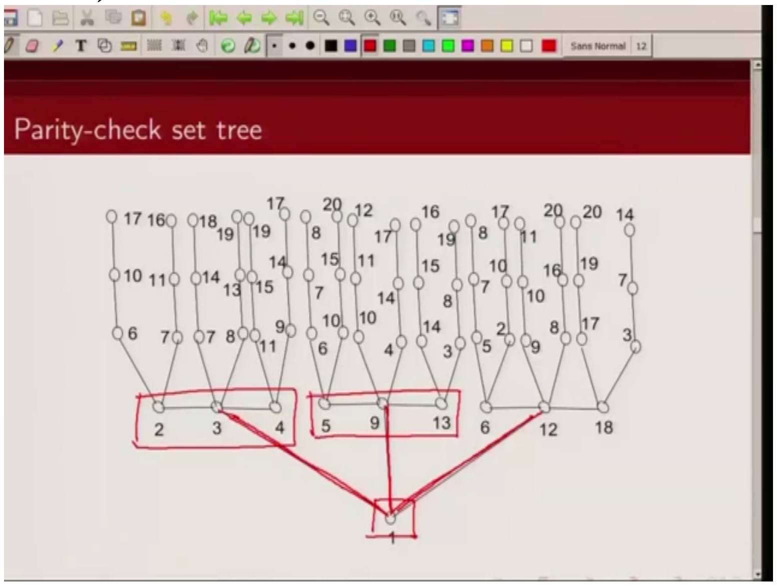
(Refer Slide Time 04:49)



Parity-check set tree

parity check equation, so that is represented by this. So that's what I

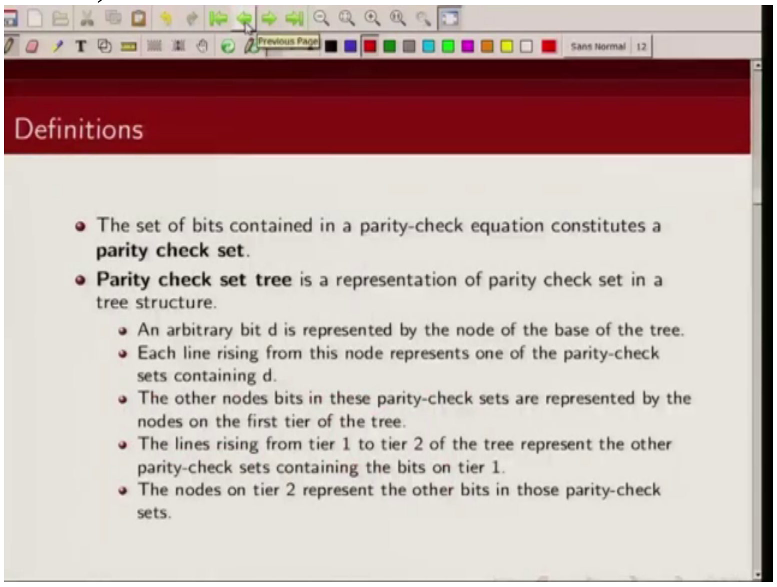
(Refer Slide Time 04:55)



Parity-check set tree

mean when I said

(Refer Slide Time 04:57)

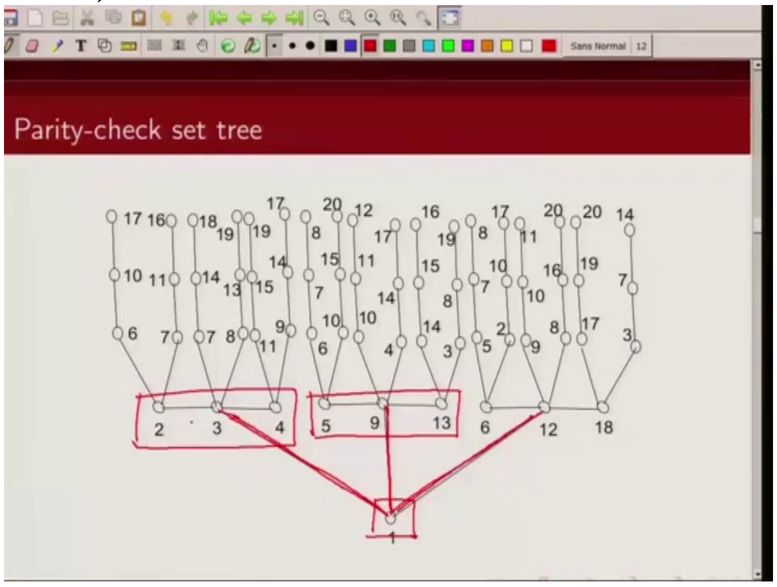


Definitions

- The set of bits contained in a parity-check equation constitutes a **parity check set**.
- **Parity check set tree** is a representation of parity check set in a tree structure.
  - An arbitrary bit d is represented by the node of the base of the tree.
  - Each line rising from this node represents one of the parity-check sets containing d.
  - The other nodes bits in these parity-check sets are represented by the nodes on the first tier of the tree.
  - The lines rising from tier 1 to tier 2 of the tree represent the other parity-check sets containing the bits on tier 1.
  - The nodes on tier 2 represent the other bits in those parity-check sets.

other nodes are represented as nodes in the first tier. Now line arises from tier 1 to tier 2 represent the other parity check constraints containing bits from tier 1. So this is my tier 0, this is tier 1.

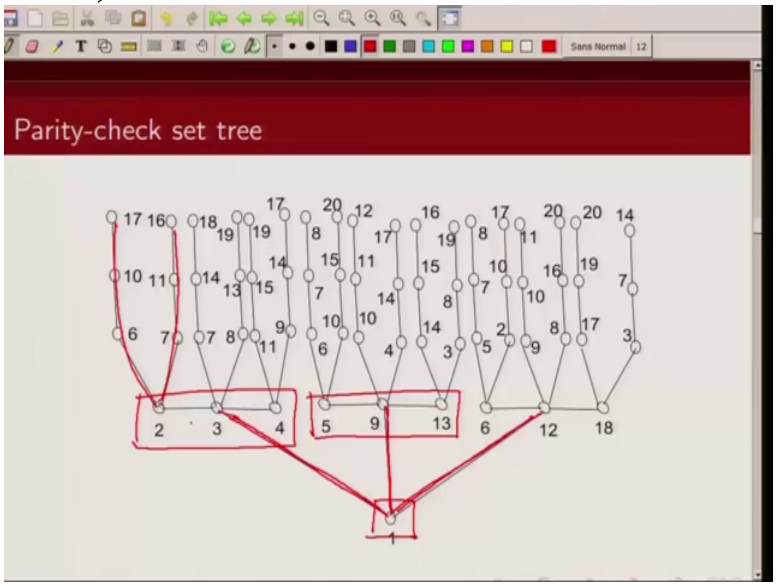
(Refer Slide Time 05:18)



Parity-check set tree

Now what is a, what are connections coming here? These are

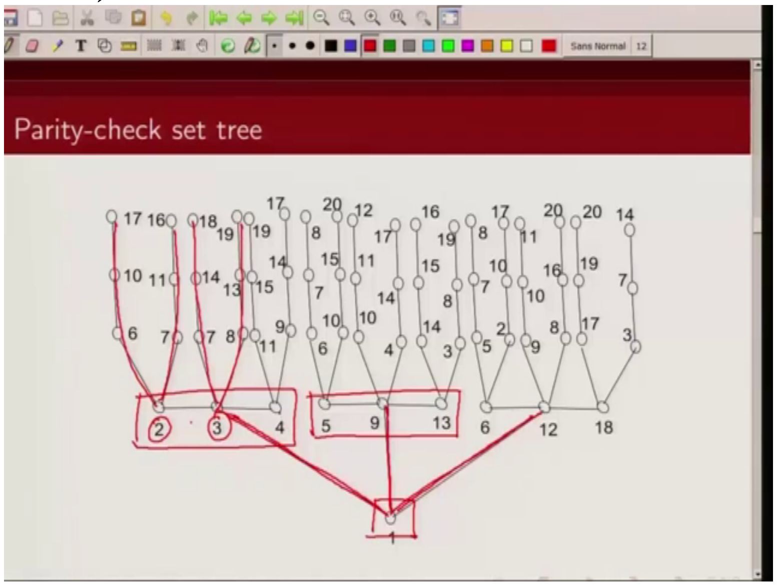
(Refer Slide Time 05:22)



Parity-check set tree

the parity check constraints involving these bits, involving 2, involving 3 is here, involving 4,

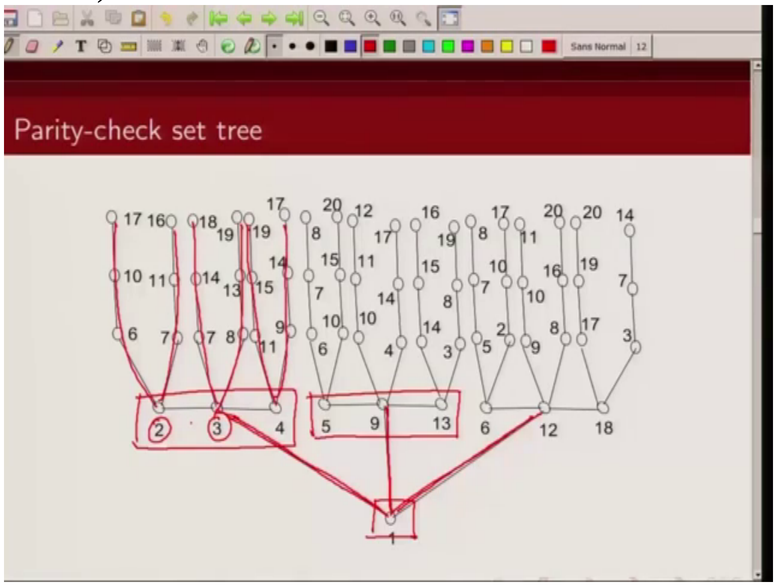
(Refer Slide Time 05:33)



Parity-check set tree

these are the parity check constraints,

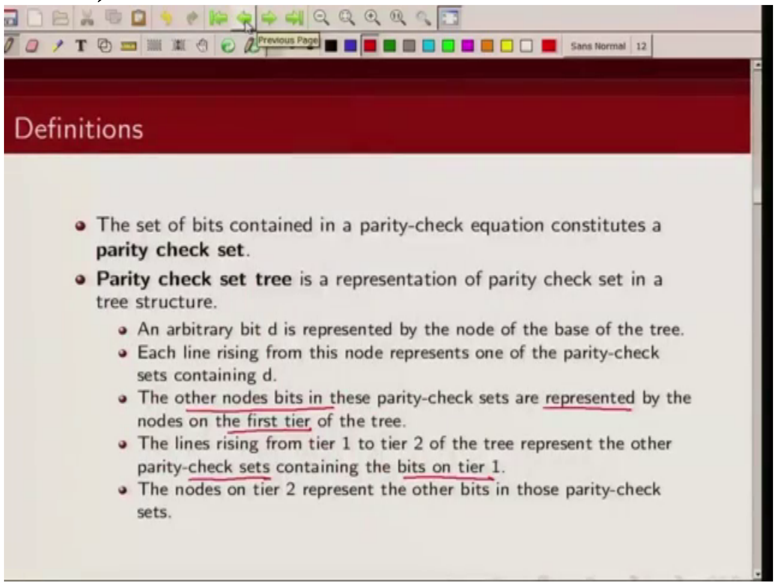
(Refer Slide Time 05:35)



Parity-check set tree

Ok. So this is how I am drawing

(Refer Slide Time 05:41)

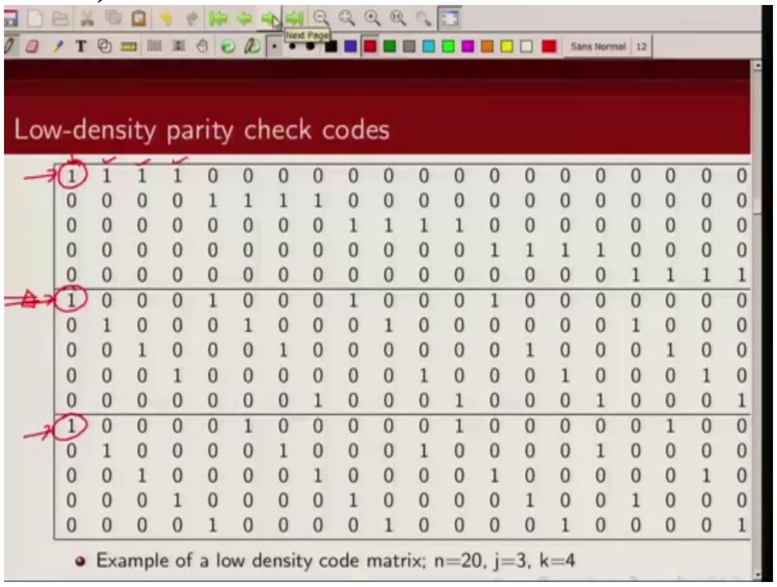


Definitions

- The set of bits contained in a parity-check equation constitutes a **parity check set**.
- **Parity check set tree** is a representation of parity check set in a tree structure.
  - An arbitrary bit d is represented by the node of the base of the tree.
  - Each line rising from this node represents one of the parity-check sets containing d.
  - The other nodes bits in these parity-check sets are represented by the nodes on the first tier of the tree.
  - The lines rising from tier 1 to tier 2 of the tree represent the other parity-check sets containing the bits on tier 1.
  - The nodes on tier 2 represent the other bits in those parity-check sets.

my parity check set tree. So again

(Refer Slide Time 05:46)



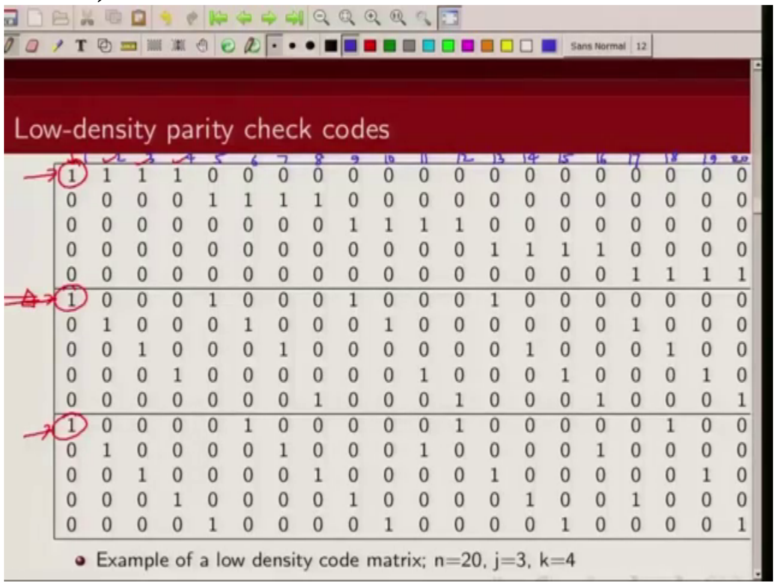
Low-density parity check codes

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

- Example of a low density code matrix; n=20, j=3, k=4

pay attention to this parity check matrix. Let's label each of them like let's say 1, 2, 3, 4, 5, 6, 7. Let's just label these columns. So that way it will be easier for us to refer to them.

(Refer Slide Time 06:12)

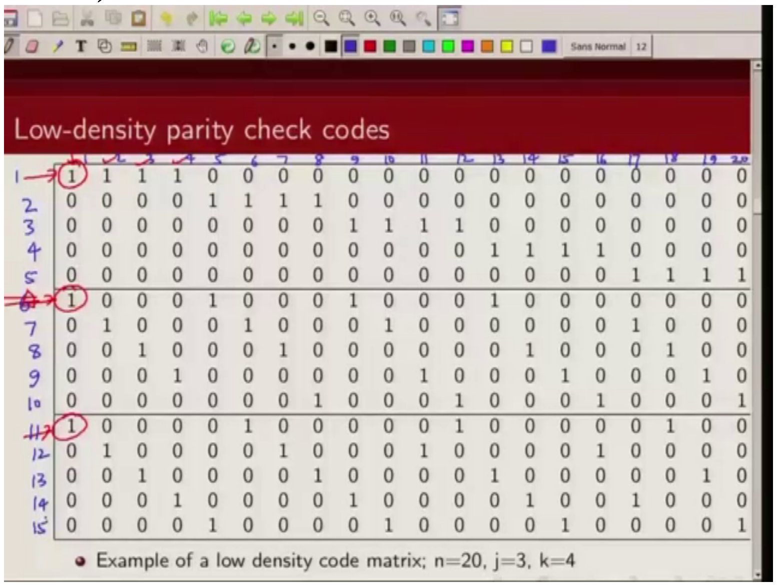


Low-density parity check codes

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

- Example of a low density code matrix; n=20, j=3, k=4

(Refer Slide Time 06:27)



Low-density parity check codes

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
6	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
7	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
8	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
9	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
10	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
11	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
12	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
13	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
14	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
15	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

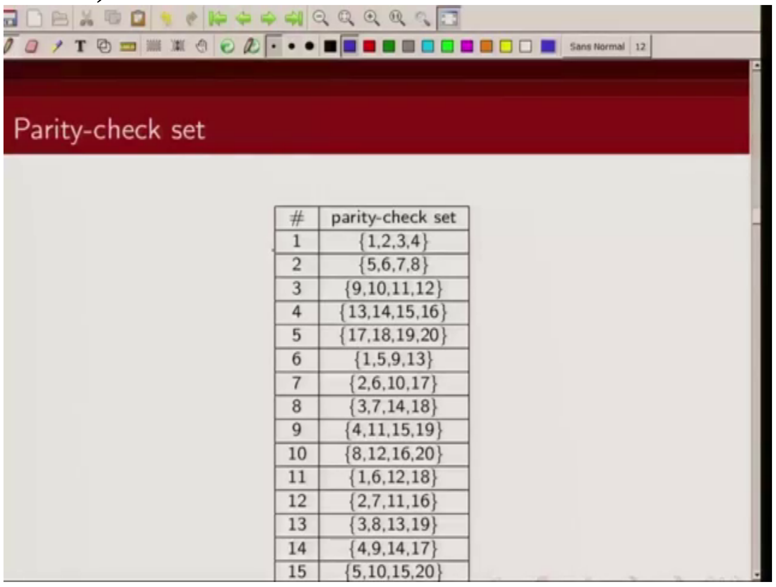
- Example of a low density code matrix; n=20, j=3, k=4

Similarly I am labeling these rows. So you can see there will be 15 parity check sets, each corresponding to each of the rows, Ok.

(Refer Slide Time 06:39)

So let us look at the parity check set. So let's first

(Refer Slide Time 06:43)

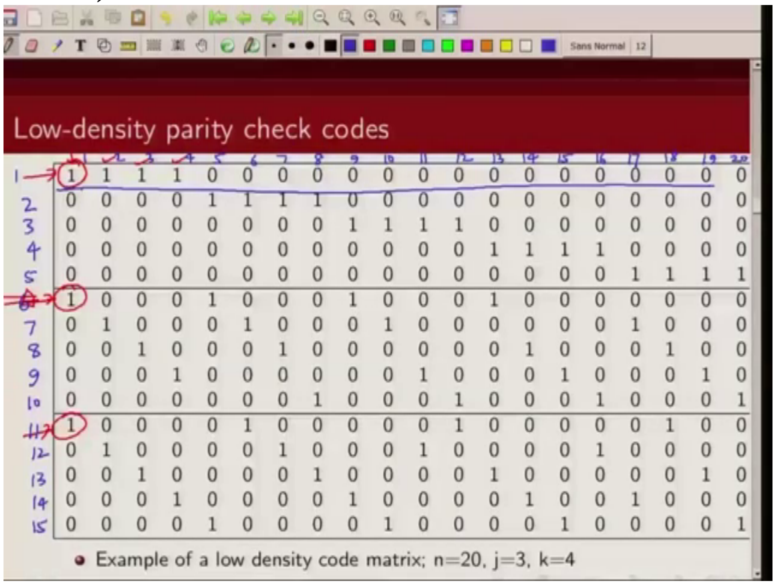


Parity-check set

#	parity-check set
1	{1,2,3,4}
2	{5,6,7,8}
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{1,5,9,13}
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{1,6,12,18}
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

let us look at this first parity check set which corresponds to this

(Refer Slide Time 06:50)



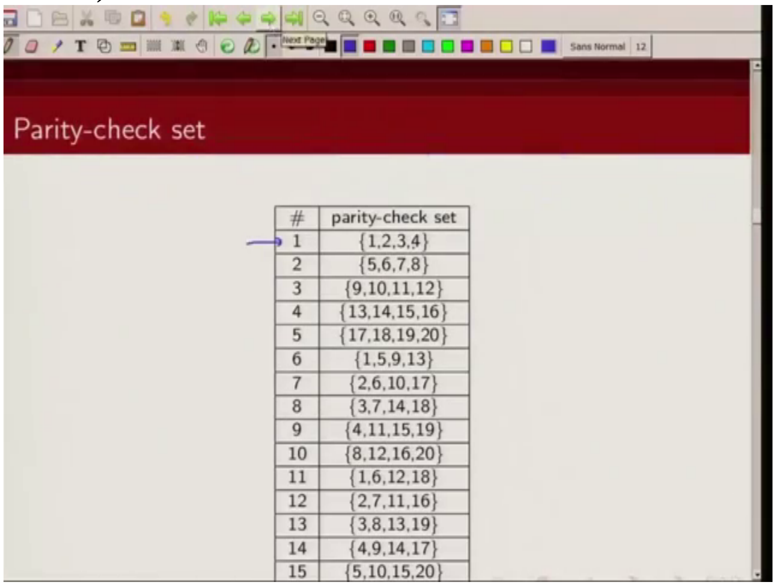
Low-density parity check codes

$$\begin{array}{c|cccccccccccccccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 \\
\hline
1 \rightarrow & \textcircled{1} & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\hline
6 \rightarrow & \textcircled{1} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
8 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
9 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\hline
11 \rightarrow & \textcircled{1} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
12 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
13 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
14 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
15 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1
\end{array}$$

- Example of a low density code matrix; n=20, j=3, k=4

first row. So note here bit number 1, 2, 3 and 4, these are participating in the parity check equation. So that's why

(Refer Slide Time 07:05)

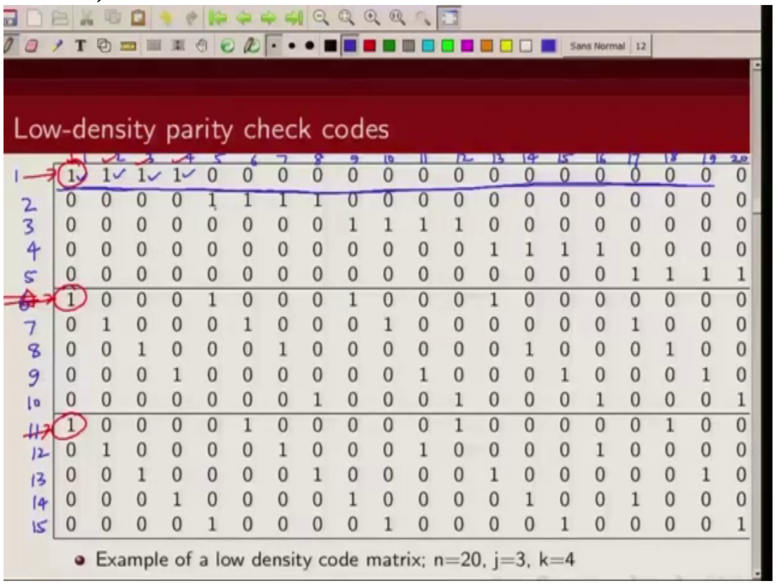


Parity-check set

#	parity-check set
1	{1,2,3,4}
2	{5,6,7,8}
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{1,5,9,13}
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{1,6,12,18}
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

this first parity check set consists of 1, 2, 3 and 4. Similarly parity check set 2, if we look at second parity check equation.

