Bit Parallel String Matching Algorithms: A Survey

Sumit Gupta
Dept. of Computer Science of Engineering
Maulana Azad National Institute of Technology,
Bhopal, India

Akhtar Rasool
Dept. of Computer Science of Engineering
Maulana Azad National Institute of Technology
Bhopal, India

ABSTRACT
The intrinsic parallelism in bit operations like AND/OR inside a computer word is known as bit parallelism. Since 1992, this bit parallelism is directly used in string matching for matching efficiency improvement. Some of the popular bit parallel string matching algorithms Shift OR, Shift OR with Q-Gram, BNDM, TNDM, SBNDM, LBNDM, FBNDM, BNDMq, and Multiple pattern BNDM. This paper discusses the working of various bit parallel string matching algorithms with example. Here we present how bit parallelism is useful for efficiency improvement in various algorithms.

Keywords
String Matching, Bit Parallelism, Shift OR, BNDM, TNDM, SBNDM, LBNDM, FBNDM, BNDMq, SBNDMq, WW Algorithm.

1. INTRODUCTION
Bit parallelism [1] is an intrinsic property of computer in which bit operations are performed parallely within the computer word in the single clock. With the use of bit parallelism in String Matching algorithm speed of matching is improved up to certain level. String matching [2] algorithms are used in most of the real world applications where pattern extraction is required like as Intrusion Detection system [3][4], Plagiarism detection [5], Data Mining [6] and Bioinformatics [7]. Bit parallel algorithms are faster than the other benchmark character based algorithms like as KMP [4][8], BM [9][10], BMH [11][12], BMHS[13], BMHS2[14], BMI[15], Improved BMHS[16], Cmmentz Walter[17][18], Wu Manber[19][20] and Aho-Corasick [21][22] etc. Bit Parallel algorithms [23] are based on the non deterministic automata but there is no such automata are present. It is simply the efficient simulation of non deterministic automata. Figure-1 shows how the bit parallel operations are performed inside the computer word. Here computer word size length is 8 bits or 1 byte.

Computer Word 1:	1	0	0	0	1	1	1	1
Computer Word 2:	1	0	0	0	0	0	1	0

After performing "&" operation at bit level in parallel, the resultant word can be obtained like

Resultant Word 3:	1	0	0	0	0	0	1	0

Figure 1: Operation perform inside the computer

First bit parallel algorithm was introduced in 1992 by the Baeza–Yates and Gonnet named as Shift OR algorithm [24]. It was an approximate multiple pattern string matching algorithm. It was faster than the previous algorithms but gives false matches.

After Shift OR in 1998 Navarro and Raffinot were introduced new bit parallel algorithm named as BNDM (Backward Non Deterministic Matching)[25]. This was an exact single pattern string matching algorithm. In BNDM algorithm we use AND or SHIFT operation which performed in parallel. BNDM is faster than character based algorithms but here pattern size must be smaller or equal than the computer word size.

BNDM set the benchmark in the string matching algorithm. After BNDM in 2003 Peltola and Tarhio introduced improved version of BNDM known as TNDM (Two way Non Deterministic Matching)[26]. In TNDM scanning is much similar to the BNDM except mismatch at last position instead of shifting it scan forward.

In 2003 another algorithm was introduced Simplified BNDM also known as SBNDM [26]. It was an improved version of BNDM. Here, it is not required to find the longest prefix that's why the average length of shift is reduced. Here, the inner most loop becomes simpler.

In 2005 Longtao He, Binxing Fang and Jie Sui proposed a bit parallel algorithm known as Wide Window Algorithm [27]. It use the wide window of size one less than the two time the pattern length to attempt m position in parallel. They combine the bit parallel technique with new wide window concept. It is exact single pattern matching algorithm which is faster in most of the cases.

In 2006 an improved version of Shift OR was introduced by Salmela, Tarhio and Kytojoki known as Shift OR with Q-Gram [28]. Same as Shift OR it is an approximate multiple string matching algorithm. It considers Q character at a time for comparison by doing that the size of automata is reduced and number of comparisons for finding pattern is reduced up to certain level. This algorithm also reduces the false matches.

