

PERANCANGAN KODE KOREKSI KESALAHAN REED SOLOMON CODE RS(31,27) DENGAN MENGGUNAKAN BAHASA VHDL

Ali Mahmudi¹⁾, Jasmani²⁾

¹⁾Institut Teknologi Nasional Malang, Malang.

¹⁾email: amahmudi@hotmail.com

²⁾Institut Teknologi Nasional Malang, Malang

Abstract

Error correction technology is widely applied in the era of data / information exchange today. This technology is widely used to reduce the error rate on the exchange of data / information to acceptable limits. Hamming Code, Reed Muller Code, Golay Code, BCH Code and Reed Solomon Code are some examples of error correction technology that is widely used. This paper presents the Reed Solomon Code. Here, the Reed Solomon Code is decoded using the Welch Berlekamp algorithm. The Reed Solomon RS code (31, 27) is designed and designed using the VHDL language, and then simulated using the Altera ModelSim application. The simulation results show that the RS code (31, 27) goes as expected and is able to correct up to 2 errors.

Keywords: Reed Solomon Code, Error Correction, RS code, Welch Berlekamp Algorithm.

PENDAHULUAN

Di era computer dewasa ini, sistem pertukaran informasi yang handal sangat diperlukan. Sistem komunikasi yang handal dapat diperoleh dengan menggunakan media komunikasi yang bersih dari gangguan / noise. Sayangnya, media komunikasi yang bersih dari gangguan / noise tersebut biasanya relatif mahal.

Cara lain adalah dengan menerapkan teknologi koreksi kesalahan pada data yang dikirimkan[1][2]. Penerapan teknologi koreksi kesalahan adalah untuk mengurangi tingkat kesalahan pada data / informasi. *Kode BCH, Kode Golay, Kode Hamming, dan Kode Reed Solomon* adalah beberapa teknologi koreksi kesalahan yang cukup populer dewasa ini. Makalah ini mengetengahkan tentang *Kode Reed Solomon*, atau dikenal dengan Kode RS. Kode RS adalah suatu kode koreksi kesalahan yang banyak dipakai pada aplikasi komputer dan data komunikasi. Kode ini dipakai pada aplikasi *Compact Disc* dan *Digital Video* sampai dengan komunikasi luar angkasa pada ekspedisi NASA *Voyager* dan *Hubble Space Telescope*[1][3][4][5].

Makalah ini mengetengahkan tentang RS enkoder dan RS dekoder, dan penekanan pada algoritma Welch Berlekamp.

Kode RS yang dibahas disini memiliki karakteristik 31 simbol, dengan rincian 27 simbol informasi dan 4 simbol *parity check*. RS (31, 27) code ini mampu mengoreksi

kesalahan sampai dengan 2 simbol, setiap simbol terdiri dari 5 bit.

KAJIAN PUSTAKA

Matematika biner medan galois

Untuk memahami Kode RS, harus dimulai dari matematika biner medan Galois, *Galois Field GF(2^m)*. Kode RS(31, 27) menggunakan GF(2⁵) dengan nilai polinomial primitif

$$p(x) = x^5 + x^2 + 1 \quad (1)$$

Nilai dari medan Galois GF(2⁵) dapat ditabulasikan seperti tabel 1.

Berdasarkan persamaan X(1)X, maka dapat dituliskan bahwa

$$\alpha^5 = \alpha^2 + 1 \equiv 00101$$

sehingga berlaku

$$\alpha^6 = \alpha^5 \cdot \alpha = (\alpha^2 + 1)\alpha = \alpha^3 + \alpha \equiv 01010$$

REED SOLOMON ENKODER

RS(n,k) code pada Medan Galois GF(2^m) mampu memperbaiki keasalan sampai dengan *t* simbol, dimana masing-masing simbol terdiri dari *m* bit. *RS(n,k)* memiliki panjang kode *n* simbol, dimana *n*=2^{*m*}-1. Dalam *n* simbol tersebut, terdapat simbol informasi sebanyak *k* (=n-2*t*) simbol dan simbol paritas sebanyak 2*t* simbol.

