

Binary to decimal \rightarrow 8421 approach

$$\begin{matrix} 32 & 16 & 8 & 4 & 2 & 1 \\ (1 & 1 & 0 & 0 & 0 & 1) \end{matrix}_2$$

$$32 + 16 + 1 = \textcircled{59}_{10}$$

Ex $\begin{matrix} 32 & 16 & 8 & 4 & 2 & 1 \\ (1 & 0 & 1 & 0 & 1 & 1) \end{matrix}_2$

$$32 + 8 + 2 + 1 = \textcircled{43}_2$$

Binary Number

$$\begin{matrix} \textcircled{1} & 0 & 1 & 1 & 0 & \textcircled{1} \end{matrix} \leftarrow \text{LSB}$$

MSB

$$\text{LSB} \rightarrow 1 \text{ (odd)}$$

$$\text{LSB} \rightarrow 0 \text{ (even)}$$

$$\begin{pmatrix} 32 & 16 & 8 & 4 & 2 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}_2 \rightarrow \text{Even}$$

$$32 + 8 + 2$$

$$= \textcircled{42}_{10}$$

Error detecting & Correcting Code

↳ Malfunction, which disturb the actual flow of Execution

↳ Change in data

↳ Change/manipulation in H/W

* Whenever a message is transmitted, it may get scrambled by noise or data gets corrupted.

* To avoid these kind of things we use error detection & Correction code

Ex 'Parity Check', 'Hamming Code', 'CRC (cyclic Redundancy Check)'

Parity Check:

It ensures accurate data transmission between two nodes during communication.

→ It can detect 1 bit error.

↳ even parity: Total no. of 1's in a bitstream are even

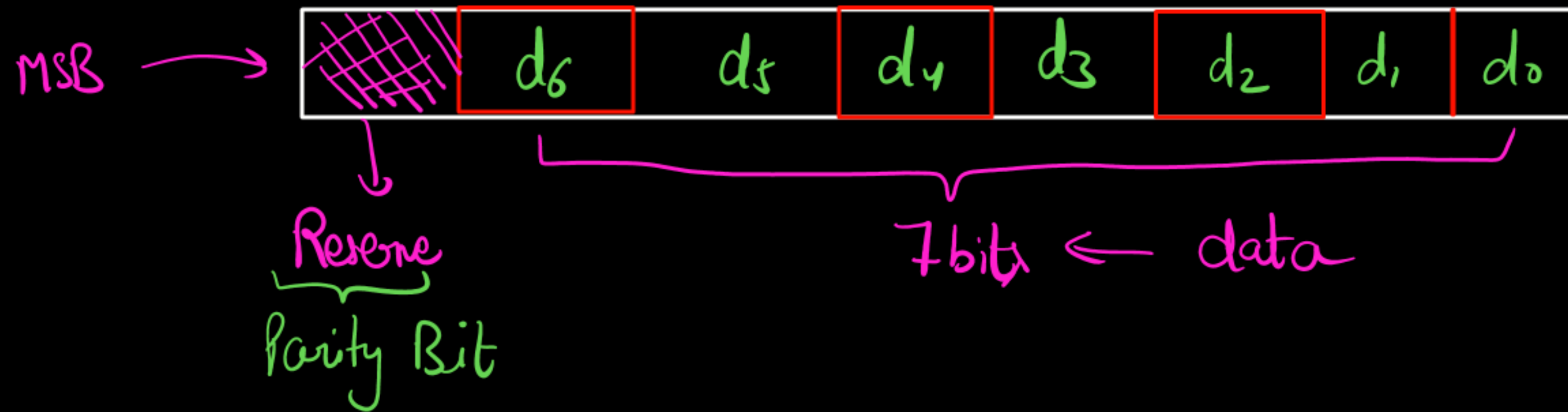
Odd parity: Total no. of 1's in a bitstream are odd

0 1 0 1
└────────┘
no. of 1's = 2
↓
Even
number
(even parity)

0 1 1 1
└────────┘
no. of 1's = 3
↓
Odd
number
(odd Parity)

Parity Check

Suppose we've 8 bit number



Ex

0 1011010

data

no. of 1's = 4
↓
even

even
parity

1 1011010

data

no. of 1's = 4 (even)

no. of 1's = 5 (odd)
↳ odd parity

Sender
0 1 0 1 1 0 1 0
 n. of 1s = 4 (even)
 parity

Receiver
0 1 0 1 1 0 1 1
 n. of 1s = 5 (odd)
 ↓ Error

odd bits
 Even parity → 1, 3, 5, 7, 9, ...
 (detectable)
 → [2n+1] bits ← detectable
 ← odd

0 1 0 1 1 0 0 1 } 2 bits change
 n. of 1s = 4 (even)
 ↓
 [Limitation]

Even Parity → n. of Bit switch ⇒ 2n (even)
 ↳ undetectable

Data Representation:

↳ Inside memory & decimal Representation

- Unsigned Magnitude
 - Signed Magnitude
 - 1's Complement
 - 2's Complement
 - 9's Complement
 - fixed & floating point representation (important)
- } important
- } Concept

(A) Unsigned Magnitude:

↳ Unsigned number does not has any sign
↳ +ve number

→ Range 0 to ∞

→ $0 \leq n \leq \infty$

If we have 'n' bits, then we can represent 2^n numbers in unsigned Mag. Representation

