ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 5, May 2015

FPGA Implementation of Fully Synchronized Multi-Rate FSM Based Memory Efficient NID System

Ms. Flemin Nicklet [1], Dr. S. Prakash [2]
PG scholar, Department of ECE, Jerusalem College of Engineering, Pallikkaranai, Chennai[1]
Professor, Department of ECE, Jerusalem College of Engineering, Pallikkaranai, Chennai[2]

Abstract: High speed and always-on network access is becoming common place around the world, creating a demand for increased network security. Network Intrusion Detection Systems (NIDS) attempt to detect and prevent attacks from the network using pattern-matching rules in a way similar to anti-virus software. For the low-cost hardware-based intrusion detection systems, this project work proposed a memory-efficient parallel string matching scheme. In order to reduce the number of state transitions, the finite state machine tiles in a string matcher adopt bit-level input symbols. Long target patterns are divided into sub patterns with a fixed length; deterministic finite automata are built with the sub patterns. Using the pattern dividing, the variety of target pattern lengths can be mitigated, so that memory usage in homogeneous string matchers can be efficient. In order to identify each original long pattern being divided, a two-stage sequential matching scheme is proposed for the successive matches with sub-patterns. As an extended work, here adding the all digital Phase locked loop (ADPLL) for the multi-rate clock synchronization. The input intrusions are divided into pages and each of the incoming pages will be in variable rate. Hence it's important to do synchronization as the clock mismatch will leads to 3 major cases. An adaptive reconfigurable PLL based clock divider is used for variable rate pattern match and gated clock system through early mismatch detection is verified through exhaustive test bench simulation. And finally FPGA implementation was carried out using ALTERA CYCLONE family FPGA.

Keywords: Network Intrusion Detection System (NIDS), Deterministic Finite Automata (DFA), Pattern matching, Pattern division.

I. INTRODUCTION

Most digital circuits designed and fabricated today are "synchronous". In essence, they are based on two fundamental assumptions that greatly simplify their design: (1) all signals are binary, and (2) all components share a common and discrete notion of time, as defined by a clock signal distributed throughout the circuit. Asynchronous circuits are fundamentally different [1-3]. For synchronous circuits, power gating can be implemented in the fine-grain or coarse-grain manner. The fine-grain power gating approach has more opportunities to reduce leakage at run-time than the coarse-grain power gating approach [4]. However, there are several design issues associated with incorporating fine-grain power-gating in synchronous circuits. First, fine-grain power-gating needs significant buffering and routing resources to distribute the sleep control signal to all the cells in the synchronous system [5].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. 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.

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.

II. NIDS For Packet Inspection

A. Software-based string match

Several string matching algorithms have been

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 5, May 2015

recently proposed in NIDS specially for SNORT's open source NIDS. First versions of SNORT used brute-force pattern matching, which was very slow, making clear that using a more efficient string matching algorithm, would improve performance. The first implementations that improved SNORT used the Boyer-Moore algorithm, and later a *"2-dimentional linked list with recursive node walking"*. This implementation improved SNORT performance 200-500%. The Boyer-Moore algorithm is one of the most well-known algorithms that use two heuristics to reduce the number of comparisons. It first aligns the pattern and the incoming data (text), the comparison begins from the right-most character, and in case of mismatch the text is properly shifted.

However, the Boyer-Moore algorithm compares each pattern independently against the incoming data, and hence substrings repeated in more than one patterns are compared multiple times. Another implementation of SNORT uses Wu-Mander multi-pattern matching algorithm. The MWM algorithm performs a hash on 2-character prefix of the input data, in order to index into a group of patterns. This SNORT implementation is much faster than previous ones.

The topic of this thesis is the analysis and design of ADPLL based multi rate NIDS systems, something that is currently not achievable.

B. Hardware based string matching

Software-based Intrusion Detection Systems can only support modest throughput. Hardware based NIDS system can easily adapt in NIDS application needs, achieving better performance with reasonable cost. Many ASIC intrusion detection systems usually store their rules using large memory blocks, and examine incoming packets in integrated processing engines. Generally, ASICs programmable security co-processors are expensive, complicated, and although they can support higher throughput compared to GPP, they do not achieve impressive performance. The memory blocks that store the NIDS rules are re-loaded, whenever an updated rule-set is available.

