(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 May 2018 (24.05.2018)

WIPO | PCT

(10) International Publication Number
WO 2018/091706 A1

(51) International Patent Classification:
G06N 3/04 (2006.01) *G06N 3/08* (2006.01)
G06N 3/063 (2006.01)

(21) International Application Number:
PCT/EP2017/079767

(22) International Filing Date:
20 November 2017 (20.11.2017)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
16306525.3 21 November 2016 (21.11.2016) EP
17305186.3 20 February 2017 (20.02.2017) EP

(71) Applicants: **CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE** [FR/FR]; 3, rue Michel Ange, 75016 PARIS (FR). **CONSEJO SUPERIOR DE INVESTIGACIONES CIENTIFICAS (CSIC)** [—/ES]; Pabellon del Peru, Av. Maria Luisa, 41013 SEVILLE (ES).

(72) Inventors: **THORPE, Simon**; En Bastide, 31540 SAINT FELIX LAURAGAIS (FR). **MASQUELIER, Timothée**; 16 bd Riquet, 31000 TOULOUSE (FR). **MARTIN, Jacob**; 222 Avenue de Muret, 31300 TOULOUSE (FR). **YOUSEFZADEH, Amir Reza**; Calle Pobladores, N10, Camas, 41900 SEVILLE (ES). **LINARES-BARRANCO, Bernabe**; Divina Enfermera 11, 41003 SEVILLE (ES).

(74) Agent: **PRIORI, Enrico**; Immeuble Visium, 22, avenue Aristide Briand, 94117 ARCUEIL Cedex (FR).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM,

WO 2018/091706 A1

(54) Title: METHOD, DIGITAL ELECTRONIC CIRCUIT AND SYSTEM FOR UNSUPERVISED DETECTION OF REPEATING PATTERNS IN A SERIES OF EVENTS

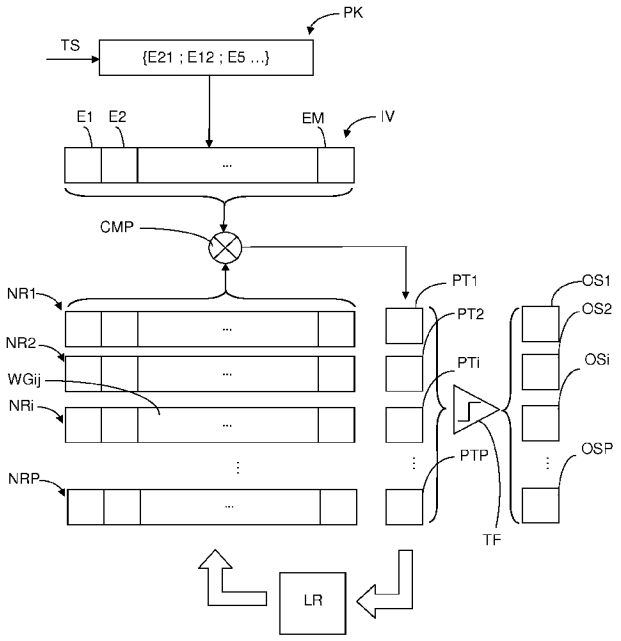

FIG.1

(57) Abstract: A method of performing unsupervised detection of repeating patterns in a series (TS) of events (E21, E12, E5...), comprising the steps of: a) Providing a plurality of neurons (NR1 - NRP), each neuron being representative of W event types; b) Acquiring an input packet (IV) comprising N successive events of the series; c) Attributing to at least some neurons a potential value (PT1 - PTP), representative of the number of common events between the input packet and the neuron; d) Modify the event types of neurons having a potential value exceeding a first threshold T_L; and e) generating a first output signal (OS1 - OSP) for all neurons having a potential value exceeding a second threshold T_F, and a second output signal, different from the first one, for all other neurons. A digital electronic circuit and system configured for carrying out such a method.

[Continued on next page]

TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

METHOD, DIGITAL ELECTRONIC CIRCUIT AND SYSTEM FOR UNSUPERVISED DETECTION OF REPEATING PATTERNS IN A SERIES OF EVENTS

DESCRIPTION

Object of the invention

The invention relates to a method, a digital circuit, a system and a computer program product for performing unsupervised detection of repeating patterns in a series of events. It belongs to the technical fields of digital electronics and of machine learning and more particularly to the sub-field of neural networks. It lends itself to several applications including – but not limited to – video stream processing (e.g. Dynamic Vision Sensors) and audio processing (e.g. artificial cochleae).

Prior art

One of the most striking features of the cerebral cortex is its ability to wire itself in order to adapt to its environment. Sensory substitution or addition experiments suggest that neurons can make sense of any kind of sensory data, presumably using the same basic mechanisms. One such mechanism is supposed to be the so called spike-timing-dependent plasticity (STDP). It has been shown that artificial neurons equipped with this mechanism can detect repeating patterns of input “spikes”, in an unsupervised manner, even when those patterns are embedded in noise. See e.g.:

- Masquelier, T., Guyonneau, R. & Thorpe, S. J. Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. PLoS One 3, e1377 (2008).
- Masquelier, T., Guyonneau, R. & Thorpe, S. J. Competitive STDP-Based Spike Pattern Learning. Neural Comput 21, 1259–1276 (2009).
- Gilson, M., Masquelier, T. & Hugues, E. STDP allows fast rate-modulated coding with Poisson-like spike trains. PLoS Comput. Biol. 7, e1002231 (2011).

This amazing ability has inspired a number of neuromorphic algorithms and architectures, used for data processing and more particularly for temporal pattern recognition.

For instance WO 2012/054109 discloses an artificial neural network having a plurality of synapses, each synapses having an adjustable weight which takes discrete values and changes in discrete steps with a probability depending on a time elapsed between a pair of spikes originating from a post-synaptic neuron circuit and a pre-synaptic neuron circuit connected to it. Operating such an artificial neural network is computationally intensive, as updating the synapses weights (essential for learning) requires multiply-accumulate operations, and multiplications are known to be the most space and power-hungry operations in the digital implementation of artificial neural networks.

WO2013119867 discloses an artificial neural network wherein synapses do not have weights – or, equivalently, have binary weights: a synapse is then either existing or non-existing. Despite this simplification, operating this artificial neural network remains computationally intensive, as it requires measuring and applying variable delays to input and output spikes.

Moreover, artificial neural networks according to the prior art generally suffer from a lack of robustness: they may fail to detect a pattern it if is slightly distorted, or if the event acquisition rates varies.

More particularly, in current STDP-based approaches, fine-tuning of the numerous parameter of the rule is required. This also adversely affects robustness. For instance, the ratio between weight reinforcement and weight depression is crucial: if too large, most weights end up saturated, leading to very active but non-selective neurons. If too low, the weights tend to decrease until the neurons do not reach their threshold anymore, which is a dead end.

The PhD thesis of Pirjo Moen “Attribute, Event Sequence, and Event Type Similarity Notions for Data Mining”, University of Helsinki, February 2000, discloses several metrics for the similarity between event sequences. These metrics are intended for application to data mining.

The invention aims at overcoming these drawbacks of the prior art by providing a method and an architecture for performing unsupervised detection of temporal patterns which is simple and economical to implement (in

terms of computing time, energy consumption and/or silicon surface), effective and robust.

According to some embodiments of the invention, these aims are achieved thanks to the following technical features:

- Input events ("spikes") are grouped into fixed-size packets. The temporal order between events of a same packet is lost, which may seem a drawback, but indeed increases robustness as it eases the detection of distorted patterns and makes the method insensitive to changes of the event rate
- Weighted or un-weighted synapses are replaced by set of binary weights. Learning only requires flipping some of these binary weights and performing sums and comparisons, thus minimizing the computational burden.
- The number of binary weights which is set to "1" for each neuron does not vary during the learning process. This avoids ending up with non-selective or non-sensible neurons.

Like other neuromorphic architectures, spiking neural networks are most often implemented by software running on general or specialized (e.g. graphical processing units, GPU) processors. In some cases, hardware implementations – based e.g. on FPGA (Field Programmable Gate Array) or other programmable circuits, or even on ASICs (Application Specific Integrated Circuit) – are used to improve performances. These hardware implementations, however, are often complex, requiring a large silicon surface, and therefore expensive. In some cases, analog or mixed-signal implementations using special devices such as memristors are used, but this approach is also complex and expensive.

According to an aspect of the invention, these drawbacks of the prior art are overcome by providing a digital hardware implementation of a spiking network which is simple and economical in terms of computing time, energy consumption and/or silicon surface, while being very effective for performing robust unsupervised detection of repeating patterns

Description of the invention

An object of the present invention is a method of performing unsupervised detection of repeating patterns in a series of events, each event of the series belonging to an event type of an M-element set of event types, the method comprising the steps of:

a) Providing a plurality of neurons, each neuron being representative of a W-element subset of the set of event types, with $1 \leq W \leq M$;

b) Acquiring an input packet comprising N successive events of the series, with $1 \leq N < M$ and preferably $1 < N < M$;

c) Attributing to at least some neurons a potential value, representative of the number of events of the input packet whose types belong to the W-element subset of the neuron;

d) for neurons having a potential value exceeding a first threshold T_L, replacing $n_{swap} \geq 1$ event types of the corresponding W-element subset, which are not common to the input packet, with event types comprised in the input packet and not currently belonging to said W-element subset; and

e) generating an output signal indicative of neurons having a potential value exceeding a second threshold TF, greater than the first threshold;

steps b) to e) being repeated a plurality of times.

In a preferred embodiment, n_{swap} and TL are set independently for different neurons. More preferred, n_{swap} and T_L are respectively decreased and increased as the potential value of the neuron increases.

According to further embodiments of the method, each neuron is implemented by a Boolean vector having M components, each component being associated to a different event type of said set, W of said components taking a first Boolean value and (M-W) of said components taking a second Boolean value.

Preferably, step a) comprises performing random initialization of the neurons.

More preferably, step b) comprises filling a M-element Boolean vector, called input vector, each element of which is associated to a different event type of said set, by setting at the first Boolean value elements associated

to event types comprised in the input packet, and at the second Boolean values elements associated to event types not comprised in the input packet.

Even more preferably, step c) comprises comparing (CMP) element-wise the input vector and the vectors implementing neurons.

Optionally, the method further comprises a step f) of generating a series of events from the output signals generated at step e), and repeating steps a) to e) by taking said series as input.

Another object of the invention is a digital (preferably integrated) electronic circuit configured for carrying out such a method, said digital electronic circuit comprising:

- an input unit, configured for receiving, on an input port of the digital electronic circuit, a series of digital signals representing respective events, each signal of the series belonging to a signal type of an M-element set of signal types, and for generating a data packet representative of N contiguous signals of the series, with $1 \leq N < M$ and preferably $1 < N < M$;
- a memory storing data defining a plurality of neurons, the data defining each one of said neurons comprising a set of binary weights representative of a subset of the set of signal types;
- a match calculating unit, connected to said input unit and said memory, configured for receiving a data packet from the input unit; for computing, for at least some of the neurons defined by the data stored by the memory, a potential value representative of the number of signals of the input packet whose types belong to the subset of the neuron; and for generating, on an output port of the digital electronic circuit, a series of output signals indicative of neurons having a potential value exceeding a threshold TF, called firing threshold; and
- a learning unit, connected to said input unit, said match calculating unit and said memory, configured for modifying, inside said memory, the set of binary weights of neurons having a potential value exceeding a threshold TL, called learning threshold, said modifying comprising swapping $n_{swap} \geq 1$ binary weights representative of signal types of the subset of the set of signal types which are not common to the input packet, with a same number of

binary weights representative of signal types comprised in the input packet and not currently belonging to said subset of the set of signal types.

Another subject of the invention is a digital electronic system comprising:

- a plurality of such digital electronic circuits, at least two of which comprise input units having filters configured to allow incoming digital signals having different values of an indicator of a neuron population comprised by each digital signal of said series, wherein the matching units of said digital electronic circuit are configured for generating a said output signal also containing an updated indicator of a neuron population; and
- a signal merger having a plurality of input ports and a single output port, the input ports of the signal merger being connected to the output ports of said digital electronic circuits and the output port of the signal merger being connected to the input ports of said digital electronic circuits and to an output port of the digital electronic system.

Yet another object of the invention is a computer program product, stored on a non-volatile computer-readable data-storage medium, comprising computer-executable instructions to cause a computer to carry out such a method.

