Double Bits Error Correction for Computational Grid with CRC

Nima Jafari Navimipour, Seyed Hasan Es-hagi

***Abstract*— In grid system error during sending information due to devastating factors like external electromagnetic sources and noise is inevitable. The Cyclic Redundancy Check (CRC) method is used for error detection. CRC is used to control such factors in received information. In this paper, the new method based on CRC has been introduced that is able to detect the exact place of double bits error and correct them. In this method the receiver divide the received data on polynomial generator, g(x), and then get the remainder. Receiver can correct double bits error by comparing the remainder and the content of the look-up table. The result show that presented method can reduce the traffic for application that have low bit error rate (BER).**

***Index Terms*—CRC; Error Correction; Computational Grid**

I. Introduction

The popularity of the Internet and the availability of powerful computers and high-speed networks as low-cost commodity components are changing the way we use computers today [1]. These technical opportunities have led to the possibility of using geographically distributed and multi-owner resources to solve large-scale problems in science, engineering, and commerce [1].Recent research on these topics has led to the emergence of a new paradigm known as grid computing [2].These powerful paradigm has been used in various sciences such as spaceship process imaging and medical science [3], [4]. To achieve the promising potentials of tremendous distributed resources, effective and efficient error detection and correction are fundamentally important.

The accuracy of the transferred data is vital for grid system. There are devastating factors like external electromagnetic sources, bandwidth limit and noise that cause error in transferred information. For increasing the quality of service, controlling this problem is one of the data link layer duties in grid. A cyclic redundancy check (CRC) is a type of function that takes as input a data stream of any length, and produces as output a value of a certain space, commonly a 32-bit integer. A CRC can be used as a checksum to detect accidental alteration of data during transmission or storage. The CRC was invented by W. Wesley Peterson, and published in his 1961 paper [5]. The IEEE-recommended 32-bit CRC used in Ethernet and elsewhere, appeared at a telecommunications conference in 1975 [6]. According to ability of CRC method of errors detection, this method is strongest control method that also is used in data link protocols, PPP[1], HDLC[2], and Ethernet and in TCP/IP[3] stack protocols like IP4[4], UDP[5], and TCP. This method can only detect the errors in a way that receiver in case of error request for resending from sender considering copy of transferred information in his buffer, will send the faulty information again. But in most applications one or two bits of error correction will be better than burst and multiple errors detection, because it can prevent resending information and also has significant role on reducing traffic. Sunil Shukla and his friends in [7] have shown that by using CRC method we can correct the single-bit error.

In this paper, we show how to produce control bits in sender and reaction of receiver for received bits, then will review Sunil Shukla method that uses CRC to correct single-bit error. In the next section, details of using CRC to correct double bits error and produced look-up table to error correction will be represented, and finally we will conclude these methods.

II. CRC Calculation Methods

CRC is one of the most famous and strongest error control methods. A CRC is an error-detecting code. Its computation resembles a long division operation in which the quotient is discarded and the remainder becomes the result, with the important distinction that the arithmetic used is the carry-less arithmetic of a finite field. The length of the remainder is always less than or equal to the length of the divisor, which therefore determines how long the result can be.

Let g(x) be the CRC generator polynomial (the selection of generator polynomial is the most important part of implementing the CRC algorithm. The polynomial must be chosen to maximize the error detecting capabilities while minimizing overall collision probabilities) of degree n-k where n is the codeword length and k is the message length. Let v(x), e(x), and r(x) represent the polynomials associated with the n bit codeword where v(x) is the sending codeword polynomial, e(x) is the error codeword polynomial, and r(x) is the receiving codeword polynomial. These three polynomials are related by r(x) = v(x) + e(x). The remainder resulting from the division of $x^{n-k}v(x)$ by g(x), denoted as $R_{g(x)}\{x^{n-k}v(x)\}$, is equal to zero or a constant sequence [8]. Therefore the remainder resulting at the

Nima Jafari Navimipour is with Islamic Azad University, Tabriz Branch, Tabriz, Iran, phone: +989144021694; fax: +984115552322; e-mail: Jafari@iaut.ac.ir and Jafari_n@ymail.com

Seyes Hasan Es-hagi is member of Young Researchers Club, Tabriz Branch, Islamic Azad University, Tabriz Branch, Tabriz, Iran phone: +989143171845; fax: +984114201024; e-mail: es-hagi@iaut.ac.ir and es_hagi@gmail.com

