ISSN:0975-9646

Efficient String Matching Using Bit Parallelism

Kapil Kumar Soni, Rohit Vyas, Dr. Vivek Sharma

TIT College, Bhopal, Madhya Pradesh, India

Abstract: Bit parallelism is an inherent property of computer to perform bitwise a parallel operation on computer word, but it is performed only on data available in single computer word. Bit parallelism inherently favors parallelism of bit operations within computer word. Parallel computing comprises bit parallelism and analyzed that it can be carried out "in parallel" which ensures utilizing the word size of computer. This technique is being properly utilized to work out on various string matching problems for increasing the efficiency of various real world applications. Since 1992 bit parallelism is being used in string matching applications to improve the matching pace. There are a variety of important bit parallel string matching algorithms exist like Shift-OR, BNDM, TNDM, SBNDM, BNDMq, Shift-OR with Q-Gram, and Multiple Patterns BNDM. This paper discusses the various important bit parallel string matching algorithms by means of example along with their advantages and disadvantages.

Keywords: String Matching, Bit Parallelism, Shift-OR, BNDM, TNDM, SBNDM, BNDMq, Shift-OR with Q-Gram, Multiple pattern BNDM.

I. Introduction

Bit parallelism [1] is a built in quality of computer in which basic bit wise operations like AND, OR, NOR etc, are performed parallel within the computer word in the single clock cycle. Equivalent to computer word there are basic registers available for holding a data unit in the computer. Figure 1 describes how the bit parallelism can be achieved in the computer.

With the use of bit parallelism in the field of String Matching, matching speed is improved up to certain level. But using such kinds of bit parallel based string matching, matching is quite difficult and having number of limitations [2]. These algorithms can be used in most of the real world applications [3] where string matching is required like as Intrusion Detection system, Plagiarism detection, Data Mining & Bioinformatics and etc.

Bit parallel string matching algorithms are faster than the other benchmark character matching based algorithms like as KMP [4], BM [5], BMH [6], BMHS [7], BMHS2 [8], BMI [9], Improved BMHS [10], Wu Manber [11] and Aho-Corasick [12] etc.

The Bit Parallel algorithms are based on the simulation of non-deterministic automata. It is simply the efficient simulation of non-deterministic automata. Figure 1 shows bit parallelism behaviour inside a computer.

Some of the important bit parallel string matching algorithms are Shift-OR [13], BNDM [14], TNDM [15], SBNDM [15], BNDMq [17], Shift-OR with Q-Gram [16], and Multiple pattern BNDM [18]. BNDM, TNDM, SBNDM and BNDMq are the single pattern string matching algorithms whereas Shift-OR, Shift-OR with Q-grams and Multiple Patterns BNDM are the multiple pattern string matching algorithms.

This paper presents the working of above mentioned bit parallel string matching algorithms with examples. Here we elaborate the advantages and disadvantages of these algorithm and gives the bit parallel string matching history.

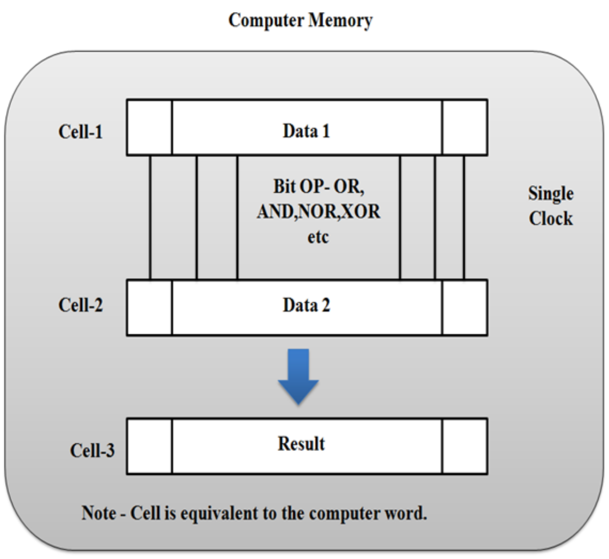
Computer Memory
Cell-1
Data 1
Bit OP- OR, AND,NOR,XOR etc
Single Clock
Cell-2
Data 2
Cell-3
Result
Note - Cell is equivalent to the computer word.

Figure 1: Bit Parallelism in Computer

II. History Of Bit Parallel String Matching

In 1992 the Baeza–Yates and Gonnet gives the first bit parallel string matching algorithm named Shift- OR algorithm [13]. It was approximate multiple patterns string matching algorithm.

