

Efficient Header Classification Architecture for Network Intrusion Detection

WEN-JYI HWANG AND CHIEN-MIN OU*

*Department of Computer Science and Information Engineering
National Taiwan Normal University
Taipei, 117 Taiwan*

**Department of Electronics Engineering
Ching Yun University
Chungli, 320 Taiwan*

In this paper, an efficient FPGA-based header classification circuit is proposed for network intrusion detection system (NIDS). The circuit is based on simple shift registers and symbol encoders for the fast packet header classification in hardware. As compared with related work, experimental results show that the proposed work achieves higher throughput and less hardware resource in the FPGA implementations of NIDS systems.

Keywords: network intrusion detection system, FPGA implementation, pattern matching, header classification architecture, shift-or algorithm

1. INTRODUCTION

With the rapid increase in malicious attacks on networks, there has been an increased awareness amongst network administrators to deploy tools that protect them from external attacks. A network intrusion detection system (NIDS) is one of the tools available to help in creating a secure network infrastructure. It monitors network traffic for suspicious data patterns and activities, and informs system administrators when malicious traffic is detected so that proper actions may be taken. Many NIDSs such as SNORT [12] classify packets based on the header fields and the strings (signatures) in the packet content. The computational complexity of NIDSs is high because of the requirement of the packet header and signature matching during their detection processes.

The SNORT system running on general purpose processors may only achieve up to 60 Mbps [7] throughput because of the high computational complexity. Since these systems do not operate at line speed, some malicious traffic can be dropped and thus may not be detected. One way to accelerate the speed for intrusion detection is to utilize the FPGA for the implementation of SNORT systems. Because the NIDS rules do not change frequently, the cost for FPGA implementations may not be high as compared with their software-based counterparts. Moreover, the hardware implementation can exploit parallelism for string matching so that the throughput of NIDSs can be increased.

Many FPGA-based SNORT architectures scan only signatures in the payload [1, 3, 4, 9, 11], which may result in high false alarm rates. Moreover, although a number of header classification architectures have been proposed [5, 10], a common drawback of these architectures is the high area cost for the FPGA implementation. This may signifi-

Received December 12, 2007; revised June 10 & September 22, 2008; accepted October 16, 2008.
Communicated by Tzong-Chen Wu.

cantly increase the consumption of logic elements, logic modules, and/or memory bits as compared with the architectures matching only the signatures in the payload.

The objective of this paper is to present a novel header classification architecture for FPGA implementation of NIDSs achieving both high throughput and low area cost. The proposed architecture is based on the shift-or algorithm for exact string matching [2]. The shift-or algorithm is an effective software approach for pattern matching because of its simplicity and flexibility. However, it may not perform well when the pattern size is larger than the computer word size, which is the case for the 5-tuple (source IP address, destination IP address, protocol, source port, destination port) header of a SNORT rule. Accordingly, the software implementation of shift-or algorithm may not be suited for header matching of SNORT systems.

The proposed architecture uses only simple shift registers and symbol encoders for the hardware implementation of shift-or algorithm. The architecture is more flexible on the pattern size as compared with its software counterpart. In the architecture, each SNORT pattern is only associated with a shift register for pattern comparison, which are designed in accordance with the pattern size. In addition, different rules also share the same symbol encoder to reduce the area cost for FPGA implementation.

Note that, source port and destination port of some SNORT rules are defined as ranges. Direct applications of the proposed shift-or architecture to match ranges may be difficult. Although a range can be converted into a series of prefixes, this process may significantly expand the rule set size. In our design, comparators are employed for matching the range. Moreover, the port range matching process is performed in parallel with other matching processes based on the shift-or architectures.

Because of its simplicity, the architecture may operate at a higher clock rate as compared with other implementations. The area cost may also be lower than the existing designs. Moreover, although the proposed architecture in its simplest form only processes one character at a time, the architecture can be extended to further enhance the throughput of the circuit. Multiple characters can be scanned and processed in one cycle at the expense of slight increase in area cost.

2. PRELIMINARIES

Suppose we are searching for a *pattern* $P = p_1p_2 \dots p_m$ inside a *text* $T = t_1t_2 \dots t_n$, where $n \gg m$. Every character of P and T belongs to the same alphabet $\Sigma = \{s_1, \dots, s_{|\Sigma|}\}$.

Let R_j be a bit vector containing information about all matches of the prefixes of P that end at j . The vector contains $m + 1$ elements $R_j[i]$, $i = 0, \dots, m$, where $R_j[i] = 0$ if the first i characters of the pattern P match exactly the last i characters up to j in the text (*i.e.*, $p_1p_2 \dots p_i = t_{j-i+1}t_{j-i+2} \dots t_j$). The transition from R_j to R_{j+1} is performed by the recurrence [2]:

$$R_{j+1}[i] = \begin{cases} 0, & \text{if } R_j[i-1] = 0 \text{ and } p_i = t_{j+1}, \\ 1, & \text{otherwise.} \end{cases} \quad (1)$$

where the initial conditions for the recurrence are given by $R_0[i] = 1$, $i = 1, \dots, m$, and $R_j[0] = 0$, $j = 0, \dots, m$. The recurrence can be implemented by the simple shift and OR

operations. Let S_k be a bit vector (containing m elements) associated with each symbol $s_k \in \Sigma$, where the k th element $S_k[i]$ is given by

$$S_k[i] = \begin{cases} 0, & \text{if } s_k = p_i, \\ 1, & \text{otherwise.} \end{cases} \quad (2)$$

Assume $t_{j+1} = s_c$. Based on Eq. (2), the recurrence shown in Eq. (1) can then be rewritten as

$$R_{j+1}[i] = R_j[i - 1] \text{ OR } S_c[i], \quad i = 1, \dots, m. \quad (3)$$

Fig. 1 shows an example of the exact string matching based on the shift-or algorithm, where $P = aab$, $T = acaab$, and $\Sigma = \{a, b, c\}$. The bit vector associated with each s_k , which is determined by Eq. (2), is given in Fig. 1 (a). Given S_c and R_{j-1} the R_j can be computed by Eq. (3), as shown in Fig. 1 (b). Note that, when $j = 5$, it can be found from Fig. 1 (b) that $R_j[3] = 0$. Therefore, one occurrence of P is found when $j = 5$.

