IJIRMET ISSN (Online): 2456-0448
International Journal of Innovative Research in Management, Engineering and Technology
Vol. 6, Issue 4, April 2021

# Reconfigrable Tcam For Payload Monitoring System

[1] S. Lakshmi,[2] D. Sathishkumar

[1] Associate Professor, Department of Electronics and Communication Engineering, Thirumalai Engineering College, Kilambi, Kanchipuram- 631551, Anna University, Chennai, India.

[2] Department of Electronics and Communication Engineering, Thirumalai Engineering College, Kilambi, Kanchipuram- 631551, Anna University, Chennai, India.

***Abstract: Software-defined networks (SDNs) are the future networks that enable the system to be more flexible and programmable using a centralized controller. Field-programmable gate arrays (FPGAs) serve as exemplary hardware to implement these adaptable networks. Ternary content-addressable memory (TCAM) is an essential part of every network to perform packet classification and forwarding, but they are missing in modern FPGAs. Researchers and FPGA vendors have proposed several designs to emulate TCAM using available memories on FPGA, but they are power inefficient due to the activating of entire circuitry in a single search operation. In this brief, we propose a novel power-aware reconfigurable FPGA-based TCAM architecture that enables only a portion of the hardware to perform the search operation. We performed an extensive design space exploration to find the optimal number of banks on Xilinx FPGAs, which provides the maximum power saving. Moreover, we propose a solution to bank overflow using backup CAM (BUC) to handle the overflowed CAM entries. The proposed TCAM improves the power consumption by 40% and maintains one-clock cycle update latency with no compromise on the throughput of the system.***

## 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.

## MOTIVATION:

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 thesis we focus on payload matching. Many different algorithms or combination of algorithms have been introduced and implemented in general purpose processors (GPP) for fast string matching, using mostly SNORT open source NIDS rule-set. However, intrusion detection systems running in 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 ASIC commercial products have been developed. These systems can support high throughput, but constitute a relatively expensive solution. On the other hand, 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. Several architectures have been proposed for FPGA-based NIDS, using regular expressions (NFAs/DFAs), CAM, discrete comparators, and approximate filtering techniques.

IJIRMET ISSN (Online): 2456-0448
International Journal of Innovative Research in Management, Engineering, and Technology
Vol. 6, Issue 4, April 2021

` Generally, the performance results of FPGA systems are promising, showing that FPGAs can be used to support the increasing needs for network security. FPGAs are flexible, reconfigurable, provide hardware speed, and therefore, are suitable for implementing such systems. On the other hand, there are several issues that should be faced. Large designs are complex and therefore hard to operate at high frequency. 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.

**SCOPE OF THIS THESIS:**

Since string matching is the most computationally intensive part of an NIDS, our proposed architectures exploit the benefits of FPGAs TCAM 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 thesis, we suggest solutions to maintain high performance and minimize area cost, show also how pattern matching designs can be updated in TCAM and partially or entirely changed, and advocate that brute-force 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 thesis proposes a pattern matching algorithm that reduces total memory requirements by sharing common infixes of target patterns using TCAM. 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 addition, throughput could decrease due to the modified state transition mechanism. The memory requirements for match vectors were reduced by relabeling states and eliminating the match vectors of non-output states. By sharing common infixes of target patterns or relabeling states and eliminating the match vectors of non-output states, the memory usage in the match vectors could be efficient. However, the variety of target pattern lengths is another serious problem in achieving regularity and scalability with low hardware cost. Each pattern consists of multiple character codes, where the number of character codes is defined as the pattern length. According to the rule sets, the distribution of pattern lengths could be different from each other. In addition, the variation of pattern lengths in each rule set is irregular. If target patterns are to be mapped onto multiple homogeneous string matchers, memory usage cannot be balanced without considering different pattern lengths.

In order to reduce the memory requirements of the CAM-based string matching engine, this proposes a memory-efficient parallel TCAM 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 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. 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 Clam AV, compared with several existing bit-split string matching approaches.

