Why do we need to introduce temporal behavior in both modern science and modern computing

Introducing temporal logic into computing science

János Végh

Abstract Classic science seemed to be completed more than a century ago, facing only a few (but growing number of!) unexplained issues. Introducing time-dependence to classic science explained those issues, and its consequent use led to a series of modern sciences, including relativistic and quantum physics. Classic computing science today seems to be completed, facing only a few (but growing number of!) unexplained issues. Introducing time-dependence to classic computing science explains those issues. Can this change also lead to a revolution, and resulting in a modern computing science, in the same way as it resulted in the birth of the modern science?

Keywords Temporal logic of computing · Modern computing paradigm · Temporal behavior in computing science · computing performance · stalling · efficiency of ANNs
1 Introduction

Computing science is on the border of mathematics and, through its physical implementation, science. From the beginning of computing, computing paradigm itself, "the implicit hardware/software contract [3]", has defined how mathematics-based theory and its science-based implementation must cooperate. Mathematics, however, considers only dependencies between its operands; it assumes that its operands are instantly available and that value of logical expressions it uses do not change. Resemblance of mathematics and classic science is evident: both of them consider instant interaction. In other words, both classic science and classic computing assume infinite interaction speed between its objects. The approach is absolutely OK as long as pure mathematics is considered, but in computing, a physical implementation of mathematical expressions is used, and that implementation (of course) follows laws of nature (NOT the laws of classic science). However, to discuss features of computing systems, and to introduce into computing science their correct, science-based logic defined by their physical components, is out of scope of "computer science", as quoted above. Classic science enables to accelerate a spaceship to a speed exceeding the speed of light, while the modern science says that it is not possible. The more than 100 years old 'modern science' did not yet touch 'computer science', and this hiatus led on one side to stalling of computing, from single-processor performance through supercomputers to Artificial Neural Network (ANN)s; on the other side, to wasting energy for heating and cooling, rather than computing.

In science, assuming that the speed of light is a limiting speed, enabled to explain the mystic events, such as adding speeds was non-linear any more, and other experiences (and a different thinking about them!) revealed the non-continuous nature of energy. Maybe time can play a significant role again in explaining some mystic issues in computing?

2 Why classic computing is untimed

At the time when von Neumann proposed his famous abstraction to interface mathematics and its engineering implementation, both time of processing and accessing data were in the milliseconds region, while physical data delivery time was in the range of microseconds. It was a plausible assumption to consider that total time of processing comprises only time of computation plus time of data access; data transfer time could be neglected.

Technological development of computing has changed relations between those timings drastically. Today, data transfer time is much longer, than time needed to process them (and led to the symptom, that moving data requires more energy [18] than manipulating it). Besides, the relative
weight of data transfer time has grown tremendously, for many reasons. We cannot neglect it anymore; even, it started to dominate computing. Firstly, miniaturizing the processors to sub-micron size, while keeping the rest of components (such as buses) above the centimeter scale. Secondly, single-processor performance has stalled [24], mainly because of reaching the limits, the laws of nature enable [15] (but, as we present, also because of tremendously extending its inherent idle waiting times). Thirdly, making truly parallel computers failed [3], and we can reach the needed high computing performance only through putting together an excessive number of segregated processors. This latter way replaced parallel computing with parallelized sequential computing, disregarding that operating rules of the latter [19][28][26] sharply differ from those experienced with segregated processors. Fourthly, the utilization mode (mainly multitasking) forced us to use an operating system (OS), which imitates a “new processor” for a new task, at serious time expenses [22][7][30]. Finally, the idea of “real-time connected everything” introduced geographically large distances with their corresponding several millisecond data transfer times. Despite all of this, the idea of non-temporal behavior was confirmed by accepting “weak scaling” [9], suggesting that all housekeeping times, such as organizing the joint work of parallelized serial processors, sharing resources, using exceptions and OS services, delivering data between processing units and data storage units, are negligible.

Classic computing science kept the idea of “instant delivery”; although even within the core, wiring (and its related transfer time) has an increasing weight [15] in the timing budget. Moreover, computing systems “have a clock signal which is distributed in a tree-like fashion into every tiny part of the chip... Approximately 30% of the total consumption of a modern microprocessor is solely used for the clock signal distribution.” [32] It seems to be the case, that the (through their technical implementation: science-based) electronics components, of course, “know” their correct (“modern”) temporal behavior. Their designers, however, attempt to keep the illusion of time-independent operating regime. Or, maybe they have no formalism to handle temporal logic?

