Multi-match Packet Classification on Memory-Logic Trade-off FPGA-based Architecture

Carlos Zerbini
Universidad Tecnológica Nacional
Department of Electrical Engineering, Córdoba, Argentina
czerbini@electronica.frc.utn.edu.ar

Jorge M. Finochietto
Universidad Nacional de Córdoba – CONICET
Digital Communications Lab, Córdoba, Argentina
jfinochietto@efn.uncor.edu

***Abstract*—Packet processing is becoming much more challenging as networks evolve towards a multi-service platform. In particular, packet *classification* demands smaller processing times as data rates increase. To successfully meet this requirement, hardware-based classification architectures have become an area of extensive research. Even if Field Programmable Logic Arrays (FPGAs) have emerged as an interesting technology for implementing these architectures, existing proposals either exploit maximal concurrency with unbounded resource consumption, or base the architecture on distributed RAM memory-based schemes which strongly undervalues FPGA capabilities. Moreover, most of these proposals target *best-match* classification and are not suited for high-speed updates of classification rulesets. In this paper, we propose a new approach which exploits rich logic resources available in modern FPGAs while reducing memory consumption. Our architecture is conceived for multi-match classification, and its mapping methodology is naturally suited for high-speed, simple updating of the classification ruleset. Analytical evaluation and implementation results of our architecture are promising, demonstrating that it is suitable for line speed processing with balanced resource consumption. With additional optimizations, our proposal has the potential to be integrated into network processing architectures demanding all aforementioned features.**

I. Introduction

As packet networks need to support an increasing number of services, classification of their traffic into flows for differentiated processing becomes essential. To this end, steadily increasing data from the packet header must be evaluated at line speed, which traduces in more demanding processing architectures. In particular, the *classification* task has the potential to become the major bottleneck due to the fact that the available processing time decreases as the data rate increases. Indeed, in 100 Gbps networks a packet of 64 bytes has to be processed (i.e., classified) in about 5 ns.

In general, a packet classifier analyzes header fields of each packet to take decisions on it. Each field value is used as a key to access one of the k dimensions of a set of n rules. Each dimension corresponds to one of the header fields, over which each rule specifies an arbitrary range of values. A rule is said to *match* a given packet if its resulting keys are within the ranges defined over their k dimensions.

The problem of packet classification is similar to that of a geometric point location problem. In general, each rule defined over k fields represents an hyperrectangle over a k-dimensional search space. In unidimensional (1D) classification, a rule defines a line segment, while for bidimensional (2D) classification it generates a rectangle. In this context, a packet represents a point in space defined by its keys (i.e., field values) as coordinates, which can belong to one or more hyperrectangles (i.e., rules) since overlaps can occur. These overlaps divide the search space in *hyperregions*, each of which has an associated set of rules. The main goal of a packet classifier is to identify the hyperregion a packet belongs to.

Moreover, two different classification problems are of interest: *best-match* and *multi-match*. The former only returns the highest priority matching rule, while the latter, all matching rules. Best-match has been widely studied and multiple approaches adopt it as motivation due to its use in longest prefix match for IP lookup, which is the main mechanism enabling the Internet up to date. However, a growing number of applications such as traffic balancing and accounting, differentiated services, Network Intrusion Detection Systems (NIDS), etc. are pushing the need for high-performance multi-match classification schemes.

In this paper, we consider the use of FPGA-based architectures for multi-match packet classification, where the classifier is asked to return the set of rules matched by a given packet. In particular, we consider the *aggregation* stage needed to determine this set of rules from partial matches on each dimension. We propose a novel aggregation scheme for this stage based on logic in contrast with most available solutions in the literature which are memory-based. This scheme is suitable for implementation on FPGAs, where logic resources are abundant through the use of logic blocks containing look-up-tables (LUTs) and registers, leaving RAM memory resources available for partial 1D lookups and additional processing tasks. As a result, our proposal provides a trade-off implementation for packet classifiers which can better exploit available resources on FPGAs. Besides, our aggregation scheme provides 1-cycle computation time and preserves a simple structure for updating rules.

The paper is organized as follows. Section II reviews relevant work related to our proposal, while Section III discusses the main drivers for it. A detailed description of the proposed architecture is presented in Section IV, with particular emphasis on the aggregation stage which is required to reduce overall memory consumption. This stage is evaluated and compared with other approaches in Section V, highlighting existing trade-offs. Finally, we conclude the paper and discuss future work in Section VI.

II. Related work

Approaches for multi-dimensional packet classification can be coarsely divided in *trie-based* and *decomposition-based* [1]. As our work follows the latter, we briefly review existing work, specially aiming at the aggregation stage where two main research lines can be identified:

Bitmap intersection[1] or region-based approach (REG). This approach is essentially based on the identification of distinct *regions* which can be associated to a set of matched rules. In its seminal work, *Parallel Bit Vector* (BV) [2], all rules are projected on each dimension and a maximum of $2n-1$ non-overlapping intervals are identified. Unidimensional searches can be made, for example, through binary search. A set of matched rules is stored for each region as an n-bit map (bitmap). Due to this biunivocal bit-rule mapping, the intersection between all partial results reduces to an extremely simple AND (i.e., conjunction) operation between each bit vector. The simplicity of the intersection comes at the cost that global information regarding the complete set of rules must be stored for each dimension. BV bitmap requires n bits per each of the $2n-1$ (i.e., $O(n)$) possible intervals on each of the k dimensions, trading unfavorable space requirement of $O(kn^2)$ for $O(k \log n + n/w)$ time complexity, when using binary-search on each field and memory of width w [2].

