Towards Brain-Inspired System Architectures

Thomas Sterling, Maciej Brodowicz, and Timur Gilmanov(✉)

Center for Research in Extreme Scale Technologies,
School of Informatics and Computing, Indiana University, 420 N. Walnut St.,
Bloomington, IN 47404, USA
{tron,mbrodowi,timugilm}@indiana.edu

Abstract. Brain-inspired computing structures, technologies, and methods offer innovative approaches to the future of computing. From the lowest level of neuron devices to the highest abstraction of consciousness, the brain drives new ideas (literally and conceptually) in computer design and operation. This paper interrelates three levels of brain inspired abstractions including intelligence, abstract graph data structures, and neuron operation and interconnection. An abstract machine architecture is presented from which a lower bound on resource requirements for intelligence is to be derived. At the lowest level a new use of cellular automata architecture is discussed that mimics the fine-grain locality of action and high degree interconnectivity of neurons and their structures. Graph structures serve as a brain inspired intermediary abstraction between these two as the neocortex is organized as a directed graph. This paper shows how all of the pieces tie together and opens a new way of considering future computing structures through brain inspired concepts.

1 Introduction

Even as future directions of supercomputing are challenged by issues of scalability, power, reliability, and usability, the human brain demonstrates radical alternatives in technologies, structure, and operation that inspire revolutionary approaches to computing at unprecedented scales compared to contemporary computers. The brain comprises almost a hundred billion neurons; each with thousands of interconnects in less than 1500 cubic centimeters with a power budget of 20 Watts. Each neuron performs a complex algorithm a thousand times a second. Today, experts in the US, Europe, and Asia as well as other parts of the world are considering what can be derived from insights related to brain structure and operation in advancing technical approaches to goals in future Exascale computing for science, engineering, industry, commerce, the arts, and security. This paper examines three brain-inspired features of future generation computing: one abstract related to knowledge understanding and one physical to achieve effective degree and diversity of interconnectivity through semiconductor technology, both logically integrated by graph structures.

The phrase "brain-inspired" is as vague as it is provocative; both of real value. It suggests many potential opportunities and stretches the realm of possibilities well beyond conventional practices. As a result, it has motivated work

L. Grandinetti et al. (Eds.): BrainComp 2013, LNCS 8603, pp. 159–170, 2014.
DOI: 10.1007/978-3-319-12084-3_13

in a number of directions over many decades. Turing, after defining computability, prescribed a test for computer intelligence [1]. The vast networks of neurons have inspired the class of algorithms known as neural networks [2] that, among other areas, has demonstrated recent advances in natural language processing. Neural nets do not duplicate the brain but rather exhibit some properties reminiscent of and perhaps motivated by brain structures. Here, three properties of the human brain that inspire consideration of innovations in computing are identified: Consciousness, Intelligence, and Cellular Automata [3].

"Consciousness" is intuitive to everyone but lacks clarity of definition and is as much a part of philosophy as science. Therefore, it is deferred in this discussion, although tantalizing and important in the long term. Here, the latter two are examined in depth and tied to practical issues of future computing systems. "Intelligence" is addressed as a class of observable computing behavior. The project, Cognitive Real-time Interactive System (CRIS), is described to determine minimum bounds on resource requirements for intelligent systems. The generalized cellular automata hardware structure, inspired by neuron structures, is an innovative yet realistic concept to achieve advantages of low-level brain elements with future semiconductor device technologies. The Continuum Computer Architecture (CCA) project explores fine-grain hardware structures that take advantage of near nano-scale semiconductor technologies through brain-inspired physical characteristics including localized functionality and rapid result dissemination to wide array of distributed component destinations. These two brain-inspired computing forms are mutually supportive. Dynamic graph data structures are an intervening abstraction relating the two. Graphs may be considered brain-inspired as the complex topologies of the neurons of the brain are graphs. But many knowledge structures such as semantic nets and search space algorithms are manifest as graphs as well.

