DARPA Report on Project: Superior AI Study, HR0011-16-0006

Task 4: Super-Turing Analog Hardware

Edward Rietman, U. Mass., Amherst

The following few pages describe work done associated with **Task 4**, Super-Turing analog hardware. The students involved in those subtasks wrote each main subsection.

One full-time grad student (Adam Kohan) has been working on simulating quasiperiodic oscillators for control of a robot (a simulated robot).

A second student, Harikrishnan Sreedharan Pillai in the EE department has been working with his professor, Sandip Kundu, in designing a chip to for neuromorphic computing and security applications. The chip is scheduled for manufacturing in September and should be ready for evaluation early in ’18.

A third student, Hia Ghosh, has previously reported on numerical simulations, now reports on initial work of comparing SPICE simulations with hardware. Her report, at this point is preliminary.

Project 1: Simulation of Robot(s) with Quasiperiodic Oscillators as Computational Engines

Adam Kohan

We developed a realistic and sophisticated genetic, or evolutionary, algorithm for online construction of organized complex networks possessing intrinsic topological organizations of the brain for capturing the representational power of neuron populations from the brain in artificial neural networks. The greater than Turing computational power of neural networks has long been shown [1][3][4]. However, we believe it is the architecture of the network that lends itself to the representational efficiency of that computational power (computational complexity) [2][5][6][7]. The architecture of the brain provides efficient computational power and is correlated with the complex interactions of functions represented by neuron populations in the brain that support higher-level intelligence [8][9][10][11]. For this reason, we devised an algorithm from relevant models in genetics, neuroscience, and computational neuroscience that elucidate the underlying processes of topological organizations in the brain. We matched the models with their corresponding time-line in the development of the brain across generations and in individual lifespans.

The time-line separates the algorithm into three distinct stages that overlap through time in the lifespan of the neural module or reservoir. The algorithm designs, constructs, and continuously shapes a reservoir of spiking neurons, wherein the brain is modeled as a

network graph with neural units as nodes linked by structural and functional connectivity as the edges. In the design stage, the neuron types, connectivity types, spatial organizational, and functional groups are determined, fine tuned, and revised. In the construction and shaping stages, the reservoir is grown by laying down scaffolding, and then by iterations of neuron proliferation, neuron migration, synapse formation, and synapse pruning during semi-supervised training of the reservoir. The construction stage focuses on learning the structures of the reservoirs. Conversely, and in complement, the shaping stage primarily directs functional learning and is defined by activity and resource dependent local learning rules mediated by reinforcement learning.

We implement a model of the spiking neuron using Leaky Integrate and Fire (LIF) dynamics with conductance based synapses whose decay is an exponential function. The dynamics of the membrane potential, V, is:

$$C_m \frac{dV}{dt} = -I_{leak} - I_{spike} - I_{syn} \qquad (1)$$

where C_m is the capacity of the membrane, and there are three currents. The equation of the leak current, I_{leak}, is:

$$I_{leak} = C_m \frac{(V(t) - V_{reset})}{t_m} \qquad (2)$$

where V_{rest}, is the resting membrane potential and t_m is the membrane time constant. The conductance based synaptic current is described by:

$$I_{syn} = \sum_{k=e,i,x} G_k(t)(V(t) - V_k) \qquad (3)$$

where V_k, is the reversal potential. The sum is over the set k of synapses: the excitatory synapses (e), the inhibitory synapses (i), and the external synapses (x). External synapses may be inputs into the network, noise, or an offset. The exponential decay of the synapse is:

$$dG_k(t)/dt = \frac{-G_k(t)}{t_k} \qquad (4)$$

where t_k is the decay time of the synaptic conductance. Finally, the spiking current, I_{spike} , is controlled programmatically such that when the membrane potential at time *t*, *V(t),* reaches the threshold V_{thresh}, the neuron spikes and the membrane potential is reset to V_{reset} for a refractory period of t_{ref}.

We chose the conductance based LIF model because it is the simplest possible biophysical representation of the neuron that captures many common features biological neurons share. In the sub-threshold regime, the membrane acts as a leaky capacitor whose potential decays to a resting voltage level. However, as soon as the membrane potential reaches the threshold, a critical value, the membrane potential spikes to a higher amplitude, propagates the corresponding short voltage pulse along the axon, and resets to a hyperpolarized voltage level for a refractory period.