(Refer Slide Time 07:18)



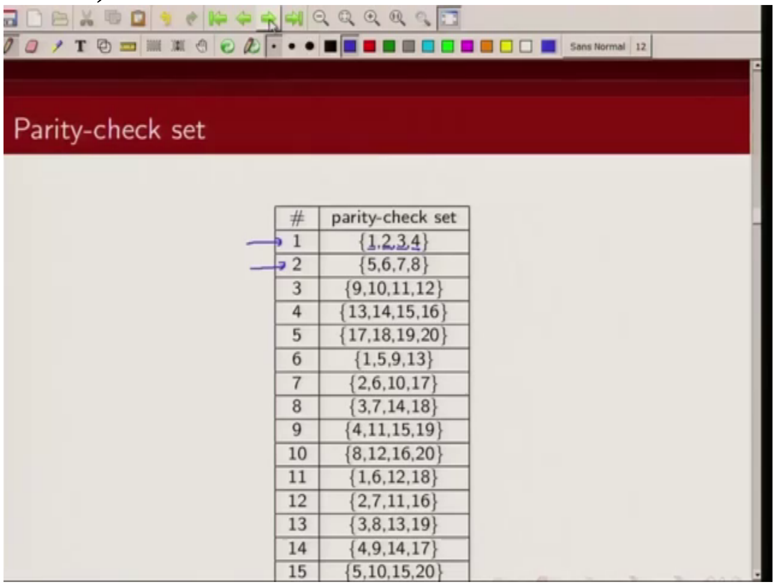
Low-density parity check codes

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1✓	1✓	1✓	1✓	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
6	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
7	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
8	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
9	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
10	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
11	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
12	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
13	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
14	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
15	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

- Example of a low density code matrix; n=20, j=3, k=4

This bit number 5, bit number 6, bit number 7, bit number 8 are participating, so then

(Refer Slide Time 07:27)

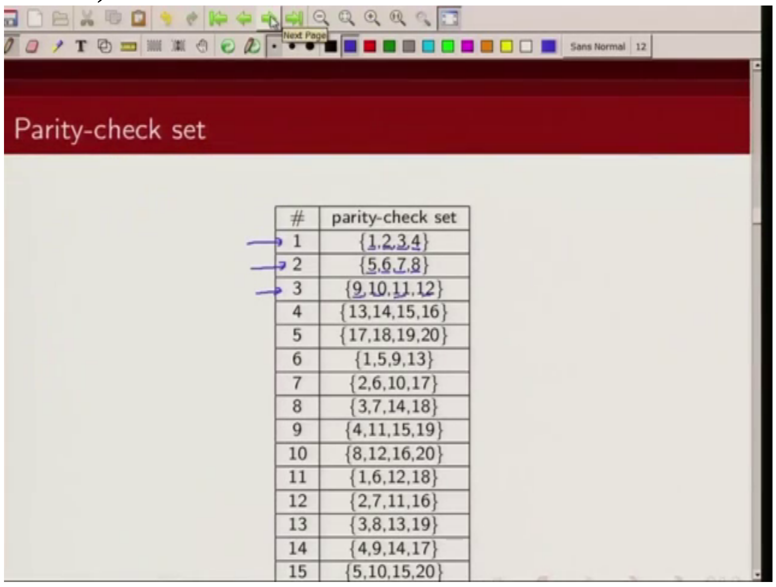


Parity-check set

#	parity-check set
1	{1,2,3,4}
2	{5,6,7,8}
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{1,5,9,13}
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{1,6,12,18}
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

parity check set will have 5, 6, 7 and 8. Similarly parity check third has 9, 10, 11, 12. So we can take any example. Let's just take this one, eighth one. Bit number 3, 7, 14, and 18; 3, 7, 14 and 18 these are participating in the parity check equation. So bit number

(Refer Slide Time 07:58)



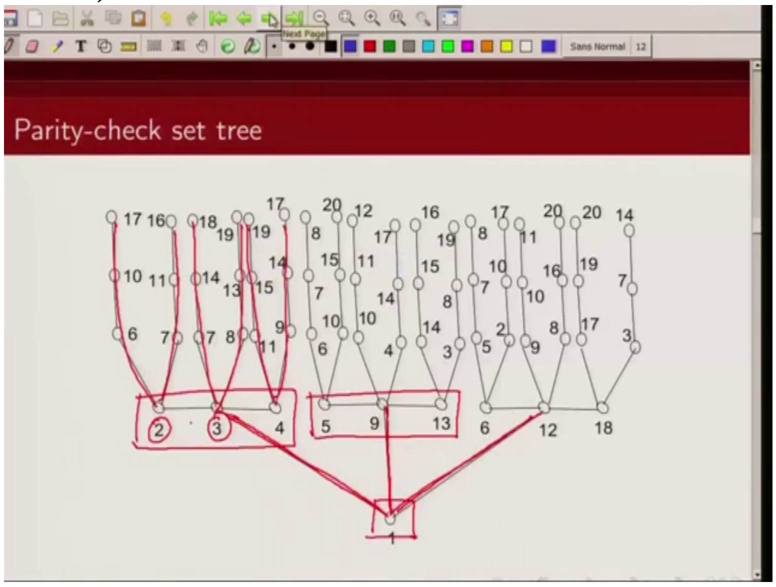
Parity-check set

#	parity-check set
1	{1,2,3,4}
2	{5,6,7,8}
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{1,5,9,13}
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{1,6,12,18}
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

3, 7, 14 and 18. So this is how for each of the parity check equations we create this parity check set. So there are 15 such parity check sets for this particular example.

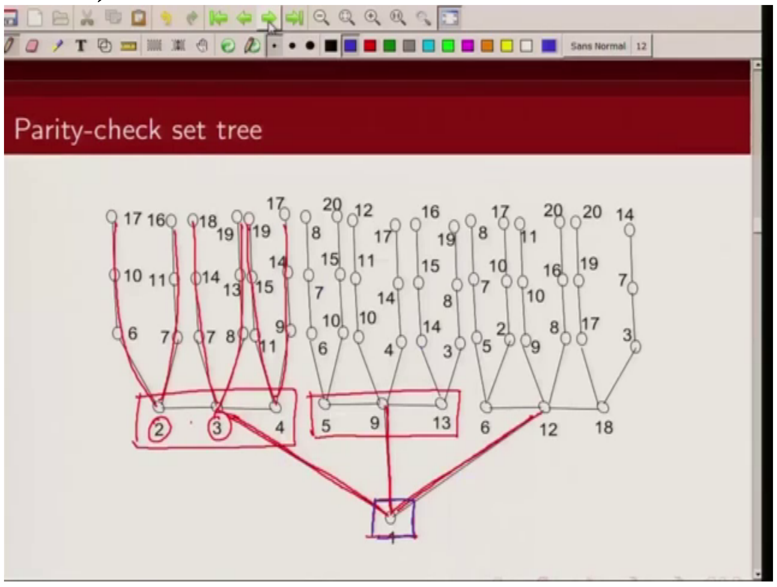
(Refer Slide Time 08:17)

Parity-check set tree



And how do we draw the parity check set tree? As I said we pick one bit. Let us say I picked number 1. Now bit number 1 appears in which parity set, how may parity check equations? Now look here bit number 1 appears in this, bit number 1 appears here, bit number 1 appears here, that's it. It appears in these 3 parity check sets. So we are going to draw 3 lines corresponding to each of these parity check sets.

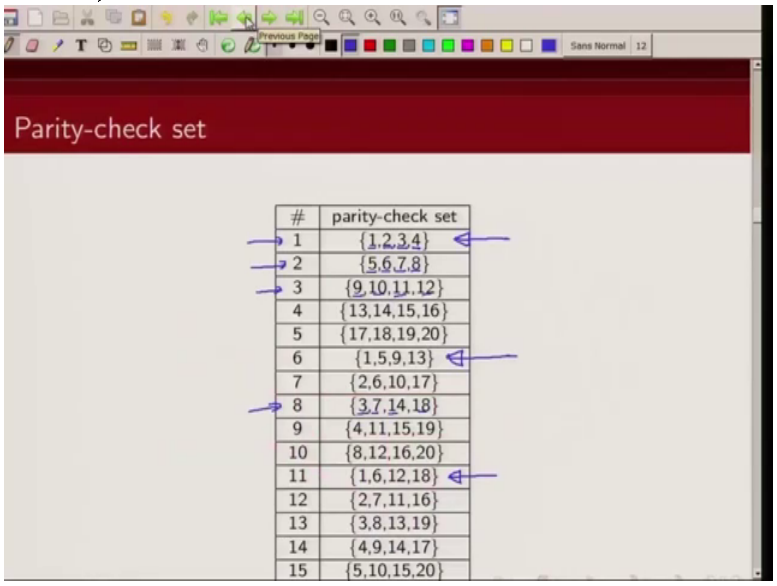
(Refer Slide Time 09:00)



Parity-check set tree

So that's what we have done. This is one line, this is another line, this is another line. Now next what we have done is we have written all the nodes that participate in the parity check set. So if you look at this one

(Refer Slide Time 09:15)

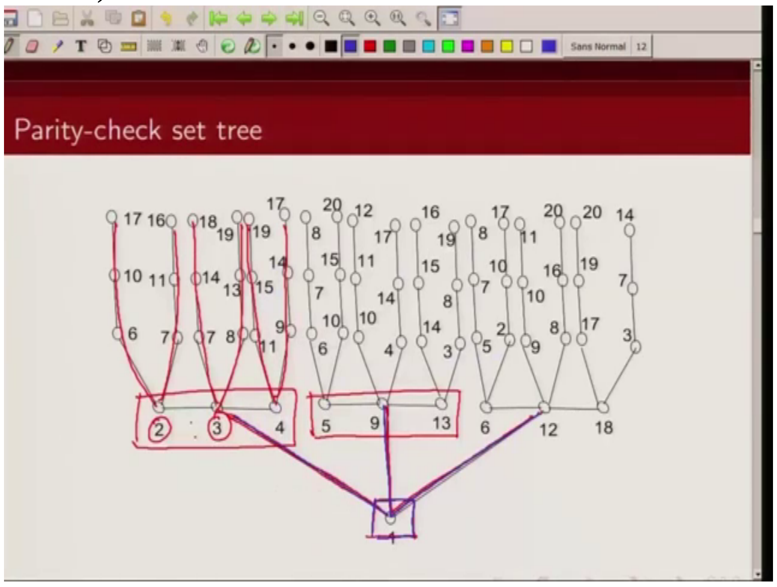


Parity-check set

#	parity-check set
1	{1,2,3,4}
2	{5,6,7,8}
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{1,5,9,13}
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{1,6,12,18}
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

in addition to 1, the other bits are 2, 3 and 4. So that we are

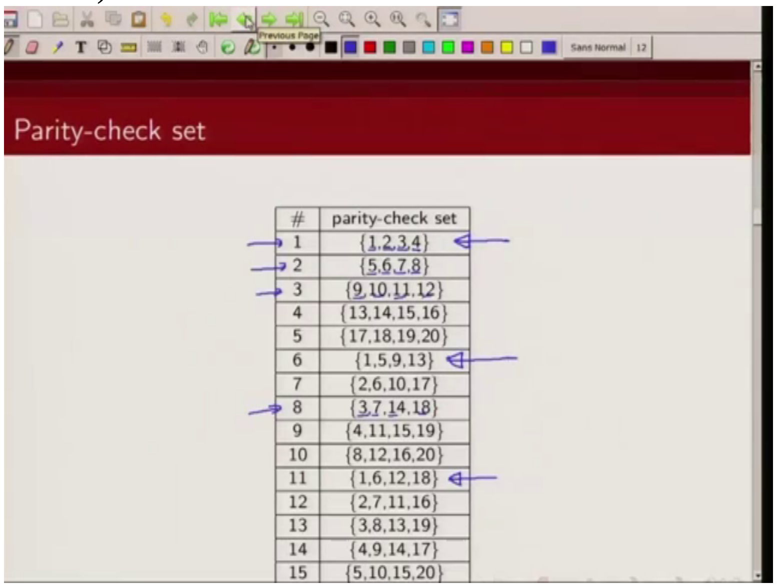
(Refer Slide Time 09:21)



Parity-check set tree

writing like this, 2, 3, and 4. Similarly

(Refer Slide Time 09:24)

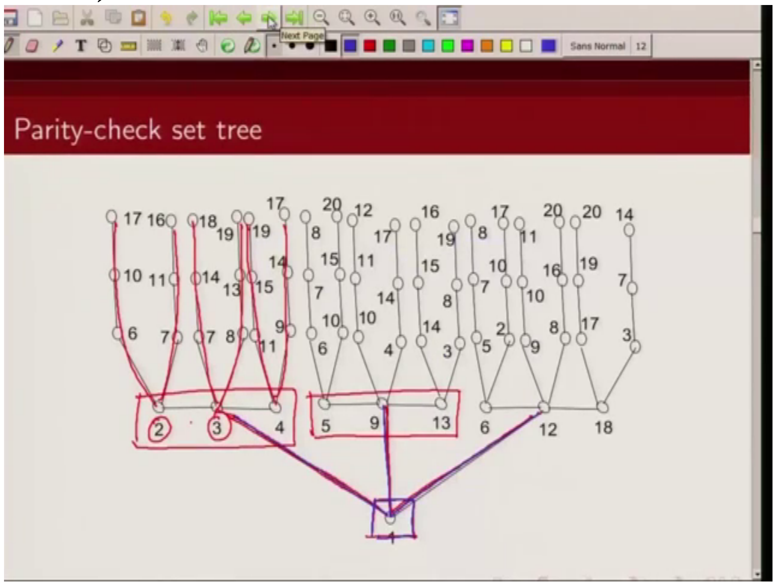


Parity-check set

#	parity-check set
1	{1,2,3,4}
2	{5,6,7,8}
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{1,5,9,13}
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{1,6,12,18}
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

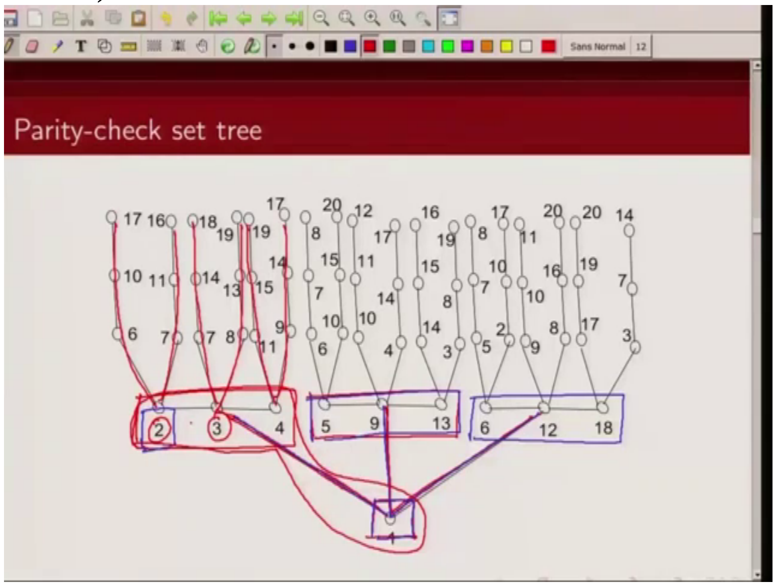
here, bit 5, 9 and 13 are participating in relation to bit number 1. So these

(Refer Slide Time 09:34)


Parity-check set tree

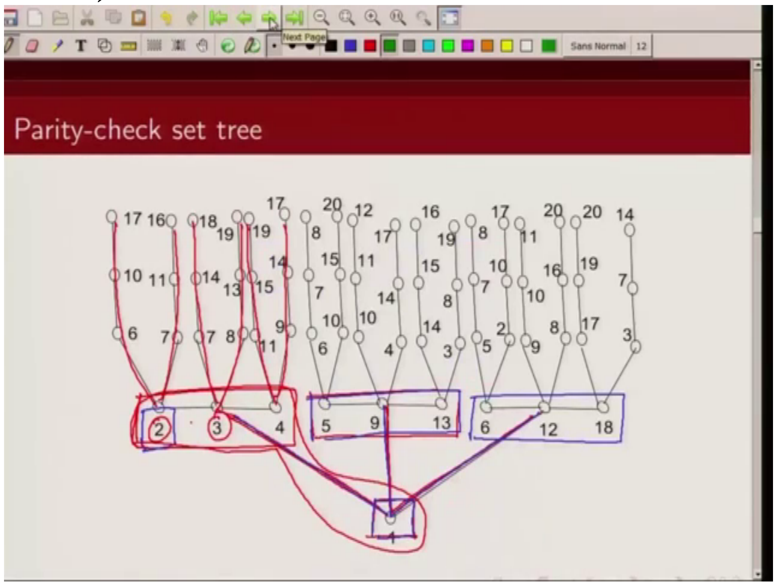
are 5, 9 and 13. And here 1, 6, 12 and 18 are participating. So then we have 6, 12 and 18. So this is our tier 1. Now how do we draw tier 2? Now ((()) this, look at 2. Now 2 appears in which, 2 appears in parity check set 1, 2 appears in parity check set 7, 2 appears in parity check set 12, right? Now this 2 appears in parity check set 1, it is already captured here. This is already captured here that 2 appears

(Refer Slide Time 10:25)


Parity-check set tree

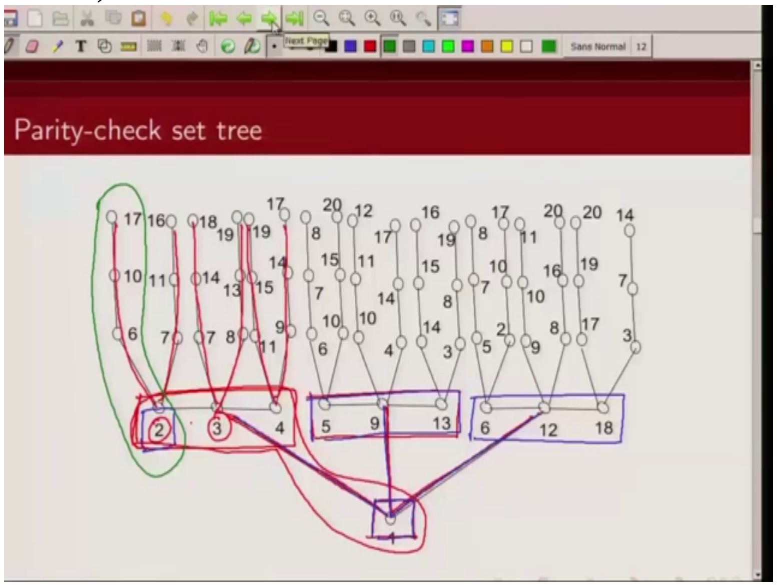
in parity check set 1. So what are the other 2 parity check sets? This is 1, is this, the other is this. So 2 appears with 6, 10 and 17. How do we show that?

(Refer Slide Time 10:38)


Parity-check set tree

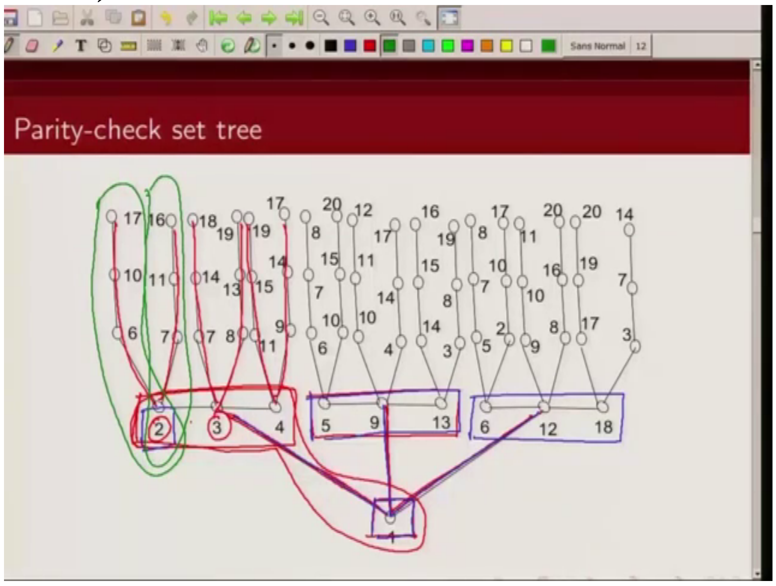
So we are showing by this particular edge. How do we show this parity check set? 2 appears with 7, 11 and 16. How do we show that?

(Refer Slide Time 10:54)


Parity-check set tree

We show that using this.

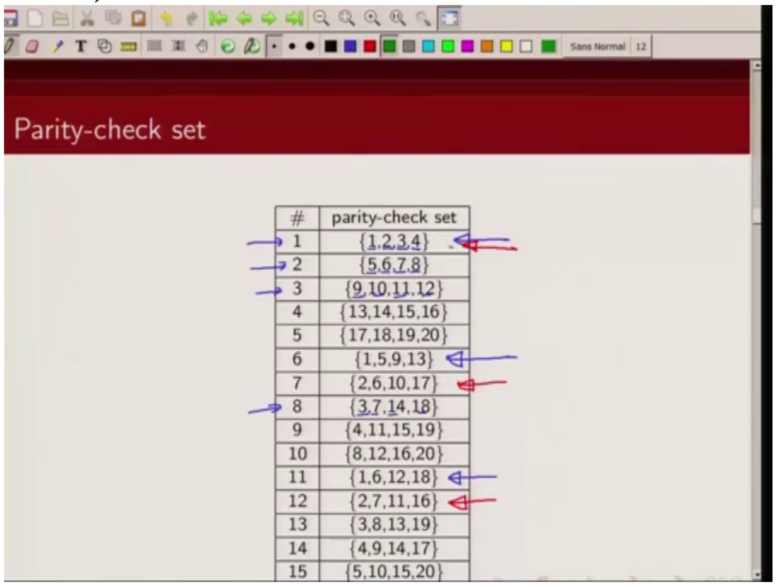
(Refer Slide Time 10:58)



Parity-check set tree

Similarly we do the same thing for other bits. So for example bit number 3; now look at bit number 3. Bit number 3 appears in

(Refer Slide Time 11:11)

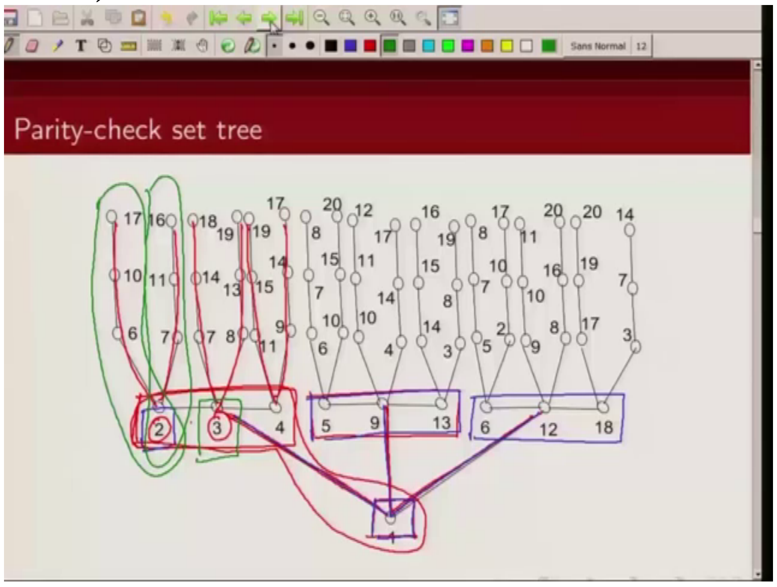


Parity-check set

#	parity-check set
1	{1,2,3,4}
2	{5,6,7,8}
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{1,5,9,13}
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{1,6,12,18}
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

parity check set 1, it appears in parity check set 8, it appears in parity check set 13. Now this parity check set 1, that is already captured, because

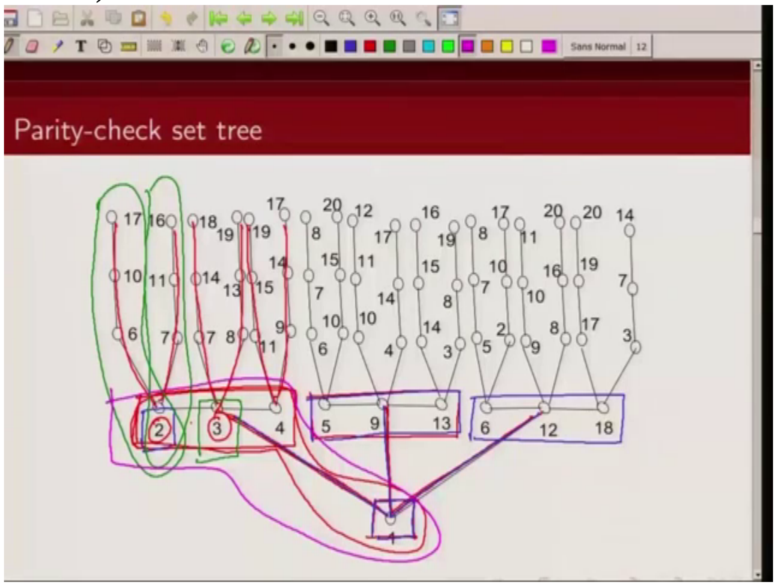
(Refer Slide Time 11:28)



Parity-check set tree

that is this one, it is already captured. So

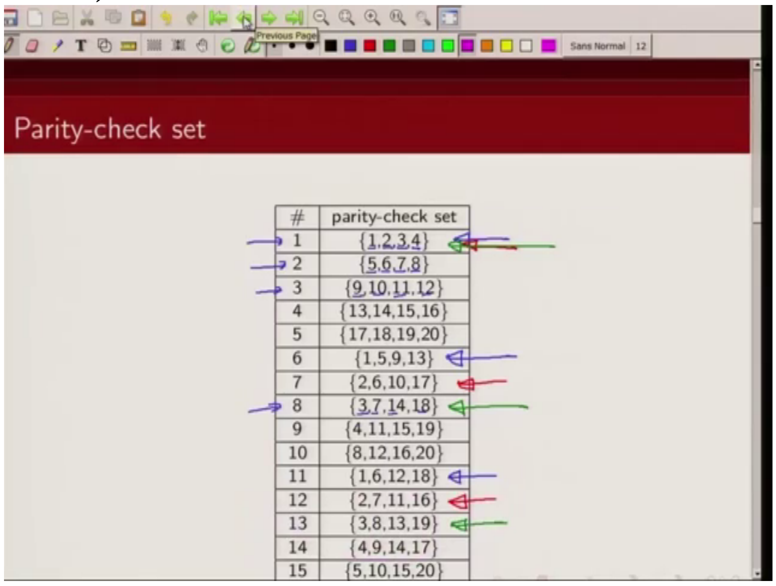
(Refer Slide Time 11:34)



Parity-check set tree

what are the other two parity check

(Refer Slide Time 11:36)

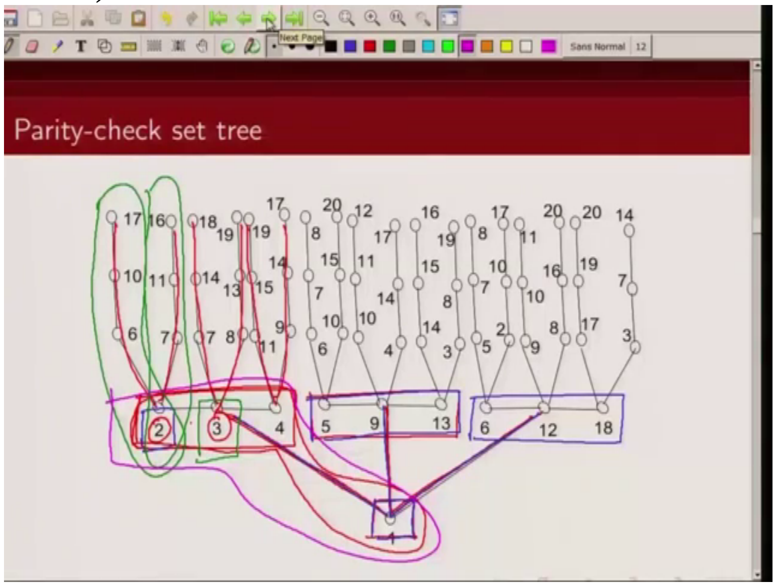


Parity-check set

#	parity-check set
1	{1,2,3,4}
2	{5,6,7,8}
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{1,5,9,13}
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{1,6,12,18}
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

sets? The one involving 3, 7, 14 and 18, so this is

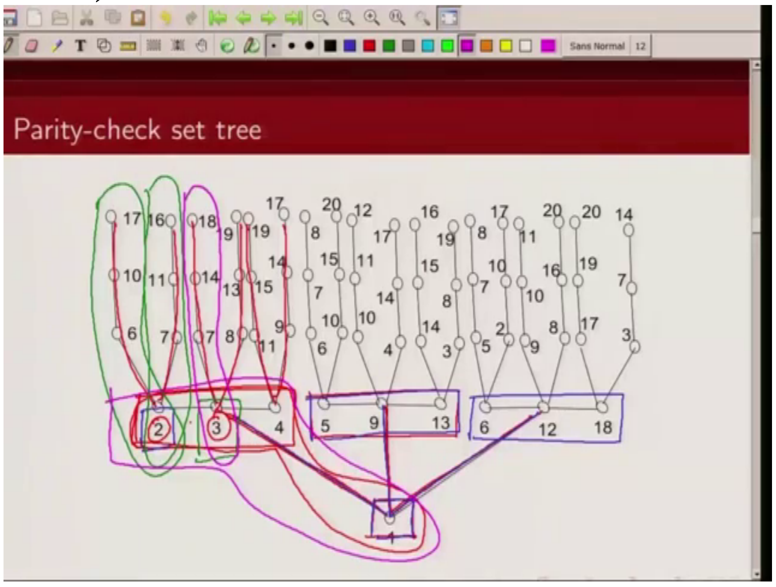
(Refer Slide Time 11:44)



Parity-check set tree

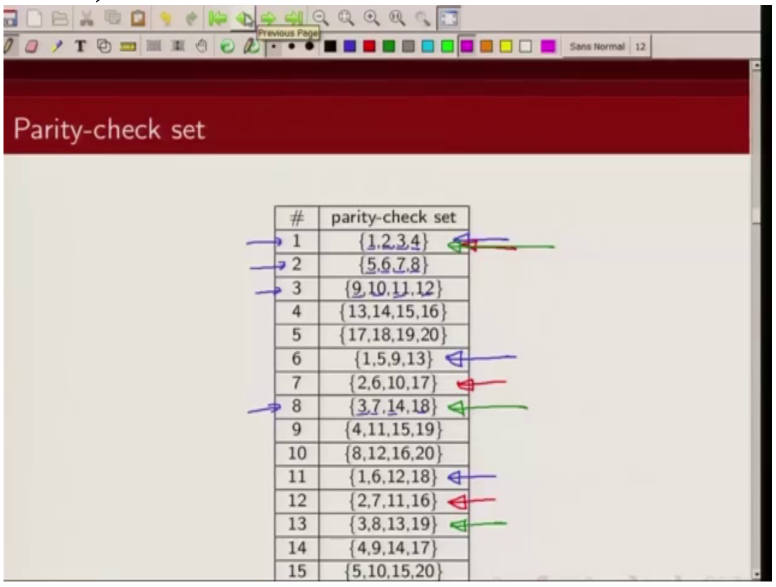
3, 7, 14 and 18, that's just 1.

