Literature study: Asynchronous neurosynaptic-chips

Christoph Zimmermann

22. January 2016

Abstract

In the summer 2014 IBM unveiled a new chip, the second generation of its neurosynaptic chip that mimics the brain. At this time it was by device count the second largest chip in the world. It is a fully digital implementation of a neuronal network with one million individually programmable neurons and 256 million individually programmable synapses. The chip contains 5.4 billion transistors, but only draws 70 milliwatts of power! This is only possible because this chip and its architecture is based on asynchronous logic. The advantages of asynchronous Logic like lower power and/or higher performance, the insensitivity to production variations or temperature have been shown before and are well investigated. But asynchronous logic is still a niche. So this chip is very important, because it shows that the understanding and more important also the tools are ready to successfully design very large designs with asynchronous logic. In fact the used high-level design and simulation tools from the Hierarchical Asynchronous Circuit Kompiler Toolkit (HACKT) are released under the GPLv2 and are available on GitHub.

Table of contents

Introduction

In the context of the “Human Interface Technologies” specialization course at the Bern University of Applied Sciences Engineering and Information Technology we had to to a homework in the realm of asynchronous logic design. I've chosen this literature study work because I wanted to know more about the technologies that are involved in the recently presented research work of IBM in the area of neuronal networks implemented in hardware. Another point was, to understand why IBM has chosen the asynchronous design methodology in comparison to all other research project and also in contrast to any other logic chip IBM has designed before.

In essence this report is a aggregation of the research results that IBM published during the six years of the DARPA financed SyNAPSE project.

neuronal networks are a research field by its own and have a long history dating back to the 1950's but only in the recent years the available computing power made it possible to simulate and therefore analyse larger neuronal networks as they are found in biological systems. At the time of writing worm brains can be simulated in useful time on desktop workstations and a model of the rat brain can be simulated on very large supercomputers (still 400 times slower than real time).

The IBM/DARPA SyNAPSE project

In 2008 IBM started in the SyNAPSE project, which is financed by the Defense Advanced Research Projects Agency (DARPA), an agency of the U.S. Department of Defense responsible for the development of emerging technologies for use by the military. Over the six years DARPA was funding this project with a total of $102.6 million US$ according to http://www.artificialbrains.com/darpa-synapse-program:

> *SyNAPSE is a backronym standing for Systems of Neuromorphic Adaptive Plastic Scalable Electronics. It started in 2008 and as of January 2013 has received $102.6 million in funding. It is scheduled to run until around 2016. The project is primarily contracted to IBM and HRL (formerly Hughes Research Laboratories) who in turn subcontract parts of the research to various US universities.*

According to the head of the project at IBM Dharmendra Modha it involved over 200 persons from different institutions:

> *I am immensely grateful to our 200+ collaborators since 2008—spanning eight IBM labs and fabs, five universities, one start-up, and two Department of Energy laboratories. Finally, DARPA's mandate, metrics, and investment were absolutely vital. [Dharmendra Modha, IBM Fellow]*

At the Supercomputing Conference 2014 IBM presented a paper which gives a short overview of the development process and the different tools/chips that now form a whole ecosystem to develop algorithms and applications for neuronal networks and the TrueNorth chips:

> *[…] starting with a series of neuroscience-inspired simulations that led to a highly-optimized kernel that in turn led to a novel parallel, distributed, modular, scalable, event-driven, fault tolerant architecture that has become the basis of TrueNorth.*
>
> *These simulations allowed us to understand computation, communication, and memory constraints as well as challenges in scaling and real-time performance. Our optimized simulator, Compass, is functionally 1:1 equivalent with TrueNorth via co design. As a*

result, we have developed a cache of applications on Compass, such as multi-sensory feature extraction and pattern recognition; association and context processing; as well as information extraction, that now run without modification on TrueNorth, orders of magnitude faster and for orders of magnitude less energy. Scaling beyond a single chip, TrueNorth has a tillable structure enabling modular, scalable cognitive supercomputers. As a first step in this direction, we demonstrate four-chip and sixteen-chip boards with tiled arrays of TrueNorth processors that communicate without any additional peripheral circuitry. We also demonstrate a rack containing eight 1Gb Ethernet cards each with a single TrueNorth processor. Looking to the future, we can imagine replicating the "1% human-scale" simulations that required 16 racks of Blue Gene/P and ran 400× slower than real-time on a TrueNorth system that requires only one rack, would run in real-time and consume an estimated 128, 000× less energy. These systems pave the way for multimedia cloud processors capable of dealing with a myriad of sensors pervasive in today's world. To enable applications across a wide spectrum spanning neural networks and machine learning, we have developed an end-to-end ecosystem [...][CAS14]

Corelets

It is going far too deep for this report to explain the higher levels of the ecosystem that IBM has developed through the SyNAPSE program. But nevertheless it is well worth to give an overview to the coals of the Corelet Programming Environment:

Applications for the TrueNorth processor are developed in the Corelet Programming Environment (CPE), a new, object-oriented, compositional language and development environment that promotes efficient, modular, scalable, and collaborative TrueNorth software. A corelet is a functional encapsulation of a network of neurosynaptic cores that collectively perform a specific task. Object-oriented corelets can seamlessly build hierarchically composable networks while sharing underlying code and unified network interfaces. Our approach is conceptually akin to Banavar's framework for compositional modularity, and dramatically improves code reuse and scalability. Corelets are collected in the corelet library — an ever growing repository of reusable corelets covering numerous TrueNorth implementations of seminal algorithms, including linear and non-linear signal and image processing; spatiotemporal filtering; saliency; object detection, classification, and recognition; and real-time audio and video analytics, a representative set of which have been selected for testing here. [CAS14]

The TrueNorth Chips

Our main focus here is the underlying hardware that was built by IBM, the TrueNorth architecture and the chips utilizing it.

The goal of the IBM project was to develop a fully digital, scalable, low-power, non-von-Neumann architecture to execute synaptic operations. An important requirement was a 1:1 correspondence between neuronal networks implemented in software and in this new hardware architecture.

Architecture

As the TrueNorth architecture is essentially a neuronal-network it is constructed out of neurons which are equivalent to the grey matter in our brains (the “computing” part) and synapses which are equivalent to the white matter, the interconnection network between the neurons.