Tabel 1. Medan Galois $GF(2^5)$

No	Exponen	Polinomial					Biner
		α^4	α^3	α^2	α^1	α^0	
0	α^0	0	0	0	0	1	00001
1	α^1	0	0	0	1	0	00010
2	α^2	0	0	1	0	0	00100
3	α^3	0	1	0	0	0	01000
4	α^4	1	0	0	0	0	10000
5	α^5	0	0	1	0	1	00101
6	α^6	0	1	0	1	0	01010
7	α^7	1	0	1	0	0	10100
8	α^8	0	1	1	0	1	01101
9	α^9	1	1	0	1	0	11010
10	α^{10}	1	0	0	0	1	10001
11	α^{11}	0	0	1	1	1	00111
12	α^{12}	0	1	1	1	0	01110
13	α^{13}	1	1	1	0	0	11100
14	α^{14}	1	1	1	0	1	11101
15	α^{15}	1	1	1	1	1	11111
16	α^{16}	1	1	0	1	1	11011
17	α^{17}	1	0	0	1	1	10011
18	α^{18}	0	0	0	1	1	00011
19	α^{19}	0	0	1	1	0	00110
20	α^{20}	0	1	1	0	0	01100
21	α^{21}	1	1	0	0	0	11000
22	α^{22}	1	0	1	0	1	10101
23	α^{23}	0	1	1	1	1	01111
24	α^{24}	1	1	1	1	0	11110
25	α^{25}	1	1	0	0	1	11001
26	α^{26}	1	0	1	1	1	10111
27	α^{27}	0	1	0	1	1	01011
28	α^{28}	1	0	1	1	0	10110
29	α^{29}	0	1	0	0	1	01001
30	α^{30}	1	0	0	1	0	10010
0	α^0	0	0	0	0	1	00001

Informasi = k symbol	Paritas = $n-k$ symbol
------------------------	------------------------

Gambar 1. Sistematik RS code.

Simbol informasi sebanyak k ($=n-2t$) simbol, dan ini dapat diperlakukan sebagai suatu polinomial seperti ditunjukkan pada persamaan berikut ini.

$$f(x) = x^{2t} (s_{n-2t} + s_{n-2t-1}x + \dots + s_2x^{n-2t-1} + s_1x^{n-2t}) \quad (2)$$

dimana s_i adalah simbol ke- i yang ditransmisikan. Banyaknya simbol paritas adalah $2t$ simbol. Nilai simbol paritas diperoleh dari nilai sisa atas hasil bagi antara simbol informasi $f(x)$ dengan polinomial pembangkit $g(x)$ ($=generator polynomial$) $X[6]XX[7]$. Hal ini ditunjukkan pada persamaan X(3)

$$f(x) \text{ } \overline{\text{ }} g(x) \quad (3)$$

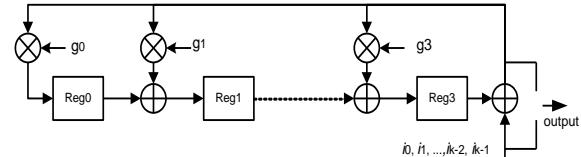
Polinomial pembangkit $g(x)$ didefinisikan pada X(4)

$$g(x) = \prod_{j=0}^{2t-1} (x - \alpha^j) = \sum_{j=0}^{2t} g_j x^j \quad (4)$$

dimana α adalah elemen dari Galois Field $GF(2^n)$, dan g_j adalah koefisien dari polinomial $g(x)$ dimana $g_{2t}=1$.

Cara memperoleh polinomial pembangkit untuk RS(31, 27) ditunjukkan dalam persamaan X(5). Berikutnya, skematis diagram RS Enkoder RS(31, 27) ditunjukkan oleh gambar 2X[6]XX[7]XX[8]X.

$$\begin{aligned} g(x) &= \prod_{j=0}^3 (x - \alpha^j) \\ &= (x - \alpha^0)(x - \alpha^1)(x - \alpha^2)(x - \alpha^3) \\ &= x^4 + \alpha^{23}x^3 + \alpha^{17}x^2 + \alpha^{26}x + \alpha^6 \\ &= g_4x^4 + g_3x^3 + g_2x^2 + g_1x + g_0 \end{aligned} \quad (5)$$



Gambar 2. Rangkaian RS (31, 27) enkoder.

Jika polinomial $t(x)$ adalah kode data yang ditransmisikan (*transmitted codeword*), $r(x)$ adalah kode data yang diterima (*received codeword*) dan $e(x)$ adalah kode data yang terjadi kesalahan (*erroneous codeword*), maka ketiga polinomial tersebut di atas dapat dituliskan X(6)X.

$$r(x) = t(x) + e(x) \quad (6)$$

RS Dekoder

RS dekoder memiliki 4 bagian utama, yakni Re-enkoder, Algoritma Welch Berlekamp, Chien Search dan Koreksi kesalahan.

