VOL. 3, NO. 4, April 2012 ISSN 2079-8407
Journal of Emerging Trends in Computing and Information Sciences

http://www.cisjournal.org

A Comparative Analysis and Application of the Compression Properties of Two 7-Bit Subsets of Unicode

'Dele Oluwade

Dewade Systems Consult, U. I. P. O. B. 20253, Ibadan, OYO 200005,
Federal Republic of Nigeria

Email: deleoluwade@yahoo.com

ABSTRACT

Data compression is important in the computing process because it helps to reduce the space occupied by a file, which normally leads to the reduction in the time taken to access the file. Files which may be compressed include text, images, video/speech and sound. Data compression algorithms have been developed and applied to several areas including neural networks, bioinformatics, database management systems, wireless systems, error detection and correction codes, fractals etc. Since computer/communication codes are binary texts, they may be compressed using binary text compression algorithms. By considering two binary texts which can be modeled as subsets of Unicode, namely the numerics of a 7-bit subset of the Information Processing Code (IPC) and the numerics of the 7-bit American Standard Code for Information Interchange (ASCII) respectively, this paper presents a comparative analysis of the compression properties of these two texts using a binary text compression algorithm. The algorithm is a generalization of a storage-based built-in test pattern generation method for reducing storage requirements and test application time in circuits. This algorithm is then applied to the design of an IPC to ASCII code converter. Code converters are circuits which accept input codes in one form and translate them to present equivalent values in a different format as the output code. They are useful in real-time control systems and data acquisition systems, such as aviation systems and patient-monitoring systems in hospitals, which require the use of different sensors to continuously monitor the computer. This paper thus presents an equivalent approach to the design of code converters via the well-known k-map method.

Keywords: *Data compression properties, Unicode, Code converter, K-map method, Information Processing Code, ASCII.*

1. INTRODUCTION

Data compression process assists in reducing the space occupied by a file, which normally leads to the reduction in the time taken to access the file. Files which may be compressed include text, images, video/speech and sound. In particular, text compression is important because of the large amount of texts which are available in e-libraries and on the internet including catalogs, bibliographic files and address books. Basically, there are two categories of text compression algorithms, namely dictionary-based algorithms (e.g. LZW algorithm) and statistics-based algorithms (e.g. Huffman coding, Arithmetic coding). In the former category, a string of characters is replaced with a pointer to an earlier occurrence of the string. The latter is however based on the collection of statistics on the frequency of appearance of the characters.

Data compression algorithms have been developed and applied to several areas including linguistics, neural networks, bioinformatics, database management systems, wireless systems, error detection and correction codes, fractals etc. Human languages to which these algorithms have been applied include English, Turkish, Hebrew, Japanese, Chinese and Arabic. (E.g. see [1, 2, 3, 4, 5, 6, 7, and 8]. In [5, 9], two distinct data compression algorithms, based on some form of partitioning, were presented respectively. The first algorithm, called layered set partitioning in hierarchical trees, was developed for telemedical applications in which the encoded bit streams are divided into a number of layers. The second algorithm is a binary text compression algorithm in which a binary string is first partitioned into two as a concatenation of words before the given set of binary strings is partitioned into disjoint equivalence classes in line with the Fundamental Theorem of Equivalence Relations in Abstract Algebra [10].

In the present paper, two hypothetical binary texts are modeled using two historically old subsets of Unicode, namely the bit patterns of the numerics of a 7-bit subset of the Information Processing Code (IPC) [11], and the numerics of the 7-bit American Standard Code for Information Interchange (ASCII) [12, 13], respectively. Although the former is virtually extinct in present day practical systems, the latter is still popular on modern computer systems. A comparative analysis of the compression properties of these two texts is then presented using the binary text compression algorithm described in [5].

The algorithm, which is a generalization of a storage-based built-in test pattern generation method for reducing storage requirements and test application time in circuits [14], is then applied to the design of an IPC to ASCII code converter. This algorithm had earlier been applied to the entire 7-bit ASCII code [16], instead of just the set S_1 considered in the present paper.

