



Institute for Information
and Communication Technologies,
Electronics and Applied Mathematics

Bottom-Up and Top-Down Neuromorphic Processor Design: Unveiling Roads to Embedded Cognition

Charlotte Frenkel

Thesis submitted in partial fulfillment of the requirements for the degree of
Docteur en sciences de l'ingénieur et technologie

Dissertation committee

Prof. David BOL	(advisor)	UCLouvain, Belgium
Prof. Jean-Didier LEGAT	(co-advisor)	UCLouvain, Belgium
Prof. Jean-Pierre RASKIN	(president)	UCLouvain, Belgium
Prof. Michel VERLEYSEN	(secretary)	UCLouvain, Belgium
Prof. Marian VERHELST		KULeuven, Belgium
Prof. Emre NEFTCI		UCIrvine, USA
Prof. Giacomo INDIVERI		UZH and ETHZ, Switzerland

JANUARY 2020

A Maman et à mon frère

Contents

Acknowledgments	vi
Abstract	ix
Acronyms	xi
Author's publication list	xii
Introduction	1
I.1 Bottom-up neuromorphic design strategy	2
I.2 Top-down neuromorphic design strategy	3
I.3 Thesis outline	5
I Bottom-Up Neuromorphic Design	9
1 Building blocks – Neurons and synapses as adaptive processing and memory elements	13
1.1 Neurons	14
1.1.1 Overview of neural codes	15
1.1.2 Comparison of silicon neuron implementations	16
1.1.3 Proposed neuron architecture	18
1.1.4 Simulation and implementation results	22
1.2 Synapses	23
1.2.1 Overview of spike-based synaptic plasticity learning rules	24
1.2.2 Comparison of STDP- and SDSP-based silicon synapse implementations	26
1.2.3 Simulation and implementation results	29
2 Integration – Neuromorphic experimentation platforms	34
2.1 Overview of neuromorphic experimentation platforms	35
2.1.1 Analog/mixed-signal designs	35
2.1.2 Digital designs	37
2.2 ODIN – A 28-nm neuromorphic processor with record neuron and online-learning synapse densities	39
2.2.1 Architecture and implementation	40
2.2.2 Specifications, measurements and benchmarking results	48

2.3	MorphIC – A 65-nm quad-core neuromorphic processor demonstrating large-scale connectivity with record binary online-learning synapse density	58
2.3.1	Architecture and implementation	60
2.3.2	Specifications, measurements and benchmarking results	70
2.4	Discussion	74
2.4.1	Comparative analysis of bottom-up neuromorphic designs	76
2.4.2	Hierarchical event routing	77
2.4.3	Bottom-up spike-based online learning performance	79
II	Top-Down Neuromorphic Design	82
3	Algorithms – Minimizing the training cost of neural networks	86
3.1	Overview of algorithms releasing backprop constraints toward biological plausibility and hardware efficiency	87
3.1.1	Releasing the weight transport problem requires random feedback	89
3.1.2	Releasing update locking requires local error or gradient prediction	89
3.2	Direct random target projection (DRTP) allows releasing both weight transport and update locking	90
3.2.1	Weight updates based only on the error sign provide learning to multi-layer networks	91
3.2.2	For classification, a feedback pathway is no longer required as the error sign is known in advance	94
3.2.3	DRTP delivers useful modulatory signals for classification	95
3.2.4	DRTP learns to classify MNIST and CIFAR-10 images without feedback	98
3.3	Discussion	101
4	Integration – Neuromorphic accelerators	104
4.1	Overview of neuromorphic accelerators	105
4.2	SPOON – Fusing spiking and non-spiking domains for low-power adaptive event-based data processing	108
4.2.1	Architecture and implementation	108
4.2.2	Specifications and benchmarking results	115
4.3	Discussion	117
	Conclusions and perspectives	121
A	Direct random target projection – Methods and proofs	143
A.1	Methods	143
A.2	Full proof of alignment for the BP and DRTP modulatory signals	145
A.3	Supplementary tables and figures	151

Acknowledgments

Embarking on academic research is the start of a long journey. Becoming a researcher requires a continuous and extensive training not only of technical skills, but also of mindset, acuteness, curiosity and patience. Only then comes the ability to ask the right questions, and thus to uncover the promising research tracks, key caveats and associated tradeoffs. If I think this four-year Ph.D. allowed me to take a first step toward this objective, it would certainly not have been possible without the guidance and support of numerous people. I would like to apologize for the ones I might forget.

First, I would like to thank my advisor Prof. David Bol for trusting me to start neuromorphic research at UCLouvain. Beyond your constant availability and support (I will definitely remember some great 20-minute meetings that degenerated into 1-hour open brainstorming sessions), I could always count on your scientific rigor and expertise when reviewing my results and papers. While I enjoyed a lot the freedom you gave me, I was lucky to have your sharp eye to benchmark potential ideas, identify hidden challenges and take a step back to keep the big picture in mind. I could not have hoped for a better supervision.

I am grateful to my Ph.D. thesis jury for their time, thoughtful comments to improve the manuscript and stimulating discussions during the defense. More specifically, I would like to thank: Prof. Jean-Didier Legat, for your great support (since the beginning of my engineering cursus) and for accepting to be co-advisor of this thesis; Prof. Jean-Pierre Raskin, for accepting to chair the jury despite your tight schedule; Prof. Michel Verleysen, for ensuring a solid machine learning background; Prof. Marian Verhelst, for your thorough comments on the manuscript, enthusiasm and responsiveness; Prof. Emre Neftci, for our great discussions in Capo Caccia, spanning the early draft of DRTP to the academic career. Last but not least, I would like to express my gratitude to Prof. Giacomo Indiveri. Thank you for being so supportive, accepting to take part in my supervisory panel since the start of my Ph.D. and inviting me for a one-month research stay at the Institute of Neuroinformatics (INI) in Zürich, an opportunity that I enjoyed a lot and had a strong positive impact at the beginning of my research. Now, three and a half years later, I am really glad to head for a postdoc at INI.

I would then like to thank the following people for their help and feedback: Prof. Denis Flandre, for taking part in my supervisory panel and for your will to further develop spike-based computation at UCLouvain; Prof. Christophe de Vleeschouwer, for your review of the DRTP preprint; Simon Carbonnelle, for the brainstorming sessions on the mysteries behind deep learning; Dr. Ning Qiao, for the many

discussions we had at INI and Capo Caccia, I am looking forward to our future collaboration; Dr. Hesham Mostafa, for proposing your help with AER libraries; Dr. Nicolas Moser (and friend since UCLouvain BSc), for offering your help to review the introduction and conclusion of the manuscript, it was always a pleasure to meet you at the ISCAS and BioCAS conferences; Prof. Frank Gürkaynak, for your help with the 65-nm layermap file; Dr. Thibaut Gurné, for allowing excellent interactions with Nokia – Bell Labs; Profs. Wulfram Gerstner and Jonathan Touboul, for your feedback on the number of Izhikevich behaviors of the AdExp neuron model; Prof. Abderazek Ben Abdallah, for your feedback on the ODIN paper and for giving me the opportunity to chair the neuromorphic track at IEEE MCSoc; Prof. Huajin Tang, for your feedback on the ODIN paper; Dr. Syed Ahmed Aamir, Yannik Stradmann, Sebastian Billaudelle and Benjamin Cramer, for our discussions clarifying the above-threshold design strategy behind the the BrainScaleS neuron model. I would also like to thank the participants of the Capo Caccia workshops and the INI students and researchers I met during my stay there for our shared research and extensive discussions. It already led to fruitful collaborations on sensor fusion, and I would like to thank Elisa Donati, Melika Payvand, Enea Ceolini, Gemma Taverni, Lyes Khacef, Sumit Bam Shrestha and Garrick Orchard for that. I am looking forward to the next ones at INI! Finally, I am grateful to the anonymous reviewers of my papers for their feedback (a special thank you goes to reviewer 2 of the ODIN paper, whose critical yet constructive feedback led to strongly clarify the final paper compared to the first preprint).

I was glad to carry out this research in the ECS research group, it was filled with a good atmosphere thanks to Maxime Schramme (adaptive body biasing, double check and team cheerfulness expert), Ludovic Moreau (digital expert, ISCAS teammate and homebrewed beer master), Rémi Dekimpe (biosignal expert), Thibault Pirson (neuromorph and sustainability expert), Pengcheng Xu (energy harvesting expert), Mathieu Xhonneux (RF and Linux expert), Roghayeh Saeidi (power management and memory expert), Adrian Kneip (in-memory expert), Marco Gonzalez (quality-of-signal expert), Gueric de Streel and Cecilia Gimeno (RF experts), Thomas Haine (imager, memory and tapeout expert), Pierre-Antoine Haddad (rectifier and graphene expert), Julien De Vos (switched-cap expert), Khoi Nguyen (digital expert and engineering teammate) and François Stas (Jedi master). A special thank you goes to my co-author Martin Lefebvre (imager and machine learning expert), with whom I enjoyed so much collaborating on the ODIN paper and, more importantly, on the DRTP paper. Achieving this final result would certainly not have been possible without your help (and your faith in the “little Balrog” inside me). Finally, I would like to thank the UCLouvain IEEE student branch officers as well as the Club ELEC and Club INFO members. It was a pleasure to chair the branch for two years, and I am glad to see it in good hands now with Adrian.

I also had the pleasure to supervise four great Master students and aspiring neuromorphs: Naomi Rousseau, Thibault Pirson, Laura Vandooren and Julien Verecken. I am glad, Julien, that you are willing to embark on the Ph.D. adventure and to pursue neuromorphic research when I will have left the lab: all the best!

Over these four years, I received excellent logistic support and would like to thank the following people: Brigitte Dupont, for your efficiency during the high emergency and desperation cases of computer crashes and license unavailability; Pascal Simon, for the WELCOME measurement room access and support; Souley Djadjandi and David Spôte, for your help with PCB assembly; Viviane Sauvage, Anne Adant, Isabelle Dargent, Nathalie Ponet, Marie-Christine Vandingenen and Christel Derzelle, for your responsiveness in daily administration; Kathrin Aguilar, for your kind and efficient handling of the Capo Caccia workshops.

I am grateful to the Fonds National de la Recherche Scientifique (F.R.S.-FNRS) of Belgium and to the Walloon Region for funding this project.

Finally, I was lucky to be able to count on the support of my friends, with special thanks to François Heremans (*mon cher binôme*) and Nicolas Torres (my yearly support novel penpal) and the *let's save the world* and *applied math* board game squads. Of course, my deepest thanks go to my family, without whom all of this would not have been possible. I would particularly like to thank my brother for his daily curiosity about my work, and my mother for being both my mainstay and my keystone since the beginning.

Charlotte

Abstract

While Moore’s law has driven exponential computing power expectations from silicon processing devices for the past fifty years, with its nearing end comes a strong incentive to seek for roads to embedded cognition not only in computing power, but also in computation organization and information representation. In an attempt to both better understand the brain and lead to more efficient silicon processing devices, these two aspects are at the core of the paradigm shift pursued in neuromorphic engineering. First, it investigates the co-location of processing and memory with neuron and synapse elements, while von-Neumann processors separate them. Second, it aims at encoding data both in space and time with all-or-none spike events, while von-Neumann processors rely on a clocked multi-bit representation. However, it is still unclear which of the *bottom-up* (neuroscience-driven) or *top-down* (application-driven) approaches to the design of neuromorphic silicon devices could unveil the most promising roads to embedded cognition. In order to clarify this question and outline design guidelines, this thesis is divided into two parts to investigate both standpoints.

The first part of the thesis focuses on the bottom-up approach, which is the preferred strategy for neuroscience experimentation and brain reverse-engineering through analysis by synthesis. From the building block level to the silicon integration, we design and demonstrate two bottom-up neuromorphic processor prototypes, both of which embed synaptic plasticity for low-overhead learning at the edge and online adaptation to uncontrolled environments. With the ODIN 256-neuron 64k-synapse chip, we show that moving synaptic plasticity computation to the neurons allows releasing design constraints on the synapses. The neurons rely on a custom model emulating phenomenologically the 20 Izhikevich behaviors and supporting spike-driven synaptic plasticity (SDSP). As opposed to spike-timing-dependent plasticity (STDP), which relies on the relative pre- and post-synaptic spike times and thus induces critical buffering overhead, SDSP updates rely on the post-synaptic neuron state at the time of the pre-synaptic spike. The proposed neuron and synapse models do not require time-stepped evaluation, allowing for an event-driven time-multiplexed digital design that leverages single-port foundry SRAMs for high-density integration. Measurement results demonstrate that ODIN consumes only 12.7pJ/SOP at 0.55V for a silicon area of 0.086mm² in 28-nm FDSOI CMOS, thus achieving record neuron and synapse densities. With the 2k-neuron 2M-synapse MorphIC chip, we further extend the ODIN architecture for large-scale integration. We propose (i) a stochastic version of SDSP (S-SDSP) for online learning with binary synapses and (ii) a hierarchical event routing infrastructure combining local crossbar, inter-core star-based and inter-chip mesh-based routing,

allowing for biological fan-in and fan-out values of 1k and 2k, respectively. MorphIC further extends the density claim of ODIN by reaching record neuron and synapse densities in a silicon area of 2.86mm^2 in 65-nm CMOS, while it demonstrates a favorable energy-area-accuracy tradeoff on MNIST compared to state-of-the-art spiking neural network (SNN) chips. However, the inherent difficulty for bottom-up designs lies in applying them to real-world scenarios beyond the scope of neuroscience-oriented applications.

The second part of the thesis investigates the top-down approach. By starting from the applicative problem, it attempts at answering the difficulty of bottom-up approaches in tackling real-world scenarios efficiently. Targeting the deployment of neuromorphic devices in adaptive edge computing applications, we first investigate how to release the key constraints that preclude the backpropagation of error (BP) algorithm from being hardware-efficient and biophysically-plausible: the *weight transport problem* and *update locking*. Based on the concept of feedback alignment, we propose the direct random target projection (DRTP) algorithm. By being entirely feedforward and allowing for layerwise training with local information, it releases the two key issues of BP. We show on the MNIST and CIFAR-10 datasets that DRTP provides a tradeoff between accuracy and computational cost that is suitable for edge computing applications. In order to demonstrate this claim *in silico*, we propose the SPOON event-driven CNN (eCNN), which embeds DRTP-based online learning at power and area overheads of only 16.8% and 11.8%, respectively. Event-driven convolutions with time-to-first-spike coding leverage sparsity from event-based sensors, while a combination with frame-based processing ensures maximum data reuse and parallelism in fully-connected layers. On MNIST, with $313\text{nJ}/\text{inference}$ at accuracies of 95.3% (online learning) or 97.5% (offline training) for an area of 0.32mm^2 in 28-nm FDSOI CMOS, pre-silicon results show that only SPOON reaches the efficiency of conventional machine learning accelerators while enabling online learning with event-based sensors. This point is further emphasized on N-MNIST, as SPOON achieves accuracies of 93.0% (online learning) or 93.8% (offline learning). However, defining a suitable target for bio-inspiration in top-down designs is difficult, as it should ensure both the efficiency and the relevance of the resulting neuromorphic device.

Therefore, we claim that each of these two design approaches can act as a *guide* to address the shortcomings of the other. We conclude this thesis by discussing perspectives for combining bottom-up and top-down design toward embedded cognition.

Acronyms

AdExp	Adaptive-exponential integrate-and-fire
AER	Address-event representation
AI	Artificial intelligence
ANN	Artificial neural network
BNN	Binary neural network
BP	Backpropagation
CMOS	Complementary metal-oxide semiconductor
CNN	Convolutional neural network
CPU	Central processing unit
DFA	Direct feedback alignment
DRAM	Dynamic random-access memory
DRTP	Direct random target projection
eCNN	Event-driven convolutional neural network
FA	Feedback alignment
FDSOI	Fully-depleted silicon-on-insulator
FIFO	First-in, first-out
FPGA	Field-programmable gate array
GALS	Globally-asynchronous locally-synchronous
GPU	Graphics processing unit
H&H	Hodgkin-Huxley
IoT	Internet-of-things
ISI	Inter-spike interval
I&F	Integrate-and-fire
LCA	Locally competitive algorithm
LFSR	Linear-feedback shift register
LIF	Leaky integrate-and-fire
LTD	Long-term depression
LTP	Long-term potentiation
MSB	Most significant bit
MSE	Mean squared error
NEF	Neural engineering framework
NoC	Network-on-chip
PVT	Process, voltage and temperature
sDFA	Sign-based direct feedback alignment
SDSP	Spike-dependent synaptic plasticity
SGD	Stochastic gradient descent
SNN	Spiking neural network
SOP	Synaptic operation
SRAM	Static random-access memory
STDP	Spike-timing-dependent plasticity
STP	Short-term plasticity
S-SDSP	Stochastic spike-dependent synaptic plasticity
S-STDP	Stochastic spike-timing-dependent plasticity
WTA	Winner-take-all

Author's publication list

Related journal papers

- JP1. C. Frenkel, M. Lefebvre, J.-D. Legat and D. Bol, "A 0.086-mm² 12.7-pJ/SOP 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28nm CMOS," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 1, pp. 145-158, 2019.
- JP2. C. Frenkel, J.-D. Legat and D. Bol, "MorphIC: A 65-nm 738k-synapse/mm² quad-core binary-weight digital neuromorphic processor with stochastic spike-driven online learning," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 5, pp. 999-1010, 2019.
- JP3. C. Frenkel^{*}, M. Lefebvre^{*} and D. Bol, "Learning without feedback: Direct random target projection as a feedback-alignment algorithm with layerwise feedforward training," *in preparation*, *arXiv preprint arXiv:1904.08513*, 2019.
- JP4. E. Ceolini^{*}, C. Frenkel^{*}, S. B. Shrestha^{*}, G. Taverni, L. Khacef, G. Orchard, M. Payvand and E. Donati, "Hand-gesture recognition based on EMG and event-based camera sensor fusion: A benchmark in neuromorphic computing," *Frontiers in Neuroscience*, 2020, *submitted*.

Related conference papers

- CP1. C. Frenkel, J.-D. Legat and D. Bol, "A compact phenomenological digital neuron implementing the 20 Izhikevich behaviors," *Proc. of IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2017.
- CP2. C. Frenkel, G. Indiveri, J.-D. Legat and D. Bol, "A fully-synthesized 20-gate digital spike-based synapse with embedded online learning," *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017.
- CP3. C. Frenkel, J.-D. Legat and D. Bol, "A 65-nm 738k-synapse/mm² quad-core binary-weight digital neuromorphic processor with stochastic spike-driven online learning," *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019.
- CP4. C. Frenkel, J.-D. Legat and D. Bol, "A 28-nm convolutional neuromorphic processor enabling online learning with spike-based retinas," *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, *accepted*.

^{*} Equal contributions.

Short-course invited presentations

- SC1. C. Frenkel, “Towards a bio-inspired computing paradigm through large-scale low-power neuromorphic circuits,” webinar broadcasted to worldwide Bell Labs locations from Nokia / Bell Labs, Antwerpen, Belgium, April 2019.
- SC2. C. Frenkel, “Neuromorphic computation – Introduction to the ODIN and MorphIC spiking neural network chips,” in the session “Intro 3: Neuromorphic computation” at the Capo Caccia cognitive neuromorphic engineering workshop, April 2019.
- SC3. C. Frenkel, *TBA*, Neuro-Inspired Computational Elements (NICE) workshop, Heidelberg, Germany, March 2020.

Unrelated journal papers

- UJP1. G. de Streel, F. Stas, T. Gurné, F. Durant, C. Frenkel, A. Cathelin and D. Bol, “SleepTalker: A ULV 802.15. 4a IR-UWB transmitter SoC in 28-nm FDSOI achieving 14 pJ/b at 27 Mb/s with channel selection based on adaptive FBB and digitally programmable pulse shaping,” *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 52, no. 4, pp. 1163-1177, 2017.
- UJP2. C. Gimeno, D. Flandre, M. Schramme, C. Frenkel and D. Bol, “A 2.24-pJ/bit 2.5-Gb/s IR-UWB receiver in 28-nm FDSOI CMOS for low-energy chip-to-chip communications,” *Elsevier International Journal of Electronics and Communications*, vol. 114, no. 152996, 2020.

Unrelated conference papers

- UCP1. C. Frenkel, J.-D. Legat and D. Bol, “A partial reconfiguration-based scheme to mitigate multiple-bit upsets for FPGAs in low-cost space applications,” *Proc. of IEEE International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, 2015.
- UCP2. C. Frenkel, J.-D. Legat and D. Bol, “Comparative analysis of redundancy schemes for soft-error detection in low-cost space applications,” *Proc. of IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, 2016.
- UCP3. G. de Streel, F. Stas, T. Gurné, F. Durant, C. Frenkel and D. Bol, “SleepTalker: A 28nm FDSOI ULV 802.15. 4a IR-UWB transmitter SoC achieving 14pJ/bit at 27Mb/s with adaptive-FBB-based channel selection and programmable pulse shape,” *Proc. of IEEE Symposium on VLSI Circuits (VLSI-Circuits)*, 2016.

- UCP4. D. Bol, M. Schramme, L. Moreau, T. Haine, P. Xu, C. Frenkel, R. Dekimpe, F. Stas and D. Flandre, “A 40-to-80MHz sub-4 μ W/MHz ULV Cortex-M0 MCU SoC in 28nm FDSOI with dual-loop adaptive back-bias generator for 20 μ s wake-up from deep fully retentive sleep mode,” *Proc. of IEEE International Symposium on Solid-State Circuits (ISSCC)*, 2019.

Introduction

Together with the development of computers came the promise for exponential integration *in silico*, known as *Moore's law*, and thus the dream to reach artificial cognition. As a first step toward bio-inspired computation, artificial neural networks (ANNs) take their roots in the observation that the brain processes information with densely-interconnected and distributed computational elements: the neurons. From the early developments starting in the 1940s with the McCulloch & Pitts neuron model [1], and through the pioneering work from Rosenblatt [2] and Hopfield [3], ANNs are now enclosed within the concept of *artificial intelligence* (AI). They are used for applications ranging from computer vision (e.g., [4–6]) to natural language processing (e.g., [7,8]), often nearing or outperforming humans in complex benchmarking datasets, games of chance and even medical diagnostic [9–11]. The massive deployment of ANNs has been enabled by the successful backpropagation of error (BP) algorithm [12] and is currently backed by significant CPU and GPU resources centralized in cloud servers, which comes in stark contrast with the recent trend for decentralized near-sensor data processing, i.e. *edge computing*, for the internet-of-things (IoT) paradigm [13,14]. Therefore, the development of hardware accelerators supporting ANN architectures within the frame of power- and resource-constrained devices is currently receiving a lot of attention, from both industry and academia [15,16]. However, state-of-the-art designs currently perform basic image recognition¹ while burning about $4\mu\text{J}$ per classification [18], thus lagging orders of magnitude behind the solution found by biological evolution: the brain [19]. Indeed, the *Holy Grail* for distributed computational agents is often seen in the honey bee: with only about one million of neurons and a $10\text{-}\mu\text{W}$ power budget, its brain is able to perform tasks ranging from real-time navigation to complex pattern recognition [20].

Therefore, taking the brain as a guide toward the goal of embedded cognition appears as a natural research direction. It all started in the late 1980s, the term “neuromorphic” was coined by Carver Mead with the discovery that direct emulation of the brain ion channels dynamics could be obtained with the MOS transistor operated in the subthreshold regime [21]. The field of neuromorphic engineering lies at the crossroads of neuroscience, computer science and electrical engineering. It encompasses the study and design of bio-inspired systems following the biological *organization principles* and *information representations*, thus implying a two-fold paradigm shift over conventional computer architectures. Firstly, while conventional

¹As for the CIFAR-10 dataset [17], comprising 10 classes of animal and vehicle images in a format of 32 by 32 pixels.

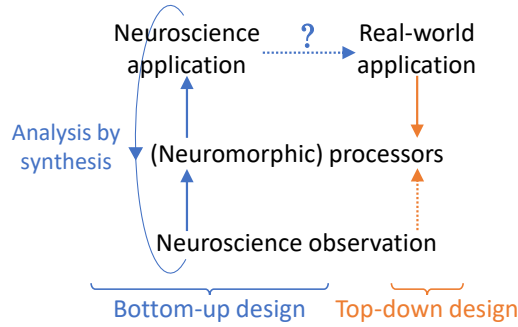


Fig. 1 Summary of bottom-up and top-down neuromorphic design approaches.

von-Neumann processor architectures rely on separated processing and memory, the brain organization principles rely on distributed computation that co-locates processing and memory with neuron and synapse elements, respectively [22]. This first paradigm shift therefore allows releasing the von-Neumann bottleneck in data communication between processing and memory, a point whose criticality is further emphasized by the recent slow down in the pace of Moore’s law, especially for off-chip DRAM memory [23]. Secondly, von-Neumann processor architectures rely on a clocked information representation whose resolution is determined by the number bits used for encoding. On the contrary, the brain processes information by encoding data both in space and time with all-or-none spike events, each single wire potentially encoding arbitrary precision in the inter-spike time interval [24]. This second paradigm shift supports sparse event-driven processing toward a reduced power consumption, especially if spikes are used from sensing to computation. The granularity at which these paradigm shifts can be realized in actual neuromorphic hardware depends on the implementation choices and the design strategy that is followed, the latter being of two types: either *bottom-up* or *top-down*.

I.1 Bottom-up neuromorphic design strategy

The vast majority of neuromorphic designs follow a *bottom-up* strategy (Fig. 1), which is also the historic one adopted since the first neuromorphic chips from the late 1980s. It takes its roots in neuroscience observations and then attempts at (i) replicating these observations *in silico* and (ii) integrating them at scales ranging from hundreds or thousands [25–29] to millions of neurons [30–34]. Integrations reaching a billion neurons are at reach when racks of neuromorphic chips are assembled as a supercomputer, envisioning the simulation in real time of about 1% of the human brain [35]. Bottom-up approaches thus allow designing experimentation platforms that cover acceleration of neuroscience simulations [30], brain reverse-engineering through *analysis by synthesis* [22, 36] and even exploration of hybrid setups between biological and artificial neurons [37, 38]. Their application to brain-

machine interfaces [39, 40] and closed sensorimotor loops for autonomous cognitive agents [41–43] is also under investigation. However, the inherent difficulty of bottom-up approaches lies in applying the resulting hardware to real-world problems beyond the scope of neuroscience, a point that is further emphasized by the current lack of appropriate and widely-accepted neuromorphic benchmarks [44].

The abstraction level chosen on the neuroscience side ranges from the biophysics of ion and protein channels to simplified functional descriptions. On the one hand, mixed-signal implementations with core analog computation and digital spike distribution lie in close analogy to the inner workings of the brain. It especially holds for subthreshold analog designs, which are optimal for low-level biophysical abstractions of neuroscience thanks to the direct emulation property between the hardware and the biophysics [21, 45]. Key mixed-signal designs include BrainScaleS from Heidelberg University [30], ROLLS and DYNAPs from the Institute of Neuroinformatics in Zürich [27, 29] and Neurogrid from Stanford [32]. On the other hand, purely digital designs, which grew in popularity over the last decade, step away from mixed-signal brain-like computation and thus appear to be particularly efficient for high-level functional abstractions of neuroscience, i.e. cognitive computing. Key digital realizations include SpiNNaker from the University of Manchester [31], TrueNorth from IBM [33] and Loihi from Intel [34]. Most of the bottom-up designs are currently being sold to universities for basic research.

Each design opts for a different tradeoff between biophysical accuracy, versatility and implementation efficiency. Optimizing this tradeoff appears as especially critical for the synapses, given that biological fan-ins range from 100 to 10k synapses per neuron [46]. As the synapses are the limiting factor in neuromorphic hardware, in order to reach large-scale integration, designers often either drop the key feature of *synaptic plasticity* (e.g., [29, 33]), which comes at the expense of adaptability and robustness to uncontrolled environments, or move synaptic resources off-chip (e.g., [31, 32]), which comes at the expense of an increase in the system power and latency. By embedding configurable synaptic plasticity in a large-scale high-density design, Loihi took a first step in the right direction [34]. However, a systematic analysis of the tradeoffs of spike-based synaptic plasticity has not yet been carried out: numerous exploration tracks are thus left open to improve the power and density efficiencies of neuromorphic designs. Therefore, the first question that this thesis focuses on is

Question 1 *How can a wide diversity of biophysical behaviors be embedded in silico without precluding an efficient large-scale integration?*

I.2 Top-down neuromorphic design strategy

The top-down neuromorphic design strategy is recently gaining in popularity and attempts at answering the key difficulty of bottom-up approaches in tackling

real-world problems efficiently, beyond neuroscience-oriented applications (Fig. 1). Taking inspiration from machine learning accelerators, top-down design strategies (i) start from the applicative problem and their related algorithms, (ii) investigate how to release constraints in order to make these algorithms hardware- and biophysically-aware, and (iii) proceed with the hardware integration. Given that *synaptic plasticity* has been highlighted by bottom-up strategies as being a key challenge, several neuromorphic accelerators have been designed with the specific objective of embedding online learning for image recognition at the edge [47–50]. However, top-down designs currently appear to be either spiking neural networks (SNNs) with event-driven processing at the expense of accuracy [47–49] or binary neural networks (BNNs) with high accuracy at the expense of conventional frame-based processing [50], thus excluding sparse event-driven sensing.

Although restricted to frame-based BNN operation, the approach from Park *et al.* is particularly interesting [50]: it proposes a silicon implementation in line with the recent body of work that aims at releasing the constraints of BP toward higher biophysical plausibility and hardware efficiency. The first key constraint is the symmetry requirement between the forward and backward weights, known as the *weight transport problem* [51], which implies dedicated inter- and intra-layer communication channels [52]. The second key constraint is *update locking*, as backpropagation updates cannot take place before the full forward and backward passes have been completed [53, 54]. It has been shown that weight transport and update locking preclude biological plausibility [51, 55], resulting from the non-locality in both weight and gradient information. However, to the best of our knowledge, no algorithm releasing both of these constraints for multi-layer training has been proposed to date. Therefore, the second question that this thesis focuses on is

Question 2 *How can a weight-transport-free and update-unlocked training be carried out toward efficient online-learning neuromorphic processors?*

Finally, in order to both better understand the brain and design more efficient neuromorphic hardware, we believe that bottom-up and top-down design strategies are not mutually exclusive, but instead should be used in combination. A first attempt has been made by Chen *et al.* with an SNN embedding bio-inspired spike-timing-dependent plasticity (STDP) that can also be programmed as a BNN using offline-trained weights [56]. A second example is the Tianjic chip, whose architecture allows cointegrating both artificial neural network and spiking neural network processing [57]. Finally, another approach is followed with the BrainDrop neuromorphic chip [58]. It is a top-down design aiming at a one-to-one compatibility with the neural engineering framework (NEF) [59], which models complex neural ensembles as dynamical systems, while leveraging the experience acquired with the previous-generation Neurogrid bottom-up design [32]. Therefore, the third question that this thesis investigates is

Question 3 *What are the synergies between bottom-up and top-down design strategies toward bringing the efficiency of online-learning spike-based neuromorphic systems closer to their biological counterparts?*

I.3 Thesis outline

In order to answer the questions outlined above, this thesis is divided into two parts. The first part follows a bottom-up design strategy: it starts with the neuron and synapse building blocks (Chapter 1) and then proceeds with silicon integration (Chapter 2). The second part follows a top-down design strategy: given the applicative problem of pattern recognition for edge computing, it starts with the training algorithm development (Chapter 3) and then proceeds with the design of a neuromorphic accelerator (Chapter 4). The thesis outline is therefore as follows.

Chapter 1 Both as a first step of a bottom-up approach to neuromorphic engineering and as an introduction to the fundamentals of neuroscience, we start by covering the key aspects, features and functions of neurons and synapses. First, we focus on the *neurons* and give an overview of the main spike-based neural codes, which justifies the need for a high versatility in the neuronal dynamics and behaviors to go beyond a spike-inefficient rate coding. We then survey the main neuron models and compare the efficiency of their silicon implementations, highlighting that mixed-signal neurons currently offer the highest versatility-efficiency tradeoff. We show that, in digital design, stepping back from mathematical equation solvers with a phenomenological implementation of the key neuron behaviors allows reaching a favorable versatility-efficiency tradeoff compared to mixed-signal neurons. We demonstrate this claim by embedding the 20 Izhikevich behaviors of cortical spiking neurons in a standalone area of $466\mu\text{m}^2$ in 28-nm FDSOI CMOS. Second, we focus on the *synapses*. Synaptic plasticity is a key requirement in order for neuromorphic devices to collect knowledge and adapt to new features in uncontrolled environments, while avoiding the overhead of an off-chip learning engine. However, embedding plasticity *in silico* is also a tough challenge due to the high biological fan-ins of 100 to 10k synapses per neuron. We first survey the main plasticity mechanisms and then focus on a comparison of two spike-based learning rules, namely spike-timing-dependent plasticity (STDP) and spike-dependent synaptic plasticity (SDSP), applied to digital design. We demonstrate that the latter can be embedded in a standalone area of only $15.7\mu\text{m}^2$ per synapse in 28-nm FDSOI CMOS. Moreover, digital SDSP opens the doors for high-density large-scale integration by allowing for time-multiplexed designs based on single-port foundry SRAMs. These results lay the grounds for an efficient neuron and synapse integration in neuromorphic processors. The contents of this chapter are adapted from [CP1] and [CP2] for the neuron and synapse building blocks, respectively.

Chapter 2 As a second step of the bottom-up approach, we investigate silicon integration of neurons and synapses. We first identify the main trends, implementation strategies and use cases by surveying the key bottom-up neuromorphic designs, referred to as *experimentation platforms*. We identify that edge computing is a promising use case for small- to medium-scale neuromorphic designs, for which we propose two chips. First, ODIN is a 256-neuron 64k-synapse neuromorphic proces-

sor prototyped in 28-nm FDSOI CMOS. Using time-multiplexed versions of the phenomenological neuron and SDSP synapse models derived in Chapter 1, ODIN demonstrates record neuron and synapse densities in a silicon area of 0.086mm^2 , while maintaining a high energy efficiency with 12.7pJ per synaptic operation (SOP). We then extend these claims with MorphIC, a quad-core 65-nm neuromorphic processor embedding 2k neurons and more than 2M synapses in a silicon area of 2.86mm^2 . We propose a stochastic SDSP (S-SDSP) synapse to further improve density by moving synaptic weights to binary resolutions, while scalability is ensured with a hierarchical routing infrastructure that can accommodate large-scale multi-chip setups. We highlight the efficiency of ODIN and MorphIC by carrying out a detailed comparative analysis of neuromorphic experimentation platforms along the key figures of merit of density, energy efficiency, versatility and flexibility. Finally, we further discuss the aspect of scalability and the relevance of local bio-inspired spike-based learning rules for real-world applications, which are difficultly addressed by bottom-up designs. The contents of this chapter are adapted from [JP1] and [JP2, CP3] for the ODIN and MorphIC neuromorphic processors, respectively, and aim at answering the first question raised in this thesis: “How can a wide diversity of biophysical behaviors be embedded *in silico* without precluding an efficient large-scale integration?”.

Chapter 3 Both as a first step of a top-down approach to neuromorphic engineering and as an introduction to the fundamentals of machine learning for adaptive edge computing, we start by studying the backpropagation of error (BP) algorithm. Two key issues currently preclude biological plausibility and hardware efficiency: (i) symmetry is required between forward and backward weights, which is known as the weight transport problem, and (ii) updates are locked before both the forward and backward passes have been completed. We show that using only the error sign is sufficient to provide learning in the hidden layers. As in classification problems the error sign information is already contained in the targets (i.e. one-hot-encoded labels), using the latter as a proxy for the error brings three advantages: (i) it eliminates the requirement for an explicit feedback pathway (i.e. no weight transport), (ii) it allows for weight updates to be computed in each layer independently as soon as the layer outputs have been evaluated (i.e. no update locking), and (iii) it allows relying on low-cost local and feedforward computation by evaluating layerwise loss gradients as a label-dependent random vector selection (i.e. relaxed computational, structural and memory requirements). We therefore propose the direct random target projection (DRTP) algorithm and demonstrate on the MNIST and CIFAR-10 datasets that, despite the absence of an explicit error feedback, DRTP provides a tradeoff between accuracy and computational cost that is suitable for edge computing applications. The contents of this chapter are adapted from [JP3], currently an arXiv preprint jointly first-authored with Martin Lefebvre.

Chapter 4 As a second step of the top-down approach, we investigate the application-driven design of neuromorphic accelerators. We first survey existing top-down designs, highlighting two trends: spiking neural networks with event-driven

processing at the expense of accuracy and binary neural networks with high accuracy at the expense of conventional frame-based processing. In order to overcome these limitations, we propose SPOON, an event-driven convolutional neural network (eCNN) in 28-nm FDSOI CMOS leveraging two key aspects. First, by embedding online learning with only 16.8-% power and 11.8-% area overheads, we show how the low computational and memory requirements of the DRTP algorithm derived in Chapter 3 allow for efficient silicon implementations. Second, we demonstrate how combining event-driven with frame-based computation (i) allows leveraging the sparsity of event-driven sensors and (ii) ensures maximum data reuse. On MNIST, with 313nJ/inference at accuracies of 95.3% (online learning) or 97.5% (offline training) for an area of 0.32mm² in 28-nm FDSOI CMOS, we demonstrate that SPOON reaches the efficiency of conventional machine learning accelerators while embedding on-chip learning and being compatible with event-based sensors, a point that we further emphasize with N-MNIST benchmarking. The contents of this chapter are adapted from [CP4] and aim at answering the second question raised in this thesis: “How can a weight-transport-free and update-unlocked training be carried out toward efficient online-learning neuromorphic processors?”.

Conclusions and perspectives Finally, we summarize the answers that we propose to the first two central questions of this thesis, while we outline the resulting research perspectives as an answer to the last central question, namely “What are the synergies between bottom-up and top-down design strategies toward bringing the efficiency of online-learning spike-based neuromorphic systems closer to their biological counterparts?”.

Part I

Bottom-Up Neuromorphic Design

*We can't afford that all of
our research is devoted to
the machine: what we are
trying to learn about isn't
the machine that we are
building, it's the brain.*

Misha Mahowald

Building blocks – Neurons and synapses as adaptive processing and memory elements

Contents

1.1	Neurons	14
1.1.1	Overview of neural codes	15
1.1.2	Comparison of silicon neuron implementations	16
1.1.3	Proposed neuron architecture	18
1.1.4	Simulation and implementation results	22
1.2	Synapses	23
1.2.1	Overview of spike-based synaptic plasticity learning rules	24
1.2.2	Comparison of STDP- and SDSP-based silicon synapse implementations	26
1.2.3	Simulation and implementation results	29

Abstract Both as a first step of a bottom-up approach to neuromorphic engineering and as an introduction to the fundamentals of neuroscience, we start by covering the key aspects, features and functions of neurons and synapses. First, we focus on the *neurons* and give an overview of the main spike-based neural codes, which justifies the need for a high versatility in the neuronal dynamics and behaviors to go beyond a spike-inefficient rate coding. We then survey the main neuron models and compare the efficiency of their silicon implementations, highlighting that mixed-signal neurons currently offer the highest versatility-efficiency tradeoff. We show that, in digital design, stepping back from mathematical equation solvers with a phenomenological implementation of the key neuron behaviors allows reaching a favorable versatility-efficiency tradeoff compared to mixed-signal neurons. We demonstrate this claim by embedding the 20 Izhikevich behaviors of cortical spiking neurons in a standalone area of $466\mu\text{m}^2$ in 28-nm FDSOI CMOS [CP1]. Second, we focus on the *synapses*. Synaptic plasticity is a key requirement in order for neuromorphic devices to collect knowledge and adapt to new features in uncontrolled environments, while avoiding the overhead of an off-chip learning engine. However, embedding plasticity *in silico* is also a tough challenge due to the high biological fan-ins of 100 to 10k synapses per neuron. We first survey the main plasticity mechanisms and then focus on a comparison of two spike-based learning rules, namely spike-timing-dependent plasticity (STDP) and spike-dependent synaptic plasticity (SDSP), applied to digital design. We demonstrate that the latter can be embedded in a standalone area of only $15.7\mu\text{m}^2$ per synapse in 28-nm FDSOI CMOS [CP2]. Moreover, digital SDSP opens the doors for high-density large-scale integration by allowing for time-multiplexed designs based on single-port foundry SRAMs. These results lay the grounds for an efficient neuron and synapse integration in neuromorphic processors.

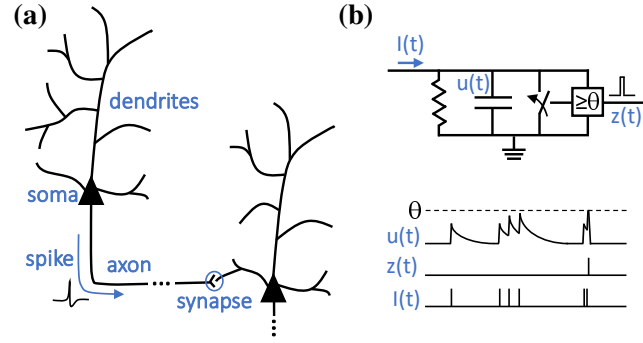


Fig. 1.1 Simplified neuronal structure and operation. (a) Neurons can be decomposed into dendrites, soma and axon. The locus of axon-to-dendrite neuron-to-neuron connection is the synapse. Adapted from [60]. (b) Simplified first-order leaky integrate-and-fire (LIF) model abstracted as an RC filter. The dendritic currents are denoted as $I(t)$, while the sub- and above-threshold parts of the membrane potential¹ are denoted as $u(t)$ and $z(t)$, respectively.

1.1 Neurons

As the key computational elements of biological systems, the *neurons* carry out nonlinear transformations of their inputs, both in space and time, and are divided into three stages (Fig. 1.1(a)) [60]: the *dendrites* act as an input stage, the core computation takes place in the *soma* and the outputs are transmitted along the *axon*, which connects to dendrites of other neurons through *synapses* (Section 1.2). The neural tissue also contains glial cells, which are believed to take a structuring and stabilizing role [61], but whose study is beyond the scope of this thesis. One of the simplest neuron models, which originates from the work of Louis Lapicque in 1907 [62], describes biological neurons as *integrating* synaptic currents into a membrane potential and *firing* a spike (i.e. action potential) when the membrane potential exceeds a firing threshold, after which the membrane potential is reset. It is thus referred to as the *integrate-and-fire* (I&F) model, while the addition of a leakage term leads to the *leaky integrate-and-fire* (LIF) model, which emphasizes the influence of recent inputs over past activity [63]. This basic linear-filter operation can be modeled by an RC circuit, which is illustrated in Fig. 1.1(b). The widespread I&F and LIF models are *phenomenological* models: they aim at computational efficiency while exhibiting, from an input/output point of view, a restricted repertoire of biophysical behaviors chosen for their prevalence or relevance for a specific application. On the other end of the neuron models spectrum, *conductance-based* models aim at a faithful correspondence with the biophysics of biological neurons. The Hodgkin-Huxley (H&H) model [64] provides the highest accuracy but is computationally-intensive as it consists of four nonlinear ordinary differential equations. The Izhikevich model is a two-dimensional reduction of the H&H model [65] that can still capture the 20 main behaviors of biological

¹ These terms are related to the neuron firing threshold θ and are not to be confused with sub- or above-threshold analog design, which is related to the transistor threshold voltage V_t .

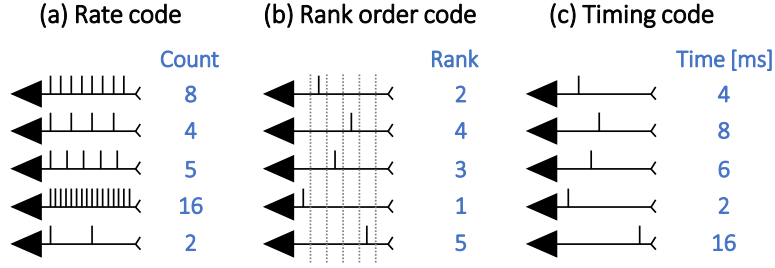


Fig. 1.2 Main encodings in spiking neural networks, as defined in [24]. (a) Conventional rate code, easy to use and accurate but inefficient in its spike use. (b) Rank order code, efficient in its spike use but with a limited representational power. (c) Timing code in the specific case of time-to-first-spike encoding, both efficient in its spike use and accurate, illustrated for an arbitrary resolution of 1ms.

spiking neurons found in the cortex [66]. The adaptive-exponential (AdExp) two-dimensional model is similar to the Izhikevich model and differs only by the spike mechanism [67]. Therefore, the tradeoff between biophysical accuracy, versatility and implementation efficiency of silicon neurons is defined by the biophysical description level (i.e. phenomenological or conductance-based) and by the underlying model. We refer the reader to [66] for a detailed neuron model summary.

In this section, we propose a phenomenological digital neuron that exhibits the full repertoire of the 20 Izhikevich behaviors with a low silicon area, making it suitable as an experimentation platform to investigate the computational primitives of the brain. The neuron is compatible with time multiplexing, has a configurable synaptic fan-in, can operate on a wide range of time scales from biological- to accelerated-time and occupies a standalone area of $22.3\mu m \times 20.9\mu m$ in 28-nm FDSOI CMOS, which is more compact than current analog and digital implementations. An overview of the main neural codes is provided in Section 1.1.1 in order to highlight why a large behavior repertoire is important for versatile experimentation platforms. We then provide an overview of silicon neuron implementations and compare them using the number of Izhikevich behaviors and the silicon area as metrics (Section 1.1.2). The architecture of the proposed neuron and its main features are then presented in Section 1.1.3, while pre-silicon results are shown in Section 1.1.4 for the validation of the 20 Izhikevich behaviors. A time-multiplexed version of this neuron will be used in the ODIN neuromorphic processor, presented in Chapter 2.

1.1.1 Overview of neural codes

In order to leverage the sparsity of spike-based processing, time must be embedded into computation [68]. As the standard LIF neuron model only behaves like a leaky integrator, it lacks the ability and behavior repertoire to efficiently compute using temporal information [69]. Therefore, the LIF model is usually restricted to the use of the *rate code* (Fig. 1.2(a)), a standard spike coding approach directly mapping con-

tinuous values into spike rates [24]. It is a popular code due to its simplicity, which also allows for straightforward mappings from artificial neural networks (ANNs) to spiking neural networks (SNNs) [70], at the expense of a high power penalty as each spike only encodes a marginal amount of information. This aspect can be mitigated by the use of the *rank order code* (Fig. 1.2(b)), where data is encoded as a sequence of successive spikes without involving relative timing: each neuron spikes once, in the order of decreasing values. We show in Chapter 2 that the rank order code is useful for an early estimation of high-confidence inference in multi-layer networks of LIF neurons for classification problems. However, without a versatile or specialized neuron model, rate coding is still required at the input by repeating spike sequences until an output neuron fires. Output rank order coding thus only partly mitigates the spike coding inefficiency implied by the LIF model.

Therefore, behavior versatility is crucial to explore codes that embed higher amounts of data bits per spike and favor sparsity by leveraging time, such as the *timing code* [24, 71], where the popular *time-to-first-spike* variant encodes information in the time taken by a neuron to fire its first spike (Fig. 1.2(c)). The 20 Izhikevich behaviors of biological cortical spiking neurons offer a variety of ways to capture time into computation [66]. For example, phasic spiking captures the stimulation onset [66] and could be useful for codes relying on the emission of a single spike per neuron [24]. Spike frequency adaptation is useful to encode time since the stimulation onset [66], while both spike frequency adaptation and threshold variability can be used to implement forms of homeostatic plasticity, which allows stabilizing the global network activity [34, 72]. Spike latency can emulate axonal delays, which are useful to induce temporal dynamics in SNNs and to enhance neural synchrony [73], another mechanism believed to increase the representational power of SNNs through population coding [24]. Finally, resonant behaviors allow selectively responding to specific frequencies and spike time intervals, thus enabling the timing code.

