

# *Design of superconducting optoelectronic networks for neuromorphic computing*

Sonia Buckley  
Adam N. McCaughan  
Jeff Chiles  
Richard P. Mirin  
Sae Woo Nam  
Jeffrey M. Shainline  
Grant Bruer  
James S. Plank  
Catherine D. Schuman

November, 2018

*IEEE International Conference on Rebooting Computing (ICRC 2018)*

**<http://icrc.ieee.org/>**

An online home for this paper may be found at: <http://neuromorphic.eecs.utk.edu>

---

## Citation – Plain Text:

```
author      S. Buckley and A. N. McCaughan and J. Chiles and R. P. Mirin and
            S. W. Nam and J. M. Shainline and G. Bruer and J. S. Plank and C. D. Schuman
title       Design of superconducting optoelectronic networks for neuromorphic computing
booktitle   IEEE International Conference on Rebooting Computing
address     Tysons, VA
month       November
year        2018
pages       36--42
```

## Bibtex:

```
@ @INPROCEEDINGS{bmc:18:dso,
  author = "S. Buckley and A. N. McCaughan and J. Chiles and R. P. Mirin and
           S. W. Nam and J. M. Shainline and G. Bruer and J. S. Plank
           and C. D. Schuman",
  title = "Design of superconducting optoelectronic networks for neuromorphic computing ",
  booktitle = "IEEE International Conference on Rebooting Computing",
  address = "Tysons, VA",
  month = "November",
  year = "2018",
  pages = "36--42"
}
```

# Design of superconducting optoelectronic networks for neuromorphic computing

Sonia Buckley, Adam N. McCaughan,  
Jeff Chiles, Richard P. Mirin,  
Sae Woo Nam, Jeffrey M. Shainline  
*National Institute of Standards and Technology*  
Boulder, CO  
sonia.buckley@nist.gov

Grant Bruer, James S. Plank  
*Department of EECS*  
*University of Tennessee*  
Knoxville, TN  
jplank@utk.edu

Catherine D. Schuman  
*Computational Data Analytics*  
*Oak Ridge National Laboratory*  
Oak Ridge, TN  
schumancd@ornl.gov

**Abstract**—We have previously proposed a novel hardware platform (SOEN) for neuromorphic computing based on superconducting optoelectronics that presents many of the features necessary for information processing in the brain. Here we discuss the design and training of networks of neurons and synapses based on this technology. We present circuit models for the simplest neurons and synapses that we can use to build networks. We discuss the further abstracted integrate and fire model that we use for evolutionary optimization of small networks of these neurons. We show that we can use the TENNLab evolutionary optimization programming framework to design small networks for logic, control and classification tasks. We plan to use the results as feedback to inform our neuron design.

## I. INTRODUCTION

The human brain is capable of complex pattern recognition and learning tasks with a fraction of the power consumption of a traditional computer. Neuromorphic computing aims to build hardware that emulates the brain to provide some of this advantage [1]. Deep learning techniques inspired by the brain have already proven to be very effective for many pattern recognition and learning tasks [2]. Many neuromorphic computers have architectures designed specifically to implement deep learning algorithms. In addition to these architecture changes, neuromorphic computers often use spikes to carry information. These “spiking neural networks” (SNNs) have very high computational power [3], [4], [5] and their hardware implementations have very low power consumption.

Spiking neuromorphic hardware may also require new physics and materials beyond traditional CMOS. We have previously proposed [6], [7] a hardware platform based on superconducting optoelectronic neurons (SOENs) that can achieve physical fanouts of one to one thousand [8], [9]. The platform utilizes the properties of superconductors to achieve energy efficiency similar to the human brain in terms of computations per second per watt, with operation energies as low as tens of attojoules per synapse event [7]. However, the proposed neurons are challenging to fabricate, and most approaches for training even on simple benchmark problems such as MNIST handwritten digit classification [10] require hundreds of neurons and tens of thousands of synapses.

Meanwhile, it is usually not easy to take advantage of the full computational power of a spiking neuromorphic hardware platform when using these training approaches. In particular, training such networks effectively to perform useful tasks presents a major challenge. Here we propose using evolutionary optimization (EO) to take advantage of all of the parameters that we can control in this hardware platform to design smaller networks that are realistic to fabricate and test in the near term. We use the programming framework implemented by TENNLab [11] to simulate these circuits in a neuromorphic computation environment and provide guidance towards early SOEN design.

In this paper we propose a simplified circuit model for a SOEN neuron which we first simulate in LTSpice to obtain their electrical characteristics. We then implement a high level model of the simplified SOEN neuron/synapse circuits in the TENNLab evolutionary optimization programming framework [11] for neural networks. We use existing logic, control and classification applications already implemented in this framework to explore SOEN responses and performance while varying their design parameters. This technique provides feedback for the important parameters of these circuits and guides us towards early implementations of SOEN networks. Our goals in this study are to use EO to evaluate (1) if we can solve logic, control and classification tasks using this hardware platform (2) which of these circuit variations performs better and at which tasks, and (3) to investigate how the input and output encoding affects the circuit performance.