The IPC, in its comprehensive form, is a 128 character 8-bit code developed for general information processing and information interchange in online systems. The code is constructed in such a way that 7, 6, 5 and 4-bit subsets can be easily derived from it. An online system in this regard is defined as a system in which multiple independent users are provided service by the computer via remote consoles which asynchronously communicate with the computer on a demand basis [11]. ASCII, on the other hand, is a 128 character 7-bit code designed primarily for use on non-IBM computers. The code is the most popular coded character sets (CCSs) used on digital computers. It consists of a set of numerics (S_1), a set of uppercase and lowercase alphabetics (S_2), a set of operation and special characters (S_3) and a set of control characters (S_4) [13, 15]. In a simple form, a CCS may be described by the following equation:

$$C = \{S_1, S_2, S_3, S_4\} \quad (1.1)$$

Generally, a CCS structurally contains two parts, namely, the set of bit patterns and the set of assigned meanings, even though only the former is relevant to this paper. Unicode ('Unification Code') was designed to serve as a universal set of all possible CCSs. By virtue of its 16-bit blocklength, it has an order of 65,536 [25]. It follows that all p-bit CCSs (where p is a natural number), $p \leq 16$, are naturally subsets of Unicode. Code converters are circuits which accept input codes in one form and translate them to present equivalent values in a different format as the output code. They are useful in real-time control systems and data acquisition systems, such as aviation systems and patient-monitoring systems in hospitals, which require the use of different sensors to continuously monitor the computer. There are basically two categories of converters, namely digital-to-analog converter (DAC) and analog-to-digital converter (ADC). A DAC converts binary output signals from computers to analog signals which can be used to position mechanical and other devices. ADC, on the other hand, enables mechanical displacements to be converted to a digital representation. It also permits the conversion of an electric analog signal to digital-coded signals [17]. An example of an ADC is the shaft encoder which directly converts a physical position to a digital value. This ADC is connected to a rotating shaft to which a rotatable coded-segment disk is usually coupled. In this converter, the Gray code (otherwise known as unit-distance code) is commonly used to form the coder disk. Two bits in this code do not change value in successive coded binary numbers. If the inputs to a system are from a coder using a Gray code, then the code groups have to be converted to conventional binary or binary coded decimal (BCD) before they are used. Algorithms exist for the conversion from Gray code to binary and vice versa. ADC converter in a cellphone converts analogue electrical signals produced by the microphone into digital signals.

An earlier work involved the design of an Excess-3 to BCD code converter using k-maps [18]. This design, which used 13 gates, and others have been documented as United States patents [19]. Usually, an Excess-3 code is used when one desires to perform arithmetic operations by the method of compliments. That is, it is a self-complementing code in which the 9's complement of a decimal digit expressed in Excess-3 code may be obtained by complementing each particular individual bit e.g. the Excess-3 code representation of the decimal digit 5 is the number 1000 whose complement on an individual bit basis is 0111. This has a decimal equivalent of 4 in the Excess-3 code. The Excess-3 code is thus a modified form of BCD code [20].

Conventionally, design of code converters is usually accomplished via the well-known k-map method which is an important tool in digital electronics for the simplification / minimization of Boolean expressions. A Boolean expression provides a greater degree of convenience in the representation and simplification of switching circuits. Ultimately, it brings about economy of construction and reliability of operation. Such an expression for a given function can be derived by using a table of combinations to list desired function. In deriving a sum-of-products expression for the function, a set of product terms is first listed. Then the terms for which the function is to have a value one are selected and logically added together to form the desired expression instead of using a table of combinations. A karnaugh map (k-map) is an alternative elegant map which facilitates the minimization of Boolean expressions. This map is a matrix-like diagram which depicts the whole of the Boolean universe. It is particularly suitable when the number of variables is small [17].

In [21], a new self-documenting method of constructing k-maps was presented. Apart from assigning a unique identifier to each element in a Boolean minterm, the method also uses the identifiers to construct the map. In [22], the authors presented a "faster procedure" for the design of synchronous counters using k-maps. The procedure allows synchronous counters to be designed using J-K and R-S flip-flop implementations while at the same time requiring only one set of k-maps for both implementations.

The present paper is organized as follows: Section 1 presents the background information to the subject matter of the paper. In Section 2, an overview of the data compression algorithm is discussed and results on its application to the IPC and ASCII presented. Section 3 presents an introduction to the k-map method and the procedure for designing an IPC to ASCII code converter. Section 4 presents a discussion of the results in the paper while Section 5 is the concluding part.

