Exploiting Refractory Period for Functional Multiplexing and Short-Term Memory in Spiking Neural Networks

Zhenduo Zhai[1] and Ismail Akturk[2]

***Abstract*— Spiking Neural Networks (SNNs) have recently received attention in robotics due to their low power and efficiency prospects. However, we argue that existing implementations of SNNs don't exploit the greater potential inherent to spiking neurons – particularly refractory period – that could enable functional multiplexing and short-term memory. We demonstrate how refractory period enables functional multiplexing and form a short-term memory in SNNs which would support complex functionalities, and learning methods with smaller number of neurons compared to traditional SNNs implementations that do not model the refractory period.**

I. INTRODUCTION

Historically, it has been believed that the intelligence is based on reasoning, where the logic (and its computation) is the foundation of reasoning [1]. In this regard, the very first attempt was made by McCulloch and Pitts, where they demonstrated the neurons are capable of compute the basic logical functions [2]. By combining such basic logic gates, complex functions can be realized. Although this is a viable approach (i.e., implementation of each basic logic gates via set of neurons and building more complex functions from them), it may render a less efficient way of utilizing the SNNs, in terms of area and energy/performance.

We argue that exploitation of intrinsic properties of spiking neurons (in particular refractory period behavior) could allow us to realize complex logical functions with smaller number of neurons compared to implementing logic gates with conventional neural models, where each neuron is considered to be ready to spike immediately regardless whether it has fired recently, or not (refractory period is just considered as intrinsic delay and has not been exploited). Fig. 1 shows behavior of a typical biologically-plausible spiking neuron. A neuron in resting state would spike, once its membrane potential reaches a certain threshold (by integrating incoming stimuli); and then, it enters in a *refractory period* in which it cannot spike until the end of this period, regardless of the strength of the stimulus it may have.

In this paper, we propose to use more biologically-plausible artificial spiking neurons (that employs refractory period), in particular to enable *functional multiplexing* and to form *short-term memory* in SNNs.

[1]Department of Electrical Engineering, and Computer Science, University of Missouri, Columbia, MO, USA zz7z9@mail.missouri.edu

[2]Department of Electrical Engineering, and Computer Science, University of Missouri, Columbia, MO, USA akturki@missouri.edu

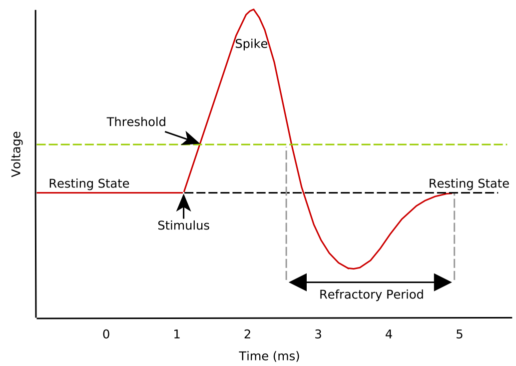


Fig. 1. Typical action model of a spiking neuron.

II. FUNCTIONAL MULTIPLEXING

Functional multiplexing is a method that allows SNN to realize different functions during distinct time periods without using separate network or extra neurons.

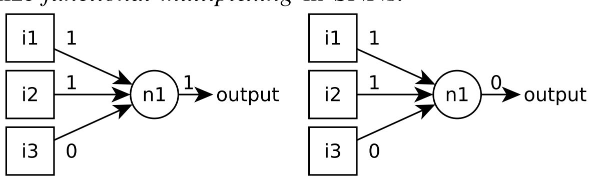
The way functional multiplexing in SNNs works as follows. Considering a network of spiking neurons, when a set of neurons spike at a given time (let's say at t_0), they cannot spike again until the refractory period ends (assume the duration of refractory period is t_r, they would not spike until t_1 where $t_1 = t_0 + t_r$). This is very unique feature that a given network would have a distinct set of active neurons at any given time, allowing it to realize different functions. In a sense, the whole network is time-multiplexed at each time step based on what has been computed in the previous time step (i.e., which neurons were active and have spiked) . Such multiplexing is promising to realize a complex functions with fewer spiking neurons compared to naive way of implementing functions with traditional neuron-based building blocks (e.g., logical gates) in which neurons were considered to be available all the time (i.e., having no refractory period).