S_k	a	b	c
$i=1$	0	1	1
$i=2$	0	1	1
$i=3$	1	0	1

j	0	1	2	3	4	5
i	R_j	S_c	R_j	S_c	R_j	S_c
0	0	0	0	0	0	0
1	1	0	0	1	1	0
2	1	0	1	1	1	0
3	1	1	1	1	1	1

(a) The bit vector S_k associated with each symbol $s_k \in \Sigma = \{a, b, c\}$ for the pattern P .

(b) The bit vector R_j for the text T , where one occurrence of P is found (encircled).

Fig. 1. An example of shift-or algorithm with pattern $P = aab$ and text $T = acaab$.

3. THE ARCHITECTURE

In the SNORT systems, some rules may share the same header. Let M and N be the number of rules and number of distinct headers in a SNORT rule set. Therefore, $N \leq M$. The proposed architecture for header classification contains N modules. Each module is responsible for the pattern matching of a single header. Rules having the same header therefore share the same module. As shown in Fig. 2, the proposed architecture can process in parallel with the architectures for the signature detection in the payload. The incoming source is first broadcasted to the proposed architecture and the architecture for signature matching. An alarm will be issued when both the header and signature hits are detected for any rule.

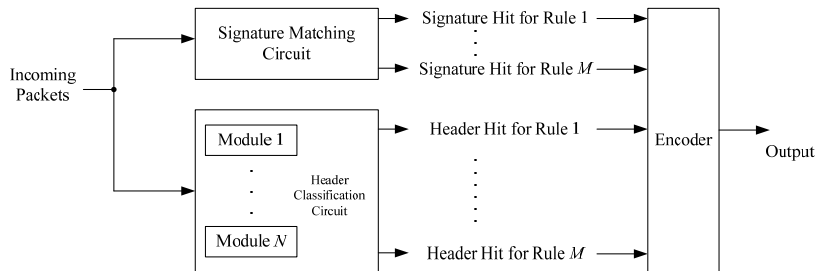


Fig. 2. The structure of the proposed header classification circuit for NIDS.

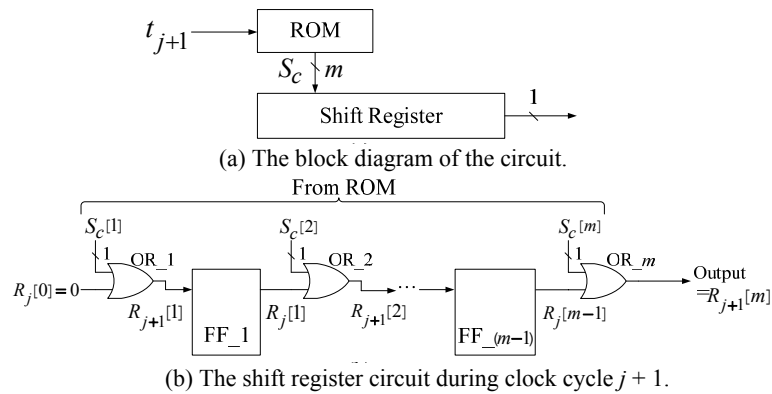


Fig. 3. The basic circuit of each module for exact pattern matching.

3.1 ROM-Based Shift-or Circuit

Our previous work [8] proposed a direct hardware implementation of shift-or algorithm, as shown in Fig. 3, where each module contains a ROM and a shift register. There are $|\Sigma|$ entries in the ROM. The k th entry of the ROM contains the m -bit vector S_k , where m is the size of the pattern associated with the module. The shift register consists of $m - 1$ flip-flops (FFs) and m OR gates.

The module operates by scanning the source string one character at a time. Therefore, after the clock cycle j , the circuit completes the string matching process up to t_j . Moreover, the character t_{j+1} is the input character to the module during the clock cycle ($j + 1$). Assume $t_{j+1} = s_c$. The input character t_{j+1} is first delivered to the ROM for the retrieval of S_c to the OR gates. Each OR gate i has two inputs: one is from the i th output bit of the ROM (*i.e.*, $S_{c[i]}$), and the other is from the output of FF ($i - 1$), which contains $R_j[i - 1]$ during the clock cycle $j + 1$. From Eq. (3), it follows that the OR gate i produces $R_{j+1}[i]$, which therefore will become the output of FF i during the clock $j + 2$ for the subsequent operations. Note that, during the clock cycle $j + 1$, the m th OR gate produces $R_{j+1}[m]$, which is identical to 0 when $p_1 p_2 \dots p_m = t_{j-m} t_{j-m+1} \dots t_{j+1}$. Therefore, the output of the OR gate m is the check point of exact string matching with pattern size m .