The most common technique for pattern matching in ASIC intrusion detection systems is the use of regular expressions. Updating the rule-set is not a trivial procedure, since the system must be able to support a variation of rules, with sometimes complex syntax, and special features. On the other hand, FPGAs are more suitable; because they are reconfigurable they provide hardware speed and exploit parallelism.

C. Algorithms in misuse detection

- Simple string matching
- State Machine Matching

Simple string matching:

– The Boyer-Moore algorithm uses two different heuristics for pattern match. The first heuristic, referred to as the bad character heuristic, works as follows: if the search pattern contains a malicious character the pattern is shifted out as a mismatching character and it is aligned with the rightmost position at which it appears inside the pattern. The second heuristic, works as follows: if a mismatch is found in the middle of the malicious character, the search pattern is shifted to the next occurrence of the matched suffix in the pattern. Both heuristics can lead to a shift distance of m.

State Machine Matching:

Aho-Corasick String Matching Automaton for a given finite set M of patterns is a (deterministic) finite automaton G accepting the set of all words containing a word of P .Formation about where to jump from one state to another in FSM machine for each character in pattern M It just follows the string to be matched making transitions via FSM states, the transition function which tells which state to jump for each patterns. Whenever we reach a final FSM state a match is reported by the engine.

D. NFA/DFA implementation at hardware level

The most common approach is the regular expressions matching, implemented using Finite Automata (NFAs or DFAs). Regular expressions produce designs with low cost, but at a modest throughput. The basic idea of is to generate regular expressions for every pattern or group of patterns, and implement them with N/DFA. A regular expression is a pattern that describes one or more strings. It consists of characters, which are considered as regular expressions, and *meta-characters* (*j*,*, (,),) that have special use. Regular expression syntax includes the following rules:

- ab, a followed by b.
- a*j*b, a or b.
- a*, zero, one, or more a.

There are also other meta-characters that lead to more complex syntaxes, and more efficient regular expressions.

Non-deterministic Finite Automata (NFAs) are direct graphs, their nodes are states, and their edges are labeled with a character. There is an *initial* and one or more *final* states. On the other hand, Deterministic Finite Automata (DFAs) are similar to NFAs, but they do not include characters. Additionally, only one state can be active

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 5, May 2015

in DFAs, while NFAs can have more than one active state. Generally, NFAs are simpler and easier to design by just listing all stored patterns. On the other hand, DFAs are easier to implement, because there are no choices to be considered, since there are no characters and there is only one state active. Theoretically, DFA can be exponentially larger than NFA, but in practice often DFAs have, as compared to NFAs, a similar number of states (O *(n)* states, where *n* is the number of expression characters). The use of parallelism (processing multiple bytes or characters per cycle) is in general difficult in finite-automata implementations that are built with the implicit assumption that the input is checked one byte at a time. One proposed solution to this problem is the usage of packet-level parallelism where multiple pattern matching subsystems operating in parallel can process more than one packet.

E. Clock phase violation

In any synchronous digital system clock signal will be used for time reference for data propagation. Due to clock skew phase of clock signal will be varying from one circuit to another one. This will lead clock mismatch problem. Sometimes it will lead Meta stability problem. In our proposed NIDS system this problem arises in three levels as follows

Clock mismatch in ASCII conversion module: With this violation leads failure to get incoming patterns for pattern match.

Clock mismatch in string matching conversion module: With this violation leads failure to match incoming patterns with data base.

Clock mismatch in pages: Even with incoming intrusions match with data base pattern match module can't assert any signals.

III. HIGHLY SYNCHRONIZED NIDS SYSTEM

A. Phase Locked Loop

The common approach for clock processing, such as de-skewing and frequency multiplication, is based on analog Phase-Locked-Loops (PLLs) or DLLs. In recent years, the interest in all-digital DLLs (ADDLL) increased, and several papers reporting such implementations have been published. The Phase Locked Loop (PLL) is one of the most ubiquitous electronic components found in almost every electronic device

B. Feedback Divider Component

As mentioned earlier the feedback frequency divider (÷N) is used in frequency synthesis to produce a PLL output signal frequency FV that is some multiple, N, of the reference signal, FR. There are two common classes of feedback divider: the integer-N and fractional-N dividers. The integer-N type divides the frequency by an integer divide ratio, while the fractional-N divider provides a fractional divide ratio.