1.1.2 Comparison of silicon neuron implementations

Depending on the underlying model, silicon neurons can be designed following two main approaches: phenomenological implementation and biophysical emulation (analog, especially subthreshold) or simulation (digital). An overview of the current state of the art for both analog and digital neurons is provided in Fig. 1.3, illustrating their expressiveness power measured by the number of Izhikevich behaviors they are able to reproduce against their silicon area. Subthreshold [27, 74] and above-threshold [30, 75] analog implementations provide a high expressiveness power for a low area. By their similarity with the Izhikevich model, AdExp neurons are believed to reach the 20 Izhikevich behaviors [76], although it has not been demonstrated. Neuron implementations from [32] and [77] should provide similar tradeoffs, but no information is provided as to their number of Izhikevich behaviors. With a reduced number of behaviors, analog implementations of the Mihalas-Niebur model in [78] and [79] were demonstrated to exhibit 9 and 15 out of the 20 Izhikevich

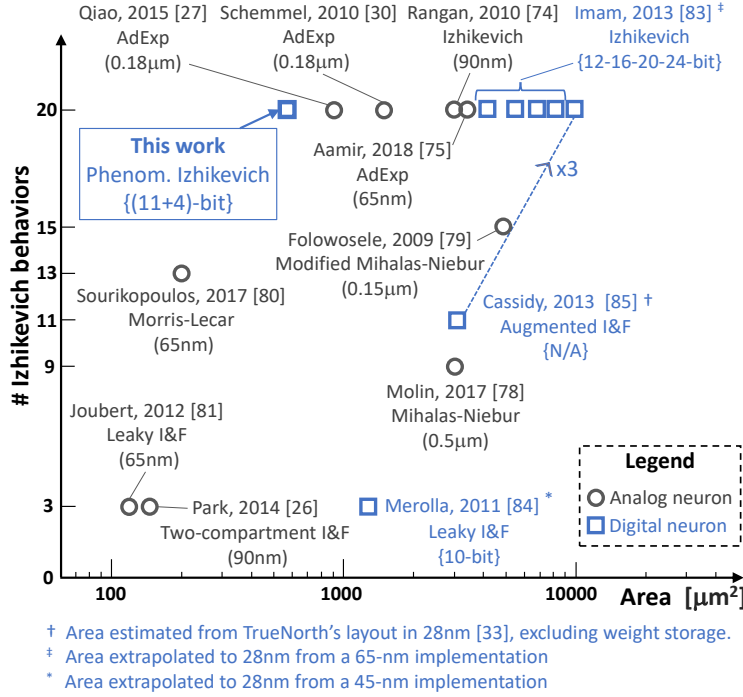


Fig. 1.3 State of the art of analog and digital neuron implementations: expressiveness power against area tradeoff. The area of digital designs has been normalized to a 28-nm node using the node factor. This normalization has not been applied to analog designs as they require redesign to compensate for performance degradation during technology scaling: original area and technology node are reported. All neurons presented in this figure are standalone (i.e. non time-multiplexed), except the one of Cassidy *et al.* for which only the update logic area is reported.

behaviors, respectively. The Morris-Lecar model is also explored in [80] and is believed to reach 13 out of the 20 Izhikevich behaviors [66]. The phenomenological approach is followed in [81] and [26] with LIF neurons, the latter being an extended two-compartment version. However, a favorable versatility-area tradeoff is achieved in analog circuits at the expense of design time, susceptibility to mismatch and noise and requirement for precise bias voltages and currents, whose mitigation can only be handled with specific techniques at both the circuit design and neural network levels [82]. On the other hand, digital designs release these constraints at the expense of going for a simulation approach lying further from the biophysics, thus inducing a high implementation overhead. This is illustrated in the neuron implementation from [83] that solves digitally the differential equations of the Izhikevich neuron model, while the phenomenological approach is followed in [84] with a 10-bit LIF neuron. Between both approaches lies the neuron model of Cassidy *et al.* [85], it is based on a LIF neuron model to which configurability and stochasticity are added. This model is used in the TrueNorth chip [33] and exhibits 11 Izhikevich behaviors, while the 20 behaviors can be reached by coupling three neurons together, showing an interesting expressiveness power against area tradeoff. However, Fig. 1.3 shows that digital designs are still far from reaching the compactness of their analog counterparts.

While Fig. 1.3 shows standalone neuron implementations, time multiplexing allows reducing the neuron area at the expense of an increase in control complexity and of a reduction in the maximum update rate. It is achieved by sharing the neuron update resources and by offloading the states to an SRAM for digital designs (e.g., as in [33]) or to capacitors for analog designs (e.g., as in [26,28]). Therefore, the choice between time-multiplexed neurons and fully-parallel standalone neurons relies on the target application: the former is preferred for low-cost large-scale integration of neuromorphic experimentation platforms, while the latter is preferred for fully-parallel biological-to-accelerated-time execution of large networks. It therefore highlights the need for a digital neuron that is suitable for both time-multiplexed and fully-parallel yet low-area networks.

1.1.3 Proposed neuron architecture

In order to bridge the area gap between the digital and analog approaches, we propose the neuron architecture shown in Fig. 1.4. It allows reproducing the 20 Izhikevich behaviors while being entirely event-driven, the state is updated only upon reception of events from pre-synaptic neurons or of time reference triggers. This allows drastically reducing the computational overhead compared to models such as the H&H and Izhikevich, which require solvers for coupled nonlinear differential equations, and thus updates at each integration timestep. Piecewise linear approximations have been proposed to save the overhead of dedicated solvers (e.g., [86,87]), however they still require an update of all neuron states after each integration timestep, which induces high computational and memory access overheads, especially at accelerated time constants. While Fig. 1.4 shows a standalone design, event-driven operation also supports time multiplexing: the neuron update logic can easily be shared among neurons whose state is stored in SRAM. Therefore, the proposed neuron implements the behaviors and main functionalities of the Izhikevich model at low area and computational costs, not its detailed dynamics. As shown in Fig. 1.4, the circuit architecture follows a synchronous digital design. While asynchronous design is attractive to avoid a clock tree overhead and to lie closer to the asynchronous event-driven processing taking place in the brain, a lack of efficient tool support currently restrains the design of large-scale digital asynchronous circuits. Indeed, the large-scale digital neuromorphic experimentation platforms of TrueNorth [33] and SpiNNaker [31] are based on a globally-asynchronous locally-synchronous (GALS) design, where clocked cores communicate using asynchronous spike/packet routing protocols. One such standard protocol is the address-event representation (AER) [88,89], a popular four-phase-handshake spike routing bus. Only the Intel Loihi is a fully-asynchronous chip, backed by a team experienced in asynchronous logic design. Therefore, in this thesis, we will avoid the design time and complexity overhead of asynchronous design by keeping a standard clocked approach. However, we will maintain compatibility with asynchronous AER-based off-chip communication for the neuromorphic processors proposed in Chapters 2 and 4.

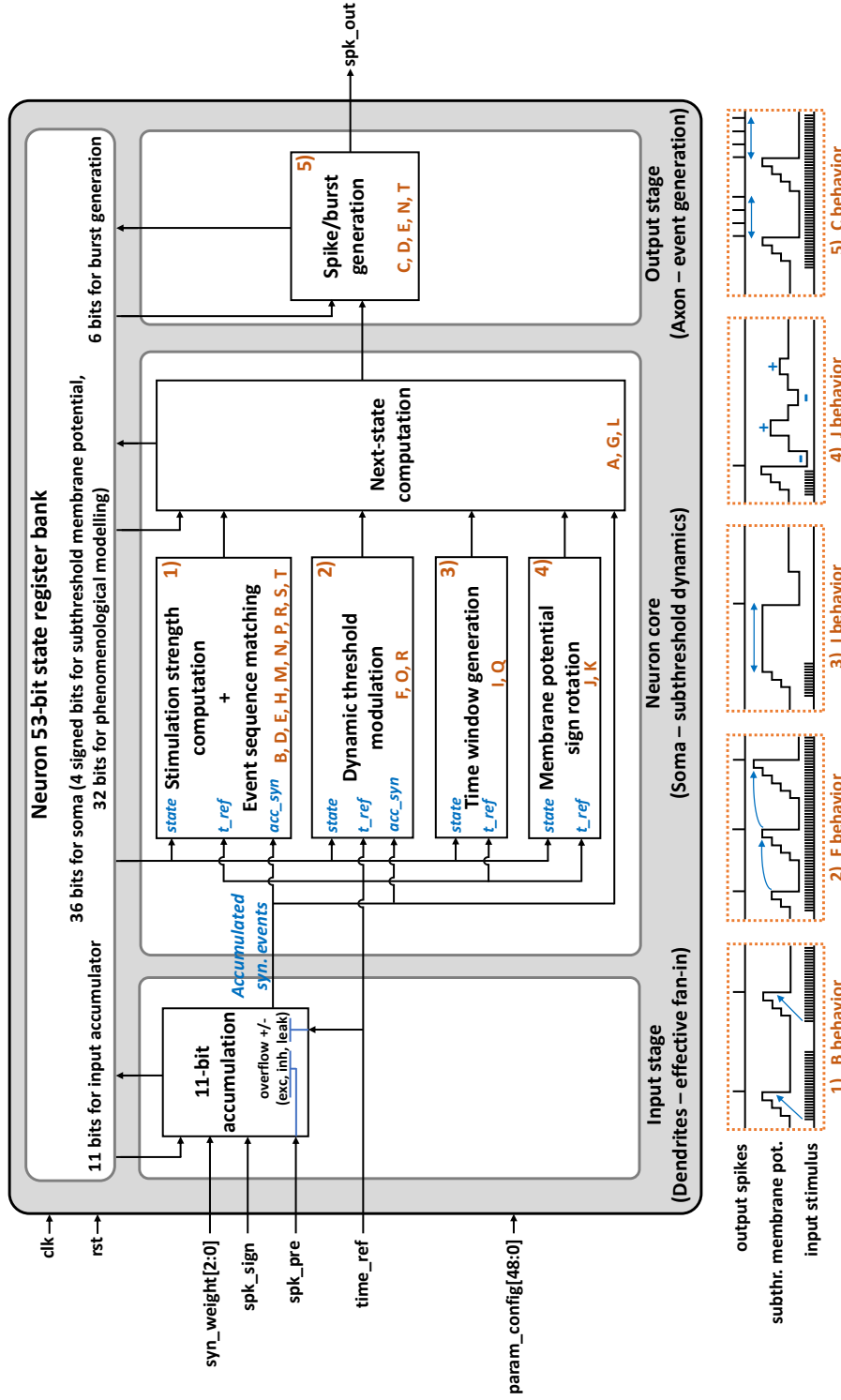


Fig. 1.4 Three-stage architecture of the proposed phenomenological digital neuron. One input event (i.e. pre-synaptic neuron spike event or time reference trigger) and the associated state update can be handled at each clock cycle. Letters inside the different combinational logic blocks indicate the Izhikevich behaviors that are phenomenologically generated, following the numbering convention provided in Fig. 1.5. A sample behavior illustration is also provided for each combinational logic block to illustrate their functionality on a practical case, as outlined in blue in the behavior illustrations.

The input stage of the proposed neuron can be assimilated to dendrites. Spikes from pre-synaptic neurons are signaled with the `spk_pre` input, which indicates when data on the `spk_sign` and the `syn_weight` inputs are valid: the former represents the event sign to handle inhibitory and excitatory pre-synaptic neurons, the latter contains the 3-bit weight of the associated synapse. The synaptic contributions are handled in an accumulator: its effective depth is configurable based on the input parameter array `param_config`. This configurability allows for an inversely proportional scaling of the relative synaptic weight contributions to the neuron membrane potential with the synaptic fan-in. In order to accommodate for fan-ins of up to 10^3 synapses per neuron, an 11-bit accumulator depth combined with 4-bit subthreshold dynamics in the neuron core allows maintaining orders of magnitude similar to those found in biology [46]. The `time_ref` trigger acts as a time reference for the different behaviors and allows neurons to operate on time scales ranging from biological- to accelerated-time depending on its frequency of occurrence. Beyond defining time constants, the time reference trigger also induces leakage in the 11-bit input accumulator, where the leakage strength is configurable. It is typically issued as a broadcast event to a neuron group.

The neuron core can be assimilated to the soma. Following the *spike* or *pulse* model assumption for cortical neurons, information lies in spike timings, not in spike shapes [46]. Therefore, only subthreshold dynamics are modeled in the neuron core in order to dedicate resolution bits to the informative part of the neuron membrane potential, similarly to the LIF abstraction shown in Fig. 1.1(b). Subthreshold dynamics are driven by four blocks that capture the functionalities necessary to phenomenologically describe the 20 Izhikevich behaviors, as indicated in Fig. 1.4 by letters corresponding to the behavior numbering of Fig. 1.5. These blocks are stimulated by accumulated synaptic events resulting from positive and negative overflows of the input stage accumulator, while all time-dependent computations are triggered by time reference events. The first block computes the strength and duration of the ongoing input stimulation to assess if specific stimulation sequences are matched, which captures phasic, mixed, class 2, rebound, bistability, accommodation and inhibition-induced behaviors. The second block allows the neuron firing threshold to depend on the input stimulation or on the output firing patterns, which captures threshold variability and spike frequency adaptation behaviors. The third block of time window generation allows delaying some neuron operations like firing and membrane potential resetting, which captures spike latency and depolarizing after-potential behaviors. Beyond emulating axonal delays with the spike latency behavior, refractory delays are also implemented with a configurable duration. The fourth block allows rotating the sign of the subthreshold membrane potential in order to capture the subthreshold oscillation and resonator behaviors. The activation, extent and effects of all four blocks used in phenomenological modeling are configured using the neuron parameters in `param_config`.

Finally, the neuron output stage can be assimilated to the axon. Upon each threshold crossing of the membrane potential in the neuron core, the latter sends to the neuron output stage the target number of spikes and inter-spike interval to be

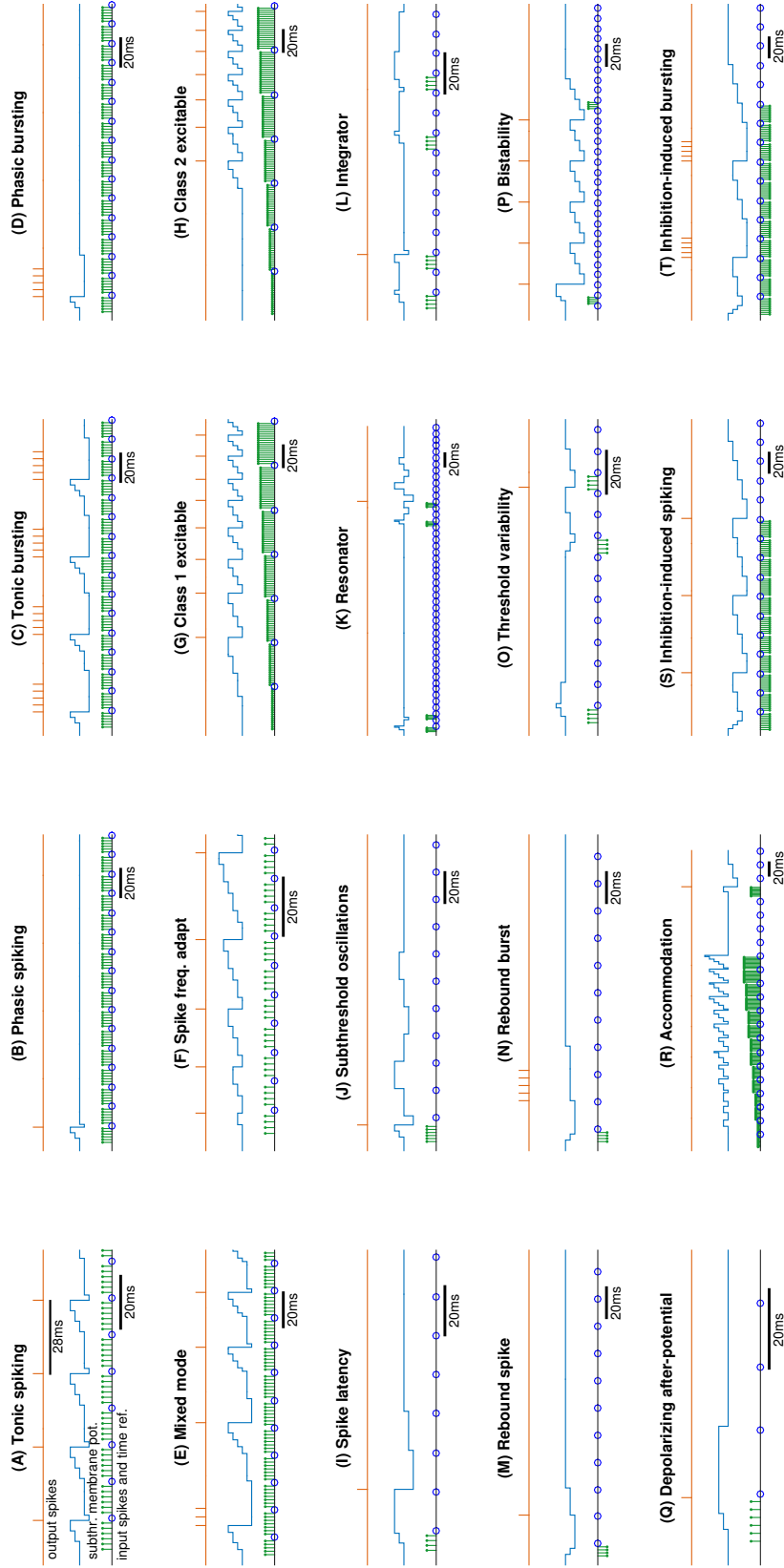


Fig. 1.5 Post-layout simulations for the 20 Izhikevich behaviors with biological time constant. Positive spikes represent inputs from excitatory neurons, while negative spikes represent inputs from inhibitory neurons. A 20-ms scale reference is provided next to each of the 20 behaviors and closely match those of the original Izhikevich behaviors figure in [66]. A clock frequency of 1kHz has been used for all behaviors.

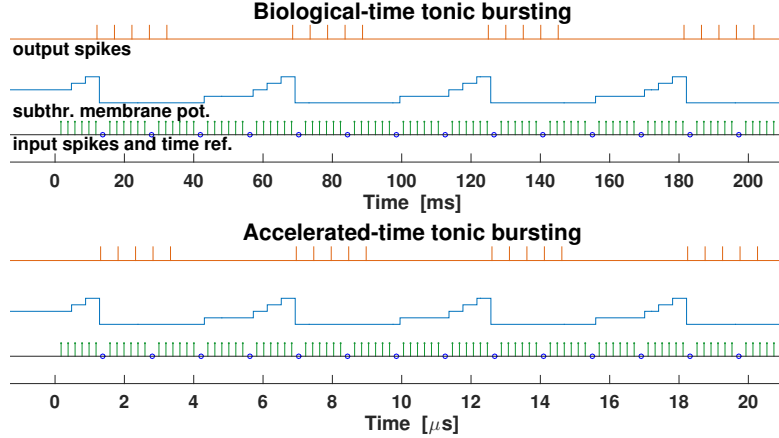


Fig. 1.6 Biological-time (up, 1-kHz clock) and accelerated-time (down, 10-MHz clock) tonic bursting, illustrating the configurable time constant of the neuron.

generated on the `spk_out` output line, both of which are adjustable with the neuron parameters `param_config`. Therefore, spiking and bursting can be reached independently from the subthreshold dynamics and handled in a separate output stage. As the burst time constants are usually much shorter than the neuron time reference events used to generate the 20 behaviors (Fig. 1.5), the output stage generates configurable burst timings based on the frequency of the input clock `clk`.

1.1.4 Simulation and implementation results

In order to allow for a direct comparison with other standalone neuron models reported in Fig. 1.3, we synthesized the proposed phenomenological digital neuron in 28-nm FDSOI CMOS from digital standard-cell libraries. The results presented in this section are obtained from post-layout simulations, while the time-multiplexed and Calcium-extended version of this neuron has been prototyped in the ODIN neuromorphic processor, presented in Chapter 2. Fig. 1.5 shows that the proposed neuron successfully exhibits each of the 20 Izhikevich behaviors: output spike patterns, subthreshold dynamics, test stimuli and input neuron time reference are illustrated. The formatting of Fig. 1.5 and the different time constants of the behaviors, indicated by a 20-ms scale reference next to each behavior, accurately match those of the original Izhikevich figure in [66].

In order to highlight the configurable time constant of the neuron, Fig. 1.6 shows the tonic bursting behavior operated at biological time (top) and with an acceleration factor of 10^4 (bottom). As the proposed neuron is entirely event-driven, time constant scaling was in this case carried out by scaling the input event rates and adapting the clock frequency consequently. Depending on the selected neuron behaviors, other levers are the neuron time reference and configuration parameters.

Table 1.1 Proposed phenomenological neuron characteristics.

Number of Izhikevich behaviors	20	
Technology	28-nm FDSOI CMOS	
Standalone post-shrink layout area	$22.3\mu m \times 20.9\mu m$	
Time constant	Biological to accelerated	
Supply voltage	0.8V	0.4V
Leakage power	49nW	12nW
Dynamic energy per input event	280fJ	64fJ
Max. frequency and input event rate	524MHz	16.9MHz

The specifications of the proposed neuron are summarized in Table 1.1. Using 8-track LVT libraries with upsized gate length (poly biasing technique by 16nm), the neuron occupies a post-shrink area of only $22.3\mu m \times 20.9\mu m$ with 694 logic gates in 28-nm FDSOI CMOS. Energy and speed performance is provided for supply voltages of 0.8V and 0.4V. The leakage power goes from 49nW to 12nW, while the consumed energy per input event under tonic spiking goes from 280fJ to 64fJ, at 0.8V and 0.4V respectively. The power consumption results at 0.4V are competitive with other neuron implementations, including analog ones (e.g., see [78] for a review). Finally, the maximum operating frequency, which also corresponds to the maximum input event rate, goes from 524MHz to 16.9MHz, at 0.8V and 0.4V respectively.

1.2 Synapses

Biological synapses embed the functions of memory and plasticity in extremely dense elements [22], allowing neurons to connect with fan-in values ranging from 100 to 10k synapses per neuron [46]. Density is thus a key requirement in the design of silicon synapses as they often dominate the area of neuromorphic processors, sometimes by more than one order of magnitude [27]. While releasing the requirement for plasticity allows keeping the synapse complexity to a minimum (e.g., as in the IBM TrueNorth chip [33]), retaining embedded online learning is important for three reasons. First, it allows low-power autonomous agents to collect knowledge and adapt to new features in uncontrolled environments, where new training data is presented on-the-fly in real time [20]. Second, in an attempt to close the loop of bottom-up neuromorphic design through analysis by synthesis (Fig. 1), it is necessary to develop biophysically-realistic silicon synapses embedding spike-based plasticity mechanisms for exploring their properties, determining the best application areas and use cases, and postulating how they might operate in the brain and support cognition [45]. Finally, from a computational efficiency point of view, neuromorphic designs deprived from synaptic plasticity rely on off-chip optimizers, thus precluding deployment in applications that are power- and resource-constrained both in the training and the inference phases. Therefore, a careful hardware-aware selection of spike-based synaptic plasticity rules is necessary for the design of efficient silicon synapses.

In this section, we propose a digital synapse with embedded spike-based plasticity requiring only 18 logic gates for an area of $15.7\mu\text{m}^2$ in 28-nm FDSOI CMOS. First, we provide an overview of synaptic plasticity learning rules in Section 1.2.1. Next, we carry out a comparison of two key spike-based learning rules in Section 1.2.2: spike-timing-dependent plasticity (STDP) and spike-dependent synaptic plasticity (SDSP). We highlight how the computational aspects of the latter can be offloaded from the synapses to the neurons, thus allowing for more compact implementations compared to STDP-based synapses. Finally, pre-silicon results are shown in Section 1.2.3 and are compared with state-of-the-art synapse implementations. A time-multiplexed version of the proposed SDSP-based synapse will be used in the ODIN neuromorphic processor, while it will be enhanced with a stochasticity mechanism for operation with binary weights in the MorphIC neuromorphic processor, both presented in Chapter 2.

1.2.1 Overview of spike-based synaptic plasticity learning rules

A wide range of plasticity mechanisms is believed to take place at different timescales in the brain, where it is common to segment them into four types [22, 90–92]. First, *short-term plasticity* (STP) operates over milliseconds and covers adaptive neuronal behaviors (Section 1.1) and short-term synaptic adaptation [93]. A few analog CMOS implementations of STP have been proposed, e.g. in [27, 28]. Second, *long-term plasticity* mechanisms operate over tens to hundreds of milliseconds and cover spike-based plasticity rules, as well as working memory dynamics [94]. An important body of work covers both analog and digital implementations of spike-based plasticity rules, whose scale is ideal to operate on both standard and spiking versions of image and sound classification datasets: long-term plasticity will be the focus of the subsequent text. Third, *homeostatic plasticity* operates over tens to hundreds of seconds and allows scaling synaptic weights to stabilize the neuron firing frequency ranges, and thus the network activity [95]. There is a particular interest for homeostatic plasticity in analog designs so as to compensate for process, voltage and temperature (PVT) variations at the network level [96]. The design of efficient strategies for circuit implementations of homeostaticity is not yet mature: achieving the long homeostatic timescales in analog CMOS design is challenging, although solutions have been proposed for subthreshold design in [72], while it incurs high control and memory access overheads in time-multiplexed digital designs. Finally, *structural plasticity* operates over days to modify the network connectivity [97]. It is usually applied to the mapping tables governing system-level digital spike routers.

Being one of the first formulations of a long-term spike-based plasticity mechanism relying on experimental data derived by Bi and Poo [98], pair-based spike-timing-dependent plasticity (STDP) is a conceptually simple and popular learning rule for silicon synapses [25, 68, 99–103]. STDP is a Hebbian learning rule relying on the relative timing of pre- and post-synaptic spikes occurring at times t_{pre} and

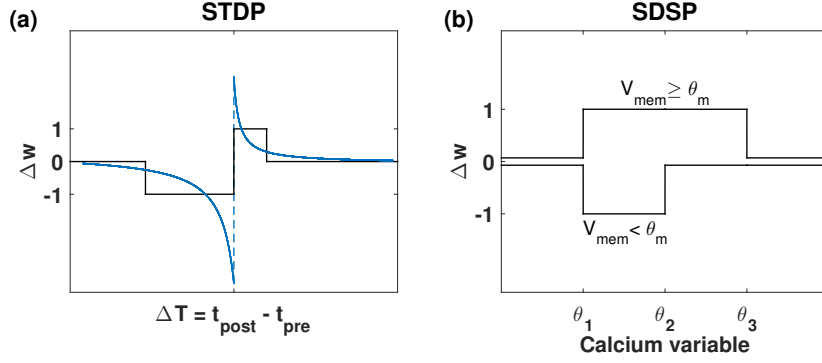


Fig. 1.7 Illustration of the STDP and SDSP spike-based learning rules. In order to highlight their suitability for digital design, the amplitude scaling factors of SDSP and of the digital version of STDP have been normalized for unit weight updates Δw . (a) STDP learning rule (blue) with the popular approximation proposed by Cassidy *et al.* in [105] (black). (b) SDSP learning rule from Brader *et al.* [106].

t_{post} , respectively. STDP strengthens correlation in the pre- and post-synaptic activities by increasing (resp. decreasing) the synaptic weight for causal (resp. anti-causal) orderings between pre- and post-synaptic spikes. It follows an exponential function shown as a blue line in Fig. 1.7(a) and formalized in Eq. (1.1), where $\Delta T = t_{\text{post}} - t_{\text{pre}}$, A_+ and A_- are amplitude scaling parameters and τ_+ and τ_- are time constants [104]. A strong simplification of the STDP shape is proposed by Cassidy *et al.* in [105] for digital implementations and is shown in black in Fig. 1.7(a).

$$\begin{cases} w \rightarrow w + A_+ \exp(-\Delta T / \tau_+) & \text{if } \Delta T > 0 \\ w \rightarrow w - A_- \exp(\Delta T / \tau_-) & \text{if } \Delta T < 0 \end{cases} \quad (1.1)$$

The spike-driven synaptic plasticity (SDSP) learning rule has been proposed by Brader *et al.* in [106] and led to several silicon implementations based on Eq. (1.2) [27, 28, 107, 108]. Instead of relying on relative pre- and post-synaptic spike timings, SDSP computes updates based on the internal state of the post-synaptic neuron at the time of the pre-synaptic spike. If the post-synaptic membrane voltage V_{mem} is above (resp. below) a given threshold θ_m , the synaptic weight undergoes a step increase (resp. decrease) upon the arrival of a pre-synaptic spike, where A_+ and A_- are amplitude scaling parameters. As for STDP, SDSP strengthens correlation between pre- and post-synaptic activities as the membrane potential indicates whether or not the post-synaptic neuron is about to spike. In order to improve the recognition of highly-correlated patterns, Brader *et al.* add a stop-learning rule based on the Calcium concentration Ca of the post-synaptic neuron [106]. The Calcium concentration provides an image of the recent post-synaptic firing activity: if it is beyond average ranges (thresholds θ_1 , θ_2 and θ_3), there is evidence that learning already occurred and that further potentiation or depression is likely to

result in overfitting. The SDSP rule with stop-learning conditions is illustrated in Fig. 1.7(b). Finally, a bistability mechanism [106] is added to the SDSP description of Eq. (1.2): if the synaptic weight is above (resp. below) a given threshold θ_w , it is attracted toward a high (resp. low) state in the long term. This mechanism helps regarding two aspects [109]. First, it prevents undesired fluctuations resulting from spontaneous activity in the network. Second, it results in a natural stochastic selection mechanism for long-term potentiation (LTP) and long-term depression (LTD) as the spike activity in neural networks is often studied as a Poisson distribution [110]. The learning performances of SDSP are similar to those of STDP but present better biophysical accuracy and generalization properties [106], which are however subject to a careful parameter choice.

$$\begin{cases} w \rightarrow w + A_+ & \text{if } V_{\text{mem}}(t_{\text{pre}}) \geq \theta_m, \theta_1 \leq \text{Ca}(t_{\text{pre}}) < \theta_3 \\ w \rightarrow w - A_- & \text{if } V_{\text{mem}}(t_{\text{pre}}) < \theta_m, \theta_1 \leq \text{Ca}(t_{\text{pre}}) < \theta_2 \end{cases} \quad (1.2)$$

Just as the Calcium concentration in SDSP corresponds to a third factor modulating the pre- and post-synaptic activities, several other third-factor learning rules have been proposed, including the Bienenstock-Cooper-Munro (BCM) model [111], the triplet-based STDP [112], and several other variants of STDP and SDSP, e.g. [113, 114], from which the silicon synapse design from [115] is inspired. This idea of adding a modulation factor to input and output activities is also important for top-down approaches designing bio-inspired versions of the backpropagation of error (BP) algorithm [12], as will be demonstrated in Chapter 3. Finally, another approach has been followed in the Intel Loihi neuromorphic chip: in order to support experimentation and analysis by synthesis, a plasticity coprocessor allows embedding configurable spike-based plasticity rules in the synapses [34]. For all approaches, the specific learning rule and resolution selected for the design determines the synapse circuit size, its task-specific learning performance and the memory lifetime of the network as a function of the number of new stimuli received (i.e. the palimpsest property) [109].

1.2.2 Comparison of STDP- and SDSP-based silicon synapse implementations

In order to assess the key differences between the two main spike-based learning rules, namely STDP and SDSP, it is necessary to carry out a detailed comparative analysis of the design tradeoffs between both. More specifically, we will carry out this analysis with a focus on digital design for two reasons. First, several analog designs have been proposed for both STDP, SDSP and their variants (Table 1.2). As technology normalization cannot be directly applied to analog designs and given the widely different technology nodes and implementation strategies at hand (i.e. time-multiplexed or fully-parallel, SRAM- or capacitor-based weight storage),

it is difficult to identify clear trends between analog STDP and SDSP designs, except that the $16\text{-}\mu\text{m}^2$ 28-nm SDSP synapse from Mayr *et al.* appears to be particularly efficient. An analysis of STDP and SDSP synapse implementations within the framework of digital design can thus help for a technology-normalized identification of design tradeoffs. This is further supported by the fact that no digital implementation of SDSP has been proposed to date. Second, as synapses are often a bottleneck in achieving area-efficient neuromorphic designs, especially when embedding online learning, two memory substrates are appealing due to their high density: memristors and single-port foundry SRAMs. On the one hand, memristors promise new density records and recent work (e.g., [116]) shows how the memristor characteristics can be used to emulate biological synapses and to implement stochastic learning while being non-volatile. However, high-yield large-scale co-integration with CMOS has yet to be demonstrated [117, 118], while the aspects linked to synaptic resolution control, mismatch and fabrication costs need to be alleviated. On the other hand, standard single-port foundry SRAMs currently have densities as high as $0.120\mu\text{m}^2/\text{bit}$ in 28-nm FDSOI CMOS [119] or $0.031\mu\text{m}^2/\text{bit}$ in the recent Intel 10-nm FinFET node [120]. Foundry SRAMs are thus an efficient substrate for low-cost synapse array design, which suits well a time-multiplexed digital design approach. However, the memory access patterns required by the considered learning rule might imply the use of custom SRAMs instead of single-port foundry SRAMs [25], thus automatically inducing design time and density penalties as the DRC rules for logic are used instead of the foundry bitcell pushed rules [121]. The suitability of single-port foundry SRAMs is thus an important aspect to be analyzed when comparing digital implementations of STDP and SDSP.

The synchronous N-bit digital STDP implementation proposed by Cassidy *et al.* [105], following the simplification of Fig. 1.7(a), is shown in Fig. 1.8(a). The synapse design is divided into two main parts. First, in order to compute the STDP weight updates Δw , which can take values of 0, +1 or -1, the UP/DOWN decoder requires a shift register inside each synapse for time window generation around pre-synaptic (spk_{pre}) and post-synaptic (spk_{post}) spikes. Second, the weight updates are applied to the state element containing the synaptic weight w_{syn} , currently represented as registers for a standalone parallel synapse implementation. The proposed synchronous N-bit digital implementation of SDSP is shown in Fig. 1.8(b) and differs from the STDP synapse only by its weight update computation in the UP/DOWN decoder. It reveals the key advantage of the SDSP learning rule: the main synaptic computations are offloaded to the post-synaptic neuron, which computes the SDSP learning conditions v_{up} and v_{down} based on Eq. (1.2) for all synapses, where the amplitude scaling factors A_+ and A_- have been fixed to 1. As opposed to STDP, it avoids the overhead of replicating shift registers that, depending on the synaptic fan-in, could amount from 100 to 10k per neuron. The post-synaptic neuron must be extended with a Calcium variable, which can be achieved using a simple output spike counter that is regularly decremented to

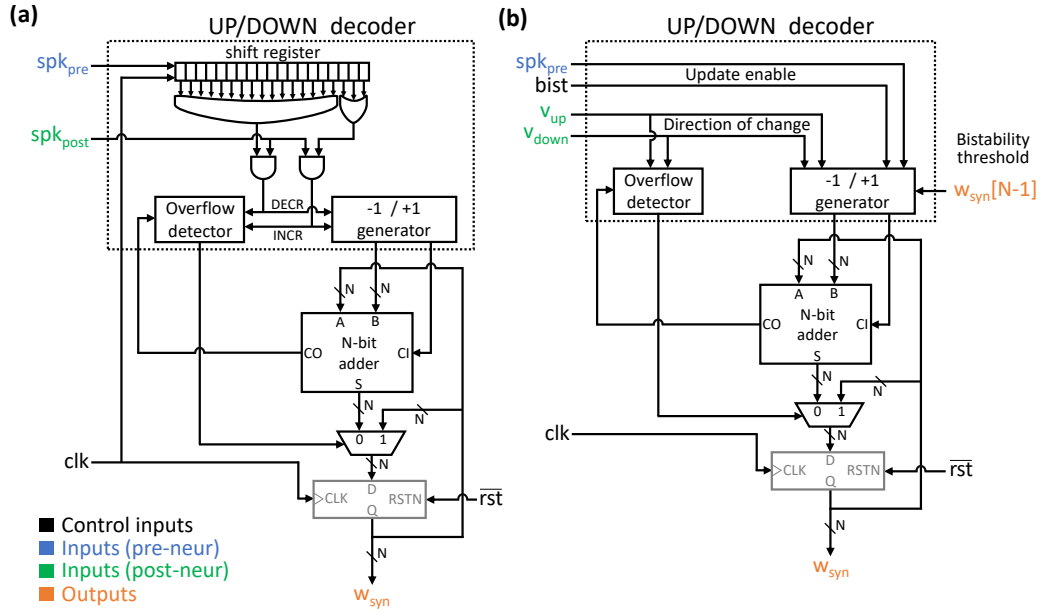


Fig. 1.8 N-bit digital implementations of STDP and SDSP synapses. (a) Digital STDP implementation based on [105], where the time window step is linked to the clock tick. (b) Proposed digital SDSP implementation.

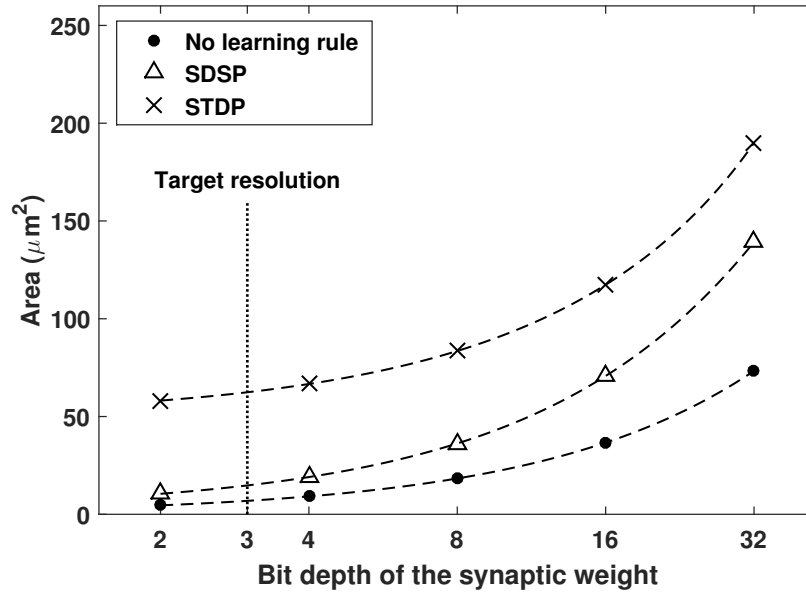


Fig. 1.9 Area comparison of synapses with STDP, SDSP and without learning. The STDP implementation is based on the design by Cassidy *et al.* in [105].

simulate the Calcium leakage mechanism. A trigger signal **bist** is also provided as a time reference for the bistability mechanism, the associated bistability threshold is given by the most significant bit of the synaptic weight w_{syn} .

The STDP and SDSP implementations are compared in Fig. 1.9 for different synaptic weight resolutions, while a synapse without learning rule made only of flip-flops for weight storage is provided as a reference. As expected, STDP suffers from a high fixed cost, which is especially critical at low resolutions. As resolutions beyond 4 bits are of marginal interest for STDP [122] and as 3 bits are sufficient for SDSP to obtain the same dynamics as those presented in [106], SDSP offers a strong potential for low-area synapse implementations compared to STDP.

In order to avoid this high fixed cost of STDP, another approach has been proposed by Seo *et al.* in [25]. In their fully-digital time-multiplexed 45-nm chip, they reach an efficient synaptic area of $14.5\mu\text{m}^2$ with a 4-bit resolution (Table 1.2). To do so, the strategy proposed in [25] is to update synaptic weights upon both pre-synaptic and post-synaptic events, which however requires the design of a custom dual-port transpose SRAM with both row and column accesses, with the previously-outlined density and design time penalties. The proposed SDSP synapse does not imply such constraints: the synaptic weights can easily be offloaded to a standard single-port foundry SRAM, which is a key driver for the synaptic efficiency of the ODIN neuromorphic processor proposed in Chapter 2. Finally, Seo *et al.* also implement a stochastic variant of STDP (S-STDP) in order to maintain learning for weight resolutions as low as 1 bit, which allows pushing the synaptic area down to $8.5\mu\text{m}^2$. A similar idea will be used in the MorphIC neuromorphic processor for a stochastic SDSP (S-SDSP) implementation on binary weights (Chapter 2).

1.2.3 Simulation and implementation results

The proposed implementation of a 3-bit SDSP-based synapse requires only 18 gates for a post-shrink area of $15.7\mu\text{m}^2$ in 28-nm FDSOI CMOS. Post-layout simulations of SDSP-based learning are shown in Fig. 1.10, where an LTP transition occurs after $t = 400\text{ms}$. The red and blue shades superimposed over the Calcium variable illustrate the SDSP signals that enable synaptic potentiation (v_{up}) and depression (v_{down}), as prescribed by Eq. (1.2), while bistability events induce updates independently. An 8-bit LIF neuron extended with Calcium concentration is used as the post-synaptic neuron. This neuron extension required by SDSP occupies only $46\mu\text{m}^2$ for the added combinational logic and either $7\mu\text{m}^2$ or about $0.4\mu\text{m}^2$ for the added register-based or SRAM-based state, which is negligible. Indeed, the combinational logic is amortized over the number of synapses in the neuron dendritic tree (i.e. synaptic fan-in) if the neurons are standalone and operate in parallel. Otherwise, if they are time-multiplexed, the combinational logic update is even amortized over all synapses in the system. Regarding the state overhead, it is amortized over the neuron dendritic tree in both cases.

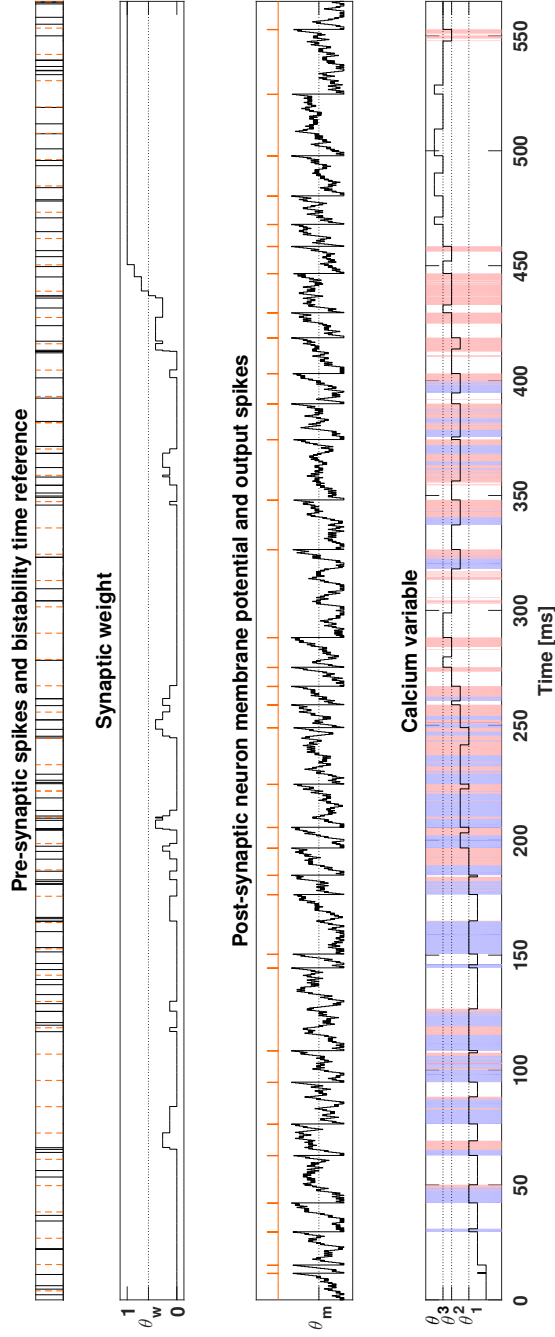


Fig. 1.10 Post-layout simulation of the proposed SDSP-based synapse. The bistability bist reference trigger is shown as dotted orange lines next to the pre-synaptic spikes. The SDSP conditions for potentiation (v_{up}) and depression (v_{down}) from the post-synaptic neuron are shown next to the Calcium variable in red and blue, respectively. Non-colored parts of the Calcium variable are associated with stop-learning conditions.

Table 1.2 State of the art of silicon synapses with embedded spike-based long-term learning rules, adapted and completed from [90].

	Technology	Plasticity	Fully-synthesizable	Weight storage (precision)	Estimated area
Schemmel, 2006 [99]	0.18 μ m	STDP	✗	SRAM (4 bits)	100 μ m ²
Arthur, 2006 [68]	0.25 μ m	STDP	✗	SRAM (1 bit)	440 μ m ²
Tanaka, 2009 [100]	0.25 μ m	STDP	✗	Capacitor (Analog)	5000 μ m ²
Cruz-Albrecht, 2012 [102]	90nm	STDP	✗	Capacitor (Analog)	4823 μ m ²
Seo, 2011 [25]	45nm SOI	S-STDP	✗ [†]	SRAM (1 bit)	8.5 μ m ² *
Seo, 2011 [25]	45nm SOI	S-STDP	✗ [†]	SRAM (4 bits)	14.5 μ m ² *
Ramakrishnan, 2011 [101]	0.35 μ m	Weight-dependent STDP	✗	Floating gate (Analog)	100 μ m ²
Bamford, 2012 [103]	0.35 μ m	Weight-dependent STDP	✗	Capacitor (Analog)	400 μ m ²
Maldonado Huayaney, 2016 [115]	0.18 μ m	Calcium-based STDP	✗	Capacitor (Bistable)	1638 μ m ²
Giulioni, 2008 [107]	0.35 μ m	SDSP	✗	Capacitor (Bistable)	3140 μ m ²
Mitra, 2009 [108]	0.35 μ m	SDSP	✗	Capacitor (Bistable)	3000 μ m ²
Qiao, 2015 [27]	0.18 μ m	SDSP	✗	Capacitor (Bistable)	252 μ m ²
Mayr, 2016 [28]	28nm	SDSP	✗	SRAM (4 bits)	16 μ m ² ‡
Proposed synapse	28nm FDSOI	SDSP	✓	Flip-flops (3 bits)	15.7 μ m ²
Proposed synapse (ODIN)	28nm FDSOI	SDSP	✓	SRAM (3+1 bits)	0.68 μ m ²
Proposed synapse (MorphIC)	65nm	S-SDSP	✓	SRAM (1 bit)	0.64 μ m ²

* Assuming that two thirds of the neuron area are occupied by time window generation and random number generation for stochastic STDP (S-STDP).

† Requires a custom SRAM design.

‡ Including the RAM block area and excluding the short-term plasticity circuitry for a fair comparison, otherwise the full synapse occupies 26.5 μ m².

A summary of previously-proposed analog and digital online-learning synapse designs is provided in Table 1.2. The most efficient previously-proposed designs are the analog 28-nm SDSP implementation from Mayr *et al.* [28] and the digital 45-nm S-STDP implementation from Seo *et al.* [25], both leveraging time multiplexing for high density. However, the former suffers from the peripheral analog driver circuitry and the latter suffers from the penalty of a custom dual-port SRAM design to accommodate for row and column accesses of the S-STDP learning rule. Even in its standalone version for a fully-parallel processing based on a 3-bit flip-flop weight storage, the proposed fully-synthesizable digital 28-nm SDSP synapse is still competitive. However, when considering a time-multiplexed implementation as demonstrated with the ODIN and MorphIC neuromorphic processors in Chapter 2, record densities of $0.68\mu\text{m}^2$ per (3+1)-bit 28-nm synapse and of $0.64\mu\text{m}^2$ per 1-bit 65-nm synapse are achieved, respectively. These numbers include the side overhead of Calcium variable extension in the neurons and, in the case of MorphIC, of random number generation.

Integration – Neuromorphic experimentation platforms

Contents

2.1	Overview of neuromorphic experimentation platforms . .	35
2.1.1	Analog/mixed-signal designs	35
2.1.2	Digital designs	37
2.2	ODIN – A 28-nm neuromorphic processor with record neuron and online-learning synapse densities	39
2.2.1	Architecture and implementation	40
2.2.2	Specifications, measurements and benchmarking results . . .	48
2.3	MorphIC – A 65-nm quad-core neuromorphic processor demonstrating large-scale connectivity with record binary online-learning synapse density	58
2.3.1	Architecture and implementation	60
2.3.2	Specifications, measurements and benchmarking results . . .	70
2.4	Discussion	74
2.4.1	Comparative analysis of bottom-up neuromorphic designs . .	76
2.4.2	Hierarchical event routing	77
2.4.3	Bottom-up spike-based online learning performance	79

Abstract As a second step of the bottom-up approach, we investigate silicon integration of neurons and synapses. We first identify the main trends, implementation strategies and use cases by surveying the key bottom-up neuromorphic designs, referred to as *experimentation platforms*. We identify that edge computing is a promising use case for small- to medium-scale neuromorphic designs, for which we propose two chips. First, ODIN is a 256-neuron 64k-synapse neuromorphic processor prototyped in 28-nm FDSOI CMOS [JP1]. Using time-multiplexed versions of the phenomenological neuron and SDSP synapse models derived in Chapter 1, ODIN demonstrates record neuron and synapse densities in a silicon area of 0.086mm^2 , while maintaining a high energy efficiency with 12.7pJ per synaptic operation (SOP). We then extend these claims with MorphIC, a quad-core 65-nm neuromorphic processor embedding 2k neurons and more than 2M synapses in a silicon area of 2.86mm^2 [JP2, CP3]. We propose a stochastic SDSP (S-SDSP) synapse to further improve density by moving synaptic weights to binary resolutions, while scalability is ensured with a hierarchical routing infrastructure that can accommodate large-scale multi-chip setups. We highlight the efficiency of ODIN and MorphIC by carrying out a detailed comparative analysis of neuromorphic experimentation platforms along the key figures of merit of density, energy efficiency, versatility and flexibility. Finally, we further discuss the aspect of scalability and the relevance of local bio-inspired spike-based learning rules for real-world applications, which are difficultly addressed by bottom-up designs.

2.1 Overview of neuromorphic experimentation platforms

The field of neuromorphic engineering takes its roots into the late 1980s with the discovery that the MOS transistor operated in the subthreshold regime can directly emulate the ion channels dynamics in the brain [21], which led to a long tradition of bottom-up design. Starting from neuroscience observation and biophysical neuron and synapse models, small- to large-scale integrations *in silico* have been achieved with a wide diversity of approaches. There are two main categories: analog/mixed-signal designs (Section 2.1.1) and digital designs (Section 2.1.2). Depending on their implementation, the resulting neuromorphic systems can be used as experimentation platforms for neuroscience or, if strict correspondence with the brain biophysics is not enforced, bio-inspired edge computing (small-scale systems) or cognitive computing (large-scale systems). The landscape of bottom-up neuromorphic experimentation platforms is summarized in Table 2.1.

2.1.1 Analog/mixed-signal designs

Mixed-signal neuromorphic designs usually rely on core analog computation with a digital spike distribution. The latter is based on asynchronous address-event representation (AER) buses [88,89] in small-scale designs (e.g., [27]) or on network-on-chip (NoC) infrastructures for spike packet routing in large-scale designs (e.g., [29,30,32]). Core analog computation can be of two main types: sub- or above-threshold.

