Neural State Machines

Tatyana Pak[1], Anna Kiepura[1], and Timothy Kurer[1]

[1]**Institute of Neuroinformatics, ETH Zürich and University of Zürich**

ABSTRACT

How the brain makes state dependent decisions is an essential, yet not fully resolved, field of research. The prefrontal cortex (PFC) is known to play a role in combining sensory information and planning responses. One way of performing state dependent computation in spiking neural networks (SNN) is using a neural state machine (NSM), a SNN implementation of finite state machines. Here we explore, characterize and adapt a previously proposed NSM based on a winner take all network (WTA) and a disinhibition gating mechanism. The rich interconnection, recurrence and disinhibition circuitry found in the superficial neocortex point to this implementation being a bioplausible way in which the brain may implement its state dependent computation. Many previous implementations of SNN cortical functions have not been tested in the noisy environment of real neurons or neuromorphic hardware, while this implementation has been tested on a neuromorphic chip (DYNAP-SE) to be robust to noise and device mismatch. Here we show the reproducibility, functionality and as well as suggest alternations to the NSM previously proposed by Liang and Indiveri (2017).

Keywords: Neuromorphic, DYNAP-SE, SNN

1 INTRODUCTION

The prefrontal cortex (PFC) is theorized to coordinate the different brain areas and responses in accordance with an internal representation of the world constructed in the PFC (Liang and Indiveri, 2019). Many models have been shown that can implement such computation in SNN using computational primitives found in the cortex (Rutishauser and Douglas, 2009), in particular one proposed in (Liang and Indiveri, 2017), that implements a neural state machine (NSM) on neuromorphic hardware, capable of not only performing the computation in SNN but also with low resolution parameters and under noisy conditions, as are found in the brain and neuromorphic hardware due to mismatch and low current application. Unlike traditional computing, neuromorphic SNN's are built on analog processors composed of populations of spiking neurons, the dynamics of which are driven by a combination of slow, low power, sub-threshold analog circuits and fast, programmable digital circuits (Chicca et al., 2014). This creates a noisy and asynchronous network, transmitting data in an event based manner as is observed in biology (Faisal et al., 2008). This work in particular was implemented on a DYNAP-SE chip from the institute of Neuroinformatics, UZH, comprised of a total of 4 chips, with 4 cores each, which at 256 neurons per core gives access to a total of 4096 Neurons, each capable of possessing 64 input synapses and transmitting to 1000 output synapses of 4 types (AMPA, NMDA, GABA A, GABA B). This network in particular utilizes a Winner Take All network (WTA) to determine and sustain its state while inhibiting conflicts and approaching a stable state. This WTA is based on recurrent connections between the states and a global inhibitory neuron, driven by the summed activity of all states and suppressing all states, allowing only the winner to sustain itself. This implementation is beneficial for noisy neuromorphic hardware, since noise is suppressed by the global inhibition but is also bio-plausible since similar recurrent connections are found in the superficial cortical layers, where a large percentage of connections arise from short range recurrent connections, which combined with longer range inhibitory input creates a feedback loop that selectively enhances a winning population while suppressing others (Rutishauser and Douglas, 2009). The proposed network further utilizes a dis-inhibition gating mechanism found frequently in the brain (Möhler and Rudolph, 2017) to allow only a single state to listen to an external signal input. Thus the whole network represents a bio-plausible, real implementation of NSM as well as potential possibility for how the brain performs state dependent computation.

2 NETWORK ARCHITETURE

The initial network architecture closely mimics the setup proposed by (Liang and Indiveri, 2017), with a few key differences. One of the main constraints of working on DYNAP-SE is that the parameters for synapse connection type as well as the neuron parameters can only be set once per core. This implementation (Figure 1) shows a single chip (4 core) implementation that sets connections such that each connection type is only used once per core, allowing for maximum control of parameters. To achieve this NMDA is used exclusively as an AMPA substitute, by setting its threshold to 0 on all cores. Thus AMPA and NMDA can be viewed interchangeably in this report.