3 Famous issues with computing

The vast computing systems can cope with their tasks with growing difficulty, enormously decreasing computing efficiency, and irrationally growing energy consumption; one can experience similar issues in the world of networked edge devices. Being not aware of that the collaboration between processors needs a different approach (another paradigm), resulted in demonstrative failures already known (such as the supercom-
puters Gyoukou and Aurora’18, or the brain simulator SpiNNaker)\(^1\) and many more may follow: such as Aurora’21 [21], the China mystic supercomputers\(^2\) and the EU planned supercomputers\(^3\). General-purpose computing systems comprising "only" millions of processors already show the issues, and the brain-like systems want to comprise four orders of magnitude higher number of computing elements [12]. Moreover, when targeting neuromorphic features such as "deep learning training", the issues start to manifest at a few couple of dozens of processors [11][27]. The scaling is nonlinear [28][29], strongly depending on the workload type, and "artificial intelligence, . . . its the most disruptive workload from an I/O pattern perspective"\(^4\) [27][29] one can run on conventional architectures.

"Successfully addressing these challenges [of neuromorphic computing] will lead to a new class of computers and systems architectures" [23]. However, as noticed by judges of the Gordon Bell Prize, "surprisingly, [among the winners of the supercomputer competition] there have been no brain-inspired massively parallel specialized computers" [4]. Despite the vast need and investments, furthermore, the concentrated and coordinated efforts, just because of the vital bottleneck: the missing modern computing theory.

4 Introducing time to computing

As suspected by many experts, the computing paradigm itself, "the implicit hardware/software contract [3]", is responsible for the experienced issues: "No current programming model is able to cope with this development [of processors], though, as they essentially still follow the classical van Neumann model" [20]. On one side, when thinking about "advances beyond 2020", the solution was expected from a "more efficient implementation of the von Neumann architecture" [14]. On the other side, it was guessed that "the von Neumann architecture is fundamentally inefficient and non-scalable for representing massively interconnected neural networks" [16]. Even publications in most prestigious journals [5][17] are missing the need to introduce temporal behavior. There are many analogies between science and computing [31]; among others, how they handle time. Both classic science and classic comput-

---

1 The explanations are quite different: Gyoukou was withdrawn after its first appearance; Aurora failed: retargeted and delayed; Despite the failure of SpiNNaker1, the SpiNNaker2 is also under construction [13]; "Chinese decision-makers decided to withhold the countrys newest Shuguang supercomputers even though they operate more than 50 percent faster than the best current US machines".

2 https://www.scmp.com/tech/policy/article/3015997/china-has-decided-not-fan-flames-super-computing-rivalry-amid-us

3 https://ec.europa.eu/newsroom/dae/document.cfm?doc_id=60156

4 https://www.nextplatform.com/2019/10/30/cray-revamps-clusterstor-for-the-exascale-era/
The temporal logic of computing

Assume instant (infinitely quick) interaction between its objects. That is, an event happening at any location can be instantly seen at all other locations; time has no specific role, and an event has an immediate effect on all other considered objects. In science, inventing that the speed of light is insurmountable, led to introducing the four-dimensional space-time. Special relativity calculates that fourth spatial coordinate from the time as the distance the light traverses in a given time.

In computing, distances get defined during fabrication of components and assembling the system. In biological systems, nature defines neuronal distances, and in 'wet' neuro-biology, signal timing rather than axon length is the right (measurable) parameter. To introduce temporal logic (meaning: the logical value of an expression depends on WHERE and WHEN is it evaluated) into computing, the reverse of the Minkowski transformation is required. We need to use a special four-vector, where all coordinates are time values: the first three are the corresponding local coordinates, divided by the speed of interaction) having time dimension, and the fourth coordinate is the physical time itself. Distances from location of the event are measured along their access path; they are not calculated from their corresponding spatial coordinates.

Actually, Minkowski invented his four-dimensional space as a purely mathematical construction, assuming only that a limiting speed exists. The space-time became famous after that Einstein discovered, that the limiting speed is the speed of light, but otherwise has not much to do with theory of special relativity. In the case of electronics, the limiting speed is connected to the speed of light. As we use time as primary variable, the formalism can also be used to describe neuronal operation (where the conduction speed is modulated), although using the formalism in that case is less straightforward). At the same time, the famous hypothetical experiment can be excellently described in our time-space system, too.

