New Gener. Comput. (2017) 35:253–269
DOI 10.1007/s00354-017-0020-4

TUTORIAL

The End of Moore's Law: Opportunities for Natural Computing?

Ferdinand Peper[1]

Received: 15 April 2017 / Accepted: 25 May 2017 / Published online: 29 June 2017

Abstract The impending end of Moore's Law has started a rethinking of the way computers are built and computation is done. This paper discusses two directions that are currently attracting much attention as future computation paradigms: the merging of logic and memory, and brain-inspired computing. Natural computing has been known for its innovative methods to conduct computation, and as such may play an important role in the shaping of the post-Moore era.

Keywords Moore's Law · Memory-based computing · Cellular Automata · Brain-inspired computing · Neural networks · Natural computing

Introduction

There is an old joke about Moore's Law stating that the number of people predicting its end doubles every 2 years [8]. Though this has not kept it from continuing unabated for more than half a century, recent developments indicate significant changes in integrated circuit industry in the coming decades. One such development is the end in 2016 of the International Technology Roadmap of Semiconductors (ITRS), which has provided biannual updates on the technology status and expected developments in the semiconductor industry since the 1990s, and has acted as a treadmill to keep the various players in the industry synchronized with each other and with Moore's Law. There is less incentive for such coordination nowadays, due to a slowing down of technology generations in recent years, witness, for example, Intel's repeated postponement of its 10 nm technology. Though ITRS has touched upon alternative devices and architectures, it has been firmly in the camp of silicon

✉ Ferdinand Peper
peper@nict.go.jp

[1] Center for Information and Neural Networks, National Institute of Information and Communications Technology, Osaka, Japan

technology, which is no surprise given this technology's extraordinary success. With the end of ITRS, new initiatives like IEEE Rebooting Computing have started, accompanied by the launch of the International Roadmap for Devices and Systems (IRDS), emphasizing novel architectures, devices, and applications.

Though all indicators are that silicon-based microelectronics will be around for the foreseeable future, there is an increasing realization that novel materials, designs, and principles of computation will play a major role in future technology. Even if these alternatives will make up only a small part of computers at first, they will require a rethinking in how information is processed. Questions on how to overcome the Von Neumann bottleneck, for example, have already led to proposals for architectures in which memory and logic are combined on much finer scales than was previously thought practical, and there is an increasing realization as well that future architectures need to incorporate at least some of the functionality of human brains.

Computing technology is moving into a direction in which functionality has priority over physical specifications like clock speed or device feature sizes. When Dennard's scaling rules [10] became unsustainable around 2004, there was a sudden shift away from the GHz war between manufacturers. A similar phenomenon has occurred recently with advertised on-chip feature sizes, witness Intel's statement[1] that there will be more focus on performance improvements rather than the underlying technology node [56]. Such improvements will not necessarily be quantitative, e.g., more Floating Point Operations per Second (FLOPS), but rather will emphasize energy efficiency (FLOPS per Watt), or quality in terms of how well applications will actually be served.

In this paper, we discuss two trends in information processing that have attracted attention for a long time, but have particular appeal in the current era of new materials, devices, architectures, and computational challenges. The merging of logic and memory has been contemplated with the removal of Von Neumann's bottleneck in mind, and the obvious solution is to bring the processing of information closer to where it is stored. Cellular Automata are a well-known model in this paradigm, and they have particular appeal for their high degree of fine-grained parallelism. Architectures for brain-inspired computing tend to require similar degrees of parallelism, but they differ from Cellular Automata by their great degree of freedom with regard to structure: whereas Cellular Automata are extremely regular, brain-inspired computing tends to rely more on random connectivity. The unparalleled performance of the brain in processing complex information at an extremely meager energy budget has greatly inspired generations of scientists, and we have now arrived in an era where this research has come to bear fruits.

This paper is organized as follows. "A Brief History of Moore's Law" places Moore's Law in a historic perspective to assist the reader in judging its future. "Merging of Logic and Memory" discusses the merging of logic and memory with an emphasis on Cellular Automata. This is followed by a discussion of brain-

[1] Statement by Venkata Renduchintala, president of Intel's Client and Internet of Things businesses and its Systems Architecture Group.

inspired computing in "Brain-Inspired Computing", in which the major trends in brain science are discussed, as well as their possible implications on algorithms and architectures. This paper finishes with a discussion of natural computing in "And Natural Computing?", giving an outlook on the post-Moore era.

A Brief History of Moore's Law

Before the advent of Moore's Law, integrated circuits were a new technology that, due to its low yields, was hardly recognized as competitive with the then more common technology based on discrete components. Fairchild Semiconductor—the young company Moore was part of—was able to cram 64 transistors on an integrated circuit, albeit at a higher per-component price than that of single components sold at the time. It was in this context that Moore formulated his now-famous law in 1965 [45], and his prediction may very well have been rooted in a desire to increase interest among potential customers for this new technology.

The 1965-prediction that the number of components on a chip would double every year could be seen in this light, but fast forward 1975 it turned out to be quite accurate, being only a factor of two below the predicted 65,000 components. Nevertheless, in that same year Moore saw reason to revise his prediction downward to a doubling of the number of components every 2 years [46]. To understand why, let's briefly examine the factors that contributed to the increase in integration density in the first decade of Moore's Law. First, there is the obvious factor of device size reduction, which is commonly attributed as key to Moore's Law. The second factor, increasing chip area, is less obvious, but it contributed almost as much as the first. The third factor is device and circuit cleverness: in the first decade there was still plenty of room to pack devices closer together through more efficient layouts, but Moore concluded that this factor was soon to run out of steam. Reasoning that the third factor accounted for almost as much improvement as the other two factors together, Moore revised his law in 1975 downward to a biannual doubling of chip component counts.

