This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TrueNorth: Design and Tool Flow of a 65mW 1 Million Neuron Programmable Neurosynaptic Chip

Filipp Akopyan[1], Jun Sawada[1], Andrew Cassidy[1], Rodrigo Alvarez-Icaza[1], John Arthur[1], Paul Merolla[1], Nabil Imam[1], Yutaka Nakamura[2], Pallab Datta[1], Gi-Joon Nam[1], Brian Taba[1], Michael Beakes[1], Bernard Brezzo[1], Jente B. Kuang[1], Rajit Manohar[3], William P. Risk[1], Bryan Jackson[1], and Dharmendra S. Modha[1]

[1] IBM Research, U.S.A.

{akopyan, sawada, andrewca, arodrigo, arthurjo, pameroll, ni49, pdatta, gnam, btaba, beakes, brezzo, kuang, risk, bryanlj, dmodha}@us.ibm.com

[2] IBM Research, Japan

xyutaka@jp.ibm.com

[3] Cornell University, U.S.A.

rajit@csl.cornell.edu

***Abstract*—The new era of cognitive computing brings forth the grand challenge of developing systems capable of processing massive amounts of noisy multi-sensory data. This type of *intelligent* computing poses a set of constraints, including real-time operation, low power consumption and scalability, which require a radical departure from conventional system design. Brain-inspired architectures offer tremendous promise in this area. To this end, we developed TrueNorth, a 65mW real-time neurosynaptic processor that implements a non-von Neumann, low-power, highly-parallel, scalable, defect-tolerant architecture. With 4096 neurosynaptic cores, the TrueNorth chip contains 1 million digital neurons and 256 million synapses tightly interconnected by an event-driven routing infrastructure. The fully digital 5.4 billion transistor implementation leverages existing CMOS scaling trends, while ensuring one-to-one correspondence between hardware and software. With such aggressive design metrics and the TrueNorth architecture breaking path with prevailing architectures, it is clear that conventional CAD tools could not be used for the design. As a result, we developed a novel design methodology that includes mixed asynchronous-synchronous circuits and a complete tool flow for building an event-driven, low-power neurosynaptic chip.**

The TrueNorth chip is fully configurable in terms of connectivity and neural parameters to allow custom configurations for a wide range of cognitive and sensory perception applications. To reduce the system's communication energy, we have adapted existing application-agnostic VLSI CAD placement tools for mapping logical neural networks to the physical neurosynaptic core locations on the TrueNorth chips. With that, we have successfully demonstrated the use of TrueNorth-based systems in multiple applications, including visual object recognition, with higher performance and orders of magnitude lower power consumption than the same algorithms run on von Neumann architectures. The TrueNorth chip and its tool flow serve as building blocks for future cognitive systems, and give designers an opportunity to develop novel brain-inspired architectures and systems based on the knowledge obtained from this work.

I. Introduction

The remarkable brain, structured with 100 billion parallel computational units (neurons) closely coupled to local memory (1,000-10,000 synapses per neuron) and high-fanout event-driven communication, attains exceptional energy efficiency (consuming only 20W), while achieving unparalleled performance on perceptual and cognitive tasks. The brain's performance arises from its massive parallelism, operating at low-frequency (10Hz average firing rate) and a low power density of 10mW/cm^2, in contrast to contemporary multi-Gigahertz processors with a power density of 100 W/cm^2. The TrueNorth architecture [1], depicted in Fig. 1, is the result of approximating the structure and form of organic neuro-biology within the constraints of inorganic silicon technology. It is a platform for low-power, real-time execution of large-scale neural networks, such as state-of-the-art speech and visual object recognition algorithms [2].

The TrueNorth chip, the latest achievement of the DARPA SyNAPSE project, is composed of 4096 neurosynaptic cores tiled in a 2D array, containing an aggregate of 1 million neurons and 256 million synapses. It attains a peak computational performance of 58 Giga-Synaptic Operations Per Second (GSOPS) and computational energy efficiency of 400 Giga-Synaptic Operations Per Second per Watt (GSOPS/W) [3]. We have deployed the TrueNorth chip in 1, 4, and 16-chip systems, as well as on a compact 2"×5" board for mobile applications. The TrueNorth native Corelet language, the Corelet Programming Environment (CPE) [4], and an ever-expanding library of composable algorithms [5] are used to develop applications for the TrueNorth architecture. Prior project achievements include the largest long-distance wiring diagram of the primate brain [6], which informed our architectural choices, as well as a series of simulations using the scalable Compass software simulator, which implements the TrueNorth logical architecture. These simulations scaled from "mouse-scale" and "rat-scale" on Blue Gene/L [7] to "cat-scale" on Blue Gene/P [8] to "human-scale" on LLNL's Sequoia 96-rack Blue Gene/Q [9,10].

Inspired by the brain's structure, we posited the TrueNorth architecture based on the following seven principles. Each principle led to specific design innovations in order to efficiently reduce the TrueNorth architecture to silicon.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

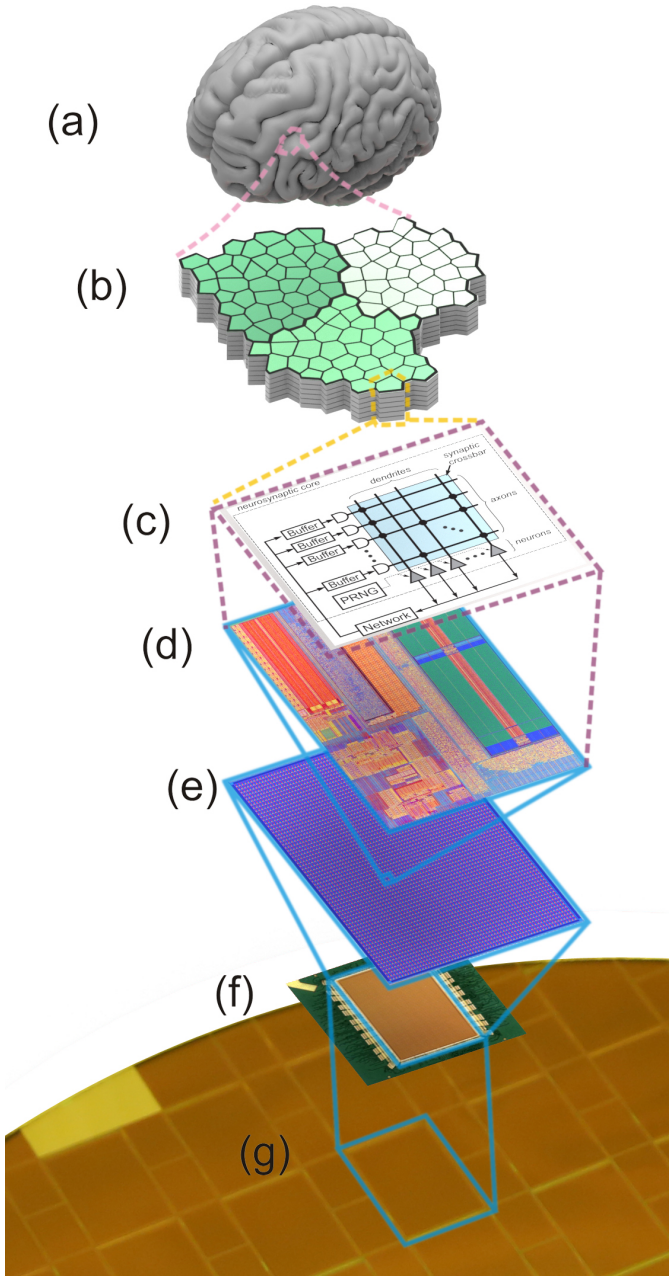

Fig. 1: The TrueNorth architecture is inspired by the structure and function of the human brain (a) and the cortical column (b), a small region of densely interconnected and functionally related neurons that form the canonical unit of the brain. Analogously, the neurosynaptic core (c) is the basic building block of the TrueNorth architecture, containing 256 input axons, 256 neurons, and a 64k synaptic crossbar. Composed of tightly coupled computation (neurons), memory (synapses), and communication (axons and dendrites), one core occupies $240\mu m \times 390\mu m$ of silicon (d). Tiling 4096 neurosynaptic cores in a 2D array forms a TrueNorth chip (e), which occupies 4.3 cm^2 in a 28nm low-power CMOS process (f) and consumes merely 65mW of power while running a typical computer vision application. The end result is an efficient approximation of cortical structure within the constraints of an inorganic silicon substrate (g).

1) **Minimizing active power**: Based on the sparsity of relevant perceptual information in time and space, TrueNorth is an event-driven architecture using asynchronous circuits as well as techniques to make synchronous circuits event-driven. We eliminated power-hungry global clock networks, collocated memory and computation (minimizing the distance data travels), and implemented sparse memory access patterns.
2) **Minimizing static power**: The TrueNorth chip is implemented in a low-power manufacturing process and is amenable to voltage scaling.
3) **Maximizing parallelism**: To achieve high performance through 4096 parallel cores, we minimized core area by using synchronous circuits for computation and by time-division multiplexing the neuron computation, sharing one physical circuit to compute the state of 256 neurons.
4) **Real-time operation**: The TrueNorth architecture uses hierarchical communication, with a high-fanout crossbar for local communication and a network-on-chip for long-distance communication, and global system synchronization to ensure real-time operation. We define real-time as evaluating every neuron once each millisecond, delineated by a 1kHz synchronization signal.
5) **Scalability**: We achieve scalability by tiling cores within a chip, using peripheral circuits to tile chips, and locally generating core execution signals (eliminating the global clock skew challenge).
6) **Defect Tolerance**: Our implementation includes circuit-level (memory) redundancies, as well as disabling and routing around faulty cores at the architecture-level to provide robustness to manufacturing defects.
7) **Hardware-software one-to-one equivalence**: A fully digital implementation and deterministic global system synchronization enable this contract between hardware and software. These innovations accelerated testing and verification of the chip (both pre- and post-tapeout), as well as enhance programmability, such that identical programs can be run on the chip and simulator.

Realizing these architectural innovations and complex unconventional event-driven circuit primitives in the 5.4 billion transistor TrueNorth chip required new, non-standard design approaches and tools. Our main contribution, in this paper, is a novel hybrid asynchronous-synchronous flow to interconnect elements from both asynchronous and synchronous design approaches, plus tools to support the design and verification. The flow also includes techniques to operate synchronous circuits in an event-driven manner. We expect this type of asynchronous-synchronous tool flow to be widely used in the future, not only for neurosynaptic architectures but also for event-driven, power efficient designs for mobile and sensory applications.

To implement efficient event-driven communication circuits, we used fully custom asynchronous techniques and tools. In computational blocks, for rapid design and flexibility, we used a more conventional synchronous synthesis approach. To

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

mitigate testing complexity, we designed the TrueNorth chip to ensure that the behavior of the chip exactly matches a software simulator [9], spike to spike, using a global synchronization trigger. For deployment, we adapted existing VLSI placement tools in order to map logical neural networks to physical core locations on the TrueNorth chip for efficient run-time operation. The end result of this unconventional architecture and non-standard design methodology is a revolutionary chip that consumes orders of magnitude less energy than conventional processors which use standard design flows.

In this paper, we present the design of the TrueNorth chip and the novel asynchronous-synchronous design tool flow. In Section II, we review related neuromorphic chips and architectures. In Section III, we give an overview of the TrueNorth architecture. In Section IV, we describe our mixed asynchronous-synchronous design approach. In Section V, we provide the chip design details, including the neurosynaptic core and chip periphery, as well as their major components. In Section VI, we discuss our design methodology, focusing on interfaces, simulation, synthesis, and verification tools. In Section VII, we report measured results from the chip. In Section VIII, we present TrueNorth platforms that we built and are currently using. In Section IX, we demonstrate some example applications. In Section X, we describe tools for mapping neural networks to cores. In Sections XI and XII, we propose future system scaling and present our conclusions.

II. Related Work

Focusing on designs that have demonstrated working silicon [11], here are some recent neuromorphic efforts.

The SpiNNaker project at the University of Manchester is a microprocessor-based system optimized for real-time spiking neural networks, where the aim is to improve the performance of software simulations [12]. SpiNNaker uses a custom chip that integrates 18 microprocessors in 102mm^2 using a 130nm process. The architecture, being fully digital, uses an asynchronous message passing network (2D torus) for inter-chip communication. Using a 48-chip board, SpiNNaker has demonstrated networks with hundreds of thousands of neurons and tens of millions of synapses, and consumes 25–36W running at real-time [13].

The Neurogrid project at Stanford University is a mixed analog-digital neuromorphic system, where the aim is to emulate continuous-time neural dynamics resulting from the interactions among biophysical components [14]. Neurogrid is a sixteen-chip system organized in a tree routing network, where neurons on any chip asynchronously send messages to synapses on other chips via multicast communication [15] and a local analog diffuser network. Long-range connections are implemented using a FPGA daughterboard. Each chip in the system is 168mm^2 in a 180nm process, and has a 2D array of 64K analog neurons. Neurons on the same chip share the same parameters (for example, modeling a specific cell type in a layer of cortex). In total, Neurogrid simulates one million neurons in real-time for approximately 3.1W (total system power including the FPGA).

The BrainScaleS project at the University of Heidelberg is a mixed analog-digital design, where the goal is to accelerate the search for interesting networks by running $1,000$ to $10,000$ times faster than real-time [16]. BrainScaleS is a wafer-scale system (20cm diameter in 180nm technology) that has 200 thousand analog neurons and 40 million addressable synapses, and is projected to consume roughly 1kW. The architecture is based on a network of High Input Count Analog Neural Network modules, each with 512 neurons and 128K synapses in 50mm^2. The communication network at the wafer scale is asynchronous, and connects to a high performance FPGA system using a hierarchical packet-based approach.

Other mixed analog-digital neuromorphic approaches include IFAT from University of California San Diego [17], and recent work from Institute for Neuroinformatics [18]. Other work has taken a fully synchronous digital approach [19]. See [11] for one comparative view of recent neuromorphic chip architectures.

