Wake-up radio receiver based on Spiking Neurons for detecting activation sequence

Guillaume Marthe, Claire Goursaud, Laurent Clavier

Wake-up radio receiver based on Spiking Neurons for detecting activation sequence

Guillaume Marthe
CITI, EA3720
Univ Lyon, INSA Lyon, Inria
69621 Villeurbanne, France
guillaume.marthe@insa-lyon.fr

Claire Goursaud
CITI, EA3720
Univ Lyon, INSA Lyon, Inria
69621 Villeurbanne, France
claire.goursaud@insa-lyon.fr

Laurent Clavier
CNRS, UMR 8520 - IEMN
IMT Nord Europe, Université de Lille
59000 Lille, France
laurent.clavier@imt-nord-europe.fr

***Abstract*—Energy consumption is a critical issue for the deployed nodes in the area of Internet of Things (IoT). This is the reason why many research projects focus on Wake-up Radio (WuR) receivers that permit the nodes to remain in sleep mode for as long as possible and to wake them up only if needed. However, current WuR use classic microcontrollers that are still too energy consuming. Meanwhile, Spiking Neural Networks (SNN) offer much lower power consumption. Thus, we propose to adapt those neural networks as a wake-up radio receiver in the IoT context. We aim at waking up the concerned node by recognising one or many activation sequences in a bit flow. We propose here a configuration for the neurons along with the design of appropriate sequences. We present the performances of our system and the impact of different parameters on the accuracy of the recognition system.**

***Index Terms*—Spiking Neural Network, Wake-Up Radios, Active sequence detection**

I. Introduction

Internet of Things (IoT) has become a major field of research due to its usage in a wide range of domains [1]. One of its historical challenges remains the energy consumption of the smart devices it connects. Among others, the Wake-up Radio receivers (WuR) are a solution allowing us to save energy by wakening devices on demand. Indeed, as a lot of nodes send very little information, our goal is to awaken them only if needed rather than periodically.

Wake-up receiver is a dedicated circuit used for continuous channel monitoring, which the interesting property that it consumes less power than the main processor [2]. These receivers wake up the main receiver only when a specific signal is detected, as shown in Figure 1. Then, asynchronous communication can be achieved, resulting in a huge decrease of energy waste. However, most wake-up receivers are still relying on low power microcontrollers that do perform signal recognition but consume peak powers higher than $200\mu W$ [3], making IoT nodes unable to reach their optimal energy efficiency. Thus, classical computing in a wake-up radio does not currently allow for a consumption that permits the battery to last as long as the intended lifespan of the node [4].

Meanwhile, Spiking Neural Networks (SNN) begin to emerge as a low power solution for data processing [5], image recognition [6] and to model brain activity in neurosciences [7]. This bio-inspired system can compute as fast as actual devices but consumes less. Their usage is still limited as an industrial or commercial solution, but some companies started to sell some neuromorphic architectures like TrueNorth from IBM. These architectures are also based on classical electronics. However, to the best of our knowledge, no study has considered the potential utility of spiking neural network for telecommunications transmitters.

In this study, we design and evaluate a more energy efficient wake-up radio using a spiking neural network to detect an activation sequence. We want to reproduce the usual device address recognition with this low-power architecture. To do so, we jointly design the signal to be transmitted and the architecture based on those special neurons.

The originality of this work is to consider a new generation of neurons in standard 65 nm CMOS technology with optimized energy efficiency [11] (4-fJ/Spike) as they do not rely on transistors.

The paper is organized as follows. Section II introduces the models used for simulations by presenting the system and the architecture of the network. Section III details the proposition for the behaviour of those neurons and network and explain their implementation. Performances results are presented in Section IV, for different cases. Finally, Section V concludes the paper.

II. Models

A. System Model

We consider a network in which a gateway is in charge of the N nodes in its vicinity. The objective of the paper is to define a system such that a node is awaken whenever a gateway needs to communicate with it by receiving a specific code sequence.