[1] Point-to-Point Protocol
[2] High-Level Data Link Control
[3] Transmission Control Protocol/Internet Protocol
[4] Internet Protocol version 4
[5] User Datagram Protocol

receiver from the division of $x^{n-k}r(x)$ by g(x) is
$$R_{g(x)}\{x^{n-k}r(x)\} = R_{g(x)}\{x^{n-k}v(x) + x^{n-k}e(x)\} = R_{g(x)}\{x^{n-k}e(x)\} \quad (1)$$

If single-bit error occurs during codeword transmission, e(x) is equal to x^i, where *i* is a number between 0 and *n*−1, indicating the position of the corrupted bit. Equation (1) is then
$$R_{g(x)}\{x^{n-k}r(x)\} = R_{g(x)}\{x^{n-k}e(x)\} = R_{g(x)}\{x^{n-k}x^i\} \quad (2).$$

Possible patterns of $R_{g(x)}\{x^{n-k}x^i\}$ and the corresponding position of the error bit are stored in a look-up table in advance when correcting single-bit errors. Upon receiving a codeword *r*(*x*) containing a corrupted bit, the bit position is easily determined by comparing the division remainder of $R_{g(x)}\{x^{n-k}v(x)\}$ with the contents of the look-up table. The correction is easily implemented by inverting the bit of the receiving codeword at that position. Regarding to the properties of CRC generator polynomial can select specific CRC that can correct as follow:

All single-bit errors.

All double bits error, if G(x) had at least three sentences.

All odd errors, if G(x) can be divided by x+1.

Many of burst error that its length be less than G(x) length.

To have the above properties the primitive generator polynomial should be used to produce CRC. There are several standard and common CRC that is described in [9]. CRC can be implemented in hardware by three techniques serial, parallel and look-up tables. Method of look-up table includes saved CRC for all input moods. For example, for given four bits there is a need for saved $2^4 = 16$ quantity in look-up table. Serial implementation uses Liner Feedback Shift Registers (LFSR) that division is done by shifting to left and subtraction by XOR [10].

One example can clarify this technique. To compute an n-bit binary CRC, line the bits are representing the input in a row, and position the (n+1)-bit patterns representing the CRC's divisor underneath the left-hand end of the row [11]. Here is the first calculation for computing a 3-bit CRC:

```
11010011101100  ß input
          1011  ß divisor (4 Bits)
----------------------
01100011101100  ß result
```

If the input bit above the leftmost divisor bit is zero, do nothing and move the divisor to the right by one bit. If the input bit above the leftmost divisor bit is 1, the divisor is exclusive-OR into the input. The divisor is then shifted one bit to the right, and the process is repeated until the divisor reaches the right-hand end of the input row. Here is the last calculation:

```
00000000001110  ß result of multiplication calculation
          1011  ß divisor
---------------------
00000000000101  ß remainder (3 bits)
```

Since the leftmost divisor bit zeroed every input bit it touched, when this process ends the only bits in the input row that can be nonzero are the n bits at the right-hand end of the row. These n bits are the remainder of the division step, and will also be the value of the CRC function.

III. Related Work

The common method for single-bit error detection based on a look-up table has been presented in [5] and [11]. In [5] the author uses one method for correction single bit error based on CRC-16 code and the length of data is measured 16. Method is that first all possible single bit error on the data of 16 bit that may happen have been measured and the remainder of divided by $G(X) = X^{15} + X^{12} + X^5 + 1$ canceled and stored in table (1). On the receiver when a data is received, first this data is divided by G(x), when the remainder is zero, there is no error and data is valid, but if reminder result of dividend on generator polynomial is not equal to zero, the reminder is compare with existing quantities in look-up table, in order to the error location to be found and corrected.

A CRC look-up table optimization method for single-bit error correction has been presented in [8]. The optimization method minimizes the address length of the pre-designed look-up table while satisfying certain restrictions. With this method the access time to table and extraction of figure to place of error is done quickly. In this method for applying hardware, associated memories have been used.