To the best of the author's knowledge, no previous work has demonstrated a neuromorphic chip in working silicon at a comparable scale to the TrueNorth chip (1 million neurons and 256 million synapses) with comparable power efficiency (65mW power consumption). This differentiation is realized by our circuit design emphasis on minimizing active/static power, and compact physical design for increasing parallelism. Also hardware-software one-to-one equivalence is a major advantage for testing and application development, a feature that is infeasible for systems based on analog circuits.

III. The TrueNorth Architecture

The TrueNorth architecture is a radical departure from the conventional von Neumann architecture. Unlike von Neumann machines, we do not use sequential programs that map instructions into a linear memory. The TrueNorth architecture implements spiking neurons coupled together by the network connecting them. We *program* the chip by specifying the behavior of the neurons and the connectivity between them. Neurons communicate with each other by sending *spikes*. The communicated data may be encoded using the frequency, time, and spatial distribution of spikes.

We designed the TrueNorth architecture to be extremely parallel, event-driven, low-power, and scalable, using a *neurosynaptic core*, as the basic building block of the architecture. The left side of Fig. 2 shows a bipartite graph of neurons, which is a small section of a larger neural network. A TrueNorth core, shown on the right side, is a hardware representation of this bipartite graph of neurons, with arbitrary connectivity between the input and output layers of the graph. A neurosynaptic core contains both the computing elements, neurons, for computing the *membrane potential* using various neuron models, and the memory to store neuron connectivity and parameters. By physically locating the memory and the computation close to each other, we localize the data movement, increasing efficiency, and reducing the power consumption of the chip. By tiling 4096 neurosynaptic cores on a TrueNorth chip, we

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

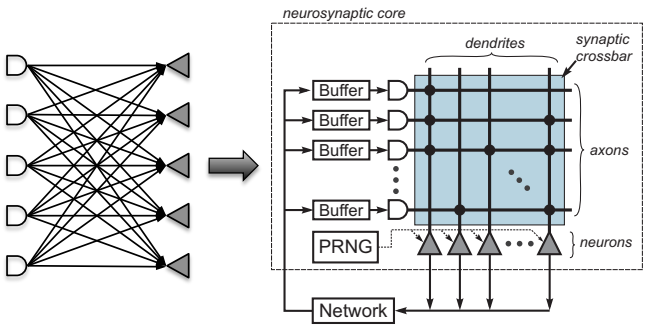

Fig. 2: A bipartite graph of a neural network (left), with arbitrary connections between axons and neurons, and the corresponding logical representation of a TrueNorth core (right).

scale up to a highly-parallel architecture, where each core implements 256 neurons and 64k synapses.

As shown in Fig. 2, a single neurosynaptic core consists of input buffers that receive spikes from the network, *axons* which are represented by horizontal lines, *dendrites* which are represented by vertical lines, and *neurons* (represented by triangles) that send spikes into the network. A connection between an axon and a dendrite is a *synapse*, represented by a black dot. The synapses for each core are organized into a *synaptic crossbar*. The output of each neuron is connected to the input buffer of the axon that it communicates with. This axon may be located in the same core as the communicating neuron or in a different core, in which case the communication occurs over the routing network.

The computation of a neurosynaptic core proceeds according to the following steps:

1) A neurosynaptic core receives spikes from the network and stores them in the input buffers.
2) When a 1kHz synchronization trigger signal called a *tick* arrives, the spikes for the current tick are read from the input buffers, and distributed across the corresponding horizontal axons.
3) Where there is a synaptic connection between a horizontal axon and a vertical dendrite, the spike from the axon is delivered to the neuron through the dendrite.
4) Each neuron integrates its incoming spikes and updates its membrane potential.
5) When all spikes are integrated in a neuron, the leak value is subtracted from the membrane potential.
6) If the updated membrane potential exceeds the threshold, a spike is generated and sent into the network.

All the computation must finish in the current tick, which spans 1ms. Although the order of spikes arriving from the network may vary due to network delays, core computation is deterministic within a tick due to the input buffers. In this sense, the TrueNorth chip matches one-to-one with the software simulator, which is one of the advantages of using a fully digital neuron implementation.

The membrane potential $V_j(t)$ of the j^{th} neuron at tick t is computed according to the following simplified equation:

$$V_j(t) = V_j(t-1) + \sum_{i=0}^{255} A_i(t) w_{i,j} s_j^{G_i} - \lambda_j.$$

Here, $A_i(t)$ is 1 if there is an incoming spike (stored temporarily in the buffer) on the i^{th} axon at time t, and $w_{i,j}$ is 1 if the i^{th} axon is connected to the j^{th} dendrite. Otherwise, the values are 0. We assign one of four types to each axon, and designate the type of the i^{th} axon with G_i, an integer from 1 to 4. The synaptic weight between any axon of type G and the j^{th} dendrite is given by the integer value s_j^G. The integer value λ_j is a leak, subtracted from the membrane potential at each tick. When the updated membrane potential $V_j(t)$ is larger than the threshold value α_j, the j^{th} neuron generates a spike and injects it into the network, and the state $V_j(t)$ is reset to R_j. The actual neuron equations implemented in the TrueNorth chip are more sophisticated [20]. For example, we support: stochastic spike integration, leak, and threshold using a pseudo random number generator (*PRNG*); programmable leak signs depending on the sign of the membrane potential; and programmable reset modes for the membrane potential after spiking. The axon and synaptic weight constraints are most effectively handled by incorporating the constraints directly in the learning procedures (for example, back propagation). We find that training algorithms are capable of finding excellent solutions using the free parameters available, given the architectural constraints.

We created the TrueNorth architecture by connecting a large number of neurosynaptic cores into a 2D mesh network, creating an efficient, highly-parallel, and scalable architecture. The TrueNorth architecture can be tiled not only at the core-level but also at the chip-level. Fig. 3 demonstrates spike communication within one chip, as well as between two adjacent chips. The green arrow shows a trace of an intra-chip spike traveling from the lower-left core to the upper-right core using the on-chip routing network. To extend the 2D mesh network of cores beyond the chip boundary, we equipped the TrueNorth chip with custom peripheral circuitry that allows seamless and direct inter-chip communication (demonstrated by the orange arrow in Fig. 3). As expected, the time and energy required to communicate between cores physically located on different chips is much higher than to communicate between cores on the same chip. We will revisit this issue in Section X.

To implement this architecture in a low-power chip, we chose to use event-driven computation and communication. This decision led us to using a mixed asynchronous-synchronous approach, as discussed in the next section.

IV. Mixed Synchronous-Asynchronous Design

The majority of typical semiconductor chips only utilize the synchronous design style due to a rapid design cycle and the availability of CAD tools. In this approach, all state holding elements of the design update their states simultaneously upon arrival of the global clock edge. In order to assure correct operation of synchronous circuits, a uniform clock

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

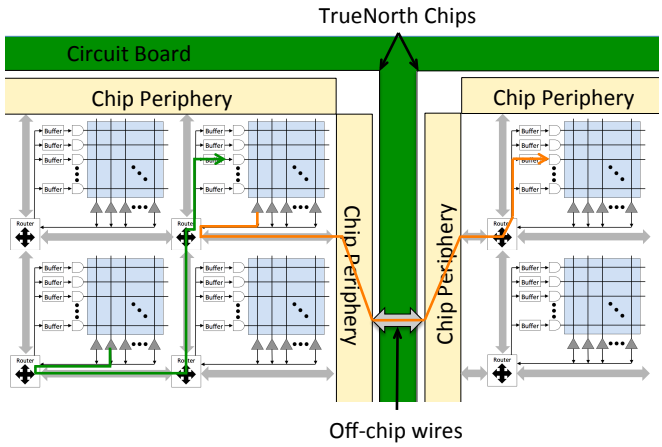

Fig. 3: Corners of two TrueNorth chips, with intra-chip (green arrow) and inter-chip (orange arrow) spike traces.

distribution network has to be carefully designed to guarantee proper synchronization of circuit blocks throughout the entire chip. Such clock distribution networks have penalties in terms of power consumption and area. Global clock trees operate continuously with high slew rates to minimize the clock skew between clocked sequential elements, and consume dynamic power even when no useful task is performed by the circuits. As a result, if used in an activity dependent design such as our neurosynaptic architecture, these clock trees would waste significant energy. Clock gating is a technique commonly used to mitigate the power drawbacks of global clock trees. However, clock gating is generally performed at a very coarse level and requires complex control circuitry to ensure proper operation of all circuit blocks. Another option for synchronous designs is to distribute slow-switching clocks across the chip and use phase-locked loops (PLLs) to locally multiply the slow clocks at the destination blocks, but PLLs have area penalties and are not natively activity-dependent.

In contrast, asynchronous circuits operate without a clock, by using request/acknowledge handshaking to transmit data. The data-driven nature of asynchronous circuits allows a circuit to be idle without switching activity when there is no work to be done. In addition, asynchronous circuits are capable of correct operation in the presence of continuous and dynamic changes in delays [21]. Sources of local delay variations may include temperature, supply voltage fluctuations, process variations, noise, radiation and other transient phenomena. As a result, the asynchronous design style also enables correct circuit operation at lower supply voltages.

For the TrueNorth chip, we chose to use a mixed asynchronous-synchronous approach. For all the communication and control circuits, we chose the asynchronous design style, while for computation, we chose the synchronous design style. Since TrueNorth cores operate in parallel, and are governed by spike generation/transmission/delivery (referred to as *events*), it is natural to implement all the routing mechanisms asynchronously. The asynchronous control circuitry inside each TrueNorth core ensures that the core is active only when it is necessary to integrate synaptic inputs and to update membrane potentials. Consequently, in our design there is no need for a high-speed global clock, and communication occurs by means of handshake protocols.

As for computational circuits, we used the synchronous design style, since this method aids in implementing complex neuron equations efficiently in a small silicon area, minimizing the leakage power. However, the clock signals for the synchronous circuit blocks are generated locally in each core by an asynchronous control circuit. This approach generates a clock pulse only when there is computation to be performed, minimizing the number of clock transitions and ensuring the lowest dynamic power consumption possible.

For the asynchronous circuit design, we selected a quasi delay-insensitive (QDI) circuit family. The benefits, as well as the limitations of this asynchronous style have been thoroughly analyzed in previous work by Martin [21]. The exception to the QDI design style was for off-chip communication circuits, where we used bundled-data asynchronous circuits to minimize the number of interface signals. The QDI design style, based on Martin's synthesis procedure [22], decomposes Communicating Hardware Processes (CHP) into many fine-grained circuit elements operating in parallel. These processes synchronize by communicating tokens over delay-insensitive channels that are implemented using a four-phase handshake protocol. One of the main primitives of such communication is a Muller C-element [23]. A single-bit communication protocol is shown in Fig. 4. Here, the sending process asserts the 'data 0' line to communicate a '0' value, shown by the rising transition (1), which the receiving process then acknowledges with a rising transition (2). Subsequently, the channel is reset with falling transitions (3,4) to a neutral state. As seen in Fig. 4, sending a '1' value, demonstrated on the next communication iteration, is quite similar. The doted arrows in Fig. 4 indicate the causality of the handshake operations.

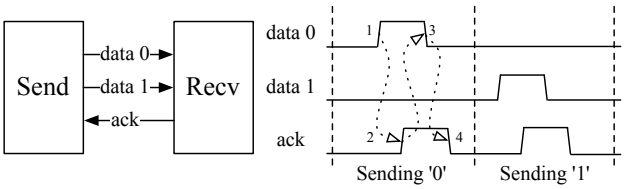

Fig. 4: Asynchronous 4-Phase Handshake

To implement QDI asynchronous pipelines we used pre-charge half-buffers (PCHB), and pre-charge full-buffers (PCFB) for computational operations based on their robust performance in terms latency and throughput, even with complex logic. As for simple fifo-type buffers and control-data slack matching, we used weak-condition half-buffers (WCHB), due to their short cycle time (10 transitions per cycle) and small size in terms of transistor count. These circuit primitives are described in detail by Lines [24].

This novel mixed asynchronous-synchronous design approach enables us to significantly reduce the active and static

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

power, by making the circuits event-driven, and tuning them for low-power operation. Nonetheless, the circuits operate at a high speed, making it possible to run the chip at real-time. For each block, depending on the requirements, we have used either synchronous or asynchronous design style, as discussed in the next section.

V. The TrueNorth Chip Design

A. High Level Chip Description

In this section we discuss the detailed design of the TrueNorth chip, implemented based on the presented architecture and the previously discussed circuit styles. We first explain the overall chip design, introducing all the TrueNorth blocks in this subsection, and then describe each individual block in the following subsections.

Fig. 5 shows the physical layout of a TrueNorth chip. The full chip consists of a 64×64 array of neurosynaptic cores with associated peripheral logic. A single core consists of the *Scheduler* block, the *Token Controller* block, the *Core SRAM*, the *Neuron* block, and the *Router* block.

The Router communicates with its own core and the four neighboring Routers in the east, west, north, and south directions, creating a 2D mesh network. Each spike packet carries a relative dx, dy address of the destination core, a destination axon index, a destination tick at which the spike is to be integrated, and several flags for debugging purposes. When a spike arrives at the Router of the destination core, the Router passes it to the Scheduler, shown as (A) in Fig. 5.

The main purpose of the Scheduler is to store input spikes in a queue until the specified destination tick, given in the spike packet. The Scheduler stores each spike as a binary value in an SRAM of 16×256-bit entries, corresponding to 16 ticks and 256 axons. When a neurosynaptic core receives a tick, the Scheduler reads all the spikes $A_i(t)$ for the current tick, and sends them to the Token Controller (B).

The Token Controller controls the sequence of computations carried out by a neurosynaptic core. After receiving spikes from the Scheduler, it processes 256 neurons one by one. Using the membrane potential equation introduced in Section III: for the j^{th} neuron, the synaptic connectivity $w_{i,j}$ is sent from the Core SRAM to the Token Controller (C), while the rest of the neuron parameters (D) and the current membrane potential (E) are sent to the Neuron block. If the bit-wise AND of the synaptic connectivity $w_{i,j}$ and the input spike $A_i(t)$ is 1, the Token Controller sends a clock signal and an instruction (F) indicating axon type G_i to the Neuron block, which in turn adjusts the membrane potential based on $s_j^{G_i}$. In other words, the Neuron block receives an instruction and a clock pulse only when two conditions are true: there is an active input spike and the associated synapse is active. Thereby, the Neuron block computes in an event-driven fashion.