Brief description of the drawings

Additional features and advantages of the present invention will become apparent from the subsequent description, taken in conjunction with the accompanying drawings, which show:

- Figure 1, a block diagram of an implementation of the inventive method;
- Figures 2A, 2B and 3, plots illustrating the technical results of the invention;
- figure 4, a high-level block diagram of a digital electronic circuit according to an embodiment of the invention;
- figure 5, a detailed block diagram of an input unit of the digital electronic circuit of figure 4;

- figure 6, a detailed block diagram of a match calculating unit of the digital electronic circuit of figure 4;
- figures 7A and 7B, a detailed block diagram of a learning unit of the digital electronic circuit of figure 4;
- FIGURE 7C, a block diagram of a logical unit of figure 7B; and
- figure 8, a high-level block diagram of a digital system comprising several instances of the circuit of figure 4.

Embodiments of the invention

The inventive method allows analyzing a series of "events". In actual digital implementations, an event may typically be expressed by a fixed-length binary word. For instance, if 10-bit words are used, there will be 2^{10}=1024 different event types. Each event type may represent e.g. the luminance values of a set of pixels of an image detector, a sound in an audio stream, etc. Let {E1 … EM} be the M-element set of allowable event types, and TS be a list of successive events, each event of TS belonging to an event type of the set. For the sake of simplicity, each event of TS will be identified by its event type. Therefore, it will be possible to write e.g. TS = (E21, E12, E5 …).

The list TS does not need to have timestamps, but it needs to be ordered. It is possible that some events are in fact simultaneous or quasi-simultaneous, in which case their order in the list is partially arbitrary.

A "pattern" is constituted by a group of nearby events that occur repeatedly within the series TS, even if different repetitions slightly differ, e.g. due to noise (some events can be missing, or additional events can be interleaved). The inventive method aims at detecting such patterns, otherwise stated at generating a distinguishable output signal when a pattern occurs. The detection is performed by unsupervised learning, i.e. it is not required to enter a list of the pattern to be detected: after some presentations, the algorithm learns alone to recognize repeating patterns within the input series TS.

As illustrated on figure 1, the series TS is read sequentially, until a predetermined number N (usually greater than 1) of events has been

acquired, forming a “packet” PK. The series can then be seen as a succession of packets.

In an advantageous embodiment of the invention, the acquired packet is used to fill a M-element Boolean vector, or array, called input vector IV, each element of which is associated to an event type: E1, E2… EM. Initially, all the elements of the input vector IV are set to “0”; whenever an element “Ei” is acquired, the corresponding element of the input vector takes the value “1”. At the end of the acquisition of a packet, the input vector IV will therefore contain N “ones” and M-N “zeros”. The following table 1 shows an example of an input vector for the case M=20, N=4, PK = {E3, E5, E10, E12}:

E 1	E 2	E 3	E 4	E 5	E 6	E 7	E 8	E 9	E 10	E 11	E 12	E 13	E 14	E 15	E 16	E 17	E 18	E 19	E 20
0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

Table 1

It is interesting to note that the input vector does not keep track of the order of the events within the packet. As explained above, this is an important feature of the invention, and does contribute to its robustness.

It will be understood that the size N of a packet PK has to be smaller – and preferably much smaller, i.e. by at least a factor of 10 – than the number M of event types (albeit this is not the case for the example of table 1, which is only provided as an illustration). Indeed, the inventive method does not allow taking into account the presence of several events of a same type within a same packet, therefore such a situation should be exceptional.

The detection of patterns is performed using a set of P “neurons”, each associated to a set of W unique event types, i.e. to a subset of the M-element set of all possible event types. Typically, W is greater than 1 and smaller than the number N of events in a packet and than the number M of event types; the number P of neuron may advantageously be equal to M, for reasons which will be explained furthers, but this is by no mean essential.

According to an advantageous embodiment of the invention, each neuron NR1, NR2…NRi …NRP is represented by a M-element Boolean

vector, or array, each element WGij of which is associated to an event type Ej; this is analogous to the representation of an event packet by the input vector. Neurons are initialized by randomly setting W of their elements to “1”, while the remaining (M-W) elements are at “0”. Elements of the neuron vector having a value of “1” are also called “weights”.

Whenever an event packet PK is acquired, it is compared (reference CMP on figure 1) to any one of the P neurons. Advantageously, the comparison consists in performing a logical AND between corresponding elements of the input vector and of each neuron. This allows identifying events of the input packet corresponding to the event types of each neuron (“matches”). The number of matches for a neuron determines its “potential” PT1, PT2…PTi…PTP. Table 2 illustrates an example, based on the input vector of table 1, where W=4. Matches are indicated in bold characters, and the potential values are underlined.

Event type	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12	E13	E14	E15	E16	E17	E18	E19	E20	Potential
IV	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	
NR1	0	1	**1**	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1
NR2	1	0	**1**	0	**1**	0	0	0	0	0	0	**1**	0	0	0	1	0	0	0	0	3
…																					
NRP	0	0	0	1	0	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0

Table 2

In the exemplary embodiment discussed here, a potential value is attributed to each one of the neuron, but this is not necessarily the case. The

potential is not necessarily equal to the number of matches, as in this exemplary embodiment. More generally, it may be any value representative of this number, e.g. a suitable, preferably monotone, function thereof.

If all the events occur independently and with the same probability, then the potential has a hypergeometric distribution with N draws from a population of size M containing W successes (or, equivalently, W draws from a population of size M containing N successes). If N<<M (which is possible for applications where M is large), then the probability of the potential being greater or equal than a threshold T quickly drops with T. For example with M=1024, N=64 and W=32, then the probability Pr of the potential being greater or equal than 9 is 10^{-5} approximately. That being said, for most real applications input events will not occur independently, and their frequencies may also differ.

All the neurons whose potential is greater or equal than a first, or "learning", threshold TLi (index "i", which will be omitted when this does not cause confusion, is used because the threshold may be – and preferably is – different for each neuron) are modified according to a learning rule (functional bloc LR on figure 1). The idea is that only neurons which are already sufficiently similar to the input packet should be trained in order to recognize it.

The training is performed by swapping a certain number n_{swap} of – randomly chosen – unused weights of the neurons (i.e. "ones" of the neuron vector which do not coincide with "ones" of the input vectors) with – also randomly chosen – active but unused inputs (i.e. "zeros" of the neuron vector which coincide with "ones" of the input vector). In other words, n_{swap} event types of the neuron vector, which are not common to the event types of the input packet, are exchanged with event types comprised in the input packet and not currently belonging to the neuron. For example, taking n_{swap}=1, the weight of the position #1 of neuron NR2, which is unused, is moved to position #10 of the same neuron, which corresponds to an active but unused input.

Table 3 shows the vector of the second neuron NR2 after the swapping.

NR2	0	0	**1**	0	**1**	0	0	0	0	**1**	0	**1**	0	0	0	1	0	0	0	0

Table 3

It is clear that this makes the neuron's weights more similar to the current input packet. Hence if the same packet arrives again, the potential of neuron NR2 will reach 4. This rule is similar to the biological learning rule known as spike-timing-dependent plasticity (STDP), which reinforces the connections with afferents that contributed in triggering a postsynaptic spike. Yet with STDP the weight modification is subtle but concerns all contributing afferents, while with the inventive rule the weight modification is drastic (i.e. from 0 to 1), but concerns only n_{swap} afferents. Importantly, with the inventive swapping rule, the number of non-zero weights is kept constant for each neuron.

When the potential of a neuron reaches or exceeds a second, or "firing", threshold TF, the neurons emits an output signal (it "fires"). According to an advantageous implementation of the invention, each neurons has a binary output OS1, OS2…OSi…OSP which normally takes a "0" value, and which commutes to "1" when the threshold TF is reached. Advantageously, the second ("firing") threshold TF is the same for all the neurons and is at least equal to the highest possible value for the first ("learning") threshold TL. The output signal may take any alternative format, provided that it allows identifying the neurons which have fired.

As it will be demonstrated further, with reference to figures 2A, 2B and 3, thanks to this learning rule some neurons will gradually become "selective" to any repeating input pattern, that is they will reach their second thresholds and fire when their "preferred" pattern appears, but hardly ever otherwise.

It has been found that it is useful to vary both the learning rate n_{swap}, and the first threshold TL independently for each neuron. When learning begins, a relatively low threshold is necessary to start moving the weights. On the same ground, one wants to move many weights, in order to rapidly forget the arbitrary random initial weights, and rapidly shape them according to the

input. As learning progresses, the potentials caused by the repeating patterns tend to increase. This means one can safely increase TL up to a maximum value T_{max} without missing the patterns, yet decreasing the false alarm rate. At this point it is also useful to decrease the learning rate, in order to secure the learned patterns. The learning speed n_{swap} can decrease until $n_{min}=0$ to completely stop learning, or until $n_{min}=1$ if one still wants to slowly adapt to the possibly changing input patterns.

According to a particular embodiment of the invention, the following heuristic rule is proposed to decrease n_{swap} and increase TLi:

TLi is initialized at a same value T_{min} for all "i", and then, whenever the i-th neuron NRi reaches its learning threshold TLi:

- $TLi := \min(T_{max}, PTi)$
- $n_{swap} := \max(n_{min}, n_{swap} - dn_{swap}*(PTi-TLi))$

where PTi is the potential of NRi and dn_{swap} is a parameter which tunes the decrease speed (e.g. 1/4W).

Advantageously, the second (firing) threshold is taken equal to the maximal value taken by the learning threshold: $TF=T_{max}$.

Interestingly, the outputs of the firing neurons also form a series of events, just like TS. More precisely, the firing on each one of the P neurons will constitute an event, and there will be P possible event types. Some of these events may happen simultaneously, if several neurons, trained to detect a same pattern, fire at the same time. This means that it is possible to stack multiple neuron layers using exactly the same algorithm; this is particularly easy if P=M. Neurons in higher layers will learn "meta-patterns", that is combinations of patterns. Besides, it is worth mentioning that neurons typically fire much less frequently than their inputs. This means that packets in higher layers will need more time to be filled, which corresponds to longer integration times.

The inventive method has been tested on synthetic data. Events have been generated randomly using independent Poisson processes (gray dots on figure 2A). On top of this activity, three repeating patterns have been introduced ("+", "x" and black dots on the same figure). The patterns consist in 16 nearly simultaneous events of different kinds. They repeat at irregular intervals.

The parameters used for this simulation were:

M = 1024

N = 64

W = 32

T_{min} = 7

T_{max} = TF = 9

n_{swap} = 32,

n_{min} = 1

dn_{swap} = 8

P= 1024

f = 20Hz (mean frequency of the inhomogeneous Poisson processes).

Figure 2B shows the output events, generated by neurons when their potential reaches TF. Neurons tend to be active if and only if some patterns repeat (that is, the random gray input spikes generate virtually no output activity). The most selective neurons for each pattern are those with the maximum number of "good weights" (i.e. corresponding to the pattern's 16 events), among the neurons that have reached the final threshold T_{max}. These are very stable neurons which will not forget what they have learned easily. Learning only takes a few presentations (~5). After learning most patterns are detected, and false alarms are very rare.

The firing threshold TF is equal to 9. According to the hypergeometric law, the probability of false alarm per packet is thus ~7.10^{-5}, that is one every ~42 seconds. It is worth mentioning that if the task is to detect larger patterns (e.g. 32 events instead of 16), then one can use larger thresholds, leading to much lower false alarm probabilities, e.g. TF=18 leads to one false alarm every 20 000 years. It is an advantageous feature of the invention that the false alarm probability for a given value of the firing threshold can be analytically computed.

In another simulation, a single pattern was used, and the number of its presentations during 1 second of input was varied. The probability Pr of having at least one selective neuron was computed as a function of the number of presentations and for three different values of the number of

neurons, P (1024, 2048 and 4096). A selective neuron is defined as having at least 8 good weights (chance level = 0.5), and having reached the final threshold T_{max} at t = 1s. Figure 3 shows the results. Learning is very fast: with 1024 neurons, 7 presentations are needed to reach a probability of 1, but with 4096 neurons, 4 presentations are enough.

The inventive method has been described with respect to a specific embodiment, but it is not limited to it. For instance, different implementations could be used for the input vector and for neurons.

The inventive method may be carried out using a suitably programmed computer – possibly including a graphic processing unit (GPU) to speed-up execution. A computer program comprising computer-executable instructions to cause a computer system to carry out the inventive method may be stored on a non-volatile computer-readable data-storage medium, e.g. a flash memory.

Alternatively, the inventive method may be carried out using dedicated hardware, typically a digital electronic, preferably integrated circuit, either specific (ASIC) or based on programmable logic (e.g. a Field Programmable Gate Array, FPGA).

Figure 4 shows a general block diagram of a digital electronic circuit SNN according to an embodiment of the invention. The circuit may e.g. be implemented in a FPGA and comprises four main functional blocks:

- An input unit IU which receives event-representing signals ES from an input port of the circuit and generates data packets PK representing a number N of events. The figure shows that the input unit comprises two sub-blocks, a FIFO (first-in first-out) buffer IB and a “packet maker” PM whose structure will be described in detail with reference to figure 5.

