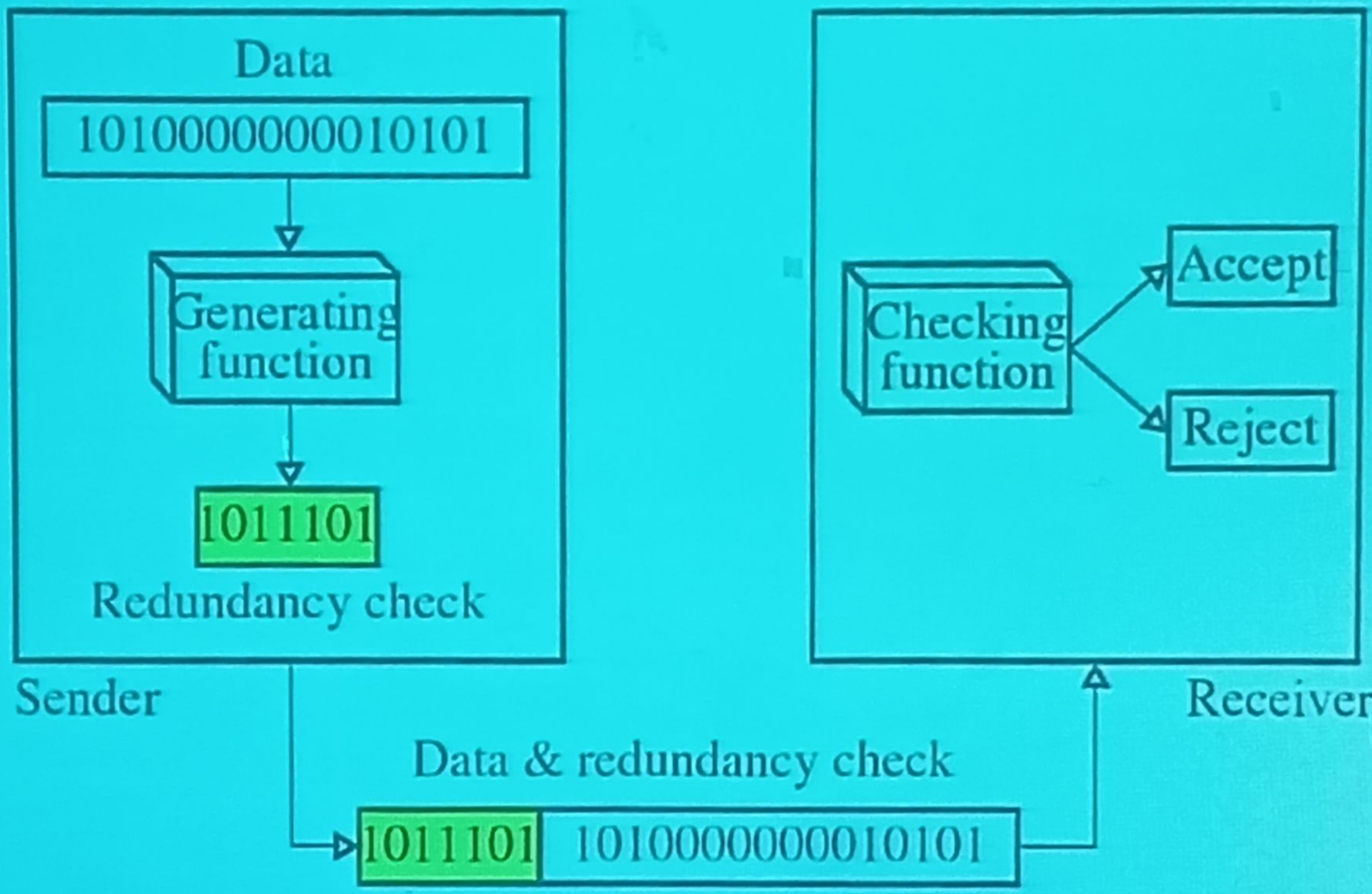
Error detection

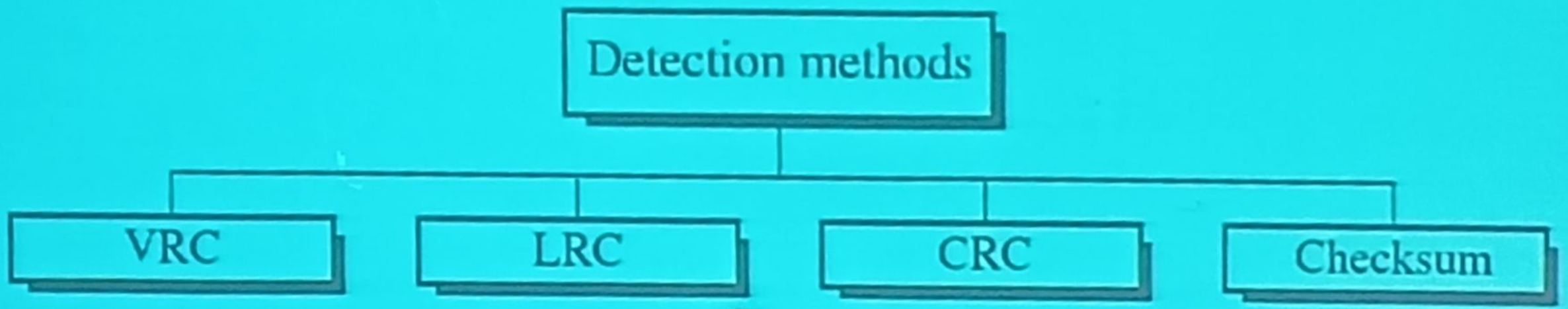
Error detection means to decide whether the received data is correct or not without having a copy of the original message.