Proses Re-enkoder ini mirip dengan proses enkoding pada RS Enkoder. Pada RS (31, 27), proses re-enkoder menerima 27 simbol informasi, kemudian diproses dan menghasilkan 4 internal paritas simbol. Output dari Re-enkoder adalah penjumlahan modulo 2 antara 4 internal paritas simbol dengan 4 paritas simbol yang diterima dari media transmisi. Hasil modulo 2 nya, disebut dengan 'parity check simbol'.

Pada RS(31, 27), lokasi parity check adalah $\mathbf{H} = \{\square^0, \square^1, \square^2, \square^3\}$ dan nilai dari parity check simbol adalah $\mathbf{R} = \{R_0, R_1, R_2, R_3\}$. Tahap berikutnya adalah parity check simbol menjadi input dari Algoritma Welch Berlekamp.

Input dari Algoritma Welch Berlekamp adalah $(R_0, \square^0), (R_1, \square^1), (R_2, \square^2)$ dan yang terakhir adalah (R_3, \square^3) . Algoritma Algoritma Welch Berlekamp dapat dituliskan sebagai berikut

```

1.      ---- Initialisasi
2.      Q0(x)=1; N0(x)=0
3.      W0(x)=x; V0(x)=1
4.      J = 1

5.      Input = (Rd, αd)
6.      for d=0 step 1 until less than 2t-1
7.      begin loop
8.      D1= Qd (αd)Rd- Nd (αd)

9.      if D1=0 then % Branch A
10.     Wd+1=Wd (x-αd)      & Vd+1=Vd (x-αd)
11.     Qd+1=Qd           & Nd+1=Nd
12.     J = J + 1

13.    Else
14.    D2=[Wd (αd)Rd- Vd (αd)]

15.    If J != 0 then % Branch C
16.    Wd+1=Wd+Qd .D2/D1   & Vd+1=Vd+Nd .D2/D1
17.    Qd+1=Qd (x-αd)     & Nd+1=Nd (x-αd)
18.    J = J - 1

19.    Else % Branch B
20.    Wd+1=Qd (x-αd)      & Vd+1=Nd (x-αd)
21.    Qd+1=Wd+Qd .D2/D1   & Nd+1=Vd+Nd .D2/D1
22.    J = 1

23.  End if % Branch B dan C
24.  End if % Branch A
25. End of loop

```

Output dari algoritma Welch Berlekamp adalah polinomial $Q(x)$ dan $N(x)$. Polinomial

$Q(x)$ adalah polinomial penunjuk lokasi kesalahan, dan polinomial $N(x)$ adalah polinomial penunjuk nilai kesalahan.

Langkah berikutnya adalah menggunakan Chien Search untuk memecahkan polinomial $Q(x)$. Algoritma Horner biasanya dipakai untuk melakukan Chien Search.

$$Q(x) = q_d x^d + q_{d-1} x^{d-1} + \dots + q_0 \quad (7)$$

Dimana nilai x adalah mulai dari $\alpha^0, \alpha^1, \alpha^2, \alpha^3$, sampai dengan α^{30} untuk GF(2⁵). Jika $Q(x)=0$, maka lokasi kesalahan telah ditemukan, dan berikutnya nilai kesalahan dipecahkan dengan menggunakan polinomial $N(x)$, polinomial penunjuk nilai kesalahan.

Untuk mendapatkan nilai kesalahan, nilai konstan C perlu ditentukan lebih dahulu dengan menggunakan persamaan berikut

$$C = [(\alpha^0 \alpha^1 \dots \alpha^{2t-2})(\alpha^0 - \alpha^1) \dots (\alpha^0 - \alpha^{2t-1})]^{-1} \quad (8)$$

Untuk nilai t=2 dalam GF(2⁵), diperoleh nilai $C = \alpha^8$. Selanjutnya, nilai kesalahan dapat ditentukan dengan menggunakan persamaan berikut

$$E_g = \begin{cases} \frac{N(\alpha^g)}{P(\alpha^g)Q'(\alpha^g)} : \text{if } \alpha^g \notin H \\ n_g + \frac{N'(\alpha^g)}{P'(\alpha^g)Q'(\alpha^g)} : \text{if } \alpha^g \in H \end{cases} \quad (9)$$

$$P(x) = Cg(x) = \alpha^8 x^4 + \alpha^0 x^3 + \alpha^{25} x^2 + \alpha^3 x + \alpha^{14}$$

