TOPICAL REVIEW • OPEN ACCESS

Constructive community race: full-density spiking neural network model drives neuromorphic computing

To cite this article: Johanna Senk *et al* 2026 *Neuromorph. Comput. Eng.* **6** 012001

NEUROMORPHIC
Computing and Engineering

OPEN ACCESS

RECEIVED
5 August 2025

REVISED
16 December 2025

ACCEPTED FOR PUBLICATION
13 January 2026

PUBLISHED
16 February 2026

TOPICAL REVIEW

Constructive community race: full-density spiking neural network model drives neuromorphic computing

Johanna Senk[1,2,*], **Anno C Kurth**[2,15], **Steve Furber**[3], **Tobias Gemmeke**[4], **Bruno Golosio**[5,6], **Arne Heittmann**[7], **James C Knight**[1], **Eric Müller**[8], **Tobias Noll**[9], **Thomas Nowotny**[1], **Gorka Peraza Coppola**[2,9], **Luca Peres**[3], **Oliver Rhodes**[3], **Andrew Rowley**[3], **Johannes Schemmel**[10], **Tim Stadtmann**[4], **Tom Tetzlaff**[2], **Gianmarco Tiddia**[6], **Sacha J van Albada**[2,11], **José Villamar**[2,9] and **Markus Diesmann**[2,12,13,14]

[1] Sussex AI, School of Engineering and Informatics, University of Sussex, Brighton, United Kingdom
[2] Institute for Advanced Simulation (IAS-6), Jülich Research Centre, Jülich, Germany
[3] Department of Computer Science, University of Manchester, Manchester, United Kingdom
[4] Lehrstuhl für Integrierte Digitale Systeme und Schaltungsentwurf (IDS), RWTH Aachen University, Aachen, Germany
[5] Department of Physics, University of Cagliari, Monserrato, Italy
[6] Istituto Nazionale di Fisica Nucleare (INFN), Sezione di Cagliari, Monserrato, Italy
[7] Neuromorphic Software Ecosystems (PGI-15), Jülich Research Centre, Jülich, Germany
[8] Kirchhoff Institute for Physics, Heidelberg University, Heidelberg, Germany
[9] RWTH Aachen University, Aachen, Germany
[10] Institute of Computer Engineering, Heidelberg University, Heidelberg, Germany
[11] Institute of Zoology, University of Cologne, Cologne, Germany
[12] JARA-Institute Brain Structure-Function Relationships (INM-10), Jülich Research Centre, Jülich, Germany
[13] Department of Psychiatry, Psychotherapy and Psychosomatics, School of Medicine, RWTH Aachen University, Aachen, Germany
[14] Department of Physics, Faculty 1, RWTH Aachen University, Aachen, Germany
[15] Current address: Hierarchical Neural Computation RIKEN ECL Research Unit, RIKEN Center for Brain Science, Wako, Japan.
[*] Author to whom any correspondence should be addressed.

E-mail: j.senk@sussex.ac.uk

Keywords: benchmark, neuromorphic computing, computational neuroscience, energy consumption, performance optimization, cortical microcircuit, spiking neural network model

Abstract

The local circuitry of the mammalian brain is a focus of the search for generic computational principles because it is largely conserved across species and modalities. In 2014 a model was proposed representing all neurons and synapses of the stereotypical cortical microcircuit below 1 mm^2 of brain surface. The model reproduces fundamental features of brain activity but its impact remained limited because of its computational demands. For theory and simulation, however, the model was a breakthrough because it is full-scale, therefore free of uncertainties of downscaling, and larger models are less densely connected. This sparked a race in the neuromorphic computing community and the model became a de facto standard benchmark. Within a few years real-time performance was reached and surpassed at significantly reduced energy consumption. We review how the computational challenge was tackled by different simulation technologies and derive guidelines for the next generation of benchmarks and other domains of science.

1. Introduction

Neuroscience increasingly depends on mathematical modeling and simulations to bridge the gap between the properties of single neurons and the function of the entire brain [1]. Thereby, neuroscience moves towards digital twins integrating anatomical and physiological data (e.g. using rat [2, 3] or mouse [4] models) and enabling virtual experiments that cannot be done in organisms. A particular structure of interest in this process is the so called canonical microcircuit [5–7] in the neocortex. The term refers to the local cortical network, the evolutionary youngest part in the mammalian brain [8]. The microcircuit is canonical in the sense that its basic architecture is conserved across species and modalities [9], making it a natural target for the search of generic computational principles in the brain. In 2014 a

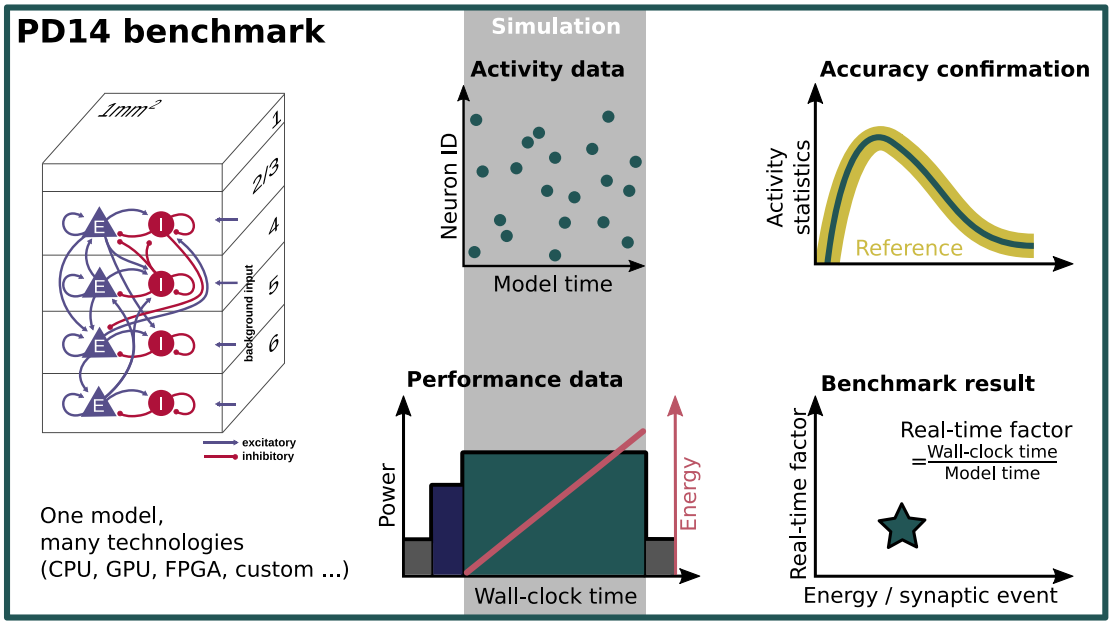

Figure 1. Spiking network model of full circuitry below 1 mm^2 surface area of cerebral cortex serves as benchmark for neuromorphic technologies. The statistics (dark cyan curve) of the simulated activity (dark cyan dots) is compared to reference data (thick yellow curve). Once sufficient accuracy is confirmed, the power measured during the simulation phase (light gray background) yields the consumed energy (dark cyan area corresponds to end point of light red curve, dark blue area indicates network construction phase, dark gray idle phase). The performance benchmark result (dark cyan star) contrasts the real-time factor (defined as the ratio of required wall-clock time and biological time covered by the model) against the required energy (expressed as energy per synaptic event). Reproduced from [10]. CC BY 4.0 and adapted with permission from [11]. The Author 2012. Published by Oxford University Press.

model was proposed representing all the nerve cells (neurons) and all their contact points (synapses) below the surface area of 1 mm^2 of cortex [11]; that is a network size of on the order of one hundred thousand neurons and one billion synapses. Since the model is full-scale, it lifts all uncertainties about how downscaling affects network activity present in earlier models [12]. Downscaling here refers to the concept of reducing the number of neurons and synapses in a model for computational tractability, while attempting to preserve the dynamics and keeping other structural characteristics like the probability of two neurons forming a connection invariant. It does not refer to representing neurons and synapses at a finer level of description, like the level of electrical compartments or the molecular level. The model represents four cortical layers by populations of excitatory and inhibitory leaky integrate-and-fire model neurons (network sketch in figure 1). Due to the simplicity of its components and the limited data available at the time, the explanatory scope of the model is limited, even though it reproduces basic features of cortical activity such as asynchronous and irregular spiking at biologically plausible population-specific rates. Initially the simulation results were confirmed by independent researchers using different simulation codes [13, 14]. The model and variations thereof have been used for various neuroscientific investigations (e.g. attention [15, 16], orientation selectivity [17], inhibitory neuron types [18–20], network connectivity [21, 22], building block of multi-area models [23, 24], forward-modeling of extracellular potentials [25, 26]) and as a test bed for theoretical methods (e.g. mean-field and linear response theory [27, 28], population density models [29, 30], stochastic population equations [31, 32]). Nevertheless, when it was first proposed, run times of several minutes were required for the simulation of 1 s of biological time, limiting the practical use of the model. A recent review [33] analyses the impact of the model on the fields of computational neuroscience and neuromorphic computing.

The opportunity to overcome a barrier in terms of network size, and the availability of a reproducible model combined with the computational challenge of bringing down simulation time of a microscopically parallel problem sparked a concrete and quantifiable race in the computational neuroscience and neuromorphic computing communities for ever faster and more energy-efficient simulation. In particular the frequent updates (at least every 0.1 ms of model-time in time-driven simulations) as well as the communication and delivery of a large number of events (also called spikes) during state propagation call for innovative solutions in hardware and software, beyond conventional approaches in high-performance computing (HPC). A few years later, the model has been simulated on a range of

platforms including a large-scale neuromorphic system (SpiNNaker [10, 34]), many-core CPU systems (NEST [10, 35]) GPUs (GeNN [36, 37], NEST GPU [38, 39] and FPGAs (CsNN [40], neuroAIx [41]). In this time span, real-time performance was reached and simulation time and energy per synaptic event dropped by two orders of magnitude. The circuit can now be simulated an order of magnitude faster than real time. The role of the model as a benchmark was never declared by any organization but retrospectively emerged over the years. The studies reproducing the work were done independently by different laboratories, in retrospect becoming evident as a race. It is therefore time to gather the results distributed in the literature, contrast the findings and identify the advantages and bottlenecks of different approaches. From this we derive limitations of present technology and requirements for future neuromorphic architectures. From the perspective of computational science we discuss what made the PD14 model successful in becoming a de facto standard benchmark for our community and what obstacles the authors faced using the benchmark. Finally, we discuss next challenges for the simulation of large-scale neuronal networks and characterize desired properties of next generation benchmark models.