**DISSERTATION OUTLINE**

The rest of the thesis is organized as follows: the next chapter presents brief description of CAM in NIDS, offers some statistics about the patterns contained in a NIDS, and present some performance results of CAM-based NIDS.
Chapter 3 describes hardware-based NIDSs, previous FPGA-based TCAM pattern matching architectures, and commercial products.
In chapter 4, proposes a memory-efficient parallel string matching scheme using the hash table dividing approach and its hardware architecture for the pattern identification.
In chapter 5, hardware and software used, and
In chapter 6, we present the results and conclusions of this work and discuss future extensions

## 2. LITERATURE SURVEY

1.TITLE : “Resources Efficient FPGA-basedTCAM”.
AUTHOR : H. Mahmood.
YEAR : 2019
DESCRIPTION:
Ternary content-addressable memory (TCAM) is a high-speed searching device that searches the entire memory in parallel in

deterministic time, unlike random-access memory (RAM), which searches sequentially. A network router classifies and forwards a data packet with the aid of a TCAM that stores the routing data in a table. Field-programmable gate arrays (FPGAs), due to its hardware-like performance and software-like reconfigurability, are widely used in networking systems where TCAM is an essential component. TCAM is not included in modern FPGAs, which leads to the emulation of TCAM using available resources on FPGA. Several emulated TCAM designs are presented but they lack the efficient utilization of FPGA's hardware resources. In this paper, we present a novel TCAM architecture, the distributed RAM based TCAM (D-TCAM), using D-CAM as a building block. Similarly, by exploiting the LUT-flip-flip (LUT-FF) pair nature of Xilinx FPGAs, the proposed TCAM architecture improves throughput by 58.8% without any additional hardware cost.

**ADVANTAGES:**
Effectiveness and improved performance

**DISADVANTAGES**:
Energy and cost problem

**2.TITLE :** "An energy-efficient SRAM-based TCAM on FPGA,"
**AUTHOR :** Z. Ullah.
**YEAR :** 2018
**DESCRIPTION:**
Ternary content-addressable memories (TCAMs) are used to design high-speed search engines. TCAM is implemented on application-specific integrated circuit (native TCAMs) and field-programmable gate array (FPGA) (static random-access memory (SRAM)-based TCAMs) platforms but both have the drawback of high power consumption. This paper presents a pre-classifier-based architecture for an energy-efficient SRAM-based TCAM. The first classification stage divides the TCAM table into several sub-tables of balanced size. The second SRAM-based implementation stage maps each of the resultant TCAM sub-tables to a separate row of configured SRAM blocks in the architecture. The proposed architecture selectively activates at most one row of SRAM blocks for each incoming TCAM word. Compared with the existing SRAM-based TCAM designs on FPGAs, the proposed design consumes significantly reduced energy as it activates a part of SRAM memory used for lookup rather than the entire SRAM memory as in the previous schemes. We implemented the proposed approach sample designs of size 512 × 36 on Xilinx Virtex-6 FPGA. The experimental results showed that the proposed design achieved at least three times lower power consumption per performance than other SRAM-based TCAM architectures

**ADVANTAGES:**
Lower power consumption
**DISADVANTAGES:**
To minimize the energy cost.

## EXISTING SYSTEM

In existing works they proposed a memory efficient and ASCII approach for string matching on FPGAs. Here high throughput is achieved using FSM based binary search tree algorithm. Also they used on-chip RAM is used to store the patterns.
In Single-Character String Matching There are three major concerns for hardware implementation of the AC algorithm, namely, the cost (memory & logic elements), speed, and pattern set updates. Two categories of hardware architectures can be found in the literature, namely, hardwired-circuit architecture and memory-based architecture.
Massive parallelism can be utilized in hardwired-circuit implementations on FPGA where the string matching algorithm is modeled as a nondeterministic automaton (NFA) to minimize the number of transition edges. Nodes in the state transition graph are mapped to independent register bits, and state transitions are implemented by physical circuits that connect the register bits corresponding to the current states and the next states of transition edges.

## DRAWBACKS

- Pre processing step is used to segment the patterns into the Sequence of block. It affects the throughput rate. If number of characters in the patterns increases, complexity will also increase.
- Hardware complexity will be very high and dedicated memories are used to store the repeated characters in the different patterns.

- Memory efficiency is not achieved and due to sequential comparison of characters worst throughput will be very poor

## 3. PROPOSED BLOCK DIAGRAM

### INTRODUCTION

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.
Most of the previous researches on hardware regex detection methods are based on pcre patterns taken from Snort. A few major differences between the Clam AV format and the pcre format are listed below 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 Clam AV, compared with several existing bit-split string matching approaches.

