Cerebra-S: a Configurable Neuromorphic Network-on-Chip Accelerator Architecture for Small-Scale Spiking Neural Networks

Saadia Jameel*
Department of Computer Engineering
Faculty of Engineering
University of Peradeniya
Peradeniya, Sri Lanka
e18147@eng.pdn.ac.lk

Thamish Wanduragala*
Department of Computer Engineering
Faculty of Engineering
University of Peradeniya
Peradeniya, Sri Lanka
e18379@eng.pdn.ac.lk

Dr. Isuru Nawinne
Department of Computer Engineering
Faculty of Engineering
University of Peradeniya
Peradeniya, Sri Lanka
isurunawinne@eng.pdn.ac.lk

Prof. Roshan Ragel
Department of Computer Engineering
Faculty of Engineering
University of Peradeniya
Peradeniya, Sri Lanka
roshanr@eng.pdn.ac.lk

Prof. Smruti Sarangi
Computer Science and Engineering
Indian Institute of Technology Delhi
Delhi, India
srsarangi@cse.iitd.ac.in

***Abstract*—Cerebra-S is a configurable neuromorphic accelerator architecture designed for inferencing Spiking Neural Networks (SNNs). It supports any SNN topology and efficiently utilizes hardware resources, whether the SNN is fully or sparsely connected. The modules of the Cerebra-S chip core are optimized for high speed and low power consumption using dedicated hardware. It features a low-power processor for management tasks. The processor runs the RISC-V standard Instruction Set Architecture (ISA). The modules exploit parallelism to maximize the efficiency of spiking neural networks. Real-world applications, such as lane-keeping vehicles, gaming, image classification using STDP learning, generative pre-trained language models, and energy-efficient natural language processing, use SNNs with fewer than a few thousand neurons and non-layered structures. We present a neuromorphic chip that simulates small-scale SNNs using hardware resources sparingly. Existing neuromorphic hardware for small-scale SNNs has many restrictions such as neuron connections, and number of neurons per layer, due to their focus on simplicity over flexibility. The design discussed in this paper balances simplicity and flexibility, enabling synchronous communication between neurons and memory distribution to reduce bottlenecks and memory access time.**

***Index Terms*—neuromorphic, artificial neural network, spiking neural network, RISC-V, accelerator**

I. Introduction

The increasing demand for neuromorphic computing has driven the development of chips that emulate the brain's neural networks, such as Intel's Loihi [10] and IBM's TrueNorth [1]. These state-of-the-art systems offer the potential for vast computational power and energy efficiency by supporting spiking neural networks (SNNs) with millions of neurons [2]. However, for many applications—especially those in resource-constrained environments like edge computing, IoT devices, and specialized AI tasks—only a fraction of this neural capacity is required. Exploiting such large-scale hardware for small-scale tasks leads to unnecessary costs and inefficiencies, both in terms of computational power and energy consumption.

This work presents Cerebra-S, a neuromorphic chip designed specifically to address these inefficiencies by tailoring neuromorphic hardware to applications that need only a few thousand neurons. By focusing on optimizing performance for smaller neural networks, Cerebra-S offers a cost-effective, energy-efficient solution for the many applications that do not require the massive neuron capacity of existing chips.

Spiking Neural Networks (SNNs) are often regarded as the third generation of neural networks, offering distinct advantages over traditional architectures like Artificial Neural Networks (ANNs) [4]. One of the key benefits of SNNs is their energy efficiency. They are event-driven, processing information only when neurons spike, which significantly reduces power consumption compared to ANNs. With ANNs, continuous data processing is required. This makes SNNs particularly suitable for low-power devices [3]. Furthermore, SNNs are more biologically plausible, as they emulate the time-based operation of real neurons, communicating information via spikes in a way that mirrors the human brain. SNNs typically achieve a superior power-to-performance ratio, making them ideal for embedded systems and edge computing, where energy constraints are critical.

Key contributions of the Cerebra-S chip include:

- Tailored design for small-scale applications: *Cerebra-S* is designed to efficiently handle tasks requiring a few

*These authors contributed equally to this work

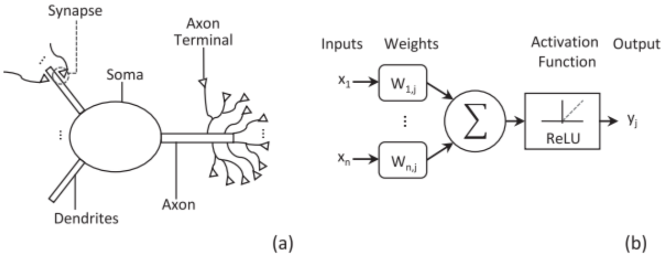

Fig. 1. (a) A biological neuron. (b) An Artificial Neural Network with n inputs with their corresponding synaptic weights. All weighted inputs are added and an activation function determines the output. Source: [20]

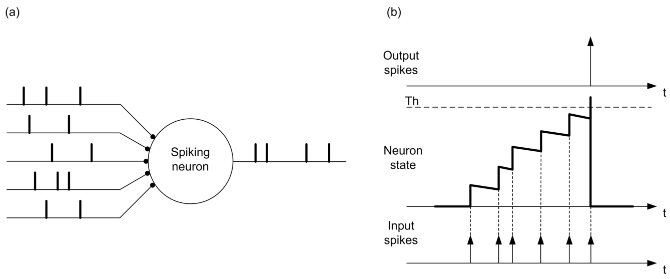

Fig. 2. (a) A spiking neuron generating output spikes due to the influence of its input spikes. (b) Evolution of the LIF model neuron state. Source: [21]

thousand neurons, avoiding the unnecessary overhead of larger chips.

- Cost and energy efficiency: The chip offers a significant reduction in production and operational costs by scaling down neuron capacity to match specific application needs.
- Optimized architecture for edge and IoT applications: *Cerebra-S* is ideal for resource-constrained environments, making it suitable for AI tasks in mobile, embedded, and IoT systems.
- Enhanced accessibility of neuromorphic computing: By lowering the cost barrier, *Cerebra-S* enables wider adoption of neuromorphic solutions in both academic research and commercial products.