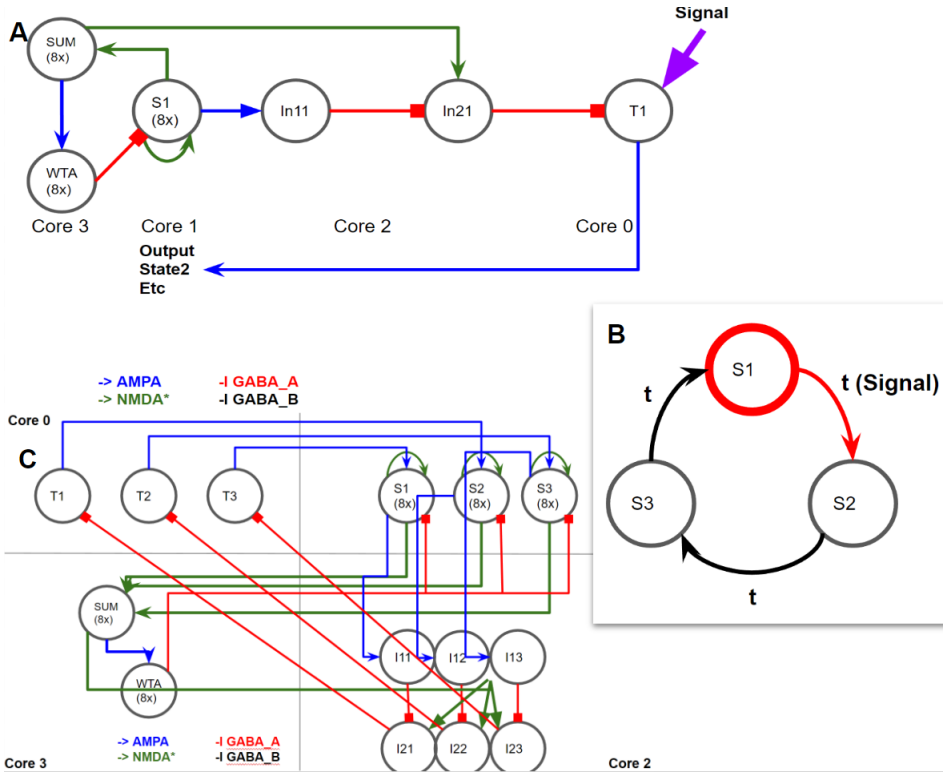

Figure 1. A. Basic network architecture for single state, with connection types marked. B: Finite state machine equivalent to implemented NSM. C: Complete architecture explored in this article.

The WTA contains 5 neuron groups composed of 8 neurons. (Figure 1 A) 3 state groups with recurrent all 2 all connections, a sum of states group and a global inhibitory group (WTA). The recurrency within states allows state groups to sustain their activity in the absence of input. The sum group is driven by all the activity in the states, and itself drives the activity of the global inhibitory neuron group as well as the inhibitory 2 neurons from the dis-inhibition gate. The global inhibitory group suppresses all state neurons equally, thus creating a feedback loop between state activity and suppression of states. The strength of connections in this loop decides key properties of the network. Primarily it suppresses activity in all but the most active state, leading to WTA behavior. The balance between negative feedback and recurrency based sustaining of states ideally leads to a constant state activity in only a single state, in the absence of any inputs. Further the ratio of inhibition and recurrency also determines the threshold a signal needs to push a non winning state to, to allow it to take over. The disinhibition gate and signal mechanism consist of single neurons only. Inhibitory 2 is driven by the sum of states and thus always active, suppressing the activity of their respective transition neurons. However the inhibitory 1 neuron can be activated by its respective state, inhibiting inhibitory 2 and thus disinhibiting the respective transition neuron. Thus a mirrored pattern of activity to the states is expected in transitions and inhibitory 1 neurons compared to the states - with only one state active only its respective inhibitory 1 and transition neurons should be firing. For inhibitory 2 this pattern is reversed. This disinhibition gate allows only a single transition to listen to a signal sent to all transition neurons simultaneously. Ultimately the positive connections

from transitions to states mean that state transition is expected if a signal is sent, only if the internal state matches the transition, implementing a simple FSM shown in (Figure 1 B). Precise and laborious tuning was necessary for the network to show all of these properties simultaneously, though both the WTA and disinhibition gate work efficiently and robustly individually. This basic network can be scaled to add further states as well as different signals and state transitioning and output patterns.

3 METHODOLOGY

3.1 Code setup

Some general remarks about our code:

- Function to initialize parameters based on core numbers;
- Neuron params unchanged for all cores;
- DC current set to 0 in all cores, except for the states neurons;
- One poisson input to a state for initialisation, and after a pause another poisson input to transitions;
- Issue with poisson generator: input gets sent to both states and transitions at the same time;
- Issue with spike generator: input gets repeated after 1s and the pause between inputs should not exceed 0.6s.

3.2 Winner-Take-All: fine-tuning

While fine-tuning the WTA part of the network, we observed the following:

- Very strong recursive connection in the states neurons (almost maxed out) was required; otherwise sustaining a state was not possible;
- Multiple neurons per state group are necessary for sustaining behaviour;
- The firing rate of the WTA neurons had to be at least 5 times higher than the firing rate of the SUM neurons.

