A MEMORY EFFICIENT HARDWARE BASED PATTERN MATCHING AND PROTEIN ALIGNMENT SCHEMES FOR HIGHLY COMPLEX DATABASES

M.AntoBennet,[1] S.Sankaranarayanan[2] M.Deepika[2] N.Nanthini[2] S.Bhuvaneshwari[2]M.Priyanka

[1] Faculty of Electronics and Communication Department, vel tech, Chennai, India.

[2] UG Students of Electronics and Communication Department, vel tech , Chennai, India.

Email:bennetmab@gmail.com

Submitted: May 27, 2017 ***Accepted: June 15, 2017*** ***Published: Sep 1, 2017***

Abstract- Protein sequence alignment to find correlation between different species, or genetic mutations etc. is the most computational intensive task when performing protein comparison. To speed-up the alignment, Systolic Arrays (SAs) have been used. In order to avoid the internal-loop problem which reduces the performance, pipeline interleaving strategy has been presented. This strategy is applied to an SA for Smith Waterman (SW) algorithm which is an alignment algorithm to locally align two proteins. In the proposed system, the above methodology has been extended to implement a memory efficient FPGA-hardware based Network Intrusion Detection System (NIDS) to speed up network processing. The pattern matching in Intrusion Detection Systems (IDS) is done using SNORT to find the pattern of intrusions. A Finite State Machine (FSM) based Processing Elements (PE) unit to achieve minimum number of states for pattern matching and bit wise early intrusion detection to increase the throughput by pipelining is presented.

Index terms*:* Systolic Arrays (SAs),Intrusion Detection Systems (IDS),Network Intrusion Detection System (NIDS), protein Data Bases (DBs).

I. INTRODUCTION

The proliferation of Internet and networking applications, coupled with the wide-spread availability of system hacks and viruses have increased the need for network security. Firewalls have been used extensively to prevent access to systems from all but a few, well defined access points (ports), but they cannot eliminate all security threats, nor can they detect attacks when they happen. Stateful inspection firewalls are able to understand details of the protocol that are inspecting by tracking the state of a connection. They actually establish and monitor connections for when it is terminated. However, current network security needs, require a much more efficient analysis and understanding of the application data. Content-based security threats and problems occur more frequently, in an everyday basis. Virus and worm inflections, Spams (unsolicited e-mails), email spoofing, and dangerous or undesirable data, get more and more annoying and cause innumerable problems. Therefore, next generation firewalls should provide deep packet Inspection capabilities, in order to provide protection from these attacks. Such systems check packet header, rely on pattern matching techniques to analyze packet payload, and make decisions on the significance of the packet body, based on the content of the payload. Network Intrusion Detection Systems (NIDS) perform deep packet inspection. They scan packet's payload looking for patterns that would indicate security threats. Matching every incoming byte, though, against thousands of pattern characters at wire rates is a complicated task. Measurements on SNORT show that 31% of total processing is due to string matching; the percentage goes up to 80% in the case of Web-intensive traffic. So, string matching can be considered as one of the most computationally intensive parts of a NIDS and in this work we focus on payload matching. Intrusion detection systems running in General Purpose Processor (GPP) can only serve up to a few hundred Mbps throughput. Therefore, seeking for hardware-based solutions is possibly the only way to increase performance for speeds higher than a few hundred Mbps.Until now several Application Specific Integrated Circuit (ASIC) commercial products have been developed. These systems can support high throughput, but constitute a relatively expensive solution. On the other hand, Field Programmable Gate Array (FPGA)-based systems provide higher flexibility and high throughput comparable to ASICs performance. FPGA-based platforms can exploit the fact that the NIDS rules change relatively infrequently,

and use reconfiguration to reduce implementation cost. In addition, they can exploit parallelism in order to achieve satisfactory processing throughput. Additionally, matching a large number of patterns has high area cost, so sharing logic is critical, since it could save a significant amount of resources, and make designs smaller and faster.

A NIDS monitors traffic on a network looking for suspicious activity, which could be an attack or unauthorized activity. A large NIDS server can be set up on a backbone network, to monitor all traffic; or smaller systems can be set up to monitor traffic for a particular server, switch, gateway, or router. In addition to monitoring incoming and outgoing network traffic, a NIDS server can also scan system files looking for unauthorized activity and to maintain data and file integrity. The NIDS server can also detect changes in the server core components. In addition to traffic monitoring, a NIDS server can also scan server log files and look for suspicious traffic or usage patterns that match a typical network compromise or a remote hacking attempt. The NIDS server can also server a proactive role instead of a protective or reactive function. Possible uses include scanning local firewalls or network servers for potential exploits.

Protein alignment is a way of arranging the sequences of protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences. Aligned sequences of nucleotide or amino acid residues are typically represented as rows within a matrix. Gaps are inserted between the residues so that identical or similar characters are aligned in successive columns. Sequence alignments are also used for non-biological sequences, such as those present in natural language or in financial data.

Systolic systems consists of an array of PE (Processing Elements) processors are called cells, each cell is connected to a small number of nearest neighbours in a mesh like topology. Each cell performs a sequence of operations on data that flows between them. Generally the operations will be the same in each cell, each cell performs an operation or small number of operations on a data item and then passes it to its neighbour.

Since string matching is the most computationally intensive part of an NIDS, our proposed architectures exploit the benefits of FPGAs to design efficient string matching systems. The proposed architectures can support between 3 to 10 Gbps throughput, storing an entire NIDS set of patterns in a single device. In this work, I suggest solutions to maintain high performance and minimize area cost, show also how pattern matching designs can be updated and partially or entirely changed, and advocate that some solutions can offer high performance, while require low

area. Techniques such as fine-grain pipelining, parallelism, partitioning, and pre-decoding are described, analyzing how they affect performance and resource consumption.

