ERROR DETECTION

Parity Check (Vertical Redudancy Check)❶
Longitudinal Redudancy Check❷
Cyclic Redudancy Check ❸
Checksum ❹

Telkom University

Budhi Irawan, S.Si, M.T

Transmisi Data

- Pengiriman sebuah informasi akan berjalan lancar apabila tidak terjadi ***hambatan*** apapun dalam pengiriman.
- Informasi yang dikirimkan akan diterima ***sama persis*** seperti data awal.
- Jarak antara pengirim dan penerima hanya akan berpengaruh ***terhadap waktu***.

Noiseless Channel

- ***Delay*** adalah ***selisih waktu*** dari informasi saat dikirimkan sampai informasi tersebut diterima oleh penerima.
- Kanal yang tidak menyebabkan error pada proses transmisi data disebut ***Noiseless Channel***.

Noisy Channel

- Pada kenyataannya tidak ada media transmisi yang benar-benar ***tanpa hambatan*** dan ***tanpa noise***.
- Pengiriman data melalui media transmisi dipengaruhi oleh berbagai macam ***gangguan*** yang terjadi selama ***perjalanan data*** tersebut.
- Kanal yang dapat menyebabkan error selama proses transmisi data disebut ***Noisy Channel***

Mekanisme Error Control

- ***Mekanisme Error Control*** merupakan suatu mekanisme untuk dapat memastikan apakah data yang dikirimkan oleh pengirim dapat diterima dengan benar oleh penerimanya.
- Apabila data yang diterima ***tidak benar***, maka data tersebut harus ***dikirim ulang*** oleh si pengirim sampai data ***diterima dengan benar*** oleh penerimanya.

Mekanisme Error Control

- *Backward Error Control (BEC)*
- *Forward Error Control (FEC)*

Metode Error Detection

- ***Pengiriman data*** menggunakan ***format biner*** memudahkan dalam mendeteksi ada atau ***tidaknya error*** pada data yang dikirim.
- Apabila penerima dapat mengetahui ***letak bit*** yang error pada sekumpulan ***bit data*** yang ***dikirim,*** maka penerimanya akan dengan mudah ***memperbaikinya.***

Metode Backward Error Control

- **Parity Check (VRC)** → paling sederhana
- **LRC** →Pengembangan dari Parity Check
- **CRC** → lebih sulit, meminta kemampuan komputasi
- **Checksum** → operasi word

Parity Check (VRC)

- Merupakan metode dimana ada penambahan bit pada deretan bit data
- **Parity Check** disebut juga dengan nama **Vertical Redudancy Check**
- Terdapat 2 jenis pariti : genap dan ganjil
  - ***Single Pariti genap*** = jumlah bit 1 dalam kode adalah **genap**
  - ***Single Pariti ganjil*** = jumlah bit 1 dalam kode adalah **ganjil**

Parity Check (VRC)

- Sistem sederhana dan mudah dibuat hardwarenya

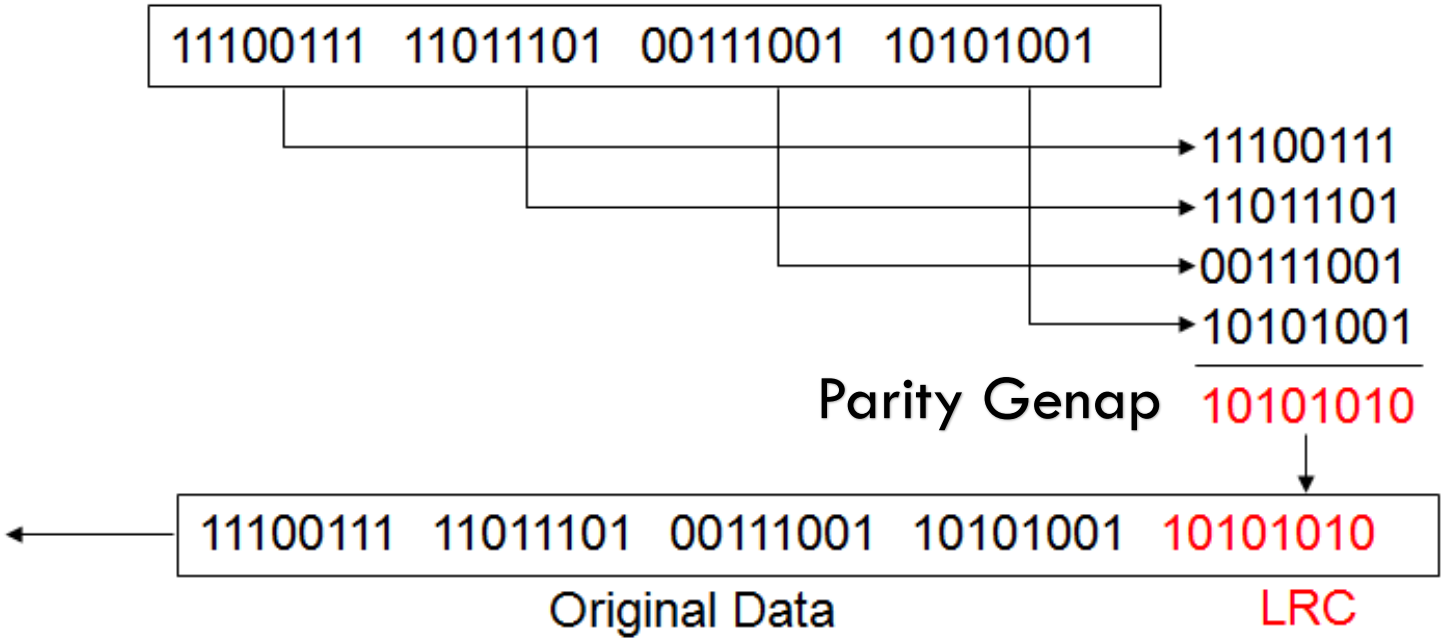
  Contoh : Karakter ASCII **A** (**1000001**)