A. Basic Assumptions and Conventions Used

Before diving into details of the proposed work, here we describe the assumptions and conventions that we used throughout the paper. Each spiking neuron has a threshold that specifies the minimum amount of stimuli (i.e., active inputs) needed for it to spike. We consider a spike represents logical '1', whereas no spike represents logical '0'. Without loss of generality, we also assume that each connection has the same weight (this is for simplifying the discussion; otherwise, it is not a constraint for the proposed scheme to work). Two types of neurons are assumed in SNNs: i) excitatory, ii) inhibitory neurons, where excitatory neurons contribute to spiking probability of the (post-)connected neurons; and inhibitory neurons prohibit (post-)connected neurons to spike. For the illustration of excitatory and inhibitory neurons, we use circle and double-circle nodes in the figures, respectively.

B. Neuron Behavior in Resting State vs. Refractory Period

Functional multiplexing in SNNs is based on the fact that the behavior of a neuron differs depending on its current state, even if the stimuli (i.e., inputs) remain intact. For our discussion, a neuron can be in one of the following three states: i) *resting*, ii) *spiking*, and iii) *refractory period*. When a neuron is in *resting* state, it accumulates the inputs connected to it, and when they reach to threshold, a spike is emitted. After a neuron spikes, it goes into refractory period in which it becomes unresponsive, regardless of the strengths of the inputs. At the end of refractory period, a neuron gets into resting state again. We propose to exploit the distinct behaviors of a neuron in resting and refractory period to realize *functional multiplexing* in SNNs.



(a) in resting state when $t < t_0$; and spiking when $t_0 \leq t < t_0 + t_s$

(b) in refractory period when $t_0 + t_s \leq t < t_0 + t_s + t_r$

Fig. 2. The behavior of a neuron in different states: a) spiking; b) refractory period while inputs remain the same, i.e., $[i_1, i_2, i_3]$ = { 1, 1, 0 }.

Fig. 2 illustrates the behavior of a neuron in (a) resting, and (b) refractory period, while the given inputs remain intact. Given that i_1 and i_2 are '1' at time t_0, the neuron n_1 has been in resting state accumulates these inputs and emits a spike when they reach the threshold (assuming threshold of n_1 is two). From the time emitting a spike to entering into refractory period (i.e., t_s), the output becomes logical '1' (Fig. 2a). Following emitting a spike, neuron n_1 enters into refractory period and remains unresponsive to the inputs until it reaches resting state, once again. From the time entering into refractory period to reaching the resting state (i.e., t_r), the output becomes logical '0' (Fig. 2b). Although we do not impose any particular value, the duration of t_s and t_r can be used as a knob in the design process of SNNs for functions to be implemented. Assessment of under which constraints and circumstances different t_s and t_r values provide better design alternatives is open question and left as a future work.

C. Neuron Behavior as Inputs Change as a Function of Time

To demonstrate functional multiplexing over time, we need to show that a neuron has a distinct behavior as: i) internal state changes while inputs remain the same, and ii) internal state changes along with the inputs (as a function of time). We illustrate a scenario for both (i) and (ii), below.

