On the distribution of clique-based neural networks for edge AI

Benoit Larras, Antoine Frappé

On the Distribution of Clique-Based Neural Networks for Edge AI

Benoit Larras, *Member, IEEE*, and Antoine Frappé, *Senior Member, IEEE*

(Invited Paper)

***Abstract*—Distributed smart sensors are more and more used in applications such as biomedical or domestic monitoring. However, each sensor broadcasts data wirelessly to the others or to an aggregator, which leads to energy-hungry sensor nodes not ensuring data privacy. To tackle both challenges, this work proposes to distribute the feature extraction and a part of a clique-based neural network (CBNN) in each sensor node. This scheme allows standardizing data at the sensor level, ensuring privacy if the data is intercepted. Besides, a lower number of bits is transmitted, thus limiting the communication overhead. The inherent redundancy of clique-based networks makes them resilient to out-of-range connections, allowing an additional power reduction in the sensor nodes. Compared with a localized CBNN in the aggregator, the distributed structure reduces the inference latency by 28%, the sensor energy consumption by 25% and increases the protocol robustness. The circuit implementation is possible with the use of single-cluster iterative clique-based circuits, and demonstrated for a posture recognition application. To this end, a hardware circuit has been fabricated and performs a classification using 115fJ per synaptic event per neuron in 83ns.**

***Index Terms*—Neural networks circuit, clique-based neural networks, analog/mixed-signal circuit, distributed architecture.**

I. Introduction

With the growing number of global monitoring applications through various parameters, such as home monitoring or healthcare monitoring, wireless sensor networks (WSN) are vastly used. The standard architecture is as follows [1]. Every sensor in the network acquires data and broadcasts it wirelessly. Then, an aggregator acquires all the data from the sensors and processes the whole flow, typically using Artificial Intelligence (AI) paradigms.

However, this structure has three main drawbacks, at different levels of the system. First, at the aggregator level, the main processor has to be able to compute in real-time the complete flow of information. This necessary computational power is paid either in additional power consumption or computation time. The drawback becomes all the more an issue if the aggregator is destined to be embedded with battery operation, like a smartwatch or smartphone. Second, at the sensor level, transmitting the whole acquired data is energy-hungry and thus harmful to the battery lifetime. Third, at the system level, sending directly the data to the aggregator does not guarantee the information privacy if someone intercepts the transmitted data, which is especially crucial for healthcare monitoring.

The concept of "Near-sensor computing" is a solution to mitigate the first two drawbacks. It consists of shifting part of the processing – feature extraction and small-scale classification – at the sensor level, as explained in [2]. Pre-processing data in the sensor aims at decreasing the use-time of the communication interface by determining the relevance of the input data in regard to the targeted application. If the acquired data is relevant, a "wake-up" signal can be generated to enable the transmission; otherwise, the communication interface is in sleep mode, as illustrated in [3]. Moreover, when the interface is activated, only pre-processed data can be transmitted for analysis instead of raw data, thereby reducing the number of transmitted bits further.

While the feature extraction operation strongly depends on the type of sensor, the classification operation can be done by a generic unit, and thus output the same type of data flow for each sensor. This scheme allows anonymously transferring data from one or another sensor, addressing thus the issue of information privacy, provided that the classifier structure is generic. This work proposes to use clique-based neural networks (CBNNs) to be the generic classifier in the previously stated architecture, applied in a WSN used for posture recognition. The structure of CBNNs [4] allows them either to be embedded as a whole network in the aggregator or to be divided into clusters and dispatched in each node of the WSN. In [5], the latter case, where each cluster is embedded with a memory storing the connections between the neurons, is explored. The circuit structure, as well as the memory organization destined to be embedded is presented. This paper is an extended version of [5] and includes the following additional contributions:

- An applicative use-case context for the distributed sensor architecture is described. In this work, the study is conducted in the context of posture recognition where the sensors are Inertial Measurement Units (IMUs) placed at different places over the body.
- The two aforementioned envisioned scenarios using an embedded CBNN are detailed. The implications of the hardware in each case are also described.
- A thorough study about the advantages of distributing or not the clusters in the sensor nodes is presented. A comparison in terms of inference latency and energy consumption per node per inference between the two scenarios is given.
- Simulation results showing the impact of out-of-range sensor nodes on the network performance are provided.

This work was supported in part by the European CHIST-ERA JEDAI program under grant number ANR-19-CHR3-0005-01.
Benoit Larras and Antoine Frappé are with Univ. Lille, CNRS, Centrale Lille, Univ. Polytechnique Hauts-de-France, Yncrea Hauts-de-France, UMR 8520 - IEMN, F-59000 Lille, France. Email: benoit.larras@yncrea.fr

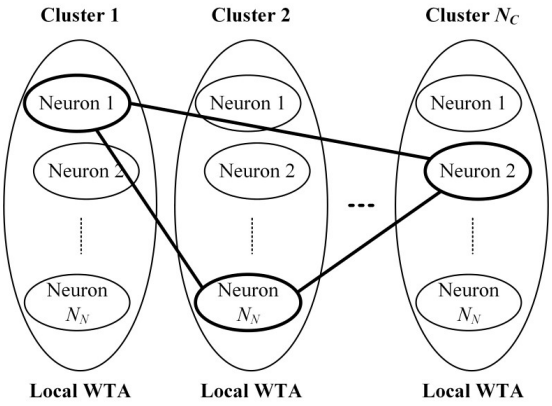

Figure 1. Topology of a clique-based neural network composed of N_C clusters of N_N neurons each. An example of a clique is highlighted in black.