2. BINARY TEXT COMPRESSION PROPERTIES OF THE IPC AND ASCII CODES

Historically, the general idea of (re)presentation in the theory of coded character sets (CCS) refers to the form in which the characters of a CCS are recorded or transmitted on certain media e.g. a data transmission line, magnetic tape, disk etc [15]. Such media representations are important in order to specify a precise relationship between the format characteristics of the medium (like rows, columns etc) and the bits of the bit pattern of a character.

A typical data in a Boolean system is usually either a numeric data or a character data. Conventional formats for representing (i.e. for coding or storing) numeric data include the sign and magnitude method, the radix/diminished radix form, the fixed point representation, decimal representation and the floating point representation [17]. In the conventional representation of character data on the byte addressable IBM 360/370 digital computer, which forms the basis for the present-day computers, an n-bit word of a uniform digital code is divided into two halves where the first half (i.e. the most significant bits) is called a zoned portion and the second half (i.e. the least significant bits) is called a numeric portion [15]. The above traditional representation provides a motivating force for the development of the data compression algorithm [5], otherwise referred to as code presentation technique [13, 16, 23, 24, 25], which has been applied in this paper. The algorithm is similar to the concept of "group presentation" in abstract algebra whereby a group is presented (i.e. written in a simple and condensed form) in terms of a set of generators and a set of relators [26, 27].

In the technique, the first n/2 bits of a word or bit pattern of an even blocklength code C is called a zoned portion while the remaining n/2 bits (i.e. the n/2 least significant bits) is called a numeric portion. For a code having an odd block length, two possible cases are considered. The case in which the zoned portion refers to the first (n + 1)/2 bits is called Type I definition while the case in which the zoned portion refers to the first (n – 1)/2 bits is called Type II definition of zoned portion. The remaining bits in each case are collectively referred to as numeric portion. A zoned portion which is common to two or more words is said to be a constant zoned portion. A subset E of the code C is said to be an equizone if all the words in E have the same zoned portion. Using the concept of equivalence relations, an element (or word) of a code is said to be in a mathematical relation with another element of the same code if the two elements have the same zoned portion. This relation is obviously an equivalence relation since it is reflexive, symmetric and transitive. By the Fundamental Theorem of Equivalence Relations, C is partitioned into disjoint equivalence classes (which are the equizones). The degree (d) of C is defined as the number of equizones in the code. The zoned set (or zoned code) refers to the order-preserving set of all the constant zoned portions of C while the numer set (or numer code) is the order-preserving set of all the code's numeric portions. The ordering used in the algorithm is the collating sequence of a code. Thus the ith equizone refers to the equizone whose zoned portion is the ith word of the zoned set. The zoned set $\{t_i\}$ for (i = 1, 2, …, d) is the ordered set in which t_1 is the first word, t_2 the second word etc. A decinumer of an equizone E is the decimal representation of a numeric portion of the equizone while the order-preserving set of all the decinumers of E is called a decinumer set. In general, the code presentation C {P} of an arbitrary uniform digital code C is defined by

$$C\{P\} = \bigcup_{i=1}^{d} \{ z_i x_{ig} \; \forall \; g \in Q_i \} \qquad (2.1)$$

where d is the degree of the code, z_i the constant zoned portion, x_{ig} the bit pattern of a word belonging to equizone E_i and Q_i the decinumer set for equizone E_i.

Table 2.1 shows the zoned sets of the numerics of the Information Processing Code (IPC) and the numerics of the American Standard Code for Information Interchange (ASCII).

Table 2.1: The Zoned Sets of the Numerics of the IPC and ASCII.

	IPC	ASCII
Type I definition	{0000,0001}	{0110, 0111}
Type II definition	{ 000}	{011}

Table 2.2: The Ordered Set of the Decinumer Sets of the Numerics of the IPC and ASCII.

	IPC	ASCII
Type I definition	{{0,1,2,…,7}, {0,1}}	{0,1}
Type II definition	{0,1,2,…, 9}	{0,1,2,…,9}

Emphasis in this paper is placed on the Type II definition of zoned portion.

