Event-Driven Digital-Time-Domain Inference Architectures for Tsetlin Machines

Tian Lan†, Rishad Shafik†, Alex Yakovlev†
†Microsystems Research Group, Newcastle University, Newcastle upon Tyne, UK
{t.lan3, rishad.shafik, alex.yakovlev}@newcastle.ac.uk

***Abstract*—Machine learning fits model parameters to approximate input-output mappings, predicting unknown samples. However, these models often require extensive arithmetic computations during inference, increasing latency and power consumption. This paper proposes a digital-time-domain computing approach for Tsetlin machine (TM) inference process to address these challenges. This approach leverages a delay accumulation mechanism to mitigate the costly arithmetic sums of classes and employs a Winner-Takes-All scheme to replace conventional magnitude comparators. Specifically, a Hamming distance-driven time-domain scheme is implemented for multi-class TMs. Furthermore, differential delay paths, combined with a leading-ones-detector logarithmic delay compression digital-time-domain scheme, are utilised for the coalesced TMs, accommodating both binary-signed and exponential-scale delay accumulation issues. Compared to the functionally equivalent, post-implementation digital TM architecture baseline, the proposed architecture demonstrates orders-of-magnitude improvements in energy efficiency and throughput.**

***Index Terms*—Machine learning, Event-driven, Time-domain computing, Asynchronous pipeline, Tsetlin machine, Winner-Takes-All**

I. Introduction

Edge computing has emerged as a key enabler for real-time machine learning (ML) applications, driven by characteristics of low-latency responses, reduced bandwidth demands, and enhanced security [1] [2]. Applications like drone collision avoidance, autonomous driving, and augmented reality [3] rely critically on rapid inference and low communication overhead, making ML algorithms deployed at the edge particularly appealing. However, edge devices are often deployed far from gateways and frequently operate within limited energy budgets. These constraints impose stringent energy management on ML algorithms implemented at the edge.

In edge ML energy management, two pressing contradictions require resolutions. **First**, synchronous systems are controlled by a global clock whose frequency must exceed the incoming-event rate and cannot be adapted dynamically as that rate fluctuates, resulting in significant energy waste. As a resolution, event-driven asynchronous ML architectures [4] [5] enable processing only when events occur, eliminating clock power during idle intervals; this sparse computing consequently achieves lower average energy consumption and improved performance.

Second, ML algorithms are inherently intensive in arithmetic operations. Some examples include: (i) the evaluation of inner products using convolutional kernels in convolutional neural networks (CNNs); (ii) dense matrix multiply-accumulate (MAC) operations in fully connected layers of deep neural networks (DNNs); (iii) the accumulation of membrane potentials in spiking neural networks (SNNs); and (iv) various nonlinear transforms such as ReLU, softmax, argmax, and their variants that convert weighted sums into class-conditional probabilities or discrete decisions. Implementing these essential operations in the digital domain requires many arithmetic logic units (ALUs), MAC arrays, and magnitude comparators (MCs). Each logic occurs through discrete voltage transitions, with every transition charging or discharging the effective capacitance of CMOS nodes. This switching activity generates significant dynamic power, resulting in substantial energy dissipation per inference, especially at the edge [6]. As a weak-capacitance approach, transferring the extensive arithmetic operations to the time domain shows considerable promise. Recent studies [7] [8] have demonstrated that the MAC operations of NNs can be represented as pulse-delay accumulation. This method greatly reduces the charging and discharging of CMOS nodes, providing a practical route toward ultra-low-power solutions for edge devices.

Multi-class Tsetlin Machines (TMs) and variants such as the Coalesced TM (CoTM) represent a promising category of ML algorithms based on propositional logic [9] [10]. They offer several advantages for edge-oriented AI hardware: (i) **Hardware Affinity**: TM inference is based on Boolean operations, eliminating the need for convolutional layers and large-scale matrix multiplications. This allows for straightforward mapping of field-programmable gate array (FPGA) or application-specific integrated circuit (ASIC) architectures. (ii) **Simple Inference**: The inference processes in TMs are conceptually simple and computationally lightweight, making them well-suited for satisfying real-time and low-power requirements. (iii) **Full Interpretability**: Every prediction made by a TM can be traced back to explicit Boolean clause combinations. This ensures that decision pathways are transparent and understandable to humans, rather than relying on the opaque internal representations of DNNs. Collectively, these properties position TMs as an energy-efficient, fast, and explainable alternative to NNs for edge ML tasks.

