APLIKASI PEMBELAJARAN METODE *HAMMING CODE* DENGAN *SINGLE ERROR CORRECTION*

Oleh:
Ruth Meivera Siburian [1)]
Jeremia Siregar [2)]
Sumilina Waruwu [3)]
Institut Sains Teknologi TD. Pardede [1,2,3)]
E-mail:
vmanut@yahoo.com [1)]
sumilina08062001@gmail.com [2)]
jeremiasiregar@istp.ac.id [3)]

ABSTRACT

When transmitting digital data, it can experience failure (error). Errors result in changes to the contents of the data being transferred. In computer science, there are various kinds of logic to detect and correct these errors. One way to detect simple errors is to use Hamming Code with single error correction. Hamming Code is an error detection method that is capable of detecting multiple errors, but only able to correct one error (single error correction). This error detection method is very suitable for use. in situations where there are some random errors (randomly occurring errors). The Hamming Code method inserts (n + 1) check bits into 2n data bits. This method uses the XOR (Exclusive – OR) operation in the error detection process. Input and output data from the Hamming Code method are binary numbers.

Key : Haming Code, Data Communication, Single Error

ABSTRAK

Pada saat transmisi data digital dapat mengalami kegagalan (*error*).*Error* mengakibatkan perubahan isi dari data yang ditransfer.Dalam ilmu komputer, terdapat bermacam – macam logika untuk mendeteksi dan mengoreksi *error* tersebut. Salah satu cara untuk mendeteksi *error* yang sederhana adalah dengan menggunakan *Hamming Code* dengan *single error correction*.*Hamming Code* adalah suatu metoda pendeteksi *error* yang mampu mendeteksi beberapa *error*, namun hanya mampu mengoreksi satu *error* (*single error correction*).Metoda pendeteksi *error* ini sangat cocok digunakan pada situasi dimana terdapat beberapa *error* yang teracak (*randomly occuring errors*). Metoda *Hamming Code* menyisipkan (n + 1) *check bit* ke dalam 2^n data *bit*. Metoda ini menggunakan operasi *XOR* (*Exclusive – OR*) dalam proses pendeteksian *error*. *Input* dan *output* data dari metoda *Hamming Code* berupa bilangan biner.

Kata Kunci : Haming Code, Komunikasi Data, Single Error

1. PENDAHULUAN

Pada saat transmisi data digital berlangsung, terkadang data yang ditransfer dapat mengalami kegagalan (Error). Error yang terjadi dapat mengakibatkan perubahan isi dari data yang ditransfer.

Dalam ilmu komputer terdapat bermacam-macam Metode dalam mendeteksi serta mengoreksi error tersebut. Salah satu metode untuk mendeteksi error adalah dengan menggunanakan Hamming Code dengan single error correntionn.

Hamming Code merupakan salah satu metode pendeteksierror yang mampu mendeteksi beberapa error, namun hanya mampu mengoreksi satu error (single error correntionn). Metode ini sangat cocok digunakan pada situasi dimana terdapat beberapa error yang teracak (randomly occurring errors). Algoritma Hamming Code menyisipkan (n + 1) check bit ke dalam 2n data bit. Algoritma ini menggunakan operasi XOR (Exclusive - OR) dalam proses pendeteksian error, dengan nilai input dan output data berupa bilangan biner.

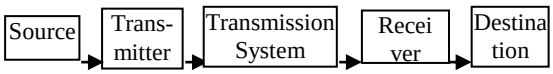
Tujuan dan Manfaat Penulisan

Tujuan penyusunan tugas akhir (skripsi) ini adalah untuk merancang suatu perangkat lunak pembelajaran yang mampu menjelaskan teknik pendeteksi error dari metoda *Hamming Code* secara tahap demi tahap.

Manfaat dari penyusunan tugas akhir (skripsi) ini yaitu :Membantu pembelajaran metoda *Hamming Code*. Sebagai fasilitas pendukungdalam proses belajar – mengajar.