From the receiver's point of view, the WuR objective is to evaluate whether the receiver's signature is present in the continuous incoming stream. If so, the main receiver is awaken to communicate with the gateway.

The input signal is modelled as:

$$y = \sum_{k=-\infty}^{\infty} \sum_{i=1}^{N} \delta_{ik} c_i (t - k \cdot T) + n \tag{1}$$

where δ_{ik} is the activity indicator (equal to 1 if the code i is transmitted in the k^{th} period, and null otherwise, and such

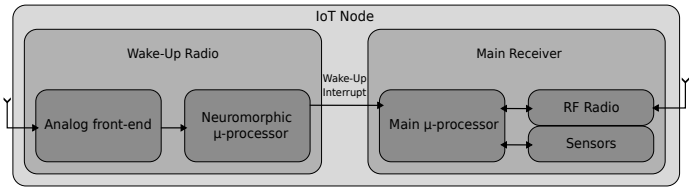

Fig. 1. Implementation structure of a WuR using a SNN.

that at most 1 code is transmitted in the k^{th} period), c_i the code to address user i, and $n \sim \mathcal{N}(0, \sigma)$ the Additive White Gaussian Noise (AWGN).

This signal can be processed by a correlator to detect whether the targeted sequence is active. The correlator is thus always running. As a consequence, we propose to use the strongly energy efficient spiking neurons to process the incoming signal.

B. SNN Model

We define a spiking neural network structure analysing the incoming bits and firing if the desired addresses is present. This kind of neural network is composed of specific neurons that mimic the human neurons. The main difference with their artificial counterparts is that spiking neurons do not transmit information in a synchronised way, but rather spike whenever their membrane potential reaches a threshold v_{th} after accumulating incoming electrical stimulus as shown on the right of Eq. 1. The value v_{th} depends on the layer of the neuron and the pattern to recognise.

After hitting the threshold, the neuron spikes and its membrane potential v decreases instantly to its resting potential v_{rest}. If it does not reach v_{th}, the membrane potential will slowly decay to a resting potential until we integrate another incoming spike. The most used model to describe the potential decay is the Leaky Integrate and Fire (LIF) model [8]:

$$\tau_m \frac{dv}{dt} = (v_{rest} - v) + R_m I_m \tag{2}$$

where τ_m is the membrane time constant, v_{rest} the resting potential of the neuron, R_m the neuron resistance and I_m the input current of the neuron.

Synapses are the links between the different neurons. They can excite (resp. inhibit) a target neuron with positive (resp. negative) weight that remains constant. They weight and propagate the incoming spikes from the output of the presynaptic neuron (the neuron just before the considered synapse) to the entrance of the postsynaptic neuron (the one just after the considered synapse).

The input of our neural network is modelled as a spike train:

$$s^{(f)}(t) = \sum_f \delta(t - t_f) \tag{3}$$

with t_f corresponding to the different spike times, and δ the Dirac distribution. It represents the flow of 1s sent at their specific time as we can see in the left box of Figure 2. We assume that the incoming signal has an On-Off Keying modulation. The main advantage of OOK is that it can be demodulated thanks to a non-coherent receiver. It thus permits to save precious energy by removing the oscillator. The received wireless signal is transformed into a spike train. When a 1 is transmitted, the received power leads to the generation of a spike. Thus, as the transmission of 1s and 0s are slotted, the absence of a spike stands for a 0.

Our desired output is a spike train fired on the timings when we get the targeted signature in the input, as depicted in the right box of Figure 2. In this example, we are looking for the signature 1011 which appears twice in the incoming pattern. Thus, we get two spikes sent to the Radio-Frequency (RF) devices triggered by our network.