This work was supported by EPSRC EP/X036006/1 Scalability Oriented Novel Network of Event Triggered Systems (SONNETS) project and by EPSRC EP/X039943/1 UKRI-RCN: Exploiting the dynamics of self-timed machine learning hardware (ESTEEM) project.

Deploying TM's inference process at the edge still needs to address the previously identified challenges of eliminating the global clock and alleviating intensive arithmetic operations. For multi-class TMs, it has been demonstrated that both asynchronous (four-phase handshake) dual-rail architectures [11] and time-domain strategies [12] are able to achieve energy efficiency. This paper presents design paradigms for multi-class TM and CoTM using asynchronous bundled data (BD) and hybrid digital-time-domain methods. Energy efficiency and throughput are quantitatively evaluated against functionally equivalent, post-implementation synchronous and asynchronous hardware, demonstrating the advantages of the proposed architecture.

Contributions:

- Novel hybrid digital-time-domain computing paradigms are proposed that integrate seamlessly with event-driven asynchronous logic.
- The proposed architecture realises an ultra-low-power, high-throughput inference process.
- Functionally equivalent synchronous and asynchronous digital TM inference circuits are implemented, providing a direct baseline for the proposed approach.

II. Methodology

Fig. 1 depicts the block diagram of the proposed architecture. At the top hierarchical level, an asynchronous controller synchronises the downstream functional modules by forwarding a *Request* signal from *Req_in* to *Req_out*. The *Acknowledge* path runs from *Ack_in* to *Ack_out*, providing reverse feedback to complete each transaction. The functional modules are designed as a hybrid data path. Specifically, literal generation and clause output are carried out in the digital domain. The class sum and argmax functions are converted to the time domain.

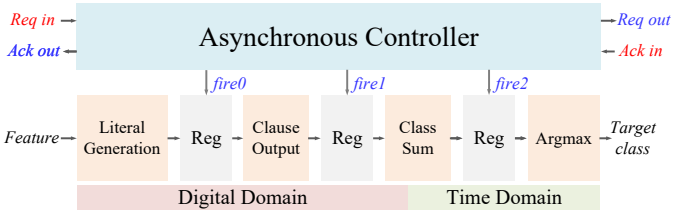


Fig. 1: Block diagram of the proposed architecture.

A. Asynchronous Controller

The asynchronous controller illustrated in Fig. 2 is constructed as a three-stage Click element. The two-phase handshake protocol achieves **Elastic throughput**: Computation proceeds only when data is available, eliminating idle switching and the global clock to handle worst-case path delays. **Gate-level implementation**: the circuit employs combinatorial logic and flip-flops, enabling direct synthesis with standard CMOS workflows and ensuring compatibility with electronic design automation (EDA) toolkits [4] [5] [13]. The pseudocode for a click element stage is listed in Algorithm 1.

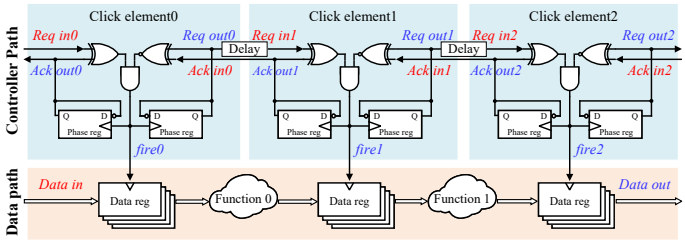


Fig. 2: Click element-based asynchronous pipeline.

Algorithm 1 Click Element-based Asynchronous Controller

function ClickElement
  Iutput rst, req_in, ack_in
  Output $req\_out,\ ack\_out,\ fire$
  Logic $phase\_in,\ phase\_out$
  if rst **then**
    $phase\_in \leftarrow 0\ ;\ phase\_out \leftarrow 0$
  else
    $fire \leftarrow (req\_in \oplus phase\_in) \wedge \neg(ack\_in \oplus phase\_out)$
    if $fire \uparrow$ **then**
      $phase\_in \leftarrow \neg phase\_in$
      $phase\_out \leftarrow \neg phase\_out$
    end if
  end if
  $req\_out \leftarrow phase\_in$
  $ack\_out \leftarrow phase\_out$
  return $req\_out, ack\_out, fire$
end function

B. Clause Evaluation

The literal generation and clause output stages are combined into the clause evaluation functional module. The descriptive hardware pseudocode is listed in Algorithm 2.