2. Results

This work covers technologies designed for large-scale spiking neuronal network simulations. These technologies have different algorithmic approaches and include software-based simulators exploiting conventional hardware and custom neuromorphic platforms. Their performance is assessed on the example of the PD14 model [11] in terms of time to solution and energy to solution. At this point, we take it for granted that the model is simulated at sufficient accuracy, meaning that the statistics of the simulated spike data are compatible with reference data (figure 1, upper panels, for details see section 4.2.3). Timestamps in the simulation scripts enable different execution phases to be identified and related directly to temporally resolved power measurements. (figure 1 lower panels). Each simulation experiment begins with a network initialization phase during which the network is constructed either directly on the simulation system or on a host system and then transferred. The beginning of the subsequent state-propagation phase should be considered a warm-up time because start-up transients may occur due to initial conditions. Even though network initialization and warm-up time can consume a substantial amount of time and energy, here we only focus on the state-propagation phase with stationary network dynamics. We define the performance metrics as follows:

Time to solution. The *real-time factor* q_{RTF} is defined as the quotient of wall-clock time T_{wall} (which is also known as real time) and model time T_{model} (which is the duration by which the state of the model is advanced in time):

$$q_{\mathrm{RTF}} = \frac{T_{\mathrm{wall}}}{T_{\mathrm{model}}} \tag{1}$$

If the real-time factor is larger than one, the simulation runs slower than wall-clock time. The reciprocal of q_{RTF} is a measure for the speed of the calculation; the higher q_{RTF}^{-1} the faster the computing system completes the task. Ten times faster than real time could be expressed by $q_{\mathrm{RTF}}^{-1} = 10$.

The speed of the calculation should not be confused with the concept of speedup. Speedup [42, 43] refers to a ratio of two wall-clock times, namely the wall-clock time required to solve a problem serially and the wall-clock time needed to solve the same problem in parallel (e.g. by distributing the work across multiple threads or processes): $q_{\mathrm{speedup}} = T_{\mathrm{wall,serial}}/T_{\mathrm{wall,parallel}}$.

Energy to solution. The *energy per synaptic event* E_{syn} is defined as the integrated power P (i.e. wall socket power including communication infrastructure) over the state-propagation phase (i.e. wall-clock time T_{wall}) divided by the total number of synaptic activations in the time interval T_{model}:

$$E_{\mathrm{syn}} = \frac{\int_0^{T_{\mathrm{wall}}} P(t)\,\mathrm{d}t}{T_{\mathrm{model}} \cdot \sum_\alpha N_\alpha \cdot K_{\mathrm{out},\alpha} \cdot \nu_\alpha} \tag{2}$$

with the neuron number N_α, the average number of outgoing connections $K_{\mathrm{out},\alpha}$ and the spike rate ν_α per neuron of population $\alpha \in \{\mathrm{L2/3E}, \mathrm{L2/3I}, \mathrm{L4E}, \mathrm{L4I}, \mathrm{L5E}, \mathrm{L5I}, \mathrm{L6E}, \mathrm{L6I}\}$ in the PD14 model. In a simulation of ten seconds of model time our neuronal network produces approximately 2.47 million spikes leading to approximately 9.6 billion synaptic events.

The PD14 has become a de facto standard benchmark. In the beginning, an executable model description was only available in the original simulation language (SLI) of the NEST code [44] and this has been included in all versions of NEST since v2.4.0 (released in 2015). But, thanks to progress in the community and large-scale projects, executable descriptions abstracted from a particular simulation engine became available. Nevertheless, a performance result is only meaningful if a benchmark reaches the same accuracy of the solution as the reference data. This is because generally, a lower accuracy can be reached with less computational effort. The neuroscientific purpose of the PD14 model is to reproduce certain characteristics of the activity of the respective biological neuronal network. Therefore, van Albada *et al* [10] define a set of measures a benchmark of PD14 needs to fulfill. A later study [45] finds that the specified correlation measure is good enough to expose corrupted activity, but that model-specific correlation structure is only exhibited at considerably longer simulation times. This study also shows that the inhomogeneity of the network structure hides the distribution of synaptic weights: the correlation measures remain identical if all weights are collapsed to the mean value of the corresponding normal distribution. Retrospectively this is apparent as a mean-field theory [27] can quantitatively reproduce the first and second-order statistics of the population activity. Thus, the characterization of the solution does not require the simulation of a spiking neuronal network with an individual representation of all neurons and synapses. The synaptic weights can be replaced by a single value per neuron population [45] and connections can be generated on-the-fly with a generator for pseudo-random numbers [40]. This dramatically reduces the amount of memory a simulation engine needs to access. Nevertheless, so far only direct simulation delivers the full distributions of the statistical measures. It remains to the good will of the scientists to carry out the benchmark with simulation engines that are in principle capable of representing synaptic weights as dynamical variables as required for plasticity and learning. Spike-based plasticity rules directly interact with the correlation structure. If constraints of the engine at hand do not allow a representation at this level of resolution, the results may still be interesting but the conditions need to be declared.

The community gain of the PD14 race is attributed to various factors and a complete disentanglement is outside the scope of this manuscript. General-purpose approaches (i.e. simulation software designed to support a plethora of different models and to run on different conventional CPU and GPU based hardware systems such as NEST [44] and GeNN [46]) are compared to dedicated approaches (i.e, neuromorphic hardware such as SpiNNaker [47] or software-hardware-model co-designs using FPGAs). Apart from model-specific simulator advancements, some progress in case of NEST and GeNN is due to later software versions and hardware generations which are continuously optimized independently of the PD14 model. However, the proportion of the contributions is unclear as both are replaced in new data points. For SpiNNaker instead, we see an improvement of the software stack only allowing better use of the existing hardware. Generally speaking, the observed difficulty of running old software on new hardware hampers a systematic comparison. For example, despite great efforts to adhere to programming language and coding standards, in practice it is a challenge to run NEST versions released only a few years ago on recent hardware. Further research may analyze why this is so difficult and what developers of a simulation software stack and developers of benchmark models can learn to make simulation codes and models more robust.

Table 1 and figure 2 summarize the progress of the community in increasing the simulation speed and reducing the energy consumption of the PD14 model. The employed computing platforms are illustrated in figure 3. The historical account (figure 2(a)) of the real-time factor starts with a number of initial data points in 2018. None of the studies came close to real time. The SpiNNaker system is specified for real-time performance, but shortcomings of the mapping of neurons to hardware exposed by the study required a temporary slow down. The CPU code, already mature at the time, showed better performance but at the highest energy consumption. GPU code for spiking neuronal networks was just emerging and immediately took the top position in speed. A year later the SpiNNaker system reached its specification and simulated the model in real time. In the following three years progress was harder and breaking the real-time barrier became a psychologically important target. Finally a new GPU code came very close and the more mature initial GPU code passed the barrier. Shortly afterwards CPU code outperformed the GPU results and reached out into the real-time regime on a single state-of-the-art commodity compute node. Nevertheless this was dwarfed in the same year by an FPGA-based neuromorphic system. A year later the rapid progress in GPU hardware enabled GPU-based systems to take the lead again. But in the same year a new dedicated FPGA-based neuromorphic system made a jump to a simulation speed twenty times faster than real time. This achievement is substantial because present CPU code requires a longer duration just for the communication between compute nodes.

In the real-time factor alone (figure 2(a)) the progress appears steep but continuous with the exception of a major jump due to the FPGA technology. The vertical divide at real time may indicate the

Table 1. Performance data of considered studies. A code name disambiguates the studies and refers to the bibliography (number in brackets). The real-time factor q_{RTF} (equation (1)) and the energy per synaptic event E_{syn} (equation (2)) are the performance results obtained with the simulation technologies characterized by simulator name, number of nodes, system specification, and the process node as the industrial specification of the chip technology. The external drive indicates the background input used (DC or Poisson).

Study		q_{RTF}	E_{syn} (μJ)	Simulator	#Nodes	System	Process node (nm)	External drive
vAl+18a	[10]	2.465	9.941	NEST CPU	12	2 Intel Xeon E52680v3	22	DC
vAl+18b	[10]	4.584	5.816	NEST CPU	3	2 Intel Xeon E52680v3	22	DC
vAl+18c	[10]	20	4.4	SpiNNaker 1	6	48 × 18 × ARM-968	130	DC
KN18	[36]	1.838	0.47	GeNN	1	Tesla V100	12	Poisson
Rho+19a	[34]	1	0.601	SpiNNaker 1	12	48 × 18 × ARM-968	130	DC
Rho+19b	[34]	1	0.628	SpiNNaker 1	12	48 × 18 × ARM-968	130	Poisson
Gol+21	[38]	1.055	0.25	NEST GPU	1	RTX 2080 Ti	12	Poisson
Kni+21	[37]	0.7	—	GeNN	1	Titan RTX	12	Poisson
Hei+22	[40]	0.25	0.284	CsNN	345	IBM INC-3000	28	Poisson
Kur+22a	[35]	0.53	0.48	NEST CPU	2	2 AMD EPYC Rome 7702	7	DC
Kur+22b	[35]	0.67	0.33	NEST CPU	1	2 AMD EPYC Rome 7702	7	DC
Gol+23a	[39]	0.386	0.104	NEST GPU	1	RTX 4090	5	DC
Gol+23b	[39]	0.272	0.074	GeNN	1	RTX 4090	5	Poisson
Kau+23	[41]	0.05	0.048	neuroAIx	35	NetFPGA SUME	28	DC

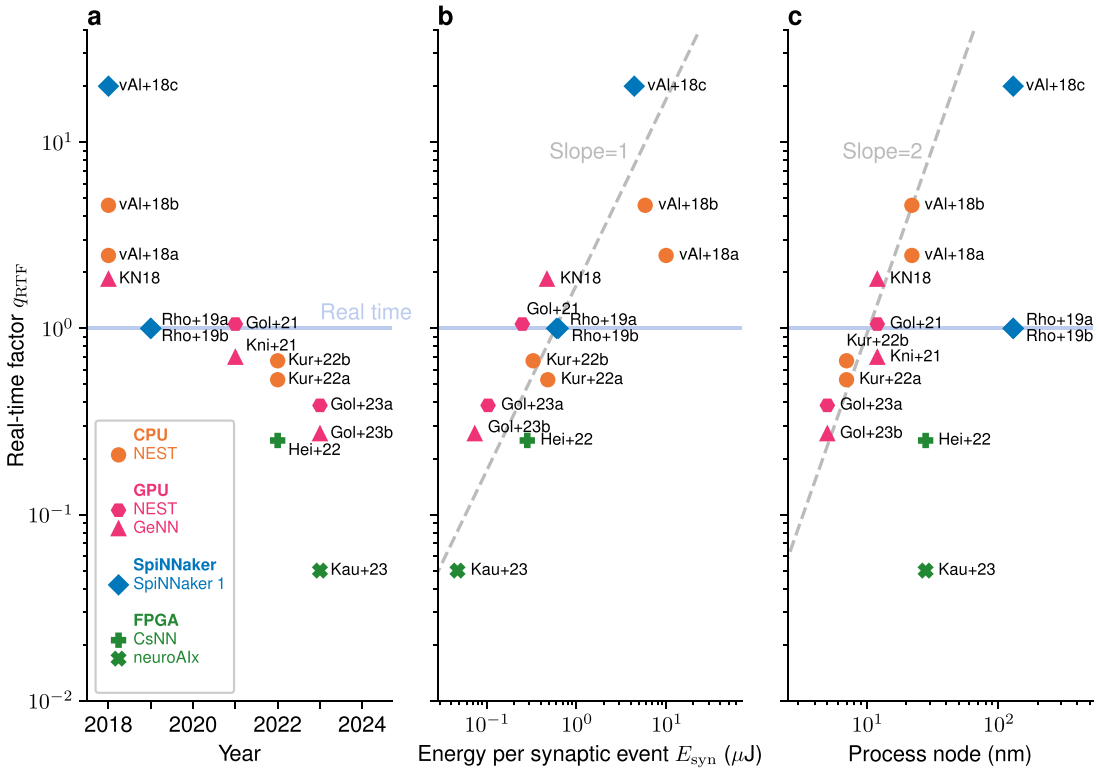

Figure 2. Progress of the community in reduction of time to solution and energy consumption for the PD14 model. Colors group hardware architectures and shapes indicate algorithmic approach (legend). Abbreviations in panels further disambiguate individual studies. (a) Ratio between time passed on wall-clock and stretch of time covered by the model (real-time factor) versus the year of publication in semi logarithmic representation. (b) Real-time factor as a function of energy per synaptic event in double logarithmic representation. Dashed line from fit through all data points with a slope of one. (c) Real-time factor versus process node in double logarithmic representation. Dashed line from fit through CPU and GPU data points with a slope of two. Citations of studies and values are given in table 1.

