Exact Worst-Case TCAM Rule Expansion

Ori Rottenstreich, Rami Cohen, Danny Raz and Isaac Keslassy

APPENDIX A TCAM ARCHITECTURES

A. Suggested Architectures

In this section we suggest several TCAM architectures that enable us to implement range encoding more efficiently using logical gates, and illustrate them with a simple example. The objective of these TCAM architectures is to illustrate the trade-off between the worst-case range expansion guarantees and the complexity of the logic module. Since they cannot be implemented directly using off-the-shelf TCAMs, it should be clear that they will probably not directly fit a large class of networking devices.

Fig. 1 illustrates the different TCAM architectures. We assume $d = 2$ fields of $W = 4$ bits each. We want to encode $k = 2$ multi-dimensional ranges R^1 and R^2, where $R^1 = [1, 14] \times [5, 14]$, $R^2 = [7, 10] \times [2, 3]$, and each range leads to a different action. We assume that the default action is predefined in the parallel encoder (PE), and therefore there is no need to add TCAM entries for it. We also use an input example, equal to $8 = \{1000\}$ in the first field and $7 = \{0111\}$ in the second, and denote in parentheses the values that are transmitted on each line.

First, Fig. 1(a) presents the standard INTERNAL-PRODUCT architecture. Using internal Binary-Prefix encoding, it encodes each range by using the product of its TCAM entries along each dimension. In this case, it uses 6×5 entries to encode R^1, and 3×1 entries to encode R^2, yielding a total of 33 entries.

Next, Fig. 1(b) introduces the proposed COMBINED-PRODUCT architecture. Instead of encoding each range only internally, the COMBINED-PRODUCT architecture encodes it using its complementary as well, in at most $f_p(W) = W$ entries instead of $2W-2$ above, and uses more logic to process the results. In this example, it uses 12 entries for R^1 and 3 for R^2, i.e. a total of 15.

Specifically, each field of each range behaves like a single TCAM block. The results of each TCAM entry are entered into a chained logic part that outputs a (1) on each line if it is the first entry that matches the header, and (0) otherwise (i.e., either there was no match on this line or there was a match on a previous line). Note that the chained logic can also be replaced with a more efficient hierarchical logic.

In the second logic part, a logic gate with a control input either behaves like a pass-through gate or like a zeroing gate, depending on whether the encoded entry corresponds to the range or to its complement. Thus, the output is a (1) iff it is the first matching entry and it belongs to the range. Last, an OR gate checks whether the first matching entry belongs to the range, i.e. whether the range is matched. The PE then outputs the first matching range.

TABLE I
UPPER BOUND ON RULE EXPANSIONS OF TCAM ARCHITECTURES

Architecture	Expansion upper bound	Values for $k = 1$, $W = 16$, $d = 2$
INTERNAL-PRODUCT	$k \cdot (2W - 2)^d$	$(30)^2 = 900$
COMBINED-PRODUCT	$k \cdot W^d$	$(16)^2 = 256$
COMBINED-SUM	$k \cdot d \cdot W$	$2 \cdot 16 = 32$

Next, Fig. 1(c) shows the proposed COMBINED-SUM architecture. Each field of each range is encoded separately, by using chaining as in the COMBINED-PRODUCT architecture. Then, in a second stage, an AND gate checks whether all fields have a match. In this example, R^1 is encoded using $3 + 4 = 7$ entries, and R^2 using $3 + 1 = 4$, with a total of 11 entries.

Table I summarizes the bounds on the worst-case rule expansion for each architecture, and provides the corresponding values for IPv4 packet headers with 2 range fields of 16 bits each. The first two results follow from Theorem 13. The last result comes from applying Theorem 7 on each field, with $W \geq 2$.