Algorithm 2 Clause Evaluation

function ClauseEvaluation
  Input $rst,\ fire0,\ feature \in \{0,1\}^{\text{num\_feature}}$
  Output $clause\_vector \in \{0,1\}^{\text{num\_clause}}$
  Logic $literal \in \{0,1\}^{2\times num\_feature}$
  if rst **then**
    $clause\_vector \leftarrow 0$
  else if $fire0 \uparrow$ **then**
    for $i \leftarrow 0$ to $num\_feature - 1$ **do**
      $literal[2i] \leftarrow feature[i]$
      $literal[2i+1] \leftarrow \neg feature[i]$
    end for
    for $j \leftarrow 0$ to $num\_clause - 1$ **do**
      $clause\_vector[j] \leftarrow \bigwedge(literal \vee TA\_states)$
    end for
  end if
  return $clause\_vector$
end function

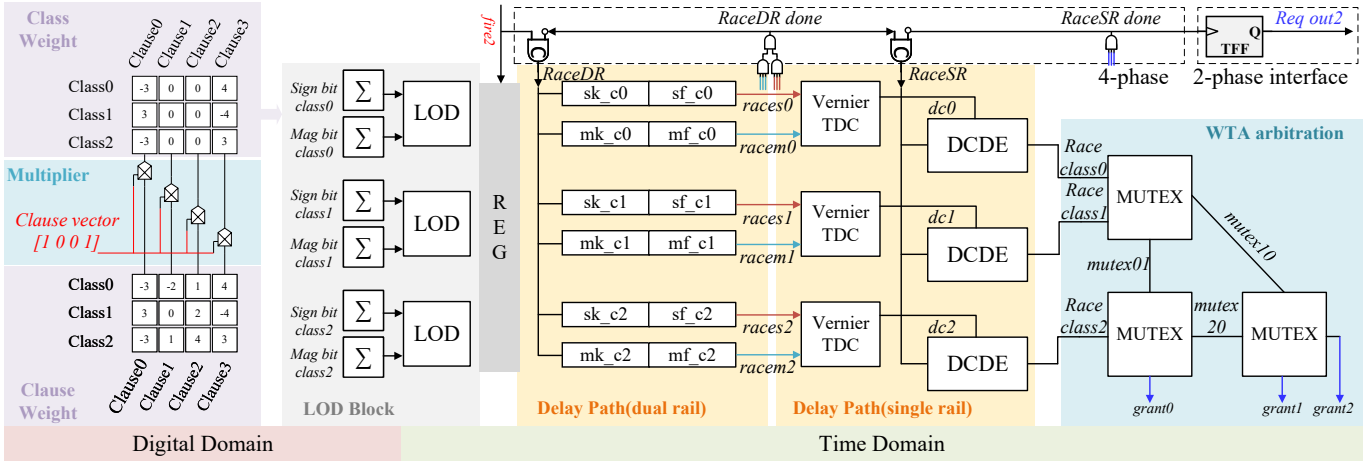


Fig. 3: The hybrid digital-time domain architecture for CoTM classification.

C. Classification

Classification is a significant functional module in the TM inference process, which aggregate the weighted contributions of active clauses to reach the final classification. Using hardware description languages (listed in Algorithm 3) to build this module is intuitive and convenient. However, conducting extensive arithmetic operations purely in the digital domain requires numerous ALUs, which leads to significant energy consumption [12]. Compared to the energy used for memory accesses, the energy overhead from arithmetic operations can be more effectively reduced through architectural optimisation. The classification function for multi-class TM inference can be defined as follows:

$$\hat{y} = \underset{i=1,\ldots,m}{\operatorname{argmax}} \left(\sum_{j=1}^{\frac{n}{2}} C_j^{1,i}(X) - \sum_{j=1}^{\frac{n}{2}} C_j^{0,i}(X) \right) \tag{1}$$

This function can be regarded as comparing Hamming distances among different classes, where contributions from ones in positive clauses and zeros in negative clauses are considered equivalent, as are contributions from zeros in positive clauses and ones in negative clauses. A detailed time-domain classification architecture has already been discussed in [12]. This work further develops a streamlined interface that seamlessly integrates the two-phase BD controller with the four-phase quasi-delay-insensitive (QDI) classification module, which will later be introduced in the CoTM part.

$$\hat{y} = \underset{i=1,\ldots,m}{\operatorname{argmax}} \left(\sum_{j=1}^{n} W_j^i \cdot \mathcal{C}_j^i(X) \right) \tag{2}$$

In the CoTM inference process, the classification function (Eq. 2) selects the target class with the highest binary MAC value. The CoTM does not pre-assign positive or negative clauses; instead, any clause can either support (with a positive integer weight) or oppose (with a negative integer weight) any class, greatly enhancing expressiveness. However, this flexibility introduces two significant challenges in designing time-domain hardware: (i) **Signed class sum addition:** The unidirectionality of time encoding makes it challenging to represent signed weight summation directly. (ii)**Exponential path delay growth:** As the number of clauses increases, the path delay for a class grows exponentially, which inflates area requirements and decreases robustness to process–voltage–temperature (PVT) variations.