III. Proposition

A. SNN Behavior

Once the wireless signal has been transformed into a spike train, we need to evaluate at the same time l successive bits, with l the number of symbols in the signature. Then, we send spikes when the signature is recognised in the received flow as can be seen in Figure 3. To do so, the incoming spikes need to be parallelized before being simultaneously processed. To begin with, our solution consists of a network composed of three layers of neurons, connected by synapses in a totally feed-forward way.

The first layer (in blue) is composed of a unique neuron. Its purpose is to demodulate the input signal into a spike train ((3)), and then send it to the second layer. In a noiseless case, the received sequence is perfectly represented. However, for a noisy channel, the demodulation leads to an erroneous stream with either missing spikes, additional spikes, or both.

The second layer (green) contains $l - 1$ neurons which resends the received spike train but with an appropriate delay. The synapses between those two layers delay the spike train from the input to the neurons in such a way that neuron i receives $s^{(f)}(t - i \cdot t_{bit})$ with t_{bit} the time between two bits in the input signal. For a l-bits pattern, the neuron of the first layer and the $l - 1$ neurons of the second layer will simultaneously provide the l last input bits received to the last layer.

Finally, the output layer (in red) can contain as many neurons as the number of signatures to be recognised (m, which will be $m = 1$ in all the following studies), where each one is connected to the preceding neurons via l synapses. Each synapse either excites or inhibits the neurons of the second layer according to whether the pattern has a 1 or a 0 at this position, as in Figure 3. If one of those neurons is meant to read a 1, it is connected to an excitatory synapse that sends a positive spike to the output neuron. Alternatively, if a neuron is meant to receive a 0 and receive a spike instead, its inhibitor synapse sends a negative spike to the output neuron in order to decrease its membrane potential to prevent the firing.

Indeed, a unique receiver can be awakened by several signatures, such as its dedicated personal one, or some collective one whenever a group is addressed. However, as the behaviour

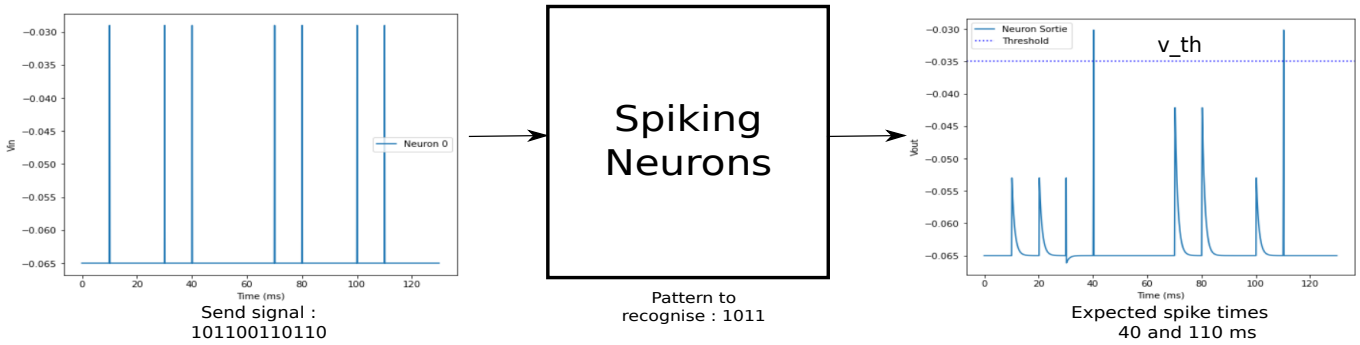

Fig. 2. Ideal input and output of the SNN.