2. TINJAUAN PUSTAKA

Komunikasi digital merupakan sesuatu hal yang penting dalam suatu infrastruktur komunikasi yang sedang dibangun. Kerangka / model komunikasi dasar yang sederhana adalah sebagai berikut:



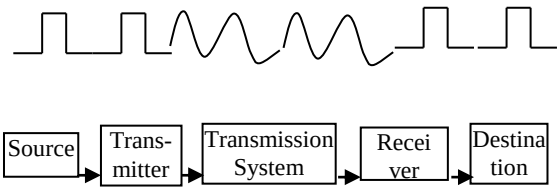
Gambar 2.1 Kerangka Komunikasi Dasar

Keterangan:

1. *Source* (sumber), merupakan suatu alat untuk membangkitkan data sehingga dapat ditransmisikan, contoh : telepon dan PC (*Personal Computer*).
2. *Transmitte*r (pengirim), berfungsi untuk mengkonversi data ke dalam bentuk signal analog untuk ditransmisikan.
3. *Source* dan *Transmitter* merupakan bagian dari *Source System.*
4. *Transmission System* (system transmisi) berupa jalur transmisi tunggal (*single transmission line*) atau jaringan kompleks (*complex network*)yang menghubungkan antara sumber dengan *destination* (tujuan).
5. *Receiver* (penerima) berfungsi untuk mengkonversi signal analog yang diterima ke dalam bentuk data digital. Destinasi berfungsi untuk menangkap data yang dihasilkan oleh receiver.

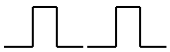
6. *Receiver* dan *Destination* merupakan bagian dari *Destination System.*

Proses transmisi data secara terperinci dapat dilihat pada gambar di bawah ini :



Gambar 2.2 Rincian Proses Transmisi Data

Keterangan :


:Digital *bit* Stream


: Analog signal

a) Informasi yang di-*input*
b) Data yang di-*input*
c) Signal yang akan ditransmisikan berupa signal analog
d) Signal yang diterima berupa signal analog
Data *output*

2.2. Error Detection

Pada saat data berada dalam transmission systemterdapat kemungkinan data terkorupsi (data error). Data error tersebut akan diperbaiki oleh receiver melalui proses error detection dan error correction.

Proses error detection dilakukan oleh transmitter dengan cara menambahkan beberapa bit tambahan ke dalam data yang akan ditransmisikan. Proses error detection dan correction ini sering digunakan pada CD Players, High speed modem, dan telepon selular (cellular phones).

Secara garis besar, metoda pendeteksi error (error detection) dan pengontrol error (error controller) dapat dibagi menjadi 2 bagian besar dengan perincian sebagai berikut:

1. Error Detection and Correction.
2. Error Controller, yaitu: Automatic Repeat Request (ARQ)

2.3. Hamming Code

Hamming Code ditemukan oleh Richard W. Hamming pada tahun 1940-an. Metode Hamming Code merupakan salah satu metoda pendeteksi error (error detection) yang mampu untuk mendeteksi beberapa error, namun hanya mampu mengoreksi satu error (single error correction).

Metode pendeteksi error ini sangat cocok digunakan pada situasi dimana terdapat beberapa error yang teracak (randomly occuring errors).

Hamming Code merupakan salah satu Metode pendeteksi error (error detection) dan pengoreksi error (error correction) yang paling sederhana. Metode

ini menggunakan operasi logika XOR (Exclusive – OR) dalam proses pendeteksian error (error detection) maupun proses pengoreksian error (error correction), sedangkan input dan output data dari metode Hamming Code berupa bilangan biner.

1. Operasi Logika

   Operasi – operasi logika dasar terdiri dari operasi NOT, operasi AND, dan operasi OR. Operasi logika lain merupakan kombinasi dari operasi - operasi logika dasar ini. Salah satu operasi logika hasil kombinasi operasi logika dasar adalah operasi XOR (Exclusive - OR).