## II. HARDWARE PLATFORM

The spikes in this hardware platform consist of 20 ns photonic pulses, generated by a semiconductor LED [12]. The spikes are distributed by a network of photonic waveguides (shown in schematic in Fig. 1(a)) fabricated in deposited dielectrics [8], [13]. The photonic waveguides allow this hardware platform to reach very high physical fanout. A schematic of an individual neuron is shown in Fig. 1 (b). Pulses are received by downstream neurons at synapses (yellow). Each synapse converts the pulses from the optical to the electrical domain using a superconducting nanowire [13], with the

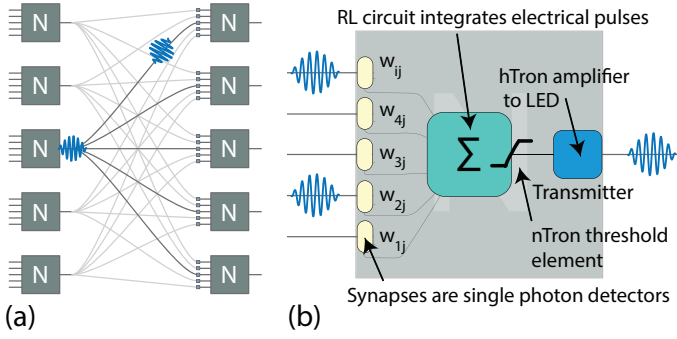


Fig. 1. (a) A network of neurons connected by photonic waveguides. (b) Schematic of an individual neuron.

amplitude and duration of the electrical pulse determined by circuit properties described in the next section.

Varying levels of complexity are possible for SOEN design. In its simplest form, the amplitude of the electrical pulse from the detector synapse can be weighted ( $w_{ij}$ ) and integrated with electrical pulses from other synapses on a thresholding unit, using a superconducting circuit element, the nTron [14] (Fig 1 (b)). Once the threshold value has been exceeded, the current in the integration loop triggers the transmitter circuit and discharges a photonic pulse. The transmitter circuit consists of an amplifier (hTron) and an LED coupled to an output waveguide (axon). This is the SOEN circuit design we will discuss in the remainder of this paper.

In more advanced versions of SOENs, the electrical pulse from the detectors is weighted by a high-efficiency superconducting Josephson junction (JJ) circuit [15], [16], which produces a number of single flux quantum (SFQ) pulses [17], [18], [19]. The number of SFQ pulses generated depends on the synaptic weight, which is in turn controlled by an input current. Further biological realism and processing capability are possible [15]. With as little as one photon from the upstream and downstream neuron, superconducting circuitry can adjust the current controlling the synaptic weight, thus implementing the biologically inspired learning mechanism spike-timing dependent plasticity (STDP). Similar circuitry can implement dendritic processing and metaplasticity. SFQ pulses from all of the synapses on a particular neuron can be integrated in a series of integration loops and compared to an adjustable threshold value.

At present, we have experimentally demonstrated the semiconductor LEDs, multiplanar waveguide routing, waveguide integrated superconducting detectors, and the superconducting electronics necessary for the thresholding and amplification elements. We have yet to demonstrate the synaptic Josephson junction circuits, although such technology is routinely implemented for voltage standards [20] and other superconducting circuits [21], [22], [23], [24]. For the earliest demonstrations, we seek to build a neuromorphic computer with simpler elements and direct electronic weight control in place of the more complicated JJ circuitry, albeit at lower energy efficiencies and with loss of some functionality. The advantage of these circuits

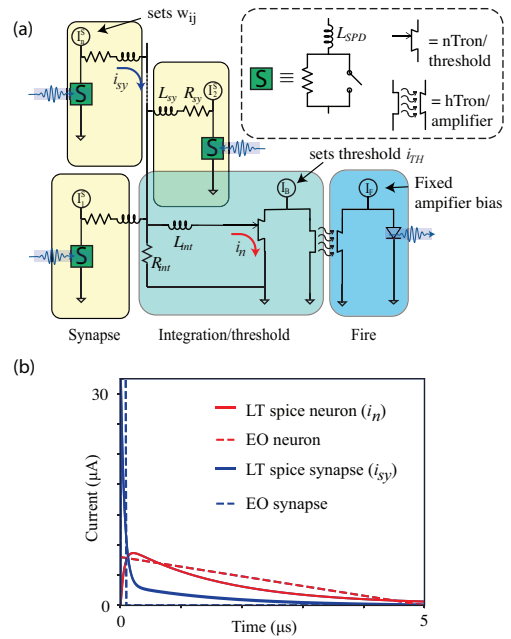


Fig. 2. (a) SOENs circuit with neuronal integration loop (SOEN1). The neuron fires a photonic pulse when  $i_n > i_{TH}$ . The equivalent circuit for the SNSPD is shown in the dashed box on the top right, as well as the nTron and hTron circuit symbols. (b) The current in the neuron integration loop/nTron ( $i_n$ ) with two synapses firing (red) and the current ejected ( $i_{sy}$ ) from the SNSPD (blue) when a synapse fires, simulated in LTspice (solid line) and in the EO programming framework (dashed line).

is they provide us with a simple testbed with which we can demonstrate and develop this technology. Therefore the circuit described in this paper is a simplified version of the full circuit described in ref. [25]. The simplified circuit will enable us to prototype neurons and neural networks more rapidly.

