Unit - 2

Data Link Layer

-> Data link layer is above the physical layer & below the network layer.
-> It is responsible for node to node delivery of messages.
-> The data link layer converts packets into frames, it depends on frame size of NIC (network interface card)
-> When frame buffer is full it stop the transmitting signals
-> The data link layer discard the data & send again

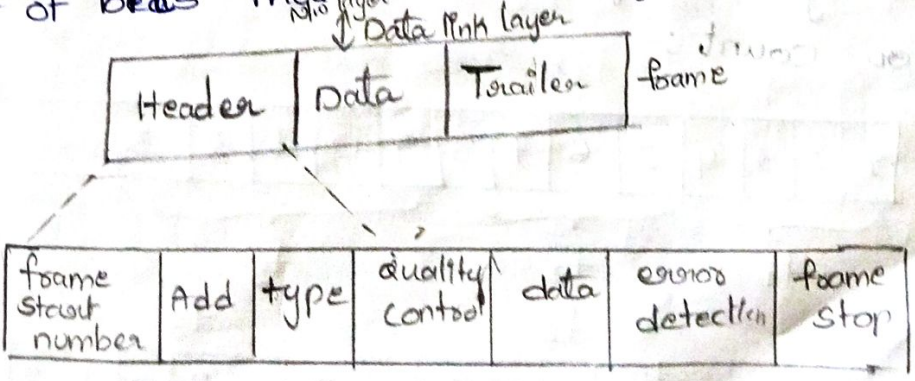
functions of data link layer:-

* framing
* Physical address
* Error control
* flow control
* Access control.

Data link layer design issues :-

1. framing
2. Error control
3. flow control

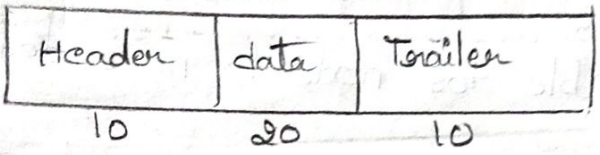
1. framing :- framing is a function of data link layer which provides the way for a ~~sent~~ sender to transmit a set of ~~bits~~ that are meaningful to the receivers.



types of frames

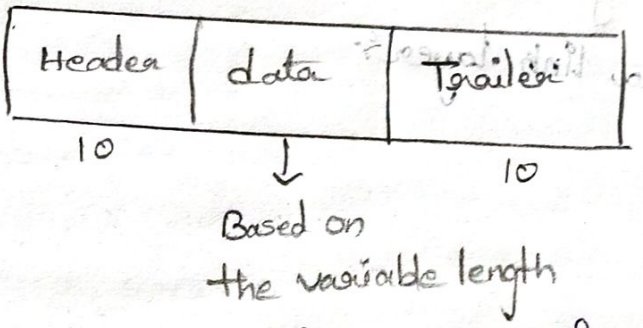
1. fixed length frame
2. variable length frame

1. fixed length frame



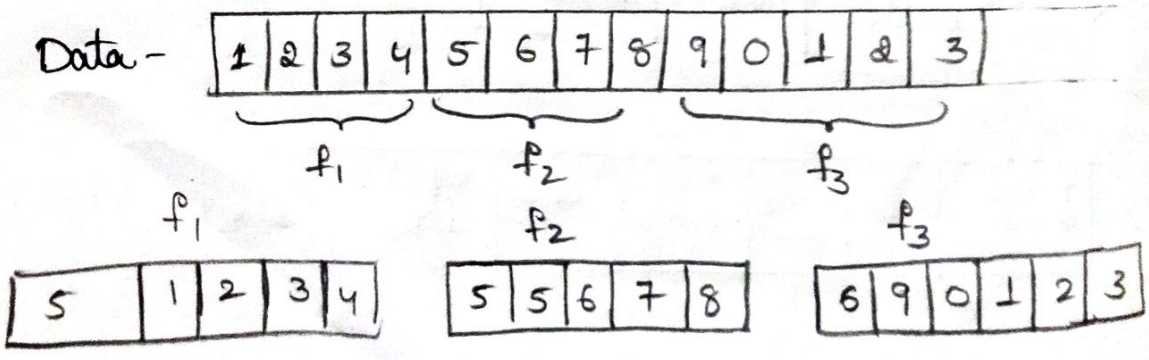
- In fixed length frame the frame size should be fixed
- If 200 bits of data to transfer from sender to reciever the frame size should be 10.

