TEKNIK KOMUNIKASI

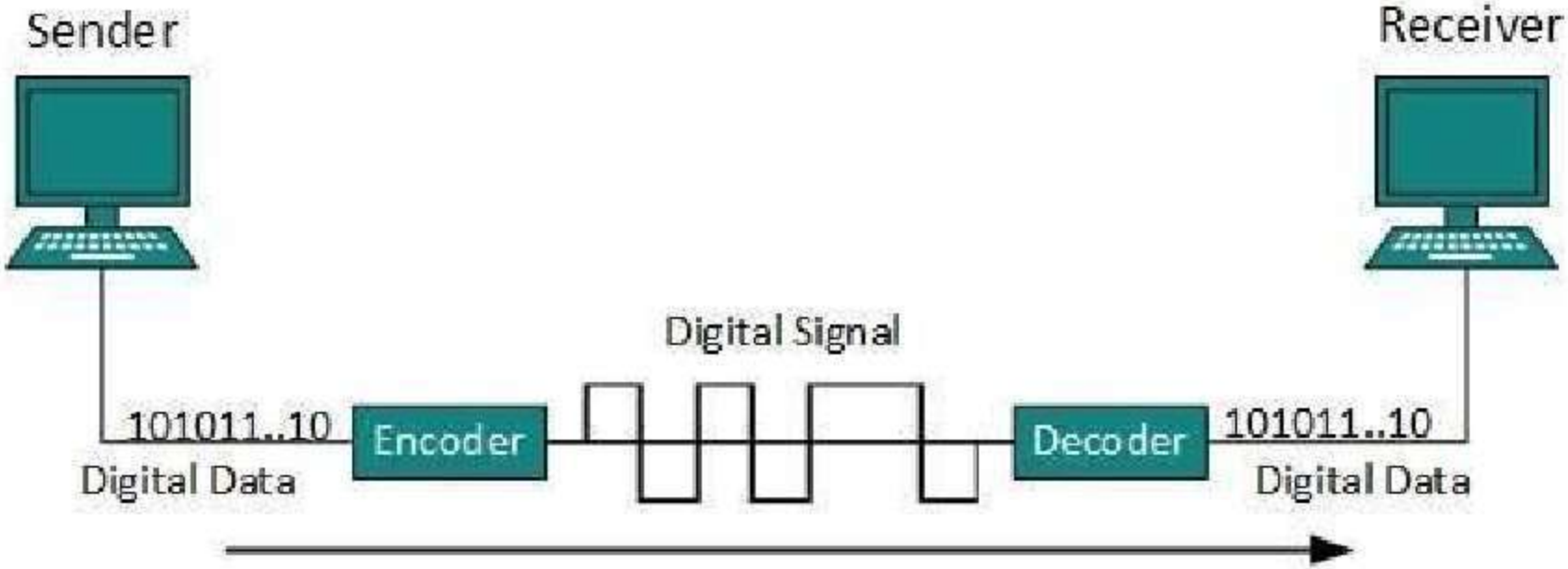
Rahmi Hidayati, S.Kom., M.Cs

TEKNIK KOMUNIKASI DATA DIGITAL

PENDAHULUAN

- Data ditransfer melalui jalur komunikasi tunggal pada transmisi data secara serial, di mana setiap elemen pensinyalan dapat berupa :

1. Kurang dari 1 bit: misal dengan pengkodean Manchester.
2. 1 bit: NRZ-L dan FSK.
3. Lebih dari 1 bit: QPSK.


Sender
Receiver
Digital Signal
101011..10
Digital Data
Encoder
Decoder
101011..10
Digital Data

PENDAHULUAN

- **Sinkronisasi** merupakan salah satu tugas utama dari komunikasi data.
- Suatu *transmitter* mengirim pesan 1 bit pada suatu waktu melalui medium ke pesawat penerima.
- *Receiver* harus mengenal awal dan akhir dari blok-blok bit dan juga harus mengetahui durasi tiap bit sehingga dapat men-*sample line* tersebut dengan waktu yang tepat untuk membaca tiap bit.

TRANSMISI ASINKRON DAN SINKRON

- Masalah *timing* membutuhkan mekanisme untuk mensinkronkan antara *transmitter* dan *receiver*.

  Contoh : aliran interval bit pada *receiver*.

  - Jika waktu tidak selaras dan bergerak sendiri maka terdapat sampel pada waktu yang salah setelah bit dikirim.

- Dua solusi untuk sinkronisasi waktu:
  - Transmisi Asinkron/*Asynchronous*
  - Transmisi Sinkron/*Synchronous*