psychologically important barrier motivating further publications. Figure 2(b) shows the time required to reach the solution as a function of the energy required by the different systems to complete the task. The double-logarithmic representation reveals an overall power-law decline. While energy consumption dropped by two orders of magnitude, the simulations became two orders of magnitude faster. The graph does not segregate technologies but data points are rather organized by the time of publication.

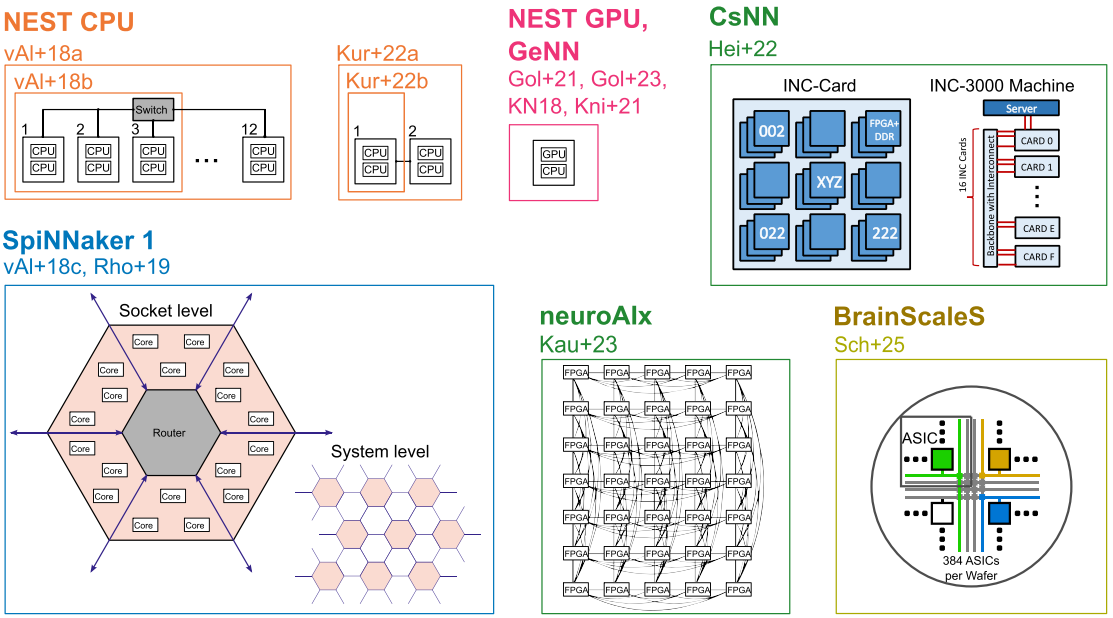

Figure 3. Sketches of computing platforms. Diagrams illustrate the major computational building blocks and the communication architecture of a particular system required during the simulation phase. Compare with columns '# Nodes' and 'System' in table 1. Same color scheme as in figure 2. **NEST CPU** uses compute nodes with two CPU sockets each. The first study (*vAl+18a/b*) uses a compute cluster with Infiniband switch. The second study uses two point-to-point connected nodes (*Kur+22a/b*). **NEST GPU** and **GeNN** simulations run on single GPUs. **SpiNNaker** uses a router and 18 low-power cores per socket, with sockets embedded in a mesh routing fabric with 6 links per socket, enabling an extensible system facilitating multicast communication between cores. The presented studies using SpiNNaker both used this architecture, with variations in how the cores were configured to perform neural processing, and small variations in the overall size of the systems. **CsNN** uses the IBM INC-3000 system which consists of 16 PCB's (INC-cards), each hosting 27 reconfigurable SoC nodes. The whole INC-3000 system consists of 432 FPGA nodes which are connected by a three-dimensional mesh. **neuroAIx** uses a cluster of 35 FPGAs connected with overlayed long hop topologies as an extension to mesh-like network topologies. **BrainScaleS** uses a circuit-switched on-wafer network for spike communication between the 384 ASICs per wafer.

Figure 2(c) shows the real-time factor over process node. Even though the communication hardware and other system-specific components are not resolved, this representation reveals a clearer trend and at the same time a separation of dedicated hardware architectures from the conventional CPU and GPU architectures. Let us assume that while reducing process node, manufacturers increase the number of transistors and keep the chip area constant, thereby also conserving the power consumption; keeping the power density constant is known as Dennard scaling [48], though such scaling has been increasingly inapplicable to digital processing systems beyond the 65 nm technology node [49]. If neuromorphic code keeps up with the increased parallelization [50] enabled by the growing number of transistors, the real-time factor should drop linearly with the number of transistors. The latter, however depends quadratically on process node. Following this argument we expect in the double-logarithmic representation of figure 2(c) the real-time factor to decline with a slope of two in dependence of process node. This holds well for the conventional CPU and GPU technologies even though the impact of the whole system used is disregarded in this argument. The argument also explains the slope of one in the dependence of real-time factor on energy per synaptic event observed in figure 2(b). If a technology completes the task in half the time, only half of the energy is required under the assumption of a constant power density. The dedicated architectures based on the SpiNNaker chip and FPGAs deviate from this trend. The expectation on a dedicated hardware is that for comparable process node it outperforms code executed on a conventional computing architecture. Initially the SpiNNaker system (*vAl+18c*) suffered from a suboptimal mapping of PD14 onto the hardware. But already at this time performance was better than the one predicted for CPU and GPU based systems at this large process node. Conversely, starting from the *vAl+18c* data point and projecting performance with a slope of two down to the process node of the conventional competitors at the time (*vAl+18a/b*) also exposes the superior performance of the design. With improved software a dramatic step down in real-time factor became possible (*Rho+19*) while process node remained unchanged. The architecture is competitive with conventional architectures using an order of magnitude smaller process node. Furthermore, a prediction starting at (*Rho+19*) with a slope of two down to smaller process nodes outperforms all recent CPU and GPU systems. The same is true for the FPGA-based systems. They already started out in the range of the best real-time factors

(*Hei+22*). A system of generic FPGA boards with customized communication and selective optimization in hardware description language (HDL) yielded a step down in real-time factor by an order of magnitude while the process node remained the same (*Kau+23*). With the optimized software the SpiNNaker and FPGA based systems fall on the same projection for decreasing process node. Following the trend of the conventional architectures for further reductions in process node it seems difficult to reach $q_{\mathrm{RTF}} = 0.1$ without major progress in algorithms. Applying the same projection with a slope of two to the SpiNNaker and FPGA architectures, however, $q_{\mathrm{RTF}} = 0.01$ seems indeed plausible.

The reduction by two orders of magnitude in simulation time since the beginning of the race enables the community to investigate plasticity and system-level learning in a full scale model where neurons are supplied with the natural number of synapses. This increase in simulation speed is of practical relevance for the number of simulation experiments a researcher can do per day, because the biological processes unfold over minutes and hours. Trivial parallelization is often not a cure here, because only the result of a simulation gives the insight and inspiration for the next one. On a more societal level, the energy costs and respective CO_2 footprint of simulating the PD14 model are reduced.

Unless a neuromorphic system needs to interact with the real world in a closed loop [51], it is worth to strive for even faster and more energy efficient simulations. For the use as a neuroscience research tool, which requires a high degree of flexibility, 100 × faster than real-time is desirable and seems to be compatible with present day conventional semiconductor technology. This was also concluded in a previous pilot project (Advanced Computing Architectures (ACAs), www.fz-juelich.de/en/aca; e.g. [52, 53]). For the PD14 this may constitute the next milestone for the community.

For conventional CPU and GPU technology, research is ongoing on improving the codes for multi-node systems. But for 100 × the time available for an update step of 0.1 ms is already shorter than the latency of an older Infiniband interconnect (5 μs), and close to the latency of a modern one (0.5 μs). The progress on more densely integrated single nodes, however, is steady and the success of strong-scaling on multi-GPU nodes is largely unexplored. In this respect, the booster nodes of the upcoming JUPITER [54] exascale computer at Juelich equipped with four NVIDIA GH200 Grace Hopper Superchips are an exciting next target for CPU as well as GPU code. However, GPU codes typically need to launch multiple kernels during each simulation timestep to ensure correct synchronization and each launch incurs a fixed latency (a minimum of 2.5 μs was measured on an NVIDIA DGX-1 system [55]). Therefore, as for example GeNN launches four kernels per timestep when simulating PD14, this launch latency alone limits q_{RTF} to around 0.1.

SpiNNaker 1 uses a relatively old semiconductor technology, 130 nm bulk CMOS, which should be taken into consideration when reviewing energy performance, but further highlights the importance of overall system architecture on optimizing execution speed of spiking neural network simulations. The successor chip, SpiNNaker 2 [56], has been developed using a more up-to-date technology (22 nm FDSOI) and is expected to deliver around a 10 × improvement in functional density and energy-efficiency. As established above, extrapolating the FPGA architecture with a slope of two shows that 100 × is in reach of reasonable process node sizes. Critical is of course a corresponding communication architecture [52].

There is a convergence of general purpose computational architectures and dedicated neuromorphic hardware in the sense that in conventional systems the parallelism becomes ever more fine grained. For the PD14 model for example, each of the 128 cores of the compute nodes used here [35] is just responsible for some 700 neurons. It remains to be seen how far this abundance of hardware can be exploited for further strong scaling. The structures in present day chips are orders of magnitude smaller than the respective structures in biology and the switching times are accordingly shorter. Therefore electronics can profit from techniques like busses and multiplexing to compensate for the lack of cables. Nevertheless, the optimal system is not necessarily one where one circuit takes care of a single neuron. The BrainScaleS architecture (figure 3 and section 4.1.7) exploits the microscopically parallel nature of neuronal networks to the extreme. Each neuron is emulated using its own analogue circuit and achieves $q_{\mathrm{RTF}} = 0.0001$ by mapping the full network to a single wafer-size chip. At present only a downscaled version of PD14 fits onto the system [57], therefore the data are not included in figure 2. Progress in the combination of analog and digital hardware as well as mapping software may change this.