Although the direct implementation is simple, it has a number of drawbacks. When the ROM is implemented by the logic elements or logic modules, the area cost may be high when the alphabet size $|\Sigma|$ is large. The ROM can be realized by the embedded memory bits. However, the memory bits may operate at slower speed as compared with the logic elements. Therefore, the ROM implemented by embedded memory bits may become the bottleneck of the throughput. In addition, the same ROM cannot be shared by different rules. The consumption of embedded memory bits will be high for the circuits containing large number of SNORT rules.

3.2 Shift-or Circuit Based on Symbol Encoder

The shift-or circuit is implemented without ROMs in this paper. Therefore, it requires no embedded memory bit, and attains higher operating frequencies. The ROMs

are replaced by a simple symbol encoder, which can be shared by different rules. Consequently, the proposed circuit is well suited for the implementation of systems containing large number of SNORT rules.

The basic symbol encoder has $|\Sigma|$ bits output [6]. When the symbol s_k is presented at the input, only the k th bit output of the encoder will be set to zero while the others are set to one. Given a pattern $P = p_1p_2 \dots p_m$, the basic symbol encoder can be used to obtain the bit vectors S_k for each symbol $s_k \in |\Sigma|$.

Fig. 4 depicts the proposed circuit based on the symbol encoder. As shown in the figure, the circuit contains only the encoder and a shift register. The shift register also consists of $m - 1$ FFs and m OR gates. Each OR gate i also has two inputs, and the first one is connected to the output of FF $(i - 1)$. However, the second input is connected to the outputs of the symbol encoder. The connection is dependent on p_i , the i th character of the pattern P . When $p_i = s_k$, we connect the second input of OR gate i to the k th output of the symbol encoder.

Note that, because p_i is a character in the pattern P , it belongs to $\Sigma = \{s_1, \dots, s_{|\Sigma|}\}$. Let $f: \Sigma \rightarrow \{1, \dots, |\Sigma|\}$ be a function such that, if $p_i = s_k$, then $f(p_i) = k$. Therefore, as shown in Fig. 4, the second input of the OR gate i is connected to the $f(p_i)$ th output of the symbol encoder.

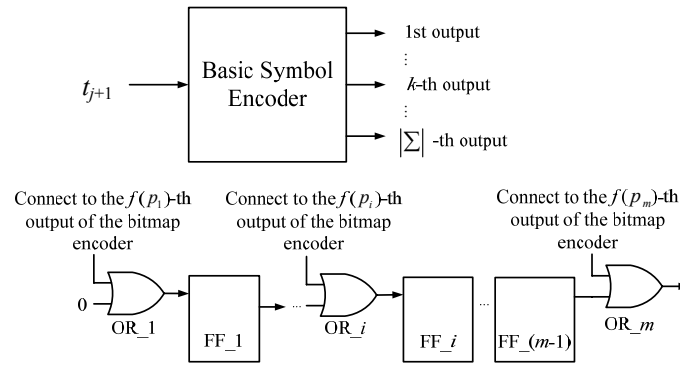
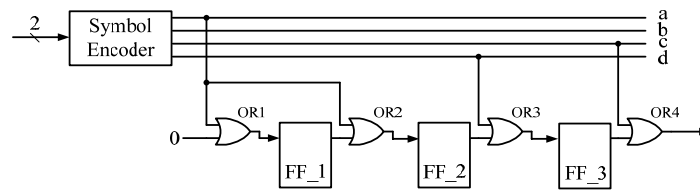


Fig. 4. The shift-or circuit based on a symbol encoder.

Let $I_k = \{i: f(p_i) = k\}$. Consequently, when s_k is presented at the input of the symbol encoder, all the OR gates having index $i \in I_k$ receives 0 from the symbol encoder, while the others receive 1. Based on Eq. (2), we conclude that each OR gate i obtains $S_k[i]$ from the symbol encoder when the input symbol is s_k . That is, the symbol encoder can replace the ROM for hardware shift-or implementation.

Fig. 5 shows a simple example of the proposed circuit for the pattern $P = aadc$ and $\Sigma = \{a, b, c, d\}$. In this example, since there are only four symbols, the symbol encoder has four outputs. We set $a = s_1$, $b = s_2$, $c = s_3$, and $d = s_4$. Because $p_1 = p_2 = a$, we have $f(p_1) = f(p_2) = 1$. Both the OR1 and OR2 are connected to the first output of the symbol encoder. Moreover, based on the facts that $p_3 = d$ and $p_4 = c$, we connect the OR3 and OR4 to the 4th and 3rd outputs of the symbol encoder, respectively. The shift register therefore can receive bit vectors 0011 (i.e., S_1), 1111 (i.e., S_2) 1110 (i.e., S_3) and 1101 (i.e., S_4) when symbols a, b, c and d are the input symbols, respectively.

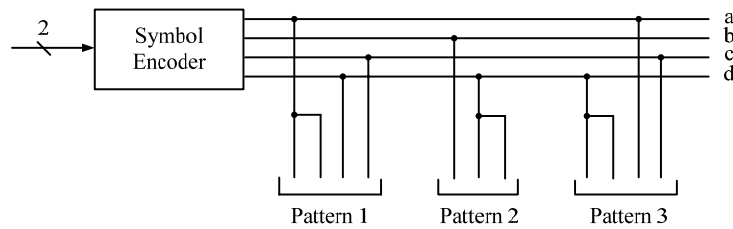


(a) The architecture.

Input	Pattern 1
a	0011
b	1111
c	1110
d	1101

(b) Table of the pattern.