Single Pariti Ganjil

Single Pariti Genap

Longitudinal Redudancy Check (LRC)

Data diorganisasikan kedalam suatu table dan tambahkan setiap parity bit pada setiap kolom.



Cyclic Redundancy Checks (CRC)

- Pada metode ini, pengirim akan melakukan proses pembagian data dengan suatu pembagi tertentu yang disebut dengan ***Generator Polynomial (Kode CRC)***.
- Bit – bit sisa pembagian disebut ***Reminder***.
- ***Reminder*** inilah yang yang ikut dikirimkan bersama ***data aslinya***.

Cyclic Redundancy Checks (CRC)

- Pada ***sisi penerima*** akan dilakukan operasi yang ***sama***, yaitu membagi seluruh data yang sampai.
- Apabila ***bit-bit sisa pembagian*** bernilai ***0***, maka dapat dipastikan bahwa data yang sampai di penerima ***tidak mengalami error***.

Algoritma CRC Dasar

CRC => konsep matematis untuk operasi polynomial (persamaan pangkat terbesar).

- **Messages Mx :** Pesan yang akan dikirim
- **Kode CRC Cx :** Generator polynomial dengan degree tertentu (k).
  - CRC-16 → 1100000000000101
  - CRC-ITU → 1000100000100001
  - CRC-32 → 10000010010000001000111011011011 1
  - Deretan bit yang diawali dan diakhiri dengan bit 1 (1xxxxxx1)

Algoritma CRC Dasar

Contoh:

Kode CRC : **1001**

k = 3 (penambahan 3 bit 0 (**000**) pada **Mx**)

Cx => polynomial dengan **derajat 4.**

- Representasi Koefisien Polynomial: **1**, **0**, **0**, **1**
- Polinomial : **Cx** = $x^3 + x^2 + x^1 + x^0$

Algoritma CRC Dasar

- **Pembentukan kode**

  Reminder Rx : Sisa hasil pembagian XOR antara (**Mx** ditambahkan k bit 0 terhadap **Cx**

- **Sender : *Kirim* => Px = Mx + Rx**

  Paket Data Kirim Px : Mx ditambahkan **Rx**

Algoritma CRC Dasar

- **Receiver:** Terima <= Px
  - Lakukan operasi pembagian **Px** dengan **Cx**.
  - Jika terdapat **sisa** (reminder) maka **error**.
  - Jika ***tidak terdapat sisa (zero)*** maka ***tidak ada error***.

Contoh Kalkulasi CRC

- Diketahui :

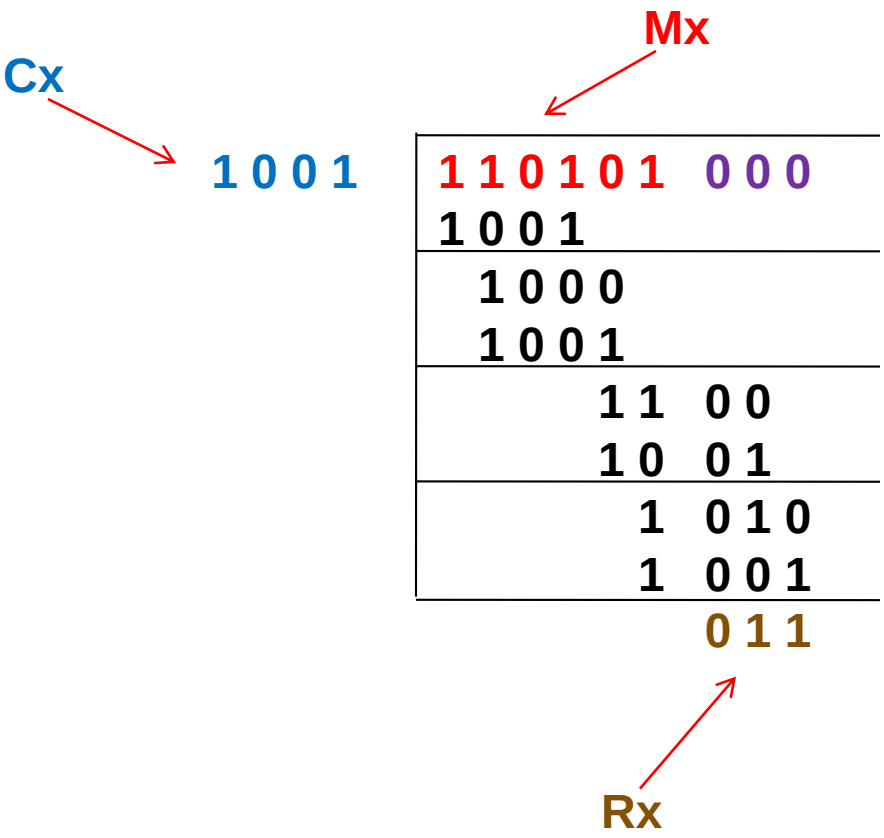
  Data yang Akan dikirimkan : **1 1 0 1 0 1 (Mx)**