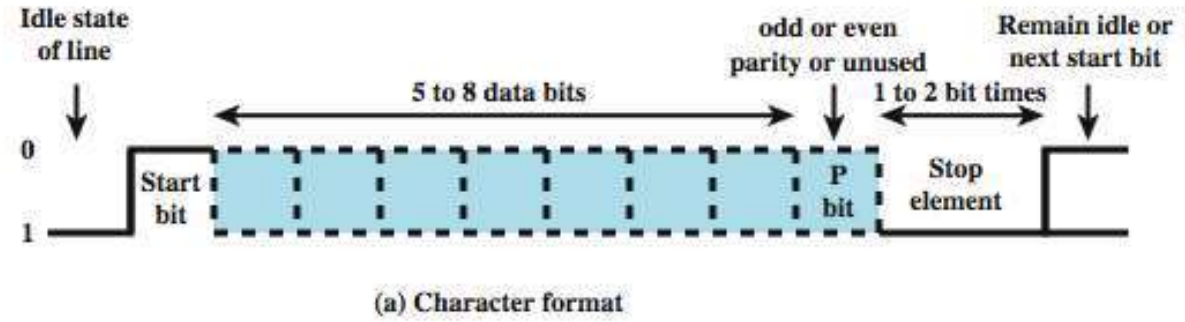
TRANSMISI ASINKRON

- Masing-masing karakter data diperlakukan secara independen.
- Strategi dengan skema ini adalah untuk menghindari masalah *timing* dengan tidak mengirimkan aliran bit yang panjang.
- Data yang ditransmisikan satu karakter pada satu waktu.

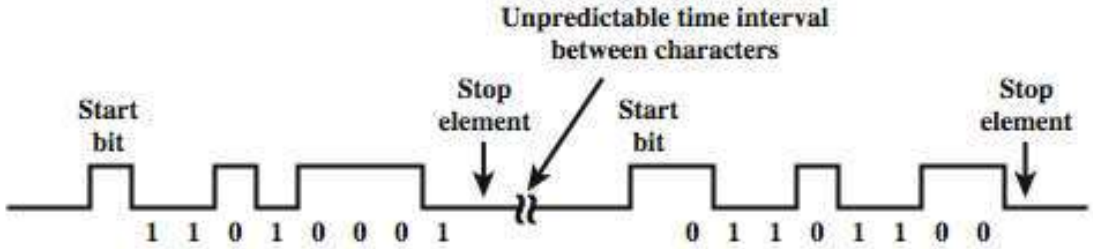
TRANSMISI ASINKRON

- Masing-masing karakter diawali dengan *start bit* yang memberitahu *receiver* bahwa karakter akan tiba.
- *Receiver* mengambil sampel setiap bit dalam karakter dan kemudian mencari awal karakter berikutnya.
- Tidak bekerja dengan baik untuk blok data yang panjang karena *clock receiver* akan bergerak keluar dari sinkronisasi dengan *clock transmitter*.

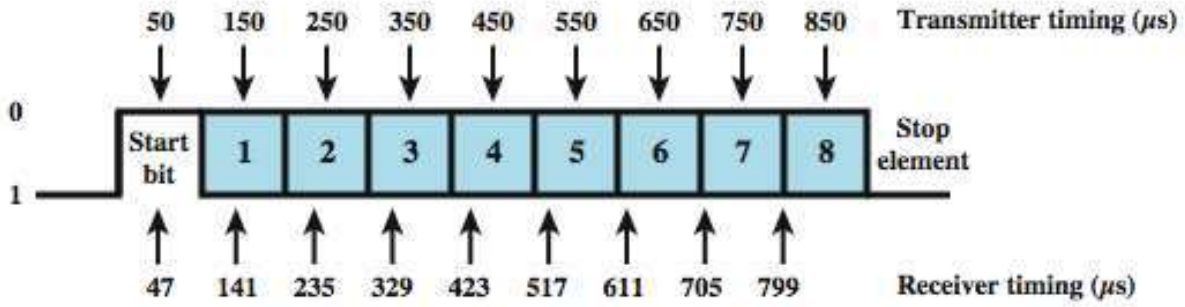
TRANSMISI ASINKRON



(a) Character format



(b) 8-bit asynchronous character stream



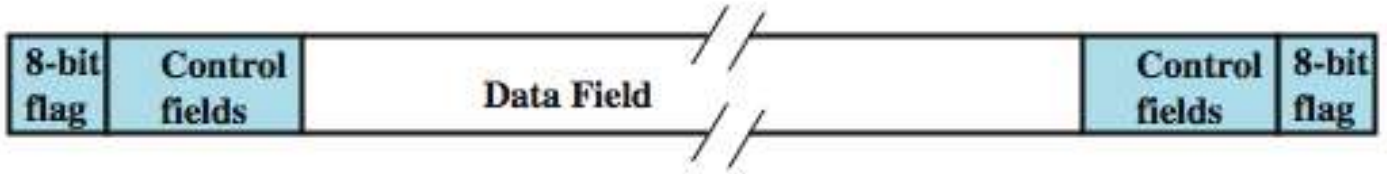
(c) Effect of timing error

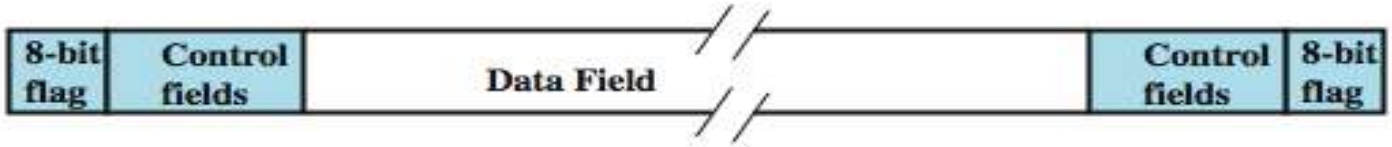
TRANSMISI ASINKRON

- Sederhana
- Murah
- Memerlukan 2 sampai 3 bit per karakter untuk sinkronisasi

TRANSMISI SINKRON

- Blok data yang ditransmisikan dikirim sebagai sebuah ***frame.***
- *Clock transmitter* dan *receiver* harus di sinkronkan
  - Dapat menggunakan jalur *clock* yang terpisah
  - atau menanamkan sinyal *clock* dalam data
- Perlu untuk menunjukkan awal dan akhir dari blok
  - Menggunakan pola *bit preamble* dan *postamble*.
- ***Frame***: data + bit preamble + postamble + kontrol informasi.
- Lebih efisien (*overhead* yang lebih rendah) daripada asinkron.