2. Cara Kerja Metode Hamming Code

   Metode Hamming Code menyisipkan beberapa buah check bit ke dalam data. Jumlah check bit yang di sisipkan ter gantung pada panjang data.

   Rumus perhitungan untuk menghitung jumlah check bit yang harus disisipkan ke dalam data adalah sebagai berikut.

   Untuk data 2n bit, jumlah check bit yang disisipkan ada sebanyak c = (n + 1) bit.

3. HASIL DAN PEMBAHASAN

3. 1Pembahasa

3.1.1Metode Hamming Code

Metode Hamming Code melakukan proses pengecekan *error* dengan cara menyisipkan n – 1 Check *bit* untuk 2^n*bit* data (*input* dan output). Lalu dilakukan perhitungan nilai dari check *bit* tersebut untuk data *input* dan data output. Setelah itu, nilai check *bit* tersebut di – XOR-kan dan dilakukan perbandingan antara nilai check *bitinput* dan nilai check *bit* output. Apabila nilainya todak sama, maka terdapat *error* (kesalahan). Sebaliknya, apabila nilainya sama maka tidak terdapat *error* dalam data output.

Proses kerja dari metode Hamming Code ini terbagi menjadi dua bagian yaitu proses encoding dan proses decoding.

Untuk memahami proses penyelesaian metode Hamming Code, amatilah contoh di bawah ini:

Misalkan data *input* 10111010 ditransmisikan. Data yang diterima oleh receiver berupa 10111110, maka proses pendeteksi *error* (*error* detection) dan pengoreksi *error* (*error* correction) dari metode Hamming Code adalah sebagai berikut:

1. Hitung panjang data *input* dan output dari metode Hamming Code. Untuk data 2^n*bit*. Jumlah check *bit* yang disisipkan ada sebanyak c = (n + 1) *bit*. Panjang *input* dan output data = 8 *bit* = 2^3 sehingga jumlah check *bit* : 3 + 1 = 4 dan panjang data *input* dan output dari metode Hamming Code = 12 *bit*.
2. Tandai semua posisi *bit* yang merupakan posisi dari check *bit*. Posisi

selain posisi check *bit* merupakan posisi dari data *bit*.

Rumus perhitungan posisi Check *bit* :

$$C_i = 2^{i-1}$$

$C_1 = 2^{1-1} = 2^0 = 1$

$C_2 = 2^{2-1} = 2^1 = 2$

$C_3 = 2^{3-1} = 2^2 = 4$

$C_1 = 2^{1-1} = 2^0 = 1$

$C_1 = 2^{1-1} = 2^0 = 1$

3. Tentukan rumus perhitungan dari check *bit* C_1, C_2, C_3 dan C_4.

C_1 = lihat posisi *bit* paling kanan dari member position di mana *bit* bernilai 1, kecuali posisi dari check *bit*. Semua data *bit* yang berada pada posisi tersebut diambil.

$C_1 = M_1 \oplus M_2 \oplus M_4 \oplus M_5 \oplus M_7$

C_2 = lihat posisi *bit* kedua dari kanan dari member position di mana *bit* bernilai 1, kecuali posisi dari check *bit*. Semua data *bit* yang berada pada posisi tersebut diambil.

$C_2 = M_1 \oplus M_3 \oplus M_4 \oplus M_6 \oplus M_7$

C_3 = lihat posisi *bit* ketiga dari kanan dari member position di mana *bit* bernilai 1, kecuali posisi dari check *bit*. Semua data *bit* yang berada pada posisi tersebut diambil.