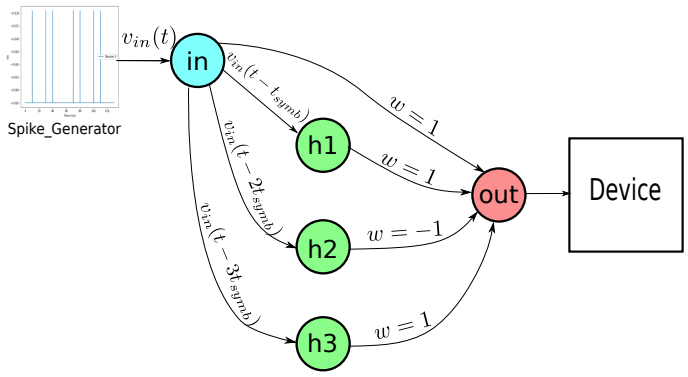

Fig. 3. Path followed by the signal in the network.

is identical for all kind of signatures, we focus in this paper on the case where a unique sequence is searched for. The respective threshold of each output neuron is set in order to fire if they receive at the same time (with a small tolerance), as much exciter spikes as the pattern contains 1s, and exactly zero spikes on the inhibitory synapses.

B. Implementation

The simulations were realised with Brian2 [9], a Python-based simulator for spiking neural networks. We created our input signal (the vector t_f introduced in (3)) with SpikeGeneratorGroup, an object for creating spike trains from the spike timings. According to the size of the signature l and the number m of desired outputs we want to recognise, we then create our neurons as described in II-B. We use the NeuronGroup function to model our $l + m$ neurons with the LIF behaviour we presented. We finally customize our synapses as either "excite" or "inhibit" according to the signatures, and connect them to the corresponding neurons. Our network thus contains $n_{neu} = l + m$ neurons and $n_{syn} = l(m + 1)$ synapses. The value l should be adapted to the network size. In this paper, we consider as an example, a network size of 1000 nodes, requiring 10 bit long sequences.

Nowadays, neurons architectures power is around $100pW$ per neuron [10]. Thus, to recognize one specific and unique sequence, our network contains 11 neurons, which means that the network power will be in the order of $1nW$. This is dramatically below the peak power of $200\mu W$ from actual wake-up receivers. Even for a 100 long sequence which would permit to generate 10^{30} different sequences, the power would be reduced by a factor of 20000.

IV. Results

A. Transmission Model

We evaluate the performances for three types of sequence activity pattern. In the first one, called *Burst* method, the sequences are sent in a continuous way, mimicking the case where there is always a node to wake up. Thus, as soon as the current sequence ends, a new one begins. Meanwhile, the second one, called the *Sporadic* method, corresponds to the case where nodes are rarely awaken. Thus, sequences are sent in a sparse way, with a silence separating two consecutive sequences, during at least the length of a sequence. In the third one, called *Synchro*, we make the assumption that the sequences are periodically scheduled, allowing the receiver to know when the sequence should start and finish if present, thanks to synchronisation.

Besides, in order to be in a realistic case, we consider an AWGN noise (characterised by its standard deviation σ), added to the input of our first neuron in, as shown in Figure 3. The noise might create errors on the generated spiking train, which can lead to FA (False Alarm) or MD (MisDetection).

The probability of MD is given by $\frac{FN}{FN+TP}$, while FA corresponds to $\frac{FP}{FP+TN}$; where FN stands for False Negatives, when a code is not detected, FP the false positives, when we have a false alarm, and TP (resp. TN) true positive (resp. negative), when we detect correctly that there is (resp. there is not) the desired pattern.

B. Initial Set-Up: unconstrained codes

As a first step, we consider that any sequence of l bits can be used as a code. We first develop the theoretical expression

of the probability of MD and FA for the Synchro method. To do so, we need to evaluate the probability that a received bit, either a 0 or a 1, will be toggled by the noise. The resting potential of our neuron is at $-65mV$ and a received spike will rise the membrane potential by $60mV$ to reach $-5mV$. The threshold will accordingly be placed between -65 and $-5mV$. In this scenario, the threshold is placed at $-35mV$ which is the middle. The noise n follow the normal centered distribution $\mathcal{N}(0, \sigma)$. The condition to flag a mistake on a bit is when $n > 30$ if the transmitted bit was a 0 and $n < -30$ if it was a 1:

$$P(E|0) = P(n > (v_{th} - v_{rest})) \tag{4}$$

$$P(E|1) = P(n < (v_{th} - v_{spike})) \tag{5}$$

As the normal distribution is a symmetrical function, $P(n > 30) = P(n < -30)$, we have $P(E|0) = P(E|1) = P(E)$. The probability to have a MisDetection is the probability that at least one bit has been flipped. Thus:

$$P(MD) = 1 - (1 - P(E))^l \tag{6}$$

Similarly, if there is a pattern that is the same than the searched pattern except for k bits, and if these k bits are flipped, then there is a False Alarm. The FA probability is thus:

$$P(FA|N) = \frac{P(FA \cap N)}{P(N)} \tag{7}$$

with

$$P(FA \cap N) = \sum_{k=1}^{l} \left(\left(\frac{1}{2}\right)^l \binom{l}{k} P(E)^k (1 - P(E))^{l-k} \right) \tag{8}$$

and

$$P(N) = 1 - \left(\frac{1}{2}\right)^l \tag{9}$$

With FA the event "The network send a false alarm" and N "The emitted signature is a negative sequence", i.e. the emitted sequence is not the one we try to recognise.

In this first set-up, we present the obtained performances with σ varying from 0 to 60 for $l = 10$ in Figure 4. We can first note that the MD seems to be more important than FA. This is due to the fact that a MD occurs when there is at least one error on the sequence. Meanwhile for a FA, an almost similar sequence needs to be sent and the differentiating bits need to be flipped, implying more constraint to fall in this case.

Besides, we can observe that performances are much better in the Synchro case than in the other scenarios. For the burst case, this is due to the sliding window that can lead to an error due to the combination of 2 consecutive sequences. As for the sporadic case, this is due to the fact that the neural network can not differentiate the absence of a spike due to a non-transmission, to the absence of a spike due to a 0 in the OOK transmitted sequence. Thus, FA cases are more likely to appear in the Sporadic case when the searched sequence contains 0s at its beginning or its end. In these cases, the generated spike arises in the middle of the transmitted sequence. If it was possible (as in the synchro mode) to focus only on spikes arising at the end of the sequences, then these unwanted alarms are removed. Nonetheless, synchronization is too energy-consuming. We thus propose in this work to force the code to begin and end with 1. By doing so, we merge a synchronisation code (with the extreme 1s with the authentication sequence).

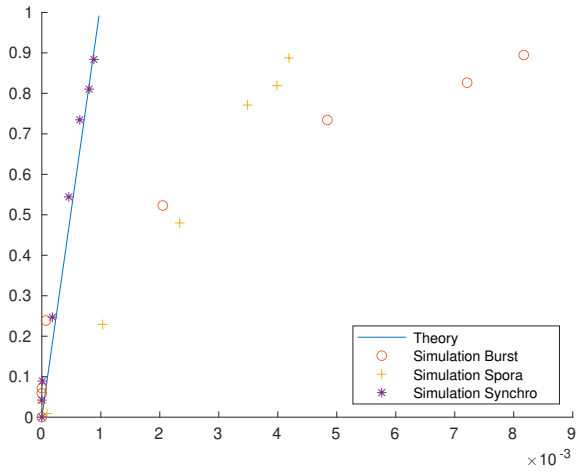

Fig. 4. Impact of the noise on FA and MD for a 10-bit long unconstrainted code

C. Enhanced Code

When considering the enhanced code, starting and ending with a 1, the MD and FA probabilities need to be adapted. If we set the first and last bit as a 1, the useful number of bits become $l - 2$. We define o as the parameter that describes whether the code is purely random ($o = 0$), or with the synchronisation part ($o = 1$). In this case, eq.8 and eq.9 are now defined as:

$$P(FA \cap N) = (1 - P(E))^{2o} \sum_{k=1}^{l-2o} \left(\left(\frac{1}{2}\right)^{l-2o} \binom{l-2o}{k} P(E)^k (1 - P(E))^{l-2o-k} \right) \tag{10}$$

$$P(N) = 1 - \left(\frac{1}{2}\right)^{l-2o} \tag{11}$$

As expected, we can see in Figure 5 that adding the 1s will lead to better performances for the same number of useful bits but worse if we compare to the same size of signature. However, this is obtained at the cost of increased signature