This work proposes a pattern matching algorithm that reduces total memory requirements by sharing common infixes of target patterns. For the pattern identification, a state should contain its own match vector with a set of bits, where each bit represents a matched pattern in the state. Even though the information of shared common infixes was stored in match vectors, the number of shared common infixes was limited by the size of the match vectors.

In order to reduce the memory requirements of the DFA-based string matching engine, this proposes a memory-efficient parallel string matching scheme using the pattern dividing approach. Long target patterns are divided into sub-patterns with a fixed length; therefore, the variety of target pattern lengths can be mitigated. Moreover, the number of shared common states increases due to both the reduced length and the increasing number of sub-patterns, compared with the cases of the string matching with long target patterns. For each string matching, DFAs are built with bit-level input symbols for the bit splitting in order to reduce the number of state transitions from each state.

II. RELATED WORKS

It is very tough to calculate the protein to protein interaction in real time because the PPI differs from one individual to another as per the metabolism of the individual[1].To detect a subset of small proteins, DALI algorithm fails to generate any significant alignment, although suchalignments do exist.Instead of looking at the total sequence, the Smith–Waterman algorithm compares segments of all possible lengths and optimizes the similarity measure and obtains the alignment[2].Multiple sequence alignment computation stands at a cross-road between computation and biology. The computational issue is ascomplex to solve as when given, any sensible biological criterion, thecomputation of an exact MSA is NP .Complete and therefore impossible for all but unrealistically small datasets[3].The major drawback of the tools used by the author in this proposal is the time needed to scan the entire protein databases (DBs), because CPUs are used in a serial manner. Several optimizations have been attempted exploiting GPUs and HW accelerator[4].Biologists use alignment algorithms to investigate similarities between proteins of different species, in order to find phylogenetic or functional correlations, or proteins of the same species, for genetic mutations studies like cancers and genetic diseases. Biologists

have several SW tools to perform their analysis. The major drawback of these tools is the time needed to scan the entire protein Data Bases (DBs), because Central Processing Units (CPUs) are used in a serial manner[5].The discriminative motifs finding is that it lacks in elegance ofgenerative such as priors, structures and uncertainity and Relationships between variables are also not explicitly mentionable and visualizable[6].When the derived SA requires Processing Elements (PEs) with internal loops , throughput could be highly affected in fact in a loop-based sequential circuit the result of a logic operation depends on the previous operations. Therefore, in sequential circuits, it is not possible to execute a logic operation at each clock cycle, because inputs must wait many cycles to synchronize with incoming feedback signals[7].Biological networks comparison is a difficult task since it involves subgraph isomorphism checking. Therefore, exact algorithms cannot be usually afforded to solve the problem, unless cases are focused on for which tractability can be achieved via Fixed Parameter Tractability (FPT) algorithms. FPT contains all polynomial-time computable problems. Moreover, it contains all optimization problems that allow a fully polynomial-time approximation scheme[8].A memory-efficient and modular approach for large-scale string pattern matching. In Network Intrusion Detection Systems (NIDSs), string pattern matching demands exceptionally high performance to match the content of network traffic against a predefined database of malicious patterns. Much work has been done in this field. An algorithm called “leaf-attaching” to preprocess a given dictionary without increasing the number of patterns was proposed. The resulting set of post processed patterns can be searched using any tree search data structure. A scalable, high-throughput, Memory-efficient Architecture for large-scale String Matching (MASM) based on a pipelined Binary Search Tree (BST) was presented. The proposed algorithm and architecture achieve a memory efficiency of 0.56. As a result, the design scales well to support larger dictionaries. Implementations on 45 nm ASIC and a state-of-the-art FPGA device show that the architecture achieves 24 and 3.2 Gbps, respectively[9].The software-based approaches are General-Purpose Processors. In software-based approaches they only need one state transition per input character, which causes at most one memory access for each character input. However the practical use is limited because of their excessive memory usage. In hardware-based approaches, memory usage is not a concern since it can accomodate large memory[10]. Several compression techniques for DFAs have been proposed, focusing on reducing the number of transitions between states. Although the methods can reduce the memory consumption significantly it is

hard to reduce the number of states in DFAs with complex regular expressions. By using 3 single bit comparators & applying bitwise early detection method in DFA, the number of states used have been reduced[11].Pattern matching is one of the most important components for the content inspection based applications of network security, and it requires well designed algorithms and architectures to keep up with the increasing network speed. For most of the solutions, derivative algorithms are widely used. They are based on the DFA model but utilize large amount of memory because of so many transition rules. An algorithm is presented in this paper for multiple pattern matching. It uses a novel model, namely Cached Deterministic Finite Automaton (CDFA)[12].

III. PROPOSED METHOD

In order to reduce the memory requirements of the DFA-based string matching engine, this proposes a memory-efficient parallel string matching scheme using the pattern dividing approach and its hardware architecture for the pattern identification. Long target patterns are divided into sub-patterns with a fixed length; therefore, the variety of target pattern lengths can be mitigated. By balancing memory usage between the string matchers, unused memory area in homogeneous string matchers decreases. Moreover, the number of shared common states increases due to both the reduced length and the increasing number of sub-patterns, compared with the cases of the string matching with long target patterns. For each string matcher, DFAs are built with bit-level input symbols for the bit splitting in order to reduce the number of state transitions from each state. For identifying the original long target patterns, the successive matches with sub-patterns are detected using the proposed two-stage sequential string matching engine. Experimental results show that memory requirements decrease on average by 47.8 percent and 62.8 percent for selected rules Snort and ClamAV, compared with several existing bit-split string matching approaches.

