Applied Information System and Management (AISM)
Volume X, (X) 2XXX, p. XX-XX
P-ISSN: 2621-2536; E-ISSN: 2621-2544; DOI: 10.15408/aism.vxix. xxxxx

Dotriacontal Number System in Computer and Error Detection

Md. Jahurul Islam[1], M. Mesbahuddin Sarker[2], Taslim Taher[3]

***Abstract*— In this paper for the first time, we have formulated the dotriacontal number system using existing number systems like binary, and hexadecimal numbers. The dotriacontal number is one with a base of 32, containing 32 single-character digits or symbols. Each symbol contains five binary digits. This number system can be used in computers for reducing memory consumption and for memory specification. It also can be used to increase the number of MAC addresses, IPv4 and IPv6 addresses, and to detect error message using the checksum method.**

***Index Terms*—Decimal, binary, hexadecimal, dotriacontal number, MAC, and EUI.**

I. Introduction

A number system is a system of writing to express numbers. The modern binary number, the basis for binary code was invented by Gottfried Wilhelm Leibniz in 1689 which was published in 1703 [18]. The hexadecimal number with a base of 16 was invented by John Nystrom which was published in 1863 [17]. Since computers use binary numbers 0 and 1 to process the data and to keep the circuitry less, when any letter or symbol or word is typed the computer translates them into numbers and then convert binary 0 and 1. Also the hexadecimal number is used in computers to define memory location, MAC (Media Access Control) addresses, error message display, and color on web pages [20]. Due to the process of hues data and transmission of audio and video, the demand for high performance, computer memory specification and internetworking protocol version (IP) have increased. It is a great challenge to the researchers to restrict the memory consumption and overcome the deficiencies of MAC (Media Access Control) address, and IP versions. These problems can be reduced by storing data in a short coded system. In this paper, we proposed that the dotriacontal number system can be used in computers as a short coded system because each digit of this system contains five binary bits. This can be used to reduce memory consumption and for memory specification and also to increase the number of MAC addresses, IPv4 and IPv6 addresses, and to detect error messages using the checksum method.

Remaining sections are built like this: In Section 2, a writing study on work is presented involving illustrations for the most important terms applied in this thesis-basic concept and symbols of existing number systems such as deciaml, binary octal, and hexadecimal number system. In Section 3, we describe dotriacontal number system, we show the relation between different number systems, also we present the conversions from one number system to dotriacontal number reversely dotriacontal to another. In Section 4, the MAC, IPv4, and IPv6 addresses are explained in dotriacontal number system. In Section 5, we discuss the applications of dotriacontal number system in computer memory addressing and also the methods of error detection in transmitting message, use the checksum method in dotriacontal number system. In Section 6, we show the comparison between the hexadecimal and dotriacontal number system and also present the comparison between IPv4 address 32 and IPv4 address 40 bits.

II. Related Work

Fiete, I. R., Seung, H. S. [1] enforced unary numerals used in the neural circuits responsible for birdsong production. Karl Menninger [2] implemented the development of numbers both as spoken and as symbolic abstract numerals that can be readily manipulated and combined. Sheraz Sheraz A. K., M. Moosa et. al. [3] used protocols and mechanisms to recover failed packets in wireless networks. Whenever bits flow from one device to another devices, they can be corrupted or changed due to interference and network problems. The corrupted bits leads to

This paragraph of the first footnote will contain the date on which you submitted your paper for review, revised, and accepted.

[1]Md. Jahurul Islam, Department of Computer Science, Govt. Mollartek Udayan School and College, Dhaka, Bangladesh (e-mail: jahurul93@gmail.com).

[2]M. Mesbahuddin Sarker, Institute of Information System, Jahangirnagar University, Dhaka, Bangladesh (e-mail: sarker@juniv.edu).

[3]Taslim Taher, Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh (e-mail: taslim.cse@aust.edu).

unpredictable data being received by the destination which is called errors. There are four types of error checking methods such as Parity check, Cyclic Redundancy Check (CRC), Longitudinal Redundancy Check (LRC)/2-D Parity, and Checksum.

Fletcher [4] described Fletcher checksum error detection information in his paper. One of the error detection mechanisms is called checksum which is generated by "summing up" all the bytes or words in a data word to generate a checksum value, this is named an FCS (Frame Check Sequence) in network applications. The checksum is added to the data word and dispatched with it, building up this a systematic code in which the data being transmitted is the code word same. Receiver recalculates the checksum of the received data and compares it to the received checksum value. If the calculated and received checksum are the same, then it is clear that there is no transmission error. The checksums generally are divided into three general areas of cost/performance trade-off. The simplest and least effective checksums include a simple "sum" function over all bytes or words in a communication. The simple "sum" functions are three types similar to XOR, two's complement addition, and one's complement addition. These checksums deal with weak error discovery content but have very veritably computational cost. Adler [5] executed an enhancement over the Fletcher checksum.

A. Nakassis [6] and K. Sklower [7] designed effective executions for the Fletcher checksum. D. Sheinwald, et al. [8] enforced an logical comparison of CRC-32, Fletcher-32, and Adler-32 (for a code word length of 8KB). The Fletcher checksum [4] and the later Adler checksum [5] are both constructed with error discovery parcels nearly as good as CRCs with significantly dropped computational cost. References [9], [10], [11], [12], and [13] proposed the error discovery effectiveness of two's complement addition and one's complement addition checksums. Stone et al. [14], [15], [16] estimated the network checksum prosecution of the one's complement addition checksum, Fletcher checksum, and CRC.