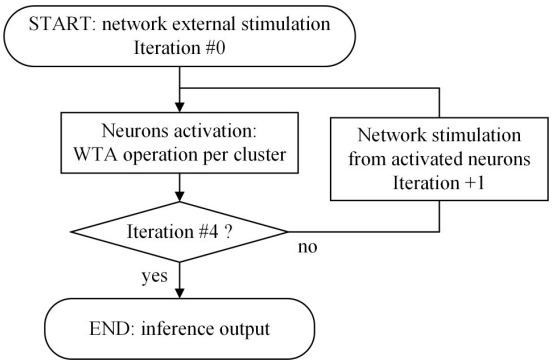

Figure 2. Algorithmic flowchart of the network inference. Four iterations of the WTA process are needed to converge to a stable state [6].

This structure has been tested with an on-chip single-cluster CBNN circuit driven by an external digital unit in an FPGA. In the ASIC, the analog structure designed for neuron computation consumes 115fJ to process a local classification, validating thus the possibility to integrate distributed CBNNs on a sensor node.

This paper is organized into the following sections. Section II provides details about the applicative context, as well as about CBNNs and how they can be distributed. Section III explains in detail the proposed architecture to be embedded. Section IV gives a comparison between a distributed CBNN and an aggregated CBNN for the same context. Finally, Section V presents a hardware realization and Section VI concludes the paper.

II. CBNN STRUCTURE

"Neural processors" circuits implement large Artificial Neural Networks (ANNs) composed of about one million neurons for embedded AI [7]-[9]. Oversized for tasks such as embedded associative memories, they are advantageously replaced by simpler clique-based ANNs [4], where specific classification tasks can be performed by networks of a few thousand of neurons in total [10]-[11]. Moreover, neuromorphic systems such as [12]-[13] operate at low frequency and are thus not adapted for real-time operation, unlike CBNNs which converge in tens of nanoseconds [14]. This Section aims at providing contextual elements about CBNNs and how they behave, as well as an applicative use-case with a WSN.

A. CBNN theory

CBNNs are not fully connected, have binary-weighted synapses and use simple activation functions. Clustered networks are formed by grouping neurons per category of information, as shown in Figure 1. There are N_C clusters, N_N neurons per cluster, and N_S synapses per neuron. Neurons from different clusters are connected *a priori* to form a clique. An in-cluster "Winner-Takes-All" (WTA) rule activates or not the neurons. Thus, only one neuron is activated per cluster. An algorithmic flowchart describing network behavior is shown in Figure 2. First, the neurons are stimulated externally and the first round of WTA is applied per cluster. Depending on the existing connections, the clusters propagate the neurons state (activated or not) to stimulate neurons in different clusters, before another round of WTA. In total, four iterations of neurons stimulation and WTA operation are performed to ensure the convergence of the network [6].

Low complexity and, hence, low power consumption are better exploited by implementing the neurons using analog CMOS circuits [4]. However, creating the neural connections *a priori* implies that they are not reconfigurable. This limits their uses for pattern recognition in a non-variable context once the system is calibrated. Proper calibration avoids the necessity of a self-configuration feature in this kind of context.

Several leads are envisioned to address the issue of reconfigurability in integrated CBNNs. One is to fabricate all the possible connections between neurons and store an activation bit in an embedded memory. This solution brings full flexibility in terms of possible stored patterns. It is thus adapted for applications where the context can change and an update of the patterns is necessary. However, the overhead in terms of silicon area compared to the non-configurable structure grows exponentially with the number of neurons, which is not suited for an integrated solution. A second solution to bring flexibility to the structure is to implement a single cluster and iterate the recovery process to emulate an entire network. The number of clusters and number of neurons can thus be changed (for a lower or equal number of neurons per cluster than in the implemented cluster), at the price of increased response time. The number of synapses per neuron N_S can also be identified to N_C in the structure, reducing the silicon area occupation. A dedicated memory is also necessary to store the connections between all the neurons, and the intermediate neuron states. In [14], this solution was chosen to emulate networks of up to 4,000 neurons with 128 neurons implemented on-chip.

B. Applicative use-case

In the context of WSN, an applicative use-case using Inertial Measurement Units (IMUs) such as ISM330DLC from ST Microelectronics® [15] for posture recognition is set. It is illustrated with five sensors in Figures 3 and 4. Several IMUs are dispatched all over the body, each of them recording

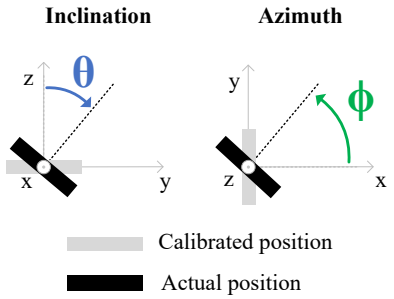

Figure 3. Definition of inclination and azimuthal angles compared to an absolute calibrated reference (in gray).