2. Variable length frame.



- In variable length frame the frame size is not fixed & it is based on variable length
- 4 methods are used to start, & end of each frame
  1. Character count
  2. flag bite with bite stuffing
  3. flag bit with bit stuffing.
  4. Physical layer coding vidation

-> Character count :-



finding correctly

5	1	2	3	4	5	5	6	7	8	6	9	0	1	2	3

1234 5678 90123

f_1 f_2 f_3

Error finding

5	1	2	3	4	7	5	6	7	8	6	9	0	1	2	3

f_1 f_2

1234 5678 69

-> flag byte with byte stuffing

- Each frame start & end with special byte called flag bite.
- If the flag byte occurs in the frame, stuff or insert an extra escape byte.
- The frame with flag byte format

FLAG	Header	Payload field	Trailer	FLAG

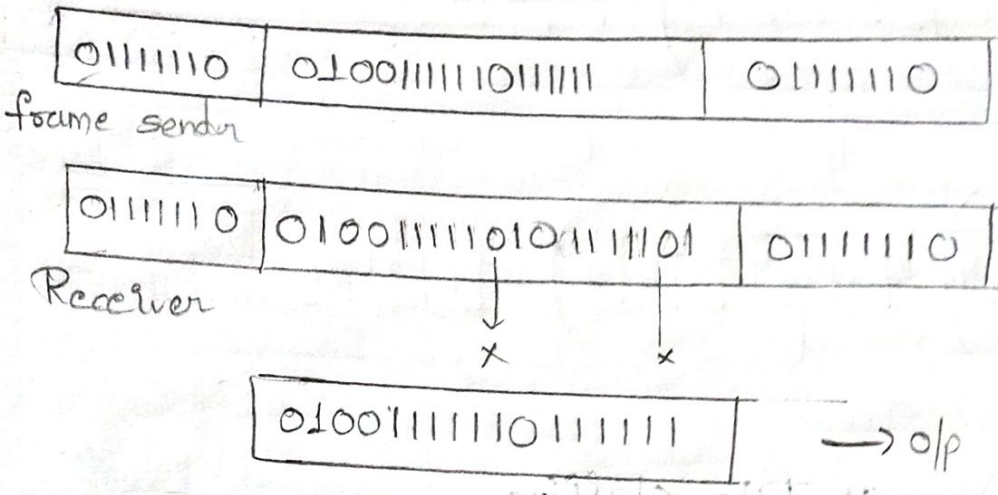
Original character Before stuffing → After stuffing

A	FLAG	B	→	A	ESC	FLAG	B
A	ESC	B	→	A	ESC	ESC	B
A	ESC	FLAG	B	→	A	ESC	ESC	ESC	FLAG	B
A	ESC	ESC	B	→	A	ESC	ESC	ESC	ESC	B

-> flag bit with bit stuffing

- The starting & end of each frame specified a bit pattern is added.
- The original data is 0100111111011111
- Through original data the flag bit 01111110 has to be transmitted.

- After 5 consecutive 1's insert 0's in flag bits.