Gottfried Wilhelm Leibniz [19] constructed the ultramodern binary number system in 1689 and appears in his article Explication de I'Arithmetique Binaire. He showed the binary bits for the decimal number zero to thirty two number and binary operations in his paper. John Nystrom's [17] executed a new system of computation and metrology, called the tonal system which was the full conception of the hexadecimal number system in 1869. The hexadecimal system was interpolated to the computing world by IBM in 1963. At first interpretation, the Bendix G-15 computer was applied to the integers 0-9 and u-z in 1956 [23]

III. THE DOTRIACONTAL NUMBER SYSTEM

The dotriacontal number is one with a base of 32, containing 32 single-character digits or symbols. The first 16 digits have been taken from the digits of the hexadecimal number system like 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. The remaining 16 digits are denoted by the symbols G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, and V representing the decimal values 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, and 31 respectively. Here the largest single digit is V or 31 which is one less than the base. Each position of the dotriacontal number system is a power of the base (32). Therefore, the decimal numeral identical of the dotriacontal numeral $(C7V)_{32}$ is in the following:

Now $(C7V)_{32} = C\times(32)^2+7\times(32)^1+V\times(32)^0$

$=12\times1024+7\times32+31\times1=12328+224+31=12583$

Hence, $(C7V)_{32} = (12583)_{10}$

Observe that since there are only 32 digits in the dotriacontal number system, 5 bits $2^5=32$ are sufficient to represent any digit of the dotriacontal number system in binary number system. Five bits of the binary digits is called nickel. That is a nickel equal to one digit of the dotriacontal number system. We also know that two nickel units are ten bits of the binary digits, which is called deckle. That is a deckle equal to two digits of the dotriacontal number system.

IV. RESEARCH METHOD

A. *Conversion from One Number System to Dotriacontal Number System*

1) Conversion from Binary number to Dotriacontal number system

To change a binary to dotriacontal, settle the bits in a class of five bits and locate the dotriacontal number equivalent of each class. However, put 0's to the left-hand side of the bits, when the bits cannot be partitioned exactly into a class of five bits.

Example 4.1: Convert binary number $(11101011011)_2$ to dotriacontal number.

Solution: First settle the bits into a class of five bits and then put the dotriacontal number of each class.

Now $(1011101011011)_2= \underline{00101}\ \ \underline{11010}\ \ \underline{11011}$

5 26(Q) 27(R)

Hence, $(11101011011)_2 = (5QR)_{32}$.

Enumeration applying dotriacontal number are given bellow:

0 to 9, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, 10, 11, 12, to 19, 1A, 1B, 1C to 1V, 20, 21, 22, to 29, 2A, 2B, 2C, to 2V, 30, 31, to 39, 3A, 3B, 3C, to 3V, 40, 41, … etc.

B. *Conversion from Octal Number System to Dotriacontal Number System*

We can convert an octal number to dotriacontal in two gradations:

Gradation 1: In the first gradation, we convert the octal number to binary.

Gradation 2: In the second gradation, we convert binary number to dotriacontal.

Example 4.2: Convert octal number $(765.6)_8$ to dotriacontal number.

Gradation 1: To change an octal number to binary, we settle 3 bits binary equivalent of each octal digit in the same order. $(765.6)_8 = (111110101.110)_2$

Gradation 2: To alter binary to dotriacontal. First settle the binary bits into a class of five bits. Beginning from the radix point, we proceed to the left in the whole number and in the fractional part, we proceed to the right.

Now $(111110101.110)_2 = \underset{F}{\underline{01111}}\ \underset{L}{\underline{10101}}\ \underset{O}{\underline{1011}}$

or $(111110101.110)_2 = (01111\ 10101\ .11000)_2 = (FL.O)_{32}$,

where 15=F, 21=L, and 24=O.

Hence, $(765.6)_8 = (\ FL.O)_{32}$.

C. Conversion from Decimal Number System to Dotriacontal Number System

A decimal integer can be converted to a dotriacontal number using the repeated division method, where the decimal integer divided by 32 until the quotient is equal to 0, and the remainders of each division will represent the dotraicontal number. The residual are ordered from last to first; that is, the first remainder will be the least significant digit of the dotriacontal number. That way, the most significant digit will be the last remainder.

Example 4.3: What is the dotriacontal number of decimal number $(912)_{10}$

Step 1: Divide the decimal number 912 by 32 = 912/32 = 28 + remaider of 16 (G).

Step 2: Divide 28 by 32 = 28/32 = 0 + remainder of 28 (S).

As with the last division a quotient less than 32 is obtained, it means that the result has been found; The remainders have to order inversely, in such a way that the dotraicontal number of decimal 912 is SG. Hence, $(912)_{10} = (SG)_{32}$.

D. The following table refers to the facile transformation of the remaining 16 digits of the dotriacontal number system from binary to decimal, octal, hexadecimal, and dotriacontal.

Table 1. Relation between different number systems

Decimal	Binary	Octal	Hexadecimal	Dotriacontal
Bs-10	Bs-2	Bs.-8	Bs-16	Bs-32
16	10000	20	10	G
17	10001	21	11	H
18	10010	22	12	I
19	10011	23	13	J
20	10100	24	14	K
21	10101	25	15	L
22	10110	26	16	M
23	10111	27	17	N
24	11000	30	18	O
25	11001	31	19	P
26	11010	32	1A	Q
27	11011	33	1B	R
28	11100	34	1C	S
29	11101	35	1D	T
30	11110	36	1E	U
31	11111	37	1F	V
32	100000	40	20	10

V. SYSTEM IMPLEMENTATION

A. Memory Specification in Dotriacontal Number System

The JEDEC memory recognized refers to the particular for semiconductor memory circuits and similar storage devices proclaimed by the Joint Electron Device Engineering Council (JEDEC) Solid State Technology Association. The particular refers the two familiar units of information:

1. The bits 0 and 1.
2. The byte (B) becomes a binary attribute string. That is, one byte is equal to 8 bits. Three prefixes of particular cites as follows:

Kilo (K): A multiplier equal to 2^{10} or 1024, Mega (M): A multiplier equal to 2^{20} or 1048576 or K^2, where K = 1024, Giga (G): A multiplier equal to 2^{30} or 1073741824 or K^3, where K = 1024.

