THE INTERNATIONAL JOURNAL OF SCIENCE & TECHNOLEDGE

Analysis of Encoding Large Integers: Using Method-II on A123367

Afolabi Godfrey
Teacher, Mathematics Department, Joda International School, Nigeria
Zaid Ibrahim
Derector Personnel, Department of Mathematics, Sokoto State University, Nigeria
Muhammad. S. Magami
Lecturer, Department of Mathematics, Usmanu Danfodiyo University, Nigeria
Saidu Yakubu
Lecturer, School of General Studies, College of Nursing and Midwifery, Sokoto, Nigeria
Ahmad Rufai
Lecturer, Department of Mathematics, Sokoto State University, Nigeria

Abstract:
In most computational analysis, large integers are obtained as results which are used, stored or transmitted across channels from a source to a specified destination. The efficiency and reliability of any result when ever in use is paramount to any analyst and this can be guaranteed by encoding such results. In this work therefore, an analysis of encoding large integer using Method-II was carried out on a selected integer from the sequence A123367. Consequently, the results obtained as presented in this paper shall be useful generally in coding theory.

Keywords*: A123367, encode, integer, method-ii and parity code*

1. Introduction

The possibility of obtaining large integers as results from computational analysis cannot be over emphasized, *Afolabi and Ibrahim, (2013)*. Therefore, encoding such results obtained for efficiency and reliability whenever in use becomes a matter of interest to every player in the field of Mathematical Sciences. It is against this background therefore that the analysis of Method-II was carried out on a selected integer from integer sequence A123367 and presented in this paper. This integer sequence A123367 is the difference between the number of rows in the truth table circuit designs involving n variables and divided by order eight of the symmetric group on n symbols, *Ibrahim, (2007)*. It is generated by the formula: $(n! - 2^n) / 8$, with n = 4, 5,…, 23, *Ibrahim (2007)*. The results obtained from the analysis of this work shall form the general basis for encoding large binary data in Coding Theory.

2. Definition of Basic Terms used

The following basic terms as used in this paper are defined to make the work self contained:

- A123367: This is a special integer sequence generated by the formula $(n! - 2^n) / 8$, with n = 4, 5,…, 23, Ibrahim, (2007).
- Encode: This process involves the addition of parity (correction) bit to the result (information) obtained (computed) meant to used, sent, or stored as the case may be. This is to enable the detection of any error(s) when ever they occur. Thus, encoding a bit sequence adds redundant information to aid the intended receiver in correcting symbol error(s). For example, to encode the given (result) data: 10010011**,** a parity code **1100,** calculated would be placed in positions 2^n, n = 0, 1, 2, … that is, positions 1, 2, 4 and 8 respectively which are parity code positions. Thus, the encoded data would be **11**1**0**001**0**0011, Ziegler, (2000).
- Integer: This is a collection of all the numbers that can be represented on a number line. They are: Negative, Neutral and Positive numbers respectively. For example, …,-3, -2, -1, 0, 1, 2, 3…
- Method-II: The analysis of Method-II involves stages I-IV.
- Parity code: Parity is the simplest and oldest error detection method. A binary digit called parity is used to indicate whether the number of bits with '1' in a given set of binary data (result obtained) is/are even or odd, it is set to a '**0**' if even and a '**1**' if odd and usually used to detect transmission or computation error(s). These parity bits set forms the parity code and then placed into their respective positions in the original data (result obtained) which allows for the restoration of an erroneous bit when its position is detected, Bhattacharrya and Nandi, (1997).

3. Method / Procedure

The following four stages are the procedures of the analysis of Method-II carried out in this work:

- Stage I: Generate the integer (result) from the integer sequence by using the formula $(n! - 2^n) / 8$, with n = 23

- Stage II: Obtain the binary equivalence of the integer (result) generated.
- Stage III: Calculate the parity code.
- Stage IV: Obtain the encoded data.

4. Data Presentation and Computation on A123367

The integer (result) generated from the integer sequence A123367, using (n! – 2^n) / 8, when n = 23, its binary equivalence, calculated parity code and the encoded data obtained are shown in TABLE 1 below:

(n! – 2^n) / 8, n = 23	323150209236062103142 4.
Binary Equivalence	1000011111100011111011100111111000100000001000010100000001100100011100000.
Parity Code	**1 0 0 0 0 1 0.**
Encoded Data	**1 0** 1 **0** 0 0 0 **0** 0 1 1 1 1 1 1 **0** 0 0 0 1 1 1 1 1 0 1 1 1 0 0 1 **1** 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 **0** 1 1 0 0 1 0 0 0 1 1 1 0 0 0 0 0.