3. APPLICATION TO THE DESIGN OF CODE CONVERTERS

Converters play an important role in real-time control systems and data acquisition systems which require the use of different sensors to continuously monitor the computer e.g. aviation systems, patient-monitoring systems in hospitals etc. Generally, code converters are circuits which accept input codes in one form and translate them to

present equivalent values in a different format as the output code. In designing such a circuit, the output codes required for each input code condition are first listed. Each output line is thereafter treated separately and then a k-map is drawn to describe the required output condition, for each input combination. The don't care states are entered for each unused input combination [28].

The k-map is a matrix-like diagram which consists of cells or boxes such that the whole of the Boolean universe is represented. In the case in which there is a single variable, the k-map is split into two equal halves. While one half represents the condition of the variable being true (i.e. having a value 1), the other half represents the condition of the variable being false (i.e. having a value 0). When there are two variables, the diagram is split along its other axis thus forming four cells. In general, when there are n variables, a k-map lists all the 2^n different product terms which can be formed in exactly n variables. Given a function of n variables, a product term in exactly these variables is referred to as a minterm. That is, a minterm refers to each combination of variables in a sum-of-products function. A combination of variables in a product-of-sums form is called a maxterm [17, 21].

In general, an n-variable k-map has 2^n cells where each cell represents a single minterm. The minterm in each cell is the product of the variables listed at the abscissa of the cell (i.e. the "header row" of the map). Thus for instance, where there are two variables A, B, there exists 4 different minterms namely A, $A^/$, B, $B^/$ where $A^/$, $B^/$ refer to the complements of A, B respectively (for example, if A = 1, $A^/$ = 0). This is described in Table 3.1.

Table 3.1: K-Map for Two Variables A and B

	$A^/$	A
$B^/$	$A^/B^/$	$AB^/$
B	$A^/B$	AB

For the cell described by $A^/B^/$, for instance, the abscissa is $A^/$ while the ordinate is $B^/$. Some designers however prefer to make the abscissa to be the header column of the map and the ordinate the header row. There is nothing wrong with this provided consistency and uniformity is maintained in drawing the maps. For the purpose of this paper, the structure in Table 3.1 is used. For each minterm which leads to a 1 output say, a k-map is filled in by placing 1s in the respectively cells. Outputs which are not specified in a given problem (called don't care outputs) are taken note of by marking an "X" in the affected cells. A basic characteristic of a k-map is that each cell differs from its adjacent cell by having exactly one variable complemented in the minterm in one cell which is not complemented in the minterm in the adjacent cell. In an n-variable map in general, there are n minterms adjacent to a given minterm.

In minimizing Boolean algebraic expressions, a set of exactly 2^m ($m < n$) adjacent cells containing 1s is called a subcube. In a subcube of two cells, a single variable differs in the cells. As such the two minterms in the cells can be described by a product term in the variables which don't differ. In an n-variable map, a subcube with 2^m cells has n – m (i.e. n minus m) variables which are the same in all the minterms and m variable which take all possible combinations of being complemented or not complemented. The set of minterms of Boolean expression doesn't necessarily form a single subcube. The largest subcube which can be found around a given minterm is sometimes referred to as a maximal subcube. If no cell in a maximal subcube intersects with another maximal subcube, then the product terms corresponding to the maximal subcubes (sometimes called prime implicants) are selected and the sum of these terms form a minimal sum-of-products expression. On the other hand, if all cells in one maximal subcube intersect with other maximal subcubes, then many cases can arise in describing the minimal expressions. A simple rule however is to note that each cell containing a 1 must necessarily be contained in some subcube which is selected.

There are basically two categories of converters. A digital to analog converter (DAC) converts binary output signals from computers to analog signals which can be used to position mechanical and other devices. Analog-to-digital converters (ADC), on the other hand, enable mechanical displacements to be converted to a digital representation. These converters also permit the conversion of an electric analog signal to digital-coded signals [17]. (An analog multiplexer (AMUX) is a device which is commonly incorporated in these converters). An example of an ADC is the shaft encoder which directly converts a physical position to a digital value. This encoder is connected to a rotating shaft, to which a coded-segment disk (which can rotate) is usually coupled. It then reads out the angular position of the shaft in digital form. The fastest ADCs are called flash (or simultaneous) converters. A counter ADC is a non flash ADC which includes a binary counter and a comparator. A tracking ADC follows the analog input up and down continuously and gives a continuous output of its value. A particularly common type of this converter is the successive approximation ADC. This ADC is similar to the counter ADC but has control logic instead of a counter logic.