Moore understood from the beginning the importance of economics in his law. Rather than focusing on the maximum or average number of components that could be placed on a chip, he used the number for which the cost per component was at a minimum [41]. Merely adding components would increase the cost per component, because of the decreased yield of chips caused by a more than proportional occurrence of defects. So, Moore's Law is not just about doubling of the number of components per chip in a certain time interval, but importantly it talks about the most economical way to do so. This sweet spot of minimum cost per component identified by Moore is still as valid today as it was 50 years ago.

From 1975 on it has become clear that the rate of integration differs by the type of chip: the biannual doubling of components applies to logic chips, such as microprocessors, but memory chips tend to double their number of transistors in a shorter period (18 months) due to their more regular structure, which offers advantages in design and layout.

Until around 2004 Moore's Law ran in conjunction with Dennard's scaling rules [10], which provided a predictable scaling of design parameters. According to these rules, scaling by a factor λ (>1) reduces device dimensions by a factor $1/\lambda$, and this allows the voltage, current, capacitance, and delay time of a transistor to decrease by the same factor $1/\lambda$ if the doping concentration is increased by the factor λ. Since the power requirement of the device then reduces by the factor $1/\lambda^2$, and the number of devices that can be placed on the same area increases by roughly the factor λ^2, the power dissipation per unit area would stay constant between generations. Dennard's scaling rules had provided a free lunch for engineers for decades, as the decreased transistor sizes delivered by Moore's Law automatically led to better performance in terms of speed and power consumption. An early sign that Moore's law was in its final stages came with the end of Dennard's scaling rules, when it became more difficult to decrease voltage at the same high pace due to increased leakage currents in devices. Thus came to an end the race between manufacturers for increasing clock speeds, only to be replaced by a race to manufacture chips by the most advanced technology node. With Dennard's scaling finished, the main benefits of shrinking transistors today are more functions per chip and a lower cost per function. Though costs of equipment, material, and processing to manufacture chips has increased exponentially over the years, the cost per square centimeter of silicon has hardly changed over time [41]. We have now arrived in an era where more transistors are imprinted on silicon each year than characters are printed on paper, and the cost to do so is lower per transistor than per character.

The initial effect of Moore's Law was a scaling up of chips into more powerful ones at the same price every couple of years—also called Moore's Law 1.0 by Mack [41]. Nowadays this version of Moore's Law governs only high-end chips, which include microprocessors for supercomputers, high-end Field Programmable Gate Arrays (FPGAs), and General Purpose Graphical Processing Units (GPGPUs) [41]. The billions of transistors available on these chips are overkill for most of the applications on the market, so it has become more common to deliver similar performance at a lower price in each generation; this scaling down of chips is also called Moore's Law 2.0 [41]. Up to 2010 there was optimism about the future of Moore's Law, and it was even predicted by some to accelerate, but this mood has gradually darkened when it became clear that the advertised feature sizes according to which technology generations were defined started to bear less and less resemblance to the on-chip feature sizes that could actually be achieved.

The efforts towards ever-higher integration densities was labeled More Moore in the ITRS report of 2011 [55]. The real interest in the 2011 report, however, was in the various other categories it introduced. Beyond CMOS, represents the research efforts towards novel devices and architectures. Though reaching back for decades, these efforts have attracted increased interest with the realization that More Moore can deliver only diminishing returns in the future. More than Moore, a category that is called Moore's Law 3.0 by Mack [41], indicates the trend to include different functionalities on chips, like sensors and RF devices, and looks at systems as a whole and their requirements for applications.

In conjunction with these trends, the question has been raised whether the Von Neumann architecture is still as relevant today as it was at its birth in 1945. The concept of storing programs in memory was conceived at a time in which a computer like the ENIAC could be "programmed" only through a complex process in which a problem was mapped onto hardware by setting switches and plugging in wires in appropriate ways. The ENIAC was in fact a primitive form of reconfigurable hardware, but its relevance was not fully appreciated at the time due to its tedious and time-consuming programming process. The last decade has seen a diversification of architectures, with elements of FPGAs and GPGPUs playing an increasing dominant role in them. The diversification in applications, away from computers to communication devices has had a significant impact on industry, where low power consumption has become the driving force behind designs. Combinations of various hardware solutions with more traditional von Neumann architectures have also become increasingly commonplace. This hybridization of architectures may be an important step in the search for alternatives to the Von Neumann architecture.

Merging of Logic and Memory

A direct consequence of the Von Neumann architecture is the intensive traffic between the central processing unit (CPU) and memory. The resulting bottleneck has grown worse over the years as both the size of memory and the speed of CPUs have radically outgrown the bandwidth between the two. Attempts to ameliorate the Von Neumann bottleneck have led to many innovations in hardware, like caching, memory access optimization, branch prediction, and multi-threading. Many of these solutions could be considered Processing in Memory (PIM) [38] in one form or the other, since they primarily aim at temporarily storing data and instructions in memory that is inside the CPU, rather than accessing the main memory.

The Von Neumann bottleneck is not only a matter of hardware organization, it is also considered a perceptual bottleneck in the sense that most programmers are trained in using programming languages that inherently assume a division between memory and processing, as argued by Backus in his Turing Award Lecture [1]. Parallel computing is one of the proposed solutions to deal with the Von Neumann bottleneck, since it allows memory to be brought closer to processing resources, but one should be careful not to introduce a separate Von Neumann bottleneck for each parallel processor employed. When we follow a very fine-grained model of parallel processing, we arrive at models in which small chunks of logic are equipped with limited amounts of memory. The most extreme form of distributing memory over processing resources are Cellular Automata, to which, ironically, Von Neumann also got his name associated with.