### PROPOSED WORK

In this work we proposed hardware efficient VLSI architectures to detect the complex NIDS patterns based on the information reduction approach using Ternary content-addressable memory (TCAM). Here we preprocessed input bytes to transform the byte-oriented matching problem to a CAM-based matching problem. The input byte stream is converted into a token-stream using dedicated hardware units which can perform parallel computations for high throughput rate. A NIDS pattern contains one or more segments will be subdivided into multiple non-trivial tokens. Finally the token-stream is processed by a NFA-based aggregation unit to determine the virus to be found.
Here FEMEs are finite state machines (FSM) designed to extract field values within a header data of the application layer as shown in Figure 3.1. FEMEs search the header data sequentially. The extraction time is a function of the length of the header and number of bytes searched in a clock cycle.

### Power-efficient TCAM architecture

As shown in Fig. 3.1 the proposed power-efficient TCAM architecture consisting of five stages. Stage-1 is the pre classification and is used in the update and search operation of the TCAM. The n bits are the selector-bits that determine the bank (using a n:2n decoder), where the input key needs to be stored, or the search key needs to be searched. Only one of the 2n = B banks is selected to perform the searching or storing operation. Stage-2 is the clock gating (CG), which is a combination of AND gates to allow or stop the clock going to each bank. The clock signal to only one bank is allowed, and all others are stopped based on the pattern of selector-bits. Stage-3a consists of the actual memory, which is composed of FFs or slice registers (SRs). Only 25% of the total FFs are activated to perform searching or storing operation.

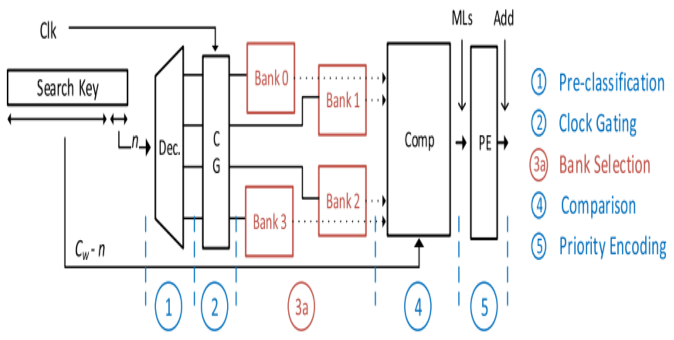


Figure: Proposed RPE-TCAM architecture
(Bank selection to save power consumption)

IJIRMET ISSN (Online): 2456-0448
International Journal of Innovative Research in Management, Engineering, and Technology
Vol. 6, Issue 4, April 2021

**Divided Patterns**

The remnant pattern Ri represents a suffix or residual sub-pattern of the target pattern Pi that succeeds the quotient vector of Pi. The match in a sub-pattern is encoded with a quotient index, which represents the unique index of the sub-pattern match. The remnant pattern Ri should be matched sequentially after the quotient vector Qi is matched.

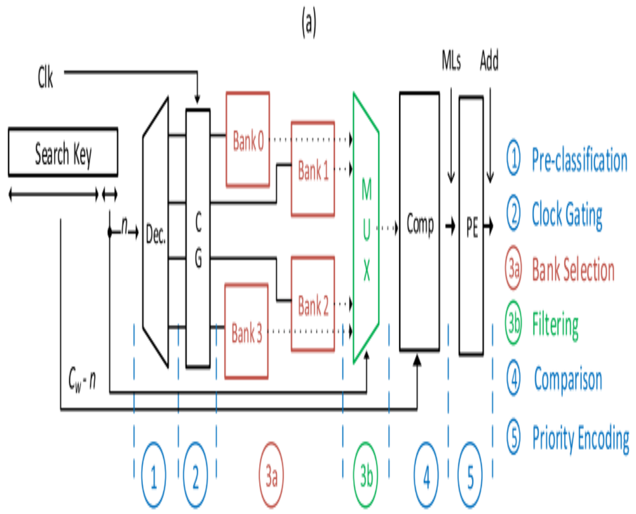


Figure: Proposed RPE-TCAM architecture

(Bank selection to save Hardware consumption)