Error detection **uses the concept of redundancy, which means** adding extra bits for detecting errors at the destination.

Redundancy


Data
1010000000010101
Generating function
1011101
Redundancy check
Sender
Checking function
Accept
Reject
Receiver
Data & redundancy check
1011101
1010000000010101

Four types of redundancy checks are used in data communications

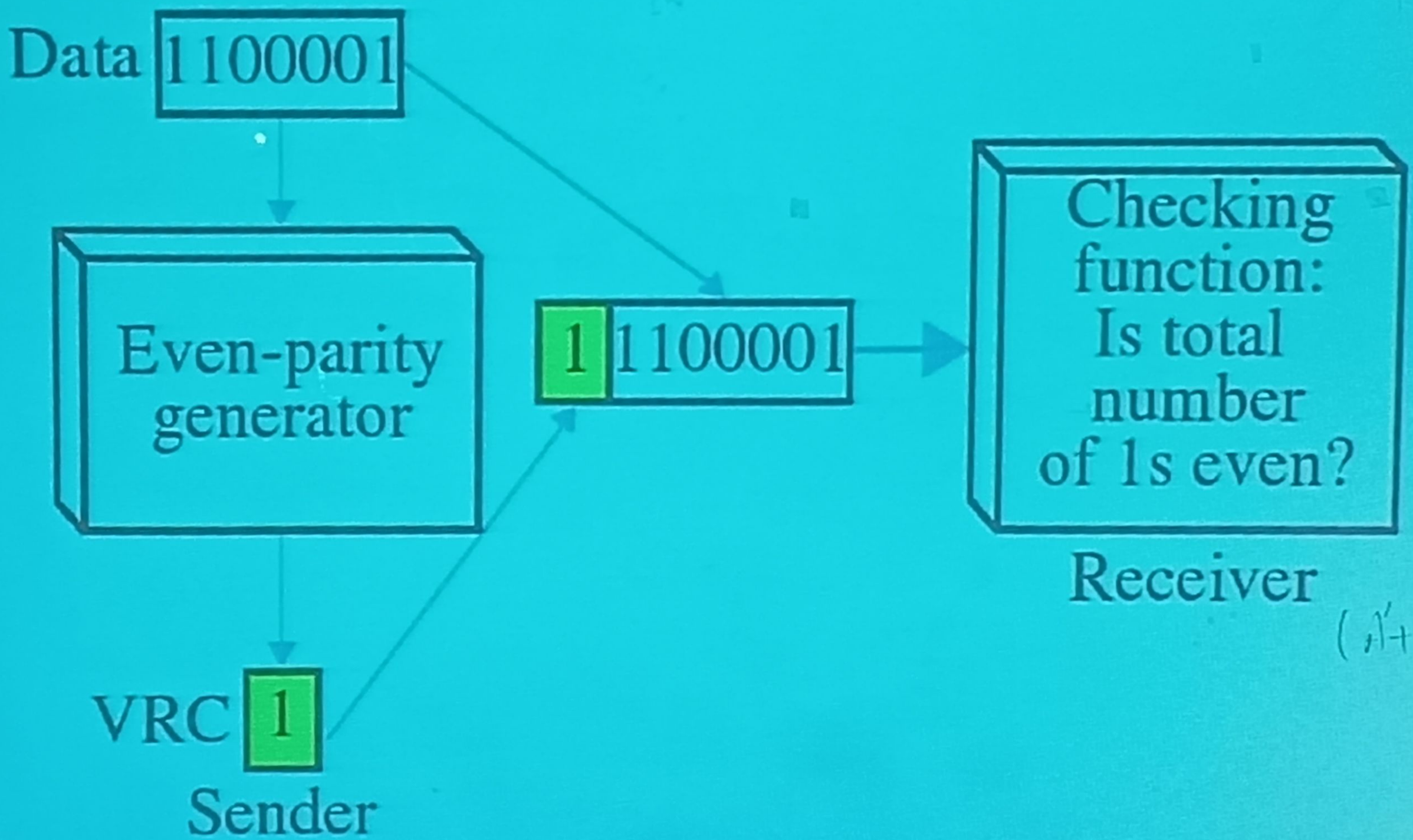


Vertical Redundancy Check

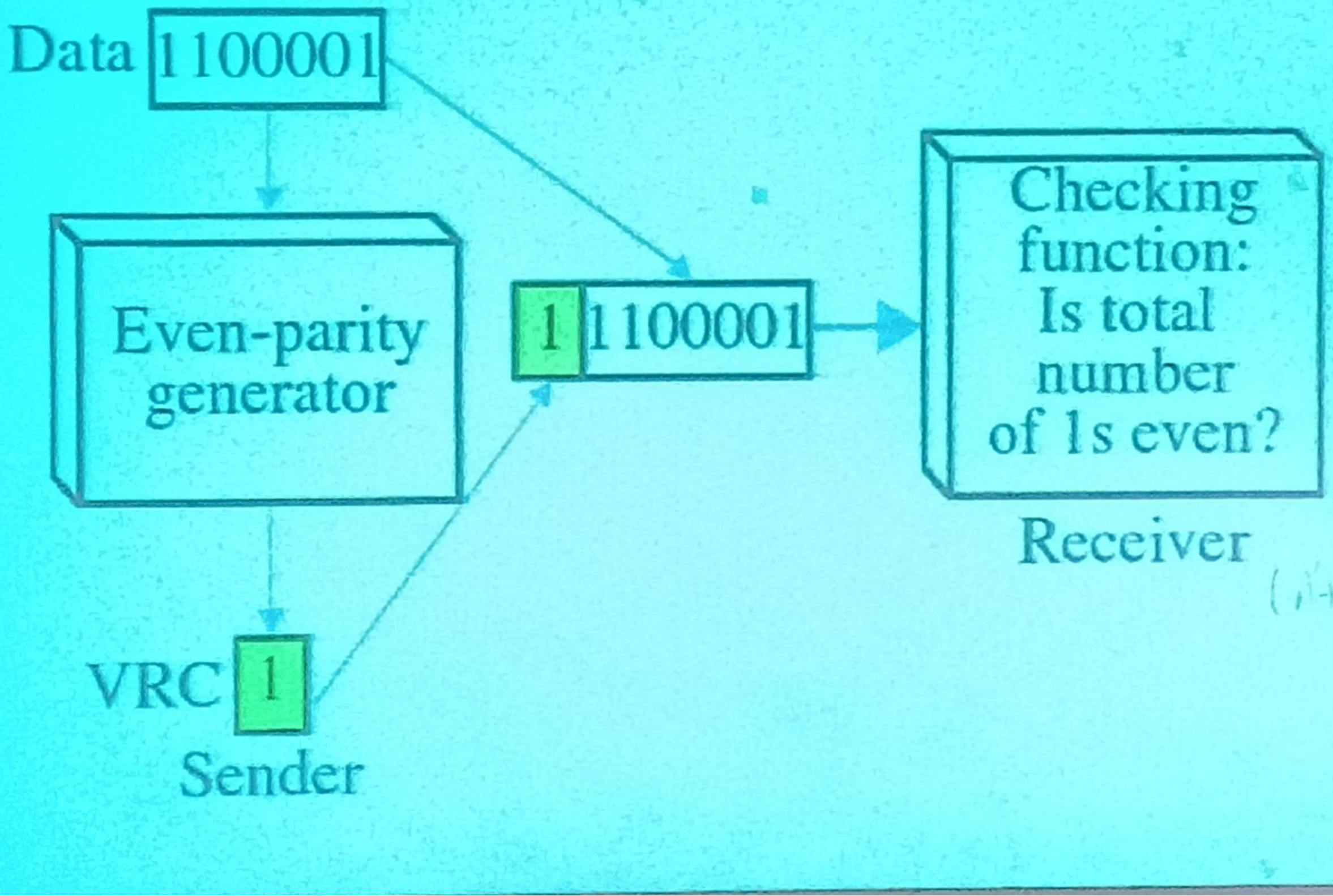
Longitudinal Redundancy Check

Cyclic Redundancy Check