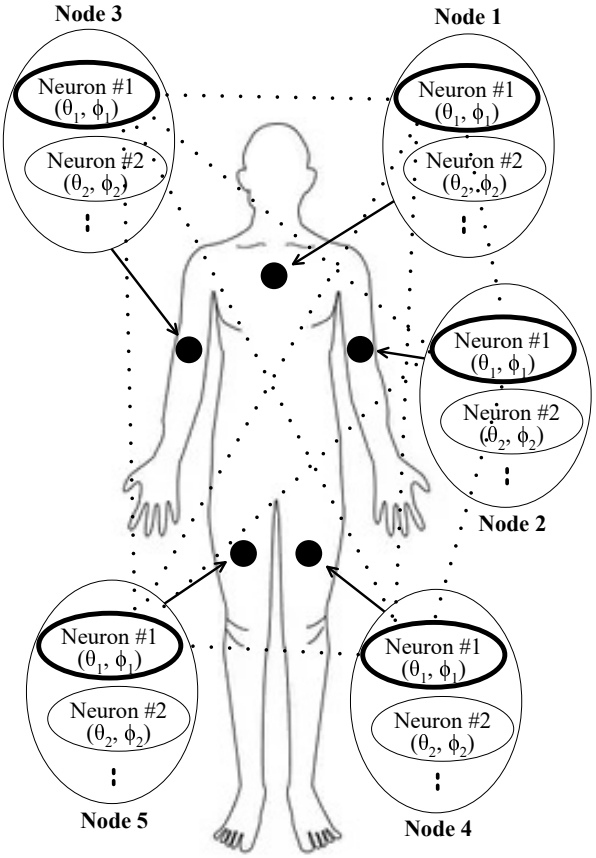

Figure 4. Example of a WSN of 5 IMUs dispatched on the body for posture recognition. Each sensor node is associated with a cluster from a CBNN, and each neuron corresponds to an absolute angular position of the sensor. A clique between neurons from different clusters is highlighted in black dotted lines, it corresponds to the "Laying down face up, azimuth 0°" posture from Table II.

the inclination angle θ and the azimuthal angle Φ of the absolute spherical coordinates with a 45° resolution, Figure 3. The angles are calculated from the calibrating position at device start-up, which is aligned with the z-axis. There are five possible values for θ (0°, 45°, 90°, 135°, 180°) and eight values for Φ (0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°), leading to 40 unique angular positions combining θ and Φ per accelerometer. One cluster in a CBNN is thus associated with one sensor in the WSN, and one neuron in a cluster corresponds to a combination (θ,Φ), Figure 4. The correspondence map between neurons in a cluster and the combinations (θ,Φ) is given in Table I.

Table I
CORRESPONDENCE MAP BETWEEN ACCELEROMETER DATA AND THE NEURONS IN A CLUSTER

Neuron	Associated couple (θ,Φ)	Neuron	Associated couple (θ,Φ)
Neuron #1 (N_1)	(0°,0°)	Neuron #21 (N_{21})	(90°,180°)
Neuron #2 (N_2)	(0°,45°)	Neuron #22 (N_{22})	(90°,225°)
Neuron #3 (N_3)	(0°,90°)	Neuron #23 (N_{23})	(90°,270°)
Neuron #4 (N_4)	(0°,135°)	Neuron #24 (N_{24})	(90°,315°)
Neuron #5 (N_5)	(0°,180°)	Neuron #25 (N_{25})	(135°,0°)
Neuron #6 (N_6)	(0°,225°)	Neuron #26 (N_{26})	(135°,45°)
Neuron #7 (N_7)	(0°,270°)	Neuron #27 (N_{27})	(135°,90°)
Neuron #8 (N_8)	(0°,315°)	Neuron #28 (N_{28})	(135°,135°)
Neuron #9 (N_9)	(45°,0°)	Neuron #29 (N_{29})	(135°,180°)
Neuron #10 (N_{10})	(45°,45°)	Neuron #30 (N_{30})	(135°,225°)
Neuron #11 (N_{11})	(45°,90°)	Neuron #31 (N_{31})	(135°,270°)
Neuron #12 (N_{12})	(45°,135°)	Neuron #32 (N_{32})	(135°,315°)
Neuron #13 (N_{13})	(45°,180°)	Neuron #33 (N_{33})	(180°,0°)
Neuron #14 (N_{14})	(45°,225°)	Neuron #34 (N_{34})	(180°,45°)
Neuron #15 (N_{15})	(45°,275°)	Neuron #35 (N_{35})	(180°,90°)
Neuron #16 (N_{16})	(45°,315°)	Neuron #36 (N_{36})	(180°,135°)
Neuron #17 (N_{17})	(90°,0°)	Neuron #37 (N_{37})	(180°,180°)
Neuron #18 (N_{18})	(90°,45°)	Neuron #38 (N_{38})	(180°,225°)
Neuron #19 (N_{19})	(90°,90°)	Neuron #39 (N_{39})	(180°,270°)
Neuron #20 (N_{20})	(90°,135°)	Neuron #40 (N_{40})	(180°,315°)

Table II
EXAMPLES OF STORED POSTURES (NON-EXHAUSTIVE LIST)

Stored clique	Corresponding posture
(N_1,N_1,N_1,N_1,N_1)	"Laying down face up, azimuth 0°"
(N_{17},N_{17},N_{17},N_{17},N_{17})	"Standing up arms down, azimuth 0°"
(N_{17},N_{17},N_{17},N_1,N_1)	"Sitting up, azimuth 0°"
(N_{17},N_{21},N_{17},N_{17},N_{17})	"Standing up left arm up, azimuth 0°"
(N_{19},N_{19},N_{19},N_{17},N_{17})	"Rotating torso 90° left standing up, azimuth 0°"

The postures are stored as cliques by activating the connections between the corresponding neurons. Examples of stored cliques are shown in Table II. In the left column, the notation used to indicate the neurons connected by a clique is the following: *(Neuron in Node 1,Neuron in Node 2, ... ,Neuron in Node 5)*. It is demonstrated in [4] that, in a CBNN with 40