Applied strategies: **(i) Encode class sum into differential delay path**: Digital logic pre-calculates all clause *sign* contributions into S and all *magnitude* contributions into M. Then, launches two independent pulses ($race_S$ and $race_M$). The interval difference encodes the final signed class sum. **(ii)Leading-ones-detector (LOD) delay compression**: Coarse delays (s^k, m^k) logarithmically segment the delay range, while a e-bit normalised fine delay (s^f, m^f) is used for high-

Algorithm 3 Classification

```
function CLASSIFICATION
    Input fire1, fire2, clause_vector ∈ {0,1}^num_clause
    Output target_class
    Logic sum_weight , class_sum ∈ ℤ^num_class
    if rst then
        sum_weight ← 0
    else if fire1 ↑ then
        sum_weight ← clause_vector ∧ clause_weight
    end if
    if fire2 ↑ then
        for i ← 0 to num_class −1 do
            class_sum[i] ← Sum(sum_weight[i])
        end for
        target_class ← Argmax(class_sum)
    end if
    return target_class
end function
```

resolution tuning [14].

The proposed CoTM classification implementation (illustrated in Fig. 3) leverages a digital-time-domain architecture. The main functional modules include: a binary multiplication matrix, an LOD block, a delay accumulation module, a Winner-Takes-All (WTA) arbitration system, a race control unit, and a four-to-two phase interface for asynchronous communication. **Specifically**:

1) ***Binary multiplication matrix****:* This module selects the weights of activated clauses based on the clause evaluation results. The chosen clause weights are then used to compute the subsequent class sum. This can be implemented as a multiplier (MUX) array.

2) ***LOD Coarse Fine Delay Extraction****:* As a front-end circuit to the delay accumulation module, the LOD block mitigates the exponential growth of delay paths caused by the increasing bit width of the class sum. The pseudocode hardware description is listed in Algorithm 4.

Algorithm 4 LOD coarse fine delay extraction

```
function LOD
    Input SumValue, e
    Output k, f
    Logic BitWidth, mask
    for i = BitWidth − 1 downto 0 do
        if SumValue[i] then
            k ← i
            break
        end if
    end for
    mask ← (1 ≪ k) − 1
    f ← SumValue & mask
    if k ≥ e then
        f ← f ≫ (k − e)
    else
        f ← f ≪ (e − k)
    end if
    return k, f
end function
```

With the values of S or M (*SumValue*), the LOD first executes a leading-ones detection to determine the coarse delay index k. This process maps the pulse to a logarithmically encoded delay segment. Next, the residual bits below k are extracted and normalised to a e-bit resolution, which is then returned as the fine delay f. By decomposing the overall path delay into pairs (k, f), the LOD compresses an exponential path space to logarithmic complexity while maintaining timing resolution.

3) ***Delay accumulation****:* The delay accumulation module comprises a differential delay path, a Vernier time-to-digital converter (TDC) [14], and a single-rail (SR) delay path. The differential delay path (Fig. 4) is controlled by the coarse fine parameter (k, f) generated by the LOD block, where the coarse unit delay is τ and the maximum coarse delay is $k\,\tau$. The fine unit delay is $\tau/2^e$, ensuring that the maximum span of fine delay remains τ while permitting a resolution of e. With the simultaneous launch event *raceDR*, the arrival instants of the two rails (*raceS* and *raceM*) encode the sign and magnitude of the class sum. A Vernier TDC digitises these race pulses to produce a compact delay code dc. Subsequently, dc configures a digitally-controlled delay element (DCDE) to realise the final SR delay path; because of the delay compression, this stage requires only a short length. Typical DCDE implementations may employ digital multiplexed segments [12], [15], or analogue structures such as current-starved inverters [16] and shunt-capacitor inverters [17].

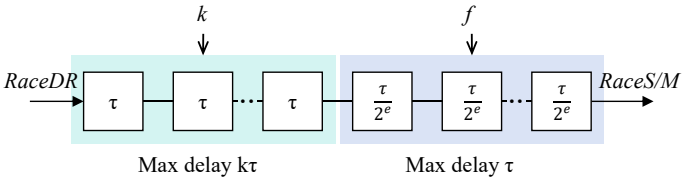


Fig. 4: Differential delay path.