When the Token Controller completes integrating all the spikes for a neuron, it sends several additional instructions to the Neuron block: one for subtracting the leak λ_i, and another for checking the spiking condition. If the resulting membrane potential $V_i(t)$ is above the programmable threshold, the Neuron block generates a spike packet (G) for the Router. The Router in turn injects the new spike packet into the network. Finally, the updated membrane potential value (E) is written back to the Core SRAM to complete the processing of a single neuron.

When a core processes all the neurons for the current tick, the Token Controller stops sending clock pulses to the Neuron block, halting the computation. The Token Controller then instructs the Scheduler to advance its time pointer to the next tick. At this point, besides delivering incoming spikes to the Scheduler, the neurosynaptic core goes silent, waiting for the next tick.

The chip's peripheral interfaces allow the extension of the 2D grid of neurosynaptic cores beyond the chip boundaries. Since the quantity of available I/O pins is limited, the Merge-Split blocks at the edges of the chip merge spikes coming from multiple buses into a single stream of spikes going out of the chip. Conversely, this peripheral unit also distributes a stream of incoming off-chip spikes to multiple buses in the core array.

Sections V-B through V-H introduce the blocks constituting the TrueNorth chip. We first introduce the blocks that reside in each TrueNorth neurosynaptic core, and then move to the chip's peripheral circuitry.

B. The TrueNorth Core Internals: The Neuron Block

The Neuron block is the TrueNorth core's main computational element. We implemented a dual stochastic and deterministic neuron based on an augmented integrate-and-fire neuron model [20]. Striking a balance, the implementation complexity is less than the Hodgkin-Huxley or Izhikevich neuron models, however, by combining 1–3 simple neurons, we are able to replicate all 20 of the biologically observed spiking neuron behaviors cataloged by Izhikevich [25]. The physical Neuron block uses time-division multiplexing to compute the states of 256 logical neurons for a core with a single computational circuit. In order to simplify implementation of the complex arithmetic and logical operations, the Neuron block was implemented in synchronous logic, using a standard ASIC design flow. However, it is event-driven, because the Token Controller only sends the exact number of clock pulses required for the Neuron block to complete its computation, as described in Section V-F.

The block diagram shown in Fig. 6, depicts the five major elements of the Neuron block, with input/output parameters from/to the Core SRAM shown in blue. The synaptic input unit implements stochastic and deterministic inputs for 4 different weight types with positive or negative values $s_j^{G_i}$. The leak and leak reversal units provide a constant bias (stochastic or deterministic) on the dynamics of the neural computation. The integrator unit sums the membrane potential from the previous tick with the synaptic inputs and the leak input. The threshold and reset unit compares the membrane potential value with the threshold value. If the membrane potential value is greater than or equal to the threshold value, the Neuron block resets the membrane potential and transmits a spike event. The random

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

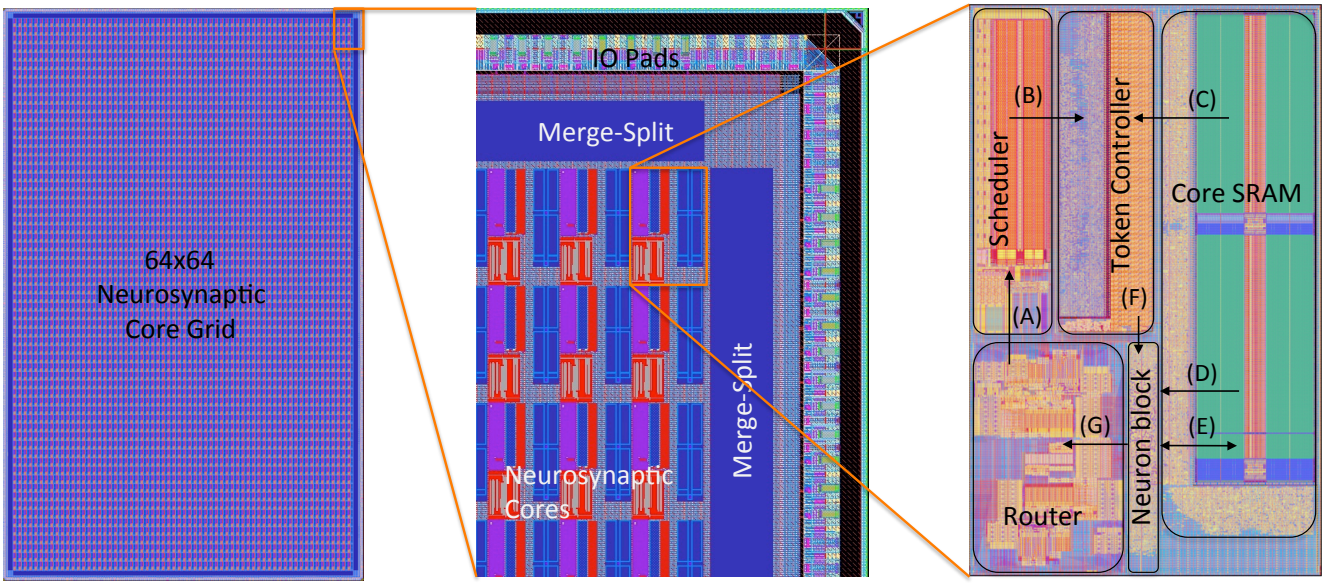

Fig. 5: The TrueNorth chip (left) consists of a 2D grid of 64×64 neurosynaptic cores. At the edge of the chip (middle), the core grid interfaces with the Merge-Split block and then with I/O pads. The physical layout of a single core (right) is overlaid with a core block diagram. See text for details on the data flow through the core blocks. The Router, Scheduler, and Token Controller are asynchronous blocks, while the Neuron and SRAM controller blocks are synchronous.

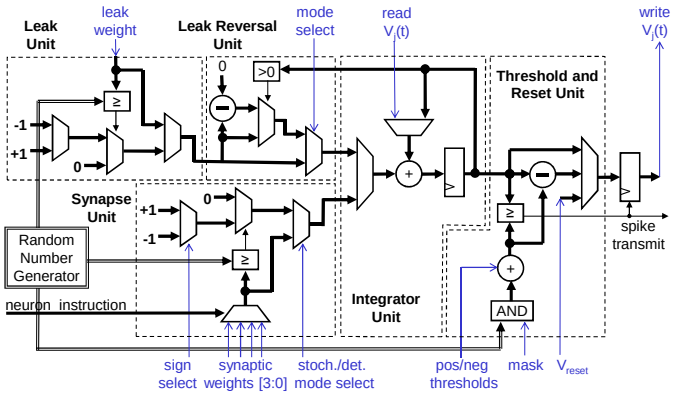

Fig. 6: Neuron Block Diagram

number generator is used for the stochastic leak, synapse, and threshold functions. The Core SRAM, external to this block, stores the synaptic connectivity and weight values, the leak value, the threshold value, the configuration parameters, as well as the membrane potential, $V_j(t)$. At the beginning and end of a neural computation cycle, the membrane potential is loaded from and then written back to the Core SRAM.

As previously mentioned, even though the Neuron block was implemented using synchronous design flow to take advantage of rapid and flexible design cycles, it is fully event-driven to minimize the active power consumption. Additionally, multiplexing a single neuron circuit for multiple logical neurons reduced the area of the block, while increasing the parallelism and reducing the static power consumption.

C. The TrueNorth Core Internals: The Router

The TrueNorth architecture supports complex interconnected neural networks by building an on-chip network of neurosynaptic cores. The 2D mesh communication network, which is responsible for routing spike events from neurons to axons with low latency, is a graph where each node is a core and the edges are formed by connecting the routers of nearest-neighbor cores together. Connections between cores, where neurons on any module individually connect to axons on any module, are implemented virtually by sending spike events through the network of routers.

As illustrated in Fig. 7, each Router is decomposed into six individual processes: From Local, Forward East, Forward West, Forward North, Forward South, and To Local. There are five input ports—one that receives the routing packets from spiking neurons within the local core and four others that receive routing packets from nearest-neighbor Routers in the east, west, north, and south directions of the 2D mesh. Upon receiving a packet, the Router uses the information encoded in the dx (number of hops in the x direction as a 9-bit signed integer) or dy (number of hops in the y direction as a 9-bit signed integer as well) fields to send the packet out to one of five destinations: the Scheduler of the local core, or nearest neighbor Routers to the east, west, north, or south directions. Thus, a spiking neuron from any core in the network can connect to an axon of any destination core within a 9-bit routing window by traversing the 2D mesh network through local routing hops. If a spike needs to travel further than this routing window (up to 4 chips in either direction), we use

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

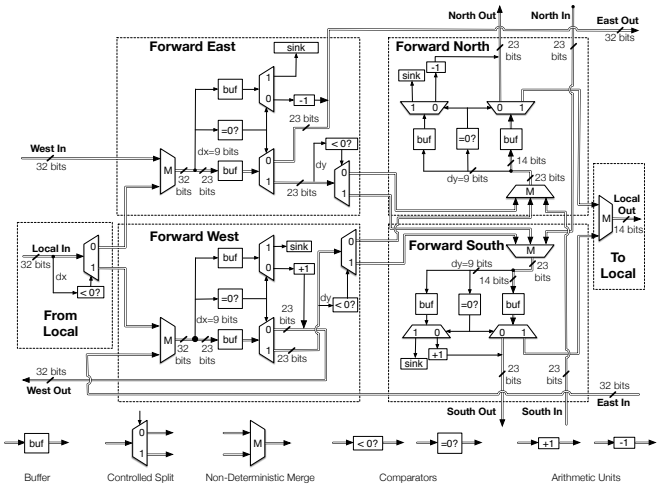

Fig. 7: Internal structure of the router (top) composed from five QDI asynchronous circuit blocks (bottom).

repeater cores.

Packets are routed first in the horizontal direction and the dx field is decremented (eastward hop, $dx > 0$) or incremented (westward hop, $dx < 0$) with each routing hop. When dx becomes 0, the associated bits are stripped from the packet and routing in the vertical direction initiates. With each vertical hop, dy is decremented (northward hop, $dy > 0$) or incremented (southward hop, $dy < 0$). When dy becomes 0, the associated bits are stripped from the packet and the remaining 14 bits are sent to the Scheduler of the local core. By prioritizing routing a packet in the east/west directions over the north/south directions, network deadlock through dependency cycles among Routers is precluded.

Because more than one message can contend for the same Router, its resources are allocated on a first-come first-serve basis (simultaneously arriving packets are arbitrated) until all requests are serviced. For any spike packet that has to wait for a resource, it is buffered and backpressure is applied to the input channel, guaranteeing that no spikes will be dropped. Depending on congestion, this backpressure may propagate all the way back to the generating core, which stalls until its spike can be injected into the network. As a result, the communication time between a spiking neuron and a destination axon will fluctuate due to on-going traffic in the spike-routing network. However, we hide this network non-determinism at the destination by ensuring that the spike delivery tick of each spike is configured to be longer than the maximum communication latency between its corresponding source and destination. In order to mitigate congestion, each Router has built-in buffering on all input and output channels in order to service more spikes at a higher throughput. The Router can route several spikes simultaneously, if their directions are not interfering.

The Router was constructed with fully asynchronous QDI logic blocks depicted at the bottom of Fig. 7, and described as follows:

- A buffer block enables slack matching of internal signals to increase Router throughput.
- A controlled split block routes its input data to one of two outputs depending on the value of a control signal.
- A non-deterministic merge block routes data from one of two inputs to one output on a first-come first-serve basis. An arbiter inside the block resolves metastabilities arising from simultaneous arrival of both inputs.
- Comparator blocks check the values of the dx and dy fields of the packet to determine the routing direction.
- Arithmetic blocks increment or decrement the value of the dx and dy fields.

The implemented asynchronous design naturally enables high-speed spike communication during times of (and in regions of) bursty neural activity, while maintaining low power consumption during times of (and in regions of) sparse activity. With the mesh network constructed from the Routers, the TrueNorth chip implements hierarchical communication; a spike is sent globally through the mesh network using a single packet, which fans out to multiple neurons locally at the destination core. This reduces the network traffic, enabling the TrueNorth chip to operate in real-time.

D. The TrueNorth Core Internals: The Scheduler

Once the spike packet arrives at the Router of the destination core, it is forwarded to the core's Scheduler. The Scheduler's function is to deliver the incoming spike to the right axon at the proper tick. The spike packet at the Scheduler input has the following format (the *dx* and *dy* destination core coordinates have been stripped by the routing network prior to the arrival of the packet to the Scheduler):

delivery tick	destination axon index	debug bits
4 bits	8 bits	2 bits

Here, *delivery tick* represents the least significant 4-bits of the tick counter; *destination axon index* points to one of 256 axons in the core; *debug bits* are used for testing and debugging purposes. The delivery tick of an incoming spike needs to account for travel distance and maximum network congestion, and must be within the next 15 ticks after the spike generation (represented by 4 bits).

The Scheduler Operation. The Scheduler contains spike write/read/clear control logic and a dedicated 12-transistor 16×256-bit SRAM. The 256 bitlines of the SRAM correspond to the 256 axons of the core, while 16 wordlines correspond to the 16 delivery ticks. The overall structure of the Scheduler is outlined in Fig. 8.