Intelligence is an attribute inspired by the human brain but neither defined nor limited by it. Further, not all mental attributes associated with the human brain need be ascribed to intelligence. A working definition of "intelligence" is required to guide the development and govern the operation of an intelligent system. Even if such a definition is not fully compliant with all possible interpretations, it must be viable, repeatable, testable, and realizable. Intelligence is the ability of an entity to understand its context including itself and react to it in real-time in response to intrinsic goals and derived objectives. This definition defers determination of the explicit class of entity or the nature of its context as well as the specification of its governing goals and objectives. It supports many possible manifestations of intelligent agents and their operational domain. It also implies a range of the property of intelligence, begging the question of a quantifiable metric by which to measure intelligence. Equally challenging is the pivotal verb: "to understand". The definition does establish the principal attributes of an intelligent system even if it alone fails to fully fix the meaning of key terms. Machine Intelligence is an algorithm representable in a mechanical system. Intelligence is an emergent behavior of a real-time system comprising the synergy of the distinct functional capabilities of learning, knowledge, planning, and understanding in a real-time context. The purpose of CRIS is to explore the resource requirements in time and space (memory capacity, execution elements, communication bandwidth, power) in order to realize the properties deemed essential to intelligence. The goal is to provide a quantifiable lower bound of such resources based on an abstract machine architecture comprising a synthesis of such functional elements. These are informed and inspired by understanding of the brain but not intended to duplicate brain functionality in all ways.

Conventional architectures, both individual processor cores and memory hierarchies, are becoming increasingly inadequate in terms of efficiency, scalability, power, generality, portability, and usability. But the brain inspires alternative structures; ones that exploit lightweight physical hardware structures while adapting to asynchronous operation. High connectivity over widely distributed destinations within the brain is a property rarely shared in conventional structures. The second extreme attribute is that of hyper-parallelism where each primitive element is capable of some independent and complex operation. Conventional systems do neither; the brain is exceptional at this at the level of neuron structures. Future nano-scale semiconductor technologies will favor tighter coupling for closer interaction while die-scale structures are loosely coupled and display asynchronous interaction. Cellular automata embodies many of the properties of neuron structures. However, conventional cellular automata are special purpose with interactions limited to nearest neighbor. CCA suggests an alternative version of cellular automata in which localized actions have global destinations through packet switching abstractions rather than line switched. Like neurons, many messages can be sent to different destinations. The unifying principle of graphs as the intermediate form representing both the abstraction of knowledge, planning, and searching for CRIS naturally lends itself to a new generation of implementation by CCA. This paper examines these levels of brain-inspired abstractions and their interrelationships for future computing systems.

2 Overview of Machine Intelligent System

Over the last six decades attempts to deliver a cognitive system that is capable of learning have been made. In spite of advances in natural language processing, planning, pattern matching, robotics, and other related disciplines, a truly cognitive system has not been delivered. With rapid technological advancements and digital information increase, new algorithms, utilizing both hardware and the available information, are being developed. Although seemingly capable of delivering certain elements of brain-inspired behavior, these algorithms lack in the critical component of being able to learn new concepts (self-adjust the algorithmic behavior of self) except in special cases.

2.1 Abstract Architecture for Machine Intelligence

An architecture that represents a Machine Intelligent (MI) system is comprised of a number of interrelated autonomous components, each of which serves key

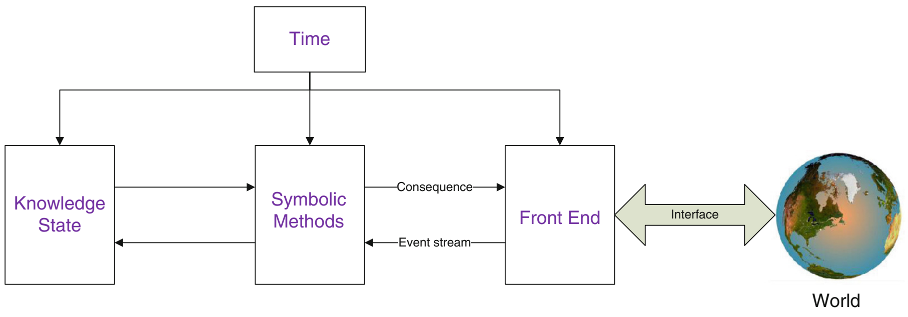

Fig. 1. Abstract architecture of a MI system.

functions in the entire system operation. These components can be divided into three groups at the top level, including the system's knowledge state, symbolic methods and front-end, see Fig. 1.