In 2009 Branislav Durian, Jan Holub, Hannu Peltola and Jorma Tarhio introduced the concept of Q gram in BNDM algorithm known as BNDMq [29]. It reads the q characters at each alignment before testing the state variable. So most of the cases the number of comparisons are less than in comparison to BNDM required.

Similarly as BNDMq Branislav Durian, Jan Holub, Hannu Peltola and Jorma Tarhio also introduced SBNDMq [29] in 2009. It is an exact single pattern string matching algorithm. Similar approach of BNDMq was used in SBNDMq which gives the better results in comparison to SBNDM.

In 2010 Changsheng Miao, Guiran Chang and Xingwei combine the concept of Q gram with BNDM and implement exact multiple string matching algorithm known as Filtering based multiple string matching algorithm [30].

Figure 2 shows the evolution of bit parallel algorithm with their description.

S.No.	Algorithm	Year	Authors
1	Shift OR	1992	Baeza–Yates and Gonnet
2	BNDM	1998	Navarro and Raffinot
3	TNDM	2003	Peltola and Tarhio
4	SBNDM	2003	Peltola and Tarhio
5	Wide Window	2005	Longtao He, Binxing Fang and Jie Sui
6	Shift OR with Q Gram	2006	Salmela, Tarhio and Kytojoki
7	BNDMq	2009	Branislav Durian, Jan Holub, Hannu Peltola and Jorma Tarhio
8	Multiple BNDM with q gram	2010	Changsheng Miao, Guiran Chang and Xingwei

Figure 2: Evolution of Bit Parallel Algorithm

This paper gives the detailed description of above bit parallel string matching algorithms. Paper starts from very first algorithm based on bit parallelism up to latest one.

2. BIT PARALLEL ALGORITHM

2.1. Shift OR Algorithm

Shift OR [13] algorithm is an approximate multiple pattern string matching algorithm which means it search number of pattern at a time but there is a possibility of error. In Shift OR algorithm order of searching is from left to right. It is design for large pattern have equal length and equal or less than the word size of a computer.

Most of the multiple pattern algorithms build a trie of the pattern in pre-processing phase so as the pattern size increases size of tree also increase which is not practical to maintain. Shift OR algorithm is a simulation of nondeterministic automata where we do not need to build any trie. They don't need to buffer the input. It is real time algorithm suitable to be implemented in hardware.

Shift OR consist into two phase:

Pre-Processing Phase 2. Searching Phase

Pre-processing phase: In pre-processing phase we find out the bit vector of the every character of alphabet. In this 'ith' bit is zero if and only if character appears at position 'i' otherwise place 1 and write it into reverse order.

Searching Phase: The automation has a transition from state 'i' to 'i+1' on character c if and only if 'ith' bit in B[c] is 0. In state vector D where 'ith' bit is 0 if and only if state 'i' in the automation is active. If 0 is occurs at MSB means we find the pattern at that position.

```
ALGORITHM Shift OR (text=t1…tn, patterns=p1,..,pk)[24]
1. Pre-processing
   [Text[i]]  <- 1m, s= 1
    for j= 0…k do
     for i= 0… m-1 do
      B [p[j][i]] <- B[p[j][i] & ~ (s<<1)
       end for
     end for
2. Searching
   While pos < n do
     D = 1m
     D = D <<1| B [text [pos + j]]
       if D> 1m-1 do pos<- pos + 1
       else do
           count <- count +1
           Report occurrence at position pos<- pos-m+1
           D <- 1m
           pos<- pos +1
       end else
   end while.
```

Let's take an example to understand Shift OR algorithm. Suppose FAST, MACC and BATC be the patterns of length 4 and STRINGFASTMATCH be the text of length 15.

Bit Vector: B[F] = 1110, B[A] = 1101, B[S] = 1011, B[T] = 0011, B[M] = 1110, B[C] = 0011, B[B] = 1110 and for others 1111.