In 1998, based on Shift-OR algorithm Navarro and Raffinot proposed a new algorithm named as BNDM (Backward Non Deterministic Matching) [14]. BNDM is exact single pattern string matching algorithm. In BNDM algorithm, AND or SHIFT operation is used. BNDM set the benchmark in the string matching algorithm.

After BNDM in 2003 Peltola and Tarhio presented an improved version of BNDM named TNDM (Two Way Non Deterministic Matching) [15]. Except mismatch at last position this algorithm is same as BNDM algorithm. In 2003 another algorithm Simplified BNDM also known as SBNDM [15] was suggested. It was an enhanced version of BNDM. Here we do not require finding the longest prefix which will give the better performance in some cases.

In 2006 an enhanced edition of Shift-OR algorithm was launched by Salmela, Tarhio and Kytojoki known as Shift OR with Q-Gram [16]. It is also an approximate multiple string matching algorithm. It considers Q character at a time for scanning.

In 2009 Branislav Durian, Jan Holub, Hannu Peltola and Jorma Tarhio presented the concept of Q-gram in BNDM algorithm known as BNDMq [17]. It scans the Q characters at each arrangement. In 2010 Changsheng Miao, Guiran Chang and Xingwei combine the concept of Q-gram with BNDM and implemented Multiple Pattern BNDM Algorithm [18]. Figure 2 shows the timeline history of bit parallel string matching algorithms.

III. Bit Parallel String Matching Algroithms

A. Shift-OR Algorithm:

Shift-OR [13] Algorithm gives the approximate matching results it means algorithm matching results may contain errors. In Shift OR algorithm searching is done from left to right. But this is designed for working on patterns of equal length plus the length less than or equal to the size of a computer word.

Maximum multiple pattern algorithms are based on the automata theory. They build automata of the patterns in pre-processing phase. When the pattern set size is increased then the size of automata also increases which is difficult to maintain. Shift-OR algorithm is a replication of a nondeterministic automata. It is suitable to be implemented in hardware. Shift OR consist of two phases pre-processing and searching.

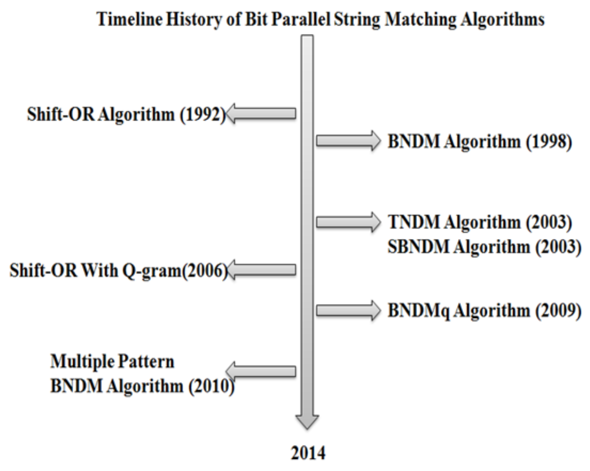
Timeline History of Bit Parallel String Matching Algorithms
Shift-OR Algorithm (1992)
BNDM Algorithm (1998)
TNDM Algorithm (2003)
SBNDM Algorithm (2003)
Shift-OR With Q-gram(2006)
BNDMq Algorithm (2009)
Multiple Pattern BNDM Algorithm (2010)
2014

Figure 2: Bit Parallel String Matching Timeline History

In pre-processing phase we find out the bit vector of the every character which is possibly exists in the text. In this ith corresponding bit is zero the same character appears at position 'i' otherwise place 1 at ith position and write this in reverse order. In searching phase automaton has a transition from state 'i' to 'i+1' on the basis of input character say 'c'. If 'ith' bit in B[c] is 0, and in state vector D where 'ith' bit is 0 indicates that 0 occurs at MSB so we find the pattern at that particular position. In Shift OR algorithm pre-processing and searching both are simple. Here only Shift and AND are used and no buffering is required. This algorithm is working on the patterns whose lengths are equal. Let's take an example to understand Shift OR algorithm.

Suppose King, thin and Apps be the patterns of length 4 and "DeepthinKing" be the text of length 12.

Bit Vector: B [K] = 1110, B [i] = 1001, B [n] = 0011, B [g] = 0111, B [t] = 1110, B [h] = 1101, B [a] = 1110, B [p] = 1001, B [s] = 0111 and for others 1111. Table 1 shows the algorithm processing for this example. And the initial Bit vector D is initialized to all 1's.