### III. CIRCUITS

We start by considering two simplified SOEN variations, SOEN1 and SOEN2. SOEN1 is described in Fig. 2 (a). The neuron circuit can be divided into a number of synaptic circuits, a single neuronal integration loop and a transmitter circuit. Photonic spikes from upstream SOENs are received at the synapses of downstream neurons (indicated by an “S” in Fig. 2 (a)). Each synapse consists of a superconducting nanowire single photon detector (SNSPD) capable of detecting single photons. Single photon detection efficiency of  $>90\%$  has been demonstrated with these detectors in the near-IR [26]. Synapses can be connected in parallel on a single neuron. When a photon is detected, the SNSPD ejects its current ( $i_{sy}$ ), converting the photonic pulse to an electrical pulse. The equivalent circuit model for the SNSPD, used in the LTspice simulation, is shown in the dashed box in Fig. 2 (a) and consists of an inductor ( $L_{SPD}$ ) in series with a switch in parallel with a  $1\text{ k}\Omega$  resistor. We model the detection of a photon on the SNSPD as opening the switch for 5 ns (with a rise time of 1 ns). The current pulse amplitude is set by the SNSPD bias current, and the actual pulse duration is given by the  $L/R$  time constant of the synapse, which depends on the

designed series resistance of the synapse,  $R_{sy}$ , the designed series inductance of the synapse,  $L_{sy}$ , the inductance of the detector,  $L_{SPD}$ , and the integration loop resistance,  $R_{int}$ . We refer to this detection of a photon and ejection of current as a synaptic firing event. Once a synapse has fired, it cannot fire again until the nanowire has returned to the superconducting state. In the present model, this time is 100 ns.

The current from the firing synapse is diverted to an accumulation element, which we refer to as the neuronal integration loop. When the synapse fires, it is stored as current ( $i_n$ ) in the neuronal integration loop for a time (the neuron leak rate) given by the time constant of this loop ( $L_{int}/R_{int}$ ). The amount of current added to this loop, and thus the synaptic weight, is varied by changing the bias current on the SNSPD in the synapse (labeled as  $I_i^S$  in Figs. 2 and 3). The synaptic pulse shown in Fig. 2 (b) is for a bias current of  $15 \mu\text{A}$ , as would be present in an SNSPD made of MoSi or NbN. The current in the neuronal integration loop when two such synapses are firing at once is shown in red in the same figure. The neuronal integration loop has a leak rate of 200 kHz, leading to an integration time on the order of  $5 \mu\text{s}$ , or 50 times longer than the SNSPD refractory period.

The thresholding element in the neuronal integration loop is an nTron superconducting current amplifier [14], which in this case is operated as a switch. The nTron will divert current to the transmitter circuit when the gate threshold,  $i_{th}$ , is exceeded, while  $i_{th}$  depends on the nTron bias current ( $I_{Bnt}$ ), and can be varied dynamically by changing  $I_{Bnt}$ . The nTron will reset once it has been triggered, as the  $L/R$  time constant in the loop suddenly decreases due to the increase in  $R$  as the nTron gate turns from superconducting to normal, causing the current to leak out over a thousand times faster. The refractory period of the neuron is set by the reset time of the nTron, which is controlled by the new  $L/R$  time constant.

The transmitter circuit is triggered by a thresholding event to generate the photonic pulse referred to as a neuronal firing event. The transmitter circuit consists of a hTron amplifier, which transduces the small voltage generated by the nTron to the larger voltage necessary to turn on the LED. It operates by heating up a superconducting element in parallel with the LED, thus suddenly increasing the parallel resistance and diverting the current through the LED, as shown in Fig. 2 (a) between the threshold and fire boxes.

While the goal of this study is to determine if we can use evolutionary optimization to design neuromorphic systems using SOEN technology, we also consider whether evolutionary optimization can inform us on the minimum requirements for the basic SOEN design. In the SOEN circuits considered here, the neuronal integration loop serves as a short-term memory for spikes, but also reduces the amount of current passing through the nTron per synapse event. For reasonable values of the nTron threshold and synapse bias currents, a single synapse cannot trigger the neuron. Therefore we also consider the circuit shown in Fig. 3. In this circuit, the neuronal integration loop is omitted; this SOEN design has a much faster leak rate, defined by the  $L/R$  time constant of the SNSPDs. This leads

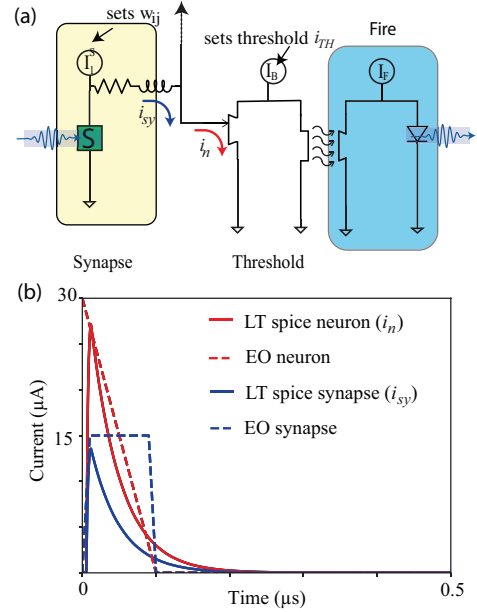


Fig. 3. (a) SOENs circuit without neuronal integration loop (SOEN2). The neuron will fire when  $i_n > i_{TH}$ . (b) Simulation of the current in nTron ( $i_n$ ) with two synapses firing (red), the current ejected ( $i_{sy}$ ) from the SNSPD (blue) when a synapse fires in LTspice and modeled in the EO programming framework.