That is, we introduce a four-dimensional time-space system. The resemblance with the Minkowski space is evident, and the name difference signals the different methods of utilization. For better visibility, the third spatial dimension is omitted in the figures. Figure 1 (essentially a light cone in 2D space plus a time dimension) shows why time must be considered explicitly in all kinds of computing. The figure shows that an event happens in our time-space system at point (0,0,0). Our observers are located on the 'x' axis; vertical scale corresponds to the time. In the classic physical hypothethical experiment, a light is switched on in the origin. The observer switches his light on, when he notices that the first light is switched on. The distance traveled by the light, is given as the value of time multiplied by the speed of light. At any point in time on the vertical axis, a circle describes the propagation of light. In our
Computing operation in time-space approach. The processing operators can be gates, processors, neurons or networked computers. The "idle waiting time", rooting in the finite interaction speed and the physical distance of computing elements (see mixed-color vectors in figure), is one of the major sources of inefficiency of computing systems.

(pseudo) 3-dimensional system, the temporal behavior is described as a conical surface, known as the future light cone.

Both light sources have some 'processing time', that passes between noticing the light (receiving an instruction) and switching their light on (performing an instruction). An instruction is received at the bottom of the green arrow. The light goes on at the head of the arrow (i.e., at the
same location, but at a later time), when 'processing time' \( T_p \) passed. Following that, light propagates in the two spatial dimensions as a circle around the axis 't'. Observers at a larger distance notice the light at a later time: a 'transmission time' \( T_t \) is needed. If 'processing time' of the light source of our first event were zero, light would propagate along the gray surface at the origo. However, because of the finite processing time of the source, light propagates along the blueish cone surface, at the head of the green arrow.

A circle marks the position of our observer on the axis 'x'. With zero 'transmission time', a second gray conical surface (at the head of the horizontal blue dotted arrow) would describe his light. However, this second 'processing time' can only begin when our observer notices the light at his position: when the mixed-color vertical dashed arrow hits the blueish surface. At that point begins 'processing time' of our second light source; the yellowish conical surface, beginning at the head of the second vertical green arrow, describes the second light propagation. The horizontal (blue dotted) arrow describes the physical distance of the observer (as a time coordinate), the vertical (mixed color dashed) arrow describes time delay of the observer light. It comprises two components: \( T_t \) transmission time (mixed color) to the observer and its \( T_p \) processing time (green). The light cone of the observer starts at \( t = 2 \cdot T_p + T_t \).

The red arrow represents the resulting apparent processing time \( T_A \): the longer is the red vector; the slower is the system. As the vectors are in the same plane, \( T_A = \sqrt{T_t^2 + (2 \cdot T_p + T_t)^2} \); that is \( T_A = T_p \cdot \sqrt{R^2 + (2 + R)^2} \). The apparent operating time is a non-linear function of both of its component times and their ratio \( R \). If more computing elements are involved, \( T_t \) denotes the longest transmission time. (Similar statement is valid if \( T_p \) times are different.) The effect is significant: if \( R = 1 \), the apparent execution time of performing the two computations is more than 3 times longer than \( T_p \). Two more observers are located on the axis 'x', at the same position, to illustrate the influence of transmission speed (and/or ratio \( R \)). For visibility, their timings are displayed at points '1' and '2', respectively. In their case the transmission speed differs by a factor of two compared to that displayed at point '0'; in this way three different \( R = T_t/T_p \) ratios are used. Notice that at half transmission speed (the horizontal blue arrow is twice as long as that in the origin) the vector is considerably longer, while at double transmission speed, the decrease of the time is much less expressed\(^5\).

In our particular coordinate system formally (x,y,t) coordinates are used. That is, what happens in time in a component at position (x,y), it happens along a line parallel with axis t, at (x,y). The objects are annotated with their spatial position coordinates 'x' and 'y', but they are time values: how much time the signal having the limiting speed

---

\(^5\) This wants only to illustrate the effect of transmission speed on observations. This phenomenon is discussed in detail in [26].
needs to reach that point. They may alternatively be positioned at some arbitrary position that corresponds to the same time distance from point (0,0,0) (a cylindrical coordinate system would be adequate, but would make both visualization and calculations much harder to follow). The interaction vectors are neither parallel with any of the axes nor are in a spatial plane: both their temporal and spatial coordinates change as the interaction propagates. The arrows in the same horizontal plane represent the same time (no transmission). The horizontal blue arrows are just helper lines: the position (annotated by x,y, but denoting time the signal from (0,0,0) needs to reach this position) is projected to the time axes x and y. The thin red arrow is the vectorial sum of the two projections, also in that plane.

Fig. 2 Performance dependence of an on-chip cache memory, at different cache operating times, in the same topology. Cores at (-0.5,0) and (0.5,0) positions access on-chip cache at (0,0.5) and (0,1), respectively. Vertical orange arrows represent physical cache operating time. Cache memories, from left to right, have physical access speed (on some arbitrary scale) 1 and 10, respectively. Vertical green arrows (from the bottom of the lower arrow to the top of the upper arrow) represent the apparent access time.