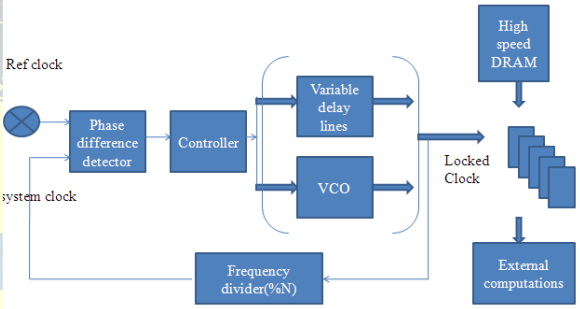

Fig.1. ADLL block diagram

C. Variable Delay Line

The register-controlled variable delay line consists of a combination of inverters and capacitors or a NAND gate chain [2], [3]. The unit delay is determined by the logic gate delay, such as that from an inverter or NAND. The unit delay of a basic logic gate is too large to meet the skew design specification. Therefore, a fine delay control scheme is employed. The variable delay line consists of both coarse and fine delay lines. In this case, in order to move the switching compensation from a coarse delay line to a fine delay line, the dual delay line is adopted.

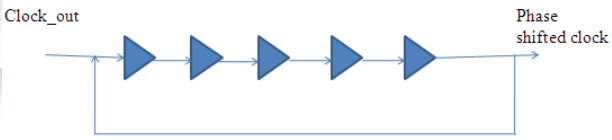

Fig.2. Delay lines

C. Pattern identification

For each target pattern, a unique identification index should be provided in order to distinguish its pattern match from other pattern matches. If multiple target patterns are mapped onto a DFA, it is possible that a target pattern can be a sub-pattern of other target patterns. For example, it is assumed that four target patterns {"abc," "abcd," "ac," "bcd"} are mapped on a DFA, where target pattern lengths

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 5, May 2015

range from 2 to 4. The fourth target pattern is a suffix of the second target pattern. If the second target pattern is matched, the fourth target pattern is always matched, but not vice versa. We let a target pattern P_i be a suffix of another target pattern P_j for P_i ¼6 P_j. If different identification indexes are provided for the matches with P_i and P_j, respectively, P_i is explicitly identified when P_j is matched.

If only the identification index for P_j is provided, P_i is implicitly identified. In this case, only the target pattern with the longest prefix from the initial state has its own identification index in the implicit identification; therefore, users could detect matches with P_i after analyzing the identification for P_j with extra effort. Let's assume that only k sub-patterns of a quotient vector, Q_i ¼ fSP_{i1}; SP_{i2}; . . . ; SP_{ik}g, are mapped onto a DFA. In this case, the number of output states for the sub-patterns is equal to or smaller than k.

D. Bit based leaf attaching algorithm

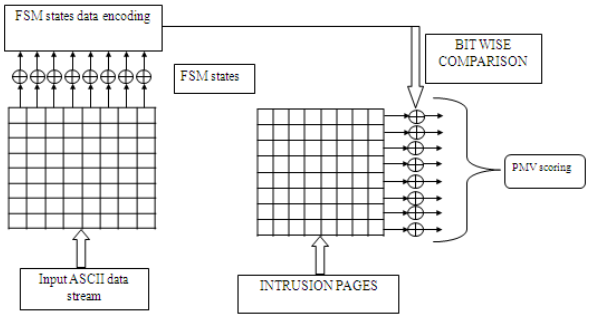

Fig.3. Proposed bit based comparison architecture

Tree search algorithm is a good choice for a string matching engine as the lookup latency does not depend on the length of the pattern, but on the number of patterns in the dictionary. In the case of the tree search, the latency is proportional to log of the number of patterns. However, in order to use tree search algorithms, the given set of patterns needs to be processed to eliminate the overlap between patterns. Leaf-pushing can be used to eliminate overlap.

E. Gated-Clock Design

To reduce power consumption in NIDS system a set of strategies termed dynamic power management (DPM) is often used. The DPMs strategy consists in disabling some of the FSM circuits by performing early detection operations during bit wise matching form LSB to MSB, thus reducing power consumption.