The Scheduler performs a *WRITE* operation to the SRAM when it receives a spike packet from the Router. The Scheduler decodes an incoming packet to determine when (tick) and where (axon) a spike should be delivered. It then writes a value 1 to the SRAM bitcell corresponding to the intersection of the destination axon (SRAM's bitline) and the delivery tick (SRAM's wordline). All other SRAM bits remain unchanged. This requires a special SRAM that allows a *WRITE* operation to a single bit. The *READ* and *CLEAR* operations to the SRAM

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

are initiated by the Token Controller. At the beginning of each tick, the Token Controller sends a request to the Scheduler to read all the spikes for the current tick (corresponding to an entire wordline of the SRAM); these spikes are sent to the Token Controller. After all spikes are integrated for all 256 neurons in the core, the Token Controller requests the Scheduler to clear the wordline of the SRAM corresponding to the current tick. Since the incoming spikes may arrive from the Router at any time, the Scheduler is designed to perform *READ/CLEAR* operations in parallel with a *WRITE* operation, as long as these operations are targeting different wordlines.

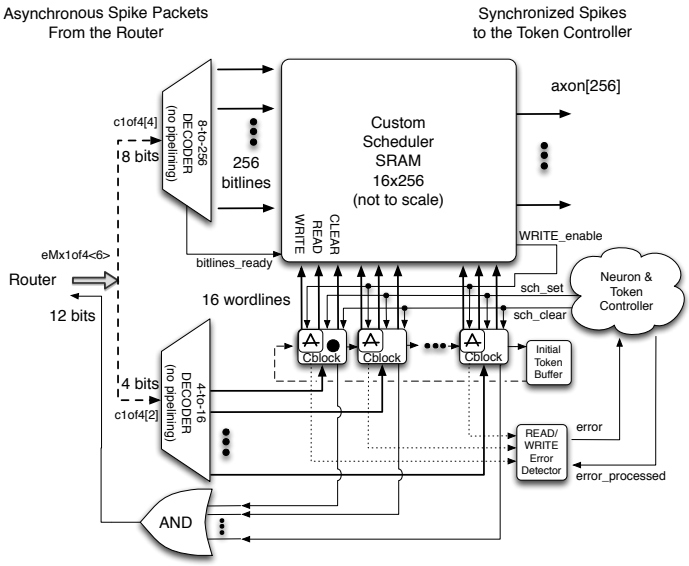

Fig. 8: Overall Scheduler Structure. Note that the Scheduler SRAM is rotated 90 degrees with respect to the bitcell orientation shown in Fig. 9.

The Scheduler SRAM. The SRAM used in the Scheduler has a special interface to store spikes that are coming from the Router (one at a time), while in parallel reading and clearing spikes corresponding to the current tick for the entire 256 axon array. The *WRITE* operation is performed by asserting the write wordline and bitline. The bitcell at the intersection of these lines will be updated with a value 1, while the rest of the SRAM is unchanged. The *READ* operation is initiated by asserting one of 16 read ports; this operation reads the entire 256 bits of the corresponding wordline. Similarly, the *CLEAR* operation is performed by asserting one of the 16 clear ports, which sets all the bitcells in the corresponding wordline to the value 0. The details of the physical implementation for the Scheduler SRAM are described in Section V-E.

The Scheduler Decoders. The asynchronous decoders of the Scheduler were implemented using 256 C-elements on the bitline side and 16 C-elements on the wordline side. From the spike packet, the bitline decoder receives the axon number (represented in 8 bits by a 1-of-4 coding scheme), and asserts one of the 256 bitlines. Similarly, the wordline decoder receives the delivery tick of a spike (represented in 4 bits by a 1-of-4 coding scheme) and asserts one of the 16 wordlines using custom self-timed control blocks. The decoders process incoming spikes one at a time to avoid race conditions. A 16-bit AND-tree combines *ready* signals from the control blocks, enabling the Router to forward the next incoming spike. The *ready* signals prevent the Router from sending the next spike until the previous one is successfully written to the SRAM.

The Scheduler Control Blocks. The Scheduler control blocks (Cblocks) asynchronously regulate the *READ*, *WRITE*, and *CLEAR* operations to the Scheduler SRAM. Cblocks receive incoming spike requests directly from the wordline decoder in a one-hot manner. The Cblock that receives a value 1 from the decoder initiates a *WRITE* to the corresponding wordline, as soon as the SRAM is ready (*WRITE_enable* $= 1$). Only one Cblock is allowed to perform *READ* and *CLEAR* operations in a given tick. The privilege of performing these operations is passed sequentially to the subsequent Cblock at the end of every tick using an asynchronous token. When the Token Controller asserts *sch_set* or *sch_clear*, the Cblock that possesses the asynchronous token performs the *READ* and *CLEAR* operations, respectively. If a WRITE is requested at the Cblock with the token, the Scheduler reports an *error* to the Neuron block, since this would imply a READ-WRITE conflict in the Scheduler SRAM. The Neuron block acknowledges this scenario with the *error_processed* signal.

The Scheduler's logic was implemented using the asynchronous design style to minimize the dynamic power consumption. The asynchronous approach is beneficial here, since spike packets may arrive from the Router at any time with variable rate. The Scheduler also serves the function of a synchronizer by aligning spikes and releasing them to axons at the proper tick. This synchronization is critical to maintain the hardware-software one-to-one equivalence by masking the nondeterminism of the routing network, while enabling real-time operation.

E. The TrueNorth Core Internals: The SRAMs

Each TrueNorth core has two SRAM blocks. One resides in the Scheduler and functions as the spike queue storing incoming events for delivery at future ticks. The other is the primary core memory, storing all synaptic connectivity, neuron parameters, membrane potential, and axon/core targets for all neurons in the core.

The Scheduler SRAM. The Scheduler memory uses a fully custom 12-transistor (12T) bitcell. (See Section V-D for block details and sequencing information.) Unlike a standard bitcell, the 12T cell performs three operations, with a port for each: READ, WRITE, and CLEAR. All signals are operated full swing, transitioning fully between GND and VDD, requiring no sense amplifiers. The READ operation fetches the state of a row of cells, requiring a read wordline, W_{RD}, and a read bitline, B_{RD}. The WRITE operation sets one bitcell (selecting both row and column) to 1, requiring a word select, W_{WR}, and a bit select, B_{WR}. The CLEAR operation resets the state of a row of cells to 0, requiring only a clear wordline, W_{CLR}.

The 12T cell is structured similar to a Set-Reset Latch, composed of a pair of cross-coupled NOR gates, which

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

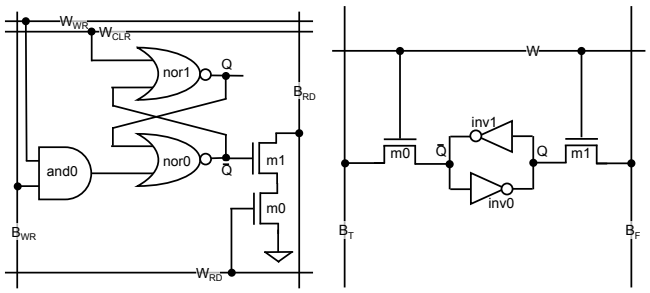

Fig. 9: SRAM bitcells: (a) 12-transistor Scheduler SRAM Cell (simplified), (b) 6-transistor Core SRAM Cell.

maintain an internal state, Q, and its inverse, $\overline{\text{Q}}$ (Fig. 9(a)). For a CLEAR operation, the W_{CLR} input to nor1 is pulled to 1, driving Q to 0, which drives $\overline{\text{Q}}$ to 1. For a WRITE operation, an input to nor0 is pulled to 1, driving $\overline{\text{Q}}$ to 0, which drives Q to 1. For this operation, the cell ANDs its row and column selects (W_{WR} and B_{WR}); however, in the physical implementation, we integrate the AND and NOR logic (and0 and nor0) into a single six transistor gate to save area (not shown). For a READ operation, we access a stack of two NFETs. When W_{RD} and $\overline{\text{Q}}$ are 1, the stack pulls B_{RD} to 0. When Q is 1 ($\overline{\text{Q}}$ is 0), the stack is inactive and does not pull B_{RD} down and it remains 1 (due to a pre-charge circuit).

The Core SRAM. The primary core memory uses a $0.152\mu\text{m}^2$ standard 6-transistor (6T) bitcell [26] (Fig. 9(b)). The standard SRAM cell uses READ and WRITE operations, sharing both wordlines and bitlines. The bitcell consists of a cross-coupled inverter pair and two access transistors. The SRAM module uses a standard sense amplifier scheme, reducing the voltage swing on bitlines during a READ operation and saving energy.

The SRAM is organized into 256 rows by 410 columns (not including redundant rows and columns). Each row corresponds to the information for a single neuron in the core (see Section V-F). The neuron's 410 bits correspond to its synaptic connections, parameters and membrane potential $V_j(t)$, its core and axon targets, and the programmed delivery tick.

synaptic connections	$V_j(t)$ & neuron parameters	spike destination	spike delivery tick
256 bits	124 bits	26 bits	4 bits

Both SRAMs were designed to minimize the power consumption and the area, while meeting the necessary performance requirement. For example, we reduced the size of the FETs in the SRAM drivers, lowering the signal slew rate to 3.5V/ns in order to save power. We also arranged the physical design floorplan in a way that minimizes the distance between the SRAM and the computational units, reducing the data movement and the dynamic power consumption. Additionally, we used redundant circuit structures to improve the yield, since the majority of the manufacturing defects occur in the SRAMs.

F. TrueNorth Core Internals: The Token Controller

Now that we have introduced the majority of the core blocks, the missing piece is their coordination. To that end, the Token Controller orchestrates which actions occur in a core during each tick. Fig. 10 shows a block diagram of the Token Controller. At every tick, the Token Controller requests 256 axon states from the Scheduler corresponding to the current tick. Next, it sequentially analyzes 256 neurons. For each neuron, the Token Controller reads a row of the Core SRAM (corresponding to current neuron's connectivity) and combines this data with the axon states from the Scheduler. The Token Controller then transitions any updates into the Neuron block, and finally sends any pending spike events to the Router.

Algorithmically, the control block performs the following events at each tick:

1) Wait for a tick;
2) Request current axon activity $A_i(t)$ from the Scheduler;
3) Initialize READ pointer of the Core SRAM to row 0;
4) Repeat for 256 neurons, indexed as j:
   a) Read neuron data from the current SRAM row;
   b) Repeat for 256 axons, indexed as i:
      i) Send compute instructions to the Neuron block only if spike $A_i(t)$ and synaptic connection $w_{i,j}$ are both 1; Otherwise skip;
   c) Apply leak;
   d) Check threshold, reset membrane potential if applicable;
   e) Send spike to the Router, if applicable;
   f) Write back the membrane potential to the Core SRAM;
   g) Increment the SRAM READ pointer to next row;
5) Request to clear *current* axon activity $A_i(t)$ in the Scheduler SRAM;
6) Report any errors.

This algorithm, known as the "Dendritic Approach", enables storing all the data for a neuron in a single Core SRAM row and only reading/writing each row once per tick. The Dendritic Approach sets a constant bound on the number of RAM operations, decoupling the SRAM read power consumption from the synaptic activity. In contrast, the more common "Axonal Approach" built into the Golden Gate neurosynaptic core [27], handles *every* input spike as an event that triggers multiple SRAM operations.

Internally, to process each neuron, the Token Controller sends a token through a chain of asynchronous shift registers (267 registers) and corresponding logic blocks. In the asynchronous register array, 256 registers contain data about which neurons need to be updated. The remaining registers handle the control logic between the Scheduler, the Core SRAM, and the Router (in case of any spikes). At each stage, a shift register may generate a timed pulse to a given block, send an instruction and generate a clock pulse for the Neuron block, or directly pass the token to the next register. Asynchronous registers responsible for executing synaptic integration instructions in the Neuron block have an input port that checks that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

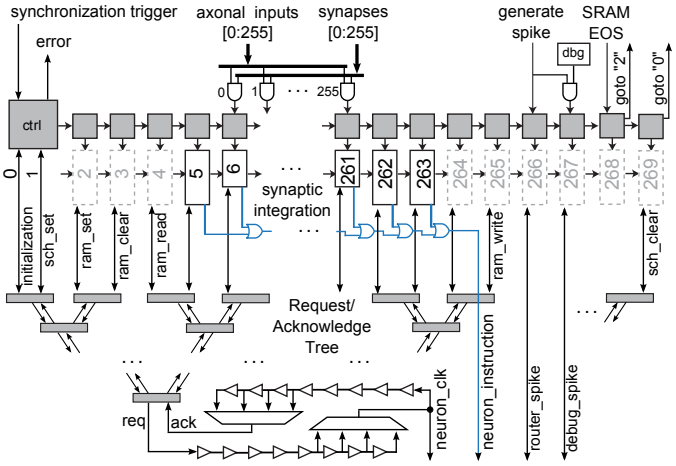

Fig. 10: Token Controller Overview (asynchronous registers shown as gray squares)

a given synapse is active and that there is a spike on that axon before generating an instruction and a trigger pulse for the associated neuron. If there is no synaptic event, the token advances to the next register, conserving energy and time.

Under extreme spike traffic congestion in the local routing network, the asynchronous token may not traverse the ring of registers within one tick, resulting in a timing violation. In such a scenario, a system-wide *error* flag is raised by the Token Controller indicating to the system that spike data may be lost and the computational correctness of the network is not guaranteed at this point, although the chip will continue to operate.

Depending on the application, dropping spike data may or may not be problematic for the system. Sensory information processing systems are designed to be robust at the algorithmic level. The system must be robust to variance in the input data in order to operate in realistic conditions. Similarly, additive noise should not destroy system performance. Thus, while losing data is undesirable, the system performance should degrade gracefully, and not catastrophically. Importantly, however, when data is lost, the errors are flagged by the system to notify the user.

When an error condition is detected, there are several approaches to correcting the situation. First, the user may retry the computation with a longer tick, a useful method when running faster than real-time. For real-time systems, however, the appearance of an error condition generally means that there is a problem in the network that needs to be corrected. This means either: (a) adjusting the I/O bandwidth requirements of the network, or (b) changing the placement to reduce the inter-chip communication (see Section X).

The Token Controller comprises both asynchronous and synchronous components to coordinate actions of all core blocks. However, the Token Controller design is fully event-driven with commands and instructions generated to other blocks only when necessary, while keeping the core dormant, if there is no work to be done. The Token controller is a key to lowering active power consumption and implementing the high-speed design required for real-time operation. The Token Controller description completes the overview of blocks inside of a TrueNorth core.