Given that apparent processing time \( T_A \) defines performance of the system, \( T_p \) and \( T_t \) must be concerted. Fig. 2 demonstrates why: two different topologies and two different physical cache operating speeds are used in the figure. The signal, requesting to access the cache, propagates along the dotted green vector (it changes both its time and position coordinates; recall that position coordinates are also mapped to time), the cache starts to operate only when the green dotted arrow hits its position. Till that time, the cache is idle waiting. After its operating time (the vertical orange arrow), the result is delivered back to the requesting core. This time can also be projected back to the "position
axes”, and their sum (thin red arrow) can be calculated. Similarly, the requesting core is also "idle waiting" until the requested content arrives. The physical delivery of the fetched value begins at the bottom of the lower thick green arrows, includes waiting (dashed thin green lines), and finishes at the head of the upper thick green vector; their distance defines the apparent cache access time that, of course, is inversely proportional with the apparent cache access speed. Notice that the apparent processing time is a monotonic function of the physical processing speed, but because of the included 'transmission times' due to the physical distance of the respective elements, their dependence is far from being linear. The apparent cache speed increases either if the cache is physically closer to the requesting core or if the cache access time is shorter (or both). The apparent processing time (represented by vertical green arrows) is only slightly affected by the physical speed of the cache memory (represented by vertical orange arrows). See also section 6.5.

As positioning cache and selecting its technology is a question of design, the figure enables to optimize timing versus expenses. The figure also explains rationale behind "in-memory" computing: most of wasted "idle waiting" time can be eliminated. Repeated operation of course can change the idle to active ratio; one must consider, however, the resources the signal delivery uses (they may use the same bus).

5 The price of being idle

The transmission time is an 'idle time' (the mixed-color arrow in Fig. 1) for the observer: it is ready to run, takes power, but does no useful work. Due to their finite physical size and limited interaction speed (both neglected in the classic paradigm), the temporal operation of the computing systems results inherently in an idle time of processing components. As it sensitively depends on many factors and conditions, it can be a significant contributor to non-payload portion of processing time. With other major contributors, originating from their technical implementation [30], these "idle waiting" times sharply decrease payload performance of the systems. In other words, "idle waiting time" leads to poor computing efficiency and/or enormously large energy consumption.

5.1 Gate-level processing

Temporal diagram of a 1-bit adder is shown in Fig. 3. The operations, the gates perform, are the same in both subfigures. The gates are aligned along axis X, and the signals along axis y. The difference between the two cases is the position of the second XOR gate. The absolute distance
from the origin and the signal sources is the same, but distance from the other involved gates is different. Notice that signal $c_o$ is produced in both cases at the same time. Signal $\text{sum}$, however, has quite different timing, just because of different wiring.

![Diagrame](image)

**Fig. 3** Temporal dependence diagram of a 1-bit adder. The second XOR gate is at $(-1,0)$ and $(+1,0)$, respectively. Notice how changing position of a gate affects signal timing. The lack of vertical arrows signals "idle waiting" time (undefined gate output)

Gates are ready to operate as well as signals are ready to be processed (at the head of the blue arrows). Logic gates have the same operating time (the length of green vectors), their access time distance includes the needed multiplexing. The signal must reach their gate (dotted green arrows), that after its operating time passes produces the output signal, that starts immediately towards the next gate. The vertical green arrows denote gate processing (their label shows the operation they perform, and one can project the arrow to axis $x$ to find out the ID of the gate). There are "pointless" arrows in the figure. For example, signal $a \& b$ reaches the OR gate much earlier, than signal to its other input. Depending on operands of OR, it may or may not result in the final sum. Signals have their presumed values only after they received both of their inputs and processed them. Before that time, value of the signal is undefined.

Notice, that considering physical distance and finite interaction speed, drastically changes the picture we have (based on "classic computing"), that operating time of an adder is simply the sum of its "gate times". For example, the first AND and XOR operations could work in parallel (at the same time), but the difference in their physical distance the signals must travel, changes the times when they can operate with their signals.
The difference in timing roots not only in the different number of gates involved: the distance traversed by the signals can contribute equally, and even counterbalance the different number of involved gates. As the $c_0$ output is the input $c_i$ for the next bit, it must be wired there. The total execution time of, say, a 64-bit adder shall be optimized at that level, rather than at bit level. Orchestrating temporal operation through considering both complexity of operation, and positions of signals and operators, can significantly enhance performance.