(Refer Slide Time 11:52)



Parity-check set tree

(Refer Slide Time 11:53)



Parity-check set

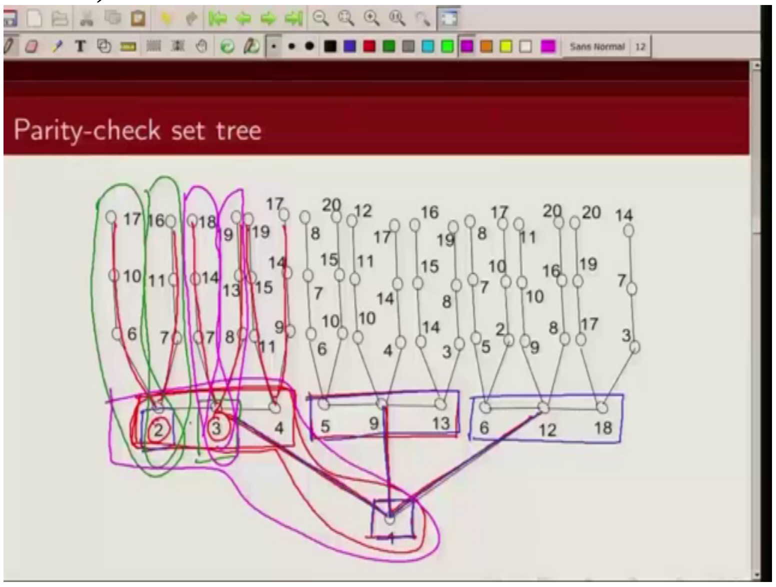
#	parity-check set
1	{1,2,3,4}
2	{5,6,7,8}
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{1,5,9,13}
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{1,6,12,18}
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

And the other one is 3, 8, 13 and 19. So this is this one, 3, 8, 13 and 19, Ok. So we are basically connecting by edges all these parity check sets. So that's how we are representing

(Refer Slide Time 12:14)

parity check set tree. Now we can do with other bits as well. We can for example

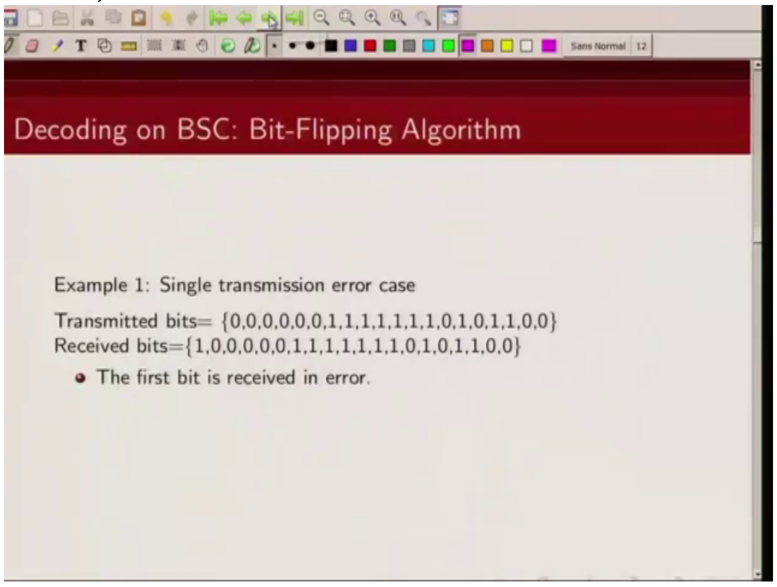
(Refer Slide Time 12:19)



Parity-check set tree

instead of making 1, if I can make this as 2, I will construct a tree around this node 2, same procedure.

(Refer Slide Time 12:28)



Decoding on BSC: Bit-Flipping Algorithm

Example 1: Single transmission error case

Transmitted bits= {0,0,0,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0}
Received bits={1,0,0,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0}

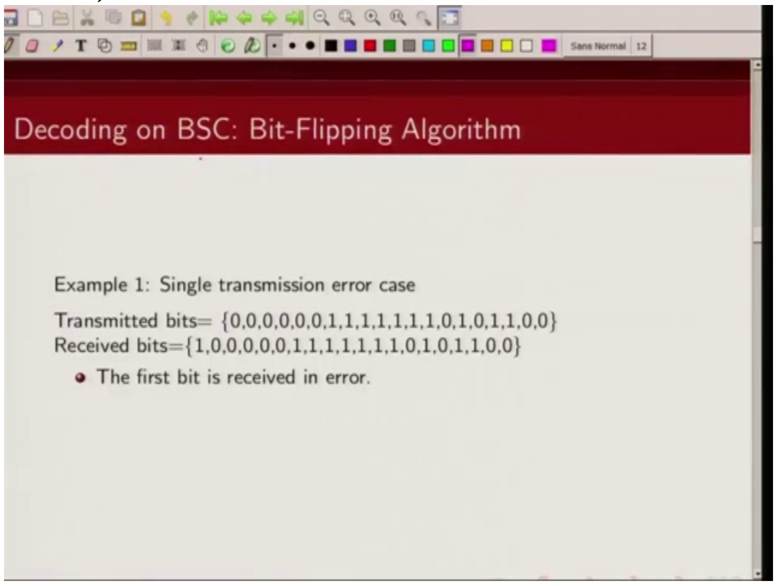
- The first bit is received in error.

Now let us look at how

(Refer Slide Time 12:31)

we can correct error. So we are considering

(Refer Slide Time 12:35)



Decoding on BSC: Bit-Flipping Algorithm

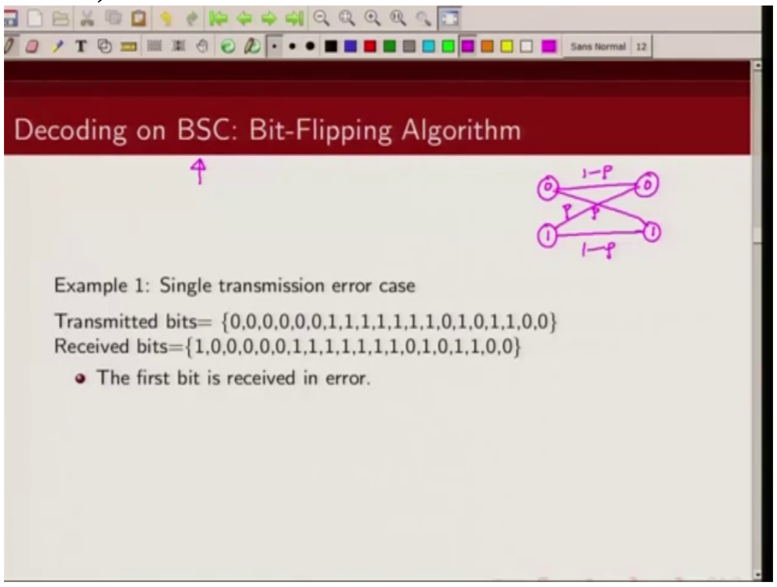
Example 1: Single transmission error case

Transmitted bits= {0,0,0,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0}
Received bits={1,0,0,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0}

- The first bit is received in error.

a binary symmetrical channel. Again recall what is a binary symmetrical channel? So there are 2 inputs, 0 and 1, 0 and 1 with probability 1 minus p you receive the bits correctly and there is a crossover probability of bits getting flipped.

(Refer Slide Time 12:54)



Decoding on BSC: Bit-Flipping Algorithm

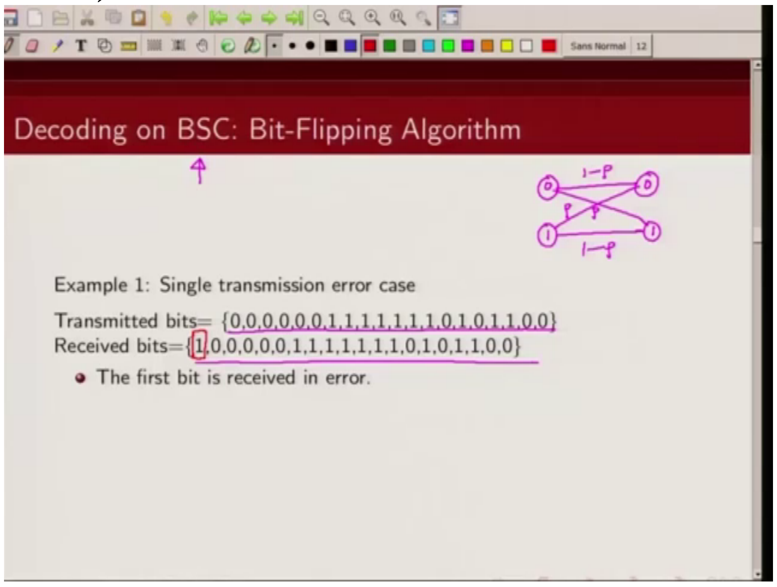
Example 1: Single transmission error case

Transmitted bits= {0,0,0,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0}
Received bits={1,0,0,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0}

- The first bit is received in error.

So let us consider that we have transmitted this information, we have transmitted this coded sequence and what we received is this. So there is an error in the first bit location. Now how do we correct this error? So to decode this what we are

(Refer Slide Time 13:18)



Decoding on BSC: Bit-Flipping Algorithm

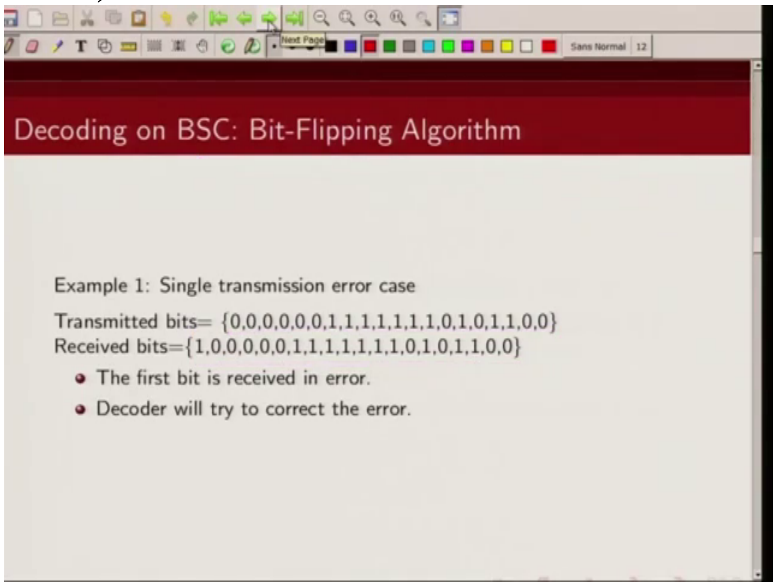
Example 1: Single transmission error case

Transmitted bits= {0,0,0,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0}
Received bits={1,0,0,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0}

- The first bit is received in error.

going to do is we are going to construct a parity check set tree around each of these bits and use that for our decoding purpose.

(Refer Slide Time 13:30)



Decoding on BSC: Bit-Flipping Algorithm

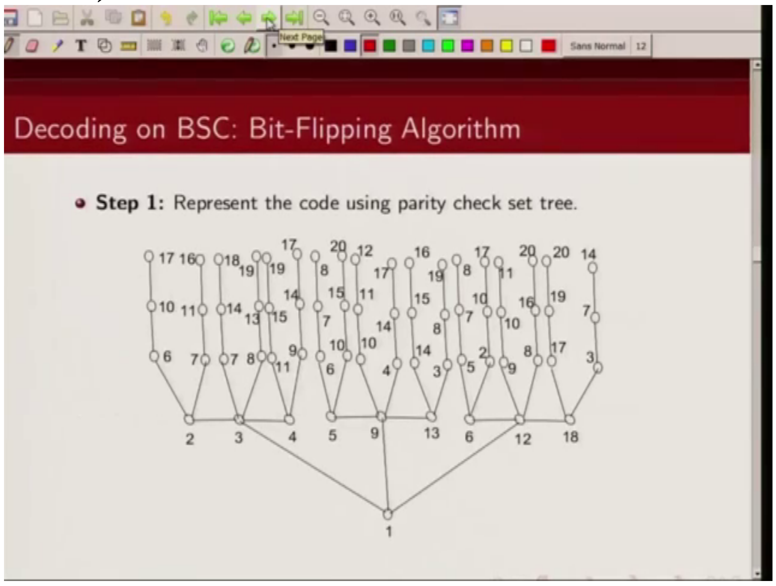
Example 1: Single transmission error case

Transmitted bits= {0,0,0,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0}
Received bits={1,0,0,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0}

- The first bit is received in error.
- Decoder will try to correct the error.

So let's see how we do that.

(Refer Slide Time 13:32)

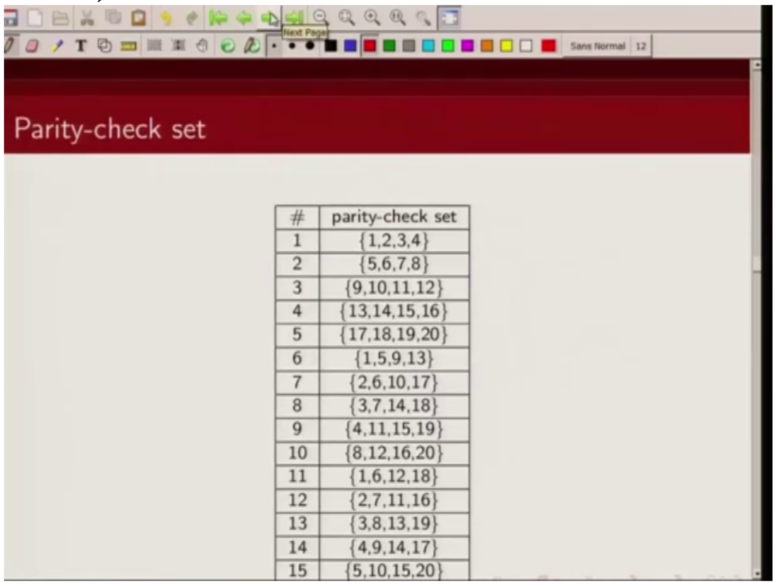


Decoding on BSC: Bit-Flipping Algorithm

- **Step 1:** Represent the code using parity check set tree.

So the first step is we construct the, we represent the code using parity check set tree and we have explained in the previous slide how this parity check set tree is constructed. So this is the parity check set tree for the bit number 1.

(Refer Slide Time 13:52)

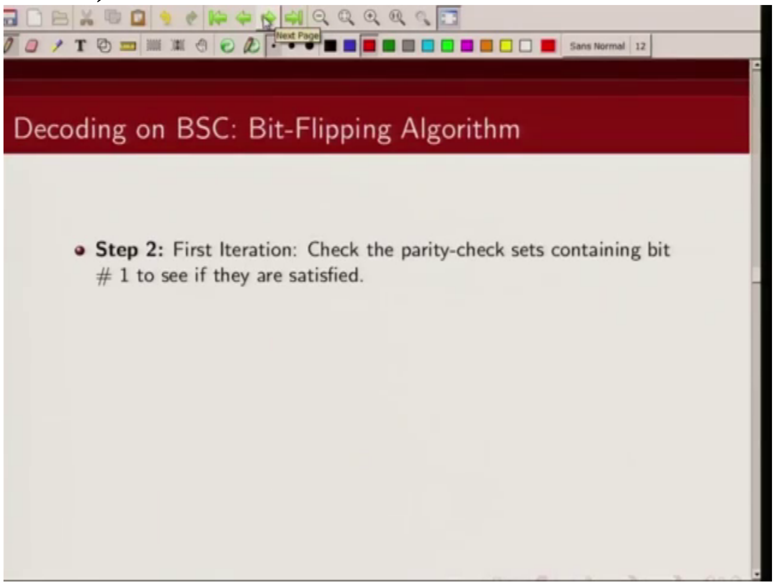


Parity-check set

#	parity-check set
1	{1,2,3,4}
2	{5,6,7,8}
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{1,5,9,13}
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{1,6,12,18}
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

And remember this parity check set, corresponding to this we have drawn this parity check set tree.

(Refer Slide Time 14:00)



Decoding on BSC: Bit-Flipping Algorithm

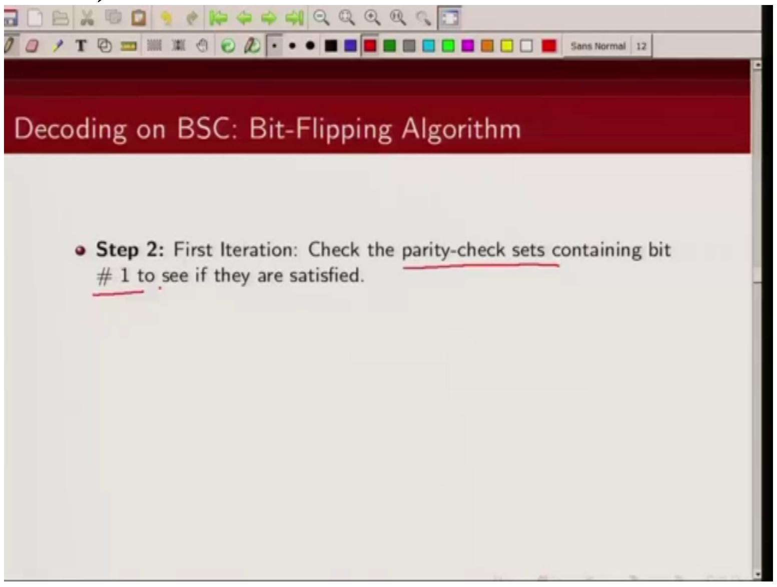
- **Step 2:** First Iteration: Check the parity-check sets containing bit # 1 to see if they are satisfied.

Now what we are going to see, first check is whether all the parity check sets containing bit number 1, if they are satisfied. If they are satisfied it is likely that bit number 1 is received correctly. If majority

(Refer Slide Time 14:19)

of them are not satisfied, it is likely that bit number 1 is in error. So

(Refer Slide Time 14:25)

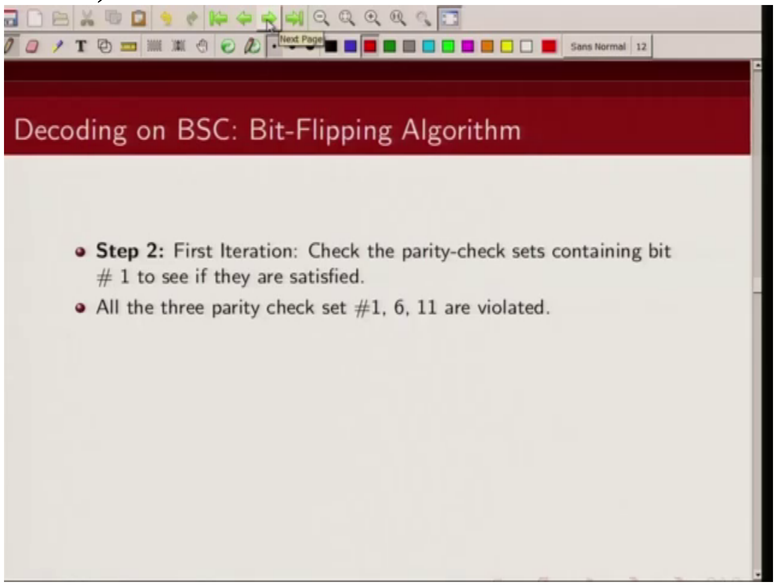


Decoding on BSC: Bit-Flipping Algorithm

- **Step 2:** First Iteration: Check the parity-check sets containing bit # 1 to see if they are satisfied.

let's see.

(Refer Slide Time 14:27)

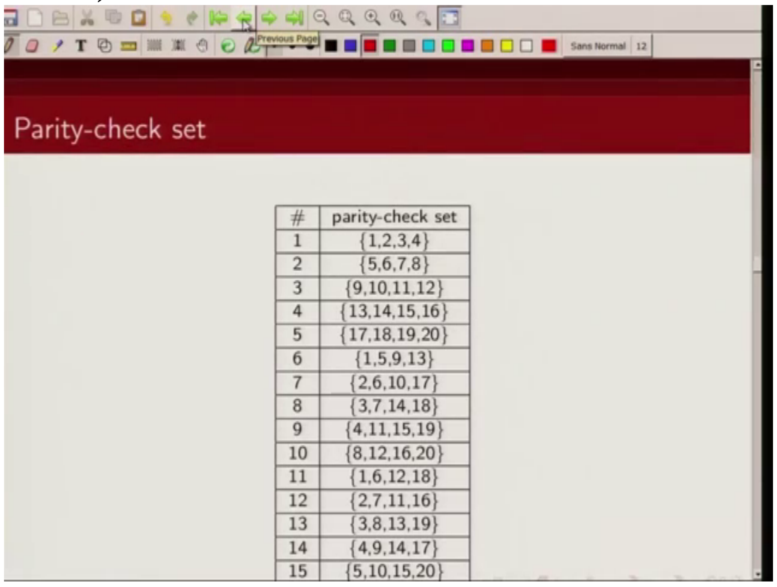


Decoding on BSC: Bit-Flipping Algorithm

- **Step 2:** First Iteration: Check the parity-check sets containing bit # 1 to see if they are satisfied.
- All the three parity check set #1, 6, 11 are violated.

Now which are the parity check sets in which bit number 1 is

(Refer Slide Time 14:31)

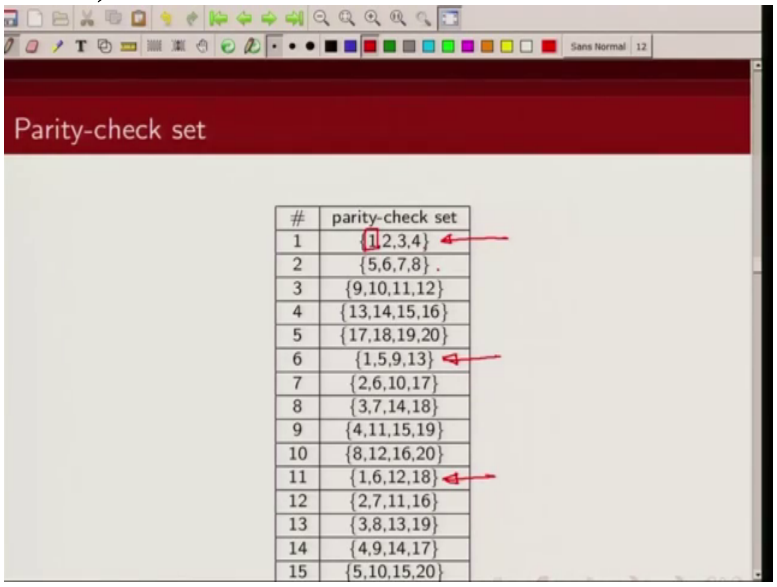


Parity-check set

#	parity-check set
1	{1,2,3,4}
2	{5,6,7,8}
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{1,5,9,13}
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{1,6,12,18}
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

participating? That is this, this one and this one. Now note in our example, there was a single error in bit number 1 location. So all other bits were received correctly only bit number 1 was in error. Then what is going to happen? This parity check set would not going to be satisfied because this bit is in error.