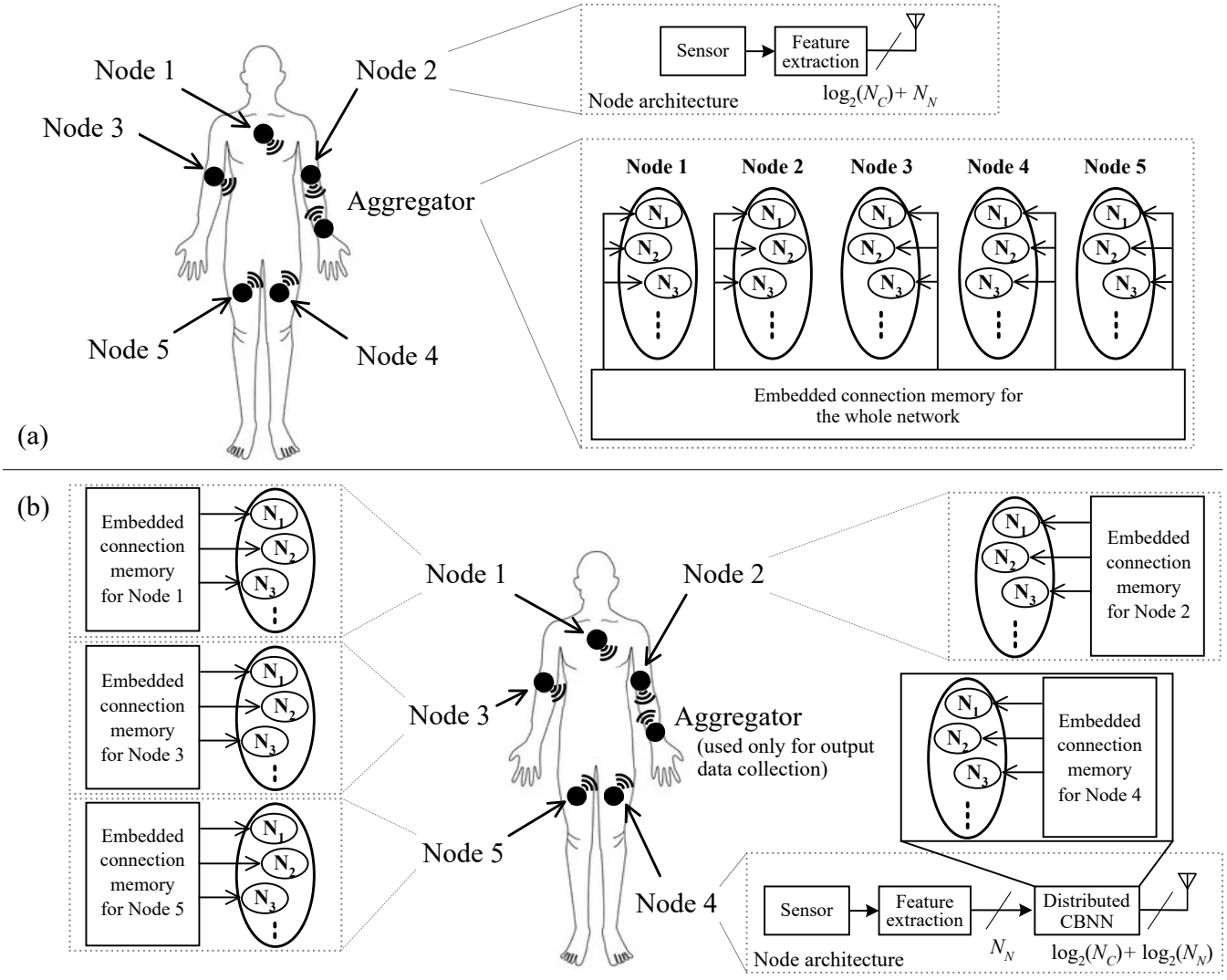

Figure 5. Structural comparison between: (a) Localized CBNN in the aggregator; and (b) Distributed CBNN in the sensor nodes. Each structure is composed of 5 sensor nodes and an aggregator.

neurons per cluster, it is possible to store 400 cliques, hence 400 different postures, without degrading the inference process with ideal stimulation conditions.

C. Distribution of a CBNN in the sensor nodes

In the literature, CBNNs are typically integrated on a single chip, as a whole like in [4], or iteratively like in [14]. In the application of WSNs, a CBNN can be integrated into the aggregator (*e.g.* a smartwatch), receiving the encoded data from the sensor nodes for the external stimulation of the neurons. This structure is illustrated in Figure 5-(a) in the applicative context, but can be extended to a generic CBNN structure of N_C clusters of N_N neurons each. In each sensor node, the angular features are extracted and formatted so that they correspond to one or several neurons to stimulate. Multiple stimulations are possible if the input data are not close enough to one feature in particular. Thus, the stimulation word, coded on N_N bits, is transmitted to the aggregator in addition to the cluster index, coded on $\log_2(N_C)$ bits. From Section II-A, a memory storing the connections between the neurons needs to be embedded in the aggregator so that the cliques can be modified in case a sensor is added or removed in the network, or if the number of stored postures needs to be changed. It contains $N_C^2 \times N_N$ words (*i.e.* the number of connections arriving on the clusters) of N_N bits each.

In the scope of the "Near-sensor computing" paradigm, instead of integrating the whole CBNN in the aggregator, the clusters can be dispatched among the nodes of the WSN. The aggregator is thus used only to collect the clusters' activation states and compare them to stored cliques. An example of a distributed CBNN for body monitoring is shown in Figure 5-(b) with the same CBNN structure as in Figure 5-(a). With this scheme, the transmitted data from the sensors are the outputs of the individual clusters instead of raw data from the sensors. Since only 1 neuron is active per cluster, only the index of the activated neuron in the cluster is transmitted, along with the cluster index, in base 2. Thus, the total number of bits transmitted by a node is reduced to $\log_2(N_C) + \log_2(N_N)$.

