Saad Saleh, Anouk S. Goossens, Sunny Shu, Tamalika Banerjee and Boris Koldehofe. **Analog In-Network Computing through Memristor-based Match-Compute Processing**. In Proceedings of the 43rd International Conference on Computer Communications (INFOCOM 2024), IEEE, 10 pages, 2024.

Analog In-Network Computing through Memristor-based Match-Compute Processing

Saad Saleh*‡, Anouk S. Goossens†‡, Sunny Shu*, Tamalika Banerjee†‡, Boris Koldehofe§‡
*Bernoulli Institute, University of Groningen, Netherlands
†Zernike Institute for Advanced Materials, University of Groningen, Netherlands
‡CogniGron (Groningen Cognitive Systems and Materials Center), University of Groningen, Netherlands
§Department of Computer Science and Automation, Technische Universität Ilmenau, Germany
s.saleh@rug.nl, a.s.goossens@rug.nl, s.shu@student.rug.nl, t.banerjee@rug.nl, boris.koldehofe@tu-ilmenau.de

***Abstract*—Current network functions consume a significant amount of energy and lack the capacity to support more expressive learning models like neuromorphic functions. The major reason is the underlying transistor-based components that require continuous energy-intensive data movements between the storage and computational units. In this research, we propose the use of a novel component, called Memristor, which can colocalize computation and storage, and provide computational capabilities. Building on memristors, we propose the concept of match-compute processing for supporting energy efficient network functions. Considering the analog processing of memristors, we propose a Probabilistic Content Addressable Memory (pCAM) abstraction which can provide analog match functions. pCAM provides deterministic and probabilistic outputs depending upon the closeness of match of an incoming query with the specified network policy. pCAM uses a crossbar array for line rate matrix multiplications on the match outputs. We proposed a match-compute packet processing architecture and developed the programming abstractions for a baseline network function, i.e., Active Queue Management, which drops packets based upon the higher-order derivatives of sojourn times and buffer sizes. The analysis of match-compute processing over a physically fabricated memristor chip showed only 0.01 fJ/bit/cell of energy consumption, which is 50 times less than the traditional match-action processing.**

***Index Terms*—Energy Efficiency; Switches; Memristors.**

I. Introduction

The Internet heavily uses complex network functions like congestion control [1], load balancing [2], packet scheduling [3], traffic analysis [4] [5], etc., in order to maintain the Quality of Service (QoS) and apply network policies for end users. Building on packet processors in routers and switches, network functions operate at remarkable line rate performance of 12.8-51.2 Tbps [6] [7]. A downside which is often neglected is the enormous energy footprint of such components e.g., 240-340 TWh to transport data between senders and receivers (equivalent to 10% of world's nuclear power generation) [8]. They also have limited ability to support more expressive learning models, like *Cognitive* functions using neuromorphic computing [9]. One major reason for these shortcomings is the architecture of packet processors which heavily builds on Ternary Content Addressable Memory (TCAM). TCAM memory nowadays builds on transistor-based components which require continuous data movements between the storage and match units (consuming up to 90% of system's energy [10]).

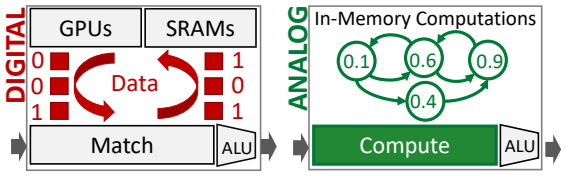


Fig. 1: Energy savings by limiting data movements in the match-compute processing vs the match-action processing.

Moreover, it lacks the capacity to support computations, like matrix multiplications, in the data plane required for learning models. Overcoming such shortcomings requires a shift to technologies with the ability to colocalize computation and storage in a single component, and enhance the computational ability of packet processors. A highly promising technology with this ability is the *Memristor* with substantial industrial effort and well-established performance, but less known to the research community [11] [12].

Memristors are nanoscale programmable components that can store data in the form of a physical property, called Resistance. Built upon the principles of in-memory computing, memristors can perform computations by providing an output depending upon the input and the resistance [13]. The efficient integration of such components in packet processors, however, requires to take into account their analog nature. In this paper, we focus on the problem of integrating such components by proposing appropriate packet processing abstractions which also work in the analog domain. In particular, we aim to enhance the traditional match-action processing, where header fields are matched against locally stored rules to take corresponding actions, by a new concept we call match-compute processing. Match-compute processing enables the execution of computation operations in the data plane (Fig. 1). Building on [14], we introduce the Probabilistic Content Addressable Memory (pCAM) abstraction for supporting the match-compute processing and demonstrate for memristor-based components how to build the abstractions. The pCAM abstraction provides analog match functions by computing the closeness of an incoming query with the stored network policy. It can be programmed to provide a discrete programmed output (maximum or minimum) for a given range, called as

Saad Saleh, Anouk S. Goossens, Sunny Shu, Tamalika Banerjee and Boris Koldehofe. **Analog In-Network Computing through Memristor-based Match-Compute Processing**. In Proceedings of the 43rd International Conference on Computer Communications (INFOCOM 2024), IEEE, 10 pages, 2024.

deterministic output, to show a complete match or mismatch. It can also be programmed to compute an analog output for a given range of input, called as probabilistic output, to indicate a partial match. The deterministic and probabilistic outputs of multiple parallel pCAM cells are used as inputs to a programmable crossbar array. The crossbar array multiplies the pCAM outputs with the programmed weights and computes the output function by taking into account all parallel matches. In this paper, we demonstrate that the analog processing of pCAM-based match-compute processing is effective and efficient in implementing advanced network functions, like Active Queue Management (AQM). With the help of analog match-compute, Packet Drop Probability (PDP) can be modeled based on the analog higher-order derivatives of sojourn times and buffer sizes. In addition, the design of pCAM relying on analog components allows for substantial energy savings.

Contributions and Research Findings. In this paper, we propose the match-compute based packet processing for supporting energy efficient network functions and analyze its performance over a physically fabricated memristor chip developed for this research[1]. Our major contributions include; (1) Development of a novel match-compute framework for packet processors; (2) Proposition of the packet processing architecture built over the pCAM abstraction; (3) Development of the programming abstractions for the match-compute processing; (4) Analysis of the match-compute processing for a network function, i.e., AQM, for a physically fabricated memristor chip of Nb-doped $SrTiO_3$. Our analysis over the experimental data set of the memristor chip showed only 0.01 fJ/bit/cell of energy consumption for match-compute processing which is 50 times less than the state-of-the-art match-action processing. We estimated the PDP for an AQM function based on the higher-order derivatives of sojourn times and buffer sizes. The match-compute based AQM showed better support for congestion management than the prior techniques due to the incorporation of line rate computations.

Paper Organization. Sec-II presents the match-action processing, memristors, and the problem statement. Sec-III presents the proposed match-compute processing. The packet processing architecture and programming abstractions are presented in Sec-IV. The performance analysis of the proposed match-compute processing is shown in Sec-V. Sec-VI presents the literature review and Sec-VII concludes the paper.

II. Background

In this section, we present the limitations of match-action processing, features of memristors, and the problem statement.