4) ***WTA arbitration****:* The WTA arbitration module monitors the rising edge of the concurrent m race signals, denoted as *RaceClass*$[m-1{:}0]$. It grants the first-arriving pulse, selecting the class index with the highest arithmetic value. Since the output *grant*$[m-1{:}0]$ is a one-hot vector, the WTA also acts as the terminal of the time-domain path, interfacing directly with the subsequent digital-domain ports. Two WTA implementations are supported:

(i) **Tree-Based Arbiter (TBA)** [12]: The TBA uses a QDI hierarchy that propagates local winner signals forward until a global winner is recognised. For a system with m competing classes, this architecture requires $\lceil \log_2 m \rceil$ arbitration layers arranged in a binary tree, along with a total of $m-1$ identical TBA cells.

(ii) **Mesh-Like Arbiter** [18]: This topology is an all-pair cyclic-comparison network; the final winner emerges after $m-1$ stages and is implemented using $\dfrac{m(m-1)}{2}$ mutual exclusion (Mutex) cells.

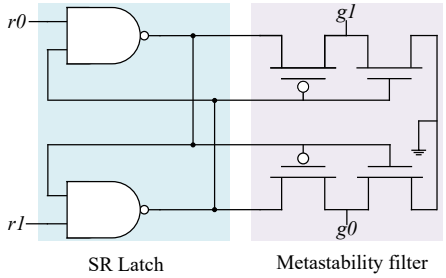


Fig. 5: Gate-level netlist of a Mutex

Figure 5 shows the gate-level netlist of the Mutex [19]. This component consists of a Set-Reset latch formed by two cross-coupled NAND gates, which detect the rising edge of competing signals. Specifically, when a faster input transitions

from low to high, the corresponding NAND pair masks the slower input. However, if two signals arrive within a time interval shorter than the intrinsic feedback propagation delay of the NAND gate, the latch may enter a metastable state. A metastability filter is utilised downstream of the Set-Reset latch to mitigate this. Additionally, Table I presents a theoretical performance analysis of these implementations.

TABLE I: Theoretical analysis of WTA implementations

Config.	Arbitration Depth	Cell Count	Arbitration Latency
TBA	$\log_2 m$	$m-1$	$\log_2 m\,(d_{\text{Mutex}} + d_{\text{OR}} + d_{\text{C-Element}})$
Mesh-Like	$m-1$	$\frac{m(m-1)}{2}$	$(m-1)\,d_{\text{Mutex}}$

5) ***Four to two phase interface****:* The proposed classification functional module follows a four-phase handshake protocol. In this protocol, the activation and deactivation of the differential and SR race signals are each controlled by a Muller C-element. In contrast, the TM inference pipeline controller uses a two-phase handshake protocol; therefore, a TFF is integrated at the boundary to provide a reliable interface.

TABLE II: Truth table of a two-input Muller C-element

a	b	c
0	0	0
0	1	c_{prev}
1	0	c_{prev}
1	1	1

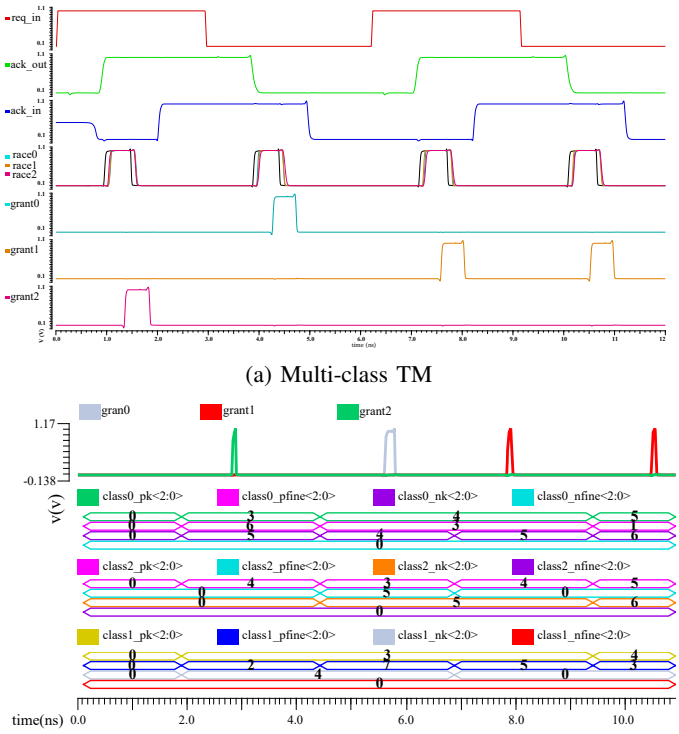
The Muller C-element, which is a state-holding component in asynchronous circuit design, exhibits the following characteristic behaviour: its output transitions to '1' only when all inputs are '1', drops to '0' only when all inputs are '0', and retains its previous state in all other scenarios. The truth table for a two-input Muller C-element is presented in Table II.