3.1PROPOSED STRINGMATCHING SCHEME

Leaf-Attaching Algorithm

algorithm to disjoint the patterns, called leaf-attaching shown in Fig 1. The algorithm takes a prefix tree as the input, and outputs a set of disjoint patterns is proposed. All the child (or non-leaf) patterns are merged with their parent (or leaf) patterns. Let L be the maximum length of the patterns. Each parent pattern has a match vector of length L bits attached to it. The match vector

is a binary string, which indicates how many child-patterns are included in a parent pattern.

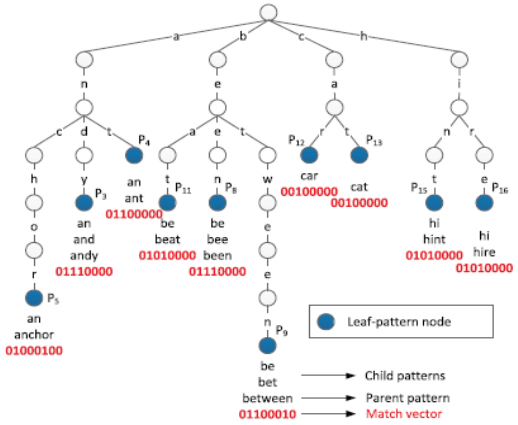

Fig.1 Leaf-attached tree

A value of 1 at position i implies that there is a child-pattern with length "I" bytes, starting from the beginning of the parent pattern. For instance, if the parent pattern is andy and its match vector is 0111, then there are three child-patterns included: an, and, and andy, corresponding to the 1 at positions 2, 3, and 4, respectively. Note that a pattern can be the child (prefix) of more than one parent pattern. Fig.2 shows the sample merging for two parent patterns, andy and between.

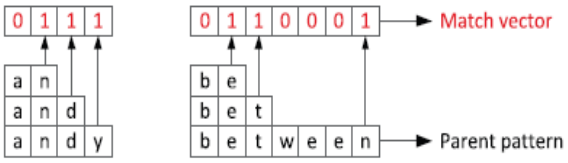

Fig 2Sample merging

3.2Memory-Efficient Structure

Memory-efficient data structure based on a complete binary search tree is presented. Complete BST is a special binary tree data structure .The binary search algorithm is a technique to find a specific element in a sorted list. In a balanced binary search tree, an element if it exists can be found in at most dlog2 (N) 1) e operations, where N is the total number of nodes in the tree. The given dictionary is leaf-attached and the leaf patterns along with their match vector are extracted. The leaf patterns are used to build the BST. Each node in the BST includes a pattern and a match vector.

3.3Sequential Matching With Divided Patterns

The match with a divided target pattern consists of successive matches with its quotient vector and remnant pattern shown in Fig 3. If a target pattern is divided by a fixed length f, the in the

quotient vector should be detected at f different points. Because the starting points of the sequential matches can be different, the points when the target pattern is matched can vary. State pointers and MSVs are held for f cycles and updated periodically every f cycles. Due to various lengths of the remnant patterns, the output states in an FSM for the remnant patterns can be reached at any cycle.

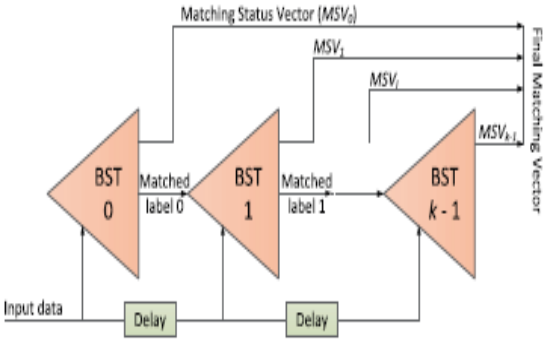

Fig 3 Block diagramof the cascading approach

3.4Pattern Partitioning and Mapping

After all target patterns are divided with a fixed length f, the unique patterns for the quotient vector matcher, the quotient index matcher, the remnant pattern matcher, and the short pattern matcher are determined. For each matcher type, sub-patterns are partitioned into multiple subsets for homogeneous string matchers. The pattern partitioning is represented in Algorithm 1 as follows: procedure denotes the pattern partitioning with patterns T and string matcher parameter M. First, patterns are sorted lexicographically to increase the number of shared common prefixes. Then, a procedure, which denotes the pattern mapping, is called for obtaining the FSM tile contents for a string matcher.The pattern mapping is repeated until there are no unmapped patterns. If the largest number of states in tries is greater than the maximum number of states in an FSM tile, next iteration is continued by reducing π by one; otherwise, the loop is broken after obtaining the map-able patterns, mapped t. The unmapped patterns are returned by a procedure Remove. After exiting for loop, a procedure “Add failing pointer” is called to add failing pointers from each state to the longest suffix state finally, for the mapped patterns mapped t, the contents are obtained by calling a procedure. Due to the hardware resource parameter such as p(M)and s(M), the pattern mapping algorithm shows the constant time complexity, O(1).

3.6Divided Pattern Matching

In order to explain the divided pattern matching with an example, the sequence of two digits between pipe symbols is the sequence of hexadecimal numbers. The length of the sub-patterns for the quotient vector is fixed as 4. All divided patterns are ordered as shown in Fig. 4, where binary code values are provided in the right column. Let us assume that the LSB of characters is adopted for the input of an FSM tile.

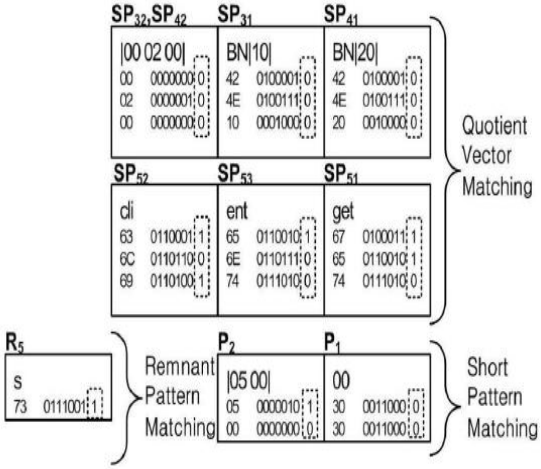

fig 4 Example of subpatterns for divided pattern matching.