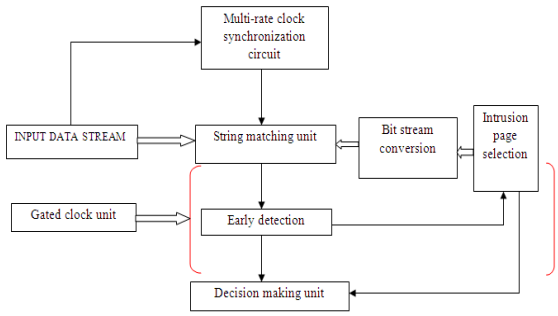

Fig.4. Early detection strategy

At circuit level, this strategy is applied by the so-called "gated clock" approach which disables the clock of FFs More specifically, for FFs without an enable signal, which is our case; we can adopt the early detection as it is shown in Fig. 4.

IV. RESULTS

Here ADPLL based synchronous NIDS system is designed with 5 page pipelined architecture using and its parameter optimization is proved over asynchronous mechanism. The leaf attaching -based algorithm is highly suitable for VLSI implementation, since it is built using prefix and parallel computations. ADPLL system components are designed and its functionality is verified using Modelsim simulation. We used to prove the efficiency of pipelined architecture through EDA tool synthesis to realize the low hardware Complexity VLSI implementation. Moreover, the computational accuracy can be selected based on the trade-off between the hardware Complexity and approximation error. In addition, since our proposed algorithm has uniform scaling factor, it is also suitable for scaled FPGA implementation. In order to evaluate the practicality of implementing NIDS system on a single FPGA, a typical top module was designed as a cascade of NIDS sections, and was implemented on an ALTERA CYCLONE Family FPGA through QUARTUS II EDA tool based hardware synthesis.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 5, May 2015