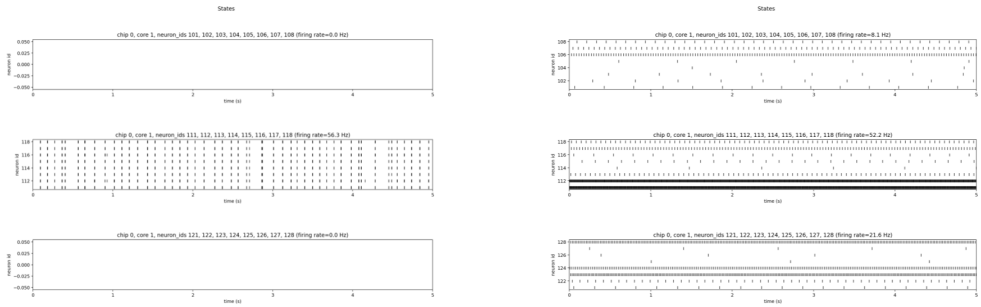

Figure 2. States with no recursion, no inhibition, no input from transitions (left) and states with no input from transitions only (right).

3.3 AND/Disinhibition gate: fine-tuning

A functioning AND gate should cause the transitions to fire only when the two conditions are met: (1) The input was sent to transitions. (2) The corresponding state is currently firing (the winner state). This implies that even though we send the input to all three transition neurons simultaneously, only one transitions neuron should fire at a time - the one that corresponds to the state that is currently active (the winner). At rest (when the transitions are not firing), all three transitions neurons are inhibited by the inhibitory neuron 2 (inh2). When the winner state is firing, it excites the inhibitory neuron 1 (in1), which in turns inhibits inh2, activating the transitions neuron via a disinhibition mechanism.

The parameters of the AMPA connections between the inputs and the transitions (leakage, gain, and weight), the AMPA connections between the states and inh1, and the GABA_B connections between inh1

and inh2 turned out to be most crucial for obtaining a functioning AND gate. Making the connections between the inputs and transitions strong enough to trigger spiking, but at the same time weak enough to avoid spiking in more that one transition neuron at a time. We tried solving this issue by altering the connection strength between inh2 and the transitions, but it turned out to be insufficient to solve the problem. Eventually, we managed to obtain the desired mechanism by including an extra layer between inh2 and transitions, which will be explained in the next section.

The figure below shows an example behaviour of a functional AND gate, however, this result is still not perfect as this figure was produced using one of the earlier versions of the network. In this recording, we were sending a Poisson input to state 1 for the first 0.5 seconds, and then to the transitions from 1.0 to 1.2 second of the recording). After we initialised the states, we can see that only one state is firing, and it is able to sustain itself in the absence of subsequent input. We also observed that in this case, the transitions fire before we send an input to them, which constitutes one of the issue we encountered while tuning the network (we explain the solution on page X). The most important part demonstrating that the AND gate is working properly is visible in the figure below 4 seconds after starting the recording; although the input is sent to all three transition neurons simultaneously, only the transition neuron that corresponds to the currently winning neuron is firing. Moreover, the activity of inh1 is mirroring the activity of the state neurons, which is due to the fact that the inh1 neurons are receiving excitatory connection from the corresponding state neurons. The activity of inh2 should be the opposite of the activity of inh1, meaning that while inh1 should only be active in the winning state, inh2 should only be inhibited in the winning state, as it is visible in the figure below.

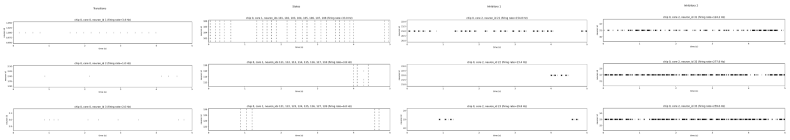

Figure 3. AND Gate Firing (Left to right: Transitions, states, inh1, inh2)

We tested the AND gate further, by examining the network behaviour in absence of inputs to the transition neurons. Activity in the transitions was still observed, but only in Transition 1, and the firing rate was constant throughout the recording. After the activity in State 1 initialised, it was able to sustain itself. The changes of the active state do not take place, demonstrating that the input to transitions is indeed a necessary condition to trigger the active state shift, which is one of the desired features of our network. Moreover, we can see that in1 is only firing in State 1, while inh2 is inhibited in State 1, which indeed constitutes the behaviour we would expect from our architecture.