Fig. 5. A simple example of the proposed circuit for the pattern *aadc* and the total symbol *a, b, c, d*.



(a) The architecture.

Input	Pattern 1	Pattern 2	Pattern 3
a	0011	111	1101
b	1111	011	1111
c	1110	111	1110
d	1101	100	0011

(b) Table of the pattern.

Fig. 6. An example of three patterns (*aadc*, *bdd* and *ddac*) share the same symbol encoder.

One advantage of using the symbol encoder is that the encoder is independent of the target pattern P . Only the connection between the encoder and shift register is dependent on P . Consequently, different Snort rules can share the same symbol encoder. An example of three patterns (*aadc*, *bdd* and *ddac*) sharing the same symbol encoder is shown in Fig. 6, where we set $\Sigma = \{a, b, c, d\}$.

Note that some output pins of the symbol encoder may left open. This is because some symbols in the alphabet Σ may not belong to the patterns considered in the circuit. Consequently, the corresponding output pins of these symbols will not be connected to the OR gates of the shift registers. It is therefore not necessary to design a symbol encoder for the alphabet Σ . We can design a symbol encoder for the alphabet excluding all the symbols not used by the patterns to further reduce the area cost and avoid open output pins.

4. HEADER CLASSIFICATION MODULE BASED ON SHIFT-OR CIRCUIT

Recall that the header of each SNORT rule contains 5 fields: source IP address, destination IP address, protocol, source port and destination port. In our design, each module for header classification shown in Fig. 2 contains two components: the first component is used for the matching of source IP address, destination IP address and protocol. The second component is used for the port matching.

The shift-or architecture can be directly used for the implementation of the first component for most of the rules. Although the length of a packet header is fixed, when some of the fields of a SNORT rule are left unspecified (*i.e.*, specified as “any”), the length of the shift register of the corresponding module can be shortened. This may effectively save the area cost for FPGA implementation.

Moreover, for the rules having HOME_NET as source IP address or destination IP address, the modification of the shift-or architecture is necessary. Note that, in the shift-or architecture, the target pattern P is assumed to be fixed. Nevertheless, HOME_NET is a variable indicating the network to be protected. Therefore, it is dependent on the location of the NIDS. Fig. 7 shows the shift-or circuit where the target patterns can be changed for different IP locations. This can be accomplished by the employment of multiplexers, which serve as switches controlling the connections between the symbol encoder and shift register. The positions of the switches are determined by the target pattern P , which are stored in a buffer. The buffer content will be updated when P is changed. For sake of simplicity, in the Fig. 7 we consider only $\Sigma = \{a, b, c, d\}$. In addition, the length of target pattern P is 4. Therefore, there are only 3 FFs, 4 OR gates and 4 multiplexers. The extension of the circuit for the actual pattern matching of HOME_NET is straightforward. We simply set $\Sigma = \{0, \dots, 255\}$. The length of the pattern remains 4.

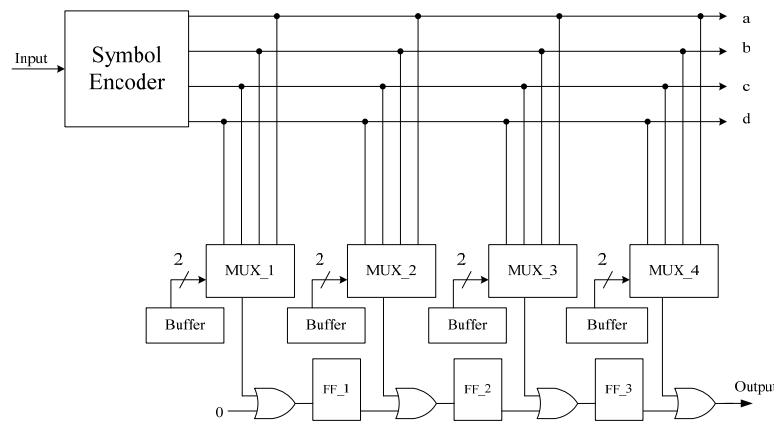


Fig. 7. The shift-or circuit where the target patterns can be changed for different IP addresses.

The implementation of the second component is dependent on the port specification. When both the source and destination ports are indicated by a single port number, the proposed shift-or architecture can be effectively employed for port matching. However, in case the ports are specified in terms of range, direct applications of the proposed archi-

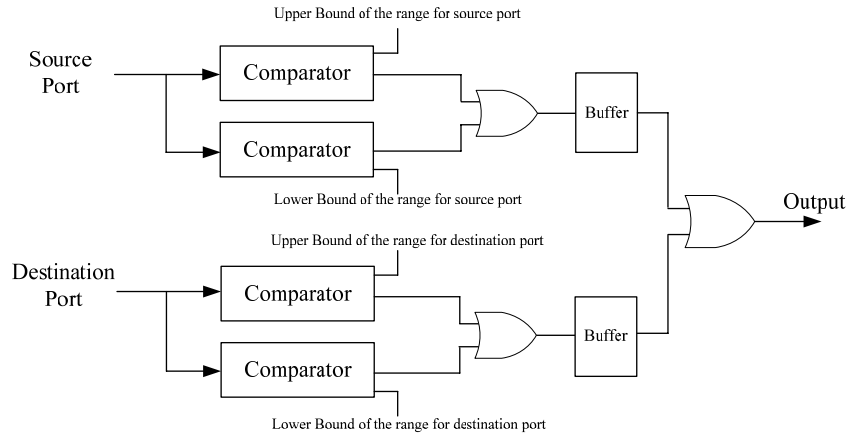


Fig. 8. Range matching circuit for port matching.

ture may be difficult. As shown in Fig. 8, comparators are then employed for range matching in our design. For each of the source and destination ports, there are two comparators, which are compared against the upper bound and the lower bound of the range, respectively. Both the upper bound and lower bound are specified in 16 bits. The circuit is therefore able to match any interval in $[0, 65535]$.