The particular prefixes involve the following familiar terms:

1 KB = 2^{10} or 1024 Byte = 8192 bits, 1 MB = 2^{10} or 1024KB = 2^{20} or 1048576 Byte = 1048576×8 bits = 8388608 bits, 1 GB = 2^{30} or 1073741824 = 1073741824× 8 bits = 8589934592 bits, 1 TB = 2^{40} or (1.0995116e+12) Byte.

On the other hand, we proposed that memory storage capacity specification can be defined in the dotriacontal digits as follows: One digit of dotriacontal number is five bits which is called nickel, and two digits of dotriacontal number is ten bits which is called deckle. That is, 1 nickel equal to 5 bits, 1 deckle equal to 10 bits.

1 KD = 2^{10} or 1024 Deckle = 1024×10 bits = 10240 bits, 1 MD = 2^{10} or 1024KD = 2^{20} or 1048576 Deckle = 1048576×10 bits = 10485760 bits, 1 GD = 2^{30} or 1073741824 Deckle = 1073741824×10 bits = 10737741824 bits, 1 TD = 2^{40} or (1.0995116e+12) Deckle = (1.0995116e+12) × 10 bits

We observed that a large amount of bits can be processed by this deckle system, so it can be used to specify memory specification.

B. Computer Memory Addressing in Dotriacontal Number System

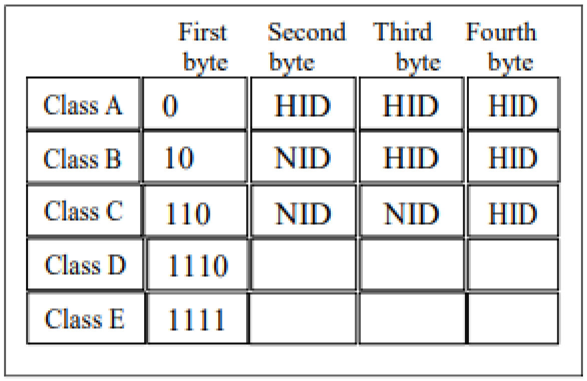
Traditionally, computer memory addresses are displayed in five hexadecimal digits such as conventional memory addresses 640K range from 0000:0 to 9FFF:F, because of 640K = 640 × 1024 = 6,55,360 and if this number is converted to hexadecimal is A0000. So, conventional memory addresses begin with 00000h and end with A0000h minus 1h or 9FFFFh, and the upper memory is defined as the memory addresses from 640K to 1024K which is displayed by A000:0 to FFFF:F in the hexadecimal number.

We proposed that computer memory addresses can be displayed in four dotriacontal integers rather of five hexadecimal integers, which will reduce the memory consumption. If the traditional memory 640K = 640 × 1024 = 6,55,360 is converted to dotriacontal integers, it is K000dt. So, conventional memory addresses begin with 0000dt and end with K000dt minus 1dt or JVVVdt. Hence, the conventional memory addresses range from 0000 to JVVV. In an analogous way, the upper memory addresses will range from K00:0 to VVVV, and the memory addresses from 1024KB to 1024MB is defined by 10000 to VVVVVV in the dotriacontal number. Observed that if we convert the memory addressing in four dotriacontal integers in-vantage of five hexadecimal integers, the memory consumption will be saved 20% memory position.

C. *The IPv4 Addresses in Dotriacontal Number System*

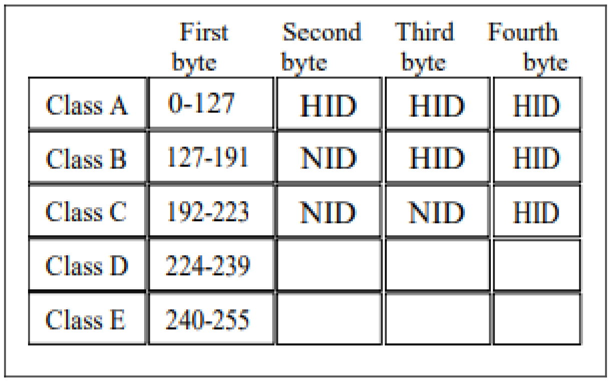
Generally, IPv4 is a 32-bit address that specifically and widely prescribes relations between devices and the internet. The IPv4 address is about 2^{32} or 4,294,967,296 addresses. The IPv4 addressing used the concept of classes at the pioneer level. This addressing is called group addressing. The address space can be classified within five groups: A, B, C, D, and E. This group in binary and dotted decimal notation are in the following Fig. 5.1 (a) and (b).

Fig. 5.1 (a) Binary notation



	First byte	Second byte	Third byte	Fourth byte
Class A	0	HID	HID	HID
Class B	10	NID	HID	HID
Class C	110	NID	NID	HID
Class D	1110			
Class E	1111			

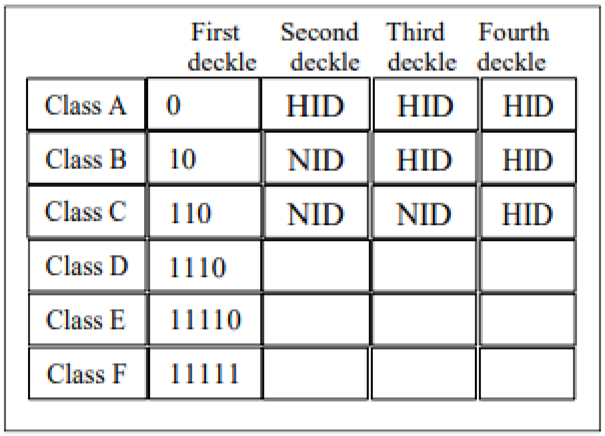
Fig. 5.1 (b) Dotted decimal notation



	First byte	Second byte	Third byte	Fourth byte
Class A	0-127	HID	HID	HID
Class B	127-191	NID	HID	HID
Class C	192-223	NID	NID	HID
Class D	224-239			
Class E	240-255			