TABLE I. The CRC Patterns of Single Bit Error

Data Bit in Error	CRC Pattern	MSB 8 bits	LSB 8 bits
0	0001000000100001	16	33
1	0010000001000010	32	66
2	0100000010000100	64	132
3	1000000100001000	129	8
4	0001001000110001	18	49
5	0010010001100010	36	98
6	0100100011000100	72	196
7	1001000110001000	145	136
8	0011001100110001	51	46
9	0110011001100010	102	98
10	1100110011000100	204	196
11	1000100110101001	137	169
12	0000001101110011	3	115
13	0000011011100110	6	230
14	0000110111001100	13	204
15	0001101110011000	27	125

IV. Double bits error correction based on CRC

The given method in this paper uses CRC-16 for correcting double bits error that are likely to happen in 16-bit data. In this paper it is supposed that only two bit errors have happened. A look-up table is used in this method that the content of the table is consists of remains and the place of eroded bits. In this method, 16-bits data with 16–bit CRC have been used that altogether form 32 bits $V(x) = (x_1, x_2, ..., x_{16}, C_{17}, C_{18}, ..., C_{32})$. Now if double bits error in the sent continued bit happens in i, j situations, $e(x) = (x^i, x^j)$ that $1 \le i, j \le 32$. Since errors may happen in data or in control bits(CRC) filed , three condition will emerge, if $1 \le i, j \le 16$ the error occurred in data field , if $17 \le j \le 32, 1 \le i \le 16$ the one bit error occurred in data field and one bit error occurred in CRC field and finally if $17 \le i, j \le 32$ the both errors have occurred in CRC field. The number of variety of double bits error can calculate with

$$C(2,32) = \frac{32!}{2! \times 30!} = 496 \quad (3)$$

Regarding to equations (3) variety of double bits error is 496. In each of these eroded conditions, a unique remainder is obtained; only in 48 conditions reminder is not unique. 48 numbers is insignificant in comparison with 496 numbers. These 48 conditions have been shown in table (2).

For example when the remainder is (0000010000000000) it is possible that $e(x) = (x^{11}, x^{27})$ or $e(x) = (x^{16}, x^{23})$.

To implementation this method two separate tables are needed. Unique remainders and the position of corresponding eroded bits with the number of 400 are presented in appendix A and non-unique remainders and the position of corresponding eroded bits with the number of 48 are presented in table (2) that i_1, j_1 refer to locations of double bits error and i_2, j_2 refer to another error locations that have same reminder. These tables are calculated by MATLAB. In this method, the received information is divided by CRC-16, if the remainder opposes zero, in order to the position of eroded bits specified ,the remainder is compared with the CRC Patterns column in table (3); now, for not being table (3) checked again T1 should be zero. If reminder to be matched with any CRC Patterns column in the table (3),we can say hit occurred and we can easily extract the position of the errors and correcting operation with inverting the position of errors in received information .Then our correcting operation finished. If the reminder is not matched with any CRC Patterns column in table (3), it's necessary to refer table (2) and the remainder be compared with CRC Patterns column in the table (2) for founding error's position. Since in this table any special reminder specifies two double bits error, first one of the couple, which indicates the place of errors is extracted, and then corrects these bits in the received information, then division

TABLE II. Non-unique CRC patterns of bits error

Column	CRC Pattern	i_1, j_1	i_2, j_2
1	0000010000000000	11,27	16,32
2	0000110111001100	27,31	15,20
3	0100000000000000	20,31	15,27
4	0100100011000100	23,27	11,16
5	0100110011001100	9,20	4,16
6	0100110101001100	13,24	8,20
7	0100110101101101	12,28	17,24
8	0100110111000100	4,20	9,16
9	0101010101001100	17,28	12,24
10	0101011101011100	16,32	21,28
11	0101110111001100	8,24	13,20
12	0101111011110101	28,32	16,21
13	1000000000000000	16,27	11,23
14	1000000100001000	15,31	20,27
15	1000010001000000	7,23	12,19
16	1000101010100110	19,30	14,26
17	1000110000000000	12,23	7,19
18	1000110001000100	3,19	8,15
19	1000110011000000	8,19	3,15
20	1000111011000100	22,26	10,15
21	1010100010100110	10,26	15,22
22	1010110001000000	26,30	14,19
23	1100010010000100	19,23	7,12
24	1100010101101101	21,32	16,28
25	1100100011100100	18,22	6,11
26	1100110001000000	15,19	3,8
27	1100110010000110	14,18	2,7
28	1100110011100101	13,17	1,6
29	1100110111001100	16,20	4,9
30	1100111011010100	17,21	5,10
31	1101010011100101	24,28	12,17
32	1101110001000100	20,24	8,13
33	1101110011000101	6,17	1,13
34	1101110011100100	1,17	6,13
35	1101110011110101	10,21	5,17
36	1101111011100101	5,21	10,17
37	1101111110110111	13,29	18,25
38	1101111111110101	14,25	9,21
39	1110100010000110	6,22	11,18
40	1110101000100010	14,30	19,26
41	1110101010100110	15,26	10,22
42	1110110010000100	2,18	7,14
43	1110110010100110	11,22	6,18
44	1110110011000110	7,18	2,14
45	1110110111000100	21,25	9,14
46	1110111111110101	18,29	13,25
47	1111110010000110	25,29	13,18
48	1111111100010000	9,25	14,21