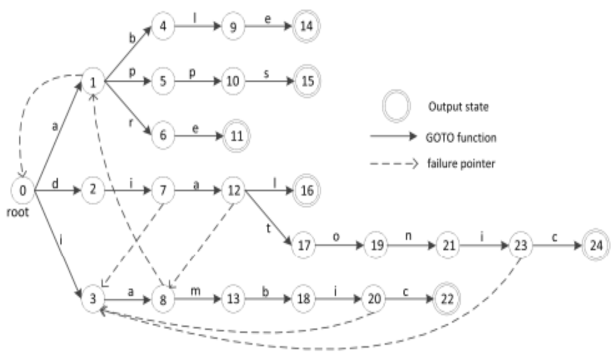
**NFA token detection unit**

The NFA regex detection method was initially developed to support intrusion detection. In principle NFA can also be used to detect the full virus patterns but the cost can be very high for long patterns and patterns with large exact-count and range-count. In this study, the virus signatures are divided into tokens, and the NFA is used to process tokens that cannot be detected by other methods.

The design of the NFA detection unit is refined for virus detection. In the Snort regex, the counting block corresponds to the repetition of a symbol by the given number of times.

**Figure 3.3: Finite state machines**

Design of the lookup tables for the string matching engine should satisfy three requirements: 1) constant lookup time; 2) good memory efficiency; and 3) flexible dynamic updates.



Design of the lookup tables for the string matching engine should satisfy three requirements: 1) constant lookup time; 2) good memory efficiency; and 3) flexible dynamic updates.

## RESULTS AND OUTPUTS

### OUTPUTS

### MODELSIM SIMULATION OUTPUT:

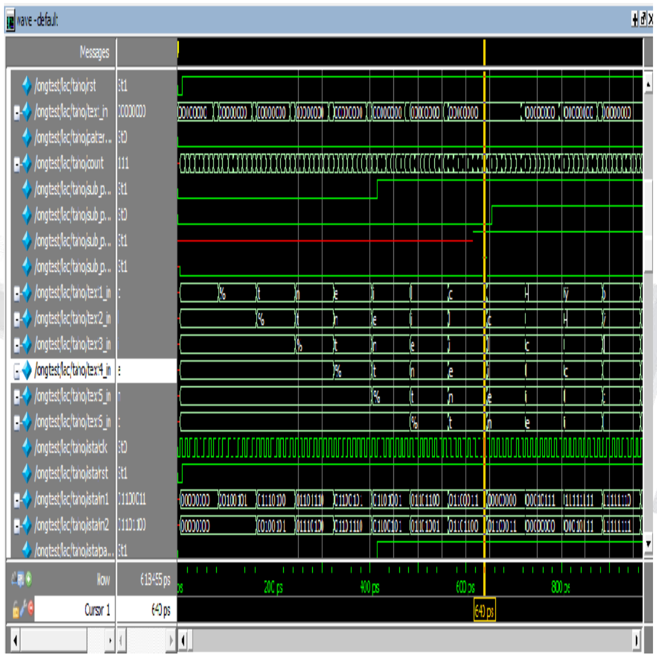
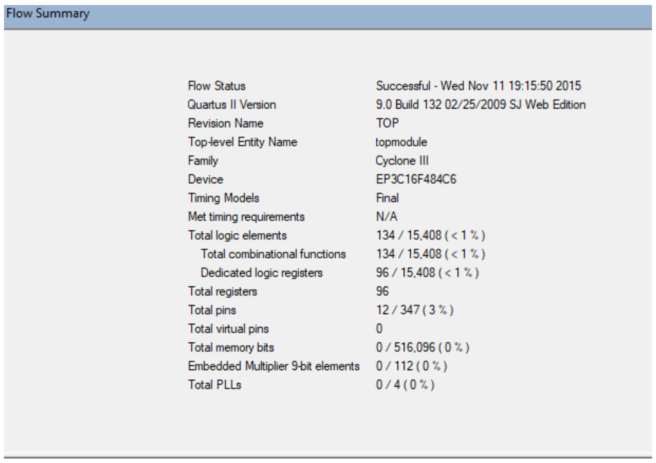