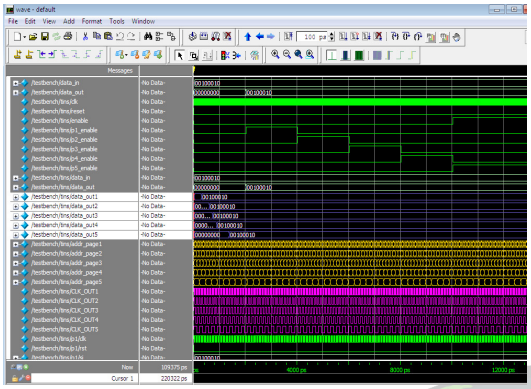

Fig.5. variable rate synchronized pattern match output

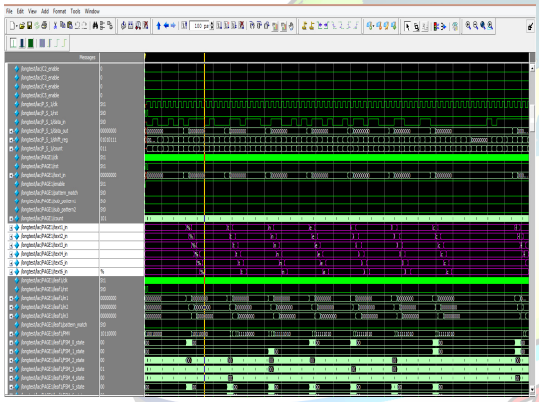

Fig.6. Clock Mismatch simulation report

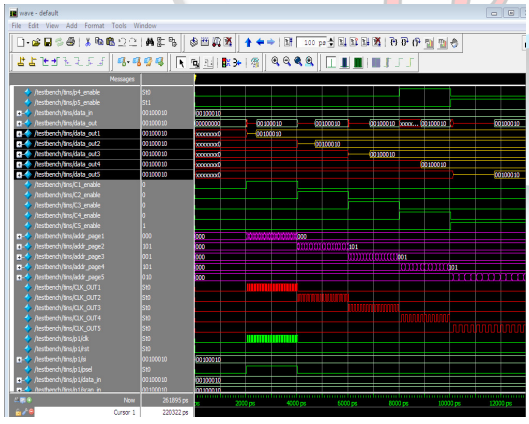

Fig.7. Gated latch multi rate NIDS system

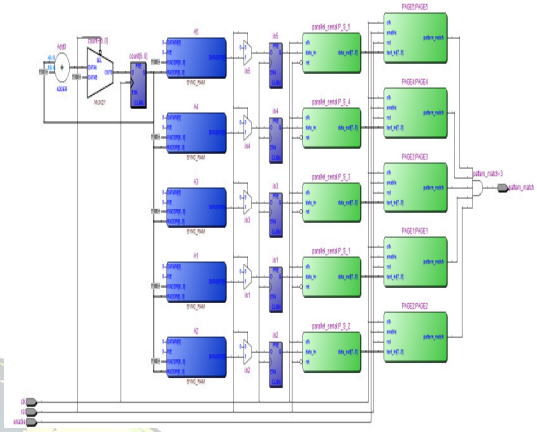

Fig.8. NIDS system RTL hardware view

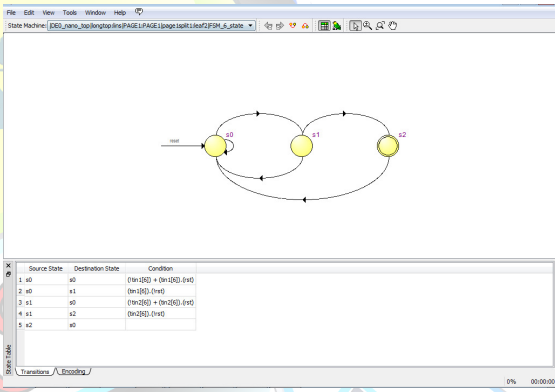

Fig.9. FSM state machine viewer

V. CONCLUSION

This paper has described an approach to the implementation of multi rate NIDS based on field programmable gate arrays (FPGAs). The proposed ADPLL-based parallel string matching scheme minimizes total memory requirements with variable rate pattern match. The problem of variable data rate can be mitigated by dividing source clock into multi rate with a phase locked. Considering the unnecessary transition that occurs with practical rule sets, it is concluded that the proposed CLOCK GATED ADPLL string matching scheme is useful for reducing total power requirements of parallel string matching engines. Because of the programmability of this technology, the method proposed in this paper can be

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 5, May 2015

extended to provide a variety of other high performance real time pattern match realizations.

REFERENCES

[1]. 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.

[2]. Snort, Ver.2.8, Network Intrusion Detection System, http:// www.snort.org., 2011.

[3]. Clam Antivirus, Ver.0.95.3. http://www.clamav.net., 2011.

[4]. 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.

[5]. Hoang Le, member, IEEE, and Viktor K. Prasanna, fellow, IEEE, "A Memory Efficient and Modular Approach for Large-Scale String Pattern Matching",vol. 62, no. 5, May 2013.

[6]. 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.

[7]. Haoyu Song and John W. Lockwood, Dept. CSE, Washington University, "Efficient Packet Classification for Network Intrusion Detection using FPGA", 2005

[8]. F. Yu, Z. Chen, Y. Diao, T.V. Lakshman, and R.H. Katz, "Fast and Memory-Efficient Regular Expression Matching for Deep Packet Inspection," Proc. Second ACM/IEEE Symp. Architecture for Networking and Comm. Systems, pp. 93-102, 2006.

[9]. 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.

[10]. 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.

[11]. L. Tan, B. Brotherton, and T. Sherwood, "Bit-Split String-Matching Engines for Intrusion Detection and Prevention," ACM Trans. Architecture and Code Optimization, vol. 3, no. 1, pp. 3-34, Mar. 2006.

[12]. Y.-J. Jeon, J.-H. Lee, H.-C. Lee, K.-W. Jin, K.-S. Min, J.-Y. Y. Chung, and H.-J. J. Park, "A 66–333-MHz 12-mW register-controlled DLL with a single delay line and adaptive-duty-cycle clock dividers for production DDR SDRAMs," *IEEE J. Solid-State Circuits*, vol. 39, no. 11, pp. 2087–2092, Nov. 2004.

[13]. K.-H. Kim, J.-B. Lee, W.-J. Lee, B.-H. Jeong, G.-H. Cho, J.-S. Lee, G.-S. Byun, C. Kim, Y.-H. Jun, and S.-I. Cho, "A 1.4 Gb/s DLL using 2nd order charge-pump scheme with low phase/duty error for high-speed DRAM application," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2004, pp. 213–214.

[14]. T. Hamamoto, K. Furutani, T. Kubo, S. Kawasaki, H. Iga, T. Kono, Y. Konishi, and T. Yoshihara, "A 667-Mb/s operating digital DLL architecture for 512-Mb DDR," *IEEE J. Solid-State Circuits*, vol. 39, no. 1, pp. 194–206, Jan. 2004.