$C_3 = M_2 \oplus M_3 \oplus M_4 \oplus M_8$

C_4 = lihat posisi *bit* keempat dari kanan dari member position di mana *bit* bernilai 1, kecuali posisi dari check *bit*. Semua data *bit* yang berada pada posisi tersebut diambil

$C_4 = M_5 \oplus M_6 \oplus M_7 \oplus M_8$

Tabel 3.1 Posisi check bit dan data bit

Bit Position	Member Position	Check Bit	Data Bit
12	1100		M_8
11	1011		M_7
10	1010		M_6
9	1001		M_5
8	1000	C_4	
7	0111		M_4
6	0110		M_3
5	0101		M_2
4	0100	C_3	
3	0011		M_1
2	0010	C_2	
1	0001	C_1	

4. Hitung nilai dari check *bit* untuk data *input* dan data output

Data *input* : 1 0 1 1 0 1 1 0

M1 M2 M3 M4 M5 M6 M7 M8

$C_1 = M_1 \oplus M_2 \oplus M_4 \oplus M_5 \oplus M_7 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 = 0$

$C_2 = M_1 \oplus M_3 \oplus M_4 \oplus M_6 \oplus M_7 = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 0$

$C_3 = M_2 \oplus M_3 \oplus M_4 \oplus M_8 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$

$C_4 = M_5 \oplus M_6 \oplus M_7 \oplus M_8 = 1 \oplus 0 \oplus 1 \oplus 0 = 0$

Nilai check *bit* untuk data *input*,

0 0 0 0

C4 C3 C2 C1

Setelah diproses maka data *input* menjadi,

0 1 0 1 **0** 1

1 0 **0** 1 **0**

0

M8 M7 M6 M5 **C4** M4

M3 M2 **C3** M1 **C2**

C1

Kemudian dilanjutkan proses perhitungan untuk data output seperti berikut,

Data output : 1 0 1 1

1 1 1 0

M1 M2 M3 M4

M5 M6 M7 M8

$C_1 = M_1 \oplus M_2 \oplus M_4 \oplus M_5 \oplus M_7 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 = 0$

$C_2 = M_1 \oplus M_3 \oplus M_4 \oplus M_6 \oplus M_7 = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 0$

$C_3 = M_2 \oplus M_3 \oplus M_4 \oplus M_8 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$

$C_4 = M_5 \oplus M_6 \oplus M_7 \oplus M_8 = 1 \oplus 1 \oplus 1 \oplus 0 = 0$

Nilai check *bit* untuk data output:

1 0 1 0

C4 C3 C2 C1

Setelah diproses maka data output menjadi:

0 1 0 1 **1** 1

1 0 **0** 1 1

0

M8 M7 M6 M5 **C4** M4

M3 M2 **C3** M1 **C2**

C1

5. Nilai check *bitinput* tidak sama dengan nilai check *bit* output berarti terdapat kesalahan (*error*).
6. Lakukan operasi XOR terhadap check *bitinput* dan check *bit* output

C4	**C3**	**C2**	**C1**	
Input	0	0	0	0
Output		1	0	1
	0			
.....				⊕
1		0	1	0

7. Konversikan hasil operasi XOR kedalam bentuk bilangan decimal.
   $(1010)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 8 + 0 + 2 + 0 = 10$
8. Nilai dari hasil operasi XOR lebih kecil daripada panjang data *input* dan nilai dari hasil operasi XOR bukan merupakan posisi dari check *bit*, maka hanya terdapat satu kesalahan (*error*) dan hasil operasi merupakan posisi *bad bit.*

3.2 Perancangan

3.2.1 Perancangan Struktur Chart

Perancangan lunak yang dirancang ini terdiri dari modul utama, yaitu metode Hamming Code, modul ini masing-masing terdiri dari tiga buah sub modul yaitu sub modul *input* pesan, encoding dan decoding.

Algoritma decoding dari metode Hamming Code dapat dilihat pada gambar berikut:

3.2.2 Penggunaan Komponen

Perangkat lunak ini dirancang dengan menggunakan bahasa pemrograman Microsoft Visual Basic 6.0 dengan menggunakan beberapa objek dasar seperti:

1. Label, yang digunakan untuk menampilkan keterangan.
2. Shape, yang digunakan sebagai dekorasi tambahan.
3. MSFlexgrid, yang digunakan untuk menampilkan table Galois Field dan BCH.
4. Command button, yang digunakan sebagai tobol eksekusi.
5. Combo box, yang digunakan untuk menyediakan pilihan kecepatan animasi.
6. Text Box, yang digunakan sebagai tempat pengisian pesan dan juga untuk menampilkan hasil eksekusi.
7. Picture Box, yang digunakan untuk menampilkan animasi proses simulasi.