The goal of this section and Fig 3 is only to call the attention to that in addition to the viewpoint of mathematics (using standard gates and logic functions) and technology (which technology enables to produce smaller gate times and smaller expenses), also the temporal behavior must be considered, when designing chips. Even inside a simple adder circuit, the performance can be changed significantly, only via changing the physical distance of gates; in strong contrast with the "classic computing". Total operating time of the adder is considerably longer than sum of operating times of its gates. Proper positioning of gates (and wiring them) is a point to be considered seriously, and maybe also the role of gates must be rethought.

Notice that this type of 'idle time' remains hidden for single-processor performance measurements. It was experienced, however, that general purpose chips are very inefficient [10]: data signals must be delivered from one gate to another. Dividing larger designs into clock regions, and distributing clock signal "in a tree-like fashion" [32], just to cover temporal behavior of the components, introduces an artificial loss that should be avoided using "modern" (time-aware) design methods.

6 Selected bottlenecks of computing due to its time-independence

Technical implementations of computing are usually designed assuming time-independence and greatly contribute to the experienced inefficiency of computing systems.

6.1 Connecting components

Elements of a computing system are prepared separately and they are connected through a several cm-long bus. That is, a sub-nanosecond processing time is associated with a nanosecond transmission time. Fig. 4 discusses, in terms of "temporal logic": why using high-speed buses for connecting modern computer components leads to very severe performance loss. It is valid for any processing units, but it is especially disadvantageous when one attempts to imitate neuromorphic operation. The processing unit is called 'neuron' here, and a workload of type ANN is assumed; see also section 6.4.
The two neurons of the hidden layer are positioned at (-0.3,0) and (0.6,0). The bus is at position (0,0.5). The two neurons make their computation (green arrows at the position of neurons), then they want to tell their result to fellow neurons. Unlike in biology, first they must have access to the shared bus (red arrows). Core at (-3,0) is closer to the bus, so its request is granted. As soon as the grant signal reaches requesting core, the bus operation is initiated, and the data starts to travel to the bus. As soon as it reaches the bus, it is forwarded by the high speed of the bus, and at that point bus request of the other core is granted, and finally, also computed result of the second neuron is bused.

At this point comes into picture the role of the workload on the system: the two neurons in the hidden layer want to use the single shared bus, at the same time, for communication. As a consequence, the apparent processing time is several times higher, than the physical processing time, and it increases linearly with the number of neurons in the hidden layer (and maybe with also the total number of neurons in the system, if a single high-speed bus is used).

In vast systems, especially when attempting to mimic neuromorphic workload, the speed of the bus is getting marginal. Notice that times shown in the figure are not proportional: the (temporal) distance between cores is in the several picoseconds range, while the bus (and the
arbiter) are at a distance well above nanoseconds, so the actual temporal behavior (and the idle time stemming from it) is much worse than the figure suggests. The figure suggests to use another design principle instead of using the bus exclusively, directly from the position of the computing component.

6.2 Synchronous and asynchronous operation

The case depicted in Fig.1 is an asynchronous operation: when the light cone arrives at the observer, the second processing can start. If we have additional observers, their transmission times may be different, and we have no way to synchronize their operation. If we have another observer at the point mirrored to the origin, the light cone arrives at it at the same time, but to synchronize the operation of the two observers, we would need a 2-fold longer extra synchronization time. Instead, we issue another light cone (the central clock) at the origin (in the case of that light cone, the processing time is zero, just a rising edge), and the observers are instructed to start their processing when this synchronizing light, rather than the event light, cone reaches their point of observation. Given that the internal wiring can be very different, we must choose the clock period according to the "worst-case". The more observers; the more waiting. This effect is considerable even inside the chip (at \( \leq 1 \text{ cm} \) distances); in the case of supercomputers, the distance is about 100 m.

A careful analysis [25] discovered that using synchronous computing (using clock signals) has a significant effect on the performance of large-scale systems mimicking neuromorphic operation. The performance analysis [2] of large-scale brain simulation facilities demonstrated another exciting parallel between modern science and large-scale computing. The commonly used 1 ms integration time, limited both the multi-thread software (SW) simulator, running on general-purpose supercomputers, and the purpose-build hardware (HW) brain simulator to the same performance. Similar shall be the case very soon in connection with building the targeted large-scale neuromorphic systems, despite the initial success of specialized neuronal chips (such as as [16,8]). Although at a higher value (about two orders of magnitude higher than the one in [2]), systems built from such chips also shall stall because of the "quantal nature of time" [31], although using asynchronous operating mode can rearrange the scene.