Function is done again, if the remainder is zero so the error has been specified properly and we finish the algorithm; otherwise the next couple refers to real place of error in information and should be corrected. The above method has been given in chart (1).

V. Result

An important reason for the popularity of CRCs for detecting the accidental alteration of data is their efficiency guarantee. Errors in both data transmission channels and magnetic storage media tend to be distributed non-randomly, making CRC properties more useful than alternative schemes such as multiple parity checks, but main drawback of CRC is the inability of it in the case of error correction. So in this paper we introduce the new method based on CRC code that is suitable for error correction in more applications. This method is based on look-up table, so have a high speed rather than other common error correction techniques because of advanced memory technology. The advantages of our method are simple circuit structure and high speed operation.

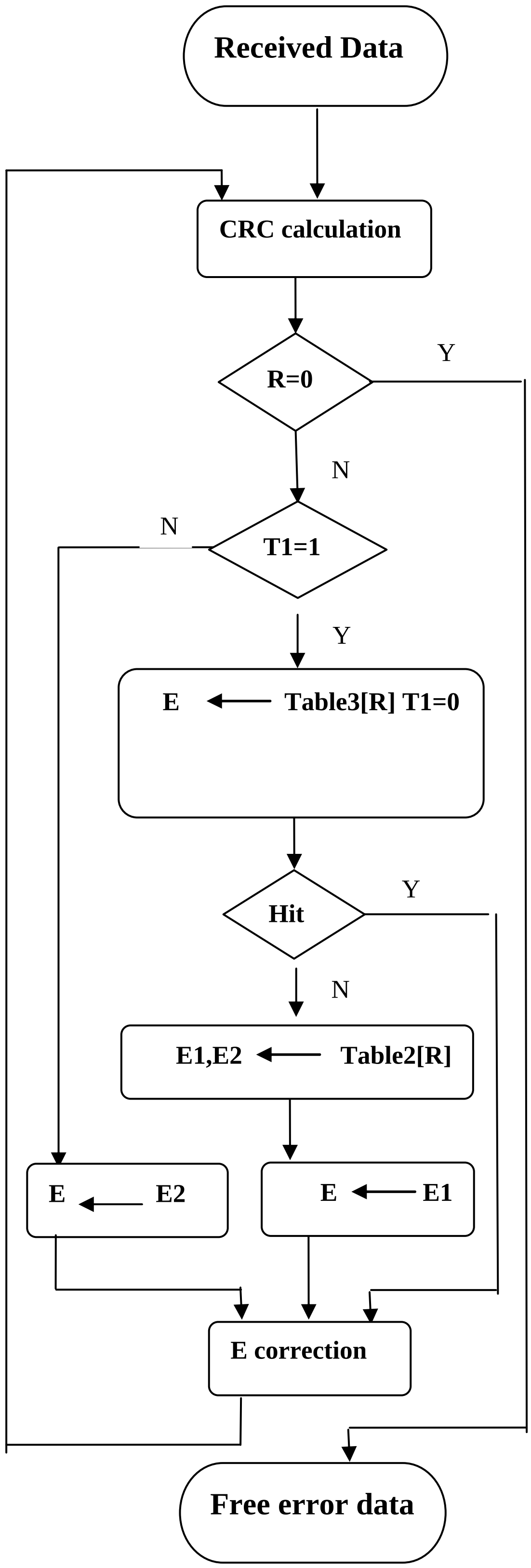

Chart 1: Double bits error correction Algorithm

CRC is the method that can detect the error in transferring data between two points in grid and can't correct the errors. But in grid error correction is important. In this paper a new method based on CRC has been introduced for double bits error correction. This method used look-up table for error correction. This method is very suitable for the applications that have low BER. In this case, the network traffic is reduced, because this method can reduce retransmission. For future work, this method can be developed for more that double bits error correction.