- A random-access (RAM) memory NM storing data describing the neurons. In the exemplary embodiment, each neuron is defined by several items of information: a vector of binary weights BWV, a value for the learning threshold TL and a value for the n_{swap} which controls learning speed. Memory NM is preferably a dual-port block RAM belonging to the FPGA. In the following it will be assumed that the memory NM contains M memory words, identified by $\log_2(M)$-bit addresses, each word storing the items of information

defining a neuron. Moreover, it will be assumed that each neuron has a M-bit binary weight vector. This is not essential, but it simplifies the implementation and, as discussed above and explained in detail with reference to figure 8, eases multi-layer processing.

- A matching unit, or match calculator MU which receives at its input data packets coming from the input unit and vectors of binary weights read from the memory, and uses them for computing neuron potentials and generating output signals (or "output spikes") OS, identifying neurons which have fired. Advantageously, the matching unit also modifies the learning threshold of neurons, and writes it into the appropriate location of the memory NM. The matching unit will be described in detail with reference to figure 6.

- A learning unit LU which implements the learning algorithm. This unit receives "learning events" from the matching unit, data packets from the input unit and binary weights from the memory, and uses these elements for computing updated data describing the neurons, which are written into the memory. The learning unit will be described in detail with reference to figures 7A, 7B and 7C.

The subdivision of the circuit in functional blocks may or may not correspond to a physical reality. For instance, in a FPGA or ASIC implementation, logic blocks belonging to a same functional block may or may not be adjacent to each other. In a multi-chip implementation, it is not necessary that all the element of a same functional block are realized on a same chip, or that all the logical block of a same chip belong to a same functional block. This is to say that the boundaries between functional blocks are largely conventional.

For reasons that will be understood later, in the exemplary embodiment event-representing signals ES consist of two fields: an "address" field EA, which identifies the event type, and a "population" field NPN. For instance, a 16-bit signal ES may comprise a 10-bit event field, allowing identifying 1024 different event types, and a 6-bit population field.

As mentioned above, the input unit comprises a FIFO buffer IB and a packet maker PM. The FIFO buffer is a conventional device and does not deserve a detailed description. It is necessary in most implementations because event-representing signals ES arrive asynchronously; without a buffer, a new

signal may arrive before that the packet maker has finished processing the previous one, and be lost.

A block diagram of the packet maker is illustrated on figure 5. Clock, enable and set/reset signals, as well as power supplies, are omitted for the sake of simplicity.

Event-representing signals received from the FIFO buffer are temporarily stored in an input register IR.

A controller PMCTR, which may be modeled by a finite state machine, may read from register IR the population number NPN to perform a "filtering": only signals having particular values of the NPN fields are processed. The usefulness of this filtering operation will be explained later, with reference to figure 8. The address field EA is provided to the select input SI of an M-output demultiplexer DMX, having a constant value of "1" applied to its data input DI. Each possible value of the address field EA, i.e. each possible signal type, uniquely identifies an output of the demultiplexer, and a logical "1" is transferred to said output to be stored in a corresponding memory element of an M-bit register PKR. Before receiving the first signal, all the memory elements of said register had been initialized to "0".

If all the received signals were different from each other, after receiving N signal the register PKR would contain exactly N "1" identifying said signals, and (M-N) "0". It would therefore correspond to what has been called, in the preceding description of the unsupervised detection method, an "input packet". However, in practice, there is a possibility that two or more incoming signal are of a same type; such signals only contribute to a single "1" in the packet – otherwise stated, duplicate signals have to be discarded in order to ensure that each packet contains exactly N "ones".

For this purpose, the address field of each incoming signal is also provided to the select input SI of an M-input multiplexer MX, whose output is connected to the controller PMCTR. This allows the controller to "read" a memory element just before a "1" is written into it. If the element contains a "0", a counter PMCN is incremented; if it already contains a "1", the incoming signal is considered a duplicate to be ignored, and the counter is not incremented. When the counter attains the value of "N" (the packet size, provided e.g. by an

external signal), the register PKR contains a fully formed" packet PK, which is transferred to another register, VPKR, in order to be provided to the packet maker output (in practice, this means that the matching unit receives a signal informing it that it can read the packet at the output of the packet maker, said output being directly connected to the register VPKR). The transfer Unlike PKR, whose content changes at each reception of an event-representing signal, VPKR always contains a "fully formed" signal, and its content only changes upon the completion of the next packet. This is why VPKR, and not PKR, is directly connected to the output of the packet maker.

The controller exchanges different signals with other blocks of the circuit: it receives a "Event_ready" signal from the FIFO buffer, indicating that the buffer is not full, and sends to it a "Read Event" prompting the transmission of the first available signal to the input register IR; it sends a "Spike Packet Ready" signal to the match calculator unit MU and receives from it a "Match Calculator Ready"; the "Spike Packet Ready" signal also triggers the transfer of the packet PK from the register PKR to VPKR. And, depending on the received signals, it controls the operation of the other logical blocks (demultiplexer, multiplexer, counter, register).

Figure 6 is a block diagram of the matching unit MU.

Like the packet maker (and, as it will be discussed later, the learning unit), the matching unit comprises a controller MCTRL – which, again, may be modeled by a finite state machine – controlling the operation of the logical blocks of the unit and exchanging signals with the controllers of other units. As discussed above, it exchanges "Spike Packet Ready" and "Match Calculator Ready" signals with the input unit in order to acquire packets PK from it. It also receives a "STDP Unit Ready" signal informing it when the learning unit is ready to update neurons.

The main task of the packet maker is to calculate the matching between a packet PK and the binary weight vectors BWV of all the neurons – or at least of a predetermined subset thereof. To do so, the controller MCTRL increments a counter SYAC, whose content successively takes values corresponding to all the memory (or "synaptic") addresses SYA of the binary weights vector of the different neurons. The content of the counter is used to

perform a reading access to the memory, allowing the matching unit to read the binary weight vector BWV and the learning threshold TL of the corresponding neuron, which are stored in respective registers. The current SYA address is also stored in a field of an output register OSR, connected to an output port of the circuit. Another field of the output register stores the population number NPN, which may be provided from a signal coming from outside the circuit, or be a constant value (e.g. if the filter implemented by the input unit controller ensures that only signals having a same population number are processed; in this case, the value of NPN stored in the output register OSR may be different from that of the processed signals; again, this issue will be discussed later with reference to figure 8). Moreover, the synaptic address SYA is also stored in a field of another register, called the "learning event register" LER, to which the learning unit has access.

The packet PK coming from the packet maker and the binary weight vector BWV read from the memory – both M-bit binary words – undergo a bit-wise logical AND operation, carried out by a bank of AND gates MAG operating in parallel. The outputs of these logical gates are stored in an M-bit register MR, called the matching register. It will easily be understood that a memory element of the matching register only contains a "1" if both the packet PK and the weight vector contains a "1" at the corresponding position, i.e. if they match. An adder MADD (e.g. a three-stage pipelined adder) computes the sum of all the bits of the matching register; the result is the potential value PT of the current neuron for the processed packet.

The potential PT is provided at the input of three logical blocks:

- A first digital comparer MCMP1 compares it with the firing threshold TF, provided by an external signal. If PT is greater or equal than TF, an "Output Spike Ready" flag is raised at an output of the circuit, to notify that an output signal OS may be read from the output register OSR. The output signal OS consists in the address of the firing neuron, SYA, and optionally of a population number NPN.

- A logic block MIN receives the potential PT on a first input and an externally provided value TL_{max}, representing a maximal allowed value for the learning threshold, on a second output. The lowest of the values present

at the inputs is transmitted to the output of the logic block, to be stored in a field TL' of the learning event register LER. TL' represents an updated value of the learning threshold – see equation (1) above. Alternatively, the potential PT could be stored in the learning event register LER to be transmitted to the learning unit, and the logic block MIN could belong to said learning unit.

- A second digital comparer MCMP2 compares the potential PT to the learning threshold TL, read from the memory. If PT is greater or equal than TL, a "Learning Event Ready" (LR) flag is raised to notify to the learning unit that it can read a "learning event" LEV from the learning event register LER, containing the address SYA of the neuron whose binary weight vector has to be updated from the learning unit and the updated learning threshold TL'.

Digital comparers are well known digital elements. Logic block MIN can be obtained by combining a digital comparer and a multiplexer.

Figures 7A and 7B form a block diagram of the learning unit LU.

The learning unit receives a learning triggering signal LTS, comprising the learning event LEV and the LR flag, from the matching unit. More precisely, the LR signal triggers the storing of LEV in a FIFO buffer LUB. When it is full, and therefore unable to acquire further LTS signals, the buffer generates a "FIFO full" signal – whose complement constitutes the "STDP Unit Ready" transmitted to the controller MCTRL of the matching unit; when "STDP Unit Ready" goes to zero, the matching unit stops its operation, and when "STDP Unit Ready" goes again to one it resumes it. When empty, the buffer also generates a "FIFO empty" signal, which is transmitted to a controller LCTR of the learning unit; the "FIFO empty" signal is also transmitted to the controller MCTRL of the matching unit, which triggers the "Match Calculator Ready" flag when it is informed that the learning unit has finished its job. The controller can be modeled as a Finite State Machine, and supervises the operation of the logical blocks of the learning unit. In particular, it generates a "FIFO Read_en" signal triggering the extraction of the first available data element from the buffer LUB.

The data element extracted from the buffer is a learning event LEV, which comprises a synaptic address SYA, i.e. the address of a neuron within the memory NM, and the updated learning threshold of the same neuron,

TL'. The synaptic address is transmitted to the address port of the memory, while TL' is temporarily stored in a corresponding field of an output register LOR.

Transmitting the synaptic address SYA to the memory NM triggers a reading operation of the data items, stored at said address, defining a neuron which has crossed the learning threshold. These data items are the binary weight vector BWV, the learning threshold TL itself and the swap number n_{swap}. They are temporarily stored in an input register IOR. The learning threshold is not processed by the learning unit, so its storage could also be omitted. Instead, the learning unit changes the values of the binary weight vector BWV and, preferably, of the swap number n_{swap}: this is the aim of the learning process. The updated values of the binary weight vector, BWV', and of the swap number, n'_{swap}, are stored in respective field of the output register LOR, together with TL' (see above). Upon reception, from the memory, of a "Write_enable" signal generated by the controller LCTR, the content of the output register is written into the memory location of address SYA, thus completing the learning process.

Updating n_{swap} simply consists in subtracting the decrement dn_{swap} from it, which is performed by a conventional subtracting block, SB1.

Updating the binary weight vector, on the contrary, is the most complex operation performed by the whole circuit. As explained above, it consists in swapping n_{swap} "ones" and "zeros" within the vector. To ensure that the exact number of bits is swapped, the current (i.e. not updated) value of n_{swap}, read from the IOR register, is transmitted to the controller LCTR, which stores it into a counter LUCN. The counter is decremented after each swapping operation; the learning process ends when its content reaches zero, and this triggers the generation of the "Write_enable" signal from the controller (see above).

The bit swapping block WSLB, which constitutes the bulk of the learning unit, receives at its inputs the data packet PK, generated from the packet maker PM, and the current binary weight vector BWV from the IOR register.

The binary weight vector BWV and the complemented (i.e. passed through a bank of M "NOT" gates) data packet PK undergo a bit-wise logical AND operation, carried out by a bank of AND gates LAG1 operating in parallel. The result, which is called the "ineffective weights vector" IWV is stored in a register. The IWV vector contains a "1" at each position corresponding to a "1" in the weight vector and a "0" in the data packet; otherwise stated, IWV represents active weights associated to missing events.

The complemented (i.e. passed through a bank of M "NOT" gates) binary weight vector BWV and the data packet PK undergo a bit-wise logical AND operation, carried out by another bank of AND gates LAG2 operating in parallel. The result, which is called the "ineffective spikes vector" ISV is stored in a register. The ISV vector contains a "1" at each position corresponding to a "0" in the weight vector and a "1" in the data packet; otherwise stated, ISV represents events of the packet corresponding to inactive weights.

The IWV and ISV vectors are transferred to respective circular shift registers CSR1, CSR2, and undergo a circular shift by a same (pseudo)random amount rv, determined by a random number generator RNG, controlled by the controller LCTR and having an output connected to a shift input port of each circular shift register. Several implementations of random generators are known in the art of digital electronics. According to a particular embodiment, RNG is a simple counter always counting up in each clock cycle. Because the exact time of processing each learning event LEV depends on several parameters, the value of this counter at a given time can be considered random for the purposes of the learning algorithm. Such an embodiment is advantageous because it requires particularly few hardware resources.