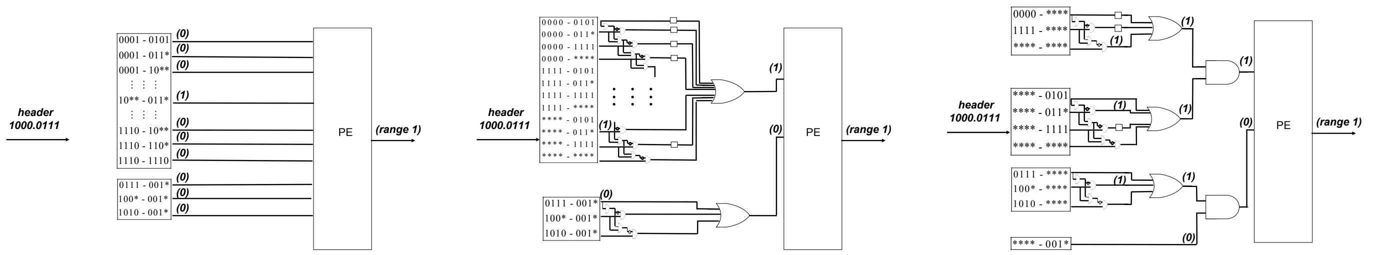
The additional logic required in the last two architectures is summarized as follows. In the COMBINED-PRODUCT architecture, we have for each range one OR gate with multiple inputs as well as one controlled logic gate, one OR gate, one AND gate and one invertor for each TCAM entry. Likewise, in the COMBINED-SUM architecture, there is one AND gate for each range, one OR gate with multiple inputs for each of the d fields as well as one controlled logic gate, one OR gate, one AND gate and one invertor for each TCAM entry.

B. Implementation Considerations

Hot Updates: Since the TCAM is clearly divided between ranges, and the implementation of each range is independent of the other ranges, hot classifier updates are surprisingly easy to apply in this architecture compared to typical TCAM architectures.

Turning Off Entries: In the figures, we only represent the active entries. A simple way to implement the TCAM is to divide it by blocks, each block representing the maximum number of entries per range (Table I). Then, when some entries are not used, it is possible to turn them off. To do so, we add a transistor to switch voltage on and off, together with an SRAM array of 1 bit per entry that remembers the correct action.

PE Size: The number of inputs and outputs of the PE is reduced. It now equals the number of ranges, i.e. the number of rules, instead of the number of TCAM entries. In a sense, the PE is implemented in a hierarchical fashion, with the first logic block being the one shown in the figure (e.g. using chaining). In addition, the size of the SRAM that follows the PE can decrease as well from the TCAM size (number of entries) to the classifier size (number of rules).



(a) INTERNAL-PRODUCT (b) COMBINED-PRODUCT (c) COMBINED-SUM

Fig. 1. TCAM Architectures implementing various encoding schemes

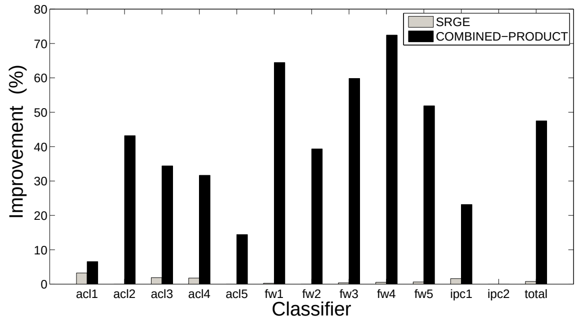


Fig. 2. The improvement of the SRGE scheme and the COMBINED-PRODUCT scheme in the average rule expansion of ClassBench synthetic classifiers in comparison with the Binary-Prefix scheme.

Multiple Actions: To implement more actions than *accept* and *deny*, the architecture does not need to be changed. The action associated with each range simply needs to be indicated in the corresponding SRAM entry.