Ketika kesalahan ditemukan pada lokasi α^g dengan nilai kesalahan E_g , maka perbaikan simbol pada lokasi tersebut dapat diperoleh dengan persamaan berikut

$$c_g = E_g + s_g \quad (10)$$

HASIL

Gambar 3 berikut adalah tampilan script VHDL codec (encoder decoder). Yang meliputi RS(31, 27) enkoder dan RS(31, 27) dekoder.

Gambar 4 berikut adalah tampilan script VHDL RS (31, 27) enkoder. Sedangkan gambar 5 menunjukkan tampilan script VHDL Algoritma Welch

Berlekamp untuk RS (31, 27) dekoder.

```

1 -- FILE codec.vhd
2 -- This file to perform decoder RS (31, 27)
3
4 ENTITY decoder IS
5   PORT (res, clk, ck, ddatain: IN BIT;
6         output: OUT BIT);
7 END decoder;
8
9 ARCHITECTURE bvh OF decoder IS
10
11   -- FILE counter.vhd
12   COMPONENT counter
13     PORT (res, clk): IN BIT;
14     begin
15       cycle, parity, endseq, start_wb, ctrl_be : OUT BIT;
16       sel_poly_ip, dual_ip, hi_input: OUT BIT_VECTOR(4 DOWNTO 0);
17       dual_low, alghdgi: OUT BIT_VECTOR(4 DOWNTO 0));
18   END COMPONENT;
19
20   -- FILE encoder.vhd
21   COMPONENT encoder
22     PORT (ddatain, clock, reset, sel5, ctrl_parity: IN BIT;
23           enc_out, reenc_out: OUT BIT);
24   END COMPONENT;
25
26   COMPONENT calcRx
27     PORT(res, clk, parity, endseq, rx_ip, ctrl_rx, cycle: IN BIT;
28          sel_in: IN BIT_VECTOR(4 DOWNTO 0);
29          | output, capRx: OUT BIT_VECTOR(4 DOWNTO 0));
30   END COMPONENT;
31
32   -- FILE wb.vhd
33   COMPONENT wb
34     PORT (res, clk2, start, endseq, parity: IN BIT;
35           R, alpha, ringcount: IN BIT_VECTOR(4 DOWNTO 0);
36           H4, H3, H2, H1, H0: OUT BIT_VECTOR(4 DOWNTO 0);
37           Q4, Q3, Q2, Q1, Q0: OUT BIT_VECTOR(4 DOWNTO 0));
38   END COMPONENT;
39
40   -- pos_x.vhd
41   COMPONENT pos_x
42     PORT (nextx, c0, c1, c2, c3, c4: IN BIT_VECTOR(4 DOWNTO 0);
43           sum, sum2: OUT BIT_VECTOR(4 DOWNTO 0));
44   END COMPONENT;
45
46   -- buffer.vhd
47   COMPONENT buffer
48     PORT(res, clk, input: IN BIT;
49           output: OUT BIT);
50   END COMPONENT;
51
52   -- FILE inverse.vhd
53   COMPONENT inverse
54     PORT(res, v: IN BIT_VECTOR(4 DOWNTO 0);
55           vinv: OUT BIT_VECTOR(4 DOWNTO 0));
56