A. Match-Action Processing and Limitations

Match-action processing is used in switches and routers for matching the incoming header fields against stored policies and applying the highest priority match among a set of matching policies. Existing architectures do matching in one clock cycle to ensure line rate performance with features like prefix matching. However, match-action processing per se is not sufficient to support all kinds of network functions, like AQM can be processed via match-action processing but requires external computational components. In consequence, additional continuous data movements are required which are considered expensive in terms of performance and energy usage [15]. It is also important to observe that the current designs of match-action processing in the form of TCAM are known to be expensive in terms of the cost in resources and energy consumption. The architecture requires typically a substantial amount of transistors. Increasing the deployed policies in the design imposes a significant cost in terms of energy since the underlying design lacks scalability. Finally, the transistor-based design can keep the state of policies only in a volatile operation and requires a continuous power supply.

[1]Anouk S. Goossens and Tamalika Banerjee fabricated the Nb-doped $SrTiO_3$ memristor chip.

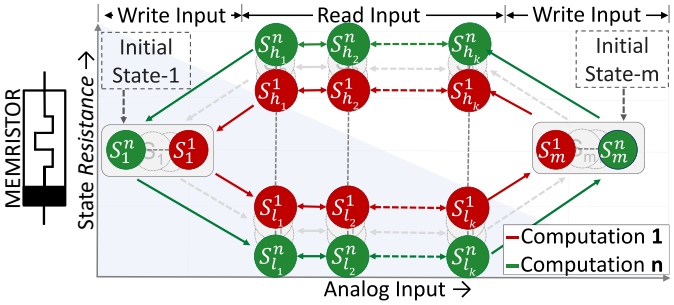


Fig. 2: The input-state response for a memristor [14].

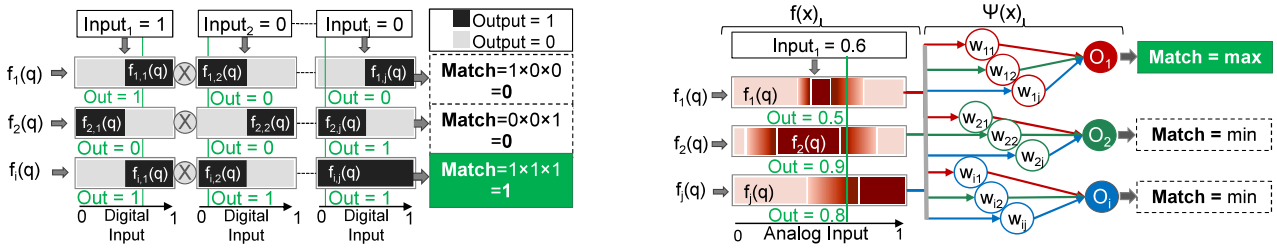
B. Memristor-based In-memory Computing

The memristor is a novel nanoscale component, which has drawn recently significant attention from industry as a possible replacement for traditional transistor-based technology. Memristor stores information in the form of a programmable state, represented as a physical material property Resistance. The state of a memristor is analog and it can be used to store the network policies in packet processors. The memristor can compute in-memory by providing an output which is a function of the applied read input and the programmed initial state. The initial state is programmed by applying a set of write signals i.e., continuous high/low voltage pulses for long (write) durations. The read operation applies voltage pulses of shorter (read) durations in order to fetch the stored state and it cannot alter the initial state. Unlike the traditional components, memristor is the only two-terminal component which can provide different state resistances against the same analog read input, as shown in Fig. 2. In the given case, same applied input can yield either $S^1_{h_1}$ or $S^1_{l_1}$ depending upon the programmed initial state of S^1_1 or S^1_m. Moreover, reprogramming the initial state to S^n_1 or S^n_m can generate a new input-state response for a memristor as shown in Fig. 2 (Computation-n).

C. Problem Statement

Integrating the properties of components like memristors is a quite challenging task as it requires the translation of features from the analog to the digital domain. In this paper, we aim at

Saad Saleh, Anouk S. Goossens, Sunny Shu, Tamalika Banerjee and Boris Koldehofe. **Analog In-Network Computing through Memristor-based Match-Compute Processing**. In Proceedings of the 43rd International Conference on Computer Communications (INFOCOM 2024), IEEE, 10 pages, 2024.



(a) TCAM-based match-action processing.

(b) pCAM-based match-compute processing.

Fig. 3: Abstract working operations of the match-action vs the proposed match-compute packet processing.

an understanding of designing the appropriate packet processing abstractions for enhancing the expressiveness of network functions and supporting energy efficient network functions. In particular, we propose a novel Match-Compute abstraction to utilize the colocalization of memory and processing.

To understand the implications for network abstractions, consider the example of an AQM function. An AQM function needs to calculate the PDP based upon the sojourn time, buffer size, and rate of change of these parameters in a single stage. The match-action processing like TCAM (Fig. 3a) only supports matching of incoming query q with the stored policies (using $f()$) in N stages, as shown in Eq. 1. The output of $f()$ is high only if the stored policy is equal to the incoming query or don't care bit X (Eq. 2). The action corresponds to the match with the highest priority p_i (Eq. 3). In this framework, it is not possible to express matching restrictions. Therefore, typical solutions either have limitations in precision and accuracy or time consumption for PDP computation. With the proposed match-compute abstraction, we aim to express the network functions requiring multiple network policies like sojourn time and buffer size for AQM in a single stage, as shown below. It enables us to specify multiple features and calculate the network function with high precision, accuracy, and less energy consumption at line rate in the data plane.

Unsolvable Network Functions:

if (a<$f_1(q)$<b) & (c<$f_2(q)$<d) $\rightarrow$ Action(1)

Examples

AQM:

$f_1(q)$ = Sojourn Time; $f_2(q)$ = Buffer size;

Firewall:

$f_1(q)$ = Packet Rate; $f_2(q)$ = Source IP;

Load Balancing:

$f_1(q)$ = Bandwidth; $f_2(q)$ = Latency;

Generalized Unsolvable Function:

$\sum_{j=1}^{N}(Priority(e,j)\times$(a<$f_j(q)$<b)) $\rightarrow$ Action(e)

Constraint: Policies $f()$ programmed in parallel TCAMs.

III. Proposed Match-Compute Processing

In this section, we present the proposed match-compute processing and its formalizations for various network functions.

$$O_i = \prod_{j=1}^{N} f_{i,j}(q); \quad \forall i \in M \tag{1}$$

$$f_{i,j}(q) = \begin{cases} 1 & if \quad Stored\_policy \;=\; Input|X \\ 0 & if \quad Stored\_policy \;\neq\; Input|X \end{cases} \tag{2}$$

$$Action = \max\{p_i * O_i : i \in \mathrm{M}, p_i \in \mathrm{Priority(i)}\} \tag{3}$$

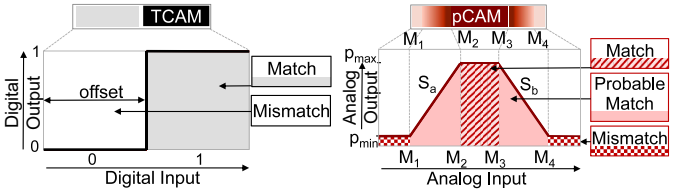
A. Match-Compute Processing Architecture

The proposed match-compute processing consists of applying the analog match process on the incoming query in order to identify the closeness of the incoming query with the stored network policy. The match process uses a programmable non-linear analog function $f(x)$ corresponding to the stored policy and it can be realized by the development of a novel abstraction called pCAM. In the next stage, the compute stage, the analog outputs of multiple parallel occurring matches are combined in order to evaluate the cumulative effect of matches. The compute stage applies a programmable multiplication and accumulation function $\psi(x)$ to the outputs of $f(x)$ in order to extract the network function output from multiple parallel occurring matches. $\psi(x)$ assigns the priorities to stored policies inside various cells. The function $\psi(x)$ is realized by a memristor-based crossbar array [16] [17]. Fig. 3b presents the proposed analog match-compute processing.