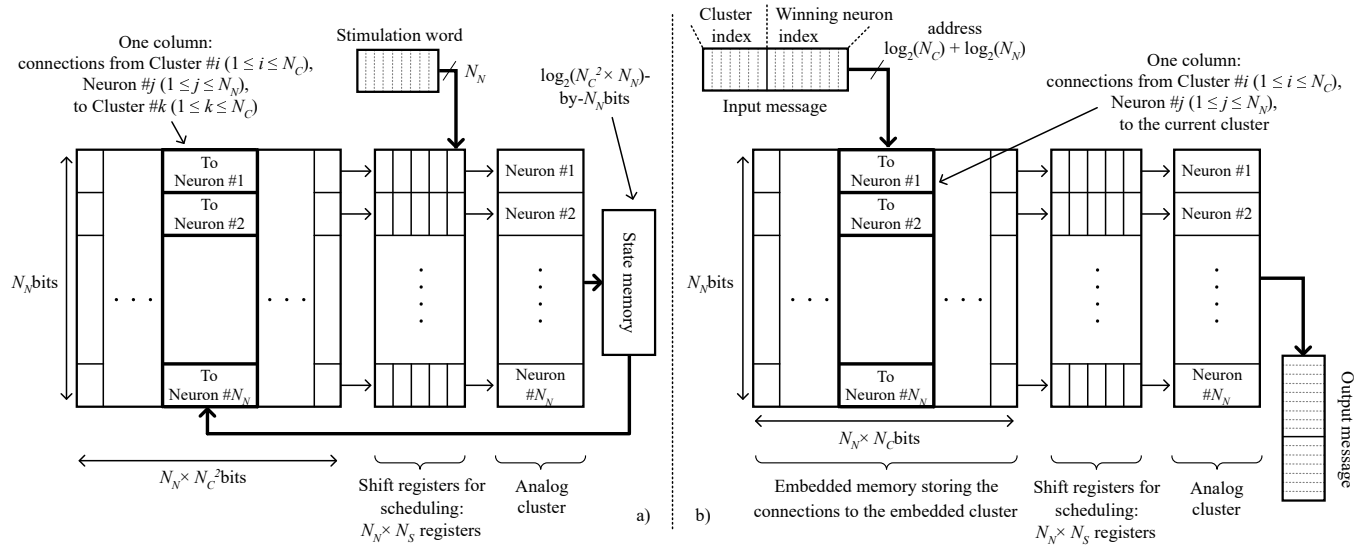

Figure 6. Organization of the embedded memory in: (a) the localized scenario; and (b) the distributed scenario in each sensor node. For the sake of clarity, memory words are displayed as columns. Each square represents a bit storing the connection state between 2 neurons from a distant cluster to the concerned integrated cluster. Each row represents a neuron in the concerned cluster (*destination neuron*), and each column a neuron in the whole network (*source neuron*).

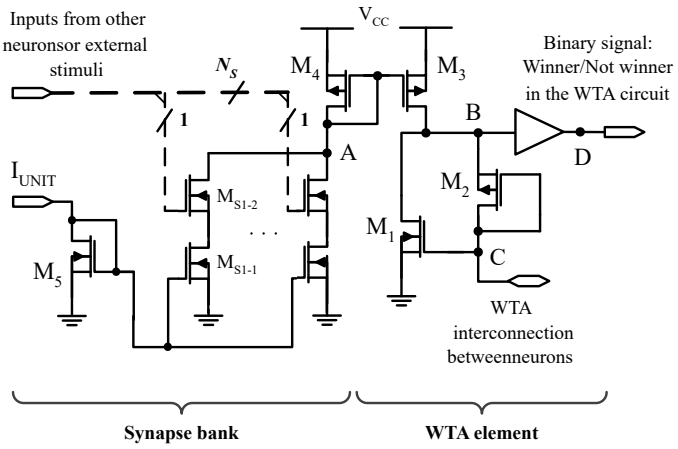

Figure 7. CMOS schematic of a neuron circuit.

Considering the embedded memory, it can be distributed as well in each node of the network. Each memory can, therefore, be reduced to $N_C \times N_N$ words of N_N bits each, storing only the incoming connections to the concerned integrated cluster. With smaller memories, the access time for each cluster is faster and thus more energy efficient. Moreover, since the cluster structure is generic, the data structure is also identical from one node to another. This means that, even if a message is intercepted, the context and information about the data can not be deduced from the transmitted message itself. It structurally ensures data privacy without having to spend resources to encrypt the transmission.

III. Circuit architecture

This section describes the implementation choices for the single-cluster circuit. The first subsection details the analog cluster circuit structure and the second subsection explains the different possible embedded memory topologies.

A. Mixed-signal cluster structure

The cluster circuit is structured as follows. Each neuron circuit adds contributions from the N_S synapses and performs the WTA operation locally, as shown in Figure 7. Synapses are switched current sources acting as V-to-I converters outputting either 0A or I_{UNIT}. The currents provided by the synapses are summed at node A.

One element of the WTA circuit is composed of M_1 and diode M_2. After being copied by the means of the M_3 - M_4 current mirror, the current flowing in the neuron sets the drain-source voltage of M_1 depending on the other currents flowing in the neurons in the same cluster, connected through node C. The operation results in a binary voltage at node B. If the current flowing in a neuron is the highest in the cluster, the drain-source voltage of M_1 is set over its saturation voltage. Otherwise, M_1 is blocked and its drain-source voltage is set below its saturation voltage.

A digital buffer converts the binary voltage at node B into a standard binary voltage at node D. The index of the winning neuron is then transmitted to the rest of the network.

B. Memory distribution and stimulation process

The embedded memory stores the binary connections between the different neurons in the network. The memory organization is shown in Figure 6 for both localized and distributed scenarios. In the localized architecture, the connection memory stores ${N_C}^2 \times N_N$ words of N_N bits each, organized in columns for the sake of clarity in Figure 6-(a). The number of words corresponds to all the possible *source neurons* ($N_C \times N_N$) for all the N_C clusters. One word is input in the cluster and is thus N_N-bit long.