TEXT	D	B[Ti]	Pos	D′	Comments
STRINGFASTMATCH	1111	-	-		D all ones
STRINGFASTMATCH	1110	1011	1	1111	D & B[S]
STRINGFASTMATCH	1110	0011	2	1111	D & B[T]
STRINGFASTMATCH	1110	1111	3	1111	D & B[R]
STRINGFASTMATCH	1110	1111	4	1111	D & B[I]
STRINGFASTMATCH	1110	1111	5	1111	D & B[N]
STRINGFASTMATCH	1110	1111	6	1111	D & B[G]
STRINGFASTMATCH	1110	1110	7	1110	D & B[F]
STRINGFASTMATCH	1100	1101	8	1101	D & B[A]
STRINGFASTMATCH	1010	1011	9	1011	D & B[S]
STRINGFASTMATCH	0110	0111	10	0111	MSB is 0, Pattern found
STRINGFASTMATCH	1110	1110	11	1110	D & B[M]
STRINGFASTMATCH	1100	1101	12	1101	D & B[A]
STRINGFASTMATCH	1010	0011	13	1011	D & B[T]
STRINGFASTMATCH	0110	0011	14	0111	MSB is 0, Pattern found
STRINGFASTMATCH	1110	1111	15	1111	D & B[H]

* Pattern are found at position 7 & 11

Figure 3: Various steps of Shift OR Algorithm

Figure 3 shows the various step involved in the shift OR algorithm. Initially we set the value of D = 1111 (All one) and update D when a character c is read from the text as follow D= (D<<1) | B[c]. Whole searching is carry out from left to right one by one if 0 is occurred at MSB it means pattern is found.

In Shift OR algorithm pre-processing and search are very simple and only bitwise logical operations Shift and AND are used and no Buffering is required. It is a real time algorithm. Time delay to process one text character is bounded by a constant depend only on pattern length. Here all pattern length must be equal.

2.2. BNDM Algorithm

BNDM stands for Backward Non Deterministic Matching [25]. It is exact single pattern string matching algorithm. In BNDM order of searching is from right to left. It uses the concept of bit parallelism from shift OR algorithm [24] and suffix automata from BDM algorithm [25]. This algorithm is a bit parallel simulation of BDM algorithm. BDM skips character using suffix automata which is deterministic in pre processing. To construct Deterministic automata is complex task. BNDM simulates the non deterministic version using bit parallelism.

BNDM algorithm consist in two phase

1. Pre-processing phase 2. Searching phase

Pre-processing: In pre-processing phase we find Bit Vector of each Character of the Pattern calculated by putting 1 for occurrence and 0 for non-occurrence and take bit vector D with initial value with all one.

Searching Phase: In searching phase pattern is searched with the help of two logical operator that are AND and SHIFT. Pattern is searched when MSB of D is 1 and value of j is 0.

```
Algorithm BNDM (P=p1p2p3..pm,T= t1t2..tn)
 Preprocessing
```

```
For c ∈ Σ do B[c] ←0m
  For i∈ 1...m do B [pm-i+1] ← B [pm-i+1] | 0m-110i-1
Searching
  Pos← 0
  While (pos<=n-m) do
   J←m, last←m
   D=1m
   While d! = 0m do
    D ←D& B [Tpos+j]
    J← j-1
    If D & 10m-1! = 0m then
     If j>0 then last ← j
     Else report an occurrence at pos+1
    D← D<<1
    End of While
   Pos←pos+last
  End of while
```

Let us understand the working of whole algorithm with the help of an example. Let Text T = 'STRINGFASTMATCH' and Pattern P= 'FAST'

Bit Vector: B [F] =1000, B [A] =0100, B[S] =0010, B [T] =0001 and other are all zero.
Initially J=4, last=4, pos. = 0 and bit vector D =1111
Than we perform the BNDM algorithm. There various step are shown in the Figure 4 with proper explanation.

