Global Journal of Computer Science and Technology: A
Hardware & Computation
Volume 20 Issue 1 Version 1.0 Year 2020
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Why do we need to Introduce Temporal Behavior in both Modern Science and Modern Computing, With an Outlook to Researching Modern Effects/Materials and Technologies

By Janos Vegh

Kalimanos BT

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 into classic science explained those issues, and its consequent use led to the birth of a series of modern sciences, including relativistic and quantum physics. Classic computing is based on the paradigm proposed by von Neumann for vacuum tubes only, which seems to be completed in the same sense. Von Neumann warned, however, that implementing computers under more advanced technological conditions, using the paradigm without considering the transfer time (and especially attempting to imitate neural operation), would be unsound. However, classic computing science persists in neglecting the transfer time and is facing a few (but growing number of!) unexplained issues, and its development stalled in most of its fields. Introducing time-dependence into the classic computing science explains those issues and discovers the reasons for its experienced stalling. It can lead to a revolution in computing, resulting in a modern computing science, in the same way, as it resulted in modern science's birth.

Keywords: temporal logic of computing, modern computing paradigm, temporal behavior in computing science, computing performance, stalling, efficiency of ANNs.

GJCST-A Classification: F.4.1

Why do we need to Introduce Temporal Behavior in both Modern Science and Modern Computing, With an Outlook to Researching Modern Effects/Materials and Technologies

Janos Vegh

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 into classic science explained those issues, and its consequent use led to the birth of a series of modern sciences, including relativistic and quantum physics. Classic computing is based on the paradigm proposed by von Neumann for vacuum tubes only, which seems to be completed in the same sense. Von Neumann warned, however, that implementing computers under more advanced technological conditions, using the paradigm without considering the transfer time (and especially attempting to imitate neural operation), would be unsound. However, classic computing science persists in neglecting the transfer time and is facing a few (but growing number of!) unexplained issues, and its development stalled in most of its fields. Introducing time-dependence into the classic computing science explains those issues and discovers the reasons for its experienced stalling. It can lead to a revolution in computing, resulting in a modern computing science, in the same way, as it resulted in modern science's birth.

Keywords: temporal logic of computing, modern computing paradigm, temporal behavior in computing science, computing performance, stalling, efficiency of ANNs.

I. Introduction

Computing science is on the border of mathematics and, through its physical implementation, science. From the beginning of computing, the computing paradigm itself, ”*the implicit hardware/software contract* [3]”, has defined how the mathematics-based theory and its science-based implementation must cooperate. The contract is based on the famous ”First Draft” [22]. Von Neumann warned, however: ”*6.3 At this point, the following observation is necessary. In the human nervous system, the conduction times along the lines (axons) can be longer than the synaptic delays, hence our above procedure of neglecting them aside of τ [the processing time] would be **unsound**. In the actually intended vacuum tube interpretation, however, this procedure is justified: τ is to be about a microsecond, an electromagnetic impulse travels in this time 300 meters, and as the lines are likely to be short compared to this, the conduction times may indeed be neglected. (**It would take an ultra-high frequency device** $- \approx 10^{-8}$ **seconds or less – to vitiate this argument.**)*” That is, according to its inventor, *the paradigm is justified only for the relationship between transfer time and processing time, represented by vacuum tubes*, and is surely unsound for workloads mimicking neural behavior. However, von Neumann did not suggest another procedure that we could follow when using a different technology.

The technological development of computing has changed the relationship between those timings drastically. Today, the data transfer time is much longer than the time needed to process it (and led to the symptom that moving data requires more energy [25] 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 the components (such as buses) above the centimeter scale. Secondly, the single-processor performance has stalled [31], mainly because of reaching the limits the laws of

Author: Project no. 136496 has been implemented with the support provided by the National Research, Development and Innovation Fund of Hungary, financed under the K funding scheme, Kalimanos BT, Hungary. e-mail: Vegh.Janos@gmail.com

nature enable [21] (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 the operating rules of the latter [26]**[36][33]** 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 [29][8]**[37]**. 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" [13], 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.*

Von Neumann was aware of the technological development and suggested to revise the validity of the paradigm when entering a new technology age. However, the theoretical basis for computing remained his original paradigm, and the solid mathematical background of computing science is still based on it; that is ***unsound*** under the present technological conditions.