4.1 High Throughput Shift-or Circuit for Header Classification

The basic module circuit shown in Fig. 3 only processes one character per cycle. The throughput of the header classification architecture can be improved further by processing q characters at a time. This can be accomplished by grouping q consecutive characters in the source into a single symbol. Note that, the shift-or algorithm actually process one symbol at a time. In our basic implementation as stated above, each character is a symbol. Now, by grouping q characters as a symbol, we are able to process q characters at a time.

Without loss of generality, we consider $q = 2$. Let $\Omega = \{x_1, \dots, x_{|\Omega|}\}$ be the alphabet for the new symbols, where $x_k = (y_1, y_2)$, and $y_1, y_2 \in \Sigma$. That is, $\Omega = \Sigma \times \Sigma$. Note that, the new symbols containing the same set of characters but with different orders are viewed as different symbols. For examples, (a, b) and (b, a) are different. Therefore, The size of the new alphabet $|\Omega| = |\Sigma|^2$.

Based on Ω , a pattern P can be rewritten as $P = u_1 u_2 \dots u_{\lceil m/2 \rceil}$, where $u_i = (p_{2i-1}, p_{2i})$. Note that $u_{\lceil m/2 \rceil} = (p_{m-1}, p_m)$ when m is even. However, when m is odd, $u_{\lceil m/2 \rceil} = (p_m, \phi)$, where ϕ denotes ‘‘don’t care,’’ and can be any character in Σ . We can then associate a bit vector X_k containing $\lceil m/2 \rceil$ elements for each symbol $x_k \in \Omega$, where the i th element of X_k is given by

$$X_k[i] = \begin{cases} 0, & \text{if } x_k = u_i, \\ 1, & \text{otherwise.} \end{cases} \quad (4)$$

A symbol encoder is required for providing $X_1, \dots, X_{|\Omega|}$. In this case, the symbol encoder contain $|\Omega| = |\Sigma|^2$ outputs, where each output i will be set to 0 when the symbol x_i is presented at the input.

Note that the string matching operations ending at j over the alphabet Ω is equivalent to the operations ending at either $2j$ or $2j + 1$ (but not both) over the alphabet Σ . It may then be necessary to perform the matching process ending at every location of the source over the alphabet Σ . Therefore, when design the circuit for signature matching, two shift registers should be employed. One is for even locations, and the other is for odd locations. However, for the header classification applications, because the packet headers have fixed locations, single shift register will be sufficient in the circuit.

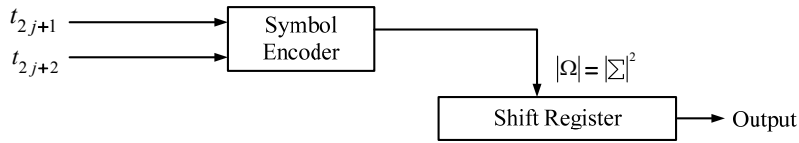


Fig. 9. The structure of a high throughput module circuit processing two characters at a time ($q = 2$) with the symbol encoder.

Fig. 9 shows the proposed circuit with $q = 2$ for header classification. Note that, since the pattern P in this case contains $\lceil m/2 \rceil$ symbols, the shift registers with $\lceil m/2 \rceil - 1$ FFs and $\lceil m/2 \rceil$ OR gates are sufficient for the operations. To perform the string matching operations for $q = 2$, we convert the source T to the sequence $T_e = e_1 e_2 \dots$ over alphabet Ω , where $e_j = (t_{2j-1}, t_{2j})$. During the clock cycle $j + 1$, symbol e_{j+1} is fetched to the symbol encoder. This is equivalent to the scanning of two characters t_{2j+1} and t_{2j+2} simultaneously for shift-or operations.

The scheme shown in Fig. 9 can be extended for any $q > 2$ by first forming the new alphabet $\Omega = \{x_1, \dots, x_{|\Omega|}\}$, where each symbol x_k is of q -tuple (*i.e.*, $x_k = (y_1, \dots, y_q)$ and $y_j \in \Sigma, j = 1, \dots, q$). The corresponding symbol encoder should then have $|\Omega| = |\Sigma|^q$ outputs. The area complexity of the symbol encoder therefore will go exponentially as q increases. The design of the proposed circuit may then become impractical for large q .

To solve the area cost problem for large q , a modified symbol encoder architecture (termed type I symbol encoder) is proposed in this paper. In the symbol encoder type I, the output can be separated into q groups G_1, \dots, G_q , where each group contains $|\Sigma|$ outputs. Suppose the symbol $x_k = (y_1, \dots, y_q)$ is presented in the input. For each $j, j = 1, \dots, q$, if $y_j = s_k$, only the k th output of G_j will be set to zero, and the other outputs of G_j will be set to one. Since the total number of outputs is $q|\Sigma|$, the area and routing cost may only grows linearly with q .