G. The TrueNorth Chip Periphery: The Merge-Split Blocks and The Serializer/Deserializer Circuits

Now we move from the design of an individual core, to the design of the peripheral circuitry that interfaces the core array to the chip I/O. The core array has 64 bi-directional ports on each of the four edges of the mesh boundary, corresponding to a total of $8,320$ wires for east and west edges and $6,272$ wires for north and south.[1] Due to the limited number of I/O pads (a few hundred per side), it is infeasible to connect the ports from the core array directly off chip without some form of time multiplexing.

To implement this multiplexing, we designed a Merge-Split block that interfaces 32 ports from the core array to the chip I/O, shown in Fig. 11. At the center of the design is the core array that has 256 bidirectional ports at its periphery, corresponding to the edge of the on-chip routing network. Groups of 32 ports interface to a Merge-Split block which multiplex and demultiplex packets to a single port before connecting to the pad ring. In total, there are eight Merge-Split blocks around the chip periphery. The Merge-Split was designed so that adjacent chips can directly interface across chip boundaries, thereby supporting chip tiling. Specifically, spikes leaving the core array are tagged with their row (or column) before being merged onto the time-multiplexed link. Conversely, spikes that enter the chip from a multiplexed link are demultiplexed to the appropriate row (or column) using the tagged information. To maintain a high throughput across chip boundaries, the Merge-Split blocks implement internal buffering via asynchronous FIFOs.

The Merge-Split is made from two independent paths: A Merge-Serializer path that is responsible for sending spikes from the core array to the I/O; and a Deserializer-Split path that is responsible for receiving spikes from the I/O and sending them to the core array. The Merge-Serializer path operates as follows. First, spikes packets leaving the core array enter a Merge, where they are tagged with an additional 5 bits to indicate which of the 32 ports they entered on. Logically, the Merge is a binary tree built from two-input one-output circuits with arbitration that combine 32 ports to a single port. Finally, this single port enters a Serializer circuit that converts the packet into an efficient bundled-data four-phase protocol suitable for off-chip communication, shown in Fig. 12.

The converse path consists of a Deserializer-Split. Here, spike packets arrive from the off-chip link using the bundled-data four-phase protocol. These packets are first converted back into a delay-insensitive four phase protocol via the

[1]Each uni-directional port uses 32-bits and 24-bits for east-west and north-south edges, respectively. These ports are encoded using 1-in-4 codes plus an acknowledge signal, corresponding to 65 and 49 wires respectively. The north and south ports require fewer bits since the Router data path in the vertical direction drops the dx routing information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

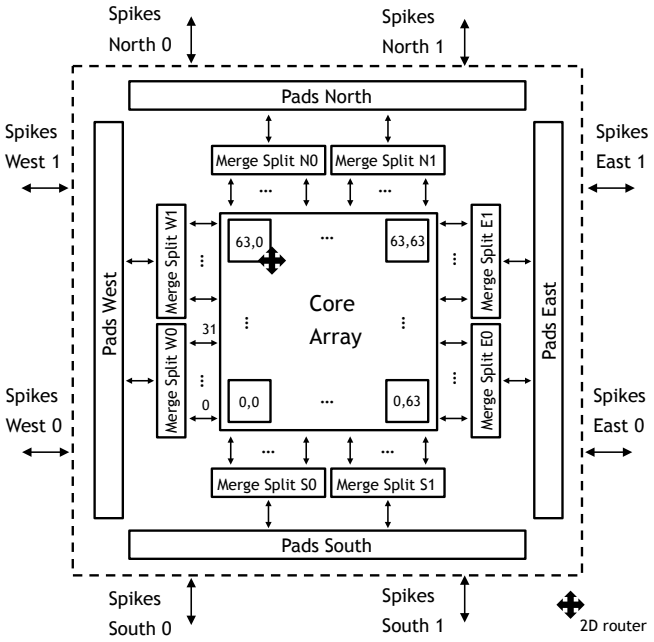

Fig. 11: Top Level Blocks of the TrueNorth Chip Architecture

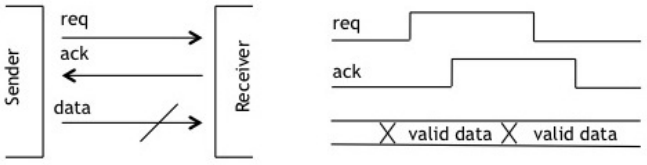

Fig. 12: Two phase bundled data protocol used for inter-chip communication. To maximize I/O bandwidth, we send data on both the rising and falling edges of the req signal (double data rate), and use bundled data.

Deserializer circuit. Next, the Split block steers the packet to the appropriate row (or column) based on the tagged information that originated from the adjacent chip. Logically, the Split is a binary tree built from one-input two-output steering circuits. Note that the row (or column) tag is stripped off before entering the core array.

The Router and the Merge-Split blocks operate in tandem, and we implemented both of them using an asynchronous design style. The QDI approach minimizes routing power consumption when the circuits are idle and maximizes throughput when the routing structures have to handle many spikes in flight at the same time. When no spikes are in flight, the Router and the Merge-Split blocks remain dormant and consume only a minimal amount of leakage power. The asynchronous design style also allows us to operate at a lower voltage and in the presence of longer switching delays.

By extending the 2D mesh network beyond the chip boundary, the Merge-Split blocks enable efficient scaling to multi-chip systems with tens of millions of neurons. The asynchronous handshake interface of the TrueNorth chip does not require high-speed clocks or any external logic for inter-chip communication, reducing the power requirement further.

H. The TrueNorth Chip Periphery: Scan Chains and I/O

Here, we cover the TrueNorth chip's interface for programming and testing. The TrueNorth chip is programmed through scan chains. Each neurosynaptic core has eight scan chains with a multiplexed scan interface. Three chains are used to read, write, and configure the core SRAM. Four additional chains are utilized to program the parameters of neurosynaptic cores such as axon types and PRNG seed values, as well as to test the synchronous logic. The last chain is a boundary scan isolating the asynchronous circuits and breaking the acknowledge feedback loop to test the asynchronous logic independently from the synchronous logic.

The scan chain runs at a modest speed of 10 MHz, without PLLs or global clock trees on the chip. In order to speed up the programming and testing of the chip, we added several improvements to the scan infrastructure. First, 64 neurosynaptic cores in a single row share a scan interface, reducing the chip I/O pin count used for scanning, while allowing cores in different rows to be programmed in parallel. Second, neurosynaptic cores in the same row may be programmed simultaneously with identical configuration, or sequentially programmed with unique configurations. This technique allows identical test vectors to be scanned into all neurosynaptic cores to run chip testing in parallel. Third, the TrueNorth chip sends out the exclusive-OR of the scan output vectors from neurosynaptic cores selected by the system. This property enables accelerated chip testing by checking the parity of the scan output vectors from multiple cores, while also having the ability of scanning out each individual core, if necessary.

This section completes the discussion of the TrueNorth chip design details. One of our key innovations is the adoption of asynchronous logic for activity-dependent blocks, while using conventional synchronous logic, driven by locally-generated clocks, for computational elements. In the following section, we discuss the custom hybrid design methodology that we developed to implement this state-of-the-art mixed asynchronous-synchronous chip.

VI. Chip Design Methodology

A. The TrueNorth Tool Flow

The combination of asynchronous and synchronous circuits in the TrueNorth chip poses a unique challenge in its design methodology. In the design approaches discussed in this section, we show how we combined conventional ASIC and custom design tools to design this chip.

The overall design flow is summarized in Fig. 13. The logic design of the synchronous circuits in the TrueNorth chip was specified using the Verilog hardware description language. The synchronous logic goes through the standard synthesis and place & route procedures. However, asynchronous circuit design and verification are not supported by commercial chip design tools. Instead we used a mix of academic tools [28] and tools developed internally. For asynchronous design we used

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

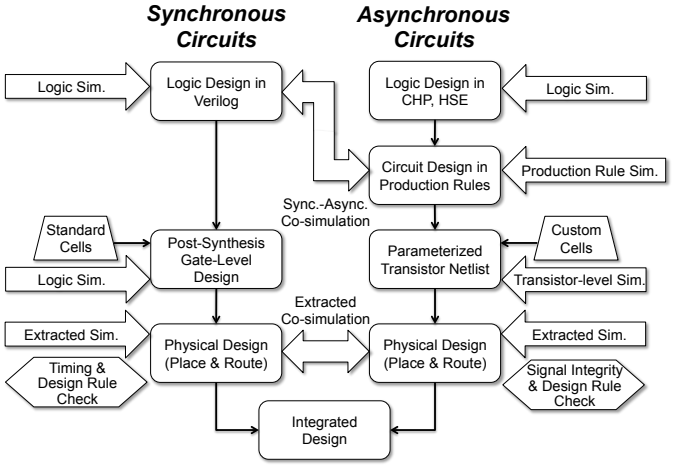

Fig. 13: TrueNorth Design Tool Flow

CHP (Communicating Hardware Processes [29]) hardware description language and HSE (Handshaking Expansions [30]) for high-level description. We then employed ACT (Asynchronous Circuit Toolkit developed at Cornell University, which evolved from CAST [31]) to aid in transforming the high-level logic into silicon, utilizing a transistor stack description format, called *production rules*. Though several techniques exist for automated asynchronous circuit synthesis [32], we performed this step manually to obtain highly-efficient circuit implementations for the TrueNorth chip.

The logic simulation of our hybrid design was carried out by a combination of a commercial Verilog simulator, Synopsys VCS, for synchronous circuits and a custom digital simulator, PRSIM [33], for asynchronous circuits. PRSIM simulates production rules and communicates with VCS using the Verilog Procedural Interface (VPI) [34]. When signal values change across the asynchronous-synchronous circuit boundaries, PRSIM and VCS notify each other via the VPI interface.

The input to PRSIM is a netlist based on production rules. A set of production rules can be viewed as a sequence of events/firings. All events are stored in a queue, and when the pre-conditions of an event become true, a timestamp is attached to that event. If the timestamp of an event coincides with PRSIM's running clock, the event (production rule) is executed. The timestamps of events can be deterministic or can follow a probability distribution. The probability distributions may be random or long-tailed (i.e most events are scheduled in the near future, while some events are far in the future). Such flexibility allows PRSIM to simulate the behavior of synchronous circuits, as well as of asynchronous circuits at random or deterministic timing. Whenever an event is executed, PRSIM performs multiple tests to verify correct circuit behavior. These tests include: 1) Verify that all events are *non-interfering*. An event is non-interfering if it does not create a short circuit under any conditions in the context of the design; 2) Verify proper codification on synchronous buses and asynchronous channels; 3) Verify the correctness of expected values of a channel or a bus (optional); 4) Verify that events are *stable*. In an asynchronous context, an event is considered stable when all receivers acknowledge each signal transition before the signal changes its value again. PRSIM is also capable of evaluating energy, power, and the transient effects of temperature and supply voltage on gate delays.

Since the TrueNorth chip is a very large design, simulating the logic of the entire chip is impractical (due to time and memory limitations). Instead, we dynamically load and simulate (using VCS and PRSIM) only the neurosynaptic cores that are actively used for a given regression. Overall, we ran millions of regression tests to verify the logical correctness of our design.

The physical design of the TrueNorth chip was carried out by a combination of commercial, academic and in-house tools, depending on the target circuit. On one hand, the blocks using conventional synchronous circuits were synthesized, placed, routed, and statically-timed with commercial design tools (Cadence Encounter). On the other hand, the asynchronous circuits were implemented exercising a number of in-house and academic tools. The high-level logic description of asynchronous circuits in the CHP language was manually decomposed into production rules. We then used an academic tool to convert production rules into a parameterized transistor netlist. These transistor-level implementations of the asynchronous circuits were constructed based on a custom cell library built in-house. With the help of in-house tools and Cadence Virtuoso, the mask design engineers created the physical design (layout) of the asynchronous implementations. Each step of converting the asynchronous logic design from CHP to silicon was verified by equivalence and design rule checkers.

QDI asynchronous circuits are delay-insensitive, with an exception of isochronic forks [21], which are verified by PRSIM. Thus, these circuits do not require conventional static timing analysis. However, a signal integrity issue, such as a *glitch* may result in a deadlock of an asynchronous design. As a result, we took several precautions to improve signal integrity and eliminate glitches. In particular, we minimized the wire cross talk noise, reduced gate-level charge sharing, maximized slew rates and confirmed full-swing transitions on all asynchronous gates. In order to avoid glitching, we enforced several rules throughout the asynchronous design. First, the signal slew rates of asynchronous gates must be faster than 5V/ns. Second, noise from charge sharing on any output node, plus cross talk noise from neighboring wires must not exceed 200mV (with the logic nominally running at 1V supply) to protect succeeding gates from switching prematurely [35]. In order to verify these conditions, we simulated the physical design of all asynchronous circuits using the Synopsys HSIM and Hspice simulators. When these conditions were violated, we exercised various techniques to mitigate the issues, including pre-charging, buffer insertion, and wire shielding.

Signal integrity in asynchronous circuits is one of the crucial issues to be addressed during design verification. It

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

requires an iterative process alternating between simulation and circuit-level updates, similar to the timing closure process for synchronous circuits. However, signal integrity verification is generally more time-consuming, because electrical circuit simulation takes longer than static timing analysis. The necessity for signal integrity analysis in the context of a large asynchronous design (over 5 billion total transistors on the TrueNorth chip) motivates a co-simulation tool that performs electrical simulation of the targeted asynchronous blocks (both pre- and post- physical design), while the remainder of the design is simulated logically, at a faster speed.

We utilized a custom tool, COSIM, which enables co-simulation of an arbitrary mix of synchronous and asynchronous circuit families at various levels of abstraction. COSIM automatically creates an interface between the Verilog simulator VCS, the PRSIM asynchronous simulator, and the HSIM transistor-level simulator using Verilog Procedural Interface. The tool supports simultaneous co-simulation of high-level behavioral descriptions (Verilog, VHDL, CHP), RTL netlists and transistor-level netlists using digital simulators (VCS, PRSIM) and an analog transistor-level simulator (HSIM). COSIM contains preloaded communication primitives, including Boolean signals, buses, and asynchronous channels. Whenever a connection needs to be made between different levels of abstraction, the tool detects the type of the interface required, and automatically generates the Verilog code that performs the necessary connection. Furthermore, COSIM has an extensive module library to send, receive, probe and check correctness of communication channels.