The addition of the conductance based synaptic current models transmitter-activated ion channels involved in transmission across the synaptic cleft that results in an excitatory or inhibitory postsynaptic current. The current depends on the difference between the reverse potential V_k and the value of the membrane potential V(t). The reverse potential adjusts the direction of current across the cell membrane to maintain the equilibrium voltage of the synapse. The time dependent conductivity, $G_k(t)$, is a superposition of exponentials, of which an exponential decay is a simple representative choice. The result is a better model of the neuron's integrative properties as effected by several fold conductance increases from synaptic activity, in contrast to increased current. The neuron can be induced to fire through increasing the excitatory presynaptic rate, decreasing the inhibitory presynaptic rate, and unbalanced changes in the variance of the excitatory and inhibitory presynaptic rate.

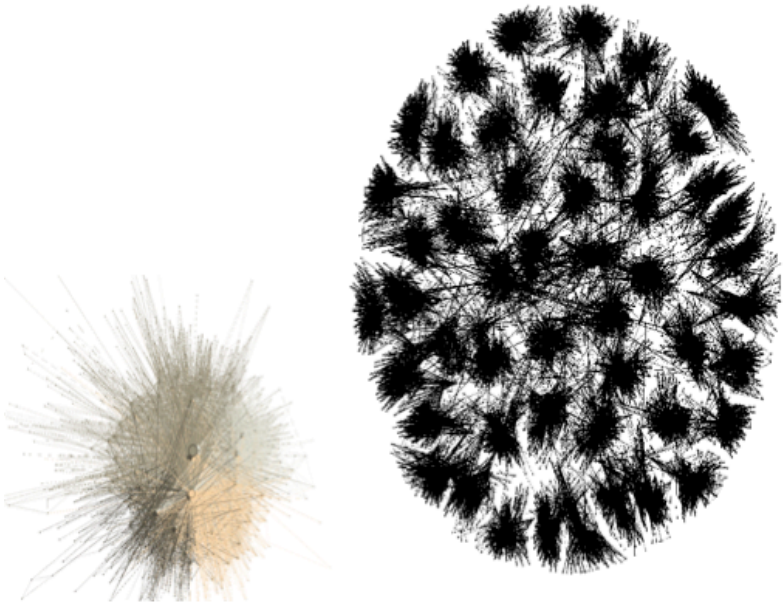


Figure: Example of "evolved" complex neural network, or artificial neocortex. The network on the left is one example of the many network modules comprising the overall structure. The entire system is built from models of spiking neurons.

At this point we have completed the software for "evolving" a complex cortical tissue-like artificial neural network. The Figure above shows an example of the overall network and one of the modules. Tests are currently underway to train it on NIST data, speech identification, and object recognition. To complement this we have designed a training regiment resembling environmental complexity in nature by providing data with variation in problem domains (vertical complexity) intermixed with periods focusing on data from a single problem domain (horizontal complexity). This is important, because the generalization and functionality of the overall system is dependent on the breadth and depth of training data processed early in the construction and shaping stages. That is to say, the neural system is then inherently multi-objective.

Results

The network design portion of the algorithm is in progress. The network design portion is responsible for the underlying organization of the network. Without an underlying organization, the networks are being generated using random fundamental units and settings. They have many of the properties of brain networks, but no underlying organization matching the problem domain.

The network construction portion of the algorithm is being tested using the datasets below.Accuracy varies widely depending on the random configuration of a generated network. This indicates that network organization is a considerable factor in learning. As such, accuracy should greatly improve once the network design portion of the algorithm is complete. In addition, the full training set has not yet been utilized for any of the datasets. Increasing the number of training samples has been shown to increase the accuracy by 5-10 percent.

The networks are treated as reservoirs with lasso regression used to assign output nodes to class labels. (Note that the network will not use lasso regression once the network design portion of the algorithm is ready for testing as the network will learn the response itself.) If the spiking network is viewed as filter, then the output nodes are indicators of class labels, either individually or as a group vote. Using lasso regression, the features (output nodes) for each class may be selected from the network and used to make a predication.

Alternatively, lasso regression was replaced with spike time dependent plasticity (STDP) in all the edges connected to output nodes. After training with STDP, the network is run on the training data with STDP disable, and output nodes for each class label are identified. The weight is updated on a postsynaptic spike event by the rule

$$\Delta w = l(x\ pre - xcon)(wmax - w)^{u}$$

where x_pre is the presynaptic trace, which increases by one for each presynaptic spike. The learning rate is l. The maximum weight is w_max. The dependence on the previous synaptic weight is determined by u. Finally, x_con controls the disconnection of presynaptic neurons that rarely fire.

The networks are tested both on the MNIST dataset and the Spoken Digit Dataset without modification. However, lasso regression is retrained for each dataset.

MNIST

With 10,000 to 30,000 out of 60,000 training samples and only a single iteration showing the training to the networks, the networks mean performance is between 35 to 50 percent accuracy on a test set of 10,000 samples.