The procedure for designing code converters using the data compression algorithm described in Section 2 is as follows:

Procedure CONVERTER DESIGN

1. Start;
2. Draw a schematic block diagram of the code converter showing the set of input codes and the set of output codes;

3. Prepare a table showing the output codes required for each input code condition;
4. Prepare the template for n converter map tables, where n is the blocklength of the set of input codes and the set of output codes;
5. List the zoned set in the outermost row (i.e. abscissa) of each of the map tables such that each zoned code occupies a distinct column;
6. List the numer set in the outermost column (i.e. ordinate) of each of the map tables such that each numer set occupies a distinct row;
7. Mark a don't care state in each of the maps;
8. Stop.

Fig. 3.1 is the schematic diagram of an IPC to ASCII converter where $I_1I_2...I_n$ represents a typical bit pattern of IPC and A_1A_2…a typical bit pattern of ASCII. In Table 3.2, the output codes required for each input code condition with respect to the two codes are described. Table 3.3((a) to (g)) indicate the converter maps for the IPC to ASCII conversion using the Type II definition of zoned portion. A don't care state in these maps is indicated by an "X" in the appropriate cell. In each of the maps in Table 3.3((a) to (g)), the zoned set (set of the most significant bits) is listed in the abscissa (i.e. the outermost row) while the numer set is listed in the ordinate (i.e. the outermost column). That is, $I_1I_2I_3$ describes a bit pattern on the horizontal position while $I_4I_5I_6I_7$ describes the bit pattern in the vertical position. For instance, with respect to the second row and second column, $I_1I_2I_3$ is '001' and $I_4I_5I_6I_7$ is '0001' and so the bit pattern of the cell is 0010001.

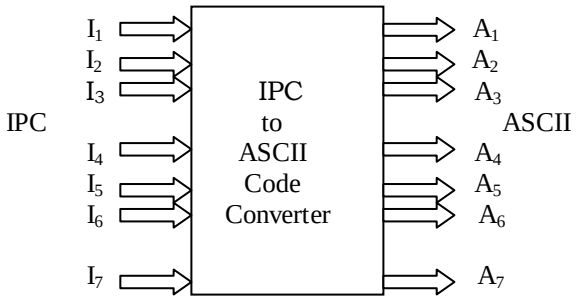

Fig. 3.1: Schematic diagram of an IPC to ASCII Code Converter

Table 3.2: The Output Codes (ASCII) Required for Each Input Code (IPC) Condition with respect to the Numerics

	INPUT (IPC)	**OUTPUT** (ASCII)
Numeric	$I_1I_2I_3I_4I_5I_6I_7$	$A_1A_2A_3A_4A_5A_6A_7$
0	0 0 0 0 0 0 0 0	0 1 1 0 0 0 0
1	0 0 0 0 0 0 0 1	0 1 1 0 0 0 1
2	0 0 0 0 0 0 1 0	0 1 1 0 0 1 0
3	0 0 0 0 0 0 1 1	0 1 1 0 0 1 1
4	0 0 0 0 0 1 0 0	0 1 1 0 1 0 0
5	0 0 0 0 0 1 0 1	0 1 1 0 1 0 1
6	0 0 0 0 0 1 1 0	0 1 1 0 1 1 0
7	0 0 0 0 0 1 1 1	0 1 1 0 1 1 1
8	0 0 0 0 1 0 0 0	0 1 1 1 0 0 0
9	0 0 0 0 1 0 0 1	0 1 1 1 0 0 1

Table 3.3(a): Converter Map for the IPC – ASCII Conversion

$I_1I_2I_3$ / $I_4I_5I_6I_7$	000	001	010	011	100	101	110	111
0000	0	X	X	X	X	X	X	X
0001	0	X	X	X	X	X	X	X
0010	0	X	X	X	X	X	X	X
0011	0	X	X	X	X	X	X	X
0100	0	X	X	X	X	X	X	X
0101	0	X	X	X	X	X	X	X
0110	0	X	X	X	X	X	X	X
0111	0	X	X	X	X	X	X	X
1000	0	X	X	X	X	X	X	X
1001	0	X	X	X	X	X	X	X