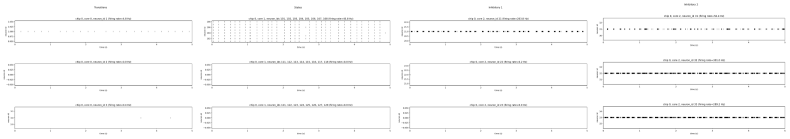

Figure 4. AND Gate without input to transitions. (Left to right: Transitions, states, inh1, inh2)

3.4 Transitions: fine-tuning

Transitions firing should trigger the switch of the current active state to the next state in the correct order ($1 \rightarrow 2 \rightarrow 3 \rightarrow 1$, as shown in the network diagram). The key parameters for obtaining the desired behaviour turned out to be the ones specifying the strength of the excitatory AMPA connections between the transition neurons and state neurons (leakage, gain, and weight of the AMPA synapses).

Figure figure below illustrates the main challenge that we encountered during tuning the network so that it follows this pattern: after the input was sent to the transitions and causes the firing of Transition 1, we immediately observe firing in Transition 2, and then Transition. 3. The activity in State 2 was most likely too brief to show spikes in the plot, but we can observe spiking in State 3. Then, another switch to State 1 was observed. This problem of "Cycling between states" is also demonstrated in Figure 4 and it most likely occurs due to the issue with spike generator that we explain on page X. In Figure 5, we can see a good example of the transition mechanism functioning correctly, as firing in Transition 1 triggers

the switch of the active state from State 1 to State 2, whose activity is sustained after the switch. However, it also suggests that our network is very sensitive to the activity in the transitions, as only one spike is enough to trigger the state transition. This constitutes one of the drawbacks to our architecture, as ideally the network should be resistant to very low activity in the transitions, as it could be triggered by noise, which is widely present and intrinsic to neuromorphic hardware architectures.

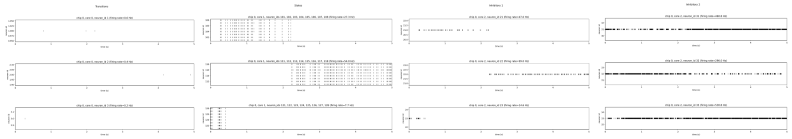

Figure 5. Transitions (Left to right: Transitions, states, inh1, inh2)

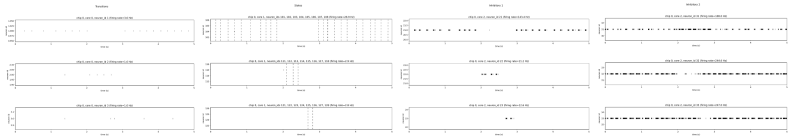

Figure 6. Transitions and States Cycling (Left to right: Transitions, states, inh1, inh2)

4 RESULTS

4.1 Final version

We took into account everything we observed in earlier versions to develop the final version. First, we addressed the issue of not self-sustaining states. We removed direct Poisson input to the states, since due to the winner-take-all architecture, the state with a slight advantage must be able to suppress the others. However, that was still not enough to have a full transition from one state to another. We switched some of the states neuron ids to ensure that all of them had similar responses to the same input.

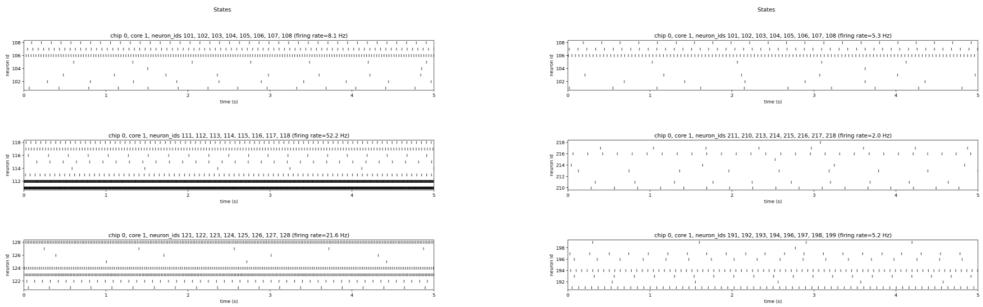

Figure 7. States with no recursion, no inhibition, no input from transitions. Before tuning (left), after tuning (right)

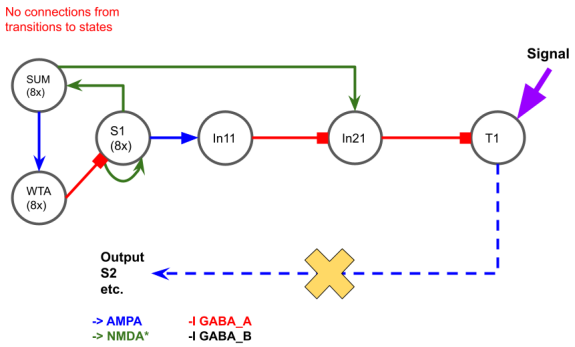

Figure 8. Network architecture with no input from transitions.