Consider $q = 2$ as an example. There are only two groups G_1 and G_2 at the output. Suppose $\Sigma = \{a, b, c, d\}$. Each group then contains 4 outputs. Let $a = s_1, b = s_2, c = s_3$, and $d = s_4$. If the symbol $x_k = (a, c)$ is presented at the input, then $y_1 = s_1$ and $y_2 = s_3$. The 1st output and 3rd output of G_1 and G_2 will be zero, respectively. The others are set to one.

The connections from shift registers to the type I symbol encoder are modified accordingly. Again use $q = 2$ for the sake of brevity. Consider the pattern $P = u_1 u_2 \dots u_{\lceil m/2 \rceil}$,

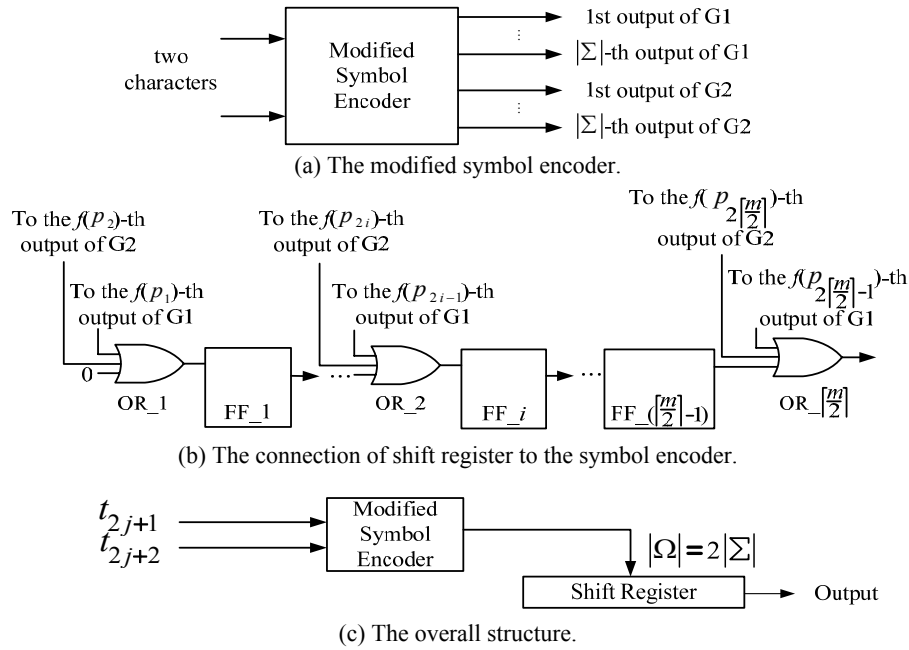


Fig. 10. The structure of proposed high throughput circuit with modified symbol encoder for $q = 2$.

where $u_i = (p_{2i-1}, p_{2i})$. Recall that, in the system shown in Fig. 4, each OR gate i has only two inputs. In the scheme based on the modified symbol encoder, as shown in Fig. 10, each OR gate i has three inputs for $q = 2$. The first one is connected to the output of FF($i - 1$). The other two are connected to the $f(p_{2i-1})$ th output and $f(p_{2i})$ th output of G_1 and G_2 respectively. It can then easily be shown that this connection will obtain X_k for the shift-or operation when the symbol x_k is presented at the input of the system.

Note that, when both q and/or $|\Sigma|$ are large, the area cost for implementing the modified symbol encoder may still be high. Suppose Σ is the cross product of a smaller set Γ . For instance, $\Sigma = \Gamma \times \Gamma$. A simple way to reduce the area cost is to design the modified symbol encoder based on Γ . That is, the output of the symbol encoder now contains $2q$ groups: $G_{11}, G_{12}, G_{21}, G_{22}, \dots, G_{q1}, G_{q2}$, where each group has $|\Gamma|$ outputs. Because $\Sigma = \Gamma \times \Gamma$, we have $|\Gamma| = \sqrt{|\Sigma|}$. We call this type of symbol encoder, the type II symbol encoder. The number of output pins of the type II symbol encoder therefore is only $2q|\Gamma| = 2q\sqrt{|\Sigma|}$. By contrast, the output pins of type I symbol encoder is $q \times |\Sigma|$. The area and routing cost may then be substantially reduced for type II symbol encoder for large q and/or $|\Sigma|$.

Fig. 11 shows an example of type II symbol encoder with $q = 2$. As shown in the figure, the output pins can be divided into 4 groups: $G_{11}, G_{12}, G_{21}, G_{22}$. Each group has $\sqrt{|\Sigma|}$ outputs. The total number of outputs of the type II symbol encoder then becomes $4\sqrt{|\Sigma|}$, as shown in Fig. 11 (b).

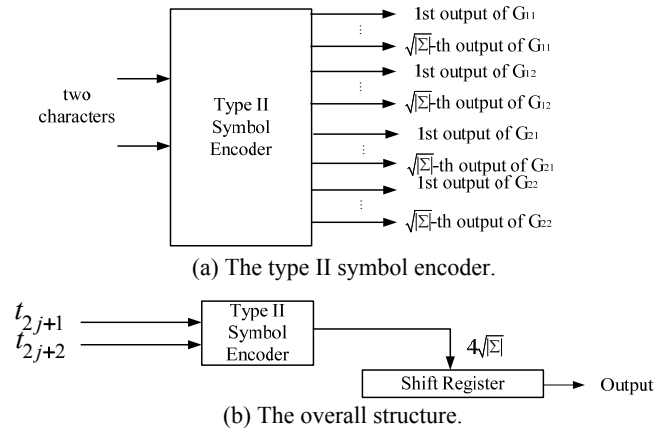


Fig. 11. An example of type II symbol encoder for $q = 2$.