Table 3.3(b): Converter Map for the IPC – ASCII Conversion

$I_1I_2I_3$ / $I_4I_5I_6I_7$	000	001	010	011	100	101	110	111
0000	1	X	X	X	X	X	X	X
0001	1	X	X	X	X	X	X	X
0010	1	X	X	X	X	X	X	X
0011	1	X	X	X	X	X	X	X
0100	1	X	X	X	X	X	X	X
0101	1	X	X	X	X	X	X	X
0110	1	X	X	X	X	X	X	X
0111	1	X	X	X	X	X	X	X
1000	1	X	X	X	X	X	X	X
1001	1	X	X	X	X	X	X	X

Table 3.3(c): Converter Map for the IPC - ASCII Conversion

$I_1I_2I_3$ / $I_4I_5I_6I_7$	000	001	010	011	100	101	110	111
0000	1	X	X	X	X	X	X	X
0001	1	X	X	X	X	X	X	X
0010	1	X	X	X	X	X	X	X
0011	1	X	X	X	X	X	X	X
0100	1	X	X	X	X	X	X	X
0101	1	X	X	X	X	X	X	X
0110	1	X	X	X	X	X	X	X
0111	1	X	X	X	X	X	X	X
1000	1	X	X	X	X	X	X	X
1001	1	X	X	X	X	X	X	X

Table 3.3(d): Converter Map for the IPC – ASCII Conversion

$I_1I_2I_3$/ $I_4I_5I_6I_7$	000	001	010	011	100	101	110	111
0000	0	X	X	X	X	X	X	X
0001	0	X	X	X	X	X	X	X
0010	0	X	X	X	X	X	X	X
0011	0	X	X	X	X	X	X	X
0100	0	X	X	X	X	X	X	X
0101	0	X	X	X	X	X	X	X
0110	0	X	X	X	X	X	X	X
0111	0	X	X	X	X	X	X	X
1000	1	X	X	X	X	X	X	X
1001	1	X	X	X	X	X	X	X

Table 3.3(e): Converter Map for the IPC - ASCII Conversion

$I_1I_2I_3$/ $I_4I_5I_6I_7$	000	001	010	011	100	101	110	111
0000	0	X	X	X	X	X	X	X
0001	0	X	X	X	X	X	X	X
0010	0	X	X	X	X	X	X	X
0011	0	X	X	X	X	X	X	X
0100	1	X	X	X	X	X	X	X
0101	1	X	X	X	X	X	X	X
0110	1	X	X	X	X	X	X	X
0111	1	X	X	X	X	X	X	X
1000	0	X	X	X	X	X	X	X
1001	0	X	X	X	X	X	X	X

Table 3.3(f): Converter Map for the IPC – ASCII Conversion

$I_1I_2I_3$ / $I_4I_5I_6I_7$	000	001	010	011	100	101	110	111
0000	0	X	X	X	X	X	X	X
0001	0	X	X	X	X	X	X	X
0010	1	X	X	X	X	X	X	X
0011	1	X	X	X	X	X	X	X
0100	0	X	X	X	X	X	X	X
0101	0	X	X	X	X	X	X	X
0110	1	X	X	X	X	X	X	X
0111	1	X	X	X	X	X	X	X
1000	0	X	X	X	X	X	X	X
1001	0	X	X	X	X	X	X	X

Table 3.3(g): Converter Map for the IPC – ASCII Conversion

$I_1I_2I_3$ / $I_4I_5I_6I_7$	000	001	010	011	100	101	110	111
0000	0	X	X	X	X	X	X	X
0001	1	X	X	X	X	X	X	X
0010	0	X	X	X	X	X	X	X
0011	1	X	X	X	X	X	X	X
0100	0	X	X	X	X	X	X	X
0101	1	X	X	X	X	X	X	X
0110	0	X	X	X	X	X	X	X
0111	1	X	X	X	X	X	X	X
1000	0	X	X	X	X	X	X	X
1001	1	X	X	X	X	X	X	X

4. DISCUSSION

The binary text data compression algorithm used in the paper is a generalization of a storage-based built-in test pattern generation method described in [14] for reducing storage requirements and test application time in circuits. When compared to the 5-bit test set T = {00000, 00111,01000,01110,10110,10111} used as illustration in the above reference, the sets T_1 and T_2 are the zoned set and numer set of T respectively, based on the Type II definition of zoned portion.