To further fine-tune the network, we removed the connection from transitions to states, and discovered that all the layers, except the transition layer, were working properly, Despite the inhibition from the Inhibitory 2 layer, the corresponding neurons of Transitions were not getting suppressed.

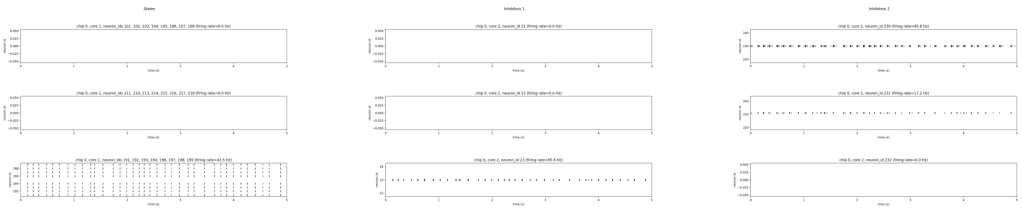

Figure 9. States, inhibitory 1 and inhibitory 2 with no input from transitions.

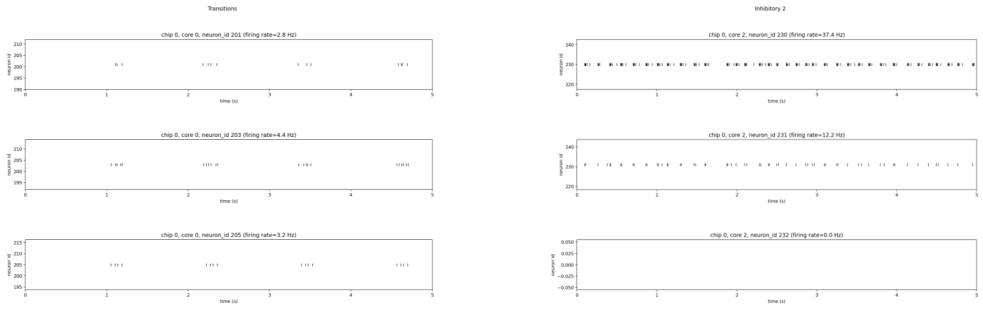

Figure 10. All neurons of transitions (left) still spiking despite the suppression from inhibitory 2 (right).

This posed a problem since:

1. Decreasing AMPA to transitions led to them not spiking at all;

2. Altering the parameters for inh2 broke the balance of excitation and inhibition to that layer.

Thus, we decided to add an extra layer between Inhibitory 2 and transitions to amplify inhibition.

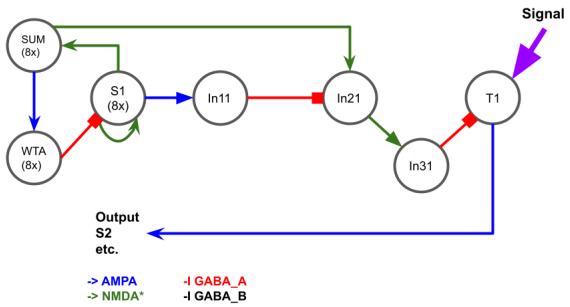

Figure 11. Network architecture with the extra inhibitory layer.

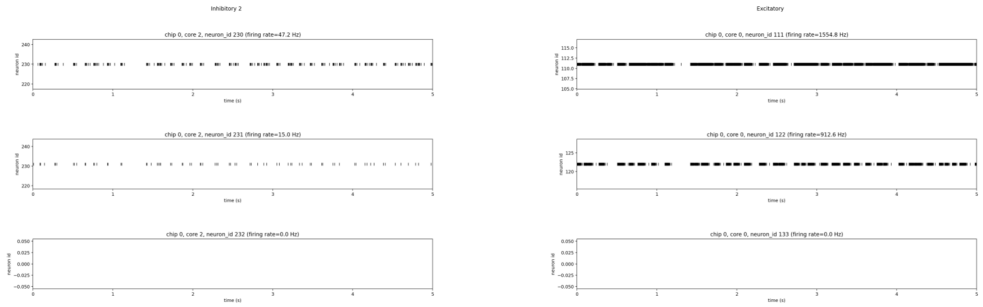

Figure 12. Inhibitory 2 (left) output amplified by Layer 3 (right).

We also removed the connection from Inhibitory 1 to Inhibitory 2 to fine-tune the responsiveness of Inhibitory 2 neurons and ensure they had similar firing rate in response to the same input.