The circularly-shifted vectors IWV and ISV then undergo a bit-wise logical AND operation with respective mask vectors WM, SM, stored by registers. Before the first swapping, all the bits of the mask vectors are set at a logical "1", therefore IWV and ISV are unchanged by the bit-wise AND operation. References WMAG, SMAG designate the banks of logical AND gates performing the operation.

The binary vectors at the outputs of the AND gates are passed through respective logical blocks FFAB1, FFAB2 which find the first active bit of each of them. Otherwise stated, FFAB1 outputs a value which represents the position of the first logical "1" in the MIVW=ISW AND WM vector; FFAB2 does the same thing for ISV (MISV=ISW AND SM).

A possible implementation of the FFAB1 block is illustrated on figure 7C; that of FFAB2 may be identical. In the embodiment of figures 7A/7B, MIWV is a 1024-bit binary word, however figure 7C uses a 16-bit word for the sake of simplicity. More precisely, the value of IWV is taken to be "0010000010000000". This 16-bit word is used to fill a 4x4 two-dimensional array, whose lines are L0: "0010"; L1: "0000"; L2: "1000" and L3: "0000". Conversely, the columns are C0 "0010"; C1: "0000"; C2: "1000" and C3: "0000". The first active bit is the third one, i.e. the bit belonging to line L0 and columns C2. Four four-input OR gates POG1 – POG4 are used for OR-ing the bits of the four columns. The output of a logical gate is a bit taking the value "1" if the corresponding line contains at least one "1", representing an ineffective weight, and "0" otherwise. In the example, the bits corresponding to lines L0 and L2 are set to "1", those of lines L1 and L3 to "0". A first priority encoder PRE1 receives at its inputs the outputs of these logical gates (1 – 0 – 1 – 0). The inputs have different priorities: the one corresponding to line L0 has the highest priority, followed by L1, then L2 and finally L3. The output of the priority encoder is a 2-bit word identifying the highest-priority input having a "1" value; in this case, this output is "00", identifying line L0. This output is provided at the "select" input SI of a multiplexer PMUX, whose four inputs are connected to respective lines of the columns. Therefore, the four bits of the selected line L0 are present on the output of the multiplexer, to be provided as inputs to a second priority encoder PRE2, whose output is another 2-bit word identifying the position of the first "1" of the line – in this case "10", corresponding to the third position. A 4-bit number is then formed by juxtaposing the outputs of the first and second priority encoders, in this order. The value of this number ("0010", i.e. 2) corresponds to the position of the first "1" within the IWV vector – i.e. the third position.

The random number rv, determining the amount of the circular shift is then subtracted from the output of logical blocks FFAB1, FFAB2 using

subtracting blocks SB2, SB3. It is important to note that these are "circular" subtractions: for instance, if the output of FFAB1, that of FFAB2 and rv are 10-bit binary words, a negative result -|r| of the subtraction will be represented by a positive number 2^{10}-|r|-1; therefore, the result will always be comprised between 0 and 2^{10}-1=1023.

It will be easily understood that the results of the subtractions represent the rv^{th} active bits of the IWV and of the ISV vectors. Otherwise stated, the subtractions readjust the positions found by logical blocks FFAB1, FFAB2, compensating the circular shift.

It has been implicitly assumed that a positive value of rv corresponds to a shift to the right of the bits of ISW and ISV. If it corresponded to a shift to the left, the subtracting blocks SB2, SB3 should simply be replaced by adders.

A first multiplexer LMUX1 has two data inputs connected to the outputs of the subtracting blocks SB2, SB3, a select input SI connected to the controller LCTR and an output connected to an address input of a register UWR, whose content has been initialized to the current value of the weight vector BWV. A second multiplexer LMUX2 has two data inputs receiving constant binary values: "1" and "0", a select input SI connected to the controller LCTR and an output connected to a data input of register UWR. The register also has a "write_enable" input receiving a signal (reference wr_en) generated by the controller. It will be understood that, with suitable control signals from the controller, the multiplexer can be used to write a "0" at the position of the rv^{th} ineffective weight of the binary weight vector, and a "1" at the position of the rv^{th} ineffective spike. This is the first of the required swapping operations, improving the match between the weight vector and the data packet.

The output of SB2 is also applied to an address input of the register storing the WM mask, said register having a constant "0" applied at its data input. Similarly, the output of SB3 is also applied to an address input of the register storing the SM mask, said register having a constant "0" applied at its data input. As a consequence, a "0" is put in each of the masks at the position of a swapped bit.

The updated masks are then bit-wise AND-ed with the circularly shifted IWV and ISV vectors, the results being applied at the input of logical blocks FFAB1, FFAB2. Due to the bit-wise AND operations with the updated mask, the first active bits of IWV and ISV are now put to zero, therefore FFAB1 and FFAB2 find the second active bits of the original IWV/ISV vectors, and so on.

These operations are repeated n_{swap} times (the controllers decreases the content of counter LUCN by one at each iteration, and stops when a value of 0 is reached; it also stops when BWV' perfectly matches the input packet, i.e. when there are no more ineffective weights and spikes, and therefore the MIWV and MISV vectors only contain zeros), thus resulting in the flipping of a same number of "ineffective weight" and "ineffective spike" bits or, equivalently, in their swapping. This condition is checked by "OR-ing" the outputs of logical gates POG1 – POG4 using an additional four-input OR gate POG5 (see figure 7C). Gate POG5 generates a one-bit signal EM which is transmitted to the controller LCTR (not represented on figures 7A/7B for the sake of simplicity).

The updated binary weight vector BWV', contained in the UWR register, is then transmitted to the corresponding field of the output vector LOR, before being written into the memory, as already explained.

The circuit described above (figures 4 – 7C) has been implemented using a XC6SLX150T-3 Spartan-6 ® FPGA by Xilinx ®. Synthesis was performed using Verilog language and a XST synthesizer, also by Xilinx. The circuit comprised 1024 neurons and 1024 synapses (i.e. M=1024); the number N of spikes in a packet was taken equal to 64 and the number W of weights in a neuron was 32. The initial value of the learning threshold was taken to be 7, and its final value (TL_{MAX}) was 12; the firing threshold was also equal to 12. The initial value of n_{swap} was 32, with a decrement dn_{swap} of 8 down to 0 (or, in some other embodiment, to a nonzero minimal value, e.g. 1). Advantageously, all these parameters are programmable in the FPGA through a JTAG port.

A set of 64 "conventional" supervised STDP neurons was added to classify the detected pattern. A supervised STDP neuron has an extra

external input (called “supervisor”), encoded also through AER (Address Event Representation), which forces post-synaptic spikes from this neuron when its representative “category” (or feature) is present at the input. Therefore, whenever a “supervisor” spike arrives, the corresponding active synapses will be potentiated. Otherwise active synapses will be depressed. Suitable STDP neurons are described e.g. in the paper by Masquelier T, Guyonneau R, and Thorpe SJ “Spike Timing Dependent Plasticity Finds the Start of Repeating Patterns in Continuous Spike Trains” PLoS ONE 3(1): e1377 (2008).

The implementation used 25% of the slices and 21% of the block RAMs of the FPGA; interface circuits and supervised neurons consumed an additional 19% of the slices and 25% of the block RAMs. The circuit was operated at a clock frequency of 100 MHz. The learning unit was able to accept about one packet every 10.24 µs, i.e. every 1024 clock cycle. Each packet contained 64 events (spikes), therefore the theoretical event acceptance rate of the circuit was of about 6 Meps (million events per seconds); in practice, an acceptance rate of 5.3 Meps was actually achieved. By way of comparison, a software simulation running on one core of an Intel ® Core™ i5 CPU could not exceed 10 keps (thousand events per second).

Power consumption at maximum speed rate was 530 mW: 143 mW of static consumption and 387 mW of dynamic consumption, corresponding to 73 nJ for each input spike. It has been estimated that a standard-cell ASIC implementation, based for instance on Toshiba’s FFSA™ technology) would reduce consumption by 70% while improving speed by a factor of 5.

The circuit has been tested using event-representing signals generated by a simulator. Repeating patterns were generated, including at least one event among 32 different event types; signals representing 992 other event types were generated randomly. It was found that 6 presentations were sufficient to ensure reliable detection of the patterns. The detection algorithm turned out to be quite insensitive to spike loss: the rate of successful detection was only reduced by 2% (from 100% to 98%) if 40% of the spikes were deleted; the rate was still of 88% with a 50% deletion rate. It also proved reasonably insensitive to jitter: at an input spike rate of 20.48 keps (average time to make a

packet: about 3.2 ms) a jitter of up to 2 ms only reduced the detection rate by 4% (from 100% to 96%).

Figure 8 illustrates a diagram of a digital system comprising a plurality of digital electronic circuits of the type described above – in the exemplary embodiment of the figure there are three such circuits: SNN1, SNN2 and SNN3. Each of these circuits implements a population filter: PF1, PF2, PF3, only allowing in event-representing signals characterized by a particular population number. On the figure, the filters are represented as separate logic blocks but, as explained above, they may be implemented by the input unit controllers. In particular, the filters PF1 and PF2 of circuits SNN1 and SNN2 only allow in signals with population number NPN=1, and filter PF3 of circuit SNN3 only allows in signals with population number NPN=2.

Optionally an additional circuit SUP implementing a different, supervised, artificial neural network (of any known type) may also be present. This circuits implements a population filter PF4 which only allows in signals with population number NPN=3.

The output ports of circuits SNN1, SNN2, SNN3 and SUP are connected to respective input ports of a digital signal merger SMG, having a single output which is connected to the inputs of the circuits, thus implementing a data loop. An event generator EG (for instance, a dynamic vision sensor, or DVS) is also connected to a respective input of the merger through an input interface circuit IIC (for instance, an Address-Event-Representation – AER – interface). An output interface circuit OIC (for instance, another AER interface) is connected to the output of the merger. The merger SMG provides arbitration to manage the collisions between output signals of circuits "firing" almost simultaneously. The output interface circuit has a population filter allowing in signals with population number NPN=4.

The input interface generates event-representing signals ES with a population number of 1; therefore they are processed by circuits SNN1 and SNN2. These circuits have a matching unit configured to generate output signals having a population number of 2; this way, these output signals constitute events suitable to be processed by circuit SNN3. In turn, this circuit produces output signals with population number of 3, suitable to be processed

by the supervised artificial neural network SUP. And the output signals of this latter circuit have a population number of 4, in order to be accepted by the output interface. Overall, the system implements a multi-layer neural network wherein circuits SNN1 and SNN2 implement a first layer, circuit SNN3 a second layer and SUP a third layer. The first and second layers perform unsupervised detection of recurring pattern, while the third (supervised) layer associates each detected pattern to a predetermined class.

Two separate circuits are advantageously used to implement the first layer to assist in the detection of patterns which are split between two successive packets. For this purpose, circuit SNN2 ignores the first N/2 events with population number NPN=1; this ensures that the first packet generated by its input unit comprises the last N/2 events of the first packet of the circuit SNN1 and the first N/2 events of its second pattern, and so on. This way, if a pattern happens to be split between two successive packets of one circuit (a situation which might result in a missed detection), it will fall right in the middle of a packet of the other circuit. Due to duplicate events, after a while the synchronization between circuits SNN1 and SNN2 risks being lost and a reset is necessary. The same approach may be used for other layers.

A general controller GCTR supervises the operation of the system. For instance, it may set the values of different parameters of the circuits of the systems (e.g. TL, TF, n_{swap}, N, the population numbers of the filters…; in figures 5 – 8, all these parameters are represented by signal coming from outside the units), perform different tests etc.

The system of figure 8 may be implemented in a single FPGA (or another kind of programmable device), or ASIC, or in the form of a multi-chip system.

The invention has been described with reference to a specific embodiment, but it is not limited to it.

For instance, the implementation of one or some logic blocks may be changed without changing that of the other blocks – except, if necessary, for ensuring the compatibility of the data formats. The disclosed implementations, however, are believed to realize an optimal trade-off between complexity, speed and power consumption.

In particular, in the swapping block WSLB of the learning unit, n_{swap} random numbers could be generated successively in order to swap non-successive ineffective weight and ineffective spike bits. This implementation, however, would be much slower.

The data representations may also be changed to a certain extent. For instance, in the exemplary embodiment; the binary weight vector of each neuron is represented by a single M-bit binary word. In an alternative representation, it could be represented by W binary words of $\log_2 M$ bits, each representing the address of a bit set to 1. If M=1024 and W=32, this representation requires 320 bits per neuron, instead of 1024 resulting in a considerable saving in memory space. This implementation, however, is less flexible and a conversion to the “explicit” M-bit representation is required in order to be able to perform the bitwise AND operation in the matching unit and in the learning unit.