III. Experimental Results

A. Functional Verification

The verification circuits for the proposed architecture were built in Cadence Virtuoso using TSMC's 65nm CMOS technology and evaluated in an analogue mixed-signal simulation environment. The Iris classification dataset [20] was adopted for functional verification, comprising 16 features, 12 clauses, and 3 classes. The resulting verification waveforms (see Fig. 6) illustrate two inference processes: (a) the multi-class TM, where class selection is determined through a Hamming-distance driven WTA arbitration mechanism (b) the CoTM, in which the class delay is first encoded in differential, then compressed using a LOD scheme into coarse and fine delay, and subsequently decided by WTA arbitration. For the identical input feature vector, both TM algorithms and their hardware realisations performed the correct prediction, supporting the target class sequence (2, 0, 1, 1).

Purely digital-domain inference implementations for both multi-class TM and CoTM were realised as baselines. Functionally identical synchronous and asynchronous pipelines

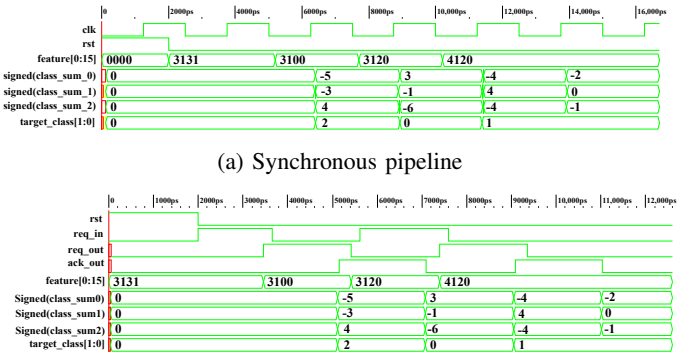


(a) Multi-class TM

(b) CoTM

Fig. 6: Waveform of the proposed DT-domain architecture

were developed in System Verilog, following Algorithm 1 2 3. Functional verification, logic synthesis, and physical implementation were executed with Cadence Xcelium, Genus, and Innovus. The synchronous and asynchronous digital-domain multi-class TM waveform is depicted in Fig. 7, while the CoTM result is shown in Fig. 8. The experiments confirm that all logically equivalent TM implementations achieve identical inference accuracy.



(a) Synchronous pipeline

(b) Asynchronous BD pipeline

Fig. 7: Waveform of digital implementation of multi-class TM

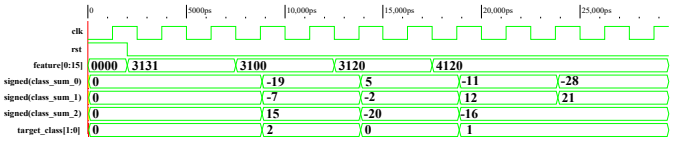
B. Performance Evaluation

The inference process of TMs are evaluated as:

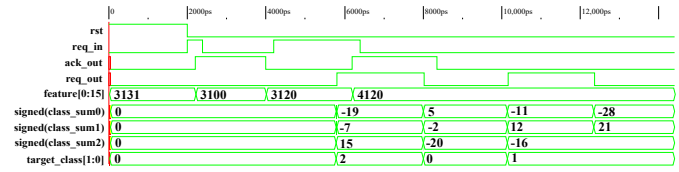
$$\text{Throughput}_{\text{TM}} = 2\,F\,C\,K\,f_{\text{infer}} \tag{3}$$

TABLE III: Comparison With Some State-of-the-Art Works.