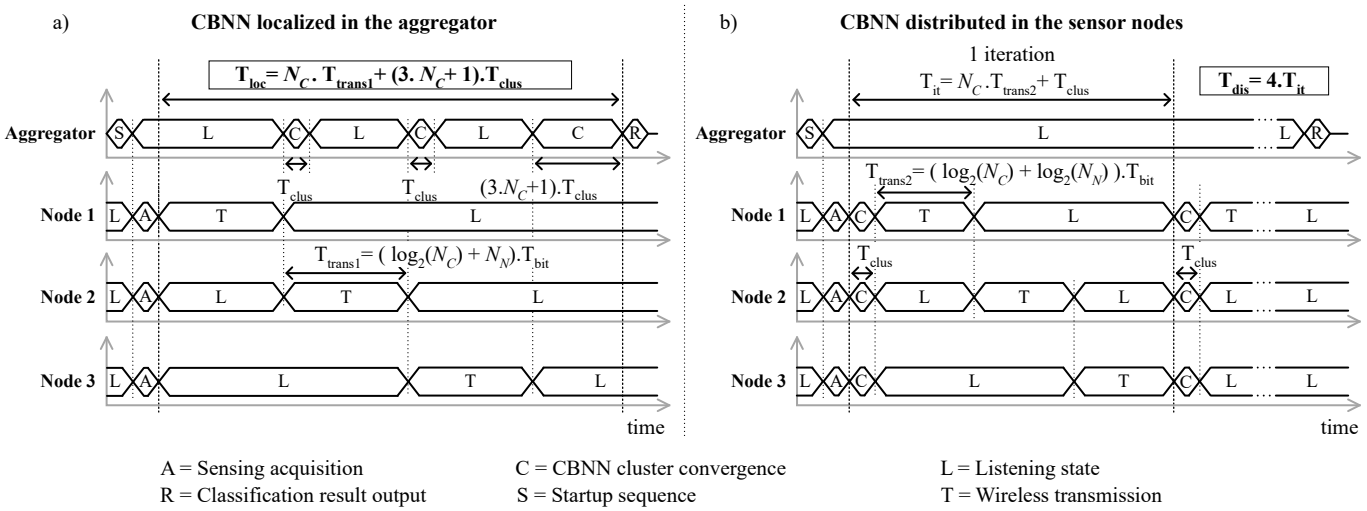

Figure 8. Timing description of the two scenarios: (a) for a localized CBNN in the aggregator; and (b) for a distributed CBNN in the sensor nodes. The topology of the structures are identical and composed of N_C nodes and N_N neurons per cluster.

In the distributed architecture, the whole connection memory does not have to be integrated into every node. Only the connections incoming from other clusters to the considered integrated cluster are implemented, Figure 6-(b). It contains $N_C \times N_N$ words, corresponding to the number of neurons in the whole network. Each word contains N_N bits, as 1 bit is used to stimulate 1 neuron in the integrated cluster.

The reading process is scheduled as follows. In the localized scenario, the N_N stimulation bits coming from the sensors are directly input in the cluster through the shift registers. The cluster index is only used for scheduling. After each WTA operation in the same iteration, the winning neuron index is stored in a state memory so that the connected neurons can be stimulated during the next iterations of the inference process.

In the distributed scenario, the stimulation bits are also directly input in the cluster through the shift registers and the first message is output. During the next iterations, each circuit receives a message of $log_2(N_C)+log_2(N_N)$ bits from another cluster indicating the winning neuron in that cluster, Figure 6. This message serves as an address to retrieve the activated connections to the current cluster in the embedded memory. The selected memory word concerns 1 cluster in the network. The word is pushed in a temporary shift register, waiting for the messages from the other clusters. Once all the clusters have sent data, all the memorized bits are input in the analog cluster at the same time. If a cluster takes too long to send its message or is out of range, a timeout signal is triggered and the shift register is sent as-is to the analog cluster. The structural redundancy of CBNNs can recover a stored clique as long as half of the neurons are stimulated.

IV. Comparison between CBNN architectures

In this Section, the objective is to compare the benefits of both scenarios envisioned in Section II-C. The comparison criteria include inference latency and energy consumption per node per inference. The case of out-of-range nodes in the WSN is also studied in both scenarios and the impact on the transmission power in each node is given.

A. Inference latency

The inference process, including 4 iterations of WTA operations in the CBNN, is depicted in Figure 8-(a) for the localized CBNN in the aggregator scenario, and Figure 8-(b) for the distributed CBNN in the sensor nodes scenario. In both cases, the wireless link is modeled from a stae-of-the-art Bluetooth© Low-Energy (BLE) transceiver/receiver [16], with a 1 Mbps data rate or a transmission bit period $\mathrm{T_{bit}} = 1\mu$s. The considered CBNN hardware is the single-cluster architecture from [14], implying successive computations in time on the same chip in the localized scenario, or multiple physical instances of the chip in each sensor in the distributed scenario.

In the localized scenario, the aggregator sends a start-up signal for the sensor nodes triggering data acquisition. Once the feature extraction step has ended, each sensor transmits one after another to the aggregator the corresponding $\log_2(N_C) + N_N$-bit word for CBNN stimulation. In the aggregator, after a stimulation word is received, the first iteration of the inference process concerning the corresponding cluster starts and lasts the cluster convergence time $\mathrm{T_{clus}}$. The value of $\mathrm{T_{clus}}$ is 83 ns from [14]. Once all the stimulation words are received, the remaining three iterations of the inference process are done cluster by cluster in the aggregator, during $3.N_C.\mathrm{T_{clus}}$. The total inference time in that scenario $\mathrm{T_{loc}}$ is thus equal to:

$$\mathrm{T_{loc}} = N_C.(\log_2(N_C) + N_N).\mathrm{T_{bit}} + (3.N_C + 1).\mathrm{T_{clus}} \quad (1)$$

In the distributed scenario, after the start-up signal and the feature extraction, a cluster convergence starts in each sensor node. Each one of them transmits a $\log_2(N_C)+\log_2(N_N)$-bit word to the other sensors to update their data with the entire network data for the next iteration. The four iterations of the inference are thus processed in the sensors and the aggregator