D = 1111, pos = 1 & D = D<<1.

Table 1 : Shift-OR Algorithm Processing Example

S.NO.	Matching Process	Comments

1	Scan "D"
	D = (1110) OR
	(B[D]=1111)
	D= 1111
		Remain in the same state because MSB is not zero. D contains All 1's.

2	Scan "e"
	pos = pos+1=2
	D= D<<1
	D=(1110) OR
	(B[e]=1111)
	D=1111
		Remain in the same state because MSB is not zero. D contains All 1's.

3	Scan "e"
	pos = pos+1=3
	D= D<<1
	D=(1110) OR
	(B[e]=1111)
	D=1111
		Remain in the same state because MSB is not zero. D contains All 1's.

4	Scan "p"
	pos = pos+1=4
	D= D<<1
	D=(1110) OR
	(B[p]=1001)
	D=1111
		Remain in the same state because MSB is not zero. D contains All 1's.

5	Scan "t"
	pos = pos+1=5
	D= D<<1
	D=(1110) OR
	(B[t]=1110)
	D=1110
		Move to the next state because D not contains All 1's.

6	Scan "h"
	pos = pos+1=6
	D= D<<1
	D=(1100) OR
	(B[h]=1101)
	D=1101
		Move to the next state because D not contains All 1's.

7	Scan "i"
	pos = pos+1=7
	D= D<<1
	D=(1010) OR
	(B[i]=1001)
	D=1011
		Move to the next state because D not contains All 1's.

8	Scan "n"
	pos = pos+1=8
	D= D<<1
	D=(0110) OR
	(B[n]=0011)
	D=0111
		MSB = 0 means patterns found at pos-m+1 i.e. 8-4-1=5.

9	Scan "K"
	pos = pos+1=9
	D= D<<1
	D=(1110) OR
	(B[K]=1110)
	D=1110
		Move to the next state because D not contains All 1's.

10	Scan "i"
	pos = pos+1=10
	D= D<<1
	D=(1100) OR
	(B[i]=1001)
	D=1110
		Move to the next state because D not contains All 1's.

11	Scan "n"
	pos = pos+1=11
	D= D<<1
	D=(1010) OR
	(B[n]=0011)
	D=1011
		Move to the next state because D not contains All 1's.

12	Scan "g"
	pos = pos+1=12
	D= D<<1
	D=(0110) OR
	(B[K]=0111)
	D=0111
		MSB = 0 means patterns found at pos-m+1 i.e. 8-4-1=5.

B. BNDM Algorithm:

BNDM i.e. Backward Non Deterministic Matching [14] is single pattern exact string matching algorithm. In BNDM we scan the text from right to left respective of pattern. BNDM matching concept is based on shift OR Algorithm and suffix automata from BDM (Backward Deterministic Matching) Algorithm. This algorithm is simulation of BDM algorithm with the use of bit parallelism. BNDM simulates the non-deterministic version using bit parallelism.

BNDM comprises of two phases pre-processing and searching. In pre-processing phase we find Bit Vectors of each character that will possibly come in text. This is done by using the pattern characters. Bit Vectors are to be calculated by putting 1 for occurrence and 0 for non-occurrence of the character and we first take bit vector D with initial value i.e. all one. Here bit vector size is less than or equal to the word length of the computer. While searching, the pattern is searched with the help AND & SHIFT operations. Pattern is searched when MSB of D is 1 and value of character count pointer is 0.

For example Let Text T = ‘DeepthinKing' and Pattern P = ‘King'. The Bit vectors of the characters are B [k] = 1000, B [i] = 0100, B [n] = 0010, B [g] = 0001 and other = 0000. Initial Value of D = 1111, pos = 0, j = 4 and last = 4. Table 2 shows the BNDM algorithm processing example.

Table 2 : BNDM Algorithm Processing Example

S.No.	**Matching Process**	**Comments**

1	Scan "p"
	D = D &B[Tpos + j]
	D= (D=1111)&(B[p]=0000)
	D=0000.
	pos = pos +last =4
	D=1111, j=4, last=4
		D is all zero means mismatch with zero suffix entry.

2	Scan "n"
	D = D &B[Tpos + j]
	D= (D=1111)&(B[n]=0010)
	D=0010.
	j = j-1= 3,
	D<<1 i.e. 0100
		Searching for suffix move right to left.