Spoken Digit Dataset

With 100 out of 900 training samples and only a single iteration, the network mean performance is between 30 to 50 percent accuracy on a test set of 100 samples.

Future Work

During development, constructed networks are tested without modification on each of the testing datasets to ensure its capability to generalize. Our goal is for the networks to generalize to different problem domains without reconstruction. Finally, the networks will drive robots in 2D and then 3D environments.

The simulated environments will include multiple problem domains with which the networks must interact in a complex manner to survive. While there is no explicit goal in the simulated environments, the networks must manage multiple goals intertwined with their intrinsic needs. Intrinsic needs are initially driven by a lack of system resources.

References

1. Cybenko., G. (1989) "Approximations by superpositions of sigmoidal functions", Mathematics of Control, Signals, and Systems, 2 (4), 303-314
2. Kurt Hornik (1991) "Approximation Capabilities of Multilayer Feedforward Networks", Neural Networks, 4(2), 251257. doi:10.1016/0893-6080(91)90009-T
3. Siegelmann, H. T. and E. D. Sontag (1991). Turing computability with neural nets, Applied Mathematics Letters, 4, 77-80.
4. Siegelmann, H. and E. Sontag (1995). On the computational power of neural nets, J. Comp. Syst. Sci., 132 – 150
5. D. Srinivasan, A.C. Liew, C.S. Chang, A neural network short-term load forecaster, Electric Power Systems Research, 28 (1994), pp. 227234
6. X. Zhang, Time series analysis and prediction by neural networks, Optimization Methods and Software, 4 (1994), pp. 151170
7. D.L. Chester, Why two hidden layers are better than one? Proceedings of the International Joint Conference on Neural Networks, 1990, pp. 12651268.
8. Daniel P. Kennedy, Eric Courchesne, The intrinsic functional organization of the brain is altered in autism, NeuroImage, Volume 39, Issue 4, 15 February 2008, Pages 1877-1885
9. C.J. Stam, E.C.W. van Straaten, The organization of physiological brain networks, Clinical Neurophysiology, Volume 123, Issue 6, June 2012, Pages 1067-1087
10. M.P. van den Heuvel, C.J. Stam, M. Boersma, H.E. Hulshoff Pol, Small-world and scale-free organization of voxel-based resting-state functional connectivity in the human brain, NeuroImage, Volume 43, Issue 3, 15 November 2008, Pages 528-539
11. E. Bullmore and O. Sporns, "The economy of brain network organization", Nature Rev. Neurosci., vol. 13, no. 5, pp. 336-349, 2012

Project 2: Neuromorphic Chip for Pattern Recognition and Security Applications

Harikrishnan Sreedharan Pillai and Sandip Kundu

In this research project, we are designing an ASIC chip, which would function as a Quasi Periodic Oscillatory Machine for pattern recognition and as a security chip for authentication purposes.