A Cellular Automaton (CA) is a regular array of cells, each of which is a finite automaton that has as input the state of the cell itself and the states of the neighbors. Depending on those states, the cell's next state is determined according to a set of transition rules. The neighborhood of a cell is often chosen to be the four cells directly orthogonally adjacent to it (Von Neumann neighborhood), but other

neighborhoods have also been investigated, including neighborhoods with additionally the diagonal neighbors (Moore neighborhood), and neighborhoods including cells up to a distance larger than 1.

The functionality of a CA is determined by its transition rules. Depending on the design, a CA may be specialized in certain operations, or it may be capable of universal computation. The latter can be accomplished through various strategies. One strategy is to simulate a universal Turing Machine on a CA, another is to simulate a logic circuit on a CA and show that such a circuit can be constructed for any computational function. Since Turing Machines are notorious for their inefficiency, simulating them on a CA is not considered a good way to use the parallel processing ability of CAs. This is a reason why simulation of logic circuits on CA is preferable in hardware implementations.

Most CA models are synchronously timed, like the well-known Game of Life [18], meaning that the updates of all cells take place simultaneously. Synchronous updating requires the distribution of a clock, as well as sufficient memory in each cell to store its current state as well as its next state. An alternative, asynchronous, timing model has been explored in [49, 50] that allows cells to operate more independently from each other, thus emphasizing the locality of their interactions. In an asynchronous CA, each cell may undergo a transition with a certain probability, but only so if the cell's state and the states of its neighbors match the left-hand side of a transition rule. Absent such a match to any transition rule, no transition will take place for that cell. In this respect, asynchronous CAs are quite different from synchronous CAs, which update all cells in each step.

The timing of CAs carries over to the logic circuits they simulate. Asynchronous CAs are typically designed such that they support so-called Delay-Insensitive (DI) circuits [29]. In DI-circuits signals are allowed to have arbitrary delays in time, which means that variations in the arrival times of signals at gates do not affect the correctness of operations. For conventional synchronously timed Boolean circuits the situation is quite different: a delay of a signal by a single clock cycle to the input of a logic gate will often dramatically affect the gate's output. Conventional logic gates are not universal for the class of DI-circuits, because they do not implement timing functionality, and as a result they are unsuitable as the basis for asynchronous CAs. DI-circuits tend to use a set of logic primitives that includes at least one gate that is able to "synchronize" input signals. Synchronization by a logic element in the context of DI-circuits means that the element will only produce output if all of its input wires have signals on them. If at least one input signal is absent, the element will keep all other input signals pending until all missing input signals are provided.

Asynchronous CAs tend to employ DI-circuits in which signals take the form of tokens. A well-known example of a token-based DI-circuit is a Petri-net [47]. A token in a CA is usually implemented as an uninterrupted finite linear area of cells that are in the same state. This area may stretch and shrink due to the randomness of the asynchronous timing of the CA, but the probability that it has its minimum size—which is usually one cell—in a certain time interval will not change over time. Designs of DI-circuits, and especially token-based circuits, are different from conventional Boolean circuits, because different logic primitives are used. Typically

a dual-rail representation is employed, in which each logic level is assigned its own wire and a signal is represented through the presence of a token on the wire corresponding with the logic value. A dual-rail representation makes certain operations less complex: a NOT-gate, for example, can be realized by simply crossing a 0-wire and a 1-wire. However, most other operations tend to become more complex, though this also depends on the particular choice of the logic primitives.

The complexity of cells is an important issue in CAs, especially when efficient implementation in hardware is required. Cell complexity is determined by the number of states a cell can assume, as well as the number of transition rules describing its functionality. Even within that framework there may be a difference in complexity due to the size of the neighborhood in a CA: a bigger neighborhood tends to result in transition rules with higher complexity. On the other hand, a bigger neighborhood may require less transition rules to specify certain functionality. Designers usually favor a smaller neighborhood in CA, because complexity tends to be less in this case when taking all factors in account. The number of cell states and rules required in asynchronous CA models based on token-based DI-circuits tend to be of the same order as their synchronous counterparts. Further reductions in the number of states and rules have been achieved by allowing signals in asynchronous CAs to fluctuate, and make them search their way from input to output in a random search process; the resulting models are called Brownian Cellular Automata [35]. The use of fluctuations may be relevant in a nano-electronic and biological context, since noise plays a significant role on scales of single particles and molecules.

Complexity of cells is also strongly dependent on the granularity of a CA. On one side of the spectrum is a fine granularity, in which a cell has very limited functionality. Since most cells in a CA simulating a logic circuit are used for wires or for unused spaces between wires, it is a waste of resources to include lots of functionality in cells. In its most extreme form of fine granularity, multiple cells will be needed to simulate a single logic gate. Coarse granularity (the other side of the spectrum) uses at most one cell per logic gate, and usually multiple logic gates can be simulated by a single cell [13]. In this case fewer cells will be required, but if a cell cannot be fully used due to it being deployed as a wire or space between wires, there is significant overhead.

It tends to be difficult for CAs to achieve the level of functional density of contemporary CPUs, but in the end the regularity of CAs may allow the use of bottom-up manufacturing methods that can achieve much higher integration densities than those possible with the top-down methods employed for conventional VLSI. This era has not yet arrived, and an important reason for the relatively low functional densities of CAs is that the transition rules of CAs need to be stored in each cell to make the model truly local, thus leading to much overhead. This motivates research into finding physical or biological materials in which rules are naturally implemented through physical interactions.