  Kode CRC : **1 0 0 1 (Cx)**

- Maka :

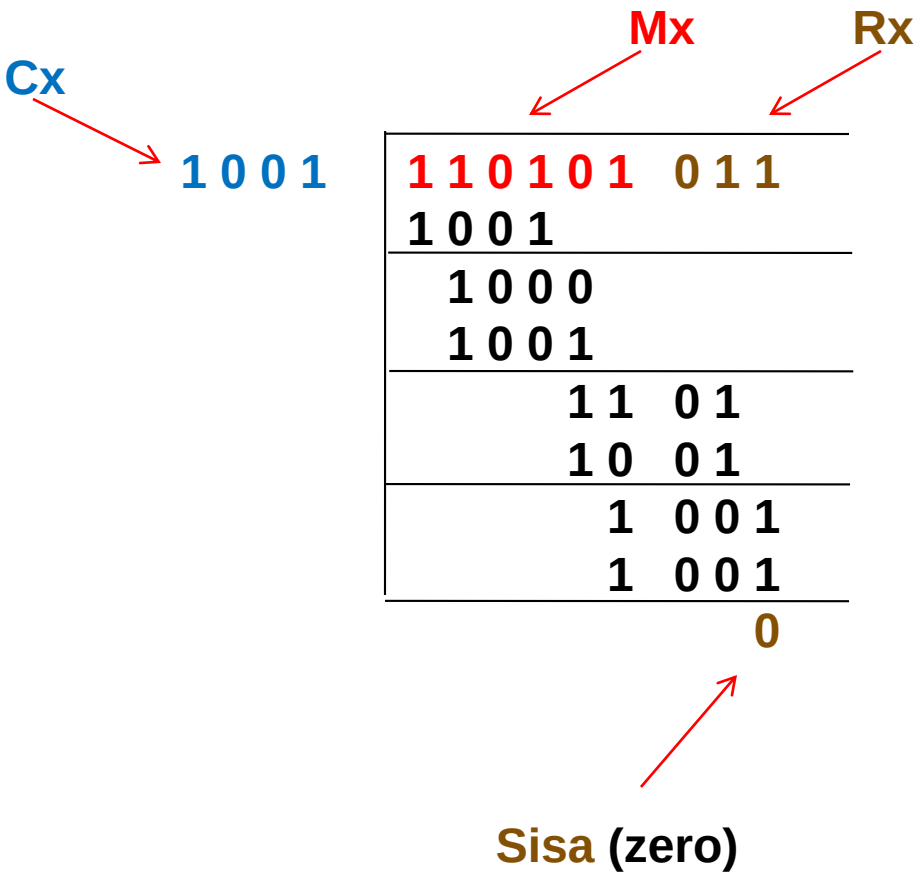
  k = 3 (penambahan 3 bit 0 (**000**)

Contoh Kalkulasi CRC



Px = Mx + Rx → 1 1 0 1 0 1 0 1 1

Contoh Kalkulasi CRC



Soal Kalkulasi CRC

1. M(x)= 10110101101,
   C(x) = 10101
2. M(x)= 10110110101,
   C(x) = 100101

Checksum

- **CRC** memerlukan perhitungan **XOR** sebanyak jumlah bit data dan memerlukan kemampuan komputasi yang cukup besar
- Diciptakan metoda **checksum** (*untuk mengurangi perhitungan*) pada beberapa jenis transmisi tidak perlu kecanggihan **CRC** atau sudah melakukan **CRC** di lapis lain

Checksum

- Cara perhitungan **checksum:**
  - Data dibagi menjadi kelompok-kelompok **16 bit (word)**
  - **Word pertama** di **XOR** dengan **word kedua**

Checksum

- Hasil di **XOR** dengan **word ketiga, keempat, … sampai word terakhir** *(jika bit-bit terakhir tidak cukup untuk menjadi word, ditambahkan padding bit '0' sampai membentuk word)*
- Hasil akhir *(16 bit)* = **checksum**

Contoh Checksum