Vertical Redundancy Check
VRC

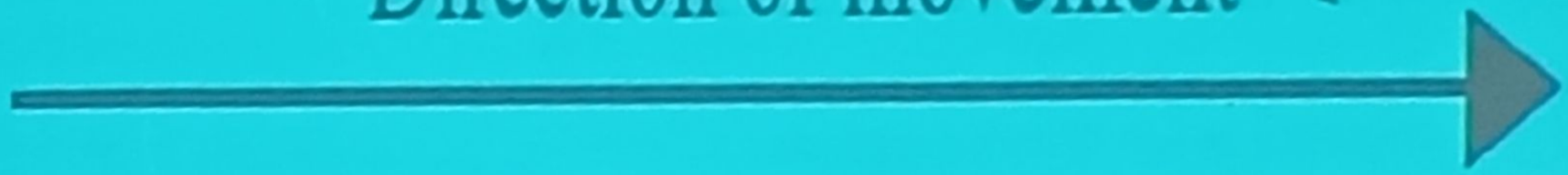


Vertical Redundancy Check
VRC



Longitudinal Redundancy Check
LRC

Direction of movement



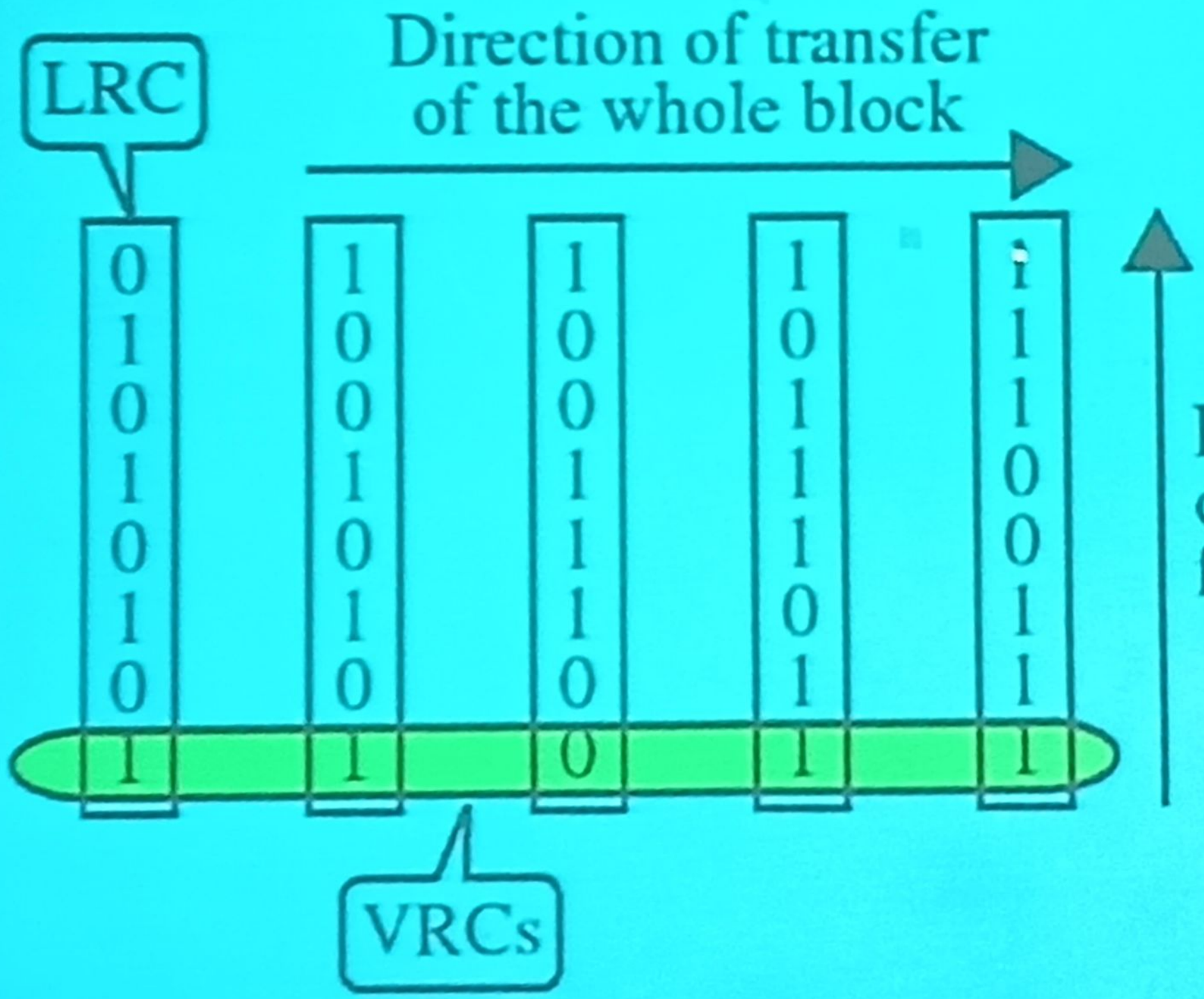
10101010	10101001	00111001	11011101	11100111