Subthreshold analog designs allow leveraging a direct *emulation* approach for compact and low-power neuromorphic circuits that lie close to the brain biophysics. Considering voltage swings of 1V for capacitors and currents on the order of 1pF and 1nA, respectively, the resulting time constants are on the order of 1ms [123]. Subthreshold analog designs are thus inherently adapted for real-time processing with time constants well-matched to those of the environment. Therefore, the high biophysical accuracy and biological time constants of subthreshold analog designs make them ideally-suited for brain emulation and basic research through analysis by synthesis. However, their high susceptibility to mismatch and PVT variations is a critical challenge that requires advanced calibration procedures [124] and exploration of mitigation techniques with robust computational primitives at the network level [82]. This approach is pursued in three main designs. First, the 0.18- μm ROLLS chip [27] is a neurosynaptic core that embeds 256 AdExp neurons (Section 1.1), 64k synapses with STP and 64k synapses with SDSP (Section 1.2). Second, the 0.18- μm DYNAPs chip [29] is a quad-core 2k-neuron 64k-synapse scale-up of ROLLS whose focus is put on the spike routing and communication infrastructure, at the expense of synaptic plasticity, which has been removed. A 28-nm version of the DYNAPs chip has been designed, which includes a plastic core embedding 64 neurons and 8k 4-bit digital STDP synapses, with preliminary results reported in [125]. Finally, the Neurogrid, a 1-million-neuron system based on

Table 2.1 Overview of bottom-up neuromorphic experimentation platforms and implementation strategies. (S) denotes small-scale chips embedding up to 256 neurons. (M) denotes medium-scale chips embedding 1k to 2k neurons and a communication infrastructure suitable for large-scale integration. (L) denotes large-scale chips or systems ranging from tens or hundreds of thousands (single chip/wafer) to millions of neurons (multi-chip setups), with up to a billion neurons for supercomputer-scale setups.

Implementation	Key designs	Main application
Analog mixed-signal	Subthreshold	ROLLS (S) [27] DYNAPs (M) [29] Neurogrid (L) [32]
	Above-threshold	BrainScaleS (L) [30] (BrainScaleS 2) (L)* [127]
	Switched- or time-muxed-cap	Mayr <i>et al.</i> (S) [28] IFAT (L) [26]
Digital	Software-based	GENESIS [134] NEURON [135] NEST [136] Auryn [138] Brian 1,2 [137,141] ANNarchy [139] GeNN [140]
		Low-cost and flexible neuroscience simulation
	Distributed von-Neumann	SpiNNaker (L) [31] (SpiNNaker 2) (L)* [143]
		Neuroscience simulation acceleration
	Full-custom	Seo <i>et al.</i> (S) [25] ODIN (S) MorphIC (M)
		Bio-inspired edge computing
	FPGA-based [‡]	TrueNorth (L) [33] Loihi (L) [34]
		Cognitive computing
		Cassidy <i>et al.</i> (L) [145] Minitaur (L) [146] Wang <i>et al.</i> (L) [147] Luo <i>et al.</i> (L) [148] Yang <i>et al.</i> (L) [149]
		Low-cost, flexible neuroscience simulation and cognitive computing

[†] Software-based approaches rely on conventional CPU and GPU hardware. The implementation scale depends on available resources and on the granularity at which the biophysics are reproduced.

* The second-generation BrainScaleS and SpiNNaker large-scale systems are currently in development, only proof-of-concept prototype chips have been reported so far. The BrainScales 2 prototype embeds only 64 neurons, while the SpiNNaker 2 prototype embeds only 4 ARM cores out of the 152 planned.

[‡] Non-exhaustive list.

sixteen 0.18- μm Neurocore chips, was designed in order to emulate the biophysics of cortical layers [32]. However, large-scale integration is achieved at the expense of synaptic weight storage, which has been moved off-chip, thus inducing power and latency overheads.

Above-threshold analog designs are ideally-suited for accelerated-time neuroscience simulations. Indeed, compared to subthreshold analog designs, the capacitor size is kept on the order of 1pF, while higher currents and reduced voltage swings allow reaching acceleration factors ranging from 10^3 to 10^5 , with an improved robustness

to mismatch and PVT variations [126]. This approach is pursued in the BrainScaleS wafer-scale design: it relies on 0.18- μm HICANN chips with 512 AdExp neurons and 112k 4-bit STDP synapses integrated at a scale of 352 chips per wafer [30]. BrainScaleS thus embeds 180k neurons and 40M synapses per wafer for large-scale simulation and exploration of cortical functions. The second-generation BrainScaleS is currently being designed, with early small-scale 64-neuron 2k-synapse prototypes embedding a programmable plasticity processor as well as multi-compartment neuron models for dendritic computation and structural plasticity, directly inspired by recent neuroscience research [127, 128].

Approaches based on *switched-capacitor* and *capacitor-based time multiplexing* have been proposed in [28] and [26]. The 28-nm chip from Mayr *et al.* is an interesting attempt at leveraging technology scaling by using digital control and SRAM-based weight storage, while maintaining the higher biophysical accuracy of analog designs for synaptic plasticity through switched-capacitor circuits [28]. Capacitor-based time multiplexing is used for neuron membrane potential storage. This small-scale chip embeds 64 neurons and 8k 4-bit synapses with both STP and SDSP, as per the implementation described in [129]. It is thus suitable for near-sensor applications at the edge, where the power and area footprints should be minimized [13, 14]. The 65-nm integrate-and-fire array transceiver (IFAT) chip from Park *et al.* relies on conductance-based neuron and synapse models with capacitor-based time multiplexing [26], embedding as high as 65k two-compartment integrate-and-fire neurons per chip. However, synapses do not embed synaptic plasticity and their weights are stored off-chip. This chip is thus appropriate for large-scale cognitive computing experiments with relaxed synaptic requirements.

Finally, similarly to the *in-memory computing* trend in machine learning accelerators [130], memristor crossbar arrays for high-density in-memory neuromorphic computing are currently being actively investigated [116]. Memristors promise both high density and characteristics similar to those of biological synapses [131], however co-integration with CMOS is still at an early stage (Section 1.2). A first proof-of-concept chip has recently been proposed in [132], though only demonstrated for very small problems (e.g., classification of 5×5 -pixel binary patterns). It embeds 5k memristor synapses at a density of $10 \mu\text{m}^2$ per synapse, which is an order of magnitude behind state-of-the-art digital integrations. Significant work is thus required to achieve optimized memristor-based neuromorphic systems and to alleviate the aspects of synaptic resolution control, mismatch and fabrication costs. A 0.35- μm flash-based STDP design has also been proposed [133], but embedded flash memory is difficult to scale beyond 40-nm CMOS and requires high programming voltages.

2.1.2 Digital designs

Digital designs have two key advantages over their analog counterparts: the robustness to noise, mismatch and PVT variations and the potential to fully leverage technology scaling. The former ensures a predictable behavior and possibly a

one-to-one hardware-software correspondence, while the latter ensures competitive energy and density for neuron and synapse integration with deep sub-micron technologies, which compensates the overhead of going for a *simulation* approach lying far from the biophysics. While neuromorphic engineering aims at a paradigm shift from von-Neumann-based architectures to distributed and co-integrated memory and computing elements, the granularity at which this paradigm shift is achieved in digital implementations strongly varies between three main approaches: software-based, distributed von-Neumann or full-custom, from high to low processing and memory separation.

Software-based approaches rely on conventional von-Neumann hardware. Dedicated spiking neural network simulators such as GENESIS [134], NEURON [135], NEST [136], Brian [137] and Aurnyn [138] allow running experiments on conventional CPUs, while simulators such as ANNarchy [139], GeNN [140] and Brian 2 [141] provide GPU support. Software-based approaches provide the highest flexibility and control over the neuron and synapse models and experiments scale. However, using von-Neumann hardware to simulate spiking neural networks comes at the cost of power and simulation time overhead, although recent work has demonstrated that GPUs can compare favorably to a SpiNNaker-based system for cortical-scale simulations [142].

SpiNNaker belongs to the *distributed von-Neumann* approaches. It was fabricated in 0.13- μm CMOS technology and embeds 18 ARM968 cores per chip in a GALS design for efficient handling of asynchronous spike data, spanning biological to accelerated time constants [31]. SpiNNaker has been optimized for large-scale spiking neural network experiments while keeping a high degree of flexibility, with the current supercomputer-scale setup reaching the billion of neurons, i.e. about 1% of the human brain [35]. The second-generation SpiNNaker system is currently being designed. Current 28-nm prototype chips embed 4 ARM Cortex M4F cores out of the 152 per chip planned for the final 22-nm SpiNNaker 2 system [143]. The objective is to simulate about two orders of magnitude more neurons per chip compared to the first-generation SpiNNaker: when integrated at supercomputer scale, real-time simulations of the whole-scale human brain will be within reach [144].

Full-custom digital designs allow reaching high-density and energy-efficient neuron and synapse integrations, thanks to memory being moved closer to computation compared to the two above-mentioned digital approaches. As all full-custom digital designs reported so far are using SRAM-based time multiplexing, this can be related to the efficiency improvement brought by caches in conventional von-Neumann processors. Full-custom designs can usually be configured to span biological to accelerated time constants. The 45-nm small-scale design from Seo *et al.* embeds 256 LIF neurons and 64k binary synapses based on S-STDP [25], it achieves high neuron and synapse densities despite the use of a custom SRAM (Section 1.2). Its scale thus makes it ideal for edge computing. In line with this application, we propose two chips: ODIN and MorphIC. The former embeds 256 neurons with the 20 Izhikevich behaviors and 64k SDSP-based (3+1)-bit synapses in 28-nm CMOS,

while the latter is a quad-core scale-up in 65-nm CMOS allowing for large-scale multi-chip setups with a total of 2k LIF neurons and more than 2M binary S-SDSP synapses per chip, as described in Sections 2.2 and 2.3, respectively. ODIN and MorphIC achieve record neuron and synapse densities with high energy efficiency. Finally, cognitive computing applications require large-scale platforms, which is currently offered by the 28-nm IBM TrueNorth [33] and the 14-nm Intel Loihi [34] neuromorphic chips. On the one hand, TrueNorth embeds as high as 1M neurons and 256M binary non-plastic synapses per chip, where neurons rely on a custom model emulating 11 Izhikevich behaviors, or 20 behaviors if three neurons are combined [85]. On the other hand, Loihi embeds up to 180k neurons and 114k (9-bit) to 1M (binary) synapses per chip. Neurons rely on a LIF model with a configurable number of compartments to which several functionalities such as axonal and refractory delays, spike latency and threshold adaptation have been added. The spike-based plasticity rule used for synapses is programmable.

Finally, it should be noted that digital approaches also encompass FPGA designs, which trade off efficiency for higher flexibility and a reduced cost compared to full-custom designs. Although beyond the scope of this thesis, a wide diversity of FPGA designs cover small- to large-scale cognitive computing (e.g., [145–147]) and neuroscience applications (e.g., [148, 149]).

2.2 ODIN – A 28-nm neuromorphic processor with record neuron and online-learning synapse densities

As digital designs follow a *simulation* approach as opposed to the direct emulation property of subthreshold analog/mixed-signal designs, optimizing the tradeoff between density, power, versatility and biological plausibility in digital SNNs is still an open challenge. In order to explore the brain computational primitives in the context of edge computing, we consider that there are two essential requirements. First, on the synapse side, event-driven embedded online learning allows low-power autonomous agents to adapt in real time to new features in uncontrolled environments, where limited training data is presented on-the-fly to the network (Section 1.2). Embedded online learning is thus critical for edge computing in order to avoid the overhead of an off-chip learning engine. Second, on the neuron side, the widespread LIF neuron model has been shown to lack the essential behavior repertoire necessary to explore the computational properties of large neural networks as well as time-based codes (Section 1.1.1): embedding the biological neuron versatility *in silico* is thus necessary. However, biophysically-accurate models such as the H&H, Izhikevich or AdExp (Section 1.1) require digital SNNs to solve coupled nonlinear differential equations and to update all neuron states at each integration timestep, which is particularly expensive for time-multiplexed designs. Current full-custom approaches rely on neuron models with a restricted set of behaviors (e.g., TrueNorth and Loihi) [33, 34], down to the simple LIF model for the chip of Seo *et al.* [25].

Therefore, we propose ODIN, an online-learning digital spiking neuromorphic processor in 28-nm FDSOI CMOS.¹ It comprises 256 neurons and 64k synapses based on the models proposed in Chapter 1 and embeds SDSP online learning at a high density with only $0.68\mu\text{m}^2/\text{synapse}$. Neurons can be programmed to emulate the 20 Izhikevich behaviors found in biological spiking neurons [66] using a custom phenomenological model that is entirely event-driven and does not require a neuron update at each integration timestep. ODIN occupies only 0.086mm^2 and consumes 12.7pJ per synaptic operation (SOP) at 0.55V . Using a single presentation of 6k pre-processed 16×16 -pixel training images from the MNIST dataset of handwritten digits [150], a single-layer fully-connected 10-neuron network with on-chip SDSP-based learning achieves a classification accuracy of 84.5% with only 15nJ per inference using rank order coding. The design of ODIN is described in Section 2.2.1, with architecture and implementation details on the event-based addressing scheme, the online-learning synapse, the phenomenological neuron and the internal event scheduler. The specifications and measurement results are presented in Section 2.2.2 with a comparison on MNIST of two learning strategies: on-chip and online with SDSP or off-chip and offline with stochastic gradient descent.

2.2.1 Architecture and implementation

The architecture of ODIN corresponds to a crossbar with all-to-all neuron connectivity, as shown in Fig. 2.1. In order to minimize the silicon area by leveraging the high-speed operation of digital logic, a time multiplexing approach is followed in which neurons and synapses are updated in a sequential manner, not in parallel. The individual states of N neurons and N^2 synapses are stored in on-chip SRAMs, a controller handles the neuron and synapse updates to emulate a $N\times N$ crossbar, where each of the N neurons thus has a fan-in of N online-learning synapses.

Input and output events at the chip level are handled through four-phase-handshake AER buses. As ODIN is implemented using a synchronous digital implementation flow (Section 2.2.2), in order for the AER to be used in an asynchronous fashion from an off-chip point of view, a double-latching synchronization barrier has been placed on the **REQ** and **ACK** lines of the input and output AER interfaces, respectively, to limit metastability issues. In order to increase versatility in experimentation and testing of ODIN, the input AER bus is extended in order to represent a wide variety of event types.

- 1) Neuron spike event – It is the standard operation of ODIN, which can be triggered externally from the input AER interface or internally from the embedded neurons. If the $\log_2 N$ -bit address of a source neuron i is received over AER, all N neurons of ODIN will be updated with the synaptic weight found at synapse i in their respective N -synapse dendritic tree: each neuron spike event thus leads to N SOPs. An SDSP weight update will also be applied to these synapses (Section 2.2.1.1).

¹ The HDL code and documentation of ODIN are publicly available at <https://github.com/ChFrenkel/ODIN>.

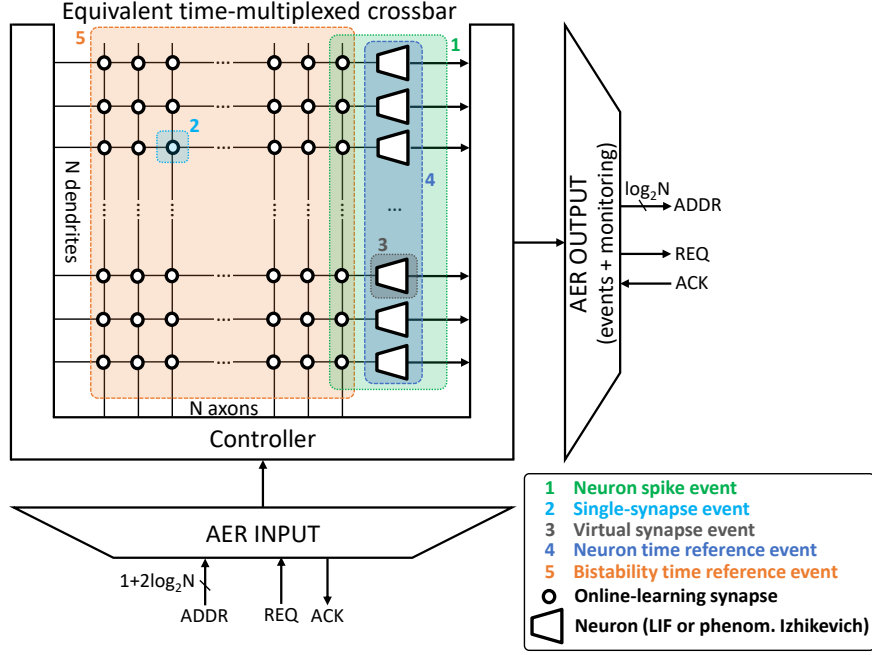


Fig. 2.1 Equivalent N-neuron N^2 -synapse time-multiplexed crossbar architecture. Input AER addresses cover neuron, single-synapse, virtual synapse and time reference events for neurons and for synaptic bistability.

- 2) Single-synapse event – Two $\log_2 N$ -bit addresses are provided in the ADDR field of the input AER interface: the address of a source neuron i and the address of a destination neuron j . This event is handled similarly to an AER neuron spike event, but only the neuron j of ODIN will be updated, together with a single SDSP weight update to synapse i of neuron j .
- 3) Virtual synapse event – The $\log_2 N$ -bit address of a target neuron j of ODIN and a fixed weight value are provided in the ADDR field, no synapse of ODIN is read nor updated.
- 4) Neuron time reference event – A specific address is used to provide an external time reference event to all neurons, which defines the time constant of the Izhikevich behaviors of the phenomenological neuron model (Section 2.2.1.2).
- 5) Bistability time reference event – A specific address is used to trigger the bistability mechanism of all synapses (Section 2.2.1.1).

Therefore, in order to represent these five event types with all neuron, synapse and specific time reference control addresses, the input AER bus needs an ADDR field width of $1+2\log_2 N$ bits (Fig. 2.1).

Two operation modes can be chosen for the $\log_2 N$ -bit output AER bus. The first one is the standard operation mode in which, as soon as one of the N neurons emits an output spike event, the source neuron address is transmitted on the $\log_2 N$ -bit

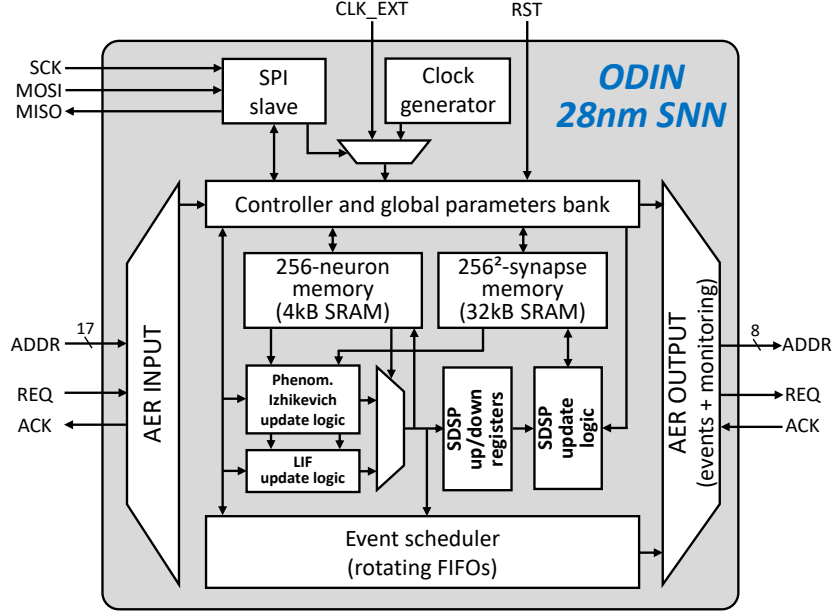


Fig. 2.2 Block diagram of the ODIN online-learning digital spiking neuromorphic processor, which implements the architecture shown in Fig. 2.1 for $N=256$ neurons.

ADDR field of the AER output. The second one is a non-standard operation mode for monitoring purposes of a specific neuron or synapse, AER events containing state data are generated. The target neuron and synapse addresses to be monitored are configurable.

Fig. 2.2 gives a block diagram overview of the ODIN neuromorphic processor, which implements the crossbar architecture presented in Fig. 2.1 for $N=256$ neurons. The individual states and parameters of the 256 neurons and 64k synapses are respectively stored in 4-kB and 32-kB high-density single-port foundry SRAMs while a global controller manages time multiplexing of the synapse and neuron update logic (Sections 2.2.1.1 and 2.2.1.2), the associated timing diagram is shown in Fig. 2.3. All chip-level control registers as well as individual neuron and synapse states and parameters can be read and written through a conventional SPI bus, while input and output events at the chip level are handled through AER buses. Finally, a scheduler based on rotating FIFOs (Section 2.2.1.3) arbitrates and orders external and internally-generated events, including bursts of spikes induced by Izhikevich bursting behaviors.

2.2.1.1 Synapse array

In order to embed online learning in each synapse of the 256×256 crossbar array, an efficient digital implementation of a learning rule is required. In the comparative analysis between the STDP and SDSP learning rules carried out in Section 1.2, we

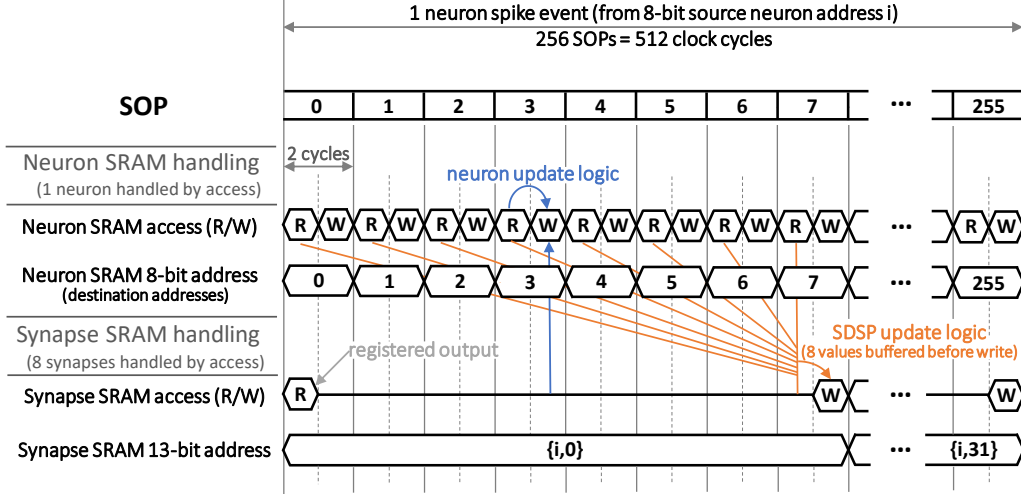


Fig. 2.3 Timing diagram of ODIN for a neuron spike event from 8-bit source neuron address i , leading to 256 SOPs. Each SOP lasts two clock cycles. First, the current state of a destination neuron with 8-bit address j is retrieved. Second, the current state of neuron j and the associated synaptic weight corresponding to the source neuron i are provided to the neuron update logic, the updated state of neuron j is then written back to the neuron memory. As ODIN has 4-bit synapses and the chosen synapse SRAM has 32-bit words for density purposes, read and write operations to the synapse memory handle 8 synapses at a time in order to minimize memory accesses. The SDSP update logic takes information from the Calcium variable and membrane potential of the destination neurons in order to update the associated synaptic weights.

demonstrated that the SDSP learning rule is particularly efficient in digital designs for two reasons. First, it allows offloading plasticity computation to the neurons, thus avoiding replication of circuit resources inside each synapse and leading to substantial resource and area savings compared to STDP. The neuron model needs to be extended with a Calcium variable, as recalled in Eq. (2.1), where V_{mem} is the neuron membrane potential (compared to threshold θ_m), Ca is the Calcium variable (compared to thresholds θ_1 , θ_2 and θ_3) and updates to synaptic weights w have been normalized to unit steps. Second, SDSP depends only on the state of the post-synaptic neuron at the time of the pre-synaptic spike, which allows leveraging the high density of single-port foundry SRAMs for time-multiplexed synaptic state storage. Indeed, as STDP depends on relative spike timings, it either implies a local time window storage inside each synapse or the design of custom dual-port SRAMs with both row and column accesses (Section 1.2.2).

$$\begin{cases} w \rightarrow w + 1 & \text{if } V_{\text{mem}}(t_{\text{pre}}) \geq \theta_m, \theta_1 \leq \text{Ca}(t_{\text{pre}}) < \theta_3 \\ w \rightarrow w - 1 & \text{if } V_{\text{mem}}(t_{\text{pre}}) < \theta_m, \theta_1 \leq \text{Ca}(t_{\text{pre}}) < \theta_2 \end{cases} \quad (2.1)$$

Each synapse of ODIN consists of a 3-bit weight and 1 bit of mapping table to enable or disable online learning locally, which results in a 4-bit synapse. All

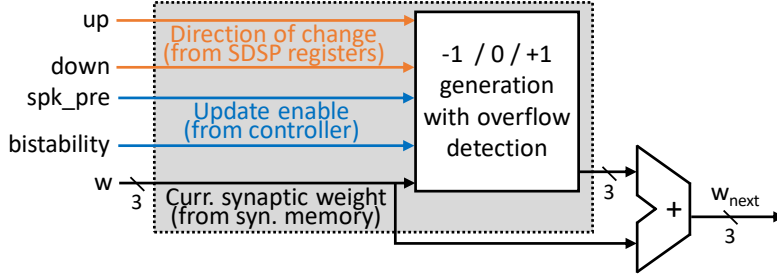


Fig. 2.4 SDSP update logic for the time-multiplexed implementation of the standalone SDSP synapse proposed in Chapter 1. The different blocks interacting with the SDSP update logic are shown in Fig. 2.2.

thresholds θ_m , θ_1 , θ_2 and θ_3 of each neuron can be individually programmed through SPI (Section 2.2.1.2). Being a time-multiplexed design, the standalone SDSP synapse circuit from Fig. 1.8(b) reduces to the update circuit shown in Fig. 2.4. The **up** and **down** signals, coming from the SDSP registers shown in Fig. 2.2, represent the increment/decrement conditions of SDSP updates in Eq. (2.1). The controller provides control signals conditionally enabling the updates: **spk_pre** is linked to incoming pre-synaptic spikes and **bistability** is associated to bistability time reference events received externally through specific AER events. The bistability mechanism is optional (Section 1.2.1): each time a bistability time reference event is received, if the current synaptic weight is above (resp. below) half its dynamic, it increments (resp. decrements) toward a high (resp. low) state. This time-multiplexed digital SDSP learning rule is implemented at high density: each 4-bit plastic synapse of ODIN occupies only $0.68\mu\text{m}^2$, taking into account the synapse SRAM area, the SDSP update logic and the addition of a Calcium variable into the neurons. Each SOP in ODIN takes two clock cycles, as shown in Fig. 2.3.

2.2.1.2 Neuron array

In order to leverage spike-based processing with efficient spike encodings, time must be included into computation, which can be captured by the Izhikevich behavior repertoire, beyond the simple integration provided by LIF neurons (Section 1.1.1). ODIN comprises two neuron models (Fig. 2.2): each of the 256 neurons of ODIN can be individually chosen between a standard 8-bit LIF neuron model and the custom phenomenological model of the 20 Izhikevich behaviors proposed in Chapter 1. Furthermore, each neuron can be defined as excitatory or inhibitory. As the neurons are time-multiplexed and their states and parameters are stored in a single-port foundry SRAM, the use of the combinational update logic of the LIF model or of the phenomenological model is determined by a neuron parameter bit. Both neuron models are extended with a 3-bit Calcium variable and associated Calcium threshold parameters to enable SDSP online learning in the synapses (Section 2.2.1.1). As it represents an image of the recent neuron firing activity, the Calcium variable

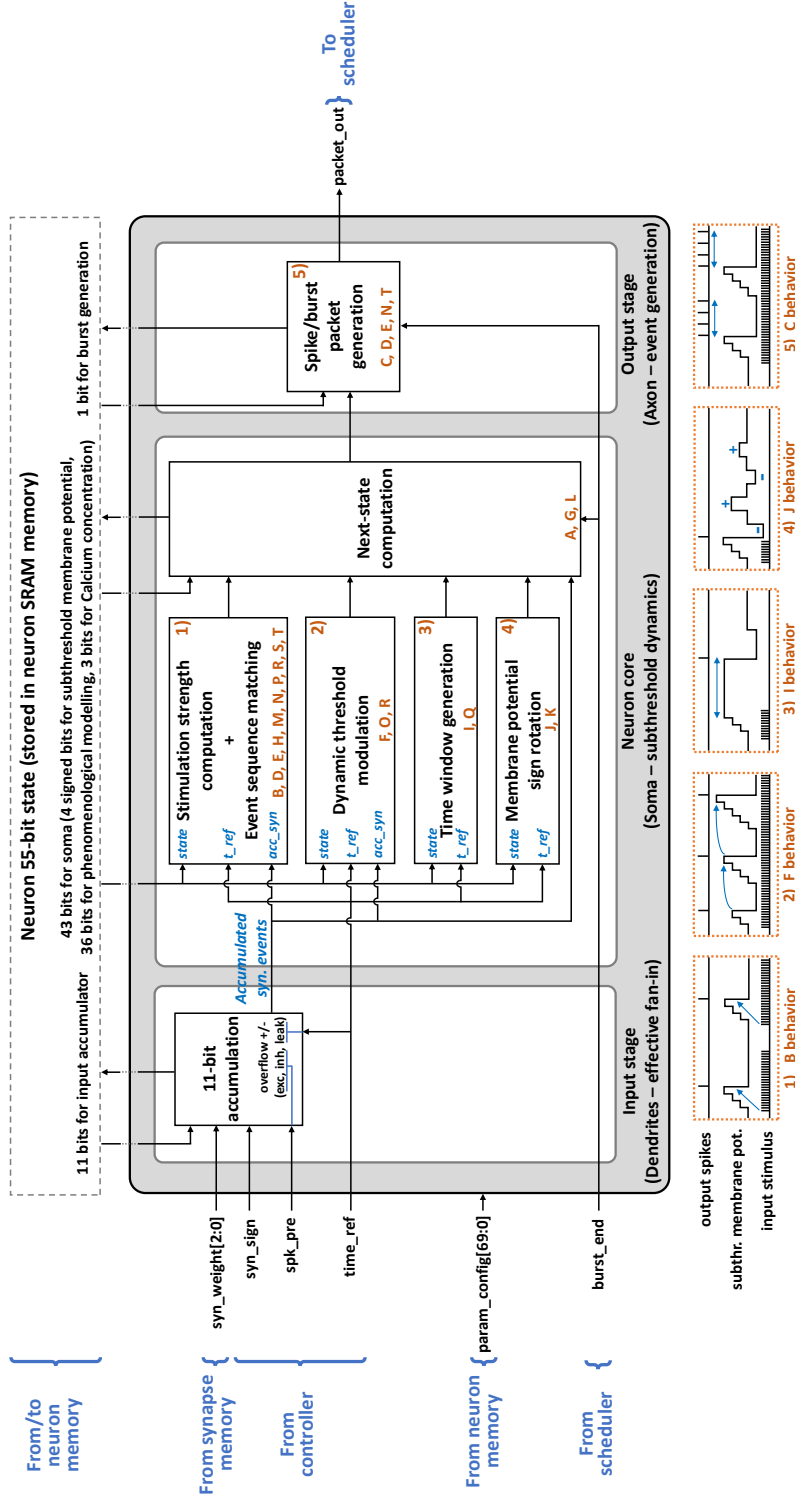


Fig. 2.5 Three-stage architecture of the proposed phenomenological digital neuron update logic, corresponding to the time-multiplexed and Calcium-extended version of the model proposed in Chapter 1. Letters inside the different combinational logic blocks indicate the Izhikevich behaviors that are phenomenologically generated, following the numbering convention provided in Fig. 2.9. A sample behavior illustration is also provided for each combinational logic block to illustrate their functionality on a practical case, as outlined in blue in the behavior illustrations. The different blocks interacting with the neuron update logic are shown in Fig. 2.2.

of a neuron is incremented each time a spike is fired. The Calcium leakage time constant is configurable and depends on the neuron time reference events provided externally through specific AER events.

The three-stage architecture of the proposed phenomenological neuron is shown in Fig. 2.5, it is the time-multiplexed and Calcium-extended version of the standalone neuron circuit from Fig. 1.4. The characteristics and behaviors of each neuron are independently controlled with a 70-bit `param_config` parameter array stored in the neuron memory, next to the 55-bit neuron state. Each neuron of ODIN thus requires 126 parameter and state bits, including one bit to select between the LIF and the phenomenological neuron models. These parameters and the initial state of all neurons are preloaded during the initial SPI configuration of ODIN. The time constants of the neuronal dynamics range from biological- to accelerated-time depending on the frequency of external time reference events. The neuron is entirely event-driven, the state is updated only upon reception of synaptic or time reference events, no mathematical equation needs to be solved at each integration timestep.

The time-multiplexed Calcium-extended version of the phenomenological Izhikevich neuron operates as per the description provided in Section 1.1.3, at the exception of the output stage. When firing conditions are met in the neuron core, the output stage generates a neuron output event packet `packet_out` containing the number of spikes and inter-spike interval (ISI) to be generated, following the parameters in `param_config`. As the neuron state is updated only after input events and as bursts usually have an ISI shorter than neuron time reference events (Fig. 2.9), the neuron alone is not able to update itself in order to generate bursts. Therefore, neuron event packets are sent to the scheduler of ODIN (Section 2.2.1.3) which arbitrates the generation of spikes and bursts from all neurons. When a neuron emits a burst event, its membrane potential is reset and then locked down until the scheduler signals the end of burst generation using the `burst_end` signal.

2.2.1.3 Scheduler

The architecture of the scheduler block in Fig. 2.2 is shown in Fig. 2.6 and can be assimilated to a priority-based ordered FIFO [151]. Its purpose is to handle spiking and bursting events from all neurons and to arbitrate between external and internally-generated neuron events. Spiking and bursting neurons of ODIN send 14-bit event packets to the scheduler. The packets contain the 8-bit address of the source neuron, a 3-bit field indicating the number of spikes to be generated minus one and a 3-bit field quantifying the ISI. The two latter parameters are zero in the case of single-spike events. All single-spike events are stored in a FIFO of 32 8-bit stages, where each stage stores the address of the source spiking neuron. In the global priority-based ordered FIFO structure of the scheduler, this single-spike FIFO always has the highest priority: all single-spike events are handled as soon as they are available.

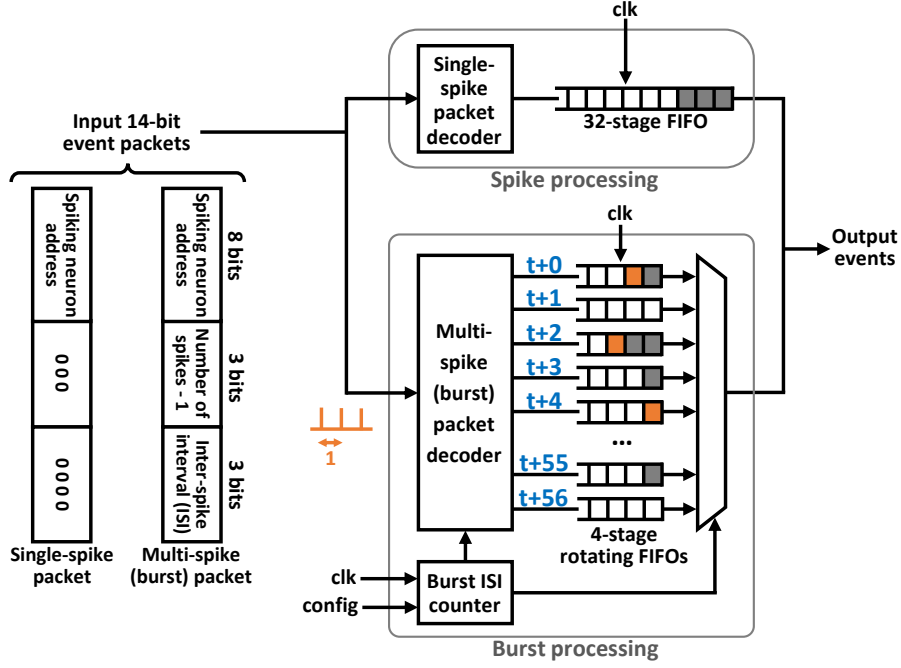


Fig. 2.6 Scheduler architecture and contents of event packets sent by spiking neurons. Handling bursts issued by the Izhikevich behaviors requires the use of rotating FIFOs in order to reproduce the inter-spike interval (ISI) value that is specified in the incoming burst packet. Orange spikes illustrate a 3-spike 1-timestep-ISI burst example being stored in the scheduler rotating FIFOs, where grayed-out stages represent previously-filled spike events waiting to be processed by the controller.

All multi-spike events (i.e. burst events) are separated by the burst packet decoder into multiple single-spike events and split among rotating 4-stage FIFOs, corresponding to successive timesteps. All FIFOs are assigned a priority token inspired from the FIFO priority queue architecture proposed in [151]. In order to generate the ISI of the different bursts, the scheduler generates its own timestep locally in the burst ISI counter, which is configurable to range from biological to accelerated time constants. As soon as one timestep is elapsed, the priorities of all 4-stage FIFOs are rotated. As neuron event packets can encode bursts of up to 8 spikes with an ISI of up to 7 timesteps, a total of $(7 \times 8) + 1 = 57$ rotating 4-stage FIFOs are necessary to encode all priorities, which are denoted in Fig. 2.6 by reference to the burst ISI timestep t , where the highest-priority current timestep is $t + 0$. To illustrate the scheduler operation, we show in Fig. 2.6 an input neuron event packet describing a 3-spike 1-timestep-ISI burst as an example. Three single-spike events will be generated by the burst packet decoder toward FIFOs associated with local ISI timesteps $+0$, $+2$ and $+4$. The FIFO stages have a 9-bit width, they contain the 8-bit address of the source spiking neuron with one bit marking the last spike of the burst, currently stored in the timestep $+4$ FIFO. The spike at timestep $+0$ will be processed immediately by the controller. After two timestep ticks of the ISI counter and two rotations of FIFO priorities, the second spike of the burst gets to

timestep +0 and is processed by the controller. After two other ISI counter ticks, the last spike of the burst is retrieved by the controller: as one bit indicates that this spike ends a burst, the scheduler unlocks state updates of the associated source neuron (Section 2.2.1.2).

2.2.2 Specifications, measurements and benchmarking results

The ODIN neuromorphic processor was fabricated in the ST Microelectronics 28-nm fully-depleted silicon-on-insulator (FDSOI) CMOS process, with a mix of 8-track LVT libraries with upsized gate length in order to reduce leakage (poly-biasing technique by 4nm and 16nm). ODIN is implemented following a standard fully-synthesizable synchronous digital implementation flow. Fig. 2.7(a) shows the chip microphotograph. ODIN occupies an area of only $320\mu\text{m} \times 270\mu\text{m}$ post-shrink, the remaining chip area is used by unrelated blocks. Fig. 2.7(a) also shows a zoom on the floorplan view, highlighting that the synapse and neuron single-port foundry SRAMs occupy 50% and 15% of the area, respectively. These numbers directly provide the area occupied by the 64k synapses and 256 neurons of ODIN as the associated combinational update logic area is negligible compared to the SRAM sizes. The scheduler, controller, neuron and synapse update logic, SPI and AER buses are implemented with digital standard cells in the remaining area of the floorplan, 80% of which is occupied by the scheduler. All measurement results presented in this section were accurately predicted by a Python-based simulator we developed for ODIN: one-to-one hardware/software correspondence is thus ensured.

2.2.2.1 Chip specifications and measurements

Specifications of the 64k-synapse 256-neuron ODIN spiking neuromorphic processor are summarized in Fig. 2.7(b). It embeds SDSP-based online learning, with each synapse consisting of a 3-bit weight and one bit of mapping table, while phenomenological neurons are able to exhibit all 20 Izhikevich behaviors from biological to accelerated time constants. A total of 9 ODIN chips were available for tests, the power and energy measurements reported in Fig. 2.7(b) are average values across ODIN test chips and have been acquired at a temperature of 24°C . ODIN remains fully functional while scaling the supply voltage down to 0.55V, where the maximum operating frequency of the chip is 75MHz, against 100MHz at the nominal supply voltage of 0.8V. The different contributions to the power consumption P of ODIN are summarized in Eq. (2.2):

$$P = P_{\text{leak}} + P_{\text{idle}} \times f_{\text{clk}} + E_{\text{SOP}} \times r_{\text{SOP}}, \quad (2.2)$$

where P_{leak} is the leakage power without clock activity and $P_{\text{leak}} + P_{\text{idle}} \times f_{\text{clk}}$ is the power consumption of ODIN without any activity in the network when a clock of

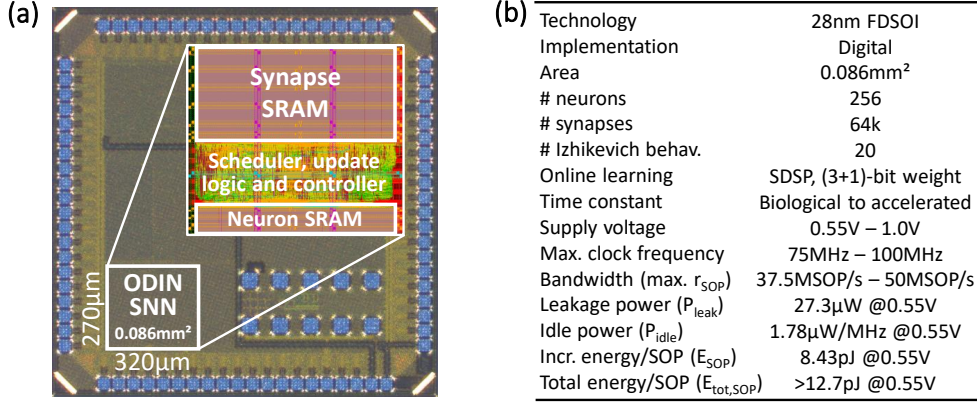


Fig. 2.7 (a) Chip microphotograph with zoom on floorplan view. Chip area outside the $320\mu\text{m} \times 270\mu\text{m}$ of the ODIN neuromorphic processor is occupied by unrelated blocks. (b) Specifications and measurements. P_{leak} , P_{idle} and E_{SOP} are parameters of the ODIN power consumption model in Eq. (2.2).

frequency f_{clk} is connected. E_{SOP} is the energy paid for each SOP, which includes the contributions of reading and updating the synaptic weight according to the SDSP learning rule, reading and updating the associated neuron state, as well as the controller and scheduler overheads. Finally, r_{SOP} is the average SOP processing rate, the maximum SOP rate ODIN can handle is equal to $f_{\text{clk}}/2$ as each SOP takes two clock cycles to complete (Fig. 2.3).

At 0.55V, P_{leak} is $27.3\mu\text{W}$ and P_{idle} is $1.78\mu\text{W}/\text{MHz}$. In order to determine E_{SOP} , we saturate the scheduler with neuron spike event requests, each consisting of 256 SOPs (Section 2.2.1), so that the achieved SOP rate r_{SOP} is at its maximum with one SOP processed every two clock cycles. By measuring the chip power consumption and subtracting leakage and idle power, we obtain an energy per SOP E_{SOP} of 8.43pJ.

Another commonly-employed definition for the energy per SOP can be found in Eq. (2.3), where the whole chip power consumption P is divided by the SOP rate r_{SOP} , without subtracting contributions from leakage and idle power (see Section 2.4.1 and Table 2.5 for an energy per SOP summary across state-of-the-art neuromorphic chips). In order to avoid confusion in the subsequent text, we will denote E_{SOP} as the *incremental* energy per SOP and $E_{\text{tot,SOP}}$ as the *global* energy per SOP. The former is appropriate for the definition of the power model in Eq. (2.2) and is useful to accurately predict the power consumption of ODIN based on its activity and clock frequency, while the latter is more application-driven and representative of the real energy per SOP performance.

$$E_{\text{tot,SOP}} = P/r_{\text{SOP}} \quad (2.3)$$

At a clock frequency of 75MHz, the maximum r_{SOP} is equal to 37.5MSOP/s and the measured power consumption P of ODIN is $477\mu\text{W}$ at 0.55V. At this

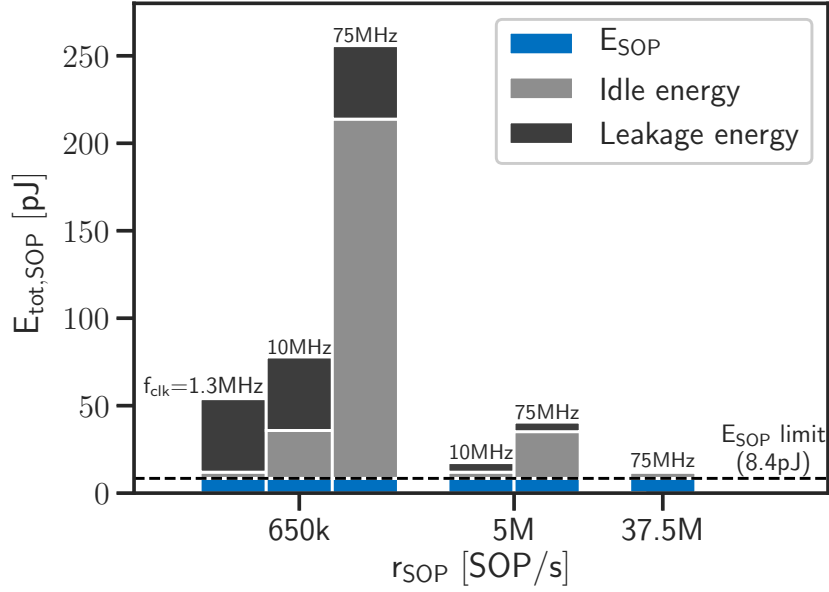


Fig. 2.8 Energy breakdown of the global SOP energy $E_{\text{tot,SOP}}$ at 0.55V, illustrating the contributions from the incremental SOP energy E_{SOP} , leakage energy and idle energy for different choices of r_{SOP} and f_{clk} . The energy efficiency of ODIN is the highest in accelerated time as it approaches the incremental energy E_{SOP} .

accelerated-time rate and maximum r_{SOP} , the global energy per SOP $E_{\text{tot,SOP}}$ is equal to 12.7pJ, which is dominated by dynamic power as the influence of leakage is negligible (6% of the total power) and the idle power accounts for 28% of the total power. In order to estimate $E_{\text{tot,SOP}}$ when ODIN operates at a biological time constant, an order of magnitude can be found by assuming that all 256 neurons of ODIN spike at a rate of 10Hz. Each neuron event leading to 256 SOPs, the corresponding SOP rate r_{SOP} is equal to 650kSOP/s, a clock frequency of at least 1.3MHz is thus sufficient to operate ODIN at a biological time constant. In this biological-time regime, the measured power consumption P of ODIN is 35 μ W at 0.55V. Therefore, the global energy per SOP $E_{\text{tot,SOP}}$ at biological time is equal to 54pJ, which is dominated by leakage (78% of the total power). These results highlight that ODIN achieves a better efficiency in accelerated-time operation thanks to leakage amortization over more SOPs per second, as shown in Fig. 2.8. In this regime, $E_{\text{tot,SOP}}$ approaches the incremental SOP energy E_{SOP} . In order to minimize the power consumption, the clock frequency of ODIN needs to be set according to the target application, taking into account the expected network activity and the required temporal resolution, which vanishes when r_{SOP} approaches its maximum value of $f_{\text{clk}}/2$. The power model of Eq. (2.2) allows evaluating this tradeoff by computing the optimal operating point of ODIN and the resulting power consumption. As the power model of Eq. (2.2) was derived at a room temperature of 24°C and 0.55V, care should be taken with the operating conditions of ODIN in the target application (e.g., the contribution of leakage increases in high-temperature operating conditions).

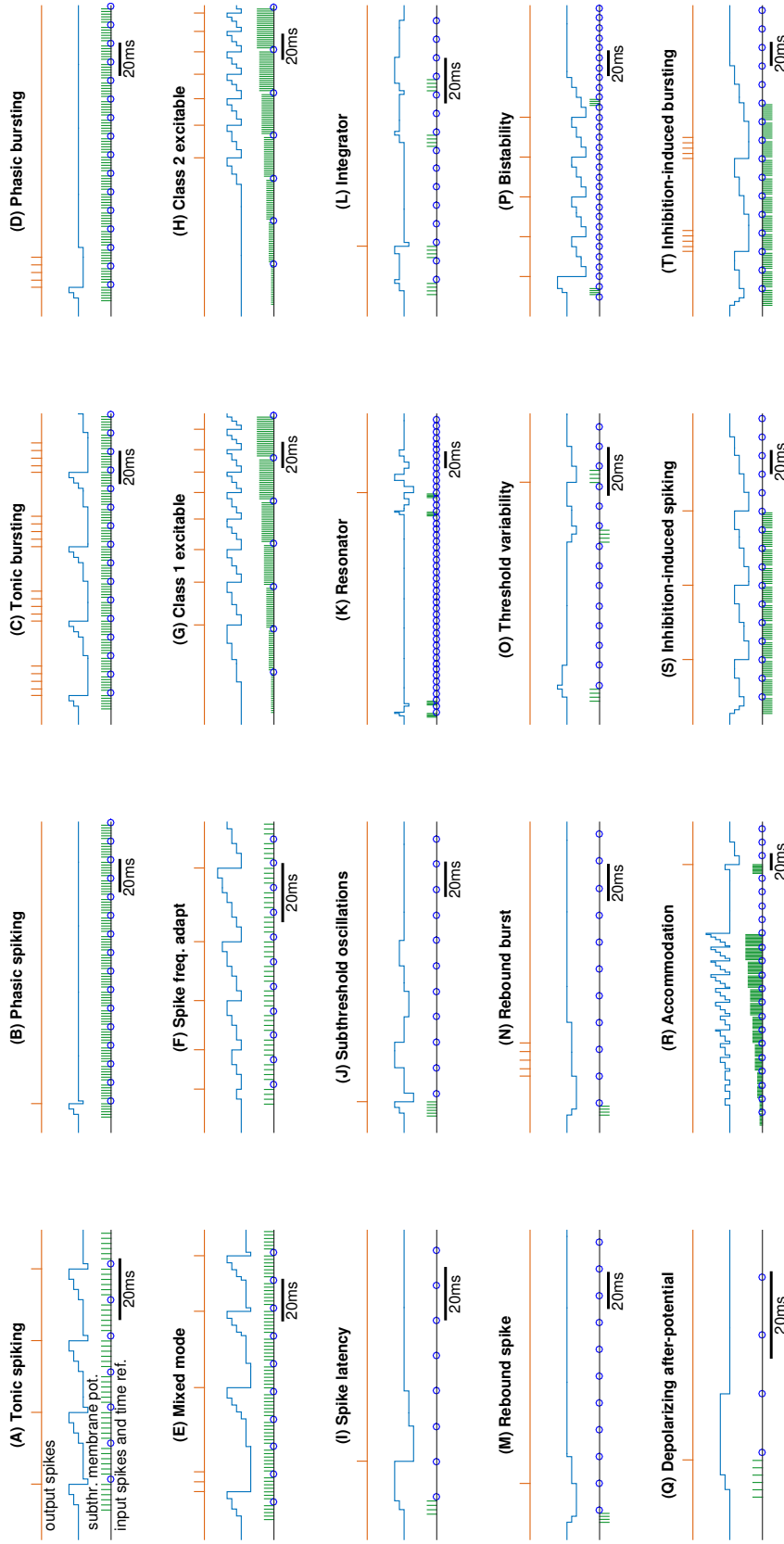


Fig. 2.9 Silicon validation of the 20 Izhikevich behaviors with biological time constant in ODIN. A 20-ms scale reference is provided next to each of the 20 behaviors and closely match those of the original Izhikevich behaviors figure in [66]. Results are in accordance with the pre-silicon validation from Fig. 1.5 in Chapter 1.