to an integration time on the same order as the refractory period of the neuron and the synapse. The synapse pulse and the current through the nTron (thresholding element) when two synapses are firing at once for the no-integration circuit is shown in Fig. 3 (b). We reiterate that these are simple circuits that can be implemented in the superconducting optoelectronic technology in the near-term. Different combinations of the same circuit elements are also possible. We refer to all of these as SOEN circuits, with different parameter sets. More complex circuits may be developed as needed if a particular functionality is found to be critical, as we discuss in Section V. It is therefore one of the goals of this paper to determine the most useful circuit parameter sets for particular applications.

#### IV. EVOLUTIONARY OPTIMIZATION

It is important to verify that we can use SOEN circuits in networks to perform various applications. Early in the development of a new technology it is difficult to design and fabricate systems of large numbers of neurons and synapses, as the infrastructure for emergent technologies is less mature. For example, even the relatively simple benchmark classification of MNIST handwritten digits is typically implemented in algorithms that use one input neuron per pixel (784 inputs) and ten output neurons (one per digit classification), in addition to order ten to one hundred hidden neurons. Moreover, implementations that simply map, for example, a convolutional neural network algorithm to a spiking hardware platform, do not take advantage of the dynamic aspect of SNNs, and the variety of ways that SNNs can encode information. It has been shown that an SNN can classify MNIST using a smaller

TABLE I  
SIMPLIFIED MODEL FOR EO

EO parameters	SOEN1 (Fig. 2)	SOEN2 (Fig. 3)	SOEN3	SOEN4	SOEN5	SOEN6
neuron threshold	5-15 $\mu\text{A}$	5-15 $\mu\text{A}$	5-15 $\mu\text{A}$	5-15 $\mu\text{A}$	5-15 $\mu\text{A}$	5-15 $\mu\text{A}$
synapse weight	-3-3 $\mu\text{A}$	-15-15 $\mu\text{A}$	-3-3 $\mu\text{A}$	-15-15 $\mu\text{A}$	-15-15 $\mu\text{A}$	-15-15 $\mu\text{A}$
post-neuron delay	0	0	0	0	0-10 $\mu\text{s}$	0-10 $\mu\text{s}$
Set parameters						
synapse delay	50 ns	"	"	"	"	"
neuron leak time	5 $\mu\text{s}$	0.1 $\mu\text{s}$	0.1 $\mu\text{s}$	5 $\mu\text{s}$	0.1 $\mu\text{s}$	5 $\mu\text{s}$
neuron refractory period	100 ns	"	"	"	"	"
synapse refractory period	100 ns	"	"	"	"	"

number of inputs [27], [28] by partially encoding the image in time. In Ref. [27], design of these relatively small networks was done by evolutionary optimization.

Motivated by the preceding arguments, in this paper we choose EO to design simple neural networks in the superconducting opto-electronic hardware platform. This allows us to take advantage of the complexity of dynamic spiking networks and to generate relatively small networks to solve particular tasks. EO has the additional advantages that it can use any of the parameters of this hardware, and will train the architecture (i.e. size and connectivity) of the network in addition to the parameters.

Table 1 shows the parameter sets we have tested for the SOENs model implemented in the TENNLab EO programming framework [11]. EO parameters are subject to mutation during training, while set parameters are defined in the model. Values are based on the LTspice simulations and realistic physical parameters, and are implemented as a first step towards a comprehensive model to design networks. SOEN1 and SOEN2 are also shown as the dashed lines in Figs. 2 (b) and 3 (b), and correspond most directly to the circuits described in the previous sections. However, there is flexibility in the design of these circuits, and therefore we also investigated other parameter combinations. For example, SOEN1 has a maximum synapse weight that is lower than the minimum neuron threshold, and therefore a single synapse cannot trigger a single neuron. Meanwhile, SOEN2 has a synapse weight range that will allow a single synapse to trigger a neuron, but with 50 times shorter integration time. Therefore, we have also implemented SOEN3 and SOEN4, to investigate whether differences between SOEN1 and SOEN2 are due to the synapse weight range or the integration time. We have not yet designed circuits with the SOEN3 and SOEN4 parameter sets, but rather we are using these parameter sets to investigate the importance of specific circuit parameters. In addition, if more parameters (e.g. synapse delay) are needed to solve tasks, more complex circuits may be designed for these tasks. Delay circuits and fixed electrical delays are possible to design using superconducting circuit elements. We therefore added post-neuron delays to the SOEN2 and SOEN4 parameter sets (generating SOEN5 and SOEN6), to investigate the effect of this additional parameter on the ability to evolve networks to perform various tasks.