**Figure:** Model Sim simulation report

### AREA UTILIZATION REPORT:

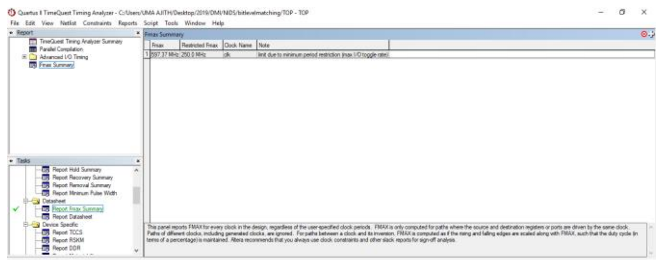


Flow Summary

| | |
|---|---|
| Flow Status | Successful - Wed Nov 11 19:15:50 2015 |
| Quartus II Version | 9.0 Build 132 02/25/2009 SJ Web Edition |
| Revision Name | TOP |
| Top-level Entity Name | topmodule |
| Family | Cyclone III |
| Device | EP3C16F484C6 |
| Timing Models | Final |
| Met timing requirements | N/A |
| Total logic elements | 134 / 15,408 ( < 1 % ) |
| Total combinational functions | 134 / 15,408 ( < 1 % ) |
| Dedicated logic registers | 96 / 15,408 ( < 1 % ) |
| Total registers | 96 |
| Total pins | 12 / 347 ( 3 % ) |
| Total virtual pins | 0 |
| Total memory bits | 0 / 516,096 ( 0 % ) |
| Embedded Multiplier 9-bit elements | 0 / 112 ( 0 % ) |
| Total PLLs | 0 / 4 ( 0 % ) |

Figure: Flow summary report.

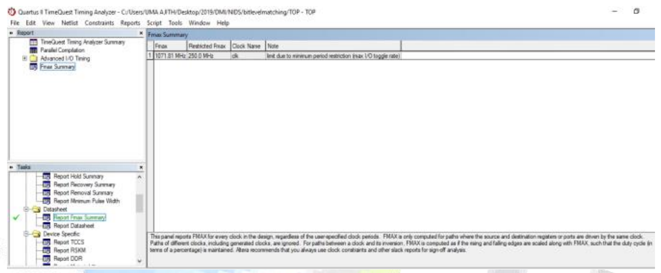
## PERFORMANCE REPORT:


Quartus II TimeQuest Timing Analyzer - C:/Users/UMA AJITH/Desktop/2019/DMI/NIDS/bitlevelmatching/TOP - TOP
File Edit View Netlist Constraints Reports Script Tools Window Help
Report
TimeQuest Timing Analyzer Summary
Parallel Compilation
Advanced I/O Timing
Fmax Summary
Tasks
Report Hold Summary
Report Recovery Summary
Report Removal Summary
Report Minimum Pulse Width
Datasheet
Report Fmax Summary
Report Datasheet
Device Specific
Report TCCS
Report RSKM
Report DDR
Fmax Summary

| | Fmax | Restricted Fmax | Clock Name | Note |
|---|---|---|---|---|
| 1 | 597.37 MHz | 250.0 MHz | clk | limit due to minimum period restriction (max I/O toggle rate) |

This panel reports FMAX for every clock in the design, regardless of the user-specified clock periods. FMAX is only computed for paths where the source and destination registers or ports are driven by the same clock. Paths of different clocks, including generated clocks, are ignored. For paths between a clock and its inversion, FMAX is computed as if the rising and falling edges are scaled along with FMAX, such that the duty cycle (in terms of a percentage) is maintained. Altera recommends that you always use clock constraints and other slack reports for sign-off analysis.

Figure: Fmax summary report (slow corner)

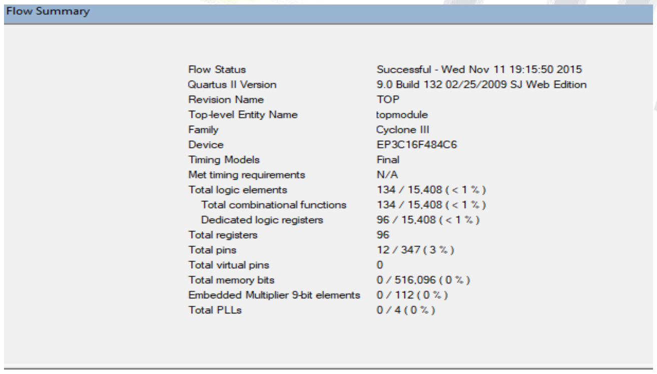

Quartus II TimeQuest Timing Analyzer - C:/Users/UMA AJITH/Desktop/2019/DMI/NIDS/bitlevelmatching/TOP - TOP
File Edit View Netlist Constraints Reports Script Tools Window Help
Report
TimeQuest Timing Analyzer Summary
Parallel Compilation
Advanced I/O Timing
Fmax Summary
Tasks
Report Hold Summary
Report Recovery Summary
Report Removal Summary
Report Minimum Pulse Width
Datasheet
Report Fmax Summary
Report Datasheet
Device Specific
Report TCCS
Report RSKM
Report DDR
Fmax Summary

| | Fmax | Restricted Fmax | Clock Name | Note |
|---|---|---|---|---|
| 1 | 1071.81 MHz | 250.0 MHz | clk | limit due to minimum period restriction (max I/O toggle rate) |