Knowledge state represents various types of knowledge that the system is continuously maintaining throughout its execution, while symbolic methods include a number of means for processing the accumulated knowledge as well as any other symbolic-based data. Finally, the front-end provides means for the system to exchange information with the outside world.

Such a system represents a closed execution loop by obtaining data stream at its front-end, which is then processed to obtain decisions about updating the knowledge state. This updated state is later used to provide an output stream to an external entity, which is referred to as the "World" in Fig. 1.

Knowledge is an essential part of any Machine Intelligence system that requires a well-defined representational hierarchy. Alternative knowledge representation techniques exist, including knowledge databases and ontology representation [4]. The proposed architecture differentiates knowledge into a number of classes beginning from relatively static foundational knowledge to most rapidly changing imperatives derived from input stream commands or otherwise implied by local context. Knowledge is further broken into universal knowledge, that any system might possess, such as fundamental facts about geography, history or physics, and unique knowledge that any system belonging to a particular environment needs to maintain, such as the machine's location, its internal system status, or what other agents it is currently interacting with in its surroundings.

One of the most important properties of the proposed MI system is that it has to be self-aware. A topic of self-awareness that goes back a few centuries has been a subject of discussion in the fields of psychology and philosophy. A definition of "self-aware" is proposed. Such systems need to identify themselves in the surrounding environment and recognizing what kind of entities (including other MI systems) it is presently interacting with. It is important for the system to model such aspects as identity (who am I?), physical location in space (where am I?), location in time (when am I?), and operation status (how am I?). Additionally, the system has to be aware of any entities in its neighborhood that can affect it, with whom/what it is presently maintaining a dialog, whether it is challenged by anything, and what are its current responsibilities alongside with how well it is advancing in achieving its goals. The system's objective function hierarchy models the latter, which is represented by a stack of goals to be satisfied. These goals are sorted by the complexity, with more coarse-grained declarative objectives located at the top of the hierarchy, and with fine-grained imperatives at the bottom.

The objective function module interacts closely with the active context stack that contains transient information about anything the system is interacting with, including knowledge, external agents, etc. Every new situation, event, objective or interaction requires a context, which is added to the active context hierarchy. Items in this hierarchy are linked to the corresponding objective function items. Due to the unstructured nature of processed information, it is possible that some of this relationships are "one:many", "many:one, or even "one:none" and "none:one". The active context stack is expected to become extremely dynamic at times when there is a need to process a large amount of fine-grained contextual information in the real-time regime. The stack, therefore, is growing and shrinking rapidly depending on the active contexts.

A mechanism that drives the knowledge query and update is referred to as Master Control Program (MCP). This mechanism is analogous to central nervous system for autonomic behavior. It ensures that a system is constantly going through an outer loop, executing required actions in bounded time. During the execution of a loop cycle, the MCP needs to perform a number of actions, such as (i) query the system's status by interacting with the self-awareness module; (ii) satisfy the objectives that are due this execution cycle; (iii) query all the I/O sensors to obtain updated information about the external environment; (iv) trigger the knowledge state update mechanisms, and (v) update current objective function and active context stack hierarchies. When satisfying the objectives, MCP needs to obey the axioms that are imposed by the developers and are hard-coded into the system's non-volatile read-only memory. This approach is necessary in order to guarantee that the system will not become dangerous to human beings under any circumstances. These rules can not be altered by the system itself and would require external interference (if need be). During the cycle of the execution loop, a number of symbolic methods are involved, which are described below.

The system's symbolic methods include learning, update mechanisms, planning, inference and conflict resolution. All of these methods are independent actors, that are operating on required tasks, taking current context information into consideration. Learning is one of the most essential methods allowing the system to modify its future actions based on the empirical information about its interaction with the environment. The update mechanisms ensure that the new information is processed and recorded in the machine's knowledge state base. Planning module supports for a plan derivation for various tasks. In case a machine does not know what type of metrics to use, or has to achieve a goal while generating a plan, it is attempting to utilize learning modules in order to express the goals through currently existing knowledge or obtain new knowledge about the problem. Finally, inference mechanisms allow to draw new conclusions about the facts available in the knowledge state, while conflict resolution ensures that all the constraints of present goals are satisfied.