→ Physical layer coding violation

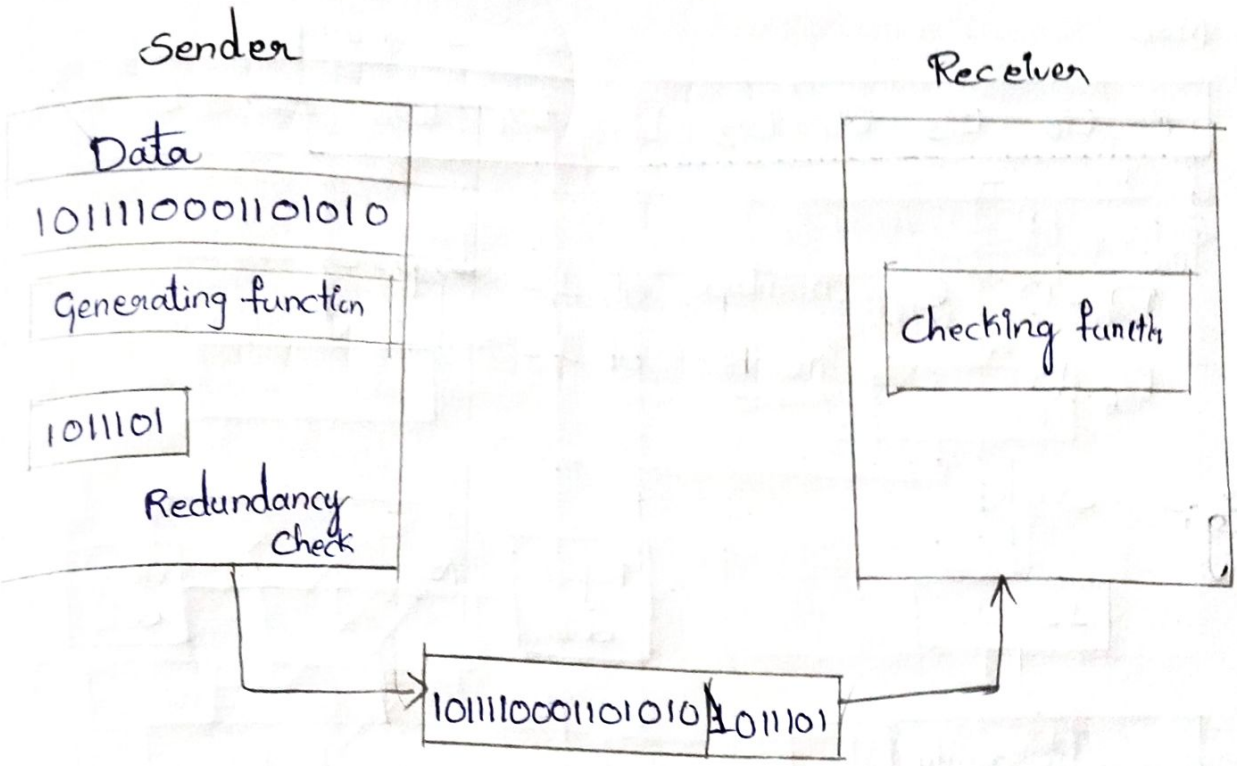
- Coding violation is a method that is used in network which encoding on physical medium including some sort of redundancy
- Local area network encoding the each bit of data is represented by two physical bits whenever bit is 1 it represents as 1,0 (1 represents high 0 represents low)
- Combination of 00, 11 are not used in encoding
- In order to operate a division bit frame in data link layer, this approach has redundancy so error codes like 00 & 11 are used as escape bits

Error detection & Correction

Error detection:-

Redundancy is the main technique to detect errors, a group of bits adding end of the data

- In detection process valid data is accepted & invalid data is rejected



Error Correction:-

There are two ways to handle the error correction

→ When error found the reciever ask the sender to retransmitt the same data.

→ The reciever can use an error correcting codes to detect & correct the errors.