Different types of FSM tiles in string matchers are adopted for the quotient vector, the remnant pattern, and the short pattern matching, respectively. The quotient vector matching adopts all possible states for the sub-patterns with the same length; only the output states should indicate nonzero PMVs, so the architecture of the FSM tile is adopted. The numbers of output states and possible states are determined according to the length of the sub-patterns. If many output states are shared, the number of mapped sub-patterns could be greater than the number of output states. In this case, the maximum number of mapped target patterns is the same as the number of bits in a PMV. DFA for the quotient vector matching where the double-circled eight states represent possible output states. In addition, the failing pointers are not shown for clarity. Sub-patterns SP32 and SP42 are identical, so only the identification index of SP32 are shown in the figure 3.4. The DFA is implemented in an FSM tile for one input bit, where the starting address of the nonzero PMV table is the same as the starting address of the FSM tile. The lengths of remnant patterns and short patterns are shorter than the fixed length of sub-patterns in the quotient vector. In this case, any states except for the initial state can be output states.

3.7String Matching EngineArchitecture

Based on the sequential matching mentioned above, architecture of the proposed string matching engine. f is the fixed length of Sub-patterns in the quotient vector. According to f, the number of the remnant pattern matchers can be varied. A character code of one byte from a payload is inputed in the quotient vector matcher. The quotient vector matcher consists of v string matchers, where the width of an FMV is equal to the number of bits in a PMV of an FSM tile, p. In the quotient vector matcher, only one bit in total temporary match vectors becomes true because only one sub-pattern can be matched in the quotient vector matcher per cycle. Therefore, the temporary match vectors are encoded using binary encoder, where the encoder output can be the quotient index.

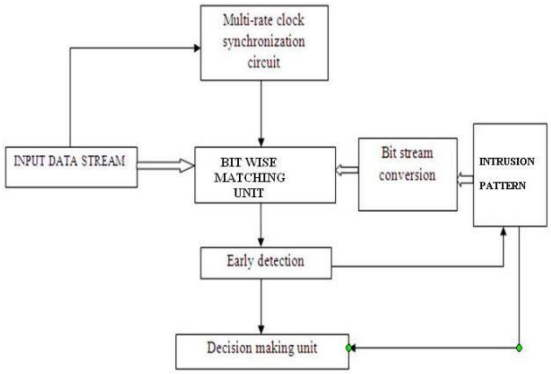

Fig 5 Pattern Matching and Detection

The above proposed block diagram in Fig 5 describes pattern matching and detection for NIDS. For the proposed NIDS the input data stream of bits is received for every clock pulse as given in the clock synchronization circuit. The pattern of intrusions have already been stored in the database. This pattern is converted into a bit stream and is compared with the input data stream with the help of pattern matching unit. Snort performs protocol analysis, content searching, and content matching. Due to bitwise matching, if there is any intrusion , pattern mismatch occurs and early detection takes place. The detected pattern is given to the intrusion pattern in database.The decision making unit obtains the patterns and identifies the type of intrusion.

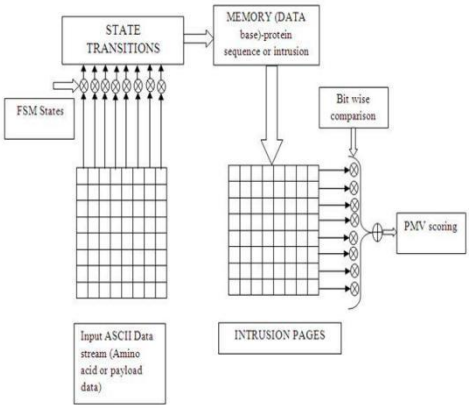

Fig 6 Computational Blocks

A Finite State Machine is any device that stores the status of something at a given time and can operate on input to change the status and/or cause an action or output to take place for any given change. The memory efficiency is analyzed by two ways. They are the number of states used in matching and the number of logical elements utilized after synthesis. As shown in Fig.6 , the bitwise comparison is done between the two bit streams and the type of intrusion pattern in identified and detected early by means of PMV (Partial Matching Vector) scoring.

EXPERIMENTALRESULTS

Some of the outputs taken using Model Sim software and Quartus II software are shown below. These results show the output for protein alignment with and without interleaving, Network Intrusion Detection Intrusion System, area utilization report, performance report, RTL viewer and other information.