6.3 Parallelized sequential computing

A major bottleneck in distributed computing is rooting also in "idle waiting", as was correctly identified decades ago [19]. One of the cores
Fig. 5 Operating diagram of parallelized sequential computing systems. One of cores, at position (0,0.5,0), orchestrates parallelization. Two more core are participating in the job, at (-0.5,0,0) and (1.0,0). Notice that the larger physical distance leads to considerable delay in delivering the result back to the coordinator core. Green arrows denote payload, dashed orange arrows non-payload processing time.

(in Fig. 5 the one at position (0,0.5,0)) starts with some sequential-only processing. In the next step it shares the job with its fellow cores, cycling through their addresses. Core at (-0.5,0,0) is the first. Notice that even in the timeless paradigm, the first core must ‘idle wait’ the sequential-only processing, plus the end of first cycle. Given that signal must propagate to it from the originating core, it can begin its part of calculation only at the beginning of the green arrow. Similarly, core at (1.0,0) is started, in the second round. Notice that its idle waiting is longer because of looping in the initiator core, and similarly, the transmission time is also longer.
Fig. 6 The surface and the figure marks show at what efficiency the top supercomputers run the ‘best workload’ benchmark HPL, and the ‘real-life load’ HPCG. [26] The right bottom part displays the expected efficiency [25] of running neuromorphic calculations on SPA computers.

After sharing the job, all cores start to make their part of calculation. We assume that all cores need the same time to perform their part, and after that they return the result to the organizer core. Orchestrator core must wait the slowest fellow core; processing time of the parallelized system is defined by the largest apparent processing time\(^6\). As shown in the figure, the looping contribution is increasing with adding more cores to the loop and the transmission delay is increasing with the physical size of the supercomputer.

The parallelized sequential computing introduces a rule of ‘adding performance’ in modern computing, that is quite similar to rule of ‘adding speed’ in modern physics [31], see Fig. 7. The effect on efficiency of supercomputers is depicted in Fig. 6. This is a natural loss, that can be mitigated, but cannot be eliminated.

Figure 6 also depicts how computing efficiencies of recent supercomputers depend (see discussion in [26]) on the number of single-threaded processors in the system and parameter \((1 - \alpha)\), describing the non-payload portion defined by the corresponding benchmark task. It is

\(^6\) Notice that looping delay can be combined with propagation delay: their rational pairing enabled in the case of Sierra a 17 % increase in payload performance with 0 % increase in the nominal performance.
known since decades that "this decay in performance is not a fault of the architecture, but is dictated by the limited parallelism" [19]; in excessive systems of modern HW, is also dictated by laws of nature [31].

6.4 Neuromorphic computing

Biology strictly considers physical distance and operating speed of its components: slight changes in values in timing, result in severe disfunctionality. Biology uses a more complex computing system: not only that \( T_p \) and \( T_t \) times are in the same order of magnitude (i.e., their timely behavior must evidently be considered) but also the conduction velocity (the interaction speed) is changed significantly, case by case (if needed, by a factor about one hundred!), to deliver needed control signals to their place [30]. However, using the proposed time-space system we can correctly describe neuronal operation, too. We must, however, consider that in their case the interaction speed is largely different for the different components/events.

In sections 6.1 and 6.3 was discussed the timely behavior of serial bus and distributed parallelized processing, respectively. The classic ANNs combine their disadvantages into one single inefficient system: *signal transition time between neurons can be orders of magnitude higher that their processing time*. Given that, as discussed in section 5.1, in a technical implementation of neuronal operation, in most portion of the total time, value of the output signal of neurons differs from the expected one. These facts, combined, mean, that when "training" ANNs, fellow neurons receive that (maybe wrong) output signal in most of their learning period, and correspondingly, they also provide (maybe) false input for the linked neurons. Given that neurons do not know which is the "right" signal, upon receiving a new input, they adjust their synaptic weights to the wrong signal. The larger the system, the worse the effect; the result is
The temporal logic of computing 17

weak-long training, even for (compared to functionality of brain) very simple tasks. The effect of undefined output is of course known in engineering: for example, in processors, adders comprising several one-bit adders, do not provide their final output until a fixed time (supposing that until that time signals in its components are relaxed).

The analysis of temporal behavior of ANNs underlines that, in general, "artificial intelligence, . . . its the most disruptive workload from an I/O pattern perspective"\(^7\). In practice it means, that imitating neuromorphic computing on conventional architectures can be performed only with very low computing efficiency \([25, 29]\).