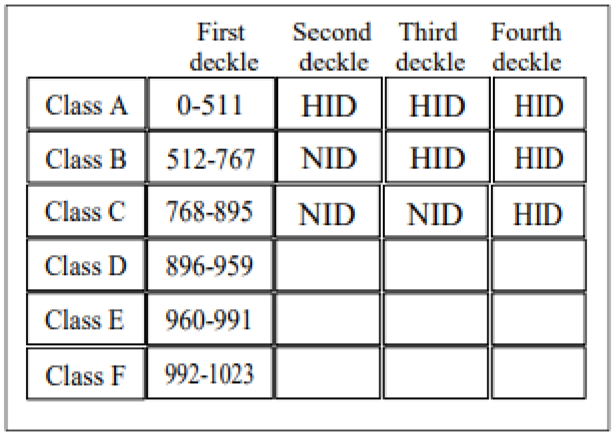
In the dotriacontal number system, one dotriacontal digit resembles a group of five contiguous binary bits, called a nickel. Similarly, two contiguous dotriacontal digits resemble a group of ten contiguous binary bits, called a deckle. We proposed that the IPv4 addresses can be defined in 8 digits of dotriacontal number as a 40-bit address that uniquely defines the connection of a device to the internet. Since $(32)^8 = (2^5)^8 = 2^{40}$, the address space of IPv4 is 2^{40} or 109951,16,27,776 whereas IPv4 address 2^{32} or 429,49,67,296. The IPv4 address will be increased by 25500 percent. In classful addressing, the address space is divided into six classes: A, B, C, D, E, and F. The classful address in binary and dotted decimal notation are given in the following Fig. 5.2 (a) and (b).

Fig. 5.2 (a) Binary notation



	First deckle	Second deckle	Third deckle	Fourth deckle
Class A	0	HID	HID	HID
Class B	10	NID	HID	HID
Class C	110	NID	NID	HID
Class D	1110			
Class E	11110			
Class F	11111			

Fig. 5.2 (b) Dotted decimal notation



	First deckle	Second deckle	Third deckle	Fourth deckle
Class A	0-511	HID	HID	HID
Class B	512-767	NID	HID	HID
Class C	768-895	NID	NID	HID
Class D	896-959			
Class E	960-991			
Class F	992-1023			

D. *MAC Addresses in Dotriacontal Number System*

Traditionally, MAC address is Media Access Control address, which is called Layer 2 address or Physical address or hardware address. These 48 bits (6 bytes) binary addresses are denoted in hexadecimal systems to build it simple for humans to recite and understand. MAC addresses are permanent numbers, which are burned into the network card. MAC tact FF:FF:FF:FF:FF:FF is fixed for Broadcast type of communication. An Ethernet Switch will deluge an Ethernet Frame with FF:FF:FF:FF:FF:FF as the goal MAC address to all its related ports. The address of MAC is typically nearby in particular. The address of MAC is defined within a Local Area Network (LAN). MAC- 01:00:5E:00:00:00 to 01:00:5E:7F:FF:FF are the scope of Ethernet multicast physical addresses that is IPv4 Multicast. The MAC address 2^{48} is defined by MAC-48 based on EUI-48 (Extended Unique Identifier - 48) [18].

We proposed that the MAC address can be defined as six deckles or 12 digits of the dotriacontal number that is $(32)^{12}=(2^5)^{12}=2^{60}$, which is MAC address 2^{60} in instead of six octet MAC address 2^{48} in hexadecimal digits. Then a large number of MAC addresses will be increased within the twelve dotriacontal digits. The MAC address 2^{60} is defined by MAC-60 based on EUI-60 (Extended Unique Identifier-60). The following table shows that 10 bits (1 deckle) can be a common binary grouping for representing MAC addresses. The binary 00000 00000 to 11111 11111 can be represented in dotriacontal number as the range 00 to VV.

We proposed that the MAC address VV:VV:VV:VV:VV:VV will be reserved for broadcast type communication. That is, the Ethernet physical address is six deckles long. From the Fig. 4.2 (b) and the table 2, we can define the range of IPv4 multicast address 896.0.0.0 to 959.1023.1023.1023 or S0:00:00:00 to TV:VV:VV:VV in dotriacontal digits. In a similar way, we proposed that an Ethernet multicast physical address could be in the range 01:00:2U:00:00:00 to 01:00:2U:3V:VV:VV in dotriacontal digits.

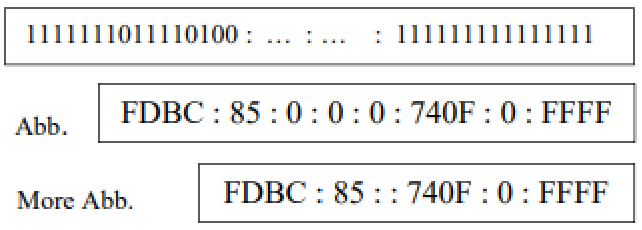
Observed that if the address of MAC is defined in the dotriacontal digits instead of hexadecimal digits, it could be increased more and more addresses.