As indicated in Table 2.1, the numerics of both the IPC and ASCII have two equizones with respect to the Type I definition of zoned portion, and one equizone with respect to the Type II definition. However, the numerics of both codes do not have the same code presentation.

The k-map may be viewed as a form of matrix representation of codes/coded character sets (i.e. sets of products of Boolean variables) of blocklength n, where n is the number of variables present. There are a total of seven converter maps for the IPC to ASCII conversion (which

equals the block length of each of the two codes). Based on the way k-maps are drawn in this paper, each of the abscissa of a cell (i.e. bit patterns in the outermost row) is a zoned portion while each of the ordinate of a cell (bit patterns in the outermost column) is a numeric portion. Also each internal column of a k-map is an equizone. Thus, the first equizone in Table 3.3((a) to (g)) has the zoned portion '000'. In Table 3.3((a) to (g)), all the cells except those in the first equizone have don't care states. In the case of Table 3.3(a) and Table 3.3(g), there exists no group of two adjacent ones in their cells. The converter maps of Table 3.3(b) and Table 3.3(c) are identical. There are nine groups of two adjacent ones in these two maps. There exists one group of two adjacent ones in Table 3.3(d) while there are three of such groups in Table 3.3(e). There are two groups of two adjacent ones in Table 3.3(f).

Apart from using k-maps, the design of an IPC to ASCII code converter can be realized using a programmable logic array in which the minimization of the Boolean expressions is accomplished via the well-known Quine-McCluskey algorithm [29]. In justifying the suitability of IPC for online computer systems, it has been argued that the amount of information to be converted to maintain complete compatibility between IPC and ASCII represents a very small percentage of the information to be processed in an online computer system [11].

5. CONCLUSION

In this paper, a comparative discourse is presented on the binary text compression properties of two distinct binary texts which can be modeled respectively as the numerics of a 7-bit subset of the Information Processing Code (IPC) and the numerics of the 7-bit American Standard Code for Information Interchange (ASCII). These properties are then related to the design of an IPC to ASCII code converter using the well-known k-map method. By virtue of the fact that their blocklength is less than the blocklength of Unicode, both the 7-bit IPC and 7-bit ASCII are subsets of Unicode. For each of the seven converter maps constructed in this paper, the output expressed as the simplest possible combination of input variables can easily be calculated as Boolean expression. Thus, an efficient IPC (or a code structurally identical to it) to ASCII code converter can be created by implementing the expressions using logic gates, just as an Excess-3 to BCD code converter was implemented using 13 gates [5]. In general, the idea expounded in this paper may also be applied to the design of decoders and synchronous counters.

REFERENCES

[1] Awajan, Arafat: "Multilayer Model for Arabic Text Compression", ***The International Arab Journal of Information Technology***, Vol. 8, No. 2 (2011), 188-196.

[2] Diri, B.: "A Text Compression System Based on the Morphology of Turkish Language", ***Proceedings of the 15th International Symposium on Computer and Information Sciences***, Turkey (2000), 156-159.

[3] Ghwanmeh, S., Al-Shalabi, R., and Kanaan, G.: "Efficient Data Compression Scheme Using Dynamic Huffman Code Applied on Arabic", ***Journal of Computer Science*** Vol.2, No. 12 (2006), 887-890.

[4] Grailu, H.: "Farsi and Arabic Document Images Lossy Compression Based on the Mixed Raster Content Model", ***International Journal on Document Analysis and Recognition*** Vol. 12, No. 4 (2009), 227-248.

[5] Oluwade, 'D.: "A Binary Text Compression Algorithm based on Partitioning", **Advances *in Computer Science & Engineering***, Vol. 3, No. 2 (2009), 165 – 174.

[6] Teahan, J; McNab, R., and Witten, H.: "A Compression-based Algorithm for Chinese Word Segmentation", ***Computer Journal of Computational Linguistics***, Vol. 26, No. 3 (2000), 375 - 392.

[7] Wiseman, Y.: "Conjugation-Based Compression for Hebrew Texts", ***ACM Transactions on Asian Language Information Processing*** Vol. 6, No. 1 (2007), 4-es.