6.5 New materials and/or physical effects

Ongoing research may result in new physical effects and/or technologies and/or materials. The general temporal behavior of matter, however, limits their usability. Fig. 2 depicts the temporal behavior of a cache operation. Using a much quicker computing element in place of a slower component has only a marginal effect if the transmission time (i.e., the physical size) limits the apparent speed of operation. Similar holds if you replace the components with some others (such as much quicker processing or storage element). Mimicking the biology is useful also here: the time window where the decision is made\(^8\) is of the same size, independently of the path traversed by the signal (the axon length) and the speed of the signal (conduction velocity); and is in the order of the 'processing time' of the neurons.\(^9\)

To fabricate smaller components without decreasing the processing time proportionally; and similarly, replacing a processing element with a very much quicker one (such as proposed in \([6][1]\), and may be proposed using any future new physical effect and/or material) is not reasonable, and it has only marginal effect, if the physical distance of the computing elements cannot be reduced proportionally at the same time. The speed of light is insurmountable and also limits performance of future computing.

7 Conclusion

This paper introduces the concept of timely behavior into computing (a temporal logic), while the model preserves the solid computing science base. The introduced formalism enables to calculate the effects

---

\(^7\) https://www.nextplatform.com/2019/10/30/cray-revamps-clusterstor-for-the-exascale-era/

\(^8\) In computing: WHEN and WHERE the logical function is evaluated

\(^9\) The biology can change the conduction velocity, that needs energy, so finding an optimum is not as simple.
of temporal behavior, rooting in science, of our computing systems. All fields of computing benefit from introducing temporal behavior for computing components, from explaining the need of "in-memory computing" to reasoning the low efficiency of the Graphic Processing Unit (GPU)s in general-purpose applications and comprehending the experienced weaks-long training times of ANNs; as well as researching more new physical effects/technologies/materials. Neglecting temporal behavior, led already to waste vast amounts of energy and introduced performance limits for important computing systems. Besides, it limits the utility of any future method, material or technology, if they are designed/developed/used in the spirit of the old (timeless) paradigm.

References

1. Building brain-inspired computing. Nature Communications 10(12), 4838 (2019). URL https://doi.org/10.1038/s41467-019-12521-x
2. van Albada, S.J., Rowley, A.G., Senk, J., Hopkins, M., Schmidt, M., Stokes, A.B., Lester, D.R., Diesmann, M., Furber, S.B.: Performance Comparison of the Digital Neuromorphic Hardware SpiNNaker and the Neural Network Simulation Software NEST for a Full-Scale Cortical Microcircuit Model. Frontiers in Neuroscience 12, 291 (2018)
3. Asanovic, K., Bodik, R., Demmel, J., Keaveny, T., Keutzer, K., Kubiatowicz, J., Morgan, N., Patterson, D., Sen, K., Wawrzynek, J., Wessel, D., Yelick, K.: A View of the Parallel Computing Landscape. Comm. ACM 52(10), 56–67 (2009)
4. Bell, G., Bailey, D.H., Dongarra, J., Karp, A.H., Walsh, K.: A look back on 30 years of the Gordon Bell Prize. The International Journal of High Performance Computing Applications 31(6), 469484 (2017). URL https://doi.org/10.1177/1094342017738610
5. Bourzac, K.: Stretching supercomputers to the limit. Nature 551, 554–556 (2017)
6. Chicca, E., Indiveri, G.: A recipe for creating ideal hybrid memristive-CMOS neuromorphic processing systems. Applied Physics Letters 116(12), 120501 (2020). DOI 10.1063/1.5142089. URL https://doi.org/10.1063/1.5142089
7. David, F.M., Carlyle, J.C., Campbell, R.H.: Context Switch Overheads for Linux on ARM Platforms. In: Proceedings of the 2007 Workshop on Experimental Computer Science, ExpCS ’07. ACM, New York, NY, USA (2007). DOI 10.1145/1281700.1281703. URL http://doi.acm.org/10.1145/1281700.1281703
8. Davies, et al., M.: Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. IEEE Micro 38, 82–99 (2018)
9. Gustafson, J.L.: Reevaluating Amdahl’s Law. Commun. ACM 31(5), 532–533 (1988). DOI 10.1145/42411.42415
10. Hameed, R., Qadeer, W., Wachs, M., Azizi, O., Solomatnikov, A., Lee, B.C., Richardson, S., Kozyrakis, C., Horowitz, M.: Understanding sources of inefficiency in general-purpose chips. In: Proceedings of the 37th Annual International Symposium on Computer Architecture, ISCA ’10, pp. 37–47. ACM, New York, NY, USA (2010). DOI 10.1145/1815961.1815968. URL http://doi.acm.org/10.1145/1815961.1815968
11. Keuper, J., Preundt, F.J.: Distributed Training of Deep Neural Networks: Theoretical and Practical Limits of Parallel Scalability. In: 2nd Workshop on Machine Learning in HPC Environments (MLHPC), pp. 1469–1476. IEEE (2016). DOI 10.1109/MLHPC.2016.006. URL https://www.researchgate.net/publication/308457837
12. Liao, Xiang-ke et al: Moving from exascale to zettascale computing: challenges and techniques. Frontiers of Information Technology & Electronic
13. Liu, C., Beller, G., Vogginger, B., Kappel, D., Partasch, J., Neumrker, F., Hppner, S., Maass, W., Furber, S.B., Legenstein, R., Mayr, C.G.: Memory-Efficient Deep Learning on a SpiNNaker 2 Prototype. Frontiers in Neuroscience 12, 840 (2018). DOI 10.3389/fnins.2018.00840. URL https://www.frontiersin.org/article/10.3389/fnins.2018.00840