In contrast to earlier projects IBM developed a fully digital architecture (More in Chapter: Comparison to chips from other projects)

Neuron

This chapter gives a very brief overview of the neuron model that is used in the TrueNorth architecture. In the literature about neuronal networks there are many proposed neuron models with different behaviour, functionality and therefore complexity. IBM has chosen the integrate-and-fire spiking neuron with binary output, which is one of the simplest neuron models. Each neuron has 256 inputs. This was also criticised by other researchers:

> *This type of neural net that has never been shown to yield accuracy anywhere close to state of the art on any task of interest[…]. Spiking neurons have binary outputs (like neurons in the brain). The advantage of spiking neurons is that you don't need multipliers[...]. But to get good results on a task like ImageNet you need about 8 bit of precision on the neuron states. To get this kind of precision with spiking neurons requires to wait multiple cycles so the spikes "average out". This slows down the overall computation. [LEC14]*

The most detailed information about the neurons are given in [CAS13]:

> *Page 3: Our neuron model is based on the leaky integrate-and-fire neural model with a constant leak, which we augmented in several ways. We begin by briefly reviewing the classic leaky integrate-and-fire neural model, followed by an in-depth description of our neuron model. […] Page 5: The primary objective of our neural specification is for building synthetic cognitive applications, motivating a neuron specification that is rich in computational expressiveness. It must not only have complex spiking output patterns to diverse inputs, but we must also be able to harness those complex behaviours to perform useful computation. In this section, we present a library of functions that are computable using our neuron specification. Most can be computed with a single neuron, while other more complex functions require a few neurons.[...] Page 9: In this paper, we have presented a digital, reconfigurable, versatile spiking neuron model that supports one-to-one equivalence between hardware and simulation and is implementable using only 1272 ASIC gates (924 gates for the model computation and 348 gates for the random number generator). We demonstrated that the parametric neuron model supports a wide variety of computational functions and neural codes.*

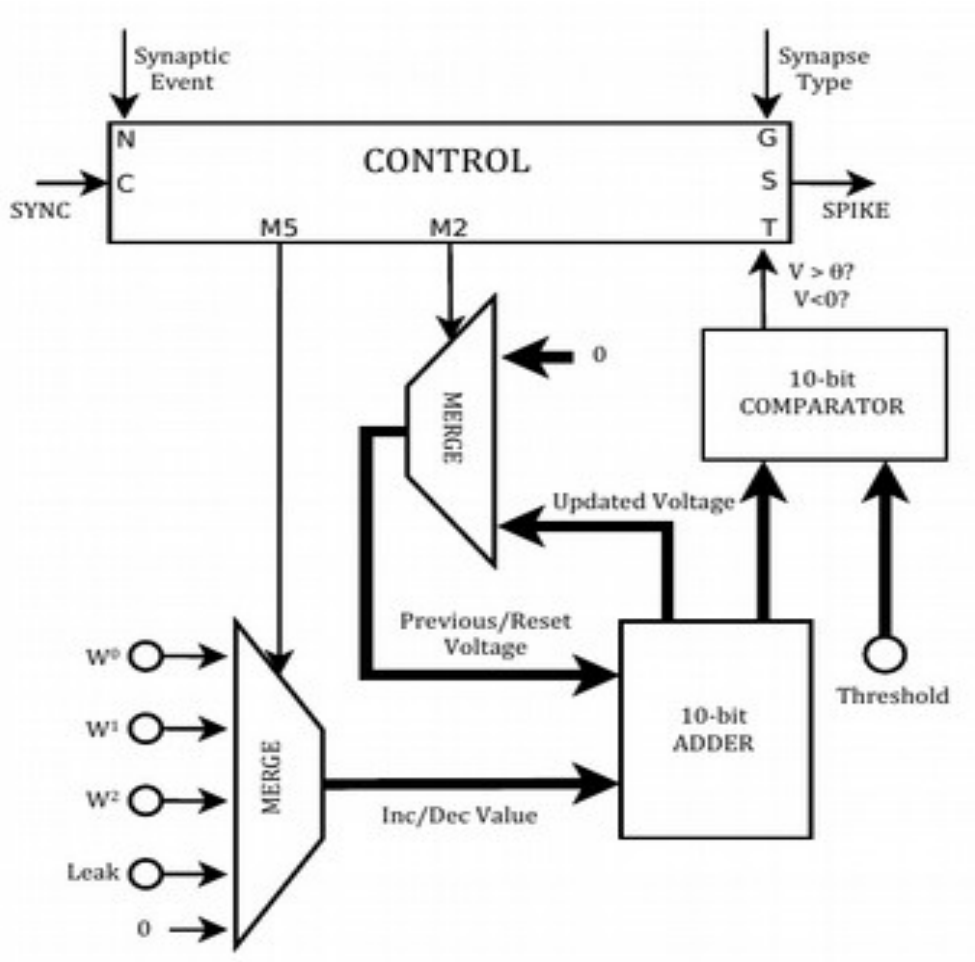


Illustration 1: Block diagram of the neuron. The control block interfaces with the crossbar, directs information flow in the datapath, synchronizes with the global time step and outputs a spike when the neuron voltage exceeds its threshold. The datapath elements update the voltage after each synaptic event and check for a spike when directed by the control. [NAB12]

Type	Name (# of neurons)
Arithmetic	absolute value (3), addition (1), division (2), fixed-gain div./mult. (1), log (1), min (2), max (2), modulation (1), multiplication (1), rate clip (2), rate match (3), sigmoid (1), square root (3), square (2), subtraction (1)
Control	bistable (1), tristable (1), delay (1), one shot (1), pass gate (1)
Data Generation	spontaneous (1), ramp up/down (3), triangle wave (5), random distributions (1)
Logic	AND (1), NAND (1), NOR (1), NOT (1), OR (1), XNOR (3), XOR (3)
Memory	binary (1), rate store (1)
"Neural"	integrate-and-fire (1), bursting (2), coincidence (1), McCulloch-Pitts (1), Boltzmann (1), motion history (1), on/off pair (2) , onset/offset pair (2)
Signal Processing	decorrelation (1), rate attractor (1), convolution (1), low-pass filter (1), high-pass filter (1), band-pass filter (1), spatiotemporal filter (1), Bloom filter (1)
Probabilistic	noisy-OR: prob. union (1), noisy-AND: prob. intersection (1)
Time-to-Spike	L1 distance (2), max (N), min (N), median (N+1), coincidence (1), anti-coincidence (1), matched filter (1)

Table 1: Neuron function library summary [CAS13]

Synapses

With this simple neuron model, with only binary outputs (spiking, not-spiking) the synapses are nothing more than a crossbar memory between axons, the outputs of neurons, and dendrites, the inputs of neurons. In older papers there was a specialised SRAM cell that was optimised for on chip learning ([SEO11]) but it seems that is not used in more recent implementations:

> *The crossbar array is configurable so that arbitrary networks can be set up in the system (e.g. Axons 1, 2, and 3 are connected to the first neuron in the 2D neuron array in illustration 2. Each row of the crossbar corresponds to an axon, each column corresponds to the input of a neuron (the dendrite), and the junctions are binary synapses implemented by a two terminal memory cell (e.g., SRAM). Thus each of the N neurons may get up to K synaptic inputs depending on the activity in the axons and the configuration of the crossbar. We chose K as 1024 and N as 256 resulting in 1024×256 crossbar synapses and an enormous configuration space. [NAB12]*

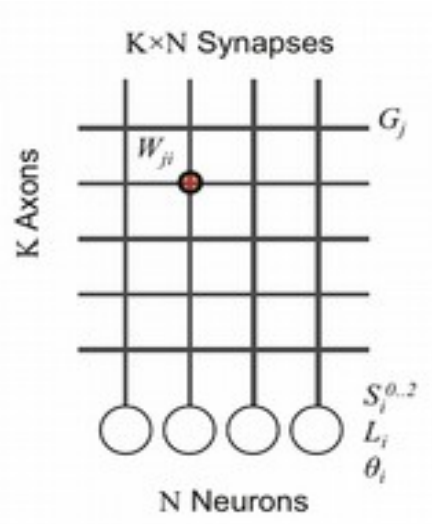


Illustration 2: The core consists of axons, represented as rows; dendrites, represented as columns; synapses, represented as row–column junctions; and neurons that receive inputs from dendrites. [MER11]

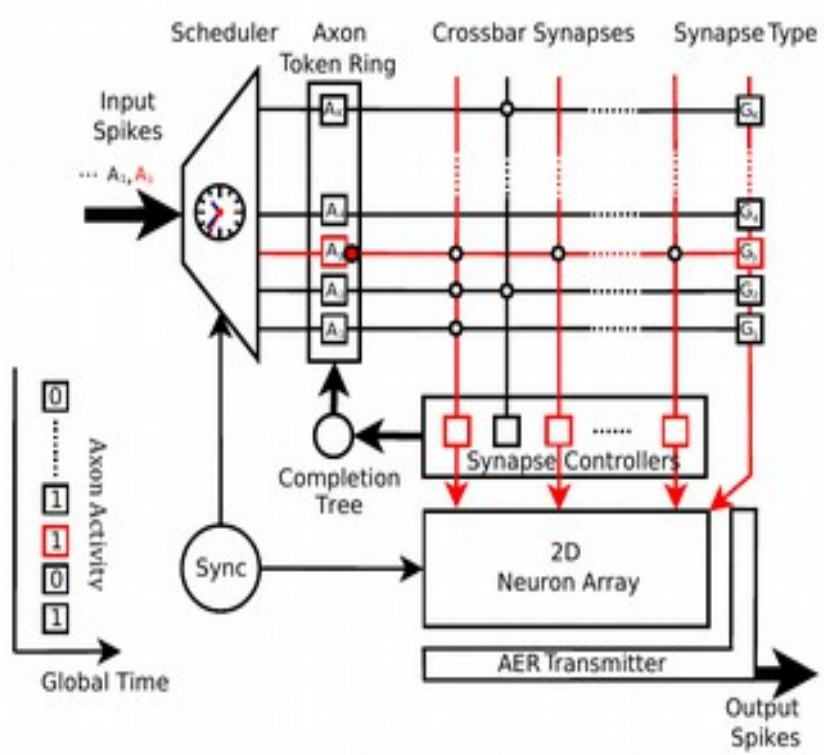
Neurosynaptic core

The neurosynaptic core is the basic building block for the TrueNorth architecture. It is in essence a very small brain, a neuronal-network, that can be connected to many other such neurosynaptic cores. This makes the architecture very scalable. It is possible to arrange several cores on one chip and also to interlink cores distributed over several chips.

One core consists of 256 neurons (outputting 256 axons) and 1024·256 synapses (representing 1024 axon inputs). Internally the operation of such a core is parallel but the communication between cores is serialized. Each output axon is encoded as a AER packet and is sent through a packet oriented on-chip network to the other cores:

> *Spikes events are sent to and from the core using address event representation (AER) packets. On the output side, an AER transmitter encodes spiking activity by sending the locations of active neurons through a multiplexed channel, leveraging the fact that the bandwidth of wires (easily larger than 100s of MHz) is orders of magnitude larger than the bandwidth of biological axons (in the 10's of Hz range). The spikes can be sent off chip, or routed to an axon of another core via a look-up table. On the input side, an AER receiver delivers incoming spikes to the appropriate axon at a predetermined time configured in a scheduler block. As spikes are serviced sequentially, their addresses are decoded to the crossbar where all 256 synaptic connections of an active axon are read out in parallel.[NAB12]*

An interesting and unexpected finding was, that the operation of such a core is dependent on a global synchronisation clock. This synchronises the operation of all interconnected cores, so from a higher level the resulting neuronal-network behaves like a fully-synchronous circuit implemented with a traditional design methodology. The details can be seen in the next two chapters.



NAME	DESCRIPTION	SIZE AND TYPE
V_i	Voltage of Neuron i	10 bit signed variable
$W_i^{0..2}$	3 Synaptic Weights of Neuron i	9 bit signed constant
L_i	Leak of Neuron i	9 bit signed constant
θ_i	Threshold of Neuron i	8 bit unsigned constant
S_{ji}	Connection between axon j and neuron i	Binary constant
G_j	Type of Axon j	3 distinct constants
A_j	State of Axon j	Binary Variable

Illustration 3: Top: Architecture of the neurosynaptic core with K axons and N neurons. Each junction in the crossbar represents a synapse between an axon (row) and dendrite (column). Each neuron has a dedicated column in the crossbar. Active synapses are represented by an open circle in the diagram. An example sequence of events in the core is illustrated. The scheduler accepts an incoming address event and communicates with the axon token-ring. The token-ring activates axon 3 (A3) by asserting the third wordline of the SRAM crossbar array. As a result, a synaptic event of type G3 is delivered to neurons N1 , N3 , and NM . The AER transmitter sends out the addresses of these neurons if they consequently spike. Bottom:State variables and parameters of the system. All values are represented as integers, and all constants are configurable at start-up [NAB12]