B. Timing Design

As discussed earlier, the TrueNorth chip contains several synchronous blocks, including the core computation circuits, memory controller, and scan chain logic. We followed a standard synthesis approach for each individual synchronous block (Fig. 13), however, this presented two unique challenges for system integration. First, there are no automated CAD tools that support mixed asynchronous-synchronous designs, and in particular the interface timing requirements between blocks. And second, we could not use hierarchical static timing analysis (STA) for timing paths between blocks at the core level, due to the presence of asynchronous blocks in the design.

In order to overcome these challenges, we defined the following interface design methodology:

- **Synchronous to Synchronous Blocks**: We divided the timing period between the two blocks and interconnecting wire, and assigned timing constraints to each of the blocks for standard static timing analysis.
- **Asynchronous to Asynchronous Blocks**: These interfaces use native QDI asynchronous handshaking and do not require additional timing constraints.
- **Synchronous to Asynchronous Blocks**: Data is set by the synchronous block one clock cycle prior to consumption by the asynchronous block. Consumption is triggered by the asynchronous Token Controller following completion of the synchronous clock cycle.
- **Asynchronous to Synchronous Blocks**: Data is set by the Token Controller and then the Token Controller sends a clock pulse (controlled by programmable delays) to latch the data. This interface has a delay assumption described in Fig. 15 in more detail.

Fig. 14 shows the mutually exclusive timing of the mixed synchronous/asynchronous timing domains. Synchronous and asynchronous transactions occur on different cycles, as dictated by the Token Controller, and both types of cycles are bounded by an asynchronous handshake in the Token Controller. So for any two interfacing blocks, the Token Controller assures that the asynchronous signals are static during a cycle while the synchronous signals transition, and the synchronous signals are static during a cycle while the asynchronous signals transition.

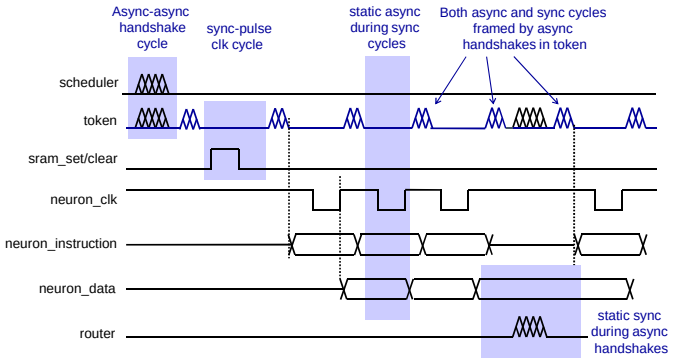

Fig. 14: Mutually Exclusive Asynchronous / Synchronous Timing Operations

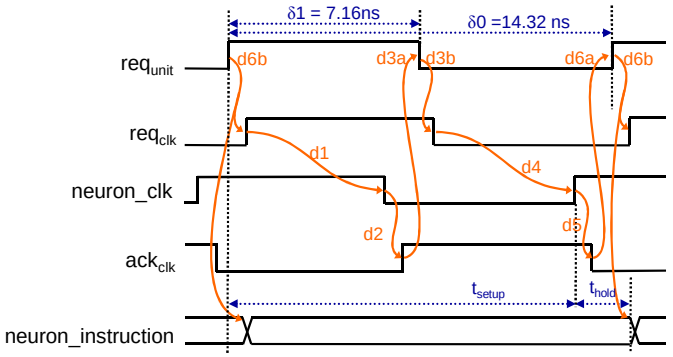

Fig. 15: Token Controller/Neuron Detailed Timing Diagram: $d1$ and $d4$ are set by the first programmable delay line; $d2$ and $d5$ are set by the second programmable delay line; $d3$ and $d6$ are the duration of an asynchronous handshake and an OR tree propagation delay ($\sim$ 1 ns). $t_{setup} = (d6b + d1 + d2 + d3 + d4)$ and $t_{hold} = (d5 + d6a)$.

The Token Controller generates the synchronous clock and data according to the timing diagram of Fig. 15. In order to mitigate the risk of a timing violation, we use two programmable delay lines (based on delay cells in the standard cell library) to configure the position of the rising clock edge

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

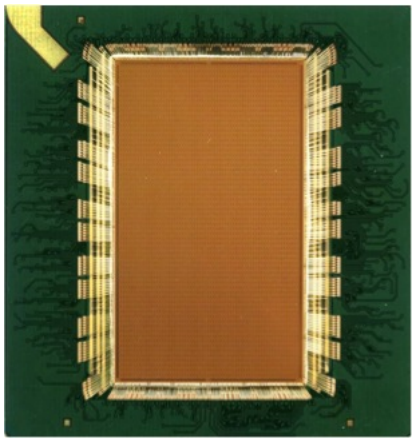

Fig. 16: The TrueNorth Chip Die Photo w/ Package Substrate

in increments of 0.27ns, as shown in Fig. 10. This governs the setup and hold times of the data going into the Neuron block.

In general, the interfaces between the asynchronous-synchronous boundaries require special care to assure the correct circuit behavior. Without automated tools to handle this, it is important to limit the asynchronous-synchronous interfaces to the block and unit boundaries to make the design and verification manageable.

C. Manufacturing Considerations

The TrueNorth chip, shown in Fig. 16, was fabricated using Samsung's 28nm LPP CMOS process technology. This silicon fabrication process was tuned for low-power devices. In order to further reduce power consumption of the chip, we used field-effect transistors with longer channel lengths. In our design, for all asynchronous state-holding gates, we used combinational feedback [33] instead of staticizers [36] to minimize switching power consumption, as well as due to process technology limitation of only few discrete transistor gate lengths.

At $4.3cm^2$ in die area and 5.4 billion transistors, the TrueNorth chip is larger than a typical ASIC chip, and manufacturing defects could have been a serious issue. To minimize the risk of defects at the design mask level, we utilized *high-yield* manufacturing rules during the physical design. We also implemented various redundancy and testing structures on the TrueNorth chip. For example, at the circuit level, the Core SRAM has ten redundant columns and sixteen redundant rows to improve the yield. We also placed dummy cells around the SRAM cell arrays to improve the lithographic rendering quality. Finally, we applied Monte-Carlo simulation to the SRAM circuits to verify sufficiently high yield.

Testing and debugging were some of the reasons why we designed the TrueNorth chip to be one-to-one equivalent with its software simulator. In the event of failure, we can compare the chip's behavior to that of the simulator, quickly isolating the deviation and identify the faulty cores. This is one advantage of using digital circuits over analog circuits in designing artificial neurons.

The arrangement of our communication network addresses the yield issues that may arise during fabrication. Faulty cores can be disabled and routed around, allowing the majority of the chip to remain functional in the face of defects. In the less likely case of a faulty router, the entire row and column of cores associated with a given router would have to be disabled, since spikes traveling to any core on the given row/column may pass throughout the defective router and potentially deadlock the system. In these cases, the configuration of the TrueNorth chip must be adjusted by the chip's software to avoid the faulty cores. Therefore, due to the high level of on-chip core redundancy, the chip still remains functional, losing only a fraction of the available cores.

The takeaway point is that while designing the TrueNorth chip we used a number of architectural, circuit-level, as well as manufacturing process-related techniques to make the chip power-efficient and resilient to manufacturing defects.

VII. Measured TrueNorth Chip Data

We tested the TrueNorth chip for logical correctness and extensively characterized it for performance and power consumption [1], [3]. Naturally, the performance and efficiency of the TrueNorth chip varies with the neural activity, routing network connectivity, and operating voltage. Here we present some important technical characteristics of the TrueNorth chip. As explained in Section IV, based on the TrueNorth chip's event-driven implementation, we are able to operate the chip at lower voltages without running into timing violations that one would encounter using a common synchronous design approach. Overall, the TrueNorth chip is operational from 1.05V down to 0.7V, with total power consumption ranging from 42mW in the low corner (0.70V, 0Hz firing rate, 0 synapses/neuron) to 323mW in the high corner (1.05V, 200Hz firing rate, 256 synapses/neuron). As an example, to show the chip's low-power capability, we pick an operating point of 0.75V. At this supply voltage, the maximum computational speed of the chip is 58 Giga-Synaptic Operations Per Second (GSOPS) and the maximum computational energy efficiency is 400 Giga-Synaptic Operations Per Second per Watt (GSOPS/W).[2] While running a typical complex recurrent neural network at 0.75V with 20Hz average firing rate and 128 active synapses per neuron at real-time (1kHz tick), the TrueNorth chip consumes only 65mW and delivers 46GSOPS/W. We also demonstrated the TrueNorth chip properly functioning faster than real-time (tick $>$ 1kHz). In our experiments we measured up to $21\times$ real-time operation, dependent on the activity rates, synaptic density, and voltage levels.

Fig. 17 shows a breakdown of the components of the overall TrueNorth chip power (@ 0.8V) for three complex recurrent networks with 128 synapses per neuron average, and three different average firing rates. Such complex networks approximate a worst-case scenario for power consumption as the cores are randomly placed on the chip. Random placement significantly increases internal communication power as

[2]Measured using a recurrent network with neuron firing rates in the range of 0–200Hz and synaptic connectivity varying between 0–100%.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

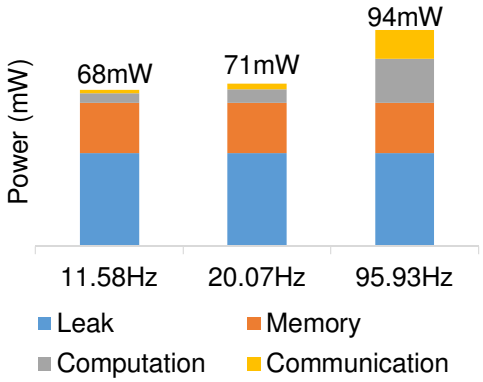

Fig. 17: Total TrueNorth chip power breakdown (@ 0.8V) for three complex recurrent networks with 128 synapses per neuron average, and three different average firing rates.

compared with the application examples where communicating cores tend to be near each other (the placement can be optimized as in Section X). Computation and communication are both event driven, thus active power scales with firing rate. In [3], we measured 2-3 orders of magnitude speedup in execution time and 5 orders of magnitude lower energy consumption for a set of computer vision neural networks run on TrueNorth, over the same networks run on a dual 2.4GHz E5-2440 processor x86 system, as well as a Blue Gene/Q system with up to 32 compute cards.

From the above data, one can see that the TrueNorth chip is a very power-efficient (65mW typical) and highly-parallel neurosynaptic chip that runs complex neural networks in real-time. Next, we present the TrueNorth-based systems that we used for collecting the test data, as well as for running the visual recognition applications described in Section IX.

VIII. The TrueNorth Platforms

We architected the TrueNorth chip with scalability as one of the major requirements. Like the cores within a chip, the chips themselves are designed to be tiled into a scalable two-dimensional array without any modification to the underlying routing algorithm or the need for off-chip interface circuits. This way we can create large networks of neurosynaptic cores with many interconnected neurons. From a logical point of view, there is no difference whether the communication between neurons occurs on the same chip or across many chips.

Using these principles, we built a single-chip testing system for characterizing the TrueNorth chips, as well as several multi-chip systems, with various arrangements of the TrueNorth chips [3]. We also built a *mobile* single TrueNorth chip system (NS1e), which is designed as a stand-alone, compact, low-power, flexible delivery platform for real-time spiking neural network applications. The multi-chip systems include a 4×1 chip platform and a 4×4 chip platform [3]. The total system power, while running the grid classifier application on the 4×4 platform at real time was 7.2W, of which the TrueNorth array (@ 1.0V) consumed 2.5W. These TrueNorth systems are currently being successfully used for the neural applications described in Section IX.

The direct communication between chips in all directions on multi-chip boards occurs by means of asynchronous bundled-data protocol, as explained in Section V-G. The TrueNorth pads are allocated and aligned in a manner that allows simple tiling and direct chip-to-chip signal connection. The communication interface signals from the peripheral chips are routed to off-board connectors to allow further scaling that creates a larger 2D mesh of neurosynaptic cores by tiling (or densely stacking) the TrueNorth boards. These off-board connectors may be mated directly to similar connectors on other boards for further seamless chip communication or through FPGA-based data-port expanders to create an even larger scale routing infrastructure.

The key for building more complex TrueNorth systems in the future, as proposed in Section XI, is the TrueNorth chip's power efficiency, which relaxes the system power delivery constraints and significantly decreases (or potentially eliminates) the cooling requirements.

IX. The TrueNorth Chip Applications

Applications for the TrueNorth architecture are developed using the Corelet Programming Environment (CPE), an object-oriented, compositional language and development environment for building neurosynaptic software that is efficient, modular, scalable, reusable, and collaborative [4]. We previously demonstrated a variety of corelet applications, including speaker recognition, musical composer recognition, digit recognition, Hidden Markov Model sequence modeling, collision avoidance, and eye detection [5].

Corelet programs describe neurosynaptic cores in a logical format that is agnostic to the physical location of each core in the actual hardware, allowing the same corelet to be deployed on different hardware configurations without modifying the source code. However, for any given hardware configuration, every logical core must be mapped to a unique physical core on a TrueNorth chip. These physical core placements can be optimized to minimize bandwidth and active power using application-agnostic algorithms, as described in Section X.

To demonstrate these optimizations, we present a selection of streaming video applications (Fig. 18), all of which can process video at 30 frames per second in real-time [3].