Projecting the linear dependence of the FPGA architecture in figure 2(b) to $q_{\mathrm{RTF}} = 0.01$ would lead to an energy consumption per synaptic event of approximately 10 nJ. BrainScaleS already reaches the same order of magnitude (12 nJ) for the downscaled version of PD14. This is three orders of magnitude lower than the value found in the initial studies in 2018, but still five to six orders away from the amount required by nature (19—760 fJ, see [10]).

Since the PD14 model was initially developed with a focus on neuroscience, there were uncertainties about model and simulation parameters to be used for benchmarks. For example, whether to use DC

or Poisson input as external drive may affect the performance and is therefore indicated in table 1. We identify the choice of initial conditions, treatment of a warm-up time, simulation duration, repeated simulations, and spike recording as further settings that are inconsistently handled across the studies involved and influence comparability (for details see section 4.1). In fact, the effect of certain model and simulation parameters on network activity and compute requirements was only exposed one at a time as byproduct of the unstructured community effort. To pool the gained insights and provide the missing instructions we here compile a Benchmarking recipe (section 4.2) with recommendations for future benchmark simulations including a model reference with detailed simulator-independent model documentation.

Most of the studies used different, hand-crafted model implementations suited to their simulators. Developing and testing these implementations is a laborious and error-prone task. With the benchmarking recipe proposed here, we aim to facilitate obtaining and porting the reference implementation and conducting the benchmarking experiments. Additionally, they highlight a number of snares impeding the communication between researchers and increasing the difficulty of comparing the results. To further simplify developing and using such benchmarks, a common language similar to the simulator-independent language PyNN [58] with backends for different systems as well as shared definitions for the measured quantities and a unified verification and validation workflow with distinguished reference data could be instrumental for the community. Another challenge lies in unifying benchmarking tools for running simulations and comparing results, addressing inherent but often overlooked issues in performance benchmarking [59].

3. Discussion

The race is ongoing but the goals to achieve a simulation speed one hundred times faster than real time and to rationalize the comparison of different architectures comes into sight. As the number of computational elements converges towards the number of neurons it seems as if the von-Neumann bottleneck widens even for conventional computing systems.

Finding challenging benchmarks remains relevant for the validation of dedicated neuromorphic computing systems. The construction of such a system is justified if it outperforms in some measure the conventional mass produced CPU and GPU based systems equipped with the best possible software. Fundamental limitations showing the superiority of neuromorphic approaches in strong-scaling or weak-scaling scenarios would give confidence and guidance for technology development.

The unfolding race highlights conditions enabling and motivating such a community wide benchmarking effort but also exposes obstacles in interpreting the benchmark definitions and initial weakness in the metrics for the verification of the simulation results.

Unlike other benchmarks in the area of brain-inspired computing, our metrics for comparing the performance of different approaches are not addressing the accuracy achieved in solving a particular task. It is already part of the verification of a certain system and a precondition that a minimal accuracy is met. The actual comparison is on the time it takes to reach the solution (at required accuracy) and the energy consumed.

The proposed benchmarking recipe (section 4.2) is a step towards more consistent benchmark runs. Porting the model to a new hardware or software environment, however, remains a challenging task due to the inherent complexity of benchmarks [59] that requires in-depth understanding of the model structure, dynamics, and the respective computing platform. Together with the model reference we publish the recipe as a living document such that the guidelines can be improved or extended if further uncertainties emerge. We also include the code to generate the performance figure (figure 2) such that new data can be included.

We believe that the insights on the conditions for the success of the study and the origins of difficulties are not specific to our field computational neuroscience but generic for benchmarking in computational science. The benchmarking recipe condenses this knowledge. In short, it highlights the importance of a well-documented model description with a reference implementation and simulation instructions.

Network initialization can require substantial amounts of resources, especially when exploring various parameters. The authors believe that the software stacks of the studies reviewed here are not yet mature enough for meaningful comparisons. Therefore our investigation concentrates on the phase of state propagation. The longer the stretch of biological time covered by the model, the more is the complete time to solution dominated by the state propagation phase. Nevertheless, with maturing software stacks future work should also include the time and energy required for network construction.

Since the PD14 model's conception, new data sets have emerged, expanding our neuroscientific understanding of cortical microcircuits. While model revisions could incorporate additional insights (e.g. target-type specific innervation preferences depending on whether a neuron is excitatory or inhibitory [60], distinct interneuron types and short-term plasticity [20], and point-neuron vs. multi-compartment neurons [4]), they are not discussed here as they would not significantly alter the computational load. Still we argue for a transparent and systematic tracking of model revisions as done with the PD14 model within the NEST code over the past ten years and from now on in a self-contained model reference (section 4.2.1) repository.

Besides, the complexity of the brain calls for complementary benchmark models with different biological detail and computational demands such that large-scale neuromorphic hardware systems will not be optimized for only a single model type. The benchmark considered here represents the smallest building block of the part of the mammalian brain responsible for higher brain functions. The strong-scaling scenario has exposed the characteristics of the hardware systems and their limits. It is now time to bring the weak-scaling scenario into perspective. For our research field this means capturing larger parts of the brain at the same level of resolution. One way of achieving this is not to increase the surface area of a model of a single cortical area, but to consider interconnected patches of surfaces in several areas. This is neuroscientifically of interest because it relates the activity in local circuits to the global dynamics of the brain. The workflows for porting a neuronal network between computing platforms are now established. This should make it easier to carry out and compare benchmarks for other, more advanced, networks models. An example is a multi-area model of the visual system of macaque monkey using an adapted version of the PD14 model for each of the 32 areas represented [61]. This model can already be simulated with NEST CPU [23], NEST GPU [62], and GeNN [63] but speed is far from real time. Another model adaptation incorporating distance-dependent connectivity has not been simulated on other systems than NEST CPU yet [64].

More detailed neuron models and slow processes like long-term plasticity, system-level learning, and development increase the computational costs of any model. Thus, at the scale of PD14 we will face longer simulation times than predicted by the data shown here, and plastic processes still cannot be studied in larger models even with state-of-the-art simulation engines. Another challenge therefore consists in devising and maturing additional representative benchmark models which require different metrics for evaluating the accuracy. It is here where benchmarks concentrating on time to solution and energy to solution converge with benchmarks concentrating on the performance in fulfilling a function [65].

Much like the supercomputing community has learned that a useful benchmarking suite needs to be multi-disciplinary for a multi-purpose system (see for example [54]). At the same time not all benchmarks are of relevance for all systems. A chip for edge computing has a different area of application than a chip designed to simulate a cortical network. The field of neuromorphic computing needs to map out its application domain and arrive at a consistent set of benchmarks [65]. The optimal application area of a given approach can then be characterized as a particular territory on this map.

Computational neuroscientists and simulation system developers symbiotically benefit from diversity in both the types of neuronal network models studied and the simulation technologies developed. On the road towards understanding the brain and at the same time making use of the gathered knowledge to advance technology, we realized the importance of points of convergence for different disciplines to come together and learn from each other through rigorously defined and executed performance benchmarks. Accepting the effort and extra complexity of using a real-world benchmark instead of a synthetic benchmark, sometimes also called mini-app, has the advantages of reducing the danger that important characteristics are overlooked and that the relevance of the benchmark is not in question. We hope that the experiences shared here advances computational science beyond the neuroscience domain.

4. Methods

4.1. Technology and study details

Here we elaborate on the different technologies and respective studies contributing to the race. We start with a general introduction to each system with main references, links to website and source code if available, and design goals. Regarding implementation languages we distinguish between the user interface and low-level components. The hardware is described as either the conventional target processor or a custom architecture; for an overview refer to figure 3. We also specify the parallelization model for software and hardware if applicable. A characterization of the employed spiking neural network simulation strategy includes the sequence of steps (e.g. code generation, compilation, network construction, and state propagation), but also details on algorithms and numerics. Each section concludes with details

on power measurement. These brief simulator profiles are by no means complete as some approaches are the results of decades of development and user experience. Features only touched upon may be the support for certain neuron and synapse models or plasticity mechanisms, or scalability with respect to network size. Our focus is on aspects that differentiate PD14 performance results on different systems, assuming that simulations achieve a sufficient accuracy. Where applicable, we highlight what we have learned from a computational point of view by contributing to the race. Following the generic introductory paragraphs, we provide details on the particular PD14 results, i.e. the data points depicted in figure 2, complementary to the performance data and system specifications in table 1. For the complete study descriptions, we refer to the original publications.

With the maturation of the model specification, the studies employ two different sets of initial conditions (section 4.2.2.2) resulting in statistically identical network activity but differing in the transient activity: the *original initial conditions* used in [11] and *amended initial conditions* suggested in [34].

4.1.1. NEST CPU

NEST [44] is a simulation software for spiking neural networks exploiting conventional CPU-based HPC infrastructure. NEST is written in C++, uses OpenMP [66] and MPI [67] for parallelization, and has a Python user interface PyNEST [68, 69] (website: www.nest-simulator.org, source code: https://github.com/nest/nest-simulator). The code represents numerical values in double-precision floating-point format, and solves sub-threshold dynamics with exact integration [70]. The network construction and simulation phase are natively executed in direct succession and both parallelized across all resources of the same system. In the studies considered here, power of NEST simulations is measured with rack Power Distribution Units (PDUs), remotely reading the active power of the compute notes under load approximately once per second.

The race encouraged NEST developers to exploit the microscopic parallelism of the neuronal dynamics on many-core CPUs, including minimizing the number of barriers and employing thread-parallel memory allocation [71].

4.1.1.1. vAl+18 [10]

The study compares the accuracy and performance of NEST and SpiNNaker simulations by executing a custom PyNN [58] code of the PD14 model with NEST 2.8 [72] using the original initial conditions. Simulations cover a duration of $T_{\text{model}} = 10\,\text{s}$ and record spikes. The subsequent analysis considers the complete simulation phase and compares grid-based simulations with precise spike timing [73, 74] as a control of accuracy. In search for the optimal hardware configuration, the study carries out a strong-scaling experiment on an HPC cluster with 32 compute nodes (two CPUs per node). The system achieves the minimum q_{RTF} (*vAl+18a*) on twelve compute nodes, and the minimum E_{syn} (*vAl+18b*) including an estimate for the network switch on three nodes.

4.1.1.2. Kur+22 [35]

This work adapts the publicly available PyNEST [68] code of the PD14 model for benchmarking and uses the amended initial conditions. Runs of NEST 2.14.1 [75] cover a duration of $T_{\text{model}} = 10\,\text{s}$ following a warm-up time of 0.1 s, and do not record neuronal activity. A strong-scaling experiment scans a system of two HPC nodes with two CPUs per node hosting 64 cores each for optimal performance. The simulations reach the minimum q_{RTF} (*Kur+22a*) on two point-to-point connected fully utilized nodes and the minimum E_{syn} (*Kur+22b*) already on one fully utilized compute node.