Building on [14], we propose a memristor-based pCAM abstraction for programming the non-linear function $f(x)$. The abstract working operation of pCAM as compared to the TCAM is shown in Fig. 4. pCAM has a range of deterministic matches providing discrete outputs, i.e., maximum or minimum outputs, based on the match of the incoming query with the stored policies. Moreover, pCAM also has a probabilistic range of matches providing analog outputs depending upon the difference of incoming query with the stored policies. By using both deterministic and probabilistic matches, pCAM can represent more expressive network functions and can compute precise analog matches. The function $\psi(x)$ is implemented by an $m \times n$ memristor-based crossbar array. It can be programmed to compute the weighted sum of inputs (pCAM outputs) for computing the most relevant match.

Fig. 5 shows the match-compute tables containing the

Saad Saleh, Anouk S. Goossens, Sunny Shu, Tamalika Banerjee and Boris Koldehofe. **Analog In-Network Computing through Memristor-based Match-Compute Processing**. In Proceedings of the 43rd International Conference on Computer Communications (INFOCOM 2024), IEEE, 10 pages, 2024.



(a) TCAM cell model. (b) pCAM cell model.

Fig. 4: The graphical output and programmable parameters of state-of-the-art TCAM vs the proposed pCAM abstraction.

Rules	Match	Priority	Action
Rule-1	0	7	No
Rule-2	1	6	Yes
Rule-3	0	10	No

(a) Match-Action table.

Rules	Match	Priority	Action
Rule-1	0.5	[1 0.1 0.1]	0.6
Rule-2	0.6	[0.6 0.2 0.1]	0.46
Rule-3	0.4	[0.1 0.7 0.2]	0.55

(b) Match-Compute table.

Fig. 5: Comparison of match-action vs match-compute tables.

analog stored policy and the priority matrix for multiple parallel occurring matches. The formal definitions of the programmable parameters in the match-compute processing are expressed in Func-1. The programmable parameters of the pCAM include the specification of the input range (M_1, M_2, M_3, M_4) for deterministic and probabilistic matches, and programmability of the output range (p_{max} and p_{min}), as shown in Fig. 4. Unlike the TCAM, pCAM provides the granular programmability of the hardware resources for minimizing the data movements. The computational process inside the match-compute unit is expressed in Algorithm-1. Based upon the analog input, the output ranges between a maximum of p_{max} and a minimum of p_{min} inside every pCAM cell. The output of multiple parallel pCAM cells is multiplied by their respective weights w and the sum of weighted outputs is used to take the actions.

M_1 : Start of Probabilistic Match
M_2 : Start of Deterministic Match
M_3 : End of Deterministic Match
M_4 : Start of Probabilistic Match
p_{max} : Programmed match probability
p_{min} : Programmed mismatch probability
w_{ij} : Programmable weights $\forall i \in M, j \in N$

Function 1: Parameters of Match-Compute Processing.

B. Match-Compute Formalizations for Network Functions

In this section, we present the formalizations of match-compute processing for network functions based on the precision requirements (flexibility) and mutual information (overlap) among parallel matches. The output of the pCAM-based match-compute units is represented in Eq. 4. $f()$, w and q represent the pCAM cell functions, associated weights, and the incoming query, respectively. N and M refer to the number of parallel pCAM cells and actions (outputs), respectively.

$$Action_i = \sum_{j=1}^{N} (w_{ij} \times f_j(q)); \quad \forall i \in M \tag{4}$$

Algorithm 1 The output function of a match-compute unit.

function COMPUTEACTION(I)
  for $j \in N$ **do**
    switch j **do**
      case $I \in (M_1, M_2)$
        $f_j(I) \leftarrow \frac{p_{max}(I-M_1)-p_{min}(I-M_2)}{M_2-M_1}$
      case $I \in (M_3, M_4)$
        $f_j(I) \leftarrow \frac{p_{max}(I-M_4)-p_{min}(I-M_3)}{M_3-M_4}$
      case $I \in [M_2, M_3]$
        $f_j(I) \leftarrow p_{\text{max}}$
      case $I \in \neg[M_1, M_4]$
        $f_j(I) \leftarrow p_{\text{min}}$
  end for
  for $i \in M$ **do**
    $O_i \leftarrow \sum_{j=1}^{N} w_{ij} f_j(I)$
  end for
end function

1) Non-flexible Non-Overlapping Network Functions: The network functions related to hard network policies represent the class of non-flexible non-overlapping functions. These functions require high precision with a definite binary output independent of the neighboring match operations. For example, IP lookup requires discrete output in every match independent of the matches in parallel cells. The formal definitions of the match-compute processing for these functions are expressed in Func-2. In the updated function, probabilistic regions of pCAM cells are eliminated by making M_2 and M_4 equal to M_1 and M_3, respectively. Moreover, the maximum and minimum outputs of pCAM are set to be 1 and 0 respectively because there is no intake from neighboring parallel matching pCAM cells. The weights for overlapping matches are also set to zero. The output of these functions depends only on the respective pCAM cells without any input from neighboring cells, as shown in Eq. 5. The function $f()$ reduces to a $MaxMin()$ function which produces only high or low outputs based on the match operation.

$M_1 \leftarrow M_2$: Eliminating the probabilistic regions
$M_3 \leftarrow M_4$: Eliminating the probabilistic regions
$p_{\text{max}} \leftarrow 1$: Maximizing the match probability
$p_{\text{min}} \leftarrow 0$: Minimizing the mismatch probability
$w_{ij} \leftarrow 0$: Minimizing the weights $\forall i \neq j$

Function 2: Non-flexible non-overlapping network functions.

$$Action_i = w_{ii} \times MaxMin_i(q); \quad \forall i \in M \tag{5}$$

2) Non-Flexible Overlapping Network Functions: It represents the class of network functions with high precision requirements (non-flexibility), but overlap with the neighboring match-compute cells. For example, the firewall function takes into account multiple parallel matches in order to express multiple policies and requires a discrete output. For these network functions, the weight factor is critical and it is programmed based upon the priority of various pCAM cells in order to perform the computation operation. The

Saad Saleh, Anouk S. Goossens, Sunny Shu, Tamalika Banerjee and Boris Koldehofe. **Analog In-Network Computing through Memristor-based Match-Compute Processing**. In Proceedings of the 43rd International Conference on Computer Communications (INFOCOM 2024), IEEE, 10 pages, 2024.

probabilistic region of pCAM is eliminated by programming M_2 and M_4 to M_1 and M_3, respectively (Func-3). p_{max} and p_{min} are also programmed to upper-threshold and lower-threshold, respectively. The output of these functions depends upon all the weights of neighboring policies, as shown in Eq. 6. Moreover, the pCAM function $f()$ also reduces to a $MaxMin()$ function for a discrete output.

```
M1 ← M2 : Eliminating the probabilistic regions
M3 ← M4 : Eliminating the probabilistic regions
pmax ← Upper-threshold : Maximum match probability
pmin ← Lower-threshold : Minimum mismatch probability
```

Function 3: Non-flexible overlapping network functions.

$$Action_i = \sum_{j=1}^{N} (w_{ij} \times MaxMin_j(q)); \quad \forall i \in M \tag{6}$$