(Refer Slide Time 15:01)

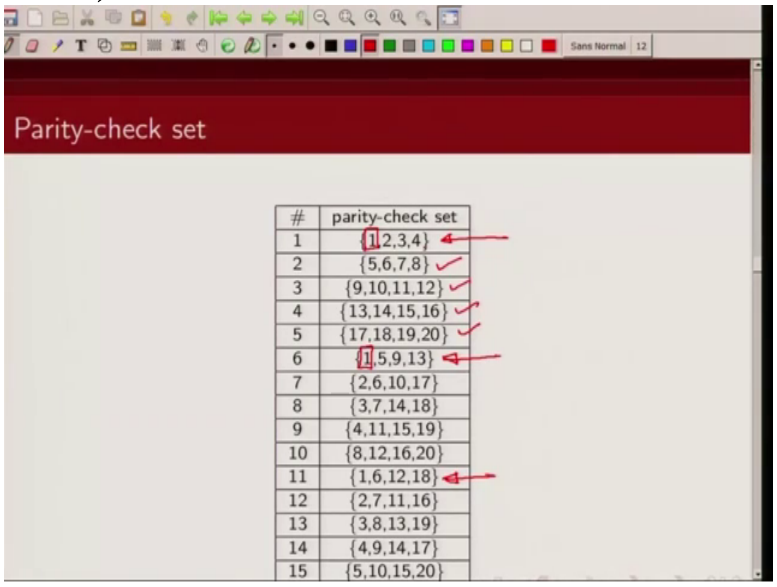


Parity-check set

#	parity-check set
1	{$\boxed{1}$,2,3,4} $\leftarrow$
2	{5,6,7,8} .
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{1,5,9,13} $\leftarrow$
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{1,6,12,18} $\leftarrow$
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

This will be satisfied, this will be satisfied, this will not be satisfied because this particular bit was

(Refer Slide Time 15:07)

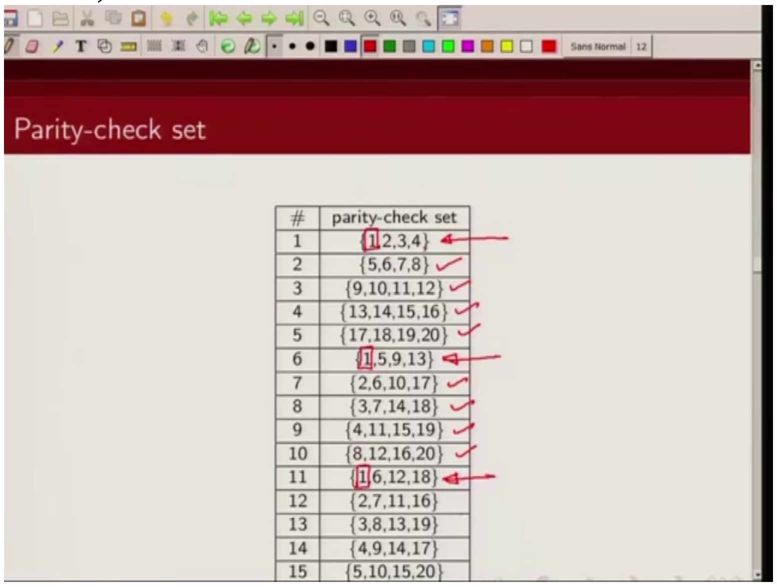


Parity-check set

#	parity-check set	
1	{$\boxed{1}$,2,3,4}	←
2	{5,6,7,8}	✓
3	{9,10,11,12}	✓
4	{13,14,15,16}	✓
5	{17,18,19,20}	✓
6	{$\boxed{1}$,5,9,13}	←
7	{2,6,10,17}	
8	{3,7,14,18}	
9	{4,11,15,19}	
10	{8,12,16,20}	
11	{1,6,12,18}	←
12	{2,7,11,16}	
13	{3,8,13,19}	
14	{4,9,14,17}	
15	{5,10,15,20}	

error. These are all satisfied. This will not be satisfied.

(Refer Slide Time 15:12)

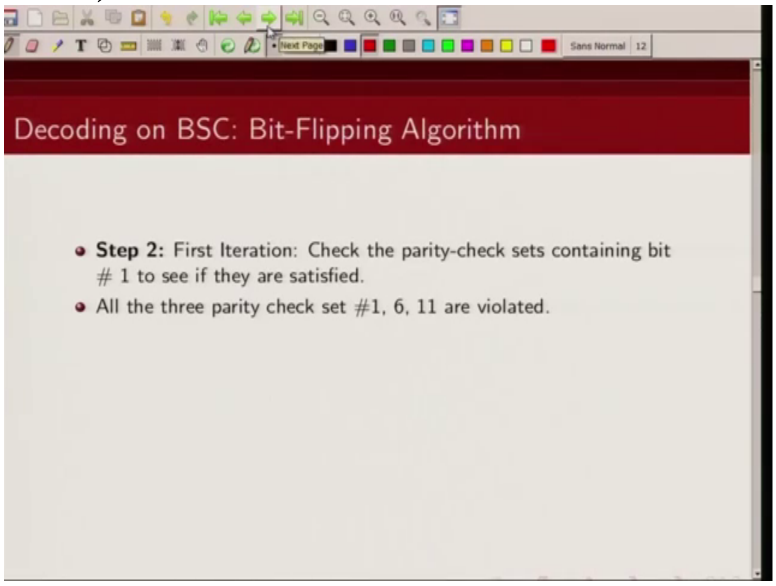


Parity-check set

#	parity-check set	
1	{$\boxed{1}$,2,3,4}	←
2	{5,6,7,8}	✓
3	{9,10,11,12}	✓
4	{13,14,15,16}	✓
5	{17,18,19,20}	✓
6	{$\boxed{1}$,5,9,13}	←
7	{2,6,10,17}	✓
8	{3,7,14,18}	✓
9	{4,11,15,19}	✓
10	{8,12,16,20}	✓
11	{$\boxed{1}$,6,12,18}	←
12	{2,7,11,16}	
13	{3,8,13,19}	
14	{4,9,14,17}	
15	{5,10,15,20}	

Again these are all satisfied. So you can see all the three parity check sets involving bit 1 are not satisfied in this particular example. So

(Refer Slide Time 15:25)



Decoding on BSC: Bit-Flipping Algorithm

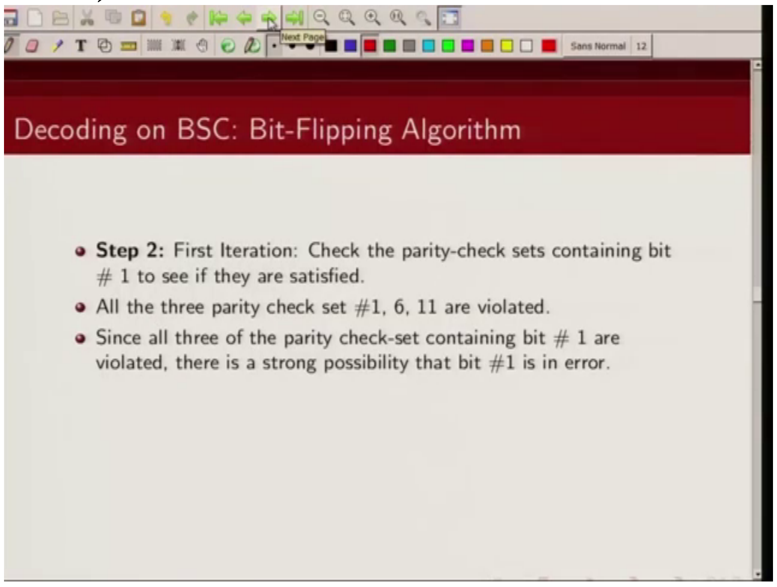
- **Step 2:** First Iteration: Check the parity-check sets containing bit # 1 to see if they are satisfied.
- All the three parity check set #1, 6, 11 are violated.

all the parity check sets containing 1, 6 and 11 are violated. What does that mean? It means that there is a very large likelihood of this particular bit

(Refer Slide Time 15:39)

being received in error.

(Refer Slide Time 15:44)

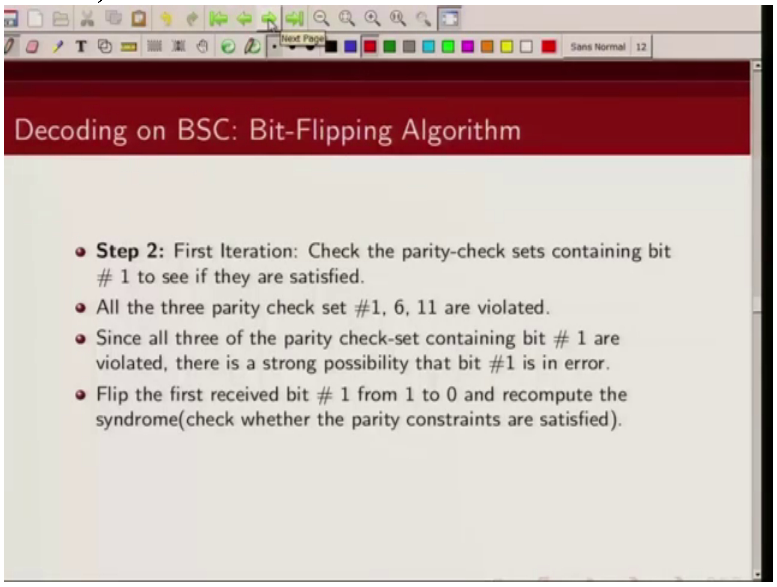


Decoding on BSC: Bit-Flipping Algorithm

- **Step 2:** First Iteration: Check the parity-check sets containing bit # 1 to see if they are satisfied.
- All the three parity check set #1, 6, 11 are violated.
- Since all three of the parity check-set containing bit # 1 are violated, there is a strong possibility that bit #1 is in error.

Hence what do we do? Then we

(Refer Slide Time 15:47)

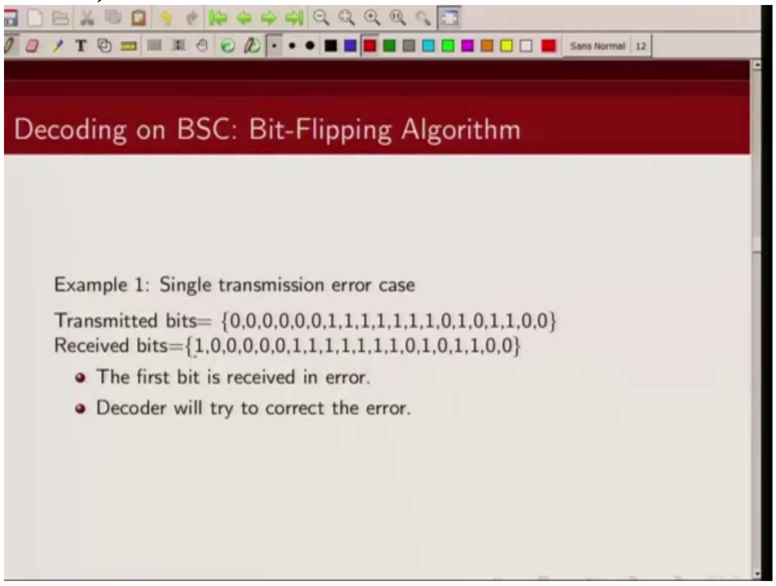


Decoding on BSC: Bit-Flipping Algorithm

- **Step 2:** First Iteration: Check the parity-check sets containing bit # 1 to see if they are satisfied.
- All the three parity check set #1, 6, 11 are violated.
- Since all three of the parity check-set containing bit # 1 are violated, there is a strong possibility that bit #1 is in error.
- Flip the first received bit # 1 from 1 to 0 and recompute the syndrome(check whether the parity constraints are satisfied).

are going to flip this bit 1. Whatever this bit was, we are going to flip it and again check the parity check constraints. So earlier this bit was received as

(Refer Slide Time 16:04)



Decoding on BSC: Bit-Flipping Algorithm

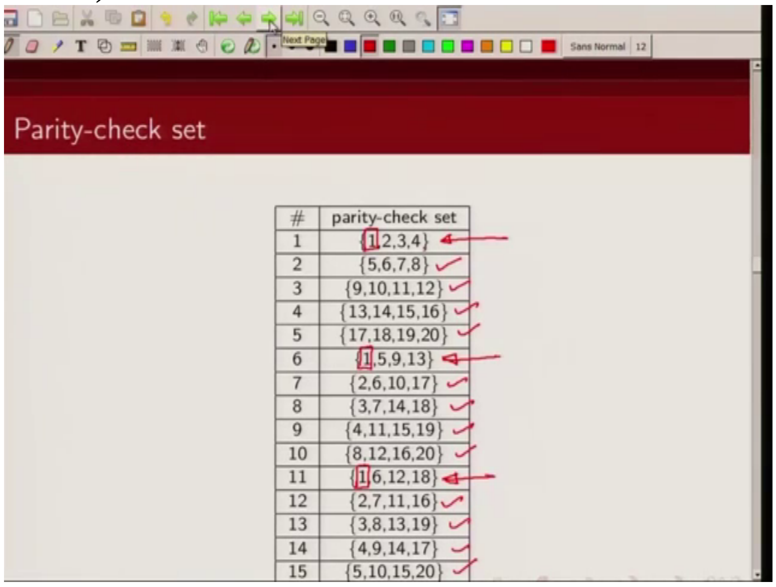
Example 1: Single transmission error case

Transmitted bits= {0,0,0,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0}
Received bits={1,0,0,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0}

- The first bit is received in error.
- Decoder will try to correct the error.

1. We are going to flip it to zero and again try to check the parity check equations. Now note

(Refer Slide Time 16:11)

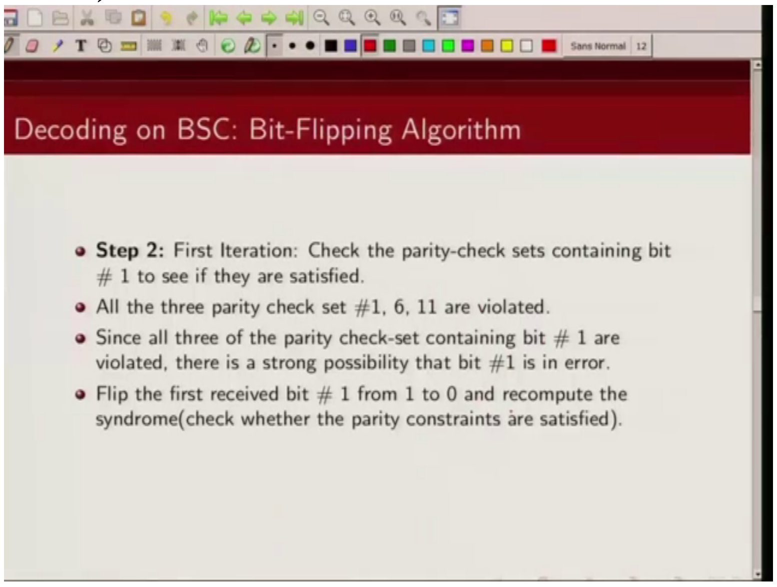


Parity-check set

#	parity-check set
1	{$\boxed{1}$,2,3,4}
2	{5,6,7,8}
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{$\boxed{1}$,5,9,13}
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{$\boxed{1}$,6,12,18}
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

that when we flip this bit, this bit is now no longer in error, these bits are no longer in error so then these parity check constraints will also be satisfied. Hence we are able to correct single error. So when

(Refer Slide Time 16:31)



Decoding on BSC: Bit-Flipping Algorithm

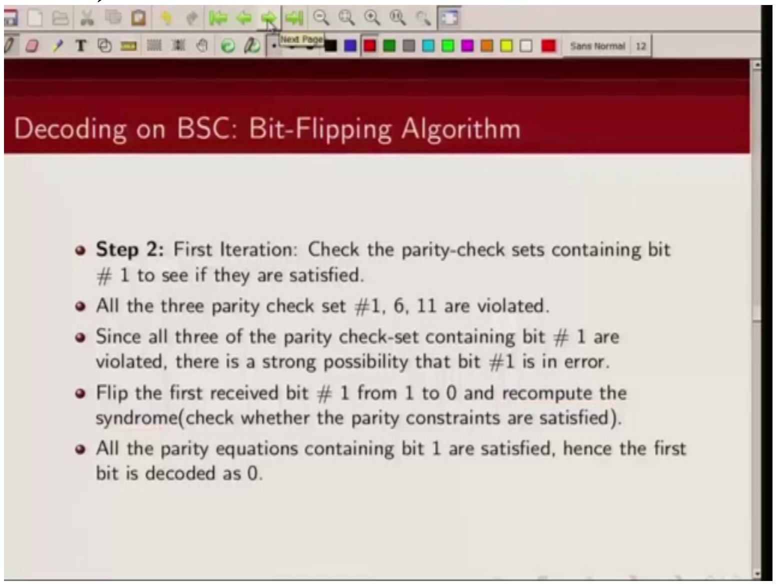
- **Step 2:** First Iteration: Check the parity-check sets containing bit # 1 to see if they are satisfied.
- All the three parity check set #1, 6, 11 are violated.
- Since all three of the parity check-set containing bit # 1 are violated, there is a strong possibility that bit #1 is in error.
- Flip the first received bit # 1 from 1 to 0 and recompute the syndrome(check whether the parity constraints are satisfied).

we recompute the syndrome we will see that all the parity check constraints because there was only single error

(Refer Slide Time 16:39)

which were able to detect

(Refer Slide Time 16:40)

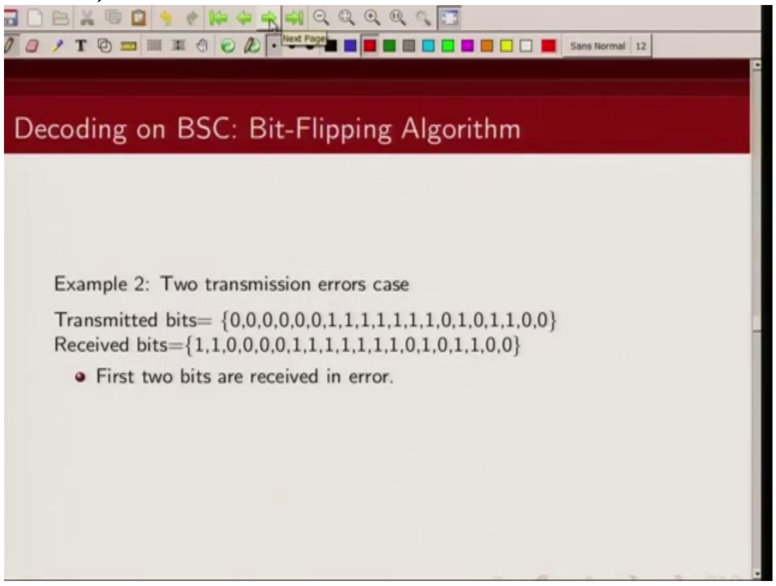


Decoding on BSC: Bit-Flipping Algorithm

- **Step 2:** First Iteration: Check the parity-check sets containing bit # 1 to see if they are satisfied.
- All the three parity check set #1, 6, 11 are violated.
- Since all three of the parity check-set containing bit # 1 are violated, there is a strong possibility that bit #1 is in error.
- Flip the first received bit # 1 from 1 to 0 and recompute the syndrome(check whether the parity constraints are satisfied).
- All the parity equations containing bit 1 are satisfied, hence the first bit is decoded as 0.

and we were able to correct it. So hence the first bit will be decoded as 0 and same procedure we will follow for other bits as well. And since there was no error in other bits all the parity check sets involving those bits will already be satisfied so we will be able to successfully

(Refer Slide Time 17:02)



Decoding on BSC: Bit-Flipping Algorithm

Example 2: Two transmission errors case

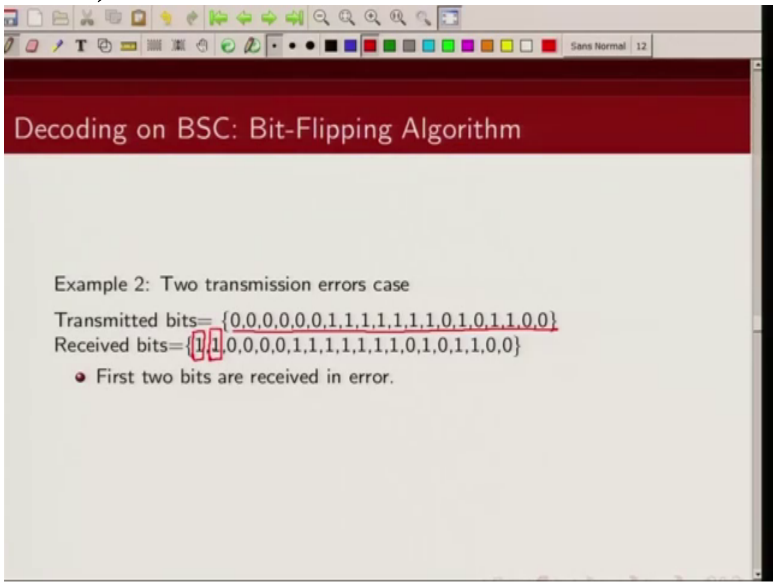
Transmitted bits= {0,0,0,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0}
Received bits={1,1,0,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0}

- First two bits are received in error.

decode it, Ok.

Now let’s look at the case when there are 2 errors. So the same transmitted codeword we have considered. In this case now we have considered there are 2 errors, in bit location 1

(Refer Slide Time 17:17)



Decoding on BSC: Bit-Flipping Algorithm

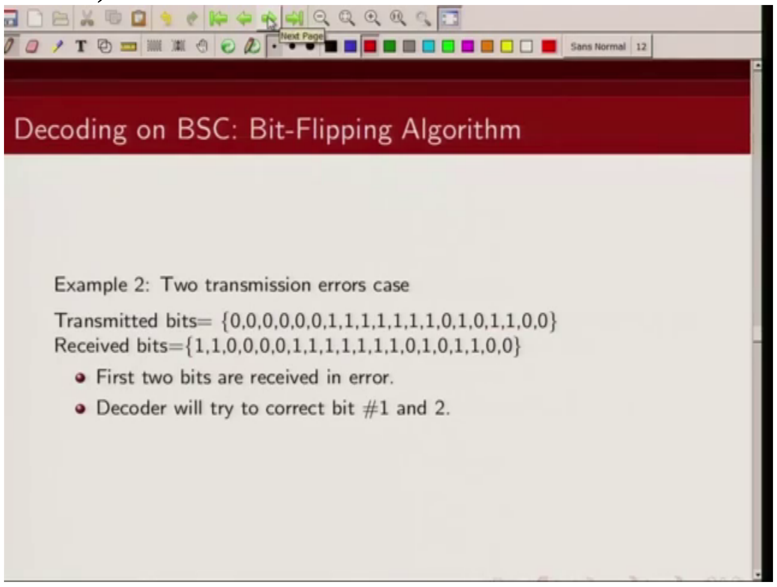
Example 2: Two transmission errors case

Transmitted bits= {0,0,0,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0}
Received bits={$\boxed{1}$,$\boxed{1}$,0,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0}

- First two bits are received in error.

and bit location 2. Now let's see how our L D P C decoder will be able to decode this.

(Refer Slide Time 17:25)



Decoding on BSC: Bit-Flipping Algorithm

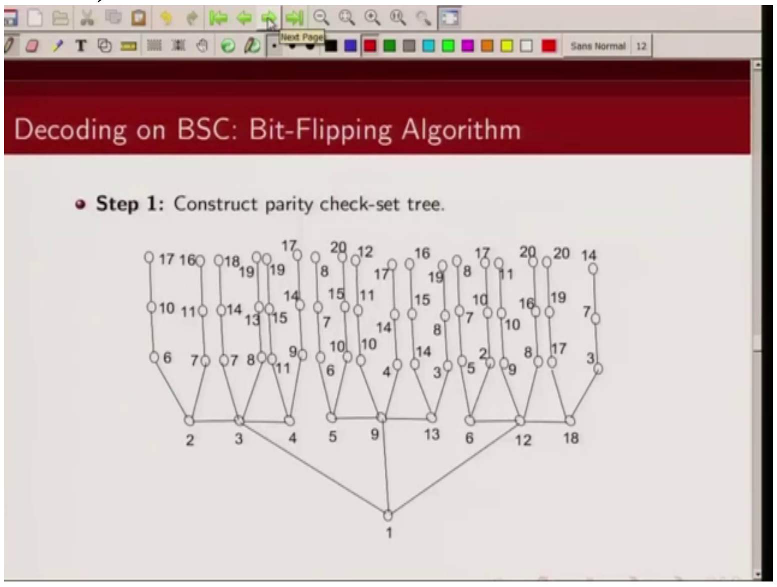
Example 2: Two transmission errors case

Transmitted bits= {0,0,0,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0}
Received bits={1,1,0,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0}

- First two bits are received in error.
- Decoder will try to correct bit #1 and 2.