Error detecting & Correcting techniques

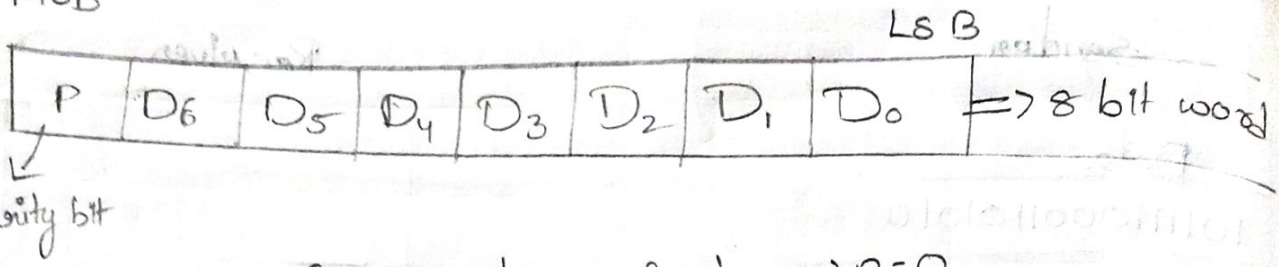
- Error detecting techniques

- The error detecting techniques are

1. Parity checker
2. Check sum.
3. Cyclic redundancy check

1. Parity checker

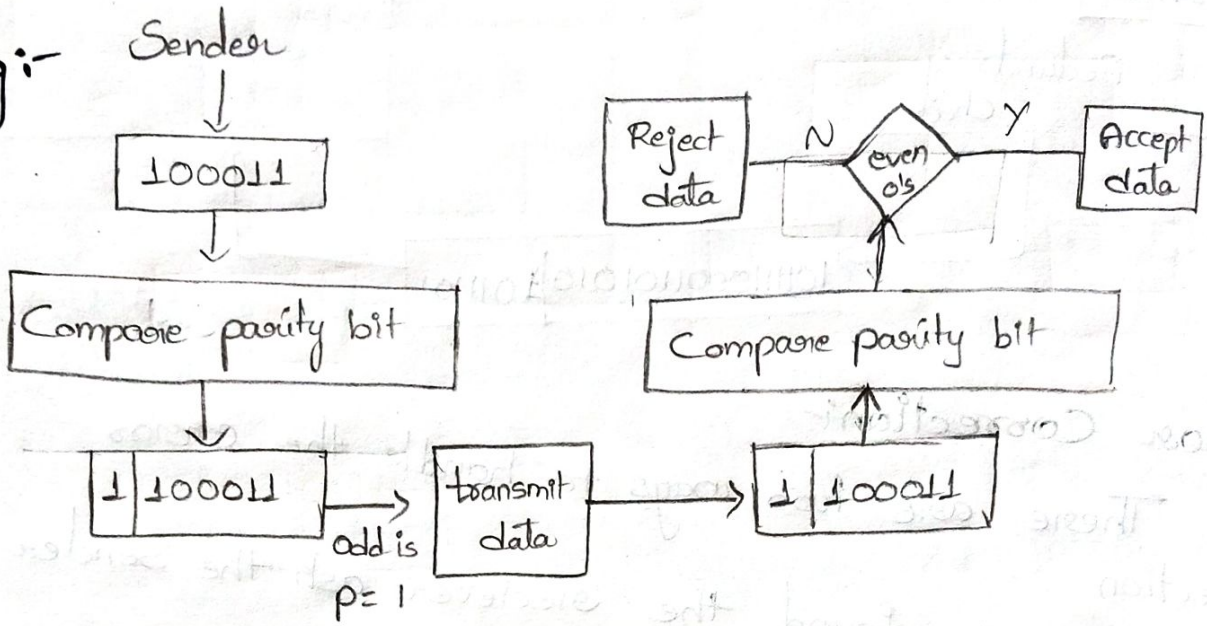
The parity checker is the simplest technique for error detection.



D_6-D_0 - Even number of 1's $\Rightarrow p=0$

D_6-D_0 - Odd number of 1's $\Rightarrow p=1$

Eg:-



Check Sum:-

In check sum the data is divided into k segments with m-bits.

-> The Sender send the segment as added +1 using 1's complement arithmetic to get the sum, the sum is complemented to get the check sum.

-> Now the check sum segment is sent along with data segment.

-> At reciever side the total recieved segments are added using 1's complement arithmetic to get the sum, The result sum is complemented again.

-> If the result is 0 then data is accepted otherwise rejected.

Eg:

10011001 | 11100010 | 00100100 | 10000100

Sender

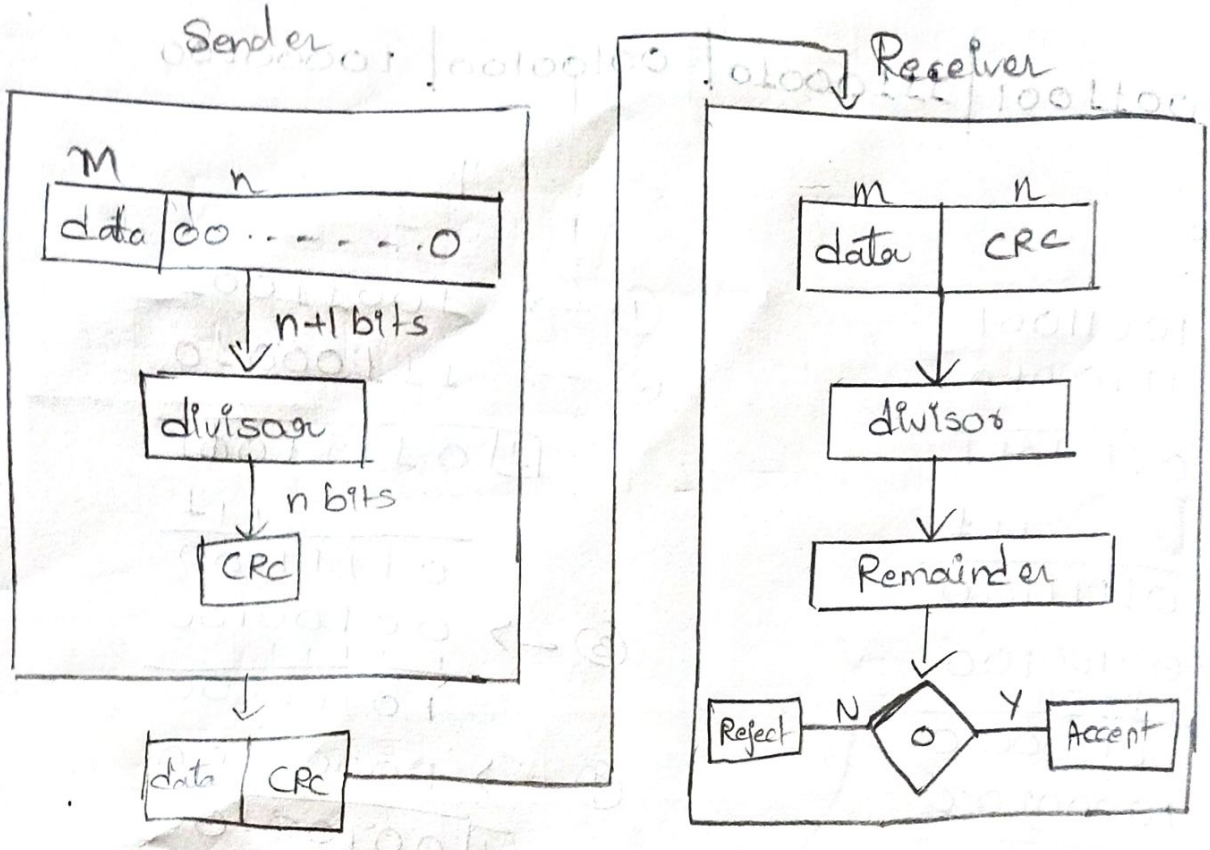
① → 10011001
② → 11100010
1 01111011
111
01111100
③ → 00100100
10000
10100000
④ → 10000100
1 00100100
1
00100101 → sum
11011010 → 1's complement