- *Haar-like features* : Haar-like features, popular for face recognition [37], are differences of summed intensity in adjacent rectangles. We extract ten Haar-like feature maps—eight sensitive to vertical or horizontal edges with various phases, plus two center-surround features—from each 200×100-pixel frame.
- *Local Binary Patterns* : Local Binary Patterns (LBP) are simple texture features used in biometrics, robot navigation, and MRI analysis [38]. We extract 20-bin Local Binary Pattern feature histograms from subpatches covering a 200×100-pixel frame.
- *Saccade generator* : Mammalian visual attention directs a succession of quick eye movements called saccades to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

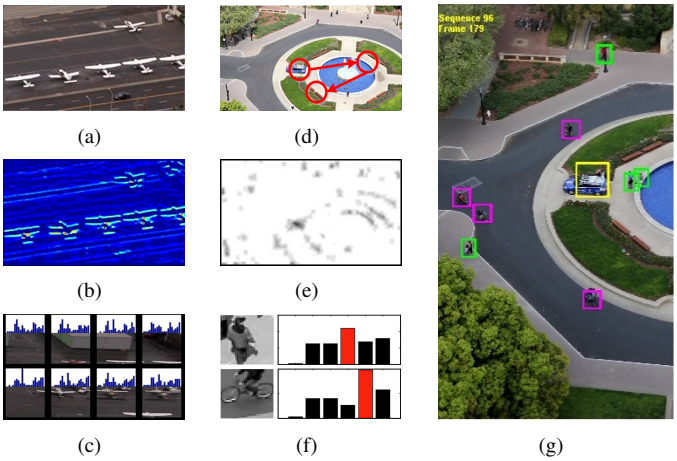

Fig. 18: Examples of streaming video applications. Left: features extracted from **(a)** one frame of a moving-camera video: **(b)** Haar-feature response map for horizontal lines; and **(c)** eight LBP histograms extracted from 8 subpatches. Middle: **(d)** Consecutive saccades, targeted to peaks of **(e)** a saliency map. **(f)** K-means classifier predictions for a person (top, fourth bar) or bicyclist (bottom, fifth bar); bar height denotes prediction strength for five foreground classes and a null class (sixth bar). Right: **(g)** Grid classifier detects and classifies five types of foreground objects in each frame of an HD video with a vertical aspect ratio (green box=person, magenta=bicyclist, yellow=car).

salient targets in the field of view [39]. Our application identifies saccade targets in an HD image by combining various feature maps into a single saliency map, performing a winner-take-all operation to find the most salient pixels, and using inhibition of return to prevent consecutive saccades from looking at the same place.

- *K-means classifier* : Once a saccade generator has successfully detected and centered a salient object in a 32×32-pixel patch, our first classifier example uses k-means clustering to identify up to one foreground object per frame [40].
- *Grid classifier* : To detect and classify all objects in an entire image simultaneously, our second two classifier examples tile the image with a grid of identically trained patch classifiers. Grid Classifier B (Fig. 18g) is tuned to maximize classification accuracy, using 16 chips to obtain an out-of-sample precision of 0.85 and 0.75 recall on the DARPA Neovision2 Tower dataset, a multi-object detection and classification task based on a collection of fixed-camera HD videos containing moving and stationary people, bicyclists, cars, buses, and trucks [41]. Grid Classifier A trades a small amount of accuracy for much lower chip area and energy consumption, requiring only 7 chips to obtain a 0.82 precision and 0.71 recall.

The total TrueNorth power while running each of these applications at real-time is listed in the final column of Table I. While the primary application that we present is our solution to the NeoVision visual-object-recognition task, which contains what/where pathways analogous to the mammalian visual cortex, TrueNorth is a general-purpose computing architecture whose design space is not confined strictly to biologically motivated algorithms. Thus, while the bio-inspired NeoVision application demonstrates depth, applications like Haar features, LBP histograms, and K-means are intended to show breadth across a range of designs. For a case study on how conventional computer-vision algorithms like Haar features can efficiently map to the TrueNorth architecture in a saliency-map application, see [42].

X. Algorithm Mapping: Core Placement

The power consumption of a TrueNorth system running a program depends on how the program is mapped on the chip. The energy required to communicate a spike between cores or even between chips increases with the distance between the source and the destination. Mapping logically connected cores to physically adjacent cores on the same chip can dramatically reduce the power consumption of the overall system. We adopted an existing VLSI design automation tool to solve this problem.

The problem of mapping neurosynaptic cores to minimize spike communication power can be formulated as a wire-length minimization problem in VLSI placement [43]:

$$\begin{aligned} \min \quad & W(\mathbf{x},\mathbf{y}) \\ s.t. \quad & D_i(\mathbf{x},\mathbf{y}) \le M_i, \qquad \text{for each tile } i, \end{aligned}$$

where $W(\mathbf{x},\mathbf{y})$ is a wire-length function and $D_i(\mathbf{x},\mathbf{y})$ and M_i encode placement legality constraints for floor-planning tile i. For neurosynaptic core placement, $D_i(\mathbf{x},\mathbf{y})$ is a potential function representing the sum of core areas in tile i, and M_t is the maximum potential value (i.e., the maximum number of cores) allowed in a tile.

To define a wire-length function for neurosynaptic core placement, we represent the network as a graph $G = (V, E)$ whose nodes $V = \{v_1, v_2, \ldots, v_n\}$ denote cores, and edges $E = \{e_1, e_2, \ldots, e_m\}$ denote nets connecting cores. For nets between cores on the same chip, the active power per spike is a linear function of the Manhattan distance between the source and the destination. Nets connecting cores on different chips consume substantially higher power while bridging chip boundaries. Accordingly, our net wire-length model has separate terms for within-chip (n_{on}) and off-chip (n_{off}) nets:

$$\begin{aligned} W(\mathbf{x},\mathbf{y}) \quad = \quad & \sum_{e \in n_{on}} [|x_i - x_j| + |y_i - y_j|] \\ & + \sum_{e \in n_{off}} \mathbf{E} * [|cx_i - cx_j| + |cy_i - cy_j|] \end{aligned}$$

where x_i and y_i are the respective x- and y-coordinates of the core v_i, $\mathbf{E}$ is a large constant penalty for off-chip communication between neighboring chips, and cx_i and cy_i denote the coordinates of chip boundaries. The off-chip term is simply the penalty constant multiplied by the number of chip boundaries a spike must traverse.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

Recall from Section V-G, each merge operation combines 32 buses down to 1 bus, followed by a 2 times serialization, resulting in a 64 times reduction in boundary bandwidth. In addition, there is a greater than 10 times disparity in frequency between the internal NoC frequency and the I/O pad frequency. Combined, the spike bandwidth is over 640 times lower at the chip boundary than internal to the chip.

In addition to its static wire-length $W(\mathbf{x}, \mathbf{y})$, each edge e_i is assigned a weight w_i that is proportional to the number of spikes communicated across that edge. Minimizing total weighted wire-length for a network of neurosynaptic cores is equivalent to minimizing active communication power of a TrueNorth system.

To perform the minimization, we use IBM CPLACE [44], an analytical placer used to design high performance server microprocessors, gaming chips, and other ASICs. When placing each network, we use four different global placement algorithms and use the result with the minimum wire length as the actual implementation[3]. In practice, all four methods produce efficient placement results.

- *Multilevel Partitioning-driven algorithm* [46]: This method recursively partitions the placement area into sub-regions and minimizes the crossing nets called "cuts" among the sub-regions. This process is repeated for each sub-region until the area becomes small enough.
- *Analytical Constraint Generation (ACG) algorithm* [47]: The partitioning-based algorithm is enhanced with cell-distribution constraints determined by an analytical free space calculation.
- *Hierarchical Quadratic Placement algorithm* [48]: The analytical top-down quadratic partitioning algorithm by Vygen et al. [49] is enhanced with a semi-persistent bottom-up clustering algorithm.
- *Quadric-based Force-directed Analytical algorithm* [44]: The force-directed analytical placer is one of the most popular placement algorithms, balancing wire-length minimization with density constraints to yield a high quality solution.

Table I lists placement optimization results and measured total power for various applications mapped to either the single-chip TrueNorth testing board or the 4×4 TrueNorth board, corresponding to networks on the order of 0.6M to 16M neurons and 150M to 4B synapses. For Grid Classifier B, a 16-chip network mapped to the 4×4 board, optimizing core placement reduces average static hop distance by a factor of 24× to only 6.7 hops (Fig. 19), and reduces the number of static cross-chip routes by a factor of 55×. Optimization reduces maximum spike traffic from about 10K spikes/tick on some ports down to 2500 spikes/tick on all ports (Fig. 20). This optimization first ensures the boundary traffic is low enough to avoid congestion delay, while maintaining real-time operation. Further optimization reduces the energy consumption.

[3]Note that the physical mapping problems are easily ported into the placement bookshelf contest formats [45]. Therefore, if you are interested in trying your own placement implementation, please contact the authors.

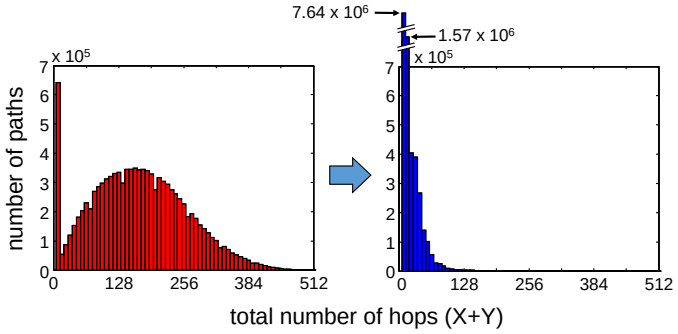

Fig. 19: Hop distribution of spikes using unoptimized (left) or optimized (right) core placement of Grid Classifier B.

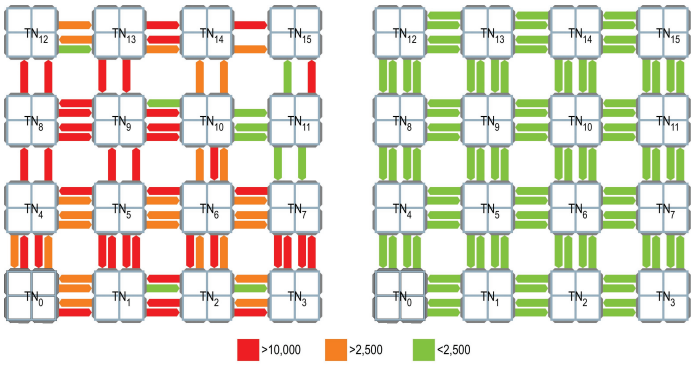

Fig. 20: Maximum spikes per tick for each port using unoptimized (left) or optimized (right) core placement of Grid Classifier B on the 4×4 TrueNorth board.

The minimum wire length solution also implicitly addresses the worst-case spike travel time by reducing congestion, as well as by reducing the number of long wire paths, both within and across chip boundaries, as shown in the hop distance distributions in Fig. 19.

This optimization technique of mapping logical cores to physical TrueNorth cores is invaluable for the software design methodology, improving the efficiency of TrueNorth programs run on hardware. It both reduces network bandwidth requirements and minimizes active communication power.

XI. The Future Large Scale TrueNorth Systems

This section discusses our perspective for scaling the TrueNorth architecture to massive neurosynaptic systems using the existing TrueNorth hardware described earlier. The extremely low-power profile of the TrueNorth chip enables us to envision large-scale neurosynaptic systems approaching the scale of a mammalian brain.

Many neural networks employ both short and long connections for communication between neurons. On a TrueNorth-based large-scale system, short distance communication is handled by the core crossbars and asynchronous routers at the intra-chip level. For longer distance communication, which may connect neurons that are physically located far apart from

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2474396, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TABLE I: Optimization Results and Measured Total TrueNorth Chip Power Results

	Network Size				Communication Power (Active)			Total Power
Applications	#chips	#cores	#nets	#pins	Default	CPLACE	× Improvement	(Watts)
Saccade generator	1	2297	17523	35046	0.40mW	0.11mW	3.6×	0.049W @0.75V
Local Binary Patterns	1	3520	22032	44064	4.41mW	2.33mW	1.9×	0.064W @0.75V
Haar-like features	1	3875	45822	91644	7.57mW	1.81mW	4.2×	0.058W @0.75V
K-means classifier	4	14832	445380	890760	20.03mW	6.44mW	3.1×	0.653W @1.0V
Grid classifier A	7	27644	461788	923576	589.90mW	130.88mW	4.5×	1.274W @1.0V
Grid classifier B	16	62375	960013	1920030	353.95mW	70.06mW	5.1×	2.515W @1.0V

each other in the system, we use robust asynchronous channels to communicate between chips. As for the ultra-long distance connections, we use conventional communication protocols. As one design solution, we can create a tree-like topology of Ethernet nodes. On each node, programmable-logic is used to convert asynchronous spike information from the periphery of a 2D mesh network of TrueNorth chips to a TCP/IP packet format, which can be sent over the Ethernet.

With low power consumption being the major characteristic of our neurosynaptic chips, TrueNorth chips may be densely packed together without cooling or power distribution concerns. Our preliminary calculations show that 4096 chips may be packed into a single rack, creating a 4 billion neuron system. The estimated power consumption for the TrueNorth chips on such a system is 300W. For communication to the outside world, as mentioned earlier, and for fast configuration purposes, TrueNorth systems need to employ FPGAs and network interfaces. As a result of this power consumption overhead, we estimate the total power of a 4 billion neuron system at a few kilowatts. Scaling such a TrueNorth system further to 96 racks will deliver 412 billion neurons with 10^{14} synapses, which is comparable to a *human brain* in terms of the number of neurons and synapses. At this scale, the power consumption attributed to the TrueNorth chips is estimated to be 29kW, although the total system power is likely to be a few hundred kilowatts due to communication and configuration devices. Such a system should be able to compute in *real-time* the 10^{14} synapse model, which was previously simulated by the BlueGene/Q Sequoia 7.9MW system $1542\times$ slower than real-time [10].