The number W of weights is not necessarily the same for all the neurons; similarly, thresholds TL and TF may be different from a neuron to another. If different neuron uses different values for TF, the matching unit will have to be slightly modified in order to include an additional register storing the TF value for the currently processed neuron.

Emphasis has been put on FPGA implementations, but this is not limiting. Any other hardware implementation, including multi-chip boards, ASIC, semi-custom IC fall within the scope of the invention.

CLAIMS

1. A method of performing unsupervised detection of repeating patterns in a series (TS) of events (E21, E12, E5 ...), each event of the series belonging to an event type of an M-element set of event types, the method comprising the steps of:

a) Providing a plurality of neurons (NR1 – NRP), each neuron being representative of a W-element subset of the set of event types, with $1 \leq W \leq M$;

b) Acquiring an input packet comprising N successive events of the series, with $1 \leq N < M$;

c) Attributing to at least some neurons a potential value (PT1 – PTP), representative of the number of events of the input packet whose types belong to the W-element subset of the neuron;

d) for neurons having a potential value exceeding a first threshold T_L, replacing $n_{swap} \geq 1$ event types of the corresponding W-element subset, which do not coincide with the input packet, with event types comprised in the input packet and not currently belonging to said W-element subset; and

e) generating an output signal (OS1 – OSP) indicative of neurons having a potential value exceeding a second threshold TF, greater than the first threshold;

steps b) to e) being repeated a plurality of times;

the method being carried out by a computer or by dedicated hardware.

2. The method of claim 1 wherein n_{swap} and TL are set independently for different neurons.

3. The method of claim 2 wherein n_{swap} and T_L are respectively decreased and increased as the potential value of the neuron increases.

4. The method of any of the preceding claims wherein each neuron is implemented by a Boolean vector having M components (WGij), each

component being associated to a different event type of said set, W of said components taking a first Boolean value and (M-W) of said components taking a second Boolean value.

5. The method of claim 4 wherein step a) comprises performing random initialization of the neurons.

6. The method of any of claims 4 or 5 wherein step b) comprises filling a M-element Boolean vector, called input vector (IV), each element of which is associated to a different event type of said set, by setting at the first Boolean value elements associated to event types comprised in the input packet, and at the second Boolean values elements associated to event types not comprised in the input packet.

7. The method of claim 6 wherein step c) comprises comparing (CMP) element-wise the input vector and the vectors implementing neurons.

8. The method of any of the preceding claims further comprising a step of

f) generating a series of events from the output signals generated at step e), and repeating steps a) to e) by taking said series as input.

9. The method of claim 8 wherein the number P of neuron is equal to the number M of event types.

10. A digital electronic circuit configured for carrying out a method according to any of the preceding claims.

11. The digital electronic circuit (SNN) of claim 10, comprising:

- an input unit, (IU) configured for receiving, on an input port of the digital electronic circuit, a series of digital signals (ES) representing respective events, each signal of the series belonging to a signal type of an M-

element set of signal types, and for generating a data packet (PK) representative of N contiguous signals of the series, with $1 \leq N < M$;

- a memory (NM) storing data defining a plurality of neurons, the data defining each one of said neurons comprising a set of binary weights (BWV) representative of a subset of the set of signal types,;
- a match calculating unit (MCU), connected to said input unit and said memory, configured for receiving a data packet (PK) from the input unit; for computing, for at least some of the neurons defined by the data stored by the memory (NM), a potential value (PT) representative of the number of signals of the input packet whose types belong to the subset of the neuron; and for generating, on an output port of the digital electronic circuit, a series of output signals (OS) indicative of neurons having a potential value exceeding a threshold TF, called firing threshold; and
- a learning unit (LU), connected to said input unit, said match calculating unit and said memory, configured for modifying, inside said memory, the set of binary weights of neurons having a potential value exceeding a threshold TL, called learning threshold, said modifying comprising swapping $n_{swap} \geq 1$ binary weights representative of signal types of the subset of the set of signal types which do not coincide with the input packet, with a same number of binary weights representative of signal types comprised in the input packet and not currently belonging to said subset of the set of signal types.

12. The digital electronic circuit of claim 11, wherein the data stored by said memory and defining each one of said neurons also comprise values of TL and n_{swap}.

13. The digital electronic circuit of claim 12, wherein at least one of said matching unit and of said learning unit is further configured for modifying the values of TL and n_{swap} stored by the memory by decreasing TL and increasing n_{swap} as the potential value of the corresponding neuron increases.

14. The digital electronic circuit of any of claims 11 to 13, wherein the data stored by said memory and defining each one of said neurons also comprise a value of TF.

15. The digital electronic circuit of any of claims 11 to 14, wherein the subsets of the set of signal types associated to all neurons have a same number (W) of elements.

16. The digital electronic circuit of any of claims 11 to 15, wherein the set of binary weights (BWV) of each neuron, representative of said subset of the set of signal types, is implemented by a Boolean vector stored by said memory, said vector having M components, each component being associated to a different signal type of said M-element set of signal types, the components associated with the elements of said subset taking a first Boolean value and said the remaining components taking a second Boolean value different from the first one.

17. The digital electronic circuit of any of claims 11 to 16, wherein the input unit comprises a demultiplexer (DMX) having a data input (DI), a select input (SI) and M outputs, and an M-bit register (PKR) having M memory elements, each having a data input connected to a respective output of the multiplexer, the demultiplexer being configured for receiving on its data input a first constant binary value and on its select input a said digital signal (ES), or a portion thereof (EA), belonging one signal type of said M-element set of signal types, whereby said first constant Boolean value is stored in one of said M memory elements associated with said signal type, the M memory elements of the register being initialized at a second constant Boolean value different from the first one.

18. The digital electronic circuit of claim 17 further comprising a counter (PMCN) and a controller (PMCTR), wherein the controller is configured for comparing the content of the M-bit register (PKR) to the digital signal present on its select input in order to determine the content of the memory

element associated to the signal type to which said digital signal (ES) belongs; for incrementing the counter if said content has said second constant Boolean value and for providing the content of the M-bit register, constituting said data packet (PK), to the match calculating unit when the counter indicates that digital signals belonging to N different signal types have been received.

19. The digital electronic circuit of claim 18 when depending from claim 16, wherein said match calculating unit comprises a bank of logical AND gates (MAG), a binary adder (MADD), a first (MCMP1) digital comparator and a controller (MCTRL), wherein the controller is configured for:

- successively reading from the memory the sets of binary weights (BWV) of a plurality of said neurons;
- using said bank of logical AND gates (MAG) for computing M-bit matching binary words by performing a bit-wise logical AND operation between each of said sets of binary weights and the data packet received from the input unit;
- using said binary adder (MADD) for computing neuron potentials (PT) by adding the bits of each matching binary word; and
- using said first digital comparator (MCMP1) for performing a comparison of each neuron potentials to said firing threshold TF and, depending on the result of said comparison, generating said output signal (OS).

20. The digital electronic circuit of claim 19 wherein said match calculating unit further comprises a second digital comparator (MCMP2), and its controller (MCTRL) is further configured for:

- using said second digital comparator (MCMP2) for performing a comparison of each neuron potentials to said learning threshold TL and, depending on the result of said comparison, generating a learning triggering signal (LTS) indicative of the neuron and transmitting it to the learning unit.

21. The digital electronic circuit of any of claims 19 and 20, wherein said match calculating unit also comprises a digital block (MIN) for

selecting the smallest of said potential value and a final value of said learning threshold and providing it to the memory as a new value (TL') of TL for the corresponding neuron.

22. The digital electronic circuit of any of claims 17 to 21, when dependent from claim 16, wherein the learning unit comprises:

- a first bank of logical AND gates (LAG1) configured for receiving at their inputs the set of binary weights (BWV) for a neuron and an inverted version of a data packet generated by the input unit, the output signals of said first bank of logical AND gates forming a first binary word (IWV) called an ineffective weights vector;
- a second bank of logical AND gates (LAG2) configured for receiving at their inputs an inverted version of said set of binary weights (BWV) and said data packet (PK), the output signals of said first bank of logical AND gates forming a second binary word (ISV) called an ineffective spikes vector;
- a first (CSR1) and second (CSR2) circular shift registers for storing said ineffective weights vector and said ineffective spikes vector;
- a random number generator (RNG), connected to the first and second circular shift registers to cause the content of said registers to undergo a circular shift of a same, random amount (rv); and
- a logical block (WSLB) configured for:
  - determining the positions of the n_{swaps} first bits of the content of the first circular shift register having a first binary value, then readjusting the determined position to compensate the random amount (rv) of said circular shift;
  - determining the positions of the n_{swaps} first bits of the content of the second circular shift register having said first binary value, then readjusting the determined position to compensate the random amount (rv) of said circular shift;
  - storing a second binary value, different from said first binary value, in a memory element of a temporary register (UWR) having, within said register, said first readjusted positions; storing the first binary value in the memory elements of the temporary register having, within

said register, said second readjusted positions; and keeping the content of the remaining memory elements of the temporary register unchanged;

the learning unit being further configured for providing the content of the temporary register to the memory as said updated set of binary weights (BWV') for the corresponding neuron.

23. The digital electronic circuit of any of claims 11 to 22, wherein each digital signal of said series (ES) also comprises an indicator of a neuron population (NPN) and wherein said input unit also comprises a filter (PF1 – PF3) configured to allow or reject incoming digital signals depending on a value of the corresponding indicator of a neuron population.

24. A digital electronic system comprising:

- a plurality of digital electronic circuits (SNN1 – SNN3) according to claim 23, at least two of which comprise input units having filters (PF1 – PF3) configured to allow incoming digital signals having different values of said indicator of a neuron population (NPN), wherein the matching units of said digital electronic circuit are configured for generating a said output signal also containing an updated indicator of a neuron population; and
- a signal merger (SMG) having a plurality of input ports and a single output port, the input ports of the signal merger being connected to the output ports of said digital electronic circuits and the output port of the signal merger being connected to the input ports of said digital electronic circuits and to an output port of the digital electronic system.

25. A digital electronic system according to claim 24, further comprising a digital circuit implementing a supervised artificial neural network (SUP) having an input port and an output port, the input port of said circuit being connected to the output port of the signal merger through a filter (PF4) configured to allow or reject incoming digital signals depending on a value of the corresponding indicator of a neuron population, the output port of said circuit being connected to an input port of the signal merger.

26. A digital electronic circuit or system according to any of claims 11 to 25, implemented by a FPGA.

27. A computer program product, stored on a non-volatile computer-readable data-storage medium, comprising computer-executable instructions to cause a computer to carry out a method according to any of claims 1 to 9.

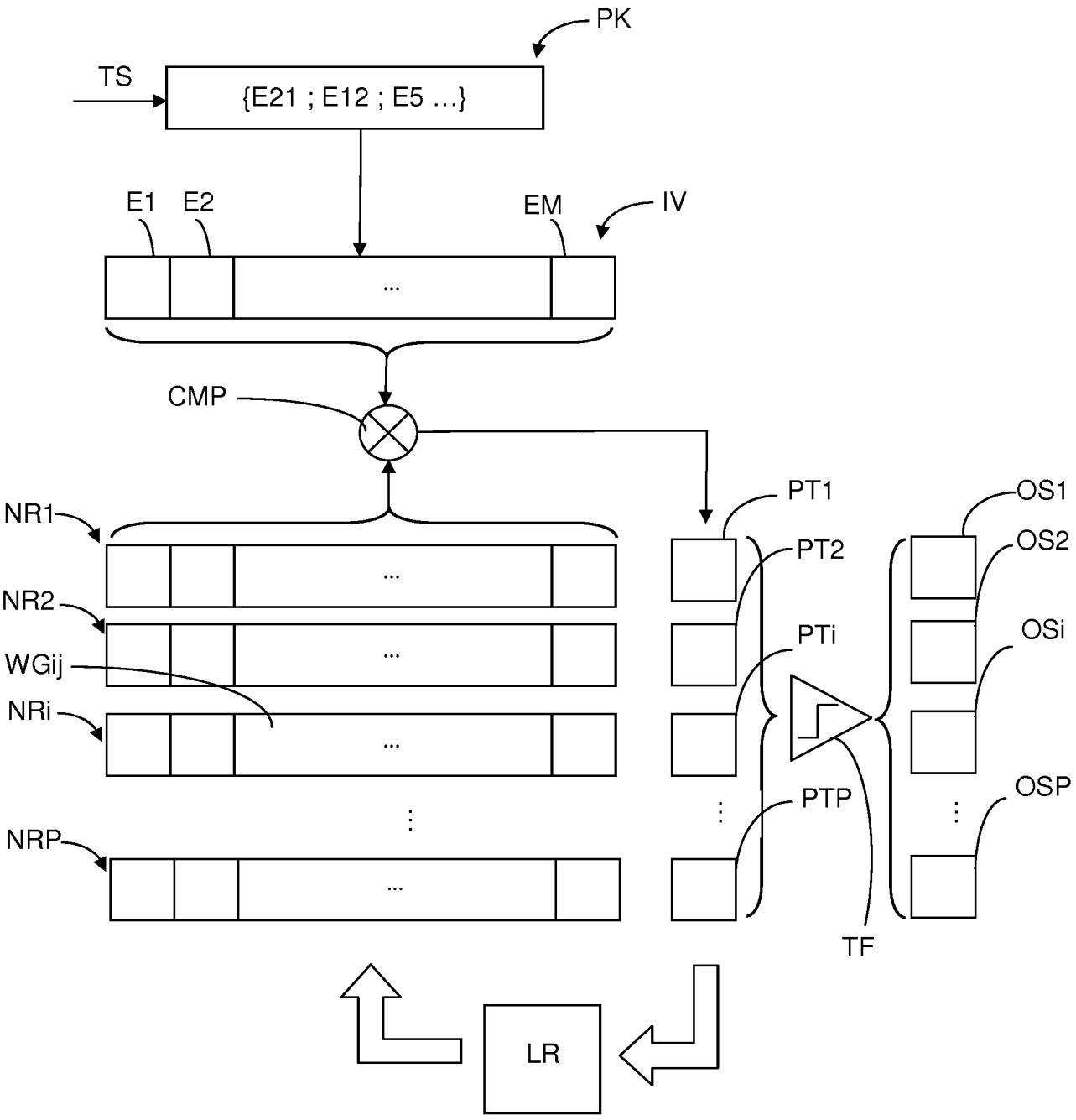
PK
TS
{E21 ; E12 ; E5 ...}
E1
E2
EM
IV
CMP
NR1
NR2
WGij
NRi
NRP
PT1
PT2
PTi
PTP
OS1
OS2
OSi
OSP
TF
LR