To begin, we solved one of the TENNLab benchmark tasks, W-bit XOR. The task has been described in detail in Ref. [11]. We use two inputs per bit. A one bit XOR problem therefore

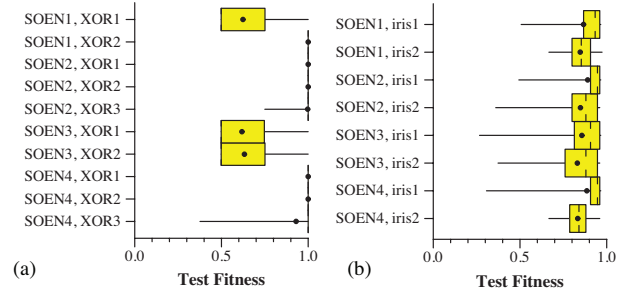


Fig. 4. (a) Statistics of testing fitness of evolved networks at XOR benchmark tests. XOR1 = (ppb = 1, W = 1), XOR2 = (ppb = 10, W = 1), XOR3 = (ppb = 10, W = 2), where W = number of bits, ppb = pulses per bin (see text). (b) Statistics for iris classification task (Iris1, ppb = 1, Iris2, ppb = 10).

TABLE II  
XOR RESULTS (MAX, MEDIAN)

Model	XOR1	XOR2	XOR3
SOEN1	1.0, 0.5	1.0, 1.0	-
SOEN2	1.0, 1.0	1.0, 1.0	1.0, 1.0
SOEN3	1.0, 0.5	1.0, 0.5	-
SOEN4	1.0, 1.0	1.0, 1.0	1.0, 1.0

requires four inputs. An “input” is a neuron to which current can be directly applied by the user or application, and serves as an initial transition between electrical inputs and photonic pulses. Depending on the magnitude of the electrical input a photonic pulse may or may not be produced by the input neuron, the magnitude of the photonic pulses are constant. Here we encode the input values in two different ways. In the first “no-rate-encoding” method a single pulse in one input neuron represents a one, a single pulse in the other input neuron represents a zero, i.e. pulse per bin is one (ppb = 1). In the second “rate-encoding” input method, 10 pulses are applied to the associated input neuron in the network (ppb = 10). In a digital problem such as XOR, “rate-encoding” is simply equivalent to adding redundancy to the inputs. For the rate-encoded inputs, there is an interval of 3 time steps between pulses, where a time step in the application translates to 1  $\mu\text{s}$  for the device. In both cases, the applied input current is high enough that the input neuron will always fire a pulse. We run this task for one bit XOR, with no-rate-encoding (XOR1) and rate-encoding (XOR2) using the first four SOEN EO models described in Figs. 2, 3 and Table 1. Once the bits are pulsed into the neuromorphic device, the device runs for a set interval of time, during which output pulses are counted. The output that pulses more decides the answer. We also ran 2 bit XOR with rate-encoding (XOR3) for SOEN2 and SOEN4, which were overall the highest performing parameter sets.

Networks are initialized with the correct number of input neurons, output neurons and with one synaptic connection per neuron. In each epoch of an EO run, we run fitness calculations on a population of 1000 networks solving 200 XOR problems. The networks in the next epoch are generated based on the fitness results [29], and an algorithm that varies the topology

TABLE III  
IRIS RESULTS (MAX, MEDIAN)

Model	Iris1	Iris2
SOEN1	0.96, 0.93	0.97, 0.85
SOEN2	0.96, 0.90	0.96, 0.88
SOEN3	0.96, 0.92	0.96, 0.88
SOEN4	0.96, 0.95	0.93, 0.79

of the networks as well as the “EO parameters” in Table I. Each EO run proceeds in as many epochs as it can complete in 5 hours (or until a fitness of 1 is reached). For each SOEN (model) and XOR (application) parameter set we perform 100 EO runs, generating 100 networks. These networks are then evaluated on five sets of 10,000 testing problems. The plots in Fig. 4 (a) show the testing fitness value statistics for the generated networks. The edges of the yellow box are the first and third quartiles of the data, the horizontal line is the median, the dot is the average, and the whisker extends to the max and min values of the data. Table II also summarizes the results for the best and median network fitness for each of the problems for both models. We find that while all of the models can generate networks with a fitness of 1, they do not all do so with equal likelihood, as can be seen from the fact that SOEN1 and SOEN3 have lower median fitnesses. In both SOEN1 and SOEN3, a single synapse cannot add enough current to the neuronal integration loop to trigger a neuron, due to the available nTron threshold range. It also appears that the added redundancy of the rate-encoding helps, in particular for the SOEN1 parameter set, which has a long integration time and performs much better for XOR2 (rate encoding) than XOR1.

We next investigate the performance of the circuits at the iris classification task from the UCI Machine Learning Repository [30], using the methods described in detail in [31]. In this task irises are classified as one of three types based on four physical measurements, the database consists of the measurement data labeled with the class. We use the same circuit and input variants as in the XOR problem, with four network inputs for the four iris characteristics. Iris1 therefore has  $\text{ppb} = 1$  and iris2  $\text{ppb} = 10$ . Each EO run is allowed to run for five hours and we report fitness in Fig. 4 (b). In this case we find very little variation, although the rate encoding tends to yield lower median network fitnesses. Overall, the XOR and iris problems seem to be too easy to solve, yielding little difference in the fitness of the evolved networks, although there are differences in the time and number of epochs it takes to arrive at these networks.