Recursive Flow Classification (RFC) [3], while still based on *bitmap intersection*, states a turning point in the *aggregation* stage by adopting a distributed approach based on minimum-size *labels* representing unique hyperregions. During classification, these labels are combined recursively into intermediate ones until that representing the decision for the packet is obtained. Updating algorithms (commonly running on general-purpose processors) keep lists of bitmaps on each dimension for pre-computing the respective labels to be stored in limited, classifier-embedded memory. This fundamental change in aggregation methodology aims at optimization of memory consumption. Space requirement is reduced to $O(tn^2)$ (t=aggregation stages) with $O(k)$ time complexity. However, the simplicity of original BV algorithm is lost by considering ruleset-specific pre-computation; that is, the correspondence between stored bits and global rules is no longer transparent, but must be established by pre-processing. Such algorithms are implemented in software, while aggregation is tied to sequential access to big memory blocks with limited concurrent operation opportunities.

Crossproducting- or value-based approach (XPROD). Originated from the seminal contribution by [4], this approach is based on the identification of *unique single values* defined from the projection of rules on each dimension. For aggregation, an optimized precomputed hash table is built storing the best rule match for each possible combination of single-field matches (i.e., crossproducts). During lookup, best matches for each field are obtained and used for accessing this table in a single step. However, due to the multiplicative nature of crossproducts, many *pseudorules* must be stored for resolving rule overlappings. These pseudorules are filters not present in the original ruleset, defined by single values from overlapped rules. This effect leads to exponential increase in memory requirements. In the worst case, space consumption is $O(n^k)$ with time requirements $O(d \log n + 1)$. This scheme was implemented in software running on a general-purpose processor based system.

To overcome crossproducting drawbacks, *Distributed Crossproducting of Field Labels (DCFL)* [5] exhaustively analyzes sample real rulesets and extracts following features:

a) the maximum number of unique field values is significantly less than the number of filters. This is due to unique field values being shared by multiple filters.

b) the maximum number of unique field values *matching any packet* is strongly limited and remains relatively constant for various filter set sizes.

c) the maximum number of *unique combinations of field values matching any packet* is bounded by twice the maximum number of matching single-field values.

Based upon these observations, a label-based aggregation scheme using Bloom filters and memory indexing was developed. Through distributed aggregation, combinations not present in the ruleset (i.e., pseudorules) are "filtered" at early stages, propagating only valid combinations for further crossproducting. Implementation results are not reported in this case, while focusing on extensive evaluation of memory consumption for sample rulesets.

While DCFL could be intuitively associated to RFC, a fundamental difference must be noted. RFC stores labels representing distinct filter overlapping regions in a k-dimensional aggregation stage. DCFL, meanwhile, assigns labels to unique field values. Both types of labels are illustrated for 1D in Fig. 1 for a typical 2D aggregation case.

Later work by Jedhe *et. al.* [6], specially aimed at reconfigurable hardware (i.e., FPGA technology), goes a step further bounding n to 128, a) to 32 and b), c) to ≤ 5. On this basis, a pure memory-based approach (without use of Bloom filters) is implemented. The number of sequential memory accesses (SMAs) is strongly tied to the maximum number of matched labels for any packet, which in this case is supposed to be ≤ 5. Implementation on FPGAs is reported in this case which makes strong use of memory resources, achieving 50 Mpps for 128 rules on a Xilinx Virtex II Pro FPGA.

An important scalability metric of these distributed schemes is the ratio between *effective* matches obtained in an aggregation stage and the number of SMAs needed for it. At a certain stage, SMAs are the product of effective matches of previous stages, while according to c) the number of effective matches obtained is bounded by twice the maximum of matches of previous stages. This leads to increasing cost for higher dimensions. For example, if 7 single-field matches are considered, $7 \times 7 + 14 \times 14 = 245$ memory acceses can be required to obtain just $14 \times 2 = 28$ effective matches. As SMAs cause the pipeline to stall, this fact can turn into an important issue.

Pre-computation required in both REG and XPROD approaches can be divided into two main stages: 1) single-field matching, and 2) aggregation. Single-field matching usually

[1]The terms *Bit Vector* and *Bitmap* are frequently interchangeably used in the literature. We adopt the term $Bitmap$ for the general case where one bit biunivocally represents some kind of match, while we reserve *Bit Vector* to explicitly refer us to the BV approach.