The classic computing science kept the idea of "instant delivery"; although even within the core, wiring (and its related transfer time) has an increasing weight [21] 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.*" [39] 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 a time-independent operating regime. Or, maybe they have no formalism to handle temporal logic?

Mathematics considers only logical *dependencies* between its operands: it assumes that its operands are instantly available, and that the value of logical expressions it uses do not change if we evaluate at a different time or different place. The 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 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 the laws of nature (NOT the laws of classic science). However, to discuss features of technological computing systems and to introduce into computing science their correct, science-based behavior is out of the scope of "computing science". Classic science enables accelerating a spaceship to a speed exceeding the speed of light, while modern science says it is impossible. The more than 100 years old 'modern science' did not yet touch 'computing 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 on heating and cooling, rather than computing.

In science, assuming that the speed of light is a limiting speed, enabled us 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. The method, transforming classic science into modern science, seems to be promising for transforming classic computing into modern computing.

II. 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 (upgraded paradigm), resulted in demonstrative failures already known (such as the supercomputers Gyoukou and Aurora'18, or the brain simulator SpiNNaker)[1] and many more may follow: such as Aurora'21 [28], the China's mystic supercomputers[2] and the EU planned supercomputers[3]. The new world champion (as of November 2020) *Fugaku* stalled at some 40% of its planned capacity [10].

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 [17]. Moreover, when targeting neuromorphic features such as "deep learning training", the issues start to manifest at a few couples of dozens of processors [16]**[34]**. The scaling is nonlinear **[36]**, strongly depending on the workload type, and "*artificial intelligence, . . . it's the most disruptive workload from an I/O pattern perspective*"[4] **[34][36]** one can run on conventional architectures.

"*Successfully addressing these challenges [of neuromorphic computing] will lead to a new class of computers and systems architectures*" [30]. 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, the concentrated and coordinated efforts, just because of the critical bottleneck: ***the missing modern computing theory***.

III. 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*" [27]. On one side, when thinking about "advances beyond 2020", the solution was expected from a "*more efficient implementation of the von Neumann architecture*" [20]. On another side, it was guessed that "*the von Neumann architecture is fundamentally inefficient and non-scalable* for representing massively interconnected neural networks" [23]. Even publications in the most prestigious journals [5][24] are missing the need to introduce temporal behavior.

[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 [18]; "Chinese decision-makers decided to withhold the country's 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/

a) *The limiting speed in science*

Classic science knows the temporal dependence of interactions (from our point of view: transferring information via physical interaction) only in the sense that if we move, for example, an electric charge, the *frequency* of the generated electromagnetic (EM) wave can be calculated. However, because of assuming instant interaction, the *speed* of the EM wave cannot: *the instant interaction is achievable only having infinitely large speed*. Due to this hiatus, *one of the vital features of computing (including neural) networks remained out of sight of the research*. This hiatus was resulting in an incomplete description of their behavior.

The fact that the speed of light is finite was known since Galilei. Moreover, Einstein discovered that interactions, such as forces between objects having electric charge or gravitational mass, have a finite interaction speed (in other words, their interaction is not instant, although very fast). In his (implicit[5]) interpretation, *there exists a universal limiting speed for interactions, that even the light (given that it is a propagating electromagnetic interaction) cannot exceed*. The scientific truth about the existence of a finite interaction speed, in general, was recently confirmed by providing experimental evidence for the existence of gravitational waves.[6] The experimental evidence also indirectly underpins that *the mathematical background of the Minkowski transform is well-established and correctly describes nature*, where the speed of interaction is limited. Modern treatments of special relativity base it on the single postulate of Minkowski spacetime [7].

b) *The time in computing*

In our electronic devices, the EM waves are propagating with a speed proportional to light's speed; that is, they have a *limiting speed*. In the first computer [15,12] (as well as in the present computers), the interaction speed was in the range of 10^8 m/s-range. The "processor size" was in the m-range, the timings (cycle length, access time, instruction's execution time) in the $msec$-range. Under those conditions, the data transfer time, also theoretically, could be safely neglected, as von Neumann emphasized it. For today, thanks to the technological development, the processor dwarfed million-fold, the timing got million-fold faster. The interaction speed, however, did not change. Since the first personal computers' appearance, the physical components' characteristic size, such as the length of buses connecting their components, did not change significantly (unlike the distance of computing gates, see Moore's observation).