This panel reports FMAX for every clock in the design, regardless of the user-specified clock periods. FMAX is only computed for paths where the source and destination registers or ports are driven by the same clock. Paths of different clocks, including generated clocks, are ignored. For paths between a clock and its inversion, FMAX is computed as if the rising and falling edges are scaled along with FMAX, such that the duty cycle (in terms of a percentage) is maintained. Altera recommends that you always use clock constraints and other slack reports for sign-off analysis.

Figure: Fmax summary report (fast corner)

## POWER DISSIPATION REPORT

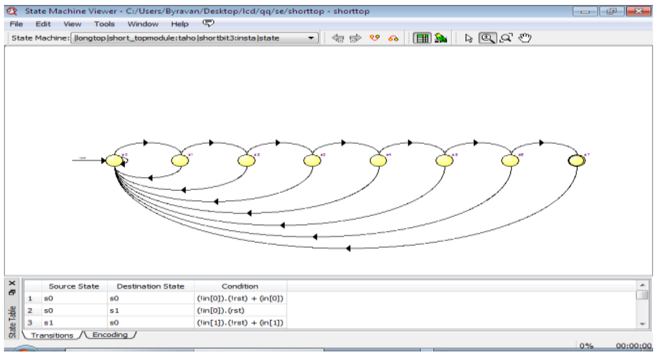


Flow Summary

| | |
|---|---|
| Flow Status | Successful - Wed Nov 11 19:15:50 2015 |
| Quartus II Version | 9.0 Build 132 02/25/2009 SJ Web Edition |
| Revision Name | TOP |
| Top-level Entity Name | topmodule |
| Family | Cyclone III |
| Device | EP3C16F484C6 |
| Timing Models | Final |
| Met timing requirements | N/A |
| Total logic elements | 134 / 15,408 ( < 1 % ) |
| Total combinational functions | 134 / 15,408 ( < 1 % ) |
| Dedicated logic registers | 96 / 15,408 ( < 1 % ) |
| Total registers | 96 |
| Total pins | 12 / 347 ( 3 % ) |
| Total virtual pins | 0 |
| Total memory bits | 0 / 516,096 ( 0 % ) |
| Embedded Multiplier 9-bit elements | 0 / 112 ( 0 % ) |
| Total PLLs | 0 / 4 ( 0 % ) |

Figure: power dissipation report

## STATE MACHINE VIEWER


State Machine Viewer - C:/Users/Byravan/Desktop/lcd/qq/se/shorttop - shorttop
File Edit View Tools Window Help
State Machine: |longtop|short_topmodule:taho|shortbit3:insta|state
s0
s1
s2
s3
s4
s5
s6
s7
State Table
Transitions
Encoding
0%
00:00:00

| | Source State | Destination State | Condition |
|---|---|---|---|
| 1 | s0 | s0 | (!in[0]).(!rst) + (in[0]) |
| 2 | s0 | s1 | (!in[0]).(rst) |
| 3 | s1 | s0 | (!in[1]).(!rst) + (in[1]) |

FIGURE: (a) ASCII based- state machine report

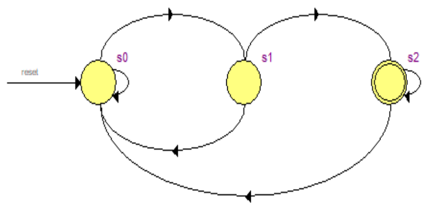


FIGURE: (b) Bit based- state machine report

**COMPARISION OF PATTERN DIVIDING USING TCAM METHODS**

| PATTERN DIVIDING METHODS | FMAX SUMMARY | AREA (Let used) | NO. OF STATES | POWER DISSIPATION |
|---|---|---|---|---|
| Aho-Corasick | 116.9 MHz | 156 | 14 | 64.44MW |
| Bit level short pattern | 597.37 MHz | 134 | 3 | 65.09MW |

## CONCLUSION

Here we verified the functionality proposed TCAM table based DFA-based parallel string matching scheme with minimized memory requirements for virus database. The problem of various pattern lengths with wild cards can be mitigated by dividing long target patterns into tokens with a fixed length.