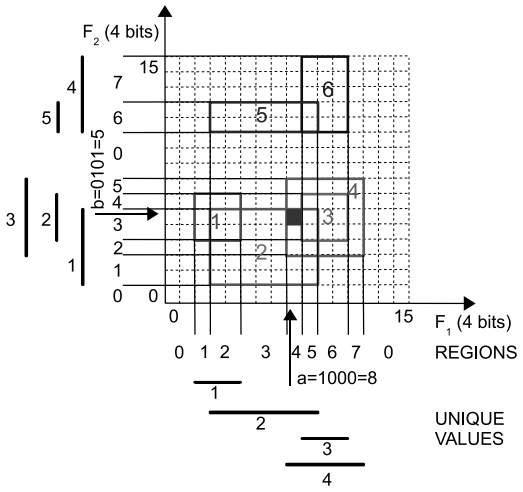

Fig. 1. Regions versus unique values for 2D aggregation case

involves moderate pre-computation of 1D regions [1]. In BV, 1D regions are then biunivocally mapped to n-bit vectors for pre-computing-free stage 2). XPROD, on the opposite end, maps 1D regions to field-local values, so stage 2) is in complete charge of mapping them to specified rules, requiring extensive pre-computation. RFC distributes pre-computation between both stages, since it considers k-D regions at both 1) and 2). XPROD techniques advantage REG ones when their aforementioned conditions are met, as they enable considerable memory savings with proper throughput. If the number of matching labels on each field where large, excessive latency would override memory savings in favor of REG techniques.

III. Motivation

From the preceeding analysis, and aiming hardware-assisted implementation on FPGAs, we can state the following premises:

a) BV bitmap makes very inefficient use of memory resources due to its intrinsic $1rule/1bit$ mapping scheme. Considering rule properties is certainly the way out from BV bindings. Label-based approaches, however, extremely alter this simple aggregation scheme requiring hardware-unfriendly, computation-intensive algorithms for $rule/label$ mapping. Moreover, labels dictate the use of memory-based architectures with limited possibilities of concurrent operation. From this considerations, bitmap-based aggregation schemes which can take advantage of rule properties seem to strike a convenient balance.

b) Pre-computation present at *aggregation stage* should be addressed, rather than pre-computation at 1D stages.

c) XPROD labels should be used, as they scale to n for n rules defined on k dimensions. REG labels, meanwhile, scale to the number of k-D regions, which is $2^{r_j} \leq \sum_{i=1}^{n} i$. That is, XPROD labels scale better than REG labels. This is because k-D regions are $disjoint$, while k-D unique values can overlap with each other.

d) Current FPGA-based work adopts either logic-intensive [5] or memory-intensive [6] approaches. While the former can suffer from degraded performance due to large routing paths, the later ignores rich and optimized logic resources available in FPGAs. We intend to achieve a balance by taking advantage of both kinds of resources. This also allows us to relax memory consumption constraints in favor of enhanced throughput.

In this context, we explore a convenient tradeoff between the simplicity (but intensive resource consumption) of the original BV scheme and the resource efficiency (but intensive pre-computation) of RFC and XPROD-based schemes. To this end, from observation c), we detect regions at early stages and map them to *disjoint bitmaps of unique values*. 1D lookup incurs acceptable pre-computation according to b), while bitmaps reduce pre-computation at aggregation stages as intended, according to a).

Our proposal has similarities with work by Sun *et al.* [7], however a fundamental difference exists between both. Sun *et al.* aims at compressing BV bitmaps by the use of a bounded number of concatenated *rule values*. Their aggregation array reduces to multiple, concurrent *conjunctions* as is the case of BV, accordingly requiring each single-field lookup to return k-dimensional *rule numbers*. We instead employ *unique-field values*, completely local to single-field lookup engines. Our aggregation array, therefore, must be able to perform distributed classification through incremental *aggregation* of partial results.

IV. Proposed Architecture

In previous work, described in detail in [8], we implemented the simple aggregation method of BV while using indexing on chunks for single-field lookup on FPGAs. We enhance that implementation through the new aggregation architecture to bring our present ideas to practice and test its performance on state-of-the-art FPGA devices. We briefly review relevant aspects of the general architecture, in order to introduce the proposed aggregation stage.

A. Unidimensional lookup stage

Unidimensional lookups are implemented through block memory indexing, where the resulting key from the packet is used as a memory address to fetch the associated set of matched rules. To avoid address space explosion, the key can be divided in *chunks* of fixed size [3] [9].

Our previous implementation [8], where no pre-computation at all is required, is based on *key/rule* mapping memory blocks. That is, the matching state of each rule is stored for each key value. This approach, named *CAM emulation*, is completely ruleset-agnostic, even more than the BV scheme. It requires $\lceil f_j/c \rceil \times 2^c \times n$ storage, where f_j= key width for field j [bits], c = chunk width [bits], and n = number of rules.