APPENDIX A

TABLE III. UNIQUE CRC PATTERNS OF DOUBLE BITS ERROR

Column	CRC Pattern	i_1, j_1
1	0000000000000001	27,01
2	0000000000000010	27,02
3	00000000000000100	27,03
4	0000000000001000	27,04
5	0000000000010000	27,05
6	00000000000100000	27,06
7	0000000001000000	27,07
8	00000000010000000	27,08
9	00000000100000000	27,09
10	0000001000000000	27,10
11	0000001101110011	29,27
12	0000010100001000	23,20
13	0000010101101101	28,15
14	0000010111101001	28,19
15	0000011011100110	30,27
16	0000100000000000	27,12
17	0000110001000000	19,16
18	0000110011000100	16,15
19	0000110101001000	20,19
20	0000110110101001	28,23
21	0001000000000000	27,13
22	0001000000100001	27,17
23	0001001000110001	27,21
24	0001010110001000	24,23
25	0001101110011000	32,27
26	0001110101001100	24,15
27	0001110111001000	24,19
28	0010000000000000	27,14
29	0010000001000010	27,18
30	0010001100001111	28,26
31	0010010001100010	27,22
32	0010101010100110	26,16
33	0010101110101110	26,20
34	0011001100110001	27,25
35	0011101100101110	26,24
36	0100000010000100	27,19
37	0100000100001000	31,16
38	0100000101101101	28,11
39	0100001110001011	30,28
40	0100010001101101	28,09
41	0100010011000100	16,12
42	0100010100101101	28,07
43	0100010101001101	28,06
44	0100010101100101	28,04
45	0100010101101001	28,03
46	0100010101101100	28,01
47	0100010101101111	28,02
48	0100010101111101	28,05
49	0100010111001100	20,12
50	0100010111101101	28,08
51	0100011000011110	29,28
52	0100011011010100	32,24
53	0100011101101101	28,10
54	0100100010100001	31,28
55	0100100111001100	20,11
56	0100101000100010	30,16