Parameter	**[21]**	**[4]**	**[8]**	**[11]**	**Proposed**	**Proposed**
Architecture	Async QDI	Async BD	Sync	Async QDI	Async BD	Async BD
Computing Domain	Digital	Digital	Time	Digital	Time	Hybrid
Technology (nm)	65	28	65	65	65	65
Voltage (V)	1.2	0.9	1.2	1.2	1.0	1.0
Energy Eff. (TOP/J)	1.87	0.42	116	873	3329	750.79
ML Algorithm	CNN	SNN	BNN	Multi-class TM	Multi-class TM	CoTM



(a) Synchronous pipeline



(b) Asynchronous BD pipeline

Fig. 8: Waveform of digital implementation of CoTM

where F represents the number of input features, C is the number of clauses, K is the number of classes, and f_{infer} is the effective inference frequency.

$$\text{EnergyEfficiency}_{\text{TM}} = \frac{\text{Throughput}_{\text{TM}}}{1000\,P} \tag{4}$$

Where P is the average power consumption. Throughput is measured in giga operations per second (GOp/s), and energy efficiency is expressed in tera operations per joule (TOp/J).

Table IV summarises the performance metrics of various implementations, covering fully digital synchronous and asynchronous BD baselines, as well as the proposed architectures for both multi-class TM and CoTM.

TABLE IV: Performance summary

Implementation	**Throughput** GOp/s	**Energy Efficiency** TOp/J
Multi-class, synchronous	380	948.61
Multi-class, asynchronous BD	510	1381.65
Multi-class, proposed	402	3290.00
CoTM, synchronous	230	304.65
CoTM, asynchronous BD	350	397.60
CoTM, proposed	419	750.79

Under identical functionality, the proposed architecture delivers substantial energy efficiency while sustaining or enhancing inference throughput. For multi-class TM, energy efficiency rises by 247% over the synchronous digital baseline, with a throughput increase of 5.8%. Compared to the asynchronous BD architecture, the proposed design sacrifices a 21% throughput, improving energy efficiency by 138%. In CoTM, the architecture simultaneously boosts throughput by 82% and energy efficiency by 146% versus the synchronous reference. Compared to the asynchronous BD counterpart, this approach improves 20% throughput and 89% energy efficiency. Therefore, across both TM variants, this approach almost matches or exceeds the digital alternatives all around.

C. Stat-of-the-art Work Comparison

Table III compares the proposed designs with several state-of-the-art ML accelerators. [21] removes the global clock in a CNN engine by introducing a Dual-Rail data path for the MAC array. [4] adopts an asynchronous pipeline for implementing SNN, where click-element-based pipelines facilitate an event-driven communication mechanism between neuron layers. Although both works benefit from clock removal and leverage computational sparsity, their arithmetic operations—matrix MACs in [21] and membrane-potential updates in [4]—are still confined to the digital domain, which limits their further energy efficiency.

In contrast, [8] presents a time-domain accelerator for BNNs. This approach employs delay-encoded weights through a laddered inverter network, followed by delay accumulation and a WTA arbitration scheme. This time-encoded processing achieves a two-order-of-magnitude improvement in energy efficiency compared to other digital-domain NN architectures.

The [11] is an ultra-low-power, multi-class TM edge chip using an asynchronous Dual-rail data path. Due to the lightweight nature of TM inference and the asynchronous characteristics, the design outperforms traditional NN accelerators from the aspect of energy consumption.

The final two columns of the table present the proposed architectures. The multi-Class TM design requires only Hamming-distance evaluation and arbitration. Integrating the time-domain logic in [12] with an asynchronous BD controller achieves fully time-domain computation, achieving the highest energy efficiency, several orders of magnitude greater than all other surveyed architectures. The CoTM version employs a hybrid digital-time-domain approach, resulting in significant energy efficiency and throughput improvements compared to functionally equivalent pure-digital synchronous and asynchronous implementations.

IV. Conclusion

This work introduces effective approaches to mitigate the energy overhead associated with the extensive arithmetic operations required in TM classification. The design illustrates

that a fully time-domain computing flow achieves optimal energy efficiency for multi-class TM through Hamming-distance evaluation and arbitration. Conversely, for CoTM, which relies on MAC operations over a signed weight matrix, the proposed design employs a hybrid digital-time-domain approach. This approach processes the sign and magnitude components digitally, followed by differential encoding and LOD compression of the delay paths. A WTA arbitration system manages the final decision-making in the time domain.

To provide a comprehensive comparison, the analogous functionality of each TM algorithm was implemented digitally in both synchronous and asynchronous styles. This demonstrates the advantages of the proposed designs while offering additional options for designers who prefer digital-domain computing.

The aforementioned methodology confirms that asynchronous control combined with time-domain computation makes TM inference well-suited for resource-constrained edge platforms. This study outlines a clear path toward scalable, low-power ML deployments at the edge, bridging the gap between conventional ML accelerators and emerging event-driven time-domain hardware.