5. EXPERIMENTAL RESULTS AND COMPARISONS

This section presents experimental results of the proposed architecture for header classification of SNORT rules. *We have implemented the header classification circuit for the headers of the entire SNORT rule set* The version of the rule set is 2.0. In our experiments, the throughput, the maximum operating frequency, and the area cost of the architecture are used as the performance measures of the NIDS systems. The throughput indicates the maximum number of bits per second the circuit can process. The maximum operating frequency of a circuit is the maximum post-layout frequency of the circuit. The area cost reveals the hardware resource consumed by the proposed circuit. The measurements of the area cost are dependent on the target FPGA devices.

Table 1 shows the performance of the proposed circuit for various q values for type I symbol encoder. The target FPGA device is Altera Stratix II. In the table, the area cost is measured by the number of adaptive logic modules (ALMs) consumed by the circuit. From the table, it can be observed that the area cost of the proposed circuit is low for small q values. In particular, when $q = 1$, the circuit only consumes 1014 ALMs, which is only 4.10% of the total number of ALMs in the FPGA device Altera Stratix II EP2S60. Nevertheless, the area cost may be high when q becomes large for the type I symbol encoder. This is because the alphabet $\Sigma = \{00, 01, \dots, FF\}$, where each element is expressed in hexadecimal form, for header classification has large size (*i.e.*, $|\Sigma| = 256$).

Table 1. The performance of the proposed circuit for various q values for type I symbol encoder. The target FPGA device is Altera Stratix II.

Design	Area Cost (Number of ALMs)	Operating Frequency (MHz)	Throughput (Gbits/sec)
Proposed architecture ($q = 1$)	1014	500	4.0
Proposed architecture ($q = 2$)	1354	500	8.0
Proposed architecture ($q = 4$)	2020	500	16.0
Proposed architecture ($q = 8$)	3422	500	32.0

Table 2. The performance of the proposed circuit for various q values for type II symbol encoder. The target FPGA device is Altera Stratix II.

Design	Area Cost (Number of ALMs)	Operating Frequency (MHz)	Throughput (Gbits/sec)
Proposed architecture ($q = 2$)	447	500	8.0
Proposed architecture ($q = 4$)	501	500	16.0
Proposed architecture ($q = 8$)	546	500	32.0

Table 3. Comparisons of the proposed architectures with the existing ones.

Design	Area Cost (Number of Slices)	Device	Throughput (Gbits/sec)
Proposed architecture ($q = 8$ and Type II symbol encoder)	725	Xilinx Virtex2	20.5
Proposed architecture ($q = 8$ and Type II symbol encoder)	723	Xilinx Virtex4 100	22.6
Dimopoulos <i>et al.</i> [5]	1756	Xilinx Virtex2	12.4
Song <i>et al.</i> [10]	2100	Xilinx Virtex4 100	10.0

To reduce the area cost, we also implement the type II symbol encoder for the header classification. The type II symbol encoder is designed using the set Γ , where $\Sigma = \Gamma \times \Gamma$. That is, $\Gamma = \{0, \dots, F\}$. As shown in the Table 2, the area cost for large q is lowered substantially for the type II encoder. It should also be noted that all the circuits considered in the table operates at 500 MHz, which is the highest frequency of the Altera Stratix II EP2S60. Because of the simplicity of the proposed circuits, the highest operating frequency can always be attained even for large q . Therefore, when $q = 8$, the throughput becomes 32 Gbits/sec. The circuit can then be effectively used for the header classification of networks with high bandwidth.

Table 3 compares the proposed circuit with the existing architectures for only header classification. Based on the same FPGA devices Xilinx Virtex2, it can be observed from Table 3 that our architecture attains higher throughput with lower area cost as compared with the architecture proposed by Dimopoulos *et al.* [5]. Our architecture also outperforms the architecture proposed by Song *et al.* [10] on the same FPGA device Xilinx Virtex 4 100. Our architecture has superior performance because it is based on the simple and flexible shift-or architecture. When the IP addresses in the header are left unspecified (*i.e.*, the IP addresses are specified as “any”), the length of the shift register in the architecture can be shortened. On the contrary, other architectures such as ternary content addressable memory (TCAM) [5, 10] have fixed structure. They may therefore need more number of slices and operate at lower speed for header classification.

To further assess the performance of the proposed architecture, the FPGA-based signature matching coprocessors for NIDS with and without the proposed header classification circuits have been implemented and compared. The signature matching (*i.e.*, payload inspection) parts of the coprocessors are also implemented by the proposed architecture with $q = 2$ and 4, as shown in Table 4. The rule set size for payload inspection is measured by the number of characters of the signatures in the payload. In this experiment, the rule set size is 8000 characters. The target FPGA device is Altera Stratix II EP2S60.

Table 4. Performance of the FPGA-based signature matching coprocessors for NIDS with and without the proposed header classification circuits.

Design	Area Cost (Number of ALMs)	Operating Frequency (MHz)	Throughput (Gbits/sec)
Signature matching without header classification ($q = 2$)	3155	476.87	7.6
Signature matching with header classification ($q = 2$ and Type II symbol encoder)	3885	408.83	6.5
Signature matching without header classification ($q = 4$)	3955	397.30	12.7
Signature matching with header classification ($q = 4$ and Type II symbol encoder)	4595	371.20	11.8

As shown in the Table 4, the area cost of the payload inspection with header classification is only slightly higher than that of the payload inspection without header classification. In particular, when $q = 4$, the area cost of the signature matching system without header classification is 3955 ALMs; whereas, the system with the proposed header classification circuit is 4595 ALMs. The employment of header classification only increases the area cost by 640 ALMs. Although larger circuits may result in slower operating speed and lower throughput, it can be observed from Table 4 that the operating frequency and throughput of the coprocessor with header classification are only slightly lower than those of the coprocessor without header classification. Consequently, the proposed header classification circuit imposes only limited overhead for the signature matching coprocessor. These results demonstrate the effectiveness of the circuit.