The nature of brain-inspired computation and its processing similarities of those in human brain suggest the use of data structure that would resemble the brain's behavior. One such structure, that naturally represents neurons and their synapses, is graphs. Graph structures can be used for both knowledge storage, with facts of various types forming a tree-like conceptual hierarchy, and context and objective stack information. The information processed by the system varies greatly in terms of its life span, with some knowledge facts that are virtually never changed (such as axioms, or very slowly changing facts about the world) on one hand and pieces of knowledge that are extremely dynamic, such as active context stack frames, or objective functions, on the other. Graph data structures, that are capable of efficient representation and processing for both types of these knowledge types will be utilized. Algorithms that support extremely dynamic graph structures need to be considered.

2.2 Quantifying Metrics for Machine Intelligence

Taking the human brain as a reference Machine Intelligent system, a question about a rough estimate of required resources for an engineered MI system arises. How large of a system would be required in order to support Machine Intelligence? In fact, how should one measure such a system and what kind of units should be used? For that purpose, a set of metrics need to be considered. The introduced metrics need to be able to accurately measure some of the components of interest for a MI system, such as operation primitives. These primitives include I/O, knowledge state update, and query. Another set of important components include concurrency, storage, as well as energy and power.

A few examples of various metrics that will be used for the experiments include computational throughput, memory capacity, storage capacity, communication latencies, bi-section bandwidth interconnect, and energy consumption rate. The proposed metrics will be utilized to measure the performance of a simulation for the MI system.

3 Continuum Computer Architecture

The Continuum Computer Architecture is a 3-D computing medium intended to efficiently and scalably support the cognitive algorithms and data structures utilized by a machine intelligence system. This architecture is inspired by prominent brain properties that are immediately applicable to graph processing. In the sections below challenges of designing the knowledge processing hardware are described, the brain features that inspired particular facets of CCA are pointed out, trade-offs that define the design constraint space are analyzed, and the resultant system architecture is described.

3.1 Challenges of Knowledge Processing

Creation of an efficient graph processing component is associated with significant technological challenges. Current digital processors are designed for diametrically different purposes; they achieve high computational throughputs on numeric workloads in which data references exhibit high levels of locality. However, their performance drops off significantly when data access becomes unpredictable and sparse – such as graph operations. Graph processor implementation for machine intelligence that simultaneously satisfies the requirements of scale, response time, and energy using electronic processing technology available today is unfeasible. Superficial analysis of brain structure reveals that the count of synaptic connections is on the order of 10^4 for each neuron. This is more than an order of magnitude greater than the highest number of ports available in a low-level switch component of a state-of-the-art high-radix network. The neuron produces action potential (fires) at the rate of few hundred pulses per second. Its underlying electric activity consists of ionic current summation and integration on the capacitance of lipid bilayer coupled with the electrical properties of the axon [5]. With approximately a hundred billion neurons in a human brain, each with thousands of synaptic connections with complex chemistry, the aggregate operation rate necessary to effectively emulate brain activity may easily reach 10^{18} (1 Exa) per second. Since the brain operates on a power budget of roughly a fifth of that of a modern high-performance CPU dissipated in a volume of just over 1 l, the challenges of building an artificial knowledge processor capable of rivaling the human brain are nothing short of staggering.

3.2 Characteristic Features of the CCA

Brain-Inspired Properties. The brain exhibits a high degree of replication; even though there are various types of neurons, the overall brain structure is attained through extensive repetition of few similar “building blocks”. CCA takes advantage of this by defining a minimal computing element, *fonton*, that connects with other elements in its immediate vicinity. The fonton is small enough to be implemented in a few thousand logic gates, reaching a diameter comparable to that of the average neuron (few tens of micrometers) in a modern CMOS process.

The neuron combines several functions: connectivity, analog signal processing, and even storage (accumulation of ionic charge). The fonton mirrors this by embedding processing logic, local registers, and a network router along with physical communication links. This approach avoids the pitfalls of discrete functional units connected by power-hungry buses with discrete memory modules and separate network hardware. The bandwidth available for each of the fonton components can be optimized to match the individual processing throughputs; their interactions may happen within a single clock cycle.