3) Flexible Non-Overlapping Network Functions: The network functions, like packet scheduling, build heavily on flexible outputs based upon the inputs but lack overlap with the neighboring match-compute cells. For example, the congestion in one queue is entirely unrelated to the other queue due to separate flow categories in different queues. For these network functions, the weight for overlapping input is programmed to be zero (Func-4). The rest of the parameters are programmed by the network function based on the requirements of different processing stages. The output of the flexible non-overlapping network functions is represented by Eq. 7. The function $f()$ defined inside pCAM cells performs computations for these network functions.

```
wij ← 0: Programmable weights  ∀i ≠ j
```

Function 4: Flexible non-overlapping network functions.

$$Action_i = w_{ii} \times f_i(q); \quad \forall i \in M \tag{7}$$

4) Mimicking the Digital Match-Action Processing: In order to implement the traditional digital algorithms, the pCAM-based match-compute abstraction can also mimic the match-action processing. In this case, the probabilistic regions of the pCAM cells are eliminated (Func-5). Moreover, the region for maximum and minimum outputs is split equally in the entire input range. The maximum and minimum outputs are programmed to 1 and 0, respectively. All weights for neighboring pCAM cells are set to zero. The output for the binary network functions is represented in Eq. 8. Similar to digital match-action processing, there is an associated priority for every rule and zero overlap between neighboring matches.

```
M1 ← M2          : Eliminating the probabilistic regions
M3 ← M4          : Eliminating the probabilistic regions
M3 − M1 ← 1/2    : Splitting the cell function
pmax ← 1         : Maximizing the match probability
pmin ← 0         : Minimizing the mismatch probability
wij ← 0          : Programmable weights  ∀i ≠ j
wii ← Priority   : Programmable rule priorities  ∀i ∈ M
```

Function 5: Digital Match-Action Processing.

$$Action_i = w_{ii} \times MaxMin_i(q); \quad \forall i \in M \tag{8}$$

IV. Programming Match-Compute Processing

In this section, we present the proposed packet processing architecture and the programming abstractions of match-compute processing for a baseline network function i.e., AQM.

A. Proposed Packet Processing Architecture

The proposed packet processing architecture for the pCAM-based match-compute processing is shown in Fig. 6. The major modules and their functions are described below.

Ingress and egress match-compute pipeline. The match-compute processing stages are used in the ingress and egress packet processing pipeline in order to support energy efficient network functions. The match-compute units are programmable by the controller based on the network function requirements. Each match-compute stage has parallel pCAM cells containing the network policies and the output is fed to the crossbar array which contains the programmed priorities for various network policies. All stages are connected with the corresponding actions in order to execute the decisions for network functions. The processing pipeline can use multiple serial stages for the processing steps. The network functions requiring complex learning models, like Spiking Neural Networks (SNN), use both ingress and egress pipelines for incorporating complex decisions for network functions.

Analog Traffic Manager. The match-compute processing units are present inside the traffic manager to deploy AQM algorithms because the current packet processors lack the programmability inside traffic manager [6]. Using the match-compute based AQM, packet processors can adapt the PDP based upon the variation in the sojourn time and buffer size.

Parser. The role of the parser is to extract the required features from incoming traffic i.e., packet header and payload fields, and feed them to the match-compute units based upon the programmability by the controller. Since the data plane supports computations, all kinds of classification features, including traffic statistics and characteristics, are used for the deployment of learning models in the data plane.

Ingress and egress packet queues. Considering the variable processing rates of different network processing units, the packet queues are present on both the ingress and egress side of the packet processing architecture. These queues store the network packets and supply them to the match-compute units based on the processing rate of the network.

Controller. The controller plays the crucial role of programming the pCAM cells and the crossbar array inside the match-compute units for computing the network function. The controller consists of the parse graph and the control program for extracting the required packet information and programming the processing pipeline.

B. Proposed Programming Abstractions

The pCAM-based match-compute processing provides the programmability of the hardware resources along with the computation operations in the data plane. In this regard, the various programmable functions are described below.

Saad Saleh, Anouk S. Goossens, Sunny Shu, Tamalika Banerjee and Boris Koldehofe. **Analog In-Network Computing through Memristor-based Match-Compute Processing**. In Proceedings of the 43rd International Conference on Computer Communications (INFOCOM 2024), IEEE, 10 pages, 2024.

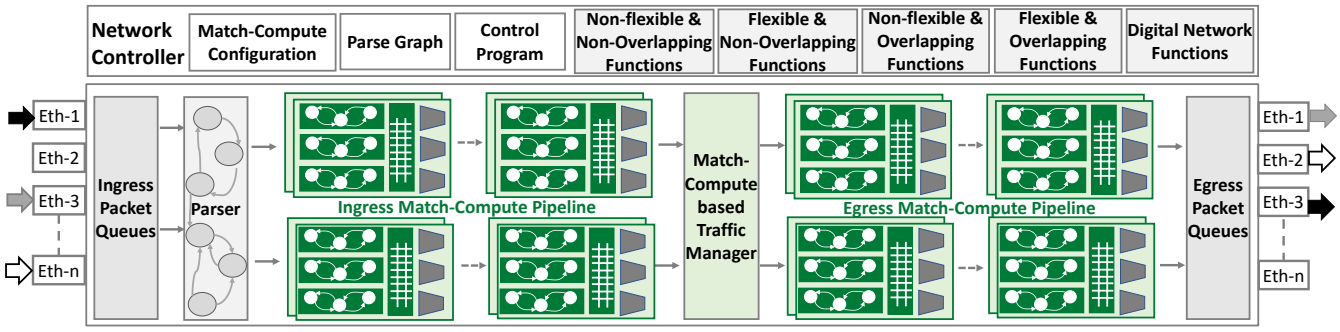


Fig. 6: The packet processing architecture for the memristor-based match-compute processing.

pCAM function. The pCAM-based match-compute units can be programmed by specifying the range of probabilistic and deterministic matches, and the maximum and minimum outputs, as shown in the function $prog\_pcam()$. Based upon the parameters, the controller applies the relevant signals (voltages) from the stored database to respective pCAM cells for programming the network policies inside the switch.

Crossbar-based computation function. The priorities of various policies are programmed by the weight function where the large weights correspond to high priorities and vice versa, as shown in the function $prog\_weights()$. At the output of the crossbar, the cumulative effect of all policies in parallel pCAM cells is obtained, as shown in the function $compute()$.

pCAM input-output function. The input-output function of every pCAM cell is expressed in the function $pcam()$. This function can be used by the controller to compute the parameters of individual pCAM cells based on the input-output response provided by the programmer.

Processing pipeline. The architecture uses a programmable processing pipeline based on the specified feature set. In order to update it, the occurrence of features is changed in the feature set, as shown in the function $pipeline()$.