3.2.3 Perancangan *Input*

Perangkat lunak memiliki dua buah rancangan tampilan *input* yaitu:

1. *Form* 'Main'

Form ini berfungsi sebagai tempat pemilihan jenis pengkodean yang ingin ditampilkan. Rancangan tampilan dari *Form* ini dapat dilihat pada gambar :

Gambar 3.2 RancanganForm 'Main'

8. *Form*' *Input* Data Hamming Code'.

Form ini berfungsi sebagai tempat pengisian data *bit* pesan untuk Hamming Code.

Rancangan tampilan dari *Form* ini dapat dilihat pada gambar 3.3

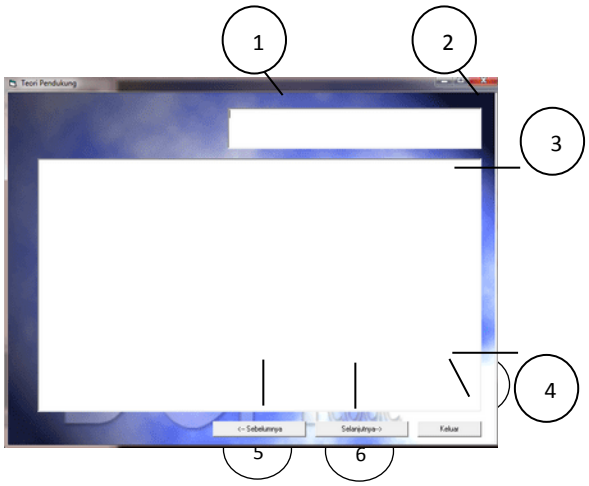
Gambar 3.3 Rancangan FormInput Data Hamming Code

3.2.4 Perancangan*Output*

Perangkat lunak ini memiliki beberapa buah rancangan tampilan *output* yaitu:

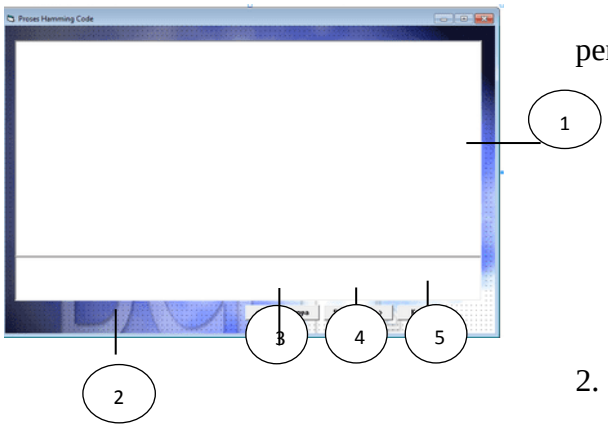
1. *Form*‘ Teori’

Form ‘Teori’ ini berfungsi untuk menampilkan teori-teori dasar yang berhubungan dengan perangkat lunak yang dirancang. Rancangan tampilan dari *Form*‘ Teori’ dapat dilihat pada gambar berikut:



Gambar 3.4 Rancangan Form Teori Data Hamming Code

***2 Form*‘ Proses Hamming’**



Gambar 3.5 Rancangan Form Proses Data Hamming Code

4. Algoritma Dan Implementasi

4.1. Algoritma

Adapun algoritma untuk pengecekan *error* pada saat transmisi data dengan *Hamming Code* dibagi menjadi 2 bagian yaitu,

1. Algoritma Pengecekan Data *Input* dan Data *Output* (pada *form input*).
2. Algoritma Penghasil Langkah – Langkah Pendeteksian *Single error* dengan *Hamming Code*.