Thanks to the decreasing density and the increasing frequency, *the speed of changing the electronic states in a computing system, more and more approached the limiting speed*. It was recognized that the system's clock signals must be delayed [39], that effort takes nearly half of the power consumption of the processor (and the same amount of energy is needed for cooling). Furthermore, only about 20% of the consumed (payload) power is used for computing [25]; the rest goes for transferring data from one place to another. Despite these shortcomings, it was not suspected that *the physical implementations of*

[5] Einstein in his classic paper "On the electrodynamics of moving bodies" [11] speaks about that "light is always propagated in empty space with a definite velocity c," given that the light represents the propagation of the electromagnetic interaction. However, in the abstract of his paper, he mentions that he speaks about "the phenomena of electrodynamics as well as of mechanics", i.e., gravity. The formalism, however, was available for electrodynamics only; for gravity only a decade later.

[6] Both assuming the non-instant nature of those interactions, and demonstrating their existence, deserved Nobel price.

the electronic components have a temporal behavior [**35**]. However, it was the final reason for many experienced issues, from the payload performance limit of supercomputers [**33**] and brain simulation [**32**], to the weeks-long training time in deep learning [**34, 36**].

c) The limiting speed in biology

In contrast with technical computing, in biological computing systems, the "spatiotemporal" behavior of dynamically interacting neurons is explicitly investigated. Their interaction speed ("conduction velocity") depends both on their inherent parameters and actual conditions. The *cm*-range distances, the m/s-range interaction speed, and the *msec*-range timings (periodicity, spike length, etc.) prove that the biological networks' apt description is feasible only with temporal logic. In this context, both theoretically and in real-time[7] simulations: the temporal behavior is a vital feature of biological systems.

The "liquid state machine" model grasps the essential point of the biological neural networks: *their logical behavior cannot be adequately described without using both time and space coordinates.* The standard interpretation is used in former studies in describing neural behavior: the mathematical formalism used time (t) as an independent parameter that was not connected (through the interaction speed of the action under study) to the spatial coordinates. Thus, that model cannot provide a full-featured description of biological neural networks' behavor, furthermore it does not enable us to analyze their technical implementations' temporal behavior.

d) The common mathematics describing the limiting speed

The Minkowski-transform become famous for its role in accepting, quickly and widely, Einstein's special relativity theory. For our paper, we only assume that a limiting speed (in both electronic and biological systems) exists, and *transferring information in the system needs a limited time.* In biology, the limiting speed (the conduction velocity) is modulated, but our statement holds for any single action: *the spatial and temporal coordinates are connected through the corresponding limiting speed.* In our approach, *Minkowski provided a mathematical method to describe information transfer phenomena in a world where the interaction speed is limited.* For example, if we have our touching sense as the only source of information from the external world, we need to walk to the object, automatically limiting the information propagation speed (both touching and being touched) to our walking speed. This case is the same when transferring information in computing systems: space-time four-coordinates describe that world. The only new assumptions we make are that the events also have a processing time, such as an atomic transition, executing a machine instruction or issuing/receiving a neural spike[8], furthermore, that the limiting interaction speed is other than the speed of light.

We can proceed, following Minkowski, merely introducing a fourth coordinate, and through the assumed limiting speed (without making further assumptions about the value and nature of that speed), we can *transform the time of propagation of an event (the interaction, aka the physical transfer of the information) to a distance*, within which the interaction can have an effect in the considered time duration. *Notice the critical aspect, that space and time not only have equal rank, but they are connected through the interaction speed, and that all coordinates have the same dimension.*

[7] The real-time in our terminology means that all computing events happen on the biologically correct time scale, instead of that, on average, the computing time matches the biological time.

e) Details of the mathematical discussion

In computing, the distances get defined during the fabrication of components and assembling the system. In biological systems, nature defines the 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, we need the reverse of the Minkowski transformation. 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 a time dimension; the fourth coordinate is the physical time itself. *The distances from the event's location are measured along with their access path; instead of calculating them from their corresponding spatial coordinates.*

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 utilization methods. For a 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; the vertical scale corresponds to the time. In the classic physical hypothetical 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 time when the observer notices the light is given as its distance from the origin divided by the speed of light. At any point in time on the vertical axis, a circle describes the propagation of light. In our

