Cortical Columns Computing Systems (C3S): Towards a Silicon Neocortex for AI Compute

Harideep Nair and John Paul Shen
Electrical and Computer Engineering Department, Carnegie Mellon University

***Abstract*—Reverse-engineering the human neocortex has been a grand challenge in machine learning and computer architecture. Although deep neural networks, born from this challenge, have achieved remarkable success, they resort to intense biologically implausible statistical processing on large datasets resulting in exponentially increasing compute demand. Recent neuroscience research suggests Cortical Columns as the fundamental processing units in the neocortex that encapsulate intelligence. This abstract presents ongoing research focusing on building such *Cortical Columns Computing Systems (C3S)* as a first step towards a silicon neocortex with brain-like capabilities and efficiency. Initial findings indicate that designing extremely energy-efficient intelligent C3S-based sensory processing chiplets, using off-the-shelf CMOS technology, is quite feasible and very promising, and certainly warrants further research exploration.**

***Index Terms*—neuromorphic computing, cortical columns, temporal neural networks, online sensory processing**

I. Introduction and Background

Human brain (neocortex) is highly adept at sensory processing tasks. Current deep neural networks (DNNs) are attempting to replicate such human-like sensory processing but using conventional hardware that employ Turing computation model and von-Neumann computer architecture targeting numerical computation. DNNs have achieved excellent results in visual object recognition, natural language processing, etc. However, the computation required is increasing at an exponential rate. Many believe this trend is not sustainable and an alternative paradigm is needed [9].

Neuromorphic spiking neural networks (SNNs) offer an alternative approach to AI compute, based on brain-inspired principles and brain-like architecture. Temporal Neural Networks (TNNs) [7] emerged recently as a more biologically plausible approach utilizing efficient temporal encoding, local spike timing dependent plasticity learning and neocortical hierarchy comprising of synapses, neurons, (mini)columns, and layers. Further, Hawkins' new theory on intelligence [2] proposes *Cortical Columns (CCs)* as the fundamental neocortical processing units. Higher forms of intelligence involve result from larger number of CCs. The neocortex gains its intelligence through CC's ability to model sensory information in structured Reference Frames (RFs), and learn complete models of objects through predict-sense-update feedback loops. There is synergy between CCs and TNNs; CCs resemble TNNs with recurrence, embodied in the RFs.

This paper presents our ongoing research in designing and implementing *Cortical Columns Computing Systems (C3S)*, capable of performing highly energy-efficient online sensory processing [6]. Unlike many neuromorphic works that focus on exotic device technologies such as resistive memory, we utilize standard off-the-shelf digital CMOS targeting near-future mass market computing. We envision C3S-based neuromorphic sensory processing units (NSPUs) as specialized chiplets that can be easily integrated into any edge AI compute substrate.

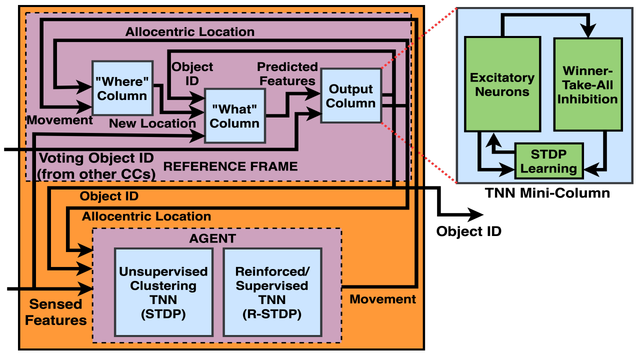

Fig. 1: Each Cortical Column (CC) consists of five TNN-style mini-columns: *Where*, *What* and *Output* mini-columns that together implement the Reference Frame (RF), and *unsupervised* and *supervised* mini-columns comprising the agent. For visual object recognition, the respective functionalities of the three RF mini-columns are: derive locations of sensor on the object, map features to locations, and derive the object ID based on the feature map. Each CC learns continuously through feedback (recurrence) by capturing meaningful patterns in its RF.

II. Cortical Columns Computing System

Inspired from Smith's macrocolumn architecture [8], our proposed Cortical Column (CC) microarchitecture (shown in Figure 1) consists of two components: 1) a *Reference Frame* that maintains a "map" of the sensory information, and 2) an *Agent* that achieves goal-oriented behavior based on information from the Reference Frame and the current input signals. Agent comprises of two TNN-type mini-columns performing unsupervised clustering and supervised classification.

Reference Frame involves three functional modules in the form of three types of mini-columns: *Where*, *What* and *Output*. In the context of visual object recognition, the three types of mini-columns together create models of objects by tracking locations of features on the object. *Output* determines the object identity. *Where* generates location of sensor on the object based on feedback from *Output* and the latest movement information from the agent. *What* predicts object features

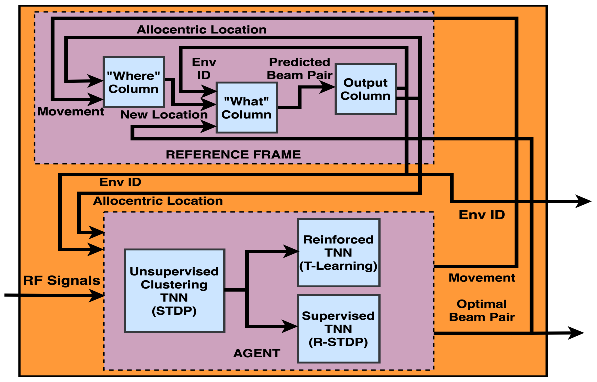

Fig. 2: Example Cortical Column for Beamforming: *What* mini-column within RF maps optimal beam pairs to UE locations. *Unsupervised*, *supervised* and *reinforced* mini-columns within agent respectively perform input signal clustering, optimal beam pair prediction, and UE movement tracking.

based on the result from *Where* and updates its model based on the actual sensory input and the feedback from *Output*.