Match-compute tables. The match-compute tables have three parameters i.e., read, output, and action. Since the computation operation is analog, the raw output of computation can also be used for network function outputs like PDP for AQM. On the input, the match-compute tables read the feature set from the packet processors. On the output, the raw analog outputs and the corresponding actions are generated, as shown in the function $MatchCompute()$.

pCAM update function. As part of the action, the individual pCAM modules and the weight function can be updated by the function $update\_pcam()$. This function reshapes the operation of the network function based on the requirements.

```
function prog_pcam(){
    program(M1, M2, M3, M4, pmax, pmin);
    //Applying voltages to the pCAM
}
```

```
function prog_weights(input,i,j){
    for(i ϵ M, j ϵ N)
        wij = input;          //Program the desired probability
    end
}
```

```
function compute(feature-i){
    analogoutput_i = Σ_{j=1}^{n} wij pcam(feature-i);
    //Output of match-compute units
}
```

```
function pcam(input,output){
    if (input ≤ M1 || input ≥ M4)
        output=pmin;
    elseif M1 < input < M2
        output=(pmax(input − M1) − pmin(input − M2))/(M2 − M1)
    elseif M3 < input < M4
        output=(pmax(input − M4) − pmin(input − M3))/(M3 − M4)
    else
        output=pmax; //Input-Output response by controller
}
```

```
function pipeline(){                  // Processing pipeline
    output = pipeline{
            compute(feature-1),       // Stage-1
            ...
            compute(feature-n)}       // Stage-n
}
```

```
table MatchCompute{
    reads{
        feature_set();}    //Reading the features for pCAM
    output{
        pipeline();}       //Computing the analog output
    actions{
        action_pCAM();     //Action set for pCAM
        update_pCAM();}    //Programmability of pCAM
}
```

```
action update_pcam(id, parameter[1:8], w){
    set_field(prog_pCAM.feature-1, M[1:8], w);
    ...
    set_field(prog_pCAM.feature-n, M[1:8], w);
}
```

C. Proof-of-Concept: Match-Compute based AQM

In this section, we present the application of match-compute processing for an AQM-based network function.

Proposed AQM technique. We propose a match-compute based AQM which utilizes the higher-order derivatives of sojourn time and buffer size to calculate the PDP. The estimation of higher-order derivatives takes huge energy and system resources in traditional digital architectures. However, the analog components, like Comparators, Op-Amps, can compute the derivatives with less energy consumption and resource utilization [18]. Based upon the first-order derivative, the proposed AQM can estimate the rate of increase of packet delay

Saad Saleh, Anouk S. Goossens, Sunny Shu, Tamalika Banerjee and Boris Koldehofe. **Analog In-Network Computing through Memristor-based Match-Compute Processing**. In Proceedings of the 43rd International Conference on Computer Communications (INFOCOM 2024), IEEE, 10 pages, 2024.

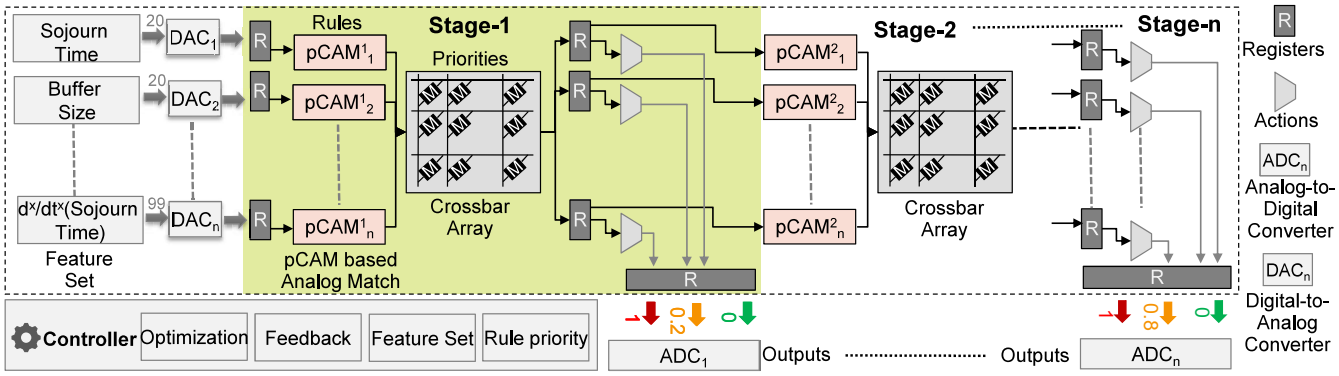


Fig. 7: The inherent semantics of the traffic manager built over match-compute processing pipeline.

and buffer size. It helps in the timely mitigation of congestion by dropping the packets. The second-order derivative provides an additional insight into the rate of change of the first-order derivative and it helps in estimating the sharp arrival/departure rate of packets for packet drop estimation. Lastly, the third-order derivative provides information about the bursty periods of the network traffic. Similarly, the higher-order derivatives of buffer size provide information about the congestion which is used in the computation of PDP.

Architecture for AQM. The architecture of the match-compute based AQM consists of the pCAM cells for estimation of the analog match and crossbar arrays to assign weights to the outputs of various pCAM cells, as shown in Fig. 7. The controller has the crucial role of collecting features from the data set by using analog components and programming the match ranges and priorities in the pCAM and the crossbar array. Moreover, the digital incoming traffic features can be converted to the analog domain through Digital-to-Analog Converters (DACs). After the computation, the processing pipeline generates the raw analog output which is the PDP. It can be used directly or converted to digital signal through Analog-to-Digital Converters (ADCs).

Programming Parameters. Building on the programming abstractions in Sec-IV-B, the match-compute based AQM consists of the features as mentioned in Func-6. The feature sets include 8 programmable parameters i.e., sojourn time, buffer size, and the consecutive three higher-order derivatives of these parameters. The feature priorities are set by the programmable weights in order to compute the PDP based upon the cumulative effect of various features.

```
function feature_set(){
    sojourn_time;          \\Packet delay in queue
    d/dt(sojourn_time);    \\1st deriv. of Packet delay
    d^2/dt^2(sojourn_time);\\2nd deriv. of Packet delay
    d^3/dt^3(sojourn_time);\\3rd deriv. of Packet delay
    buffer_size;           \\Buffer Capacity
    d/dt(buffer_size);     \\1st deriv. of Buffer capacity
    d^2/dt^2(buffer_size); \\2nd deriv. of Buffer capacity
    d^3/dt^3(buffer_size); \\3rd deriv. of Buffer capacity
}
```

Function 6: Feature set for AQM.

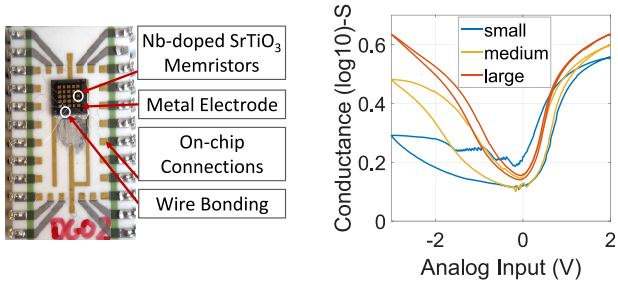
V. Performance Analysis

In this section, we analyze the energy consumption and performance metrics of the match-compute based AQM in comparison to the prior state-of-the-art AQM algorithms.

A. Energy Consumption

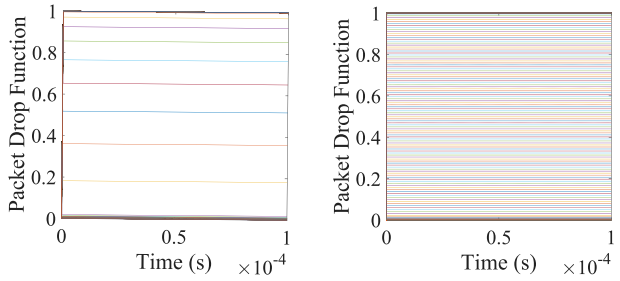
In order to validate the support of match-compute processing for network functions, like AQM, a physical memristor chip was fabricated using the Nb-doped $SrTiO_3$. This chip contains 25 memristors and all memristors share a common binding topology. All memristors can be fabricated in three different configurations, referred as small, medium, and large devices [19]. We developed the match-compute models in Matlab comprising of pCAM with crossbar array connections. The match-compute models use the experimental dataset of the memristor chip for analyzing the energy efficiency of the analog packet processing. Fig. 8 shows the memristor chip and the states of the memristors. Since resistance is very large at small voltages, conductance (inverse of resistance) has been plotted from the experimental dataset in Fig. 8.