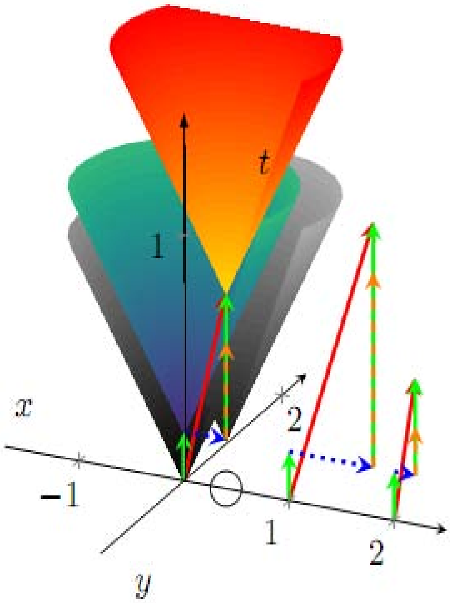


Fig.1 The Computing operation in *time-space* approach. The processing operators (depending on the granularity concerned) can be gates, processors, neurons, or networked computers. The "idle waiting time", rooting in the finite interaction speed and the physical distance of computing elements (see mixed-color vectors in figure), is one of the major sources of inefficiency of computing systems.

[8] Receiving a neural spike, however, is a little bit particular case: because of the integration, in some cases, the "processing time" can vary.

(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 (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 the '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 the 'processing time' of the light source of our first event were zero; the light would propagate along the gray surface at the origo. However, because of the finite processing time of the source, the 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 the 'processing time' of our second light source; the yellowish conical surface, beginning at 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 the time delay of the observer light. It comprises two components: the T_t transmission time (mixed color) to the observer and its T_p processing time (green). The light cone of the observer starts at $t = 2 * T_p + T_t$.

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 on 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 the T_p times are different.) The effect is significant: if $R = 1$, the apparent execution time of performing the two computations is more than three times longer than T_p. Two more observers are located on the axis 'x', at the same position, to illustrate the influence of the 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[9].

f) The role of time in performance

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), is depicted along a line parallel with axis t, at (x,y). *The objects are annotated with their spatial position coordinates 'x' and 'y'. Those coordinates are time values: how much time the signal having the limiting speed needs to reach that point.* They may alternatively be positioned at some arbitrary position that corresponds to the same *time distance* from the point (0,0,0) (a cylindrical coordinate system

[9] This wants only to illustrate the effect of transmission speed on observations. This phenomenon is discussed in detail in [**33**].

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 the 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.

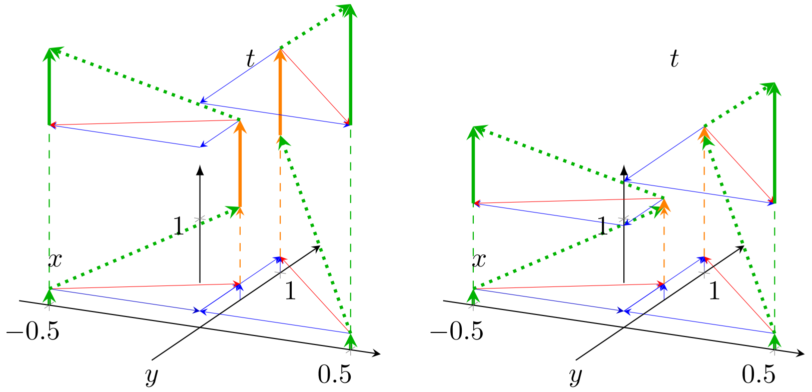


Fig. 2 The dependence of on-chip cache memory's operating speed at different physical cache operating times, in the same topology. The cores at (-0.5,0) and (0.5,0) positions access on-chip cache at (0,0.5) and (0,1), respectively. Vertical orange arrows represent physical cache operating time. Cache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

Given that the *apparent processing time* T_A defines the 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 cache memory's physical speed (represented by vertical orange arrows). See also section 5.5.

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

IV. 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 they are neglected in the classic paradigm), *the temporal operation of the computing systems results inherently in an idle time of their processing components.* As it sensitively depends on many factors and conditions, it can be a significant contributor to the processing time’s non-payload portion. With other significant contributors, originating from their technical implementation [**37**], these ”idle waiting” times sharply decrease the payload performance of the systems. In other words, the ”idle waiting time” *leads to low computing efficiency and/or enormously large energy consumption.*

a) Gate-level processing

The 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 are the same, but the other involved gates’ distances are different. Notice that the signal c_o is produced in both cases at the same time. The signal sum, however, has entirely different timing, just because of the different wiring.