Reciever

① → 10011001
② → 11100010
1 01111011
111
01111100
③ → 00100100
11111
10100000
④ → 10000100
1 00100100
1
00100101
11011010 → 1's complement
11111111
00000000 → 1's complement

Cyclic redundancy check :-

→ It is an error detection method based on binary division.
→ In CRC a sequence of redundant bits called Cyclic redundancy check bits are appended to the end of data.
→ The resulting data unit exactly divisible by second pre determined binary number.
→ The destination result, source data is divided by the same number.
→ If the remainder is 0 then accept the data otherwise reject the data.



Ex: 1101011011 → Data

$x^4 + x + 1$ → polynomial

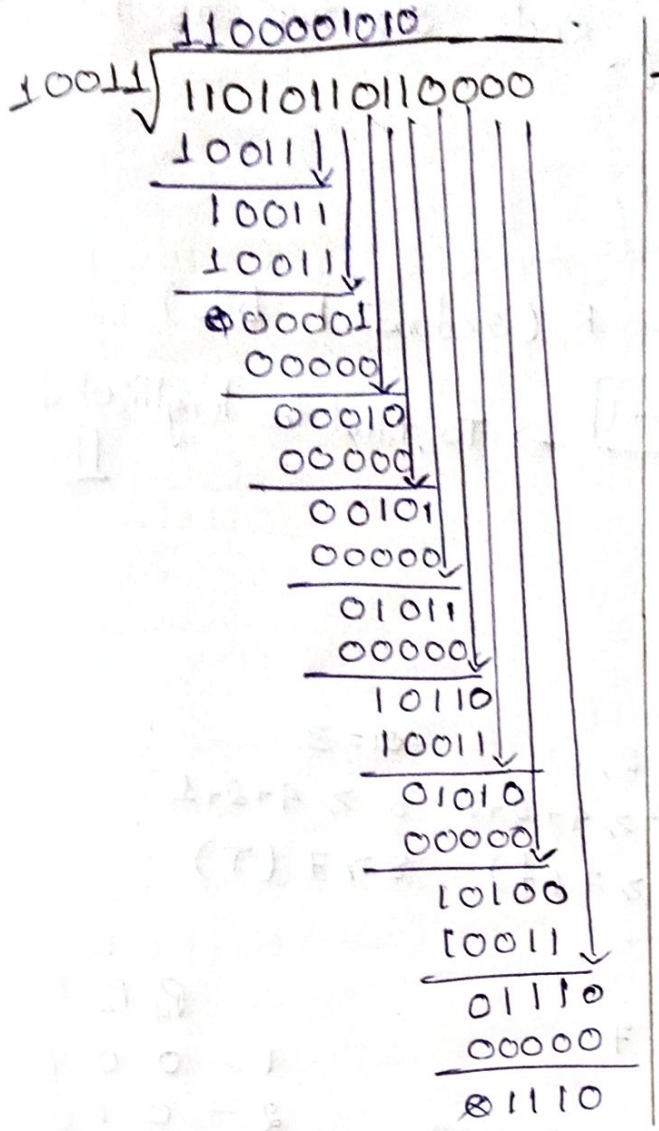
$1 \cdot x^4 + 0x^3 + 0x^2 + 1 \cdot x^1 + 1 x^0$