Global discrete-time operation

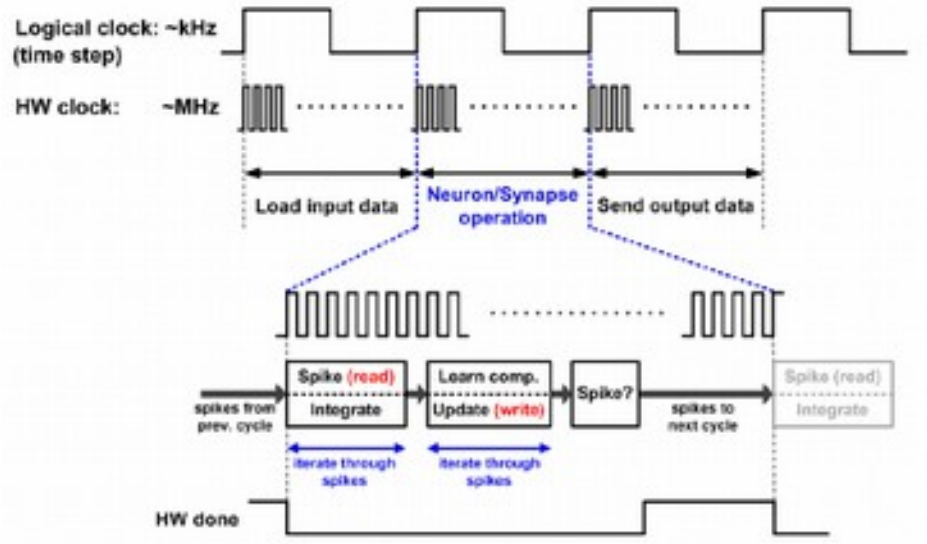


Illustration 4: Timing operation of overall chip [SEO11]

An example sequence of operation in the core is illustrated in Illustration 4. The operation has two phases during each time step. The positive edge of a global synchronization clock (Sync) initiates the first phase of operation. In this phase, address-events along with their time stamps are sent to the core and are received by the scheduler. The scheduler evaluates the time stamps and asserts the appropriate axons that go into a token-ring. The units in the token-ring that receive active axons assert the rows of the crossbar in a mutually exclusive manner. Once a word line in the crossbar is activated all the neurons that are connected to the axon (corresponding to the 1's in the row) receive an input spike along with information about the type of the axon. The neurons update their voltages as axon events come in. The first phase needs to complete within the first half of the global synchronization clock (that usually appropriate has a period amount of 1 of millisecond)—giving us a precise margin in which neural updates need to complete for potentially all 1024 axon inputs. In the second phase of operation, the negative edge of the synchronization clock is detected by all the neurons. On receiving this event, neurons whose voltages have exceeded their respective thresholds produce spikes in their output ports. The spiking addresses are encoded by the AER transmitter and sent out of the core sequentially. This phase needs to complete within the other half of the global clock — i.e. the AER transmitter has to guarantee that it can service 256 potential spikes within the global time step. A 1 millisecond global clock period (typical temporal precision in biological neural networks) means that the performance requirements of the circuits in the two phases of operation are easily met. ***Breaking neural updates into two phases ensures that the hardware is always in sync with an equivalent software simulation at the end of each time step****. [NAB12]*

Local event driven implementation (QDI)

The inner working of the neurosynaptic core is fully asynchronous implemented with quasi-delay-insensitive (QDI) circuits. To implement this circuits IBM is using Communicating Hardware Processes (CHP) which was invented by Alain J. Martin from the California Institute of Technology (Caltech), but it seems that Martin was not involved in the design of the TrueNorth architecture. CHP and it's design flow will be discussed in the chapter Design Flow and Tools.

We implement the architecture of the neurosynaptic core using asynchronous QDI circuits that are synchronized with the global time step. In this section we describe the concurrent processes of our architecture using Communicating Hardware Processes that can be synthesized into QDI asynchronous circuits using Martin's synthesis method. [NAB12]

Silicon implementations

At the time of writing there are two known implementations of the TrueNorth neurosynaptic architecture. The first is a small prototype chip using 45 nm-SOI technology. As explained later there are four different version of this chip to test different neuron implementations.

The second TrueNorth chip is a giant step forward as IBM did the step from a small prototype to a big-iron in one iteration. The second chip contains one million neurons using 5.4 billion transistors on an area of 4.3 $cm^{2.}$

45 nm version, 256 neurons, 2011

Technology:

- 45 nm CMOS-SOI (silicon on insulator), probably from IBM.
- Near-threshold operation at 0.53 V.

Chip size: 2500 μm^2.

Transistor count is unknown but with the assumption that this chip is 4096 time smaller than it's successor it contains around 1.3 million transistors.

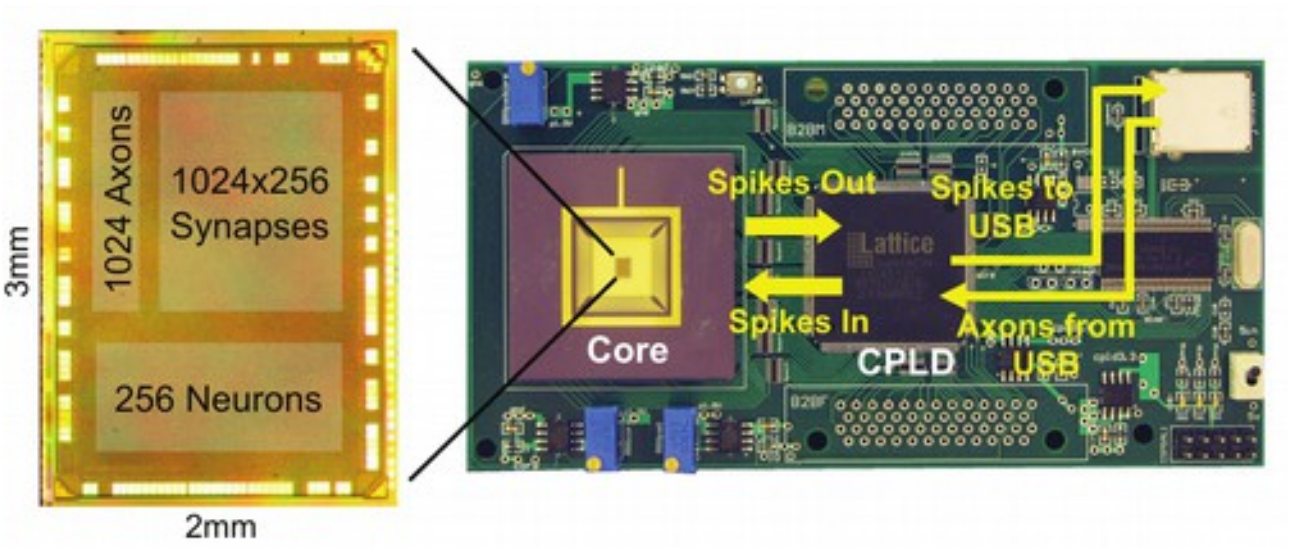


Illustration 5: (left) Neurosynaptic die measures 2mm × 3mm including the I–O pads. (right) Test board that interfaces with the chip via a USB 2.0 link. Spike events are sent to a PC for data collection, and are also routed back to the chip to implement recurrent connectivity. [MER11]

The details of this chip and the different neuron implementations were published in 2011:

The chip operates in a synchronous fashion with a ~1kHz "logical clock" (corresponding to ~1ms biological time step) that enables "real-time" communication with the external world. Internally, the hardware control circuits run at a faster rate

(~1MHz) to step through the integrate-and-spike phase and two learning phases (pre- and post-synaptic) for each neuron spiking event. [...]Through the use of clock-gating techniques, the hardware clock is applied only when signals are queued for processing. In this way, an event-driven system with minimal power dissipation is realized; only those neurons that exceed the threshold in a time step participate in communication and only their associated synapses participate in weight updates. […]

Slim Neuron: Since configuration latches can consume significant area, one variant fixes all spiking and learning parameters for a two-layer learning network. This achieves a 56% neuron area reduction relative to the base design.