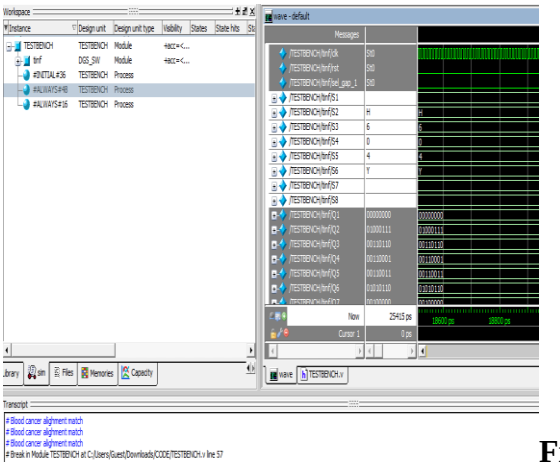

Fig 7 output for protein alignment

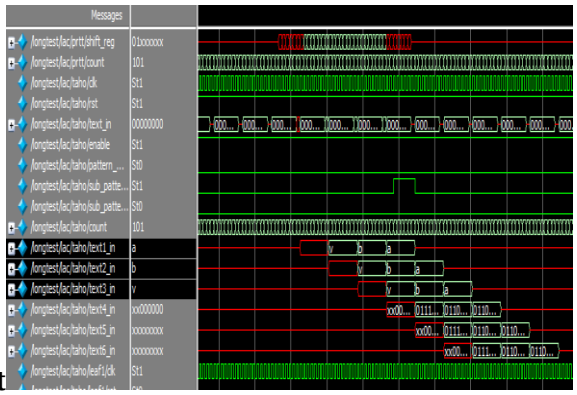

Fig8 simulation output for NIDS

It can be seen from the above figure 7 that the input protein sequence has been matched with the protein sequence that has been already stored in the database when it is run in MODELSIM. The protein sequence match here is shown as " H604Y " which is an alignment for Blood Cancer. The transcript window gives the name of the output when a match is detected.

It is seen from the above figure 8 that there has been a pattern match for a single virus pattern. Here "abv" has been fed as a virus pattern in the database and when the "abv" virus is present in the input it gets detected. The pattern match is shown by a binary 1 at the point where the "abv" virus pattern gets matched by the use of bit wise early detection method and is detected.

4.1Performance Report For Protein Alignment

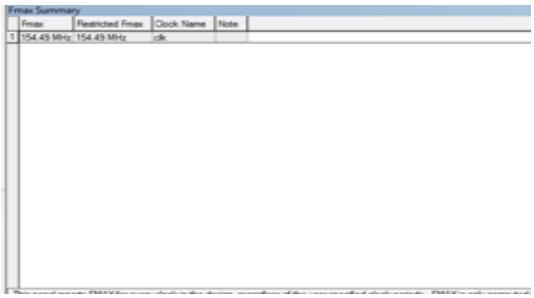

Fig9 Fmax summary without interleaving (slow corner)

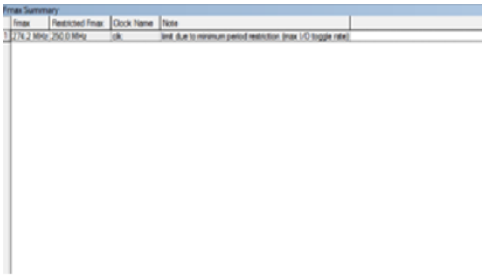

Fig10Fmax summary without interleaving (fast corner)

The above figure 9 shows the Fmax summary for protein alignment for slow corner without interleaving. Slow corner represents the operating frequency when the kit is being operated under worst conditions like dusty environment etc. which slows down the efficiency of the output rate when the environment is not under best suited conditions. The Fmax obtained for slow corner without interleaving here is 154.49 MHz.The above figure 10 shows the Fmax summary for protein alignment for fast corner without interleaving. Fast corner represents the operating frequency when the kit is being operated under best conditions like non-dusty environment etc. which speeeds up the efficiency of the output rate when the environment is under best suited conditions.The Fmax obtained for fast corner without interleaving here is 274.2 MHz.

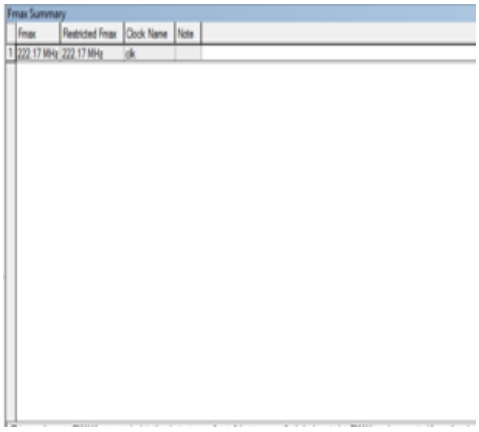

Fig 11Fmax summary with interleaving (slow corner)

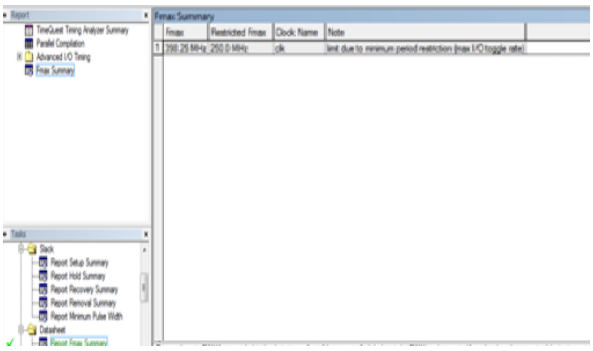

Fig 12 Fmax summary with interleaving (fast corner)

The above figure 11 shows the Fmax summary for protein alignment for slow corner with interleaving. Slow corner represents the operating frequency when the kit is being operated under worst conditions like dusty environment etc. which slows down the efficiency of the output rate when the environment is not under best suited conditions.The Fmax obtained for slow corner with interleaving here is 222.17 MHz.The above figure 12 shows the Fmax summary for protein alignment for fast corner with interleaving. Fast corner represents the operating frequency when the kit is being operated under best conditions like non-dusty environment etc. which speeeds up the efficiency of the output rate when the environment is under best suited conditions.The Fmax obtained for fast corner with interleaving here is 398.25 MHz.

