ABSTRACT
Pattern matching in network applications is characterized by intensive computation. In conventional hardware accelerating methods, performance versus cost is a trade-off. In this paper, we proposed a new Hash-CAM hybrid architecture using state-of-the-art FPGA technology that is customized for a given pattern matching problem for DPI.

1. INTRODUCTION
The evolution of the Internet into a global medium for information and communication has enabled new services, applications and novel business opportunities known as e-businesses. In parallel, sophisticated and continuously evolving Internet security treats by viruses, worms, trojans and unauthorized network intrusions have grown into one of the principal concerns of the Internet. The success and advancement of the Internet is therefore inherently linked to the evolution of defensive network security systems.

The fundamental concept of network security is to let authenticated and authorized users have access to the right information. One aspect of this issue concerns how to authenticate and secure the safe transmission of data, in which data encryption is applied. Another aspect of network security is related to the access control, policing and authorization of users and services to networks and network resources. Successful policing and access control depends upon sophisticated network and network-data surveillance methods. Deep-Packet-Inspection (DPI), based on fast pattern matching became the fundamental function of network surveillance methods and emerging network security systems.

Known methods for fast pattern matching on streamed network data are based on Finite State Machines (FSM) [6] and fast lookup methods including Hashing [3,4,8] and Content Addressable Memory (CAM) [2,7]. Related research work has shown that data inspection throughput, the number of supported signatures and hardware/memory resource costs are key metrics for pattern matching architectures. For example, hash based pattern matching presents a low-cost solution, however it is restricted due to inevitable hash-collision problems [9], whereas CAM allows fast exact matching of patterns at a significantly higher hardware cost [7].

In this paper, a hybrid fast-pattern-matching architecture for hardware-based DPI is proposed and proof-of-concept implementation based on FPGA technology is presented. The architecture is a hybrid of a Hash and CAM circuit, customized to the deployed DPI pattern-set. Reconfigurable FPGA technology is deployed to map a custom specific Hash-CAM pattern matching circuit that is tailored to the known signature-patterns. The reconfigurable nature of the underlying FPGA technology is an essential and integral part of the proposed architecture.

2. RELATED WORK
Most of the proposed architectures for string-matching problems, either CAM or FSM based methods [2-8], have a common problem of excessive use of hardware and memory resources. RAM based FSM [6] architectures require a large amount of memory for input word-size $n > 16$ bits.

Cho et al. firstly proposed prefix matching [10], where the prefixes of patterns are used as hashing keys. In this approach, a CAM-like component is used to store the prefixes and a back-end ROM is used to store the suffixes. In this approach a prefix-match produced a suffix ROM address. If prefix and suffix of a signature are found to be equal, a successful match is reported. The main drawback of Cho’s architecture is its limitation of storing any two strings with an identical prefix. This problem can be seen as a collision phenomenon that arises in most hash functions.

In order to overcome the collision problem, Sourdis [3] and Giorgos et al [4], proposed a collision-avoidance method based on perfect hashing. The perfect-hash function is derived by analysing the signature string-set for unique bits that can represent each string. By Sourdis’s [3] approach, each unique bit-pattern is used as a hashing tree to find a guaranteed 1-to-1 map from the original
strings to the hash keys. Giorgos et al. [4], however, adopts a simpler Cyclic Redundancy Check (CRC) based hash function. Collision is avoided by carefully choosing a CRC polynomial for the given signature-set. Sourdis and Giorgos’s concepts are applicable for static string sets. However, for any applications that require dynamic string-set update, perfect hashing cannot be deployed without false negative/positive.

3. PROPOSED HASH-CAM ARCHITECTURE

For lookup and string matching problems, hashing has proven to be the most efficient method in terms of memory utilisation and lookup operation [9]. Hash tables allow a constant lookup time of $O(1)$ on average, regardless of the number of table entries. However, hash functions, excluding perfect hashing, can produce two or more identical keys for distinct input values. This inevitable case is referred to as hash collision. Assuming that each collision will require the reiteration of the hash lookup function, the worst-case lookup time can reach $O(n)$.

Perfect hashing is designed to eliminate collisions for a specific string-set, but it is difficult to derive if the number of strings is large and efficient hashable (memory) utilisation becomes a priority. Furthermore, perfect hashing is unsuitable for applications for which the string set is incrementally (dynamic) updated. The ideal scheme for DPI must therefore satisfy collision-free storage, efficiency resource utilisation, and flexibility for incremental (dynamic) string-set update.