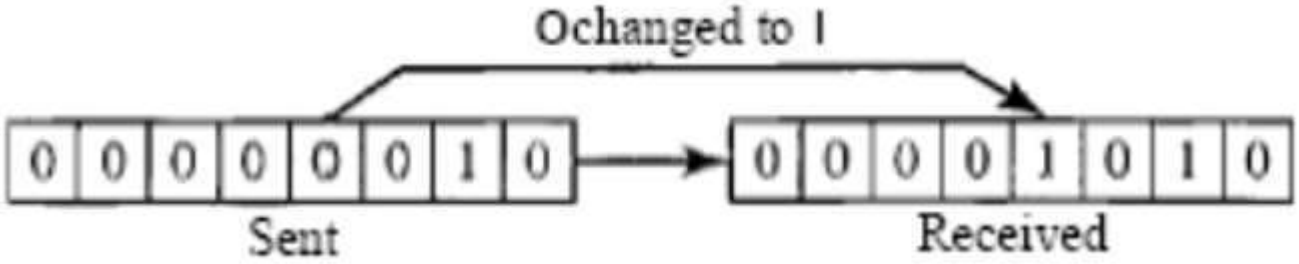
- Gambar diatas menunjukkan format frame umum untuk transmisi sinkron.
  1. **Preamble** yang disebut bendera, yang merupakan 8 bit panjangnya. Bendera yang sama digunakan sebagai **postamble**.
  2. **Bidang kontrol** (berisi *data link* mengendalikan protokol informasi, memuat informasi tentang tipe frame).
  3. **Data field,** frame yang merupakan *field* data.
  4. Untuk blok data yang cukup besar, transmisi sinkron jauh lebih efisien daripada asinkron. Transmisi asinkron memerlukan 20% atau lebih *overhead*. Informasi kontrol, *preamble* dan *postamble* dalam transmisi sinkron biasanya kurang dari 100 bit.

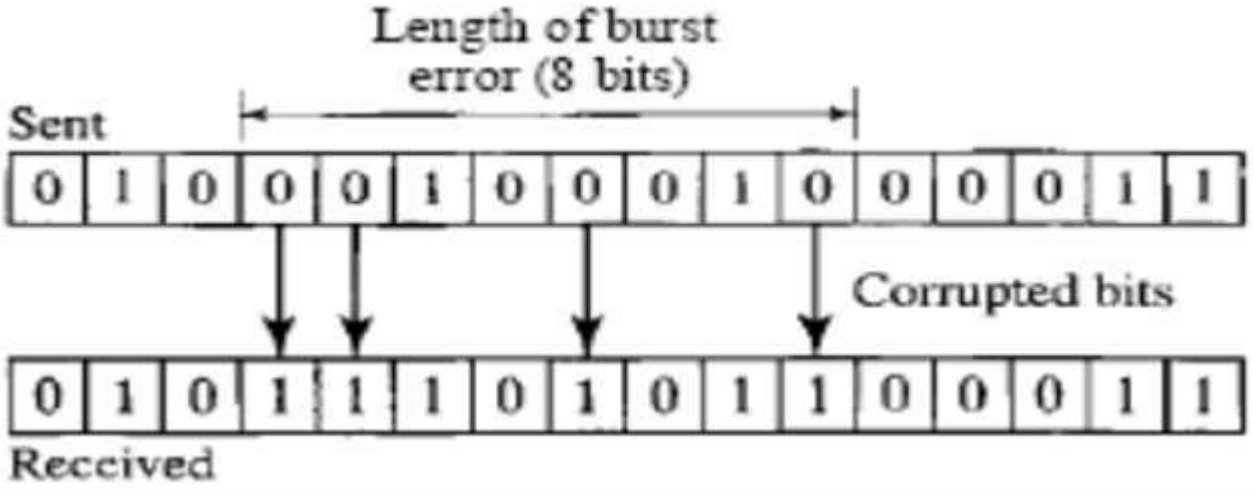
DETEKSI DAN KOREKSI KESALAHAN

JENIS ERROR

- Error terjadi ketika sebuah bit diubah antara transmisi dan penerimaan (1 jadi 0 atau 0 jadi 1).
- Kesalahan bit tunggal:
  - Hanya satu bit diubah
  - Disebabkan oleh *white noise*
- Error *burst*:
  - Rangkaian berdekatan bit B, dimana bit pertama, terakhir dan sejumlah bit tengah diterima dalam keadaan error.
  - Disebabkan oleh *noise impulse* pada lingkungan *wireless*.
  - Efek lebih besar pada kecepatan data yang lebih tinggi.