This paper presents the design of the scalable and configurable accelerator architecture, Cerebra-S, controlled by a processor based on the RISC-V instruction set architecture (ISA). This design allows for hardware-level support to exploit the advantages of processing using spiking neural networks. The simulation of the design is done on an FPGA. RISC-V was chosen as the base ISA since it is not only highly practical and popular, but also completely open source, amenable to custom extensions, and is viable for low power applications.

The proposed design consists of hardware nodes that are specialized to perform operations required to implement spiking networks. The nodes are connected using a specialised interconnect. The architecture adopts an event-driven messaging mechanism to effectively emulate the activity of spiking neurons. The nodes are designed with hardware-level flexibility, enabling them to efficiently run various types and models of SNNs.

II. Background Study

A. Spiking Neural Networks

SNNs follow complex design rules and represent a newer variant of ANNs. Unlike conventional ANNs, which transmit information in propagating cycles, SNNs transmit spikes as events throughout the neural network. A spike is considered a rapid change in voltage that occurs over a short period of time. In the context of the brain, when a synapse receives an action potential, or spike, from one of its pre-synaptic neurons, it transmits the spike to its post-synaptic neurons.

A comparison between a biological neuron and an artificial neuron is shown in Fig. 1. The figure shows how the inputs and outputs of an artificial neuron correspond to synapses of a brain, through which spike information is transmitted. This transmission leads to an update of the post-synaptic neuron's membrane potential, which is influenced by factors such as the weight of the synapse and other neuron parameters. Fig 2. shows how a neuron fires and generates a spike on its output. This functionality is replicated in an SNN. These SNNs can be modeled in various ways to provide the aforementioned functionality.

To understand how SNNs operate, it is essential to first define a generic model of a neuron. A basic neuron model consists of three key components:

- Input synapses, which receive signals (spikes) from other neurons.
- Membrane potential, which accumulates input signals and determines whether the neuron will fire.
- Output synapses, through which the neuron sends spikes to downstream neurons when a certain threshold is reached.

B. Neuron Models

Various neuron models have been proposed to govern how a single neuron functions. They have different levels of complexity. Leaky Integrate and Fire (LIF) [5], Quadratic Integrate and Fire [22], Izhikevich [6], Hodgkin-Huxley [7] and Fitzhugh-Nagumo [8] are a few commonly found neuron models. While the Hodgkin-Huxley model is the most biologically plausible, it is also prohibitively high in terms of power consumption. The LIF model is relatively simple and computationally efficient. However, this simplicity makes it inadequate for realistically imitating the rich spiking dynamics of natural neurons. The Izhikevich and Fitzhugh-Nagumo models offer a compromise between these two extremes, being computationally effective and also sufficient to emulate the behaviour of natural neurons.

The Izhikevich neuron model could be described by the following two-dimensional system of ordinary differential equations,

$$v' = 0.04v^2 + 5v + 140 - u + I \quad (1)$$

$$u' = a(bv - u) \quad (2)$$

with after spike behaviour shown as,

$$\text{if } v \geq v_{\text{th}}, \text{ then } \begin{cases} v & \leftarrow c \\ u & \leftarrow u + d \end{cases} \tag{3}$$

The variables v, u, and I denote the membrane potential, the membrane recovery variable, and the input of the neuron respectively, whereas a,b,c and d are parameters that govern the behaviour of the neuron.

In the LIF neuron model, when a spike arrives at a neuron, the weight of the synapse through which the spike arrived is integrated to the membrane potential. Once the membrane potential value reaches the neuron's threshold value, the neuron fires and generates an output spike and the corresponding membrane potential value is reset. The membrane potential value decreases with time continuously and is found to be analogous to a resistor- capacitor (RC) circuit taking the form of the following equation,

$$RCv' = -v + RI \tag{4}$$

v and I are the membrane potential and the input of the neuron respectively. R and C are parameters representing the resistance and the capacitance of the RC circuit. This equation can be further simplified to a general form similar to that of the Izhikevich model.

$$v' = av + bI \tag{5}$$

Here, a and b are parameters governing the behaviour of the neuron.

The reset voltage can be set in different ways. It can either be set to a constant value or can be described by the following equation,

$$\text{if } v \geq v_{\text{th}}, \text{ then } v \leftarrow v - v_{\text{th}} \tag{6}$$

The Quadratic Integrate-and-Fire Neuron model is a type of LIF neuron model and is one of the simplest of models. Compared to the LIF neuron model, this model accommodates a quadratic term into the calculation of the neuron membrane potential value in each time step, hence the name. The quadratic term allows for a more realistic representation of the neuron's firing behaviour. Upon reaching a predefined peak potential value it is reset to a specific reset value. The following equation describes the membrane potential value update.

$$v' = v^2 + I \tag{7}$$

In addition to the parameters mentioned above, the refractory time (T_R) and leak rate are two other crucial parameters that influence the behaviour of an LIF neuron model. The T_R regulates the firing rate of neurons and ensures that neurons don't fire too rapidly. The leak rate determines the responsiveness of a neuron to incoming signals.

Furthermore, accurate biomimetic neuron models and their hardware have been achieved with analog electronics. A common analog emulator is Neurogrid [9]. It simulates one million neurons and six billion synapses in real time while consuming less than 2W of electrical energy. Consequently, several neuromorphic architecture designs are in use.

C. *Neuromorphic Architecture Implementations*

Neuromorphic research is still highly exploratory, with several proposed implementations aimed at simulating and accelerating SNNs. There are hardware solutions that support both large and small scale SNNs. Our research focuses on developing a more flexible, resource-efficient, and power-optimized design for SNNs with a reduced number of neurons, typically between 1000 and 4000.

1) Loihi: Intel's advanced neuromorphic chip Loihi 1 [10] adopts a variation of the CUBA leaky-integrate-and-fire model[23]. It features a manycore mesh comprising of 128 neuromorphic cores. An asynchronous network-on-chip (NoC) transports all communication between cores as packetized messages. A neuromorphic core implements 1024 spiking neural units. All these units, including their fan-in and fan-out connectivity, share configuration and state variables in ten architectural memories within the neuromorphic core. Such a design allows for a single event to be expanded into a large number of specialized dependent events that can operate in parallel to reduce the neuron update and spike resolution latency. It boasts of 5000 times lower energy-delay product (EDP) in solving large problems compared to conventional LARS [11] and FISTA [12] solvers.