57	0100101100101010	30,20
58	100110001000100	16,08
59	0100110010000100	16,07
60	00100110011000000	16,03
61	0100110011000101	16,01
62	0100110011000110	16,02
63	0100110011010100	16,05
64	0100110011100100	16,06
65	0100110110001100	20,07
66	0100110111001000	20,03
67	0100110111001101	20,01
68	0100110111001110	20,02
69	0100110111011100	20,05
70	0100110111101100	20,06
71	0100111010111111	29,20
72	0100111011000100	16,10
73	0100111101111101	24,21
74	0100111110110111	29,16
75	0100111111001100	20,10
76	0101000010000000	31,24
77	0101010101101101	28,13
78	0101011001010100	32,20
79	0101100101001100	24,11
80	0101101110101010	30,24
81	0101110001001100	24,09
82	0101110011000100	16,13
83	0101110011100101	17,16
84	0101110100001100	24,07
85	0101110101000100	24,04
86	0101110101001000	24,03
87	0101110101001101	24,01
88	0101110101001110	24,02
89	0101110101011100	24,05
90	0101110101101100	24,06
91	0101110111101101	20,17
92	0101111000111111	29,24
93	0101111101001100	24,10
94	0101111111111101	21,20
95	0110000100001111	28,22
96	0110010100101111	28,18
97	0110010101101101	28,14
98	0110011001100010	27,26
99	0110100010100110	22,16
100	0110100110101110	22,20
101	0110110010000110	18,16
102	0110110011000100	16,14
103	0110110110001110	20,18
104	0110110111001100	20,14
105	0110111001111101	25,24
106	0111011001011100	28,25
107	0111100100101110	24,22
108	0111110100001110	24,18
109	0111110101001100	24,14
110	0111111011111101	25,20
111	0111111111110101	25,16
112	1000000110001100	31,19
113	1000001011100110	30,23
114	1000010000000001	23,01
115	1000010000000010	23,02
116	1000010000000100	23,03
117	1000010000001000	23,04
118	1000010000010000	23,05
119	1000010000100000	23,06
120	1000010010000000	23,08
121	1000010011000100	15,12
122	1000010100000000	23,09
123	1000011000000000	23,10
124	1000011101110011	29,23
125	1000100001000000	19,11
126	1000100011000100	15,11
127	1000100110101001	28,27
128	1000100111001100	31,23
129	1000101000100010	30,15
130	1000101011100100	26,18
131	1000110001000001	19,01
132	1000110001000010	19,02
133	1000110001001000	19,04
134	1000110001010000	19,05
135	1000110001100000	19,06
136	1000110010000100	15,07
137	1000110011000101	15,01
138	1000110011000110	15,02
139	1000110011001100	15,04
140	1000110011010100	15,05
141	1000110011100100	15,06
142	1000110101000000	19,09
143	1000110111000100	15,09
144	1000111001000000	19,10
145	1000111100110011	29,19
146	1000111110110111	29,15
147	1001000110001000	27,24
148	1001010000000000	23,13
149	1001010000100001	23,17
150	1001011000110001	23,21
151	1001011101011100	32,15
152	1001011111011000	32,19
153	1001100110010111	26,25
154	1001110001000000	19,13
155	1001110001100001	19,17
156	1001110011000100	15,13
157	1001110011100101	17,15
158	1001111001110001	21,19
159	1001111011110101	21,15
160	1001111110011000	32,23
161	1010000001100010	23,22
162	1010001010100110	26,12
163	1010010000000000	23,14
164	1010010001000010	23,18
165	1010011101101010	31,26
166	1010100000100010	22,19
167	1010100111010101	29,26
168	1010101000100110	26,08
169	1010101010000110	26,06
170	1010101010100010	26,03
171	1010101010100100	26,02
172	1010101010100111	26,01
173	1010101010101110	26,04
174	1010101010110110	26,05
175	1010101011100110	26,07
176	1010101110100110	26,09
177	1010110000000010	19,18
178	1010110010000110	18,15
179	1010110011000100	15,14
180	1010111010100110	26,11
181	1011000100111110	32,26
182	1011011100110001	25,23
183	1011100010010111	26,21
184	1011101010000111	26,17
185	1011101010100110	26,13
186	1011111101110001	25,19

187	1011111111110101	25,15
188	1100000000001000	31,09
189	1100000011000100	12,11
190	1100000100000000	31,04
191	1100000100001001	31,01
192	1100000100001010	31,02
193	1100000100001100	31,03
194	1100000100011000	31,05
195	1100000100101000	31,06
196	1100000101001000	31,07
197	1100000110001000	31,08
198	1100001000100010	30,12
199	1100001001111011	31,29
200	1100001100001000	31,10
201	1100010000000000	23,15
202	1100010001000100	12,08
203	1100010001100101	28,20
204	1100010011000000	12,03
205	1100010011000101	12,01
206	1100010011000110	12,02
207	1100010011001100	12,04
208	1100010011010100	12,05
209	1100010011100100	12,06
210	1100010100001000	31,11
211	1100010111000100	12,09
212	1100011011000100	12,10
213	1100011101011100	32,13
214	1100011101111101	32,17
215	1100011110110111	29,12
216	1100011111101110	31,30
217	1100100000100010	30,10
218	1100100001000100	11,08
219	1100100010000100	11,07
220	1100100010100110	22,14
221	1100100011000000	11,03
222	1100100011000101	11,01
223	1100100011000110	11,02
224	1100100011001100	11,04
225	1100100011010100	11,05
226	1100100100001000	31,12
227	1100100101010001	30,29
228	1100100111000100	11,09
229	1100101000000010	30,06
230	1100101000100000	30,02
231	1100101000100011	30,01
232	1100101000100110	30,03
233	1100101000101010	30,04
234	1100101000110010	30,05
235	1100101001100010	30,07
236	1100101010100010	30,08
237	1100101011000100	11,10
238	1100101100100010	30,09
239	1100101110110111	29,11
240	1100110000000100	08,07
241	1100110001000101	08,01
242	1100110001000110	08,02
243	1100110001001100	08,04
244	1100110001010100	08,05
245	1100110001100100	08,06
246	1100110010000000	07,03
247	1100110010000101	07,01
248	1100110010001100	07,04
249	1100110010010100	07,05
250	1100110010100100	07,06
251	1100110011000001	03,01