Table 2: Some selected examples of decimal to binary to dotriacontal number

Decimal	Binary	Dotriacontal number
16	00000 10000	0G
17	00000 10001	0H
18	00000 10010	0I
19	00000 10011	0J
20	00000 10100	0K
31	00000 11111	0V
94	00010 11110	2U
240	00111 10000	7G
255	00111 11111	7V
511	01111 11111	FV
512	10000 00000	G0
768	11000 00000	O0
896	11100 00000	S0
959	11101 11111	TV
960	11110 00000	U0
1023	11111 11111	VV

E. *IPv6 Addresses in Dotriacontal Number System*

Typically, the IPv6 addresses contain 16 bytes (octets), that is, 128 bits extensive in 32 hexadecimal integers. The IPv6 address in binary and hexadecimal colon notation are in the following [18]. 128 bits = 16 bytes = 32 hexadecimal integers.

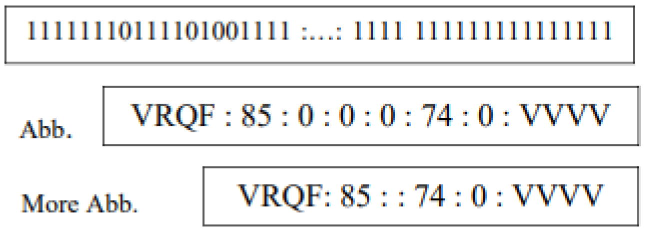


1111111011110100 : ... : ... : 1111111111111111

Abb. FDBC : 85 : 0 : 0 : 0 : 740F : 0 : FFFF

More Abb. FDBC : 85 : : 740F : 0 : FFFF

We proposed that if the IPv6 is defined in 32 digits of the dotriacontal number, it will be 160 bits long. Since $(32)^{32}=(2^5)^{32}=2^{160}$, the address space of IPv6 is about 2^{160} or 1.46150164E48 whereas IPv6 address 2^{128} or 3.40282367E38. The IPv6 address can be increased more and more by the dotriacontal number system. The IPv6 address in binary and dotriacontal colon notation is in the following.



11111110111101001111 :...: 1111 111111111111111

Abb. VRQF : 85 : 0 : 0 : 0 : 74 : 0 : VVVV

More Abb. VRQF: 85 : : 74 : 0 : VVVV

VI. Simulation and Result

A. *Checksumn Error Detecting Method in Dotriacontal Number System*

Generally, 2 byte (16-bits) checksum is applied in the internet. But we proposed that n-byte (8n-bit) checksum can be used to detect errors on the internet. In the source site the checksum is enumerated by subsequent stages.

Source site:

1. The message is grouped within n-byte (8n-bit) segment.
2. The ASCII is applied to convert every byte to a 2-integers of the dotriacontal.
3. Set 0 to the initial value of checksum.
4. Aggregate all segments with initial checksum using one's complement addition.
5. The complement of the aggregate is the checksum.
6. The checksum is transmitted along the datum.

Assigner site:

The assigner applies subsequent stages for error discovery:

1. The message (including checksum) is grouped within 8 digit segments of the dotriacontal integers.
2. All segments are aggregated using one's complement addition.
3. The new checksum is multipliered of the aggregate value.

4. If the checksum is 0, the message contains no error; otherwise it contains an error.
5. If 8 digit segments are divided within 2 contiguous digits of the dotriacontal and these digits are changed to letter or symbol using ASCII code, then the original message will display.

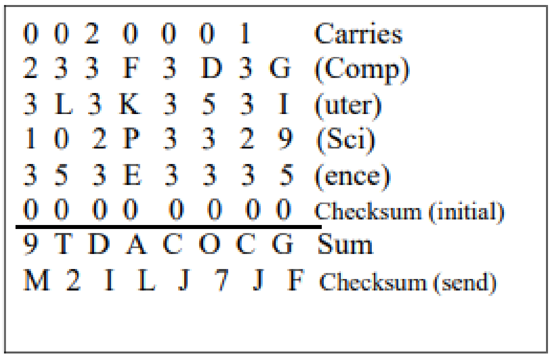
The checksum is applicable for software execution. The programs could be applied to enumerate the checksum in both sites and the receiver site could test the correctness of the message.

Example 6.1: Suppose we have to send the message "Computer Science" to receiver.

The checksum can be enumerated for the given message using subsequent stages:

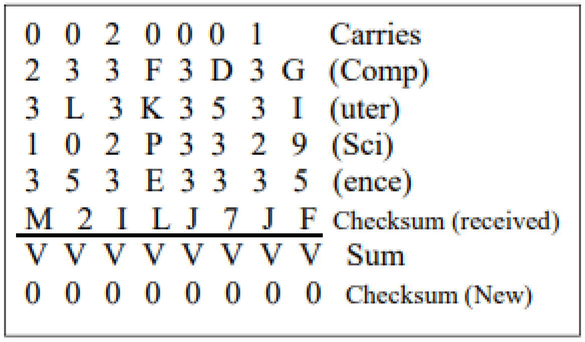
1. The message needs to be grouped within 4-byte (32-bit) segments.
2. From the table 3, we apply ASCII code to change each byte to 2 digits of the dotriacontal number.
3. The first character C is defined as 23dt and O is defined as 3Fdt.
4. In Figure 6.4 (a) the first column is enumerated in dotriacontal digits that is 1G dt. We put right digit G and left digit 1 sum up to the second column as carry. This procedure is repetition for every column.
5. Figure 6.5 (b) presents how the transmitted checksum and segmented are enumerated at the assigner site and the new checksum is complemented by the aggregate value. Dotriacontal numbers are reviewed in Table 3.