So

(Refer Slide Time 17:28)

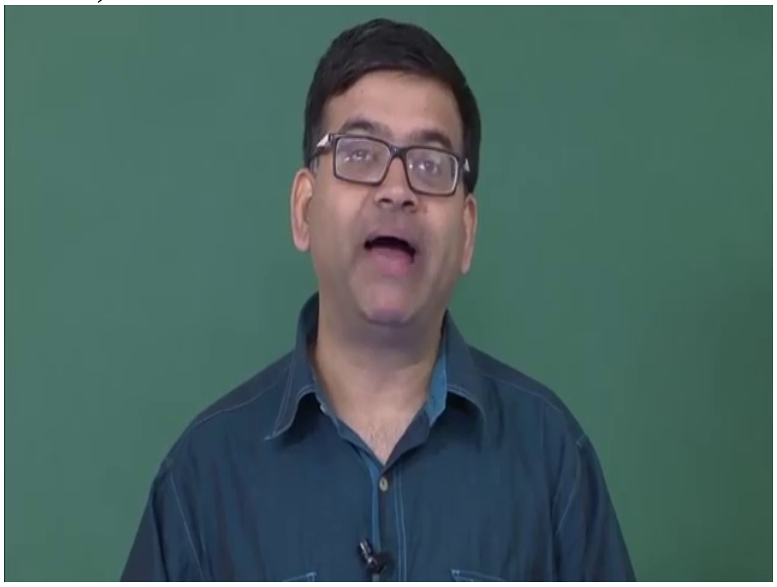


Decoding on BSC: Bit-Flipping Algorithm

- **Step 1:** Construct parity check-set tree.

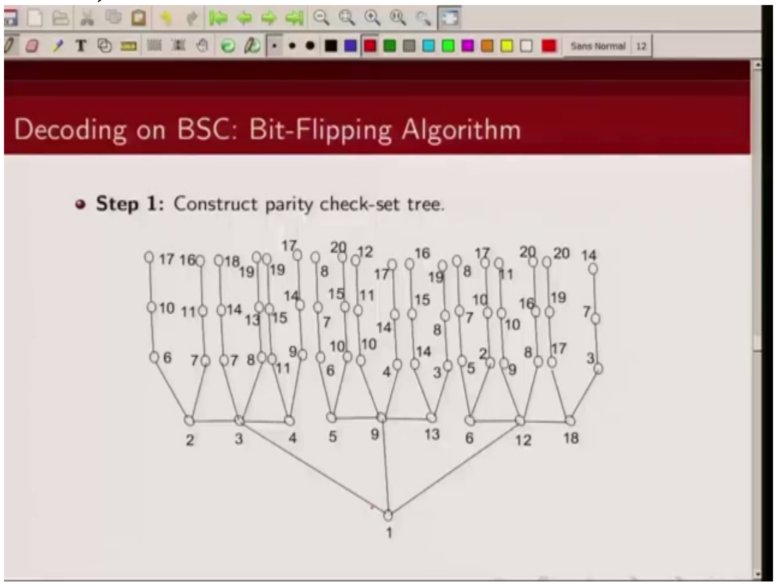
again we follow the same procedure. We draw the parity

(Refer Slide Time 17:32)



check set tree with each node at its base. So we start

(Refer Slide Time 17:37)

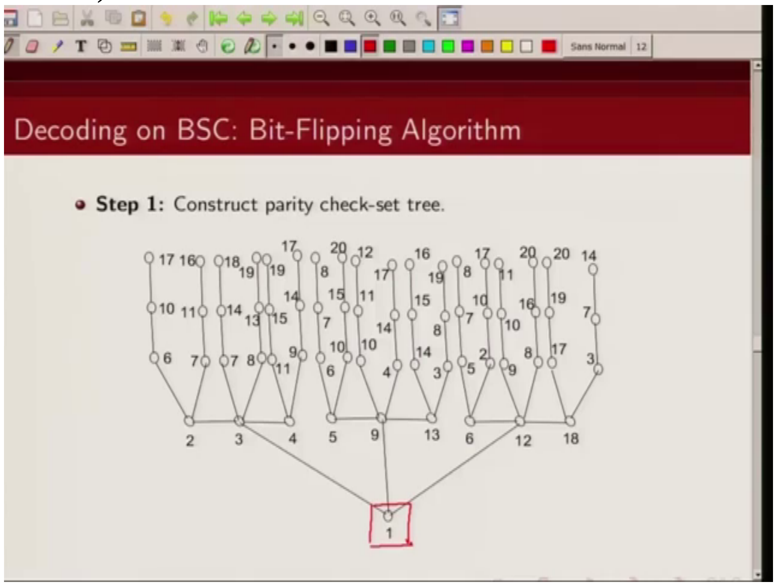


Decoding on BSC: Bit-Flipping Algorithm

- **Step 1:** Construct parity check-set tree.

with node number 1, we construct the

(Refer Slide Time 17:40)

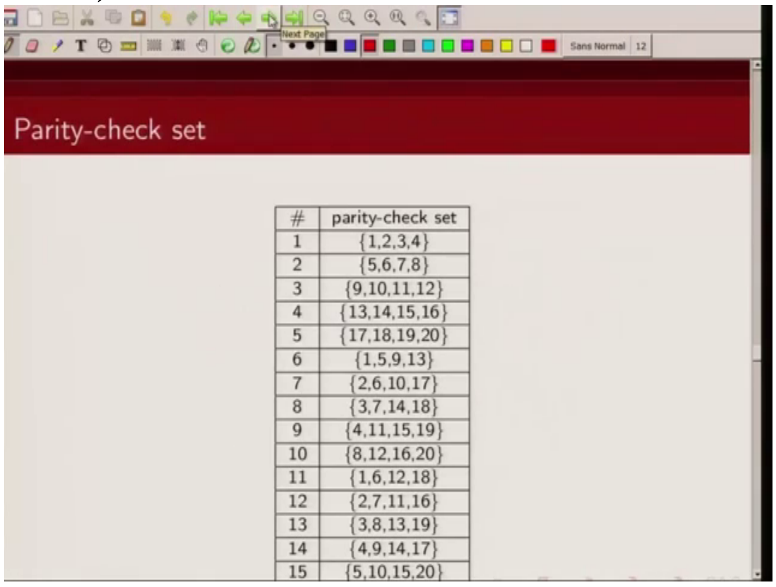


Decoding on BSC: Bit-Flipping Algorithm

- **Step 1:** Construct parity check-set tree.

parity check set tree.

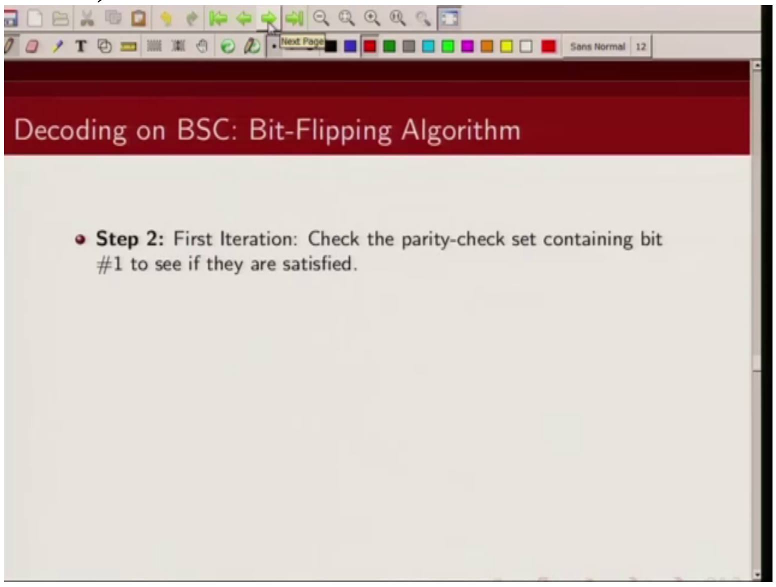
(Refer Slide Time 17:44)


Parity-check set

#	parity-check set
1	{1,2,3,4}
2	{5,6,7,8}
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{1,5,9,13}
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{1,6,12,18}
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

And these are the parity check sets, 15 parity check sets corresponding to the parity check matrix given to us.

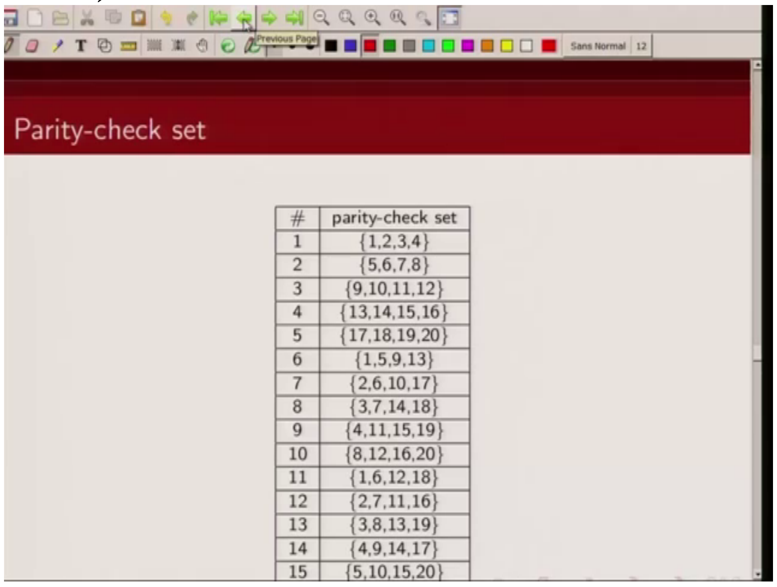
(Refer Slide Time 17:55)


Decoding on BSC: Bit-Flipping Algorithm

- **Step 2:** First Iteration: Check the parity-check set containing bit #1 to see if they are satisfied.

Now in the first step what we do is we check the parity set containing bit number 1 and we see if all the parity check constraints are satisfied. So what all are we going to do? We are

(Refer Slide Time 18:13)

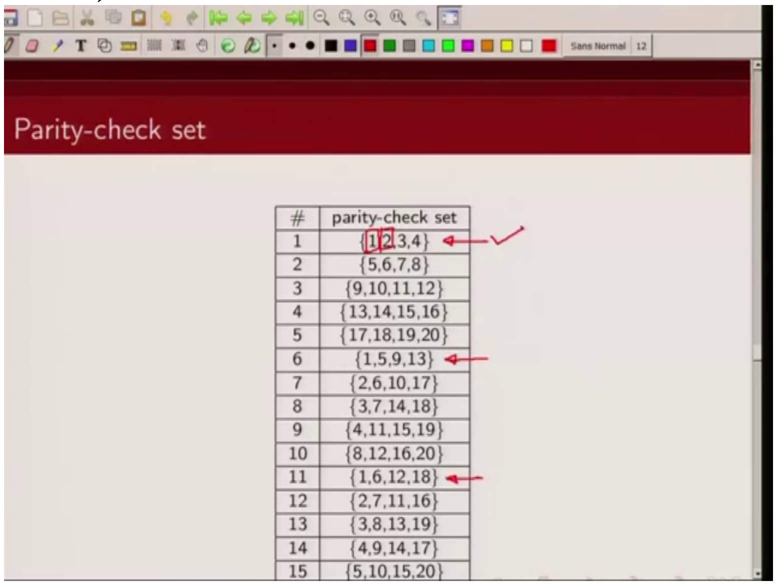


Parity-check set

#	parity-check set
1	{1,2,3,4}
2	{5,6,7,8}
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{1,5,9,13}
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{1,6,12,18}
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

going to look at all these parity check sets which have 1 in them. So this is our parity check set 1, 6 and 11. Now will this be satisfied? Yes it will be satisfied. Why? Because this was also in error and this was also in error. So this parity check

(Refer Slide Time 18:34)

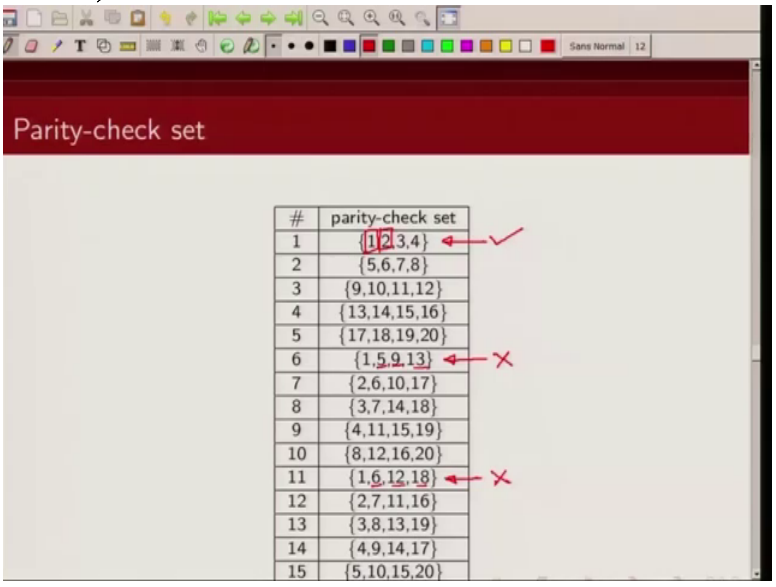


Parity-check set

#	parity-check set
1	{$\boxed{1}\boxed{2}$3,4} ← ✓
2	{5,6,7,8}
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{1,5,9,13} ←
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{1,6,12,18} ←
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

equation will be satisfied because 2 bits are in error, Ok. What about this? This parity check set will not be satisfied. Why? Because 5, 9 and 13 were received correctly but 1 was not received correctly so this parity check

(Refer Slide Time 19:06)



Parity-check set

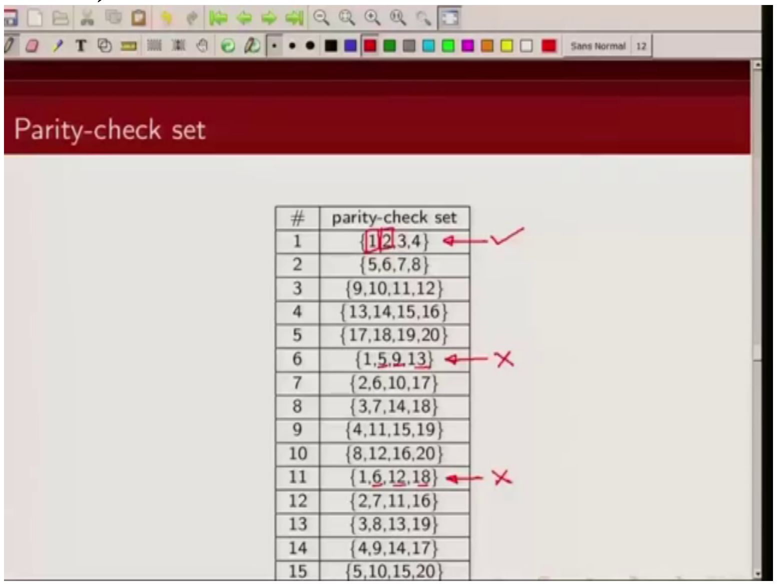
#	parity-check set	
1	$\{\boxed{1}\boxed{2}3,4\}$	$\leftarrow$ ✓
2	{5,6,7,8}	
3	{9,10,11,12}	
4	{13,14,15,16}	
5	{17,18,19,20}	
6	$\{1,\underline{5},\underline{9},\underline{13}\}$	$\leftarrow$ ×
7	{2,6,10,17}	
8	{3,7,14,18}	
9	{4,11,15,19}	
10	{8,12,16,20}	
11	$\{1,\underline{6},\underline{12},\underline{18}\}$	$\leftarrow$ ×
12	{2,7,11,16}	
13	{3,8,13,19}	
14	{4,9,14,17}	
15	{5,10,15,20}	

set will not be satisfied. Similarly here 6, 12 and 18 will be, are received correctly but 1 is not. So then this parity check set will not be satisfied. So what we have seen here in the case of double error is

(Refer Slide Time 19:13)

two of the parity check set involving 1 is not satisfied where as 1 is satisfied. Now what does

(Refer Slide Time 19:21)

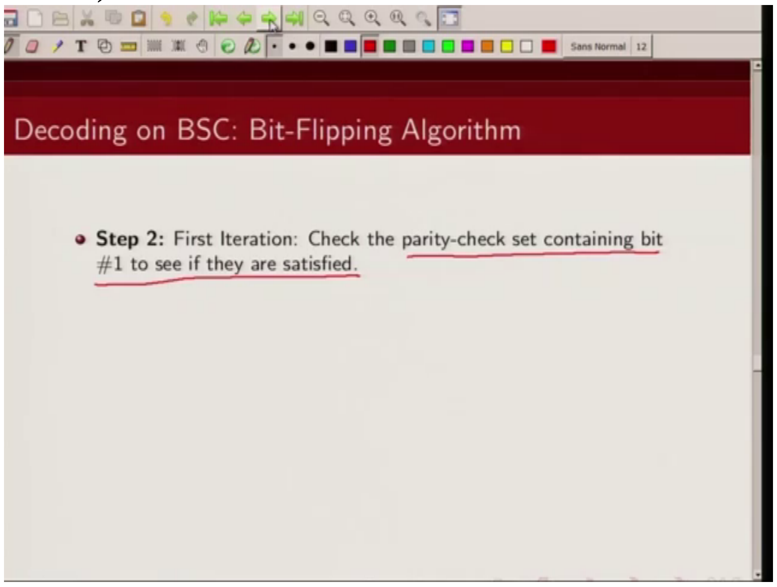


Parity-check set

#	parity-check set	
1	{1,2,3,4}	✓
2	{5,6,7,8}	
3	{9,10,11,12}	
4	{13,14,15,16}	
5	{17,18,19,20}	
6	{1,5,9,13}	✗
7	{2,6,10,17}	
8	{3,7,14,18}	
9	{4,11,15,19}	
10	{8,12,16,20}	
11	{1,6,12,18}	✗
12	{2,7,11,16}	
13	{3,8,13,19}	
14	{4,9,14,17}	
15	{5,10,15,20}	

that tell us? It tells us since majority of them are not satisfied it is likely that bit 1 was in error so we are going to flip it and try to

(Refer Slide Time 19:32)

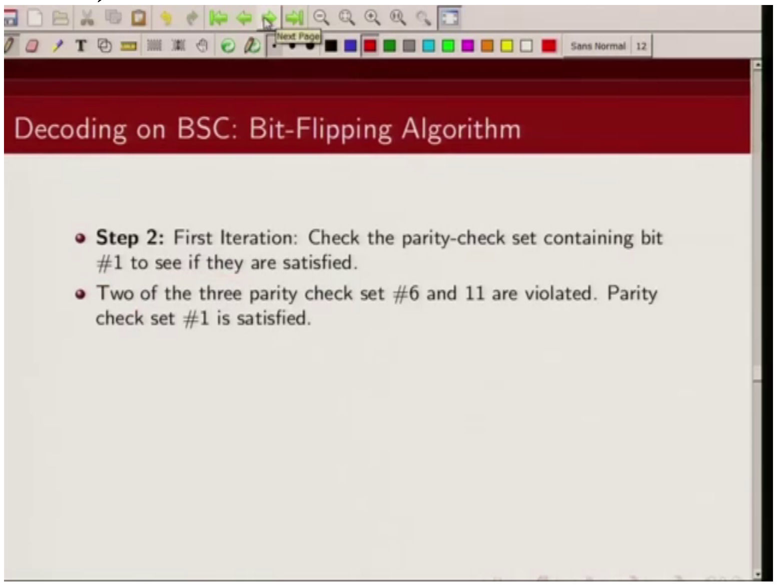


Decoding on BSC: Bit-Flipping Algorithm

- **Step 2:** First Iteration: Check the parity-check set containing bit #1 to see if they are satisfied.

do the same thing again. So

(Refer Slide Time 19:35)

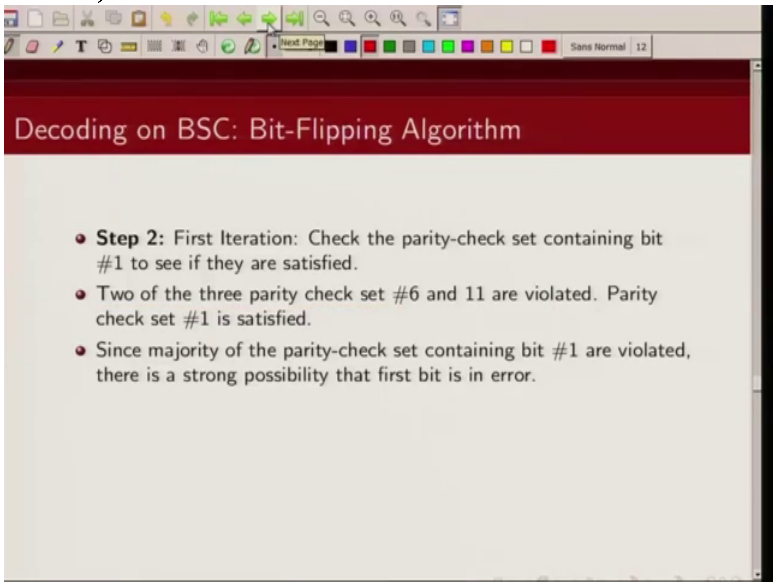


Decoding on BSC: Bit-Flipping Algorithm

- **Step 2:** First Iteration: Check the parity-check set containing bit #1 to see if they are satisfied.
- Two of the three parity check set #6 and 11 are violated. Parity check set #1 is satisfied.

since 2 of the parity check sets are violated, it is likely that bit 1

(Refer Slide Time 19:43)

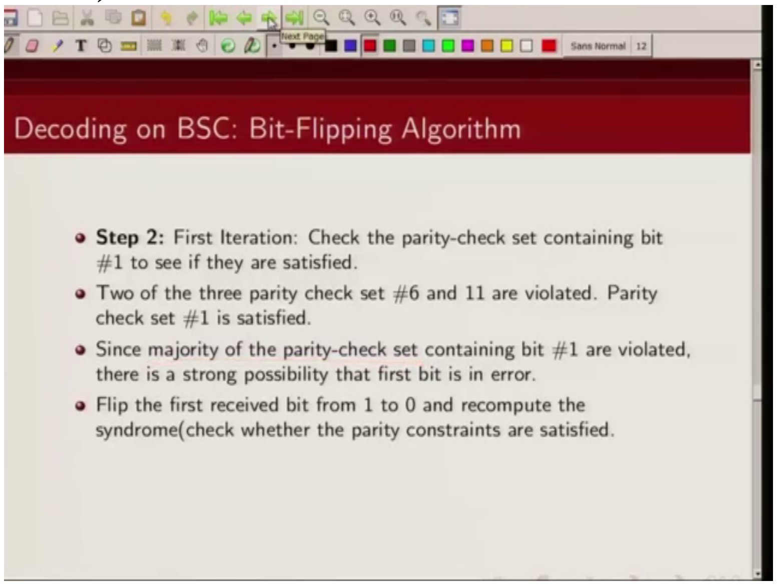


Decoding on BSC: Bit-Flipping Algorithm

- **Step 2:** First Iteration: Check the parity-check set containing bit #1 to see if they are satisfied.
- Two of the three parity check set #6 and 11 are violated. Parity check set #1 is satisfied.
- Since majority of the parity-check set containing bit #1 are violated, there is a strong possibility that first bit is in error.

is in error because majority of the parity check sets containing 1 are not satisfied.

(Refer Slide Time 19:53)



Decoding on BSC: Bit-Flipping Algorithm

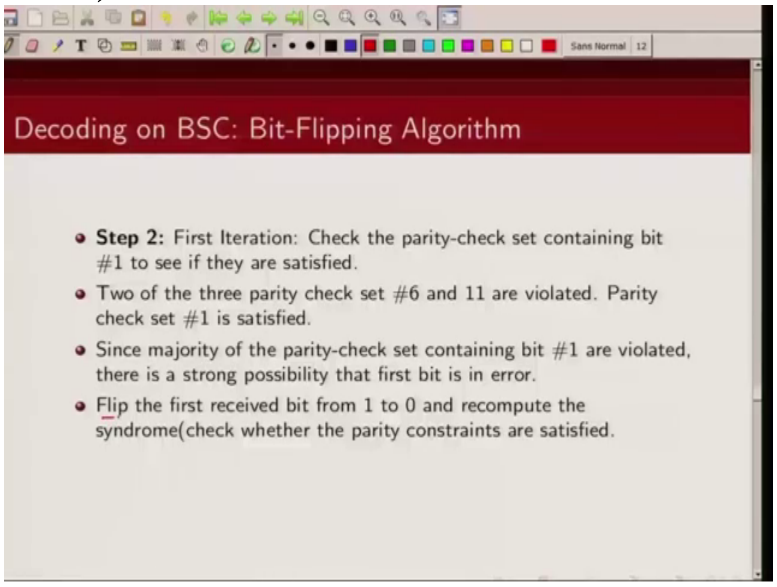
- **Step 2:** First Iteration: Check the parity-check set containing bit #1 to see if they are satisfied.
- Two of the three parity check set #6 and 11 are violated. Parity check set #1 is satisfied.
- Since majority of the parity-check set containing bit #1 are violated, there is a strong possibility that first bit is in error.
- Flip the first received bit from 1 to 0 and recompute the syndrome(check whether the parity constraints are satisfied.