Encoding Rules Defined on More Than Two Fields: In many cases, a classification *rule* $R = ((R_1, \ldots, R_d) \to a)$ is defined on additional fields besides the two fields with ranges. For instance, besides the ranges defined in the source and destination ports, a rule might require specific values of the source and destination IP and protocol. In these fields the rule is limited to an exact match or to a prefix-match and in both cases the encoding requires only one TCAM entry. To encode the rule itself and not just the fields with ranges, we can simply concatenate the encodings of the additional rules to each of the entries in the range encoding. The TCAM width is changed accordingly. A packet header is then compared in parallel against all the rule fields in one search and there is no requirement for additional searches the additional fields.

C. Additional Simulation Results

Effectiveness on Synthetic Packet Classifiers: We evaluate the suggested architectures on large synthetic classifiers generated by the ClassBench benchmark tool [32], using the 12 standard available files based on real classifiers that were also used in [17], [18]. These 12 files are of three types: access control lists (files acl1-acl5), firewalls (files fw1-fw5) and IP chains (files ipc1-ipc2). We compared, for each of the files, the expansion of the Binary-Prefix encoding, the SRGE encoding [17] and the suggested scheme from Section III-B implemented in the COMBINED-PRODUCT architecture. The file parameters and the detailed results of this comparison are presented in [34]. Figure 2 shows for each of the twelve classification databases the improvement in the average rule expansion of the SRGE scheme and the COMBINED-PRODUCT in comparison with the Binary-Prefix encoding. The last column summarizes the results for all the databases.

TABLE II
AVERAGE RULE EXPANSION WITH REAL-LIFE CLASSIFIERS.

Parameters	All rules	1 range-field	2 range-fields
Fraction of all rules (%)	100%	26%	1.5%
Binary-Prefix [19]	2.68	7.32	47.18
SRGE [17]	2.67	7.27	46.92
INTERNAL-PRODUCT	2.68	7.32	47.18
COMBINED-PRODUCT	1.63	3.38	20.09
COMBINED-SUM	2.45	3.69	8.80

We can see that in the first 11 databases the expansion of the new encoding scheme is smaller than the expansion of the binary encoding as well as of the SRGE encoding. In the last database (ipc2), all the rules are trivial and require only one TCAM entry in each of the three schemes. The expansion of the the Binary-Prefix encoding is reduced by up to 72.46% in the COMBINED-PRODUCT scheme, and by at most 3.25% by the SRGE encoding. The average expansion of the suggested encoding scheme for all the 118217 rules in all the 12 databases is 1.223 instead of 2.330 and 2.312 for the binary and SRGE encoding, respectively. The improvement in the total average expansion in all the databases in our scheme is 47.49% instead of less than 1% in the SRGE scheme.

Of course, please note that the improvement comes with a cost, as our suggested COMBINED-PRODUCT architecture requires additional logic (Fig. 1). Therefore, this comparison is meant to provide some insights on the fundamental TCAM efficiency, and by no means does it imply that this is a straight apples-to-apples comparison.

Effectiveness on Real-life Packet Classifiers: We evaluate the suggested architectures on a real-life database of 120 separate rule files and about $215,000$ rules originating from various applications (such as firewalls, ACL-routers and intrusion prevention systems). The database was previously used in [4], [17], [18], [29]. We compare these architectures with the Binary-Prefix scheme [19] and the SRGE scheme [17]. The results are presented in Table II.

Since the standard INTERNAL-PRODUCT architecture uses the internal Binary-Prefix encoding, it requires the same number of entries as the Binary-Prefix encoding. The SRGE scheme suggests a minor improvement of less than 1% for the average expansion. The COMBINED-PRODUCT architecture performs well and improves by 39.2% the total number of TCAM entries in comparison with Binary-Prefix and in particular by 53.8% and 57.4% the entries needed for rules with (at least) one and two range-fields, respectively. Due to its linear expansion in the number of dimensions, the COMBINED-PRODUCT architecture has much better average expansion of two range-fields. However, its average expansion over all rules is worse than the expansion of COMBINED-PRODUCT mainly because of the low proportion of two range-fields.

Of course, our schemes rely on additional logic, and therefore provide clear tradeoffs between improved range expansion guarantees and more complex logic within the TCAM.