4.1.1 Algoritma Pengecekan Data Input dan Data Output

Algoritma ini melakukan pengecekan terhadap data *input* dan data *output* apakah panjang data *input* dan *output* sesuai dengan jumlah *bit* yang dipilih..

Berikut ini adalah algoritma pengecekan,

1. Jika panjang *text* pada *textbox* txt*Input* tidak sama dengan 2 ^ (cbo*Bit*.ListIndex + 1), maka tampilkan pesan “Data *input* belum lengkap !” dan keluar darialgoritma.
2. Jika panjang *text* pada *textbox* txtOutput tidak sama dengan 2 ^ (cbo*Bit*.ListIndex + 1), maka tampilkan pesan “Data output belum lengkap !” dan keluar dari algoritma.
3. Set nilai cData*Input* = txt*Input*.Text.
4. Set nilai cDataOutput = txtOutput.Text

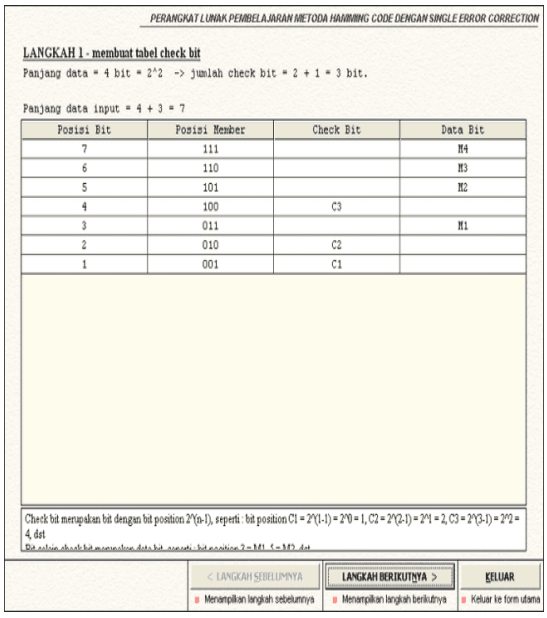
5. Lanjutkan algoritma ke algoritma penghasil langkah – langkah pendeteksian *error* dengan *Hamming Code*.

Hasil Proses Pendeteksian Single Error dengan metoda Hamming Code

Misalkan panjang data *input* dan output = 4 *bit*,

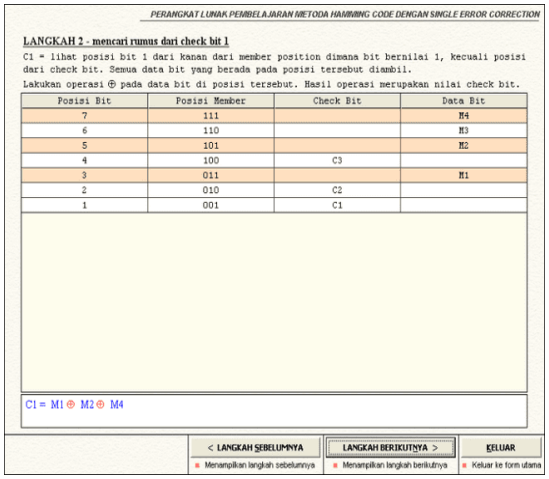
- *Data Input* = 1001.
- Langkah – langkah pendeteksian *single error* dengan *Hamming Code* adalah sebagai berikut,

o Langkah 1 – Membuat tabel *check bit*.



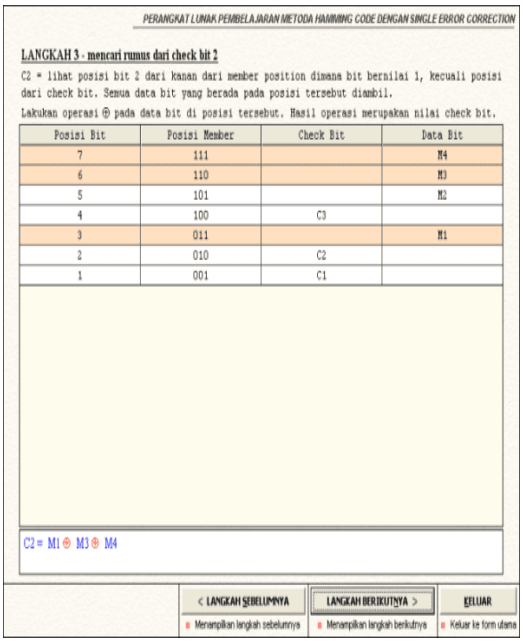
Gambar 4.1 : Membuat tabel check bit untuk data input = 1001, data output = 1000 (panjang data = 4 bit)