14. Machine Intelligence Research Institute: Erik DeBenedictis on supercomputing (2014). URL https://intelligence.org/2014/04/03/erik-debenedictis/

15. Markov, I.: Limits on fundamental limits to computation. Nature 512(7513), 147–154 (2014)

16. Sawada et al., J.: TrueNorth Ecosystem for Brain-Inspired Computing: Scalable Systems, Software, and Applications. In: SC ’16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 130–141 (2016)

17. Service, R.F.: Design for U.S. exascale computer takes shape. Science 359, 617–618 (2018)

18. Simon, H.: Why we need Exascale and why we won’t get there by 2020. In: Exascale Radioastronomy Meeting, AASCTS2 (2014). URL https://www.researchgate.net/publication/261879110

19. Singh, J.P., Hennessy, J.L., Gupta, A.: Scaling parallel programs for multiprocessors: Methodology and examples. Computer 26(7), 42–50 (1993). DOI 10.1109/MC.1993.274941

20. (S)oOS project: Resource-independent execution support on exa-scale systems. http://www.soos-project.eu/index.php/related-initiatives (2010)

21. Top500.org: Retooled Aurora Supercomputer Will Be Americas First Exascale System. https://www.top500.org/news/retooled-aurora-supercomputer-will-be-americas-first-exascale-system/ (2017)

22. Tsafrir, D.: The context-switch overhead inflicted by hardware interrupts (and the enigma of do-nothing loops). In: Proceedings of the 2007 Workshop on Experimental Computer Science, ExpCS ’07, pp. 3–3. ACM, New York, NY, USA (2007)

23. US DOE Office of Science: Report of a Roundtable Convened to Consider Neuromorphic Computing Basic Research Needs. https://science.osti.gov/-/media/ascr/pdf/programdocuments/docs/Neuromorphic-Computing-Report_FNLBLP.pdf (2015)

24. US National Research Council: The Future of Computing Performance: Game Over or Next Level? (2011). URL http://science.energy.gov/~/media/ascr/ascac/pdf/meetings/mar11/Yelick.pdf

25. Végh, J.: How Amdahl’s Law limits the performance of large artificial neural networks: . Brain Informatics 6, 1–11 (2019). URL https://braininformatics.springeropen.com/articles/10.1186/s40708-019-0097-2/metrics

26. Végh, J.: Finally, how many efficiencies the supercomputers have? The Journal of Supercomputing (2020). URL https://doi.org/10.1007%2Fs11227-020-03210-4

27. Végh, J.: How deep the machine learning can be, pp. 141–169. A Closer Look at Convolutional Neural Networks. Nova, In press (2020). URL https://arxiv.org/abs/2005.00872

28. Végh, J.: Re-evaluating scaling methods for distributed parallel systems. IEEE Transactions on Distributed and Parallel Computing ??, in review (2020). URL https://arxiv.org/abs/2002.08316

29. Végh, J.: Which scaling rule applies to Artificial Neural Networks. In: Computational Intelligence (CSCE) The 22nd Int’l Conf on Artificial Intelligence (ICAI’20), p. Accepted ICA2246. IEEE (2020). URL http://arxiv.org/abs/2005.08942
30. Végh, J., Berki, A.J.: Do we know the operating principles of our computers better than those of our brain? Neurocomputing p. in review (2020). URL https://arxiv.org/abs/2005.05061

31. Végh, J., Tisan, A.: The need for modern computing paradigm: Science applied to computing. In: Computational Intelligence CSCI The 25th Int'l Conf on Parallel and Distributed Processing Techniques and Applications, pp. 1523–1532. IEEE (2019). DOI 10.1109/CSCI49370.2019.00283. URL http://arxiv.org/abs/1908.02651

32. Waser, R. (ed.): Advanced Electronics Materials and Novel Devices. Nanoelectronics and Information Technology. Wiley (2012)