4.2 Performance Report For NIDS

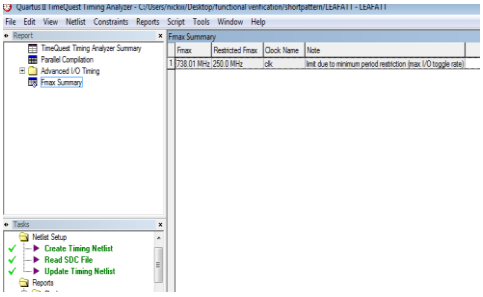

Fig 13 Fmax summary for NIDS (slow corner)

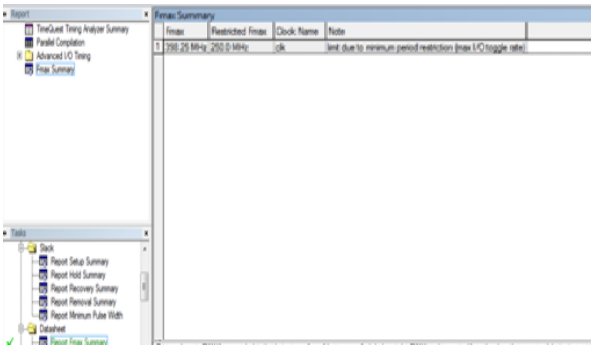

Fig 14 Fmax summary for NIDS (Fast corner)

It is seen from the above performance report for NIDS that by using Leaf Attached Algorithm the operating frequency has been increased. The above figure 13 shows the Fmax for slow corner. Slow corner represents the operating frequency when the kit is being operated under worst conditions like dusty environment etc. which slows down the efficiency of the output rate when the environment is not under best suited conditions. The Fmax here is shown as 422.83 MHz.It is seen from the above performance report for NIDS that by using Leaf Attached Algorithm the operating frequency has been increased. The above figure 14 shows the Fmax for fast corner. Fast corner represents the operating frequency when the kit is being operated under best conditions like non-dusty environment etc. which speeeds up the efficiency of the output rate when the environment is under best suited conditions. The Fmax here is shown as 738.01 MHz.

4.4RTL Viewer For Protein Alignment

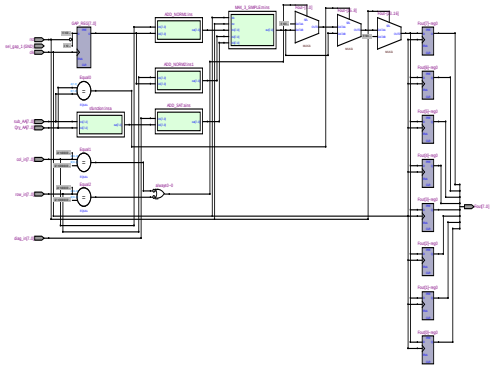

Fig 15 RTL Viewer for protein alignment without interleaving

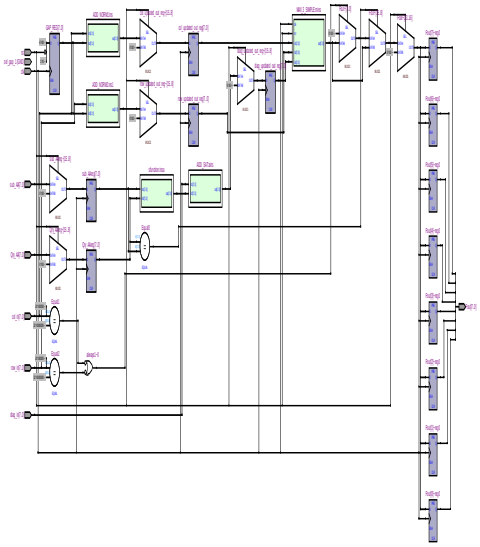

Fig 16 RTL Viewer for protein alignment with interleaving

figure 15 shows the RTL (Register Transfer Level) viewer for the protein alignment scheme. It shows the schematic view for protein alignment without interleaving. The logical elements used in designing the circuit are shown above.The figure 16 shows the RTL (Register Transfer Level) viewer for the protein alignment scheme. It shows the schematic view for protein alignment without interleaving. The logical elements used in designing the circuit are shown above.

4.5RTL Viewer For NIDS

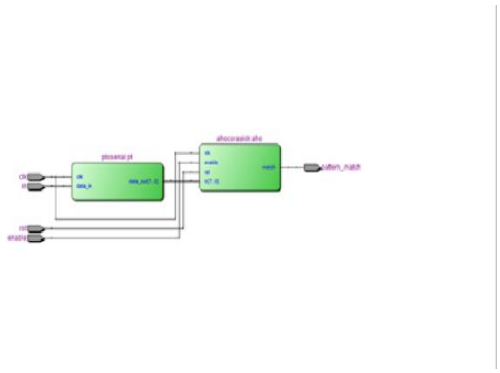

Fig 17 RTL viewer for NIDS

The above figure 17 shows the RTL (Register Transfer Level) viewer for NIDS. It shows the elements used for the hardware implementation of NIDS. The logical elements used in designing the circuit are shown above.

4.6Area Utilization Report For Protein Alignment

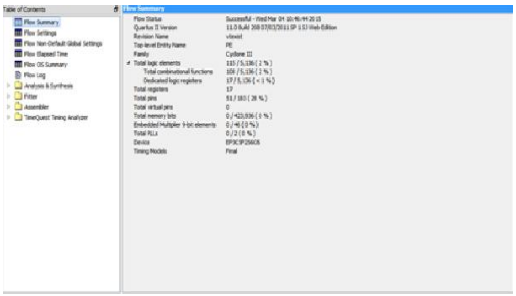

Fig 18 Flow summary report without interleaving

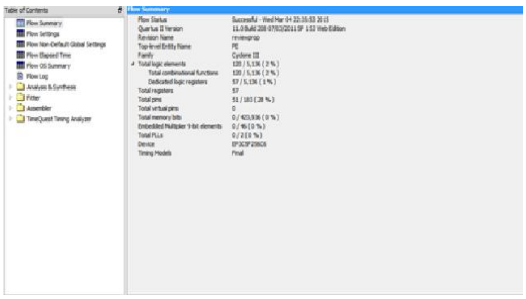

Fig19Flowsummary report with interleaving

The above figure 18 shows the report for area utilization for Protein alignment without interleaving. The total number of logical elements, registers, pins, combinationals functions used for implementation is shown.The figure

5.12 shows the report for area utilization for Protein alignment without interleaving. The total number of logical elements, registers, pins, combinationals functions used for implementation.