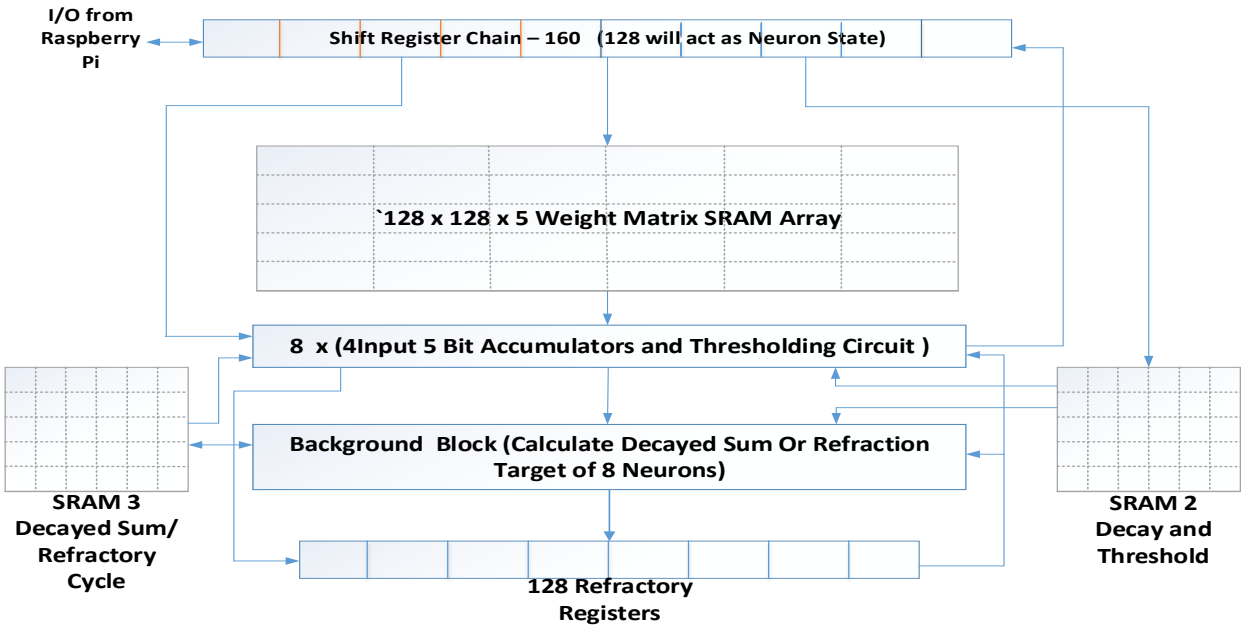
Introduction

The ASIC chip being designed in this project would function as a Quasi Periodic Oscillatory Machine(QPOM) for pattern recognition and as a security chip for authentication purposes.

The spiking neural network based Quasi Periodic Oscillatory Machine (Q-POM) would perform real-time computations on continuous streams of data. The information is stored as spatio-temporal patterns inside the recurrent neural network during training phase. This can be used like an associative memory for pattern recognition based applications. We will also be exploring possible computational properties from the oscillations of Q-POM.

Neuromorphic Chip

The neurons of our chip are based on Integrate and Fire model. It has the capability to have a memory of accumulated weighted sum in case the they did not fire. It can also decay that sum based on specified decay rate. The neurons have the capability to remain idle for specified refractory period after it fires.



Block Diagram of Neuromorphic Chip

The fully digital chip being designed has 128 neurons. Each neuron can have its own programmable threshold for firing and one among the 16 decay rates. The chip can have up to 16 iterations. The SRAM array will store the 128 x 128 weight matrix that defines the connections between neurons. The weights are stored in 2's complemented representation to support both excitatory and inhibitory connections and are of 5 bits. All neurons will have the same refractory period and it can be up to 16 iterative cycles.

The chip has a 160-register long shift register chain to which the inputs are driven from a Raspberry Pi. The weights are stored in the main SRAM array which is 128*(128 *5) cells big. Threshold (12 bits) and decay rate (4 bits) of each neuron will be stored in SRAM 2. If the neuron does not fire, the accumulated sum decayed by the specified rate, is stored in SRAM 3. Else, the iteration cycle up to which the neuron will be refracting is stored in SRAM3. The chip processes 8 neurons at a time in the 8 accumulators and thresholding circuitry. Background Block calculates the decayed sum based on decay rate simultaneously. It also calculates the target iterative cycle till which neuron would refract in case neuron does fire. This block will also have the comparator to check if current iterative cycle is the one till which the neuron refracts. The chip has another 128 registers to store the refractory information of each neuron.

Neuromorphic Chip as QPOM

While using the chip as Quasi Periodic Oscillatory Machine, we will first give a weight matrix based on training data. Then we drive in the decay rate and threshold of each neuron and configure the iteration count and the refractory period. 24 of the 128 neurons are kept aside as the input neurons. We drive in the 24 bit first input and iterate for specified number of cycles to get the first output. This is read out. For the music application simulated, this itself is the next input. In simulation, we generated 1 bar of music from a single note.

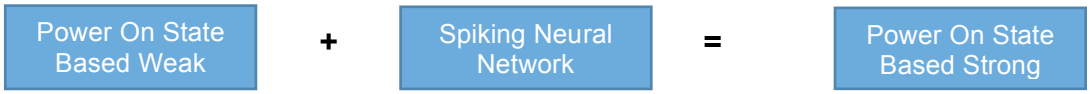
Neuromorphic Chip for Pattern Recognition (Hopfield Network)

SRAM3, background block and refractory registers are not used when chip is used in this mode. The iteration count is configured and trained weight matrix is driven in from Raspberry Pi. The thresholds for all 128 neurons are set. Then the input which is 128-bit messed up pattern is driven in. The chip would iterate for specified number of iterations. After the iterations are over, the output can be read out and is the correct pattern. We also plan to store more patterns in the matrix than the theoretical limit of Hopfield network and is planning to explore the oscillations for possible computations.

Neuromorphic Chip for Hardware Security

Physically Unclonable Functions (PUFs) have been successfully used for authenticating hardware modules at a low cost. The main advantage of using PUFs is that, the secret key used for authentication is a function of process variations of individual chips. Invasive attempts to decipher such a key would affect the characteristics of the chip and would effectively alter the key.