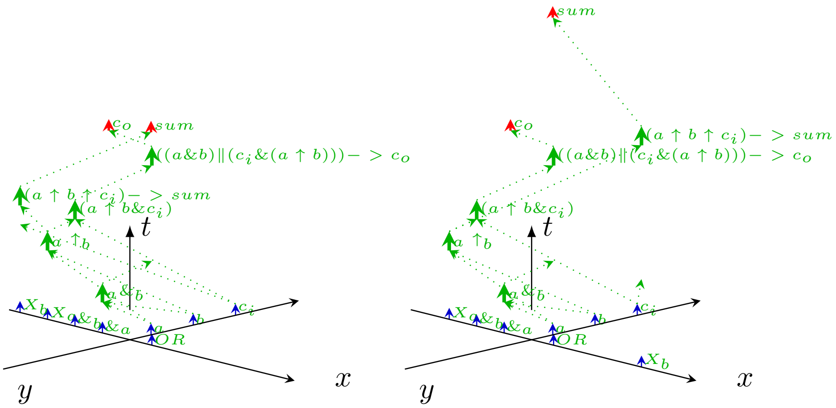


Fig. 3 The temporal dependence diagram of a 1-bit adder. The second XOR gate is at (-1,0) and (+1,0), respectively. Notice how changing the position of a gate affects signal timing. *The lack of vertical arrows* signals ”idle waiting” time (undefined gate output)

The gates are ready to operate, and the signals are ready to be processed (at the head of the blue arrows). The logic gates have the same operating time (the length of the 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 which 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 gate's ID). There are "pointless" arrows in the figure. For example, the signal $a\&b$ reaches the OR gate much earlier than the signal to its other input. Depending on the operands of OR, it may or may not result in the final sum. *The signals have their presumed values only after they received both of their inputs and processed them. Before that time, the value of the signal is undefined.*

Notice that considering the physical distance and the finite interaction speed drastically changes the picture we have (based on "classic computing"), that the 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 they can counterbalance the different number of the involved gates. As the c_o 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 the bit level. Orchestrating the gates' temporal operation by considering both the complexity of their operation and the positions of signals and operators can significantly enhance their performance.

The goal of this section and Fig 3 is only to call attention to that *in addition to the viewpoint of mathematics (using standard gates and logic functions) and technology (which technology enables to produce shorter gate times and smaller expenses), also the temporal behavior must be considered when designing chips.* Even inside a simple adder circuit, one can change the performance significantly, only via changing the physical distance of the gates; in contrast with the "classic computing". *The total operating time of the adder is considerably longer than the sum of its gates' operating times.* The 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 [14]: 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" [39], just to cover the temporal behavior of the components, introduces an artificial loss that should be avoided using "modern" (time-aware) design methods.

V. The Temporal Behavior of Selected Bottlenecks of Computing

The technical implementations of computing are usually designed assuming time-independence and significantly contribute to the experienced inefficiency of computing systems.

a) Connecting components

The elements of a computing system are prepared separately, and they are connected through a several cm-long buses. A sub-nanosecond processing time is associated with a nanosecond transmission time, clearly making the paradigm *unsound*. The "in memory" processing reduces this time and decreases the "idle waiting" time, increasing the apparent processing speed.

Fig. 4 discusses, in terms of "temporal logic", why *using high-speed buses for connecting modern computer components leads to very severe performance loss; furthermore, that it strongly distorts the true timing relations.* 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 5.4.

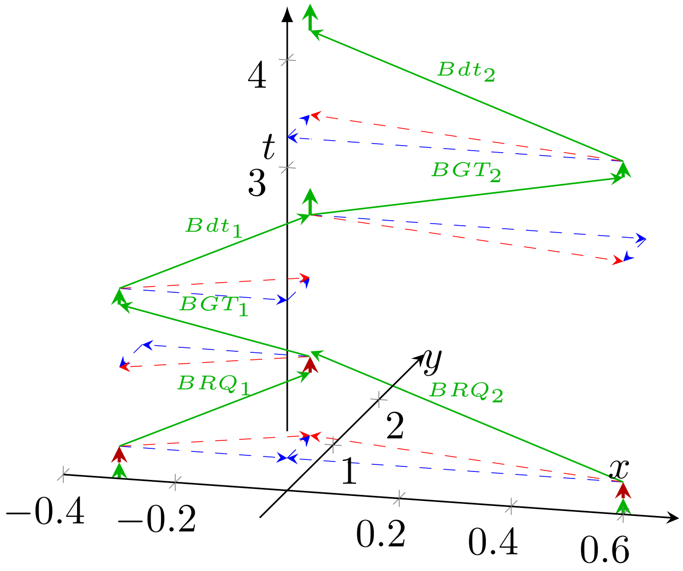


Fig. 4 The operation of the sequential bus in the time-space coordinate system. Near to axis t, *the lack of vertical arrows* signals ”idle waiting” time