o Langkah 2 – Mencari rumus dari c*heck bit* - 1.



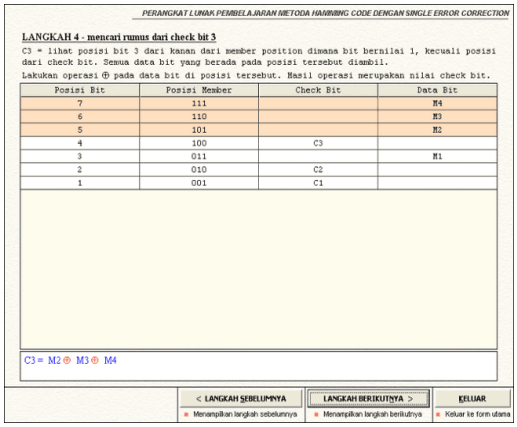
Gambar 4.2 : Mencari rumus dari check bit - 1 untuk data input = 1001, data output = 1000 (panjang data = 4 bit)

o Langkah 3 – Mencari rumus dari *check bit* - 2.



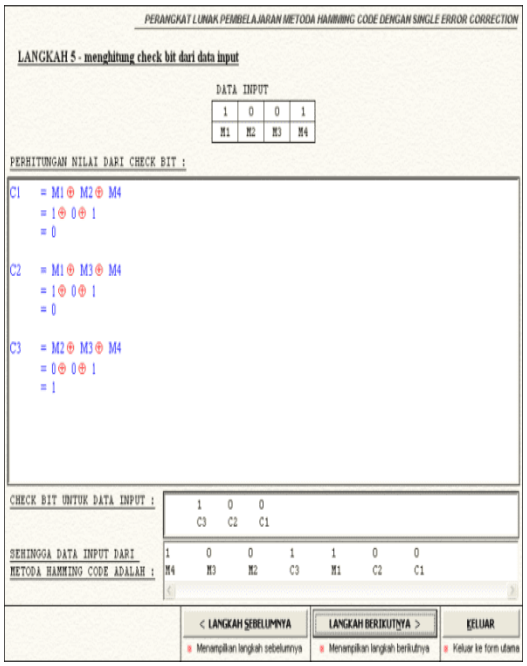
Gambar 4.3 : Mencari rumus dari check bit - 2 untuk data input = 1001, data output = 1000 (panjang data = 4 bit)

o Langkah 4 – Mencari rumus dari *check bit*-3.



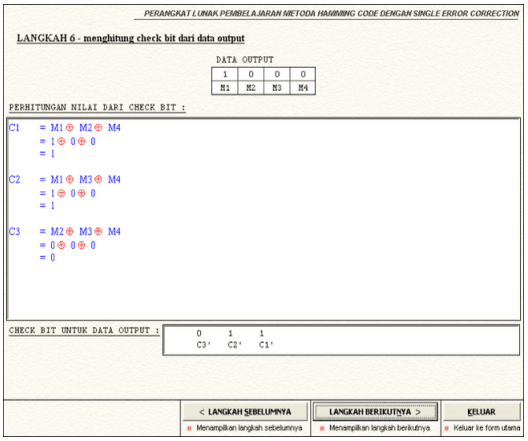
Gambar 4.4 : Mencari rumus dari check bit - 3 untuk data input = 1001, data output = 1000 (panjang data = 4 bit)

- Langkah 5 – Menghitung *check bit* dari data *input*.