TEXT WINDOW	D	B[Tpos+j]	D′	j	Last	Pos.	Comment
[STRI]NGFASTMATCH	1111	-	-	4	4	0	Initial window arrangement
[STRI]NGFASTMATCH	1111	0000	0000	3	4	4	D & B[I] Decrement j by 1
STRI[NGFA]STMATCH	1111	0100	0100	3	4	4	All zero update pos. value
STRI[NGFA]STMATCH	1000	1000	1000	2	2	4	MSB 1 & j> 0 update last=j
STRING[FAST]MATCH	1111	0001	0001	3	4	6	D & B[T] Decrement j by 1
STRING[FAST]MATCH	0010	0010	0010	2	4	6	D & B[S] Decrement j by 1
STRING[FAST]MATCH	0100	0100	0100	1	4	6	D & B[A] Decrement j by 1
STRING[FAST]MATCH	1000	1000	1000	0	4	6	MSB 1 & j-0 pattern found
STRINGFAST[MATC]H	1111	0000	0000	3	4	10	All zero exit

Figure 4: Shows the various steps perform by BNDM

BNDM algorithm become very fast string matching algorithm except very sort (0-6) or very long (90-150) pattern. It is faster than the previous algorithm Shift OR, BDM and Occupies very less space perform various operation in parallel. It is very Simple and flexible algorithm. But we have all pattern assume to less than or equal to the word size of computer.

2.3. TNDM Algorithm

TNDM stands for Two ways Non Deterministic Matching [26]. It is Exact Single pattern string matching algorithm. It is almost same as the BNDM algorithm [25] there is slight changes in the case of mismatch occur at first position instead of shifting TNDM look forward to find suffix of reverse pattern. The number of examined characters is less than BNDM therefore matching is faster.
The simulation of TNDM Algorithm can be understood by the help of the Figure 5 here mismatch is occurred at last so it scan forward to find the maximum suffix of pattern after shifting to maximum suffix the algorithm is run same as BNDM algorithm[25].

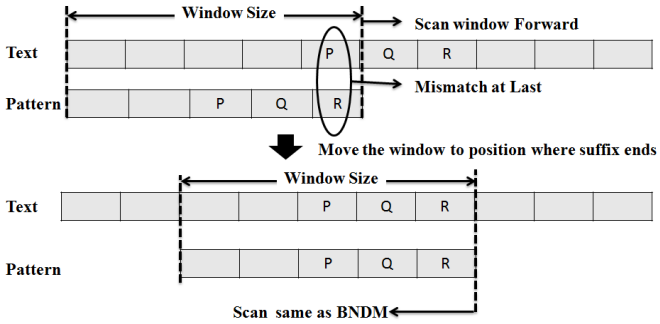

Figure 5: Working explanation of TNDM algorithm

2.4. SBNDM Algorithm

SBNDM [26] stands for Simple Backward Non Deterministic Matching. SBNDM is much similar to the BNDM [25] algorithm there is a slight change in term of shifting. Due to this it is quit faster than the BNDM algorithm. Here we do not require finding the longest prefix. Let T is the text of length n and P is the pattern of length m to be searched. At each alignment window of P in T, Scan T from right to left until the suffix of the window is not a factor of P or an occurrence of P is found. Shifting of SBNDM is done according to these i.e. shift window by m if no factor is found, shift by 1 if P found otherwise next alignment is start at last factor. Figure 6 describes the various steps of the SBNDM algorithm.

Pattern= 'DESIGN', Text = 'SFZIGNBACDESIGN'

ALIGNMENT	S	F	Z	I	G	N	B	A	C	D	E	S	I	G	N
						N									
					G	N									
				I	G	N									
NOT A FACTOR			Z	I	G	N									
NEXT ALIGNMENT	S	F	Z	I	G	N	B	A	C	D	E	S	I	G	N
NOT A FACTOR									C						
NEXT ALIGNMENT	S	F	Z	I	G	N	B	A	C	D	E	S	I	G	N
PATTERN FOUND										D	E	S	I	G	N

Figure 6: Example of SBNDM

By using the SBNDM concept the average length of shift is reduced by doing so the innermost loop of algorithm become simpler. It is faster than the BNDM algorithm.

2.5. Wide Window String Matching Algorithm

Longtao He, Binxing Fang and Jie Sui proposed an algorithm known as WW (wide window) Algorithm [27] in which text is divided into n/m where n is the length of text and m is the length of pattern. It opens a window with size 2m-1. in the window the algorithm attempts m possible occurrence position in parallel. In this window is dividing it into two parts. First one is denote as A1 of size p-1 and second part as A2 of size p. With the help of the Figure 7 we can understand how pattern is divided into window.