So what do we do if majority of them are saying they are not satisfied, we are going to

(Refer Slide Time 19:59)

flip that bit. So we are going to

(Refer Slide Time 20:02)

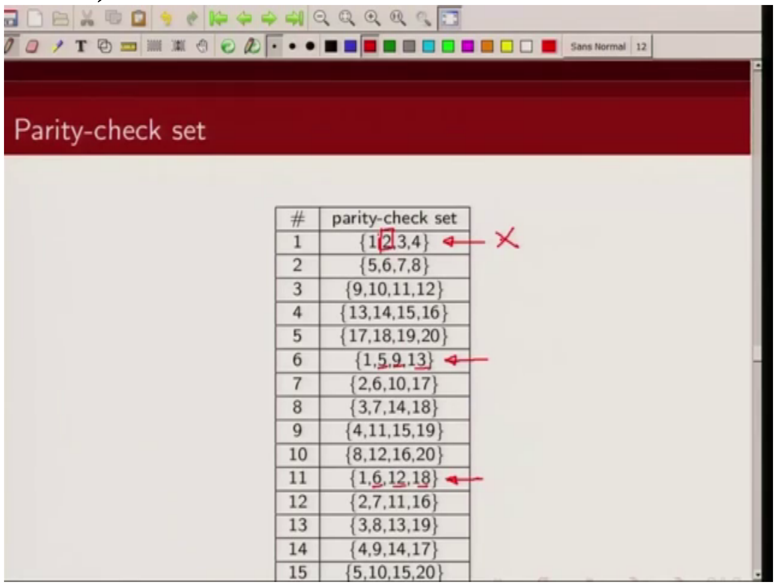


Decoding on BSC: Bit-Flipping Algorithm

- **Step 2:** First Iteration: Check the parity-check set containing bit #1 to see if they are satisfied.
- Two of the three parity check set #6 and 11 are violated. Parity check set #1 is satisfied.
- Since majority of the parity-check set containing bit #1 are violated, there is a strong possibility that first bit is in error.
- Flip the first received bit from 1 to 0 and recompute the syndrome(check whether the parity constraints are satisfied.

flip the first bit from 1 to 0 and again recompute our parity check constraints. So let's do that. So this bit has been flipped. Now if this bit has been flipped what's going to happen? If this bit is flipped, now this bit has been corrected but 2 was in error so this parity check set which was already getting satisfied

(Refer Slide Time 20:29)

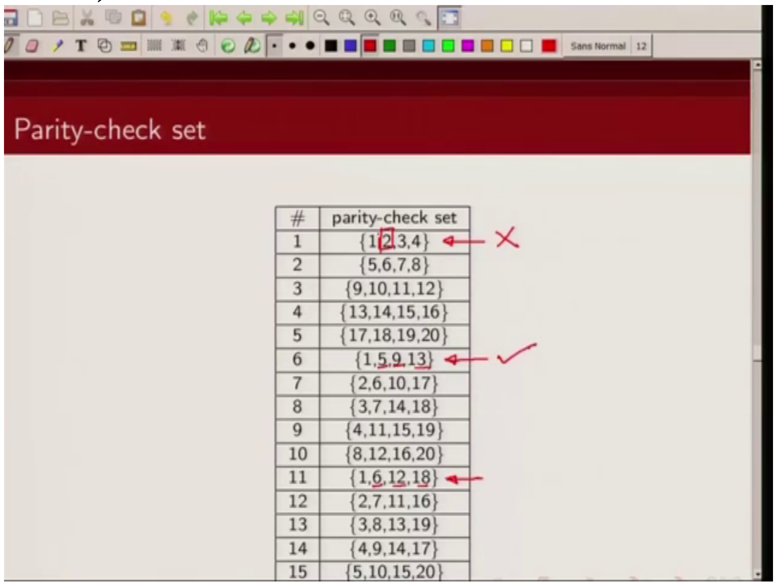


Parity-check set

#	parity-check set
1	{1,$\boxed{2}$,3,4}
2	{5,6,7,8}
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{1,5,9,13}
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{1,6,12,18}
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

is now not getting satisfied. What about this? It is getting satisfied.

(Refer Slide Time 20:34)

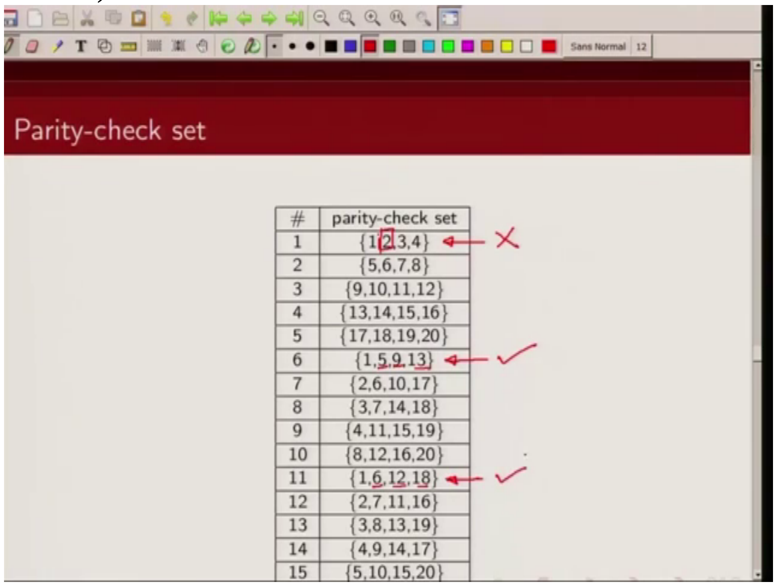


Parity-check set

#	parity-check set	
1	{1,$\boxed{2}$,3,4}	← ✗
2	{5,6,7,8}	
3	{9,10,11,12}	
4	{13,14,15,16}	
5	{17,18,19,20}	
6	{1,5,9,13}	← ✓
7	{2,6,10,17}	
8	{3,7,14,18}	
9	{4,11,15,19}	
10	{8,12,16,20}	
11	{1,6,12,18}	←
12	{2,7,11,16}	
13	{3,8,13,19}	
14	{4,9,14,17}	
15	{5,10,15,20}	

What about this? It is getting satisfied.

(Refer Slide Time 20:38)

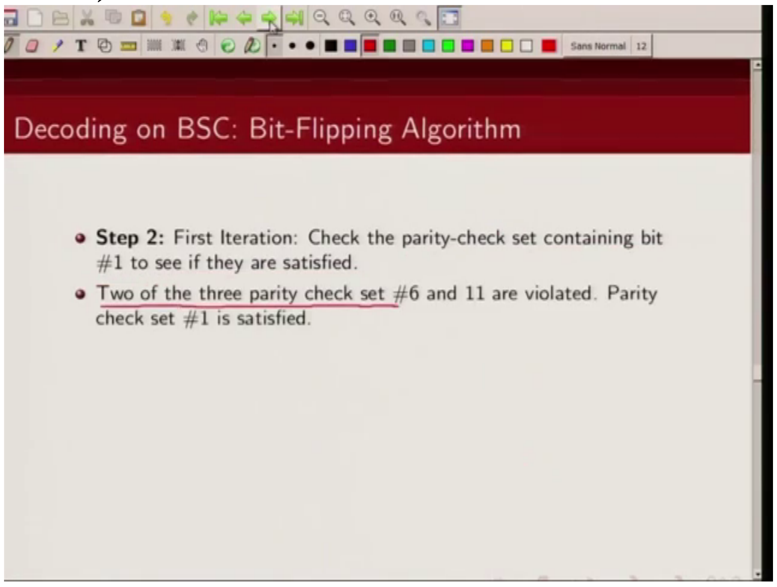


Parity-check set

#	parity-check set	
1	{1,$\boxed{2}$,3,4}	← ✗
2	{5,6,7,8}	
3	{9,10,11,12}	
4	{13,14,15,16}	
5	{17,18,19,20}	
6	{1,5,9,13}	← ✓
7	{2,6,10,17}	
8	{3,7,14,18}	
9	{4,11,15,19}	
10	{8,12,16,20}	
11	{1,6,12,18}	← ✓
12	{2,7,11,16}	
13	{3,8,13,19}	
14	{4,9,14,17}	
15	{5,10,15,20}	

So two of them are getting satisfied, one of them is not getting satisfied. So then first iteration is not enough to

(Refer Slide Time 20:46)

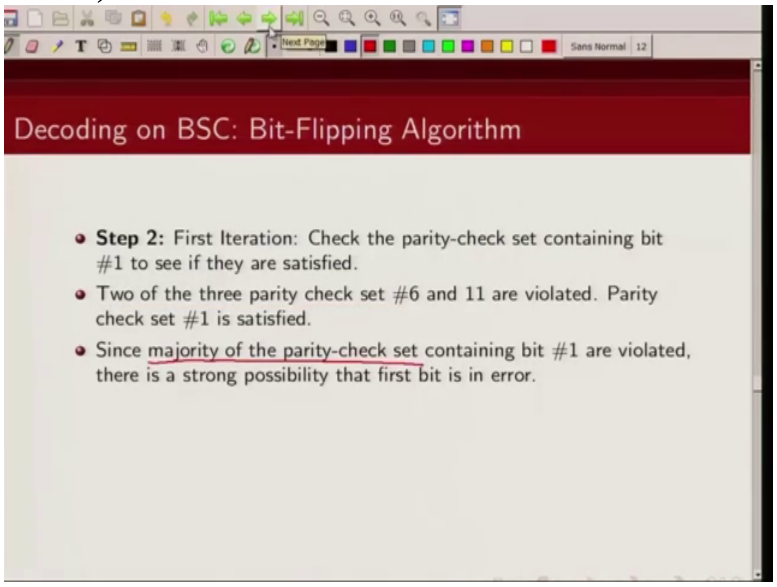


Decoding on BSC: Bit-Flipping Algorithm

- **Step 2:** First Iteration: Check the parity-check set containing bit #1 to see if they are satisfied.
- Two of the three parity check set #6 and 11 are violated. Parity check set #1 is satisfied.

decode this bit

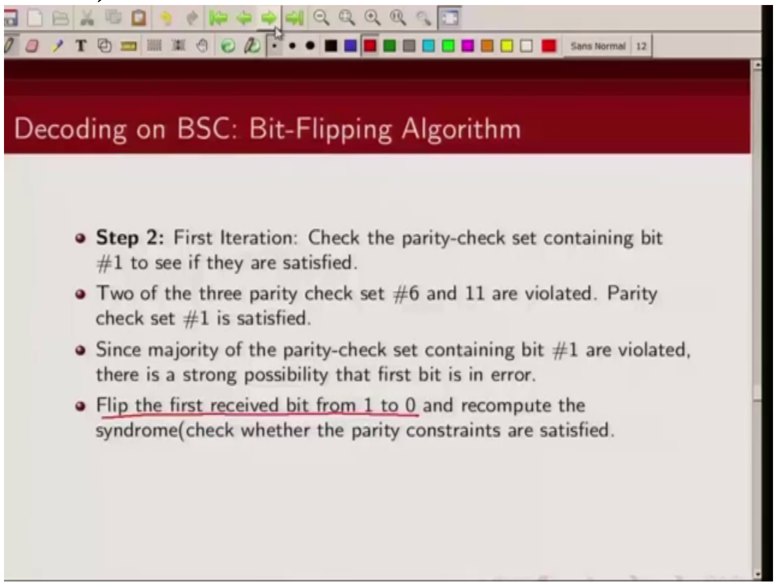
(Refer Slide Time 20:48)



Decoding on BSC: Bit-Flipping Algorithm

- **Step 2:** First Iteration: Check the parity-check set containing bit #1 to see if they are satisfied.
- Two of the three parity check set #6 and 11 are violated. Parity check set #1 is satisfied.
- Since majority of the parity-check set containing bit #1 are violated, there is a strong possibility that first bit is in error.

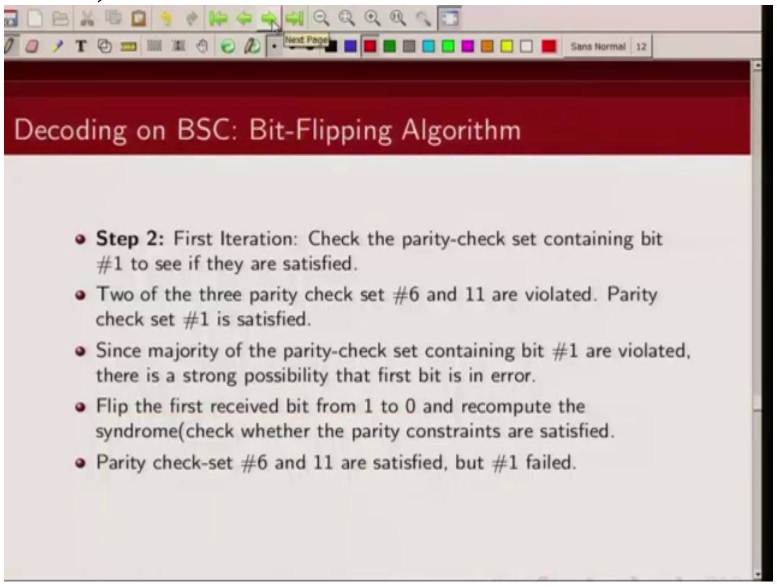
(Refer Slide Time 20:49)



Decoding on BSC: Bit-Flipping Algorithm

- **Step 2:** First Iteration: Check the parity-check set containing bit #1 to see if they are satisfied.
- Two of the three parity check set #6 and 11 are violated. Parity check set #1 is satisfied.
- Since majority of the parity-check set containing bit #1 are violated, there is a strong possibility that first bit is in error.
- Flip the first received bit from 1 to 0 and recompute the syndrome(check whether the parity constraints are satisfied.

(Refer Slide Time 20:51)

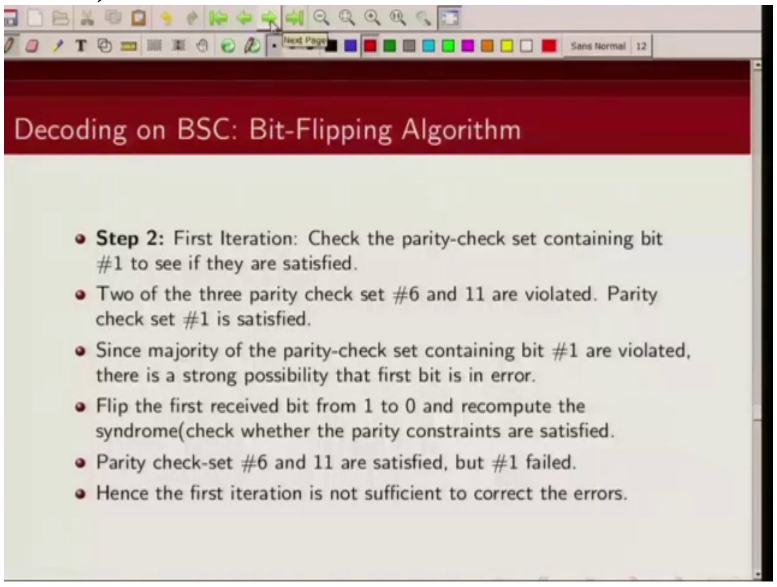


Decoding on BSC: Bit-Flipping Algorithm

- **Step 2:** First Iteration: Check the parity-check set containing bit #1 to see if they are satisfied.
- Two of the three parity check set #6 and 11 are violated. Parity check set #1 is satisfied.
- Since majority of the parity-check set containing bit #1 are violated, there is a strong possibility that first bit is in error.
- Flip the first received bit from 1 to 0 and recompute the syndrome(check whether the parity constraints are satisfied.
- Parity check-set #6 and 11 are satisfied, but #1 failed.

because parity check set first fail, there were 2 single errors,

(Refer Slide Time 20:57)

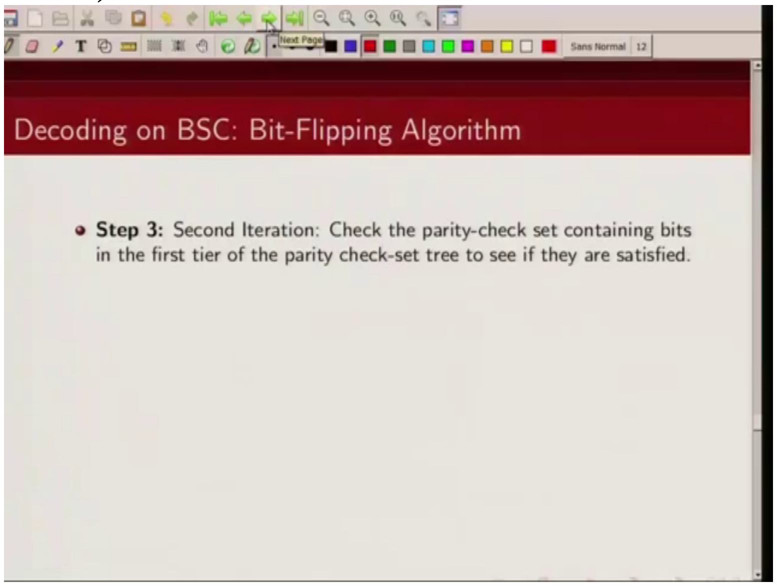


Decoding on BSC: Bit-Flipping Algorithm

- **Step 2:** First Iteration: Check the parity-check set containing bit #1 to see if they are satisfied.
- Two of the three parity check set #6 and 11 are violated. Parity check set #1 is satisfied.
- Since majority of the parity-check set containing bit #1 are violated, there is a strong possibility that first bit is in error.
- Flip the first received bit from 1 to 0 and recompute the syndrome(check whether the parity constraints are satisfied.
- Parity check-set #6 and 11 are satisfied, but #1 failed.
- Hence the first iteration is not sufficient to correct the errors.

two errors so first iteration is not sufficient to correct the errors. So then we will go to the next tier.

(Refer Slide Time 21:04)

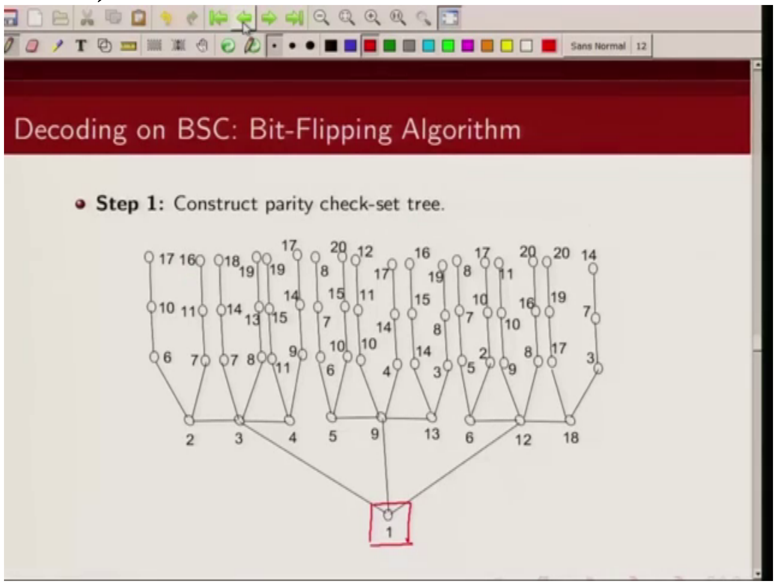


Decoding on BSC: Bit-Flipping Algorithm

- **Step 3:** Second Iteration: Check the parity-check set containing bits in the first tier of the parity check-set tree to see if they are satisfied.

Next iteration we will check parity check set containing bits in the first tier of the parity check constraint of the tree. And we will see if they are satisfied. So what we are going to do is we are going to go

(Refer Slide Time 21:22)

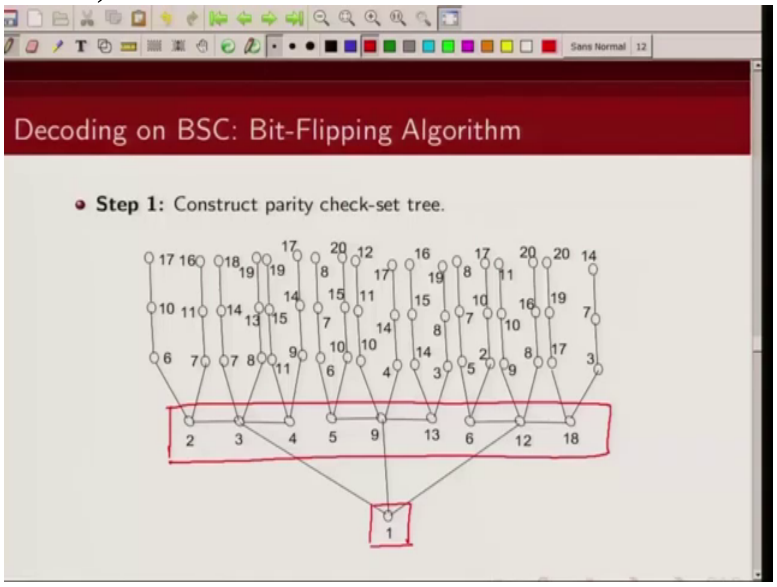


Decoding on BSC: Bit-Flipping Algorithm

- **Step 1:** Construct parity check-set tree.

in the first tier. So we are now going to look at these bits. And we are going to

(Refer Slide Time 21:27)

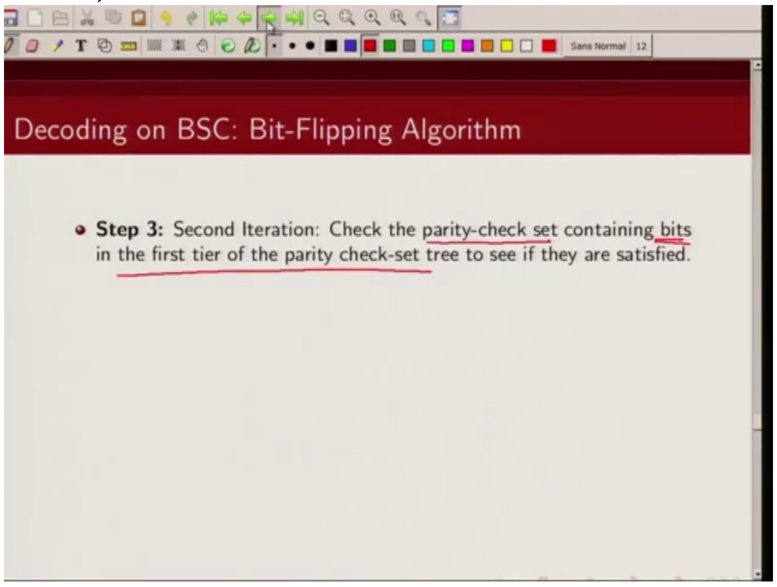


Decoding on BSC: Bit-Flipping Algorithm

- **Step 1:** Construct parity check-set tree.

see if the parity check sets involving these bits, 2, 3, 4, 5 if the parity check sets involving these bits, are they getting satisfied? If they are getting satisfied, fine. If they are again not getting satisfied then we will have to again flip the bits to make them satisfied. So this is how we are going to proceed. So let's look at second iteration.

(Refer Slide Time 21:52)

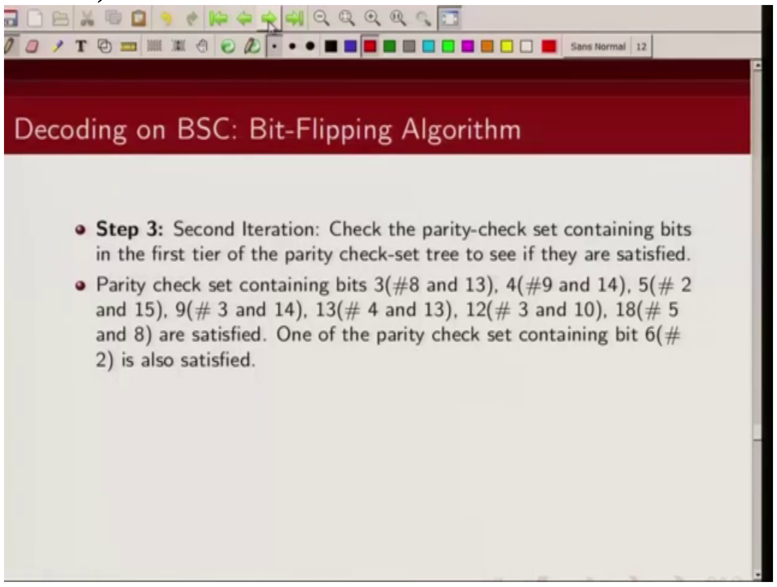


Decoding on BSC: Bit-Flipping Algorithm

- **Step 3:** Second Iteration: Check the parity-check set containing bits in the first tier of the parity check-set tree to see if they are satisfied.

Now what we are

(Refer Slide Time 21:55)

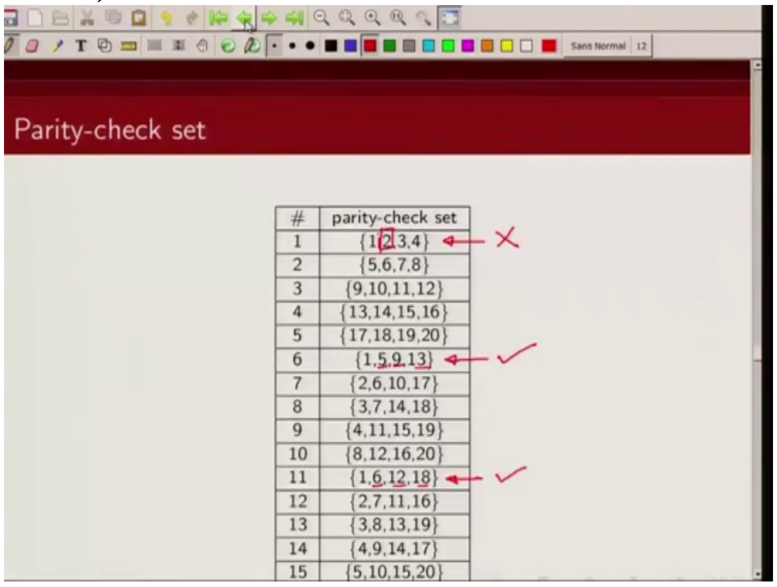


Decoding on BSC: Bit-Flipping Algorithm

- **Step 3:** Second Iteration: Check the parity-check set containing bits in the first tier of the parity check-set tree to see if they are satisfied.
- Parity check set containing bits 3(#8 and 13), 4(#9 and 14), 5(# 2 and 15), 9(# 3 and 14), 13(# 4 and 13), 12(# 3 and 10), 18(# 5 and 8) are satisfied. One of the parity check set containing bit 6(# 2) is also satisfied.

going to notice is that since bit 3 was not in error, bit 3, 4, 5, 9, 13, 12 and 18 these are, these will get satisfied.