The TCAM memory-efficient architectures were proposed for string matching which can reduce the total memory requirements. Considering the reduced memory requirements for the real rule sets, it is concluded that the proposed matching scheme is useful for reducing total memory requirements of parallel string matching engines.

## FUTURE ENHANCEMENT

- Have to prove the memory efficiency of modified FSM with shared memory for different patterns.
- Have to perform bit wise parallel string matching & its efficiency over previous methods.
- Have to prove the throughput rate & accuracy level of proposed methods with real time implementation.

## REFERENCES

1. A.V. Aho and M.J. Corasick, “Efficient String Matching: An Aid to Bibliographic Search,” Comm. ACM, vol. 18, no 6, pp. 333-340, 1975.
2. A. Ahmed, K. Park, and S. Baeg, “Resource-efficient SRAM-based Ternary content addressable memory,” IEEE Trans. Very Large Scale Integer. (VLSI) Syst., vol. 25, no.4, pp. 1583–1587, Apr. 2017.
3. M.Irfan and Z.Ullah, “G-AETCAM: Gate-based area-efficient ternary content-addressable memory on FPGA,” IEEE Access, vol. 5, pp. 20785–20790, 2017.
4. H. Kim, H. Hong, H.-S. Kim, and S. Kang, “A Memory-Efficient Parallel String Matching for Intrusion Detection Systems,” IEEE Comm. Letters, vol. 13, no. 12, pp. 1004-1006, Dec. 2009.
5. R. Karam, R. Puri, S. Ghosh, and S. Bhunia, “Emerging trends in design and applications of memory-based computing and content addressable memories,” Proc. IEEE, vol. 103, no. 8, pp. 1311– 1330, Aug. 2015.

6. K. Locke, “Parameterizable content-addressable memory,” Xilinx, San Jose, CA, USA, Appl. Note XAPP1151, 2011.
7. P.-C. Lin, Y.-D. Lin, T.-H. Lee, and Y.-C. Lai, “Using String Matching for Deep Packet Inspection,” IEEE Computer, vol. 41, no. 4, pp. 23-28, Apr. 2008.
8. C.-H. Lin, Y.-T. Tai, and S.-C. Chang, “Optimization of Pattern Matching Algorithm for Memory Based Architecture,” Proc. Third ACM/IEEE Symp. Architecture for Networking and Comm. Systems, pp. 11-16, 2007.
9. N. Mohan, W. Fung, D. Wright, and M. Sachdev, “Design techniques and test methodology for low-power TCAMs,” IEEE Trans. Very Large Scale Integer. (VLSI) Syst., vol. 14, no. 6, pp. 573–586, Apr. 2006.
10. K. Pagiamtzis and A. Sheikholeslami, “Content-addressable memory (CAM) circuits and architectures: A tutorial and survey,” IEEE J. Solid- State Circuits, vol. 41, no. 3, pp. 712–727, Mar. 2006.
11. Z. Qian and M. Margala, “Low power RAM-based hierarchical CAM on FPGA,” in Proc. Int. Conf. Reconfigurable Comput., Dec. 2014
12. M. S. Riazi, M. Samragh, and F. Koushanfar, “Camsure: Secure content addressable memory for approximate search,” ACM Trans. Embedded Comput. Syst., vol. 16, no. 5s, p. 136, 2017.
13. P. Reviriego, A. Ullah, and S. Pontarelli, “PR-TCAM: Efficient TCAM emulation on xilinx FPGAs using partial reconfiguration,” IEEE Trans. Very Large Scale Integer. (VLSI) Syst., vol. 27, no. 8, pp. 1952-56.
14. M. Somasundaram, “Circuits to generate a sequential index for an input number in a pre-defined list of numbers,” U.S. Patent 7 155 563, Dec. 26, 2006.
15. L. Tan and T. Sherwood, “A High Throughput String Matching Architecture for Intrusion Detection and Prevention,” Proc. 32nd IEEE/ACM Int’l Symp. Computer Architecture, pp. 112-122, 2005.