ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 4, April 2018

Through Packed Bcd Achieving Memory Utilization Efficiently

[1] Dr.R.Viswanathan, [2]Chandrasegar.T, [3]P.Manjula
[1]Associate Professor, [2][3] Assistant Professor
[1]Galgotias University, [2][3] VIT University

Abstract: In the recent world most of the projects are involved with reducing the space complexity of the system. When the memory is utilized efficiently then the outcome leads to less space. In this paper we proposed a new packed BCD algorithm to manage the memory efficiently. Through this proposed algorithm around 50% memory wastage can be reduced and we can maximize the utilization of memory just by reducing the space complexity. We intended to achieve memory utilization efficiently using packed BCD.

Keywords□ Memory Utilization, Packed BCD, Uncompressed BCD, and Data compression.

I. INTRODUCTION

In the recent world most of the projects are involved with reducing the space complexity of the system. Binary-Coded decimal (BCD) which encrypts the digits 0 through 9 by representing 4-bit unsigned binary. Through this algorithm we are intended to achieve memory utilization by using Packed BCD.BCD is broadly divided into two categories, Packed and unpacked BCD. In unpacked BCD only one value can be inserted in a single byte. The digit is stored in the least significant 4 bit and the most significant 4 bits are not relevant to the value of being inserted. While in compressed BCD two values can be inserted into a single byte.

E.g.: In uncompressed BCD a number 91 can be stored as follows:

Decimal: 9	1
Binary: 0000 1001	0000 0001
Binary: 1001	0001

We can reduce the above memory wastage by using packed BCD.
0000 1001 0000 0001(Unpacked BCD) =1001 0001(Packed BCD)
The above example shows that how 2-byte unpacked BCD number is packed into a single byte by creating a packed BCD number. And so one reason to use packed BCD is that it is twice as efficient in storing data.

II. LITERATURE REVIEW

Lung-Jen Lee [1] presents an idea about a new pattern run-length compression Method is given whose decompress or is simple and easy to implement. The analytical results show that it can achieve an average compression Ratio of 67.64%.The run-length-based compression Algorithm encodes repeated pattern runs. It encodes2|n| runs of compatible patterns. Christian Pataumer [2] presents a compression system optimized for the reduction of data using pulse digitizing electronics. Such systems are widely used in High Energy Physics experiments.

HaroonAltarawneh [3] described the different methods of data compression algorithms on English text files such as LZW, Huffman and Fixed-length code (FLC).The important principle of data compression algorithms on text Files are to transform a string of characters into a new string which contains the same information.

Wang Lei [4] proposed a new distributed algorithm of data compression based on hierarchical cluster model for sensor networks. The result of the above new algorithm has got good performance of approximation can compress data and also reduce the amount of data efficiently. Jacob Ziv [5] considered the case where consecutive blocks of N letters of an individual sequence X over a finite-alphabet are being compressed into their binary sequences. Here we have discussed the best possible compression that may be achieved by any universal data compression algorithm for finite N-blocks.

Kedarnath J. Balakrishnan [6] discussed relationship between Entropy and Data Compression. The entropy is the measurement of set of amount of data whose information contained in it. In this paper we also extended the concept of entropy for incompletely specified test data. It has also been described the impact of partitioning the test data into symbols on entropy.

Now a days Storing data in compressed form is becoming Common in high-performance systems. In this paper it has been suggested a hardware-assisted data compression as a tool for reducing energy consumption of processor-based systems. In this proposed algorithm we have also described in details the architecture of the compression/decompression unit presented in H Allen [7]. Luca Benini [8] represented how large numbers perform basic arithmetic operations. The best way of representing the large number is through the analysis of three types of representation of numbers is through binary-coded decimal, binary, packed binary-coded decimal. The conclusion has been made on the basis of the analysis that which number's representation is possible.

Maxime Crochemore [9] proposed a new text-compression scheme is given on the basis of forbidden Words "ant dictionary". We have also shown that this algorithm attain the Entropy for balanced binary sources. One of the main advantages of this approach is that it produces very fast decompresses .All the methods used in this paper are from Theory of computation.