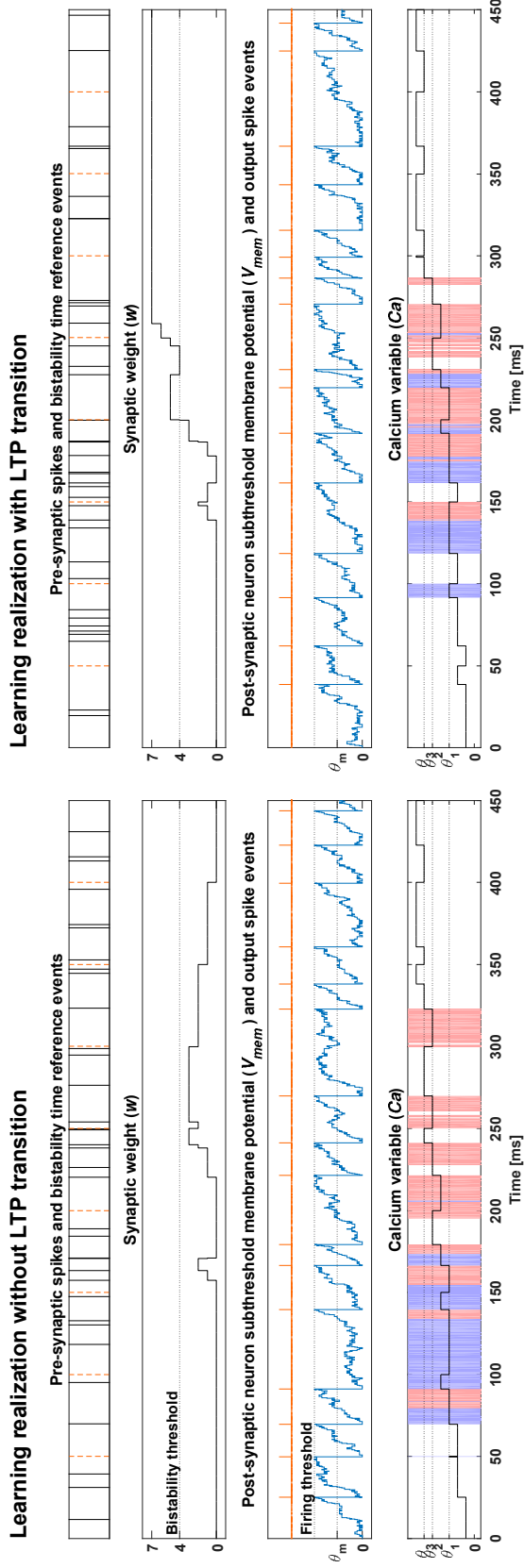


Fig. 2.10 Silicon validation of the stochastic synaptic selection mechanism of the SDSP learning rule, in accordance with the pre-silicon results from Fig. 1.10 in Chapter 1. Left: no realization of a long-term potentiation (LTP). Right: realization of an LTP after 200ms. All stimuli have been applied to a neuron in LIF configuration and generated with identical Poisson-distributed statistics: the pre-synaptic neuron fires at a rate of 70Hz, while the post-synaptic neuron fires at a rate of 40Hz. Bistability time reference events are shown in dotted orange lines at a frequency of 20Hz next to the pre-synaptic spikes. The bistability threshold at half the synaptic weight dynamic represents the value above (resp. below) which the synaptic weight increments (resp. decrements) to a high (resp. low) state upon bistability events. The SDSP conditions for weight increment and decrement from the post-synaptic neuron (up and down signals in Fig. 2.4) are shown next to the Calcium variable with red and blue areas, respectively. Non-colored areas of the Calcium variable are associated with stop-learning conditions.

2.2.2.2 Neuron and synapse characterization

Fig. 2.9 provides a silicon validation of the 20 Izhikevich behaviors in the phenomenological neuron model implemented in ODIN. Output spike patterns, subthreshold membrane potential dynamics, external input spike events and time reference events are shown with a biological time constant. The time constants of the different behaviors, indicated by a 20-ms scale reference next to each behavior, accurately match those of the original Izhikevich figure of the 20 behaviors in [66], as specified for the pre-silicon validation of the standalone neuron proposed in Chapter 1 (Fig. 1.5).

A silicon validation of the SDSP online learning mechanism is shown in Fig. 2.10. The neuron is stimulated with pre-synaptic input spikes following a Poisson distribution with a 70-Hz average rate. Neuron output spikes follow a Poisson distribution with a 40-Hz average rate. Two different learning realizations follow from stimuli generated with identical statistics, illustrating the data-induced learning stochasticity and random selection mechanism of SDSP. Indeed, on the left, no LTP occurs, while on the right an LTP transition occurs after 200ms. As for the neuron, this behavior matches the pre-silicon validation of the standalone synapse proposed in Chapter 1 (Fig. 1.10). While SDSP is an inherently unsupervised learning rule, it is possible to carry out supervised learning with the use of a *teacher* signal providing neurons with a *target* activity. A careful sizing of teacher signals and input statistics allows controlling the probability at which a neuron will potentiate or depress the synapses in its dendritic tree [106].

2.2.2.3 Comparison of online SDSP- and offline gradient-based supervised learning for spiking neural networks in ODIN

As a mapping table bit enables or disables online learning locally in each synapse (Section 2.2.1.1), the weights of a spiking neural network in ODIN can be learned either offline and off-chip (e.g., with gradient-descent-based algorithms) or online and on-chip with a low-cost implementation of SDSP that acts locally in the synapses. In order to identify the approach that is suitable for a given application, we compare these two learning strategies and highlight the associated tradeoffs. Given the limited hardware resources of ODIN, we chose the MNIST dataset of handwritten digits [150] as a support to discuss and quantify the comparison results on simple SNN topologies, the objective is therefore not to solve the MNIST problem with a record accuracy (already achieved with error rates as low as 0.2%, see [150] for a review).

The setup is presented in Fig. 2.11. The pre-processing steps applied to the MNIST dataset are downsampling 28×28 -pixel images to a 16×16 format, thus matching the number of synapses per neuron available in ODIN, together with deskewing and soft thresholding, which are common operations for small networks given the

limited neuron resources of ODIN [150]. In order to provide these images to an SNN in ODIN, one further step of converting pixel values to spikes is necessary: rate-based Poisson-distributed spike trains are used during the SDSP-based learning phase, other spike coding schemes can be explored for inference. Given that the downsampled 256-pixel MNIST images use all the 256 available neuron spike event addresses (Section 2.2.1), we chose to use a single-layer fully-connected network of 10 LIF neurons, one for each class of digit.

Fig. 2.11(a) shows the setup for offline off-chip training. As the synaptic weights of ODIN have a 3-bit resolution, offline training is carried out with quantization-aware stochastic gradient descent (SGD) following [152], as implemented in [153] using Keras with a TensorFlow backend. The chosen optimizer is Adam, which optimizes the weights by minimizing the categorical cross-entropy loss function during several epochs, each epoch consisting of one presentation of all labeled images in the training set. Care should be taken regarding two key differences between the offline-trained network and the LIF-based SNN in ODIN. First, softmax units were used during offline training, while the LIF neurons in ODIN behave as ReLU units in the frequency domain [70]. Second, offline-trained weights are in a $[-4, 3]$ range, while the synaptic weights in ODIN are in a $[0, 7]$ range. However, in the case of a single-layer network, the weights retrieved from Keras can simply be shifted to a $[0, 7]$ range and uploaded to ODIN, without further consideration: during inference, it does not impact the network prediction as the most active neuron will be the same in both cases, except if the softmax units receive a negative sum of inputs [70]. We verified that both the weight range conversion and the shift from softmax units to LIF neurons only incur a negligible accuracy drop compared to the theoretical accuracy predicted for the offline-trained ANN in Keras. However, while a direct weight upload is a valid approach for the simple single-layer structure used in this case, a specific weight conversion is necessary when resorting to more complex network topologies (e.g., multi-layer networks or convolutional neural networks), which can be carried out through dedicated ANN-to-SNN toolboxes, such as [70] or [154], or through the approach that we propose for the MorphIC neuromorphic processor in Section 2.3.2.

Fig. 2.11(b) shows the setup for on-chip online training with the embedded SDSP learning rule. Supervised learning with SDSP requires a teacher signal. During the training phase, the teacher signal is used to drive the neuron corresponding to the class of the currently applied digit to a high firing activity, while non-target neurons are driven to low firing activities. This activity in turn defines the value of the neuron Calcium variable. Fig. 2.12 shows how to properly set the target and non-target teacher signals with regard to the synaptic weight increment and decrement probabilities of the SDSP learning rule, resulting from Eq. (1.2). When MNIST characters are learned (i.e. active synapses of the target neuron have potentiated), the Calcium variable crosses the stop-learning thresholds due to increased output neuron firing activity, thus preventing overfitting. During inference, no teacher signal is applied and online learning is disabled.

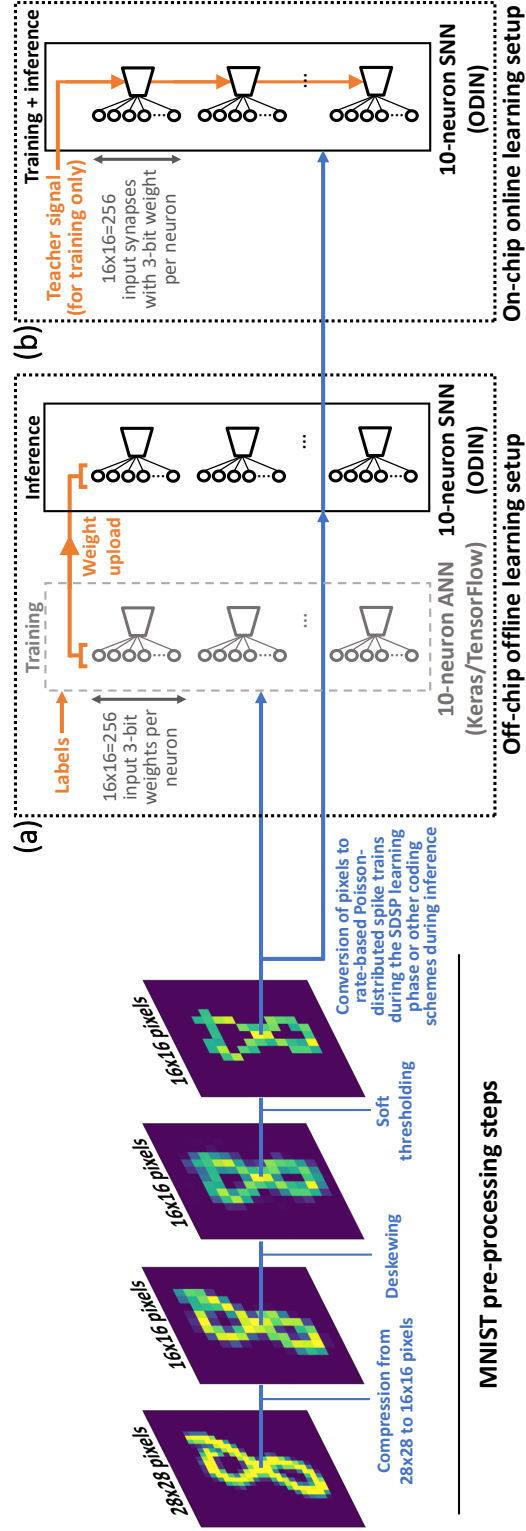


Fig. 2.11 Benchmark for testing classification accuracy on the MNIST dataset of handwritten digits. Pre-processing steps are shown on the left, while on the right are shown the two considered setups for training the weights of a LIF-based 10-neuron SNN implemented in the ODIN chip. (a) Off-chip offline weight training is carried out with quantization-aware stochastic gradient descent on a 10-neuron single-layer ANN with softmax units, implemented using Keras with a TensorFlow backend. The chosen optimizer and loss function are Adam with categorical cross-entropy. (b) On-chip online teacher-based weight training with the local SDSP learning rule.

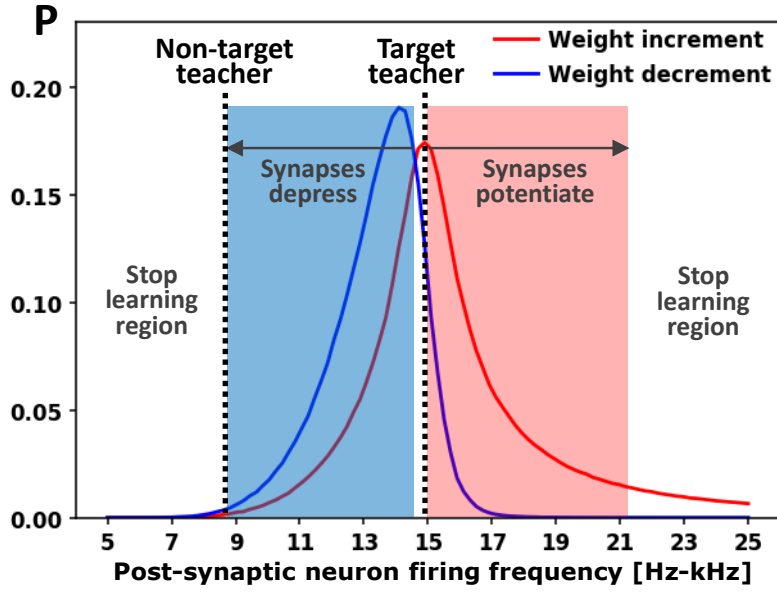


Fig. 2.12 Probability of synaptic weight increment and decrement as a function of the post-synaptic neuron firing frequency. The scale of post-synaptic neuron firing frequencies can range from biological time (Hz) to accelerated time (kHz) if the neuron time reference events (Section 2.2.1) are scaled accordingly. The teacher signal sets the minimum firing activity of the neuron, while red- and blue-shaded regions represent the contribution of input spike trains activity to the output neuron firing frequency.

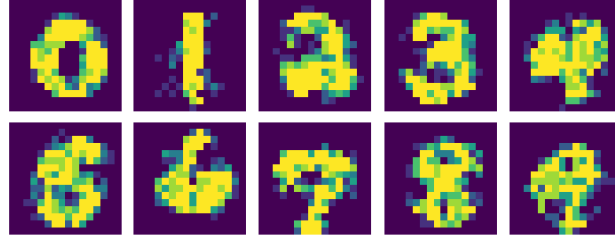


Fig. 2.13 Synaptic weights retrieved after on-chip supervised online learning of MNIST digits in ODIN with 6k training samples, leading to an accuracy of up to 85% (rate code) or 84.5% (rank order code) on the full 10k-sample test set. On-chip results correspond one-to-one with the Python-based ODIN simulator results.

The maximum attainable accuracy of the 10-neuron 3-bit-weight SNN trained online with SDSP saturates at 85% for training set sizes equal or higher than 6k samples, which corresponds to the synaptic weights retrieved from ODIN after on-chip online learning shown in Fig. 2.13. The 6k training samples were presented only once to the network, increasing the training set size or the number of presentations does not further improve the performance of the final classifier. At 0.55V, ODIN consumes 105nJ/sample during the learning phase relying on the SDSP-based supervised learning setup described previously, which includes contributions from both the teacher signal (95nJ) and the input character (10nJ). A higher accuracy

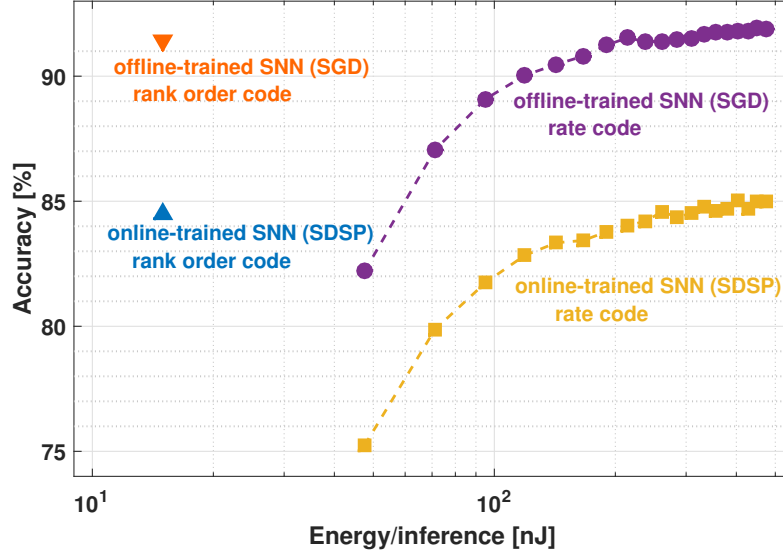


Fig. 2.14 Accuracy vs. energy per inference tradeoff for the rank order code and the rate code on the MNIST dataset. For the rate code, this tradeoff depends on the character presentation time: the longer the presentation, the higher the accumulated statistics and the consumed energy. For the rank order code, the character presentation is stopped as soon as the decision is made by a single output spike, leading to an efficient spike use. Results are provided for the ODIN chip using either weights learned off-chip offline with stochastic gradient descent (SGD) or weights learned on-chip online with SDSP, following the setup from Fig. 2.11.

of 91.9% can be reached in ODIN using offline SGD-based training with all the 60k MNIST training samples over 100 epochs (the theoretical accuracy obtained in Keras, before the weight upload to ODIN, was 92.5%).

During the inference phase, different coding schemes can be explored for the weights learned either online with SDSP or offline with SGD, resulting in different tradeoffs between accuracy and energy per inference, as shown in Fig. 2.14. As outlined in Section 1.1.1, on the one hand, the rate code is a standard spike coding approach in which pixel values are encoded into spike rates, the output neuron that spikes the most defines the inferred class. In this case, ODIN consumes 404nJ/inference after on-chip learning with 85% accuracy and 451nJ/inference after off-chip learning with 91.9% accuracy. Input characters must be presented for a much longer time than during the on-chip learning phase in order to discriminate them, sufficient statistics need to be accumulated in the output neurons. The rate code is thus inefficient in its spike use. On the other hand, the rank order code gives a promising approach for a low-power spike coding alternative that is easy to use with LIF neurons. In the rank order code, each character is provided to ODIN as a sequence of successive spikes where no relative timing is involved, each pixel spikes once and all pixels are arranged in the order of decreasing values in the sequence. The sequence is then presented repeatedly to ODIN until one output neuron spikes, this

first-spiking neuron defines the inferred class. In this case, ODIN consumes only 15nJ/inference for 84.5% accuracy after on-chip learning (resp. 91.4% accuracy after off-chip learning), which demonstrates a high efficiency. To the best of our knowledge, this is the first time MNIST classification is demonstrated with on-chip SDSP-based learning. Therefore, Fig. 2.14 highlights that, on the one hand, the SDSP learning rule allows carrying out spiking neural network training on-chip and online with low hardware resources at the cost of a limited accuracy degradation. It corresponds to applications that are constrained in power and resources during both the training and the inference phases and where data is presented on-the-fly to the classifier. On the other hand, resorting to the use of an off-chip learning engine is the optimum approach for applications in controlled environments that are not power- or resource-constrained during the training phase: it allows reaching a higher accuracy while keeping power efficiency during the inference phase.

Given more neuron resources (e.g., by connecting several ODIN chips together), a higher accuracy on MNIST could be attained with more complex network topologies. We will demonstrate this idea with the quad-core 2k-neuron MorphIC chip in Section 2.3. In the case of online SDSP-based learning, two-layer reservoirs and unsupervised winner-take-all (WTA) networks are interesting topologies. Both of these network topologies have recently been studied for subthreshold analog implementations of SDSP (e.g., see [39] and [155], respectively). If high accuracy is needed through global gradient-based optimization with deep multi-layer networks, the weights from an offline-learned ANN topology can be mapped to an SNN in ODIN. As the synaptic weights of ODIN can be individually configured to be static or plastic, hybrid networks with SDSP-based plastic and pre-trained static layers can also be investigated. A basic network architecture can thus be tuned online in order to adapt to the environment, which can be related to transfer learning [156].

2.3 MorphIC – A 65-nm quad-core neuromorphic processor demonstrating large-scale connectivity with record binary online-learning synapse density

It has been shown in Section 1.2 that synapses are the limiting factor in neuromorphic designs due to the high connectivity of biological neural networks [46]. We demonstrated record synaptic densities with the efficient SDSP-based 4-bit synapse design of ODIN in Section 2.2, however ODIN only embeds 256 neurons and 64k synapses, corresponding to fan-in and fan-out values of 256. Therefore, new techniques are required to further reduce the synaptic footprint and achieve the high fan-in and fan-out values found in biology. A promising approach currently being investigated in the field of conventional ANN and CNN neural network accelerators is weight quantization, sometimes pushed to extreme levels with binarization. Beyond a minimization of the memory footprint, it avoids the high energy cost of off-chip

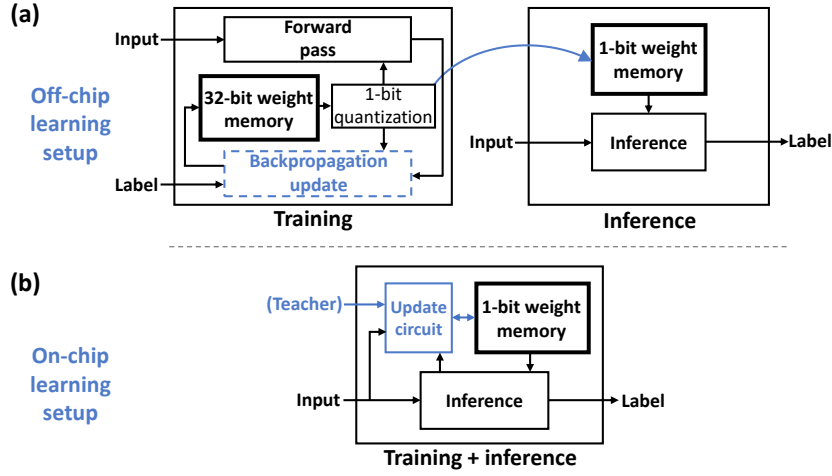


Fig. 2.15 Learning strategies for binary-weight neural networks. (a) Quantization-aware off-chip learning setup: binary weights are used during the forward pass while full-resolution weights are kept for backpropagation updates [152]. Training is carried out in an off-chip high-performance optimizer, while inference is carried out in the power- and resource-constrained device. (b) On-chip online learning setup, where data-driven weight updates are carried out in parallel with inference in the power- and resource-constrained device. A teacher signal is required for supervised online learning, whereas teacher-less learning is unsupervised.

memory accesses if all the weights can be stored into on-chip memory [153]. The accuracy drop induced by quantization can be mitigated to acceptable levels for many applications with the use of quantization-aware training techniques that propagate binary weights during the forward pass and keep full-resolution weights for backpropagation updates [152]. The associated off-chip learning setup for quantization-aware training is shown in Fig. 2.15(a): this strategy allows binary-weight neural networks to perform inference with a favorable energy-area-accuracy tradeoff, as recently demonstrated by binary CNN chips (e.g., [18, 157, 158]).

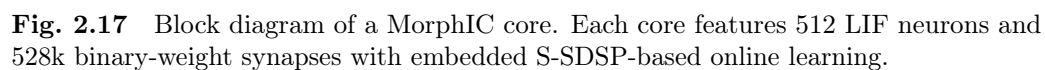
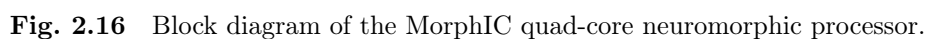
However, only three chips have been proposed previously to leverage the density and power advantage of binary weights with SNNs. The first one is the IBM TrueNorth chip, the largest-scale neuromorphic chip with 1M neurons and 256M 1-bit synapses, but it does not embed online learning [33]. The second one is the Intel Loihi chip, it has a configurable synaptic resolution that can be reduced to 1 bit and embeds a programmable co-processor for on-chip learning, though not demonstrated using a binary synaptic resolution to the best of our knowledge [34]. The third one is the chip from Seo *et al.*, which embeds stochastic STDP (S-STDP) for online learning in binary synapses [25]. However, S-STDP requires the design of a custom transpose SRAM memory with both row and column accesses, which severely degrades the density advantage of their approach (Section 1.2.2). Therefore, in order to leverage the biological plausibility and efficiency of SDSP with binary weights, we propose a stochastic SDSP (S-SDSP) learning rule, which maintains compatibility with single-port foundry SRAMs for high-density synaptic online learning integration.

Beyond plasticity, a second key aspect of SNNs lies in connectivity. The brain organization follows *small-world networks* with dense local connectivity and sparse long-range wiring, leading to an efficient clustering of neuronal activity and hierarchical information encoding [159]. Network-on-chip (NoC) design applied to multi-core SNNs is thus an active research topic [29, 30, 32–34, 160, 161]. We propose a hierarchical combination of mesh-based routing for inter-chip connectivity, star-based routing for intra-chip inter-core connectivity and crossbar-based routing for local intra-core connectivity. We store all the connectivity information locally in the neuron memory to enable the design of memory-less routers that do not require local mapping table accesses. With only 27 connectivity bits per neuron, this low-memory hierarchical connectivity allows reaching biologically-realistic fan-in and fan-out values of 1k and 2k neurons, respectively.

We demonstrate this two-fold plasticity and connectivity approach with MorphIC, a quad-core digital neuromorphic processor. MorphIC was prototyped in 65-nm CMOS and embeds 2k LIF neurons and more than 2M synapses in an active silicon area of 2.86mm^2 , therefore achieving a high density of 738k 1-bit online-learning synapses per mm^2 . It results in an order-of-magnitude density improvement compared to the only previously-proposed binary-weight online-learning SNN processor from Seo *et al.* [25]. On the MNIST image recognition task [150], MorphIC achieves a test set accuracy of 97.8% with offline-trained binary weights. It demonstrates an order-of-magnitude improvement in the area-accuracy tradeoff compared to other SNNs, while having no penalty in the energy-accuracy tradeoff using rank order coding. Embedded online learning is validated by learning to discriminate eight patterns with S-SDSP. The architecture and implementation of MorphIC are presented in Section 2.3.1, together with detailed descriptions of the hierarchical event routing infrastructure and of the S-SDSP learning rule. The specifications, measurements and benchmarking results are provided in Section 2.3.2.

2.3.1 Architecture and implementation

A block diagram of the synchronous digital MorphIC quad-core spiking neuromorphic processor is shown in Fig. 2.16, illustrating its hierarchical routing fabric for large-scale chip interconnection. Level-2 (L2) routers handle inter-chip connectivity, level-1 (L1) routers handle inter-core connectivity and level-0 (L0) routers handle intra-core connectivity (Section 2.3.1.1). The clock can be either provided externally or generated internally using a configurable-length ring oscillator. A block diagram of the MorphIC core is shown in Fig. 2.17: each core embeds 512 LIF neurons configured as a crossbar array with 256k L0 1-bit synapses and 256k L1 1-bit synapses, while 16k L2 synapses can be accessed independently. Each synapse embeds online learning with a stochastic implementation of the spike-dependent synaptic plasticity (S-SDSP) learning rule (Section 2.3.1.2). Each axon can be configured to multiply its associated synaptic weights by a factor of 1, 2, 4 or 8. Time multiplexing is used to increase the neuron and synapse densities by using shared



update circuits and storing neuron and synapse states to local SRAM memory, based on the strategy we previously proposed for ODIN in Section 2.2. Fig. 2.18 illustrates the time-multiplexed crossbar operation of a MorphIC core when it processes a spike event from a neuron in the local core (L0 connectivity) or from a neuron in another core in the same chip (L1 connectivity). The core controller goes

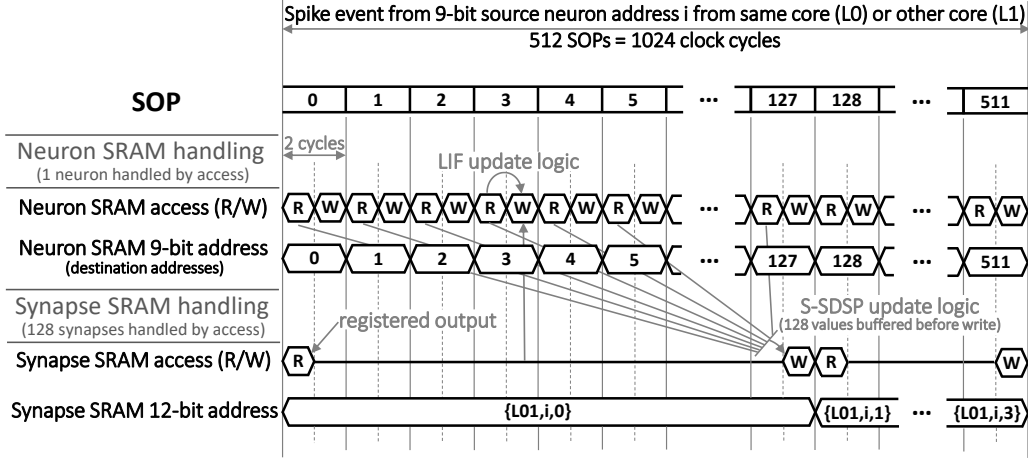


Fig. 2.18 Timing diagram of the crossbar operation in a MorphIC core, adapted from the time multiplexing scheme proposed for ODIN in Section 2.2. It illustrates the time-multiplexed crossbar operation for a spike event from 9-bit source neuron address i , leading to 512 SOPs, where each SOP lasts two clock cycles. The core controller goes sequentially through all the local 512 neurons, it first reads their state in the local SRAM memory and then writes back the updated state retrieved from the LIF neuron update logic. The synapse SRAM has 128-bit words for density purposes: as MorphIC has 1-bit synapses, 128 synapses are handled by access and stochastic SDSP (S-SDSP) updates are buffered before being written back to the synapse SRAM memory. Depending on whether the source neuron is located in the local core (L0 connectivity) or on another core from the same MorphIC chip (L1 connectivity), the MSB of the synapse SRAM address (L01 flag bit) selects whether L0 or L1 synapses are accessed.

sequentially through all the 512 local neurons, leading to 512 SOPs, and handles the local SRAM memory accesses accordingly. As L2 events target a specific synapse of a neuron (Section 2.3.1.1), they lead to a single SOP.

2.3.1.1 Hierarchical event routing

Clustering groups of neurons with dense local and sparse long-range connectivity allows minimizing the memory requirements while keeping flexibility and scalability [29]. This organization is found in the brain and is known as *small-world networks*. Hierarchy is therefore a key concept in SNN event routing infrastructures for large-scale networks [29, 30, 32–34, 160, 161]. MorphIC uses a heterogeneous hierarchical routing fabric with different router types at each level, as shown in Fig. 2.19: the L2 router follows a unicast mesh-based dimension-ordered destination-driven operation, the L1 router follows a multicast star-based source-driven operation, while the L0 router handles decoding and encoding of the different packet types for local core crossbar-based processing. The individual routing levels and their combination do not contain cyclic path dependencies and are thus deadlock-free. This heterogeneous event routing infrastructure allows for the three connectivity pat-

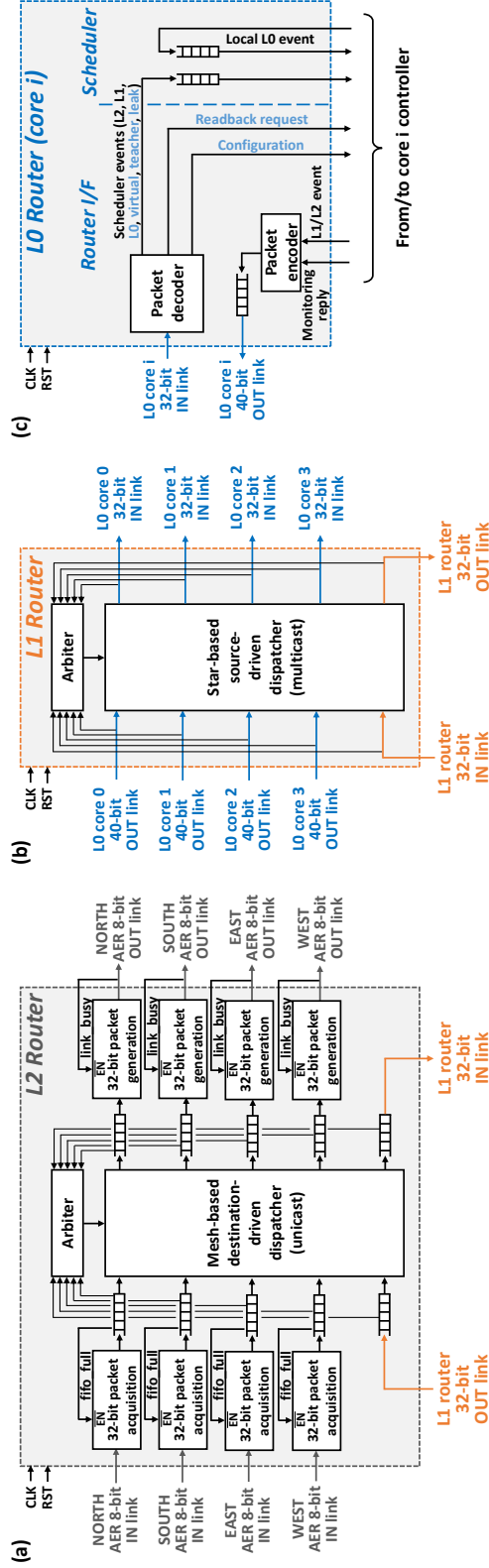


Fig. 2.19 Architecture of the hierarchical three-level event routing fabric of MorphIC. (a) The level-2 (L2) router handles high-level inter-chip connectivity with four bidirectional address-event-representation (AER) links, events are dispatched following a unicast mesh-based strategy. Packet buffering in FIFOs ensures that all links can operate independently. (b) The level-1 (L1) router handles mid-level intra-chip inter-core connectivity with four local links, one for each MorphIC core. Events are dispatched following a multicast star-based strategy. (c) The level-0 (L0) router handles low-level connectivity, it decodes incoming packets and sorts them toward either the controller or the scheduler of the local core. When a local neuron configured for L1 and/or L2 outward connectivity spikes, all its connectivity information is encapsulated in a routing packet before exiting the L0 router. Event types indicated in light blue are testbench-type events that cannot be generated by MorphIC chips.

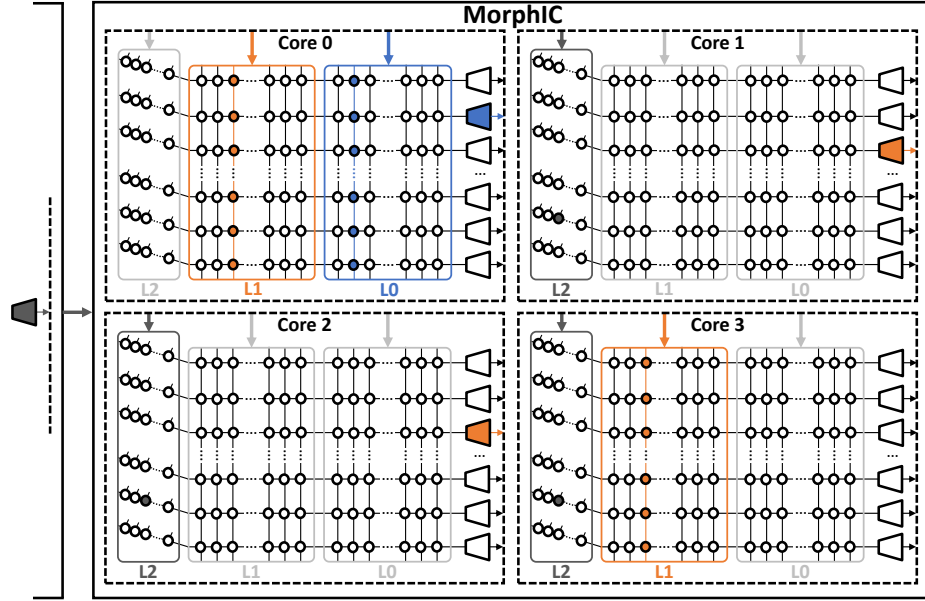


Fig. 2.20 Examples of L0, L1 and L2 connectivity handling at the core level. Blue: L0 connectivity inside core 0, following a typical crossbar operation. Orange: L1 connectivity from neurons in cores 1 and 2 to cores 0 and 3, following a crossbar operation in the destination cores. In this example, as the source neurons have identical 9-bit addresses, they map to the same L1 synapses in the destination cores. Gray: L2 connectivity from a neuron in another MorphIC chip to a specific L2 synapse of a target neuron, broadcasted to cores 1, 2 and 3 of the destination chip.

terns illustrated in Fig. 2.20, depending on the source neuron location and outward connectivity:

- The source neuron targets neurons in the same core (L0 connectivity): the time-multiplexed crossbar approach of Fig. 2.18 is followed with the local L0 synapses (e.g., blue pattern in core 0 in Fig. 2.20).
- The source neuron targets neurons in any combination of other cores in the same chip (L1 connectivity): the time-multiplexed crossbar approach of Fig. 2.18 is followed with the L1 synapses of the destination cores. The same L1 synapses are shared with up to three cores (e.g., orange pattern from source neurons in cores 1 and 2 to destination cores 0 and 3 in Fig. 2.20).
- The source neuron is located in another MorphIC chip (L2 connectivity): the target is a specific L2 synapse address in any combination of cores in one destination chip (e.g., gray pattern from a source neuron retrieved from the *West* link toward identical L2 synapse addresses in cores 1, 2 and 3 in Fig. 2.20). As each neuron has 32 L2 synapses, an L2 synapse address has a width of 14 bits (9 bits for the neuron, 5 bits for the L2 synapse).

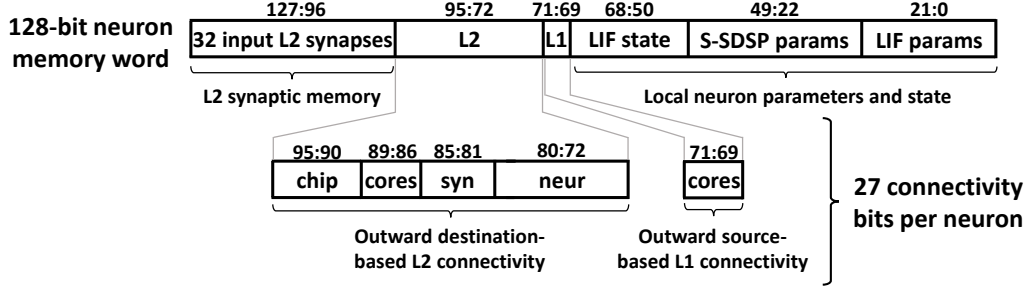


Fig. 2.21 Neuron memory map: structure of a 128-bit word in the neuron SRAM memory. Each word contains the parameters, state, outward connectivity and the 32 1-bit input L2 synapses of a neuron. At runtime, only the LIF state (i.e. an 11-bit membrane potential, a 4-bit Calcium variable and a 4-bit counter to emulate Calcium leakage) and the L2 synapses can be modified by the LIF and S-SDSP update logic blocks (Fig. 2.17), respectively. The L2 and L1 connectivity fields occupy a total of only 27 bits per neuron.

Each neuron of MorphIC can use any combination of the aforementioned three types of L0, L1 and L2 connectivities, which allows reaching a fan-in of 512 (L0) + 512 (L1) + 32 (L2) and a fan-out of 512 (L0) + 3×512 (L1) + 4 (L2).

The entire connectivity of a network of MorphIC chips is determined by only 27 connectivity bits per neuron, which are stored in the neuron 8-kB SRAM memories located inside each core (Fig. 2.17). It consists of 512 128-bit words, one word for each of the 512 LIF neurons per core, whose structure is outlined in Fig. 2.21. Destination-based L2 connectivity requires 24 bits in total: the 6-bit *chip* field stores 3-bit dx and dy fields encoding the destination chip, the 4-bit *cores* field encodes the combination of target cores, while the 5-bit *syn* and the 9-bit *neur* fields encode the 14-bit L2 synapse address. Source-based L1 connectivity requires only 3 bits per neuron in order to target any combination of the other cores in a MorphIC chip. Except if disabled in the core parameter bank, L0 crossbar connectivity is automatic and does not require further connectivity information. As all the connectivity information is decentralized next to the neurons and then encapsulated in the event packets, the routers do not require local or external mapping tables: they are memory-less beyond simple packet buffering. Let us now discuss the architectural details of the L2, L1 and L0 routers.

Level-2 (L2) router The L2 router (Fig. 2.19(a)) handles high-level inter-chip connectivity with four links along the *North*, *South*, *East* and *West* directions that operate independently and in parallel. Events from/to the four chip-level links and from/to the L1 router are buffered into FIFOs before being dispatched following a standard unicast mesh-based strategy with dimension-ordered routing (i.e. x direction before y direction). Two dx and dy fields in the chip-level packet contain the information necessary for destination-based routing. dx and dy have a 3-bit width each (one sign bit, two data bits), which allows routing packets to

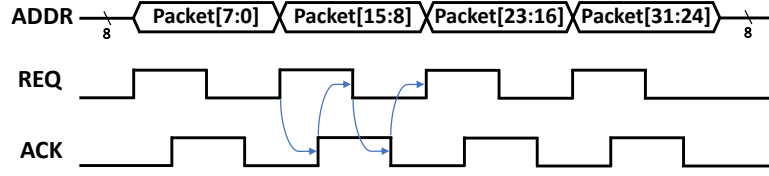


Fig. 2.22 32-bit packet transmission multiplexed into four 8-bit AER transactions at the L2 level. As a double-latching synchronization barrier is used on the receiver REQ line, the ADDR data can be asserted at the same time as the REQ line on the sender side.

up to three MorphIC chips in any direction. At each *East* or *West* (resp. *North* or *South*) hop, the L2 router decrements the value of the dx (resp. dy) data field. When both dx and dy are zero, the packet is then forwarded to the L1 router. Distance information d is also maintained separately in the event packet: d is 0 for local L0 events and 1 for events received from local L1 connectivity, it then increases for each L2 hop up to a maximum of 7 for events received from a chip located at $dx=\pm 3$ and $dy=\pm 3$. As synapses at all routing levels of MorphIC embed online learning (Section 2.3.1.2), the probability of synaptic weight update can be modulated by the distance information, following a small-world network modeling strategy. To the best of our knowledge, MorphIC is the first neuromorphic processor to embed online hierarchical learning.

The mesh-based dispatcher is controlled by an arbiter, which can be configured either for round-robin or for priority-based operation. Round-robin operation, by cycling through each link independently of the FIFO usage, guarantees a maximum latency for packet processing, while priority-based operation is a greedy approach that allocates processing time to the most active links based on FIFO usage.

Links in each direction consist of two AER buses, a sender and a receiver, for a total of eight AER buses per MorphIC chip. The MorphIC design being pad-limited, the width of the AER buses has been reduced to 8 bits. Transmission and reception of 32-bit event packets are thus multiplexed into four 8-bit AER transactions, as illustrated in Fig. 2.22. In order to ensure an asynchronous operation of the AER buses between MorphIC chips, double-latching synchronization barriers have been placed on the receiver REQ and sender ACK handshake lines to limit metastability issues. As the pads are the speed bottleneck for off-chip L2 packet routing (Table 2.2), L2 packet activity should be sparse compared to L1 and L0 activity: L2 events should thus represent high-level features, as illustrated in the experiments outlined in Section 2.3.2. The L2 routing speed could be improved by using a two-phase-handshake AER variant instead of the standard four-phase handshake.

Level-1 (L1) router The L1 router (Fig. 2.19(b)) handles mid-level intra-chip inter-core connectivity with the four local MorphIC cores. This router is based on a star topology and relies on a simple dispatcher that multicasts events to local

cores following a source-based approach. It does not contain any FIFO buffering as awaiting packets are already buffered in the L2 and L0 routers. An arbiter controls the dispatcher following a configurable round-robin or greedy priority-based operation, similarly to the L2 router.

The L1 router is at the center of the hierarchy. For neuron events from local cores (i.e. ascending-hierarchy events), it handles multicasting to any combination of other cores toward L1 synapses and/or forwarding to the L2 router toward another MorphIC chip. For events retrieved from the L2 router (i.e. descending-hierarchy events), it handles multicasting to any combination of MorphIC cores toward L2 synapses.

Level-0 (L0) router The L0 router (Fig. 2.19(c)) handles low-level intra-core connectivity. This router is divided into two blocks: an interface and a scheduler. The interface handles packet decoding and encoding from/to the L1 router. The packet decoder segments input packets into different types:

- *configuration* packets are used to program the local neuron and synapse SRAMs and the core parameter bank (Fig. 2.17), they are handled by the controller,
- *monitoring request* packets query one byte from the neuron or synapse SRAM, they are handled by the controller,
- *scheduler* events are buffered by a FIFO in the core scheduler, they include *L2* events targeting a single L2 synapse, *L1* events targeting L1 synapses, *L0* events targeting L0 synapses, *virtual* events that directly update a neuron without accessing any physical synapse, *teacher* events that control the S-SDSP supervision mechanism through the neuron Calcium variables (Section 2.3.1.2) and *leak* events that drive the LIF leakage time constant.

Locally-generated *L0* events are buffered directly in a scheduler FIFO, they are not visible from the L1/L2 router hierarchy. Locally-generated events that need to go up the router hierarchy are handled by the packet encoder:

- *monitoring reply* packets contain the neuron or the synapse state byte previously queried by a *monitoring request* packet,
- *L1/L2* events forward the L1 and L2 connectivity information of a source neuron to the L1 router.

2.3.1.2 Stochastic spike-dependent synaptic plasticity (S-SDSP)

We showed with ODIN in Section 2.2.1.1 that time-multiplexed SDSP implementations allow for low-overhead high-density integration of online learning in 3-bit-weight synapses. However, as SDSP relies on discrete positive and negative

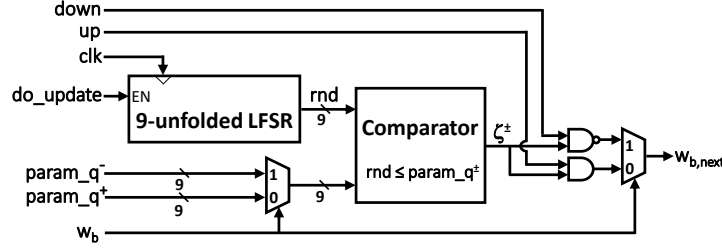


Fig. 2.23 Time-multiplexed S-SDSP update logic. *up* and *down* signals represent the values of the S-SDSP update conditions in Eq. (2.4).

steps (Eq. (2.1)), it cannot be applied directly to binary weights. Senn and Fusi proposed a bio-inspired stochastic learning rule for binary synapses in [162], where the update conditions rely on the total synaptic input of the post-synaptic neuron at the time of the pre-synaptic spike. However, this information is not easily available in time-multiplexed implementations: as shown in Fig. 2.18, the *destination* neurons are processed sequentially, while obtaining the total post-synaptic input of a neuron would require sequential processing of the *source* neurons instead, which is incompatible with an event-driven operation. Therefore, we propose a stochastic spike-dependent synaptic plasticity (S-SDSP) learning rule suitable for binary weights, as formulated in Eq. (2.4). It results from the fusion of the stochastic mechanism proposed in [162] with the SDSP update conditions from Brader *et al.* [106]. ζ^+ and ζ^- are binary random variables with probabilities q^+ and q^- of being at 1, respectively. The synaptic weight w_b therefore goes from 0 to 1 (resp. from 1 to 0) with probability q^+ (resp. q^-), depending on the update conditions. The Calcium concentration is implemented as a 4-bit variable, it is stored next to all S-SDSP parameters in the neuron SRAM (Fig. 2.21).

$$\begin{cases} w_b \rightarrow w_b + \zeta^+(1 - w_b) & \text{if } V_{\text{mem}}(t_{\text{pre}}) \geq \theta_m, \theta_1 \leq \text{Ca}(t_{\text{pre}}) < \theta_3 \\ w_b \rightarrow w_b - \zeta^- w_b & \text{if } V_{\text{mem}}(t_{\text{pre}}) < \theta_m, \theta_1 \leq \text{Ca}(t_{\text{pre}}) < \theta_2 \end{cases} \quad (2.4)$$

The proposed S-SDSP update logic is shown in Fig. 2.23. The binary random variables $\zeta^\pm$ can be generated with $q^\pm$ probabilities using linear feedback shift register (LFSR)-based pseudo-random number generation. In order to generate $q^\pm$ with a resolution similar to the probabilities down to 0.01 used in [162], approximately 6 bits of resolution are required. Distance-based modulation of $q^\pm$ from small-world network modeling requires another 3 bits of resolution as the distance information ranges from 0 to 7 (Section 2.3.1.1). Therefore, we selected a 9-bit resolution for $q^\pm$ probabilities. As S-SDSP updates must be computed in a single clock cycle, it is possible to parallelize successive iterations of an LFSR by using the unfolding algorithm from [163], as suggested in [164] to avoid instantiating parallel LFSRs and save switching power. The number of parallelized successive iterations is governed by the unfolding factor, which is 9 in this case. The unfolding process

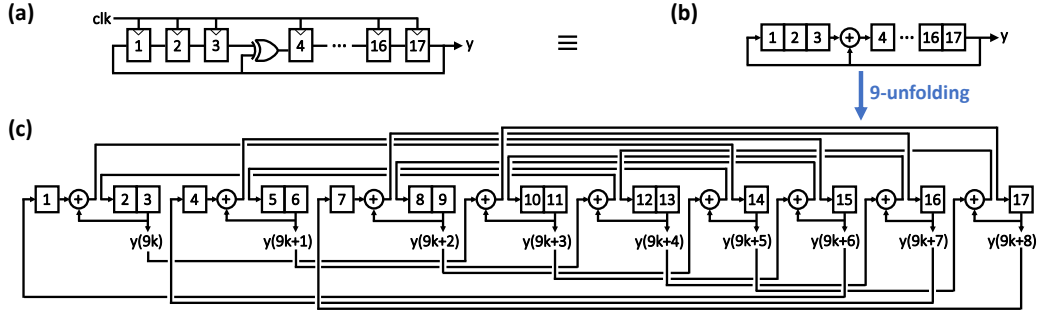


Fig. 2.24 (a) Circuit diagram of a Galois 17-bit LFSR with characteristic polynomial $x^{17} + x^3 + 1$. (b) Equivalent compact representation. (c) 9-unfolded 17-bit Galois LFSR for single-cycle 9-bit pseudo-random word generation for the S-SDSP online learning rule.