We next investigate the performance of SOEN neurons for a benchmark control task, balancing a pole on a cart. In this task, the network must keep a pole balanced on a cart as described in detail in Ref. [31]. The network can apply a fixed force to push the cart to the left or to the right to keep the pole balanced. The network receives values for the position and velocity of the cart and the angular position and velocity of the pole (four variables total). Each of these variables is binned

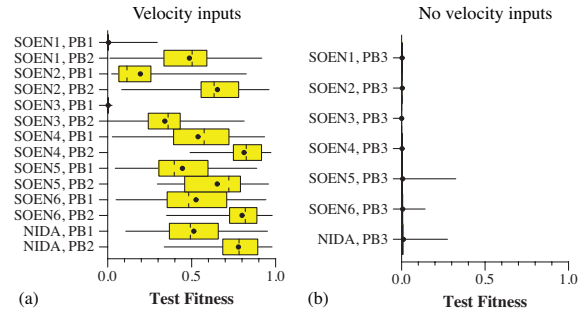


Fig. 5. (a) Statistics of testing fitness of evolved networks at a polebalance control task using position and velocity inputs. PB1 = ( $\text{ppb} = 1$ ), PB2 = ( $\text{ppb} = 10$ ) (b) Statistics of testing fitness at polebalance control task with only position inputs ( $\text{ppb} = 10$ ). NIDA = TENNLab software model, SOEN parameter sets described in Table I.

into two categories, left and right. Thus, polebalance networks have a total of 8 input neurons. The current state of game is input to the network with activity on four of the input neurons, either with a single pulse (PB1) or ten pulses separated by 3 time steps (PB2). In PB1, the amplitude of the current pulse is allowed to vary, and this represents the value of the particular variable, for example cart position. The maximum amplitude will always cause the input neuron to fire, while intermediate amplitudes may or may not cause the input neuron to fire, depending on the evolved neuron threshold. In PB2, pulses are digital and the number of pulses represents the value. For example, if the cart position  $x$  can vary between  $-3$  and  $3$ , a value of  $x = 1.2$  will cause 4 pulses to fire in the second bin, while a value of  $-1.2$  will cause 4 pulses to fire in the first bin. In this case, an input current pulse will always cause an input neuron to fire. The network has two outputs, which effectively vote on which direction to apply the force.

We ran the SOEN model for all six parameter sets and the TENNLab NIDA model [11] at the PB1 and PB2 tasks. The results of the tests are shown in Fig. 5 (a) and Table IV. All of the models performed worse for PB1. However, SOEN1 and SOEN3 had the most difficulty solving this task, again showing the importance of a single synapse triggering a neuron. This may prove possible to overcome by the selection of suitable input parameters, EO mutation weights or initial conditions. SOEN4, which has a long integration time and a high maximum synapse weight, performed the best, with SOEN2, which has a shorter integration time and the same maximum synapse weight performing second best. This shows that the addition of an integration time is beneficial in this task. Comparing SOEN2/SOEN5 or SOEN4/SOEN6 pairs, which are the same except for the addition of a post-neuron delay, we find that there is a small improvement with the addition of this parameter.

In principle, we should be able to develop networks that do not require velocity as an input, but rather use only the position of the cart and pole to estimate the velocity within the network. Such networks would “remember” previous positions and infer the velocity information. We test the ability of the

TABLE IV  
POLE BALANCER RESULTS (MAX, MEDIAN)

Model	PB1	PB2	PB3
SOEN1	0.30, 0.002	0.92, 0.5	0.004, 0.009
SOEN2	0.82, 0.11	0.96, 0.63	0.004, 0.004
SOEN3	0.03, 0.002	0.81, 0.36	0.002, 0.001
SOEN4	0.93, 0.57	0.97, 0.82	0.004, 0.003
SOEN5	0.89, 40	0.96, 0.72	0.32, 0.003
SOEN6	0.94, 0.48	0.98, 0.82	0.14, 0.003
NIDA	0.95, 0.49	0.98, 0.78	0.27, 0.004

SOENs models to perform this task with only the positions as inputs. In this case, there are only two input bins for cart position and two for pole angular position (for a total of four input neurons), and we always use rate encoding with up to ten pulses per bin. We first use the TENNLab NIDA model [11] and show that it is capable of solving the problem. We find that NIDA has more difficulty than in the case with velocity as an input, evolving many low fitness networks (the average, median and first and third quartiles are all close to zero), but is still able to produce a few higher fitness networks. We next test it for the SOEN parameter sets in the previous test. Even SOEN2 and SOEN4 were not able to balance the pole without the velocity inputs (using the same time limit for the each EO run as in the previous tests). The main difference between the SOEN model with parameter sets 1-4 and NIDA as implemented here, is that NIDA has no leak (so it holds charge forever) and second, the synapses have a variable and evolvable delay. This indicates that we may need to implement such a variable synapse delay, or some other surrogate for it, in these circuits in order to perform certain tasks. SOEN5 and SOEN6 have the same parameter sets as SOEN2 and SOEN4, but with the addition of an evolvable post-neuron delay. We find that both models perform similarly to NIDA, evolving a few networks with reasonable fitnesses. This suggests that while variable delays may not be important for some tasks (e.g. the polebalance with velocity inputs) it may be worthwhile for other tasks (e.g. the polebalance without velocity inputs). Therefore it is likely worthwhile to design some variable delay circuits, but it may be only necessary to implement them after each neuron, rather than an element per synapse.