4-bit Synapse: To compare the binary synapse design with conventional analog weight approaches, one variant implements synapses as 4-bit weights. The memory element combines four copies of the transposable SRAM cell with common word line terminals. Synapse updates occur in a three-step read-modify-write procedure in accordance with learning rules. The LFSR module is not needed in this variant as synapse weights are not updated probabilistically, which enables a 20% neuron area reduction.

Low Leakage Variant: In a system targeting real-time operation with a ~1MHz hardware clock, leakage power far outweighs active power. By leveraging ultra-high-Vt devices, a ~3X leakage current reduction is achieved compared to the base design with super-high-Vt devices, at the cost of increased σVt due to increased transistor channel doping, which raises the minimum operating voltage of the memory array. [MER11]

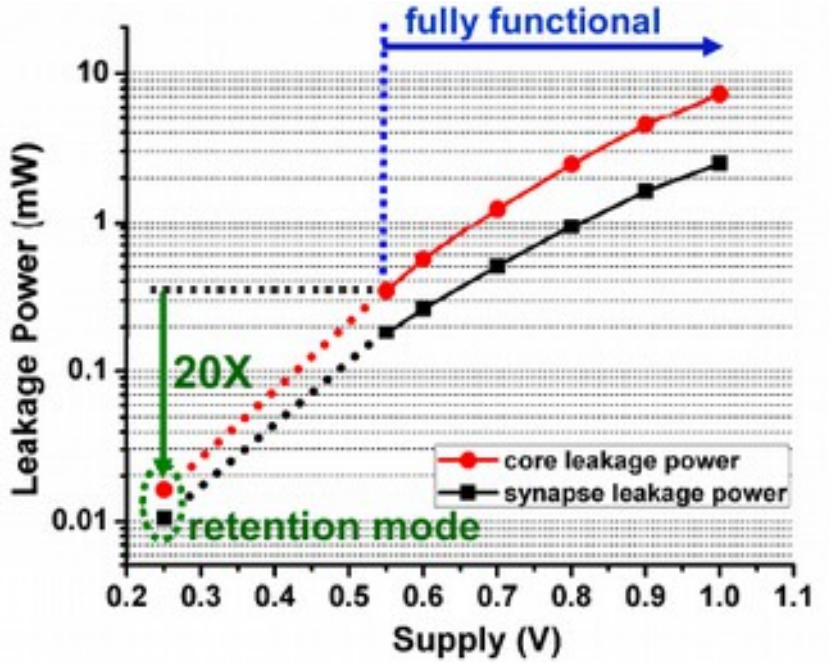


Illustration 6: Voltage scaling and power-gating of the low-leakage variant chip.'core' includes 256 neurons, FSM, priority encoder, and clock distribution.'synapse' includes 64K synapses and according peripheral circuits. [SEO11]

28 nm version, one million neurons, 2014

In 2014 IBM presented the TrueNorth chip with one million neurons implemented on one single chip. The presentation of chip was the reason to write this report, because it is several steps ahead of all other known neuronal-network implementations together with a unbelievable low power consumption of only 70 mW with full workload.

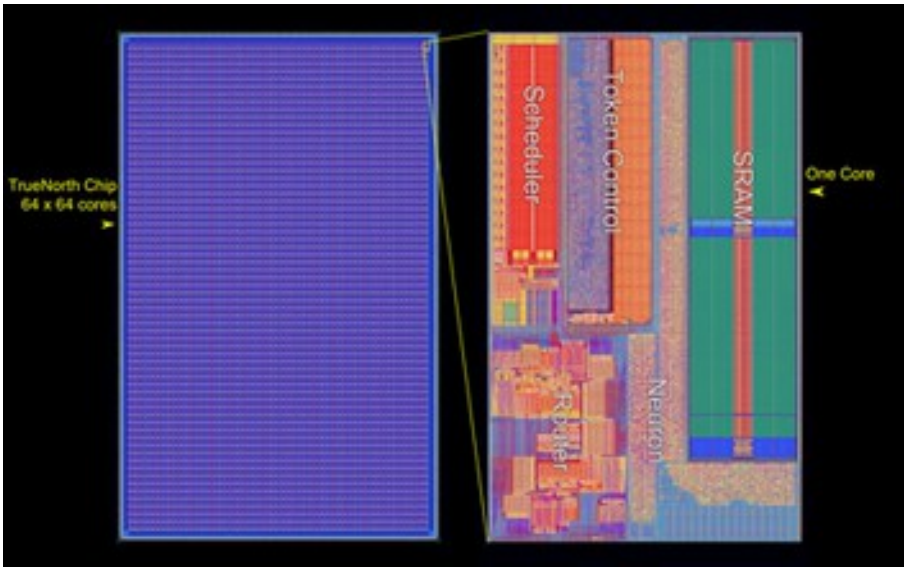


Illustration 7: TrueNorth Chip Core Array

Technology:

- Samsung's 28 nm CMOS process
- 5.4 billion transistors in 4.3 cm^2, at that time (Summer 2014) second largest CMOS chip in the world
- Power consumption of only 70 mW

Features (all according to IBM):

- One million individually programmable neurons
- 256 million individually programmable synapses on chip
- 4096 parallel and distributed cores, interconnected in an on-chip mesh network
- Over 400 Mbit of local on-chip memory (~100 Kbit per core) to store synapses and neuron parameters
- 26 pJ per synaptic event, which is the lowest recorded energy for any large-scale neuromorphic system, and five orders of magnitude lower than von Neumann computers
- 20 mW/cm^2 power density which comparable to cortex but is three to four orders of magnitude lower compared to 50-100 W/cm^2 for a CPU
- 30 times less area per neuron than than previous approaches (14.7 μm^2 compared to 400 μm^2 in analog)

Interesting was also to see the influence of the ever increasing CMOS process development costs:

> *Collaboration with Samsung was critical in gaining access to their advanced 28nm foundry process that allowed balancing the low active power of the architecture with matching low power of the underlying silicon technology. [Dharmendra Modha, IBM Fellow]*

Comparison to chips from other projects

The TrueNorth architecture is not the first hardware implementation of a neuronal network, it is the next step in a longer history of other project. As mentioned before, traditionally neurosynaptic chips have been built using analog circuits or a hybrid of analog neurons and digital synapses. The biggest advantage is that analog neurons are using only a few transistors for each neuron. So it was natural to use them to implement large neuron arrays. But they always have the problem that due to process variations the behaviour of the neurons is not deterministic. The other recent problem is that manufacturing analog chips is getting harder and harder at smaller processing nodes. So the down-scaling has more or less stopped at 180 nm. In comparison digital circuits can be manufactured at the moment with 14 nm processes.