2) Loihi2: The Loihi 2 chip by Intel [13] measures 30 square milimeters, exactly half in size and ten times faster than its predecessor. The speed improvement is the result of the increase in the number of digital neurons on the chip and that it stores data in tiny amounts spread across the mesh of neurons and not one big bank of memory. It has slightly less aggregate memory than Loihi1.

3) OpenSpike: The OpenSpike project [14] by Modaresi, F. et al. and research by Zhang, J. et al. [15] adopts the leaky integrate-and-fire neuron model at the hardware level and uses a time-multiplexed accelerator design. It has also a network implementation for interconnection between hardware neurons. While this ASIC implementation offers a clock speed of 40MHz and a supply requirement as low as 1.8V, it lacks the flexibility in programming and configuration.

4) SpiNNaker: On the other hand, SpiNNaker [16] focuses more on being large scale and able to simulate the behaviour of aggregates of up to a billion neurons in real time. It utilizes a network of nodes where each node consists of 18 ARM968 cores plus a 128Mbyte off-die Synchronous Dynamic Random Access Memory (SDRAM). 16 cores are used to support the spiking application, one core for system management tasks and another for fault tolerance. Inter-processor communication is based on an efficient multicast infrastructure using source-routed packets.

5) Low power design: Research groups have also focused on small-scale, low power accelerators for running SNNs. These designs mainly focus on small scale applications and

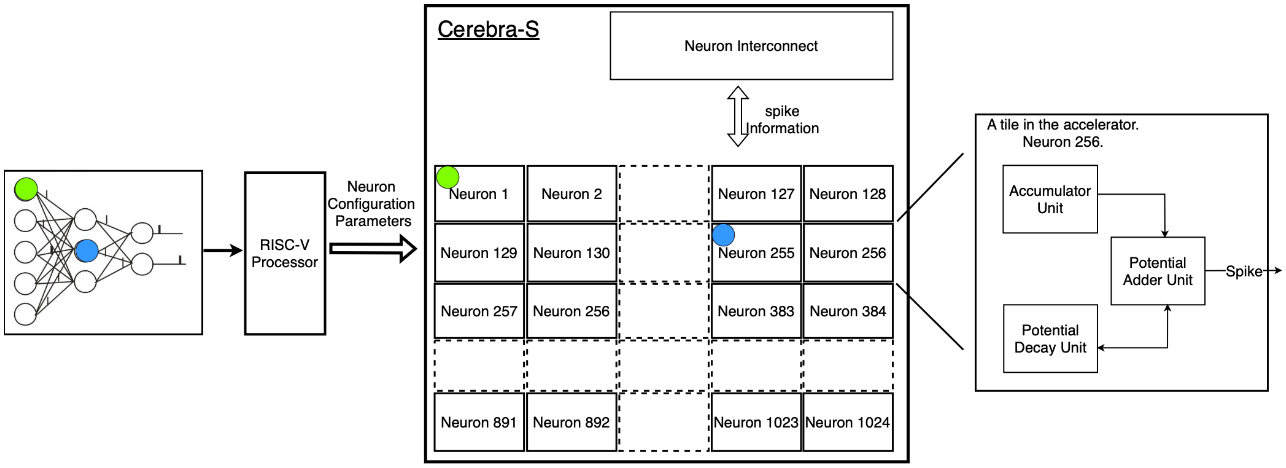

Fig. 3. The SNN is loaded onto Cerebra-S by the RISC-V processor. Every neuron in the SNN is allocated a dedicated hardware neuron as shown using the green and blue neurons. Spike information is communicated within the SNN network via the neuron interconnect.

support only a few neurons per chip compared to large-scale hardware implementations. While Loihi by Intel supports a maximum of 1024 neurons per core and 128 cores on a 60 mm^2 chip, a small-scale design chip by Chen et al. [17] supports only 4096 neurons on a 1.72 mm^2 chip and consumes less energy per synaptic operation. These low power processors use different techniques to improve their energy efficiency. Some of the techniques used with Chen et al., mainly focus on increasing the sparsity of an SNN according to the application. Sparsity, in a neural network refers to the phenomenon where a significant portion of the weights, activations, or connections within the network are zero or close to zero. The user has the ability to control the level of sparsity of the neural network according to the accuracy of the tested model. The hardware is able to classify the MNIST digits [18] at 89 percent accuracy at full connectivity and 88 percent accuracy for 50 percent weight sparsity. These designs use techniques such as power gating, voltage scaling, negative feedback derived from the current spike rate and reducing the number of clock cycles per time step, to reduce energy per classification by 17.4 times compared to full connectivity.

6) Zheng et al. [19]: introudces a low power consumption event driven hardware architecture for SNNs with Spike-timing-dependant plasticity (STDP) learning algorithm which is a commonly used learning algorithm in training SNNs. The topology of the whole system is changed to reduce the total expenses associated with the hardware, including power consumption, physical space, and operational complexity.

Chip architectures that support large-scale SNNs offer great performance when running smaller-scale SNNs as well. However, it goes without saying that a lot of resources will be wasted when an SNN with only a few hundred neurons is run on an architecture designed to support millions of neurons. While there are chips made to support smaller-scale SNNs, they often prioritize simplicity over flexibility. The hardware is structured in layers that correspond to the layers in the spiking network, inherently limiting the number of neurons per layer in the supported SNNs. These designs primarily support fully connected SNNs, with simple modifications to accommodate sparsely connected SNNs, such as by setting corresponding synaptic weights to zero. However, these modifications often result in wasted resources.