Table 1: Results obtained from computation on A123367.
Source: Researcher's Computation

5. Analysis of Method-II on A123367

The outcomes of each stage I-IV of Method-II as shown in TABLE 4.0.1 above were obtained as follows:

- Stage I: Generate the integer (result) from the integer sequence A123367 using the formula (n! – 2^n) / 8, with n = 23 as used in this work. Thus we have: 323150209236062103142 4, *Ibrahim, (2007).*
- Stage II: Obtain the binary equivalence of the integer (result) in stage I above. This is shown in TABLE 2 below:

2	323150209236062103142 4	R
2	1665751046180310515712	0
2	832875523040155252856	0
2	426437761520077626428	0
2	213218880760038813214	0
2	106609440380019406607	0
2	53304720190009703303	1
2	26652360095004851601	1
2	13326180047502425800	1
2	6613090023751212900	0
2	3306545011875606450	0
2	1653272505937203225	0
2	826636252968601612	1
2	423313126484300806	0
2	211656563242150403	0
2	105828281621075201	1
2	52914140810537600	1
2	26457070405268800	0
2	13228535202634400	0
2	6614267101317200	0
2	3307133550658600	0
2	1653566775329300	0
2	826783387664650	0
2	423391693832325	0
2	211695846916162	1
2	105847923458081	0
2	52923961729040	1
2	26461980864520	0
2	13230990432260	0
2	6610495216130	0
2	3305247608065	0
2	1657623804032	1
2	828811902016	0
2	414405951008	0
2	207202975504	0
2	103601487752	0
2	51800743876	0
2	25900371938	0

2	12950185969	0
2	6475092984	1
2	3237546492	0
2	1618773246	0
2	804386623	0
2	402193311	1
2	201096655	1
2	100548327	1
2	50274163	1
2	25137081	1
2	12568540	1
2	6284270	0
2	3142135	0
2	1571067	1
2	785533	1
2	392766	1
2	196383	0
2	98191	1
2	49095	1
2	24547	1
2	12273	1
2	6136	1
2	3068	0
2	1534	0
2	767	0
2	383	1
2	191	1
2	95	1
2	47	1
2	23	1
2	16	1
2	8	0
2	4	0
2	2	0
2	1	0
	0	1

Table 2: Obtaining binary equivalence of the integer (result)
Source: Researcher's Computation

By taking the values on the remainder (R) column from bottom to top, we have:
10000111111000111110111001111110001000000010000101000000011001000111000000.

- Stage III: Calculating the parity code

According to *Afolabi, Ibrahim and Zaid, (2014)*, the following steps are involved in the calculation of parity code using method-II:

1. Arrange in a table the given data in their respective positions, leaving space for parity bits, that is, powers of 2 from 0, 1, 2, … so, we have positions 1, 2, 4, 8, 16, 32 and 64.
2. Identify the data positions having '1', obtain their respective binary equivalences and set parity bit to '**1**' if odd number of 1's and to '**0**' if even numbers of 1's along their respective columns.
3. The parity bit set are reversed to form the parity code and placed back into their respective spaces left in the initial table.

Thus, the parity code is obtained. An encoded data is formed when the parity codes are placed back into their respective positions. We have as shown below in TABLES 3 & 4 respectively:

BP	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
DP	**P1**	**P2**	1	**P4**	0	0	0	**P8**	0	1	1	1	1	1	1	**P16**	0	0	0	1	1	1	1	1	0	1	1	1	0	0	1	**P32**	1	1	1	1	1	0	0

Table 3: Showing Bit Positions (BP) and Data Positions (DP) 1-39 respectively
Source: Researcher's Computations

BP	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
DP	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	**P64**	1	1	0	0	1	0	0	0	1	1	1	0	0	0	0	0

Table 4: Showing Bit Positions (BP) and Data Positions (DP) 40-80 respectively.
Source: Researcher's Computation

The data positions (**DP**) having 1, their respective binary equivalence and parity bit set (**PBS**) along their respective columns are shown in TABLE 5 below:

DP	BINARY EQUIVALENCE
3	0 0 0 0 0 1 1
10	0 0 0 1 0 1 0
11	0 0 0 1 0 1 1
12	0 0 0 1 1 0 0
13	0 0 0 1 1 0 1
14	0 0 0 1 1 1 0
15	0 0 0 1 1 1 1
20	0 0 1 0 1 0 0
21	0 0 1 0 1 0 1
22	0 0 1 0 1 1 0
23	0 0 1 0 1 1 1
24	0 0 1 1 0 0 0
26	0 0 1 1 0 1 0
27	0 0 1 1 0 1 1
28	0 0 1 1 1 0 0
31	0 0 1 1 1 1 1
33	0 1 0 0 0 0 1
34	0 1 0 0 0 1 0
35	0 1 0 0 0 1 1
36	0 1 0 0 1 0 0
37	0 1 0 0 1 0 1
41	0 1 0 1 0 0 1
49	0 1 1 0 0 0 1
54	0 1 1 0 1 1 0
56	0 1 1 1 0 0 0
65	1 0 0 0 0 0 1
66	1 0 0 0 0 1 0
69	1 0 0 0 1 0 1
73	1 0 0 1 0 0 1
74	1 0 0 1 0 1 0
75	1 0 0 1 0 1 1
PBS	**0 1 0 0 0 0 1**

*Table 5: Showing **DP** having 1, their respective binary equivalence and **PBS***
Source: Researcher's Computation

The parity bit set (**PBS**) is then reversed to form the parity code: **1 0 0 0 0 1 0,** and placed back into their respective positions to form the encoded data. Therefore, the encoded data will be: **1 0** 1 **0** 0 0 0 **0** 0 1 1 1 1 1 1 **0** 0 0 0 1 1 1 1 1 0 1 1 1 0 0 1 **1** 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 **0** 1 1 0 0 1 0 0 0 1 1 1 0 0 0 0 0.

The efficiency and reliability of this encoded data when ever in use, transmitted or stored as the case maybe is guaranteed. Thus, the comparison between the parity codes of the computed result or transmitted data with that of the received data in case of transmission will indicate whether an error has occurred or not. If the parity codes are found to be the same, that is result all 0's then, no error has occurred but if otherwise, that is any difference, then an error has occurred and practical steps are therefore taken to indentify the erroneous bit and flip it (that is interchange the bit from '**0**' to '**1**' or from '**1**' to '**0**' as the case may be) to correct it.

6.Conclusion

Generally, the results obtained from computational analysis are either for the present or future use. Whatever is the case, the efficiency and reliability of such results when ever in use is of great importance if communication must be effective. This can only be guaranteed by encoding the results in order to ascertain any occurrence of error(s). Although the analysis of Method-II presented in this work was carried out on a selected integer from the sequence A123367, the results obtained however, has a general application on any large integer involving binary numbers in Coding Theory.

7. References

1. Afolabi, G. & Ibrahim, A. A. The use of Algorithmic Method of Hamming Code Techniques for the Detection and Correction of Computational Errors in a Binary Coded Data: Analysis on an Integer Sequence A119626; IOSR Journals of Mathematics (IOSR- JM) e – ISSN: 2278, p – ISSN: 2319 – 7676. Volume 9, Issue 2 (Nov. – Dec. 2013) pp 33 -37. www.iosrjournals.org.
2. Afolabi, G, Ibrahim, A.A & Zaid, I, The use of Computational Method of Hamming Code Techniques for the Detection and Correction of Computational Errors in a Binary Coded Data: Analysis on an Integer Sequence A119626International Journal of Computational Engineering Research (IJCER). ISSN: 2250 – 3005. Volume 04, Issue 1 (January 2014) 6 -15. www.ijcer.org.
3. Bhattacharryya, D.K and Nandi S, (1997) An efficient class of SEC-DED-AUED codes: International symposium on parallel Architectures, Algorithms and Networks (ISPAN). 1, 410-415.
4. Ibrahim, A.A, Academic Journal Inc., An Enumeration scheme and Algebraic properties of A special (123) - avoiding class of permutations pattern; Trends in applied sciences Research. 2(4) (2007), 334-340.
5. Ziegler, J.F, An M.Sc Thesis submitted to the faculty of Information Technology and Engineering of George Mason University. Automatic Recognition and Classification of Forward Error Correcting Code. (Spring 2000, George Mason University Fairfax, Virginia).