JENIS ERROR


Ochanged to 1
0 0 0 0 0 0 1 0
Sent
0 0 0 0 1 0 1 0
Received


Length of burst error (8 bits)
Sent
0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1
Corrupted bits
0 1 0 1 1 1 0 1 0 1 1 0 0 0 1 1
Reccived

DETEKSI KESALAHAN

- Terjadinya kesalahan di deteksi menggunakan kode pendeteksi kesalahan (fungsi dari bit yang ditransmisikan).
- Kode ditambahkan oleh *transmitter*.
- Dihitung kembali dan diperiksa oleh *receiver*.
- Masih memiliki peluang kesalahan yang tidak terdeteksi.
- Skema pendeteksian sederhana:
  - Menambahkan bit paritas pada ujung blok data.
  - Paritas bit diatur sehingga mempunyai sejumlah karakter genap (paritas genap) atau ganjil (paritas ganjil).

DUA PENDEKATAN UNTUK DETEKSI KESALAHAN

1. ***Forward Error Control :*** Dimana setiap karakter yang ditransmisikan atau frame berisi informasi tambahan (*redundant*) sehingga penerima tidak hanya dapat mendeteksi dimana error terjadi, tetapi juga menjelaskan dimana aliran bit yang diterima error.
2. ***Feedback (backward) Error Control :*** Dimana setiap karakter atau frame memiliki informasi yang cukup untuk memperbolehkan penerima mendeteksi bila menemukan kesalahan tetapi tidak menemukan lokasinya. Sebuah transmisi kontrol digunakan untuk meminta pengiriman ulang, menyalin informasi yang dikirimkan.

DUA PENDEKATAN UNTUK DETEKSI KESALAHAN

- ***Feedback error control*** dibagi menjadi 2 bagian, yaitu :

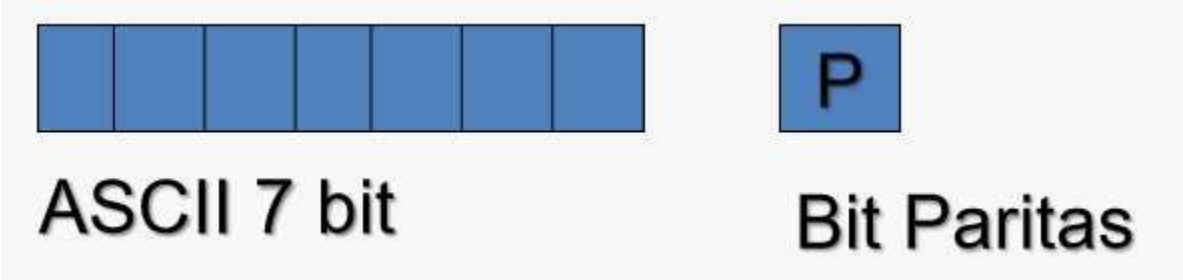
1. Teknik yang digunakan untuk deteksi kesalahan.
2. Kontrol algoritma yang telah disediakan untuk mengontrol transmisi ulang.

- Metode Deteksi Kesalahan :

1. **Echo** : Metode sederhana dengan sistem interaktif. Operator memasukkan data melalui terminal dan mengirimkan ke komputer. Komputer akan menampilkan kembali ke terminal, sehingga dapat memeriksa apakah data yang dikirimkan dengan benar.
2. **Kesalahan Otomatis**: Metode dengan tambahan bit paritas.

BIT PARITAS

- Bit tambahan yang digunakan untuk mendeteksi terjadinya error dengan hanya menambahkan 1 bit paritas pada data yang akan ditransmisikan.
- Jika terdapat k-bit *dataword* maka akan diubah menjadi n-bit *codeword*, (n = k+1).



ATURAN BIT PARITAS

a. Even Parity (paritas genap).

- Bit paritas bernilai 1, bila jumlah bit 1 adalah ganjil.
- Bit paritas bernilai 0, bila jumlah bit 1 adalah genap.

Contoh: 10101010 (data) =10101010 0.

b. Odd Parity (paritas ganjil)

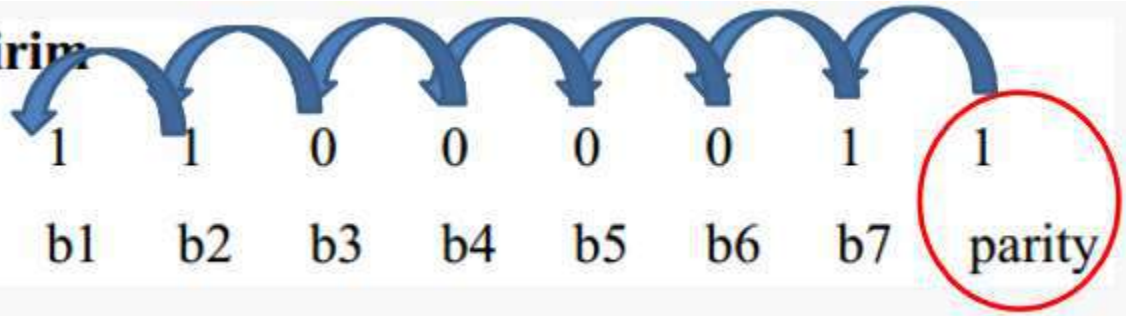
- Bit paritas bernilai 1, bila jumlah bit 1 adalah genap
- Bit paritas bernilai 0, bila jumlah bit 1 adalah ganjil

Contoh: 10101010 (data) =10101010 1.