CCA inherits from the brain distributed control and quasi-independent operation. Due to limited amount of processing a single neuron can perform, every significant brain function requires formation and activation of ensembles of neurons. One of the side benefits of this is operational redundancy, in which functions affected by minor neural damage might be assumed by nearby healthy neurons. Due to inevitable production defects and component degradation affecting complex systems, this also introduces the necessary fault tolerance for contiguous operation of the CCA system. Even though some brain structures are developed for specific functionality (such as cerebellum with fast feed-forward neural paths optimized for motor control), many of them are largely interchangeable. CCA mimics it by providing uniform distribution of identical processing elements throughout the chip, forming a nearly isotropic computing medium throughout which higher level functions may be distributed arbitrarily.

Finally, the cerebral cortex, considered to be the locus of intelligence, is a stratified, but primarily 2-D structure. Its thickness is thought to be critically related to cognitive reasoning abilities. The CCA trivially replicates the 2-D layout with transistors instantiated on the surface of a flat piece of silicon. However, recently introduced die stacking [6] enables building in the third dimension.

Emergent Behavior. Unlike the brain, a CCA machine cannot redefine its physical connections over time. However, packet switching network permits logical organization of arbitrary aggregations of fontons, as long as routing is efficient. Packet switching is also a foundation of message-driven computing, in which parts of a program react to specific events rather than actively waiting or polling for them. This results in improved energy efficiency, as inactive software components may stay idle and minimize their resource footprint until a triggering event occurs. Since the speed of message packets in the medium is finite, locality and spatial distribution of the individual program components matter.

Even though the physical implementation of CCA hardware is envisioned as 3-D, this does not impose limits on logical dimensionality. Three dimensions provide natural simulation medium for many physical phenomena. However, logical organization of the interconnecting graph may be arbitrary. Moreover, since this connectivity is defined by routing tables modifiable by software, it can be shaped during the program execution.

The high degree of replication coupled with co-location of basic functions in all elementary components provides unprecedented aggregate processing bandwidth. Assuming a clock speed of 1 GHz and a conservative 10,000 fontons per die, the peak memory bandwidth achieves 240 TB/s using 8-byte wide register banks supporting two concurrent reads and one write access per cycle. For computations, a peak of 10 Tera-ops per die is possible.

Trade-Offs. Embedding the graph processor on a CCA platform offers a number of advantages with respect to in-brain processing. Unfortunately, nearly all of them are subject to trade-offs that reduce their effectiveness:

Speed: While the firing cycle of a neuron is measured in milliseconds, synchronous logic may be clocked at gigahertz frequencies, possibly higher if local clock domains are constrained to individual fontons or more exotic technologies are applied (Josephson junctions [7,8], quantum dots [9]). In CMOS process power dissipation increases with clock speed, often forcing the thermal constraints on practical designs even before technological clock limits are reached.

Scale: Unlike the brain, the volume of a CCA system is not restricted to that of the enclosing cranial cavity. However, crossing the die boundary may be associated with substantial performance sacrifice. Even 3-D stacking alone may reduce the number of closely interconnected fontons due to manufacturing rules. Increasing the overall processing power by connecting multiple 3-D stacks is possible, but results in a non-homogeneous structure. As the largest mass produced dies rarely exceed 500 mm^2, manufacturing a homogeneous CCA system the size of human brain is difficult with available technology.

Connectivity: The ability to define and manipulate network properties in software is very appealing, but it still has to be mapped onto the 3-D physical mesh. This may result in increased diameter (counted in the number of inter-fonton links) of the implementation of a particular function. Fontons that act as routing intermediaries may have to be added to the resource pool. Increased diameter directly impacts the average number of communication hops a message must traverse on a path to destination, increasing latency, response time, and potentially energy consumed by the computation.

Energy: The power draw of a CCA implementation may be approximated through the analysis of current GPU designs, since they consist of highly replicated small processing cores operating at close to 1 GHz. Thus, a 10,000-fonton die would use approximately 90 W to power 1.5 billion transistors on a 12×12 mm chip manufactured in 28 nm process. This transistor count already provides for additional structures supporting off-die I/O and vertical interconnect. Stacking 10 dies brings the fonton count to 100,000 per structure in a volume just over 1 cm^3, raising the power envelope to close to a kilowatt (!). Assuming 50 % loss of volume for power delivery, cooling, network, and structural enforcement, a brain-sized machine with peak performance of 65 Peta-ops would dissipate close to 0.6 MW, demanding cooling water flow of about 15 liters per second (10 K coolant temperature raise over ambient).