Fig. 6.4 (a) Checksum at the source site



0	0	2	0	0	0	1		Carries
2	3	3	F	3	D	3	G	(Comp)
3	L	3	K	3	5	3	I	(uter)
1	0	2	P	3	3	2	9	(Sci)
3	5	3	E	3	3	3	5	(ence)
0	0	0	0	0	0	0	0	Checksum (initial)
9	T	D	A	C	O	C	G	Sum
M	2	I	L	J	7	J	F	Checksum (send)

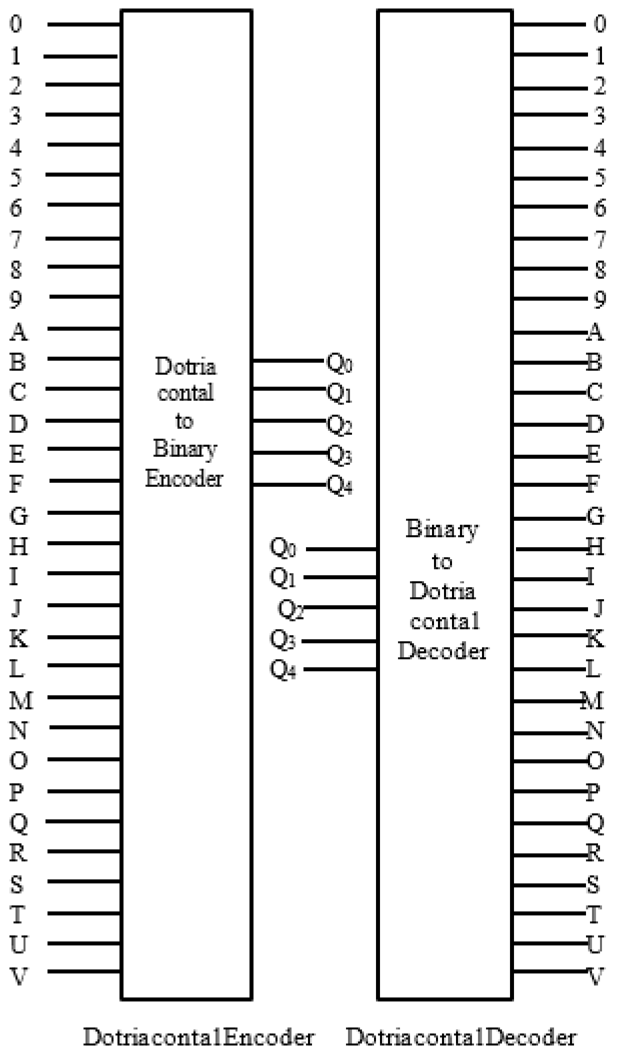
Fig. 6.4 (b) Checksum at the assigner site



0	0	2	0	0	0	1		Carries
2	3	3	F	3	D	3	G	(Comp)
3	L	3	K	3	5	3	I	(uter)
1	0	2	P	3	3	2	9	(Sci)
3	5	3	E	3	3	3	5	(ence)
M	2	I	L	J	7	J	F	Checksum (received)
V	V	V	V	V	V	V	V	Sum
0	0	0	0	0	0	0	0	Checksum (New)

B. Dotriacontal to Binary Encoder and Decoder

Fig. 6.1 Block diagram of dotriacontal to binary encoder and decoder


0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L M N O P Q R S T U V
Dotria contal to Binary Encoder
Q0 Q1 Q2 Q3 Q4
Q0 Q1 Q2 Q3 Q4
Binary to Dotria contal Decoder
0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L M N O P Q R S T U V
DotriacontalEncoder
DotriacontalDecoder

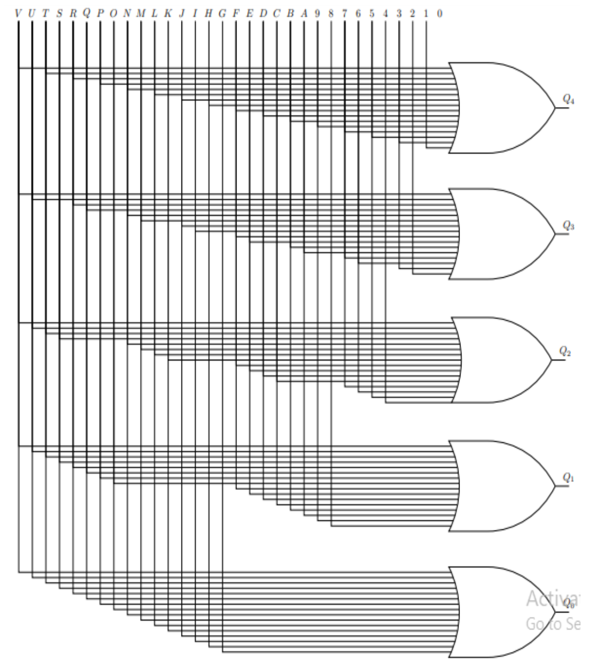
Logical expression Q_0, Q_1, Q_2, Q_3, and Q_4 using the truth table

$$Q_0 = \sum(G,H,I,J,K,M,N,O,P,Q.R,S,T,U,V)$$
$$Q_1 = \sum(8,9,A,B,C,D,E,F,O,P,Q,R,S,T,U,V)$$
$$Q_2 = \sum(4,5,6,7,C,D,E,F,K,L,M,N,S,T,U,V)$$
$$Q_3 = \sum(2,3,6,7,A,B,E,F,I,J,M,N,Q.R,U,V)$$
$$Q_4 = \sum(1,3,5,7,9,B,D,F,H,J,L,N,P,R,T,V)$$

1) Dotriacontal to Binary Encoder (32 to 5 Encoder)

This encoder contains 32 inputs and 5 outputs, which is called dotriacontal to binary encoder. Because dotriacontal number system consists of 32 digits and it produces 5 digit binary code corresponding to input line.