XII. Conclusion

In this paper we presented the design and tool flow innovations of the groundbreaking TrueNorth chip that implements our brain-inspired, event-driven, non-von Neumann architecture. The TrueNorth chip is the world's first 1 million neuron, 256 million synapse fully-digital neurosynaptic chip, implemented in a standard-CMOS manufacturing process. The chip is highly-parallel, defect-tolerant, and operates at real-time with an extremely low typical power consumption of *65mW*. Inside our low-power TrueNorth chip, we implement architectural innovations and complex unconventional event-driven circuit primitives to adhere to our seven main design principles.

We covered the mixed asynchronous-synchronous design approach, and the methodology that we developed while creating the TrueNorth chip. This novel tool flow, along with the event-driven techniques that we demonstrated, give designers an opportunity to mix asynchronous and synchronous circuits in order to make their future *state-of-the-art* designs more flexible and energy efficient. We now turn to the CAD research community for collaboration in automating and improving this tool flow further.

In contrast to general purpose von Neumann machines, which are optimized for high-precision integer and floating-point operations, TrueNorth broadly targets sensory information processing, machine learning, and cognitive computing applications using synaptic operations. However, like the early von Neumann machines, the task is now to create efficient neurosynaptic systems and optimize them in terms of programming models, algorithms, and architectural features. Using the TrueNorth chip, we can create large-scale and low-power cognitive systems as a result of the architecture's *scalability* property. We have already built the first multi-chip TrueNorth-based neurosynaptic platforms (with up to 16 million neurons and 4 billion synapses) and demonstrated example applications, such as visual object recognition, running on these platforms at *real-time* with orders of magnitude lower power consumption than conventional processors. With such outstanding technical characteristics, TrueNorth systems significantly advance the field of cognitive research and revolutionize the current state of real-world multi-sensory systems, creating new opportunities for applications ranging from robotics to battery-powered consumer electronics to large-scale analytics platforms.

Acknowledgments

This research was sponsored by Defense Advanced Research Projects Agency (DARPA) under contract No. HR0011-09-C-0002. The views, opinions, and/or findings contained in this article/presentation are those of the author(s)/presenter(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. We are grateful to many collaborators: Ankur Agrawal, Chuck Alpert, Arnon Amir, Alexander Andreopoulos, Rathinakumar Appuswamy, Sameh Asaad, Christian Baks, Davis Barch, Ralph Bellofatto, David Berg, Heinz Baier, Todd Christensen, Charles Cox, Steven Esser, Myron Flickner, Daniel Friedman, Sue Gilson, Chen Guo, Scott Hall, Ruud Haring, Charles Haymes, John Ho, Thomas Horvath, Ken Inoue, Subramanian Iyer, Jeffrey Kusnitz, Scott Lekuch, Zhuo Li, Jerry Liu, Michael Mastro, Emmett McQuinn, Steven

Millman, Roger Mousalli, Don Nguyen, Hao Nguyen, Tuyet Nguyen, Norm Pass, Kavita Prasad, Carlos Ortega-Otero, Ziad Saliba, Kai Schleupen, Jae-sun Seo, Ben Shaw, Koki Shimohashi, Arash Shokoubakhsh, Frank Tsai, Jose Tierno, Kyle Wecker, Shao-yi Wang, Ivan Vo, and Theodore Wong.

REFERENCES

[1] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[2] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[3] A. Cassidy, R. Alvarez-Icaza, F. Akopyan, J. Sawada, J. Arthur, P. Merolla, P. Datta, M. Gonzalez-Tallada, B. Taba, A. Andreopoulos, A. Amir, A. Esser, J. Kusnitz, R. Appuswamy, C. Haymes, B. Brezzo, R. Moussalli, R. Bellofatto, C. Baks, M. Mastro, K. Schleupen, C. Cox, K. Inoue, S. Millman, N. Imam, E. McQuinn, Y. Nakamura, I. Vo, C. Guo, D. Nguyen, S. Lekuch, S. Assad, D. Friedman, B. Jackson, M. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "Real-Time Scalable Cortical Computing at 46 Giga-Synaptic OPS/Watt with $\sim 100\times$ Speedup in Time-to-Solution and $\sim 100,000\times$ Reduction in Energy-to-Solution," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2014.

[4] A. Amir, P. Datta, W. Risk, A. S. Cassidy, J. A. Kusnitz, S. K. Esser, A. Andreopoulos, T. M. Wong, M. Flickner, R. Alvarez-Icaza, E. McQuinn, B. Shaw, N. Pass, and D. S. Modha, "Cognitive computing programming paradigm: A corelet language for composing networks of neuro-synaptic cores," in *International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2013.

[5] S. K. Esser, A. Andreopoulos, R. Appuswamy, P. Datta, D. Barch, A. Amir, J. Arthur, A. S. Cassidy, P. Merolla, S. Chandra, N. Basilico, S. Carpin, T. Zimmerman, F. Zee, M. Flickner, R. Alvarez-Icaza, J. A. Kusnitz, T. M. Wong, W. P. Risk, E. McQuinn, and D. S. Modha, "Cognitive computing systems: Algorithms and applications for networks of neurosynaptic cores," in *International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2013.

[6] D. S. Modha and R. Singh, "Network architecture of the long distance pathways in the macaque brain," *Proceedings of the National Academy of the Sciences USA*, vol. 107, no. 30, pp. 13485–13490, 2010.

[7] R. Ananthanarayanan and D. S. Modha, "Anatomy of a cortical simulator," in *Supercomputing 07*, 2007.

[8] R. Ananthanarayanan, S. K. Esser, H. D. Simon, and D. S. Modha, "The cat is out of the bag: cortical simulations with 10^9 neurons, 10^{13} synapses," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, (New York, NY, USA), pp. 63:1–63:12, ACM, 2009.

[9] R. Preissl, T. M. Wong, P. Datta, M. Flickner, R. Singh, S. K. Esser, W. P. Risk, H. D. Simon, and D. S. Modha, "Compass: A scalable simulator for an architecture for cognitive computing," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, p. 54, IEEE Computer Society Press, 2012.

[10] T. M. Wong, R. Preissl, P. Datta, M. Flickner, R. Singh, S. K. Esser, E. McQuinn, R. Appuswamy, W. P. Risk, H. D. Simon, and D. S. Modha, "10^{14}," *IBM Research Division, Research Report RJ10502*, 2012.

[11] J. Hsu, "How IBM got brainlike efficiency from the TrueNorth chip," *IEEE SPECTRUM (http://spectrum.ieee.org/computing/hardware/how-ibm-got-brainlike-efficiency-from-the-truenorth-chip)*, September 2014.

[12] E. Painkras, L. Plana, J. Garside, S. Temple, F. Galluppi, C. Patterson, D. Lester, A. Brown, and S. Furber, "SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation," *IEEE Journal of Solid-State Circuits*, vol. 48, pp. 1943–1953, Aug 2013.

[13] E. Stromatias, F. Galluppi, C. Patterson, and S. Furber, "Power analysis of large-scale, real-time neural networks on SpiNNaker," in *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2013.

[14] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J. Bussat, R. Alvarez-Icaza, J. Arthur, P. Merolla, and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, 2014.

[15] P. Merolla, J. Arthur, R. Alvarez, J.-M. Bussat, and K. Boahen, "A multicast tree router for multichip neuromorphic systems," *IEEE Transactions on Circuits and Systems I*, vol. 61, pp. 820–833, March 2014.

[16] J. Schemmel, A. Grubl, S. Hartmann, A. Kononov, C. Mayr, K. Meier, S. Millner, J. Partzsch, S. Schiefer, S. Scholze, *et al.*, "Live demonstration: A scaled-down version of the BrainScaleS wafer-scale neuromorphic system," in *International Symposium on Circuits and Systems (ISCAS)*, pp. 702–702, IEEE, 2012.

[17] J. Park, S. Ha, T. Yu, E. Neftci, and G. Cauwenberghs, "A 65k-neuron 73-Mevents/s 22-pJ/event asynchronous micro-pipelined integrate-and-fire array transceiver," in *Biomedical Circuits and Systems Conference (BioCAS), 2014 IEEE*, pp. 675–678, IEEE, 2014.

[18] S. Moradi and G. Indiveri, "An event-based neural network architecture with an asynchronous programmable synaptic memory," 2013.

[19] A. S. Cassidy, J. Georgiou, and A. G. Andreou, "Design of silicon brains in the nano-CMOS era: Spiking neurons, learning synapses and neural architecture optimization," *Neural Networks*, vol. 45, pp. 4–26, 2013.

[20] A. S. Cassidy, P. Merolla, J. V. Arthur, S. Esser, B. Jackson, R. Alvarez-Icaza, P. Datta, J. Sawada, T. M. Wong, V. Feldman, A. Amir, D. B.-D. Rubin, F. Akopyan, E. McQuinn, W. P. Risk, and D. S. Modha, "Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores," in *International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2013.

[21] A. J. Martin, "The Limitations to Delay-Insensitivity in Asynchronous Circuits," in *ARVLSI*, 1990.

[22] C. G. Wong and A. J. Martin, "High-level synthesis of asynchronous systems by data-driven decomposition," in *Proc. Design Automation Conference*, pp. 508–513, 2003.

[23] D. E. Muller and W. S. Bartky, "A theory of asynchronous circuits," in *Proc. Int'l Symp. Theory of Switching*, pp. 204–243, 1959.

[24] A. M. Lines, "Pipelined asynchronous circuits," tech. rep., Caltech, 1998.

[25] E. Izhikevich, "Which model to use for cortical spiking neurons?," *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1063–1070, 2004.

[26] Y. Shin, K. Shin, P. Kenkare, R. Kashyap, H.-J. Lee, D. Seo, B. Millar, Y. Kwon, R. Iyengar, M.-S. Kim, *et al.*, "28nm high-metal-gate heterogeneous quad-core CPUs for high-performance and energy-efficient mobile application processor," in *International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 154–155, IEEE, 2013.

[27] J. V. Arthur, P. A. Merolla, F. Akopyan, R. Alvarez, A. Cassidy, S. Chandra, S. K. Esser, N. Imam, W. Risk, D. B. D. Rubin, *et al.*, "Building block of a programmable neuromorphic substrate: A digital neurosynaptic core," in *International Joint Conference Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2012.

[28] D. Fang, S. Peng, C. LaFrieda, and R. Manohar, "A three-tier asynchronous FPGA," in *International VLSI/ULSI Multilevel Interconnection Conference*, 2006.

[29] A. J. Martin, "Compiling communicating processes into delay-insensitive VLSI circuits," *Distributed Computing*, pp. 226–234, 1986.

[30] R. Manohar, "An analysis of reshuffled handshaking expansions," in *Async7*, (Salt Lake City, Utah), March 2001.

[31] A. J. Martin and M. Nystrom, "CAST: Caltech asynchronous synthesis tools," in *Proc. of Fourth Asynchronous Circuit Design Working Group Workshop*, (Turku, Finland), June 2004.

[32] S. Burns and A. Martin, "Syntax-directed translation of concurrent programs into self-timed circuits," tech. rep., DTIC Document, 1988.

[33] R. Manohar, "Asynchronous VLSI systems." Class Materials for ECE 574 at Cornell University, 1999.

[34] "IEEE standard for Verilog hardware description language," *IEEE Standard 1364-2005*, 2006.

[35] K. L. Shepard and V. Narayanan, "Noise in deep submicron digital design," in *IEEE/ACM International Conference on Computer-aided Design (ICCAD)*, pp. 524–531, 1996.

[36] M. Nystrom, "Asynchronous pulse logic," tech. rep., Caltech, 2001. PhD Thesis, Chapter 4.

[37] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition*, 2001.

[38] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[39] A. Andreopoulos and J. K. Tsotsos, "50 years of object recognition: Directions forward," *Computer Vision and Image Understanding*, vol. 117, no. 8, pp. 827–891, 2013.

[40] A. Coates and A. Y. Ng, "The importance of encoding versus training with sparse coding and vector quantization," in *International Conference on Machine Learning (ICML)*, pp. 921–928, 2011.

[41] R. Kasturi, D. Goldgof, R. Ekambaram, G. Pratt, E. Krotkov, D. D. Hackett, Y. Ran, Q. Zheng, R. Sharma, M. Anderson, M. Peot, M. Aguilar, D. Khosla, Y. Chen, K. Kim, L. Elazary, R. C. Voorhies, D. F. Parks, and L. Itti, "Performance evaluation of neuromorphic-vision object recognition algorithms," in *Proc. 22nd International Conference on Pattern Recognition (ICPR'14)*, Aug 2014.

[42] A. Andreopoulos, B. Taba, A. Cassidy, R. Alvarez-Icaza, M. Flickner, W. Risk, A. Amir, P. Merolla, J. Arthur, D. Berg, J. Kusnitz, P. Datta, S. Esser, R. Appuswamy, D. Barch, and D. Modha, "Visual saliency on networks of neurosynaptic cores," *IBM Journal of Research and Development*, vol. 59, no. 2/3, pp. 9–1, 2015.

[43] J. Cong and G.-J. Nam, "Modern circuit placement: best practices and results," in *Springer Verlag*, 2007.

[44] N. Viswanathan, G.-J. Nam, C. Alpert, P. Villarrubia, H. Ren, and C. Chu, "RQL: Global placement via relaxed quadratic spreading and linearization," in *ACM/IEEE Design Automation Conference*, pp. 453–458, 2007.

[45] G.-J. Nam, C. Alpert, P. Villarrubia, B. Winter, and M. Yildiz, "The ispd2005 placement contest and benchmark suite," in *International Symposium on Physical Design*, pp. 216–220, 2005.

[46] A. Caldwell, A. Kahng, and I. Markov, "Can recursive bisection alone produce routable placements?," in *ACM/IEEE Design Automation Conference*, pp. 693–698, 2000.

[47] C. Alpert, G.-J. Nam, and P. Villarrubia, "Effective free space management for cut-based placement via analytical constraint generation," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, pp. 1343–1353, 2003.

[48] G.-J. Nam, S. Reda, C. Alpert, P. Villarrubia, and A. Kahng, "A fast hierarchical quadratic placement algorithm," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 678–691, 2006.

[49] J. Vygen, "Algorithms for large-scale flat placement," in *ACM/IEEE Design Automation Conference*, pp. 746–751, 1997.