252	1100110011000010	03,02
253	1100110011000111	02,01
254	1100110011001000	04,03
255	1100110011001101	04,01
256	1100110011001110	04,02
257	1100110011010000	05,03
258	1100110011010101	05,01
259	1100110011010110	05,02
260	1100110011011100	05,04
261	1100110011100000	06,03
262	1100110011100110	06,02
263	1100110011101100	06,04
264	1100110011110100	06,05
265	1100110101000100	09,08
266	1100110110000100	09,07
267	1100110110110111	29,10
268	1100110111000000	09,03
269	1100110111000101	09,01
270	1100110111000110	09,02
271	1100110111010100	09,05
272	1100110111100100	09,06
273	1100111000100010	30,11
274	1100111001000100	10,08
275	1100111010000100	10,07
276	1100111010110111	29,09
277	1100111011000000	10,03
278	1100111011000101	10,01
279	1100111011000110	10,02
280	1100111011001100	10,04
281	1100111011100100	10,06
282	1100111011110101	21,13
283	1100111100110111	29,08
284	1100111110010111	29,06
285	1100111110100111	29,05
286	1100111110110011	29,03
287	1100111110110101	29,02
288	1100111110110110	29,01
289	1100111110111111	29,04
290	1100111111000100	10,09
291	1100111111110111	29,07
292	1101000100001000	31,13
293	1101000100101001	31,17
294	1101000110111010	32,30
295	1101001100111001	31,21
296	1101001101011100	32,11
297	1101010000101111	32,29
298	1101010011000100	13,12
299	1101010101011100	32,10
300	1101011001011100	32,09
301	1101011011110101	21,12
302	1101011100011100	32,07
303	1101011101001100	32,05
304	1101011101010100	32,04
305	1101011101011000	32,03
306	1101011101011101	32,01
307	1101011101011110	32,02
308	1101011101111100	32,06
309	1101011111011100	32,08
310	1101100000010011	30,21
311	1101100011000100	13,11
312	1101100011100101	17,11
313	1101101000000011	30,17
314	1101101000100010	30,13
315	1101101010010000	32,31
316	1101101011110101	21,11

317	1101101110010111	25,22
318	1101110001100101	17,08
319	1101110010000100	13,07
320	1101110010100101	17,07
321	1101110011000000	13,03
322	1101110011000110	13,02
323	1101110011001100	13,04
324	1101110011010100	13,05
325	1101110011100001	17,03
326	1101110011100111	17,02
327	1101110011101101	17,04
328	1101110101001100	24,16
329	1101110110000110	29,21
330	1101110111000100	13,09
331	1101110111100101	17,09
332	1101111001110101	21,08
333	1101111010110101	21,07
334	1101111011000100	13,10
335	1101111011010101	21,06
336	1101111011110001	21,03
337	1101111011110100	21,01
338	1101111011110111	21,02
339	1101111011111101	21,04
340	1101111101011100	32,12
341	1101111110010110	29,17
342	1110000010100110	22,12
343	1110000100001000	31,14
344	1110000101001010	31,18
345	1110001001100010	26,23
346	1110010001101101	32,25
347	1110010010000110	18,12
348	1110010011000100	14,12
349	1110010101101010	31,22
350	1110100000100110	22,08
351	1110100010100010	22,03
352	1110100010100100	22,02
353	1110100010100111	22,01
354	1110100010101110	22,04
355	1110100010110110	22,05
356	1110100011000100	14,11
357	1110100011100110	22,07
358	1110100110100110	22,09
359	1110101001100000	30,18
360	1110101111010101	29,22
361	1110110000000110	18,08
362	1110110001000100	14,08
363	1110110010000010	18,03
364	1110110010000111	18,01
365	1110110010001110	18,04
366	1110110010010110	18,05
367	1110110011000000	14,03
368	1110110011000101	14,01
369	1110110011001100	14,04
370	1110110011010100	14,05
371	1110110011100100	14,06
372	1110110110000110	18,09
373	1110111001000000	30,22
374	1110111010000110	18,10
375	1110111011000100	14,10
376	1110111110110111	29,14
377	1110111111010100	25,17
378	1111001000111001	31,25
379	1111001100111110	32,22
380	1111011100011110	32,18
381	1111011101011100	32,14
382	1111011111110101	25,12
383	1111100010000111	22,17
384	1111100010100110	22,13
385	1111100100010011	30,25
386	1111101010010111	22,21
387	1111101111110101	25,11
388	1111110010100111	18,17
389	1111110011000100	14,13
390	1111110011100101	17,14
391	1111110111110101	25,10
392	1111111010110111	21,18
393	1111111101110101	25,08
394	1111111110110101	25,07
395	1111111111010101	25,06
396	1111111111100101	25,05
397	1111111111110001	25,03
398	1111111111110100	25,01
399	1111111111110111	25,02
400	1111111111111101	25,04