6. CONCLUSION

A novel FPGA implementation of header classification architecture based on shift-or algorithm is presented in this paper. The proposed algorithm in the basic form process one character at a time, and contain only a ROM and a simple shift register for each pattern matching. The throughput can be further enhanced by replacing the ROM with a simple symbol encoder, and by processing multiple characters in parallel.

To reduce the area cost of the symbol encoder when multiple characters are processed by parallel, two modified symbol encoder architectures, termed type I and type II symbol encoders, are proposed. In particular, the type II symbol encoder is able to effectively lower the area cost even when large number of characters are processed in parallel.

The proposed algorithm is implemented up to processing eight characters at a time in our experiments. Experimental results show that the header classification circuit based on type II encoder consumes only 546 ALMs, which is only 2.25% of the total number of ALMs in the FPGA device Altera Stratix II EP2S60. Moreover, based on the same FPGA device, our architectures consume lower hardware resource while attaining higher throughput for header classification as compared with existing architectures. These results reveal that our design is one of the cost-effective solutions to the FPGA implementations of the NIDS systems.

REFERENCES

1. M. Aldwairi, T. Conte, and P. Franzon, "Configurable string matching hardware for speeding up intrusion detection," *ACM SIGARCH Computer Architecture News*, Vol. 33, 2005, pp. 99-107.
2. R. Baeza-Tates and G. H. Gonnet, "A new approach to text searching," *Communications of the ACM*, Vol. 35, 1992, pp. 74-82.
3. C. Clark and D. Schimmel, "Scalable multi-pattern matching on high speed networks," in *Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines*, 2004, pp. 249-257.
4. Y. H. Cho and W. H. Mangione-Smith, "Deep packet filter with dedicated logic and read only memories," in *Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines*, 2004, pp. 125-134.
5. V. Dimopoulos, G. Papadopoulos, and D. Pnevmatikatos, "On the importance of header classification in hw/sw network intrusion detection systems," in *Proceedings of the 10th Panhellenic Conference on Informatics*, 2005, pp. 661-671.
6. W. J. Hwang, C. M. Ou, Y. N. Shih, and C. T. D. Lo, "High throughput and low area cost FPGA-based signature match circuit for network Intrusion detection," *Journal of the Chinese Institute of Engineers*, Vol. 32, 2009, pp. 397-405.
7. T. Ramirez and C. D. Lo, "Rule set decomposition for hardware network intrusion detection," in *Proceedings of the International Computer Symposium*, 2004, pp. 1224-1229.
8. H. C. Roan, C. M. Ou, W. J. Hwang, and C. T. D. Lo, "Efficient logic circuit for network intrusion detection," *Lecture Notes in Computer Science*, Vol. 4096, 2006, pp. 776-784.
9. J. Singaraju, L. Bu, and J. A. Chandy, "A signature match processor architecture for network intrusion detection," in *Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines*, 2005, pp. 235-242.
10. H. Song and J. Lockwood, "Efficient packet classification for network intrusion detection using FPGAs," in *Proceedings of IEEE Symposium on Field-Programmable Gate Arrays*, 2005, pp. 238-245.
11. I. Sourdis and D. N. Pnevmatikatos, "Pre-decoded CAMs for efficient and highspeed NIDS pattern matching," in *Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines*, 2004, pp. 258-267.
12. SNORT official web site, <http://www.snort.org>.



Wen-Jyi Hwang (黃文吉) received his diploma in Electronics Engineering from National Taipei Institute of Technology, Taiwan, in 1987, and M.S.E.C.E. and Ph.D. degrees from the University of Massachusetts at Amherst in 1990 and 1993, respectively.

From September 1993 until January 2003, he was with the Department of Electrical Engineering, Chung Yuan Christian University, Taiwan. In February 2003, he joined the Graduate Institute of Computer Science and Information Engineering, Na-

tional Taiwan Normal University, where he is now a Full Professor. His research interests are centered on multimedia communications systems with particular emphasis on image/video transmission.

Dr. Hwang is the recipient of the 2000 Outstanding Research Professor Award from Chung Yuan Christian University, 2002 Outstanding Young Researcher Award from the Asia-Pacific Board of the IEEE Communication Society, and 2002 Outstanding Young Electrical Engineer Award from the Chinese Institute of the Electrical Engineering.



Chien-Min Ou (歐謙敏) received his diploma in Telecommunication Engineering from Chien Shin Institute of Technology, Chungli, Taiwan, in 1978 and M.S., and Ph.D. degrees in Electrical Engineering from Chung Yuan Christian University, Chungli, Taiwan, in 2000, and 2003, respectively. He joined the Faculty of the Department of Electronics Engineering, Ching Yun Institute of Technology, Chungli, Taiwan, as a Lecturer in 1978. Since 2004, He has been an Assistant Professor of the Department of Electronics Engineering, Ching Yun University. He is also a member of the honor society Phi Tau Phi. His research topics include VLSI design and testing, image processing, video compression, motion estimation.