A straightforward enhancement resembles the BV approach, by introducing a first *key/region* mapping memory stage. For each dimension j, region labels of width r_j are defined and mapped by a first memory block named *reg_mem*. Each region is, in turn, linked to a unique rule bitmap through a second *region/rule* memory block, *bmp_mem*. As $r_j \leq n$, the total memory consumption is so reduced to $\lceil f_j/c \rceil \times 2^c \times r_j + 2^{r_j} \times n$. Both unidimensional schemes are illustrated in Fig. 2(a) and (b) respectively for the match case shown in Fig. 1.

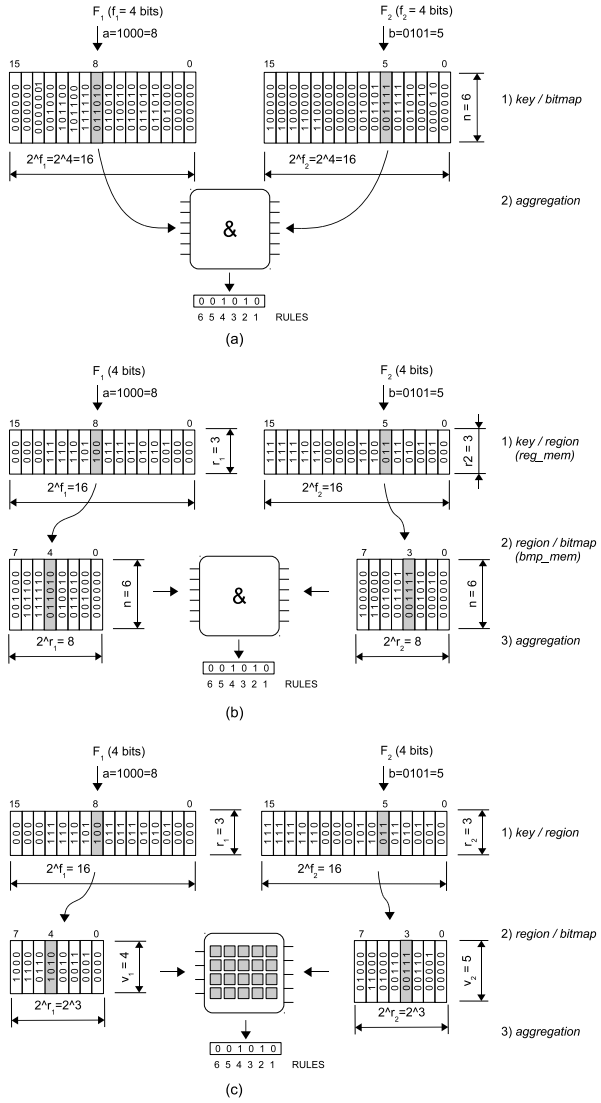

Fig. 2. General architecture: (a) CAM emulation scheme, (b) BV scheme, (c) proposed scheme

The address space of *reg_mem* is given by 2^{f_j} (or 2^c if $c < f_j$). From the condition $r_j \leq n$, it contributes with a small percentage to the 1D stage memory consumption. Address space of *bmp_mem* is strongly bounded by the number of regions resulting in a positive global balance, but it still contributes with most of the storage due to its required width n for aggregation purposes. Regarding *address space* of *bmp_mem*, previous work [10] proposes encoding of ranges into multiple *primitive ranges* by setting bounds on the number of overlaps. The main goal of that approach is to avoid update complexity of RFC while reducing the excessive resource requirements of Parallel BV. This proposal can be easily applied to our implementation, with obvious pre-computation penalities.

We hereafter focus on reducing the *word width* of *bmp_mem*. As stated in our proposal, *unique value bitmaps* can be used in order to explore a tradeoff between label- and rule bitmap-based aggregation approaches. This in turn requires a suitable aggregation stage, which we analyze and evaluate in the rest of the paper. A general view of the proposed scheme is shown in Fig. 2(c).

B. Aggregation stage

Aggregation in our previous work consisted on simple, pipelined bit-level conjunction (AND) of n-bit bitmaps from unidimensional *bmp_mem* blocks. These bitmaps represent global rules which tend to demand much memory resources. Instead, we propose to store bitmaps which are *local* to each dimension (i.e. field), with its bit positions representing *unique field values*. As a result, memory consumption can be reduced at the cost of increasing the complexity of the aggregation stage. Indeed, the *combinations* of their elements (i.e., bits) rather than their *conjunction* must be now checked. Two states are related to the combination of bitmap elements: *valid*, which is stored during rule updating, and *match*, to be set during classification.

Let us consider two field bitmaps A and B of widths (i.e., number of unique values) v_A and v_B respectively, resulting in an aggregated bitmap C of width v_C. From possible $v_A \times v_B$ combinations of their possible values (i.e., their cartesian product), only those which are (a) *valid* in the set of rules and (b) *matched* by the incoming packet header will result in relevant outcomes. Combinations which obey *both* conditions are extremely limited, as argued for disregarding SMAs of XPROD-based approaches [5]. For our present purposes, we only take advantage of condition (a) for assuming some $v_C < v_A \times v_B$ and cover such combinations through high-speed, concurrent matching.