Materials that implement CAs in a natural way are hard to find in nature, though some examples approaching computation ability have been demonstrated [2, 23]. A problem with such rules is that physical systems tend to converge to a state with minimal potential energy, so it is difficult to continue a computation for an extended

time in such systems. Rather, a one-time computation can be achieved, in which the elements used in the computation stabilize into a lower-energy state, and need to be reset afterwards [23]. Alternatively, energy needs to be fed into the system in order to sustain a computation [5], but as of today, such CA-based systems have not been found in nature.

Minimization of the number of rules has resulted in computation-universal CAs with only three rules [35], and recently even further improvements have been made [48]. Nevertheless, it would be of great practical significance if rule-less CAs could be designed. Such models have actually been considered in the history of computer science. Called Microcellular Arrays, these models consist of an array of cells, each of which can be configured as either a logic gate or as a wiring pattern, depending on the contents of a memory of a few bits associated with the cell. Cells in microcellular arrays are organized in a uniform structure with neighbors connected to each other by fixed wires. Microcellular logic was first proposed by Minnick in 1964 (see also [44]) as a way to integrate many logic devices on a single wafer without having to dice the wafer into separate integrated circuit components to be assembled into packages. The approach was one of the first attempts to realize reconfigurable logic, but it was largely unsuccessful because it was quickly overtaken by the commercial successes of the early microprocessors. This was also the era in which interconnect had a huge advantage over logic in terms of area, speed, and energy consumption, so it was considered not very cost-effective to use cells with logic in them just for the purpose of wiring. These assumptions are not valid anymore in modern VLSI, where interconnect causes more delay in signals than logic does, and it has started to compare unfavorable in terms of area and energy consumption too. These problems may merit a reconsideration in favor of the cellular approach.

Brain-Inspired Computing

Neurons come in many forms and functionalities and form a complex network that gives the brain its remarkable abilities. It is, therefore, no surprise that research to make hardware mimicking the brain has a long history (see for example [57] for a review). Rather than giving a comprehensive overview, we discuss the principal differences with conventional computers and the future potential of brain-inspired computing in this section.

The number of neurons in the human brain (100 billion) is expected[2] to be surpassed by the number of transistors on a chip by 2026 [24], but even if this extrapolation of Moore's Law turns out to be correct, it does not necessarily mean that a chip will have the same abilities as the brain. A neuron is much more complex than a transistor, having an intricate dynamics that depends on many parameters [7], including whether the neuron is inhibitory or excitatory, whether an action potential is generated from the neuron's soma or from its axon initial segment, what its topological features are such as the location of the axon initial segment and the

[2] According to Intel's senior vice-president Mooly Eden in Intel's CES 2014 keynote.

spatial structure of dendrites, and what the history is of the neuron's firing. There is a variety of operations that can be conducted by single neurons, including addition, subtraction, multiplication, division, filtering (high-pass or low-pass), and toggle switching [25].

Connections in the brain are very different from connections in computers. Axons, which form an important part of the connections in the brain, have a complex dynamics in themselves. Though they are traditionally thought of as the transmission cables along which the impulses—called spikes—fired by neurons propagate, they are increasingly recognized as having a functional and computational repertoire that is much richer [9]. Geometric properties of axons have been shown to play an important role in synaptic efficacy and neuronal timing.

An important function of connections in the brain is the storage of knowledge. The main mechanism for storage is through synapses, which form the interfaces between axons and dendrites of neurons. The strength of a synapse determines the degree to which signals pass through, and by adjusting synaptic strengths according to conditions surrounding synapses, a neural network can learn new knowledge. Whereas the huge number of connections in the human brain, estimated to be between 10^{14} and 10^{15}, would be considered a significant overhead in traditional computers, in the brain they represent the main functionality.

The mechanisms via which learning takes place are intimately tied to the representation of signals between neurons. While it has been known for an extended time that most neurons use spikes as signals, it is still an open question how spikes are exactly used to encode information [6]. An early model assumed a so-called rate encoding, in which the average number of spikes fired by a neuron in a certain time window defines an analog value that is considered the output of the neuron. Rate encoding has long been used in artificial neural networks, and it is compatible with Hebb's learning rule, which states that "neurons that fire together wire together". Hebb's rule follows the principle that if a pre-synaptic neuron and a post-synaptic neuron both fire at high rates, the synapse in between will be strengthened. While Hebb did not clearly describe when a synaptic strength will be decreased, it is commonly assumed that this will happen in all cases when neurons do not "fire together". Many artificial neural networks use some form of Hebb's rule to learn new knowledge.

There are compelling reasons to discount rate encoding as the only mechanism used to represent information in signals in the brain. Since a single spike takes approximately 1 ms of time, and a neuron's refractory period—during which it cannot fire—takes at least the same time, there will be at most 50 spikes fired by a neuron in the minimal time of 100 ms it takes for a human to react to a stimulus, but this means that a statistically significant averaging over a time window allows no more than a few time windows in this 100 ms interval. In other words the processing in the brain required to bring about a reaction to a stimulus would consist of only a few stages if rate encoding is employed, but this small number of stages is considered too meager to conduct useful cognitive processing. This indicates that spikes are used in another way in the brain, i.e., through their timings, though this does not completely exclude rate encoding. Using the timing of spikes opens a

whole new spectrum of possibilities to encode signals, because spikes can be timed over a continuous temporal dimension with resolutions well below the typical 1 ms width of a spike. Evidence suggests that spike timing is used in the motor cortex [53], the primary visual cortex [52], and mammalian olfactory systems [26].