III. System Architecture Overview

The high-level diagram of our implementation is in Fig. 3. Each node within the Cerebra-S core is designated a single neuron at initialization. Each node comprises three functional units, namely, the accumulator unit, potential decay unit and the reset unit. Each neuron has and its own built-in memory. The memory in each node contains all local parameters and values needed for its computations, as seen in the equations previously stated. The neuron interconnect includes a memory segment specifically for storing the neuron placement graph. This graph maps out how neurons are connected to each other in the Spiking Neural Network (SNN). Essentially, it shows the layout of the network which is crucial for resolving spikes generated. These memory components are written to at initialization by the RISC-V processor and are referred to during each spike calculation for information passing. The model of the neurons and the assignment of logical neurons in an SNN to nodes, are configurable, tailored to corresponding use cases. During operation, the RISC-V-based processor coordinates the timing of spike processing through a clock pulse. Custom extensions to the standard RISC-V instruction set architecture (ISA) are needed to handle the initialization of the chip with information regarding the SNN.

IV. Cerebra-S Chip Overview

The Cerebra-S core uses the RISC-V processor for control operations. The term ’core’ hereafter refers to the Cerebra-S Chip Core. Positioned at the heart of the entire process, the core emulates the neurons of the SNN.

The core is designed to model a specific number of neurons, currently fixed at 1024, in a manner shown in Fig.3. Each logical neuron in the SNN is mapped to a corresponding node in the core, as illustrated in Fig. 3. Neurons can be loaded onto any available hardware node, unlike in other existing small-scale neuromorphic designs, where a neuron must be loaded correctly onto the hardware corresponding to its layer in the SNN. Our design thus does not limit the number of neurons that can be supported per layer. For clarity and ease of understanding, each node in the core will hereafter be referred to as a neuron.

Each neuron within the core consists of three key components mimicking the firing and refractory period of biological neurons: an accumulator unit, a potential decay unit, and a potential adder and reset unit, as shown in Fig. 4. The memory units within each module are also shown as colored boxes.

The entire core can be visualized in Fig. 6, as tiles of neuron modules where each tile represents an individual neuron. All neurons are connected to an interconnect, through dedicated first-in-first-out (FIFO) buffers at the interconnect's input and output. The buffers store and pass spikes. A neuron is identified using a neuron identifier, which is a combination of 12 bits, as shown in Fig. 5. The most significant 2 bits denote the core to which the neuron belongs. The remaining bits are its unique address within the core, thus the name "local neuron address". Although this paper explains Cerebra-S as having a single core, we allow for and anticipate scalability in the future to accommodate certain applications that may go beyond our current specification of 1024 neurons, thus employing a global addressing scheme from now on.

A. Neuron Interconnect

The neuron interconnect is responsible for transmitting spikes from one neuron to another. It has a look-up table structured as a Compressed Sparse Row (CSR), which details all downstream connections for a given neuron.

In SNNs, the number of synaptic connections per neuron can vary significantly. While some neurons may have only a few connections, others may have far more. Using a Compressed Sparse Row (CSR) matrix to store synaptic connections is particularly advantageous in this context compared to a dense adjacency matrix.

Neurons with fewer connections use less space, while neurons with more connections can utilize the memory saved by their sparser counterparts. This dynamic allocation avoids the rigid, wasteful structure of an adjacency matrix, where all neurons must allocate memory for a fixed number of connections, regardless of their actual connectivity.

This approach enables the storing of connections as shared memory in the neuron interconnect and not in an off-chip storage as it acts as a lightweight version of the adjacency matrix.

Each neuron in the core has two dedicated connections to the interconnect, one at its input interface and one at the output interface. When a neuron spikes, it communicates the spike information to the neuron interconnect via its dedicated FIFO buffer. The neuron interconnect then references the CSR to identify all downstream connections of the neuron spiked and sends a spike packet to each downstream neuron. The packets generated by the neuron interconnect include the source neuron address and take the structure of that shown in Fig. 5. These packets are received by the accumulator units of the target neurons. However, if Cerebra-S is scaled to have multiple cores, given a certain application's requirements, then the packet will have to include both the source and destination neuron address.

B. Accumulator Unit

The Accumulator Unit serves as the entry point to a neuron, responsible for adding all incoming spikes to the neuron's potential.

The input to the Accumulator Unit is a packet from the neuron interconnect. When a spike/ packet is received by a neuron from the neuron interconnect, the accumulator unit extracts the source address and looks up the corresponding weight for the connection. Each weight is represented by 32 bits, and is stored locally in this unit itself. The accumulator unit then aggregates all weights corresponding to input spikes coming from its upstream connections. As standard addition cannot be performed directly, a separate floating point addiction units are employed for this task. The output of the accumulator unit is a 32-bit value, which represents the aggregation of the incoming spike connections.

C. Potential Decay Unit

The Potential Decay Unit (PDU) is responsible for updating the neuron's potential over time, simulating natural neuronal behavior. The decay operation may vary depending on the selected neuron model, as different functions are applied to the membrane potential. For instance, in the case of the Leaky-Integrate and Fire neuron model, the membrane potential decays by a constant factor. This factor is dependent on the inference and is typically determined by the trained model.

The starting membrane potential value of every neuron is initially written to each PDU's local memory, by the processor. As the SNN progresses, this value must be continually updated. The update is facilitated by the Potential Adder and Reset Unit, which will be discussed next. The inputs to the PDU include a synchronization signal, provided by the processor, and the updated membrane potential value, generated by the Potential Adder and Reset Unit.

The output of the PDU is a 32-bit number representing the decayed potential of a given neuron. This decayed potential value is then forwarded to the Potential Adder and Reset Unit for further processing.