Tenkasi V. Ramabadran [10] explained the scheme for compressing computer data by treating them as sequences of bytes. An alphabet reduction Technique which permits handling of each bit within a byte separately is also introduced. The scheme allows the complexity of the Source model, and thus compress the performance.

S. Rahil Hussian et al. [12] presented the reversible implementation of DPD (Densely packed Decimal) converter to and from conventional BCD format. Conversion is smeared to the adder circuits everywhere they follow BCD code for the arithmetic addition such that converting them to DPD. It will result in the better storage capacity by decreasing the less density of storage devices for faster access to memory.

Er.Aradhana Raju et al. [13] illustrate the implementation of the Densely Packed Decimal Encoding, projected by M. F. Cowlishaw and simulate using available software platforms. Then transmit the compressed data by using secure communication technique.

M.Cowlihaw [14] presented a lossless compression which converts three binary coded decimal (BCD) digits to 10 bits using an algorithm. Here simple Boolean operations and functions is used in reversed BCD. Technique is not restricted to multiples of three digits. In new system, use any length of string efficiently while keeping decimal digit boundaries accessible.

J.H.M. Bonten [15] describes a method of encoding decimal numbers is known as Packed Decimal Encoding. This technique is proposed to freeze out all vacant space in the set of available bit-patterns. Normally this leftover occurs when decimal digits are stored. Packed Decimal Encoding method relies on a technique for compressing decimals called Densely Packed Decimal (DPD).

Hafiz Md et al. [16] describes a method for the reversible circuit of binary coded decimal (BCD) adder. Proposed circuit has the capacity to add two 4-bits binary variables and it converts the addition into the appropriate BCD number with efficient error correcting modules where the operations are reversible.

H. Che et al. [17] describes a dynamic range encoding scheme (DRES) to considerably increase the TCAM storage efficiency for range matching. DRES uses the TCAM coprocessor itself to support range encoding. It can be programmed in a network processor using a TCAM coprocessor for packet classification.

Y.-K. Chang, C.-C. Su [18] describes a set of encoding schemes based on Gray code. Encoding techniques are used to improve the existing elementary interval- based range encoding schemes. Experiment's results show that the proposed Gray code-based schemes consume less TCAM storage space than the existing schemes.

Stephen Hines [19] describes an architectural features and complier optimization technique target one, two or more design goals expense of the others. In this a novel architectural and complier approach used to escape power requirements, minimize code size, and improve performance by adding an IRF (instruction register file) into the architecture.

Yoshiyuki okada [20] describes a compression of lossless data commonly use on personal type of computers for increase storage capacity. When we take example, we can get double of the normal capacity by using lossless data compression strategies. In this research, it is very necessary to locate compressed data of variable length in a fixed –length block by little fragmentation as much as possible.

III. PROBLEM DEFINITION

In this paper we are trying to achieve memory utilization through packed BCD (Binary Coded Decimal) and implementing the same using c language along with file handling. In uncompressed BCD only one integer number can be stored in a single byte thus there was wastage of memory. Therefore packed BCD came into existence here; we can store two integer numbers in a single byte thus saving the memory space leading to memory utilization efficiently.

IV. SOLUTION METHODOLOGY

In this paper we proposed a new algorithm for achieving the objectives.