4.1.2. NEST GPU

NEST GPU [38, 62] is an open-source simulator designed to simulate large-scale spiking neuron networks. Originally developed under the name NeuronGPU, it is now part of the NEST Initiative e.V. (source code: https://github.com/nest/nest-gpu). The simulator is written in CUDA-C++ and supports multi-GPU simulations through MPI. Moreover, it provides a Python interface, which closely mirrors that of the original NEST CPU code, allowing researchers to define neurons, connections, and synapse properties using similar commands.

The PD14 model size is too small to take advantage of the massive parallelism available at modern data centers, which offer researchers thousands of GPUs simultaneously. Yet, this model has been instrumental for investigating bottlenecks in single-GPU simulations, fostering the development of efficient algorithms for different simulation phases.

4.1.2.1. Gol+21 [38]
This study presents the prototype library NeuronGPU, verifies the code in terms of simulation results, and validates performance by implementing the PD14 model and comparing with NEST CPU and GeNN using the amended initial conditions. The simulations span $T_{\text{model}} = 10\,\text{s}$ with an additional 1 s warm-up time at the beginning of each simulation to avoid startup transients. Spikes emitted after the first 1 s of activity are recorded for verification, whereas the recording was disabled to assess the library performance. Simulations are performed on Tesla V100 and GeForce RTX 2080Ti GPU cards, with the latter being the one with minimum q_{RTF} (*Gol+21*). For the purpose of the present review, we revisit the configuration of *Gol+21* and estimate E_{syn} for the same workstation and software versions by measuring the wall socket power during the simulation phase. An earlier estimate of energy consumption for an analogous configuration, but only taking the GPU power consumption into account, comes from [41].

4.1.2.2. Gol+23a [39]
The study presents a novel runtime network construction method for NEST GPU, enabling the execution of this phase entirely on the GPU hardware. In this framework, the PD14 is employed both for benchmarking and verification purposes. Regarding benchmarking, simulations are run for $T_{\text{model}} = 10\,\text{s}$ with an additional warm-up time of 0.5 s at the beginning of the simulation, amended initial conditions were used, and spike recording is disabled. The study performs the benchmarking on both data center and consumer GPUs (i.e. Tesla V100, Ampere A100, GeForce RTX 2080Ti, and GeForce RTX 4090). Among the different GPU hardware systems, the GeForce RTX 4090 achieves the minimum q_{RTF} (*Gol+23a*). Here, we revisit the configuration of *Gol+23a* and estimate E_{syn} for the simulation codes and the workstation specified in [39].

4.1.3. GeNN
GPU enhanced Neuronal Networks (GeNN) [46] is a C++ library for generating efficient GPU kernels in CUDA or HIP for spiking neural network simulation (website: https://genn-team.github.io, source code: https://github.com/genn-team/genn). In more recent versions, this library is exposed to Python as PyGeNN [37]. GeNN is designed to be highly flexible, and one of the key enablers of this flexibility is that all neuron and synapse models are specified in a C-like language (GeNNCode), which can be written directly in Python model descriptions. While GeNN supports *data-parallel* training of spiking machine learning models across multiple GPUs using NCCL [76], its current focus is performance on single GPUs. Spiking neural network simulations in GeNN comprise a code generation stage, compilation of generated C++ and CUDA/HIP code, followed by network initialization and simulation. Network initialization can largely be offloaded onto the GPU. Users of GeNN define neuron dynamics in terms of an update procedure for a timestep, meaning that the model definition already encompasses both the underlying differential equations and the numerical solver.

4.1.3.1. KN18 [36]
This study presents extensions to GeNN that support heterogeneous dendritic delays and on-GPU initialization of state variables and connectivity (specified in GeNNCode). The authors benchmark a C++ implementation [77] of the PD14 model on GeNN 3.2.0 [78] and a variety of GPU hardware (Jetson TX2, GeForce GTX 1050Ti, Tesla K40c and Tesla V100). The simulations use the original initial conditions, record all spikes, and the external drive consists of Poisson spike trains, directly delivered to each neuron. The benchmarks cover the duration of $T_{\text{model}} = 10\,\text{s}$ and separately assess the wall clock time required for initialization and state propagation. The authors record the power usage over time at the main socket and report energy to solution as well as energy per synaptic event. The fastest simulation is achieved on the V100 (*KN18*). The investigation did not have physical access to the V100 system and therefore estimates power consumption by subtracting the GPU power reported by `nvidia-smi` on the K40-based system and adding the values for the V100.

4.1.3.2. Kni+21 [37]
This study presents a new Python frontend to GeNN (PyGeNN) as well as a new GPU-side spike recording system. In order to demonstrate the benefits of the new spike recording system and show that orchestrating simulations from Python has minimal performance overhead, the researchers benchmark GeNN 4.4.0 [79] on some newer GPU hardware (GeForce GTX 1650, Jetson Xavier NX and Titan RTX) using a new PyGeNN implementation [80] of the PD14 model. The initialization strategy and external drive are the same as in *KN18*. Simulations cover the duration of $T_{\text{model}} = 1\,\text{s}$ and assess the wall clock time required for state propagation while recording all spikes. The fastest simulation is achieved on the Titan RTX (*Kni+21*).

4.1.3.3. Gol+23b [39]

In the context of [39], the NEST GPU team considers GeNN as a reference and performs analogous benchmarks with GeNN 4.8.0. Also for GeNN, the study achieves the best performance on the GeForce RTX 4090. Revisiting the earlier work we estimate E_{syn} (*Gol+23b*) using the same workstation as in [39].

4.1.4. SpiNNaker

SpiNNaker [47] is a neuromorphic hardware platform designed to run spiking neural network simulations in biological real-time (source code: https://github.com/spinnakermanchester). This highly-parallel architecture uses low-power ARM-968 cores running at 200 Mhz, with 64KB local data memory and 32 KB local instruction memory, and 32-bit fixed-point calculations. Thus, there is no hardware support for floating-point operations. SpiNNaker chips made of up to 18 such cores are organized into boards of 48-chips which can be connected together. The biggest SpiNNaker machine built to date consists of over 1-million cores operating as a single machine. SpiNNaker machines execute spiking neural network simulations using the PyNN interface and sPyNNaker [81], and SpiNNTools [82] software APIs for facilitating problem description, partitioning, mapping, execution, and results extraction. To measure energy use in the studies considered here, a wall-socket power meter monitors an entire SpiNNaker system: SpiNNaker boards, communications switch, power supplies, and cooling fans. The meter used to profile PD14 simulations provides a reading accurate to 0.01 kWh, with measurements taken at the beginning and end of simulations via software controlled readings. The authors subsequently convert this energy to solution into energy per synaptic event by dividing by the total number of synaptic events, and assuming constant power consumption throughout the simulation period [10, 34].

Efforts to run the cortical microcircuit model have led to significant developments in the SpiNNaker software tool chain, and the PD14 model is now run regularly, in real-time and using the Poisson-drive, as part of the SpiNNaker software integration tests. This ensures that this model and others of a similar nature can be run on all future versions of the SpiNNaker software, demonstrating the lasting impact of the PD14 model on SpiNNaker development. The model has also helped inform the design of the successor system, SpiNNaker2 [56], which is expected to deliver around a 10x improvement in functional density and energy-efficiency.

4.1.4.1. vAl+18c [10]

This study reports the first successful execution of the PD14 model on SpiNNaker. The architecture was originally designed to run networks with 1 ms timesteps in real-time, however the authors demonstrate that to accurately model the dynamics the PD14 model requires a simulation timestep of 0.1 ms. While this validation was a key achievement, the resulting computation requires a slow-down of the machine to a real-time-factor of 20, due to the way the software environment maps the PD14 model to the hardware. The simulations use the original initial conditions as well as both DC (*vAl+18c*) and Poisson drive, and gather energy and wall-clock time across: preprocessing, execution, and the extraction of results and subsequent post processing. The investigations highlight issues of the original software stack with the speed of loading data onto, and saving data from, SpiNNaker. This inspired development work enabling generation of synaptic data in parallel on the cores of the machine prior to execution, speeding up the loading process by several orders of magnitude, and significantly reducing the host memory requirements.

4.1.4.2. Rho+19 [34]

This study reports improvements to the SpiNNaker software stack to enable SpiNNaker to run the PD14 model in biological real time. It should also be noted this is a hard real-time solution, where every 0.1 ms simulation time-step was computed in 0.1 ms wall-clock time. This was achieved through revisions to the model itself, updating the initialization to reduce the number of neurons initialized with membrane potential state above threshold (leading to the amended Initial conditions (section 4.2.2.2)), and hence reducing excessive spike firing in the first timestep of the simulation, which had previously compromised the hard real-time compute requirements. Additional updates include revisions to the way computation is mapped to the SpiNNaker architecture, taking inspiration from previous research [83, 84], to enable parallelization of spike processing, a fundamental processing bottleneck in the study above [10]. Additional modifications improve synchronization of the SpiNNaker cores to ensure correct processing of the model, and include *spike coloring* to identify delays of spikes being transmitted across the machine with the smaller timesteps in use. As a result the study reports simulations with a real-time factor $q_{RTF} = 1$ for both DC (*Rho+19a*) and Poisson (*Rho+19b*) drive, where energy measurements extend over 12 hours of wall-clock time—an up-time with a consistent simulation never before achieved with SpiNNaker.

4.1.5. CsNN

CsNN specifies a simulation architecture that can be prototypically mapped to an FPGA cluster as a hardware substrate. For CsNN, here the IBM INC-3000 system serves as the hardware platform. The IBM INC-3000 system consists of 16 PCB's (INC-cards, size 56 cm × 35 cm), each hosting 27 reconfigurable Xilinx ZYNQ XC7Z045 SoC nodes. The whole INC-3000 system consists of 432 FPGA nodes which are connected by a three-dimensional mesh. Earlier work already looked at single FPGAs (e.g. [85]). The IBM system, however, was initially part of the IBM General Artificial Intelligence project, in which networks of non-spiking neurons were investigated. It was then shown that this FPGA cluster architecture can also be used to advantage for the simulation of large-scale networks of spiking neurons.

4.1.5.1. Hei+22 [40]

This study completely expresses the system architecture of CsNN in a high-level programming language and maps the specification to FPGA logic using high-level synthesis tools. This design flow enables the implementation and characterization of a large number of architecture variants as part of a broad design space exploration. The authors employ the framework to characterize different advanced ODE solvers in terms of their performance, with the perspective of integrating more realistic neuron and synapse models as used in PD14. Non-linear equations arise, for example, when considering bio-realistic conductance-based synaptic couplings, AdEx neurons [86], Izhikevich neurons [87], or a non-linear membrane dynamics based on ion channels. In particular, all arithmetic operators are implemented in single-float precision. The work also extends the concept of procedural connectivity [88, 89] to the representation of synaptic connections with the attributes of synaptic weight and delay. Instead of storing synaptic parameters in a memory system, the algorithm derives all connection parameters at runtime by addressing individual seeds. The authors propose dedicated random number generators for the sampling from arbitrary distribution functions for the runtime-efficient online generation of synaptic parameters. The approach maps the connectivity of PD14 to a highly parallel near-memory computing architecture with a high compression factor and rapid look-up of individual connections.