D. Potential Adder and Reset Unit

The Potential Adder and Reset Unit is tasked with two primary functions. Firstly, it adds the outputs of the Accumulator Unit and the PDU. Secondly, it determines whether the neuron must spike. When the membrane potential reaches its threshold value, the neuron is considered to have spiked. It then manages the resetting of the neuron's membrane potential value based on the spiking status. This value is dynamically influenced by

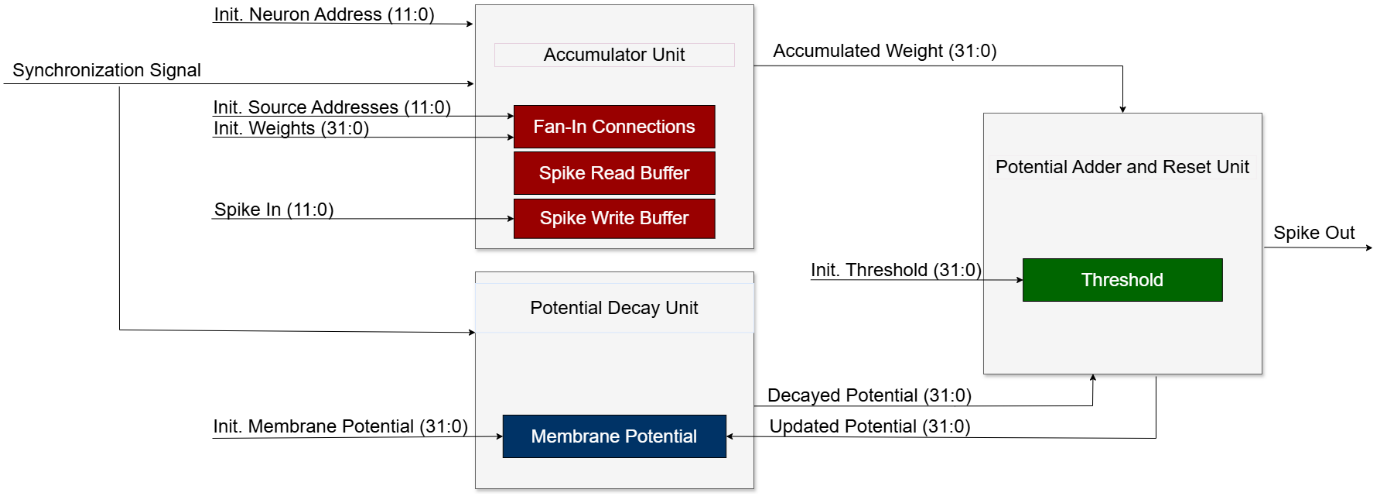

Fig. 4. A single neuron is made up of three units. Each unit has its own memory, storing parameters and data corresponding to the logical neuron that it is running. The memory units are shown in red, blue and green in the accumulator and potential decay and potential adder and reset unit respectively.

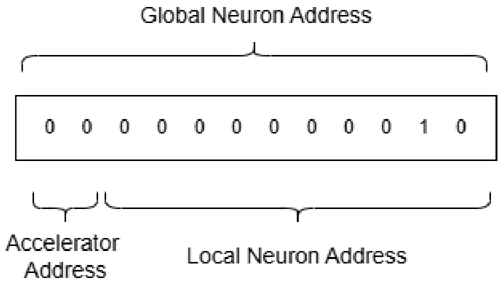

Fig. 5. Neuron Addressing Scheme. Packets generated at the neuron interconnect have the same format.

incoming spikes and the specific neuron model in operation. Upon a spike event, it resets to a value specified by the neuron model.

One output of this unit is the updated neuron potential value, which is written back to memory. The write is performed by the PDU and will be utilized in the subsequent time step. The other output is the spiking status of the neuron. This information is communicated to the neuron interconnect through a dedicated connection. Upon receiving this signal, the neuron interconnect initiates the required actions, as outlined previously.

E. Memory Management

Simulating an SNN in hardware requires storing a substantial amount of information, including synaptic weights, membrane potential values, threshold values, and the topology of the entire SNN connection. It is important to note that none of the operations require memory to be shared among neurons. Therefore, to reduce the memory access time, each neuron in the core is programmed with its own private memory, storing its parameters necessary for calculations.

Each neuron stores the synaptic weights of its upstream connections, through which it may receive spike information. This is stored as a CSR. To store received spike information there are two buffers in a neuron. One to which spikes are written to and one buffer from which the neuron reads spikes. Two read and write buffers are maintained to avoid spike overriding and reading wrong information. The write buffer is written to whenever a neuron receives spike information. Reading from the read buffer begins only at the start of a time step. The neuron also stores the current membrane potential value and its neuronal threshold value.

However, the downstream connections of all neurons are stored in the neuron interconnect, through which a spike travels before reaching the destinations that require the corresponding spike information. Since all spikes generated by every neuron must access this memory location, the neuron interconnect has potential bottlenecks. As each neuron has a dedicated connection to the neuron interconnect, spikes from different neurons will be processed in parallel. Multiple spikes originating from the same neuron will be resolved in a time-multiplexed manner.

V. RISC-V Processor Core

A RISC-V processor is integrated into the Cerebra-S core to manage memory writes and handle interrupts. The processor is a simple single-cycle RV32I core. The core is treated as a co-processor. The processor has a custom instruction set defined to facilitate communication with the core's memory. The processor is responsible for initializing the core memory with the SNN architecture that inferencing will be done on. Fig. 3. shows, at a high level, where the processor is placed in the design of the system. At the end of each time step, the processor sends interrupts to synchronize the operations of the core.

A. Custom Instructions

A modification made to the RV32I processor is the addition of custom instructions to handle data transfer between the