In order to keep Processing Elements (PEs) as simple as possible and to keep routes short on FPGA, a number of possible pipelining schemes are applicable with varying space-time tradeoffs [11]. A simple 1-dimensional (1-d)[2], bit-grained (1-b) aggregation pipeline, is illustrated in Fig. 3(a) for the case of two 3-bit bitmaps A and B. In this figure, points stand for delay elements (DE) implemented through flip-flops, V stands for *valid* state (valid unique-field combination in ruleset), and M stands for *match* state of each combination. For $v_A = 3$ and $v_B = 3$ unique field values in A and B fields respectively, the 1-d 1-b aggregation pipeline consists of $v_A \times v_B = 3 \times 3 = 9$ PEs. The number of input DEs, meanwhile, is $2 \times \sum_{i=1}^{v_A \times v_B} i$. An essential aspect of this scheme regards the *aggregation bitmap* to be used. Unlike similar pattern-matching pipelines proposed in [12], where just one bit (match) is propagated, we must keep track of *multiple matches* against valid rules. For sake of hardware simplicity, we implemented aggregation bitmaps on shift registers. If *valid=1*, the register is shifted one place with the match state for the particular combination, as shown in the PE of Fig. 3(b). An aggregation example through this scheme is also illustrated for the ruleset of Fig. 3(c) and packet $P_{A=a,B=b}$.

The scalability of this simple scheme soon degrades, but can be improved as follows. Given that bit combinations *internal* to A and B are not considered for aggregation, unique values in A can be checked in parallel against those in B, obtaining parallel partial results of width v_A after v_A clock cycles. At the output of each of the so-defined rows, a barrel shift controlled by the number of *valids=1* prepares

[2]We use 1D to refer to unidimensional classification, while 1-d to refer to 1-dimensional pipeline

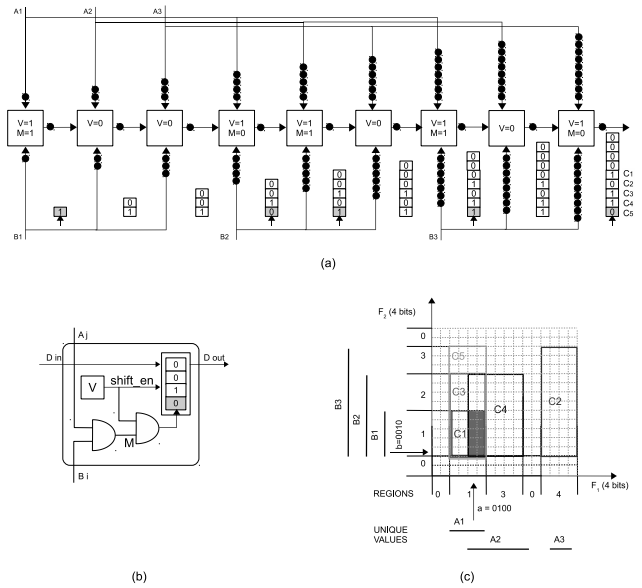

Fig. 3. 1-d, 1-b architecture: (a) aggregation, (b) PE, (c) Match case

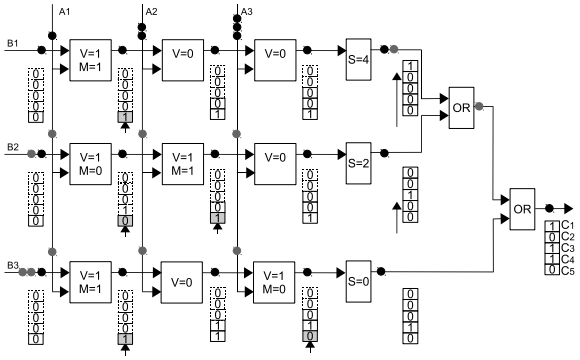

Fig. 4. 2-d, 1-b architecture

the partial bitmap for final aggregation. Finally, an output stage aggregates the partial results from each row through union (ORing) into a match vector of width v_C. As one of the ports is broadcasted, as shown in Fig. 4, we will refer to this architecture as 2-d 1-b *semi-systolic* [11]. A further improvement involves 2-d pipelining and fine-grain pipelining of the output OR stage as shown in additional grey points of Fig. 4. This architecture will be called 2-d 1-b *pure-systolic*. Unlike [7], aggregation bitmaps must not only return single matches for each row but *all valid matches* on it; therefore smaller, row-local shift register-based bitmaps are again adopted.