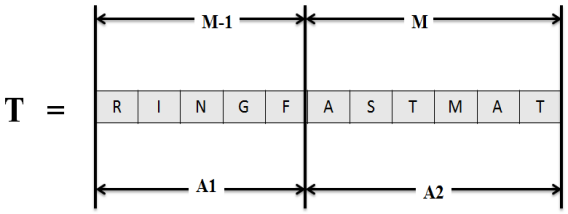

Figure 7: Shows how Pattern is divided into Window

Now in this algorithm we have to find out the all prefixes of A2 which is also the suffixes of the pattern m and shift the window directly if it fails. Let r denotes the length of longest prefix. Than we find the suffix of A1 which is also prefix of pattern m equal to the length of m-r if such cases are found means we got the pattern otherwise not.

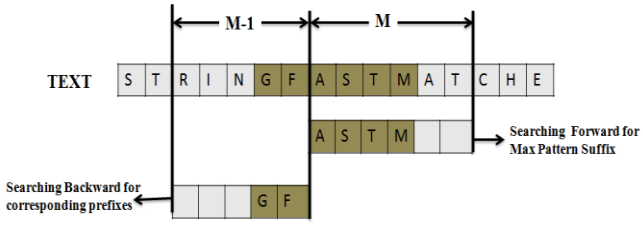

Figure 8: Demonstrate process of searching in WW Algorithm

Figure 8 shows the process of WW algorithm here we have a pattern 'GFASTM' of size 6 so window size is 2m-1 that is 11. Now longest suffix of pattern in A2 is 'ASTM' from A1 we got 'GF' it means pattern is found. WW is efficient for searching short pattern. It suits for off line pattern matching as well as high speed online pattern matching.

2.6. Shift OR with Q-Gram

Shift OR with Q-gram [28] is an enhanced version of the Shift OR algorithm. In Shift OR with Q-gram algorithm we take Q character at a time for comparison by doing so the size of automata is reduced and the number of comparison for finding pattern is reduced up to certain level. The Q-gram can be of two types Consecutive Q-gram or Overlapped Q-gram. In Consecutive Q-gram we read pattern in a sequence of q character at a time while in Overlapped Q-gram we take q character from each character of the patterns. Here the pattern length of each pattern must be same. Shift OR with Q-gram carried out in three phases First phase: Initialization Phase where initialization of the variable is carried out. Second phase: Pre-processing phase where bit vector of the various q gram are taking place. Third phase: Searching phase here searching of the pattern in the text is carried out.

Let us take an example of Shift OR Consecutive 2-Gram where Text is STRINGFASTMATCH and Patterns are 'GFASTM', 'ABATCH' and 'TMACCT' hence the consecutive 2-gram of patterns are "GF", "AS", "TM", "AC", "CT", "AB", "AT" and "CH" by doing so the number of bit in bit vector is reduced to the half of the pattern length. Here 2-gram of the pattern is treated as single character. The ith bit of the bit vector is set to zero if there is an occurrence of 2-gram in the ith position of the pattern otherwise set to one. So bit vector of our pattern is as follow.

B [GF] = 110, B [AS] = 101, B [TM] = 010, B [AC] = 101, B [CT] = 011, B[AB] = 110, B[AT] = 101 and B [CH] = 011.

Matching the text: STRINGFASTMATCH

Initialise D = 111 and Update D when a character c is read from the text by D= (D<<1) | B[c].

Various steps involved are discussed in the Figure 9 here patterns are searched similar to Shift OR by taking q character at a time. Searching is start from left of the text taking ST first we got all one means mismatch so shift the window and take the next q gram TR and same process is repeated till the text end.

TEXT	D	B[T]	D′	Comment
STRINGFASTMATCH	111	-	-	Initial D all one
STRINGFASTMATCH	110	111	111	D OR B[ST]
STRINGFASTMATCH	110	111	111	All one means no match
STRINGFASTMATCH	110	111	111	Shift one position right
STRINGFASTMATCH	110	111	111	
STRINGFASTMATCH	110	111	111	
STRINGFASTMATCH	110	110	110	Partial match found
STRINGFASTMATCH	100	101	101	D OR B[AS]
STRINGFASTMATCH	010	010	010	MSB 0 means pattern found
STRINGFASTMATCH	100	101	101	
STRINGFASTMATCH	010	011	011	MSB 0 means pattern found

MSB 0 Means pattern found. Hence pattern found at 6 & 10

Figure 9: Shows the various steps of Shift OR with Q Gram

In the example two patterns is found at 6 & 10 but second one is not in the patterns so it is a false match. By taking the q gram the time efficiency of the algorithm is improved but other factor is unchanged it is still not working for pattern lager than the word size and unequal pattern length. Here the number of false matches is reduced up to the certain level.