3	Scan "i"
	D = D &B[Tpos + j]
	D= (D=0100)&(B[i]=0100)
	D=0100.
	j=j-1=2.
	D<<1 i.e. 1000
		Searching for suffix move right to left.

4	Scan "h"
	D = D &B[Tpos + j]
	D= (D=1000)&(B[h]=0000)
	D=1000.
	pos = pos +last =8
	D=1111, j=4, last=4
		Mismatch at h. Shift by h.

5	Scan "g"
	D=1111 , pos=8, j=4 and last=4
	D = D &B[Tpos + j]
	D= (D=1111)&(B[g]=0001)
	D=0001
	j=j-1=3.
	D<<1 i.e. 0010.
		Searching for suffix move right to left.

6	Scan "n"
	D = D &B[Tpos + j]
	D= (D=0010)&(B[n]=0010)
	D=0010
	j=j-1=2.
	D<<1 i.e. 0100.
		Searching for suffix move right to left.

7	Scan "i"
	D = D &B[Tpos + j]
	D= (D=0100)&(B[i]=0100)
	D=0100
	j=j-1=1.
	D<<1 i.e. 1000.
		Searching for suffix move right to left.

8	Scan "K"
	D = D &B[Tpos + j]
	D= (D=1000)&(B[K]=1000)
	D=1000
	j=j-1=0.
		MSB having 1 means pattern found at position 5.

BNDM algorithm is faster than the previous algorithm Shift OR, BDM and other character based algorithms. It Occupies very less space and perform various operations in parallel. It is very simple and flexible algorithm. Here pattern is assume to the less than or equal to the word size of computer.

C. TNDM Algorithm:

TNDM i.e. Two Ways Non Deterministic Matching [15] is Single pattern exact string matching algorithm. It is similar to the BNDM algorithm, but having only a single change, if mismatch occur at last position instead of shifting like BNDM, TNDM go forward to find suffix of reverse pattern. Due to this sometimes number of examined characters is less than BNDM. Therefore matching is faster than the BNDM in such cases. Figure 3 shows the basic processing structure of TNDM algorithm.

D. SBNDM Algorithm:

SBNDM [15] i.e. Simple Backward Non Deterministic Matching is same as BNDM algorithm. SBNDM is having some changes, for calculating the shifts. Because of that mechanism SBNDM algorithm is faster than the BNDM algorithm. Here we are not going to finding the largest prefix for shifting. Here we always go for first prefix.

In SBNDM, the average length of shift is reduced. Here the innermost loop of algorithm becomes simple. It is faster than the BNDM algorithm in some of the favourable cases. Figure 4 shows the shifting logic of the SBNDM Algorithm.

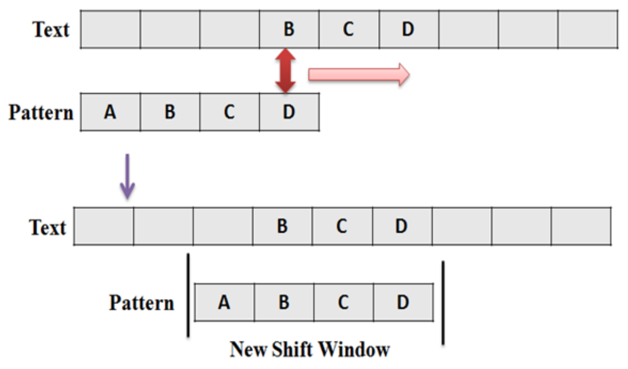

Figure 3: TNDM shifting logic

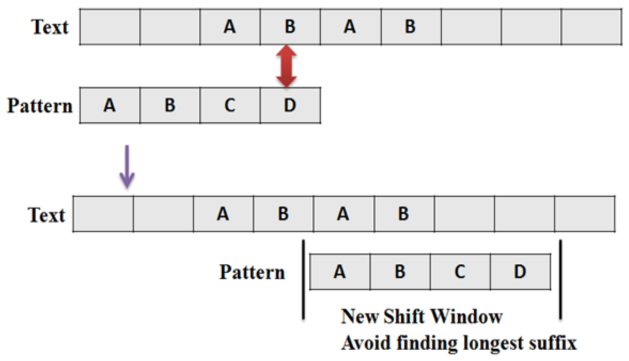

Figure 4: SBNDM shifting logic

E. BNDMq Algorithm:

BNDM with Q-gram [17] is an enhanced variation of the Basic BNDM algorithm. In BNDMq, the q-characters are read at each alignment before testing the state variable. In this algorithm loop has been made in order to quickly advance m-q+1 position, where m is the length of pattern. Here q can be varied according to our requirement. The whole algorithm can be easily understood with the help of the example given below.