[8] Yoshida, S., Morihara, T., Yahagi, H., and Satoh, N.: "Application of a Word-based Text Compression Method to Japanese and Chinese Texts", ***Proceedings of Data Compression Conference***, (1999), 561.

[9] Hwang, Wen-Jyi, Chine, Ching-Fung and Li, Kuo-Jung: "Scalable Medical Data Compression and Transmission Using Wavelet Transform for Telemedicine Applications", ***IEEE Transactions on Information Technology in Biomedicine*** Vol. 7, No. 1 March (2003), 54-63.

[10] Herstein, I.: ***Topics in Algebra***, John Wiley & Sons, New York (1975)

[11] Morenoff, E., and McLean, J.B.: "A Code for Non-numeric Information Processing Applications in Online Systems", ***Communications of the ACM,*** Vol. 10, No. 1 (1967), 19 – 22.

[12] Bannister, B. R. and Whitehead, D.G.: ***Fundamentals of Modern Digital Systems***, Macmillan Education Ltd, Basingstoke, Hampshire (1999).

[13] Oluwade, 'D.: ***Design and Analysis of***

Computer-Coded Character Sets; PhD thesis, Department of Computer Science, University of Ibadan (2004).

[14] Pomeranz, I., and Reddy, S.M.: "A Storage-Based Built-in Test Pattern Generation Method For Scan Circuits based on Partitioning and Reduction of a Precomputed Test Set", ***IEEE Transactions on Computers***, Vol. 51, No. 11 (2002), 1282-1293.

[15] Mackenzie, C.E.; ***Coded Character Sets, History and Development*** *(The Systems Programming Series)* ; Addison Wesley Publishing, Philippines (1980).

[16] Oluwade, 'D.: "Application of a Data Compression Technique to the American Standard Code for Information Interchange (ASCII)", ***International Journal of Information Science & Computer Mathematics***, Vol. 1, No. 1 (2010), 1-7.

[17] Bartee, T.C.: ***Digital Computer Fundamentals***, McGraw-Hill Inc, Singapore, Sixth Edition (1985).

[18] Frim and Miller: "Here are more Digital Converters", ***Electronics Design***, Vol. 17, No. 25 (1969), 87.

[19] http://www.freepatentsonline.com/3706878.html <Retrieved in 2009>

[20] Ryder, John D.: ***Electronic Fundamentals and Applications (Integrated and Discrete Systems)***, Prentice-Hall of India Private Limited, New Delhi (1989).

[21] Holder, M.E.: "A Modified Karnaugh Map Technique"; ***IEEE Transactions on Education,*** Vol. 48, No. 1 (2005), 206 – 207.

[22] Kehinde, L.O., and Makinwa, K.A.: "An Alternative Procedure for Synchronous Counter Design using Karnaugh Maps", ***Ife Journal of Technology***, Vol. 5, No. 1 (1995), 53 – 57.

[23] Oluwade,'D.: "Applications of 2-code Error Detection Techniques", ***Proceedings of the 14th National Conference of COAN*** (Nigeria Computer Society), Vol. 9 (1998), 245 – 251.

[24] Oluwade, 'D.: "Algebraic Characteristics of the CCITT#2 Communication Code", **International Journal of Applied Mathematics & Statistics** (IJAMAS), Vol. 2, M04 (2004), 50-59.

[25] Oluwade, 'D.: "Design of a Byte-Based Coded Character Set", ***The Journal of Computer Science and Its Application***, Vol.18, No.2 (2011), 42-54.

[26] Lucchini, A.: "On the Number of Generators of Finite Images of Free Products of Finite Groups", ***Journal of Algebra***, Vol. 245 (2001), 552 – 561.

[27] Magnus, W., Karrass, A., and Solitar, D.: ***Combinatorial Group Theory (Presentations of Groups in Terms of Generators and Relations)***, Dover Publications Inc., New York (1976).

[28] Sayers, I. L., Robson, A. P., Adams, A.E., and Chester, E.G.: ***Principles of Microprocessors***; CRC Press Inc., Boca Raton, Florida (1991).

[29] Nelson, V.P.: ***Digital Circuit Analysis and Design***, Prentice-Hall, New York (1995).