From Fig. 3(b), we can observe that computation at PEs is trivial, while significant overhead is required to aggregate results at bit-level. We can achieve a better balance by adding processing capability at elements and accordingly relaxing aggregation granularity G. Resource consumption, latency and speed are improved at cost of slight increase of PE complexity. Aggregation bitmap, meanwhile, is slightly changed to support variable-sized shifts at each PE. This modification is illustrated for $G = 2$ bits in Fig. 5(a), resulting in 2-d 2-b semi- (black) and pure-systolic (black + grey) schemes. High-level implementation of the new processing element is shown in

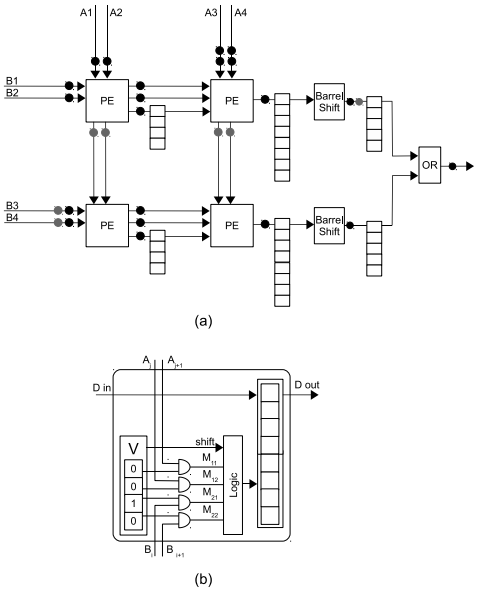

Fig. 5. 2-d, 2-b architecture: (a) aggregation datapath, (b) processing element

Fig. 5(b).

It is worth to note that all proposed schemes preserve *1 rule/1 bit* mapping. Inserting/changing a rule implies simple decoding of its fields and accordingly setting *valid* bits. At the same time, they trade high memory consumption of BV for rich logic resources available in FPGAs. This simplicity is possible thanks to the massive concurrency of our fine-grain pipeline architecture.

V. Results

Main performance metrics for packet classifiers are, namely, (a) resource consumption, (b) speed, and (c) updating capabilities. Metric (a), in turn, can be divided in (a1) registers, (a2) combinational logic (LUTs), and (a3) memory blocks (i.e., SRAM). Metric (a) will be analyzed in detail in the first subsection, while it is checked along with (b) on real FPGA implementation in the second one. Even if updating capabilities are already implemented in our aggregation architecture, analysis of metric (c) also requires complete implementation of preprocessing at 1D stage. Such algorithms are common to all REG and XPROD schemes, and at the time of writing are added to our platform for complete evaluation of updating capabilities. As we were intentionally careful in preserving *1rule/1bit* mapping, however, moderate updating complexity (i.e., similar to that of BV) is to expect.

A. Performance analysis

For sake of clarity, we must narrow our performance analysis to a representative trade-off configuration. Our first, 1-d 1-b pipeline, on the one hand, has poor scalability as discussed so it is not further analyzed. On the opposite end, pure-systolic architectures demonstrated modest returns for our medium-sized test cases on FPGAs, even if they should be the best option for larger cases. We will then focus our present analysis on semi-systolic architectures whithout loss of generality.

In order to evaluate gains as aggregation granularity varies, we first concentrate on aggregation architecture and compare its resource consumption for 1-b, 2-b, 3-b, and 4-b aggregation granularities. Both LUT and register consumption vary with changing granularity. LUT consumption, however, is not simple to estimate for coarse granularities, since synthesis tools would optimize LUT usage. We thus evaluate register consumption at this subsection, while in the following subsection we check that register estimations are correct and that LUT consumption is not a limitation. In following analysis, we will refer to our proposal as *Distributed Crossproduct of Field Values (DCFV)* as opposed to DCFL which is based on labels.

As it can be checked in Fig. 4, the 2-d 1-b semi-systolic implementation consumes $\sum_{i=1}^{v_A} i + v_B + v_A v_B + v_B \sum_{i=1}^{v_A} i + v_B v_C + v_C + v_A v_B$ registers, while the 2-d 2-b semi-systolic version of Fig. 5(a) reduces it to $\sum_{i=1}^{v_A} i + v_B + \lceil \frac{v_A}{2} \rceil v_B + \lceil \frac{v_B}{2} \rceil \sum_{i=1}^{\lceil \frac{v_A}{2} \rceil} 4i + \lceil \frac{v_B}{2} \rceil v_C + v_C + 4 \lceil \frac{v_A}{2} \rceil \lceil \frac{v_B}{2} \rceil$ registers. Coarser granularity cases (i.e., 3-b, 4-b, and so on) can be similarly checked. Consumptions of (from top to bottom) 1-b, 2-b, 3-b and 4-b granularities are compared in Fig. 6(a) for typical values $v_A = v_B = 16$ (four bottom curves), $v_A = v_B = 32$ (four middle curves), $v_A = v_B = 64$ (four top curves), and $0.1 \leq \frac{v_C}{v_A v_B} \leq 0.5$. For $v_A = v_B = 32$, e.g., $v_C \leq .5 \times 32 \times 32 = 512$ is covered, which represents quite realistic overlappings. In Fig. 6(b), meanwhile, storage consumption is evaluated for a mean case $v_C = 100$ with varying v_A and v_B. In this case, v_A and v_B are controlled so that $\lfloor \frac{v_A v_B}{v_C} \rfloor = 1$; i.e., the cartesian product stays as near to the crossproduct as possible. As shown, the 2-d 1-b aggregation clearly has the highest resource consumption in both cases. 3-b and 4-b aggregation, meanwhile, trades modest resource reduction for steadily increasing PE complexity. Thus, we chose 2-b aggregation for further evaluation.