## V. DISCUSSION AND FUTURE WORK

We were able to design networks for logic, control and classification using the SOEN model in the TENNLab programming framework. We investigated various parameter sets for the SOEN model to help inform the designs of our early circuits. We found that a larger synapse weight range, such that it is possible for a single synapse to trigger a single neuron, works better for every application we tested, and was more important than a long integration time. However, secondary to the synaptic weight range, a longer neuron integration time does seem to add some benefit. Because of this, we may work to engineer an nTron with lower threshold values or SNSPDs with higher threshold currents, e.g. using thicker superconducting films. Alternatively, we could investigate changing the

EO starting parameters such that neurons start with a large number of synaptic connections. We also find evidence that the presence of time-dependent variables, such as synapse delay may be critical for solving certain tasks. In this case, we could only balance a pole without velocity information (requiring memory of past events) if the model included an evolvable timing delay.

One important lesson for SOEN circuit design using this programming framework is that it is important to use the right input encoding for a particular device. For example, in the case of pole balancer, the SOEN circuit would not solve the task without rate encoding for some parameter sets. The encoding that is used will effect the networks generated, and may effect the outcome of the optimization (i.e. whether or not it is possible to find networks to solve a particular task).

There are many further questions that arise from this study. Intuitively, we expect that integration is more important in applications where inputs arrive to the network stochastically. In future work we will investigate whether we find this to be the case. We also plan to investigate whether it is possible to use other time-dependent variables in place of the synapse delay in SOEN circuits. It would also be interesting to investigate if a variable refractory period or diverse synaptic leak rates could be used to similar effect as neuron and synapse delay, and for which tasks this is important. If these features are crucial for many tasks, then we will need to include them in future SOEN circuit models.

This is an early model that we hope to use as a stepping stone towards more concrete designs and, ultimately, experimental demonstrations. Future work will be to verify these models against network simulations using low level software such as Verilog A. We will also verify the designed networks against experimentally fabricated networks. In this case, we will also see if further optimization can be performed on the hardware itself. Finally, we are also interested in extending the framework to implement an energy and area evaluation of the produced networks. Ultimately this could allow the reward of networks that use lower energy and minimize total wiring.

This is a contribution of NIST, an agency of the US government, not subject to copyright.

## REFERENCES

- [1] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A Survey of Neuromorphic Computing and Neural Networks in Hardware," may 2017. [Online]. Available: <http://arxiv.org/abs/1705.06963>
- [2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, may 2015. [Online]. Available: <http://www.nature.com/articles/nature14539>
- [3] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, dec 1997. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0893608097000117>
- [4] W. Gerstner and W. Kistler, *Spiking neuron models*, 1st ed. Cambridge: Cambridge University Press, 2002.