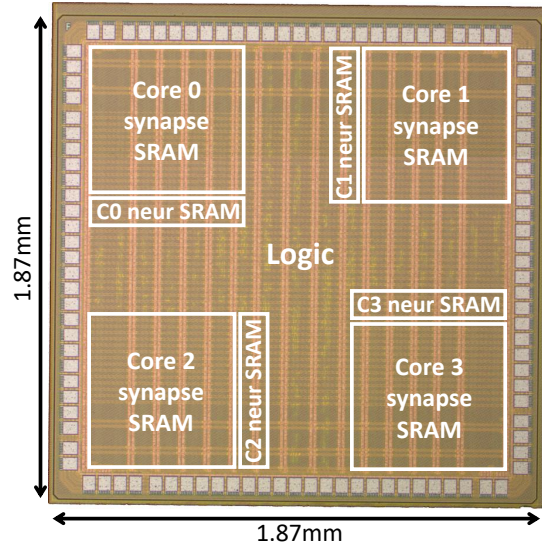


Fig. 2.25 MorphIC chip microphotograph, illustrating the floorplan of the neuron and synapse SRAM macros of each core.

and the resulting unfolded LFSR are illustrated in Fig. 2.24. Unfolding leads the combinational logic resources (here, a single XOR gate) to be multiplied by the unfolding factor, while the LFSR period is divided by the unfolding factor. In order to avoid inducing correlation between synapses, the period of the unfolded LFSR must be one order of magnitude higher than the number of synapses per neuron. We thus selected a 17-bit depth for the LFSR to be unfolded (Fig. 2.24(a-b)). The 9-unfolded LFSR is shown in Fig. 2.24(c). The overhead incurred by the resulting S-SDSP update logic is negligible as it is shared with time multiplexing for all the L0, L1 and L2 synapses in a MorphIC core.

Table 2.2 Specifications and measurements of MorphIC.

Technology	65nm LP CMOS	
Implementation	Digital	
Area	3.50mm ² (chip, incl. pads)	
	2.86mm ² (chip, excl. pads)	
	1.59mm ² (SRAM macros)	
Total SRAM memory (type)	256kB (syn), 32kB (neur)	
Number of cores	4	
Total # neurons (type)	2048 (LIF)	
Total # synapses (hier.)	1M (L0), 1M (L1), 64k (L2)	
Fan-in (hier.)	512 (L0), 512 (L1), 32 (L2)	
Fan-out (hier.)	512 (L0), 3×512 (L1), 4 (L2)	
Online learning	Stochastic SDSP, 1-bit weights	
Time constant	Biological to accelerated	
Supply voltage	0.8V	1.2V
Max. clock frequency	55MHz	210MHz
Max. acceleration [°]	10×	40×
Leakage power (P_{leak})	45μW	190μW
Idle power (P_{idle})	41.3μW/MHz	94.0μW/MHz
Energy per SOP (E_{SOP})	30pJ	65pJ
Energy per L2 hop [†]	9.0pJ	20.3pJ
Energy per L1 hop*	1.7pJ	3.8pJ
L2 router bandwidth (AER)	2.3Mpackets/s/link	5.7Mpackets/s/link [†]
L1 router bandwidth	55Mpackets/s	210Mpackets/s
Core bandwidth (max. r_{SOP})	27.5MSOP/s/core	105MSOP/s/core

[°] Compared to biological-time processing with all neurons assumed to spike at 10Hz.[†] Excluding IO power. * Simulation results. † Limited by pad delay at high speed.**Table 2.3** Silicon area breakdown (logic and SRAM without I/O pads).

Cores	Synapses	66.27%
	Neurons	16.61%
	Parameters	5.88%
	Controller	0.36%
Routers	L2	5.47%
	L1	0.18%
	L0	5.19%
Others	Clock generator	0.04%

2.3.2 Specifications, measurements and benchmarking results

MorphIC was prototyped in the UMC 8-metal 65-nm low-power (LP) CMOS process. A chip microphotograph is presented in Fig. 2.25, while specifications and measurement results are provided in Table 2.2. A detailed area breakdown is provided in Table 2.3. As derived in Section 2.2.2.3 for ODIN, the power consumption P of MorphIC can be modeled by

$$P = P_{\text{leak}} + P_{\text{idle}} \times f_{\text{clk}} + E_{\text{SOP}} \times r_{\text{SOP}}, \quad (2.5)$$

where P_{leak} is the leakage power, P_{idle} is the idle power (i.e. active clock, without network activity), E_{SOP} is the incremental energy per SOP, f_{clk} is the clock

Table 2.4 Summary of simple architectural and implementation improvements to MorphIC that would allow further reducing the MNIST energy per inference at 0.8V, 55MHz.

Optimization	E_{leak}	E_{idle}	E_{SOPs}	E_{infer}
MorphIC, rank order code, no improvement	$0.18\mu\text{J}$	$9.24\mu\text{J}$	$12.4\mu\text{J}$	$21.8\mu\text{J}$
+ crossbar opt.	$0.13\mu\text{J}$	$6.72\mu\text{J}$	$9.15\mu\text{J}$	$16.0\mu\text{J}$
+ sym. weight opt.	$0.09\mu\text{J}$	$4.48\mu\text{J}$	$6.10\mu\text{J}$	$10.7\mu\text{J}$
+ clock gating opt.	$0.09\mu\text{J}$	$2.02\mu\text{J}$	$6.10\mu\text{J}$	$8.2\mu\text{J}$

frequency and r_{SOP} is the SOP processing rate. Based on Table 2.2, MorphIC consumes a total energy of 51pJ per SOP at 0.8V when including the leakage and idle power contributions at maximum f_{clk} and r_{SOP} (i.e. 55MHz and 110MSOP/s using all four cores, each SOP taking two clock cycles as shown in Fig. 2.18).

Offline learning performance with quantization-aware training can be demonstrated with the MNIST dataset of handwritten digits [150]. Using the four cores of MorphIC and all the available neuron resources with the network topology shown in Fig. 2.26, an accuracy of 97.8% is reached using conventional rate-based coding (Section 1.1.1). As the synaptic weights trained in Keras have -1 and $+1$ values while the MorphIC synapses have 0 and 1 values instead, it is necessary to compensate for the asymmetric weight distribution of MorphIC. This compensation can be made in a layer by subtracting half the sum of its inputs, which can be achieved by layerwise inhibitory neurons connected with weight 1 to all the layer inputs and having a firing threshold of 2, as shown in Fig. 2.26. As the rate code is inefficient in its spike use, it results in a high energy per classification of $205\mu\text{J}$ at 0.8V and 55MHz. It has been shown with ODIN in Section 2.2.2 that the rank order code (i.e. values are encoded in the order in which neurons spike) is a simple yet much more efficient coding strategy than rate coding. The inferred class can be retrieved from the neuron in the output layer that spikes first. Using rank order coding, MorphIC consumes 5.45mW for 250 classifications per second at 0.8V and 55MHz, which allows reaching a 10-fold energy improvement down to $21.8\mu\text{J}$ per classification, at the expense of a drop of 1.9% in accuracy.

Following the MorphIC power model from Eq. (2.5), we can break down the MNIST rank-order energy per inference E_{infer} of $21.8\mu\text{J}$ at 0.8V and 55MHz as follows: $E_{\text{infer}} = E_{\text{leak}} + E_{\text{idle}} + E_{\text{SOPs}}$. Three simple improvements to the MorphIC architecture or implementation would allow reducing these contributions while maintaining the rank-order classification accuracy of 95.9%, as detailed in Table 2.4. First, while crossbar operation is highly-efficient for the implementation of fully-connected layers, systematic processing of all neurons in the array for each input spike can lead to a lot of dummy operations (e.g., as shown in Fig. 2.26, a spike from a hidden layer neuron should lead to only 10+1 SOPs to process the output layer neurons, not 512). This could be improved by adding only two 9-bit parameters

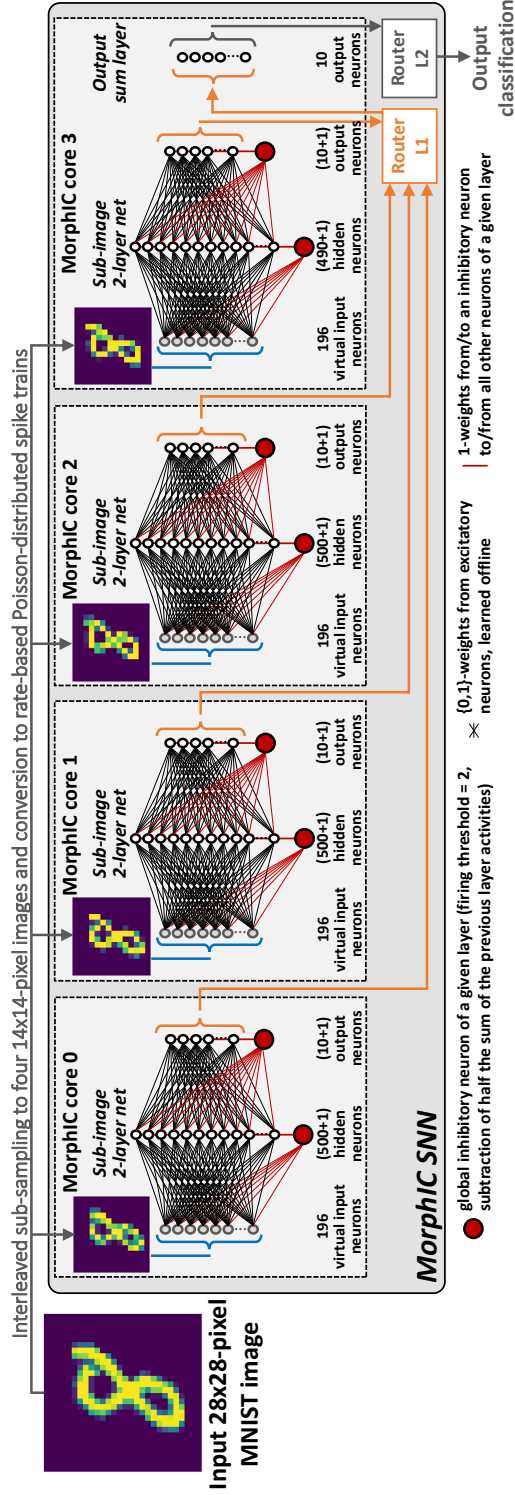


Fig. 2.26 MNIST classification setup. Input images are split with interleaved sub-sampling into four independent 14×14 images. The sub-image pixels are converted to rate-based Poisson-distributed spike trains and sent to four one-hidden-layer fully-connected networks resulting from Adam-based quantization-aware training in Keras following [152, 153]. Layerwise inhibitory neurons are used to compensate for rescaling of synaptic weights trained with -1 and $+1$ values in Keras to values of 0 and 1 in MorphIC. Average-pooling the core activities into a global output sum layer leads to a 97.8% classification accuracy using the rate code. With rank order coding, an accuracy of 95.9% can be achieved. L1 (resp. L2) connectivity carries inferences from sub-classifiers (resp. combined sub-classifiers), illustrating that the level of the encoded features increases with the connectivity hierarchy.

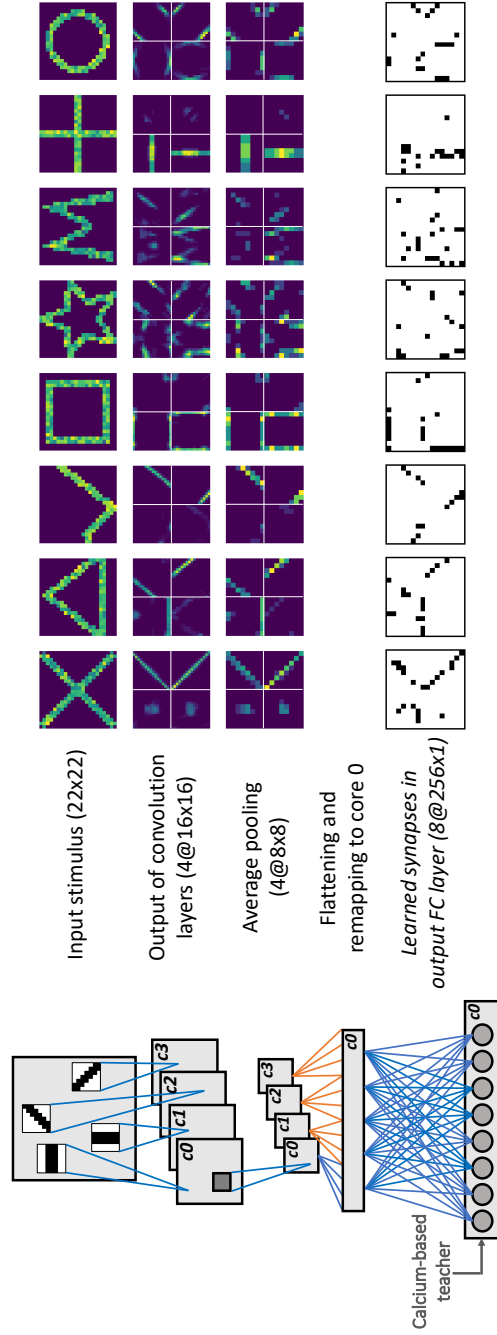


Fig. 2.27 MorphIC implementation of the 8-pattern CNN classification benchmark proposed in [165]. Left: the CNN architecture consists of four 7×7 line detection kernels at 0° , 45° , 90° and 135° , followed by average-pooling layers and a fully-connected (FC) 8-neuron output layer. Right: the input pattern activations, neuron activations of convolution and pooling layers and synaptic weights learned online with S-SDSP are illustrated. The test set consists of 100 different Poisson realizations of each input pattern, all 800 test samples are correctly classified.

per neuron that define the start and end indices of the destination neurons. Adding this architectural improvement to MorphIC would reduce the number of SOPs to be processed and also the time per inference, bringing E_{infer} down to $16.0\mu\text{J}$ per inference. Second, as shown in Fig. 2.26, inhibitory neurons have been added to the hidden and output layers to compensate for rescaling of the synaptic weights trained offline with -1 and $+1$ values to 0 and 1 values in MorphIC. However, it comes at the expense of a 50-% increase in SOP activity. In combination with the previous improvement, allowing the MorphIC binary weights to be interpreted as -1 and $+1$ values instead of 0 and 1 would allow further reducing E_{infer} to $10.7\mu\text{J}$ per inference. Finally, as clock gating has only been inserted automatically by the synthesis tool, simple architectural clock gating could be applied (i) to the parameter banks after initial chip configuration and (ii) to S-SDSP up/down registers (Fig. 2.17) during inference. This would lead to an idle power reduction by 55%, further optimizing E_{infer} down to a value of $8.2\mu\text{J}$ per inference. With a low leakage of only $45\mu\text{W}$ at 0.8V to ensure full retention of the neuronal and synaptic data, MorphIC is ideally-suited for always-on event-driven processing. A full comparison of area-accuracy and energy-accuracy tradeoffs for bottom-up and top-down neuromorphic designs that have been benchmarked on the full non-preprocessed MNIST dataset will be carried out in Section 4.3 of Chapter 4.

Finally, S-SDSP online learning is demonstrated in Fig. 2.27. We reproduced the benchmark that was proposed in [165] for an analog SDSP implementation, where eight patterns are classified by a spiking CNN. Each MorphIC core implements a fixed-weight convolutional layer with a line detection kernel followed by an average-pooling layer. The pooling layers from cores 1 to 3 are then mapped back to core 0 through L1 connectivity so as to form a single flattened layer. The flattened layer is connected with plastic weights to an 8-neuron fully-connected output layer in core 0. The resulting weights allow correctly discriminating all test samples in a test set consisting of 100 different Poisson realizations of each input pattern.

2.4 Discussion

Comparison with the state of the art of bottom-up neuromorphic experimentation platforms can be carried out along several axes, all of which uncovering design tradeoffs and guidelines. Section 2.4.1 focuses on the key figures of merit of neuron versatility (i.e. Izhikevich behavior repertoire), synaptic plasticity, density and energy per SOP. The aspects linked to scalability and learning performance require a discussion on their own, they are offloaded to Sections 2.4.2 and 2.4.3, respectively.

Table 2.5 Comparison of specifications and measured performances across state-of-the-art bottom-up neuromorphic chips.

Author Publication Chip name	Schemmel [30] ISCAS, 2010 HICANN	Benjamin [32] PIEEE, 2010 Neurogrid	Qiao [27] Front. NS, 2015 ROLLS	Moradi [29] TBICAS, 2017 DYNAPs	Park [26] BioCAS, 2014 IFAT	Mayr [28] TBCAS, 2016	Painkras [31] JSSC, 2013 Spinnaker	Seo [25] CICC, 2011	Alkopyan [33] TCAD, 2015 TrueNorth	Davies [34] IEEE Micro, 2018 Loihi	Frenkel TBCAS, 2019a ODIN	Frenkel TBCAS, 2019b Morphic
Implementation	Mixed-signal	Mixed-signal	Mixed-signal	Mixed-signal	Mixed-signal	Mixed-signal	Digital	Digital	Digital	Digital	Digital	Digital
Technology	0.18 μ m	0.18 μ m	0.18 μ m	0.18 μ m	90nm	28nm	0.13 μ m	45nm SOI	28nm	14nm FinFET	28nm FDSOI	65nm LP
# cores ^o	1	16	1	4	32	1	18	1	4096	128	1	4
Neuromorphic core area [mm ²]	49	168	51.4	7.5	0.31	0.36	3.75	0.8	0.095	0.4	0.086	0.715
# Izhikevich behaviors [†]	(20)	N/A	(20)	(20)	3	3	max. 1000 ^o	3	11 (3 near: 20)	(6)	20	3
# neurons per core	512	64k	256	256	2k	64	max. 1000 ^o	256	256	max. 1024	256	512
Synaptic weight storage	4-bit (SRAM)	Off-chip	Capacitor	12-bit (CAM)	Off-chip	4-bit (SRAM)	Off-chip	1-bit (SRAM)	1-bit (SRAM)	1- to 9-bit (SRAM)	(3+1)-bit (SRAM)	1-bit (SRAM)
Embedded online learning	STDP	No	SDSP	No	No	SDSP	Programmable	S-STDP	No	Programmable	SDSP	S-SDSP
# synapses per core	112k	–	128k	16k	–	8k	–	64k	64k	1M to 114k (1-9 bits)	64k	528k
Time constant	Accelerated	Biological	Biological	Biological	Biological	Bio. to accel.	–	Biological	Biological	N/A	Bio. to accel.	Bio. to accel.
Flexibility	Medium	Medium	Low	Medium	Medium	Low	High	Low	Medium	High	Low	Medium
Routing	Low	–	Low	Low	–	Low	–	Low	Low	High	Low	Low
Neuron core density [neur/mm ²] [†]	10.5	390	5	34	6.5k	178	max. 267 ^o	320	2.6k	max. 2.5k	3.0k	716
Neuron core density [neur/mm ²] [†] raw	–	–	–	–	–	–	max. 5.8k	826	2.6k	max. 1k	3.0k	3.9k
Synapse core density [syn/mm ²] [†]	2.3k	–	2.5k	2.1k	–	22.2k	–	80k	674k	2.5M to 282k	741k	738k
Synapse core density [syn/mm ²] [†] raw	–	–	–	–	–	–	–	207k	674k	1M to 113k	741k	4M
Supply voltage	1.8V	3.0V	1.8V	1.3V-1.8V	1.2V	0.75V, 1.0V	1.2V	0.53V-1.0V	0.7V-1.05V	0.5V-1.25V	0.55V-1.0V	0.8V-1.2V
Energy per SOP [‡]	N/A	(941pJ) [†]	>77fJ [†]	134fJ [†] /30pJ [†] (1.3V)	22pJ [†]	>850pJ [†]	>11.3nJ [†] /26.6nJ [†]	N/A	26pJ [†] (0.775V)	>23.6pJ [†] (0.75V)	8.4pJ [†] /12.7pJ [†]	30pJ [†] /51pJ [†] (0.8V)
Energy per SOP [‡] raw	–	–	–	–	–	–	>2.4nJ [†] /5.7nJ [†]	–	20pJ [†]	(66.1pJ [†])	–	12.9pJ [†] /22pJ [†]
Energy per SOP [‡] norm.	–	–	–	–	–	–	–	–	–	–	–	–

^o When chips are composed of several neuromorphic cores, we report the density data associated to a single core. Care should be taken that, depending on the core definition in the different chips, routing resources might be included (all single-core designs, IFAT, TrueNorth, Loihi and MorphIC) or excluded (Neurogrid, DYNAPs and Spinnaker). As opposed to the other reported designs, we consider the full Neurogrid system, which is composed of 16 Neurocore chips, each one considered as a core; routing resources are off-chip. For DYNAPs and Spinnaker, sharing routing overhead among cores would lead to 28-% and 37-% density penalties compared to the reported results, respectively. The HICANN chip can be considered as a core of the BrainScaleS wafer-scale system. Pad area is excluded from all reported designs.

[†] By its similarity with the Izhikevich neuron model, the AdExp neuron model is believed to reach the 20 Izhikevich behaviors [76], but it has not been demonstrated in HICANN, ROLLS and DYNAPs. The neuron model of TrueNorth can reach 11 behaviors per neuron and 20 by combining three neurons together [85]. The neuron model of Loihi is based on a LIF model to which threshold adaptation is added: the neuron should therefore reach 6 Izhikevich behaviors, although it has not been demonstrated.

^o Experiment 1 reported in Table III from [31] is considered as a best-case neuron density: 1000 simple LIF neuron models are implemented per core, each firing at a low frequency.

* Neuron (resp. synapse) core densities are computed by dividing the number of neurons (resp. synapses) per neuromorphic core by the neuromorphic core area. Regarding the synapse core density, Neurogrid, IFAT and Spinnaker use an off-chip memory to store synaptic data. As the synapse core density cannot be extracted when off-chip resources are involved, no synapse core density values are reported for these chips. Values normalized to a 28-nm CMOS technology node are provided for digital designs using the node factor, at the exception of the 14-nm FinFET node of Loihi for which Intel data from [120] has been used.

[‡] The synaptic operation energy measurements reported for the different chips do not follow a standardized measurement process. There are two main categories for energy measurements in neuromorphic chips. On the one hand, incremental values (denoted with [†]) describe the amount of energy paid per each additional SOP computation, they are measured by subtracting the leakage and idle power consumption of the chip, as in Eq. (2.2), although the exact power contributions taken into account in the SOP energy vary across chips. On the other hand, global values (denoted with ^{*}) are obtained by dividing the total chip power consumption by the SOP rate, as in Eq. (2.3). Values normalized to a 28-nm CMOS technology node are provided for digital designs using the node factor, including for the 14-nm FinFET node of Loihi in the absence of reliable data for power normalization in [120]. The conditions under which all of these measurements have been done can be found hereafter. For Neurogrid, a SOP energy of 941pJ is reported for a network of 16 Neurocore chips (1M neurons, 8B synapses, 413k spikes/s); it is a board-level measurement, no chip-level measurement is provided [32]. For ROLLS, the measured SOP energy of 77fJ is reported in [165]; it accounts for a point-to-point synaptic input event and includes the contribution of weight adaptation and digital-to-analog conversion, it represents a lower bound as it does not account for synaptic event broadcasting. For DYNAPs, the measured SOP energy of 134fJ at 1.3V is also reported in [165], while the global SOP energy of 30pJ can be estimated from [29] using the measured 800- μ W power consumption with all 1k neurons spiking at 100Hz with 25% connectivity (26.2MSOP/s), excluding the synaptic input currents. For IFAT, the SOP energy of 22pJ is extracted by measuring the chip power consumption when operated at the peak rate of 73M synaptic events/s [26]. In the chip of Mayr *et al.*, the SOP energy of 850pJ represents a lower bound extracted from the chip power consumption, estimated by considering the synaptic weights at half their dynamic at maximum operating frequency [28]. For Spinnaker, an incremental SOP energy of 11.3nJ is measured in [166], a global SOP energy of 26.6nJ at the maximum SOP rate of 16.56MSOP/s can be estimated by taking into account the leakage and idle power; both values represent a lower bound as the energy cost of neuron updates is not included. For TrueNorth, the measured SOP energy of 26pJ at 0.775V is reported in [167], it is extracted by measuring the chip power consumption when all neurons fire at 20Hz with 128 active synapses. For Loihi, a minimum SOP energy of 23.6pJ at 0.75V is extracted from pre-silicon SDF and SPICE simulations, in accordance with early post-silicon characterization [34]; it represents a lower bound as it includes only the contribution of the synaptic operation, without taking into account the cost of neuron update and learning engine update. For ODIN and MorphIC, the detailed measurement process is described in Sections 2.2.2 and 2.3.2, respectively.

2.4.1 Comparative analysis of bottom-up neuromorphic designs

A performance and specification summary of state-of-the-art bottom-up neuromorphic chips is provided in Table 2.5. Mixed-signal designs with core analog neuron and synapse computation and high-speed digital periphery are grouped on the left [26–30, 32], digital designs are grouped together with ODIN and MorphIC on the right [25, 31, 33, 34].

Among SNNs, the 28-nm IBM TrueNorth chip is the largest-scale single-chip design, with neuron and synapse densities well beyond those of all mixed-signal approaches proposed to date [33]. Compared to ODIN, its neurosynaptic cores have identical numbers of neurons and synapses and both chips are in 28-nm CMOS, which allows for a direct comparison. While TrueNorth does not embed plasticity into its binary synapses, we show with ODIN that it is possible to quadruple the number of bits per synapse and to add online learning while slightly reducing the overall area, thus improving neuron and synapse densities. We then demonstrate with MorphIC how leveraging stochastic online learning on binary weights leads to the highest densities reported to date, considering technology-normalized numbers. MorphIC also demonstrates an order-of-magnitude density improvement compared to the only previously-proposed binary SNN with demonstrated online learning from Seo *et al.* [25], a point that is further emphasized when considering process normalization.

The 14-nm Intel Loihi chip has recently been proposed in [34] and embeds a configurable spike-timing-based learning rule. Loihi is also an experimentation platform and has specific features, it is an interesting design offering a good balance between a diverse configurable feature set and a high density. While the neuron Calcium variable in ODIN corresponds to an SDSP eligibility trace with configurable time constant through Calcium leakage, Loihi offers several types of eligibility traces (e.g., multiple spike traces and reward traces) with extended configurability in the context of its programmable learning rule. The axonal and refractory delays, stochasticity and threshold adaptation for homeostasis offered by Loihi can be captured in ODIN by the stochastic transitions of SDSP (Fig. 2.10) and by the refractory, spike latency and threshold variability behaviors of the phenomenological Izhikevich neuron. Loihi also allows for dendritic computation with multi-compartment neurons and has a configurable synaptic fan-in/resolution tradeoff, while ODIN further extends the behavior repertoire at a higher density. It should be noted that, while the cores of MorphIC rely on the design proposed for ODIN, only the LIF update logic is available in MorphIC, thus reducing its feature set. This choice has been made to minimize the scheduler overhead, which takes a significant amount of logic in ODIN in order to accommodate for the Izhikevich bursting behaviors (Section 2.2.2). A better choice would have been to reuse the phenomenological Izhikevich neuron model proposed in Chapter 1 while removing bursting behaviors. For deployment in real-world applications such as sensor fusion, where input data statistics and encodings can vary strongly among sensors [168], it would therefore be interesting to investigate setups combining ODIN and MorphIC

chips. We demonstrate a sensor fusion setup in [169], where electromyography and spiking retina recordings of hand gestures are classified in ODIN and MorphIC, respectively.

Regarding the power consumption, as indicated in Table 2.5, the SOP energies reported in the state of the art are difficult to compare directly due to non-standardized measurement processes across the different chips. However, it allows extracting orders of magnitude. The flexibility-power tradeoff of the SpiNNaker software approach appears clearly: high programmability for both neuron and synapse models leads to a global energy per SOP as high as 26.6nJ. The next SpiNNaker generation might improve this tradeoff through advanced power reduction techniques and dedicated hardware accelerators (e.g., [143, 170, 171]). For ODIN, supply voltage scaling down to 0.55V in 28-nm FDSOI CMOS results in a minimum global energy per SOP of 12.7pJ at the maximum SOP rate. The global energy per SOP of MorphIC amounts to 51pJ at 0.8V, which corresponds to an energy of 22pJ/SOP considering normalization to 28-nm CMOS, which is competitive with state-of-the-art designs. MorphIC therefore achieves record densities without incurring a power penalty. Table 2.5 also shows that the hundred-fJ incremental energy of subthreshold analog designs makes them appear as particularly energy-efficient given that they exclude the contributions of leakage, idle power and network operation. However, when taking these elements into account, the high ratio between global and incremental energies per SOP of DYNAPs shows that these designs do not scale efficiently. Preliminary results from a 28-nm implementation of DYNAPs claim a promising energy per SOP of 2.8pJ [125], although it is not yet clear which SOP definition is used.

2.4.2 Hierarchical event routing

Several chips in Table 2.5 embed specific routing schemes to allow for large-scale integration of interconnected chips, it is the case for HICANN, Neurogrid, DYNAPs, SpiNNaker, TrueNorth, Loihi and MorphIC, while the HiAER hierarchical routing AER topology has been recently proposed for IFAT in [161]. The other chips compared in Table 2.5, i.e. ROLLS, ODIN and the chips from Mayr *et al.* and Seo *et al.*, consist of a single neurosynaptic core with standard input and output AER interfaces. These chips can achieve large-scale interconnection only through the use of external routing tables in order to define the inter-chip connectivity, and thus cannot scale efficiently. This limitation of ODIN is alleviated in MorphIC, whose key feature is that only 27 bits per neuron are required to entirely define the connectivity of a multi-chip network, while the routers at all the hierarchy levels are memory-less beyond simple event buffering (i.e. no dedicated access to a stored mapping table). It contrasts with Neurogrid [32] and HiAER [161], which achieve low-cost large-scale routing at the expense of requiring external mapping table storage, thus indirectly inducing high resource and power overheads. SpiNNaker [31, 160] embeds the largest-scale multicast connectivity infrastructure

proposed so far. It avoids external accesses but requires local mapping tables inside the routers to store all connectivity information for source-based 2-D triangular toroidal mesh-based routing. The same holds for DYNAPs [29], which also avoids external accesses but still requires local 2.5-kB SRAM storage in the R1 router of each 256-neuron core, beyond a CAM-based 640-bit tag storage per neuron that allows reaching a high fan-out of 4k at the expense of density. Moreover, in DYNAPs, the synaptic connections are defined within the connectivity infrastructure, as opposed to MorphIC which has independent online-learning synapses on the top of the connectivity infrastructure. Therefore, none of Neurogrid, HiAER, SpiNNaker or DYNAPs offer low-cost memory-less routers throughout the hierarchy. The routing infrastructure of TrueNorth [33] holds similar advantages to the one of MorphIC: the routers are also memory-less and require neither external nor internal storage. However, there are only two levels in the TrueNorth hierarchy: the 64k-synapse 256-neuron crossbar cores and the large-scale mesh-based routers. While an arbitrary number of neurons can map to the same axons², thus reaching arbitrary fan-ins with 256 shared synapses, the neuron fan-out of TrueNorth is limited to a maximum value of $256+256$. In MorphIC, the addition of an intermediate L1 star-based router brings a two-fold advantage over TrueNorth: it divides by two the number of L2 hops between any two cores and allows extending the neuron fan-out. Finally, the hierarchical routing infrastructure of Loihi [34] differs strongly from all other previously-proposed approaches and, as DYNAPs, does not rely on a crossbar operation at the lowest level of the hierarchy. Instead, it is highly configurable in order to adapt to the target application: 120-kB SRAM memories are used to store the entire synaptic fan-in state of each core, which allows trading off the number of synapses with flexibility in the connectivity patterns. The high-level routing infrastructure of Loihi relies on a unicast mesh-based NoC: in order to process neuron spike multicasting with up to 4096 output axons, one packet needs to be generated for each destination. Therefore, Loihi achieves multicasting at the expense of high router load overhead. While the three-level connectivity infrastructure of MorphIC is less flexible, multicasting is only handled at the L1 level and a clear hierarchy segmentation avoids overloading higher-level routers. It ensures that the higher the hierarchy, the sparser the events and the higher the level of the encoded features, which leads to high efficiency if the hierarchical nature of the event routing infrastructure matches the intrinsic hierarchy in the data representation of a given task. In the case of MorphIC, as shown in Section 2.3.2, the chosen hierarchical routing infrastructure is ideal to process tasks of the complexity of MNIST in a single chip. The cores implement four weak classifiers that carry out inference on independent sub-images, star-based L1 routing allows all cores to be at equidistance (as opposed to mesh-based routing) to combine weak classifications in a single core, while L2 routing is used to transmit sparse spikes encoding the inference on the full MNIST image.

² Axons are defined as inputs in the TrueNorth terminology, each axon connecting to 256 neurons through 256 synapses in a local crossbar array.

As the hierarchical event routing fabric of MorphIC strongly relies on all-to-all crossbar operation (Fig. 2.20), the proposed connectivity infrastructure is ideal to explore recurrent and fully-connected network configurations. It has recently been illustrated in [172], where MorphIC is used to process the fully-connected layers of a convolutional network classifying N-CARS images [173]. Although convolutional layers can also be implemented using crossbars, the absence of an efficient weight reuse mechanism implies copying the kernels in the dendritic tree of each output neuron. As the receptive field of each neuron in a convolutional layer usually consists of only a few inputs, there is a poor utilization of the synaptic fan-in resources while the time-multiplexed controller goes through a majority of dummy SOPs with zero weights (Fig. 2.18). The latter aspect can be mitigated by a controller update, similarly to the crossbar optimization with start and end addresses proposed in Section 2.3.2 for fully-connected layers.

Finally, it is worth noting that all the routers in the aforementioned approaches operate asynchronously, except in HiAER [161] and MorphIC. In MorphIC, the choice of clocked operation for the routers allows for a straightforward design at the expense of efficiency. Indeed, the timing critical path being located at the core level, using one global clock common to both the cores and the routers unnecessarily limits the bandwidth of the latter which, as they are memory-less, could operate at much higher speed. Asynchronous router design would alleviate this problem. In order to avoid the design time and complexity overhead of asynchronous digital design, another option would be to locally generate a high-speed clock directly in the L1 and L2 routers, for example with a local ring oscillator that is enabled only when packets await routing.

2.4.3 Bottom-up spike-based online learning performance

The development of silicon implementations for bio-inspired learning rules such as STDP and SDSP is an inherently *bottom-up* process: the first step lies in neuroscience experimentation to understand biological phenomena, the second step lies in the development of efficient analog or digital circuits that emulate neuroscience observations, the third step consists in finding a suitable application. Therefore, while bottom-up approaches lead to efficient silicon implementations of bio-inspired plasticity models and are ideal for the design of experimentation platforms, bridging the gap from local brain-inspired learning toward more complex real-life applications is difficult. To the best of our knowledge, only simplified benchmarks have been used so far to demonstrate silicon implementations of STDP, SDSP or their variants. A summary is provided in Table 2.6: no STDP- or SDSP-based learning rule has yet been successfully applied *in silico* to at least the full MNIST dataset without any pre-processing step. In all cases, STDP- or SDSP-based training is only demonstrated for a single-layer network or for the output layer of a network with frozen hidden layers (i.e. shallow learning). The S-SDSP learning rule that we propose for MorphIC allows reaching the highest density of

Table 2.6 Benchmark summary for silicon implementations of STDP- and SDSP-based learning rules.

Chip(s)	Implementation	Learning rule	Benchmark
BrainScaleS [30]	Mixed-signal	4-bit STDP	–
DYNAPs + ROLLS [165]	Mixed-signal	SDSP	8-pattern classif.
Mayr <i>et al.</i> [28]	Mixed-signal	4-bit SDSP	–
Seo <i>et al.</i> [25]	Digital	1-bit S-STDP	2-pattern recall
Chen <i>et al.</i> [56]	Digital	7-bit STDP	Denoising / Pre-processed MNIST
Loihi [34]	Digital	STDP-based	Pre-processed MNIST
ODIN	Digital	3-bit SDSP	16×16 deskewed MNIST
MorphIC	Digital	1-bit S-SDSP	8-pattern classif.

online-learning synapses and has successfully been demonstrated on the 8-pattern benchmark from [165]. However, while MorphIC embeds sufficient neuron and synapse resources to tackle more complex problems, it also follows a bottom-up design approach. Scaling S-SDSP to more complex tasks is thus not straightforward as it would require going beyond single-layer training. Moreover, while quantization-aware training maintains full-resolution weights to accommodate for high-resolution updates (Fig. 2.15), only a binary resolution is available in S-SDSP, which tends to reduce its ability to discriminate highly correlated patterns. This is the reason why the 8-pattern benchmark from [165] was preferred over MNIST for S-SDSP benchmarking. Further research is required to leverage brain-inspired local plasticity primitives with multi-layer networks for online learning on real-world tasks, as highlighted by the recent S-STDP study by Yousefzadeh *et al.* [174].

On the other hand, *top-down* approaches start from the applicative problems (e.g., image recognition), where the backpropagation of error algorithm [12] has already been successfully applied to specific datasets such as MNIST, CIFAR-10 or ImageNet. Such approaches then attempt to design variants of the backpropagation algorithm that are more in line with brain observation [55, 175, 176], such as moving data representation to spiking or dropping the requirement for symmetric weights [177, 178]. Silicon implementations would come as a last step. In order to overcome the challenges of bottom-up approaches, the development of new multi-layer spike-based learning rules following top-down approaches has gained growing interest in the recent years (e.g., [178–181]). Further research is yet required to realize efficient silicon implementations of such learning rules and to make them both compatible with an online-learning setup and able to leverage weight quantization. This will be the purpose of Part II of this thesis.

Part II

Top-Down Neuromorphic Design

*When we write programs
that “learn”, it turns out
that we do and they don’t.*

Alan Perlis

Algorithms – Minimizing the training cost of neural networks

Contents

3.1 Overview of algorithms releasing backprop constraints toward biological plausibility and hardware efficiency	87
3.1.1 Releasing the weight transport problem requires random feedback	89
3.1.2 Releasing update locking requires local error or gradient prediction	89
3.2 Direct random target projection (DRTP) allows releasing both weight transport and update locking	90
3.2.1 Weight updates based only on the error sign provide learning to multi-layer networks	91
3.2.2 For classification, a feedback pathway is no longer required as the error sign is known in advance	94
3.2.3 DRTP delivers useful modulatory signals for classification . .	95
3.2.4 DRTP learns to classify MNIST and CIFAR-10 images without feedback	98
3.3 Discussion	101

Abstract Both as a first step of a top-down approach to neuromorphic engineering and as an introduction to the fundamentals of machine learning for adaptive edge computing, we start by studying the backpropagation of error (BP) algorithm. Two key issues currently preclude biological plausibility and hardware efficiency: (i) symmetry is required between forward and backward weights, which is known as the weight transport problem, and (ii) updates are locked before both the forward and backward passes have been completed. We show that using only the error sign is sufficient to provide learning in the hidden layers. As in classification problems the error sign information is already contained in the targets (i.e. one-hot-encoded labels), using the latter as a proxy for the error brings three advantages: (i) it eliminates the requirement for an explicit feedback pathway (i.e. no weight transport), (ii) it allows for weight updates to be computed in each layer independently as soon as the layer outputs have been evaluated (i.e. no update locking), and (iii) it allows relying on low-cost local and feedforward computation by evaluating layerwise loss gradients as a label-dependent random vector selection (i.e. relaxed computational, structural and memory requirements). We therefore propose the direct random target projection (DRTP) algorithm and demonstrate on the MNIST and CIFAR-10 datasets that, despite the absence of an explicit error feedback, DRTP provides a tradeoff between accuracy and computational cost that is suitable for edge computing applications [JP3].

3.1 Overview of algorithms releasing backprop constraints toward biological plausibility and hardware efficiency

Artificial neural networks (ANNs) were proposed as a first step toward bio-inspired computation by emulating the way the brain processes information with densely-interconnected neurons and synapses as computational and memory elements, respectively [2, 159]. In order to train ANNs, it is necessary to identify how much each neuron contributed to the output error, a problem referred to as *credit assignment* [182]. The backpropagation of error (BP) algorithm [12] is a direct application of the chain rule to multi-layer ANNs: it computes the loss gradients with respect to the network parameters at each layer and updates them in the direction of steepest descent. By solving the credit assignment problem, BP enabled the development of deep networks for applications ranging from computer vision (e.g., [4–6]) to natural language processing (e.g., [7, 8]). However, two critical issues preclude BP from being biologically plausible.

First, BP requires symmetry between the forward and backward weights, which is known as the *weight transport problem* [51]. Beyond implying a perfect and instantaneous communication of parameters between the feedforward and feedback pathways, error backpropagation requires each layer to have full knowledge of all the weights in the downstream layers, making BP a non-local algorithm for both weight and error information. From a hardware efficiency point of view, weight symmetry also impacts memory access patterns. Therefore, there is an increasing interest in developing training algorithms that release the weight symmetry constraint, as it has been shown in [183] that weight symmetry is not mandatory to reach near-BP performance (Section 3.1.1).

The second issue of BP is its requirement for a full forward pass before parameters can be updated during the backward pass, a phenomenon referred to as *update locking* in [53, 54]. Beyond making BP biologically implausible, update locking has critical implications for BP implementation as it requires buffering all the layer inputs and activations during the forward and backward passes in order to compute the weight updates, leading to a high memory overhead. Specific techniques enabling local error handling or gradient approximation are thus required in order to tackle update locking (Section 3.1.2).

The key problems of weight transport and update locking highlight how aiming at breaking accuracy records on standard datasets has led to leave aside hardware efficiency considerations. While accuracy is the key driver for applications backed by significant GPU and CPU resources, the development of decentralized adaptive smart sensors calls for keeping the hardware requirements of learning algorithms to a minimum. Moreover, it has been shown that weight transport and update locking preclude biological plausibility [51, 55], resulting from the non-locality in both weight and gradient information. Therefore, there is currently an increasing interest in releasing these constraints in order to achieve higher hardware efficiency and to understand the mechanisms that could underlie biological synaptic plasticity.

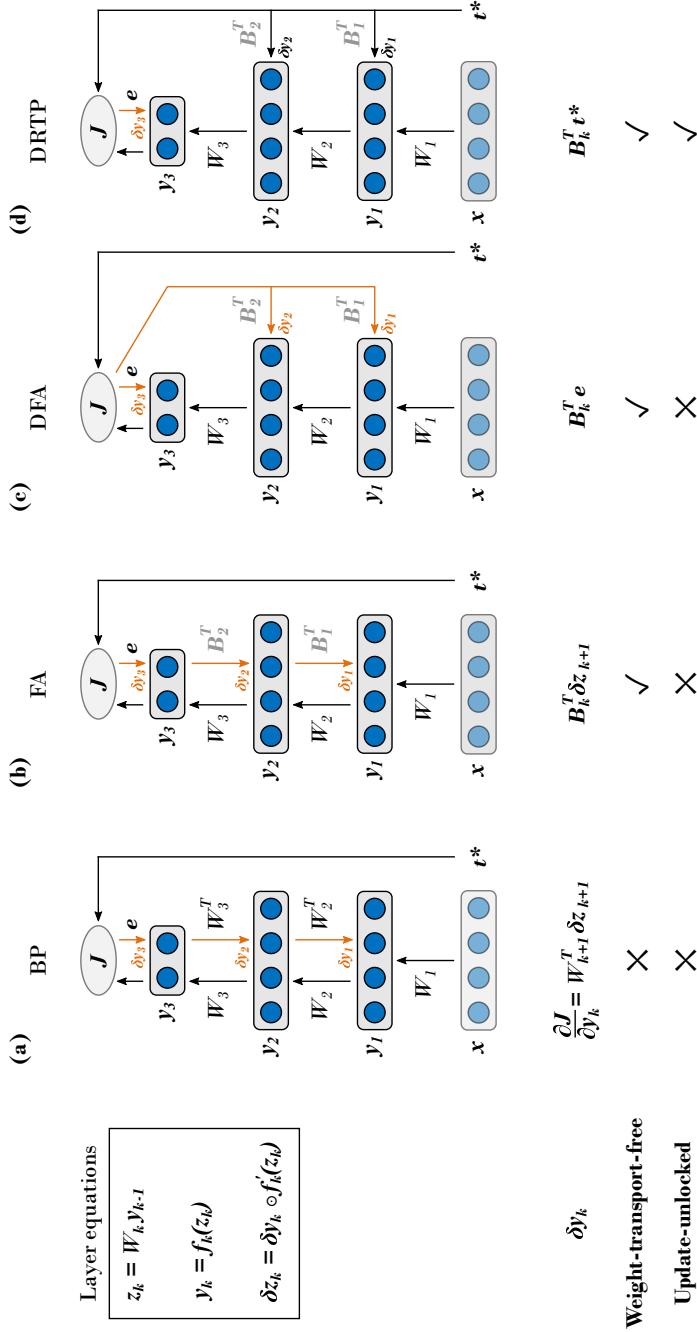


Fig. 3.1 The proposed direct random target projection algorithm builds on feedback-alignment-based algorithms to tackle the weight transport problem while further releasing update locking. Black arrows indicate the feedforward pathways and orange arrows the feedback pathways. In the k -th layer, the weighted sum of inputs y_{k-1} is denoted as z_k , the activation function as $f_k(\cdot)$ and its derivative as $f'_k(\cdot)$, with $k \in [1, K]$ and K the number of layers. Trainable forward weight matrices are denoted as W_k and fixed random connectivity matrices as B_k . The input vector is denoted as x , the target vector as t^* and the loss function as $J(\cdot)$. The estimated loss gradients for the outputs of the k -th hidden layer, denoted as δy_k , are provided for each training algorithm. The layer equations for z_k , y_k and δz_k , defined as the modulatory signals, are provided in the upper left corner, with $\odot$ denoting the elementwise multiplication operator. (a) Backpropagation of error (BP) algorithm [12]. (b) Feedback alignment (FA) algorithm [175]. (c) Direct feedback alignment (DFA) algorithm [176]. (d) Proposed direct random target projection (DRTP) algorithm. Adapted from [54, 176]. A PyTorch implementation of these algorithms is publicly available at <https://github.com/ChFrenkel/DirectRandomTargetProjection>.

3.1.1 Releasing the weight transport problem requires random feedback

The feedback alignment (FA) algorithm [175], also called random backpropagation in [55], demonstrates that using fixed random weights in the feedback pathway allows conveying useful error gradient information: the network learns to align the forward weights with the backward ones. Direct feedback alignment (DFA) [176] builds on these results and directly propagates the error between the network predictions and the targets (i.e. one-hot-encoded labels) to each hidden layer through fixed random connectivity matrices. DFA demonstrates a limited accuracy penalty compared to BP on the MNIST [150] and CIFAR-10 [17] datasets, while using the output error as a global modulation and keeping weight information local. Therefore, DFA bears important structural similarity with learning rules that are believed to take place in the brain, known as three-factor synaptic plasticity rules [114], which rely on local pre-synaptic and post-synaptic spike-based activity together with a global modulation. The SDSP learning rule studied in Chapter 1 can also be seen as a three-factor rule, with the Calcium concentration taking the role of a modulatory signal. This property of DFA was leveraged in [177] and [178], where it was formulated as a biologically-plausible spike-based learning rule. The BP, FA and DFA algorithms are summarized in Figs. 3.1(a)–(c), respectively.

3.1.2 Releasing update locking requires local error or gradient prediction

In order to avoid waiting for a full forward pass before the backward pass can be carried out to compute weight updates, early predictions are required for layerwise errors or gradients. The former is tackled in the *error locality approach*, which relies on layerwise loss functions to train layers independently [184–186]. In [184], local errors are generated using auxiliary fixed random classifiers and allow for near-BP performance on the MNIST and CIFAR-10 datasets. This strategy has been ported in [185] to a biologically-plausible spike-based three-factor synaptic plasticity rule. In [186], it is shown that a combination of two layerwise loss functions enables scaling to ImageNet [187] without compromising performance. However, the error locality approach still suffers from update locking at the layer scale as single-layer forward and backward passes are required. In [184], Mostafa *et al.* also investigate the weight transport problem, however weight symmetry is only partially relaxed: in order to maintain performance, it is necessary to keep at least the weight sign information during the backward pass in the auxiliary fixed random classifiers. On the other hand, the *synthetic gradients approach* proposed in [53, 54] relies on layerwise predictors of subsequent network computation. However, training local gradient predictors still requires backpropagating gradient information from deeper layers. Finally, the approach pursued in [188] defines layerwise target values, relying on auto-encoders at each layer. However, as these layerwise target values are still generated sequentially, this approach fails at solving the update locking problem [53].