The two neurons of the hidden layer are positioned at (-0.3,0) and (0.6,0). The bus is at a 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 their peer neurons. Unlike in biology, first, they must have access to the shared bus (red arrows). The bus requests need time to reach the arbiter. The core at (-.3,0) is closer to the bus, so its request is granted. As soon as the grant signal reaches the requesting core, the bus operation is initiated, and the data starts to travel to the bus. As soon as it reaches the bus, the bus’s high speed forwards it, and at that point, the bus request of the other core is granted. Finally, the computed result of the second neuron is bused.

At this point comes into the 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. It increases linearly with the number of neurons in the hidden layer* (and maybe also with 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 bus’s speed is getting marginal. Notice that the times shown in the figure are not proportional: the (temporal) distance between cores are in the several picoseconds range, while the bus (and the arbiter) is 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.* This effect is why *”The idea of using the popular shared bus to implement the communication medium is no longer acceptable, mainly due to its high contention.”* [19]. The figure suggests using another design principle instead of exclusively using the bus directly from the computing component’s position.

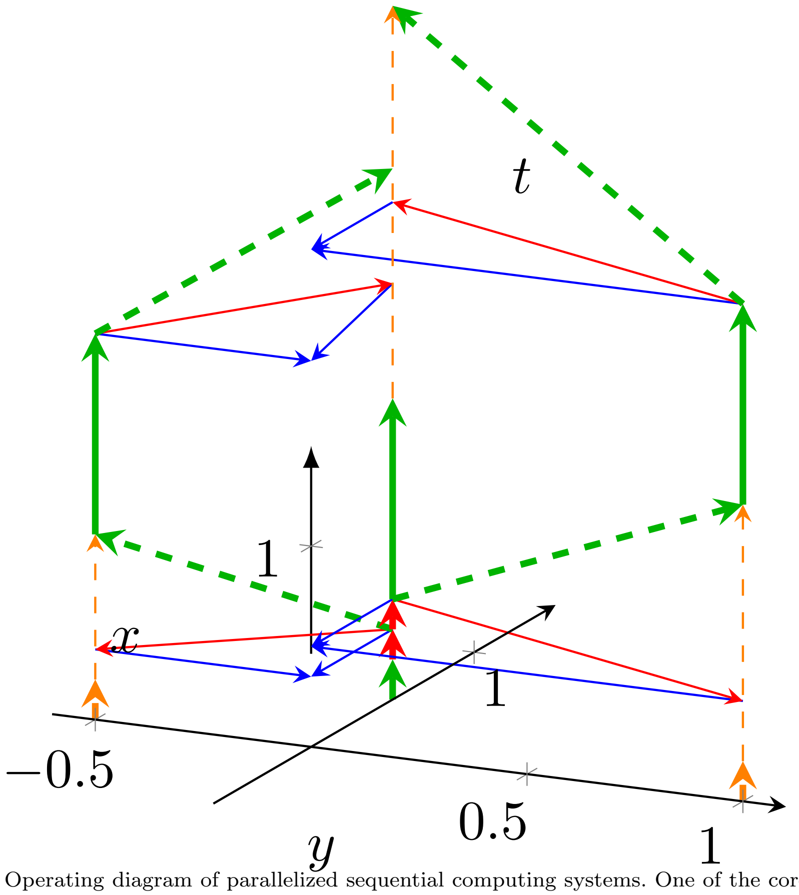
b) Synchronous and asynchronous operation

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. Furthermore, 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 simultaneously. However, to synchronize the two observers' operation, 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, reaches their observation point. If the synchronization period is large enough, all observers will notice the event light: they will be within the synchronization light cone. After noticing the synchronization light, they can all start their processing at that time, which equals to the sum of the two processing times plus the synchronization time. *The idle time for all observers increases.* Given that the internal wiring can be very different, we must choose the clock period according to the "worst-case". All observers must wait for the slowest one. The more observers, the more waiting. This effect is considerable even inside the chip (at $\leq cm$ distances); in supercomputers, the distance is about 100 m.