The energy consumption of the match-compute based AQM is shown in Fig. 9. The results show that the memristors can provide a range of states for mapping network policies and the energy consumption can go up to 0.16 nJ/bit/cell. However, match-compute based AQM also offers a range of states with very low energy consumption. The minimum energy consumption for match-compute processing is only 0.01 fJ/bit/cell. In the results, the *Input* refers to the search query extracted from either the raw packet header fields or the feature extraction modules. In case of AQM, the input range varies corresponding to the features, like sojourn times and higher-order derivatives, mapped to the range of -2 to 4. The analysis of programmable PDP is shown in Fig. 10. The results show the distinct programmable PDP by configuring the memristor in unique states for input range $[-2,1]$ and $[1,4]$. The less number of PDPs in the range $[-2,1]$ refer to the limitations of the underlying hardware. However, it's possible to map all the input fields to the range of $[1, 4]$ for high precision AQM flows e.g., TCP flows, and use the range $[-2,1]$ for low precision AQM network flows e.g., UDP flows. It is pertinent to mention that energy consumption



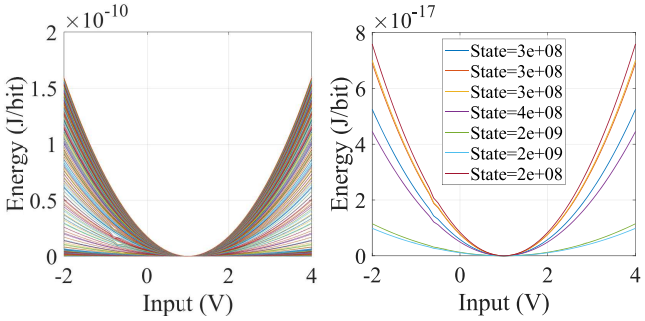
(a) Memristor chip. (b) Experimental readings.

Fig. 8: Multiple memristive states against the analog inputs for the physically fabricated memristor chip.



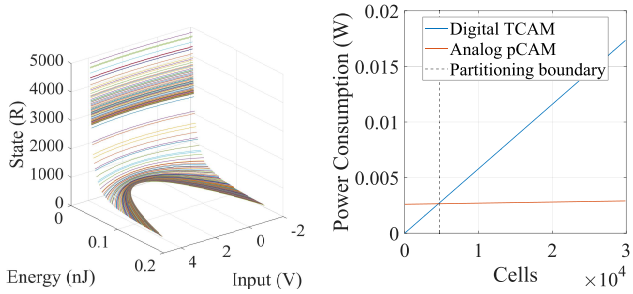
(a) Input ∈ [-2,1]. (b) Input ∈ [1,4].

Fig. 10: The range of analog PDPs for various input header ranges (queuing delays).



(a) Complete range. (b) Selected range.

Fig. 9: Energy consumption of match-compute based AQM.



(a) Energy consumption per bit. (b) Digital vs Analog.

Fig. 11: The trends of energy consumption.

depends both on the stored state and the input query, and the complete trends can only be represented in three dimensions (Fig. 11a). The analog computation has an associated one-time cost of converting packet fields from the digital to the analog domain, but it's much more scalable with an increasing number of pCAM cells, as shown in Fig. 11b. Since AQM can collect analog fields using analog components, this cost is not applicable for the AQM. The comparison of energy consumption for match-compute based AQM with state-of-the-art techniques showed at least 50 times improvement in performance (Tab. I).

B. Quality of Service

In order to analyze the performance of the match-compute based AQM, we developed the pCAM-based AQM models in the ns-3 simulator [28]. The pCAM model was programmed to drop the network packets based on traffic statistics like sojourn time, buffer size, and higher-order derivatives of these features. We used a fair-queuing approach in which different traffic classes are assigned to different queues and the PDP of each class is independent of the neighboring class. The performance of pCAM-based AQM was compared with the state-of-the-art AQM algorithms including RED [29], PIE [30], CoDel [31], FQ-CoDel [32] and COBALT [33]. pCAM was programmed with three different configurations in order to show the diversity and programmability. The results for video streaming applications by using 100 clients, 5 servers, and Poisson distributed flows (5 Mbps/client) are shown in Fig. 12. The results show that pCAM provides comparable throughput to the traditional algorithms. However, pCAM can be configured for lowest or highest sojourn time based upon the constraints of Packet Loss Ratio (PLR) and Queue length. A lower sojourn time in pCAM results in lower queue length but higher PLR and vice versa. Overall, pCAM provides the most configurable options for providing different QoS to various network traffic flows based on the requirements.

The performance of pCAM was also analyzed by increasing the network traffic load and measuring the impact on performance metrics, as shown in Fig. 13. The results show that pCAM provides throughput similar to the traditional algorithms. The PLR for pCAM is much lower than the traditional algorithms except for FQ-CoDel which has a much higher queue length. The algorithms, like PIE, CoDel, provided less sojourn time but suffered from excessive packet losses. On the contrary, pCAM managed an optimum queue length and minimized the packet losses. The comparison shows that pCAM provides satisfactory QoS by increasing the traffic load.

TABLE I: The performance comparison of Transistors(**T**)/ Memristors(**M**)-based Digital(**D**)/Analog(**A**) computations.

Researches	[20]	[21]	[22]	[23]	[24]	[25]	[26]	[27]	**pCAM**
Computation (D/A)	D	D	D	D	D	D	D	D	**A**
Technology (T/M)	T	T	M	M	M	M	M	M	**M**
Latency (ns)	1	1.9	1	0.29	0.18		2.3	8	**1**
Energy (fJ/bit)	0.58	1.98	1-16	1.04	1.2	2.15	3	7.4	**0.01**

Saad Saleh, Anouk S. Goossens, Sunny Shu, Tamalika Banerjee and Boris Koldehofe. **Analog In-Network Computing through Memristor-based Match-Compute Processing**. In Proceedings of the 43rd International Conference on Computer Communications (INFOCOM 2024), IEEE, 10 pages, 2024.

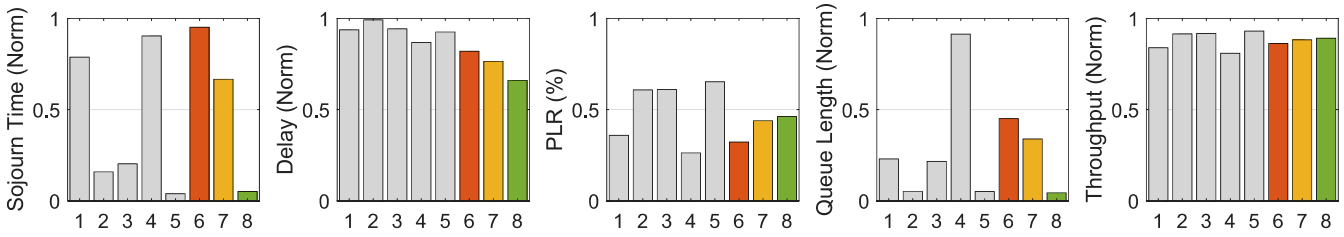


Fig. 12: Video streaming analysis of pCAM $(6, 7, 8)$ vs RED (1), PIE (2), CoDel (3), FQ-CoDel (4) and COBALT (5).