FIG.1

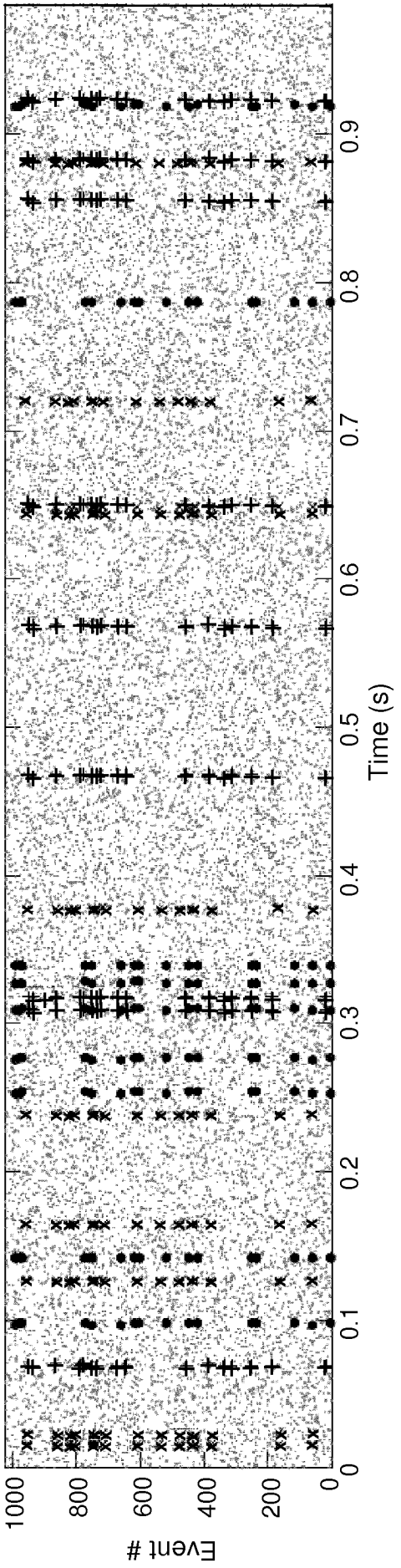
Event #
1000
800
600
400
200
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Time (s)

FIG.2A

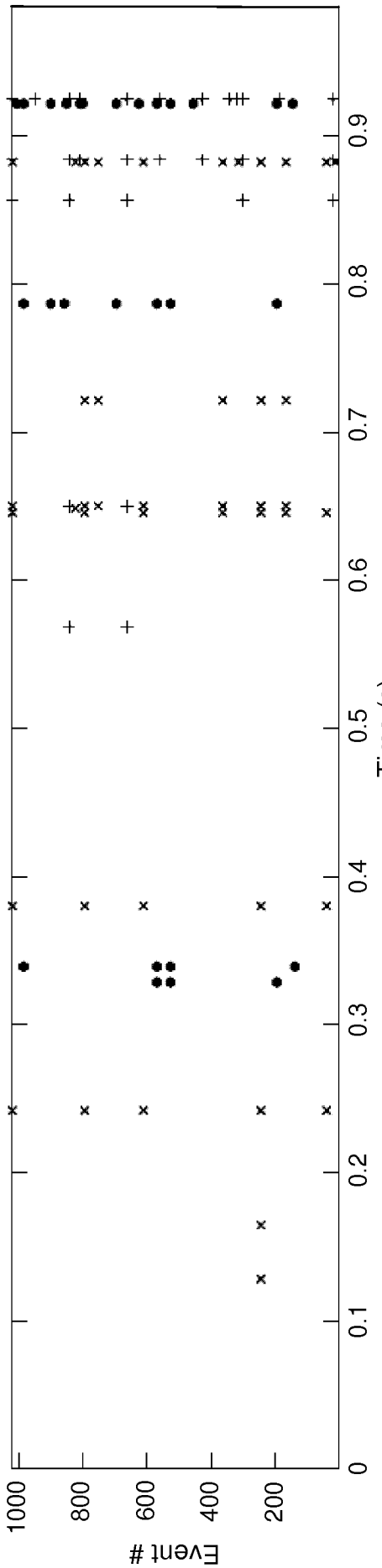
Event #
1000
800
600
400
200
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Time (s)

FIG.2B

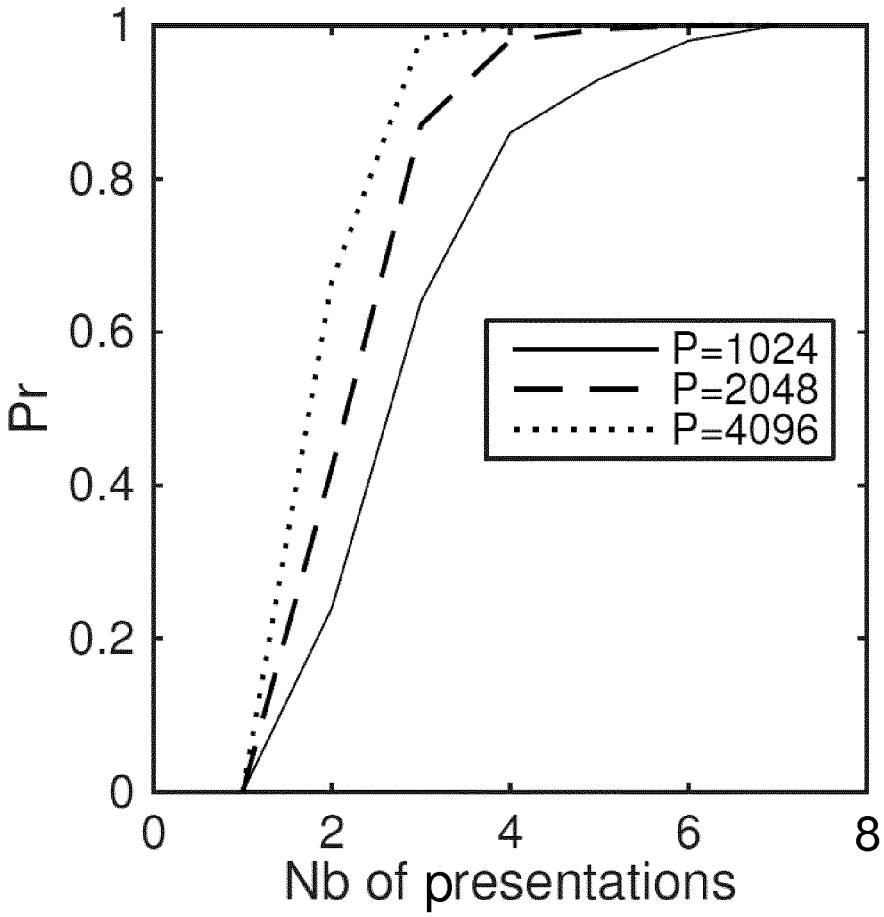

FIG.3

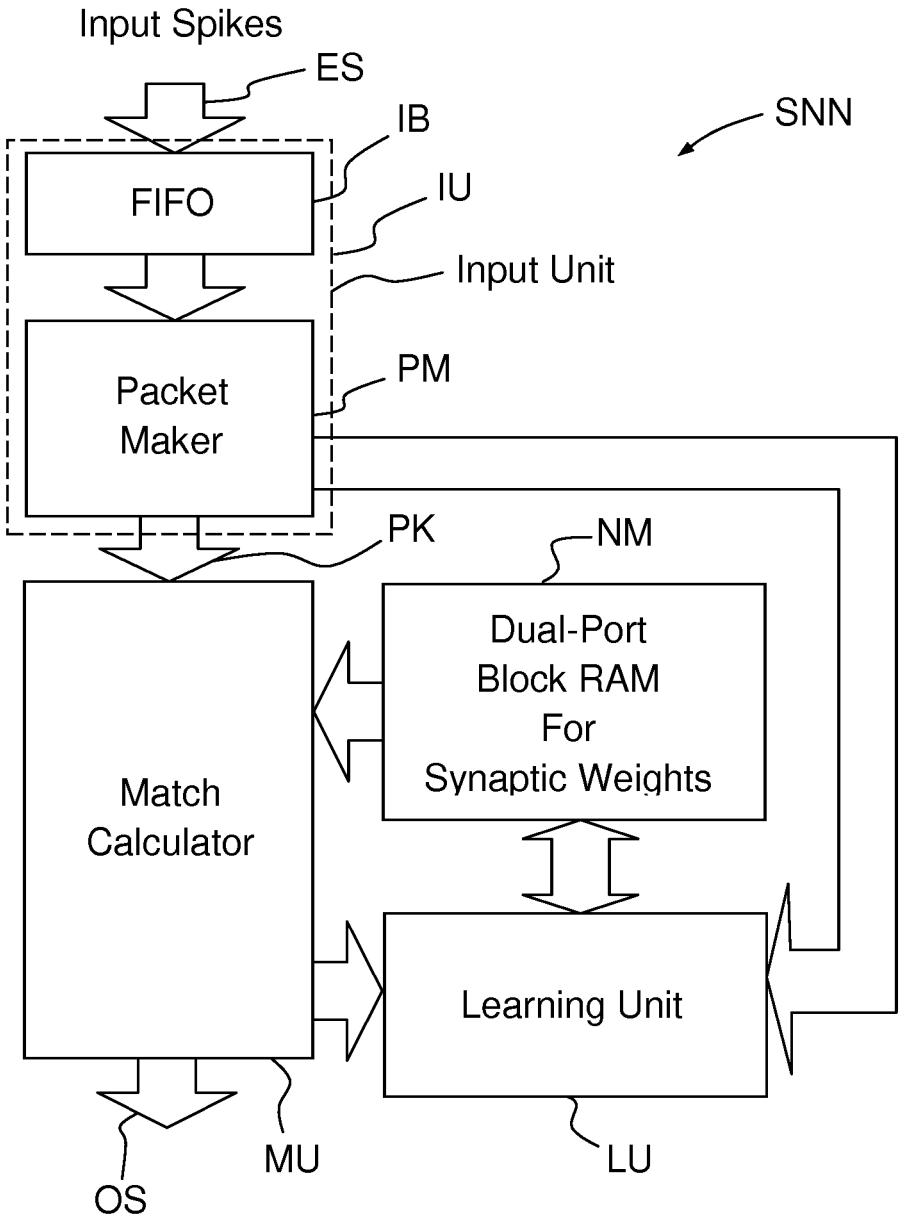
Input Spikes
ES
IB
SNN
FIFO
IU
Input Unit
Packet
Maker
PM
PK
NM
Dual-Port
Block RAM
For
Synaptic Weights
Match
Calculator
Learning Unit
OS
MU
LU