A careful analysis **[32]** 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 many-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 neural chips (such as [23,9]). 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" **[38]**, although using asynchronous operating mode can rearrange the scene.

c) Parallelized sequential computing

A major bottleneck in distributed computing is rooting also in "idle waiting", as was correctly identified decades ago [26]. One of the cores (in Fig. 5, the one



Operating diagram of parallelized sequential computing systems. One of the cores,

Fig.5 The Operating diagram of parallelized sequential computing systems. One of the cores, at position (0,0.5,0), orchestrates parallelization. Two more cores 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.

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. The core at (-0.5,0,0) is the first one. Notice that even in the timeless paradigm, the first core must 'idle wait' the sequential-only processing, plus the end of the first cycle. Given that the signal must propagate to it from the originating core, it can begin its part of computation only at the beginning of the green arrow. Similarly, the core at (1,0,0) is started in the second round. Notice that its idle waiting time is longer than that of the other core because of the looping in the initiator core, and similarly, its transmission time is also longer.

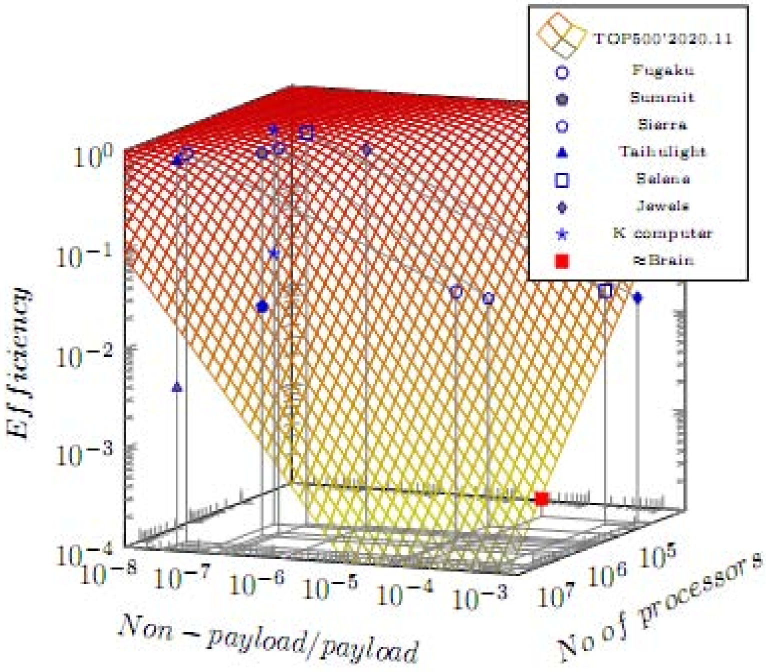


Fig. 6 The 2-parameter efficiency surface (in the function of parallelization efficiency measured by benchmark HPL and number of processing elements) as concluded from Amdahl's Law. Some sample efficiency values for some selected supercomputers are shown, measured with benchmarks HPL and HPCG, respectively. Also, the estimated efficacy of brain simulation using conventional computing is shown.

After sharing the job, all cores start to make their part of the computation. We assume that all cores need the same time to perform their part, and after that, they return the result to the organizer core. The orchestrator core must wait for the slowest fellow core; the processing time of the parallelized system is defined by the most considerable *apparent processing time*[10]. As shown in the figure, the looping (non-payload) contribution is increasing by adding more cores to the loop. Moreover, the transmission delay is increasing with the physical size of the supercomputer.

The parallelized sequential computing introduces a rule of 'adding performance' values in modern computing, which is quite similar to the rule of 'adding speed' values in modern physics **[38]**, see Fig. 7. The effect on the

[10] Notice that looping delay can be combined with propagation delay: their rational pairing enabled in the case of Sierra a 10+ % increase in its *payload performance* with 0 % increase in the nominal performance. Also, as the examples of recent world champions demonstrate, using assisting cores for both organizing the cooperation and performing other non-payload job, significantly increases their efficiency.