10011 → 5 bits

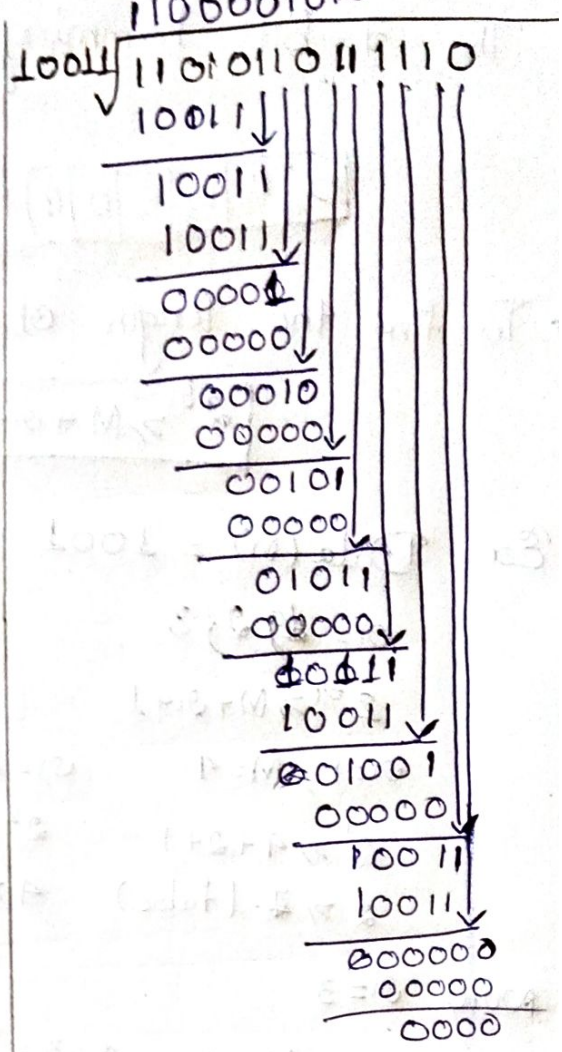
5 → (n-1) = 4 bits

11010110110000

Sender



Receiver



Data :- 11010110111110

Error correcting techniques

The following are error correcting techniques

1. Hamming code
2. Binary convential code
3. Low density parity check code

1. Hamming code :-

- The hamming code can be applied to data units of any length & it is used to detect & correct single bit errors
- The hamming code structure consist n no. of bit positions that are power of 2 & those are marked as parity bits.

and other bits are for normal data.

- The 7-bit hamming code structure as follows

D_7	D_6	D_5	P_4	D_3	P_2	P_1

- To find the length of R (redundant bits)

$2^R \geq M + R + 1$ → To find the length of data $M+R$

Ex: Data (M) = 1001

$R = 0, 1, 2, 3$

$2^R \geq M + R + 1$

$R = 1, M = 4$: $2^1 \geq 4 + 1 + 1$, $2 \geq$ ~~4~~ (false)

$R = 2$, : $2^2 \geq 4 + 2 + 1$, $4 \geq 7$ (f)

$R = 3$: $8 \geq 4 + 2 + 1$, $8 \geq 7$ (T)

Now $R = 3$

Length = $M + R = 4 + 3 = 7$

D_7	D_6	D_5	P_4	D_3	P_2	P_1

1	0	0	P_4	1	P_2	P_1

	P_4	P_2	P_1
1 —	0	0	1
2 —	0	1	0
3 —	0	1	1
4 —	1	0	0
5 —	1	0	1
6 —	1	1	0
7 —	1	1	1

P_1 → 1, 3, 5, 7 → P_1 1 0 1 → $P_1 = 0$

P_2 → 2, 3, 6, 7 → P_2 1 0 1 → $P_2 = 0$

P_3 → 4, 5, 6, 7 → P_4 0 0 1 → $P_4 = 1$

Even no 1's = 0

Odd no 1's = 1

1	0	0	1	1	0	0
7	6	5	4	3	2	1