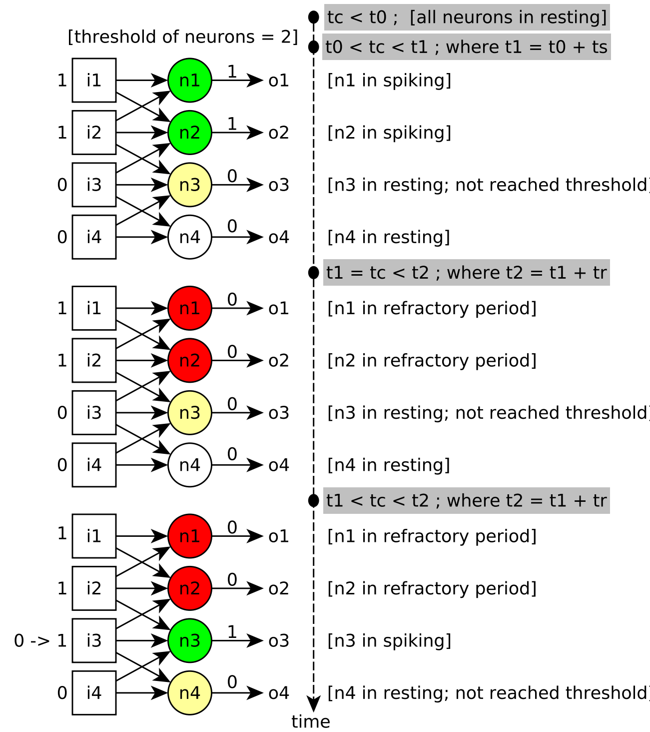


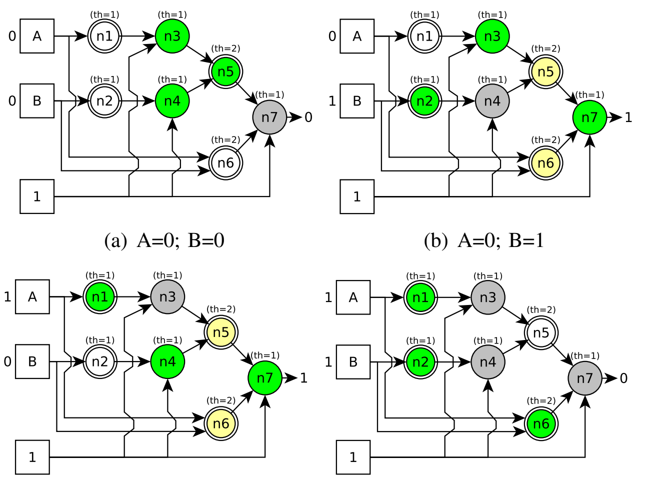
Fig. 3. Neurons' responses differ as both internal states and inputs change over time (from t_0 to t_2; t_c represents current time). A set of responsive neurons changes over time as some of them get into refractory period, illustrating a possibility of functional multiplexing. Colors represent the following. Green: spiking; yellow: resting, not reached threshold; white: resting; red: refractory period.

In Fig. 3, there are four spiking neurons – from n_1 to n_4 with a threshold of two – that are connected to four distinct inputs (i_1 to i_4). Initially, all neurons are in resting state. When the inputs are set to be $\{1, 1, 0, 0\}$ at time t_0, neurons n_1 and n_2 reach the threshold, so they emit spikes (i.e., the output for both are logical '1'). On the other hand, since neuron n_3 has received input of logical '1' only from input i_2, it could not reach the threshold; so it remains silent (i.e., output logical '0'). Neuron n_4 has received no input, so it also remains silent in resting state. After neurons n_1 and n_2 spike, they go into refractory period at time t_1, in which their output becomes logical '0'. Notice that both inputs i_1 and i_2 remain $\{1, 1\}$, but they cannot let neurons n_1 and n_2 to spike, since they are in refractory period. Virtually, these neurons disappear from the network, and so cannot contribute to the computation (or function) to be performed during this period, while the rest of the network (i.e., neurons n_3, and n_4) remain responsive to the changes to inputs. While the neurons n_1 and n_2 are still in refractory period, the input i_3 changes from logical '0' to '1' (at time t_c where $t_1 < t_c < t_2$). By this change, neuron n_3 reaches the threshold and emit a spike (i.e., output logical '1'). Although, neuron n_4 receives an input of logical '1' from i_3, it remains silent in resting state, as it couldn't reach the threshold. Once again, notice that with the input change in place, both neurons n_1 and n_2 remain unresponsive due to being in refractory period (although they could have spiked if they were in resting state). In short, the internal states of the neurons dictates how to react the changing inputs that can be exploited as functional multiplexing (i.e., the given network could be used to realize a particular function from time t_0 to t_1; and other function from time t_1 to t_2, as the responsive neurons of the network changes over time).

In the following, we show an SNN that employs functional multiplexing to realize both XOR and NAND.

D. Case Study: Multiplexing for XOR and NAND

In Fig. 4, n_1, n_2, n_5 and n_6 are inhibitory neurons, whereas n_3, n_4 and n_7 are excitatory neurons. Neurons n_5 and n_6 have threshold of two, while the rest of the neurons have threshold of one (thresholds are indicated on top of each neuron). There are two controlled inputs (A and B) changing over time, and a constant input that always provide logical '1' (at anticipated time). The illustrated network can act as XOR or NAND at distinct time periods. The logic to be realized at a given time depends on the current internal states of the neurons (i.e., which neurons are in refractory period, and which are in resting state – which are determined by the recent values of A and B).



(a) A=0; B=0

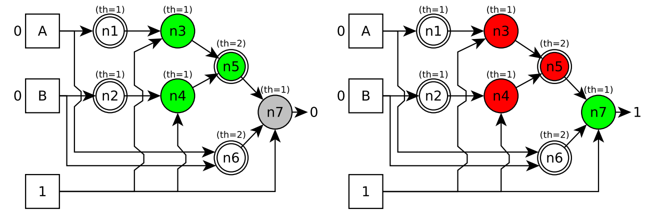
(b) A=0; B=1

(c) A=1; B=0

(d) A=1; B=1

Fig. 4. A network acts as XOR and NAND (except when A=B=0, in which case network acts as XOR, initially) when all neurons are in resting state. Color code is the same as in Fig. 3, with an addition of gray: inhibited.