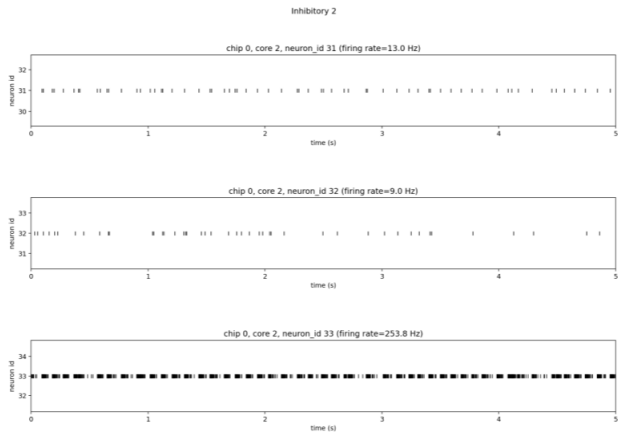

Figure 13. Varying responses of Inhibitory 2 neurons.

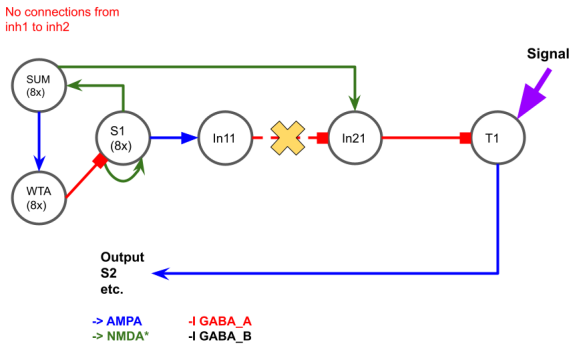

Figure 14. Network architecture with no connection from Inhibitory 1 to Inhibitory 2.

Finally, we could observe transitions neurons being properly suppressed and only firing in accordance with the current state. We added all connections back and had a state transition and self-maintain from initial state 3 to state 1. We were also able to demonstrate the same for initial states 1 and 2.

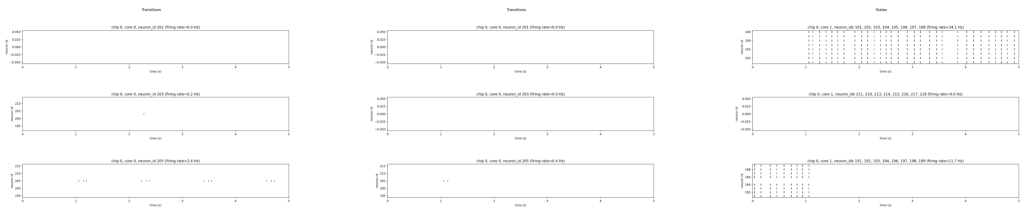

Figure 15. Transitions, properly suppressed by the inhibitory layer: with no connection to the states (left), with connection to the states (center), and states changing due to transitions (right).

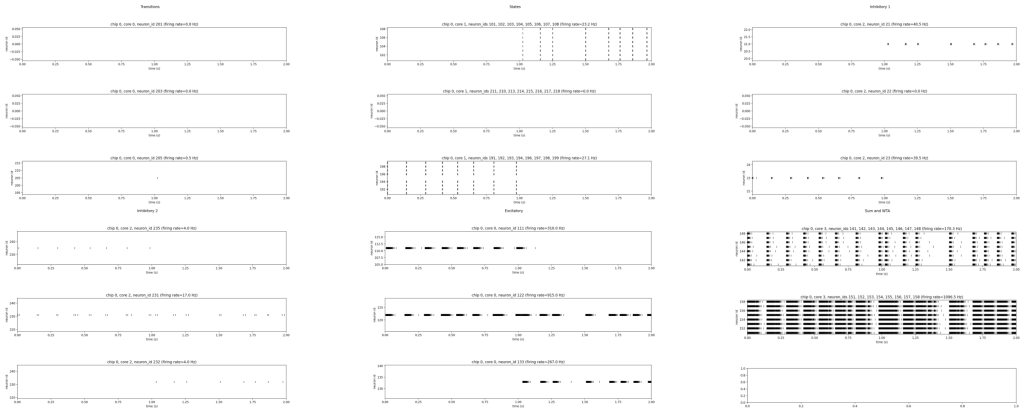

Figure 16. All layers of the network demonstrating a state change from state 3 to state 1. (Top left to bottom right: Transitions, states, inh1, inh2, layer3, sum & WTA)