CONTOH EVEN PARITY

Data	1	1	0	0	0	0	1
	b1	b2	b3	b4	b5	b6	b7

Pada sisi pengirim



Pada sisi penerima

Menggunakan algoritma sebagai berikut:

- Hitung jumlah bit 1 ➔ x
- Jika x = genap, dapat disimpulkan tidak ada error
- Jika x = ganjil, dapat disimpulkan terjadi error

KEKURANGAN BIT PARITAS

- Hanya mampu mendeteksi (bukan mengoreksi kesalahan), karena tidak dapat menunjukkan posisi bit yang salah.
- Hanya mampu mendeteksi satu atau sejumlah ganjil kesalahan. Bila ada dua kesalahan akan dianggap benar.

METODE DETEKSI KESALAHAN

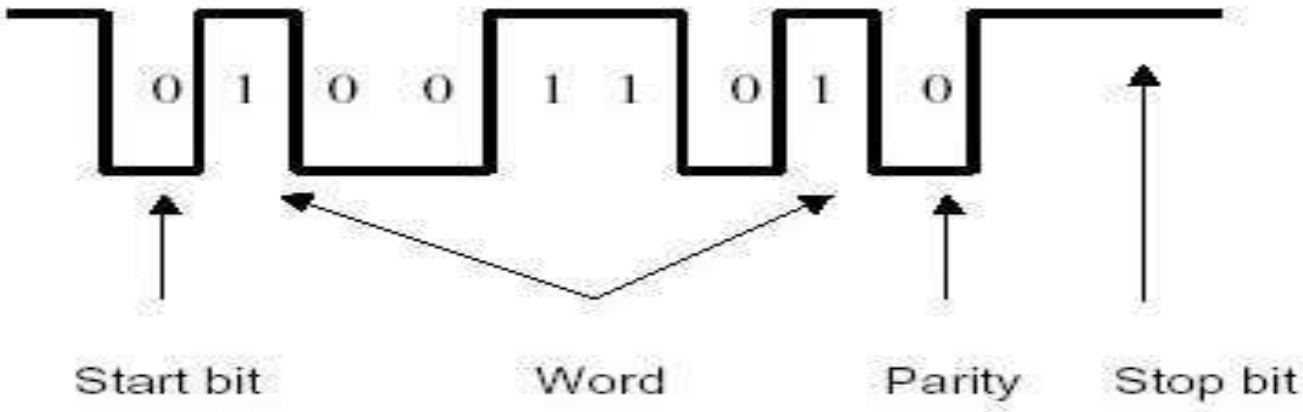
- *Vertical Redundancy Checking (VRC).*
- *Longitudinal Redundancy Checking (LRC).*
- *Cyclic Redundancy Checking (CRC).*

VERTICAL REDUNDANCY CHECKING (VRC)

- Metode ini lebih umum disebut *parity-checking* karena menggunakan sistem pengecekan paritas dan merupakan sistem untuk mencari kesalahan data yang paling sederhana.
- *Parity-checking* dibagi menjadi dua yaitu *odd-parity* (paritas ganjil) dan *even-parity* (paritas genap).
- Aturan pada *odd-parity* yaitu jumlah bit satu dalam setiap byte harus ganjil. Komputer selalu mengecek bit paritas setiap karakter yang akan dikirim, bila jumlah bit satu dalam 7bit pertama adalah genap, maka bit paritas diubah jadi 1, sebaliknya jika jumlah bit satu dalam 7bit pertama adalah ganjil, maka bit paritas diubah menjadi 0.

VERTICAL REDUNDANCY CHECKING (VRC)

- Misal, didalam komunikasi data digunakan sistem *odd parity*, maka jika huruf ***A*** disusun dalam kombinasi data biner berupa 1000001, dimana jumlah bit satu dalam 7bit pertama adalah genap, maka bit paritas diubah menjadi 1.
- Sedangkan dalam sistem *even-parity* jika huruf ***M*** disusun dalam kode biner adalah 1001101, dimana didalam 7bit pertama jumlah bit satu adalah genap, maka bit paritas ini diubah menjadi 0.



LONGITUDINAL REDUNDANCY CHECKING (LRC)

- LRC digunakan untuk memperbaiki kelemahan yang ada pada VRC (*parity-checking*). Pada sistem LRC data dikirim secara per blok (*frame*) berisi 8byte dan setiap frame terdapat satu bit paritas, fungsi dari bit ini sebagai kontrol seperti pada *parity-checking*.
- Bit paritas ini memuat 7bit paritas dari byte sebelumnya, sedangkan cara untuk mengubah nilai ketujuh bit ini yaitu dengan melihat jumlah bit satu dari seluruh byte secara vertikal.