Weak PUF generally have only one response which is dependent on process variation and is used for key generation. Strong PUFs have many challenge response pairs (CRPs) and can be used for authentication. SRAM PUF is a weak PUF wherein the power on state of the SRAM provides unique signature to a chip. In this work, we combine the weak SRAM PUF with spiking neural networks to form a Strong PUF with multiple challenge response pairs (CRPs).



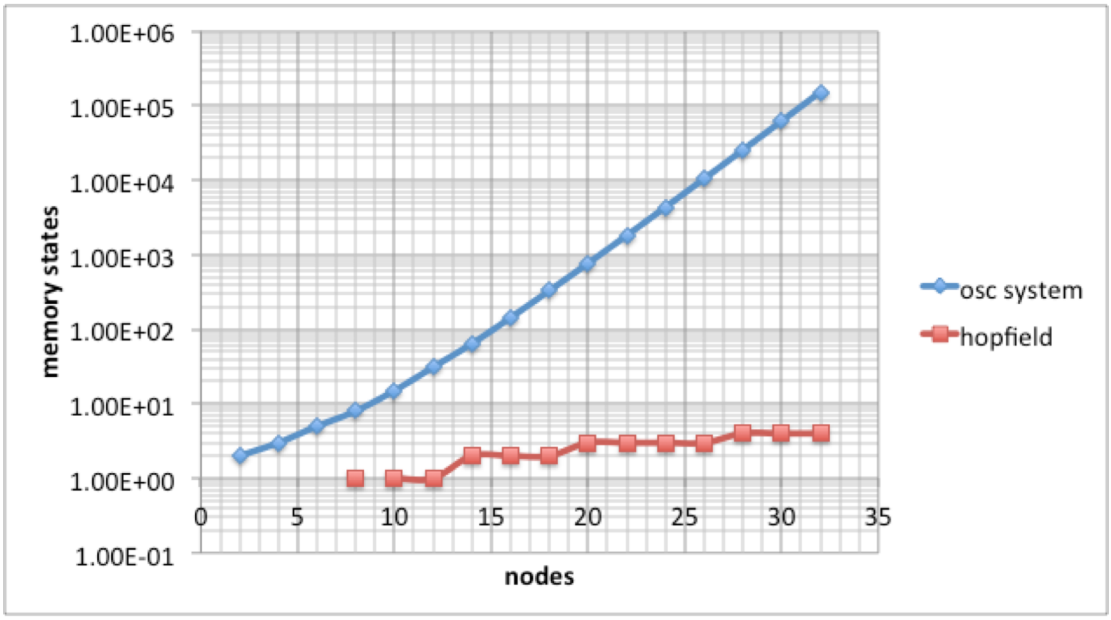
When the chip is used as Strong PUF, we do not drive in any weight matrix into the array. The power on State of the SRAM (fingerprint) will be the weight matrix. We drive in the decay rate and threshold of each neuron. Then we should configure the iteration count and the refractory period. Now we could give the challenge, and let the chip iterate for the specified iteration count and read out the response. We can have up to 2^{128} challenges. This PUF is expected to be machine learning resilient as it has 80kbits of states of weight matrix to be learned. Another hypothesis is that this PUF would not require denoising circuitry since the spiking neural network would be resilient to faults (unexpected 'power on states' of some of the bit cells).

Project Status

We have simulated the music application in software and could play the whole Daisy-Daisy song from a single music note and 4 weight matrixes. The Hopfield network based pattern recognition was also simulated in software wherein weight matrix was trained with 8 patterns each of 128 bits. The chip was able to successfully detect messed up patterns. Based on these simulations, the specifications for neuromorphic chip were finalized. The architecture of chip was then finalized after considering different design alternatives with minimum area and power in mind. The RTL design and behavioral simulation of the chip is completed except for the interface module. Design libraries for chip have been procured from MOSIS. The memory compilers were obtained from ARM. The chip is expected to be given to fab for fabrication in October.

Project 3: Hardware Construction of Neuromorphic Systems Based on Oscillators.

Hia Ghosh



Small ring circuits of inhibitory neurons exhibit an increasing number of stable oscillatory states. As the number of nodes in the ring circuit increases the number of stable states increases exponentially.

As shown in the above graph our goal is to demonstrate in simulations (SPICE or numerical) and in hardware the usefulness for massive content addressable memory based on oscillatory computing. This will have clear applicability in advanced AI. Quantitatively we will achieve results similar to the graph above, i.e. 3 to 5 orders of magnitude more memory states than a Hopfield network.

From prior experiments we have demonstrated that the number of states in a 14-node machine is capable of 64 stable states.