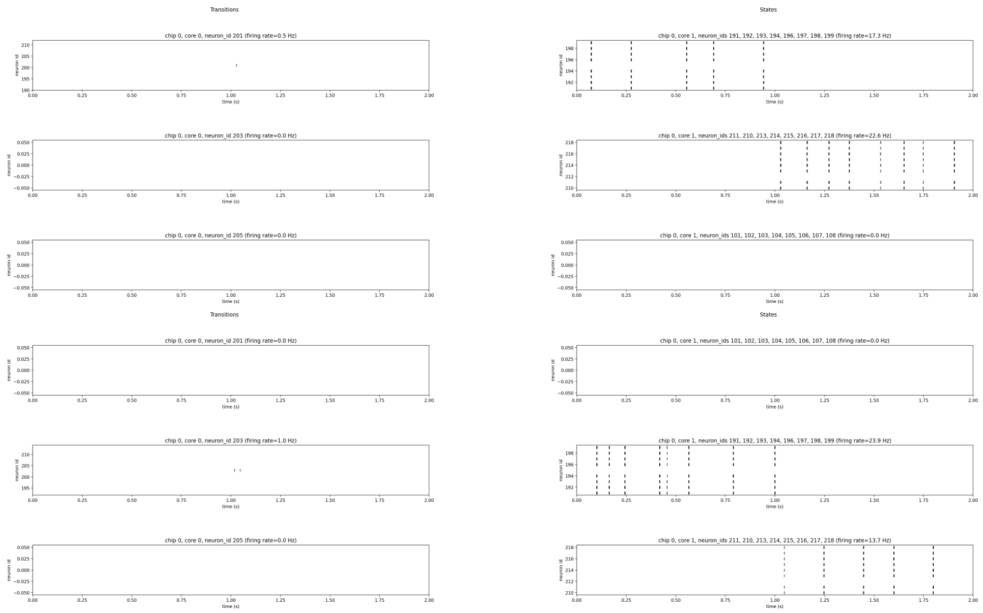

Figure 17. Transitions (left) and states (right) demonstrating a state change from state 1 to state 2 (up) and from state 2 to state 3 (down).

Making use of the repeated input sent by the spikegenerator approximately every second, we ran the network for 4 seconds, and observed correct state changes due to each input at about 1s, 2s and 3s. We also plotted the states' mean firing rate, which once again demonstrated the states changing from 3 $\rightarrow$ 1 $\rightarrow$ 2 $\rightarrow$ 3, immediately following the input to transitions.

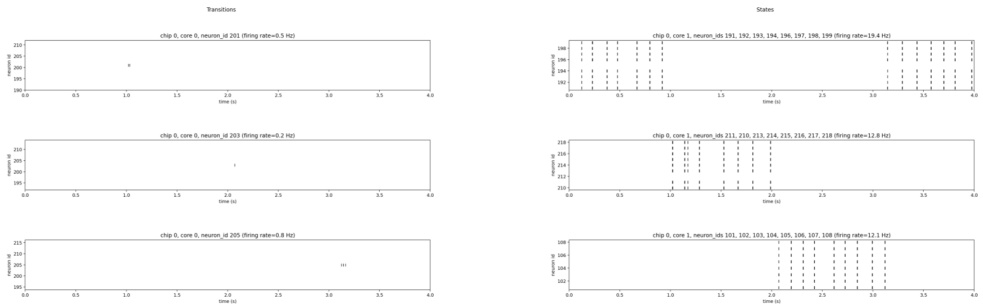

Figure 18. Transitions (left) and states (right) of a neural state machine with multiple inputs to transitions.

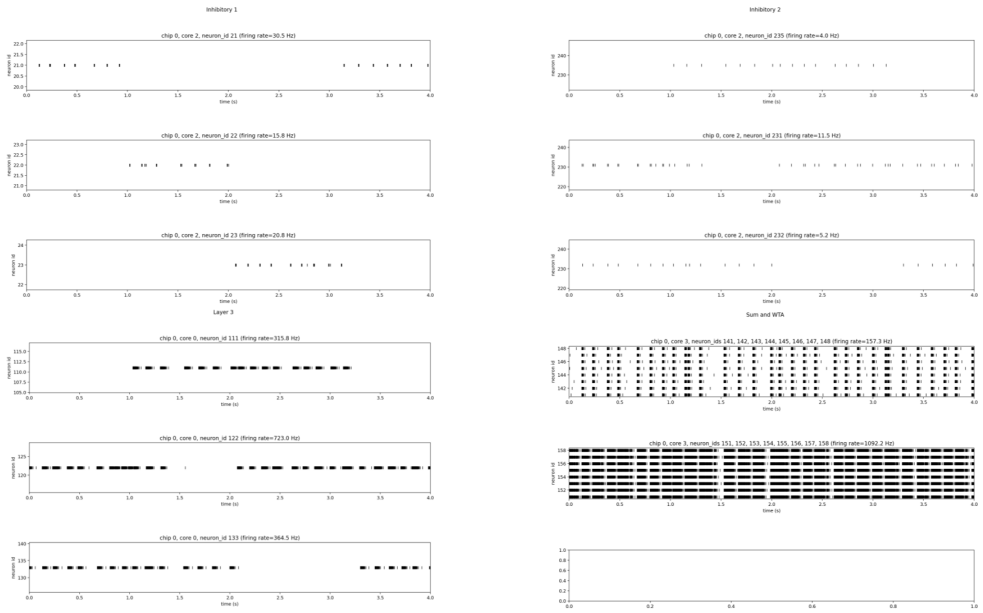

Figure 19. Secondary layers of a neural state machine with multiple inputs to transitions. (Top left to bottom right: inh1, inh2, layer3, sum & WTA)