Our second analysis considers resource consumption against those of typical REG and XPROD approaches, i.e., BV and DCFL (namely [6] for the FPGA case). For proper evaluation, it is essential to target key differences with respect to each of them. To this end, storage consumed by two generic *bmp_mem* blocks and one aggregation stage is considered for each approach. Only memory consumption at *bmp_mem* of width v_C is analyzed for BV, since aggregation implies just v_C parallel 2-input AND gates. For our proposal, meanwhile, two *bmp_mem* memory blocks of width v_A and v_B respectively and registers at systolic array are considered separately. In order to fairly compare against DCFL, finally, both of its *bmp_mem* blocks are supposed to have width $5 \times log_2 v_A$ and $5 \times log_2 v_B$ respectively according to mentioned considerations of [6], while aggregation memory consumption is added to them. In the same way as Fig. 6, v_A and v_B are kept constant in Fig 7(a) for varying crossproduct v_C, while in Fig. 7(b) v_C is constant against controlled-varying cartesian product $v_A \times v_B$. Regions defined in 1D, which affect *bmp_mem* addressing space, are considered rather pessimist $2v_A - 1$ and $2v_B - 1$ respectively in all cases.

BV shares with us the use of memories storing bitmaps for simple bitwise aggregation. It however pays for extremely simple aggregation by being unable to exploit size differences between v_A, v_B and v_C as clearly shown in Fig. 7. DCFL, on the opposite end, exploits optimal $log_2 v_A$, $log_2 v_B$ and $log_2 v_C$ size labels at the cost of purely sequential accesses. That is, it is completely unable to represent more than one unique value per label. Our architecture, meanwhile, trades moderate additional resource consumption for single-cycle multimatch operation. It is also worth to note that, as we do not stick to observation c) of DCFL, line speed multi-match operation is only bounded by the considered maximum number v_C of *valid combinations* of field values, a much more tolerant consideration than the maximum number of *matched* field values on which DCFL bases its efficiency.

TABLE I. 1-BIT AGGREGATION, SEMI-SYSTOLIC ARCHITECTURE

v_A_v_B_v_C	Mpps	Registers	LUTs
8_8_16	717	588	450
8_8_32	735	732	450
16_16_32	545	3352	1314
16_16_64	528	3896	1314
32_32_64	386	21552	4262

TABLE II. 2-BIT AGGREGATION, SEMI-SYSTOLIC ARCHITECTURE

v_A_v_B_v_C	Mpps	Registers	LUTs
8_8_16	704	254	62
8_8_32	714	356	82
16_16_32	717	1570	282
16_16_64	656	1896	322
32_32_64	493	10442	1202

B. Performance evaluation on FPGA

In order to check previous analysis, and to explore speed bounds on modern devices, semi- and pure-systolic, 2-d 1-b and 2-d 2-b architectures were implemented, simulated and synthethized on a state-of-the-art Altera Stratix V FPGA. Available resources for our tests are 370000 LUTs, 740000 registers, and 2100x20 Kbit Static RAM memory blocks. Implementation results are focused on our new aggregation stage, since 1D lookup stage is mainly based on Static RAM and already reported in [8]. In order to fix some bound for v_C, it was supposed to be at most twice $max(v_A, v_B)$.

Post place-and-route results are shown in Tables I and II for 2-d semi-systolic 1-bit and 2-bit granularities respectively. We effectively confirm that coarser granularity greatly reduces register consumption to about one half, while additionaly achieving optimal LUT consumption due to optimized synthesis. Operation speed, meanwhile, keeps well above 310 Mpps (i.e., 100 Gbps @ 40-byte packets) for all considered configurations. It should be noted that, due to the adopted architecture, influence of v_C on register consumption is minimized, so variation of v_C while keeping v_A and v_B constant has minimal impact as shown. On the other side, variation of v_A or v_B in steps G (2 in this case) strongly affects resource consumption of the array.

VI. CONCLUSION

In this paper we proposed and discussed a novel multi-match packet classification architecture which exploits both rich logic resources as well as RAM memory ones available in modern FPGAs. To assess its value, we performed analytical evaluation while comparing with existing results. From implementation results, our architecture has demonstrated to