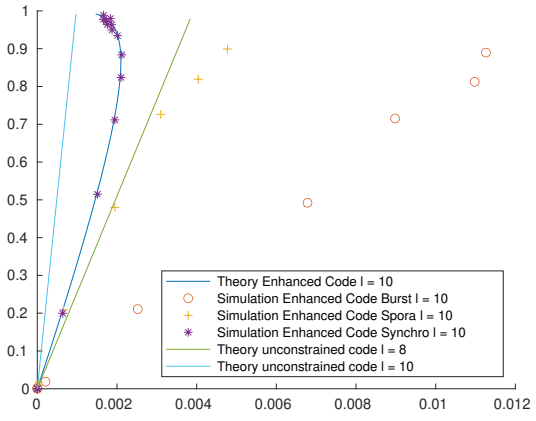

Fig. 5. Impact of the noise on FA and MD for an enhanced code

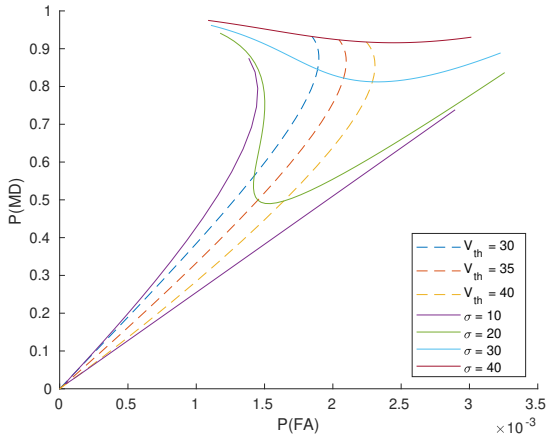

Fig. 6. Theoretical performances depending on σ and v_{th} for l = 10

length (and so its energy consumption). The interesting part of this figure is the curve for the signature being surrounded by the 1s. We can see that from a certain value of noise (30mV) we continue to get the probability of MD rising, but the one of FA starts to fall until it reaches its limit value, which is around 10^{-3}. This is due to the fact that the first and last bits of the targeted and emitted codes are always 1. Therefore, if there is an error on one of those bits, it is not possible that we get a FA and, for high values of sigma, the probability that the surrounding bits toggle rises faster than the one that the wrong bits toggles to become the targeted signature.

D. Threshold Analysis

In order to protect the synchronisation 1s, we can modify the threshold. With a threshold not being at equal distance between the rest potential and the potential reached when we receive a spike, we get two different equations for $P(E)$ as (4) and (5) will not be the same anymore. We now have to evaluate the probability to make a mistake over all combinations of signatures so the equations (6) and (8) will change too:

$$\begin{aligned} P(FA \cap N) = (1-P(E|1))^{2o} \sum_{k=1}^{l-2o} \Bigg(\left(\frac{1}{2}\right)^{l-2o} \binom{l-2o}{k} \\ \times \sum_{m=0}^{k} \left(\left(\frac{1}{2}\right)^{k} \binom{k}{m} P(E|0)^m P(E|1)^{k-m} \right) \\ \times \sum_{n=0}^{l-2o-k} \Bigg(\left(\frac{1}{2}\right)^{l-2o-k} \binom{l-2o-k}{n} \\ \times (1-P(E|0))^n (1-P(E|1))^{l-2o-k-n} \Bigg) \Bigg) \end{aligned} \tag{12}$$

$$P(MD) = 1 - \frac{1}{2^{l-2o}} \sum_{k=1}^{l-2o} \Bigg(\binom{l-2o}{k} (1-P(E|0))^k (1-P(E|1))^{l-k} \Bigg) \tag{13}$$