2.7. BNDM with Q-gram

BNDM with Q-gram [29] is an improved variation of the BNDM algorithm which reads the q gram at each alignment before testing the state variable. In this algorithm loop has been made as sort as possible in order to quickly advance m-q+1 position. Here q can be varies according to our requirement. The whole algorithm can be easily understood with the help of the example given below.

Text T = 'STRINGFASTMATCH' and Pattern P= 'FAST' Assume we have 2-gram

Pre-processing: In pre-processing phase we find Bit Vector of each Character of the Pattern calculated by putting 1 for occurrence and 0 for non-occurrence.

B[F] =1000, B[A] =0100, B[S] =0010 and B[T] =0001

Searching Phase: Initially we take variable i and set to m-q+1 where m is pattern length

Than we perform the BNDMq algorithm whose various step are described in the Figure 10 where we don't enter the loop until the q gram is matched.

Text	i	F(i,q) / D	j	First	D′	Comment
STRINGFASTMATCH	3	0000	-	-	-	D= B[R] & (B[i]<<1)
STRINGFASTMATCH	6	0000	-	-	-	D all zero shift window
STRINGFASTMATCH	9	0010	8	6	0100	D not zero enter in loop
STRINGFASTMATCH	9	0100	7	6	1000	D & B[A]
STRINGFASTMATCH	9	1000	6	6	-	MSB =1, pattern found
STRINGFASTMATCH	12	0000	-	-	-	

Figure 10: Step by step Description of BNDMq algorithm

It is become very fast string matching algorithm in the case of no match is found. In this case it does not enter the main loop and directly shift the window. It occupies same space as take in BNDM and word size limitation is still exists.

2.8. Multiple Pattern BNDM

BNDM algorithm is a single pattern string matching algorithm which is very fast one because it uses the concept of bit parallelism. Changsheng Miao, Guiran Chang and Xingwei Wang convert the BNDM algorithm in multiple pattern BNDM algorithms [30]. They develop Filtering Based Multiple String Matching Algorithm by Combining q-Grams and BNDM.

Let's understand the concept of algorithm by considering an example Text: STRINGFASTMATCH, Text size: 15, Patterns: FAST, MACC and BATC pattern length (m): 4

In preprocessing phase bit vector of the various character are calculated which is shown in the Figure 11 which are calculated in similar fashion as calculated in BNDM algorithm.

	0	1	2	3
B[F]	1	0	0	0
B[A]	0	1	0	0
B[S]	0	0	1	0
B[T]	0	0	1	1
B[M]	1	0	0	0
B[C]	0	0	1	1
B[B]	1	0	0	0

Figure 11: Shows calculation of bit vector

Initialize q = 2, i = m-q+1. The whole searching process is described in the Figure 12 step by step.

TEXT	i	F(i,q) / D	j	First	D′	Comment
STRINGFASTMATCH	-	-	-	-	-	Initial window
STRINGFASTMATCH	3	0000	-	-	-	D = B[R] & (B[I]<<1)
STRINGFASTMATCH	6	0000	-	-	-	D =0, update window by i window
STRINGFASTMATCH	9	0010	8	6	-	D ≠ 0, Enter in the loop
STRINGFASTMATCH	9	0100	7	6	0100	D & B[A]
STRINGFASTMATCH	9	1000	6	6	1000	MSB =1, Pattern found
STRINGFASTMATCH	12	0100	11	9	-	Update value of i
STRINGFASTMATCH	12	1000	10	9	1000	
STRINGFASTMATCH	10	0000	10	9	0000	D =0, update window by i
STRINGFASTMATCH	13	0011	12	10	-	
STRINGFASTMATCH	13	0110	11	10	0100	
STRINGFASTMATCH	13	1000	10	10	1000	

Pattern found at Position 7

Figure 12: Shows the various steps perform by Multiple BNDM

In multiple pattern BNDM string matching algorithm multiple patterns can be searched which were not possible in BNDM Algorithm. We need filtering after pattern match to find the exact match which will take extra time to search.