Range $[0 \text{ to } 2^n - 1]$

If we have 4 bits,

Binary: Range 0000 to 1111

Decimal: Range 0 to 15

0 to $2^n - 1$

$$1 \text{ bit} = 0, 1 \rightarrow 2^0$$

$$2 \text{ bits} = 4 \rightarrow 2^2$$

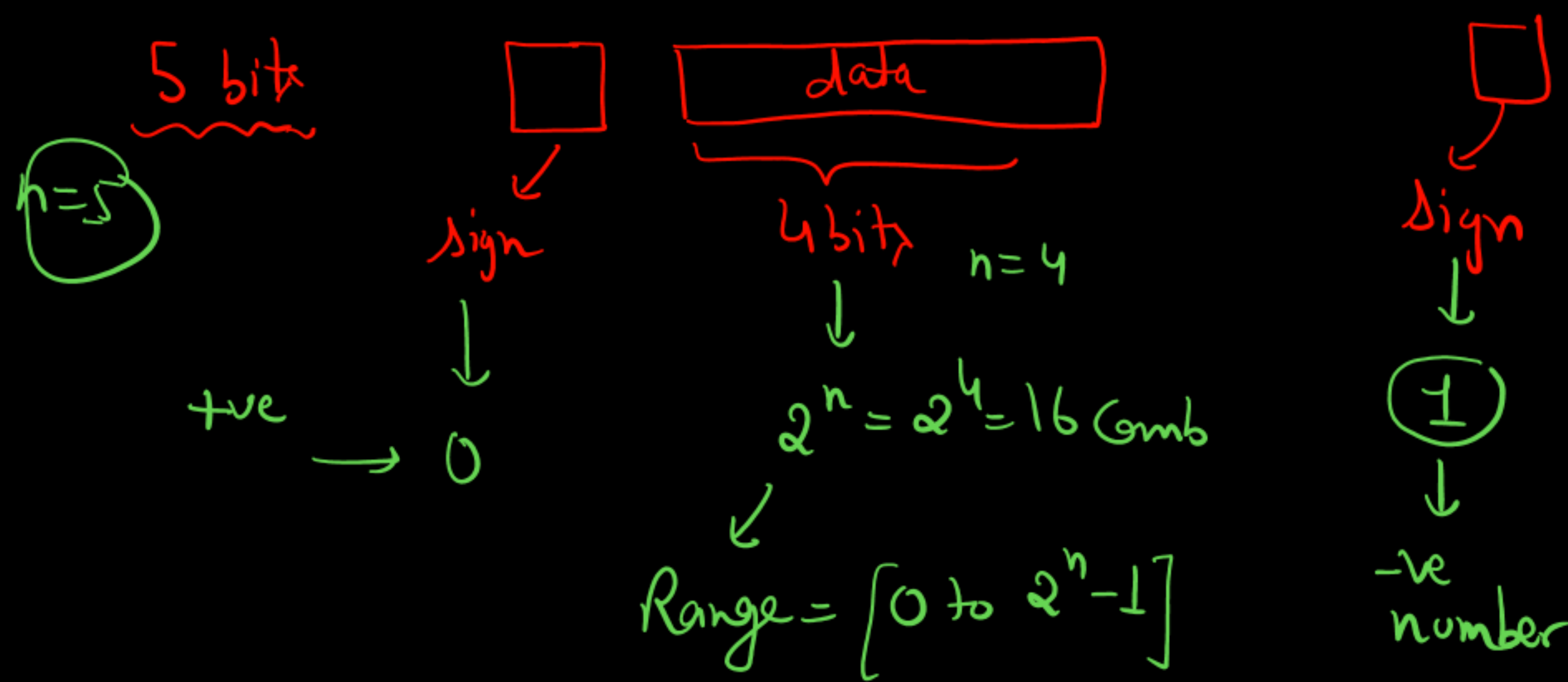
$$3 \text{ bits} = 8 \rightarrow 2^3$$

$$4 \text{ bits} = 16 \rightarrow 2^4$$

$$5 \text{ bits} = 32 \rightarrow 2^5$$

$$n \text{ bits} = 2^n$$

* No negative numbers.



data
 4 bits \rightarrow
 \downarrow
 $2^n = 2^4$ Comb. = 16
 \downarrow
 Range $[0 \text{ to } 2^n - 1]$
 $\Rightarrow - [0 \text{ to } 2^n - 1]$
 $= [-0 \text{ to } -2^n - 1]$

total Range $-2^{(n)} - 1$ to $2^n - 1$

$\left[\underbrace{-2^{n-1} - 1}_{\text{-ve number}} \text{ to } \underbrace{2^{n-1} - 1}_{\text{+ve}} \right]$

$\rightarrow n$ bits \rightarrow 1 bit sign $\underbrace{(n-1) \text{ bits data}}_{\text{Range}} \rightarrow [2^{(n-1)} - 1]$

$\rightarrow - [2^{(n-1)} - 1]$
 to
 $[2^{(n-1)} - 1]$

Representation

Range

Unsigned Magnitude	0 to $2^n - 1$
Signed Magnitude	$-(2^{n-1} - 1)$ to $(2^{n-1} - 1)$
Signed 1's Complement	$-(2^{n-1} - 1)$ to $(2^{n-1} - 1)$
Signed 2's Complement	$-(2^{n-1})$ to $(2^{n-1} - 1)$

1 bit reserve