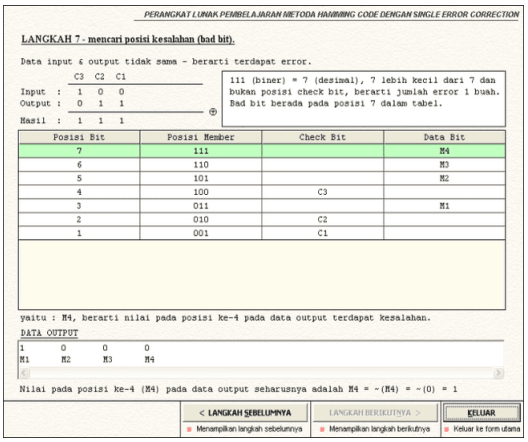
Gambar 4.5 : Menghitung check bit dari data input untuk data input = 1001, data output = 1000 (panjang data = 4 bit)

- Langkah 6 – Menghitung *check bit* dari data *output*.



Gambar 4.6 : Menghitung check bit dari data output untuk data input = 1001, data output = 1000 (panjang data = 4 bit)

- Langkah 7 – Mencari posisi kesalahan (*bad bit*).



Gambar 4.7 : Mencari posisi kesalahan (bad bit) untuk data input = 1001, data output = 1000 (panjang data = 4 bit)

5. SIMPULAN

Setelah menyelesaikan skripsi ini, maka penulis dapat menarik kesimpulan sebagai berikut:

1. Metode Hamming Code hanya mampu melakukan pengoreksian terhadap satu buah error(single error correction) saja.
2. Data input dan output pada metode HammingCode harus berupa hasil dari perpangkatan 2n dengan n harus lebih besar dari 1.
3. Panjang data input dan output pada metode

   Hamming Code minimal harus sama dengan 4 bit. Atau dengan perkataan lain, metode Hamming Code tidak mendukung data dengan panjang 2 bit.Metode hamming code tidak mampu melakukan pengecekkan terhadap posisi data error (bad bit) yang lebih dari satu buah.

Saran

Penulis ingin memberikan beberapa saran untuk pengembangan lebih lanjut pada perancangan perangkat lunak pembelajaran metoda pendeteksi dan pengoreksi kesalahan (error detection and correction). Saran-saran tersebut antara lain:

1. Dapat dikembangkan suatu perangkat lunak dengan menggunakan metoda pendeteksi dan pengoreksi error (error detection and correction method) lainnya yang lebih canggih daripada metoda Hamming Code seperti metoda CRC (Cyclic Redundancy Check), gabungan metoda LRC (Longitudinal Redundancy Check) dan VRC (Vertical Redundancy Check) yang mampu melakukan pengecekan terhadap lebih dari satu buah error dan sebagainya.
2. Perangkat lunak dapat ditambahkan beberapa animasi lainnya agar lebih menarik. Perangkat lunak juga dapat dikembangkan menjadi sebuah perangkat lunak multimedia denganmenambahkan efek suara, efek grafik dan efek-efek lainnya

6. DAFTAR PUSTAKA

Ariyus dan Aandri, 2008 **komunikasi data,** Andi Yogyakarta

Green, D. C.,2007.**Komunikasi Data**, Andi Yogyakarta.

Malvino, Albert P., Tjia May On, 1994.***Elektronika Komputer Digital***, Edisi Kedua, Pener*bit* Erlangga, Jakarta.

Djoko Pramono,2000. **Mudah Menguasai Visual Basic ,**PT. Elex Media Komputindo, Jakarta.

Rahadian, 2001. **Pemrograman Microsoft Visual Basic** PT. Elex Media Komputindo, Jakarta.

Rahmat Putar, 2005. ***The Best Source Code Visual Basic***, PT. Elex Media Komputindo, Jakarta.

Raplh J. Smith, 1992.**Rangkaian, Piranti dan Sistem**, Edisi Keempat, Jilid 1, Pener*bit* Erlangga.