REFERENCES

[1] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[2] K. Arabi, "Low power design techniques in mobile processes," in *Proceedings of the 2014 International Symposium on Low Power Electronics and Design*, ser. ISLPED '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 1–2. [Online]. Available: https://doi.org/10.1145/2627369.2631634

[3] A. Younis, B. Qiu, and D. Pompili, "Latency-aware hybrid edge cloud framework for mobile augmented reality applications," in *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE Press, 2020, p. 1–9. [Online]. Available: https://doi.org/10.1109/SECON48991.2020.9158429

[4] D. Huo, J. Zhang, J. Zhang, and H. Chen, "A 28nm energy-efficient asynchronous snn accelerator with on-chip learning for gas recognition," in *2023 28th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, 2023, pp. 42–47.

[5] J. Wang, S. Xiao, J. Luo, B. Li, L. Zhou, and Z. Yu, "An end-to-end bundled-data asynchronous circuits design flow: From rtl to gds," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 33, no. 1, p. 154–167, Jan. 2025. [Online]. Available: https://doi.org/10.1109/TVLSI.2024.3464870

[6] L. Chang, X. Zhao, C. Li, S. Yang, S. Lin, and J. Zhou, "Towards intelligent-edge computing: Application, architecture, circuit, and device perspective," in *2021 IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA)*, 2021, pp. 125–126.

[7] Y. Kong, X. Chen, X. Si, and J. Yang, "Evaluation platform of time-domain computing-in-memory circuits," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 3, pp. 1174–1178, 2023.

[8] H. A. Maharmeh, N. J. Sarhan, M. Ismail, and M. Alhawari, "A 116 tops/w spatially unrolled time-domain accelerator utilizing laddered-inverter dtc for energy-efficient edge computing in 65 nm," *IEEE Open Journal of Circuits and Systems*, vol. 4, pp. 308–323, 2023.

[9] O.-C. Granmo, "The tsetlin machine – a game theoretic bandit driven approach to optimal pattern recognition with propositional logic," 2021. [Online]. Available: https://arxiv.org/abs/1804.01508

[10] S. Glimsdal and O.-C. Granmo, "Coalesced multi-output tsetlin machines with clause sharing," 2021. [Online]. Available: https://arxiv.org/abs/2108.07594

[11] A. Wheeldon, A. Yakovlev, R. Shafik, and J. Morris, "Low-latency asynchronous logic design for inference at the edge," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2021, pp. 370–373.

[12] T. Lan, O. Ghazal, S. Ojukwu, K. Krishnamurthy, R. Shafik, and A. Yakovlev, "An asynchronous winner-takes-all arbitration architecture for tsetlin machine acceleration," in *2024 22nd IEEE Interregional NEWCAS Conference (NEWCAS)*, 2024, pp. 16–20.

[13] A. Peeters, F. te Beest, M. de Wit, and W. Mallon, "Click elements: An implementation style for data-driven compilation," in *2010 IEEE Symposium on Asynchronous Circuits and Systems*, 2010, pp. 3–14.

[14] P. Dudek, S. Szczepanski, and J. Hatfield, "A high-resolution cmos time-to-digital converter utilizing a vernier delay line," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 2, pp. 240–247, 2000.

[15] D. Miyashita, R. Yamaki, K. Hashiyoshi, H. Kobayashi, S. Kousai, Y. Oowaki, and Y. Unekawa, "An ldpc decoder with time-domain analog and digital mixed-signal processing," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 1, pp. 73–83, 2014.

[16] J. Dunning, G. Garcia, J. Lundberg, and E. Nuckolls, "An all-digital phase-locked loop with 50-cycle lock time suitable for high-performance microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 4, pp. 412–422, 1995.

[17] M. Abas, G. Russell, and D. Kinniment, "Built-in time measurement circuits – a comparative design study," *IET Computers Digital Techniques*, vol. 1, pp. 87–97, 2007. [Online]. Available: https://digital-library.theiet.org/doi/abs/10.1049/iet-cdt%3A20060111

[18] S. Golubcovs, "Multi-resource approach to asynchronous soc : design and tool support," 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:36343957

[19] V. Khomenko, D. Sokolov, A. Mokhov, and A. Yakovlev, "Waitx: An arbiter for non-persistent signals," in *2017 23rd IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, 2017, pp. 33–40.

[20] R. A. FISHER, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x

[21] S. Xiao, W. Liu, J. Lin, and Z. Yu, "A data-driven asynchronous neural network accelerator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 9, pp. 1874–1886, 2021.