LRC

Data

$(A'+B')$

VRC and LRC

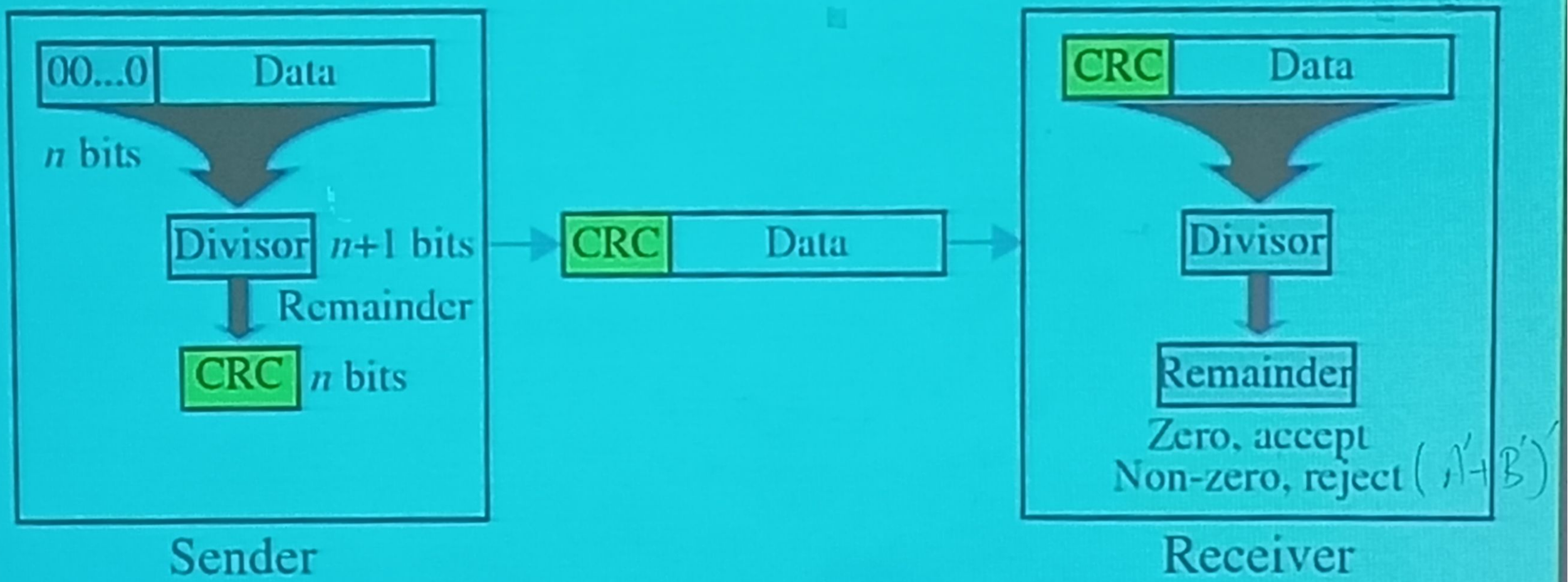

LRC
Direction of transfer of the whole block
0 1 0 1 0 1 0 1
1 0 0 1 0 1 0 0
1 0 0 1 1 1 1 0
1 0 1 1 1 0 0 1
1 1 0 0 1 1 0 1
1 1 0 0 1 1 0 1
VRCs
Direction of transfer for each unit

Performance

- LRC increases the likelihood of detecting burst errors.
- If two bits in one data units are damaged and two bits in exactly the same positions in another data unit are also damaged, the LRC checker will not detect an error.

$(A'+B')'$

Cyclic Redundancy Check
CRC



For example, if data to be transmitted is 1001 and predetermined divisor is 1011. The procedure given below is used:

Sol:

1. String of 3 zeroes is appended to 1001 as divisor is of 4 bits. Now newly formed data is 1001000.
2. Data unit 1001000 is divided by 1011.

```
Divisor   1011 | 1 0 0 1 0 0 0 | 1010 Quotient
              | 1 0 1 1       |
              |---------------|
              |   0 1 0 0     |
              |   0 0 0 0     |
              |---------------|
              |     1 0 0 0   |
              |     1 0 1 1   |
              |---------------|
              |       0 1 1 0 |
              |       0 0 0 0 |
              |---------------|
                        1 1 0 Remainder
```