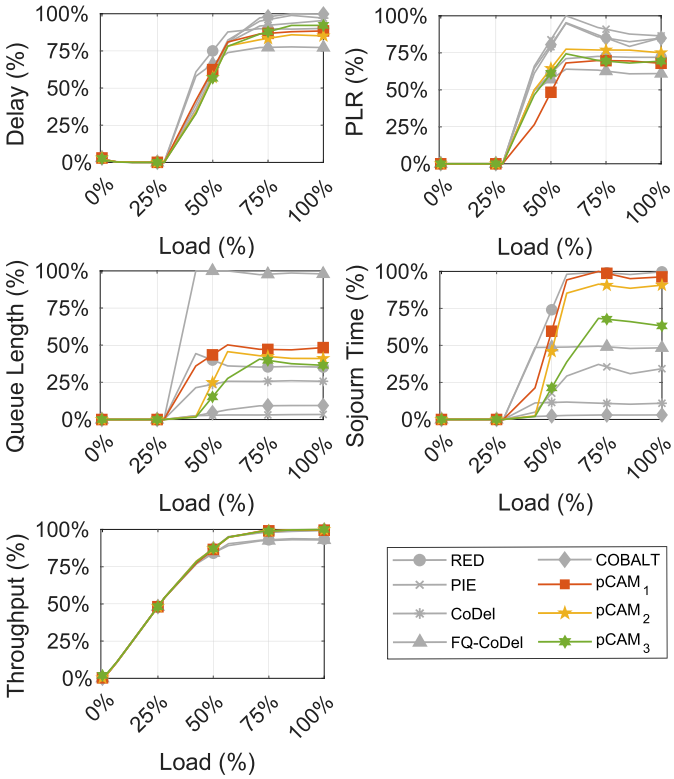


Fig. 13: Performance of pCAM with prior state-of-the-art AQM algorithms by increasing the network traffic load.

VI. Literature Review

The requirement of more expressive network functions like cognitive functions is a fundamental requirement of packet processors [34]. Saleh et al. [8] identified the applications of cognitive models for enabling energy efficient and self-learning network functions. Since the current data plane cannot take actions based upon multiple parallel matching policies, Shrivastav [35] [36] proposed multi-dimensional match-action tables using FPGAs and showed line rate performance for network functions requiring multiple policies. In [15], Zulfiqar et al. highlighted the latency issues by continuous data movements between the switch's operating systems and the control plane. The authors suggested future research on incorporating line rate computational capabilities in the data plane.

The use of memristors has been actively studied for supporting energy efficient high-performance operations [37] [38]. Saleh et al. developed memristor-based TCAMs for supporting energy efficient network functions, like IP lookup, in the data plane [22] [39]. Due to non-volatility, the proposed match-action processing did not consume any energy in the standby mode and required only 1-16 μW of energy during the match operations. In [40] [41], Graves et al. developed memristor-based TCAMs for supporting network functions like Regular Expression Matching. The research showed that memristor-based TCAMs can increase the throughput by 12 times (upto 47.2 Gbps) as compared to the FPGAs. However, these researches did not exploit the analog nature of the memristors. Recent researches [42]–[47] focused on the development of analog Content Addressable Memory (CAM) for storing policies in the analog domain. The output is either digital or analog based on the application. The results showed that memristor-based analog CAM can provide space and energy savings by 18 and 10 times, respectively. However, unlike pCAM, these analog CAM designs do not support the programmability of a nonlinear function at the cell level. A comparison of all studies shows that memristor-based CAMs have not been used for deterministic and probabilistic computations in the data plane.

VII. Conclusion and Future Work

In this paper, we showed the support of line rate computations for energy efficient network functions by leveraging the novel technology of memristors. We proposed the match-compute processing building on pCAM abstraction which supports both deterministic and probabilistic matches in the data plane. pCAM provides line rate matrix multiplications through crossbar array interconnections. We developed the packet processing architecture, programming abstractions, and analyzed the performance for an AQM function. The proposed AQM function drops packets based upon the analog higher-order derivatives of sojourn times and buffer sizes. The analysis over the dataset of a physically fabricated memristor chip showed only 0.01 fJ/bit/cell of energy consumption which is 50 times less than the traditional processing. In the future, we would focus on supporting cognitive models, like SNN, at the packet processors. Moreover, we would study the self-learning network functions to develop autonomous packet processors.

Acknowledgment

The authors would like to acknowledge the financial support of the CogniGron research center and the Ubbo Emmius Funds (University of Groningen).

Saad Saleh, Anouk S. Goossens, Sunny Shu, Tamalika Banerjee and Boris Koldehofe. **Analog In-Network Computing through Memristor-based Match-Compute Processing**. In Proceedings of the 43rd International Conference on Computer Communications (INFOCOM 2024), IEEE, 10 pages, 2024.

REFERENCES

[1] X. Chen *et al.*, "Fine-Grained Queue Measurement in the Data Plane," in *Proceedings of the International Conference on Emerging Networking Experiments And Technologies*. ACM, 2019, pp. 15–29.

[2] R. MacDavid, X. Chen, and J. Rexford, "Scalable Real-Time Bandwidth Fairness in Switches," in *Proceedings of the International Conference on Computer Communications*. IEEE, 2023, pp. 1–10.

[3] Z. Yu *et al.*, "Programmable Packet Scheduling with a Single Queue," in *Proceedings of the SIGCOMM Conference*. ACM, 2021, pp. 179–193.

[4] S. Saleh *et al.*, "Breaching IM Session Privacy Using Causality," in *Proceedings of the Global Communications Conference*. IEEE, 2014, pp. 686–691.

[5] S. Saleh, M. U. Ilyas, K. Khurshid, A. X. Liu, and H. Radha, "IM Session Identification by Outlier Detection in Cross-correlation Functions," in *Proceedings of the Annual Conference on Information Sciences and Systems*. IEEE, 2015, pp. 1–5.

[6] Intel®. (2023) Tofino 2 12.8 Tbps, Reduced Stage, 4 Pipelines. Intel. [Online]. Available: https://www.intel.com.au/content/www/au/en/products/details/network-io/intelligent-fabric-processors.html

[7] NVIDIA. (2023) Spectrum-4. 51.2 Tb/s Ethernet Switch ASIC. NVIDIA. [Online]. Available: https://www.nvidia.com/en-us/networking/ethernet-switching/

[8] S. Saleh and B. Koldehofe, "On Memristors for Enabling Energy Efficient and Enhanced Cognitive Network Functions," *IEEE Access*, vol. 10, pp. 129 279–129 312, 2022.

[9] C. D. Schuman, S. R. Kulkarni, M. Parsa, J. P. Mitchell, P. Date, and B. Kay, "Opportunities for Neuromorphic Computing Algorithms and Applications," *Nature Computational Science*, vol. 2, pp. 10–19, 2022.

[10] A. K. Ramanathan *et al.*, "Look-Up Table based Energy Efficient Processing in Cache Support for Neural Network Acceleration," in *Annual International Symposium on Microarchitecture*. IEEE/ACM, 2020, pp. 88–101.

[11] O. Vaughan, "A History of Memristors in Five Covers," *Nature Electronics*, vol. 6, no. 1, pp. 7–7, 2023.