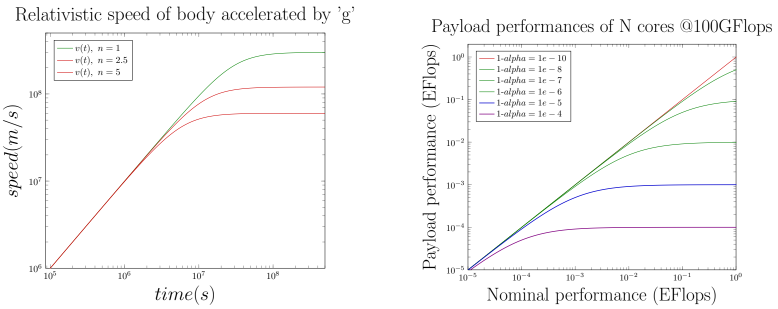


Fig. 7 The limiting effect is considered in "modern" theories. On the left side, the speed limit, as explained by the relativity theory, is illustrated. The refractory index of the medium defines the value of the speed limit. On the right side, the payload performance limit of parallelized sequential computing systems, as explained by the "modern paradigm", is illustrated. The ratio of non-payload to payload processing defines the value of payload performance.

efficiency of supercomputers is depicted in Fig. 6. This loss is natural that can be mitigated but cannot be eliminated.

Figure 6 also depicts how computing efficiencies of recent supercomputers depend (see its discussion in [**33**]) 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 known since decades that "*this decay in performance is not a fault of the architecture, but is dictated by the limited parallelism*" [26]; in excessive systems of modern HW, *is also dictated by laws of nature* [**38**].

d) Neuromorphic computing

Biology strictly considers both the physical distance and its components' operating speed: slight changes in values in their timing result in severe dysfunctionality. 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 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 the needed control signals to their place [**37**]. However, using the proposed *time-space system*, we can correctly describe the neuronal operation (that would be unsound, using the classic paradigm), too. However, we must consider that the interaction speed is different for the different components/events in their case.

In sections 5.1 and 5.3 was discussed the timely behavior of the serial bus (shared medium) and the distributed parallelized processing, respectively. The classic ANNs combine their disadvantages into one single inefficient system: *the signal transition time between neurons can be orders of magnitude higher than their processing time*. Given that, as discussed in section 4.1, *in a technological implementation of the neuronal operation, in most portion of the total time, the 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 weeks-long training, even for (compared to the functionality of the brain) straightforward 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, . . . it's the most disruptive workload from an I/O pattern perspective*"[11]. In practice, it means that imitating neuromorphic computing on conventional architectures can be performed only with very low computing efficiency [**32, 36**].

e) 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 the 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 one replaces the components with others (such as much quicker processing or storage element). *Mimicking the biology is also useful here: the time window where the decision is made*[12] *is of the same size, independently from both the path traversed by the signal (the axon length) and the signal's speed (conduction velocity). Furthermore, it is in the order of the 'processing time' of the neurons.*[13]

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 a 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 the performance of future computing.

VI. 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 us to calculate the effects 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 weeks-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 critical 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 Références Referencias

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)

[11] https://www.nextplatform.com/2019/10/30/cray-revamps-clusterstor-for-the-exascale-era/

[12] In computing: WHEN and WHERE the logical function is evaluated

[13] The biology can change the conduction velocity, which needs energy, so finding an optimum is not as simple.

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), 469–484 (2017). URL https://doi.org/10.1177/1094342017738610
5. Bourzac, K.: Streching 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. Das, A.: The special theory of relativity: a mathematical exposition, 1 edn. Springer (1993)
8. 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
9. Davies, et al, M.: Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. IEEE Micro **38**, 82–99 (2018)
10. Dongarra, J.: Report on the Fujitsu Fugaku System. Tech. Rep. Tech Report ICL-UT-20-06, University of Tennessee Department of Electrical Engineering and Computer Science (2016). URL http://bit.ly/fugaku-report
11. Einstein, A.: On the Electrodynamics of Moving Bodies. Annalen der Physik (in German) **10**(17), 891–921 (1905). DOI 10.1002/andp.19053221004
12. Godfrey, M.D., Hendry, D.F.: The Computer as von Neumann Planned It. IEEE Annals of the History of Computing **15**(1), 11–21 (1993)
13. Gustafson, J.L.: Reevaluating Amdahl's Law. Commun. ACM **31**(5), 532–533 (1988). DOI 10.1145/42411.42415