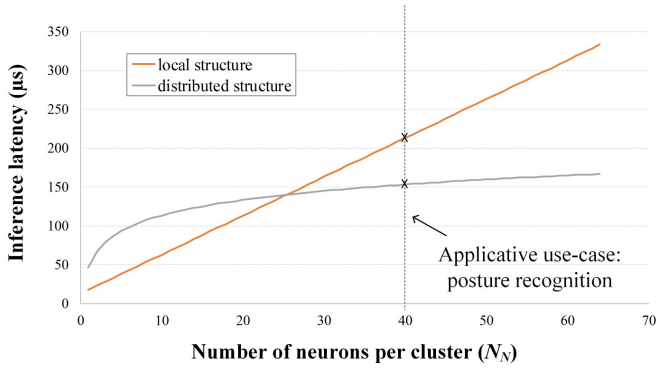

Figure 9. Inference duration model depending on the number of neurons in each sensor node N_N in both localized and distributed scenarios. The time values are calculated for $N_C = 5$ sensor nodes.

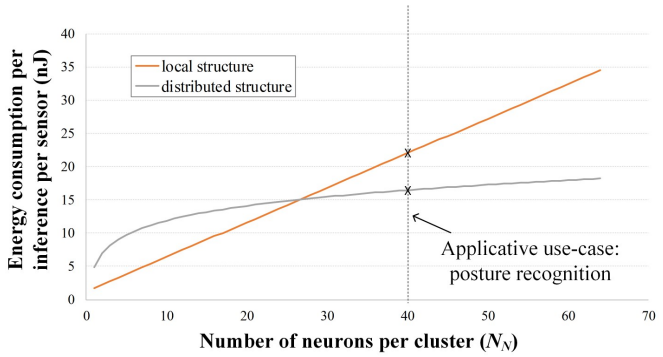

Figure 10. Energy consumption model per sensor node per inference depending on the number of neurons N_N in both localized and distributed scenarios. The energy values are calculated for $N_C = 5$ sensor nodes.

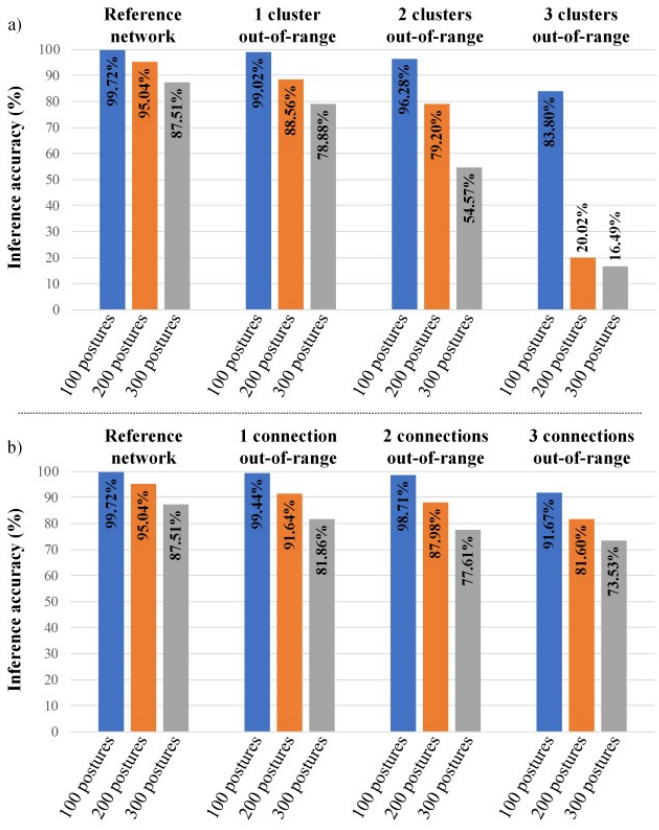

Figure 11. Inference accuracy of a CBNN composed of $N_C = 5$ sensor nodes used in the posture recognition application, for 100, 200 and 300 stored postures, simulated with *Matlab*®. (a) Impact of out-of-range clusters in a localized architecture; and (b) Impact of out-of-range connections in a distributed architecture.

only outputs the final result. The total inference time in that scenario $\mathrm{T_{dis}}$ is equal to:

$$\mathrm{T_{dis}} = 4.(N_C.(\log_2(N_C) + \log_2(N_N)).\mathrm{T_{bit}} + \mathrm{T_{clus}}) \quad (2)$$

Figure 9 shows the evolution of the inference time in both scenarios depending on the number of neurons per cluster N_N, *i.e.* the quantization of each parameter. The values of $\mathrm{T_{loc}}$ and $\mathrm{T_{dis}}$ are given for $N_C = 5$ sensor nodes, but the behavior shown in Figure 9 is the same for other values of N_C since both $\mathrm{T_{loc}}$ and $\mathrm{T_{dis}}$ are proportional to N_C. However, this graph shows a critical value of N_N after which the distributed architecture becomes interesting in terms of inference time. This value is 26 in this work but can change depending on the hardware used for data transmission or CBNN inference. The posture recognition application will thus benefit from the distributed architecture in terms of inference duration. The values of $\mathrm{T_{loc}}$ and $\mathrm{T_{dis}}$ for $N_N = 40$ neurons are 213 μs and 153 μs, respectively.