(Refer Slide Time 22:10)

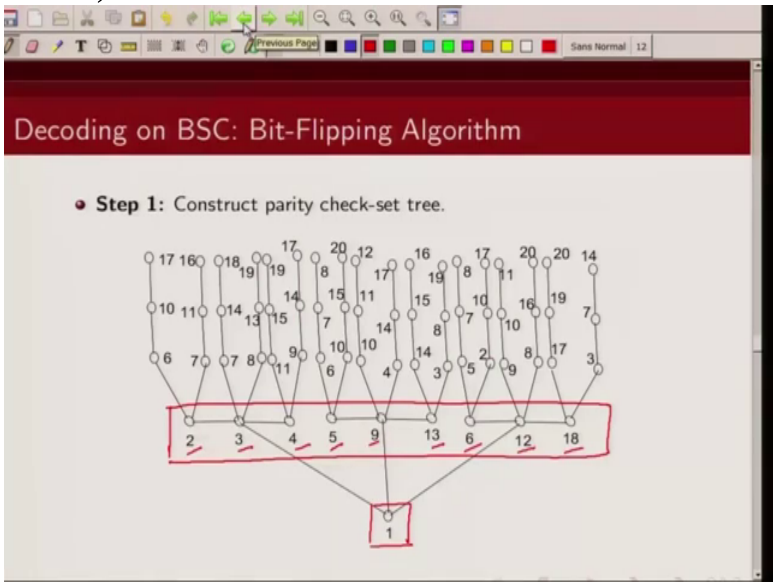


Parity-check set

#	parity-check set
1	{1,$\boxed{2}$,3,4} ← ✗
2	{5,6,7,8}
3	{9,10,11,12}
4	{13,14,15,16}
5	{17,18,19,20}
6	{1,5,9,13} ← ✓
7	{2,6,10,17}
8	{3,7,14,18}
9	{4,11,15,19}
10	{8,12,16,20}
11	{1,6,12,18} ← ✓
12	{2,7,11,16}
13	{3,8,13,19}
14	{4,9,14,17}
15	{5,10,15,20}

So if you go back to the parity check set diagram,

(Refer Slide Time 22:15)

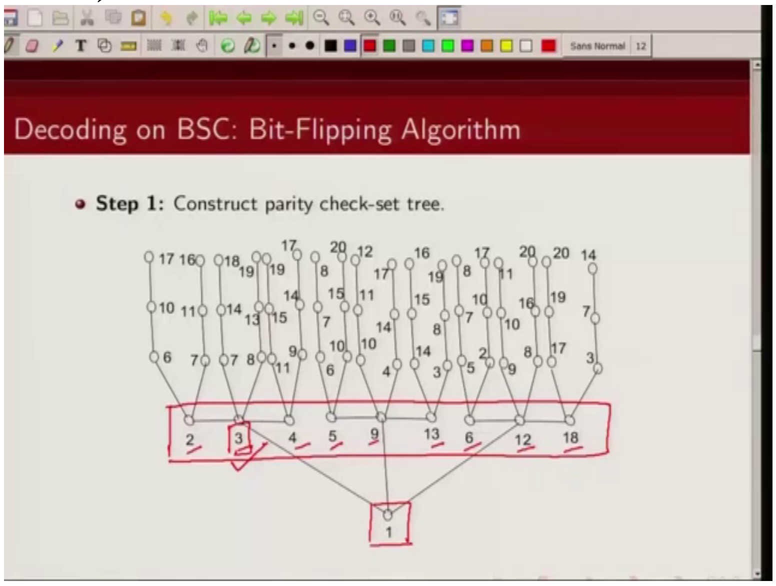


Decoding on BSC: Bit-Flipping Algorithm

- **Step 1:** Construct parity check-set tree.

this was not in error and this involves 18, 14, 7, none of these was in error. Similarly this involves 8, 13, 19, these were not in error. So all the parity check sets

(Refer Slide Time 22:28)

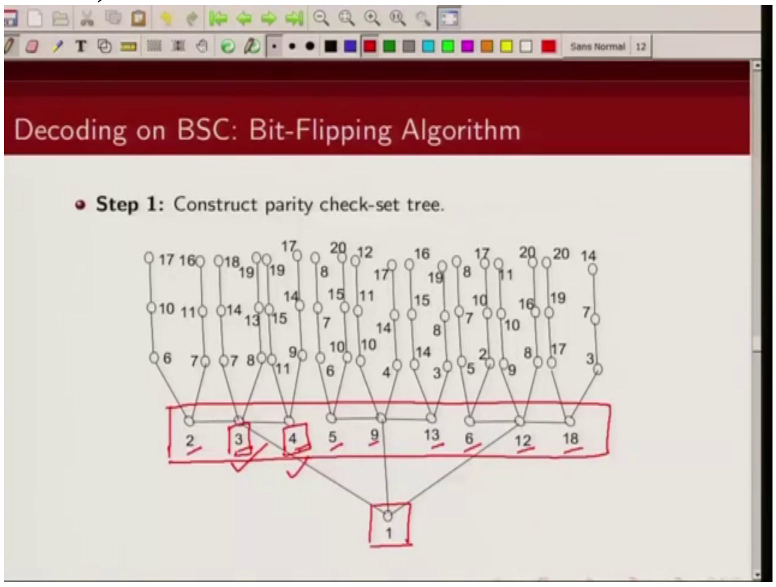


Decoding on BSC: Bit-Flipping Algorithm

- **Step 1:** Construct parity check-set tree.

containing 3 will be satisfied. What about 4? 4, 11, 15, 19 they were not received in error. Similarly 4, 9, 14, 17 were not received in error. So these parity check sets were satisfied.

(Refer Slide Time 22:41)

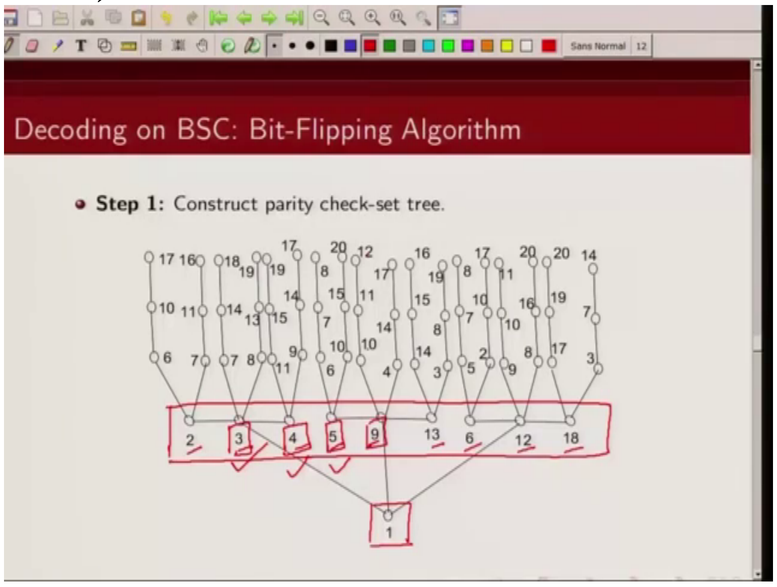


Decoding on BSC: Bit-Flipping Algorithm

- **Step 1:** Construct parity check-set tree.

5, 6, 7, 8 these will be satisfied. 5, 10, 15, 20 again these will be satisfied. Similarly 9, 10, 11, 12 no error in any of the bits. So this parity check equation will be satisfied. Similarly 17, 14, 4 so this will be satisfied.

(Refer Slide Time 22:59)

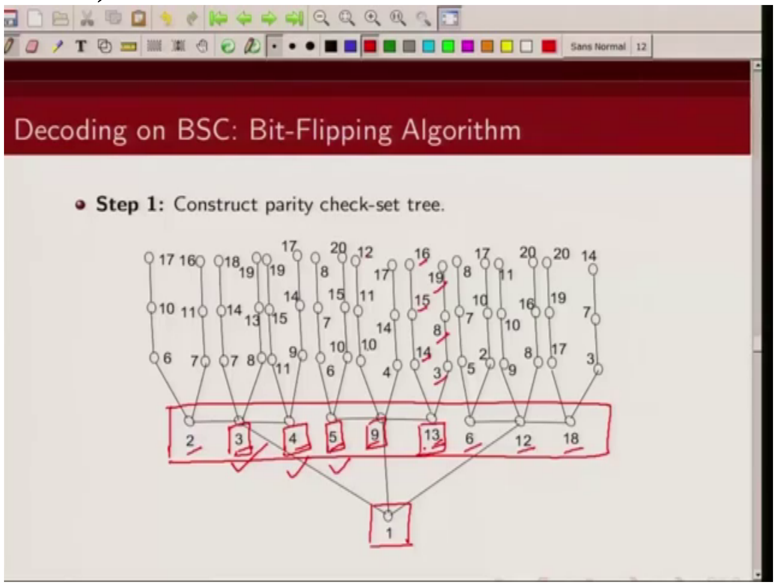


Decoding on BSC: Bit-Flipping Algorithm

- **Step 1:** Construct parity check-set tree.

13, this has 16, 15, 14 and 13, none of the bits are in error. So this will be satisfied. Then 3, 8, 19, 13 again this will be satisfied. What about

(Refer Slide Time 23:14)

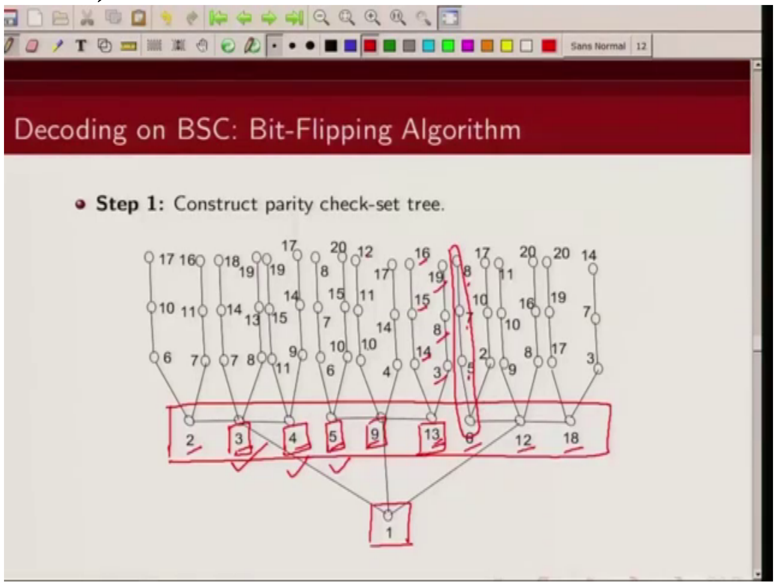


Decoding on BSC: Bit-Flipping Algorithm

- **Step 1:** Construct parity check-set tree.

this? 6, 5, 7 and 8 none of those bits are in error so this will be satisfied. This will be satisfied.

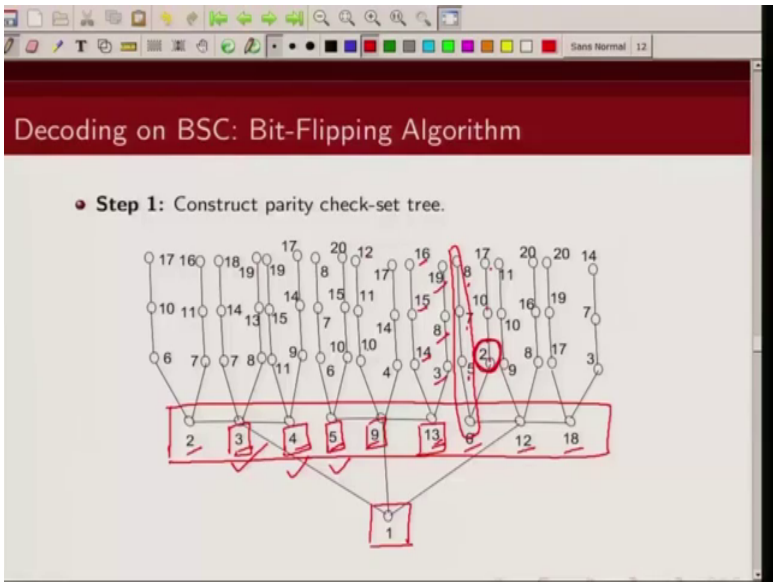
(Refer Slide Time 23:23)


Decoding on BSC: Bit-Flipping Algorithm

- **Step 1:** Construct parity check-set tree.

But what about this? 6, 2, 10 and 17; now this bit is in error.

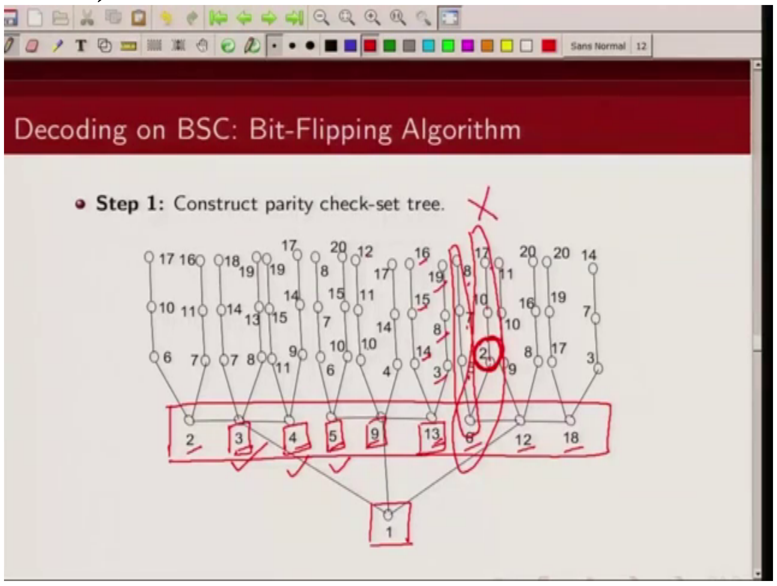
(Refer Slide Time 23:30)


Decoding on BSC: Bit-Flipping Algorithm

- **Step 1:** Construct parity check-set tree.

This bit is in error. So this particular parity check equation will not

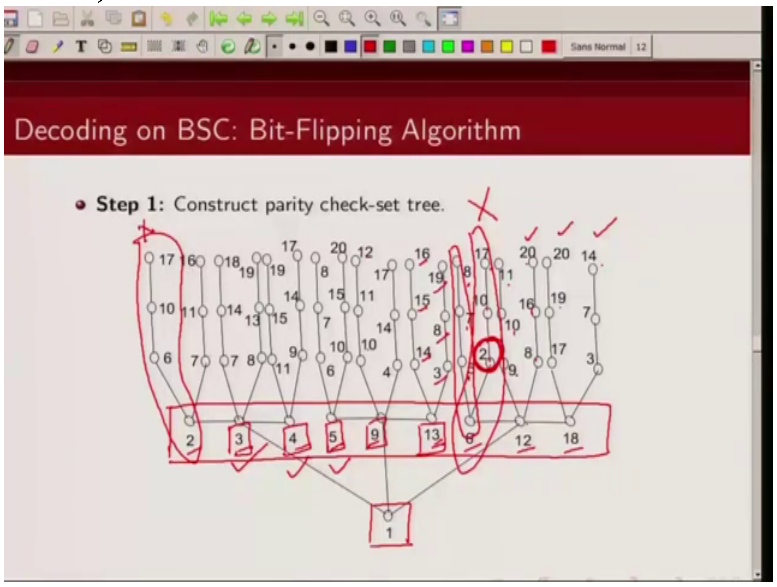
(Refer Slide Time 23:37)


Decoding on BSC: Bit-Flipping Algorithm

- **Step 1:** Construct parity check-set tree.

get satisfied. What about this, 12, 9, 10, 11, this will be satisfied. 12, 8, 16, 20 this will be satisfied. Similarly 18, 17, 19, 20 this will be satisfied. 18, 3, 7, 14 this is also satisfied. What about 2? This will be not satisfied.

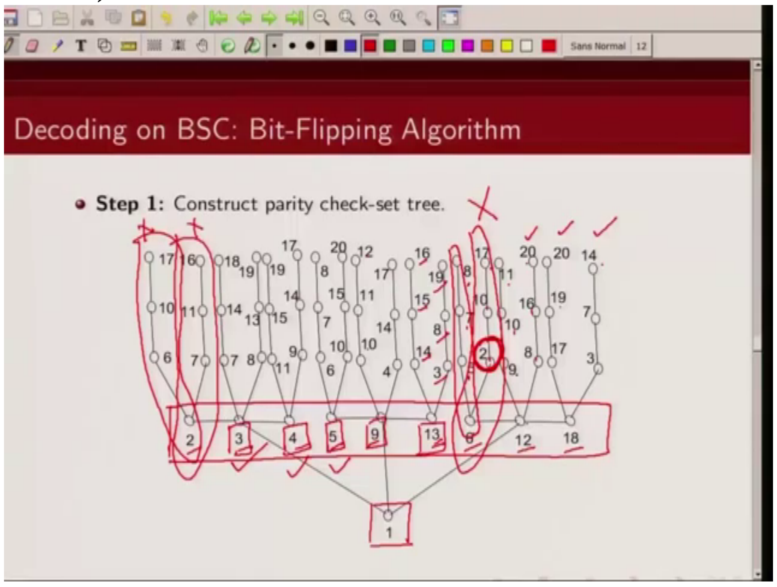
(Refer Slide Time 24:02)


Decoding on BSC: Bit-Flipping Algorithm

- **Step 1:** Construct parity check-set tree.

And similarly this will be not satisfied.

(Refer Slide Time 24:06)



Decoding on BSC: Bit-Flipping Algorithm

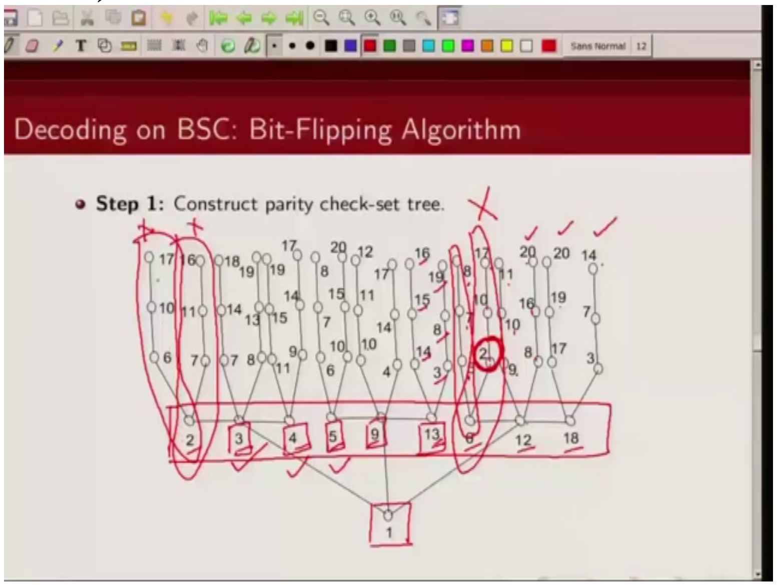
- **Step 1:** Construct parity check-set tree.

So what we can see is the parity check sets involving 2 is not

(Refer Slide Time 24:14)

getting satisfied. Because here

(Refer Slide Time 24:16)


Decoding on BSC: Bit-Flipping Algorithm

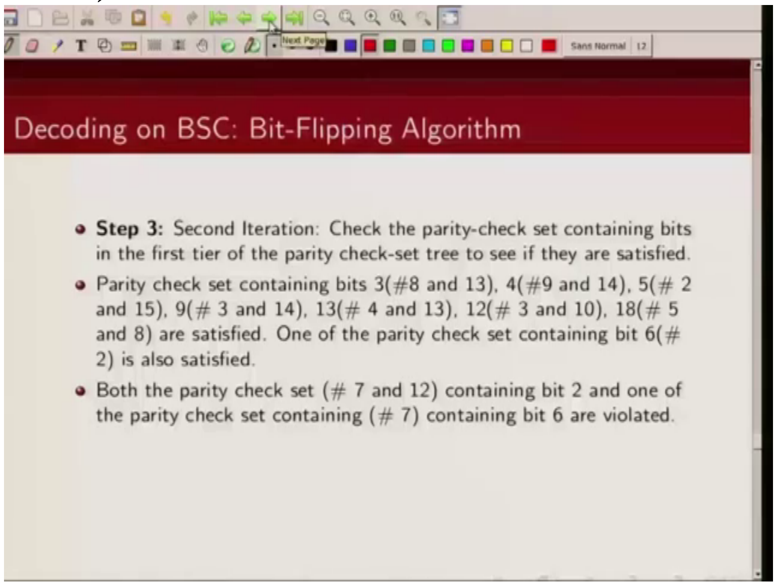
- **Step 1:** Construct parity check-set tree.

there was 2, here there was 2, here there was 2, it is not getting satisfied. Two of them are not getting satisfied. And the third one is which involves 1, 1, 2, 3, 4 this is getting satisfied. This is also not getting satisfied because 1 was corrected, 3 and 4 are correct so this is also not getting satisfied. So what we notice is parity check sets containing 2 are not

(Refer Slide Time 24:44)

getting satisfied.

(Refer Slide Time 24:46)

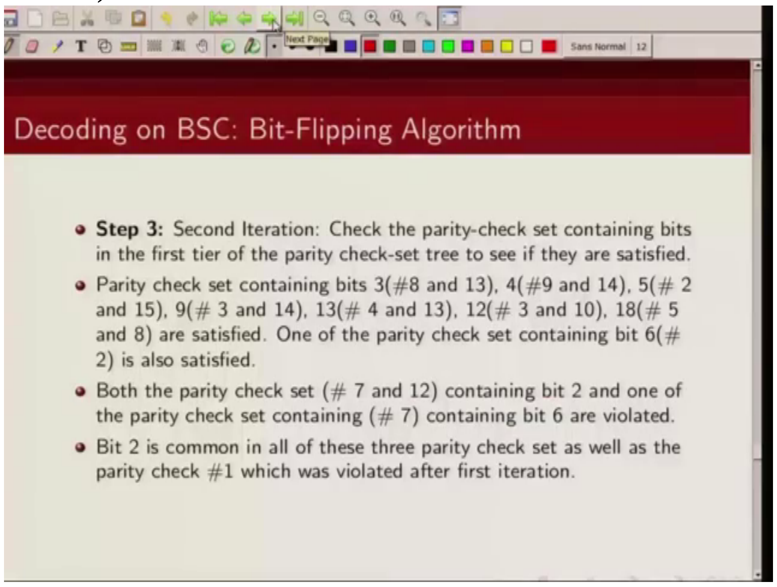


Decoding on BSC: Bit-Flipping Algorithm

- **Step 3:** Second Iteration: Check the parity-check set containing bits in the first tier of the parity check-set tree to see if they are satisfied.
- Parity check set containing bits 3(#8 and 13), 4(#9 and 14), 5(# 2 and 15), 9(# 3 and 14), 13(# 4 and 13), 12(# 3 and 10), 18(# 5 and 8) are satisfied. One of the parity check set containing bit 6(# 2) is also satisfied.
- Both the parity check set (# 7 and 12) containing bit 2 and one of the parity check set containing (# 7) containing bit 6 are violated.

In that case what do we do, we are going to flip the bits. So parity check sets containing bit 2 are not getting satisfied. Again one of the constraints containing bit 6 has bit number 2 and it was not getting satisfied.

(Refer Slide Time 25:05)

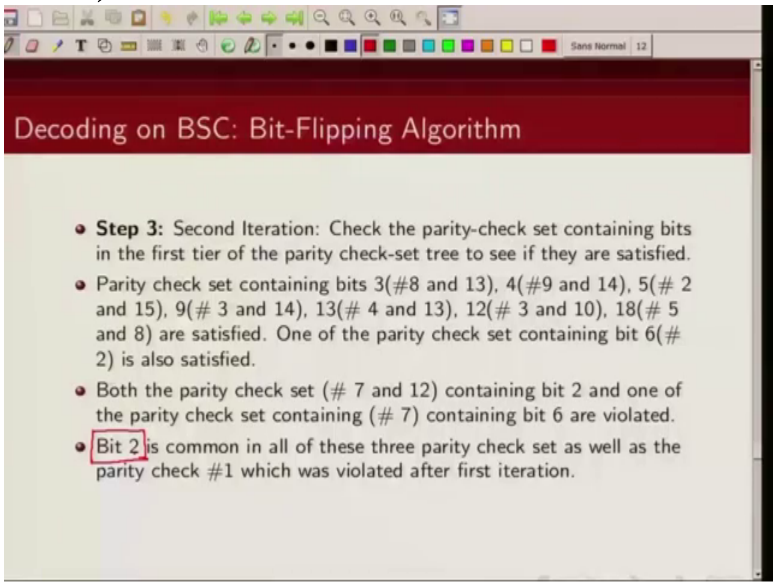


Decoding on BSC: Bit-Flipping Algorithm

- **Step 3:** Second Iteration: Check the parity-check set containing bits in the first tier of the parity check-set tree to see if they are satisfied.
- Parity check set containing bits 3(#8 and 13), 4(#9 and 14), 5(# 2 and 15), 9(# 3 and 14), 13(# 4 and 13), 12(# 3 and 10), 18(# 5 and 8) are satisfied. One of the parity check set containing bit 6(# 2) is also satisfied.
- Both the parity check set (# 7 and 12) containing bit 2 and one of the parity check set containing (# 7) containing bit 6 are violated.
- Bit 2 is common in all of these three parity check set as well as the parity check #1 which was violated after first iteration.