3.2 Direct random target projection (DRTP) allows releasing both weight transport and update locking

In order to fully solve both the weight transport and the update locking problems, we propose the direct random target projection (DRTP) algorithm (Fig. 3.1(d)). Compared to DFA, the targets are used in place of the output error and projected onto the hidden layers. We demonstrate both theoretically and experimentally that, in the framework of classification problems, the error sign information contained in the targets is sufficient to maintain feedback alignment between the estimated loss gradients δz_k for the weighted sum of inputs in layer k , denoted as the *modulatory signals* in the subsequent text, and allows training multi-layer networks, leading to three key advantages. First, DRTP solves the weight transport problem by entirely removing the need for dedicated feedback pathways. Second, layers can be updated independently and without update locking as a full forward pass is not required, thus reducing memory requirements by releasing the need to buffer inputs and activations of each layer. Third, DRTP is a purely feedforward and low-cost algorithm whose updates rely on layerwise information that is immediately available upon computation of the layer outputs. Estimating the layerwise loss gradients δy_k only requires a label-dependent random vector selection, contrasting with the error locality and synthetic gradients approaches that require the addition of side networks for error or gradient prediction. DRTP even compares favorably to DFA, as the latter still requires a multiplication between the output error and a fixed random matrix.

Therefore, DRTP is ideal for implementation in edge computing devices, enabling adaptation to uncontrolled environments while meeting stringent power and resource constraints. Suitable applications thus range from distributed smart sensor networks for the Internet-of-Things (IoT) [13] to embedded systems and cognitive robotic agents [42]. DRTP can also be formulated as a three-factor learning rule for biologically-plausible learning and embedded neuromorphic computing, in which high-density synaptic plasticity can currently not be achieved without compromising learning performance, as previously discussed in Section 2.4.3. In order to outline the rationale behind DRTP, we first demonstrate in Section 3.2.1 that keeping only the error sign in DFA still provides learning in multi-layer networks, both for regression and classification tasks. Next, we show in Section 3.2.2 that, in the framework of classification problems, error-sign-based DFA does not require a dedicated feedback pathway. We then show in Section 3.2.3 how DRTP can be derived from error-sign-based DFA while further releasing computational requirements and improving performance. We demonstrate mathematically that the DRTP modulatory signals are within 90° of those prescribed by BP. Finally, we demonstrate on the MNIST and CIFAR-10 datasets that DRTP provides a tradeoff between accuracy and computational cost that is suitable for edge computing applications (Section 3.2.4).

3.2.1 Weight updates based only on the error sign provide learning to multi-layer networks

We demonstrate with two experiments, respectively on a regression task and a classification problem, that modulatory signals based only on the error sign are within 90° of those prescribed by BP, thus providing learning in multi-layer networks. To do so, we use an error-sign-based version of DFA, subsequently denoted as sDFA, in which the error vector is replaced by the error sign in the global feedback pathway.

3.2.1.1 Regression

This first experiment aims at demonstrating that the error sign provides useful modulatory signals to multi-layer networks by comparing training algorithms on a regression task. The objective is to approximate 10 nonlinear functions $T_j(x) = \cos(\bar{x} + \phi_j)$, where $\phi_j = -\pi/2 + j\pi/9$ for $j \in [0, 9]$ and $\bar{x}$ denotes the mean of x , a 256-dimensional vector whose entries are drawn from a normal distribution with a mean lying in $[-\pi, \pi]$ (Appendix A.1). A 256-100-100-10 fully-connected network is trained to approximate $T(\cdot)$ with five training algorithms: shallow learning (i.e. frozen random hidden layers and trained output layer), BP, FA, DFA and sDFA.

The mean squared error (MSE) loss on the training set is shown in Fig. 3.2(a). While shallow learning fails to learn a meaningful approximation of $T(\cdot)$, sDFA and DFA show the fastest initial convergence, which is a consequence from the separate direct feedback pathway precluding gradients from vanishing. While DFA demonstrates the highest performance on this task, sDFA comes earlier to stagnation as it does not account for the output error magnitude reduction as training progresses, thus depriving a reduction of the effective learning rate in the hidden layers as the output error decreases. sDFA would therefore benefit from the use of a learning rate scheduler. Similar conclusions hold for the loss on the test set (Fig. 3.2(b)). The angle between the modulatory signals prescribed by BP and by feedback-alignment-based algorithms is shown in Figs. 3.2(c) and 3.2(d) for the first and second hidden layers, respectively. While all feedback-alignment-based algorithms lie close to each other within 90° of the BP modulatory signals, FA has a clear advantage during the first 100 epochs on the 5k-example training set. sDFA performs on par with DFA in the first hidden layer, while it surprisingly provides a better alignment than DFA in the second hidden layer, though not fully leveraged due to the absence of modulation in the magnitude of the updates from the output error.

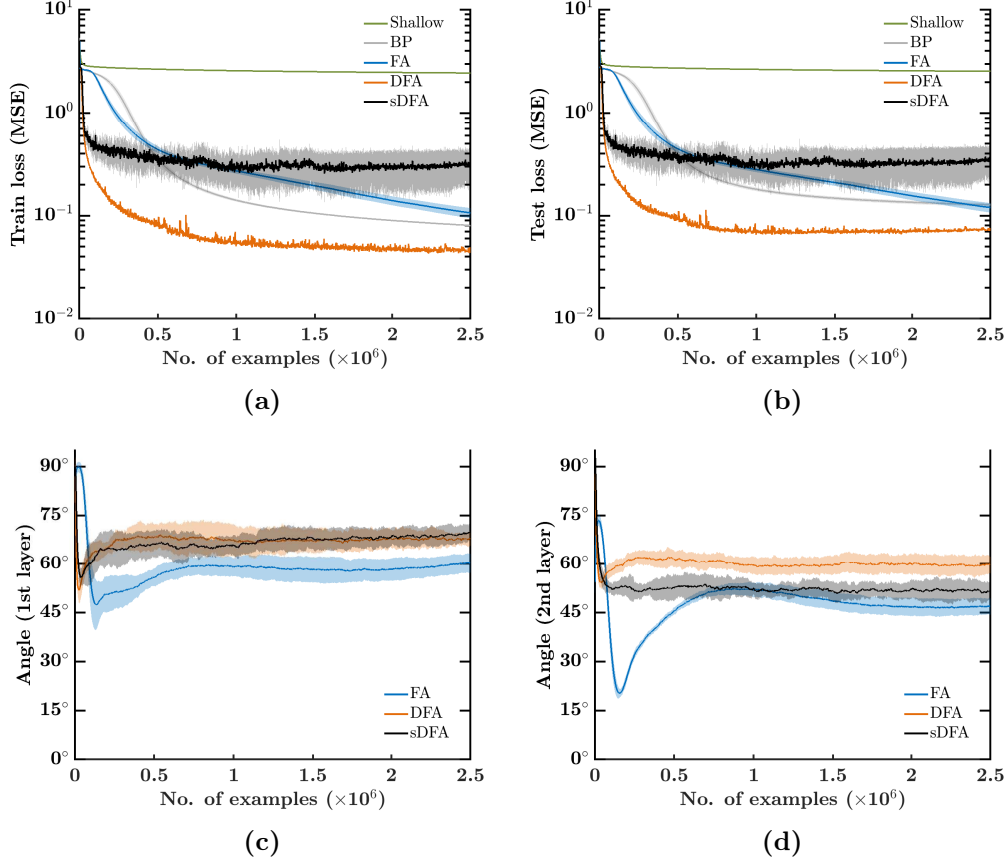


Fig. 3.2 Error-sign-based direct feedback alignment (sDFA) provides useful modulatory signals in regression tasks. A 256-100-100-10 network with tanh hidden and output units is trained to learn cosine functions with five training algorithms: shallow learning, BP, FA, DFA and sDFA. As for other feedback-alignment-based algorithms, sDFA updates are within 90° of the backpropagation updates. The train and test losses and the alignment angles are monitored every 1k samples, error bars are one standard deviation over 10 runs. Angles have been smoothed by an exponentially-weighted moving average filter with a momentum coefficient of 0.95. (a) Mean squared error loss on the 5k-example training set. (b) Mean squared error loss on the 1k-example test set. (c) Angle between the modulatory signals δz_k prescribed by BP and by feedback-alignment-based algorithms in the first hidden layer. (d) Angle between the modulatory signals δz_k prescribed by BP and by feedback-alignment-based algorithms in the second hidden layer.

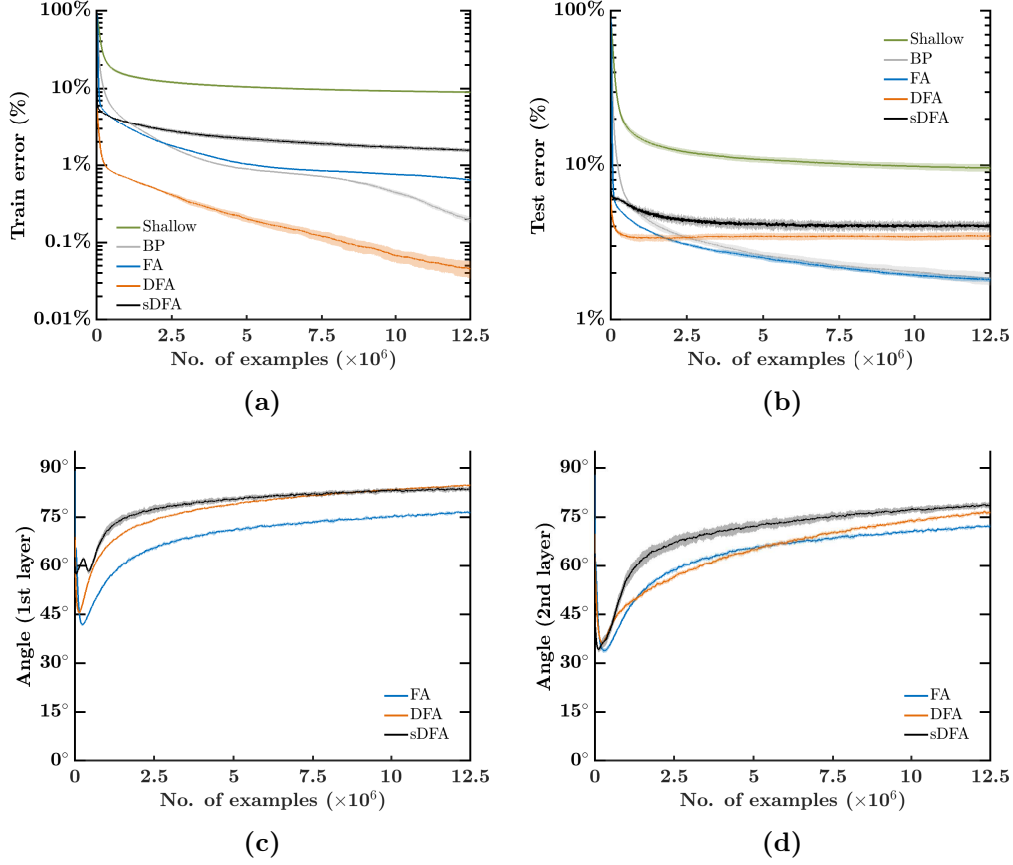


Fig. 3.3 Error-sign-based direct feedback alignment (sDFA) provides useful modulatory signals in classification tasks. A 256-500-500-10 network with tanh hidden units and sigmoid output units is trained to classify a synthetic dataset of 16×16 -pixel images into 10 classes with five training algorithms: shallow learning, BP, FA, DFA and sDFA. The update directions of the sDFA algorithm are within 90° of the backpropagation updates and are comparable to other feedback-alignment-based algorithms. The train and test losses and the alignment angles are monitored every 2.5k samples, error bars are one standard deviation over 10 runs. Angles have been smoothed by an exponentially-weighted moving average filter with a momentum coefficient of 0.95. **(a)** Error on the 25k-example training set, reaching on average 0.19% for BP, 0.64% for FA, 0.05% for DFA, 1.54% for sDFA and 8.95% for shallow learning after 500 epochs. **(b)** Error on the test set, reaching on average 1.85% for BP, 1.81% for FA, 3.48% for DFA, 4.07% for sDFA and 9.57% for shallow learning after 500 epochs. **(c)** Angle between the modulatory signals δz_k prescribed by BP and by feedback-alignment-based algorithms in the first hidden layer. **(d)** Angle between the modulatory signals δz_k prescribed by BP and by feedback-alignment-based algorithms in the second hidden layer.

3.2.1.2 Classification

With this second experiment, we demonstrate that, in addition to providing useful modulatory signals for regression problems, the error sign information allows training multi-layer networks to solve classification problems. The task consists in training a 256-500-500-10 network to solve a synthetic classification problem with 16×16 -pixel images and 10 classes; the data to classify is generated automatically with the Python `sklearn` library [189] (Appendix A.1). As for regression, the network is trained with shallow learning, BP, FA, DFA and sDFA.

Fig. 3.3(a) shows that, after 500 epochs with a 25k-example training set, DFA provides the fastest and most accurate training with a misclassification rate of 0.05%, followed by BP, FA and sDFA with 0.19%, 0.64% and 1.54%, respectively. Shallow learning lags almost an order of magnitude behind with 8.95%. However, Fig. 3.3(b) shows that DFA also has a higher overfitting and lies close to sDFA on the test set, with 3.48% and 4.07%, respectively. The lowest misclassification rates are of 1.85% for BP and 1.81% for FA, while shallow learning lags behind at 9.57%. The angle between the modulatory signals prescribed by BP and by feedback-alignment-based algorithms is shown in Figs. 3.3(c) and 3.3(d), for the first and second hidden layers, respectively. As for the regression task in Section 3.2.1.1, all feedback-alignment-based algorithms exhibit alignments close to each other. Here, alignments tend to level off after 50 epochs, with the lowest angle provided by FA, followed by DFA and sDFA. As sDFA is always within 90° of the BP modulatory signals, it is able to train multi-layer networks.

3.2.2 For classification, a feedback pathway is no longer required as the error sign is known in advance

In the framework of classification problems, training examples (x, c^*) consist of an input data sample to classify, denoted as x , and a label c^* denoting the class x belongs to, among C possible classes. The target vector, denoted as t^* , corresponds to the one-hot-encoded class label c^* . The output layer nonlinearity is conventionally chosen as a sigmoid or softmax function, yielding output values that are strictly bounded between 0 and 1. Denoting the output vector of a K -layer network as y_K , the error vector is defined as $e = y_K - t^*$. Under the aforementioned conditions, it results that an entry e_c of the C -dimensional error vector e is defined as

$$e_c = \begin{cases} y_{Kc} - 1 & \text{if } c = c^*, \\ y_{Kc} & \text{otherwise.} \end{cases}$$

As the entries of y_K are strictly bounded between 0 and 1, the error sign is given by

$$\text{sign}(e_c) = \begin{cases} -1 & \text{if } c = c^*, \\ 1 & \text{otherwise.} \end{cases}$$

Due to the nonlinearity in the output layer forcing the output values to remain bounded between 0 and 1, the error sign is class-dependent and known in advance as training examples (x, c^*) already provide the error sign information with the label c^* . A feedback pathway is thus no longer required as it has been shown that the error sign allows providing useful modulatory signals to train multi-layer networks (Section 3.2.1). Therefore, beyond being free from the *weight transport problem* as DFA, sDFA also releases *update locking* and the associated memory overhead in classification problems.

3.2.3 DRTP delivers useful modulatory signals for classification

This section aims at providing the grounds to show why the proposed direct random target projection (DRTP) algorithm delivers useful modulatory signals to multi-layer networks in the framework of classification problems. First, we show how DRTP can be viewed as a simplified version of sDFA in which the target vector t^* is used as a surrogate for the error sign. Next, we demonstrate mathematically that, in a multi-layer network composed of linear hidden layers and of a nonlinear output layer, the modulatory signals prescribed by DRTP and BP are always within 90° of each other, thus providing learning in multi-layer networks.

DRTP is a simplified version of error-sign-based DFA. As it has been shown in Section 3.2.2 that sDFA solves both the *weight transport* and the *update locking* problems in classification tasks, we propose the DRTP algorithm as a simplified version of sDFA while enhancing both performance and computational efficiency. In sDFA, the feedback signal randomly projected to the hidden layers is the sign of the error vector $e = y_K - t^*$, while in DRTP, this feedback signal is replaced by the target vector t^* . Being a one-hot encoding of c^* , t^* has a single positive entry corresponding to the correct class and zero entries elsewhere:

$$t_c^* = \frac{1 - \text{sign}(e_c)}{2} = \begin{cases} 1 & \text{if } c = c^*, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, t^* corresponds to a surrogate for the error sign vector used in sDFA, where shift, rescaling and sign inversion operations have been applied to $\text{sign}(e)$. As the connectivity matrices B_k in the DRTP gradients $\delta y_k = B_k^T t^*$ are fixed and random (Fig. 3.1(d)), they can be viewed as comprising the sign inversion and rescaling operations. Only the shift operation applied to $\text{sign}(e)$ makes a critical

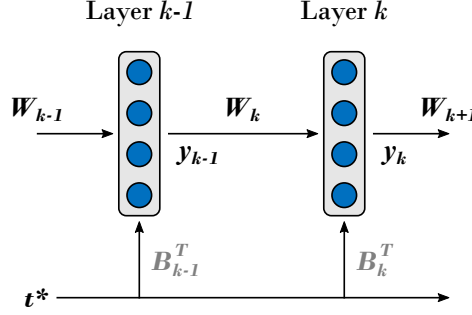


Fig. 3.4 Network of DRTP-updated linear hidden layers considered in the context of the mathematical proof of alignment between the DRTP and BP modulatory signals. The same conventions as in Fig. 3.1 are used.

difference between DRTP and sDFA, which is favorable to DRTP for two reasons. First, DRTP is computationally cheaper than sDFA. Indeed, projecting the target vector t^* to the hidden layers through fixed random connectivity matrices is equivalent to a label-dependent selection of a layerwise random vector. On the contrary, sDFA requires multiplying the error sign vector with the fixed random connectivity matrices for each training example, as all entries of the error sign vector are non-zero. Second, experiments on the MNIST and CIFAR-10 datasets show that DRTP systematically outperforms sDFA (Tables A.2 and A.3, Figs. A.1(a) and A.2(a) in Appendix A). Indeed, when the feedback information only relies on the error sign and no longer on its magnitude, the weight updates become less selective to the useful information: as all entries of the error sign vector have unit norm, the $C - 1$ entries corresponding to incorrect classes outweigh the single entry associated to the correct class and degrade the alignment (Figs. A.1(b) and A.2(b) in Appendix A).

The directions of the DRTP and BP modulatory signals are within 90° of each other. We provide a mathematical proof of alignment between the DRTP and BP modulatory signals. The structure of our proof is inspired from the FA proof of alignment in [175], which we expand in two ways. First, we extend this proof for the case of DRTP. Second, while [175] demonstrates the alignment with the BP modulatory signals for a network consisting of a single linear hidden layer, a linear output layer and a mean squared error loss, we demonstrate that alignment can be achieved for an arbitrary number of linear hidden layers, a nonlinear output layer with sigmoid activation and a binary cross-entropy loss for classification problems. Both proofs are restricted to the case of a single training example. Under these conditions, it is possible to guarantee that the DRTP modulatory signals are aligned with those of BP. This comes from the fact that the prescribed weight updates lead to a soft alignment between the product of forward weight matrices and the fixed random connectivity matrices. The mathematical details, including the lemma and theorem proofs, have been abstracted out to Appendix A.2.

In the case of the multi-layer neural network composed of linear hidden layers shown in Fig. 3.4, the output of the k -th hidden layer is given by

$$y_k = z_k = W_k y_{k-1} \quad \text{for } k \in [1, K-1],$$

where K is the number of layers and $y_0 = x$ is the input vector. The output layer is described by

$$\begin{aligned} z_K &= W_K y_{K-1} \\ y_K &= \sigma(z_K), \end{aligned}$$

where $\sigma(\cdot)$ is the sigmoid activation function. The loss function $J(\cdot)$ is the binary cross-entropy loss, computed over the C output classes:

$$J(y_K, t^*) = -\frac{1}{C} \sum_{c=1}^C \left(t_c^* \log(y_{Kc}) + (1 - t_c^*) \log(1 - y_{Kc}) \right).$$

Lemma. In the case of zero-initialized weights, i.e. $W_k^0 = 0$ for $k \in [1, K]$, and hence of zero-initialized hidden layer outputs, i.e. $y_k^0 = 0$ for $k \in [1, K-1]$ and $z_K^0 = 0$, considering a DRTP-based training performed recursively with a single element of the training set (x, c^*) and t^* denoting the one-hot encoding of c^* , at every discrete time step t , there are positive scalars $s_{y_k}^t$ and $s_{W_k}^t$ for $k \in [1, K-1]$ and a C -dimensional vector $s_{W_K}^t$ such that

$$\begin{aligned} y_k^t &= -s_{y_k}^t (B_k^T t^*) & \text{for } k \in [1, K-1] \\ W_1^t &= -s_{W_1}^t (B_1^T t^*) x^T \\ W_k^t &= s_{W_k}^t (B_k^T t^*) (B_{k-1}^T t^*)^T & \text{for } k \in [2, K-1] \\ W_K^t &= -s_{W_K}^t (B_{K-1}^T t^*)^T. \end{aligned}$$

Theorem. Under the same conditions as in the lemma and for the linear-hidden-layer network dynamics described above, the k -th layer modulatory signals prescribed by DRTP are always a positive scalar multiple of the Moore-Penrose pseudo-inverse of the product of forward matrices of layers $k+1$ to K , located in the feedback pathway between the output layer and the k -th hidden layer, multiplied by the error. That is, for $k \in [1, K-1]$ and $t > 0$,

$$\left(\prod_{i=k+1}^{K-1} W_i^t \right)^+ e = s_k^t B_k^T t^* \quad \text{with } s_k^t > 0.$$

Alignment. In the framework of classification problems, as the coefficients s_k^t are strictly positive scalars for $t > 0$, it results from the theorem that the dot product

between the BP and DRTP modulatory signals is strictly positive, i.e.

$$\begin{aligned}
 e^T \left(\prod_{i=k+1}^K W_i^T \right)^T (B_k^T t^*) &> 0 \\
 e^T \underbrace{\left(\prod_{i=k+1}^K W_i^T \right)^T \left(\prod_{i=K}^{k+1} W_i \right)^+}_I \frac{e}{s_k^t} &> 0 \\
 \frac{e^T e}{s_k^t} &> 0.
 \end{aligned}$$

The BP and DRTP modulatory signals are thus within 90° of each other. $\square$

3.2.4 DRTP learns to classify MNIST and CIFAR-10 images without feedback

In this section, we compare DRTP with BP and other feedback-alignment-based algorithms, namely FA and DFA, on the MNIST and CIFAR-10 datasets. Both datasets have 10 output classes, they respectively consist in classifying 28×28 grayscale images of handwritten digits for MNIST and 32×32 RGB images of vehicles and animals for CIFAR-10. The network topologies considered in our experiments are, on the one hand, fully-connected (FC) networks with one or two hidden layers, respectively denoted as FC1 and FC2, each hidden layer being constituted of either 500 or 1000 tanh units. On the other hand, convolutional (CONV) networks are used with either fixed random or trainable kernels. The CONV network for MNIST consists of one convolutional layer followed by a max-pooling layer and one fully-connected hidden layer, while for CIFAR-10 it consists of two convolutional layers, each followed by a max-pooling layer, and two fully-connected hidden layers. Detailed topologies are provided in Appendix A.1.

3.2.4.1 MNIST

The results on the MNIST dataset are summarized in Table 3.1. In FC networks, BP, FA and DFA perform similarly, the accuracy degradation of FA and DFA is marginal. While there is a higher accuracy degradation for DRTP, it compares favorably to shallow learning, which suffers from a high accuracy penalty. It shows that DRTP allows training hidden layers to learn MNIST digit classification without feedback. The CONV network topology leads to the lowest error, highlighting that extracting spatial redundancy information, even with random kernels, is a key to solve the MNIST task. The accuracy slightly degrades along the FA, DFA and DRTP algorithms, with a higher gap for shallow learning. When kernels are trained, BP provides the highest improvement compared to the error obtained with random

Table 3.1 Mean and standard deviation of the test error on the MNIST dataset over 10 trials. DO stands for dropout and indicates the dropout probability used in the fully-connected layers of both FC and CONV networks. The FC networks consist of one (FC1) or two (FC2) hidden layers comprising 500 or 1000 tanh units, with an output fully-connected layer of 10 units. The CONV network topology is as follows: a convolutional layer with $32 \times 5 \times 5$ kernels, a stride of 1 and a padding of 2, a max-pooling layer with 2×2 kernels and a stride of 2, a fully-connected layer of 1000 tanh units and an output fully-connected layer of 10 units.

Network		BP	FA	DFA	DRTP	Shallow
FC1-500	DO 0.0	1.65±0.06%	1.71±0.05%	1.76±0.05%	4.61±0.13%	8.25±0.09%
	DO 0.1	1.59±0.03%	1.63±0.05%	1.68±0.03%	4.92±0.13%	9.17±0.11%
	DO 0.25	1.76±0.05%	1.74±0.04%	1.86±0.03%	5.75±0.09%	10.15±0.11%
FC1-1000	DO 0.0	1.57±0.04%	1.62±0.05%	1.67±0.03%	4.10±0.07%	7.92±0.10%
	DO 0.1	1.48±0.03%	1.55±0.05%	1.58±0.05%	4.31±0.06%	9.29±0.12%
	DO 0.25	1.54±0.04%	1.56±0.02%	1.63±0.03%	4.94±0.06%	10.01±0.17%
FC2-500	DO 0.0	1.46±0.08%	1.72±0.04%	1.69±0.06%	4.58±0.09%	8.25±0.10%
	DO 0.1	1.46±0.04%	1.51±0.04%	1.57±0.06%	5.00±0.07%	9.33±0.09%
	DO 0.25	1.38±0.04%	1.69±0.02%	1.52±0.03%	5.94±0.06%	11.01±0.12%
FC2-1000	DO 0.0	1.50±0.09%	1.57±0.06%	1.65±0.07%	4.00±0.10%	7.85±0.09%
	DO 0.1	1.46±0.02%	1.46±0.03%	1.57±0.03%	4.25±0.06%	8.73±0.08%
	DO 0.25	1.38±0.03%	1.50±0.05%	1.45±0.03%	5.05±0.09%	9.84±0.05%
CONV (random)	DO 0.0	1.21±0.05%	1.30±0.06%	1.25±0.08%	1.82±0.11%	2.83±0.19%
	DO 0.1	1.25±0.03%	1.33±0.06%	1.30±0.06%	2.06±0.08%	4.74±0.30%
	DO 0.25	1.29±0.04%	1.32±0.06%	1.33±0.05%	2.60±0.14%	6.49±0.35%
CONV (trained)	DO 0.0	0.93±0.04%	1.22±0.06%	1.31±0.06%	1.48±0.15%	—
	DO 0.1	1.03±0.04%	1.27±0.06%	1.34±0.06%	1.50±0.17%	—
	DO 0.25	1.00±0.03%	1.29±0.04%	1.40±0.06%	1.81±0.20%	—

Table 3.2 Mean and standard deviation of the test error on the CIFAR-10 dataset over 10 trials. DO stands for dropout and indicates the dropout probability used in the fully-connected layers of both FC and CONV networks. DA stands for data augmentation, which consists in horizontal flipping of the training images. No dropout is used for DA. The FC networks consist of one (FC1) or two (FC2) hidden layers comprising 500 or 1000 tanh units, with an output fully-connected layer of 10 units. The CONV network topology is as follows: two convolutional layers with respectively 64 and 256 3×3 kernels, a stride and a padding of 1, both followed by a max-pooling layer with 2×2 kernels and a stride of 2, then two fully-connected layers of 1000 tanh units and an output fully-connected layer of 10 units.

Network		BP	FA	DFA	DRTP	Shallow
FC1-500	DO 0.0	48.45±0.38%	49.38±0.22%	49.62±0.29%	53.92±0.23%	58.83±0.27%
	DO 0.1	47.48±0.39%	48.94±0.22%	48.85±0.23%	53.77±0.17%	59.33±0.17%
	DO 0.25	47.80±0.21%	48.62±0.23%	48.65±0.29%	54.26±0.16%	60.44±0.14%
	DA	45.87±0.22%	47.11±0.34%	47.34±0.26%	52.73±0.31%	58.60±0.20%
FC1-1000	DO 0.0	47.52±0.30%	48.47±0.18%	48.44±0.34%	53.34±0.10%	57.91±0.17%
	DO 0.1	46.42±0.28%	47.72±0.19%	47.79±0.31%	53.15±0.15%	58.35±0.24%
	DO 0.25	46.21±0.16%	47.11±0.18%	47.11±0.25%	53.39±0.15%	59.20±0.18%
	DA	45.01±0.33%	46.15±0.36%	46.24±0.32%	51.87±0.32%	57.40±0.24%
FC2-500	DO 0.0	49.03±0.22%	50.66±0.24%	50.45±0.36%	53.41±0.35%	59.62±0.34%
	DO 0.1	48.32±0.16%	49.64±0.23%	49.58±0.30%	54.06±0.46%	60.34±0.24%
	DO 0.25	49.96±0.18%	50.80±0.16%	50.00±0.13%	54.57±0.33%	61.77±0.18%
	DA	46.62±0.10%	48.55±0.25%	48.75±0.28%	52.54±0.34%	58.99±0.20%
FC2-1000	DO 0.0	48.81±0.22%	49.87±0.18%	50.05±0.21%	52.68±0.25%	58.59±0.14%
	DO 0.1	46.58±0.24%	47.97±0.18%	48.68±0.34%	52.45±0.15%	59.12±0.13%
	DO 0.25	47.65±0.16%	48.82±0.15%	48.08±0.14%	53.29±0.31%	60.52±0.14%
	DA	46.05±0.14%	47.41±0.27%	47.90±0.19%	51.27±0.21%	57.90±0.23%
CONV (random)	DO 0.0	29.83±0.25%	30.27±0.45%	29.98±0.30%	32.65±0.38%	44.89±0.67%
	DO 0.1	29.49±0.36%	29.58±0.33%	29.44±0.31%	32.57±0.34%	48.38±0.33%
	DO 0.25	30.39±0.32%	30.55±0.28%	30.31±0.35%	33.90±0.53%	52.27±0.34%
	DA	27.87±0.25%	28.52±0.40%	28.46±0.43%	31.04±0.45%	44.23±0.42%
CONV (trained)	DO 0.0	25.31±0.25%	29.92±0.26%	31.38±0.38%	35.82±0.59%	—
	DO 0.1	27.12±0.23%	28.98±0.36%	30.56±0.41%	35.17±0.91%	—
	DO 0.25	25.61±0.23%	28.95±0.17%	31.23±0.38%	35.51±0.61%	—
	DA	25.27±0.26%	28.16±0.45%	29.49±0.49%	34.39±0.64%	—

kernels, followed by DRTP, while no significant change can be observed for FA and DFA. This is likely due to the fact that there is not enough parameter redundancy in convolutional layers to allow for an efficient training with feedback-alignment-based algorithms, a *bottleneck effect* that we will further discuss in Section 3.3. Indeed, the angle between the BP loss gradients and the feedback-alignment-based ones is roughly 90° , leading to random updates (Fig. A.3 in Appendix A). Regarding dropout, a positive impact is shown on BP, FA and DFA: a moderate dropout probability is beneficial for FC1 networks, while increasing it to 0.25 can be used for FC2 networks. Dropout has no positive impact for CONV networks, while it degrades the accuracy obtained with DRTP and shallow learning in all cases.

3.2.4.2 CIFAR-10

The results on the CIFAR-10 dataset are summarized in Table 3.2, highlighting conclusions similar to those already drawn for the MNIST dataset. Compared to BP, accuracy degrades along the FA, DFA and DRTP algorithms. The gap is higher for DRTP, yet it again compares favorably to shallow learning, demonstrating that DRTP also allows training hidden layers to learn CIFAR-10 image classification without feedback. For CONV networks, if kernels are trained, only BP is able to provide a significant advantage. Due to the bottleneck effect, FA only provides a slight improvement, while DFA and DRTP are negatively impacted. Regarding dropout, a moderate probability of 0.1 works fairly well for BP, FA, DFA and DRTP, while a higher probability of 0.25 rarely provides any advantage. Dropout always leads to an accuracy reduction for shallow learning. Finally, data augmentation (DA) improves the accuracy of all algorithms and is more effective than dropout.

3.3 Discussion

The proposed DRTP algorithm successfully addresses both the weight transport and the update locking problems, which has only been partially demonstrated in previously-proposed approaches. Indeed, the FA and DFA algorithms only address the weight transport problem [175, 176]. The error locality approach still suffers from the weight transport problem in the local classifiers [184–186], while the synthetic gradients approach requires backpropagating gradient information from deeper layers in order to train the layerwise gradient predictors [53, 54]. Both the error locality and the synthetic gradients approaches also incur computational overhead by requiring the addition of side local networks for error or gradient prediction. On the contrary, DRTP is a strikingly simple rule that alleviates the two key BP issues by enabling each layer to be updated with local information as the forward evaluation proceeds. In order to estimate the layerwise loss gradients δy_k for each layer, the only operation required by DRTP is a label-dependent random vector selection (Fig. 3.1(d)). Despite the absence of dedicated feedback

pathways, we demonstrated on the MNIST and CIFAR-10 datasets that DRTP allows training hidden layers at low computational and memory costs, thus highlighting its suitability for deployment in adaptive smart sensors at the edge and for embedded systems in general.

By solving the weight transport and update locking problems, DRTP also releases key biological implausibility issues. Neurons in the brain separate forward and backward information in somatic and dendritic compartments, a property that is highlighted in the formulation of three-factor synaptic plasticity rules [114]: pre-synaptic and post-synaptic activities are modulated by a third factor corresponding to a local dendritic voltage. Lillicrap *et al.* build on the idea that a separate dendritic compartment integrates higher-order feedback and generates local teaching signals, where the errors could be viewed as a mismatch between expected and actual perceptions or actions [175]. This aspect is further emphasized in the subsequent work of Guerguiev *et al.* when framing DFA as a spike-based three-factor learning rule [177]. In the case of DRTP, compared to DFA, the error signal is replaced by the targets, which could correspond to a modulation that bypasses the actual perceptions or actions, relying only on predictions. Such observations can have critical implications for a top-down approach to neuromorphic engineering, in an attempt to port biological computational principles to hardware toward higher energy efficiency [190, 191]. Indeed, bottom-up neuromorphic processor designs currently struggle to achieve both implementation efficiency and learning performance for real-world scenarios when embedding local bio-inspired learning rules such as STDP [98] and SDSP [106], as previously discussed in Section 2.4.3. A spike-based three-factor formulation of DRTP would release the update locking problem in the spike-based three-factor formulations of DFA [177, 178], which currently imply memory and control overhead in their hardware implementations [50, 192]. Porting DRTP to neuromorphic hardware is thus the subject of Chapter 4.

Finally, while DRTP relaxes structural, memory and computational requirements toward decentralized hardware deployment, the accuracy degradation over DFA comes from the fact that only the error sign is taken into account, not its magnitude. This aspect could be mitigated by keeping track of the error magnitude over the last samples in order to modulate the layerwise learning rates, at the expense of releasing the purely feedforward nature of DRTP. Otherwise, as outlined in Section 3.2.1.1, a fixed learning rate scheduler can be used. As for all other feedback-alignment-based algorithms, DRTP only slightly improves or even degrades the accuracy when applied to convolutional layers. Convolutional layers do not provide the parameter redundancy that can be found in fully-connected layers, a *bottleneck effect* that was first highlighted for FA in [175] and has recently been studied for DFA in [193]. If fixed random convolutional layers do not meet the performance requirements of the target application, a combination of DRTP for fully-connected layers together with error locality or synthetic gradients approaches for convolutional layers can be considered. This granularity in the selection of learning mechanisms, trading off accuracy and hardware efficiency, comes in accordance with the wide spectrum of plasticity mechanisms that are believed to operate in the brain (Section 1.2).

Integration – Neuromorphic accelerators

Contents

4.1 Overview of neuromorphic accelerators	105
4.2 SPOON – Fusing spiking and non-spiking domains for low-power adaptive event-based data processing	108
4.2.1 Architecture and implementation	108
4.2.2 Specifications and benchmarking results	115
4.3 Discussion	117

Abstract As a second step of the top-down approach, we investigate the application-driven design of neuromorphic accelerators. We first survey existing top-down designs, highlighting two trends: spiking neural networks with event-driven processing at the expense of accuracy and binary neural networks with high accuracy at the expense of conventional frame-based processing. In order to overcome these limitations, we propose SPOON, an event-driven convolutional neural network (eCNN) in 28-nm FDSOI CMOS leveraging two key aspects [CP4]. First, by embedding online learning with only 16.8-% power and 11.8-% area overheads, we show how the low computational and memory requirements of the DRTP algorithm derived in Chapter 3 allow for efficient silicon implementations. Second, we demonstrate how combining event-driven with frame-based computation (i) allows leveraging the sparsity of event-driven sensors and (ii) ensures maximum data reuse. With an energy per classification of 313nJ at 0.6V and a 0.32-mm² area for accuracies of 95.3% (on-chip training) and 97.5% (off-chip training) on MNIST, we demonstrate that SPOON reaches the efficiency of conventional machine learning accelerators while embedding on-chip learning and being compatible with event-based sensors, a point that we further emphasize with N-MNIST benchmarking.

4.1 Overview of neuromorphic accelerators

As opposed to bottom-up neuromorphic designs, which are suitable for neuroscience-oriented applications but whose efficiency in tackling real-world problems has yet to be demonstrated, top-down approaches start from the applicative problem and their related algorithms (Fig. 1). They can thus be distinguished from the bottom-up designs studied in Chapter 2 in that they can hardly be applied to another purpose than the one they were designed for (e.g., speech instead of image recognition). As the scopes, implementations and applications of top-down designs vary widely, comparing them directly is difficult, except when standard benchmarks are used. However, in order to extract the main trends, a summary of top-down neuromorphic designs is provided in Table 4.1.

The three chips from Knag *et al.* [47], Kim *et al.* [48] and Buhler *et al.* [49] follow a similar approach for image sparse coding based on an SNN implemented as a locally competitive algorithm (LCA). The LCA is implemented as a systolic ring of SNN cores, each of which is fully-connected to input pixels with feedforward excitatory connections, while lateral connections between neurons are inhibitory to favor sparsity in image representation. The 65-nm digital chip from Knag *et al.* furthermore implements SAILnet, a bio-inspired algorithm with local spike-based plasticity for adaptation of the neuron receptive fields [194]. Its main purpose is thus image feature extraction applied to denoising, however it lacks an inference module for image recognition and classification. This point is addressed by the chips from Kim *et al.* and Buhler *et al.* The former is a 65-nm digital design that embeds a classifier trained with SGD. The latter is a 40-nm mixed-signal design embedding analog LIF neurons, it is also claimed to embed online learning, but without specifying the associated algorithm. Both chips are benchmarked on MNIST, although with limited accuracies ranging from 84% to 90%.

Another approach is proposed by Park *et al.* [50], whose claim is to leverage the advantages of both ANNs (i.e. single-shot frame-based processing) and SNNs (i.e. sparse binary activations). The proposed architecture is thus equivalent to a binary neural network (BNN). It embeds the bio-inspired spike-based version of the DFA algorithm proposed by Guerguiev *et al.* in [177]. Although DFA suffers from update locking (Chapter 3), which implies a pipelined weight update scheme, Park *et al.* demonstrate a low-power design achieving an accuracy of 97.8% on MNIST with on-chip online learning.

Therefore, top-down neuromorphic designs seem to split among two categories: SNNs with event-driven processing at the expense of accuracy [47–49] or BNNs with high accuracy at the expense of conventional frame-based processing [50]. In order to solve this issue, we propose the 28-nm SPOON event-driven CNN (eCNN), combining both event-driven and frame-based processing to leverage sparsity from spiking neuromorphic retinas [195–197], while ensuring efficiency and accuracy in training and inference. It embeds the DRTP algorithm proposed in Chapter 3

Table 4.1 Comparison of top-down neuromorphic chips. The three designs on the right combine bottom-up and top-down approaches.

Author Publication Chip name	Knag [47] JSSC, 2015	Kim [48] VLSI-C, 2015	Buhler [49] VLSI-C, 2017	Park [161] ISSCC, 2019	Frenkel ISCAS, 2020 SPOON	Chen [56] JSSC, 2019	Pei [57] Nature, 2019 Tianjie	Neckar [58] PIEEE, 2019 Braindrop
Implementation	Digital	Digital	Mixed-signal	Digital	Digital	Digital	Digital	Mixed-signal
Technology	65nm	65nm	40nm	65nm	28nm FDSOI	10nm FinFET	28nm HPL	28nm FDSOI
Architecture	Spiking LCA	Spiking LCA	Spiking LCA	BNN	eCNN	SNN/BNN	SNN/ANN	SNN
Resources	256 neur 128k syn	256 neur 89k syn	512 neur 32k syn	FC200-FC200 -FC10	C5x5@10 -FC128-FC10	4k neur 1M syn	40k neur 10M syn	4k neur 64k syn
Embedded online learning	SAILnet	SGD (last layer)	<i>Yes</i> (<i>unspecified</i>)	DFA	DRTP	STDP	No	No
Demonstrated application	Image sparse coding	Image sparse coding & recog.	Image sparse coding & recog.	Image recog.	Image recog.	Image sparse coding & recog.	Real-time image, sound recognition & control	NEF-based networks
Benchmark(s) [‡]	Denoising	MNIST (84%-90%)	MNIST (88%)	MNIST (97.8%)	MNIST (95.3% , 97.5%), N-MNIST (93.0% , 93.8%)	Denoising, MNIST (98.6%)	Autonomous bike driving*	Function fitting, integrator
Energy metric	48pJ/pix	5.7pJ/pix	48.9pJ/pix	302pJ/pix	1.7nJ per pixel event	3.8pJ/SOP	0.78pJ/OP, 1.54pJ/SOP	0.38pJ/SOP

[‡] Accuracy results in bold font are obtained with on-chip online learning.^{*} Pei *et al.* also use N-MNIST and MNIST to quantify the efficiency and throughput improvement over a GPU and the improvement brought by hybrid SNN-ANN processing over SNN-only processing, respectively. However, the reported results are used only for relative comparisons, the provided data is not sufficient to be included in this table and in Section 4.3.

for adaptive edge computing. We demonstrate this claim on MNIST and on the neuromorphic MNIST (N-MNIST) dataset [198], which is a spiking version of MNIST generated by an ATIS silicon retina [196] mounted on a pan-tilt unit and moved in three saccades. SPOON reaches accuracies of 95.3% (on-chip training) and 97.5% (off-chip training) on MNIST, and of 93.0% (on-chip training) and 93.8% (off-chip training) on N-MNIST. The architecture, implementation and benchmarking details of SPOON are presented in Section 4.2. Finally, as SPOON and the chips from Kim *et al.*, Buhler *et al.*, Park *et al.* and Chen *et al.* have all been benchmarked on MNIST, it is possible to compare their energy-area-accuracy tradeoff with conventional neural network accelerators. This is the purpose of Section 4.3, where we demonstrate that, to the best of our knowledge, only SPOON reaches the efficiency of conventional machine learning accelerators while enabling online learning with event-based sensors.

Finally, three recently-published chips highlight that embedding bottom-up insight into a top-down approach is beneficial to neuromorphic computing (Table 4.1): the chip from Chen *et al.* [56], Tianjic [57] and Braindrop [58]. The first one is another attempt to bridge the gap between the BNN and SNN trends with a low-power STDP-based SNN in 10-nm FinFET that can also be programmed as a BNN. However, these two modes are still segmented at the application level: SNN operation with STDP is chosen for image denoising and BNN operation with offline-trained weights is chosen for image recognition. Indeed, Chen *et al.* show that an offline-trained BNN achieves 98.6% on MNIST, while a single-layer SNN with STDP training only achieves 89% on a pre-processed Gabor-filtered version of MNIST. Event-driven computation can thus not be leveraged in this device if high accuracy is required. The second one is Tianjic, a 28-nm digital design allowing for hybrid ANN-SNN setups and embedding as high as 40k neurons and 10M synapses per chip. This scale allows multi-chip Tianjic setups to be benchmarked on an autonomous bike driving task, demonstrating how both the ANN and SNN paradigms can be combined for real-time image recognition, sound recognition and vehicle control. The third one is Braindrop, a 28-nm mixed-signal design that relies, together with its software frontend, on an efficient set of mismatch- and temperature-invariant abstractions to provide one-to-one correspondence with the neural engineering framework (NEF) [59]. As the NEF approaches modeling of complex neural ensembles as dynamical systems of nonlinear differential equations, Braindrop allows mapping dynamical systems onto neuromorphic hardware made of somas and synaptic filters in an encode-transform-decode architecture, directly inspired by the previous-generation bottom-up Neurogrid design [32]. It is benchmarked on nonlinear 1D and 2D function fitting tasks and on integrator modeling, while a large scope of NEF applications has already been studied in the literature (e.g., see [199] for a recent review). These three chips demonstrate a high energy efficiency, i.e. 3.8pJ/SOP for the chip of Chen *et al.*, 0.78pJ/OP (ANN setup) or 1.54pJ/SOP (SNN setup) for Tianjic and 0.38pJ/SOP for Braindrop. However, Braindrop and Tianjic do not embed online learning and require an offline setup for network training and programming.

4.2 SPOON – Fusing spiking and non-spiking domains for low-power adaptive event-based data processing

As outlined in Section 4.1, in order to find a suitable tradeoff between the inherent sparsity of event-driven SNN-based processing and the accuracy of conventional frame-based BNNs, we propose SPOON (standing for spiking online-learning convolutional neuromorphic processor), an event-driven CNN (eCNN) for adaptive edge computing. Event-driven convolutions leverage sparsity from spiking neuromorphic retinas [200], an idea now also explored for conventional machine learning accelerators [201], while a combination with frame-based processing ensures maximum data reuse and parallelism in fully-connected layers. SPOON embeds online learning at low power and area overheads with the biologically-plausible DRTP algorithm that we proposed in Chapter 3, which releases the key issues of the successful backpropagation of error (BP) algorithm [12] precluding hardware efficiency and biological plausibility. The architecture and implementation of SPOON are described in Section 4.2.1, specifications and benchmarking results on MNIST and N-MNIST are given in Section 4.2.2.

4.2.1 Architecture and implementation

An overview of the eCNN topology with generic network size parameters is shown in Fig. 4.1 and highlights how event-driven and frame-based processing can be efficiently combined. Each event from the spiking neuromorphic retina passes through the kernel weights and is added to the feature map elements they correspond to, depending on the input pixel location. Upon the reception of an external trigger signaling the end of input event acquisition, maxpooling is processed in a frame-based fashion, after which all the inputs of the first hidden layer are available and can be processed in a frame-based way. The evaluation of each hidden neuron constitutes an event to update all neurons in the output layer.

The advantages of this combination between event-driven and frame-based processing for both the convolutional and fully-connected parts are highlighted in the block diagram of SPOON in Fig. 4.2. First, as neuromorphic vision sensors send spikes encoding temporal contrast [202], pixels with the highest luminosity change spike first with **ON** (positive delta) or **OFF** (negative delta) events, thus conveying useful data for edge detection. In order to efficiently extract this information, we use time-to-first-spike encoding in the convolutional layers, while event-driven processing allows leveraging the sparsity inherent to this timing code (Section 1.1.1). In order to match the time constant of SPOON with a given application, time ticks can be retrieved either from an external reference pin `TICK_EXT` or generated internally by a configurable synchronous on-chip tick generator. These computations are handled in the CONV core (Section 4.2.1.1). Second, the combination between frame-based and event-driven processing in the fully-connected layers allows for a

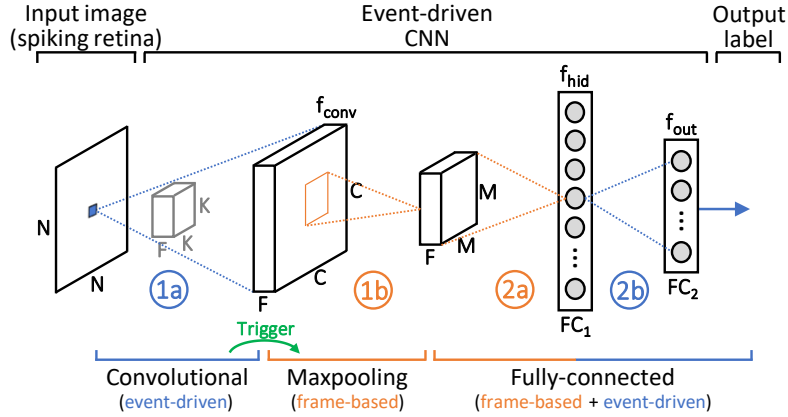


Fig. 4.1 Event-driven CNN (eCNN) network overview. Event-driven operation is shown in blue, it covers convolution of the events from the $N \times N$ -pixel neuromorphic retina by F $K \times K$ kernels and processing of the output fully-connected layer with FC_2 units. Frame-based operation is shown in orange, it covers maxpooling and processing of the first fully-connected layer with FC_1 units. Accumulation in the convolutional layer is stopped by a time-reference-dependent trigger shown in green, which starts the frame-based maxpooling operation. No padding is used, C is thus equal to $N - K + 1$ for a convolution of stride 1, while M is equal to C divided by the maxpooling stride. The activation functions are denoted by f_{conv} , f_{hid} and f_{out} .