We perform the BNDMq algorithm whose various step are described in the example shown in Table 3. Here we don't enter the loop until the q gram is matched.

Let us assume the text is "DeepthinKing" and pattern is "King". Bit vector B [K] = 1000, B [i] = 0100, B [n] = 0010, B [g] = 0001 and other = 0000.

Table 3: BNDMq Processing Example

S.No.	Matching Process	Comments

1	Q=2,i=3
	D <= f(i,q) i.e. f(3,2)
	D=(B[e]=0000&B[p]<<1=0000)
	D=0000
	So i=i+m-q+1=6
		No match shifts the window.

2	D<= f(6,2)
	D=(B[h]=0000&B[i],<<1=1000)
	D=0000
	So i=i+m-q+1=9
		No match shifts the window.

3	D<= f(9,2)
	D=(B[K]=1000&B[i]<<1=1000)
	D=1000
	First = i-(m-q+1)=6
		Match shift the window by next Q.

4	D<= f(11,2)
	D=(B[n]=0010&B[g]<<1=0010)
	D=0010
	First = i-(m-q+1)=8, j=10
	D=(D=0100 & B[i]=0100)
	D=0100
	Left shift
	J=9
	D=(1000&B[K]=1000)
	D=1000
		MSB 1 means pattern found.

F. Shift-OR with Q-Grams:

Shift OR with Q-gram [16] is an improved version of the Shift-OR algorithm. In Shift-OR with Q-gram algorithm we take Q character at a time for comparison by doing this, the size of simulation automata will be reduced and the number of comparisons for finding pattern is reduced up to certain level. The Q-gram can be Successive Q-gram or Overlie Q-gram. In Successive Q-gram we read pattern in a sequence of q character at a time while in Overlie Q-gram we take q character from each character of the patterns.

Let us take an example of Shift-OR Successive 2-Gram where Text is "DeepthinKing" and Patterns are 'inKing', and 'epthin'. Hence the Successive 2-gram of patterns are B [in] = 010, B [Ki] = 101, B [ng] = 011, B [ep] = 110, B [th] = 101 and other = 111. Table 4 shows the processing of the algorithm.

Table 4: Shift-OR with Q-grams Example

S.No.	Matching Process	Comments

1	D=111 ,D<<1
	D=((D=110)OR(B[De]=111)
	D=111
		No match remain the same state

2	D=111 ,D<<1
	D=((D=110)OR(B[ee]=111)
	D=111
		No match remain the same state

3	D=111 ,D<<1
	D=((D=110)OR(B[ep]=110)
	D=110
		Move to next state

4	D=110 ,D<<1
	D=((D=100)OR(B[th]=101)
	D=101
		Move to next state

5	D=101 ,D<<1
	D=((D=010)OR(B[in]=010)
	D=010
		MSB 0 means pattern match.

6	D=101 ,D<<1
	D=((D=100)OR(B[Ki]=101)
	D=101
		Move to next state

7	D=101 ,D<<1
	D=((D=010)OR(B[ng]=011)
	D=011
		MSB 0 means pattern match.

Because of the Q-Grams technique Shift-OR Algorithm becomes faster and the total number of false matches is reduced.

G. Multiple Patterns BNDM Algorithm:

Changsheng Miao, Guiran Chang and Xingwei Wang convert the BNDM algorithm in multiple pattern BNDM algorithms [17]. They develop Filtering Based Multiple String Matching Algorithm by Combining Q-Grams and BNDM. It is very simple algorithm and may contain false matches. In multiple pattern BNDM string matching algorithm multiple patterns can be searched this was not possible in normal BNDM algorithm. We need filtering after pattern match to find the exact match which will take extra time to search. This algorithm is the similar to Shift-OR with Q-gram only the change is instead of Shift-OR it uses the BNDM matching process.

IV. Conclusion

Bit parallelism is the inherent behaviour of the computer through which we can get the speed-up in various real world applications where bitwise operations are used. Bit parallel string matching algorithms are the new series of efficient algorithms. BNDM, TNDM, SBNDM, BNDMq are the bit parallel single pattern string matching algorithms and Shift-OR, Shift-OR with Q-Grams, multiple pattern BNDM are used for multiple pattern matching bit parallel algorithms. These are efficient while comparing to standard character based algorithm. These algorithms have some disadvantages, major disadvantage of these algorithm is working on patterns of length less than or equal to computer word. Multiple pattern algorithms were working on the pattern of equal length and it allows the approximation in matching results.