If information can be coded through the timing of spikes, why would there be a need for rate encoding? The answer to this is speculative, but a higher rate of firing in a group of neurons may somehow be related to where attention of the mind goes. A more systematic answer to this question is offered in [39], which shows, based on data from various studies, that spikes appear to occur in temporal groups of a few hundred milliseconds called packets, such that at the beginning of a packet, encoding based on spike times is more important, but later in a packet rate encoding is more prominent. In other words, though both encodings are likely to be used at any time, their relative importance appears to change over the course of a packet. It has been argued by Szymanski [59] that the timing of the spike arriving first at a set of neurons may represent important information, because the first spike will more likely be associated with the stimulus that is most significant to the organism receiving it. This resembles the well-known winner-takes-all principle in competitive learning, whereby the winner is the neuron firing the first spike.

Hebb's rule as described earlier is no longer sufficient to describe learning when spike timing is used. More relevant in this context is so-called Spike-Timing-Dependent Plasticity (STDP) [42], according to which a synaptic strength is adjusted based on the relative timing of the spikes output by a pre-synaptic neuron and a post-synaptic neuron. Simply put, when a pre-synaptic spike occurs immediately before a post-synaptic spike, the synapse will typically be strengthened in a process called Long-Term Potentiation (LTP), but if the order of the spikes is reversed, weakening of the synapse will ensue [Long-Term Depression (LTD)]. In other words, when there is a causal relationship between the firings of the pre-synaptic neuron and the post-synaptic neuron, LTP will take place, whereas an anti-causal relationship results in LTD.

If timing of spikes is so important, how can it be manipulated? An obvious mechanism is synaptic strength: since a spike from a pre-synaptic neuron will add more to a post-synaptic neuron's action potential if the corresponding synapse is strong, the timing of spikes from the post-synaptic neuron will on average be earlier in this case. This effect, however, does not give precise control over spike timing, so there is speculation that there is another mechanism that can manipulate spike timing more directly. Delays of spikes along axons have been reported to depend on the characteristics of a sheath around axons that consists of myelin, which is an electrically insulating white fatty substance [16] making up part of the brain's white matter. The myelin sheath is periodically interrupted by nodes of Ranvier, and these serve as points between which electrical charges jump, thus propagate signals along axons. The velocity by which this propagation takes place increases with the thickness of the myelin sheath, up to a point that the myelin is so thick that it starts to cover the nodes of Ranvier. Not only does myelin around axons influence the delays of spikes, it has also been reported to change over the course of weeks when experimental subjects undergo a training process [15, 19], leading to speculation that learning processes affect not merely neurons and their synapses, but also white

matter. To date no satisfactory model has been formulated that can explain, or even describe, these processes in sufficient detail.

It has been pointed out [27] that the use of spike timing in combination with delays by which spikes propagate between neurons has the potential for a highly complex dynamical behavior, in which groups of neurons fire together in intricate patterns. Called polychronization, this model assumes different delays between different pairs of pre- and post-synaptic neurons, and this allows a different post-synaptic neuron to become potentiated to fire a spike, depending on the timing of the spikes it receives from the pre-synaptic neurons. In other words, even if the same pre-synaptic neurons fire, a different post-synaptic neuron could fire in response, depending on the order and timing in which the spikes from the pre-synaptic neurons are fired. The amount of information that can be represented by the huge number of combinations in which polychronous groups can occur is overwhelming, but this resource in our brain has hardly been exploited in artificial neural network models.

In the early days of Turing and colleagues computing was assumed to be organized according to a batch model in which input and a program is offered to a computer, which then processes the input according to the program, and finally produces output. This view of computing has traditionally pervaded the artificial neural network community too: the hierarchical organization of the brain is generally thought of as defining the structure in which processing take place, i.e., from the bottom up to the highest layer, stage by stage. There is evidence, however, that supports a more dynamic view of the brain, in which autonomous processes take place even if there are no input stimuli. In this framework, input merely serves to modulate the autonomous activity taking place in the brain [11]. An artificial neural network with similar behavior is the Liquid State Machine [40]. It consists of a "reservoir" of neurons that are randomly connected to each other, as well as input neurons that are randomly connected to the neurons in the reservoir, and output neurons that are randomly connected to the neurons in the reservoir as well. The reservoir has an internal dynamics of its own, but this dynamics is modulated by signals from the input neurons. All connections have random fixed weights, except the connections to the output neurons, which can be trained according to desired input-output combinations. This training is relatively fast compared to multilayer neural networks, but it tends to be difficult to control the dynamics in the reservoir. Though there is a considerable gap between the liquid state machine and brains, the model sheds some light on the dynamics that likely play a role in the brain. Interestingly, computers have developed over the last decades to include humans in the loop of information processing in many applications. For example in gaming and networking a human typically interacts with a computer on a continuous basis, whereby the computer runs an autonomous process modulated by input from a human. It has been argued that results on computability in the traditional batch processing paradigm may be less relevant to this new framework.

Whereas the architectures of early computers in the 1940s represented a serial mode of processing, the artificial neural networks proposed in the same era initiated a more parallel distributed approach. The history of neural networks is quite tumultuous, with periods of frenzied excitement followed by "winters" in which

interest in the field and funding was minimal (see [32] for a well-written non-technical overview). Artificial neural networks have long been perceived as performing poorly in the machine learning community, but that started to change when improved learning algorithms, some relabeled as Deep Learning because of the many layers of neurons they use, were combined with fast GPGPU hardware and big training sets. This combination allowed dramatic improvements in image recognition, speech recognition, natural language understanding, and the analysis of data on drug molecule activity, particle accelerator data, and gene expression and disease [34]. Noticeable in the success of neural networks is that though they were initially inspired by the workings of the brain, in the end it was Moore's Law that facilitated their performance improvements.