```

Gambar 3. Desain VHDL untuk RS(31, 27) codec.

```

1 -- File encoder.vhd
2 -- The Design of the PIPELINE (31, 27) ENCODER
3 -- Using BERLEKAMP MULTIPLIER and k=0
4
5
6 ENTITY encoder IS
7   PORT (ddatain, clock, reset, sel5, ctrl_parity: IN BIT;
8         enc_out, reenc_out: OUT BIT);
9 END encoder;
10
11 ARCHITECTURE bvh OF encoder IS
12
13   --FILE berl_mult5.vhd
14   COMPONENT berl_mult5
15     PORT(din, clk, res, sel5: IN BIT;
16           alpha0, alpha1, alpha2, alpha3, alpha4: OUT BIT);
17   END COMPONENT;
18
19   --FILE shiftreg5.vhd
20   COMPONENT shiftreg5
21     PORT(cclk, res, din: IN BIT;
22           reg1_out, reg2_out: OUT BIT);
23   END COMPONENT;
24
25   FOR ALL: berl_mult5 USE ENTITY WORK.berl_mult5(bhv);
26   FOR ALL: shiftreg5 USE ENTITY WORK.shiftreg5(bhv);
27
28   SIGNAL berl_in, reg1_out: BIT;
29   SIGNAL reg2_out, reg2_out2: BIT;
30   SIGNAL alpha0, alpha1, alpha2, alpha3, alpha4: BIT;
31   SIGNAL alpha4, alpha26, alpha17, alpha23: BIT;
32   SIGNAL parity, temp1, temp2: BIT;
33
34 BEGIN
35
36   -- Berlekamp Multiplier, multiplication @I symbol
37   a5: berl_mult5 PORT MAP(berl_in, clock, reset, sel5,
38                           alpha0, alpha1, alpha2, alpha3, alpha4);
39
40   -- Take alpha0 from multiplier into 1 bit shift register
41   a5: shiftreg5 PORT MAP(clock, res, alpha0, reg1_out);
42
43   -- Take alpha4 and reg1_out into 5 bits shift register
44   a6: shiftreg5 PORT MAP(clock, res, temp1, reg2_out);
45
46   -- Take alpha1 and reg2_out into 5 bits shift register
47   a7: shiftreg5 PORT MAP(clock, res, temp2, reg2_out);
48
49   alpha6 <= alpha5 XOR alpha1;
50   alpha26 <= alpha7 XOR alpha2;
51   alpha17 <= alpha7 XOR alpha1 XOR alpha0;
52   alpha23 <= alpha7 XOR alpha3 XOR alpha0;
53

```

Gambar 4. Desain VHDL untuk RS(31, 27) enkoder

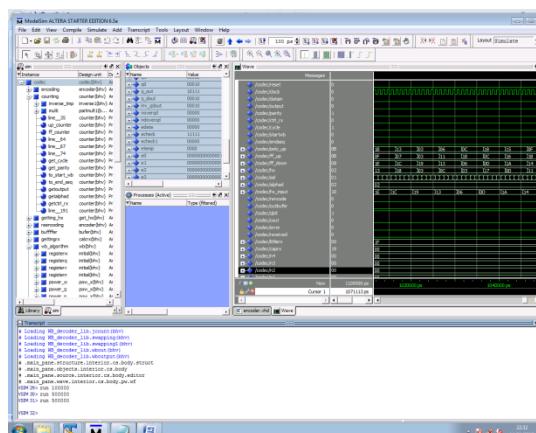
```

1 -- FILE wb.vhd
2 -- Perform WELCH BERLEKAMP ALGORITHM
3
4 ENTITY wb IS
5   PORT (res, clk2, start, endseq, parity: IN BIT;
6         R, alpha, ringcount: IN BIT_VECTOR(4 DOWNTO 0);
7         H4, H3, H2, H1, H0: OUT BIT_VECTOR(4 DOWNTO 0);
8         Q4, Q3, Q2, Q1, Q0: OUT BIT_VECTOR(4 DOWNTO 0));
9 END wb;
10
11 ARCHITECTURE bvh OF wb IS
12
13   -- FILE swapping.vhd
14   COMPONENT swapping
15     PORT (Q0in, Q1in, Q2in, Q3in, Q4in: IN BIT_VECTOR(4 DOWNTO 0);
16           Q0in, Q1in, Q2in, Q3in, Q4in: OUT BIT_VECTOR(4 DOWNTO 0));
17   END COMPONENT;
18
19   -- FILE initial.vhd
20   COMPONENT initial
21     PORT (res, clk, sell, start, parity: IN BIT;
22           sel1: IN BIT_VECTOR(4 DOWNTO 0);
23           swapout: OUT BIT);
24   END COMPONENT;
25
26   -- FILE rwg.vhd
27   COMPONENT rwg
28     PORT (D: IN BIT;
29           D0ver51, Rn: IN BIT_VECTOR(4 DOWNTO 0);
30           Q0in, Q1in, Q2in, Q3in, Q4in: IN BIT_VECTOR(4 DOWNTO 0));
31     END COMPONENT;
32
33   -- FILE genid.vhd
34   COMPONENT genid
35     PORT(R, Q, N, V: IN BIT_VECTOR(4 DOWNTO 0);
36           idixer0: OUT BIT);
37     END COMPONENT;
38
39   -- FILE initial.vhd
40   COMPONENT initial
41     PORT (res, clk, sell, start, parity: IN BIT;
42           set0, set1, set2, set3, set4: IN BIT_VECTOR(4 DOWNTO 0));
43     END COMPONENT;
44
45   -- FILE var.vhd
46   COMPONENT var
47     PORT (D: IN BIT;
48           D0ver51, Rn: IN BIT_VECTOR(4 DOWNTO 0));
49     END COMPONENT;
50
51   -- FILE rwg.vhd
52   COMPONENT rwg
53     PORT (D: IN BIT;
54           D0ver51, Rn: IN BIT_VECTOR(4 DOWNTO 0));
55     END COMPONENT;
56

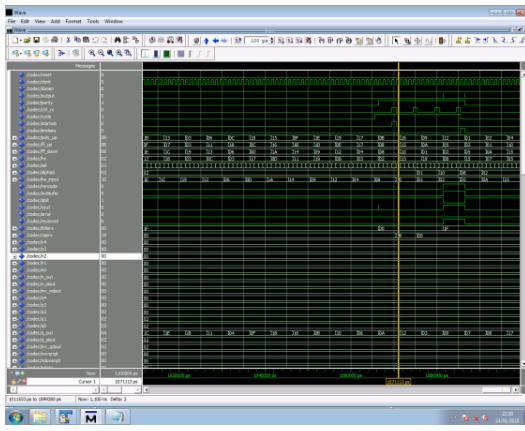
```