LONGITUDINAL REDUNDANCY CHECKING (LRC)

Bit ke :	1	2	3	4	5	6	7	Parity
1	0	1	0	1	1	0	0	1
2	1	1	0	0	1	1	0	0
3	0	1	0	1	1	0	1	0
4	0	0	1	0	1	1	0	1
5	1	0	1	1	0	1	0	0
6	0	1	0	1	0	1	0	1
7	0	1	1	0	1	1	0	0
Parity	1	0	1	0	1	1	1	1

Bit ke :	1	2	3	4	5	6	7	Parity
1	0	1	0	1	1	0	0	1
2	1	1	0	0	1	1	0	0
3	0	1	0	1	1	0	1	0
4	0	0	1	0	1	0	1	1
5	1	0	1	1	0	0	1	0
6	0	1	0	1	0	1	0	1
7	0	1	1	0	1	1	0	0
Parity	1	0	1	0	1	1	1	1

CYCLIC REDUNDANCY CHECK (CRC)

- Pendeteksian kesalahan dengan cara membagi nilai bilangan biner dari data dengan suatu nilai bilangan lainnya (constanta/*pattern*).
- Pengecekkan dilakukan dengan mencocokkan sisa bagi (*remainder*).
- Teknik CRC :

1. Modulo 2 arithmetic
2. Polynomials

MODULO 2 ARITHMETIC

- Merupakan operasi X-OR.
- Menggunakan penjumlahan *binary* tanpa *carry* pada proses penjumlahan dan tanpa *borrow* pada proses pengurangan.

```
 1111        1111
+1010       -0101
------      ------
 0101        1010
```

Tabel Ekslusive OR

+	0	1
0	0	1
1	1	0

MODULO 2 ARITHMETIC

- Contoh :

Data 1001 -> M = 4 bit

Constant 1011 -> r = 4 bit

Sisi Pengirim

- Tambahkan data yang dikirim dengan r-1 bit 0
- M ->1001 000
- Bagi M (1001000) dengan r (1011), maka akan didapat hasil bagi (*quotient*) dan sisa pembagian (*remainder*).
- Tambahkan *remainder* ke data asal (M asal) menggantikan data tambahan r-1 bit 0 -> *Frame Check Sequence* (FCS).
- 1001 000 -> 1001 110

MODULO 2 ARITHMETIC

- Sisi Penerima
- Frame yang diterima (1001 110) dibagi dengan r, jika tidak ada *remainder* maka dianggap tidak ada error.

CONTOH

- **Send**
  - M(x) = 110011 → x^5+x^4+x+1 (6 bits)
  - P(x) = 11001 → x^4+x^3+1 (5 bits, n = 4) → 4 bits of redundancy
  - Form x^n M(x) → 110011 0000 → $x^9+x^8+x^5+x^4$
  - Divide x^nM(x) by P(x) to find C(x)

```
          100001
11001 ) 1100110000
        11001
            10000
            11001
             1001 = C(x)
```

Send the block 110011 1001

- **Receive**

```
11001 ) 1100111001
        11001
            11001
            11001
            00000
```

↓

No remainder → Accept

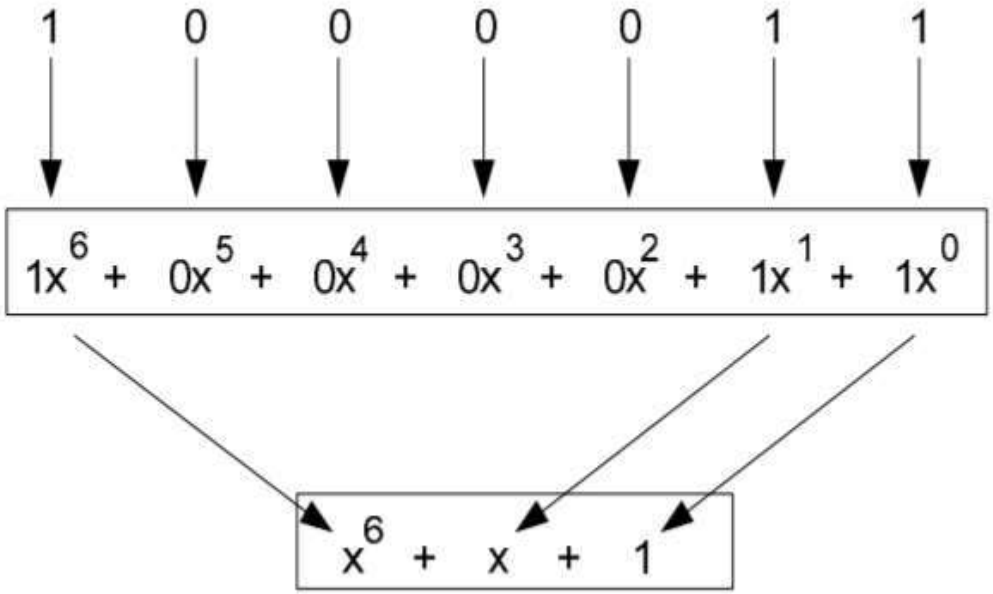
POLYNOMIALS

- Pola bit 0 dan 1 dapat direpresentasikan sebagai polynomial dengan koefisien 0 atau 1.