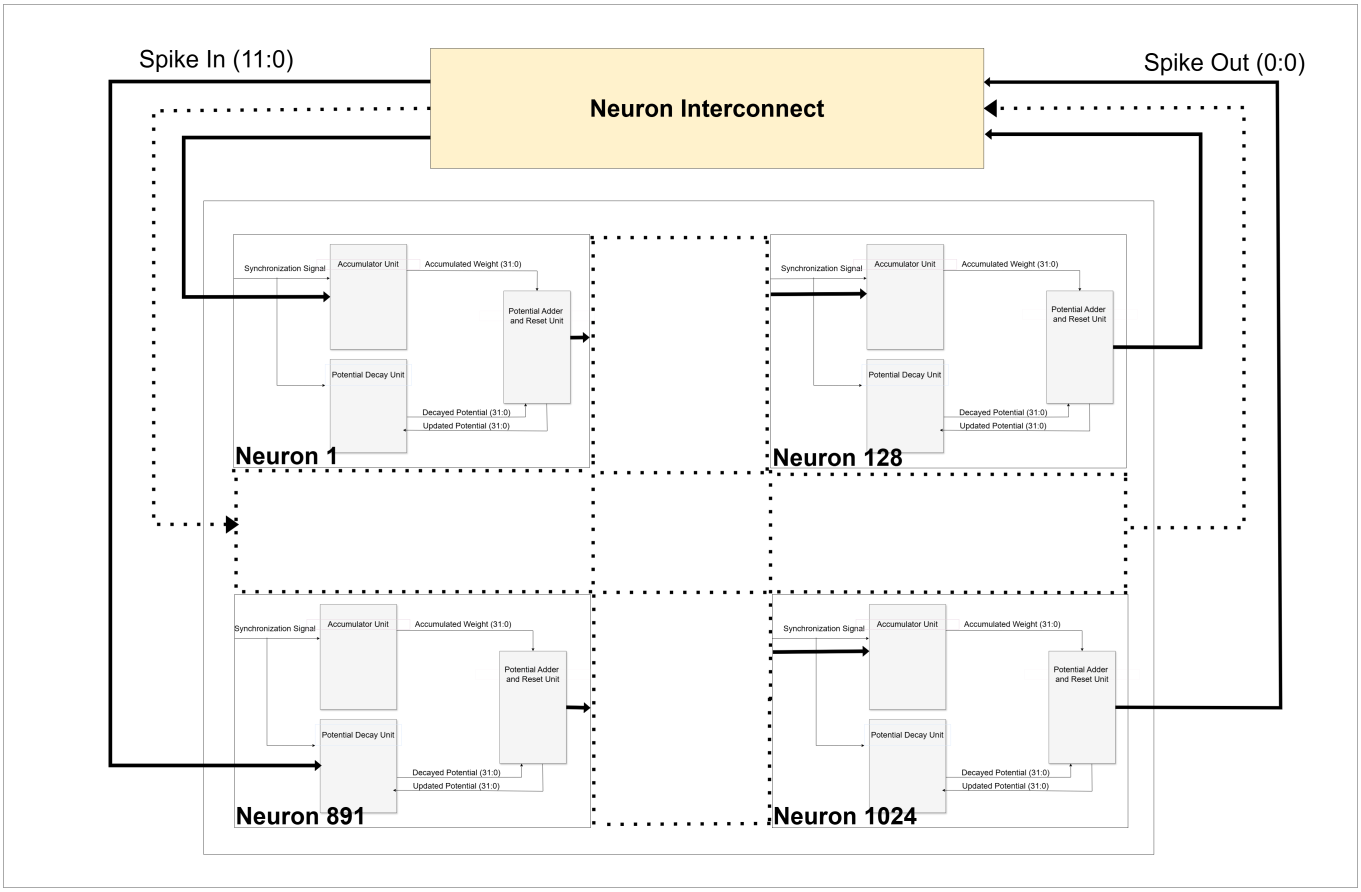

Fig. 6. Top- Level Architecture of the Cerebra-S Chip. A single neuron is represented as a tile. Each neuron has its own accumulator, potential decay, and potential adder and reset unit. The three units together make up a single neuron.

core and the processor core. The processor connects to the core via a shared bus interface. This interface enables data transfer, control signalling and synchronization. The interface acts similarly to the main memory. Custom instructions used are defined below:

- **SWACC**: Store word in accelerator core memory. It is similar to the standard store word instruction. The instruction includes the destination memory unit identifier. This information is used to store information such as the synaptic weight, threshold value and membrane potential for a particular neuron in its corresponding memory.

VI. System Operation

This section describes how each component of the system described previously comes together to execute the operation of a spiking neural network model. As shown in Fig. 3, the SNN to be inferred is loaded onto the core through the processor using the SWACC instruction. Thereafter, all neurons begin operation at algorithmic step 0. At the start of a time step, the accumulator and potential decay units work in parallel. In the accumulator unit, all spikes received from the previous time step are stored in the input buffer. The neurons iterate through their buffers to extract incoming spike information and calculate the weighted input. In parallel, the potential decay unit decays the current potential of the neuron according to the type of the SNN model running. The outputs of these two units are sent to the potential adder and reset unit.

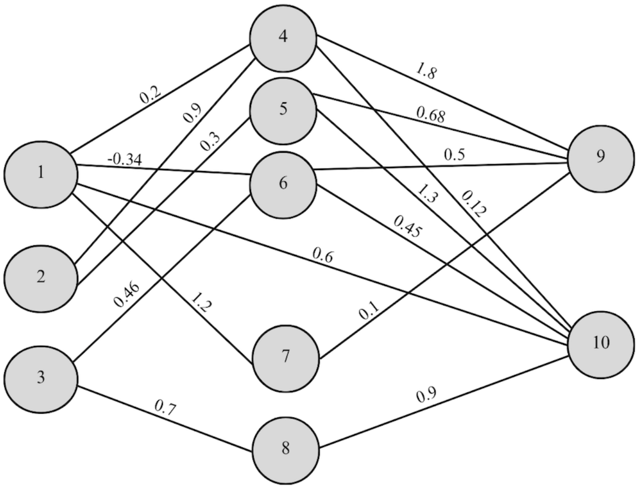

Fig. 7. Sample SNN. An SNN with 10 neurons was used for design experiments.

The potential adder and reset unit performs two operations: adding the two inputs received from its preceding units to calculate the new membrane potential, and determining whether the neuron has entered a firing state. If the neuron has entered a firing state, the membrane potential value is reset and written to memory, and the spike information is sent to the neuron interconnect. If not, the newly calculated potential value is written to the memory in the PDU.

As and when the neuron interconnect receives spikes, it performs a CSR table lookup to identify the destination of the spikes, and creates packets. An arbitration unit helps handle contention for the shared memory in which the CSR table is placed.

These packets are then sent to the corresponding neuron's spike-write buffers. Then, the write spike buffer is copied to the read spike buffer, making the write buffer ready to receive spikes in the new time step. Subsequent operations are performed thereafter, starting from reading all spikes received via the read buffer.

VII. Experiments

The design underwent initial testing to ensure its functionality, which involved running it with two distinct of Spiking Neural Networks (SNNs). The first SNN used was a simple model consisting of only 10 neurons and 16 synaptic connections, as shown in Fig. 7. This basic network served as an initial evaluation to verify the fundamental operational capabilities of the hardware design.