Gambar 5. Desain VHDL untuk Algoritma Welch Berlekamp

Aplikasi codec RS(31, 27) disimulasikan dengan menggunakan aplikasi ModelSim Altera 6.5e. Hal ini ditunjukkan pada gambar 6. Hasil simulasinya ditunjukkan oleh gambar 7, dan hasil simulasinya menunjukkan bahwa desain VHDL untuk RS (31,27) telah berjalan sesuai dengan harapan.



Gambar 6. Aplikasi ModelSim Altera 6.5e.



Gambar 7. Hasil simulasi RS(31, 27).

KESIMPULAN

Kode Reed Solomon (31, 27) sudah dirancang dengan menggunakan algoritma Welch Berlekamp. Kode Reed Solomon (31, 27) ditulis dengan menggunakan bahasa VHDL dan disimulasikan dengan menggunakan aplikasi ModelSim Altera 6.5e. Hasil simulasi menunjukkan bahwa rancangan VHDL untuk Reed Solomon (31, 27) telah berjalan sesuai dengan harapan.

REFERENCES

- [1] S. Wicker and V. Bhargava "Reed-Solomon Codes and Their Applications". *IEEE Press*, 1994.
- [2] R. Blahut, "Theory and Practice of Error Control Codes", Reading MA, Addison-Wesley, 1983.
- [3] H. Chang and C.B. Shung. "A (208, 192;8) Reed-Solomon Decoder for DVD Application", *IEEE International Conf on Communications*, vol 2, pp 957-960, 1998
- [4] S. Whitaker, J. Canaries, and K.Cameron. "Reed Solomon VLSI Codec for Advanced television". *IEEE Trans on Circuits and Systems for Video technology*, pp 230-236, 1991.
- [5] I.E.G. Richardson, "H.264 and MPEG-4 Video Compression Video Coding for Next generation Multimedia", Wiley Publisher, ISBN 0-470-84837-5, 2003.
- [6] E.R. Berlekamp, "Bit Serial Reed-Solomon Encoders", *IEEE Trans. Inform. Theory*, vol. 28, pp. 869-874, 1982.
- [7] K.Y. Liu, "Architecture for VLSI Design of a Reed-Solomon Encoder", *IEEE Trans. On Comp*, 1982.
- [8] A. Mahmudi, S. Achmadi, "Reed Solomon Code untuk Koreksi Kesalahan di Era Komputer", The 5th EECCIS 2010, Brawijaya University, Malang, pp. E-13.
- [9] Mahmudi, A. "Using Welch Berlekamp Algorithm to Implement Generalised Minimum Distance Decoding for Reed Solomon Codes", a report to support the transfer from Master of Philosophy to Doctor of Philosophy.