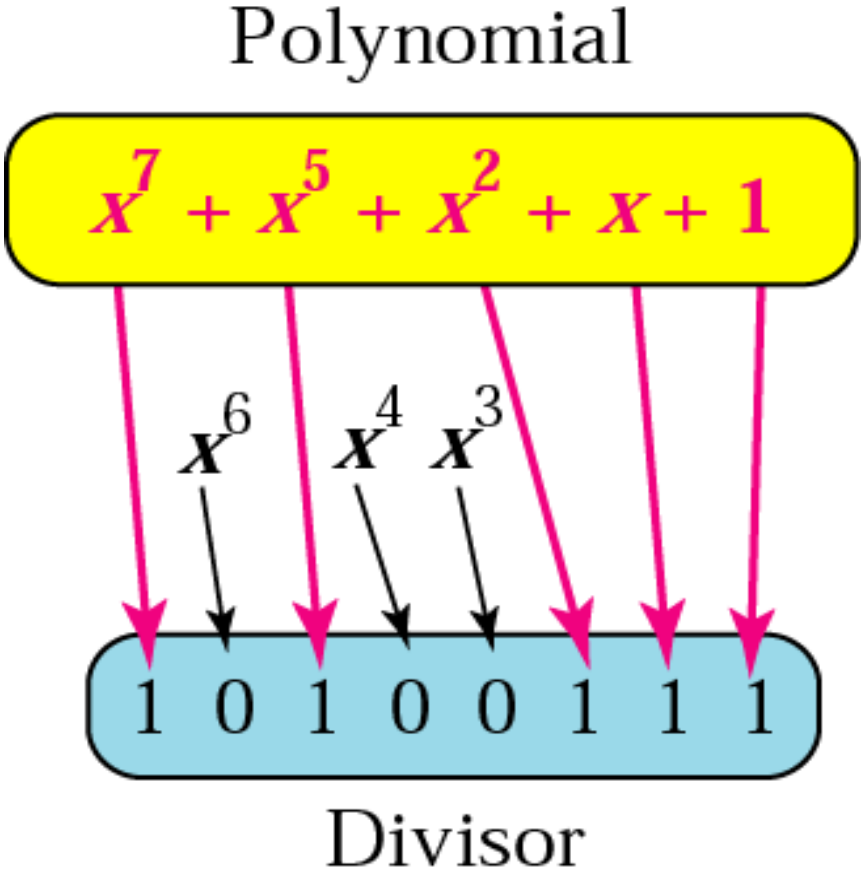
$$1x^6 + 0x^5 + 0x^4 + 0x^3 + 0x^2 + 1x^1 + 1x^0$$

- Koefisien merupakan nilai dari sebuah bit.
- Pangkat merupakan posisi dari bit.

POLYNOMIALS



$$\begin{array}{r} x^5 + x^4 + x^2 \\ \hline x^6 + x^4 + x^2 \\ x^6 + x^5 \end{array} +$$


Polynomial
$x^7 + x^5 + x^2 + x + 1$
x^6 x^4 x^3
1 0 1 0 0 1 1 1
Divisor

OPERASI POLYNOMIALS

- Proses perkalian suku polynomial, dengan menjumlahkan pangkat dari setiap suku.
- Contoh: $X^4 * X^3 = X(4+3)=X^7$
- Proses pembagian polynomial, dengan mengurangi pangkat suku pertama dengan pangkat suku ke-2.
- Contoh: $X^5 : X^3 = X(5-3) = X^2$
- Proses pembagian polynomial konsepnya sama dengan pembagian biner.

OPERASI POLYNOMIALS

Contoh:

Message M = 1001 (4 bit)

Constant r = 1011 (4 bit)

- **Sisi Pengirim**

Tambahkan data yang dikirim dengan r-1 bit 0

- M menjadi 1001 000
- Bagi M dengan r(dalam format polynomial)
- M = 1001 000 = **X6+ X3**
- r = 1011= **X3 + X + 1**