3.3 Architecture and Principles of Operation

Fonton. Figure 2 depicts the elementary building block of a CCA system. Its main components include minimal ALU, associative register file, and network interface. The processing within a fonton is controlled by a local state machine that coordinates the flow of requests that originate internally as well as those arriving from the network. There is no notion of explicit program counter; the operands necessary to execute a sequence of micro-operations are bundled with the relevant instructions to form a *token*, an atomic execution unit. Token execution modifies internal state of a fonton and potentially results in emission of tokens targeting the state of remote fontons.

The fonton is equipped with a minimal ALU that supports integer operations, along with pattern matching and bitwise permutations. Floating-point and extended integer arithmetic are higher-level functions that are synthesized

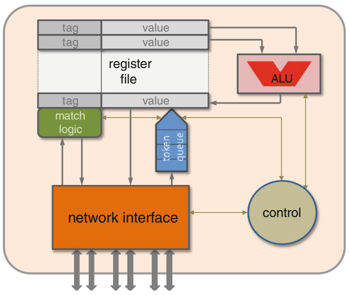

Fig. 2. *Fonton* element.

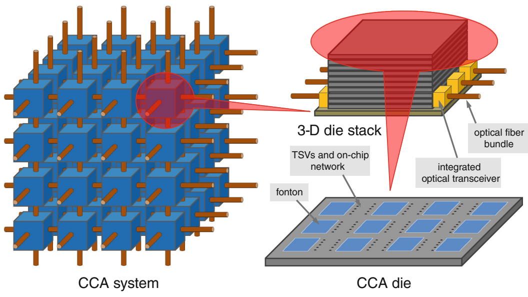

Fig. 3. Breakdown of CCA system components.

using fonton groups in adjacent locations. Although potentially slower, this provides unparalleled flexibility of matching the precision, resource footprint, and energy requirements to specific application.

Since physical memory is distributed across the system, traditional addressing schemes don't apply. Instead, fontons explicitly store associative tags along with the memory contents. The tags are unique for each entry, but also implement a form of wildcard addressing. As the location of fonton containing specific tag is not known a priori, part of the system manages a distributed address resolution service for non-local memory accesses by storing routing information in register files of predetermined fontons.

The network interface consists of six bidirectional links (two per dimension), connecting the neighboring fontons. A token packet, if traffic conditions permit, is sent over a link in a single cycle. The network interface can perform associative lookup on register tags to identify whether a token's target is local (the token is absorbed) or whether fonton contains related routing information (token's movement is modified).

Scaled Structures. Designing systems with large fonton counts requires special consideration (Fig. 3). Die stacking is a viable way to improve the resource count while preserving the homogeneous makeup of the device. While the inter-die interconnect has a different characteristics from that of the on-chip network, it uses matching technology – electrical signaling (for example, using Through-Silicon Vias [10]). Noting that the peak aggregate bandwidth of 2 PB/s is necessary to accommodate the maximum token flux through the stack boundary (256-byte packets assumed), a radically different approach is needed, such as on-die photonics with fiber optic links. Recent bandwidth record of just over 1 Pbit/s achieved in a 12-mode fiber [11] confirms the necessity of further development.

Operation. The remarkable similarity between the heavily cross-linked CCA hardware and vertex sets connected by edges of a graph helps efficiently map graph data structures onto execution resources. For small vertices, fontons provide sufficient storage to encode their local state and neighbor information;

otherwise, fontons are clustered to store complex node state. Token based operation is vital in implementation of graph algorithms. For example, searching a graph for vertices with certain properties would be inefficient if only one token is emitted per cycle for each link in a high-degree node. Instead, a parallel traversal may be initiated, in which tokens propagate as a 3-D "wavefront" with sufficient number of packets instantiated within few cycles after the launch of operation. This is typical of the brain as well.