In the simulation, the authors parameterize online generators for the network generation in such a way that they exactly reproduce the detailed connection statistics of the genuine PD14 model. The algorithm not only reproduces network densities but also the properties of synaptic multiples (multiple connections between pairs of neurons [90]). The study avoids initial transients by determining the initial conditions for the membrane potentials, synapse states, and the delay buffers from the steady-state of a prior test simulation. Manufacturer data sheets provide first worst-case estimates of energy consumption for CsNN [41]. For the present review, we revisit the original configuration and determine energy consumption in simulations covering an interval of $T_{\text{model}} = 15\,\text{min}$ (*Hei+22*).

4.1.6. neuroAIx

neuroAIx is a simulation platform for computational neuroscience, comprising a cluster of 35 FPGAs [41]. Its development was driven by identifying key requirements of such a system: flexibility, observability, scalability and replicability. The study first analyzes the computational properties of biological networks of spiking neurons in the human brain. This results in an application-specific architecture for tackling the three key bottlenecks limiting the performance of previous platforms: computation, memory and communication [41]. Firstly, to solve the computational bottleneck, neuroAIx implements two separate solutions for computing neuronal state updates at extremely low latencies: a dedicated register-transfer level design for prevalent neuron models like leaky integrate-and-fire, and the custom neuron processor architecture nAIXt [91]. Secondly, the architecture tackles the memory bottleneck by directly accessing off-chip memory bypassing any processor cores and with latency hiding techniques like prefetching synapse data. Thereby, the system avoids latency overheads (beyond the technical limitations of the underlying memory modules). Thirdly, regarding the communication bottleneck, the authors introduce overlayed long hop topologies [52] as an extension to mesh-like network topologies. Their key features are symmetric connections between distant network nodes to minimize the maximum number of hops needed to communicate spike messages. The study combines long hops with a customized routing algorithm and deadlock prevention [92], a condensed version of the Go-Back-N ARQ flow control protocol and a synchronization scheme ensuring minimum latency.

The resulting platform can replay simulations and produce bit-equivalent results, while exhibiting spiking behavior statistically equivalent to simulations in NEST. First analyses additionally confirm weak and strong scaling properties. Future iterations of neuroAIx will further increase its performance and applicability, by building on newer FPGAs, and introducing support for (three-factor) plasticity rules and a user-friendly cloud interface with NEST/NESTML integration.

4.1.6.1. Kau+23 [41]

The work introduces the neuroAIx FPGA cluster and demonstrates its performance on relevant neural network sizes using the PyNEST PD14 model [68] with NEST v3.3 [93] as a reference. This code includes the original initial conditions and uses DC input to drive activity. For comparability, the study uses the same generated connectome for simulations on neuroAIx and an HPC cluster. Finally, a simulation covering a duration of $T_{\text{model}} = 15\,\text{min}$ assesses performance, where it discards the first second to ignore transient activity. All spikes are recorded and transmitted to a host computer to verify the resulting dynamics using second-order statistics like inter-spike intervals (with spike recording on neuroAIx incurring no performance hits). neuroAIx achieves the minimum q_{RTF} by using the entire cluster of 35 FPGAs. The authors estimate E_{syn} by measuring the power consumption of individual FPGAs during simulation using current clamps at the wall socket and verify results against data reported by the internal power management units.

4.1.7. BrainScaleS

The physical implementation of neuron and synapse dynamics in analog microelectronic circuits can offer significant advantages in energy efficiency and speed compared to numerical simulation. BrainScaleS-1 [94] is a mixed-signal wafer-scale accelerator (implemented in 180 mm CMOS) for plastic spiking neural networks providing high-speed model dynamics and energy-efficient operation. A multi-layered software stack [95] provides automated conversion between user-defined experiments in the high-level description language PyNN and the neuromorphic substrate.

4.1.7.1. Sch+25 [57]

The study implements the PD14 model on a single BrainScaleS-1 wafer-scale module, featuring 196k AdEx neuron circuits and 43 M plastic conductance-based synapses. The authors adapt the model to fit the hardware's properties, including down-scaling to accommodate the biological fan-in following the approach outlined in [12]: in particular, the number of synapses in the PD14 model is greater than the number of available synapses on the silicon substrate. In addition, biological fan-in, or connection density, requires BrainScaleS neuron circuits to be combined, thereby decreasing the number of available model neurons per wafer module. The modified model represents the external drive by DC current. Additionally, the work replaces the original current-based synapse model to match the substrate's conductance-based dynamics. This adaptation process, transitioning from a numerical model to a neuromorphic implementation, is non-trivial, but promises order-of-magnitude improvements in energy efficiency and latency compared to conventional software simulation as well as hardware-accelerated numerical simulation. For the down-scaled model implementation running at a real-time factor of $q_{\text{RTF}} = 0.0001$ the system achieves a peak rate of 162×10^9 spikes per second and an upper bound of $E_{\text{syn}} < 0.012\,\mu\text{J}$.

4.2. Benchmarking recipe

The joined experience of the individual studies (section 4.1) reviewed here suggests guidelines for the evaluation of the performance of computing systems using the PD14 model as a benchmark. The steps to be taken are: 1) to obtain the reference model from a reliable source, 2) to implement a simulator-specific algorithmic interpretation of the model, 3) to achieve sufficient accuracy, and finally 4) to assess performance in terms of time and energy to solution (compare with figure 1). Here we elaborate on the reasoning behind simulation and parameter recommendations and provide a checklist in table 2.

4.2.1. Model reference

The original neuroscientific publication introducing the PD14 model and describing the data integration process as well as the neuroscientific conclusions is [11]. However, on the basis of the progress of the community in the formal description of neuronal network models since the first publication of PD14 (including the scientific case for large-scale brain simulations [1], connectivity concepts [90], and the modeling language NESTML [96]) and the results of the present study, we recommend to use the novel model reference: The open access web site https://github.com/INM-6/microcircuit-PD14-model provides a detailed, implementation-agnostic mathematical description of the model, as well as a documented PyNEST reference implementation. The repository also collects the performance results of this study and supports the inclusion of new data points as they become available.

Table 2. Checklist with recommended model and simulation parameters for the PD14 model.

Model reference	https://github.com/INM-6/microcircuit-PD14-model
External drive	State whether DC or Poisson is used
Initial conditions	Amended initial conditions: distribute membrane potentials normally with population-specific mean and variance
Warm-up time	Discard the initial 500 ms of model time from the data to be analyzed
Simulation duration	Accuracy: $T_{\text{model}} = 15\,\text{min}$, performance: $T_{\text{model}} \geqslant 10\,\text{s}$
Repeated simulations	Average across ten random seeds
Spike recording	Accuracy: yes, performance: no
Accuracy	Compute distributions of 1) single-neuron *firing rate* (FR), 2) *coefficient of variation* (CV) of the inter-spike intervals (ISI), and 3) short-term spike-count *correlation coefficients* (CC), and compare with reference data
Performance	Measure real-time factor q_{RTF} and the energy per synaptic event E_{syn} (include all contributions necessary for running the simulations at the power outlet)

4.2.2. Model and simulation parameters

4.2.2.1. External drive

In the PD14 model, each neuron receives input spikes from an independent Poisson process with a population-specific rate. This rate is parameterized by a global base rate multiplied by a population-specific in-degree. According to the original study [11], the Poisson input can be replaced by a DC input corresponding to the mean current delivered by the Poisson input (see their figure 7) without compromising the accuracy of the network activity statistics. While Poisson input might be biologically more realistic, DC input is less computationally expensive and avoids the dependence of the performance data on the efficiency of the implementation of the random number generators. Some studies considered in this review use Poisson and others DC input as stated in table 1. We recommend to clearly state which external drive is used for performance comparability.

4.2.2.2. Initial conditions

Simulating neurons firing at higher rates (also temporarily) requires more communication and is, for event-driven algorithms, therefore computationally more expensive than at lower rates. The dynamics of the PD14 model are mostly stable with only weak fluctuations in global and per-neuron firing rates [45]. Convergence speed into this state is determined by initial conditions. The original study [11] uses normally distributed initial membrane potentials with the same mean and standard deviation for all populations. Here we recommend to distribute initial conditions with the population-specific means and variances of the stationary network state after initial transients have decayed [97]. The present work calls this set of parameters the amended initial conditions. The stable network states resulting from the two sets are statistically identical. While avoiding transient activity with high firing rates, population-specific distributions of initial membrane potentials also ensure fast convergence to the stable state.

4.2.2.3. Warm-up time

To avoid analyzing data from initial transients in the dynamics of the model, we recommend discarding a warm-up time of 500 ms. This holds for both the statistical analysis of spike data and the measurement of simulation speed.

4.2.2.4. Simulation duration

The specificity of verification measures relies on the observation duration. For an investigation of this relationship on the example of the PD14 model see [45]. Simulation times need to be large enough to distinguish model specific spike correlations from spurious correlations of independent spike trains of finite length. To assess accuracy, we recommend simulating for a biological time of $T_{\text{model}} = 15\,\text{min}$. To assess performance, we recommend simulating for at least $T_{\text{model}} = 10\,\text{s}$. An increased speed of the simulation requires a longer stretch of biological time covered by the model to keep the time it takes the computing system to complete the simulations large enough for reliable measurements of wall-clock time and energy consumption.

4.2.2.5. Repeated simulations
The PD14 model is a highly irregular system and the network activity is evaluated on a statistical level. Random numbers are involved in drawing connections during network construction, initializing membrane potentials, and possibly providing Poisson input during the simulation. We recommend averaging simulation results across at least ten random seeds to gain statistics across a range of microscopically different network instantiations and dynamics. In addition, we also recommend repeating simulations for performance measurements to account for fluctuations in the computing system.

4.2.2.6. Spike recording
Recording spike data to files or transfer of the data to a companion job is required for accuracy assessment. A simulation engine may buffer spikes locally and only write them to files after a data limit has been reached. Consequently, the performance of the simulation may depend on the stretch of time covered by the simulation. To disentangle the performance of the simulation engine from the performance of analysis backends, we suggest not to record spikes when assessing the performance.

4.2.3. Accuracy
We recommend a statistical comparison of the obtained spike data to reference data created by a trusted simulation code. The distributions of single-neuron *firing rate* (FR), distributions of *coefficient of variation* (CV) of the inter-spike intervals (ISI), and distributions of short-term spike-count *correlation coefficients* (CC) should be computed. For details, refer to [45, Sec. 2.3.1] and see also [98]. Demonstrating the visual overlap of these distributions with the reference is the minimum requirement. Computing scores to quantify the match of the distributions is appreciated such as Kullback–Leibler divergence, Kolmogorov-Smirnov test, or the Earth Mover Distance.