Theoretically, the number of stable states follows a 2-ary necklace function.

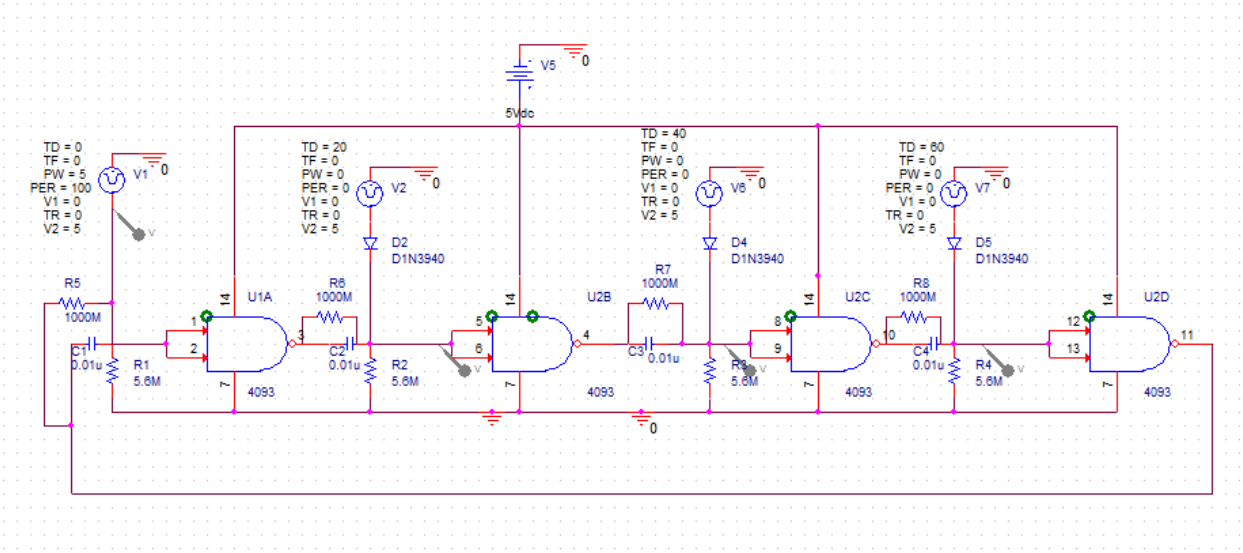
$$N(n,2) = \frac{1}{n}\sum_{i=1}^{v(n)} \phi(d_i)[F(d_i - 1) + F(d_i + 1)] \,, \qquad [1]$$

where d_i are the divisors of *n* with $d_1 = 1, \quad d_{v(n)} = n$; $v(n)$ is the number of divisors of *n*; $\phi(n)$ is the totient function, F(.) is the Fibonacci sequence (see mathworld.wolfram.com for details). The totient function is also called the Euler Totient function is given as the number of positive integers less than *n*, which are relatively prime to *n* (Rietman, et al. 2003).

We also know from previous work that connecting ring circuits together results in computation that can be interpreted as Boolean logic. Indeed much of the ground work for applications of these oscillatory circuits have been developed by Izhikevich and his colleagues.

PSPICE simulation results:

Previously we reported on preliminary numerical results for quasiperiodic oscillators. Now we extend that with PSPICE simulations and preliminary hardware experiments. Basically our findings are that SPICE simulations do not behave as the hardware experiments. Though these results are preliminary, it appears that the simulated circuits do not even oscillate. This non-oscillatory behavior in SPICE of simulations of theses circuits has been discussed with personnel in the EE department and we will be following up on this.



The figure above shows a schematic of the circuit for a four-node machine. There were two initial problems.

1) Initial conditions for capacitor: Since this is a feedback circuit, the simulator returned errors thinking the capacitor wasn't fully connected. Setting an initial voltage to the capacitors didn't help. As a result, we added resistors of large value in parallel to the capacitors to "complete" the circuit.

2) There is no way to manually toggle switches for input to the network in spice simulations. Once a switch is turned on, it remains on for the entirety of the simulation. As a result there could be no real time changes observed with triggering voltages in the circuit. To deal with this problem, we used pulse voltage sources instead of switches. We sent a short duration pulse as an input and then the pulse source would remain at zero voltage for 100 seconds. The parameters of these sources are shown as follows:

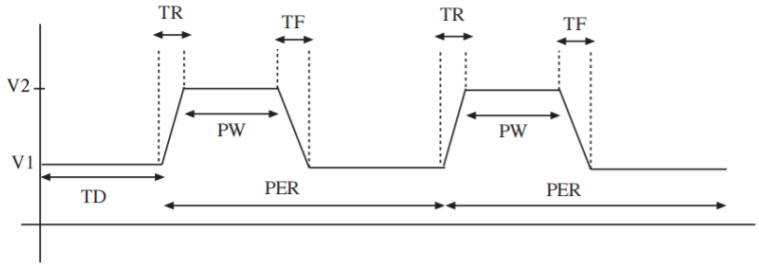
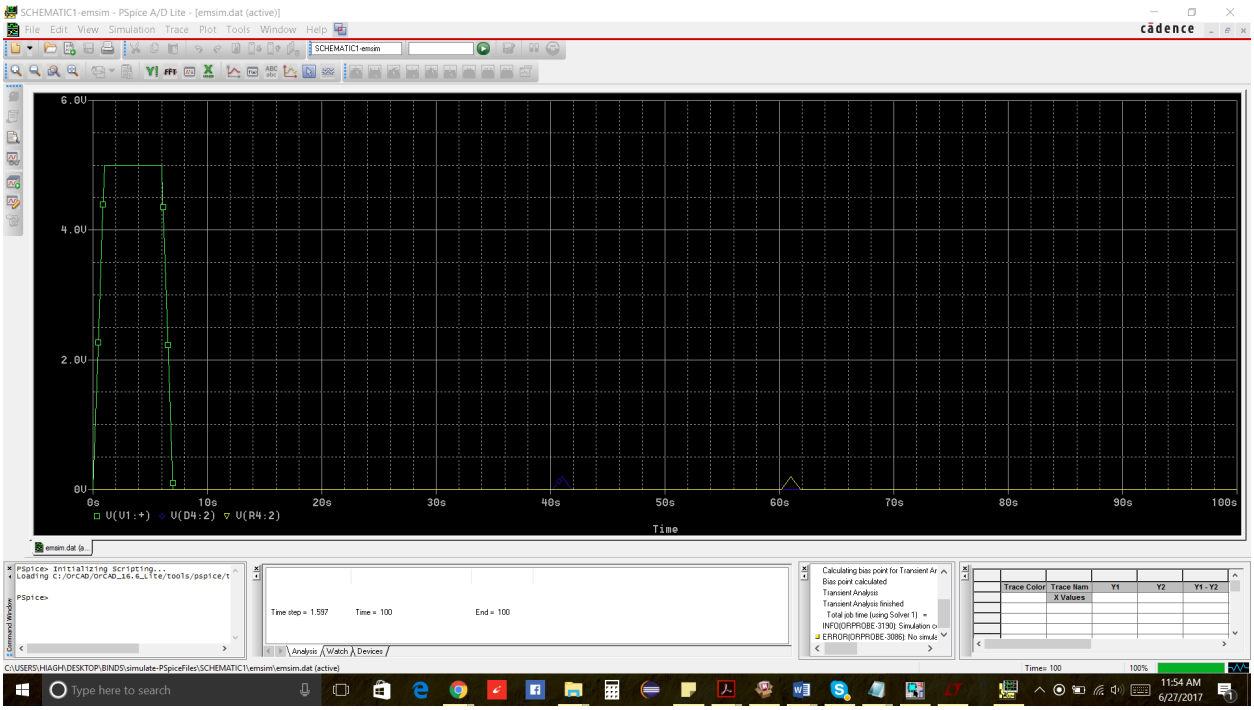