4 Conclusions

Brain-inspired computing informs innovations in form and function for future generations of computing systems. Brains represent among the most complex systems known. They are exemplars of density, energy efficiency, performance, interconnectivity, 3-D structures, heterogeneity, hierarchical structures, real-time operation, intelligence, and self-aware behavior, as well as consciousness. Each of these conveys possible new concepts that may influence aspects of future computer system design and methods of operation. Brain-inspired computing is not the duplication of the brain but rather the borrowing of concepts derived from nature that suggest alternative approaches from those conventionally applied to computing. This paper has examined three general facets of brain-inspired computing structure and operation at three corresponding layers of abstraction. These are the high-level of the emergent behavior referred to as "intelligence", the low-level physical element and structure inspired by neurons, and the intermediate abstraction of the dynamic graph data structure, which is how the neurons of the brain are organized.

The CRIS project is exploring the high level abstraction of intelligence to provide a lower bound of the total resources required to achieve the functionality of one possible definition of "intelligence". The CRIS abstract architecture is defined to reflect functionality associated with and inspired by the behavior of the human brain rather than the emerging understanding of the physical distribution within the brain of distinguishable behavioral properties. While the CRIS architecture may not fully achieve intelligence, all functional components understood to contribute to intelligence is incorporated and therefore needs to be supported in real time. Therefore, analysis of means of implementing the functions and their respective duty cycles will yield the lower bound resource (time and space) assessment. The target abstract architecture, to which such high-level functionality is assumed to be implemented, is a graph-processing engine, the intermediate level brain-inspired concept.

The CCA project is exploring low-level implementation details of an innovative application of cellular automata to reflect brain-inspired neuronal properties and architecture. The key attributes to be considered are the localization of primitive operations within the separate basic components, the high degree connectivity of each component for input and output, event-driven operation, and graph topologies of structures. Conventional components do not work well in this class of high-density structures. Classical cellular automata are high density but physically of very limited interconnectivity physically and logically constrained to nearest neighbor. CCA extends the logical connectivity potentially to many orders of magnitude through packet switching treating the cellular fabric both as local operational units and global interconnectivity. CCA demonstrates that abstract graph structures can be implemented with neuronal-like packet switched components. This can provide a brain-inspired hardware implementation of the graph abstraction needed for the high-level also brain-inspired abstraction of intelligence. The implication of this work is that there are many and interrelated ways in which our emerging understanding of the brain is yielding new practical concepts for innovations in high performance computing.

References

1. Turing, A.M.: Computing Machinery and Intelligence. Mind **59**(236), 433–460 (1950)
2. Haykin, S.: Neural Networks: A Comprehensive Foundation, 1st edn. Prentice Hall PTR, Upper Saddle River (1994)
3. Toffoli, T., Margolus, N.: Cellular Automata Machines: A New Environment for Modeling. Scientific Computation. MIT Press, Cambridge (1987). http://books.google.com/books?id=HBlJzrBKUTEC
4. Doherty, P., Lukaszewicz, W., Skowron, A., Szalas, A.: Knowledge Representation Techniques - A Rough Set Approach. Studies in Fuzziness and Soft Computing, vol. 202. Springer, Heidelberg (2006)
5. Hodgkin, A.L., Huxley, A.F.: A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Physiol. **117**(4), 500–544 (1952)
6. Topol, A.W.: Three-dimensional integrated circuits. IBM J. Res. Dev. **50**(4), 491–506 (2006)
7. Josephson, B.D.: Possible new effects in superconductive tunnelling. Phys. Lett. **1**, 251 (1962)
8. Likhariev, K.K., Semenov, V.K.: RSFQ logic/memory family: a new Josephson-junction digital technology for sub-terahertz-clock-frequency digital system. IEEE Trans. Appl. Supercond. **1**, 3–28 (1991)
9. Lent, C.S., Tougaw, P.D.: A device architecture for computing with quantum dots. Proc. IEEE **85**, 541–557 (1997)
10. Black, B., Annavaram, M., Brekelbaum, N., DeVale, J., Jiang, L., Loh, G.H., McCauley, D., Morrow, P., Nelson, D.W., Pantuso, D., Reed, P., Rupley, J., Sadasivan, S., Shen, J., Webb, C.: Die stacking (3D) microarchitecture. In: Proceedings of the 39^{th} Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-39, pp. 469–479, December 2006
11. NTT Corp.: World record one petabit per second fiber transmission over 50 km: equivalent to sending 5,000 HDTV videos per second over a single fiber. Press release, September 2012