Learning in artificial neural networks is characterized by a gradual update of the connection weights between neurons each time a sample of the training data is presented. This mode is also called incremental learning, and it tends to use a trial-and-error strategy and to require large numbers of training samples to adequately train a network. The sensory systems of many organisms are constantly bombarded by stimuli from a very early age, so it can be argued that they do not differ much in this respect from their artificial counterparts. When it comes to higher functionality, however, learning tends to proceed on a much faster scale. Humans, for example, are able to learn new knowledge through exposure of merely one or a few training samples. Called one-shot learning, this mode of learning takes advantage of knowledge learned at an earlier stage, and humans are able to use this knowledge to quickly learn new information [14], even if it differs substantially from knowledge they already have. There is evidence that humans quickly switch to a mode of one-shot learning from incremental learning if there is a weak causal relationship between stimulus and outcome [37]. Such a weak causality indicates novelty of information, and thus encourages the brain to employ knowledge about past experiences in its learning of the new information. Artificial neural networks, on the other hand, face difficulties in this task, because new information will interfere with previously learned knowledge [30]. Though deep learning excels in finding intricate structures in big data sets, it does less well than the brain when only small quantities of training data are available. The need to quickly learn new information based on only few data samples has recently led to proposals in which a memory is added to a neural network such that no retraining is required; rather the augmented memory retains old information while allowing quick encoding and retrieval of new information [54]. Though such models carry names like Neural Turing Machine [20] and Differentiable Neural Computer [21], suggesting that they are nothing more than their non-neural counterparts attached to neural networks, they differ in that they can be trained by gradient-based methods, just like most artificial neural networks. These models, however, do not truly integrate and combine data, and it is unclear how well they perform on data sets containing a wide variety of information organized in categories that may partially overlap each other. Models based on polychrony [58] may have better potential in this respect: in such models memories are represented by overlapping groups of neurons that fire in time-locked patterns defining polychronous groups. Though a neuron may be part of multiple groups, the

groups are relatively independent, which may facilitate the encoding of new memories without disturbing old ones.

Key to these models is the use of spike timing. The GPGPU hardware typically used for deep neural networks is not optimized to handle individual spikes, and neither does it excel in terms of low power consumption. Computers designed for running simulations of neural systems, on the other hand, generally have a mechanism to handle spikes. IBM's TrueNorth [43], the Neurogrid [4], and SpiNNaker [17], for example, all employ an Address Event Representation (AER) scheme [33], in which spikes between arbitrary neurons are provided with an address to which they can be routed via a bus, as well as with other information, like their firing time and the delay between the pre-synaptic and post-synaptic neurons. These computers, however, adopt a Von Neumann architecture as their underlying design, and can thus not be called genuinely neuromorphic. The future will likely bring us architectures that can efficiently handle neural models in which spike timing plays a central role, but these architectures will need to adopt a high degree of localized configurability to accomodate the flexible structure of biological neural systems. It has been argued [22] that alternative materials and devices that can implement neuron-like functionalities in a natural way may also play an important role in delivering efficient brain-inspired architectures.

And Natural Computing?

Cellular Automata and Brain-like computing are among the first examples of computing systems that were inspired by nature [28], and it did not take long before they were joined by a diversity of models like evolutionary algorithms, swarm-based models, artificial immune systems, artificial life, and DNA computing. The practitioners of these natural computing paradigms tend to share a belief that nature hosts a myriad of information processing mechanisms. These mechanisms may occur at many levels. An ignoble example of natural computing is the spaghetti sorter of Dewdney [12], which consists of merely a handful of uncooked spaghetti sticks of different lengths of which the longest can be selected in constant time by placing all of them with one end down on a table. Though quite trivial, this model actually resembles the timing mechanism in the neural systems mentioned in "Brain-Inspired Computing", whereby the dimension of time replaces the length of a spaghetti stick, and the neuron that fires a spike first is declared winner [59].

Signals are represented in digital computers in a binary fashion by a high or low voltage, and they are usually controlled by a central clock, but in many natural computing paradigms this is no longer true. Brain-inspired architectures and dynamical systems generally use analog values, whereas quantum computing [3] and noise-based computing [31] use superpositions of binary values. Bio- and nano-systems, on the other hand, tend to use signals that have a discrete character. This may take the form of molecular cascades [23] that behave like a set of domino stones triggering each others' transitions into lower energy states. Signals may also behave like discrete particles, like the tokens used in Petri nets. When signals consist not of single particles, but 10's or 100's of them, like in bio-systems, in

which molecules are typically available in relatively low concentrations, we encounter new challenges. Such meso-scale systems tend to fall in between the paradigms employing voltage-encoded signals and token-based signals, and whereas circuit theory is available for each of these, the computational framework for meso-scale systems still has to be developed.

Timing of signals may also be radically different in natural computing systems. While asynchronous architectures lack a clock, and thus effectively use an analog value in the temporal dimension, neuromorphic computing tends to go even further and uses spiking signals that are not only timed asynchronously, but also synchronize the spikes among groups of neurons. Bio- and nano-systems are typically governed by probabilistic interactions, and for this reason alone they are best timed asynchronously, because it is difficult to guarantee that interactions take place within the interval defined by a clock. Brownian circuits [36, 51] have been developed with probabilistic interactions in mind, but they are mostly geared towards signals consisting of single particles.

Implicitly assumed in this paper—and indeed in many discussions about a post-Moore era—is that the ultimate goal is to squeeze as much performance in terms of operations per second or per Watt out of devices, circuits, and architectures, and though this assumption makes perfect sense for designing computing systems inspired by nature, it is less so when the purpose is to employ materials from nature to realize computation. This is especially true when such systems are intended for use in a biological environment, like the human body. MFLOPS do not mean much inside the body, and they are likely to be harmful because of the high energy consumption and heat dissipation that usually accompany them. Engineered systems operating in the body, like molecular robots, need to be bio-compatible, and in order for them to be able to operate autonomously, they need to employ control algorithms that effectively utilize the characteristics of their environment, while requiring only very limited input of instructions and energy from the outside. Nature offers fascinating opportunities to realize information processing systems that have hardly been touched upon by computer science. The end of Moore's Law will likely be credited with highlighting these opportunities.