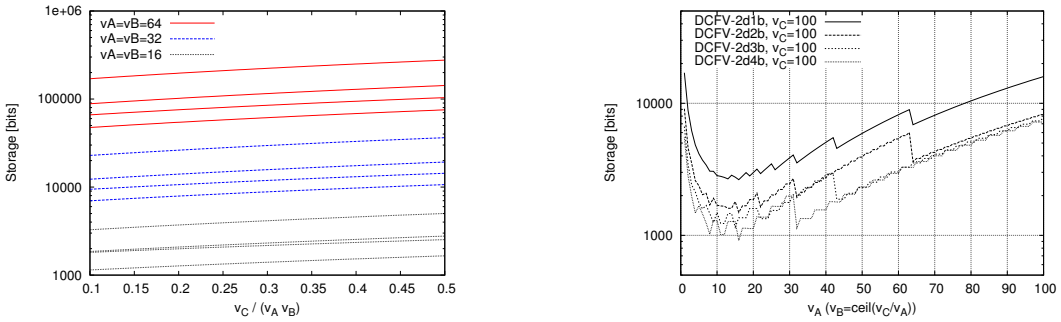

Fig. 6. Resource consumption: (a) with varying v_C, (b) with varying v_A and v_B

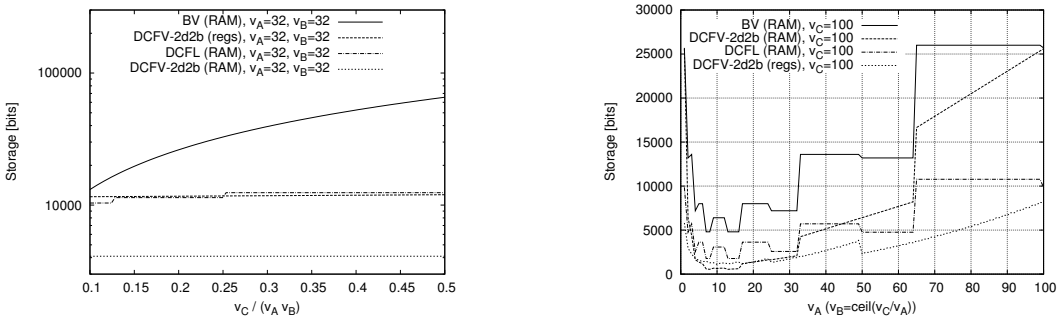

Fig. 7. Resource consumption of BV vs. DCFV vs. DCFL: (a) with varying v_C, (b) with varying v_A, v_B

be suitable for line speed processing whith balanced resource consumption. On this basis, we can conclude that our approach has interesting opportunities for application in modern packet networks.

It must be mentioned that, even if unique-value bitmap has reduced width with respect to BV, it still bases on *bit weights*. It is as such unable to fit resource consumption to the number of effectively *matched* rules as DCFL does, but fits *valid* values on each dimension (no matter if matched or not). As a consequence, unique-value based *matched* bitmaps can be moderately sparse (i.e., contain several 0's) as the second stage of Fig. 2(c) shows, standing at a midpoint between BV (Fig. 2(b)) and DCFL. As future work, utilization rate could be enhanced by (a) allowing for multiple packets/cycle classification, or (b) adopting hashing techniques.

ACKNOWLEDGMENT

This work was partially funded by the FONCyT-National Technological University IP-PRH 2007 Posgraduate Grant Program, FONCyT PICT 2011-2527 Research Grant, and Altera Corporation.

REFERENCES

[1] G. Varghese, *Network Algorithmics*, Morgan Kaufmann, 2005.

[2] T. V. Lakshman and D. Stiliadis, *High-speed policy-based packet forwarding using efficient multi-dimensional range matching*, in ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication SIGCOMM '98, pp. 203-214, 1998.

[3] P. Gupta and N. McKeown, *Packet classification on multiple fields*, in ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication SIGCOMM '99, pp. 147-160, 1999.

[4] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, *Fast and scalable layer four switching*, in ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication SIGCOMM '98, pp. 191-202, 1998.

[5] D. E. Taylor and J. S. Turner, *Scalable packet classification using distributed crossproducting of field labels*, in IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM 2005, vol.1, pp. 269-280, 2005.

[6] G. S. Jedhe, A. Ramamoorthy, and K. Varghese, *A scalable high throughput firewall in FPGA*, in 16th International Symposium on Field-Programmable Custom Computing Machines FCCM '08 , pp. 43-52, 2008.

[7] L. Sun, H. Le, and V. K. Prasanna, *Optimizing Decomposition-Based Packet Classification Implementation on FPGAs*, in International Conference on Reconfigurable Computing and FPGAs ReConFig 2011, pp. 170-175, 2011.

[8] C. Zerbini and J. M. Finochietto, *Performance evaluation of packet classification on FPGA-based TCAM emulation architectures*, in IEEE Global Telecommunications Conference GLOBECOM 2012, pp. 2790-2795, 2012.

[9] T. Ganegedara and V. K. Prasanna, *StrideBV: Single chip 400G+ packet classification*, in IEEE 13th International Conference on High Performance Switching and Routing HPSR 2012, pp. 1-6, 2012.

[10] J. Van Lunteren and T. Engbersen, *Fast and scalable packet classification*, in IEEE Journal on Selected Areas in Communications, vol. 21, pp. 560-571, 2003.

[11] H. T. Kung, *Why systolic architectures?*, Computer, vol.15, no.1, pp. 37-46, Jan. 1982.

[12] M. J. Foster and H. T. Kung, *The Design of Special-Purpose VLSI Chips*, Computer, vol. 13, no.1, pp. 26-40, Jan. 1980.