3. CONCLUSION

Bit parallel string matching algorithms are the new series of efficient algorithm. Each algorithm is efficient in compare to standard character based algorithm. BNDM, TNDM, SBNDM, WW, BNDMq are the bit parallel single pattern string matching algorithm whereas Shift OR, Shift OR with Q-gram, multiple pattern BNDM are used for multiple pattern matching. Multiple pattern matching is difficult in compare to single pattern matching for multiple matching no efficient exact string matching is possible using bit parallelism. These algorithms have some disadvantages. Major disadvantage is these algorithm is working on patterns of length less than or equal to computer word. These algorithms were working on the pattern of equal length.

4. FUTURE WORK

Most of the bit parallel algorithm is single pattern string matching algorithm so using this concept multiple patterns string matching algorithm can be implemented. Word size limitation present in the bit parallel algorithm can also be avoided.

5. 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] Ali Peiravi, "Application of string matching in Internet Security and Reliability", Marsland Press Journal of American Science, 6(1), pp. 25-33, 2010.

[4] Pei-fei Wu and Hai-juanShen,"The Research and Amelioration of Pattern-matching Algorithm in Intrusion Detection System", In the proc. of IEEE 14th International Conference on High Performance Computing and Communication & IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), pp. 1712-1715, 25-27 June 2012.

[5] Ramazan S. Aygün "structural-to-syntactic matching similar documents", Journal Knowledge and Information Systems, ACM Digital Library, Volume 16 Issue 3, pages 303-329, Aug 2008.

[6] Sanchez D., Martin-Bautista M.J., Blanco I. and Torre C.," Text Knowledge Mining: An Alternative to Text Data Mining", In the proc. of IEEE International Conference on Data Mining Workshops, ICDMW '08, pp. 664-672, 15-19Dec. 2008.

[7] Robert M. Horton,Ph.D. "Bioinformatics Algorithm Demonstrations in Microsoft Excel", California State University, Sacramento, 2004.

[8] 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.

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

[10] Zhengda Xiong,"A Composite Boyer-Moore Algorithm for the String Matching Problem", In the proc. of International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), pp. 492-496, 8-11 Dec 2010.

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

[12] 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.

[13] 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.

[14] Yuting Han and Guoai Xu, "Improved Algorithm of Pattern Matching based on BMHS", In the proc. of IEEE International Conference on Information Theory and Information Security (ICITIS), pp. 238-241, 17-19 Dec 2010.

[15] 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.

[16] 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.

[17] Commentz-Walter, "A string matching algorithm fast on the average," In the Proc. of 6th International Colloquium on Automata, Languages, and Programming, pp. 118–132,1979.

[18] Kouzinopoulos, C.S. and Margaritis, K.G.,"A Performance Evaluation of the Pre-processing Phase of Multiple Keyword Matching Algorithms", In the proc. of 15th Panhellenic Conference on Informatics (PCI), pp. 85-89, 30 Sept 2011- 2 Oct 2011.

[19] Yang Dong hong, XuKe and Cui Yong,"An improved Wu-Manber multiple patterns matching algorithm", In the proc. Of 25th IEEE International Performance, Computing, and Communications Conference, IPCCC, pp. 680, 10-12 April 2006.

[20] 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.

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

[22] Tao Tao and Mukherjee A.,"Multiple-pattern matching in LZW compressed files using Aho-Corasick algorithm", In the proc. of Data Compression Conference, 21-31 March 2005.

[23] Faro S. and Lecroq T,"The exact online string matching problem: A review of the most recent results", ACM Comput. Survey, Article 13, 42 pages, February 2013.

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

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

[26] 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.

[27] Longtao Hea, Binxing Fangaand Jie Sui," The wide window string matching algorithm" In the procd. Of Theoretical Computer Science of Elsevier, Vol. 332, pp. 391-404, 2005.

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

[29] 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.

[30] 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.