Experimental results with CAM and Hash lookup tables have shown that only a hybrid solution comprised of a Hash and CAM circuit, partially tailored to the signature-set will provide a solution that will satisfy $O(1)$ lookup time while achieving reasonable resource utilisation for the given pattern-set. State-of-the-art FPGA technologies are equipped with memory and logic resources that are an ideal technology platform for hybrid pattern-matching circuit architectures.

The proposed architecture is based on Altera Stratix-II technology [1], deploying two concurrent lookup circuits, a Hash and a CAM circuit, for pattern matching. The hash circuit is comprised of a hash function, comparison circuits and a dual-port embedded memory. The embedded memory stores the signatures and the corresponding Signature-IDs (S-ID).

The hash function generates a hash key for any input string that is used to address the embedded memory to obtain the suspected pattern. Both the input string and the retrieved pattern are compared. In case of a match the S-ID of the pattern is flagged to indicate the matching pattern. Inevitably the hash memory utilisation is inherently linked to the collision probability of the hash function. Higher memory utilisation means also higher hash collision probability. For DPI, the patterns are known, subsequently the patterns that collide for the given hash function are also known. Pre-analysis of patterns for a known hash function allows the identification of all colliding patterns.

As collision resolution, the second, CAM-based lookup circuit is used. All colliding patterns obtained for the first hash-based lookup circuit are now stored at the second CAM-based lookup circuit. The same input string used by the hash circuit to obtain hash key are now used by the CAM to obtain the matching signature S-ID.

In case of a collision, the first hash-based circuit will produce a valid hash key. However the obtained memory content will not match as it stores only 1 or 2 of the 3, 4 or $k$ matching strings. In this case the second CAM-based circuit will compare the same input string with its entries and output the matching S-ID.

This proposed hybrid approach allows a lookup performance of $O(1)$, equivalent to a purely CAM based architecture at a significantly lower cost and with more efficient memory utilisation.

The architecture can be customised by trading off hash circuit memory utilisation versus CAM size for a given string set. An overview of the Hash-CAM architecture is shown in figure 1. In order to reduce the cost of the embedded CAM circuit, a hash circuit based on a single-collision tolerance method is deployed. In this method two items stored in two different RAM cells but addressed by the same hash key are used.
4. HASH CAM TRADE-OFF ANALYSIS

Both lookup circuits, Hash and CAM, are different in nature and hardware cost. Both circuits can accommodate a certain number of entries that determine the circuit cost and performance. Both circuits are inherently linked as the CAM-based circuit stores only the strings that have a colliding hash-key value stored at the Hash-based circuit. It is therefore important to determine the correlation between the two lookup circuits in order to optimise the pattern matching architecture for the deployed technology.

Embedded memory resources and logic resources are limited. A high memory utilisation can introduce a logic cost explosion due to the size of the embedded CAM circuit [5]. Therefore, the trade off must be finely balanced in obtaining the most appropriate hash function, the dimensioning of the hash memory utilisation and the affordable size of the CAM memory.

A good hash function uniformly distributes the keys into the different locations of hash table. In an extreme scenario, a specific hash function can even achieve the injective mapping, i.e. perfect hashing. However, perfect hash function also claims quadratic memory space requirement [9]. When the number of keys is large, such a quadratic space requirement is probably unaffordable for the on-chip memory utilisation and the affordable size of the CAM memory.

A good hash function uniformly distributes the keys into the different locations of hash table. In an extreme scenario, a specific hash function can even achieve the injective mapping, i.e. perfect hashing. However, perfect hash function also claims quadratic memory space requirement [9]. When the number of keys is large, such a quadratic space requirement is probably unaffordable for the on-chip memory utilisation and the affordable size of the CAM memory.

Assume that there are \( m \) keys to be inserted into a hash table with size of \( n \) and \( l \) number of keys that are hashed into a single given slot. Overall, there will be \( \lambda \) number of such slots each with \( l \) keys hashed in. The expectation \( E \) of \( \lambda \) can be expressed as:

\[
E[\lambda] = n \left( \frac{m}{n} \right) \left( 1 - \frac{1}{n} \right)^{m-l} = n \left( \frac{m}{n} \right) e^{-\frac{m}{n}}
\]  

(1)

Equation (1), allows the calculation of the number of slots with \( l \) hashed-in keys, e.g. \( l = 0, 1, \) and \( 2 \).