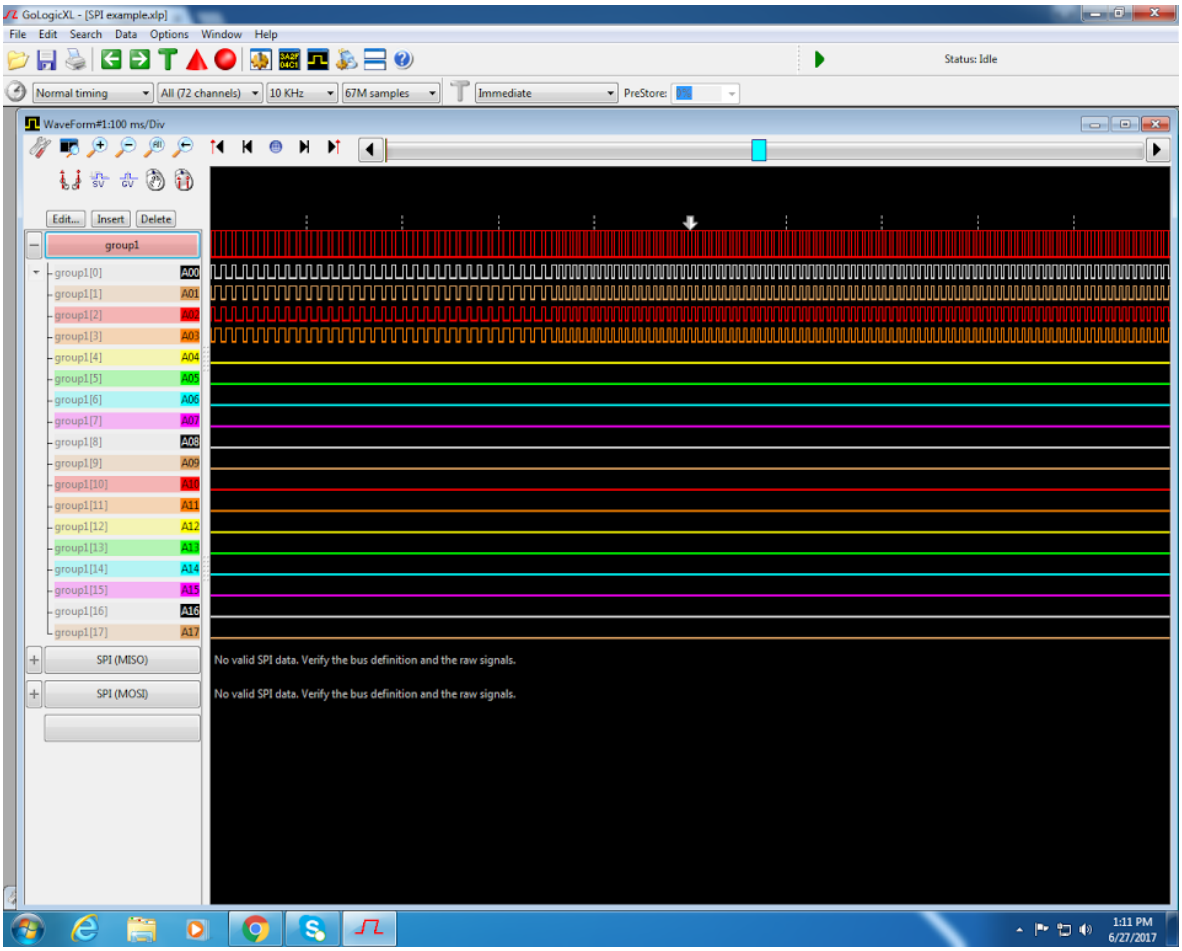
Figure 1. Parameters of VPULSE



In the above graph there is an initial pulse, the input signal to the neural oscillator, of duration 8 seconds. The system is allowed to then “free run”. As can be seen, no oscillations are observed. This is ongoing research.

Hardware results

We built a similar circuit, without the added 1GOhm shunt resistor on the capacitor and observed oscillations in the four-node machine. These are the results of triggering a 4-node circuit and observing the changes in oscillation and the stable states thereafter. In the screen dump from the digital analyzer the first trace is the group of all. The next four traces are for each of the individual neurons. The following 12 traces are ground. (The logic analyzer has probe sets of 16 channels. Unconnected channels are ground.)



Conclusion:

These preliminary results indicate that the dynamics in the hardware is richer than in the SPICE simulations. We expect to, in the future, observe real interference in pulse trains of connected quasiperiodic oscillators. One of our main goals, as pointed out above, is to demonstrate 3 to 5 orders of magnitude in memory storage. We also plan to demonstrate that with a programmable resistor array (similar to the chip we are having manufactured through MOSIS) spike time dependent plasticity or similar analog behavior. This, however, is a little way into the future yet.

Our next tasks are to finalize the PSPICE/hardware discrepancy and to outline a method to analyze these circuits using group theory.

References

Hopfield (1982), “Neural Networks and Physical Systems with Emergent Collective Computational Abilities,” *Proc. Natl. Acad. Sci. USA*, 79, 2554-2558.

Hopfield (1984), “Neurons with Graded Response have Collective Computational Properties like those of two-state Neurons,” *Proc. Natl. Acad. Sci. USA*, 81, 3088-3092.

Hoppensteadt and Izhikevich (1999), “Oscillatory Neurocomputers with Dynamic Connectivity,” *Physical Rev. Lett*. 82, 2983-2986.

Hoppensteadt and Izhikevich (2000), “Pattern Recognition via Synchronization in Phase-Locked Loop Neural Networks,” *IEEE Trans. on Neural Networks*, 11, 734-738

Rietman, et al. (2003), “Analog Computation With Rings of Quasiperiodic Oscillators: The Microdynamics of Cognition in Living Machines,” *Robotics and Autonomous Systems*, 45, 249-263.