4.7Area Utilization For NIDS

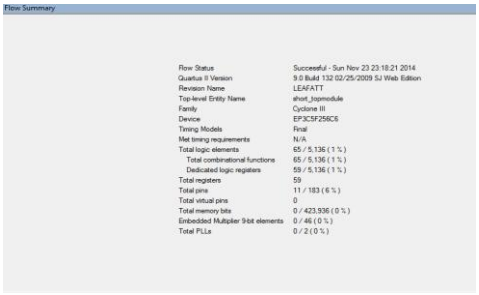

Fig 20 Flow summary report for NID

The above figure 20 shows the report of area utilization for NIDS. The total number of logical elements, registers, pins, combinationals functions used are shown above

STATE MACHINE VIEWER FOR NIDS

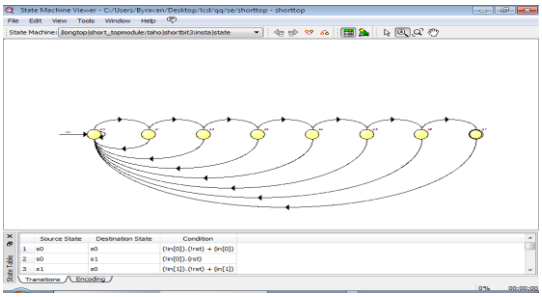

Fig 21 State machine report

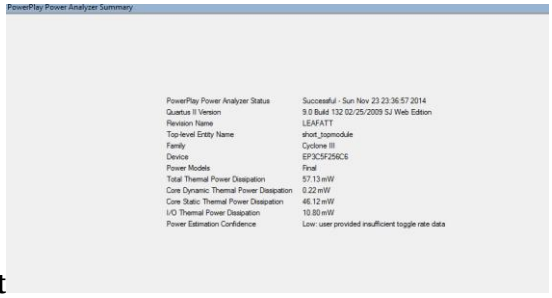

Fig 22 Power Dissipation Report

The above figure 21 shows the state machine viewer for NIDS. The total number of states for FSM is shown. The arrow marks indicate the present, previous and next states accordingly.The above figure 22 shows the report of power dissipationfor NIDS. The power models, total power dissipation, static power dissipation, dynamic power dissipation informations are shown.

Table 1 comparison table for pattern dividing methods

PATTERN DIVIDING METHODS	FMAX SUMMARY	NO.OF STATES	POWER DISSIPATION
Aho-Corasick	132.31MHz	14	66.36mw
Bit level short pattern	319.9MHz	7	64.25mw
Quotient Vector(LEAF based)	738.01MHz	3	57.13mw

Hence in proposed system throughput is increased by 350%-450% in case of quotient vector matching .The throughput is increased by 150% in case of short pattern matching .Experimental results shows that total memory requirements decreases on average by 50% in comparing with existing Aho-Corasick method as shown in the above table 1 and the corresponding graph is shown n fg 23..

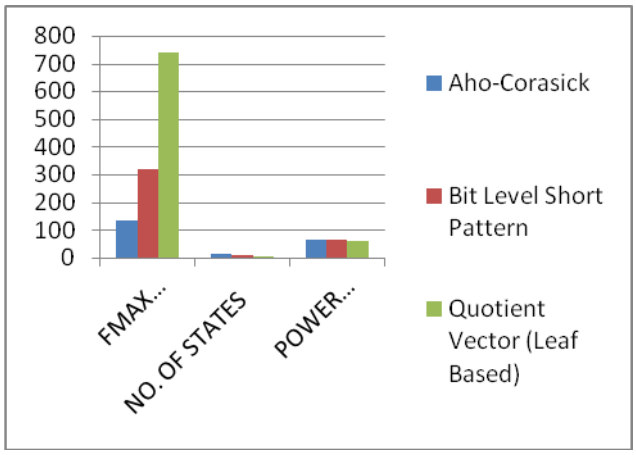

Fig23 Graph for pattern dividing methods

IV. 5CONCLUSION

Network Intrusion Detection System the FPGA hardware based NIDS proves to be more efficient than the software based NIDS since the number of users has been increasing rapidly. Based on the hardware implementation, it is concluded that hardware based approaches are more efficient in terms of speed, memory size and power consumption than software based approaches. The FPGA hardware based NIDS has been proposed to achieve higher rates using bit wise early

detection techniques. The architecture for a hardware based network intrusion detection system (NIDS) using pattern match processor was implemented. Now a days, hardware based string matching architectures are highly preferred in various real life applications. Based on the current implementation of the processor, incoming streams at rates of over 2 Gbps can be processed. This is more than sufficient to handle intrusion detection on current gigabit networks, and future enhancements by improvements in the VLSI process technology would enable the architecture to meet the demands of future 10 Gbps networks.

REFERENCES

[1] Aizat Azmi, Ahmad Amsyar Azman, Sallehuddin Ibrahim, and Mohd Amri Md Yunus, "Techniques In Advancing The Capabilities Of Various Nitrate Detection Methods: A Review", International Journal on Smart Sensing and Intelligent Systems., VOL. 10, NO. 2, June 2017, pp. 223-261.

[2] Tsugunosuke Sakai, Haruya Tamaki, Yosuke Ota, Ryohei Egusa, Shigenori Inagaki, Fusako Kusunoki, Masanori Sugimoto, Hiroshi Mizoguchi, "Eda-Based Estimation Of Visual Attention By Observation Of Eye Blink Frequency", International Journal on Smart Sensing and Intelligent Systems., VOL. 10, NO. 2, June 2017, pp. 296-307.