In [CAS14] IBM gives a small overview of former neurosynaptic chips and there size:

> *Historically, neuro-inspired computers followed Carver Mead's pioneering work, modeling biological neural circuits using silicon analog electronics, including neurons,ion channel models, and winner-take-all circuits. Larger systems, for example Neurogrid (65k neurons), CAVIAR (45k neurons), and IFAT (65k neurons) combine arrays of these analog components with an external memory to store connectivity information. More recent architectures include the BrainScaleS project, demonstrated a wafer-scale system (20cm diameter wafer) with a total of 200 thousand analog neurons and 40 million addressable synapses, and consumes roughly a kilowatt. The SpiNNaker project demonstrated a 48-chip system (each chip has 18 ARM processors) simulating a total of 250 thousand neurons and 80 million synapses (implemented via off-chip memory), which consumes 36W. [CAS14]*

And in From [ART12]:

> *A long standing goal of the neuromorphic community is to build compact, scalable, and power-efficient neural hardware that is supported by a corresponding software environment. Two recent notable projects with similar aims are PyNN [20] and the SpiNNaker project. [ART12]*

The BrainScaleS project is from the University of Heidelberg and the SpiNNaker project is from the University of Manchester (https://spinnaker.cs.manchester.ac.uk). Both projects are part of the Human Brain Project (HBP) which is a European Commission Future and Emerging Technologies Flagship project (Horizon 2020): https://www.humanbrainproject.eu

Comparison with neuFlow

More interestingly is the comparison between the asynchronous TrueNorth architecture with the synchronous neuFlow-SoC architecture as it is presented in [HUN12] because this is also a fully digital implementation of a non-von-Neumann architecture. It is based on much more complex convolution kernels and is more orientated to implement streaming data flow graphs but is also usable as neuronal network.

Process	IBM SOI 45nm
Chip area	2.5 x 5 mm^2
Supply Voltages	1V V_{core}, 1.8V and 3.3V $V_{I/O}$
Target frequency	400MHz
Estimated average power	570mW
Peak performance	160 GOPS
GOPs per Watt	$\sim$ 254
Number of transistors, memories, etc.	23.6 million transistors and 75KB 2-port RAM
Pin count	317 (299 I/Os and 18 P/Gs)
Packaging	Flip-chip

Table 2: Summary of chip specifications. The neuFlow die area is 12.5 mm^2 [HUN12]

	CPU[1]	mGPU[2]	GPU[3]	neuV6[4]	neuIBM[5]
Peak GOPs	10	182	1350	160	**160**
Real GOPs	1.1	54	294	147	**147**
Power (W)	30	30	220	10	**0.579**
GOPs/W	0.04	1.8	1.34	14.7	**254**

[1] CPU: Intel DuoCore, 2.7GHz, optimized C code
[2-3] mGPU, GPU: a mobile Nvidia GT335m and a high-end GTX480
[4] neuV6: neuFlow prototyped Xilinx Virtex 6 FPGA
[5] neuIBM: 45nm IBM SOI process neuFlow (*this work*)

Table 3: neuFlow-SoC performance comparison

The two systems are not easy to compare from this figures because it is unknown how to convert the operation/second numbers from neuFlow to synaptic-operations/second from TrueNorth.

What can be seen is, that the neuFlow SoC is seventeen times larger than the 45 nm TrueNorth but its power consumption is 500 times larger.

Design Flow and Tools

To implement this circuits IBM is using Communicating Hardware Processes (CHP). IBM has partnered with the Computer Systems Laboratory of the Cornell University for this work. This can be seen by the several IBM papers ([MER11], [ART12], [NAB12], [CAS14]) which include Rajit Manohar as an author.

The CHP Language

Designing with the CHP language family is strongly formalised. CHP is only the high-level language that is used to describe the system behaviour. During the design flow two other domain-specific languages HSE and PRS come into account. Alain J. Martins explanation:

> *Page 12: We also believe that it is not practical or wise to try and express the whole synthesis of a VLSI system from a high-level specification entirely within one single notation: the hierarchy of the three notations—CHP, HSE, PRS—is a better solution. The design of a programming language is a difficult compromise between simplicity and soundness on the one hand, and expressive power on the other hand.[ALA11]*

Designing with this languages follows the design flow as it can be seen in the illustrations 8 and 9. In the paper [MAN00] from Rajit Manohar is a complete description:

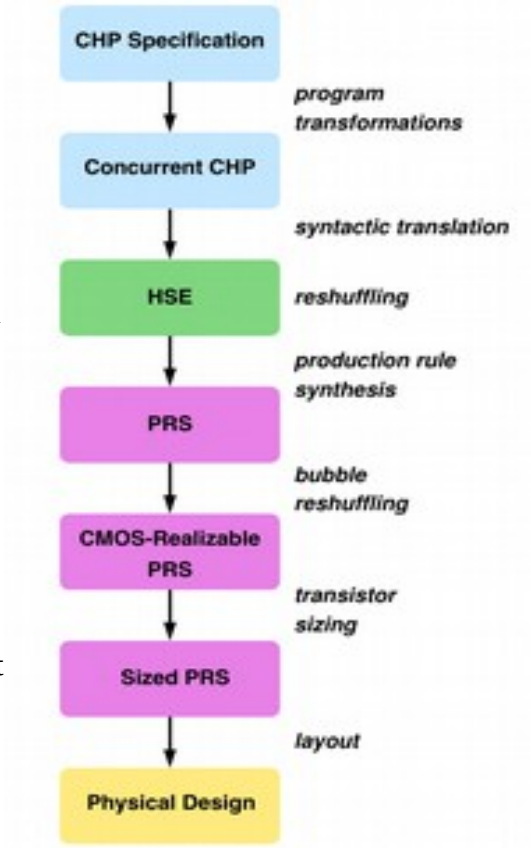


Illustration 8: Top-down design flow [MAN00]