Subsequently, a more complex SNN, trained on the MNIST dataset using snnTorch, was deployed on the hardware. This SNN comprised 994 neurons and roughly 150,000 synaptic connections, representing a significantly larger and more intricate neural network model. All parameters of the trained SNN were taken from a serialized PyTorch state dictionary, which stores the state of the snnTorch model, including weights, biases, and other parameters. The purpose of using this trained SNN was to evaluate the correct functioning of the hardware design under real-world conditions, simulating more sophisticated neural processing tasks, such as image recognition.

An important experiment was conducted to test a slightly different architecture for the network interconnect. Two scenarios were considered. The first involved using a single FIFO buffer at the output of the network interconnect to send spiking information. The second scenario used dedicated FIFO buffers at the output of the network interconnect for each neuron. This experiment modeled an SNN with a gradually increasing number of neurons and was performed using the Static Timing Analysis Tool provided by Quartus software.

VIII. Results and Discussion

SNNs were run on the Cerebra-S core, simulated on an FPGA using a digital logic implementation. The prediction accuracy of an SNN trained on the MNIST dataset, that can be supported on the Cerebra-S core was around 92%. This SNN had 994 neurons to be exact. The key point is that the SNN on the Cerebra-S core achieved significantly better performance in terms of execution time, running nearly five times faster than on a general-purpose CPU hardware. This result supports our initial stance that the true efficiency of the event-driven characteristic of an SNN can be realized when run on specialized hardware. By leveraging the inherent advantages of the Cerebra-S core, which is specifically designed to handle the asynchronous, event-driven nature of SNNs, we can achieve higher performance and efficiency.

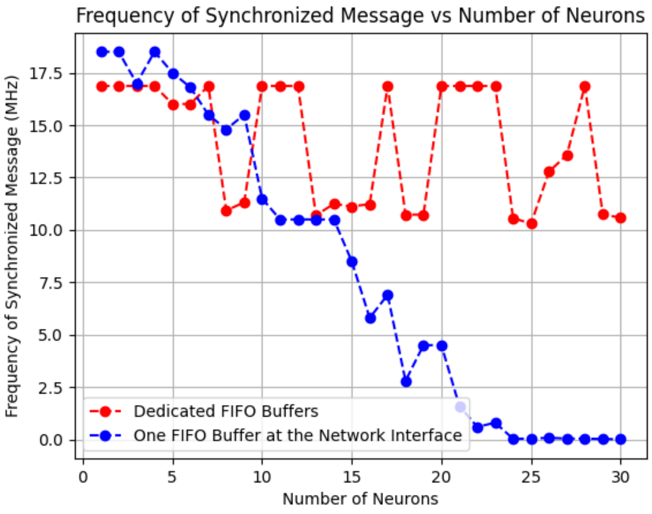

Fig. 8. Change of frequency of operation by having dedicated FIFO buffers and a single FIFO buffer at the output side of the network interconnect.

In both cases—using a single FIFO buffer versus dedicated FIFO buffers at the output of the network interconnect—the results shown in Fig. 8 were observed. The operating frequency declines sharply as the number of neurons increases when a single FIFO buffer is placed at the output of the network interconnect leading to the accumulator units. Conversely, with dedicated FIFO buffers for each accumulator unit, the frequency remains relatively stable, showing only minor variations within a certain range. This stability is attributed to the absence of bottlenecks as the system scales.

Given the high cost associated with large-scale neuromorphic chips like Loihi, utilizing a lower-cost chip for smaller-scale applications, while maintaining similar timing measures, represents a significant advancement in neuromorphic computing. This shift not only makes neuromorphic computing more accessible and cost-effective but also demonstrates that specialized, lower-cost hardware can effectively support high-performance neuromorphic applications, potentially broadening the adoption and impact of neuromorphic technology in various fields.

Table 1 illustrates the key attributes of the design as a summary. The architecture supports SNNs for inferencing. The input to the design should be weight connections among neurons along with information such as neuron model type, threshold values, initial potential values. Once these parame-

TABLE I
DESIGN NUMBERS

Network Type	SNN
Technology	FPGA
Implementation	Digital
Network Topology	No constraints
Number of Neurons	1024
Time for Spike Resolution on MNIST	$250\mu s$

ters are provided the design will set the circuit needed for the particular inferencing and output the spiking resolution results.

The design was simulated on a DE2-115 FPGA by Altera, with the implementation based entirely on digital parameters rather than analog. While the number of connections supported per neuron is limited, there is no restriction on layer-wise neuron connections, allowing for flexible network configurations. The accelerator is capable of supporting spiking neural networks (SNNs) with up to 1,024 neurons, regardless of the specific SNN topology. Notably, the MNIST classification task is completed in just 250 µs on the FPGA, demonstrating exceptional performance. This rapid processing time underscores the accelerator's ability to efficiently handle complex classification tasks in real-time, making it highly suitable for low-latency, performance-critical applications.

The Cerebra-S core neuromorphic chip implementation stands out from other small-scale designs by offering exceptional resource efficiency and flexibility in SNN topology, while also demonstrating excellent scalability. It shows the potential to outperform existing solutions in both performance and cost, while maintaining the adaptability required for future enhancements. This makes it a strong candidate for a variety of neuromorphic computing applications, potentially encouraging broader adoption and innovation in the field.

The ongoing development and testing of our platform will provide further insights to guide the optimization of the Cerebra-S core. The flexibility of the FPGA-based testing platform allows for iterative improvements and refinements based on empirical results. This adaptability ensures that the Cerebra-S core can evolve to meet specific application requirements and address emerging challenges in neuromorphic computing.

IX. CONCLUSION

While numerous neuromorphic architectures exist, there is a scarcity of designs that fully exploit the programming flexibility and platform maturity offered by existing architectures. It is well-established that highly specialized hardware, as seen in current designs, can achieve remarkably high performance levels. However, the authors of this paper posit that a novel neuromorphic architecture, grounded in the general-purpose RISC-V ISA, which provides greater flexibility, combined with the integration of custom accelerator hardware units, has the potential to rival existing designs in terms of both performance and power consumption. Even if it does not precisely match their performance, it will offer a valuable compromise by being more flexible, resource-efficient, and configurable. Further optimizations of the architecture and implementation are planned.