4.2.4. Performance
Time to solution and energy to solution should be assessed by the real-time factor and the energy per synaptic event, respectively. For definitions, see the beginning of section 2. In general, we recommend that power measurements account for the total system power: all contributions necessary for running the simulations measurable at the power outlet. This definition excludes additional potential energy costs stemming, for example, from keeping the computing system at a stable temperature. While important for assessing the true cost for the operator of a computing facility, it is a not required to assess the power of the different systems in a pragmatic and comparable fashion.

Data availability statement

All data that support the findings of this study are included within the article. Model specification and reference data are available at: https://github.com/INM-6/microcircuit-PD14-model.

Funding

This project received funding from the European Union's Horizon Europe Programme under the Specific Grant Agreement No. 10 114 7319 (EBRAINS 2.0 Project); the Joint Lab 'Supercomputing and Modeling for the Human Brain'; Juelich Research Centre intramural STEF fund for the update of instruments; HiRSE_PS, the Helmholtz Platform for Research Software Engineering—Preparatory Study, an innovation pool project of the Helmholtz Association; the Helmholtz Association's Initiative and Networking Fund under project number SO-092 (Advanced Computing Architectures, ACA); the Federal Ministry of Education and Research (BMBF, Germany) through the projects NEUROTEC II (grant number 16ME0399) and Clusters4Future-NeuroSys (grant number 03ZU1106CA); EPSRC (grant numbers EP/P006094/1, EP/S030964/1 and EP/V052241/1); the European Union's Horizon 2020 research and innovation program under Grant Agreement 945 539 (HBP SGA3); and Google Summer of Code. The contribution of the University of Cagliari to this study was supported by the Project e.INS Ecosystem of Innovation for Next Generation Sardinia—spoke 10-CUP F53C22000430001—MUR Code No. ECS00000038. Authors involved in NEST GPU development acknowledge the use of Fenix Infrastructure resources, which are partially funded from the European Union's Horizon 2020 research and innovation programme through the ICEI project under the Grant Agreement No. 800 858. Open access publication funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)–49 111 1487.

ORCID iDs

Johanna Senk 0000-0002-6304-062X
Anno C Kurth 0000-0002-9557-1003
Steve Furber 0000-0002-6524-3367
Tobias Gemmeke 0000-0003-1583-3411
Bruno Golosio 0000-0001-5144-6932
Arne Heittmann 0000-0001-9466-0568
James C Knight 0000-0003-0577-0074
Eric Müller 0000-0001-5880-2012
Tobias Noll 0000-0003-0613-7860
Thomas Nowotny 0000-0002-4451-915X
Gorka Peraza Coppola 0009-0008-0926-3305
Luca Peres 0000-0001-9748-9073
Oliver Rhodes 0000-0003-1728-2828
Andrew Rowley 0000-0002-2646-8520
Johannes Schemmel 0000-0003-1440-4375
Tim Stadtmann 0009-0007-7452-8245
Tom Tetzlaff 0000-0001-5794-5425
Gianmarco Tiddia 0000-0001-7524-0285
Sacha J van Albada 0000-0003-0682-4855
José Villamar 0009-0007-8791-7100
Markus Diesmann 0000-0002-2308-5727

References

[1] Einevoll G T *et al* 2019 The scientific case for brain simulations *Neuron* **102** 735–44
[2] Markram H *et al* 2015 Reconstruction and simulation of neocortical microcircuitry *Cell* **163** 456–92
[3] Reimann M W et al 2024 Modeling and simulation of neocortical micro- and mesocircuitry. part I: anatomy *BioRxiv Preprint* (https://doi.org/10.1101/2022.08.11.503144)
[4] Billeh Y N *et al* 2020 Systematic integration of structural and functional data into multi-scale models of mouse primary visual cortex *Neuron* **106** 388–403
[5] Mountcastle V 1978 An organizing principle for cerebral function: the unit module and the distributed system *The Mindful Brain*
[6] Douglas R J, Martin K A C and Whitteridge D 1989 A canonical microcircuit for neocortex *Neural Comput.* **1** 480–8
[7] Da Costa N M M and Martin K 2010 Whose cortical column would that be? *Front. Neuroanat.* **4** 1265
[8] Florio M and Huttner W B 2014 Neural progenitors, neurogenesis and the evolution of the neocortex *Development* **141** 2182–94
[9] Douglas R J and Martin K A C 2004 Neuronal circuits of the neocortex *Annu. Rev. Neurosci.* **27** 419–51
[10] van Albada S J *et al* 2018 Performance comparison of the digital neuromorphic hardware spinnaker and the neural network simulation software NEST for a full-scale cortical microcircuit model *Front. Neurosci.* **12** 291
[11] Potjans T C and Diesmann M 2014 The cell-type specific cortical microcircuit: relating structure and activity in a full-scale spiking network model *Cereb. Cortex* **24** 785–806
[12] van Albada S J, Helias M and Diesmann M 2015 Scalability of asynchronous networks is limited by one-to-one mapping between effective connectivity and correlations *PLOS Comput. Biol.* **11** e1004490
[13] Shimoura R O, Kamiji N L, Oliveira Pena R F, Cordeiro V L, Ceballos C C, Romaro C and Roque A C 2018 [Re] the cell-type specific cortical microcircuit: relating structure and activity in a full-scale spiking network model *ReScience* **4**
[14] Romaro C, Najman F A, Lytton W W, Roque A C and Dura-Bernal S 2021 NetPyNE implementation and scaling of the Potjans-Diesmann cortical microcircuit model *Neural Comput.* **33** 1993–2032
[15] Wagatsuma N, Potjans T C, Diesmann M and Fukai T 2011 Layer-dependent attentional processing by top-down signals in a visual cortical microcircuit model *Front. Comput. Neurosci.* **5** 31
[16] Wagatsuma N, Potjans T C, Diesmann M, Sakai K and Fukai T 2013 Spatial and feature-based attention in a layered cortical microcircuit model *PLOS One* **8** e80788
[17] Merkt B, Schüßler F and Rotter S 2019 Propagation of orientation selectivity in a spiking network model of layered primary visual cortex *PLOS Comput. Biol.* **15** e1007080
[18] Lee J H, Koch C and Mihalas S 2017 A computational analysis of the function of three inhibitory cell types in contextual visual processing *Front. Comput. Neurosci.* **11** 28
[19] Wagatsuma N, Nobukawa S and Fukai T 2023 A microcircuit model involving parvalbumin, somatostatin and vasoactive intestinal polypeptide inhibitory interneurons for the modulation of neuronal oscillation during visual processing *Cereb. Cortex* **33** 4459–77
[20] Jiang H -J, Qi G, Duarte R, Feldmeyer D and van Albada S J 2024 A layered microcircuit model of somatosensory cortex with three interneuron types and cell-type-specific short-term plasticity *Cereb. Cortex* **34** bhae378
[21] Diaz-Pier S, Naveau M, Butz-Ostendorf M and Morrison A 2016 Automatic generation of connectivity for large-scale neuronal network models through structural plasticity *Front. Neuroanat.* **10** 57
[22] Giacopelli G, Tegolo D, Spera E and Migliore M 2021 On the structural connectivity of large-scale models of brain networks at cellular level *Sci. Rep.* **11** 4345
[23] Schmidt M, Bakker R, Shen K, Bezgin G, Diesmann M and van Albada S J 2018 A multi-scale layer-resolved spiking network model of resting-state dynamics in macaque visual cortical areas *PLOS Comput. Biol.* **14** e1006359