> *The initial specification of the circuit to be designed is described in the programming notation CHP. The constructs of the language [...] include primitives for computation and communication. The next step in the design flow is to decompose the sequential CHP program into a number of concurrently operating CHP programs using program transformations. This is the part of the design flow where most architectural choices are made. Decisions about pipeline structure, degree of pipelining, and control/data partitioning are taken at this step. Transformations such as process decomposition and projection are used to ensure that correctness is preserved when translating a sequential CHP program into a number of concurrent parts. The next step in the design flow is to translate each CHP process into handshaking expansions (HSE). Handshaking expansions can be thought of as a restricted subset of CHP, where all variables are Boolean-valued and all communication actions are replaced with synchronization protocols (handshake protocols). This transformation is syntax-directed, since each CHP action can be locally replaced with the appropriate HSE. Once the HSE for a process is obtained, it can be reshuffled for performance optimization.*
>
> *The next step in the design flow is to translate HSE into a production rule set (PRS), a digital abstraction for CMOS circuits. A production rule is of the form $G \rightarrow x\uparrow$ or $G \rightarrow x\downarrow$, where G is a Boolean expression and x is a variable. The rule $G \rightarrow x\uparrow$*

corresponds to a pull-up network, and G →x↑ corresponds to a pull-down network. The translation from HSE to PRS guarantees that the resulting PRS satisfies two key properties: stability and non-interference. Stability and non-interference are necessary and sufficient for hazard-free circuit behaviour. Once production rules are obtained, they must be made CMOS implementable - rules of the form G →x↓ must be implementable with n-transistors, and rules of the form G →x↑ must be implementable with p-transistors. This transformation is known as bubble reshuffling, because it involves the insertion or movement of inverter "bubbles" in the circuit.

The next step in the design flow is to take the bubble reshuffled PRS and determine the appropriate transistor sizing to optimize the desired performance metric. Once transistor sizes are chosen, we can implement the circuit in CMOS. For QDI circuits, transistor sizing does not affect correctness except for the isochronic fork assumption. [MAN00]

More information can be found in the several papers published by the asynchronous VLSI group at Caltech (http://async.caltech.edu/publications.html) and the Computer Systems Laboratory of the Cornell University (http://www.csl.cornell.edu/~rajit/pub_all.html)

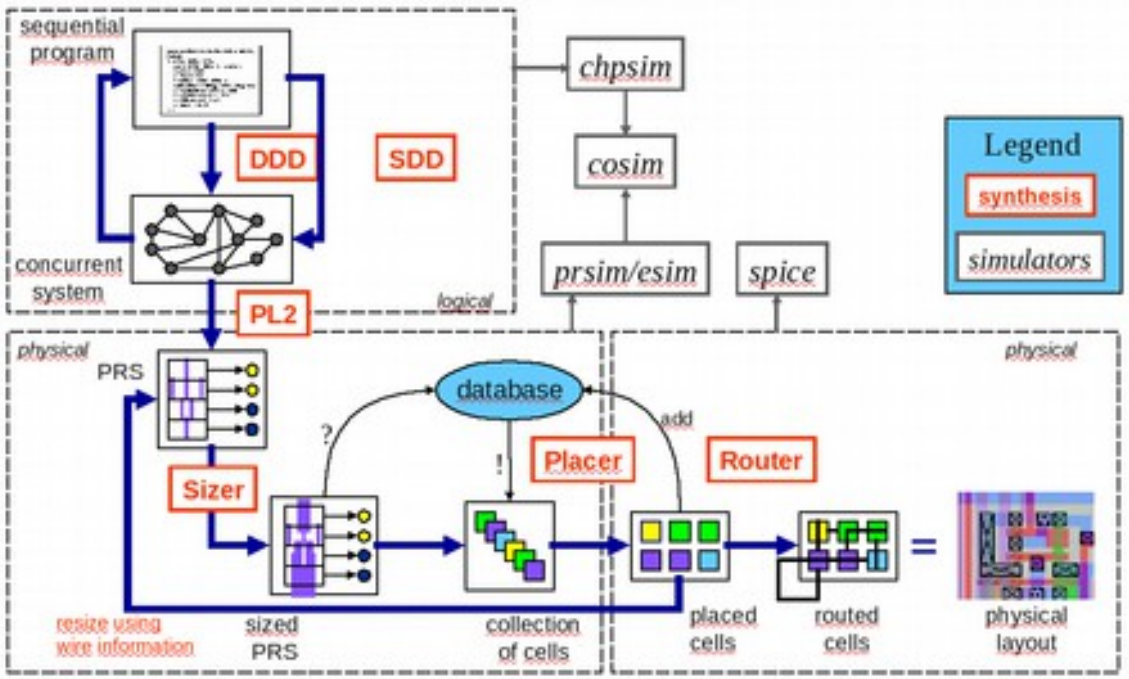


Illustration 9: More detailed design flow ([ALA07], Slide 39)

Brief CHP language history

CHP was invented by Alain J. Martin from the California Institute of Technology (Caltech), as described by himself it is related to the older CSP from C.A.R. Hoare but has a different approach:

Page 1: CHP already has a long history. It took shape in the early 1980's to describe distributed computations in general. Starting from C.A.R. Hoare's CACM version of CSP, and Dijkstra's guarded commands, the early prototype of CHP introduced the notion of channels, and the probe construct, neither of which exists in CSP. The combination of processes with channels, send/receive communication, and probe operation on channel produced a very useful, albeit restricted, distributed programming model which was successfully used both for teaching distributed computing and for

> *implementing distributed computations on experimental multiprocessors. But it was the development of a synthesis method for asynchronous VLSI that prompted the extension of the prototype notation into CHP. [ALA11]*

In the beginning of 1999 Rajit Manohar a former PhD student of Alain J. Martin and involved in several projects like the asynchronous MIPS R3000 processor moved from Caltech to the the Cornell University.

In 2003 a team around Alain J. Martin at Caltech and in his own company, Situs Logic, started developing a complete CAD toolset to design asynchronous circuits: CAST. It was used for several projects from different teams. Situs Logic is specialised in radiation-hard-by-design microprocessors and microcontrollers based on there own proprietary asynchronous technology.

In 2004 at the Cornell University an effort to replace the previous generation of asynchronous design tools was started. The result is the HACKT which is explained in the next chapter.

Another outcome of the research at the Cornell University was the founding of Achronix Semiconductor Corporation, the multi-GHz FPGA company that is based on asynchronous FPGA technology. The foundation of this was the PhD. thesis of John Teifel. Teifel is one of the founding members along with Rajit Manohar and others.