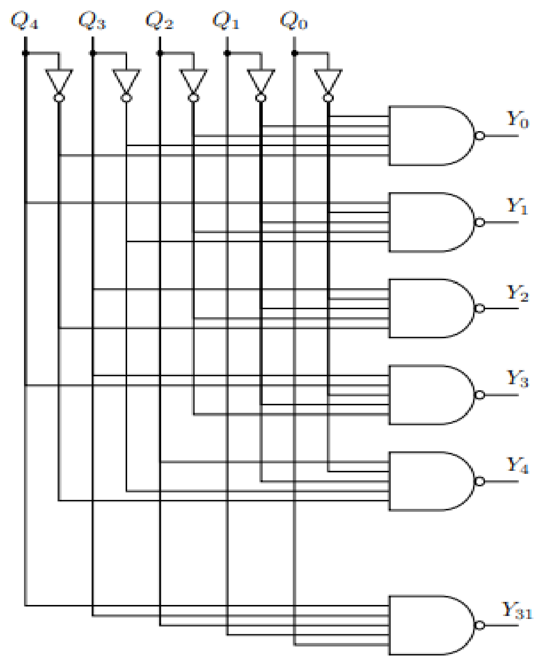
Fig. 6.2 Implementation of logic circuit of encoder


V U T S R Q P O N M L K J I H G F E D C B A 9 8 7 6 5 4 3 2 1 0
Q_4
Q_3
Q_2
Q_1
Q_0

1) Binary to Dotriacontal Decoder (5 to 32 Decoder)

This decoder is formed of 5 inputs and 32 outputs. Like the existing decoder only one output will be low and all other outputs are high at a given time using maxterms. It can be applied by 5 NOT gates and 32 NAND gates. Here NAND gates are required to produce the active low outputs, which is shown in the Fig. 5.3. Since its outputs are the 32 digits of dotriacontal numbers, this decoder is said to be binary-to-dotriacontal decoder.

Fig. 6.3 Implementation of logic circuit of decoder



C. The following table refers to the easy conversion from letter or symbol to decimal, and dotriacontal number.

Table 3: Some selected examples of ASCII codes to decimal to dotriacontal number

Letter/ Symbol	**Decimal**	**Dotriacontal/ dt**
Space	32	10
C	67	23
S	83	2P
c	99	33
d	100	34
e	101	35
f	102	37
i	105	29
m	109	3D
n	110	3E
o	111	3F
p	112	3G
q	113	3H
r	114	3I
s	115	3J
t	116	3K
u	117	3L

VII. Discussion and Finding

A. The following table refers to the easy comparison between IPv4 address 32 bits and IPv4 address 40 bits

Table 4: The comparison between IPv4 address 32 bits and IPv4 address 40 bits.

Meth.	**IPv4 Addresses of 32 bits**		**IPv4 Addresses of 40 bits**		**App.**
Class	No. of Blocks	Block Size	No. of Blocks	Block Size	
A	2^7 or 128	2^{24}	2^9 or 512	2^{30}	Unicast
B	2^{14} or 16384	2^{16}	2^{18} or 2,62,144	2^{20}	Unicast
C	2^{21} or 20,97,152	2^8	2^{27} or 13,42,17,728	2^{10}	Unicast
D	1	2^{28}	1	2^{40}	M. cast
E	1	2^{28}	1	2^{40}	Reserve
F	No blocks	No size	1	2^{40}	Reserve
S.net mask	0 – 255		0 – 1023		
Add. space	2^{32} or 429,49,67,296		2^{40} or 109951,16,27,776		

B. The Comparison between hexadecimal number and dotriacontal number based on the memory addresses and MAC addresses is in the following. During comparison it is found that 20% memory location saved in dotriacontal number system over the hexadecimal number system.

Table 5: The comparison between hexadecimal number and dotriacontal number system.

Hexadecimal number	**Dotriacontal number**
One digit is four binary bits, which is called nibble.	One digit is five binary bits, which is called nickel.
Two digits are equal to eight binary bits, which is called byte.	Two digits are equal to ten binary bits, which is called deckle.
Hexadecimal numbers are compact and use less memory, so more numbers can be stored in computer systems.	Dotriacontal numbers are more compact and use less memory, so more numbers can be stored than hexadecimal in computer systems.
In memory addressing, we need five hexadecimal digits.	In memory addressing, we need four dotriacontal digits and save 20% memory location.
The conventional memory addresses range from 0 to 640K, which is defined by 00000 to 9FFFF hexadecimal digits.	The conventional memory addresses range from 0 to 640K, which is defined by 000:0 to JVVV dotriacontal digits
The upper memory addresses from 640K to 1024K, which is defined by A0000 to FFFFF hexadecimal digits.	The upper memory addresses from 640K to 1024K, which is defined by K00:0 to VVVV dotriacontal digits.
The MAC address 2^{48} is defined by MAC-48 (6 bytes) based on EUI-48.	The MAC address 2^{60} is defined by MAC-60 (6 deckles) based on EUI-60.
8 hexadecimal digits represent 32 bits (4 bytes) IPv4 multicast addresses	8 dotriacontal digits can represent 40 bits (4 deckles) IPv4 multicast addresses
The IPv6 addresses is consists of (octets) that is 128 bits (16 bytes) long.	The IPv6 addresses is consists of (deckles) that is 160 bits (16 deckles) long.

VIII. CONCLUSION

We have constructed the dotriacontal number system for computing memory addresses, MAC, IPv4, and IPv6 addresses, and also detect error messages instead of the hexadecimal number system in checksum error detecting method. We observe that if this number system is displayed for memory addressing in the computer instead of a hexadecimal number. The memory consumption can be reduced 20% and also a large number of MAC, IPv4, and IPv6 addresses can be created by using dotriacontal digits instead of hexadecimal digits. Also, to track out errors the message of any size can be divided within n-byte (8n-bit) words in checksum method applying this number system. If the memory specification can be specified in nickel and deckle of dotriacontal digits instead of bytes, the computer processing speed will be very faster.