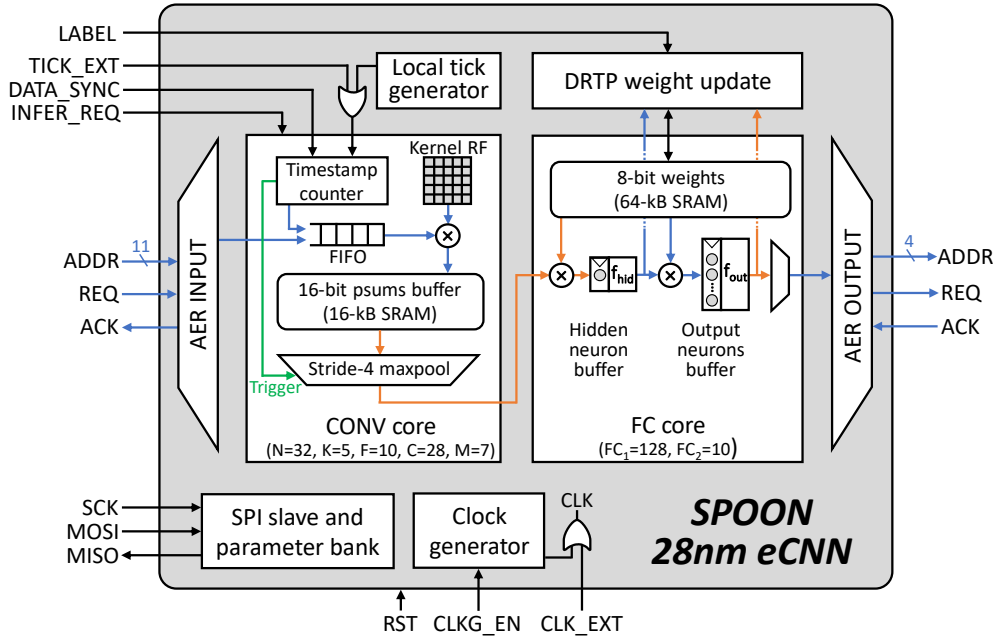


Fig. 4.2 Block diagram of the SPOON event-driven CNN (eCNN) processor, colors for event-driven and frame-based processing and network size parameters are specified according to the definitions from Fig. 4.1. f_{conv} (hidden in the overflow protection mechanism of the 16-kB SRAM psums update) and f_{hid} are hardtanh functions, f_{out} is a hardsigmoid function.

strong reduction in the buffering requirements, which is handled in the FC core (Section 4.2.1.2). Indeed, only the partial sums (psums) of the currently-processed hidden neuron and of the output neurons need to be buffered, which maximizes data reuse and favors an efficient combination with DRTP updates (Section 4.2.1.3). Four-phase-handshake AER buses are used for event-driven handling of input sensor spikes and of output inferences. All weights and parameters can be programmed and readback with an SPI bus.

4.2.1.1 Convolutional (CONV) core

The CONV core handles 32×32 -pixel images and consists of a convolutional layer with 10 5×5 8-bit programmable kernels followed by a stride-4 maxpooling layer, corresponding to parameters $N = 32$, $F = 10$, $K = 5$ and resulting in $C = 28$ and $M = 7$ in Fig. 4.1. Kernels are initialized to a hardcoded random content upon reset. The circuit architecture is shown in Fig. 4.3(a). Input AER events from the sensor are encoded with an 11-bit address, which covers the pixel $\{x, y\}$ coordinates and an ON/OFF polarity bit. Based on the TICK time reference and the DATA_SYNC pin that signals the start of an input sample, input events are concatenated with an 8-bit timestamp before being pushed into a 32-stage FIFO.

Event-driven convolutions follow the timing diagram of Fig. 4.4(a), where the 10 kernels are processed sequentially. The 9-bit timestamp, including the input event polarity bit, is first multiplied with all values in the current kernel i in a 5×5 multiplier array. The psums of the feature map elements associated to kernel i and input pixel $\{x, y\}$ coordinates, stored in a 16-kB SRAM, are then updated. Due to SRAM aspect ratio constraints, this update is split into four 256-bit read/write accesses whose locations are given by an address decoder. An overflow protection mechanism emulates a hardtanh activation function. Frame-based maxpooling is automatically carried out after the FIFO has been emptied and the 8-bit timestamp counter falls to zero. It is followed by a quantization to 6 bits with configurable rescaling. The CONV core then outputs 490 6-bit activations (CONV_OUT) and a CONV_DONE trigger to enable the FC core, which is clock-gated otherwise.

Depending on the event-driven sensor use case, two features can be used to adjust the accuracy-energy tradeoff. First, the *first-spike gating* block can be enabled to keep only the most-informative first spike of each pixel, thus dropping subsequent events. Second, the INFER_REQ pin can be used to request inference at any time by triggering maxpooling before the 8-bit timestamp counter falls to zero, thus ignoring all subsequent less-informative events.

4.2.1.2 Fully-connected (FC) core

The FC core consists of a 128-neuron fully-connected hidden layer followed by a 10-neuron output layer, corresponding to parameters $FC_1 = 128$ and $FC_2 = 10$ in

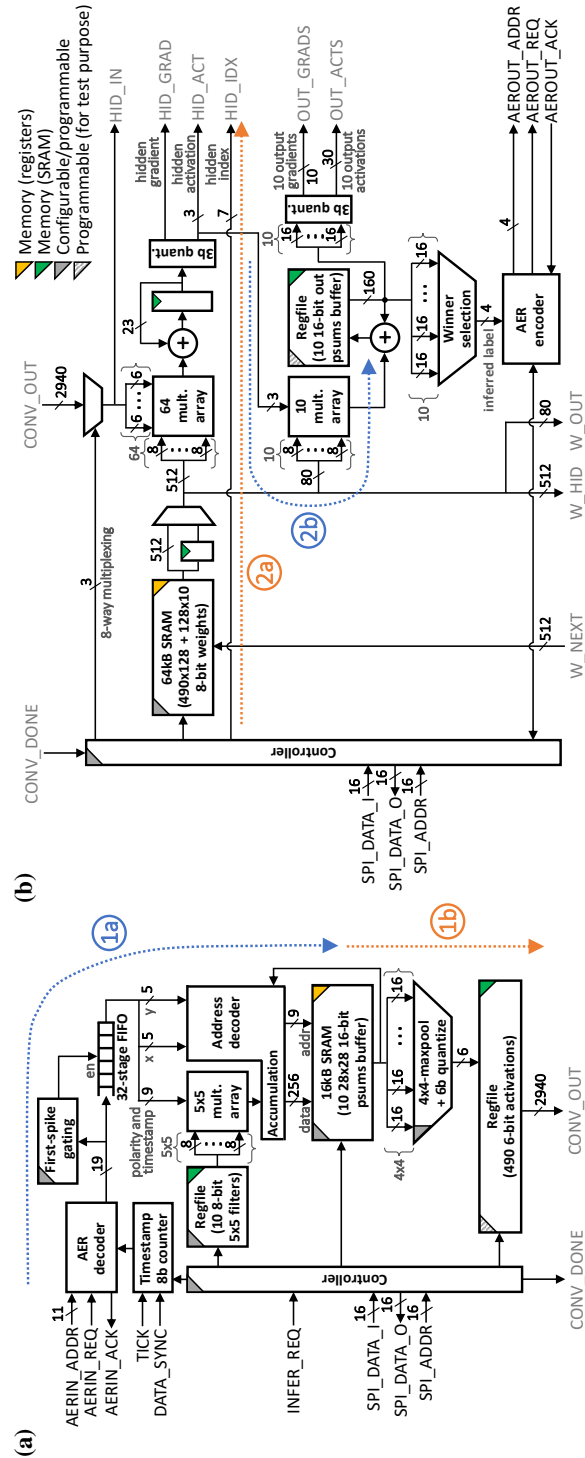


Fig. 4.3 Circuit architecture of (a) the CONV core and of (b) the FC core. Detailed control signals, data gating and overflow protection mechanisms are not shown for clarity. Event-driven and frame-based processing are highlighted following the conventions of Fig. 4.1.

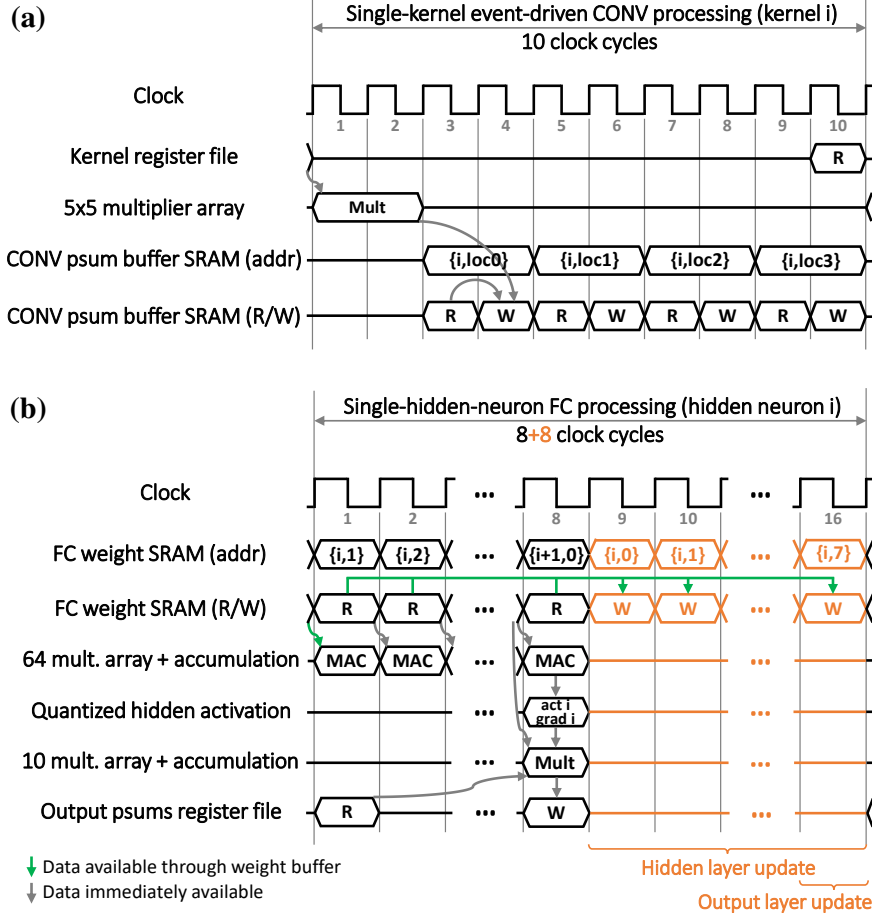


Fig. 4.4 Timing diagrams for (a) event-driven convolution in the CONV core and (b) combination of hidden and output layer processing in the FC core. Black: cycles required for inference. Orange: optional DRTP update cycles.

Fig. 4.1. All weights have an 8-bit width, are programmable and automatically initialized to zero for online learning (Section 4.2.1.3). The circuit architecture and the associated timing diagram are shown in Figs. 4.3(b) and 4.4(b), respectively. As highlighted in Fig. 4.1, the hidden layer output $y_{hid} = f_{hid}(W_{hid}x)$ is computed with a conventional frame-based approach as all the inputs are immediately available when receiving the CONV_DONE trigger, where W_{hid} represents the hidden layer weights, f_{hid} is the hidden layer activation function and x is the input from the CONV core. The hidden neurons are evaluated sequentially and inputs are processed by batch of 64. It requires 8 cycles to retrieve the 500 weights associated to a hidden neuron, including both the 490 weights $W_{hid,i}$ connecting to the inputs and the 10 weights $W_{out,i}$ connecting to the output layer neurons, where the index i denotes hidden neuron i . Once the weighted sum of inputs $W_{hid,i}x$ of hidden neuron i has been computed, output layer processing is triggered in an event-driven fashion to ensure maximum data reuse:

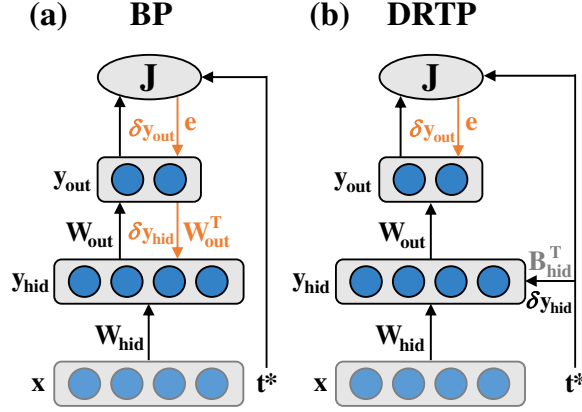


Fig. 4.5 Summary of (a) the backpropagation of error (BP) algorithm [12] compared to (b) the proposed direct random target projection (DRTP) algorithm. Adapted from Fig. 3.1 for the single-hidden-layer case of SPOON.

- the activation $y_{hid,i}$ is obtained by quantizing $W_{hid,i}x$ to 3 bits with a hardtanh function, whose binary derivative is one in the linear range and zero elsewhere (HID_ACT and HID_GRAD in Fig. 4.3(b)),
- if the derivative is non-zero, DRTP updates can be directly applied to $W_{hid,i}$ (Fig. 4.4, orange), otherwise they are skipped (Section 4.2.1.3),
- $W_{out,i}y_{hid,i}$ is added to the 10 output psums,
- as the final activation and derivative of the output neurons are not yet available for the current sample, a DRTP update of $W_{out,i}$ is triggered based on buffered previous sample data (Section 4.2.1.3).

Finally, when all the hidden neurons have been processed, the output layer activation $y_{out} = f_{out}(W_{out}y_{hid})$ is obtained by quantizing the output psums to 3 bits with a hardsigmoid function, whose binary derivative is one in the linear range and zero elsewhere (OUT_ACTS and OUT_GRADS in Fig. 4.3(b)).

4.2.1.3 On-chip online training with direct random target projection (DRTP)

Building on feedback alignment techniques [175, 176], which were proposed to solve the *weight transport problem* of BP [12], the direct random target projection (DRTP) algorithm that we proposed in Chapter 3 releases not only the weight transport problem, but also *update locking*. As these are the two key issues that preclude BP from being hardware-efficient and biologically plausible, DRTP is a low-cost algorithm suitable for deployment at the edge. It relies only on feedforward and local computation and estimates the hidden layer loss gradient δy_{hid} as a projection of the target vector t^* with a fixed random matrix B_{hid} (Fig. 4.5). This operation

corresponds to a simple label-dependent random vector selection, which can be quantized down to binary resolutions with only a negligible impact on DRTP performance. The hidden layer weight updates ΔW_{hid} can then be computed following

$$\Delta W_{hid} = -\eta_{hid} (\delta y_{hid} \odot f'_{hid}(W_{hid}x)) x^T, \quad (4.1)$$

where $\delta y_{hid} = B_{hid}^T t^*$, η_{hid} is the hidden layer learning rate and $\odot$ denotes element-wise multiplication. As opposed to BP, δy_{hid} is always non-zero in DRTP, the weights can thus be initialized to zero without precluding training convergence. The DRTP output layer update is identical to the BP update, i.e.

$$\Delta W_{out} = -\eta_{out} (e \odot f'_{out}(W_{out}y_{hid})) y_{hid}^T, \quad (4.2)$$

where η_{out} is the output layer learning rate.

The circuit architecture for the DRTP weight update module of SPOON is shown in Fig. 4.6. According to Eq. (4.1), the derivative f'_{hid} (HID_GRAD) of hidden neuron i , taking values 0 or 1, can be used as an enable signal for the hidden layer update module (Section 4.2.1.2). The fixed random binary matrix B_{hid} is stored in a register file. A specific bit is selected based on the current hidden layer neuron index (HID_IDX) and the training sample label (LABEL). Therefore, the only required computation is a label-dependent sign inversion to the inputs from the CONV core (HID_IN), processed by batch of 64. The obtained values are then used as probabilities conditioning random increments/decrements to the hidden layer weights $W_{hid,i}$ (W_HID), following the random numbers generated by a linear feedback shift register (LFSR) and a configurable learning rate. In order to parallelize the generation of 64 12-bit seeds with a single LFSR, we applied the unfolding technique [163], whose application to SNN stochastic updates was proposed for MorphIC in Section 2.3.1.2.

The output layer update follows Eq. (4.2), whose terms are buffered so that updates from the previous sample to output layer weights $W_{out,i}$ (W_OUT) can be applied concurrently with the hidden layer updates of the current sample. The error e is computed based on the previous label and output activations (OUT_ACTS), where the binary output derivatives (OUT_GRADS) act as a gating signal. If the previous hidden neuron activation is zero, updates are skipped. Otherwise, it is multiplied with the error and used in the stochastic updates array, which operates as in the hidden layer update module.

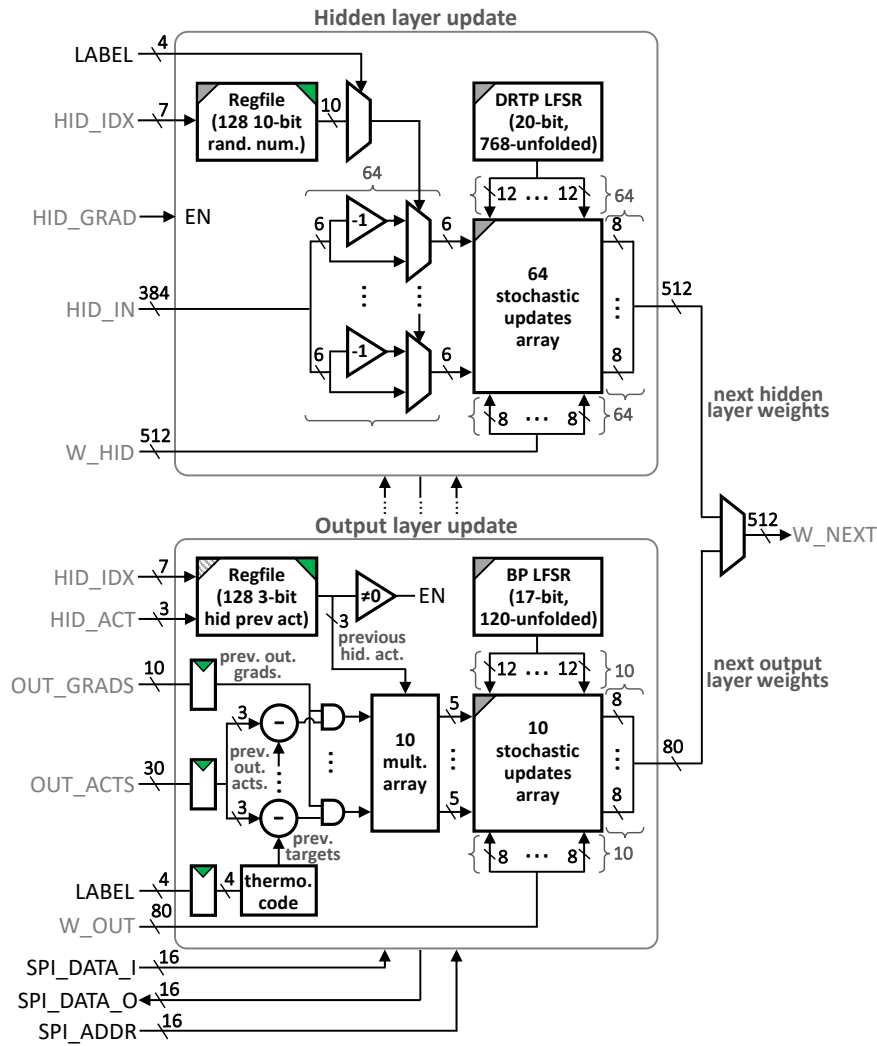


Fig. 4.6 Circuit architecture for the DRTP weight update module.

4.2.2 Specifications and benchmarking results

SPOON has been taped out at the time of writing this thesis in a 28-nm FDSOI CMOS process, the layout is presented in Fig. 4.7, while the specifications and pre-silicon performance metrics are reported in Table 4.2. It occupies a post-shrink area of only 0.32mm². At 150MHz, the maximum throughput in the CONV core amounts to 1.5Mevents/s, while the FC core adds 6.8μs per inference. If online learning is enabled, depending on the amount of neuron updates, the FC core latency can reach a maximum of 13.6μs (Fig. 4.4). At 0.6V, SPOON has a leakage power of 61μW and the convolutional layer update consumes 1.7nJ per event in the CONV core. When convolution is over, the FC core requires 55nJ to update the fully-connected layers and to send the inferred label with the output AER bus. The high suitability of DRTP for adaptive edge computing is highlighted by its

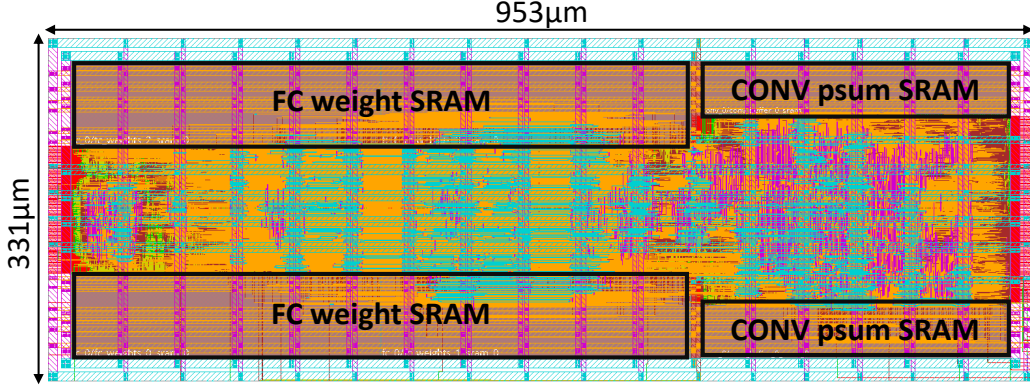


Fig. 4.7 SPOON layout view obtained under Cadence Innovus.

Table 4.2 Specifications and pre-silicon performance metrics of SPOON.

Technology	28nm FDSOI CMOS
Implementation	Digital
Area	0.32mm ² (0.26mm ² excl. rails)
Topology	C5×5@10-FC128-FC10
Online learning	Stochastic DRTP, 8-bit weights
Time constant	Biological to accelerated
Supply voltage	0.6V – 1.0V
Max. clock frequency	150MHz
Leakage power	61μW at 0.6V
CONV core throughput	1.5Mevents/s at 150MHz
FC core time	6.8μs/inference at 150MHz
Energy for CONV core	1.7nJ/event at 0.6V
Energy for FC core	55nJ/inference at 0.6V
Online learning overhead	16.8% in power, 11.8% in area

low power and area overheads of 16.8% and 11.8% compared to a design without online learning, respectively. In order to accelerate benchmarking, the accuracy results in the subsequent text were retrieved from an FPGA implementation of SPOON.

With the MNIST dataset of handwritten digits [150], SPOON requires only 313nJ per inference using time-to-first-spike encoding. Training SPOON for MNIST using an off-chip optimizer based on PyTorch [203] with quantization-aware training [204], we reach a test-set accuracy of 97.5%. When enabling on-chip DRTP-based online learning, where the convolution kernels are initialized and fixed to random values and plastic fully-connected weights are initialized to zero, SPOON reaches a test-set accuracy of 92.8% after one epoch and 95.3% after 100 epochs on the 60k-sample training set. We will analyze the energy-area-accuracy tradeoff of SPOON on MNIST compared to previously-proposed BNNs, SNNs and conventional ANN/CNN machine learning accelerators in Section 4.3.

With the N-MNIST dataset [198], as each active pixel spikes in average 4.8 times per saccade and thus leads to redundant information when interpreted as a time-to-first-spike code, we use the single-spike-per-pixel mode of SPOON. Using only the first saccade of each sample, SPOON reaches a test-set accuracy of 93.8% with offline-trained weights and of 90.2% (one epoch) or 93.0% (100 epochs) using on-chip online training, while consuming 665nJ per inference.

4.3 Discussion

While bottom-up SNN designs favor a comparison based on low-level criteria such as neuron behaviors, synaptic plasticity and resolution, neuron and synapse densities, energy per SOP, fan-in and fan-out (Section 2.4), top-down neuromorphic approaches require a comparison based on benchmark performance as they start from the applicative problem. As most of the top-down designs presented in Table 4.1 have been benchmarked on MNIST, we will adopt this benchmark as a common ground for comparison in this section. Among bottom-up SNN designs, only MorphIC and TrueNorth have been benchmarked on MNIST¹, which shows that MNIST is still a difficult dataset for bottom-up neuromorphic chips. As MNIST is popular among conventional ANN and CNN machine learning accelerators, it allows us to extract the main trends between SNN, BNN, ANN and CNN accelerators. While MNIST allows revealing that the efficiency of neuromorphic chips is catching up with state-of-the-art machine learning accelerators, MNIST does not capture the key dimension inherent to spiking neural networks and neuromorphic computing: time [43]. N-MNIST is a first step in this direction, however its time dimension is artificial as it is generated with a spiking retina from static images. Moreover, while it is popular for the development of spike-based algorithms and software- or FPGA-based SNNs (e.g., see [173] for a review), to the best of our knowledge, none of the bottom-up and top-down neuromorphic designs discussed in this thesis were benchmarked on N-MNIST, except in [57] where Pei *et al.* use this dataset to quantify the efficiency and throughput improvement of Tianjic over GPUs. This further highlights the need for widely-accepted neuromorphic datasets embedding relevant timing information, as recently called for by Mike Davies, the lead designer of the Intel Loihi chip [44].

The tradeoff analysis of energy, area and accuracy on the MNIST dataset is shown in Fig. 4.8, which has been normalized to a 28-nm technology node to allow for fair comparisons. SNNs appear to lag clearly behind conventional ANN and CNN accelerators [158, 205, 206], the BNN from Park *et al.* [50] and SPOON. It can be seen that, among SNNs, the MorphIC neuromorphic processor proposed in Chapter 2 achieves a high area efficiency without incurring a power penalty. We demonstrate with Fig. 4.8 that only SPOON reaches the efficiency of conventional

¹ We do not include results obtained on pre-processed versions of MNIST, such as the those reported for ODIN in Section 2.2.2.3.



Fig. 4.8 Analysis of tradeoffs between accuracy, area and energy per classification on the MNIST dataset for SNNs, BNNs, ANNs and CNNs. Although MorphIC and the chip from Chen *et al.* embed online learning, the MNIST experiments of these two chips were obtained with offline-learned weights. All chips are digital and allow for technology normalization, except the design from Buhler *et al.* which is mixed-signal [49]. **(a)** Area-accuracy tradeoff. Silicon area (excluding pads) has been normalized to a 28-nm technology node using the node factor (e.g., a $(28/65)^2$ -fold reduction for normalizing 65nm to 28nm), except for the 10-nm FinFET node from Chen *et al.* [56] where data from [120] was used for normalization. The TrueNorth area varies as Esser *et al.* used different numbers of cores for their experiments (5, 20, 80 and 120 cores, in the order of increasing accuracy) [207]. A 1920-core configuration is also reported in [207], leading to a 99.42-% accuracy on MNIST with TrueNorth, a record for SNNs. However, as this configuration would lead to a normalized area of 980mm², we only included TrueNorth configurations whose scale are comparable with the other chips. **(b)** Energy-accuracy tradeoff. Energy has been normalized to a 28-nm technology node using the node factor (e.g., a $(28/65)$ -fold reduction for normalizing 65nm to 28nm). The point shown in light gray for MorphIC is a prediction resulting from the power optimizations proposed in Table 2.4 of Section 2.3.2.

machine learning accelerators while enabling online learning with event-based sensors, thanks to a tight combination of event-driven and frame-based processing supported by DRTP training. Similar trends were also recently outlined in Tianjic by Pei *et al.*, where a hybrid ANN-SNN network was demonstrated to outperform the equivalent SNN-only network. These findings form an interesting trend worth investigating for the deployment of top-down neuromorphic designs in real-world applications.

Finally, another aspect that does not appear in Fig. 4.8 is related to the image throughput of these systems, a metric that is often reported for conventional machine learning accelerators. However, the time dimension is intrinsically tied to the spiking output of event-based sensors such as neuromorphic retinas, reflecting the dynamic contents of the scene. Therefore, the relevance of assessing the image throughput for SPOON as well as for SNNs in the context of edge computing is debatable: a better metric for these systems is the maximum input event throughput that they can accommodate for, as it will limit their application scope for real-time operation. However, for the sake of comparison on MNIST, SPOON achieves an image throughput of 8.5k img/sec. It compares favorably to MorphIC (250 img/sec at 0.55V, could be increased by $4\times$ at maximum frequency and 1.2V), TrueNorth (1k img/sec) [207] and the chip from Moons *et al.* (150, 486 or 1.7k img/sec at minimum energy point as for the data reported in Fig. 4.8, could be accelerated by $8\times$ if operated at maximum frequency) [158], while it is similar to the chip from Chen *et al.* (14.3k img/sec) [206]. However, the chips from Whatmough *et al.* (with 62k img/s in low-power mode to 112k img/s in high-throughput mode) [205] and from Park *et al.* (with 100k img/sec) [50] are able to demonstrate a clear throughput advantage. Finally, the chips from Kim *et al.* (with 1.6M to 9.9M img/sec) [48] and from Buhler *et al.* (with 1.7M img/sec) [49] provide the highest throughput given their relaxed classification accuracy.

Conclusions and perspectives

Pursuing the far-reaching goal of embedding cognition *in silico* requires as a first step to unveil the roads worth investigating. Taking a neuromorphic engineering perspective, we analyzed whether this objective should be approached *bottom-up*, *top-down* or with a combination of both approaches by respectively raising the three following questions:

Question 1 *How can a wide diversity of biophysical behaviors be embedded in silico without precluding an efficient large-scale integration?*

Question 2 *How can a weight-transport-free and update-unlocked training be carried out toward efficient online-learning neuromorphic processors?*

Question 3 *What are the synergies between bottom-up and top-down design strategies toward bringing the efficiency of online-learning spike-based neuromorphic systems closer to their biological counterparts?*

By delving into neuromorphic processor design from both the bottom-up and top-down standpoints in Parts I and II of this thesis, we can now take a step back and outline answers to the first two questions, which are backed by three neuromorphic processor prototypes. Leveraging the synergies between both approaches requires articulating them in a coherent framework, which is left open-ended as it actually might underlie the very concept of cognition. As further research is required along the three neuromorphic axes of neuroscience, computer science and electrical engineering, we will conclude this thesis by answering the third question in the form of perspectives.

Insight from the bottom-up standpoint

The bottom-up road to embedded cognition is fed by neuroscience observations and hindered by the silicon implementation efficiency of neuron versatility, synaptic plasticity and communication infrastructure scalability. In order to solve this challenge, we first studied the main neuron and synapse models under the light of three implementation-driven requirements: (i) a purely event-driven operation to avoid the control and power overheads of time-stepped integration, (ii) a minimization of in-synapse computation, and (iii) the compatibility with single-port foundry SRAMs for high neuron and synapse densities with time-multiplexed operation. We demonstrated that, while subthreshold analog designs offer a natural answer with a direct emulation of the brain biophysics, digital designs offer a pragmatic answer if

the right neuroscience abstraction level is chosen, at the expense of a few simplifications (e.g., pulse model assumption, point neurons, single plasticity mechanism). For the neurons, we showed that an event-driven digital implementation of the 20 Izhikevich behaviors from a phenomenological viewpoint provides a biophysical versatility over area tradeoff outperforming those of analog neuron implementations. For the synapses, we showed that spike-dependent synaptic plasticity (SDSP), as opposed to spike-timing-dependent plasticity (STDP), allows meeting the last two requirements by offloading online learning computation to the neurons, leveraging plasticity updates driven by pre-synaptic activity.

These findings drove the design of a first proof-of-concept neuromorphic processor: ODIN. By embedding 256 neurons with the 20 Izhikevich behaviors and 64k 4-bit SDSP-based synapses at densities of 3k neurons and 741k synapses per mm^2 in 28-nm FDSOI CMOS, ODIN demonstrates record neuron and synapse densities compared to state-of-the-art neuromorphic designs while maintaining a high energy efficiency with 12.7pJ/SOP. We then built on these results for a second neuromorphic chip, this time aiming at large-scale integration: MorphIC. It achieves fan-in and fan-out values of 1k and 2k, similarly to the values found in the brain. By embedding four cores totaling 2k LIF neurons and more than 2M S-SDSP-based binary synapses in an active silicon area of 2.86mm^2 in 65-nm LP CMOS, corresponding to 28-nm normalized densities of 3.8k neurons and 4M synapses per mm^2 , MorphIC further demonstrates the scalability of the proposed approach. Therefore, we answer the first question of this thesis as follows:

Claim 1 Hardware-aware neuroscience model design and selection allows reaching record neuron and synapse densities with low-power operation for large-scale integration *in silico*.

However, we also showed that bottom-up designs struggle to achieve the efficiency of conventional machine learning accelerators for real-world problems: identifying suitable applications beyond the scope of neuroscience is thus difficult. This aspect can be alleviated by following a top-down approach.

Insight from the top-down standpoint

The top-down road to embedded cognition is fed by efficient engineering solutions to real-world problems, under both the constraint and guidance of bio-inspiration. With the scope of adaptive edge computing in mind, we thus started by looking at the algorithm at the roots of neural network training: the backpropagation of error (BP) algorithm. As two aspects of BP preclude both bio-inspiration and hardware efficiency, i.e. the weight transport problem and update locking, we studied how to release them while maintaining learning in multi-layer neural networks. Building on feedback alignment (FA) algorithms, which only release weight transport, we derived the direct random target projection (DRTP) algorithm: it (i) eliminates the

requirement for an explicit feedback pathway (i.e. no weight transport), (ii) allows for weight updates to be computed in each layer independently as soon as they have been evaluated (i.e. no update locking) and (iii) relies on low-cost local and feedforward computation by evaluating layerwise loss gradients with a label-dependent random vector selection (i.e. relaxed computational, structural and memory requirements). We demonstrated on MNIST and CIFAR-10 that, despite its simplicity and low cost, DRTP still provides accuracy metrics comparable to BP and FA-based algorithms.

In order to show the efficiency of DRTP for hardware devices constrained during both the training and the inference phases, we designed a third neuromorphic processor: SPOON. In order to support spiking neuromorphic retinas with embedded online learning while minimizing the power and area footprints, we leveraged event-driven processing for convolution of input events and combined it with frame-based processing in the fully-connected layers to maximize data reuse. SPOON occupies only 0.32mm^2 of silicon area in 28-nm FDSOI CMOS and keeps the online learning cost to a minimum with 16.8-% power and 11.8-% area overheads. We demonstrated the suitability of SPOON for event-based data processing with N-MNIST, while we used the MNIST dataset to highlight that, with $313\text{nJ}/\text{inference}$ at accuracies of 95.3-% (online learning) or 97.5-% (offline training), only SPOON reaches the efficiency of conventional machine learning accelerators while supporting adaptive edge computing with event-based sensors. Therefore, we answer the second question of this thesis as follows:

Claim 2 Combining event-driven and frame-based processing with weight-transport-free update-unlocked training supports low-cost adaptive edge computing with spike-based sensors.

However, the efficiency and relevance of top-down design for neuromorphic engineering is conditioned by the bio-inspired elements that are considered as essential. This judgment bears key importance, yet it is often not sufficiently grounded on theoretical and/or experimental evidence. More bottom-up investigation is thus needed to identify useful computational primitives and mechanisms of the brain, and to distinguish them from artifacts induced by evolution to compensate for the non-idealities of the biological substrate.

Open-ended perspectives toward embedded cognition

While our bottom-up and top-down investigations allowed identifying guidelines for the design of efficient neuromorphic hardware, our conclusion from both standpoints is that each approach can act as a *guide* to address the shortcomings of the other. Early steps in this direction can be seen in the recent Braindrop and Tianjic chips. We may therefore answer the last question of this thesis as follows:

Claim 3 Top-down guidance helps pushing bottom-up neuron and synapse integration beyond the purpose of neuroscience experimentation platforms, while bottom-up guidance supports top-down design toward brain reverse-engineering.

Finally, let us mention that engineering, by breaking problems into pieces, is an inherently *divergent* approach, while the goal of embedding cognition in neuromorphic systems requests a *convergent* and *integrative* view. It is therefore necessary to come up with new methodologies, which we think are required not only for the global approach (i.e. bottom-up or top-down), but also along the processing chain (i.e. from sensing to decision through computation) and down to the technological design choices, as it can shed a new light on the long-standing question of whether analog or digital implementation should be pursued.

Bibliography

- [1] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115-133, 1943.
- [2] F. Rosenblatt, *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*, Sparta, NJ, USA: Spartan Books, 1961.
- [3] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554-2558, 1982.
- [4] A. Krizhevsky, I. Sutskever and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Proc. of Advances in Neural Information Processing Systems (NIPS)*, pp. 1097-1105, 2012.
- [5] Y. LeCun, Y. Bengio, and G. E. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, p. 436, 2015.
- [6] K. He et al., “Deep residual learning for image recognition,” *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778, 2016.
- [7] G. E. Hinton et al., “Deep neural networks for acoustic modeling in speech recognition,” *IEEE Signal Processing Magazine*, vol. 29, 2012.
- [8] D. Amodei et al., “Deep speech 2: End-to-end speech recognition in english and mandarin,” *Proc. of International Conference on Machine Learning (ICML)*, vol. 173-182, 2016.
- [9] K. He et al., “Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification,” *Proc. of IEEE International Conference on Computer Vision (ICCV)*, pp. 1026-1034, 2015.
- [10] M. Moravčík, et al., “Deepstack: Expert-level artificial intelligence in heads-up no-limit poker,” *Science*, vol. 356, no. 6337, pp. 508-513, 2017.
- [11] J. Olczak et al., “Artificial intelligence for analyzing orthopedic trauma radiographs: Deep learning algorithms – Are they on par with humans for diagnosing fractures?,” *Acta Orthopaedica*, vol. 88, no. 6, pp. 581-586, 2017.
- [12] D. Rumelhart, G. Hinton and R. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, pp. 533-536, 1986.

- [13] D. Bol, G. de Streel and D. Flandre, "Can we connect trillions of IoT sensors in a sustainable way? A technology/circuit perspective," *Proc. of IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, 2015.
- [14] W. Shi et al. "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637-646, 2016.
- [15] V. Sze et al., "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295-2329, 2017.
- [16] Y. LeCun, "Deep Learning Hardware: Past, Present, and Future," *Proc. of IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 12-19, 2019.
- [17] A. Krizhevsky, *Learning multiple layers of features from tiny images*, Technical Report, University of Toronto, 2009.
- [18] D. Bankman et al., "An always-on 3.8 μ J/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 158-172, 2019.
- [19] C. S. Poon and K. Zhou, "Neuromorphic silicon neurons and large-scale neural networks: challenges and opportunities," *Frontiers in Neuroscience*, vol. 5, no. 108, 2011.
- [20] F. Sandin et al., "Concept learning in neuromorphic vision systems: What can we learn from insects?," *Journal of Software Engineering and Applications*, vol. 7, no. 5, pp. 387-395, 2014.
- [21] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.
- [22] G. Indiveri and S.-C. Liu, "Memory and information processing in neuromorphic systems," *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1379-1397, 2015.
- [23] M. Horowitz, "Computing's energy problem (and what we can do about it)," *Proc. of IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 10-14, 2014.
- [24] S. Thorpe, A. Delorme and R. Van Rullen, "Spike-based strategies for rapid processing," *Neural Networks*, vol. 14, no. 6-7, pp. 715-725, 2001.
- [25] J.-S. Seo et al., "A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," *Proc. of IEEE Custom Integrated Circuits Conference (CICC)*, 2011.
- [26] J. Park et al., "A 65k-neuron 73-Mevents/s 22-pJ/event asynchronous micro-pipelined integrate-and-fire array transceiver," *Proc. of IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2014.

- [27] N. Qiao et al., "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses," *Frontiers in Neuroscience*, vol. 9, no. 141, 2015.
- [28] C. Mayr et al., "A biological-realtime neuromorphic system in 28 nm CMOS using low-leakage switched capacitor circuits," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 1, pp. 243-254, 2016.
- [29] S. Moradi et al., "A scalable multicore architecture with heterogeneous memory structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 1, pp. 106-122, 2018.
- [30] J. Schemmel et al., "A wafer-scale neuromorphic hardware system for large-scale neural modeling," *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1947-1950, 2010.
- [31] E. Painkras et al., "SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 8, pp. 1943-1953, 2013.
- [32] B. V. Benjamin et al., "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699-716, 2014.
- [33] F. Akopyan et al., "TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537-1557, 2015.
- [34] M. Davies et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82-99, 2018.
- [35] S. B. Furber et al., "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652-665, 2014.
- [36] G. Cauwenberghs, "Reverse engineering the cognitive brain," *Proceedings of the National Academy of Sciences*, vol. 110, no. 39, pp. 15512-15513, 2013.
- [37] R. J. Vogelstein et al., "A silicon central pattern generator controls locomotion in vivo," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 2, no. 3, pp. 212-222, 2008.
- [38] R. George et al., "Event-based softcore processor in a biohybrid setup applied to structural plasticity," *Proc. of IEEE International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, 2015.
- [39] F. Corradi and G. Indiveri, "A neuromorphic event-based neural recording system for smart brain-machine-interfaces," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 9, no. 5, pp. 699-709, 2015.

- [40] F. Boi et al., "A bidirectional brain-machine interface featuring a neuromorphic hardware decoder," *Frontiers in Neuroscience*, vol. 10, no. 563, 2016.
- [41] Y. Sandamirskaya, "Dynamic neural fields as a step toward cognitive neuromorphic architectures," *Frontiers in Neuroscience*, vol. 7, no. 276, 2014.
- [42] M. B. Milde et al., "Obstacle avoidance and target acquisition for robot navigation using a mixed signal analog/digital neuromorphic processing system," *Frontiers in Neurorobotics*, vol. 11, no. 28, 2017.
- [43] G. Indiveri and Y. Sandamirskaya, "The importance of space and time in neuromorphic cognitive agents," *IEEE Signal Processing Magazine*, vol. 32, no. 6, pp. 16-28, 2019.
- [44] M. Davies, "Benchmarks for progress in neuromorphic computing," *Nature Machine Intelligence*, vol. 1, no. 9, pp. 386-388, 2019.
- [45] G. Indiveri, E. Chicca and R. J. Douglas, "Artificial cognitive systems: from VLSI networks of spiking neurons to neuromorphic cognition," *Cognitive Computation*, vol. 1, no. 2, pp. 119-127, 2009.
- [46] C. Koch, *Biophysics of computation: information processing in single neurons*, Oxford university press, 1999.
- [47] P. Knag et al., "A sparse coding neural network ASIC with on-chip learning for feature extraction and encoding," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 4, pp. 1070-1079, 2015.
- [48] J. K. Kim et al., "A 640M pixel/s 3.65 mW sparse event-driven neuromorphic object recognition processor with on-chip learning," *IEEE Symposium on VLSI Circuits (VLSI-C)*, pp. C50-C51, 2015.
- [49] F. N. Buhler et al., "A 3.43 TOPS/W 48.9 pJ/pixel 50.1 nJ/classification 512 analog neuron sparse coding neural network with on-chip learning and classification in 40nm CMOS," *IEEE Symposium on VLSI Circuits (VLSI-C)*, pp. C30-C31, 2017.
- [50] J. Park, J. Lee and D. Jeon, "A 65nm 236.5 nJ/classification neuromorphic processor with 7.5% energy overhead on-chip learning using direct spike-only feedback," *Proc. of IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 140-142, 2019.
- [51] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," *Cognitive Science*, vol. 11, no. 1, pp. 23-63, 1987.
- [52] E. O. Neftci, "Data and power efficient intelligence with neuromorphic learning machines," *iScience*, vol. 5, pp. 52-68, 2018.

- [53] M. Jaderberg et al., “Decoupled neural interfaces using synthetic gradients,” *Proc. of International Conference on Machine Learning (ICML)*, vol. 70, pp. 1627-1635, 2017.
- [54] W. Czarnecki et al., “Understanding synthetic gradients and decoupled neural interfaces,” *Proc. of International Conference on Machine Learning (ICML)*, vol. 70, pp. 904-912, 2017.
- [55] P. Baldi, P. Sadowski and Z. Lu, “Learning in the machine: Random backpropagation and the deep learning channel,” *Artificial intelligence*, vol. 260, pp. 1-35, 2018.
- [56] G. K. Chen et al., “A 4096-neuron 1M-synapse 3.8pJ/SOP spiking neural network with on-chip STDP learning and sparse weights in 10-nm FinFET CMOS,” *IEEE Journal of Solid-State Circuits*, vol. 54, no. 4, pp. 992-1002, 2019.
- [57] J. Pei et al., “Towards artificial general intelligence with hybrid Tianjic chip architecture,” *Nature*, vol. 572, no. 7767, p. 106, 2019.
- [58] A. Neckar et al., “Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model,” *Proceedings of the IEEE*, vol. 107, no. 1, pp. 144-164, 2019.
- [59] C. Eliasmith and C. H. Anderson, *Neural engineering: Computation, representation, and dynamics in neurobiological systems*, MIT press, 2004.
- [60] W. Gerstner et al., *Neuronal dynamics: From single neurons to networks and models of cognition*, Cambridge University Press, 2014.
- [61] K. R. Jessen, “Glial cells,” *The International Journal of Biochemistry & Cell Biology*, vol. 36, no. 10, pp. 1861-1867, 2004.
- [62] L. Lapicque, “Recherches quantitatives sur l’excitation électrique des nerfs traitée comme une polarisation,” *Journal de Physiologie et de Pathologie Générale*, vol. 9, pp. 620-635, 1907.
- [63] A. N. Burkitt, “A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input,” *Biological cybernetics*, vol. 95, no. 1, pp. 1-19, 2006.
- [64] A. L. Hodgkin and A. F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve,” *Journal of Physiology*, vol. 117, no. 4, pp. 500-544, 1952.
- [65] E. M. Izhikevich, “Simple model of spiking neurons,” *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569-1572, 2003.
- [66] E. M. Izhikevich, “Which model to use for cortical spiking neurons?,” *IEEE Trans. on Neural Networks*, vol. 15, no. 5, pp. 1063-1070, 2004.

- [67] R. Brette and W. Gerstner, "Adaptive exponential integrate-and-fire model as an effective description of neuronal activity," *Journal of Neurophysiology*, vol. 94, no. 5, pp. 3637-3642, 2005.
- [68] J. V. Arthur and K. Boahen, "Learning in silicon: Timing is everything," *Proc. of Advances in Neural Information Processing Systems (NIPS)*, pp. 75-82, 2006.
- [69] G. Indiveri, F. Stefanini and E. Chicca, "Spike-based learning with a generalized integrate and fire silicon neuron," *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1951-1954, 2010.
- [70] B. Rueckauer et al., "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in Neuroscience*, vol. 11, no. 682, 2017.
- [71] Q. Yu, et al., "Rapid feedforward computation by temporal encoding and learning with spiking neurons," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 10, pp. 1539-1552, 2013.
- [72] N. Qiao, C. Bartolozzi and G. Indiveri, "An ultralow leakage synaptic scaling homeostatic plasticity circuit with configurable time scales up to 100 ks," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 6, pp. 1271-1277, 2017.
- [73] S. Sheik, E. Chicca and G. Indiveri, "Exploiting device mismatch in neuromorphic VLSI systems to implement axonal delays," *Proc. of IEEE International Joint Conference on Neural Networks (IJCNN)*, 2012.
- [74] V. Rangan et al., "A subthreshold aVLSI implementation of the Izhikevich simple neuron model," *Proc. of Int. Conf. of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 4164-4167, 2010.
- [75] S. A. Aamir et al., "A mixed-signal structured AdEx neuron for accelerated neuromorphic cores," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 5, pp. 1027-1037, 2018.
- [76] R. Naud et al., "Firing patterns in the adaptive exponential integrate-and-fire model," *Biological Cybernetics*, vol. 99, no. 335, 2008.
- [77] J. Wijekoon and P. Dudek, "Compact silicon neuron circuit with spiking and bursting behaviour," *Neural Networks*, vol. 21, no. 2, pp. 524-534, 2008.
- [78] J. L. Molin et al., "Low-power, low-mismatch, highly-dense array of VLSI Mihalas-Niebur neurons," *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2533-2536, 2017.
- [79] F. Folowosele et al., "A switched capacitor implementation of the generalized linear integrate-and-fire neuron," *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2149-2152, 2009.

- [80] I. Sourikopoulos et al., "A 4-fJ/spike artificial neuron in 65 nm CMOS technology," *Frontiers in Neuroscience*, vol. 11, no. 123, 2017.
- [81] A. Joubert et al., "Hardware spiking neurons design: Analog or digital?," *Proc. of IEEE International Joint Conference on Neural Networks (IJCNN)*, 2012.
- [82] D. Liang and G. Indiveri, "A neuromorphic computational primitive for robust context-dependent decision making and context-dependent stochastic computation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 5, pp. 843-847, 2019.
- [83] N. Imam et al., "Neural spiking dynamics in asynchronous digital circuits," *Proc. of IEEE Int. Joint Conf. on Neural Net. (IJCNN)*, 2013.
- [84] P. Merolla et al., "A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm," *Proc. of IEEE Custom Integrated Circuits Conference (CICC)*, 2011.
- [85] A. S. Cassidy et al., "Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores," *Proc. of IEEE International Joint Conference on Neural Networks (IJCNN)*, 2013.
- [86] H. Soleimani, A. Ahmadi and M. Bavandpour, "Biologically inspired spiking neurons: Piecewise linear models and digital implementation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 12, pp. 2991-3004, 2012.
- [87] S. Yang et al., "Cost-efficient FPGA implementation of basal ganglia and their Parkinsonian analysis," *Neural Networks*, vol. 71, pp. 62-75, 2015.
- [88] A. Mortara and E. A. Vittoz, "A communication architecture tailored for analog VLSI artificial neural networks: intrinsic performance and limitations," *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 459-466, 1994.
- [89] K. A. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 47, no. 5, pp. 416-434, 2000.
- [90] M. R. Azghadi et al., "Spike-based synaptic plasticity in silicon: design, implementation, application, and challenges," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 717-737, 2014.
- [91] F. Zenke, E. J. Agnes and W. Gerstner, "Diverse synaptic plasticity mechanisms orchestrated to form and retrieve memories in spiking neural networks", *Nature Communications*, vol. 6, no. 6922, 2015.
- [92] G. Indiveri, "Computing cycle – Neuromorphic computing," *IMEC Academy Tutorial*, 2015.