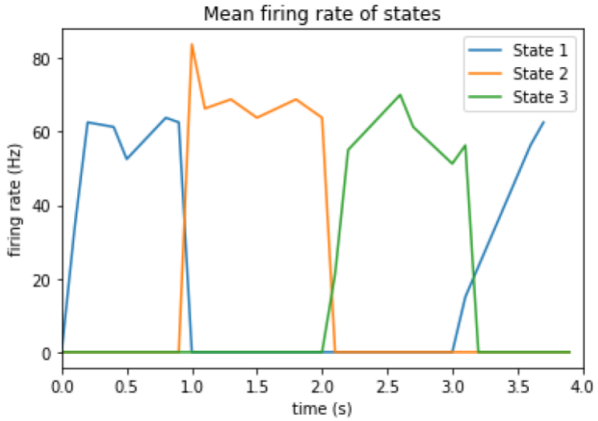

Figure 20. Mean firing rate of states with multiple inputs to transitions.

4.2 DYNAP-SE API & Board issues

For future work to avoid, a few key and pitfalls errors discovered in the May 2022 version of the SAMNA API and DYNAP-SE are as follows:

- The poissongen function seemingly created incorrect connections on chip, if multiple poissongens and target neurons were used sequentially (i.e. turning one ON after the other has been turned off, to mimick signal intervals). The poissongen signal was sent to other neuron groups besides the target. This seems to have been an API issue but was not further explored since FPGA spikengen functions are preferably used
- At an ISI of 900, no more than 0.6s of no input should be passed to the FPGA spikegen, which can be especially troublesome when utilizing low frequency poisson inputs that can stochastically create such a gap.
- When ran remotely from Jupyter Notebook, FPGA spikegen seems to send its signal correctly but then repeat itself periodically even though repeat mode is set to false

- Noisy neurons change over time and need to be carefully checked periodically, particularly in neuron groups with strong recurrent connections in which noise from a single neuron is amplified to the group
- Events reported by SAMNA seem to not always correspond to on chip activity and need to be interpreted with caution
- Daily changes between noisy neurons, temperature etc call for veray robust networks when working with DYNAP-SE

5 CONCLUSION

We have shown that the network shown by Liang and Indiveri (2017) can be implemented on DYNAP-SE and used for simple FSM computations. The balance achieved shows promise to be expanded further for computing more complex FSM calculations as the basic network structure is easily scaleable, though retuning will be required. Additionally changes in the network architecture, such as an additional excitatory layer to amplify inhibition have been proposed and show to solve issues with robustness and silencing of signal input. However one of the main requirements of SNN implementation on neuromorphic hardware, robustness, has not been fully achieved and would require further tuning. Due to the large time effort of tuning a new strategy to achieve robustness should be aimed for, whether that be in changing the network architecture, tuning neuron parameters as well as synapse parameters or following a different tuning routine. Further characterization of the network, particularly in which signals it can process is necessary, including signal ranges, thresholds and FF curves. For FSM type computation a secondary network that processes signals into symbols with intervals as is done for digital FSM could also be implemented and may also be biologically accurate as oscillating brain waves may implement a similar interval for signal processing in the brain.

REFERENCES

Chicca, E., Stefanini, F., Bartolozzi, C., and Indiveri, G. (2014). Neuromorphic electronic circuits for building autonomous cognitive systems. *Proceedings of IEEE*, 102.

Liang, D. and Indiveri, G. (2017). Robust state-dependent computation in neuromorphic electronic systems. In *2017 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–4.

Liang, D. and Indiveri, G. (2019). A neuromorphic computational primitive for robust context-dependent decision making and context-dependent stochastic computation. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 66(5):843–847.

Möhler, H. and Rudolph, U. (2017). Disinhibition, an emerging pharmacology of learning and memory. *F1000Research*, 6:101.

Rutishauser, U. and Douglas, R. J. (2009). State-Dependent Computation Using Coupled Recurrent Networks. *Neural Computation*, 21(2):478–509.