A. Algorithm

```
START
STEP 1 declare file pointers fp1 fp2
STEP 2 fp1:=open (file1)
        fp2:=open (file2)
STEP 3 if(!fp1 or !fp2)
        {//cannot open file...}
        else
STEP 4 read char a
        fp1=to_ascii(a)
        fp2:=to_hex (a)
         repeat STEP 4 untill  a!='.'
STEP 5 print from starting fp1, fp2
STEP 6 END
```

B. Coding

```
#include<stdio.h>
#include<conio.h>
int main()
{
        char a;
FILE *fp_dec;
        FILE *fp_hex;

        fp_dec=fopen ("project.txt","w+");
        fp_hex=fopen ("project1.txt","w+");
        if ((fp_dec==NULL) || (fp_hex==NULL))
        {
                puts ("/nError in memory");
                //exit ();
        }
        else
        {
                while (a! ='.')
                {
                        scanf ("%c",&a);
                        fprintf (fp_dec,"%d",a);
                        fprintf (fp_dec,"\t");
                        fprintf (fp_hex,"%x",a);
                        fprintf (fp_hex,"\t");
                }//end of while...
                rewind (fp_dec);
                rewind (fp_hex);
                printf ("\n\nThe ASCII representation of the string is...\n");
                while (1)
                {       a=fgetc(fp_dec);
                        if(a==EOF)
                        {
break;
}
printf("%c",a);}//end of while
printf("\n\nThe hexadecimal representation of the string is...\n");

while(1)
{
a=fgetc(fp_hex);
        if(a==EOF)
{
break;
}
 printf("%c",a);
}//end of while
    }//end of else
        getch();
  }// end of main
```

C. Packed BCD to ASCII Conversion

Above program is written in 8051 C for transform Packed BCD to ASCII and store bytes on C1 and C2

```
#include<reg51.h>
      void main (void)
{
unsigned char A, B;
unsigned char byte=0x29;
A=byte & 0x0F;
C1=A | 0x30;
B=byte & 0xF0;
B=B >> 4;
C2=B | 0x30;
}
```

D. ASCII to Packed BCD conversion:

8051 C program is written to convert ASCII digits of '3' and'5' to packed BCD and store bytes on A1.

```
        #include<reg51.h>
void main( void)
{
unsigned char byte;
unsigned char X='3';
unsigned char Y='5';
X=X & 0x0F;
X=X << 4;
Y=Y & 0x0F;
byte=X | Y;
A1=byte;
}
```

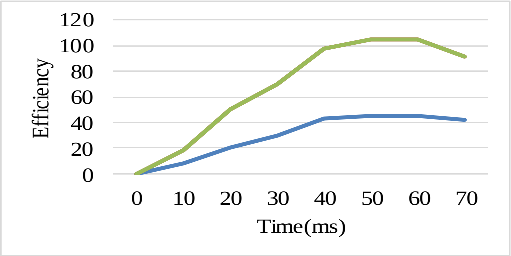
Table 1 shows that ASCII code for digits o through 9

Key	ASCII (Hex)	Binary	BCD (Unpacked)
0	30	011 0000	0000 0000
1	31	011 0001	0000 0001
2	32	011 0010	0000 0010
3	33	011 0011	0000 0011
4	34	011 0100	0000 0100
5	35	011 0101	0000 0101
6	36	011 0110	0000 0110
7	37	011 0111	0000 0111
8	38	011 1000	0000 1000
9	39	011 1001	0000 1001

Table 1 ASCII Code for digits 0-9

V. RESULT

Graph 1 shows that efficiency of modified packed BCD is superior than standard packed BCD in terms of time. Efficiency and time is measured in percentage and milli seconds. Blue line and Green line indicates standard packed BCD and modified packed BCD simultaneously.



Graph 1 Standard Packed BCD v/s Modified Packed BCD

VI. EXPERIMENT

Input From User: abcdefghijklmnopqrstuvwxyz
2345678910
ABCDEFGHIJKLMNOPQRSTUVWXYZ

97	98	99	100	101	102	103	104	105	106
107	108	109	110	111	112	113	114	115	116
117	118	119	120	121	122	10	49	50	51
52	53	54	55	56	57	49	48	10	65
66	67	68	69	70	71	72	73	74	75
76	77	78	79	80	81	82	83	84	85
86	87	88	89	90	46				

Table 2 ASCII representation of the string

61	62	63	64	65	66	67	68	69	6a
6b	6c	6d	6e	6f	70	71	72	73	74
75	76	77	78	79	7a	a	31	32	33
34	35	36	37	38	39	31	30	a	41
42	43	44	45	46	47	48	49	4a	4b
4c	4d	4e	4f	50	51	52	53	54	55
56	57	58	59	5a	2e				

Table 3 Hexadecimal representation of the string

CONCLUSION

In this paper we are emphasizing mainly on minimizing the memory wastage. From this paper we came to many results among all we are giving much importance to memory utilization. Based on the entropy analysis of the file we can increase the compression ratio which is further matter of analysis. We came to few results using this algorithm although it needed to be further implemented in future.