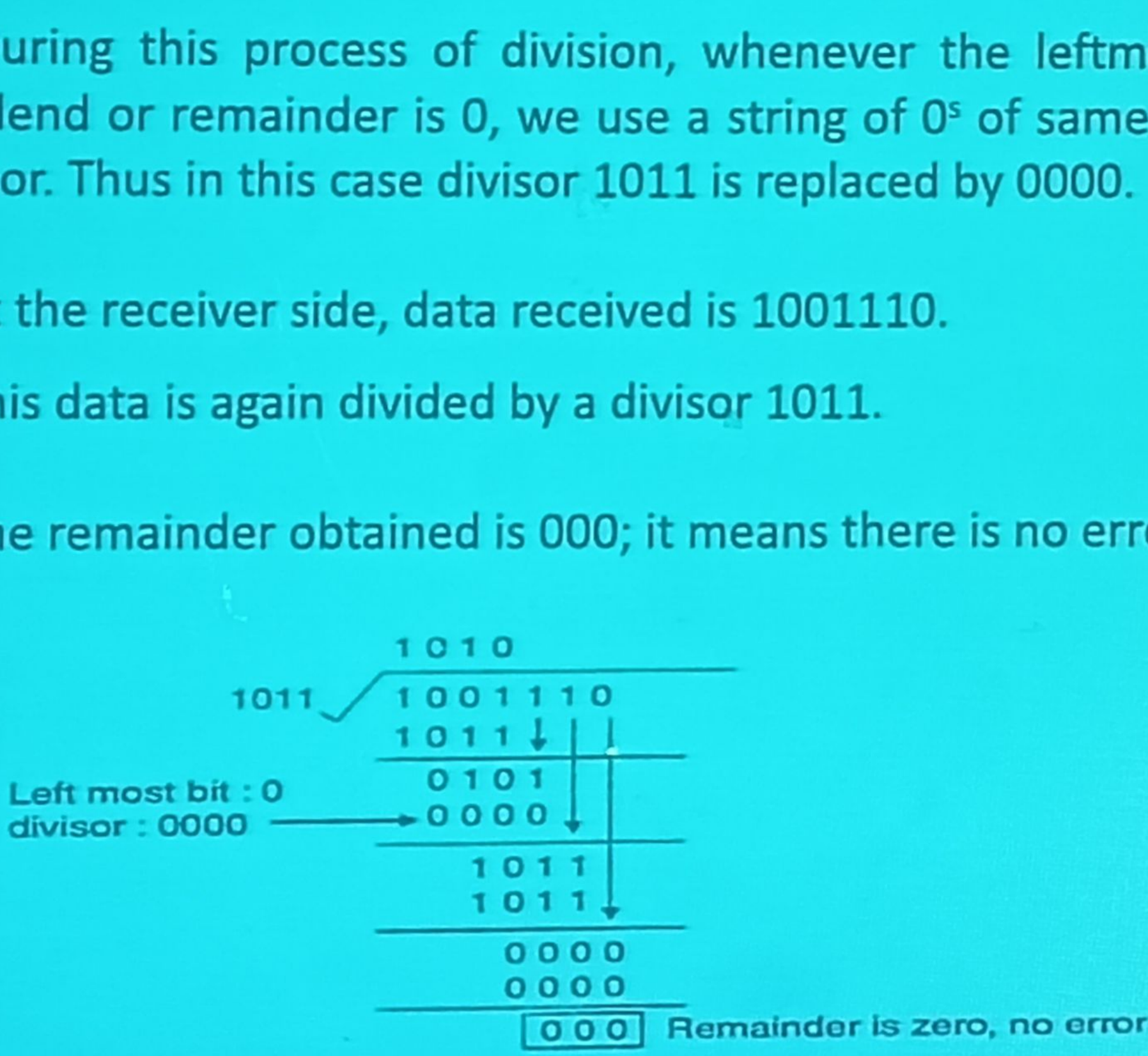
Data: 1001 CRC: 110 (Remainder)

3. During this process of division, whenever the leftmost bit of dividend or remainder is 0, we use a string of 0^s of same length as divisor. Thus in this case divisor 1011 is replaced by 0000.

4. At the receiver side, data received is 1001110.

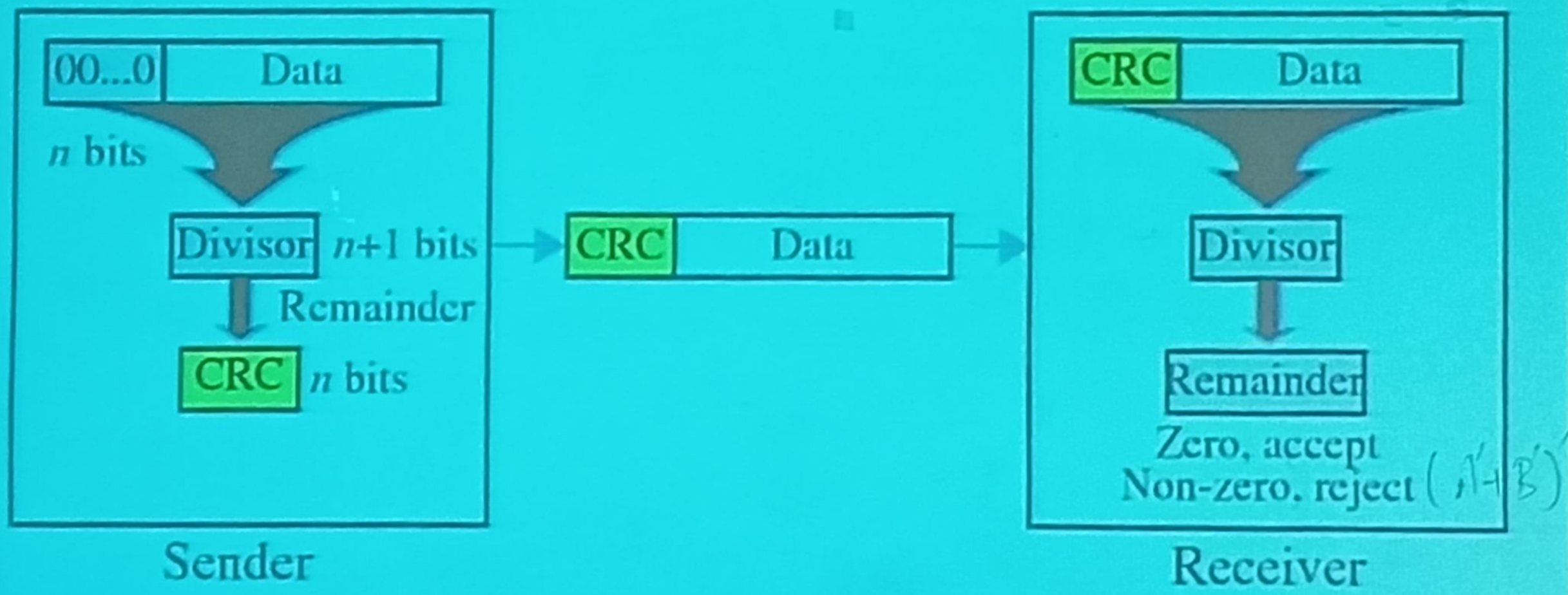
5. This data is again divided by a divisor 1011.