FIG.4

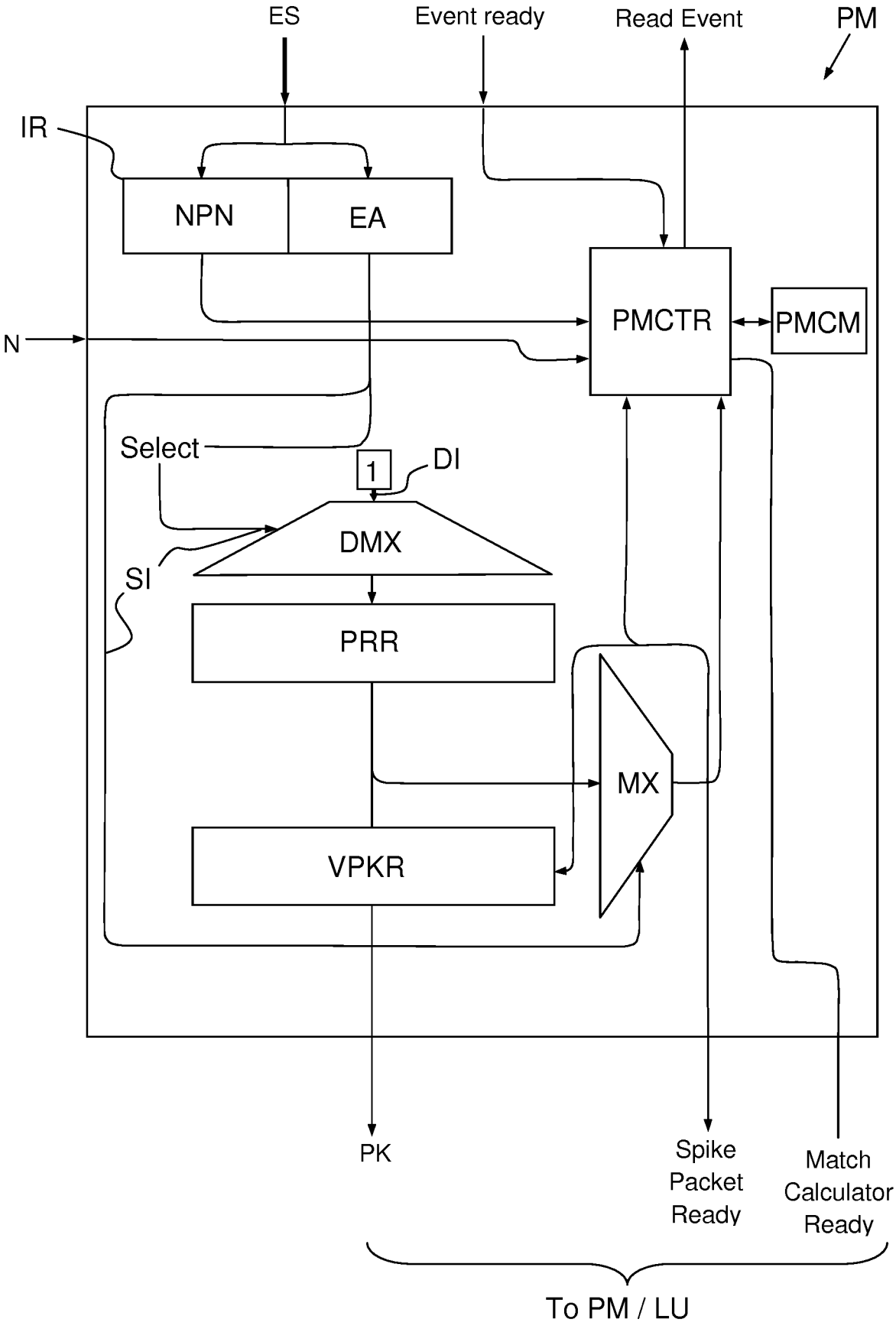
ES
Event ready
Read Event
PM
IR
NPN
EA
PMCTR
PMCM
N
Select
1
DI
DMX
SI
PRR
MX
VPKR
PK
Spike Packet Ready
Match Calculator Ready
To PM / LU

FIG.5

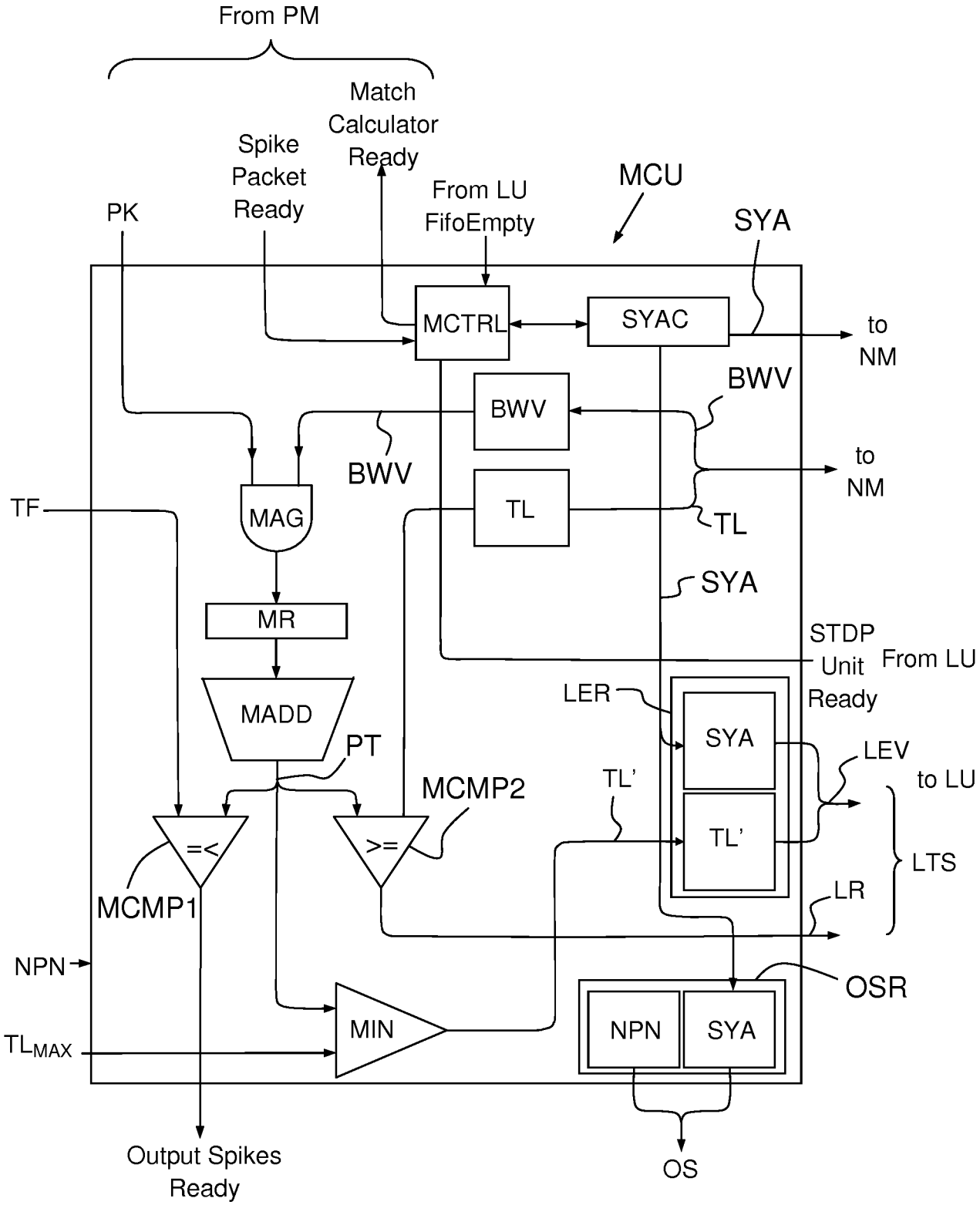
From PM
Match Calculator Ready
Spike Packet Ready
From LU FifoEmpty
MCU
PK
SYA
MCTRL
SYAC
to NM
BWV
BWV
BWV
to NM
TF
MAG
TL
TL
SYA
MR
STDP Unit
From LU Ready
MADD
LER
SYA
LEV
PT
MCMP2
TL'
to LU
=<
>=
TL'
LTS
MCMP1
LR
NPN
OSR
MIN
NPN
SYA
TL_MAX
Output Spikes Ready
OS

FIG.6

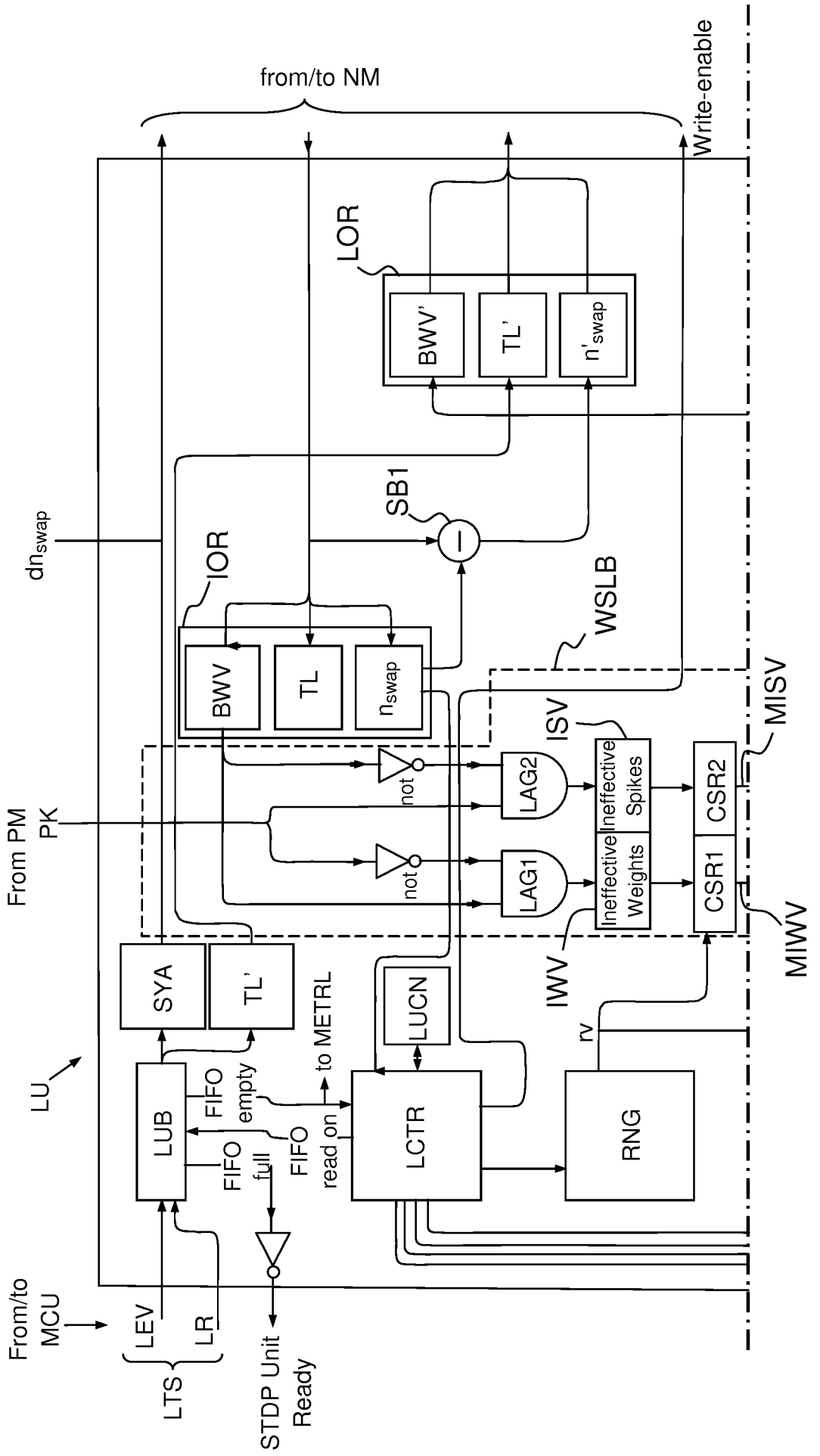
from/to NM
Write-enable
LOR
BWV'
TL'
n'swap
SB1
dnswap
IOR
BWV
TL
nswap
WSLB
ISV
MISV
From PM
PK
not
LAG2
LAG1
Ineffective Spikes
Ineffective Weights
CSR2
CSR1
MIWV
IWV
rv
SYA
TL'
LUCN
to METRL
LU
LUB
FIFO empty
FIFO full
FIFO read on
LCTR
RNG
From/to MCU
LEV
LR
LTS
STDP Unit Ready