V. Future Work

Multiple pattern bit parallel exact string matching algorithm is not available yet. So multiple pattern exact string matching is required for matching. Word size limitation present in the bit parallel algorithm can also be removed in future.

References

[1].Vidya Saikrishna, Akhtar Rasool and Nilay Khare, “Time Efficient String Matching Solution for Single and Multiple Pattern using Bit Parallelism”, In procd. Of International Journal of Computer Applications (0975 – 8887) Volume 46– No.6, May 2012.

[2].Christian Charras and Thierry Lecroq,” Handbook of Exact String_Matching Algorithms”, Published in King’s college publication, Feb 2004.

[3].Vidya Saikrishna, Akhtar Rasool and Nilay Khare, "String Matching and its Applications in Diversified Fields",IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012 Page-219-226. ISSN (Online): 1694-0814.

[4]. Knuth D E, Morris Jr J. H and Pratt V. R,” Fast pattern matching in strings”, In the procd. Of SIAM J.Comput., Vol. 6, 1, pp. 323–350, 1977.

[5]. Boyer R S and Moore J S,”A fast string searching algorithm”, Communication of ACM 20, Vol. 10, pp. 762–772, 1977.

[6] Horspool R N,”Practical fast searching in strings”, In proc. Of Software Practical Exp, Vol. 10, 6, pp. 501–506, 1980.

[7].Timo Raita,”Tuning the Boyer–Moore–Horspool String Searching Algorithm”, In the proc. of Software Practice and Experience, Vol. 22(10), pp. 879–884, Oct. 1992.

[8].Jingbo Yuan, Jisen Zheng and Shunli Ding, “An Improved Pattern Matching Algorithm”, In the proc. of Third International Symposium on Intelligent Information Technology and Security Informatics (IITSI), pp. 599-603, 2-4 April 2010.

[9] Linquan Xie, Xiaoming Liu and Guangxue Yue, “Improved Pattern Matching Algorithm of BMHS”, In the proc. of International Symposium on Information Science and Engineering (ISISE), pp. 616-619, 24-26 Dec 2010.

[10]. JingboYuan, Jinsong Yang and Shunli Ding,” An Improved Pattern Matching Algorithm Based on BMHS”, In the proc. Of 11th International Symposium on Distributed Computing and Applications to Business, Engineering & Science, 2012.

[11]. Baojun Zhang , XiaoPing Chen , Lingdi Ping , Wu, Zhaohui, ”Address Filtering Based Wu-Manber Multiple Patterns Matching Algorithm”, In the proc. of 2009 Second International Workshop on Computer Science and Engineering[WCSE], Qingdao, Vol.1, pp. 408 – 412,28-30 Oct. 2009.

[12]. Alfred v aho and Margaret j corasick,”efficient string matching: an aid to bibliographic search” communication of acm, vol. 18, June 1975.

[13].Ricardo A. Baeza-Yates and Gaston H. Gonnet,”A New Approach to Text Searching”, In Communications of the ACM, pp. 74-82, Oct 1992.

[14]. G. Navarro and M. Raffinot, “Fast and flexible string matching by combining bit-parallelism and suffix automata”,ACM Journal. Experimental Algorithmics 1998.

[15]. Hannu Peltola and Jorma Tarhio,” Alternative Algorithms for Bit-Parallel String Matching”, String Processing and Information Retrieval, Spire Springer, LNCS 2857, pp. 80-93, 2003.

[16]. L. Salmela, J. Tarhio, and J. Kytojoki, “Multi pattern string matching with q-grams”, Journal of Experimental Algorithms, Volume 11, pp. 1-19, 2006.

[17].Branislav Durian, Jan Holub, Hannu Peltola and Jarma Tarhio,”Tuning BNDM with q-grams”, In the proc. Of workshop on algorithm engineering and experiments, SIAM USA, pp. 29-37, 2009

[18].Changsheng Miao, Guiran Chang and Xingwei Wang,” Filtering Based Multiple String Matching Algorithm Combining q-Grams and BNDM”, In proc. Of Fourth International Conference on Genetic and Evolutionary Computing, 2010.