Acknowledgements This work was supported by JSPS KAKENHI Grant Number JP16H01719.

References

1. Backus, J.: Can programming be liberated from the Von Neumann style?: A functional style and its algebra of programs. Commun. ACM **21**(8), 613–641 (1978)
2. Bandyopadhyay, A., Pati, R., Sahu, S., Peper, F., Fujita, D.: Massively parallel computing on an organic molecular layer. Nat. Phys. **6**(5), 369–375 (2010)

3. Benioff, P.: The computer as a physical system: a microscopic quantum mechanical hamiltonian model of computers as represented by Turing machines. J. Stat. Phys. **22**(5), 563–591 (1980)
4. Benjamin, B.V., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A.R., Bussat, J.M., Alvarez-Icaza, R., Arthur, J.V., Merolla, P.A., Boahen, K.: Neurogrid: a mixed-analog-digital multichip system for large-scale neural simulations. Proc. IEEE **102**(5), 699–716 (2014)
5. Biafore, M.: Cellular automata for nanometer-scale computation. Phys. D Nonlinear Phenom. **70**(4), 415–433 (1994)
6. Brette, R.: Philosophy of the spike: rate-based vs. spike-based theories of the brain. Front. Syst. Neurosci. **9**, 151 (2015)
7. Brunel, N., Hakim, V., Richardson, M.J.: Single neuron dynamics and computation. Curr. Opin. Neurobiol. **25**, 149–155 (2014)
8. Cross, T.: After Moore's law: double, double, toil and trouble. The Economist, Technology Quarterly, vol. 1. http://www.economist.com/technology-quarterly/2016-03-12/after-moores-law (2016). Accessed 9 June 2017
9. Debanne, D., Campanac, E., Bialowas, A., Carlier, E., Alcaraz, G.: Axon physiology. Physiol. Rev. **91**(2), 555–602 (2011)
10. Dennard, R.H., Gaensslen, F.H., Rideout, V.L., Bassous, E., LeBlanc, A.R.: Design of ion-implanted MOSFET's with very small physical dimensions. IEEE J. Solid State Circuits **9**(5), 256–268 (1974)
11. Destexhe, A.: Intracellular and computational evidence for a dominant role of internal network activity in cortical computations. Curr. Opin. Neurobiol. **21**(5), 717–725 (2011)
12. Dewdney, A.K.: Computer recreations: on the spaghetti computer and other analog gadgets for problem solving. Sci. Am. **250**(6), 15–19 (1984)
13. Durbeck, L.J.K., Macias, N.J.: The cell matrix: an architecture for nanocomputing. Nanotechnology **12**(3), 217 (2001)
14. Fei-Fei, L., Fergus, R., Perona, P.: One-shot learning of object categories. IEEE Trans. Pattern Anal. Mach. Intell. **28**(4), 594–611 (2006)
15. Fields, R.D.: Change in the brain's white matter. Science **330**(6005), 768–769 (2010)
16. Fields, R.D.: A new mechanism of nervous system plasticity: activity-dependent myelination. Nat. Rev. Neurosci. **16**(12), 756–767 (2015)
17. Furber, S.B., Galluppi, F., Temple, S., Plana, L.A.: The SpiNNaker project. Proc. IEEE **102**(5), 652–665 (2014)
18. Gardner, M.: Mathematical games: the fantastic combinations of John Conway's new solitaire game "life". Sci. Am. **223**(4), 120–123 (1970)
19. Gibson, E.M., Purger, D., Mount, C.W., Goldstein, A.K., Lin, G.L., Wood, L.S., Inema, I., Miller, S.E., Bieri, G., Zuchero, J.B., et al.: Neuronal activity promotes oligodendrogenesis and adaptive myelination in the mammalian brain. Science **344**(6183), 1252304 (2014)
20. Graves, A., Wayne, G., Danihelka, I.: Neural Turing machines. arXiv preprint arXiv:1410.5401 (2014)
21. Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S.G., Grefenstette, E., Ramalho, T., Agapiou, J., et al.: Hybrid computing using a neural network with dynamic external memory. Nature **538**(7626), 471–476 (2016)
22. Grollier, J., Querlioz, D., Stiles, M.D.: Spintronic nanodevices for bioinspired computing. Proc. IEEE **104**(10), 2024–2039 (2016)
23. Heinrich, A., Lutz, C., Gupta, J., Eigler, D.: Molecule cascades. Science **298**, 1381–1387 (2002)
24. Henderson, R.: Intel claims that by 2026 processors will have as many transistors as there are neurons in a brain. http://www.pocket-lint.com/news/126289-intel-claims-that-by-2026-processors-will-have-as-many-transistors-as-there-are-neurons-in-a-brain (2014). Accessed 9 June 2017
25. Herz, A.V., Gollisch, T., Machens, C.K., Jaeger, D.: Modeling single-neuron dynamics and computations: a balance of detail and abstraction. Science **314**(5796), 80–85 (2006)
26. Hopfield, J.J.: Pattern recognition computation using action potential timing for stimulus representation. Nature **376**(6535), 33 (1995)
27. Izhikevich, E.: Polychronization: computation with spikes. Neural Comput. **18**(2), 245–282 (2006)
28. Kari, L., Rozenberg, G.: The many facets of natural computing. Commun. ACM **51**(10), 72–83 (2008)
29. Keller, R.: Towards a theory of universal speed-independent modules. IEEE Trans. Comput. **C-23**(1), 21–33 (1974)

30. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. In: Proceedings of the National Academy of Sciences. p. 201611835 (2017)
31. Kish, L.B., Khatri, S., Sethuraman, S.: Noise-based logic hyperspace with the superposition of 2^N states in a single wire. Phys. Lett. A **373**(22), 1928–1934 (2009)
32. Kurenkov, A.: A 'brief' history of neural nets and deep learning. http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning (2015). Accessed 12 April 2017
33. Lazzaro, J., Wawrzynek, J., Mahowald, M., Sivilotti, M., Gillespie, D.: Silicon auditory processors as computer peripherals. IEEE Trans. Neural Netw. **4**(3), 523–528 (1993)
34. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
35. Lee, J., Peper, F.: On Brownian Cellular Automata. In: Proc. of Automata 2008, pp. 278–291. Luniver Press, Bristol, UK (2008)
36. Lee, J., Peper, F., Cotofana, S., Naruse, M., Ohtsu, M., Kawazoe, T., Takahashi, Y., Shimokawa, T., Kish, L., Kubota, T.: Brownian circuits: designs. Int. J. Unconv. Comput. **12**(5–6), 341–362 (2016)
37. Lee, S.W., O'Doherty, J.P., Shimojo, S.: Neural computations mediating one-shot learning in the human brain. PLoS Biol **13**(4), e1002137 (2015)
38. Loh, G.H., Jayasena, N., Oskin, M., Nutter, M., Roberts, D., Meswani, M., Zhang, D.P., Ignatowski, M.: A processing in memory taxonomy and a case for studying fixed-function PIM. In: Workshop on Near-Data Processing (WoNDP), Davis, California (2013)
39. Luczak, A., McNaughton, B.L., Harris, K.D.: Packet-based communication in the cortex. Nat. Rev. Neurosci. **16**, 745–755 (2015)
40. Maass, W., Natschläger, T., Markram, H.: Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput. **14**(11), 2531–2560 (2002)
41. Mack, C.: The multiple lives of Moore's law. IEEE Spectr **52**(4), 31–31 (2015)
42. Markram, H., Lübke, J., Frotscher, M., Sakmann, B.: Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. Science **275**(5297), 213–215 (1997)
43. Merolla, P.A., Arthur, J.V., Alvarez-Icaza, R., Cassidy, A.S., Sawada, J., Akopyan, F., Jackson, B.L., Imam, N., Guo, C., Nakamura, Y., et al.: A million spiking-neuron integrated circuit with a scalable communication network and interface. Science **345**(6197), 668–673 (2014)
44. Minnick, R.C.: A survey of microcellular research. J. ACM **14**(2), 203–241 (1967)
45. Moore, G.E.: Cramming more components onto integrated circuits. Electronics **38**(8), 114–117 (1965)
46. Moore, G.E.: Progress in digital integrated electronics. In: Digest of the 1975. International Electron Devices Meeting, pp. 11–13. Washington, DC (1975)
47. Murata, T.: Petri nets: properties, analysis and applications. Proc. IEEE **77**(4), 541–580 (1989)
48. Peper, F.: Simplifying brownian cellular automata: two states and an average of two rules per cell. In: 2012 Third International Conference on Networking and Computing, pp. 367–370. Naha, Japan (2012)
49. Peper, F., Lee, J., Abo, F., Isokawa, T., Adachi, S., Matsui, N., Mashiko, S.: Fault-tolerance in nanocomputers: a cellular array approach. IEEE Trans. Nanotechnol. **3**(1), 187–201 (2004)
50. Peper, F., Lee, J., Adachi, S., Mashiko, S.: Laying out circuits on asynchronous cellular arrays: a step towards feasible nanocomputers? Nanotechnology **14**(4), 469–485 (2003)
51. Peper, F., Lee, J., Carmona, J., Cortadella, J., Morita, K.: Brownian circuits: fundamentals. ACM J. Emerg. Technol. Comput. Syst. **9**(1), 3-1–3-24 (2013)
52. Reich, D., Mechler, F., Victor, J.: Temporal coding of contrast in primary visual cortex: when, what, and why. J. Neurophysiol. **85**(3), 1039–1050 (2001)
53. Riehle, A., Grn, S., Diesmann, M., Aertsen, A.: Spike synchronization and rate modulation differentially involved in motor cortical function. Science **278**(5345), 1950–1953 (1997)
54. Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., Lillicrap, T.: One-shot learning with memory-augmented neural networks. arXiv preprint arXiv:1605.06065 (2016)
55. Semiconductor Industry Association and others: International technology roadmap of semiconductors (ITRS), chapter on emerging research devices (ERD) (2011), International Roadmap Committee
56. Shah, A.: Intel Will Change Its Approach to PC Chip Upgrades, Deemphasize Process Sizes. PC World, San Francisco, California (2017)
57. Smith, L.S.: Neuromorphic systems: past, present and future. In: Brain Inspired Cognitive Systems 2008, Advances in Experimental Medicine and Biology, vol. 657. Springer, New York (2010)

58. Szatmáry, B., Izhikevich, E.M.: Spike-timing theory of working memory. PLoS Comput. Biol. **6**(8), e1000879 (2010)
59. Szymanski, B.K., Chen, G.G.: Computing with time: from neural networks to sensor networks. Comput. J. **51**(4), 511–522 (2008)

Ferdinand Peper, Ph.D received his Ph.D. in Computer Science from the Delft University of Technology in 1989. In 1990, he joined the National Institute of Information and Communications Technology (NICT) in Japan (then named Communications Research Laboratory) as a researcher and has been at NICT since then, currently as an Associate Director of the Neural Information Engineering Laboratory. His research interests include artificial neural networks, cellular automata, nanocomputer architectures, asynchronous systems, distributed computing, noise and fluctuations, and sensor networks.