- [93] R. S. Zucker and W. G. Regehr, "Short-term synaptic plasticity," *Annual Review of Physiology*, vol. 64, no. 1, pp. 355-405, 2002.
- [94] D. J. Amit, *Modeling brain function: The world of attractor neural networks*, Cambridge university press, 1992.
- [95] G. G. Turrigiano and S. B. Nelson, "Homeostatic plasticity in the developing nervous system," *Nature Reviews Neuroscience*, vol. 5, no. 2, p. 97, 2004.
- [96] C. Bartolozzi, O. Nikolayeva and G. Indiveri, "Implementing homeostatic plasticity in VLSI networks of spiking neurons," *Proc. of IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 682-685, 2008.
- [97] R. Lamprecht and J. LeDoux, "Structural plasticity and memory," *Nature Reviews Neuroscience*, vol. 5, no. 1, p. 45, 2004.
- [98] G. G. Bi, and M. M. Poo, "Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type," *Journal of Neuroscience*, vol. 18, no. 24, pp. 10464-10472, 1998.
- [99] J. Schemmel et al., "Implementing synaptic plasticity in a VLSI spiking neural network model," *Proc. of IEEE International Joint Conference on Neural Network (IJCNN)*, 2006.
- [100] H. Tanaka, T. Morie and K. Aihara, "A CMOS spiking neural network circuit with symmetric/asymmetric STDP function," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 92 no. 7, pp. 1690-1698, 2009.
- [101] S. Ramakrishnan, P. E. Hasler and C. Gordon, "Floating gate synapses with spike-time-dependent plasticity," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 5, no. 3, pp. 244-252, 2011.
- [102] J. M. Cruz-Albrecht, M. W. Yung and N. Srinivasa, "Energy-efficient neuron, synapse and STDP integrated circuits," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 6, no. 3, pp. 246-256, 2012.
- [103] S. A. Bamford, A. F. Murray and D. J. Willshaw, "Spike-timing-dependent plasticity with weight dependence evoked from physical constraints," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 6, no. 4, pp. 385-398, 2012.
- [104] S. Song, K. D. Miller and L. F. Abbott, "Competitive Hebbian learning through spike-timing-dependent synaptic plasticity," *Nature Neuroscience*, vol. 3, no. 9, p. 919, 2000.
- [105] A. Cassidy, A. G. Andreou and J. Georgiou, "A combinational digital logic approach to STDP," *Proc. of IEEE International Symposium of Circuits and Systems (ISCAS)*, pp. 673-676, 2011.

- [106] J. M. Brader, W. Senn and S. Fusi, "Learning real-world stimuli in a neural network with spike-driven synaptic dynamics," *Neural Computation*, vol. 19, no. 11, pp. 2881-2912, 2007.
- [107] M. Giulioni et al., "A VLSI network of spiking neurons with plastic fully configurable "stop-learning" synapses," *Proc. of IEEE Int. Conference on Electronics, Circuits and Systems (ICECS)*, pp. 678-681, 2008.
- [108] S. Mitra, S. Fusi and G. Indiveri, "Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI," *IEEE Trans. on Biomedical Circuits and Systems*, vol. 3, no. 1, pp. 32-42, 2009.
- [109] S. Fusi, "Hebbian spike-driven synaptic plasticity for learning patterns of mean firing rates," *Biological Cybernetics*, vol. 87, pp. 459-470, 2002.
- [110] G. Maimon and J. A. Assad, "Beyond Poisson: Increased spike-time regularity across primate parietal cortex," *Neuron*, vol. 62, no. 3, pp. 426-440, 2009.
- [111] E. L. Bienenstock, L. N. Cooper and P. W. Munro, "Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex," *Journal of Neuroscience*, vol. 2, no. 1, pp. 32-48, 1982.
- [112] J.-P. Pfister and W. Gerstner, "Triplets of spikes in a model of spike timing-dependent plasticity," *Journal of Neuroscience*, vol. 26, no. 38, pp. 9673-9682, 2006.
- [113] M. Graupner and N. Brunel, "Mechanisms of induction and maintenance of spike-timing dependent plasticity in biophysical synapse models," *Frontiers in Neuroscience*, vol. 4, no. 136, 2010.
- [114] R. Urbanczik and W. Senn, "Learning by the dendritic prediction of somatic spiking," *Neuron*, vol. 81, no. 3, pp. 521-528, 2014.
- [115] F. L. Maldonado Huayaney, S. Nease and E. Chicca, "Learning in silicon beyond STDP: A neuromorphic implementation of multi-factor synaptic plasticity with calcium-based dynamics," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 12, pp. 2189-2199, 2016.
- [116] M. Payvand et al., "A neuromorphic systems approach to in-memory computing with non-ideal memristive devices: From mitigation to exploitation," *Faraday Discussions*, pp. 487-510, 2018.
- [117] P. Lin, S. Pi and Q. Xia, "3D integration of planar crossbar memristive devices with CMOS substrate," *Nanotechnology*, vol. 25, no. 40, ID 405202, 2014.
- [118] J. Rofeh et al., "Vertical integration of memristors onto foundry CMOS dies using wafer-scale integration," *Proc. of IEEE Electronic Components and Technology Conference (ECTC)*, pp. 957-962, 2015.

- [119] O. Thomas et al., "Dynamic single-p-well SRAM bitcell characterization with back-bias adjustment for optimized wide-voltage-range SRAM operation in 28nm UTBB FD-SOI," *Proc. of IEEE International Electron Devices Meeting (IEDM)*, 2014.
- [120] K. Mistry, "10nm technology leadership," *Leading at the Edge: Intel Technology and Manufacturing Day*, 2017 [Online]. Available: <https://newsroom.intel.com/newsroom/wp-content/uploads/sites/11/2017/03/Kaizad-Mistry-2017-Manufacturing.pdf>.
- [121] G. Chen et al., "A dense 45nm half-differential SRAM with lower minimum operating voltage," *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 57-60, 2011.
- [122] T. Pfeil et al., "Is a 4-bit synaptic weight resolution enough? – Constraints on enabling spike-timing dependent plasticity in neuromorphic hardware," *Frontiers in Neuroscience*, vol. 6, no. 90, 2012.
- [123] G. Indiveri et al., "Neuromorphic silicon neuron circuits," *Frontiers in Neuroscience*, vol. 5, no. 73, 2011.
- [124] E. Kauderer-Abrams et al., "A population-level approach to temperature robustness in neuromorphic systems," *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017.
- [125] B. De Salvo, "Brain-inspired technologies: Towards chips that think?," *IEEE International Solid-State Circuits Conference-(ISSCC)*, pp. 12-18, 2018.
- [126] J. Schemmel et al., "Modeling synaptic plasticity within networks of highly accelerated I&F neurons," *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 3367-3370, 2007.
- [127] S. Friedmann et al., "Demonstrating hybrid learning in a flexible neuromorphic hardware system," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 1, pp. 128-142, 2017.
- [128] S. Billaudelle et al., "Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate," *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, *accepted*.
- [129] M. Noack et al., "Switched-capacitor realization of presynaptic short-term-plasticity and stop-learning synapses in 28 nm CMOS," *Frontiers in neuroscience*, vol. 9, p. 10, 2015.
- [130] N. Verma et al., "In-memory computing: Advances and prospects," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 3, pp. 43-55, 2019.
- [131] G. Indiveri et al., "Integration of nanoscale memristor synapses in neuromorphic computing architectures," *Nanotechnology*, vol. 24, no. 38, p. 384010, 2013.

- [132] F. Cai et al., "A fully integrated reprogrammable memristor-CMOS system for efficient multiply-accumulate operations," *Nature Electronics*, vol. 2, no. 7, pp. 290-299, 2019.
- [133] S. Brink et al., "A learning-enabled neuron array IC based upon transistor channel models of biological phenomena," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 7, no. 1, pp. 71-81, 2013.
- [134] J. M. Bower and D. Beeman, *The book of GENESIS: exploring realistic neural models with the GEneral NEural SIMulation System*, Springer Science & Business Media, 2012.
- [135] N. T. Carnevale and M. L. Hines, *The NEURON book*, Cambridge University Press, 2006.
- [136] M.-O. Gewaltig and M. Diesmann, "NEST (NEural Simulation Tool)," *Scholarpedia*, vol. 2, no. 4, 2007.
- [137] D. Goodman and R. Brette, "Brian: a simulator for spiking neural networks in Python", *Frontiers in Neuroinformatics*, vol. 2, no. 5, 2008.
- [138] F. Zenke and W. Gerstner, "Limits to high-speed simulations of spiking neural networks using general-purpose computers," *Frontiers in Neuroinformatics*, vol. 8, no. 76, 2014.
- [139] J. Vitay, H. U. Dinkelbach and F. H. Hamker, "ANNarchy: a code generation approach to neural simulations on parallel hardware," *Frontiers in Neuroinformatics*, vol. 9, no. 19, 2015.
- [140] E. Yavuz, J. Turner and T. Nowotny, "GeNN: a code generation framework for accelerated brain simulations," *Scientific reports*, vol. 6, no. 18854, 2016.
- [141] M. Stimberg, R. Brette and D. Goodman, "Brian 2, an intuitive and efficient neural simulator," *eLife*, vol. 8, no. e47314, 2019.
- [142] J. C. Knight and T. Nowotny, "GPUs outperform current HPC and neuromorphic solutions in terms of speed and energy when simulating a highly-connected cortical model," *Frontiers in Neuroscience*, vol. 12, no. 941, 2018.
- [143] C. Liu et al., "Memory-efficient deep learning on a SpiNNaker 2 prototype," *Frontiers in Neuroscience*, vol. 12, no. 840, 2018.
- [144] S. Höppner and C. Mayr, "SpiNNaker 2 – Towards extremely efficient digital neuromorphics and multi-scale brain emulation," *Proc. of Neuro Inspired Computational Elements Workshop (NICE)*, 2018.
- [145] A. Cassidy, J. Georgiou and A. G. Andreou, "Design of silicon brains in the nano-CMOS era: Spiking neurons, learning synapses and neural architecture optimization," *Neural Networks*, vol. 45, pp. 4-26, 2013.

- [146] D. Neil and S.-C. Liu, "Minitaur, an event-driven FPGA-based spiking network accelerator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 12, pp. 2621-2628, 2014.
- [147] R. Wang and A. van Schaik, "Breaking Liebig's law: An advanced multi-purpose neuromorphic engine," *Frontiers in Neuroscience*, vol. 12, no. 593, 2018.
- [148] J. Luo et al., "Real-time simulation of passage-of-time encoding in cerebellum using a scalable FPGA-based system," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 3, pp. 742-753, 2016.
- [149] S. Yang et al., "Real-Time Neuromorphic System for Large-Scale Conductance-Based Spiking Neural Networks," *IEEE Transactions on Cybernetics*, 2018.
- [150] Y. LeCun and C. Cortes, "The MNIST database of handwritten digits," 1998 [Online]. Available: <http://yann.lecun.com/exdb/mnist/>.
- [151] S.-W. Moon, J. Rexford and K. G. Shin, "Scalable hardware priority queue architectures for high-speed packet switches," *IEEE Transactions on Computers*, vol. 49, no. 11, pp. 1215-1227, 2000.
- [152] I. Hubara et al., "Binarized neural networks," *Proc. of Advances in Neural Information Processing Systems (NIPS)*, pp. 4107-4115, 2016.
- [153] B. Moons et al., "Minimum energy quantized neural networks," *Proc. of IEEE Asilomar Conference on Signals, Systems, and Computers*, 2017.
- [154] P. U. Diehl et al., "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," *Proc. of International Joint Conference on Neural Networks (IJCNN)*, 2015.
- [155] R. Kreiser et al., "On-chip unsupervised learning in winner-take-all networks of spiking neurons," *Proc. of IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 424-427, 2017.
- [156] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345-1359, 2009.
- [157] R. Andri et al. "YodaNN: An architecture for ultralow power binary-weight CNN acceleration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 48-60, 2018.
- [158] B. Moons et al., "BinarEye: An always-on energy-accuracy-scalable binary CNN processor with all memory on chip in 28nm CMOS," *Proc. of IEEE Custom Integrated Circuits Conference (CICC)*, 2018.
- [159] D. Bassett and E. D. Bullmore, "Small-world brain networks," *The Neuroscientist*, vol. 12, no. 6, pp. 512-523, 2006.

- [160] J. Navaridas et al., “Understanding the interconnection network of SpiNNaker,” *Proc. of ACM International Conference on Supercomputing (ICS)*, pp. 286-295, 2009.
- [161] J. Park et al., “Hierarchical address event routing for reconfigurable large-scale neuromorphic systems,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2408-2422, 2017.
- [162] W. Senn and S. Fusi, “Convergence of stochastic learning in perceptrons with binary synapses,” *Physical Review E*, vol. 71, no. 6, p. 061907, 2005.
- [163] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*, John Wiley & Sons, 1999.
- [164] C. Cheng and K. K. Parhi, “High-speed parallel CRC implementation based on unfolding, pipelining, and retiming,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, no. 10, pp. 1017-1021, 2006.
- [165] G. Indiveri, F. Corradi and N. Qiao, “Neuromorphic architectures for spiking deep neural networks,” *Proc. of IEEE International Electron Devices Meeting (IEDM)*, 2015.
- [166] E. Stomatias et al., “Scalable energy-efficient, low-latency implementations of trained spiking deep belief networks on SpiNNaker,” *Proc. of IEEE International Joint Conference on Neural Networks (IJCNN)*, 2015.
- [167] P. A. Merolla et al., “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science*, vol. 345, no. 6197, pp. 668-673, 2014.
- [168] E. Ceolini et al., “Sensor fusion using EMG and vision for hand gesture classification in mobile applications”, *Proc. of IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2019.
- [169] E. Ceolini et al., “Hand-gesture recognition based on EMG and event-based camera sensor fusion: A benchmark in neuromorphic computing,” *Frontiers in Neuroscience*, 2020, *submitted*.
- [170] S. Höppner et al., “Dynamic voltage and frequency scaling for neuromorphic many-core systems,” *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017.
- [171] J. Partzsch et al., “A fixed point exponential function accelerator for a neuromorphic many-core system,” *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017.
- [172] T. Pirson, *Training ultra-low-power spiking neural networks for neuromorphic IoT vision sensing and recognition*, Master thesis, Université catholique de Louvain, 2019.

- [173] A. Sironi et al., “HATS: Histograms of averaged time surfaces for robust event-based object classification,” *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1731-1740, 2018.
- [174] A. Yousefzadeh et al., “On practical issues for stochastic STDP hardware with 1-bit synaptic weights,” *Frontiers in Neuroscience*, vol. 12, p. 665, 2018.
- [175] T. P. Lillicrap et al., “Random synaptic feedback weights support error backpropagation for deep learning,” *Nature Communications*, vol. 7, no. 13276, 2016.
- [176] A. Nøkland, “Direct feedback alignment provides learning in deep neural networks,” *Proc. of Advances in Neural Information Processing Systems (NIPS)*, pp. 1037-1045, 2016.
- [177] J. Guerguiev, T. P. Lillicrap and A. Richards, “Towards deep learning with segregated dendrites,” *ELife*, vol. 6, no. e22901, 2017.
- [178] E. Neftci et al., “Event-driven random back-propagation: Enabling neuro-morphic deep learning machines,” *Frontiers in Neuroscience*, vol. 11, p. 324, 2017.
- [179] N. Zheng and P. Mazumder, “Online supervised learning for hardware-based multilayer spiking neural networks through the modulation of weight-dependent spike-timing-dependent plasticity,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 9, pp. 4287-4302, 2017.
- [180] H. Mostafa, “Supervised learning based on temporal coding in spiking neural networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 7, pp. 3227-3235, 2017.
- [181] F. Zenke and S. Ganguli, “SuperSpike: Supervised learning in multilayer spiking neural networks,” *Neural computation*, vol. 30, no. 6, pp. 1514-1541, 2018.
- [182] M. Minsky, “Steps toward artificial intelligence,” *Proceedings of the IRE*, vol. 49, no. 1, pp. 8-30, 1961.
- [183] Q. Liao, J. Z. Leibo and T. Poggio, “How important is weight symmetry in backpropagation?,” *Proc. of AAAI Conference on Artificial Intelligence*, 2016.
- [184] H. Mostafa, V. Ramesh and G. Cauwenberghs, “Deep supervised learning using local errors,” *Frontiers in Neuroscience*, vol. 12, no. 608, 2018.
- [185] J. Kaiser, H. Mostafa and E. Neftci, “Synaptic plasticity dynamics for deep continuous local learning,” *arXiv preprint arXiv:1811.10766*, 2018.
- [186] A. Nøkland and L. H. Eidnes, “Training neural networks with local error signals,” *Proc. of International Conference on Machine Learning (ICML)*, 2019.

- [187] J. Deng et al., "ImageNet: A large-scale hierarchical image database," *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248-255, 2009.
- [188] D. H. Lee et al., "Difference target propagation," in *Proc. of Springer Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 498-515, 2015.
- [189] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research (JMLR)*, vol. 12, pp. 2825-2830, 2011.
- [190] C. S. Thakur et al., "Large-scale neuromorphic spiking array processors: A quest to mimic the brain," *Frontiers in Neuroscience*, vol. 12, no. 891, 2018.
- [191] B. Rajendran et al., "Low-Power Neuromorphic Hardware for Signal Processing Applications", *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 97-110, 2019.
- [192] G. Detorakis et al., "Neural and synaptic array transceiver: A brain-inspired computing framework for embedded learning," *Frontiers in Neuroscience*, vol. 12, no. 583, 2018.
- [193] J. Launay, I. Poli and F. Krzakala, "Principled Training of Neural Networks with Direct Feedback Alignment", *arXiv preprint arXiv:1906.04554*, 2019.
- [194] J. Zylberberg, J. T. Murphy and M. R. DeWeese, "A sparse coding model with synaptically local plasticity and spiking neurons can account for the diverse shapes of V1 simple cell receptive fields," *PLoS Computational Biology*, vol. 7, no. 10, p. e1002250, 2011.
- [195] P. Lichtsteiner, C. Posch and T. Delbruck, "A 128×128 120 dB $15\mu\text{s}$ latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no.2 , pp. 566-576, 2008.
- [196] C. Posch, D. Matolin and R. Wohlgenannt, "A QVGA 143 dB dynamic range frame-free PWM image sensor With lossless pixel-level video compression and time-domain CDS," *IEEE Journal of Solid State Circuits*, vol. 46, no. 1, pp. 259-275, 2011.
- [197] C. Brandli et al., "A 240×180 130 db $3\mu\text{s}$ latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333-2341, 2014.
- [198] G. Orchard et al., "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in Neuroscience*, no. 9, p. 437, 2015.
- [199] A. Voelker, *Dynamical systems in spiking neuromorphic hardware*, Doctoral dissertation, University of Waterloo, 2019.

- [200] G. Orchard et al., "HFirst: A temporal approach to object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 10, pp. 2028-2040, 2015.
- [201] K. Goetschalckx and M. Verhelst, "Breaking High Resolution CNN Bandwidth Barriers with Enhanced Depth-First Execution," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 323-331, 2019
- [202] A. Vanarse, A. Osseiran and A. Rassau, "A review of current neuromorphic approaches for vision, auditory, and olfactory sensors," *Frontiers in Neuroscience*, no. 10, p. 115, 2016.
- [203] A. Paszke et al., "Automatic differentiation in PyTorch", *Proc. of Annual Conf. on Neural Information Processing Systems (NIPS) Workshop*, 2017.
- [204] I. Hubara et al., "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869-6898, 2017.
- [205] P. N. Whatmough et al., "A 28nm SoC with a 1.2 GHz 568nJ/prediction sparse deep-neural-network engine with >0.1 timing error rate tolerance for IoT applications," *Proc. of IEEE International Solid-State Circuits Conference (ISSCC)*, 2017.
- [206] Y. Chen et al., "A 2.86-TOPS/W current mirror cross-bar-based machine-learning and physical unclonable function engine for Internet-of-Things applications" *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 6, pp. 2240-2252, 2019.
- [207] S. K. Esser et al., "Backpropagation for energy-efficient neuromorphic computing," *Advances in Neural Information Processing Systems (NIPS)*, pp. 1117-1125, 2015.

Direct random target projection – Methods and proofs

A.1 Methods

The training on both the synthetic regression and classification tasks and the MNIST and CIFAR-10 datasets has been carried out with PyTorch [203], one of the numerous Python frameworks supporting deep learning. The PyTorch code allowing to reproduce our experiments can be found at <https://github.com/ChFrenkel/DirectRandomTargetProjection>.

Regression. The examples in the training and test sets are denoted as (x, t^*) . The 10-dimensional target vectors t^* are generated using $t_j^* = T_j(x) = \cos(\bar{x} + \phi_j)$, where $\phi_j = -\pi/2 + j\pi/9$ for $j \in [0, 9]$. $\bar{x}$ denotes the mean of x , a 256-dimensional vector whose entries are initialized from a normal distribution with a mean sampled from a uniform distribution between $-\pi$ and π and with a unit variance. The training and test sets respectively contain 5k and 1k examples. The trained network has a 256-100-100-10 topology with tanh hidden and output units, whose forward weights are drawn from a He uniform distribution [9] and are zero-initialized for feedback-alignment-based algorithms. The random connectivity matrices of feedback-alignment-based algorithms are also drawn from He uniform distributions. The weights are updated after each minibatch of 50 examples, and the network is trained for 500 epochs with a fixed learning rate $\eta = 5 \times 10^{-4}$ for all training algorithms. As this is a regression task, the loss function is the mean squared error. The losses on the training and test sets and the alignment angles with BP updates are monitored every 1k samples. The experiment is repeated 10 times for each training algorithm, with different network initializations for each experiment run.

Synthetic data classification. The examples in the training and test sets are generated using the `make_classification` function from the Python library `sklearn` [189]. The main inputs required by this function are the number of samples to be generated, the number of features n in the input vectors x , the number of informative features n_{inf} among the input vectors, the number of classes, the number of clusters per class and a factor `class_sep` which conditions the class separation. In this work, we have used $n = 256$ and $n_{inf} = 128$, ten classes, five clusters per class and `class_sep` = 4.5. Using this set of parameters, the `make_classification` function then generates examples by creating for each class

Table A.1 The learning rate values for the MNIST and CIFAR-10 datasets are selected based on a grid search. A different learning rate is chosen for each training algorithm, dataset and network type (Section 3.2.4).

Dataset	Network	BP	FA	DFA	sDFA	DRTP	Shallow
MNIST	FC1	1.5×10^{-4}	5×10^{-4}	1.5×10^{-4}	5×10^{-4}	1.5×10^{-4}	1.5×10^{-2}
	FC2	5×10^{-4}	1.5×10^{-4}	5×10^{-4}	5×10^{-4}	1.5×10^{-4}	5×10^{-3}
	CONV (random)	5×10^{-5}	1.5×10^{-4}	5×10^{-5}	5×10^{-4}	5×10^{-4}	5×10^{-3}
	CONV (trained)	5×10^{-4}	5×10^{-5}	5×10^{-5}	1.5×10^{-4}	1.5×10^{-4}	–
CIFAR-10	FC1	1.5×10^{-5}	1.5×10^{-5}	1.5×10^{-5}	5×10^{-5}	1.5×10^{-4}	1.5×10^{-4}
	FC2	5×10^{-6}	5×10^{-6}	5×10^{-6}	5×10^{-5}	5×10^{-5}	5×10^{-4}
	CONV (random)	5×10^{-6}	5×10^{-6}	5×10^{-6}	1.5×10^{-4}	1.5×10^{-4}	1.5×10^{-3}
	CONV (trained)	1.5×10^{-4}	5×10^{-6}	5×10^{-6}	1.5×10^{-5}	5×10^{-5}	–

clusters of points normally distributed about the vertices of an n_{inf} -dimensional hypercube. The remaining features are filled with normally-distributed random noise. The generated examples are then separated into training and test sets of 25k and 5k examples, respectively. The trained network has a 256-500-500-10 topology with tanh hidden units and sigmoid output units. The forward and backward weights initialization, as well as the forward weight updates, are performed as for regression. As this is a classification task, the loss function is the binary cross-entropy loss. The network is trained for 500 epochs with a fixed learning rate $\eta = 5 \times 10^{-4}$. The losses on the training and test sets and the alignment angles with BP updates are monitored every 2.5k samples. For each training algorithm, the experiment is repeated 10 times with different network initializations.

MNIST and CIFAR-10 images classification. A fixed learning rate is selected based on a grid search for each training algorithm, dataset and network type (Table A.1). For both MNIST and CIFAR-10 experiments, the chosen optimizer is Adam with default parameters. A sigmoid output layer and a binary cross-entropy loss are used for all training algorithms. The entries of the forward weight matrices W_k are initialized with a He uniform distribution, as well as the entries of the fixed random connectivity matrices B_k of feedback-alignment-based algorithms. When used, dropout is applied with the same probability to all fully-connected layers. For MNIST, the networks are trained for 100 epochs with a minibatch size of 60. The CONV network topology consists of a convolutional layer with 32 5×5 kernels, a stride of 1 and a padding of 2, a max-pooling layer with 2×2 kernels and a stride of 2, a fully-connected layer of 1000 tanh units and an output fully-connected layer of 10 units. For CIFAR-10, a minibatch size of 100 is used and early stopping is applied, with a maximum of 200 epochs. The CONV network topology consists of two convolutional layers with respectively 64 and 256 3×3 kernels, a stride and a padding of 1, both followed by a max-pooling layer with 2×2 kernels and a stride of 2, then two fully-connected layers of 1000 tanh units and an output fully-connected layer of 10 units. For all experiments, the test error is averaged over the last 10 epochs of training. The results reported in Tables 3.1, 3.2, A.2 and A.3 are the mean and standard deviation over 10 trials.

A.2 Full proof of alignment for the BP and DRTP modulatory signals

This full version of the alignment proof between the BP and DRTP modulatory signals is derived under the assumptions outlined in Section 3.2.3, i.e. a neural network composed of linear hidden layers (Fig. 3.4) and a single training example (x, c^*) , where x is the input data sample and c^* the label. The C -dimensional target vector t^* corresponds to the one-hot encoding of c^* , where C is the number of classes. Our developments build on the alignment proof of [175], which demonstrates that the FA and BP modulatory signals are within 90° of each other in the case of a single linear hidden layer, a linear output layer and a mean squared error loss. In the framework of classification problems, we extend this proof for the case of DRTP and to an arbitrary number of linear hidden layers, a nonlinear output layer of sigmoid units and a binary cross-entropy loss.

Network dynamics. The output of the k -th linear hidden layer is given by

$$y_k = z_k = W_k y_{k-1} \quad \text{for } k \in [1, K-1],$$

where K is the number of layers and $y_0 = x$ is the input vector. The output layer is described by

$$\begin{aligned} z_K &= W_K y_{K-1} \\ y_K &= \sigma(z_K), \end{aligned}$$

where $\sigma(\cdot)$ is the sigmoid activation function. The loss function $J(\cdot)$ is the binary cross-entropy loss, computed over the C output classes,

$$J(y_K, t^*) = -\frac{1}{C} \sum_{c=1}^C \left(t_c^* \log(y_{Kc}) + (1 - t_c^*) \log(1 - y_{Kc}) \right).$$

The network is trained with stochastic gradient descent. In the output layer, the weight updates of both BP and DRTP follow

$$W_K \leftarrow W_K - \eta \frac{\partial J}{\partial y_K} \frac{\partial y_K}{\partial z_K} \frac{\partial z_K}{\partial W_K}.$$

The factors in this update can be computed as

$$\frac{\partial J}{\partial y_{Kc}} = \begin{cases} \frac{-1}{C} \frac{1}{y_{Kc}} & \text{if } c = c^* \\ \frac{-1}{C} \frac{-1}{(1 - y_{Kc})} & \text{otherwise} \end{cases}$$

$$\frac{\partial y_K}{\partial z_K} = y_K (1 - y_K).$$

It results that

$$\frac{\partial J}{\partial z_K} = \frac{1}{C} (y_K - t^*) = \frac{e}{C},$$

where e is the error vector, and therefore that

$$W_K \leftarrow W_K - \frac{\eta}{C} e y_{K-1}^T.$$

In the hidden layers, the weight updates follow

$$W_k \leftarrow W_k - \eta \delta y_k y_{k-1}^T.$$

On the one hand, if the training relies on the BP algorithm, the modulatory signals $\delta z_k = \delta y_k$ correspond to the loss function gradient:

$$\delta y_k = \delta z_k = \frac{\partial J}{\partial y_k} = \frac{1}{C} \left(\prod_{i=k+1}^K W_i^T \right) e.$$

On the other hand, if the DRTP algorithm is used, the modulatory signals are projections of the one-hot-encoded target vector t^* through fixed random connectivity matrices B_k :

$$\delta y_k = \delta z_k = B_k^T t^*.$$

In order to provide learning, the modulatory signals prescribed by BP and DRTP must be within 90° of each other, i.e. their dot product must be positive:

$$e^T \left(\prod_{i=k+1}^K W_i^T \right)^T B_k^T t^* > 0.$$

Lemma. In the case of zero-initialized weights, i.e. $W_k^0 = 0$ for $k \in [1, K]$, and hence of zero-initialized hidden layer outputs, i.e. $y_k^0 = 0$ for $k \in [1, K-1]$ and $z_K^0 = 0$, considering a DRTP-based training performed recursively with a single element of the training set (x, c^*) and t^* denoting the one-hot encoding of c^* , at every discrete time step t , there are positive scalars $s_{y_k}^t$ and $s_{W_k}^t$ for $k \in [1, K-1]$

and a C -dimensional vector $s_{W_K}^t$ such that

$$\begin{aligned} y_k^t &= -s_{y_k}^t (B_k^T t^*) & \text{for } k \in [1, K-1] \\ W_1^t &= -s_{W_1}^t (B_1^T t^*) x^T \\ W_k^t &= s_{W_k}^t (B_k^T t^*) (B_{k-1}^T t^*)^T & \text{for } k \in [2, K-1] \\ W_K^t &= -s_{W_K}^t (B_{K-1}^T t^*)^T. \end{aligned}$$

Proof. The lemma is proven by induction.

For $t = 0$, the conditions required to satisfy the lemma are trivially met by choosing $s_{y_k}^0, s_{W_k}^0 = 0$ for $k \in [1, K-1]$, and $s_{W_K}^0$ as a zero vector, given that $y_k^0 = 0$ for $k \in [1, K-1]$ and $W_k^0 = 0$ for $k \in [1, K]$.

For $t > 0$, considering that the conditions are satisfied at a given discrete time step t , it must be shown that they still hold at the next discrete time step $t + 1$. In the hidden layers, the weights are updated using the modulatory signals prescribed by DRTP. For the first hidden layer, we have

$$\begin{aligned} W_1^{t+1} &= W_1^t - \eta B_1^T t^* x^T \\ &= -s_{W_1}^t (B_1^T t^*) x^T - \eta (B_1^T t^*) x^T \\ s_{W_1}^{t+1} &= s_{W_1}^t + \eta = s_{W_1}^t + \Delta s_{W_1}^t \end{aligned}$$

and for subsequent hidden layers, i.e. for $k \in [2, K-1]$, we have

$$\begin{aligned} W_k^{t+1} &= W_k^t - \eta B_k^T t^* y_{k-1}^{tT} \\ &= s_{W_k}^t (B_k^T t^*) (B_{k-1}^T t^*)^T + \eta s_{y_{k-1}}^t (B_k^T t^*) (B_{k-1}^T t^*)^T \\ s_{W_k}^{t+1} &= s_{W_k}^t + \eta s_{y_{k-1}}^t = s_{W_k}^t + \Delta s_{W_k}^t. \end{aligned}$$

The weights in the output layer are updated according to the loss function gradient, thus leading to

$$\begin{aligned} W_K^{t+1} &= W_K^t - \frac{\eta}{C} (y_K^t - t^*) y_{K-1}^{tT} \\ &= W_K^t + \frac{\eta}{C} (y_K^t - t^*) s_{y_{K-1}}^t (B_{K-1}^T t^*)^T \\ &= -s_{W_K}^t (B_{K-1}^T t^*)^T + \frac{\eta s_{y_{K-1}}^t}{C} (y_K^t - t^*) (B_{K-1}^T t^*)^T \\ s_{W_K}^{t+1} &= s_{W_K}^t - \frac{\eta s_{y_{K-1}}^t}{C} (y_K^t - t^*). \end{aligned}$$

The output of the first hidden layer is

$$\begin{aligned}
y_1^{t+1} &= W_1^{t+1} x \\
&= (W_1^t - \eta B_1^T t^* x^T) x \\
&= \underbrace{W_1^t x}_{y_1^t} - \eta x^T x (B_1^T t^*) \\
&= -s_{y_1}^t (B_1^T t^*) - \eta \|x\|^2 (B_1^T t^*) \\
s_{y_1}^{t+1} &= s_{y_1}^t + \eta \|x\|^2 = s_{y_1}^t + \Delta s_{y_1}^t
\end{aligned}$$

and the output of the k -th hidden layer for $k \in [2, K-1]$ is given by

$$\begin{aligned}
y_k^{t+1} &= W_k^{t+1} y_{k-1}^{t+1} \\
&= -s_{W_k}^{t+1} (B_k^T t^*) (B_{k-1}^T t^*)^T s_{y_{k-1}}^{t+1} (B_{k-1}^T t^*) \\
&= -\left(s_{W_k}^t + \eta s_{y_{k-1}}^t\right) \left(s_{y_{k-1}}^t + \Delta s_{y_{k-1}}^t\right) \|B_{k-1}^T t^*\|^2 (B_k^T t^*) \\
&= -s_{W_k}^t s_{y_{k-1}}^t \|B_{k-1}^T t^*\|^2 (B_k^T t^*) \\
&\quad - \left(s_{W_k}^t \Delta s_{y_{k-1}}^t + \eta s_{y_{k-1}}^t \left(s_{y_{k-1}}^t + \Delta s_{y_{k-1}}^t\right)\right) \|B_{k-1}^T t^*\|^2 (B_k^T t^*) \\
&= -s_{y_k}^t (B_k^T t^*) - \left(s_{W_k}^t \Delta s_{y_{k-1}}^t + \eta s_{y_{k-1}}^t \left(s_{y_{k-1}}^t + \Delta s_{y_{k-1}}^t\right)\right) \|B_{k-1}^T t^*\|^2 (B_k^T t^*) \\
s_{y_k}^{t+1} &= s_{y_k}^t + \left(s_{W_k}^t \Delta s_{y_{k-1}}^t + \eta s_{y_{k-1}}^t \left(s_{y_{k-1}}^t + \Delta s_{y_{k-1}}^t\right)\right) \|B_{k-1}^T t^*\|^2 = s_{y_k}^t + \Delta s_{y_k}^t.
\end{aligned}$$

The coefficients $s_{W_1}^t$ and $s_{y_1}^t$ are updated with strictly positive quantities $\Delta s_{W_1}^t$ and $\Delta s_{y_1}^t$ at each time step t and are thus strictly positive for $t > 0$. Furthermore, the coefficients $s_{W_k}^t$ and $s_{y_k}^t$ are updated based on the coefficients of the previous layer and will therefore be strictly positive for $k \in [1, K-1]$. $\square$

Theorem. Under the same conditions as in the lemma and for the linear-hidden-layer network dynamics described above, the k -th layer modulatory signals prescribed by DRTP are always a positive scalar multiple of the Moore-Penrose pseudo-inverse of the product of forward matrices of layers $k+1$ to K , located in the feedback pathway between the output layer and the k -th hidden layer, multiplied by the error. That is, for $k \in [1, K-1]$ and $t > 0$,

$$\left(\prod_{i=k}^{K-1} W_i^t\right)^+ e = s_k^t B_k^T t^* \quad \text{with} \quad s_k^t > 0.$$

Proof. When replacing the forward weights W_i^t by the expressions given in the lemma, the above equality becomes

$$\begin{aligned}
& - \left[\left(\prod_{i=K-1}^{k+1} s_{W_i}^t \right) s_{W_K}^t \left(\prod_{i=K-1}^{k+1} \|B_i^T t^*\|^2 \right) (B_k^T t^*)^T \right]^+ (y_K^t - t^*) = s_k^t B_k^T t^* \\
& - \left(\prod_{i=k+1}^{K-1} s_{W_i}^t \right)^{-1} \left(\prod_{i=k+1}^{K-1} \|B_i^T t^*\|^2 \right)^{-1} \left[s_{W_K}^t (B_k^T t^*)^T \right]^+ (y_K^t - t^*) = s_k^t B_k^T t^* \\
& - \left(\prod_{i=k+1}^{K-1} s_{W_i}^t \right)^{-1} \left(\prod_{i=k+1}^{K-1} \|B_i^T t^*\|^2 \right)^{-1} (B_k^T t^*)^{T+} s_{W_K}^{t+} (y_K^t - t^*) = s_k^t B_k^T t^* \\
& - \left(\prod_{i=k+1}^{K-1} s_{W_i}^t \right)^{-1} \left(\prod_{i=k+1}^{K-1} \|B_i^T t^*\|^2 \right)^{-1} \underbrace{\|B_k^T t^*\|^{-2} (B_k^T t^*)}_{(B_k^T t^*)^{T+}} \underbrace{\|s_{W_K}^t\|^{-2} s_{W_K}^{tT}}_{s_{W_K}^{t+}} (y_K^t - t^*) = s_k^t B_k^T t^* \\
& - \left(\prod_{i=k+1}^{K-1} s_{W_i}^t \right)^{-1} \left(\prod_{i=k}^{K-1} \|B_i^T t^*\|^2 \right)^{-1} \|s_{W_K}^t\|^{-2} s_{W_K}^{tT} (y_K^t - t^*) (B_k^T t^*) = s_k^t B_k^T t^*.
\end{aligned}$$

By identification, it is found that

$$s_k^t = \frac{-s_{W_K}^{tT} (y_K^t - t^*)}{\left(\prod_{i=k+1}^{K-1} s_{W_i}^t \right) \left(\prod_{i=k}^{K-1} \|B_i^T t^*\|^2 \right) \|s_{W_K}^t\|^2}.$$

Moreover, the update formula for the vector $s_{W_K}^t$ is given by

$$s_{W_K}^{t+1} = s_{W_K}^t - \frac{\eta s_{y_{K-1}}^t}{C} (y_K^t - t^*),$$

where η , C and $s_{y_{K-1}}^t$ are positive scalars. As shown in Section 3.2.2, for any example (x, c^*) in the training set, the error vector $e = (y_K - t^*)$ has a single strictly negative entry $(y_{Kc} - 1)$ at the class label index $c = c^*$, all the other entries y_{Kc} with $c \neq c^*$ being strictly positive. This sign information is constant as the network is trained with a single training example. Given that $s_{W_K}^0 = 0$ from zero-weight initialization and that $s_{W_K}^t$ is updated in a direction opposite to that of e , we have at every discrete time step t

$$\text{sign}(s_{W_K}^t) = -\text{sign}(y_K^t - t^*),$$

and thus

$$-s_{W_K}^{tT} (y_K^t - t^*) > 0.$$

Therefore, the scalars s_k^t are strictly positive for $t > 0$. $\square$

Alignment. In the framework of classification problems, as the coefficients s_k^t are strictly positive scalars for $t > 0$, it results from the theorem that the dot product between the BP and DRTP modulatory signals is strictly positive, i.e.

$$\begin{aligned}
 e^T \left(\prod_{i=k+1}^K W_i^T \right)^T (B_k^T t^*) &> 0 \\
 e^T \underbrace{\left(\prod_{i=k+1}^K W_i^T \right)^T \left(\prod_{i=K}^{k+1} W_i \right)^+}_I \frac{e}{s_k^t} &> 0 \\
 \frac{e^T e}{s_k^t} &> 0.
 \end{aligned}$$

The BP and DRTP modulatory signals are thus within 90° of each other. $\square$

A.3 Supplementary tables and figures

Table A.2 Comparison of the sDFA and DRTP training algorithms on the MNIST dataset, illustrating that DRTP systematically outperforms sDFA. The mean and the standard deviation of the test error over 10 trials are provided. The network definitions and conditions are identical to those of Table 3.1. The learning rates are summarized in Table A.1.

Network		sDFA	DRTP
FC1-500	DO 0.0	4.74±0.15%	4.61±0.13%
	DO 0.1	5.10±0.13%	4.92±0.13%
	DO 0.25	6.06±0.10%	5.75±0.09%
FC1-1000	DO 0.0	4.22±0.11%	4.10±0.07%
	DO 0.1	4.42±0.12%	4.31±0.06%
	DO 0.25	5.23±0.12%	4.94±0.06%
FC2-500	DO 0.0	4.78±0.11%	4.58±0.09%
	DO 0.1	5.16±0.13%	5.00±0.07%
	DO 0.25	6.13±0.10%	5.94±0.06%
FC2-1000	DO 0.0	4.24±0.09%	4.00±0.10%
	DO 0.1	4.51±0.12%	4.25±0.06%
	DO 0.25	5.39±0.05%	5.05±0.09%
CONV (random)	DO 0.0	1.88±0.10%	1.82±0.11%
	DO 0.1	2.17±0.13%	2.06±0.08%
	DO 0.25	2.80±0.17%	2.60±0.14%
CONV (trained)	DO 0.0	1.69±0.10%	1.48±0.15%
	DO 0.1	1.83±0.11%	1.50±0.17%
	DO 0.25	2.20±0.15%	1.81±0.20%

Table A.3 Comparison of the sDFA and DRTP training algorithms on the CIFAR-10 dataset, illustrating that DRTP systematically outperforms sDFA. The mean and the standard deviation of the test error over 10 trials are provided. The network definitions and conditions are identical to those of Table 3.2. The learning rates are summarized in Table A.1.

Network		sDFA	DRTP
FC1-500	DO 0.0	54.80±0.29%	53.92±0.23%
	DO 0.1	54.79±0.24%	53.77±0.17%
	DO 0.25	55.48±0.27%	54.26±0.16%
	DA	53.83±0.32%	52.73±0.31%
FC1-1000	DO 0.0	53.73±0.33%	53.34±0.10%
	DO 0.1	53.92±0.31%	53.15±0.15%
	DO 0.25	54.60±0.38%	53.39±0.15%
	DA	52.95±0.32%	51.87±0.32%
FC2-500	DO 0.0	54.75±0.26%	53.41±0.35%
	DO 0.1	55.35±0.38%	54.06±0.46%
	DO 0.25	55.81±0.37%	54.57±0.33%
	DA	53.85±0.34%	52.54±0.34%
FC2-1000	DO 0.0	53.78±0.24%	52.68±0.25%
	DO 0.1	53.87±0.49%	52.45±0.15%
	DO 0.25	54.87±0.43%	53.29±0.31%
	DA	52.59±0.20%	51.27±0.21%
CONV (random)	DO 0.0	33.08±0.31%	32.65±0.38%
	DO 0.1	33.04±0.42%	32.57±0.34%
	DO 0.25	34.71±0.37%	33.90±0.53%
	DA	31.52±0.25%	31.04±0.45%
CONV (trained)	DO 0.0	38.69±0.78%	35.82±0.59%
	DO 0.1	39.23±0.82%	35.17±0.91%
	DO 0.25	40.08±1.03%	35.51±0.61%
	DA	38.43±0.86%	34.39±0.64%

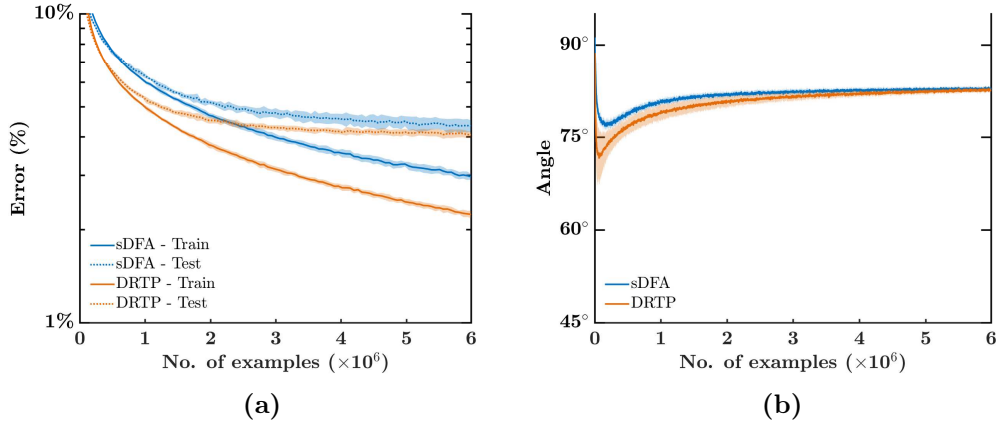


Fig. A.1 DRTP outperforms sDFA on the MNIST dataset. Both figures are with error bars of one standard deviation over 10 runs. The training and test errors are measured after each epoch, while the angle is measured after each minibatch of 60 examples. Both training methods use Adam with a fixed learning rate of 1.5×10^{-4} . **(a)** A 784-1000-10 network with tanh hidden units and sigmoid output units is trained to classify MNIST handwritten digits with the sDFA and DRTP algorithms. On average, the error on the training set reaches 2.97% for sDFA and 2.24% for DRTP, while the error on the test set reaches 4.33% for sDFA and 4.05% for DRTP after 100 epochs. **(b)** While the loss gradients δy_k estimated by both sDFA and DRTP are within 90° of the ones prescribed by BP, the alignment angle is initially better for DRTP than for sDFA. The gap vanishes as the training progresses.

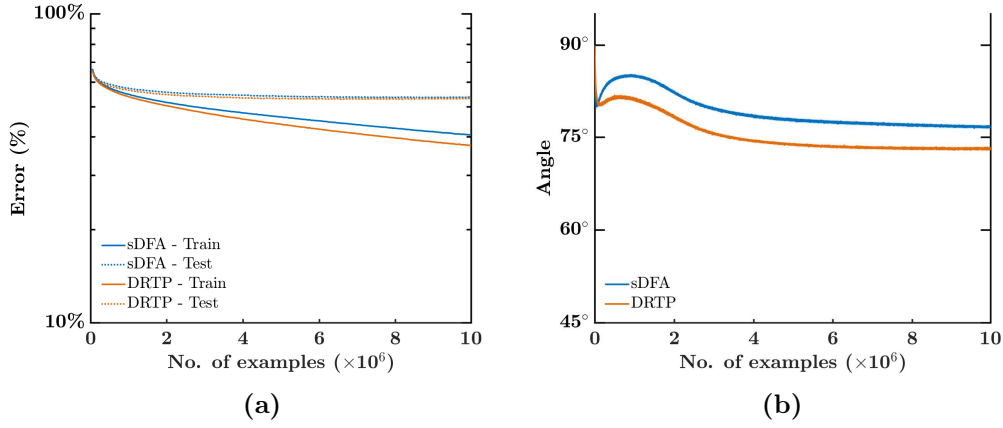


Fig. A.2 DRTP outperforms sDFA on the CIFAR-10 dataset. Both figures are with error bars of one standard deviation over 10 runs. The training and test errors are measured after each epoch, while the angle is measured after each minibatch of 100 examples. Both training methods use Adam with a fixed learning rate of 5×10^{-5} . (a) A 784-1000-10 network with tanh hidden units and sigmoid output units is trained to classify CIFAR-10 images with the sDFA and DRTP algorithms. On average, the error on the training set reaches 40.74% for sDFA and 37.39% for DRTP, while the error on the test set reaches 53.53% for sDFA and 53.12% for DRTP after 200 epochs. No early stopping was applied. (b) While the loss gradients δy_k estimated by both sDFA and DRTP are within 90° of the ones prescribed by BP, the alignment angle is approximately 3.40° better for DRTP than for sDFA.

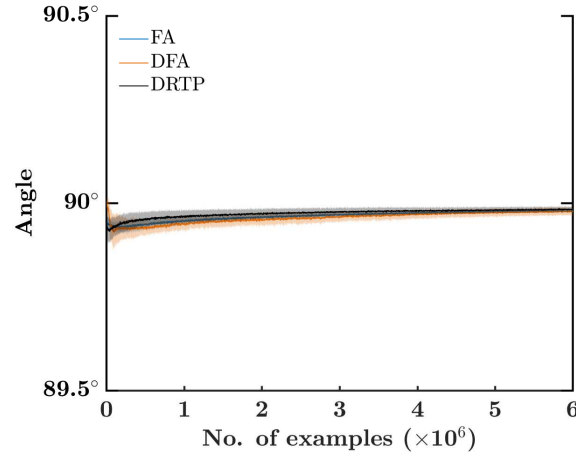


Fig. A.3 Updates to the convolutional layer weights prescribed by feedback-alignment-based algorithms are random due to a 90° -alignment with the BP loss gradients δy_k . A convolutional network is trained on the MNIST dataset with FA, DFA and DRTP. The network topology and training parameters are identical to those used for the trained CONV network in Section 3.2.4.1. Error bars are one standard deviation over 10 runs, the angle is measured after each minibatch of 60 examples. Angles have been smoothed by an exponentially-weighted moving average filter with a momentum coefficient of 0.95.