Both NAND and XOR generates logical '1', if any one of the inputs is 1 (but not both: A=1 & B=0, or A=0 & B=1). Similarly, both NAND and XOR generates logical '0' if both inputs are 1 (i.e., A=B=1). However, if both inputs are '0' (i.e., A=B=0), then NAND would generate '1', whereas XOR would generate '0'. Now, assuming all the neurons are in resting state initially, the network would behave as both XOR or NAND, unless the inputs A and B are '0' (if both A and B are '0', then network acts as XOR, initially). Specifically, let us consider when both inputs are 0 to demonstrate the network acts as XOR, but not NAND when all the neurons are in resting state. In Fig. 5(a), excitatory neurons n_3 and n_4 would spike, which are connected to inhibitory neuron n_5 that would spike, as well (since it would reach the threshold). Then, neuron n_5 would inhibit neuron n_7. Although neuron n_7 has input of logical '1', it cannot spike due to inhibition imposed by neuron n_5, so the output would be logical '0', confirming to XOR behavior (when both inputs are '0'). For all other input combinations (i.e., A=0, B=1; A=1, B=0; A=B=1), the given network would confirm to both XOR and NAND, when all the neurons are in resting state initially (as shown in Fig. 4b–d).



(a) network acts as XOR at $t_n < t_c < (t_n + t_s)$ where t_n is the time n_5 emits a spike

(b) network acts as NAND, as neurons n_3, n_4, and n_5 are in refractory period during $(t_n+t_s) \leq t_c < (t_n+t_s+t_r)$

Fig. 5. A network acts as NAND during the neurons n_3, n_4, and n_5 are in refractory period upon acting as XOR, when both inputs A and B are '0'. Color code is the same as in Fig. 4.

After the neurons n_3, n_4, and n_5 spike, they would be in refractory period for t_r amount of time. During this period of time, the network would behave as NAND gate, as illustrated in Fig. 5 when both inputs remain the same (i.e., A=B=0). Particularly, since the inhibitory neuron n_5 (and its connected neurons n_3 and n_4) is in refractory period, it cannot inhibit the neuron n_7, thus neuron n_7 spikes, confirming to NAND (when both inputs are '0', the output becomes '1').

While this case study demonstrates the feasibility of exploiting refractory period for XOR and NAND in SNNs; we believe that exploitation of refractory period can easily be extended to more complex functions, allowing to realize them without adding separate building blocks or resources (i.e., more functionality with less number of neurons – this would translate into area, power and likely performance improvements). For the given XOR and NAND study, it would require a total of 10 spiking neurons to implement them, separately (4 inhibitory and 4 excitatory neurons for XOR, and single inhibitory and excitatory neurons for NAND). However, with functional multiplexing by exploiting refractory period of neurons, both XOR and NAND can be realized within the same network of 7 spiking neurons (4 inhibitory and 3 excitatory neurons, as shown in Fig. 5).

Considering the constraints of robotics applications in terms of area, power and performance, SNNs appears to be an attractive path to pursue for enabling real-time interaction and learning. While conventional SNNs do not model the refractory period, we argue that modeling and exploiting refractory period would further open up a path for improved efficiency and performance in these domains. Particularly, functional multiplexing (as discussed above) and inherent short-term memory feature of refractory period (as discussed in the next section) are two directions that can be pursued.

III. Short-Term Memory

As mentioned earlier, a spiking neuron can be in one of the states at any given time: i) resting state, ii) spiking, and iii) refractory period. Particularly, the existence of refractory period gives an inherent short-term memory capability to a neuron. A neuron can be in refractory period only if it recently spiked (i.e., $(t_x + t_s) \leq t_c < (t_x + t_s + t_r)$ where t_x is the time a neuron spiked most recently, t_s is the time spent in spiking state, t_c is current time, and t_r is the length of refractory period). In a sense, a neuron keeps a short history of its recent activity (for t_r amount of time). This can be exploited as short-term memory which can have extended uses and provide flexibility in low-power robotics applications, particularly for building efficient learning methods. Particular demonstration of how such a short-term memory (due to refractory period) can be exploited in SNN-based robotics applications is left for an extended paper. In our group, we are currently investigating in exploiting refractory period in recurrent spiking neural networks that can play a key role in processing and acting on time-series sensory signals (e.g., audio, streaming video), and manipulation of sensory guided movement (e.g., reaching, grasping).