In our proposed Dual-RAM architecture, the first block of RAM is designed to accommodate the first element that is hashed in a given slot. The percentage of used slots, or load factor \( L_1 \), of the first RAM block can be expressed using equation (1) for \( \lambda = 1 \):

\[
L_1 = 1 - e^{-\frac{m}{n}}
\]  

(2)

Consequently, the load factor \( L_2 \) of second RAM can be expressed as:

\[
L_2 = 1 - e^{-\frac{m-n}{n}} - \left( \frac{m}{n} \right) e^{-\frac{m}{n}}
\]  

(3)

Finally, the number of keys \( C \) that needs to be stored in the CAM is the remaining keys \( C \).

\[
C = m - n \cdot L_1 - n \cdot L_2
\]  

(4)

\( \lambda \) is define as the ratio of:

\[
\lambda = \frac{m}{n},
\]  

(5)

Equation (4) can now be expressed as follows:

\[
C = m \cdot \left( \lambda + 2 \right) e^{-\lambda} + \left( \lambda - 2 \right)
\]  

(6)

The number of expected keys that need to be stored in the CAM for a given \( \lambda \) is donated by equation (6), assuming that the hash function produces a uniform distribution of the input data into \( m \) keys. It is therefore essential to obtain a hash function that has the best scrambling properties. A suitable hash function is derived by simulating a number of known hash functions with the best scrambling properties. For the analysis, 4000 unique 32-bit string patterns have been arbitrarily selected from the latest Snort rule-set.

As the hash function, a standard 16-bit Cyclic Redundancy Check (CRC) circuit, based on the original IBM CRC-16 polynomial (0xA001), has been used. Because a total of 4000 sub-patterns have been hashed, only the least 12 bits of the CRC-16 value have been used as the hash key.

Figure-2 shows the two results, (a) the expected results based on equation (6) and (b) the simulated results based on the arbitrarily chosen 4000 snort sub-patterns and the IBM CRC-16 function.

Both, the analytically driven and the simulated result with real data, follows approximately the same trend with a maximum deviation of less than 5%. This is evidence that the selected CRC16 based hash function scrambles the target data set as expected.
These results also prove that equation (6) can be used to estimate the CAM size for a given $m$ and $\lambda$, or the collision graph in Fig. 2 can be used to obtain $\lambda$ if the CAM size and $m$ is known. An experimental study of implementing CAM circuit using FPGA technology by McLaughlin et al [5] has shown that CAM size is restricted by the FPGA LUT resources and entries up to 128 can be build using standard FPGA technology.

Because of a targeted line-rate of 10Gbps and data-path of 128-bit, a CAM size of 64 entry and 128 bit-data-word has been chosen. A 64 entry CAM can accommodate up to 64 collisions. Based on the graph in Fig. 2 this results $\lambda \approx 0.4$. Assuming that $m=1000$, the memory size $n$ can be obtained using equation (5):

$$n = \frac{m}{\lambda} = \frac{1000}{0.4} = 2500$$

(7)

Subsequently, for $m=1000$, the following design parameters for the proposed Hash-CAM architecture in Fig.1 have been chosen:

- Hash Entry $n = 2500$
- CAM Entry $C = 64$

5. IMPLEMENTATION

The Hash-CAM circuit has been implemented using Altera Stratix II FPGA technology. The design was described in VHDL and synthesized using Altera Quartus Tools for the parameters determined in section 4. The post layout synthesis results are shown in Table 1.

| Table 1: Post layout synthesis results
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Stratix II - 2S130F1508</td>
</tr>
<tr>
<td>Clock frequency</td>
</tr>
<tr>
<td>Hash Circuit</td>
</tr>
<tr>
<td>CAM (64x128)</td>
</tr>
<tr>
<td>Total</td>
</tr>
<tr>
<td>% of Device</td>
</tr>
<tr>
<td>Throughput</td>
</tr>
</tbody>
</table>

Overall the circuit utilizes 10% of the FPGA resources for approximately 1000 signatures at a 13.7 Gbps line-rate (128-bit data-path at 107 MHz). Considering that the device can be utilized up to 70%, the proposed pattern matching circuit is capable of searching for up to 7000 patterns, each 128 bit wide, at approximately 10Gbps line rate.

6. CONCLUSION

In this paper, we proposed a novel Hash-CAM circuit architecture for high-performance pattern matching for network applications. The proposed architecture achieves a constant lookup time of O(1). This performance is comparable to purely CAM based circuits at a fraction of the CAM circuit cost. Standard FPGA technology has been used, allowing the integration of two lookup methods within a single device. Furthermore, an analytical model for estimating the CAM cost for a given hash-table-entry ($n$) and number of searched patterns ($m$) has been derived for resource estimation and RAM/CAM trade-off analysis.

The design study for 1000 signatures has shown that FPGA technology is ideal for implementing hybrid custom-specific pattern-matching circuits, outperforming known CAM or Hash based solutions. The memory and logic resource usage can be optimised to achieve the best resource utilisation for a given pattern matching problem such as DPI or packet classification.

REFERENCES


Acknowledgment: This research project is co-sponsored by Altera Corporation and InvestNI.