So bit 2 was common in all the

(Refer Slide Time 25:10)



Decoding on BSC: Bit-Flipping Algorithm

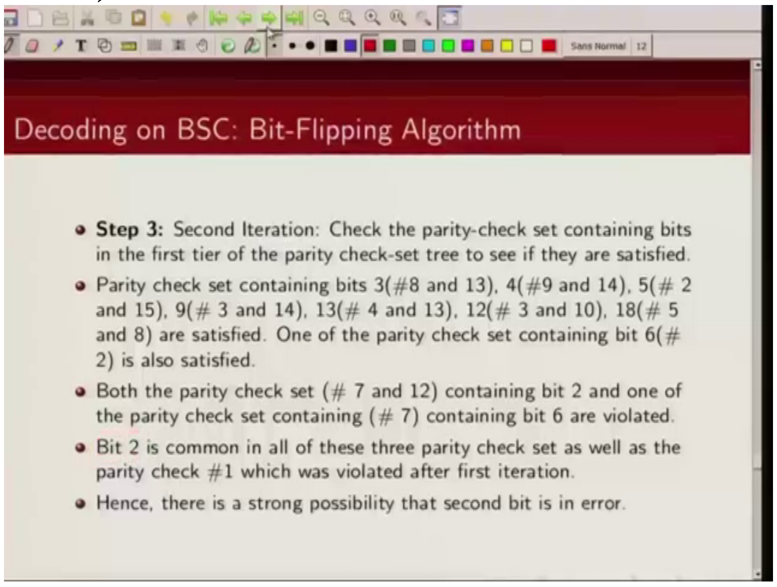
- **Step 3:** Second Iteration: Check the parity-check set containing bits in the first tier of the parity check-set tree to see if they are satisfied.
- Parity check set containing bits 3(#8 and 13), 4(#9 and 14), 5(# 2 and 15), 9(# 3 and 14), 13(# 4 and 13), 12(# 3 and 10), 18(# 5 and 8) are satisfied. One of the parity check set containing bit 6(# 2) is also satisfied.
- Both the parity check set (# 7 and 12) containing bit 2 and one of the parity check set containing (# 7) containing bit 6 are violated.
- $\boxed{\text{Bit 2}}$ is common in all of these three parity check set as well as the parity check #1 which was violated after first iteration.

parity check sets which

(Refer Slide Time 25:12)

were not getting satisfied. So

(Refer Slide Time 25:15)

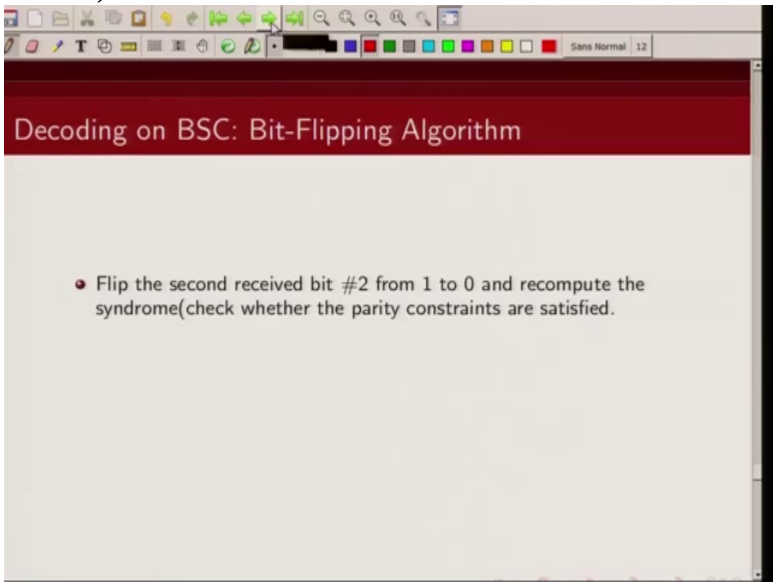


Decoding on BSC: Bit-Flipping Algorithm

- **Step 3:** Second Iteration: Check the parity-check set containing bits in the first tier of the parity check-set tree to see if they are satisfied.
- Parity check set containing bits 3(#8 and 13), 4(#9 and 14), 5(# 2 and 15), 9(# 3 and 14), 13(# 4 and 13), 12(# 3 and 10), 18(# 5 and 8) are satisfied. One of the parity check set containing bit 6(# 2) is also satisfied.
- Both the parity check set (# 7 and 12) containing bit 2 and one of the parity check set containing (# 7) containing bit 6 are violated.
- Bit 2 is common in all of these three parity check set as well as the parity check #1 which was violated after first iteration.
- Hence, there is a strong possibility that second bit is in error.

what we do is we think that second bit is in error and we are going

(Refer Slide Time 25:20)



Decoding on BSC: Bit-Flipping Algorithm

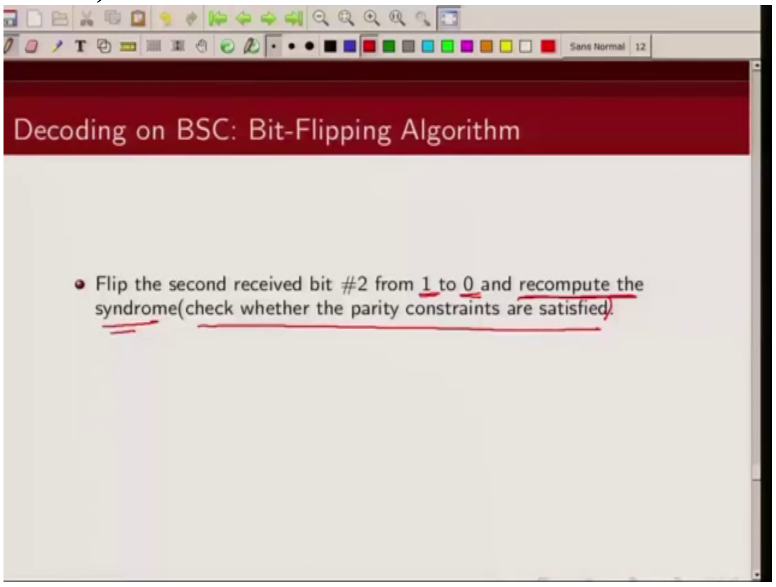
- Flip the second received bit #2 from 1 to 0 and recompute the syndrome(check whether the parity constraints are satisfied.

to flip the second bit. So we flip the, second bit was 1, we flip it to 0 and we are going to recompute all the syndrome. Now we notice that parity check constraints are satisfied. Because the 2 bit was in error; after we have flipped

(Refer Slide Time 25:38)

it, we will see that all the bits involving 2, all the parity check sets

(Refer Slide Time 25:45)

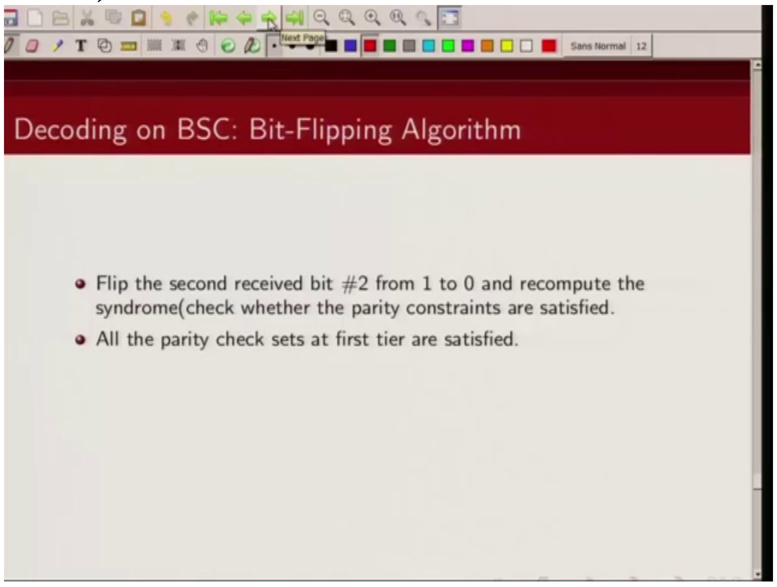


Decoding on BSC: Bit-Flipping Algorithm

- Flip the second received bit #2 from 1 to 0 and recompute the syndrome(check whether the parity constraints are satisfied).

involving bit 2 are now getting satisfied

(Refer Slide Time 25:47)



Decoding on BSC: Bit-Flipping Algorithm

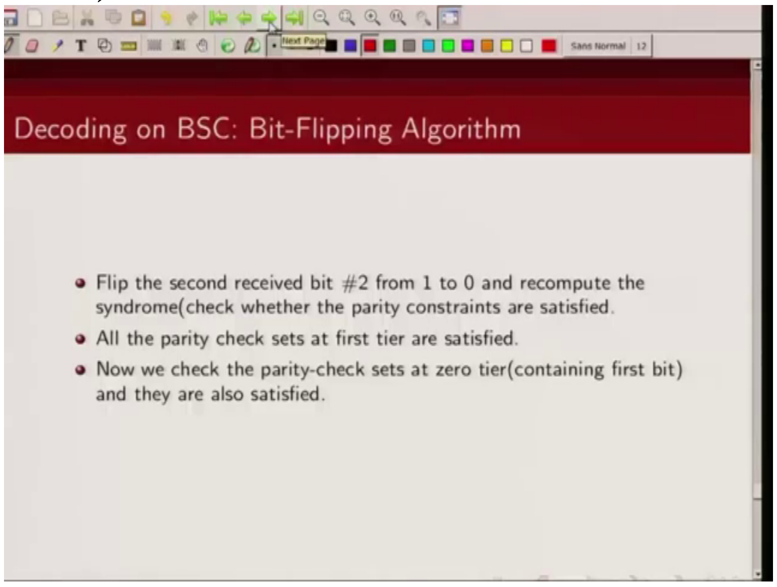
- Flip the second received bit #2 from 1 to 0 and recompute the syndrome(check whether the parity constraints are satisfied.
- All the parity check sets at first tier are satisfied.

and hence we are able to correct all errors. So if there are 2 errors, you can see that

(Refer Slide Time 25:55)

one iteration was not enough, we had to go

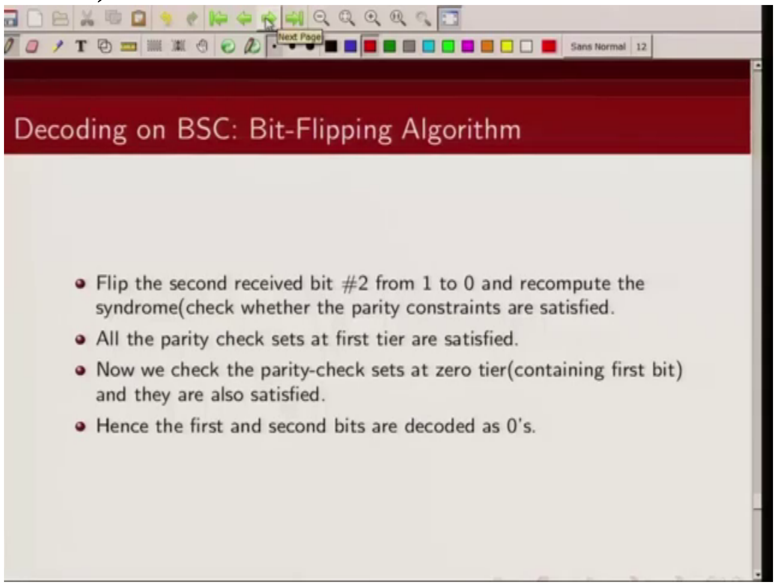
(Refer Slide Time 25:58)


Decoding on BSC: Bit-Flipping Algorithm

- Flip the second received bit #2 from 1 to 0 and recompute the syndrome(check whether the parity constraints are satisfied.
- All the parity check sets at first tier are satisfied.
- Now we check the parity-check sets at zero tier(containing first bit) and they are also satisfied.

for 2 iterations. Ok. Now we go back and check at 0 tier and we see that at 0 tier also all the parity check sets are satisfied. So hence we have successfully

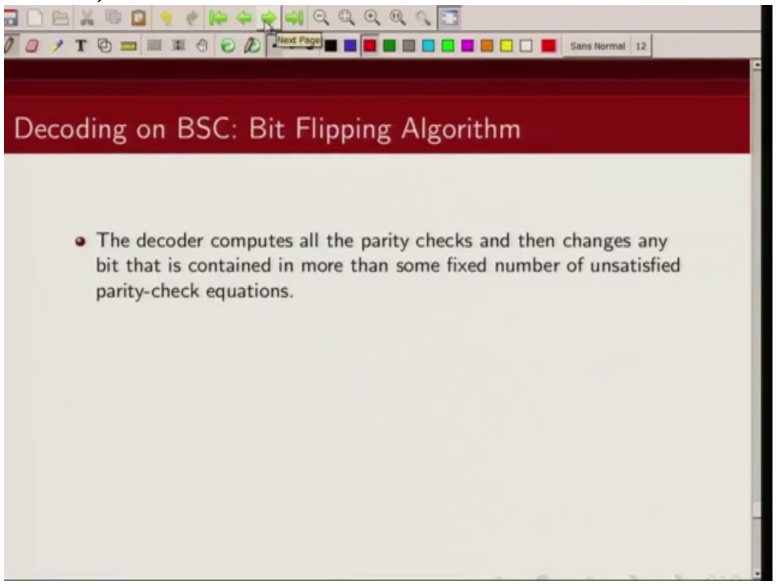
(Refer Slide Time 26:12)


Decoding on BSC: Bit-Flipping Algorithm

- Flip the second received bit #2 from 1 to 0 and recompute the syndrome(check whether the parity constraints are satisfied.
- All the parity check sets at first tier are satisfied.
- Now we check the parity-check sets at zero tier(containing first bit) and they are also satisfied.
- Hence the first and second bits are decoded as 0's.

decoded the first and second bit to be zeroes. And all other bits were received

(Refer Slide Time 26:18)



Decoding on BSC: Bit Flipping Algorithm

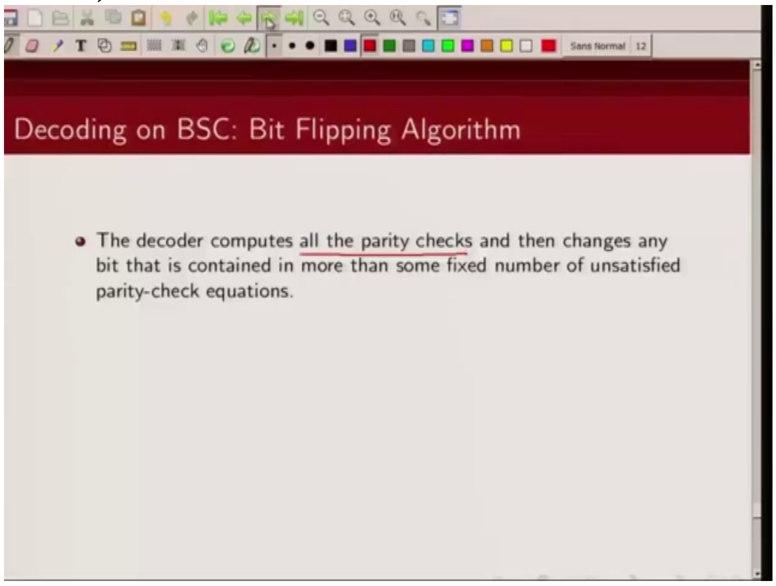
- The decoder computes all the parity checks and then changes any bit that is contained in more than some fixed number of unsatisfied parity-check equations.

in correctly so there is no error. So then what the decoder does? It basically computes all the parity check sets and then changes any bits that are contained in more than a fixed number of unsatisfied parity check equations and then we recompute the syndrome, recompute the parity check constraints and hopefully by flipping the bits which are

(Refer Slide Time 26:44)

common in most of the parity check constraints that are getting violated, we will be able to

(Refer Slide Time 26:51)

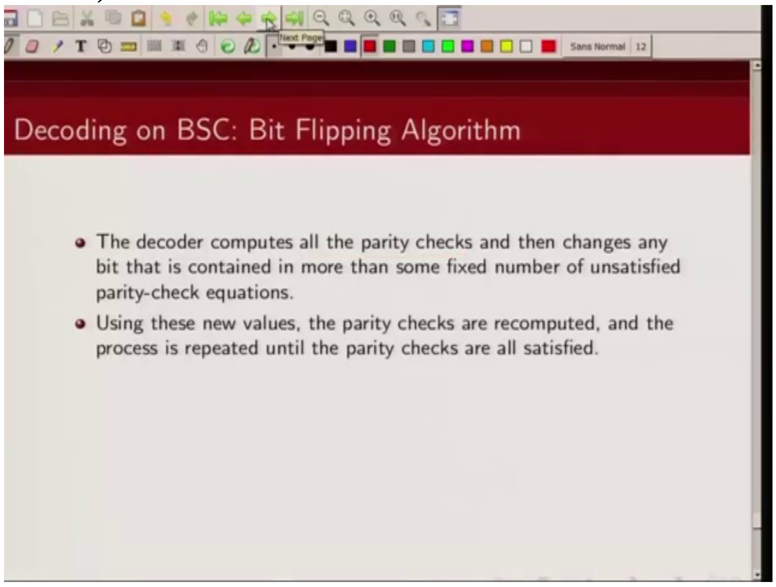


Decoding on BSC: Bit Flipping Algorithm

- The decoder computes all the parity checks and then changes any bit that is contained in more than some fixed number of unsatisfied parity-check equations.

finally correct

(Refer Slide Time 26:52)

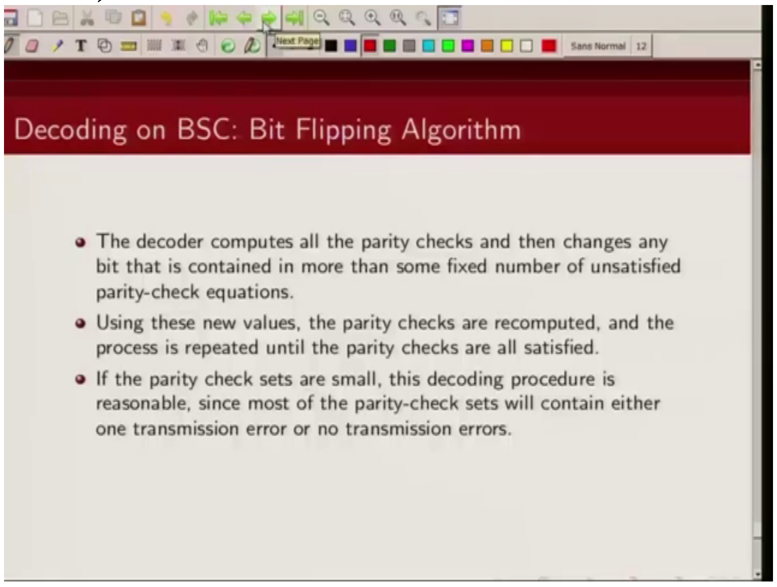


Decoding on BSC: Bit Flipping Algorithm

- The decoder computes all the parity checks and then changes any bit that is contained in more than some fixed number of unsatisfied parity-check equations.
- Using these new values, the parity checks are recomputed, and the process is repeated until the parity checks are all satisfied.

those errors. And each time after we flip the bits, we recompute the syndrome; check whether the syndromes are satisfied. When all the syndromes are getting satisfied we have successfully

(Refer Slide Time 27:05)



Decoding on BSC: Bit Flipping Algorithm

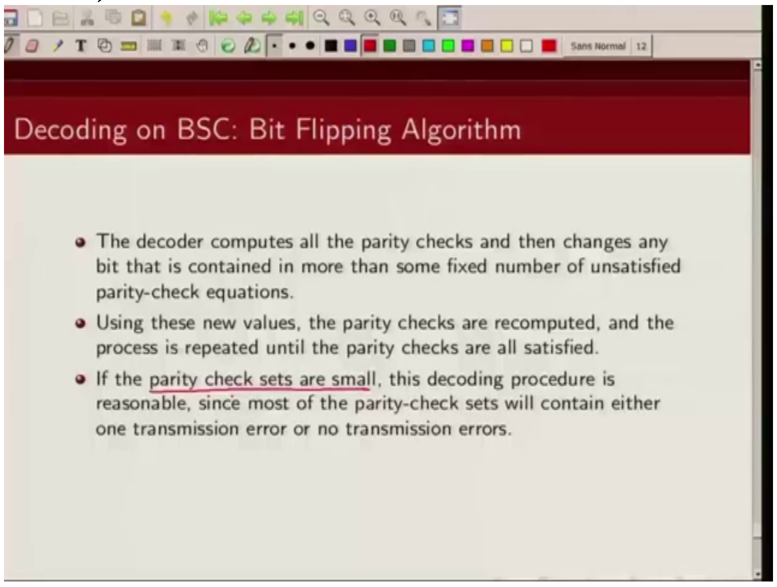
- The decoder computes all the parity checks and then changes any bit that is contained in more than some fixed number of unsatisfied parity-check equations.
- Using these new values, the parity checks are recomputed, and the process is repeated until the parity checks are all satisfied.
- If the parity check sets are small, this decoding procedure is reasonable, since most of the parity-check sets will contain either one transmission error or no transmission errors.

decoded the L D P C code. And since this size of parity check set is small, this decoding is reasonable, it is not very hard and we can also do this

(Refer Slide Time 27:18)

process parallely. We can have a, for each parity check tree, for each of these bits and we can

(Refer Slide Time 27:24)

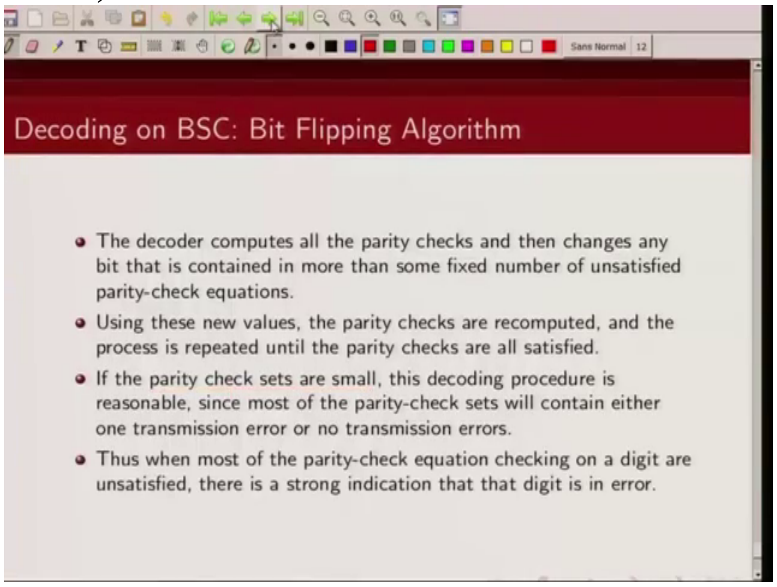


Decoding on BSC: Bit Flipping Algorithm

- The decoder computes all the parity checks and then changes any bit that is contained in more than some fixed number of unsatisfied parity-check equations.
- Using these new values, the parity checks are recomputed, and the process is repeated until the parity checks are all satisfied.
- If the parity check sets are small, this decoding procedure is reasonable, since most of the parity-check sets will contain either one transmission error or no transmission errors.

try to do this decoding in a parallel fashion.

(Refer Slide Time 27:27)



Decoding on BSC: Bit Flipping Algorithm

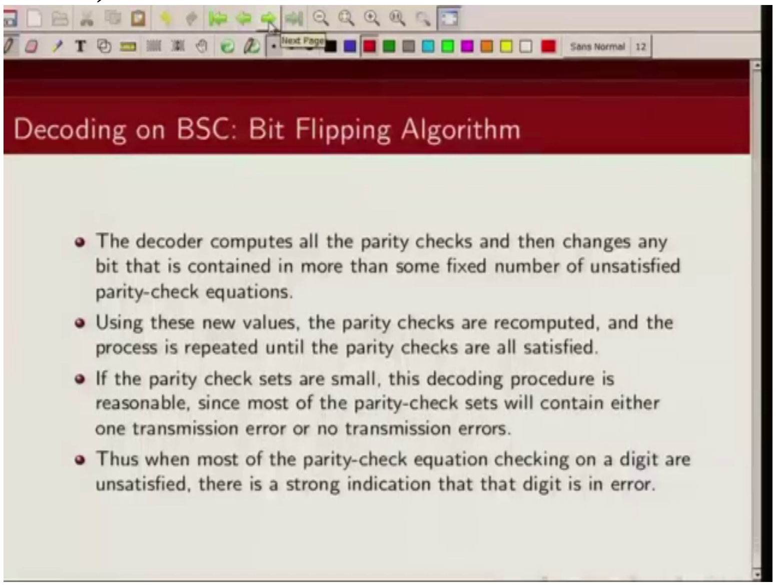
- The decoder computes all the parity checks and then changes any bit that is contained in more than some fixed number of unsatisfied parity-check equations.
- Using these new values, the parity checks are recomputed, and the process is repeated until the parity checks are all satisfied.
- If the parity check sets are small, this decoding procedure is reasonable, since most of the parity-check sets will contain either one transmission error or no transmission errors.
- Thus when most of the parity-check equation checking on a digit are unsatisfied, there is a strong indication that that digit is in error.

And again this relies on the logic that a bit that is appearing in most of the unsatisfied parity check equation that is most likely culprit.

(Refer Slide Time 27:42)

That's the one which is appearing most likely to be in error. And we are flipping that bit to correct it,

(Refer Slide Time 27:49)



Decoding on BSC: Bit Flipping Algorithm

- The decoder computes all the parity checks and then changes any bit that is contained in more than some fixed number of unsatisfied parity-check equations.
- Using these new values, the parity checks are recomputed, and the process is repeated until the parity checks are all satisfied.
- If the parity check sets are small, this decoding procedure is reasonable, since most of the parity-check sets will contain either one transmission error or no transmission errors.
- Thus when most of the parity-check equation checking on a digit are unsatisfied, there is a strong indication that that digit is in error.

Ok. So with this I am going to conclude our discussion on decoding

(Refer Slide Time 27:56)

of L D P C codes over a binary symmetric channel. We will continue the discussion on decoding of L D P C codes in the next lecture by discussing the probabilistic decoding algorithm, thank you.