FIG.7A

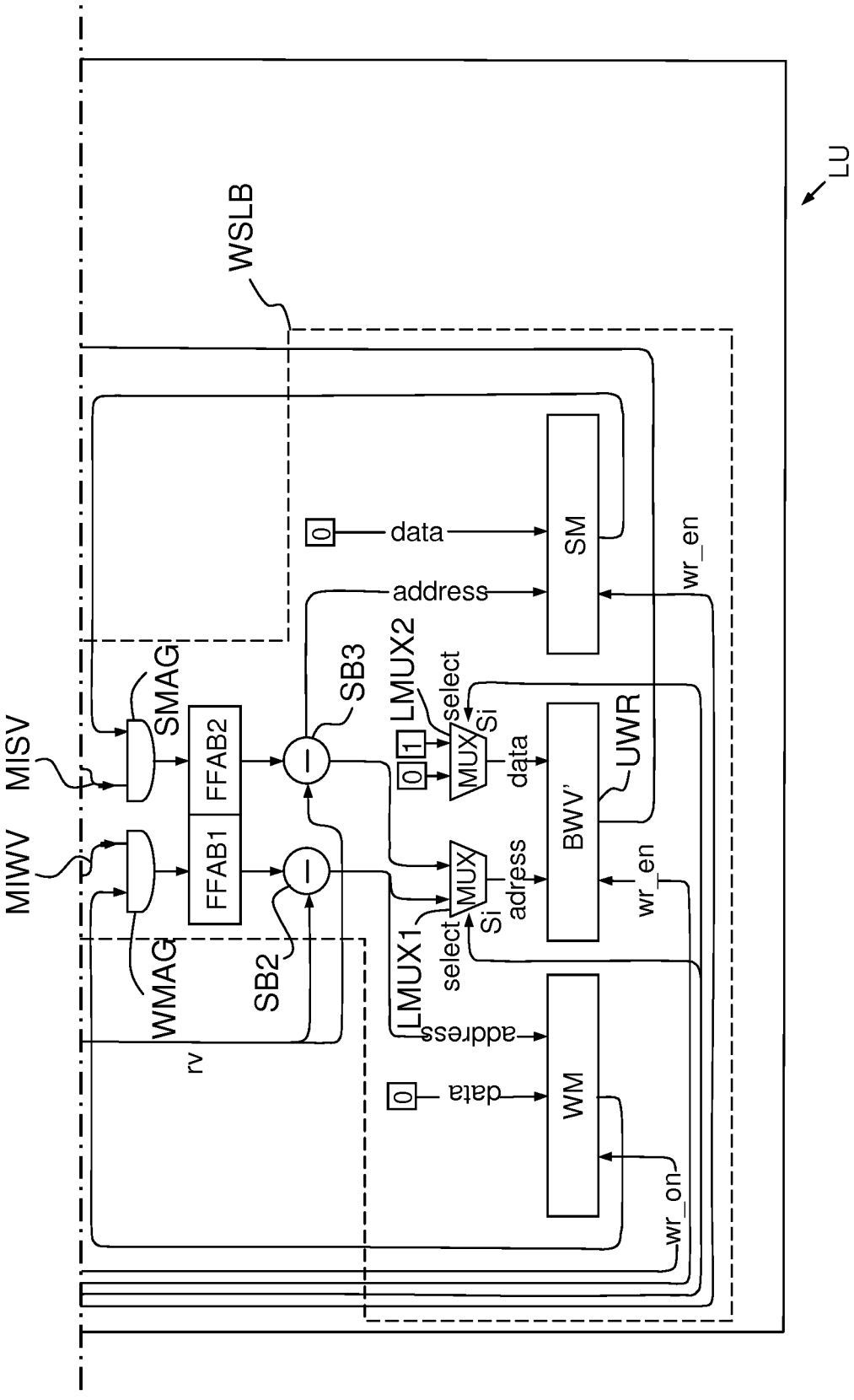
LU
WSLB
SM
data
address
wr_en
SB3
LMUX2
select
MUX
Si
data
UWR
BWV'
adress
wr_en
LMUX1
select
MUX
Si
address
WM
data
wr_on
SMAG
MISV
FFAB2
FFAB1
MIWV
WMAG
SB2
rv

FIG.7B

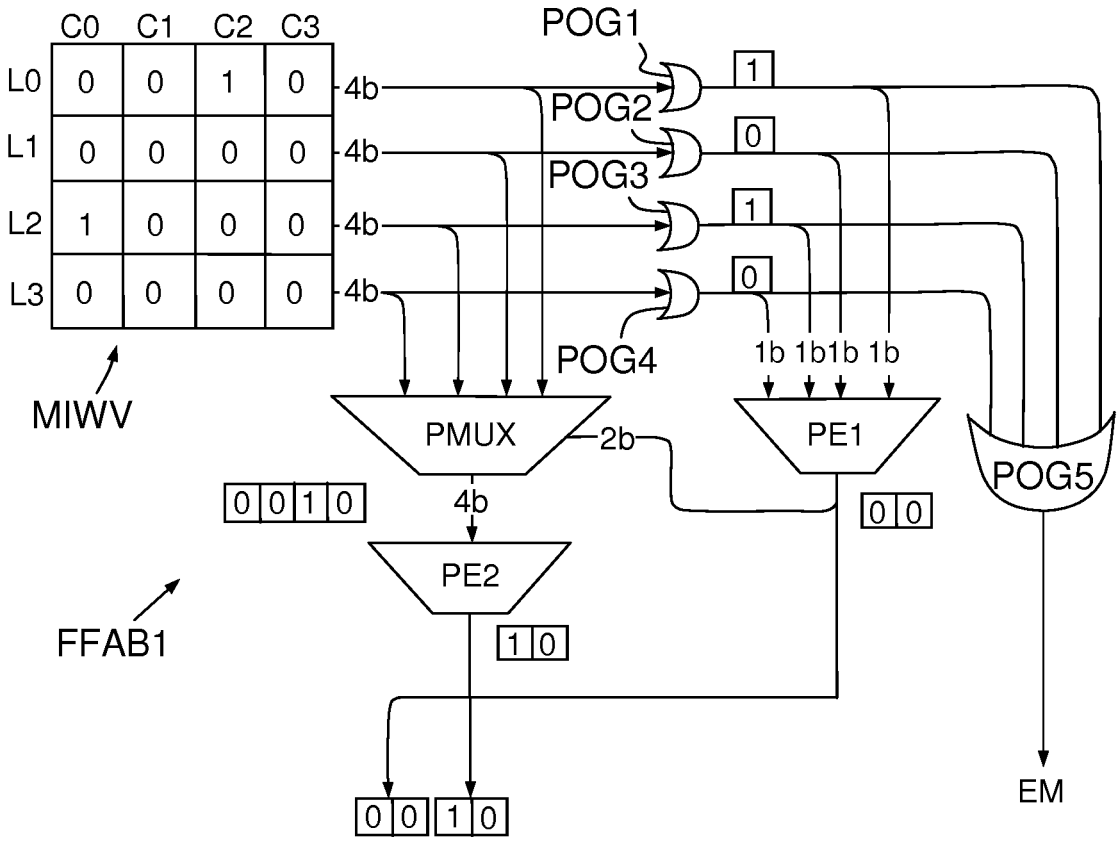
C0 C1 C2 C3
L0 0 0 1 0
L1 0 0 0 0
L2 1 0 0 0
L3 0 0 0 0
4b
4b
4b
4b
MIWV
POG1
POG2
POG3
POG4
POG5
1
0
1
0
1b 1b1b 1b
PMUX
2b
PE1
4b
PE2
0 0 1 0
0 0
1 0
0 0 1 0
FFAB1
EM

FIG.7C

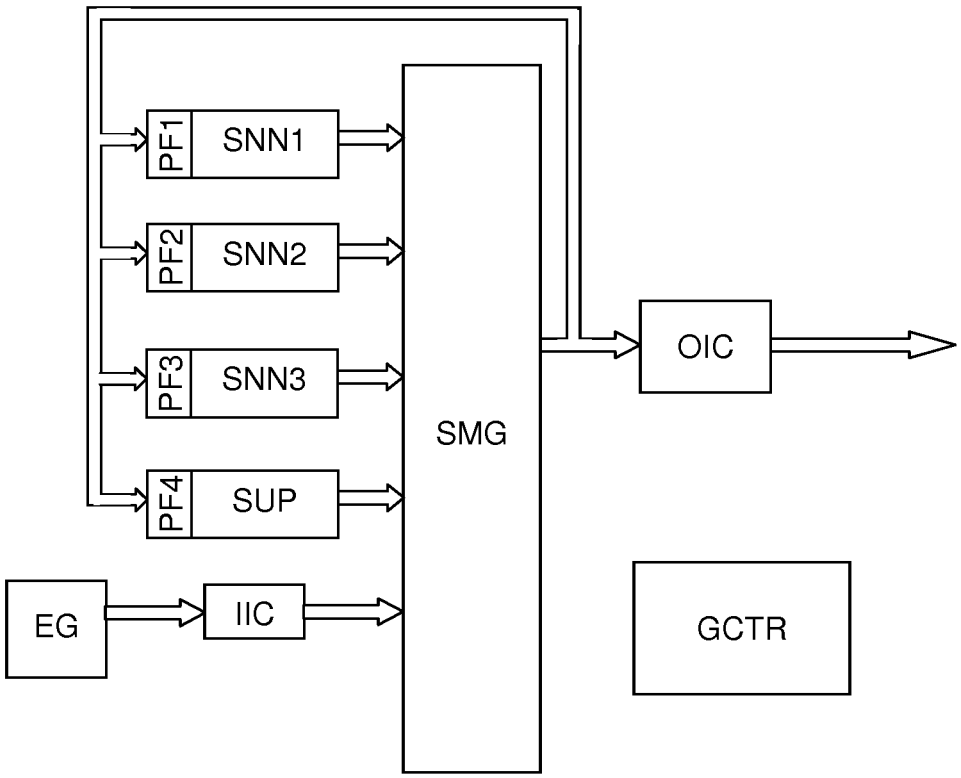
PF1
SNN1
PF2
SNN2
PF3
SNN3
PF4
SUP
SMG
OIC
EG
IIC
GCTR

FIG.8

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2017/079767

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06N3/04 G06N3/063 G06N3/08
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06N G06K

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal, WPI Data, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	P. Moen: "Attribute, event sequence, and event type similarity notions for data mining", Thesis for PhD in Computer Science at the University of Helsinki, February 2000 (2000-02), XP055375914, ISSN: 1238-8645 Retrieved from the Internet: URL:http://hdl.handle.net/10138/21371 [retrieved on 2017-05-19] cited in the application chapters 4 and 5 ----- -/--	1-16, 23-27

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
25 January 2018	07/02/2018

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Douarche, Nicolas

2

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X,P	S. Thorpe et al: "Unsupervised learning of repeating patterns using a novel STDP based algorithm", Abstracts of the 17th annual meeting of the Vision Sciences Society (VSS'17) to be held 19-24 May 2017, 53.4096, 1 May 2017 (2017-05-01), XP055375943, Retrieved from the Internet: URL:http://www.visionsciences.org/programs/VSS_2017_Abstracts.pdf [retrieved on 2017-05-19] abstract	1-27
X,P	A. Yousefzadeh et al: "Hardware implementation of convolutional STDP for on-line visual feature learning", Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS'17), 28 May 2017 (2017-05-28), XP055411618, Retrieved from the Internet: URL:http://www.capocaccia.cc/media/filer_public/8d/26/8d261621-1109-4e32-99ad-3e7c468df41b/iscas_convstdp.pdf [retrieved on 2017-09-15] sections II and III	1-27
A	C. DOMENICONI ET AL: "A classification approach for prediction of target events in temporal sequences", LECTURE NOTES IN COMPUTER SCIENCE, vol. 2431, 19 August 2002 (2002-08-19), pages 125-137, XP055375921, DOI: 10.1007/3-540-45681-3_11 section 4	1-27
A	J. MISRA, I. SAHA: "Artificial neural networks in hardware: A survey of two decades of progress", NEUROCOMPUTING, vol. 74, no. 1-3, 5 May 2010 (2010-05-05), pages 239-255, XP027517200, DOI: 10.1016/J.NEUCOM.2010.03.021 sections 1, 3, 4 and 5	1-27

-/--

2

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	D. QUERLIOZ ET AL: "Bioinspired networks with nanoscale memristive devices that combine the unsupervised and supervised learning approaches", PROCEEDINGS OF THE 2012 IEEE/ACM INTERNATIONAL SYMPOSIUM ON NANOSCALE ARCHITECTURES (NANOARCH'12), 4 July 2012 (2012-07-04), pages 203-210, XP032332257, DOI: 10.1145/2765491.2765528 sections II and IV -----	1-27

2