[3] Ismail Ben Abdallah, Yassine Bouteraa, and Chokri Rekik , "Design And Development Of 3d Printed Myoelctric Robotic Exoskeleton For Hand Rehabilitation", International Journal on Smart Sensing and Intelligent Systems., VOL. 10, NO. 2, June 2017, pp. 341-366.

[4] S. H. Teay, C. Batunlu and A. Albarbar, "Smart Sensing System For Enhanceing The Reliability Of Power Electronic Devices Used In Wind Turbines", International Journal on Smart Sensing and Intelligent Systems., VOL. 10, NO. 2, June 2017, pp. 407- 424

[5] SCihan Gercek, Djilali Kourtiche, Mustapha Nadi, Isabelle Magne, Pierre Schmitt, Martine Souques and Patrice Roth, "An In Vitro Cost-Effective Test Bench For Active Cardiac Implants, Reproducing Human Exposure To Electric Fields 50/60 Hz", International Journal on Smart Sensing and Intelligent Systems., VOL. 10, NO. 1, March 2017, pp. 1- 17

[6] P. Visconti, P. Primiceri, R. de Fazio and A. Lay Ekuakille, "A Solar-Powered White Led-Based Uv-Vis Spectrophotometric System Managed By Pc For Air Pollution Detection In Faraway And Unfriendly Locations", International Journal on Smart Sensing and Intelligent Systems., VOL. 10, NO. 1, March 2017, pp. 18- 49

[7] Samarendra Nath Sur, Rabindranath Bera and Bansibadan Maji, "Feedback Equalizer For Vehicular Channel", International Journal on Smart Sensing and Intelligent Systems., VOL. 10, NO. 1, March 2017, pp. 50- 68

[8] Yen-Hong A. Chen, Kai-Jan Lin and Yu-Chu M. Li, "Assessment To Effectiveness Of The New Early Streamer Emission Lightning Protection System", International Journal on Smart Sensing and Intelligent Systems., VOL. 10, NO. 1, March 2017, pp. 108- 123

[9] Iman Heidarpour Shahrezaei, Morteza Kazerooni and Mohsen Fallah, "A Total Quality Assessment Solution For Synthetic Aperture Radar Nlfm Waveform Generation And Evaluation In A Complex Random Media", International Journal on Smart Sensing and Intelligent Systems., VOL. 10, NO. 1, March 2017, pp. 174- 198

[10] P. Visconti ,R.Ferri, M.Pucciarelli and E.Venere, "Development And Characterization Of A Solar-Based Energy Harvesting And Power Management System For A Wsn Node Applied To Optimized Goods Transport And Storage", International Journal on Smart Sensing and Intelligent Systems., VOL. 9, NO. 4, December 2016 , pp. 1637- 1667

[11] YoumeiSong,Jianbo Li, Chenglong Li, Fushu Wang, "Social Popularity Based Routing In Delay Tolerant Networks", International Journal on Smart Sensing and Intelligent Systems., VOL. 9, NO. 4, December 2016 , pp. 1687- 1709

[12] Seifeddine Ben Warrad and OlfaBoubaker, "Full Order Unknown Inputs Observer For Multiple Time-Delay Systems", International Journal on Smart Sensing and Intelligent Systems., VOL. 9, NO. 4, December 2016 , pp. 1750- 1775

[13] Rajesh, M., and J. M. Gnanasekar. "Path observation-based physical routing protocol for wireless ad hoc networks." International Journal of Wireless and Mobile Computing 11.3 (2016): 244-257.

[14] Rajesh, M., and J. M. Gnanasekar. "Path Observation Based Physical Routing Protocol for Wireless Ad Hoc Networks." Wireless Personal Communications: 1-23.

[15] M. Rajesh., Traditional Courses into Online Moving Strategy. The Online Journal of Distance Education and e-Learning 4 (4), 19-63.

[16] Rajesh M and Gnanasekar J.M. Error- Lenient Algorithms for Connectivity Reinstallation in Wireless Adhoc Networks. International Journal of Advanced Engineering Technology; 7(1), pp 270-278, 2016.

[17] M. Rajesh and J.M. Gnanasekar., GCC over Heterogeneous Wireless Ad hoc Networks. Journal of Chemical and Pharmaceutical Sciences, 195-200.

[18] Rajesh, M and J.M. Gnanasekar., "Congestion Controls Using AODV Protocol Scheme For Wireless Ad-Hoc Network." Advances in Computer Science and Engineering 16 (1-2), 19.

[19] Rajesh M, Gnanasekar J. M. Sector Routing Protocol (SRP) in Ad-hoc Networks, Control Network and Complex Systems 5 (7), 1-4, 2015.

[20] Rajesh M, Gnanasekar J. M. Routing and Broadcast Development for Minimizing Transmission Interruption in Multi rate Wireless Mesh Networks using Directional Antennas, Innovative Systems Design and Engineering 6 (7), 30-42.

[21] F. Frattolillo, "Watermarking Protocol for Web Context," in iEEE Transactions on Information Forensics and Security,voI.2,no.3,sept , pp.350-363, 2007.

[22] H. Wang and C. Liao, "Compressed-Domain Fragile Watermarking Scheme for Distinguishing Tampers on Image Content or Watermark,“ in IEEE, pp.480-484, 2009.

[23] A .P. Singh, V. Kumar, S. S. Senger, and M. Wairiya, "Detection and Prevention of Phishing Attack using Dynamic Watermarking," inVInternational Conference on Advances in Information Technology and Mobile Communication ,vol. 147, pp 132-137,2011.