Hierarchical Asynchronous Circuit Kompiler Toolkit (HACKT)

As to my knowledge the most advanced and best maintained toolkit for the CHP design flow is HACKT. From the project homepage at http://www.csl.cornell.edu/~fang/hackt/index.php;

> *The Hierarchical Asynchronous Circuit Kompiler Toolkit (HACKT) is a project originally developed at Cornell University for aiding the design of asynchronous circuits. The project continues to be maintained by David Fang, the original author.*
>
> *HACKT contains a collection of tools to aid asynchronous circuit design. (It is also usable for synchronous circuit design.)*
>
> - *Compiler -- the HAC input is compiled into object files, which are passed to other back-end tools*
> - *Simulators -- high-level and digital circuit simulators, and a VPI co-simulation module*
> - *Netlist Generator -- producing SPICE decks of circuits*
> - *Analysis -- the simulation environments also have strong support for analyzing concurrent behavior*
>
> *As a testament to its scalability, HACKT has been used to design, simulate, and test generations of multi-billion transistor chips at Achronix Semiconductor since 2008.*

David Fang was a PhD student of Rajit Manohar at the Cornell university and is working as Principal Engineer (Core Technology Team) for Achronix Semiconductor Corp. So it is most likely that IBM was using the HACKT to develop the TrueNorth chips.

The source code of HACKT is open source and licensed under the GNU General Public License (GPLv2+). It is available from GitHub: https://github.com/fangism/hackt

Conclusion

When I started digging into the topic it grew more and more. There are very different topics involved in it and in each of them is a huge amount of research papers available. Especially the history of the development of the CHP language based design flow is not a straight line and took a while until I found the most recent state of research. But it was well worth the effort.

As it is shown above CHP is silicon proven for very large asynchronous design ($> 5 \cdot 10^9$ transistors). In fact the high-level design and simulation tools are released under the GPLv2 and are available on GitHub!

In my opinion the other neurosynaptic chip projects which are using analog circuits to construct neuronal networks (University of Heidelberg, Stanford) are flogging a dead horse. Analog manufacturing processes have more or less stopped in down scaling at 180 nm. Process variances are always problematic for analog neurosynaptic chips and they get worse with smaller structures. This makes it also problematic to develop and simulate the behaviour of a neuronal network without the actual chip.

One interesting finding, which was very unexpected, was that the TrueNorth architecture is locally asynchronous but globally synchronous. From my perspective I would guess that this structure was chosen to avoid a lot of complexity on the higher levels. With the global synchronisation all the higher levels (Simulators, compilers, software developers) don't need to handle or understand the asynchronous behaviour of the chip because the resulting neuronal network is synchronous. The future will tell us if this global synchronisation is a limitation to get rid of (with all the consequences in the higher levels) or that it is a good compromise that survives many chip generations.

Bibliography

CAS14: Andrew S. Cassidy, Rodrigo Alvarez-Icaza, Filipp Akopyan, Jun Sawada, John V. Arthur, Paul A. Merolla, Pallab Datta, Marc Gonzalez Tallada, Brian Taba, Alexander Andreopoulos, Arnon Amir, Steven K. Esser, Jeff Kusnitz, Rathinakumar Appuswamy, Chuck Haymes, Bernard Brezzo, Roger Moussalli, Ralph Bellofatto, Christian Baks, Michael Mastro, Kai Schleupen, Charles E. Cox, Ken Inoue, Steve Millman, Nabil Imam, Emmett McQuinn, Yutaka T. Nakamura, Ivan Vo, Chen Guo, Don Nguyen, Scott Lekuch, Sameh Assad, Daniel Friedman, Bryan L. Jackson, Myron D. Flickner, William P. Risk, Rajit Manohar, Dharmendra S. Modha, Real-Time Scalable Cortical Computing at 46 Giga-Synaptic OPS/Watt with ~100x Speedup in Time-to-Solution and ~100,000x Reduction in Energy-to-Solution, 2014, Proceedings of Supercomputing 2014

LEC14: Yann LeCun, My comments on the IBM TrueNorth neural net chip, 2014, https://www.facebook.com/yann.lecun/posts/10152184295832143?_fb_noscript=1

CAS13: Andrew S. Cassidy , Paul Merolla , John V. Arthur , Steve K. Esser , Bryan Jackson , Rodrigo Alvarez-icaza , Pallab Datta , Jun Sawada , Theodore M. Wong , Vitaly Feldman , Arnon Amir , Daniel Ben-dayan Rubin , Emmett Mcquinn , William P. Risk , Dharmendra S. Modha, Cognitive computing building block: A versatile and efficient digitalneuron model for neurosynaptic cores, 2013, International Joint Conference on Neural Networks (IJCNN). IEEE

NAB12: Nabil Imam, Filipp Akopyan, Paul Merolla, John Arthur, Rajit Manohar, and Dharmendra Modha, A Digital Neurosynaptic Core Using Event-Driven QDI Circuits, 2012, Proceedings of the 18th IEEE International Symposium on Asynchronous Circuits and Systems

MER11: Paul Merolla, John Arthur, Filipp Akopyan, Nabil Imam, Rajit Manohar, Dharmendra

Modha, A Digital Neurosynaptic Core Using EmbeddedCrossbar Memory with 45pJ per Spike in 45nm, 2011, Proceedings of the IEEE Custom Integrated Circuits Conference (CICC)

SEO11: Jae-sun Seo and Bernard Brezzo and Yong Liu and Benjamin D. Parker and Steven K. Esser and Robert K. Montoye and Bipin Rajendran and José A. Tierno and Dharmendra S. Modha and Daniel J. Friedman, A 45nm cmos neuromorphic chip with a scalable architecture for learning in networks of spiking neurons, 2011, Custom Integrated Circuits Conference (CICC), IEEE

ART12: John Arthur, Paul Merolla, Filipp Akopyan, Rodrigo Alvarez, Andrew Cassidy, Shyamal Chandra, Steven Esser, Nabil Imam, William Risk, Daniel Rubin, Rajit Manohar and Dharmendra Modha, Building Block of a Programmable Neuromorphic Substrate: A Digital Neurosynaptic Core, 2012, International Joint Conference on Neural Networks (IJCNN)

HUN12: Phi-Hung Pham, Darko Jelaca, Clement Farabet, Berin Martini, Yann LeCun and Eugenio Culurciello, NeuFlow: Dataflow Vision Processing System-on-a-Chip, Proc., 2012

MAN00: Rajit Manohar, A Case for Asynchronous Computer Architecture, 2000, Proceedings of the ISCA Workshop on Complexity-Effective Design

ALA11: Alain J. Martin and Christopher D. Moore, CHP and CHPsim: A Language and Simulator for Fine-Grain Distributed Computation, 2011, Caltech Technical Report CS-TR-1-2011

ALA07: Alain J. Martin, Asynchronous VLSI Design: An Introduction, 2007, http://www.async.caltech.edu/general07.ppt