REFERENCES

[1] Fangpeng Dong and Selim G. Akl, "Scheduling Algorithms for Grid Computing": State of the Art and Open Problems, School of Computing, Queen's University Kingston, Ontario January 2006

[2] R. Buyya and D. Abramson and J. Giddy and H. Stockinger, "Economic Models for Resource Management and Scheduling in Grid Computing", in J. of Concurrency and Computation: Practice and Experience, Volume 14, Issue.13-15, pp. 1507-1542,Wiley Press, December 2002.

[3] F. Berman, High-Performance Schedulers, chapter in The Grid: Blueprint for a Future Computing Infrastructure, edited by I. Foster and C. Kesselman, Morgan Kaufmann Publishers, 1998.

[4] Imtiaz Ahmad , Muhammad K. Dhodhi, "Short Communication Multiprocessor Scheduling in a Genetic Paradigm", Elsevier , pp 395-706, 1996

[5] Peterson, W. W. and Brown, D. T. "Cyclic Codes for Error Detection "Proceedings of the IRE 49: 228. doi: 10.1109/JRPROC.1961.287814, January 1961.

[6] Brayer, K; Hammond, J L Jr. "Evaluation of error detection polynomial performance on the AUTOVON channel" in National Telecommunications Conference, New Orleans, La. Conference Record 1: p. 8-21 to 8-25, New York: Institute of Electrical and Electronics Engineers, December 1975.

[7] Shukla S, Bergmann N W. "Single bit error correction implementation in CRC-16 on FPGA". In: IEEE International Conference on Field-Programmable Technology. Brisbane, Australia, 2004: 319-322.

[8] PAN Yun, GE Ning, DONG Zaiwang. "CRC Look-up Table Optimization for Single-Bit Error Correction". In: TSINGHUA SCIENCE AND TECHNOLOGY ISSN 1007-0214 18/19 pp620-623 Volume 12, Number 5, October 2007.

[9] T. V. Ramabadran and S. S. Gaitonde. "A tutorial on CRC computations". IEEE Micro, Vol. 8, No. 4, 1988, pp. 62-75.

[10] W.W Peterson, E. J. Weldon. "Cyclic Codes for Error Detection". Second Edition Published by MIT Press, 1972 ISBN 0262160390, 9780262160391.

[11] Johnston C A, Chao H J. "The ATM layer chip: An ASIC for B-ISDN applications". IEEE Journal on Selected Areas in Communications, 1991, 9(5): 741-750.

Nima Jafari Navimipour received his B.S. in computer engineering, software engineering, from Islamic Azad University, Tabriz Branch, Tabriz, Iran, in 2007, the M.S. in computer engineering, computer architecture, from Islamic Azad University, Tabriz Branch, Tabriz, Iran, in 2009. From 2007, he worked as a researcher with the Young Research Club, Tabriz Branch, Islamic Azad University, Tabriz Branch.

He is the author/co-author of more than 15 publications in technical journals and conferences. His research interests include Traffic Control, Traffic Modeling, Computational Intelligence, Evolutionary Computing, Computational Grid, and Wireless Networks.

Seyed Hasan Es-hagi received his B.S. in computer engineering, hardware engineering, from Shomal University, Amol, Iran, in 2007, the M.S. in computer engineering, computer architecture, from Islamic Azad University, Tabriz Branch, Tabriz, Iran, in 2009. From 2008, he worked as a researcher with the Young Research Club, Tabriz Branch, Islamic Azad University, Tabriz Branch.

He is the author/co-author of more than 10 publications in technical journals and conferences. His area of research is in Digital Communications, Error Correction Codes, and Wireless Networks