Without loss of generality, Fig. 6 illustrates how the internal states of a neuron (in particular, refractory period) can be exploited to form a short-term memory. There are three neurons, n_1, n_2, and n_3 whose interactions would be used to demonstrate the concept. At time t_1, neuron n_1 spikes which also causes neuron n_2 to spike. Following a spike, n_2 enters a refractory period at time $t_1 + t_s$; and it remains in refractory period for t_r amount of time. During this period, neuron n_3 spikes to probe the short-term memory of n_2. If n_2 responds to input coming from n_3, then this means that n_2 is not in refractory period; otherwise, if it remains unresponsive to the probe of n_3, then it means that n_2 is in refractory period. When n_3 probes n_2 at the time t_c where $(t_1 + t_s) \leq t_c < (t_1 + t_s + t_r)$, the n_2 does not spike since it is in refractory period, and that can be considered as a logical '1' being kept in short-term memory of n_2. The content of the short-term memory will be lost once n_2 reaches the resting state again (at time $t_1 + t_s + t_r$).

Likewise, if we look at the state of neuron n_2 at time t_N, we see that it is in resting state (no spike emitted from n_1, so there is no input for n_2 to make it spike at that moment). Waiting enough time for n_2 to change its internal state in case there could be an input from n_1 (which is not the case this time), neuron n_3 spikes to probe n_2's short-term memory at time t_c where $(t_N + t_s) \leq t_c < (t_N + t_s + t_r)$. This time, n_2 responds to input from n_3 since it was in resting state, thus emit a spike, which can be considered as a logical '0' being kept in short-term memory of n_2.

Once again, the retention of the current state (and thus its corresponding logical value) is restricted by the duration of refractory period (i.e., t_r). Although it is possible to build a complex SNNs that can retain the value longer (via feedback looped network), or by extending the refractory period (which may have side-effects on performance of the computation carried over the network), we did not explore them here. Rather, we focus on the basic principles and provide a proof-of-concept example, in this paper.

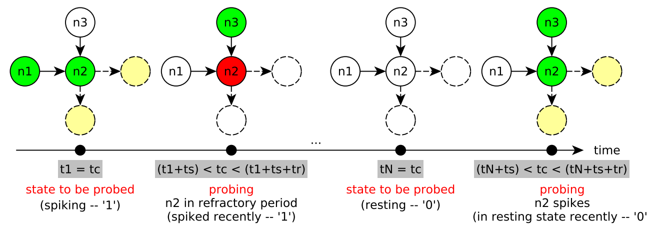


Fig. 6. Illustration of how refractory period serves as short-term memory. Neuron n_2 does not respond to probing input (from n_3) if it spiked recently (i.e., in refractory period) – this can be regarded as logical '1' being kept in short-term memory. However, neuron n_2 responds to probing input (from n_3) and spikes, if it was in resting state, recently – this can be regarded as logical '0' being kept in short-term memory.

IV. Related Work

The refractory period has been exploited in the context of associative memory, in which the network activity is a proxy to memory capacity (higher the activity means larger the capacity)[3]. When the neurons are in refractory period, the network activity reduces and makes the memory capacity smaller. However, in return, the recall ability of the network increases. The authors have modeled the refractory period and played with the threshold to improve the recall rate of the associative memories in neural networks. In contrast, we exploit refractory period itself as a way to form a memory (rather than adjusting a recall rate), as opposed to whole network acting as aassociative memory. We also exploit refractory period to support functional multiplexing in SNNs, which is the first attempt of its kind, to the best of our knowledge.

V. Conclusion

We propose two novel ways to exploit the refractory period of neurons in SNNs. First, it can be exploited to enable functional multiplexing (that is the SNN acts as different network at distinct time periods, depending on which neurons are in refractory period), and second, it can be used to build a short-term memory (based on recent spike activity, i.e., whether a neuron has spiked or not, in the recent past). Both functional multiplexing and short-term memory (based on refractory period) would be key to build efficient learning, sensory information processing, and motion planing in robotics applications (in terms of area, power and performance).

References

[1] H. Paugam-Moisy and S. M. Bohte, "Computing with spiking neuron networks," in *Handbook of Natural Computing*, 2012.

[2] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, Dec. 1943.

[3] M. Oda and H. Miyajima, "Autoassociative memory using refractory period of neurons and its on-line learning," in *ICECS 2001. 8th IEEE International Conference on Electronics, Circuits and Systems (Cat. No.01EX483)*, vol. 2, pp. 623–626 vol.2, Sep. 2001.