[12] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The Missing Memristor Found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[13] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory Devices and Applications for In-Memory Computing," *Nature Nanotechnology*, vol. 15, no. 7, pp. 529–544, 2020.

[14] S. Saleh and B. Koldehofe, "The Future is Analog: Energy-Efficient Cognitive Network Functions over Memristor-Based Analog Computations," in *Proceedings of the Workshop on Hot Topics in Networks*. ACM, 2023, p. 254–262.

[15] A. Zulfiqar, B. Pfaff, W. Tu, G. Antichi, and M. Shahbaz, "The Slow Path Needs an Accelerator Too!." *ACM SIGCOMM Computer Communication Review*, vol. 53, no. 1, pp. 38–47, 2023.

[16] C. Wang *et al.*, "Parallel In-Memory Wireless Computing," *Nature Electronics*, pp. 1–9, 2023.

[17] S.-i. Yi, J. D. Kendall, R. S. Williams, and S. Kumar, "Activity-Difference Training of Deep Neural Networks using Memristor Crossbars," *Nature Electronics*, vol. 6, no. 1, pp. 45–51, 2023.

[18] M. A. Zidan *et al.*, "A General Memristor-based Partial Differential Equation Solver," *Nature Electronics*, vol. 1, no. 7, pp. 411–420, 2018.

[19] A. S. Goossens, M. Ahmadi, D. Gupta, I. Bhaduri, B. J. Kooi, and T. Banerjee, "Memristive Memory Enhancement by Device Miniaturization for Neuromorphic Computing," *Advanced Electronic Materials*, vol. 9, no. 4, p. 2201111, 2023.

[20] I. Arsovski, T. Hebig, D. Dobson, and R. Wistort, "A 32 nm 0.58-fJ/bit/Search 1-GHz Ternary Content Addressable Memory Compiler Using Silicon-Aware Early-Predict Late-Correct Sensing with Embedded Deep-Trench Capacitor Noise Mitigation," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 4, pp. 932–939, 2013.

[21] I. Hayashi *et al.*, "A 250-MHz 18-Mb Full Ternary CAM With Low-Voltage Matchline Sensing Scheme in 65-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 11, pp. 2671–2680, 2013.

[22] S. Saleh, A. S. Goossens, T. Banerjee, and B. Koldehofe, "TCAmM$^{\text{CogniGron}}$: Energy Efficient Memristor-Based TCAM for Match-Action Processing," in *Proceedings of the International Conference on Rebooting Computing*. IEEE, 2022, pp. 89–99.

[23] S. Matsunaga *et al.*, "Fully Parallel 6T-2MTJ Nonvolatile TCAM with Single-Transistor-Based Self Match-line Discharge Control," in *Symposium on VLSI Circuits-Digest of Technical Papers*. IEEE, 2011, pp. 298–299.

[24] K. Gnawali and S. Tragoudas, "High-Speed Memristive Ternary Content Addressable Memory," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 3, pp. 1349–1360, 2021.

[25] V. Bontupalli, C. Yakopcic, R. Hasan, and T. M. Taha, "Efficient Memristor-Based Architecture for Intrusion Detection and High-Speed Packet Classification," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 14, no. 4, pp. 1–27, 2018.

[26] L. Zheng, S. Shin, S. Lloyd, M. Gokhale, K. Kim, and S.-M. Kang, "RRAM-Based TCAMs for Pattern Search," in *International Symposium on Circuits and Systems*. IEEE, 2016, pp. 1382–1385.

[27] W. Xu, T. Zhang, and Y. Chen, "Design of Spin-Torque Transfer Magnetoresistive RAM and CAM/TCAM with High Sensing and Search Speed," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 18, no. 1, pp. 66–74, 2009.

[28] Ns3 network simulator. [Online]. Available: https://www.nsnam.org/

[29] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[30] R. Pan, P. Natarajan, F. Baker, and G. White, "Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem," RFC 8033, Feb. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8033

[31] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Controlled Delay Active Queue Management," RFC 8289, Jan. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8289

[32] T. Høiland-Jørgensen, P. McKenney, D. Taht, J. Gettys, and E. Dumazet, "The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm," RFC 8290, Jan. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8290

[33] J. Palmei *et al.*, "Design and Evaluation of COBALT Queue Discipline," in *International Symposium on Local and Metropolitan Area Networks*. IEEE, 2019, pp. 1–6.

[34] S. Saleh, S. Shu, and B. Koldehofe, "Adaptive In-Network Queue Management using Derivatives of Sojourn Time and Buffer Size," in *Proceedings of the Network Operations and Management Symposium*. IEEE, 2024, p. 5 pages.

[35] V. Shrivastav, "Programmable Multi-Dimensional Table Filters for Line Rate Network Functions," in *Proceedings of the SIGCOMM Conference*. ACM, 2022, p. 649–662.

[36] V. Shrivastav, "Stateful Multi-Pipelined Programmable Switches," in *Proceedings of the SIGCOMM Conference*. ACM, 2022, p. 663–676.

[37] S. Saleh and B. Koldehofe, "Memristor-based Network Switching Architecture for Energy Efficient Cognitive Computational Models," in *Proceedings of the International Symposium on Nanoscale Architectures*. ACM, 2023, p. 4 pages.

[38] S. Saleh, A. S. Goossens, T. Banerjee, and B. Koldehofe, "PAmM: Memristor-based Probabilistic Associative Memory for Neuromorphic Network Functions," in *Proceedings of the Non-Volatile Memory Technology Symposium*. IEEE, 2023, pp. 1–5, in Press.

[39] S. Saleh, A. S. Goossens, T. Banerjee, and B. Koldehofe, "Towards Energy Efficient Memristor-based TCAM for Match-Action Processing," in *Proceedings of the International Green and Sustainable Computing Conference*. IEEE, 2022, pp. 1–4.

[40] C. E. Graves *et al.*, "Memristor TCAMs Accelerate Regular Expression Matching for Network Intrusion Detection," *IEEE Transactions on Nanotechnology*, vol. 18, pp. 963–970, 2019.

[41] C. E. Graves *et al.*, "Regular Expression Matching with Memristor TCAMs," in *Proceedings of the International Conference on Rebooting Computing*. IEEE, 2018, pp. 1–11.

[42] C. Graves, C. Li, K. Ozonat, and J. P. Strachan, "Hardware Accelerator with Analog-Content Addressable Memory (a-CAM) for Decision Tree Computation," Apr. 21 2022, US Patent App. 17/071,924.

[43] C. Li, C. Graves, and J. P. Strachan, "Analog, Non-volatile, Content Addressable Memory," Nov. 24 2020, US Patent 10,847,238.

[44] C. Li, C. Graves, and J. P. Strachan, "Methods and Systems for an Analog CAM with Fuzzy Search," May 4 2021, US Patent 10,998,047.

[45] C. Li *et al.*, "Analog Content-Addressable Memories with Memristors," *Nature Communications*, vol. 11, no. 1, pp. 1–8, 2020.

[46] S.-C. Liu, J. P. Strachan, and A. Basu, "Prospects for Analog Circuits in Deep Networks," in *Analog Circuits for Machine Learning, Current/Voltage/Temperature Sensors, and High-speed Communication: Advances in Analog Circuit Design*. Springer, 2021, pp. 49–61.

[47] G. Pedretti *et al.*, "Differentiable Content Addressable Memory with Memristors," *Advanced Electronic Materials*, vol. 8, no. 8, p. 2101198, 2022.