REFERENCES

[1] Bialek, W., et al. (2007). Efficient representation as a design principle for neural coding and computation. arXiv preprint arXiv:0712.4381.

[2] Katare, D., Perino, D., Nurmi, J., Warnier, M., Janssen, M., & Ding, A. Y. (2023). A Survey on Approximate Edge AI for Energy Efficient Autonomous Driving Services. *IEEE Communications Surveys and Tutorials*, **25**(4), 2714–2754. https://doi.org/10.1109/comst.2023.3302474

[3] Bailer-Jones, C., Gupta, R., Singh, H. (2001). An introduction to artificial neural networks. arXiv (Cornell University). https://doi.org/10.48550/arxiv.astro-ph/0102224

[4] Maass, W. (1996). Networks of Spiking Neurons: The Third Generation of Neural Network Models. Electronic Colloquium on Computational Complexity, 3. http://dblp.uni-trier.de/db/journals/eccc/eccc3.html#ECCC-TR96-031

[5] Abbott, L. (1999). Lapicque's introduction of the integrate-and-fire model neuron (1907). Brain Research Bulletin, 50(5–6), 303–304. https://doi.org/10.1016/s0361-9230(99)00161-6

[6] Izhikevich, E. (2003). Simple model of spiking neurons. IEEE Transactions on Neural Networks, 14(6), 1569–1572. https://doi.org/10.1109/tnn.2003.820440

[7] Hodgkin, A. L., Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. Journal of Physiology, 117(4), 500–544. https://doi.org/10.1113/jphysiol.1952.sp004764

[8] FitzHugh, R. (1961). Impulses and Physiological States in Theoretical Models of Nerve Membrane. Biophysical Journal, 1(6), 445–466. https://doi.org/10.1016/s0006-3495(61)86902-6

[9] Benjamin, B. V., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A. R., Bussat, J. M., Alvarez-Icaza, R., Arthur, J. V., Merolla, P. A., Boahen, K. (2014). Neurogrid: A Mixed-Analog-Digital Multichip System for Large-Scale Neural Simulations. Proceedings of the IEEE, 102(5), 699–716. https://doi.org/10.1109/jproc.2014.2313565

[10] Davies, M., Srinivasa, N., Lin, T. H., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., Liao, Y., Lin, C. K., Lines, A., Liu, R., Mathaikutty, D., McCoy, S., Paul, A., Tse, J., Venkataramanan, G., . . . Wang, H. (2018). Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. IEEE MICRO/IEEE Micro, 38(1), 82–99. https://doi.org/10.1109/mm.2018.112130359

[11] Efron, B., Hastie, T., Johnstone, I., Tibshirani, R. (2004). Least angle regression. Annals of Statistics, 32(2). https://doi.org/10.1214/009053604000000067

[12] Beck, A., Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. SIAM Journal on Imaging Sciences, 2(1), 183–202. https://doi.org/10.1137/080716542

[13] Orchard, G., Frady, E. P., Rubin, D. B. D., Sanborn, S., Shrestha, S. B., Sommer, F. T., Davies, M. (2021). Efficient Neuromorphic Signal Processing with Loihi 2. https://doi.org/10.1109/sips52927.2021.00053

[14] Modaresi, F., Guthaus, M., Eshraghian, J. (2023). OpenSpike: An OpenRAM SNN Accelerator. https://doi.org/10.29363/nanoge.neumatdecas.2023.027

[15] Zhang, J., Wu, H., Wei, J., Wei, S., Chen, H. (2019). An Asynchronous Reconfigurable SNN Accelerator With Event-Driven Time Step Update. https://doi.org/10.1109/a-sscc47793.2019.9056903

[16] Rowley, A. G. D., Brenninkmeijer, C., Davidson, S., Fellows, D., Gait, A., Lester, D., Plana, L. A., Rhodes, O., Stokes, A., Furber, S. B. (2019). SpiNNTools: The Execution Engine for the SpiNNaker Platform. Frontiers in Neuroscience, 13. https://doi.org/10.3389/fnins.2019.00231

[17] Chen, G. K., Kumar, R., Sumbul, H. E., Knag, P. C., Krishnamurthy, R. K. (2019). A 4096-Neuron 1M-Synapse 3.8-pJ/SOP Spiking Neural Network With On-Chip STDP Learning and Sparse Weights in 10-nm FinFET CMOS. IEEE Journal of Solid-state Circuits, 54(4), 992–1002. https://doi.org/10.1109/jssc.2018.2884901

[18] Deng, N. L. (2012). The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. IEEE Signal Processing Magazine, 29(6), 141–142. https://doi.org/10.1109/msp.2012.2211477

[19] Zheng, N., Mazumder, P. (2018). A Low-Power Hardware Architecture for On-Line Supervised Learning in Multi-Layer Spiking Neural Networks. https://doi.org/10.1109/iscas.2018.8351516

[20] Bouvier, M., Valentian, A., Mesquida, T., Rummens, F., Reyboz, M., Vianello, E., Beigne, E. (2019). Spiking Neural Networks Hardware Implementations and Challenges. ACM Journal on Emerging Technologies in Computing Systems, 15(2), 1–35. https://doi.org/10.1145/3304103

[21] Camuñas-Mesa, L. A., Linares-Barranco, B., Serrano-Gotarredona, T. (2019). Neuromorphic Spiking Neural Networks and Their Memristor-CMOS Hardware Implementations. Materials, 12(17), 2745. https://doi.org/10.3390/ma12172745

[22] Wu, W. C., Yeh, C. F., White, A. J., Wang, C. T., Yeh, Z. W., Hsieh, C. C., Liu, R. S., Tang, K. T., Lo, C. C. (2021). Integer Quadratic Integrate-and-Fire (IQIF): A Neuron Model for Digital Neuromorphic Systems. https://doi.org/10.1109/aicas51828.2021.9458572

[23] Mohamed Sadek Bouanane, Dalila Cherifi, Elisabetta Chicca23, Lyes Khacef. Impact of spiking neurons leakages and network recurrences on event-based spatio-temporal pattern recognition https://arxiv.org/pdf/2211.07761