- [5] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C.-K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y.-H. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, jan 2018. [Online]. Available: <http://ieeexplore.ieee.org/document/8259423/>
- [6] J. Shainline, S. Buckley, R. Mirin, and S. Nam, "Superconducting optoelectronic circuits for neuromorphic computing," *Phys. Rev. App.*, vol. 7, p. 034013, 2017.
- [7] J. M. Shainline, S. M. Buckley, A. N. McCaughan, J. Chiles, R. P. Mirin, and S. W. Nam, "Superconducting Optoelectronic Neurons I: General Principles," may 2018. [Online]. Available: <http://arxiv.org/abs/1805.01929>
- [8] J. Chiles, S. Buckley, N. Nader, S. W. Nam, R. P. Mirin, and J. M. Shainline, "Multi-planar amorphous silicon photonics with compact interplanar couplers, cross talk mitigation, and low crossing loss," *APL Photonics*, vol. 2, no. 11, p. 116101, nov 2017. [Online]. Available: <http://aip.scitation.org/doi/10.1063/1.5000384>
- [9] J. Chiles, S. M. Buckley, S. W. Nam, R. P. Mirin, and J. M. Shainline, "Design, fabrication, and metrology of 10 100 multi-planar integrated photonic routing manifolds for neural networks," *APL Photonics*, vol. 3, no. 10, p. 106101, oct 2018. [Online]. Available: <http://aip.scitation.org/doi/10.1063/1.5039641>
- [10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. [Online]. Available: <http://ieeexplore.ieee.org/document/726791/>
- [11] J. S. Plank, G. S. Rose, M. E. Dean, C. D. Schuman, and N. C. Cady, "A Unified Hardware/Software Co-Design Framework for Neuromorphic Computing Devices and Applications," in *2017 IEEE International Conference on Rebooting Computing (ICRC)*. IEEE, nov 2017, pp. 1–8. [Online]. Available: <http://ieeexplore.ieee.org/document/8123655/>
- [12] S. Buckley, J. Chiles, A. N. McCaughan, G. Moody, K. L. Silverman, M. J. Stevens, R. P. Mirin, S. W. Nam, and J. M. Shainline, "All-silicon light-emitting diodes waveguide-integrated with superconducting single-photon detectors," *Applied Physics Letters*, vol. 111, no. 14, p. 141101, oct 2017. [Online]. Available: <http://aip.scitation.org/doi/10.1063/1.4994692>
- [13] J. Shainline, S. Buckley, N. Nader, C. Gentry, K. Cossel, J. Cleary, M. Popović, N. Newbury, S. Nam, and R. Mirin, "Room-temperature-deposited dielectrics and superconductors for integrated photonics," *Opt. Express*, p. 10322, 2017.
- [14] A. N. McCaughan and K. K. Berggren, "A Superconducting-Nanowire Three-Terminal Electrothermal Device," *Nano Letters*, vol. 14, no. 10, pp. 5748–5753, oct 2014. [Online]. Available: <http://pubs.acs.org/doi/10.1021/nl502629x>
- [15] J. M. Shainline, A. N. McCaughan, S. M. Buckley, C. A. Donnelly, M. Castellanos-Beltran, M. L. Schneider, R. P. Mirin, and S. W. Nam, "Superconducting Optoelectronic Neurons III: Synaptic Plasticity," may 2018. [Online]. Available: <http://arxiv.org/abs/1805.01937>
- [16] J. M. Shainline, S. M. Buckley, A. N. McCaughan, M. Castellanos-Beltran, C. A. Donnelly, M. L. Schneider, R. P. Mirin, and S. W. Nam, "Superconducting Optoelectronic Neurons II: Receiver Circuits," may 2018. [Online]. Available: <http://arxiv.org/abs/1805.02599>
- [17] M. Tinkham, *Introduction to Superconductivity*, 2nd ed. Dover, 1996.
- [18] T. V. Duzer and C. Turner, *Principles of superconductive devices and circuits*, 2nd ed. USA: Prentice Hall, 1998.
- [19] A. M. Kadin, *Introduction to superconducting circuits*, 1st ed. USA: John Wiley and Sons, 1999.
- [20] S. P. Benz and C. A. Hamilton, "A pulsed driven programmable Josephson voltage standard," *Applied Physics Letters*, vol. 68, no. 22, p. 3171, jun 1998. [Online]. Available: <https://aip.scitation.org/doi/abs/10.1063/1.115814>
- [21] K. Likharev, "Superconductor digital electronics," *Physica C*, vol. 482, p. 6, 2012.
- [22] N. Takeuchi, D. Ozawa, Y. Yamanashi, and N. Yosikawa, "An adiabatic quantum flux parametron as an ultra-low-power logic device," *Superconductor Science and Technology*, vol. 1126, p. 1, 2013.
- [23] Q. Herr, A. Herr, O. Oberg, and A. Ioannidis, "Ultra-low-power superconductor logic," *J. Appl. Phys.*, vol. 109, p. 103903, 2011.
- [24] G. Wendin, "Quantum information processing with superconducting circuits: a review," *Rep. Prog. Phys.*, vol. 80, p. 106001, 2017.
- [25] J. M. Shainline, A. N. McCaughan, S. M. Buckley, R. P. Mirin, and S. W. Nam, "Superconducting Optoelectronic Neurons IV: Transmitter Circuits," may 2018. [Online]. Available: <http://arxiv.org/abs/1805.01941>
- [26] F. Marsili, V. B. Verma, J. A. Stern, S. Harrington, A. E. Lita, T. Gerrits, I. Vayshenker, B. Baek, M. D. Shaw, R. P. Mirin, and S. W. Nam, "Detecting single infrared photons with 93% system efficiency," *Nature Photonics*, vol. 7, no. 3, pp. 210–214, feb 2013. [Online]. Available: <http://dx.doi.org/10.1038/nphoton.2013.13>
- [27] C. D. Schuman, J. D. Birdwell, and M. Dean, "Neuroscience-inspired inspired dynamic architectures," in *Proceedings of the 2014 Biomedical Sciences and Engineering Conference*. IEEE, may 2014, pp. 1–4. [Online]. Available: <http://ieeexplore.ieee.org/document/6867735/>
- [28] C.-K. Lin, A. Wild, G. N. Chinya, Y. Cao, M. Davies, D. M. Lavery, and H. Wang, "Programming Spiking Neural Networks on Intel's Loihi," *Computer*, vol. 51, no. 3, pp. 52–61, mar 2018. [Online]. Available: <http://ieeexplore.ieee.org/document/8303802/>
- [29] C. D. Schuman, J. S. Plank, A. Disney, and J. Reynolds, "An evolutionary optimization framework for neural networks and neuromorphic architectures," in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, jul 2016, pp. 145–154. [Online]. Available: <http://ieeexplore.ieee.org/document/7727192/>
- [30] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: <http://archive.ics.uci.edu/ml>
- [31] C. D. Schuman, A. Disney, S. P. Singh, G. Bruer, J. P. Mitchell, A. Klibisz, and J. S. Plank, "Parallel Evolutionary Optimization for Neuromorphic Network Training," in *2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC)*. IEEE, nov 2016, pp. 36–46. [Online]. Available: <http://ieeexplore.ieee.org/document/7835813/>