III. Initial TNN Results

We have developed a microarchitecture model and custom buiding block macros for implementing highly efficient TNN designs [4], [5]. We illustrated that single TNN mini-columns can perform unsupervised time-series clustering within 40 μW power while outperforming majority of the state-of-the-art works. Further, multi-layer TNNs can achieve state-of-the-art 99% accuracy on MNIST digit recognition within 18 mW power, while enabling on-chip online fast continuous learning. More details on these results can be found in [1], [4], [5].

IV. Beamforming as a Killer Application

Beamforming [3] is indispensable to 5G and future 6G communications, where one or more user equipments (UEs) need to communicate with a base station via the most optimal transmit-receive beam pairs. The environmental map of UEs as observed by a base station can be stored in the RF of a CC, which can be used to perform the adaptive beamforming task. We use this as an application driver for our C3S development.

The example CC microarchitecture for beamforming (Figure 2) consists of 1) an agent that keeps track of moving UEs and learns optimal beam pairs from input signals, and 2) an RF that stores learned information from agent and infers predicted beam pairs from UE locations. Here, *Where* mini-column maps optimal beam pairs to UE locations. This is a topic of active research, as part of our C3S effort.

V. Future Research Directions

The overarching goal of our current research is to develop an end-to-end framework (Figure 3) that can automatically translate application-specific C3S models in software to highly customized NSPU chiplets. The framework consists of two main components: 1) *C3S-Sim* which consists of a PyTorch application simulator for C3S functional modeling, and a C++ cycle-accurate architectural simulator for hardware performance estimation; and 2) *C3S-Syn* that includes PyVerilog conversion for automated RTL generation from PyTorch, automated RTL-to-GDSII flow that leverages C3S-specialized custom macro cells and generates application-specific post-layout netlist for a specific NSPU chiplet design. We believe such C3S-based NSPU chiplets can be truly "intelligent" as per Hawkins' definition, and can enable contextualization and personalization of applications and services for edge AI.

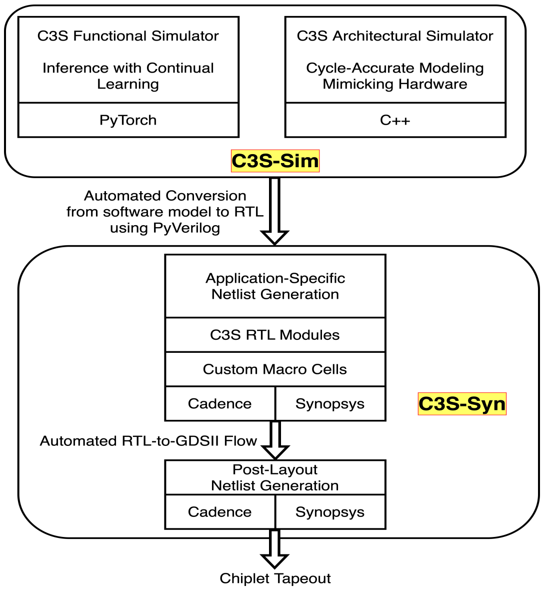
C3S Functional Simulator
Inference with Continual Learning
PyTorch
C3S Architectural Simulator
Cycle-Accurate Modeling Mimicking Hardware
C++
C3S-Sim
Automated Conversion from software model to RTL using PyVerilog
Application-Specific Netlist Generation
C3S RTL Modules
Custom Macro Cells
Cadence
Synopsys
C3S-Syn
Automated RTL-to-GDSII Flow
Post-Layout Netlist Generation
Cadence
Synopsys
Chiplet Tapeout

Fig. 3: Envisioned end-to-end *Cortical Columns Computing System (C3S)* design framework: *C3S-Sim* consists of a PyTorch tool to design application-specific C3S functional models and a cycle-accurate architectural simulator for hardware performance estimation. *C3S-Syn* incorporates the extended microarchitecture model and functional building blocks for C3S implementation, with an automated design flow to translate PyTorch functional models to application-specific chiplets.

References

[1] S. Chaudhari, H. Nair, J. M. Moura, and J. P. Shen, "Unsupervised clustering of time series signals using neuromorphic energy-efficient temporal neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.
[2] J. Hawkins, *A thousand brains: A new theory of intelligence*, 2021.
[3] K. Ma, Z. Wang, W. Tian, S. Chen, and L. Hanzo, "Deep learning for mmwave beam-management: State-of-the-art, opportunities and challenges," *IEEE Wireless Communications*, 2022.
[4] H. Nair, J. P. Shen, and J. E. Smith, "A microarchitecture implementation framework for online learning with temporal neural networks," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2021.
[5] H. Nair, P. Vellaisamy, S. Bhasuthkar, and J. P. Shen, "Tnn7: A custom macro suite for implementing highly optimized designs of neuromorphic tnns," in *IEEE Computer Society Annual Symposium on VLSI*, 2022.
[6] J. P. Shen and H. Nair, "Cortical columns computing systems: Microarchitecture model, functional building blocks, and design tools," 2023.
[7] J. E. Smith, "Space-time computing with temporal neural networks," *Synthesis Lectures on Computer Architecture*, vol. 12, no. 2, 2017.
[8] J. E. Smith, "A macrocolumn architecture implemented with temporal (spiking) neurons," *arXiv preprint arXiv:2207.05081*, 2022.
[9] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," *arXiv:2007.05558*, 2020.