[24] Pronold J, Morales-Gregorio A, Rostami V and van Albada S J 2024 Cortical multi-area model with joint excitatory-inhibitory clusters accounts for spiking statistics, inter-area propagation and variability dynamics *BioRxiv Preprint* https://doi.org/10.1101/2024.01.30.577979)
[25] Hagen E, Dahmen D, Stavrinou M L, Lindén H, Tetzlaff T, van Albada S J, Grün S, Diesmann M and Einevoll G T 2016 Hybrid scheme for modeling local field potentials from point-neuron networks *Cereb. Cortex* **26** bhw237
[26] Næss S, Halnes G, Hagen E, Hagler Jr D J, Dale A M, Einevoll G T and Ness T V 2021 Biophysically detailed forward modeling of the neural origin of EEG and MEG signals *NeuroImage* **225** 117467
[27] Bos H, Diesmann M and Helias M 2016 Identifying anatomical origins of coexisting oscillations in the cortical microcircuit *PLOS Comput. Biol.* **12** e1005132
[28] Schuecker J, Schmidt M, van Albada S J, Diesmann M and Helias M 2017 Fundamental activity constraints lead to specific interpretations of the connectome *PLOS Comput. Biol.* **13** e1005179
[29] Cain N, Iyer R, Koch C and Mihalas S 2016 The computational properties of a simplified cortical column model *PLOS Comput. Biol.* **12** e1005045
[30] Osborne H, Deutz L and Kamps M 2021 Multidimensional dynamical systems with noise: population density techniques for neuroscience *Computational Modelling of the Brain* (Springer) pp 159–78
[31] Schwalger T, Deger M and Gerstner W 2017 Towards a theory of cortical columns: from spiking neurons to interacting neural populations of finite size *PLOS Comput. Biol.* **13** e1005507
[32] René A, Longtin A and Macke J H 2020 Inference of a mesoscopic population model from population spike trains *Neural Comput.* **32** 1448–98
[33] Plesser H E *et al* 2025 Building on models–a perspective for computational neuroscience *Cereb. Cortex* **35** bhaf295
[34] Rhodes O, Peres L, Rowley A G D, Gait A, Plana L A, Brenninkmeijer C and Furber S B 2019 Real-time cortical simulation on neuromorphic hardware *Phil. Trans. R. Soc. A* **378** 20190160
[35] Kurth A C, Senk J, Terhorst D, Finnerty J and Diesmann M 2022 Sub-realtime simulation of a neuronal network of natural density *Neuromorph. Comput. Eng.* **2** 021001
[36] Knight J C and Nowotny T 2018 GPUs outperform current HPC and neuromorphic solutions in terms of speed and energy when simulating a highly-connected cortical model *Front. Neurosci.* **12** 1–19
[37] Knight J C, Komissarov A and Nowotny T 2021 PyGeNN: a python library for GPU-enhanced neural networks *Front. Neuroinformatics* **15** 659005
[38] Golosio B, Tiddia G, Luca C D, Pastorelli E, Simula F and Paolucci P S 2021 Fast simulations of highly-connected spiking cortical models using GPUs *Front. Comput. Neurosci.* **15** 627620
[39] Golosio B *et al* 2023 Runtime construction of large-scale spiking neuronal network models on GPU devices *Appl. Sci.* **13** 9598
[40] Heittmann A, Psychou G, Trensch G, Cox C E, Wilcke W W, Diesmann M and Noll T G 2022 Simulating the cortical microcircuit significantly faster than real time on the IBM INC-3000 neural supercomputer *Front. Neurosci.* **15** 728460
[41] Kauth K, Stadtmann T, Sobhani V and Gemmeke T 2023 neuroAIx-framework: design of future neuroscience simulation systems exhibiting execution of the cortical microcircuit model 20× faster than biological real-time *Front. Comput. Neurosci.* **17** 1144143
[42] Wilkinson B and Allen M 2004 *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers* 2nd ed. (Prentice Hall)
[43] van Albada S J, Pronold J, van Meegen A and Diesmann M 2021 Usage and scaling of an open-source spiking multi-area model of monkey cortex *Lecture Notes in Computer Science* (Springer) pp 47–59
[44] Gewaltig M -O and Diesmann M 2007 NEST (NEural Simulation Tool) *Scholarpedia J.* **2** 1430
[45] Dasbach S, Tetzlaff T, Diesmann M and Senk J 2021 Dynamical characteristics of recurrent neuronal networks are robust against low synaptic weight resolution *Front. Neurosci.* **15** 757790
[46] Yavuz E, Turner J and Nowotny T 2016 GeNN: a code generation framework for accelerated brain simulations *Sci. Rep.* **6** 18854
[47] Furber S and Bogdan P 2020 *SpiNNaker: A Spiking Neural Network Architecture* (John Soldatos) (https://doi.org/10.1561/978-1-68083-653-0)
[48] Dennard R H, Gaensslen F H, Yu H -N, Rideout V L, Bassous E and LeBlanc A R 1974 Design of ion-implanted MOSFET's with very small physical dimensions *IEEE J. Solid-State Circuits* **9** 256–68
[49] Bohr M 2007 A 30 year retrospective on Dennard's MOSFET scaling paper *IEEE Solid-State Circuits Newslett.* **12** 11–13
[50] Aimone J B, Awile O, Diesmann M, Knight J C, Nowotny T and Schürmann F 2023 Editorial: neuroscience, computing, performance and benchmarks: why it matters to neuroscience how fast we can compute *Front. Neuroinformatics* **17** 1157418
[51] Mahowald M and Douglas R 1991 A silicon neuron *Nature* **354** 515–8
[52] Kauth K, Stadtmann T, Brandhofer R, Sobhani V and Gemmeke T 2020 Communication architecture enabling 100x accelerated simulation of biological neural networks *Proc. Workshop on System-Level Interconnect: Problems and Pathfinding Workshop* pp 1–8
[53] Trensch G and Morrison A 2022 A system-on-chip based hybrid neuromorphic compute node architecture for reproducible hyper-real-time simulations of spiking neural networks *Front. Neuroinformatics* **16** 884033
[54] Herten A *et al* 2024 Application-driven exascale: the JUPITER benchmark suite *SC24: Int. Conf. for High Performance Computing, Networking, Storage and Analysis* (IEEE) pp 1–45
[55] Zhang L, Wahib M and Matsuoka S 2019 Understanding the overheads of launching CUDA kernels *Proc. 48th Int. Conf. on Parallel Processing* (Association for Computing Machiner) pp 5–8
[56] Mayr C, Hoeppner S and Furber S 2019 SpiNNaker 2: A 10 million core processor system for brain simulation and machine learning (arXiv:1911.02385)
[57] Schmidt H, Grübl A, Montes J, Müller E, Schmitt S and Schemmel J 2025 Demonstrating the advantages of analog wafer-scale neuromorphic hardware *Neuro-Inspired Computational Elements Workshop (NICE)*
[58] Davison A, Brüderle D, Eppler J M, Kremkow J, Muller E, Pecevski D, Perrinet L and Yger P 2009 PyNN: a common interface for neuronal network simulators *Front. Neuroinformatics* **2** 10
[59] Albers J *et al* 2022 A modular workflow for performance benchmarking of neuronal network simulations *Front. Neuroinformatics* **16** 837549
[60] Kurth A C, Albers J, Diesmann M and van Albada S J 2024 Cell-type specific projection patterns promote balanced activity in cortical microcircuits *BioRxiv Preprint* (https://doi.org/10.1101/2024.10.03.616539)
[61] Schmidt M, Bakker R, Hilgetag C C, Diesmann M and van Albada S J 2018 Multi-scale account of the network structure of macaque visual cortex *Brain Struct. Funct,* **223** 1409–35
[62] Tiddia G *et al* 2022 Fast simulation of a multi-area spiking network model of macaque cortex on an MPI-GPU cluster *Front. Neuroinformatics* **16** 883333

[63] Knight J C and Nowotny T 2021 Larger GPU-accelerated brain simulations with procedural connectivity *Nat. Comput. Sci.* **1** 136–42
[64] Senk J, Hagen E, van Albada S J and Diesmann M 2024 Reconciliation of weak pairwise spike–train correlations and highly coherent local field potentials across space *Cereb. Cortex* **34** bhae405
[65] Yik J *et al* 2025 The neurobench framework for benchmarking neuromorphic computing algorithms and systems *Nat. Commun.* **16** 1545
[66] OpenMP Architecture Review Board 2008 Specification *OpenMP Application Program Interface* (available at: www.openmp.org/mp-documents/spec30.pdf) (Accessed 27 September 2016)
[67] Message Passing Interface Forum 2009 MPI: a message-passing interface standard, version 2.2 *Technical Report*
[68] Eppler J M, Helias M, Muller E, Diesmann M and Gewaltig M 2009 PyNEST: a convenient interface to the NEST simulator *Front. Neuroinformatics* **2** 12
[69] Zaytsev Y V and Morrison A 2014 CyNEST: a maintainable Cython-based interface for the NEST simulator *Front. Neuroinformatics* **8** 23
[70] Rotter S and Diesmann M 1999 Exact digital simulation of time-invariant linear systems with applications to neuronal modeling *Biol. Cybern.* **81** 381–402
[71] Ippen T, Eppler J M, Plesser H E and Diesmann M 2017 Constructing neuronal network models in massively parallel environments *Front. Neuroinformatics* **11** 30
[72] Eppler J M *et al* 2015 NEST 2.8.0 *Zenodo* (https://doi.org/10.5281/zenodo.32969)
[73] Morrison A, Straube S, Plesser H E and Diesmann M 2007 Exact subthreshold integration with continuous spike times in discrete-time neural network simulations *Neural Comput.* **19** 47–79
[74] Hanuschkin A, Kunkel S, Helias M, Morrison A and Diesmann M 2010 A general and efficient method for incorporating precise spike times in globally time-driven simulations *Front. Neuroinformatics* **4** 113
[75] Peyser A *et al* 2021 NEST 2.14.1 *Zenodo* (https://doi.org/10.5281/zenodo.4018724)
[76] NVIDIA Corporation *NVIDIA Collective Communications Library (NCCL)* (available at: https://developer.nvidia.com/nccl)
[77] Knight J C and Nowotny T 2018 *BrainsOnBoard/frontiers_genn_paper: Publication* (https://doi.org/10.5281/zenodo.15101746)
[78] Knight J C, Nowotny T, Turner J P, Yavuz E, Zhang M, Diamond A, Cope A, Stimberg M, Kern F and Givon L E 2018 *genn-team/genn: GeNN 3.2.0* (https://doi.org/10.5281/zenodo.1478540)
[79] Knight J C et al 2021 genn-team/genn: GeNN 4.4.0 *Zenodo* (https://doi.org/10.5281/zenodo.4419159)
[80] Knight J C 2021 BrainsOnBoard/pygenn_paper: publication *Zenodo* (https://doi.org/10.5281/zenodo.15101950)
[81] Rhodes O *et al* 2018 sPyNNaker: a software package for running pynn simulations on spinnaker *Front. Neurosci.* **12** 816
[82] Rowley A G D *et al* 2019 SpiNNTools: the execution engine for the spinnaker platform *Front. Neurosci.* **13** 231
[83] Knight J C and Furber S B 2016 Synapse-centric mapping of cortical models to the SpiNNaker neuromorphic architecture *Front. Neurosci.* **10** 420
[84] Galluppi F, Lagorce X, Stromatias E, Pfeiffer M, Plana L A, Furber S B and Benosman R B 2015 A framework for plasticity implementation on the SpiNNaker neural architecture *Front. Neurosci.* **8** 429
[85] Guerrero-Rivera R, Morrison A, Diesmann M and Pearce T C 2006 Programmable logic construction kits for hyper-real-time neuronal modeling *Neural Comput.* **18** 2651–79
[86] Brette R and Gerstner W 2005 Adaptive exponential integrate-and-fire model as an effective description of neuronal activity *J. Neurophysiol.* **94** 3637–42
[87] Izhikevich E M 2003 Simple model of spiking neurons *IEEE Trans. Neural Netw.* **14** 1569–72
[88] Roth U, Jahnke A and Klar H 1995 Hardware requirements for spike-processing neural networks *From Natural to Artificial Neural Computation* (Springer) pp 720–7
[89] Roth U, Eckardt F, Jahnke A and Klar H 1997 Efficient On-Line Computation of Connectivity: Architecture of the Connection Unit of NESPINN *Proc. 6th Int. Conf. on Microelectronics for Neural Networks, Evolutionary & Fuzzy Systems (Microneuro'97)* (Technische Universität Dresden) pp 31–39
[90] Senk J, Kriener B, Djurfeldt M, Voges N, Jiang H -J, Schüttler L, Gramelsberger G, Diesmann M, Plesser H E and van Albada S J 2022 Connectivity concepts in neuronal network modeling *PLOS Comput. Biol.* **18** e1010086
[91] Kauth K, Lanius C and Gemmeke T 2024 nAIxt: a Light-Weight Processor Architecture for Efficient Computation of Neuron Models *Architecture of Computing Systems* (Springer) pp 3–17
[92] Sobhani V, Kauth K, Stadtmann T and Gemmeke T 2022 Deadlock-freedom in computational neuroscience simulators *IEEE Design Test* **39** 70–78
[93] Spreizer S *et al* 2022 NEST 3.3 *Zenodo* (https://doi.org/10.5281/zenodo.6368024)
[94] Schmidt H *et al* 2023 From clean room to machine room: commissioning of the first-generation BrainScaleS wafer-scale neuromorphic system *Neuromorph. Comput. Eng.* **3** 034013
[95] Müller E *et al* 2022 The operating system of the neuromorphic BrainScaleS-1 system *Neurocomputing* **501** 790–810
[96] Linssen C, Babu P N, Eppler J M, Koll L, Rumpe B and Morrison A 2025 NESTML: a generic modeling language and code generation tool for the simulation of spiking neural networks with advanced plasticity rules *Front. Neuroinformatics* **19** 1544143
[97] Rowley A G D, Stokes A B, Knight J, Lester D R, Hopkins M, Davies S, Rast A, Bogdan P and Davidson S 2015 PyNN on spinnaker software 2015.004 *Zenodo* (https://doi.org/10.5281/zenodo.19230)
[98] Gutzen R, Papen M, Trensch G, Quaglio P, Grün S and Denker M 2018 Reproducible neural network simulations: statistical methods for model validation on the level of network activity data *Front. Neuroinformatics* **12** 90