REFERENCES

[1] Lung-Jen Lee, “2n Pattern Run-Length for Test Data Compression", IEEE Transactions on Computer-Aided Design Of Integrated Circuits and Systems, Vol. 31, No. 4, April 2012.

[2] Christian Patauner, "A Lossless Data Compression System for A Real-Time Application in HEP Data Acquisition", IEEE Transactions on Nuclear Science, Vol. 58, No. 4, August 2011.

[3] Haroon Altarawneh, " Data Compression Techniques on Text Files: A Comparison Study", International Journal of Computer Applications (0975 – 8887), Volume 26– No.5, July 2011.

[4] Wang Lei, “Data Compression Algorithm Based On Hierarchical Cluster Model for Sensor Networks”, International Journal of Advanced Science and Technology Vol. 2, January, 2009.

[5] Jacob Ziv, “On Finite Memory Universal Data Compression and Classification of Individual Sequences", IEEE Transactions on Information Theory, Vol. 54, No. 4, April 2008.

[6] Kedarnath J. Balakrishnan, “Relationship between Entropy and Test Data Compression", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 26, No. 2, February 2007.

[7] H. Allen, “Representation of the Large Numbers with accounting Of Microprocessor Capacity", 6'b International Siberian Workshop and Tutorials Edm'2005, Session V, July 1-5, Erlagol.

[8] Luca Benini, "Memory Energy Minimization by Data Compression:Algorithms, Architectures and Implementation", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 12, No. 3, March 2004.

[9] Maxime Crochemore, "Data Compression Using Antidictionaries“, Proceedings of the IEEE, Vol. 88, No. 11, November 2000.

[10] V.Tenkasi, Ramabadran, “An Adaptive Algorithm for the Compression of Computer Data", IEEE Transactions on Communications, Vol31. No. 4, April 1989.

[11] http://academic.evergreen.edu/projects/biophysics/technotes/program/bcd.htm.

[12] S. Rahil Hussian et al., “VLSI Implementation of Densely Packed Decimal Converter to and from Binary Coded Decimal using Reversible Logic Gates”, International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622.

[13] Er.Aradhana Raju , Purabi Mahato , Ritto K. Babu , Richi Patnaik, “Secure Communication With Lossless Data Compression Using Dpd Encoding”, IJMER, ISSN: 2249–6645, Vol. 6, Iss. 5, May 2016.

[14] M.Cowlihaw, “Densely packed decimal encoding, Computers and Digital Techniques”, IEEE 2002.

[15] “Packed decimal encoding” IEEE-754-2008 by: J.H.M. Bonten.

[16] Hafiz Md., H. Babu and A. R. Chowdhury, "Design of a Reversible Binary Coded Decimal Adder by Using Reversible. 4-bit Parallel Adder"', 18th Int. Conf. VLSI Design, pp. 255-260, 2005.

[17] H. Che, Z. Wang, K. Zheng, B. Liu, "Dres: Dynamic range encoding scheme for tcam coprocessors", IEEE Transactions on Computers, pp. 902-915, 2008.

[18] Y.-K. Chang, C.-C. Su, "Efficient team encoding schemes for packet classification using gray code", GLOBECOM'07, 2007.

[19] Stephen Hines, “Improving program efficiency by packing instruction into registers”, International symposium on computer architecture, 2005.

[20] Yoshiyuki okada, ”Effective management of compressed data with Packed files System”, Data compression conference, 1997.