6. The remainder obtained is 000; it means there is no error.



CRC decoded (binary division)

Cyclic Redundancy Check
CRC



Cyclic Redundancy Check
CRC

- At the sender side, the data unit to be transmitted is divided by a predetermined divisor (binary number) in order to obtain the remainder. This remainder is called CRC.

- The CRC has one bit less than the divisor. It means that if CRC is of n bits, divisor is of n+ 1 bit.

- The sender appends this CRC to the end of data unit such that the resulting data unit becomes exactly divisible by predetermined divisor i.e. remainder becomes zero.

➢ At the destination, the incoming data unit i.e. data + CRC is divided by the same number (predetermined binary divisor).

➢ If the remainder after division is zero then there is no error in the data unit & receiver accepts it.

➢ If remainder after division is not zero, it indicates that the data unit has been damaged in transit and therefore it is rejected.

➢ CRC is based on binary division. A sequence of redundant bits called CRC or CRC remainder is appended at the end of a data unit such as byte.

Requirements of CRC :

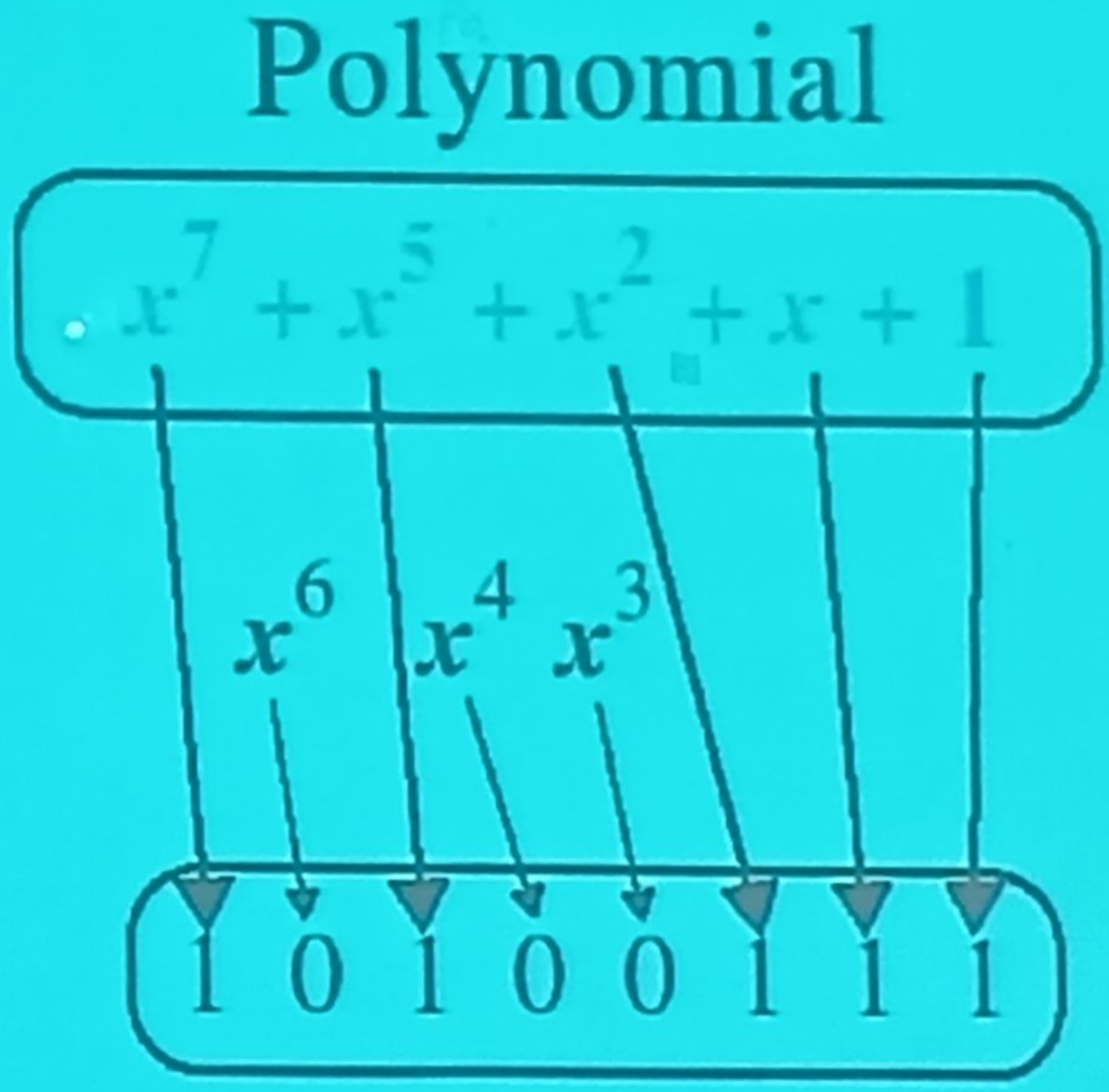
A CRC will be valid if and only if it satisfies the following requirements:

1. It should have exactly one less bit than divisor.

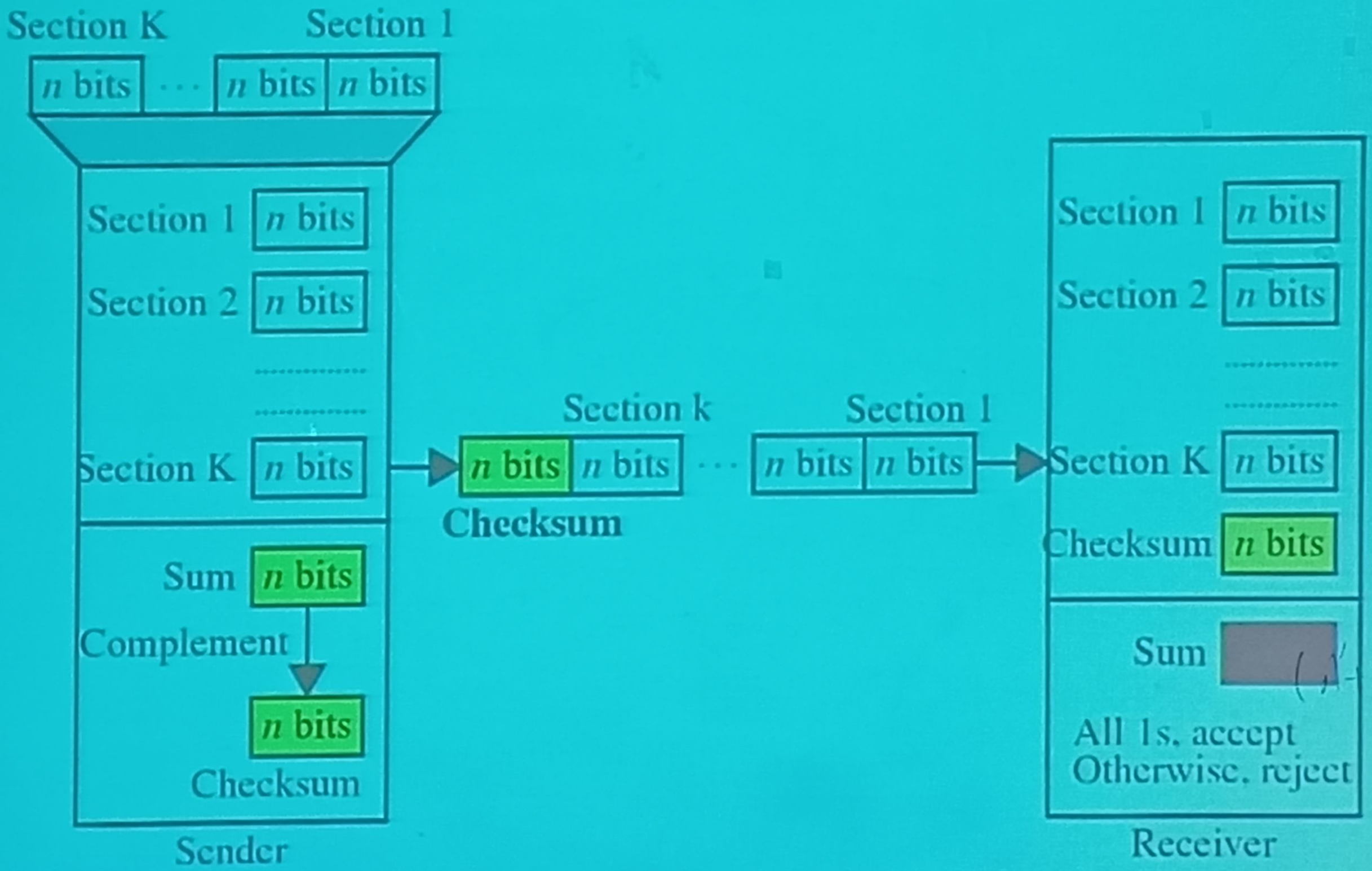
2. Appending the CRC to the end of the data unit should result in the bit sequence which is exactly divisible by the divisor.

$(A'+B')'$

Polynomial and Divisor



Checksum


Section K
Section 1
n bits
n bits
n bits
n bits
Section 1
Section 2
Section K
Sum
Complement
Checksum
Sender
Section k
Section 1
Checksum
Section 1
Section 2
Section K
Checksum
Sum
All 1s. accept
Otherwise. reject
Receiver