OPERASI POLYNOMIALS

- Tambahkan remainder ke data asal (M asal)
- X6+ X3+ X2+ X

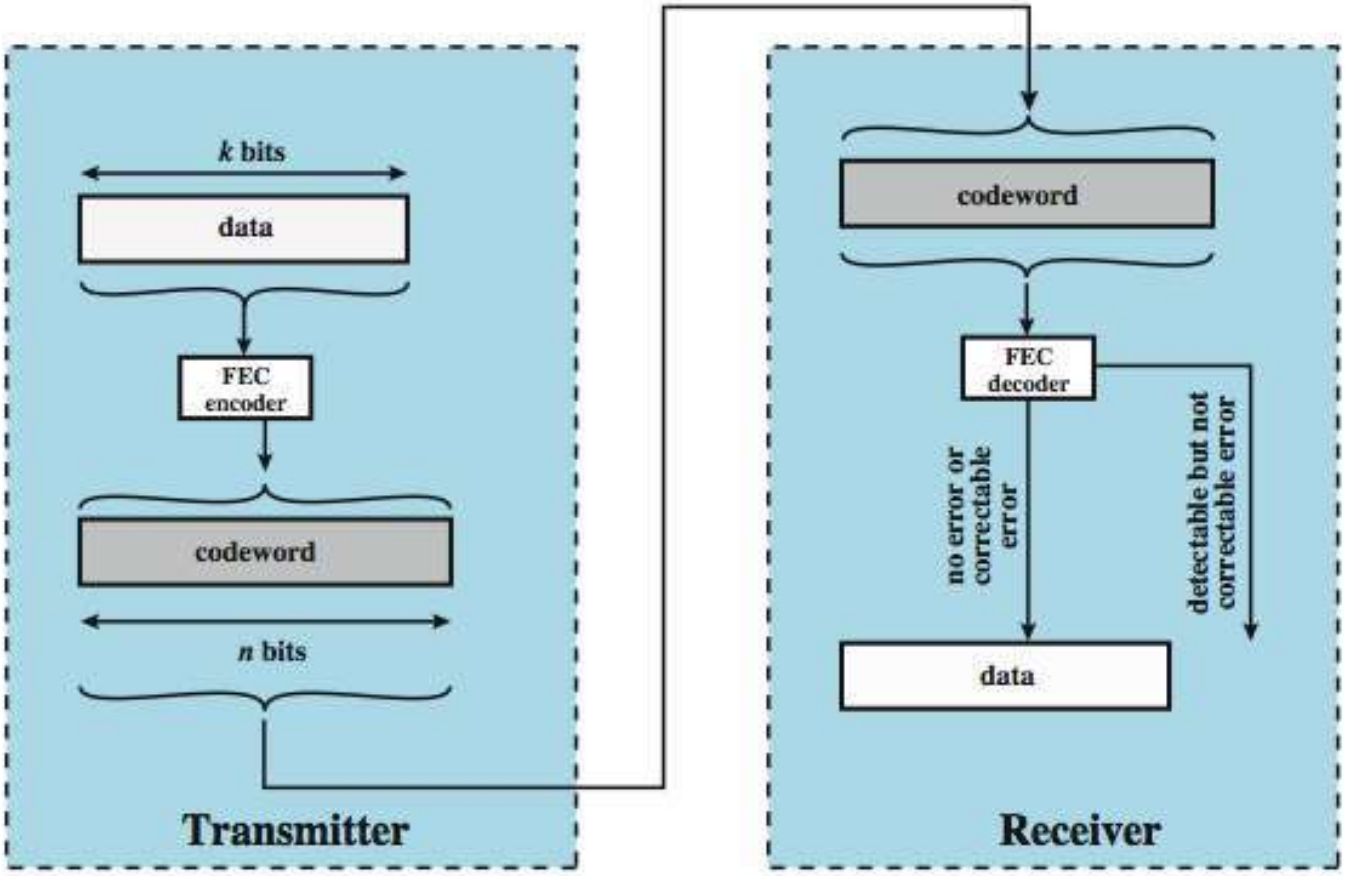
$$
\begin{array}{r|l}
 & X^3 + X \\
\hline
X^3 + X + 1 & X^6 + \quad + X^3 \\
 & X^6 + X^4 + X^3 \\
\hline
 & X^4 \\
 & X^4 + X^2 + X \\
\hline
 & X^2 + X
\end{array}
$$

OPERASI POLYNOMIALS

- **Sisi Penerima:**
- Kode polynomial yang diterima:(X6+ X3+ X2+ X) dibagi dengan r.
- Jika tidak ada *remainder* maka dianggap tidak ada error.

$$
\begin{array}{r}
X^3+X \\
\hline
X^3+X+1 \,/\, X^6+X^3+X^2+X \\
X^6+X^4+X^3 \\
\hline
X^4+X^2+X \\
X^4+X^2+X \\
\hline
0
\end{array}
$$

ERROR DETECTION PROCESS



- Pada akhir transmisi, setiap blok k-bit data dipetakan ke dalam n-bit block (n > k) disebut ***codeword***, menggunakan ***encoder* FEC (*correction error forward*)**. *Codeword* tersebut kemudian ditransmisikan.
- Selama transmisi, sinyal dapat mengalami gangguan, yang dapat menghasilkan kesalahan bit dalam sinyal.
- Pada *receiver*, sinyal yang masuk di demodulasi untuk menghasilkan bit string yang mirip dengan *codeword* asli tapi mungkin mengandung kesalahan. Blok ini dilewatkan melalui *decoder* FEC, dengan satu dari empat kemungkinan hasil:

1. Jika tidak ada kesalahan bit, *input* ke *decoder* FEC identik dengan *codeword* asli, dan *decoder* menghasilkan blok data asli sebagai *output.*
2. Untuk pola kesalahan tertentu adalah mungkin bagi *decoder* untuk mendeteksi dan memperbaiki kesalahan-kesalahan, *decoder* FEC mampu memetakan blok ini ke blok data asli.
3. Untuk pola kesalahan tertentu*, decoder* dapat mendeteksi tetapi tidak memperbaiki kesalahan, *decoder* hanya melaporkan kesalahan yang tidak dapat diperbaiki.
4. Untuk tertentu, biasanya jarang, pola kesalahan, *decoder* tidak mendeteksi bahwa kesalahan telah terjadi dan peta blok data yang masuk ke blok yang berbeda dari aslinya.

MEKANISME KOREKSI ERROR

- Menambahkan redudansi ke pesan yang ditransmisikan.
- *Receiver* dapat menyimpulkan pesan asli meskipun dengan beberapa error.
- Contoh kode koreksi error blok
  - Petakan *input k* bit *input* kedalam sebuah *codeword n* bit.
  - Masing-masing secara jelas berbeda.
  - Jika terdapat error maka diasumsikan panjang *codeword* yang dikirim mendekati jumlah yang diterima.
- Mengurangi keefektifan kecepatan data.

REFERENSI

1. Komunikasi Data, William Stalling.

2. Komunikasi Data, Dony Ariyus dan Rum Andri K.R.

3. Fahrudin.dosen.ittelkom-pwt.ac.id/wp-content/uploads/sites/70/2018/01/Deteksi-dan-koreksi-kesalahan.pdf