On Figure 6 we plotted the theoretical probabilities of FA and MD in two cases. First whe plot the curves for a specific value of sigma, while v_{th} is varying (solid lines). It allows us to define, for a specific group of parameters, the threshold that leads to the best performances. It can be observed that each of those curves has an optimal value, which we try to find. To this end, we also plotted the curves for a varying sigma while we set v_{th} (dashed lines). If our signature had included on average as much 1 than as 0, we would have expected the optimal threshold to be placed around $v_{th} = -35mV$, $30mV$ higher than the rest potential, with $v_{spike} = 60mV$ the potential added to the membrane potential when a spike is received. As we set the extremity bits to 1, there are on average more 1 than 0 in the signature, so the optimal threshold is shifted around $v_{th} = -30mV$ for $l = 10$. This phenomenon disappears if the size of the signature increases because the the average number of 1s gets closer to $\frac{l}{2}$.

E. Timing Constraint Impact

Finally, we now focus on a hardware constraint of the neurons. Neurons behavior relies on their specific time constant. In particular, the voltage decay is not instantaneous, but takes some time. If the sequence rate is too fast, the voltage might not have time to return to the resting voltage, favoring an alarm for the following sequence. In this last scenario, we evaluate the performances while varying the duration of a bit. For this study we use a third parameter, $tBit$ that represents this duration. We have conducted the following study in synchro mode with the signature surrounded by the 1s.

We can see in Figures 7, 8 and 9 that letting more time to our neurons between receiving two successive bits contribute

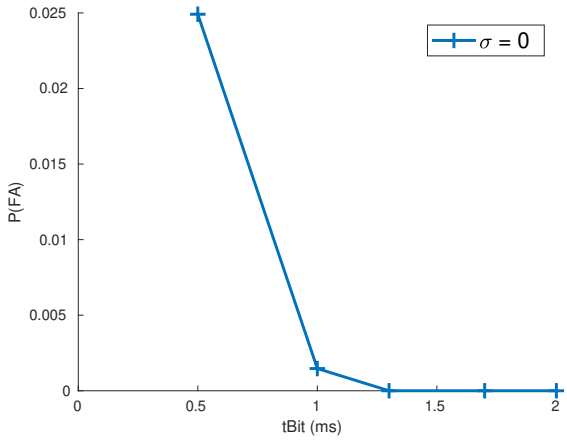

Fig. 7. Evolution of the performances on FA for different chip sending times for l = 10 and σ = 0

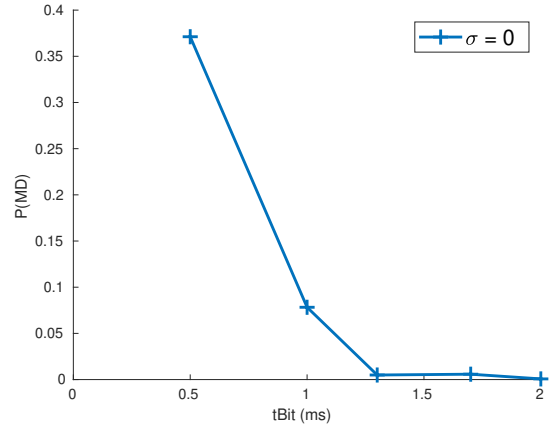

Fig. 8. Evolution of the performances on MD for different chip sending times for l = 10 and σ = 0

to improve our accuracy in term of FA and MD. Otherwise, the neuron does not have enough time for potential decay to its resting potential between 2 spikes. We can also see that for our model, there is a duration from which we have no impact on our accuracy for our set of parameters. This is due to the fact that the neuron dynamic will be too slow to decay to our resting potential between two received bits. The delay between 2 OOK symbols is thus constrained. Thus, the code rate is upper bounded. However, we can adapt τ_m, the height of the spikes, or the internal characteristics of the neuron to get faster to v_{rest} to overcome this potential limit in the code rate.