IX. FUTURE WORK

Dotriacontal number system is a large number system than exiting number systems like Decimal, Binary, Octal, and Hexadecimal number systems because it has 32 single-character digits or symbols and each symbol contains five binary bits. The contiguous five binary bits is called nickle and ten contiguous bits or two nickles is called deckle. There are many more to assignment or research in future to make this number system most implement and efficient to apply in computer and datum communication. At first we have to implement this number system for memory address displaying in computer instead of hexadecimal number. Then computer will be very faster for datum processing and reduce memory consumption and also increase the number of addresses of MAC, IPv4, and IPv6 address, if this number system is implemented. In datum communication, this number system will be very applicable for error detection and correction.

ACKNOWLEDGMENT

The authors are grateful to the honorable director of IIT, Jahangirnagar University, Bangladesh for the continuous support and encouragement.

REFERENCES

[1] Fiete, I. R.; Seung, H. S., "Neural network models of birdsong production, learning, and coding". In Squire, L.; Albright, T.; Bloom, F.; Gage, F.; Spitzer, N. New Encyclopedia of Neuroscience, 2007.

[2] Number Word and Number Symbols: A Culture History of Numbers Paperback, Dover Publications, Inc., New York, November, 1969, 2011(Republication).

[3] Sheraz A. K., M. Moosa, "Protocols and Mechanisms to Recover Failed Packets in Wireless Networks: History and Evolution", IEEE, July, 2016.

[4] J. G. Fletcher, "An Arithmetic Checksum for Serial Transmissions," IEEE Trans. Comm., vol. 30. No. 1, pp. 24/-252, Jan. 1982.

[5] P. Deutsch and J.-L. Gailly, *ZLIB Compressed Data Format Specification Version 3.3,* IETF RFC 1950, May 1996.

[6] A. Nakassis. Fletcher's error detection algorithm: How to implement it efficiently and how to avoid the most common pitfalls. Computer Communication Review, 18(5):63–88, Oct. 1988.

[7] K. Sklower. Improving the efficiency of the OSI checksum calculation. Computer Communication Review, 19(5):44–55, Oct. 1989.

[8] D. Sheinwald, J. Satran, P. Thaler, and V. Cavanna. Internet protocol small computer system interface (iSCSI) cyclic re-dundancy check (CRC) / checksum considerations. Network Working Group Request for Comments (RFC) 3385, Sept. 2002.

[9] N. R. Saxena and E. J. McCluskey, "Analysis of Checksums, Extended-Precision Checksums, and Cyclic Redundancy Checks," *IEEE Trans. Computers,* vol. 39, no. 7, pp. 969-975, July 1990.

[10] A. M. Usas, "Checksum versus Residue Codes for Multiple Error Detection," *Proc. Eighth Ann. Int'l Symp. Fault-Tolerant Computing (FTCS '78),* p. 224, 1978.

[11] S. C. Tzou Chen and G.S. Fang, "A Closed-Form Expression for the Probability of Checksum Violation," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 10, no. 7, pp. 407-410, July 1980.

[12] C. Jiao and L. Schwiebert, "Error Masking Probability of 1's Complement Checksums," *Proc. 10th Int'l Conf. Computer Comm. and Networks (ICCCN '01),* pp. 505-510, Oct. 2001.

[13] Y. Desaki, K. Iwasaki, Y. Miura, and D. Yokota, "Double and Triple

Error Detecting Capability of Internet Checksum and Estimation of Probability of Undetectable Error," *Proc. Pacific Rim Int'l Symp. Fault-Tolerant Systems (PRFTS '97),* pp. 47-52, Dec. 1997.

[14] J. Stone and C. Partridge, "When the CRC and TCP Checksum Disagree," *Computer Comm. Rev., Proc. ACM SIGCOMM '00,* vol. 30, no. 4, pp. 309-319, Oct. 2000.

[15] J. Stone, M. Greenwald, C. Partridge, and J. Hughes, "Performance of Checksums and CRC's over Real Data," *IEEE/ACM Trans. Networking,* vol. 6, no. 5, pp. 529-543, Oct. 1998.

[16] C. Partridge, J. Hughes, and J. Stone, "Performance of Checksums and CRCs over Real Data," *Computer Comm. Rev., Proc. ACM SIGCOMM '95,* vol. 25, no. 4, pp. 68-76, Oct. 1995.

[17] John Nystrom's "A New System of Arithmetic and Metrology, called the Tonal System", pp 263-275, 337-348, 402-407 (concluded), and illustrated, including a folding plate. In Journal of the Franklin Institute, third series vol. 46 overall vol. 76, 1863, 432pp.

[18] Behrouz A Frorouza, "Data Communications and Networking" Tata McGraw-Hill Publishing Company Limited, ISBN-13: 978-0-07-063414-5, 2006.

[20] Gottfried Wilhelm Leibnitz, "Explication de l'arithmétique binaire", Académie royale des sciences, 1703. Ads-00104781. https://hal.archives-ouvertes.fr/ads-00104781/document

or Explanation of Binary Arithmetic, De mathematische schriften von Gottfried Wilhelm Leibnitz, vol. VII, pp 223—227, 1703. https://www.leibniz-translations.com/binary.htm

[21] Nikolai Weibull, "An historical Survey of Number Systems."

File:///E:/Articles of project purpose/ Number_system.pdf

[22] Teach Computer Science: https://teachcomputerscience.com/uses-of-hexadecimal/

[23] History of hexadecimal number systems, https://www.slideshare.net/nandini44/history-of-hexadecimal-number-system