V. Conclusion

In this paper, we proposed an architecture based on spiking neurons for sequence pattern recognition, along with the appropriate code sequence definition. We have shown that adding the 1s in the sequence permitted us to get an accuracy closer to the synchro mode, without synchronisation circuit.

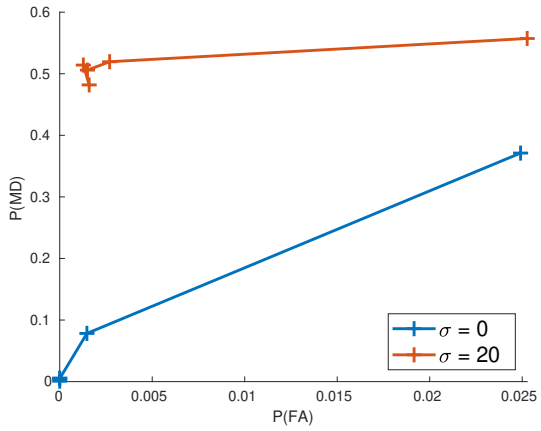

Fig. 9. Evolution of the performances for different chip sending times for l = 10 with $\sigma = 0$ and $\sigma = 20$

It implied to adapt the threshold according to our activation sequence to improve at the same time FA and MD. We also observed that we cannot receive our bits too fast because of the dynamic of our neuron, so for an optimal detection we face a limitation in our flow rate. However, this limitation is not really an issue because we only want to wake up the node, and not transmit some data.

Meanwhile, this solution is even more low-power than WUR with dedicated microcontrollers for pattern recognition, which is promising. Further work will be dedicated to study how to adapt the parameters to a faster received spike train, and the impact of the noise inside the neurons to pursue in this promising solution.

References

[1] Rishika Mehta and Jyoti Sahni and Kavita Khanna, “Internet of Things: Vision, Applications and Challenges”, Procedia Computer Sciences (2018)

[2] Ilker Demirkol Cem Ersoy Ertan Onur, “Wake-up receivers for wireless sensor networks: Benefits and challenges” , IEEE Wireless Communications (2009)

[3] N. S. Mazloum and O. Edfors, “Performance Analysis and Energy Optimization of Wake-Up Receiver Schemes for Wireless Low-Power Applications” (12/2014)

[4] Halil YETGIN et al., “A Survey of Network Lifetime Maximization Techniques in Wireless Sensor Network” (2017).

[5] Michael Hopkins, Garibaldi Pineda-García, Petruţ A. Bogdan and Steve B. Furber, “Spiking neural networks for computer vision”, Interface Focus (2018)

[6] Peter U. Diehl and Matthew Cook, “Unsupervised learning of digit recognition using spike-timing-dependent plasticity”, Frontiers in Computational Neurosciences (2015)

[7] X. Zhang, Z. Xu, C. Henriquez and S. Ferrari, “Spike-Based Indirect Training of a Spiking Neural Network-Controlled Virtual Insect”, 52nd IEEE Conference on Decision and Control (2013)

[8] Eric Hunsberger Chris, Eliasmith, “Spiking Deep Networks with LIF Neurons” (10/2015)

[9] Stimberg, M, Brette, R, Goodman, DFM. “Brian 2, an Intuitive and Efficient Neural Simulator”, eLife 8 (2019)

[10] F. DANNEVILLE et al, “A Sub-35 pW Axon-Hillock artificial neuron circuit”, Solid-State Electronics 153 (2019)

[11] I. Sourikopoulos, S. Hedayat, C. Loyez, F. Danneville, V. Hoel, E. Mercier et A. 2. Cappy, “A 4-fJ/spike artificial neuron in 65 nm CMOS technology””, Frontiers in Neuroscience, vol. 11, p. 123, March 2017.