B. Energy consumption per inference per sensor

From the process described in Figure 8 and the power consumption figures from [14] and [16], the energy consumption per inference per sensor node is derived. Its value for both scenarios is shown in Figure 10 for $N_C = 5$ sensor nodes. From [16], the power consumption has been scaled by a factor of 30 so that the distance range of the BLE specifications matches the typical distance range of the applied WSN, *i.e.* 2 m. The energy consumption of the BLE transceiver/receiver still averages 97% of the inference energy consumption per sensor node in the distributed scenario. Therefore, the behavior of the energy values follows that of the inference time, also mostly depending on the wireless data transmission. A critical value of N_N where the distributed architecture becomes interesting in terms of energy consumption per inference per sensor node also exists. Its value is 26 for the considered hardware. Thus, the distributed architecture allows reducing the energy consumption per inference per sensor node in the posture recognition use-case from 22 nJ (in the localized architecture) to 16.5 nJ. This study also highlights that the potential gain in power consumption (and latency) effectively increases with larger values of N_N.

C. Network robustness and power optimization

In the specific context of body WSNs, the relative positions of the sensor nodes change over time. Therefore, the power of the wireless transceiver/receiver has to be set including a margin in the distance range so that the sensor nodes are never out of range from one another. However, by using CBNN

for the inference, it is possible to cut off several connections between sensor nodes and still manage to classify data with only a slight loss of accuracy. The power of the transmission front-end can thus be reduced as the distance range margin is not necessary anymore. Besides, since most of the sensor power is used for data transmission, a noticeable energy gain (depending on the distance between the sensors) can make up for the accuracy loss.

A CBNN of five clusters of 40 neurons each is simulated using *Matlab*® to estimate the effect of out-of-range sensor nodes on the classification accuracy. Three cases have been simulated for 100, 200, and 300 postures stored. The simulation results are shown in Figure 11-(a) for the localized scenario and in Figure 11-(b) for the distributed scenario. In the localized scenario, an out-of-range sensor node means that the corresponding feature for cluster stimulation is lost. The corresponding cluster is therefore not stimulated during the first iteration, but still plays a role in the last three iterations of the inference process. In the distributed scenario, a sensor node can be in the range of all the other sensors but one. In that case, the messages between the two clusters will not be received at all, as explained in Section III-B.

The inference results of 10,000 simulations are averaged in each case. In each simulation, a 10% stimulation error probability models the sensing error of the IMU for each sensor node. The CBNN can detect the postures in 99.72%, 95.04%, 87.51% of the cases if there are 100, 200, 300 postures stored in the CBNN, respectively. The more stored cliques, the more a stimulation error has a chance to lead to a wrong classification. This is all the more impacting in a localized architecture since stimulation errors due to out-of-range sensors stack with stimulation errors due to the sensors themselves, as shown in Figure 11-(a). However, in the distributed architecture, out-of-range cluster-to-cluster connections have a limited impact on the global inference, Figure 11-(b). This is because data output by the distant clusters are still propagated by the other in-range clusters, thus, at the network level, the impact of the lost connection is mitigated. It is therefore possible to reduce further the power of the transceiver (and its range) with minimal effect on the global inference. Finally, another way to limit the impact of out-of-range connections is to increase the number of clusters in the CBNN. It increases the size of the cliques and thus the redundancy of the network. As an example, three lost connections in a CBNN of $N_C = 7$ clusters storing 300 postures still yield an inference accuracy of 96.98% (versus 99.1% for the 7-cluster reference network).

V. Hardware Prototype and Testing

The single-cluster architecture has been integrated on-chip using the ST 65-nm CMOS technology, Figure 12. A cluster of 128 neurons with 31 synapses each has been fabricated, as well as the 128×31 shift registers loading the data in the synapses. The whole circuit occupies a silicon area of 0.21mm^2. The circuit operates at a supply voltage V_{CC} of 1V, with a unitary current I_{UNIT} of 300nA to ensure transistor operation in the moderate inversion regime. With these parameters, the measured static power consumption is of 23.4μW.

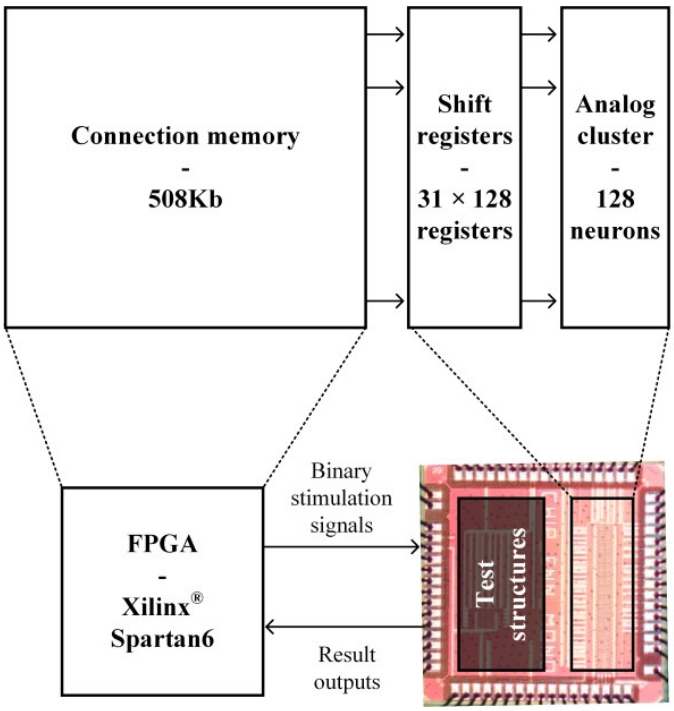

Figure 12. Chip integration of the proposed architecture. The analog cluster and the shift registers have been integrated on-chip, while the connection memory is implemented externally on an FPGA for simplicity reasons.

For the sake of simplicity, the connection memory of 508Kb is implemented off-chip on a *Xilinx*® *Spartan6*-based FPGA board. The silicon area of the connection memory has been estimated for the 65-nm CMOS technology process using *Synopsys*® and SRAM memory density provided by the circuit manufacturer. The estimated silicon area is 0.53mm^2. The energy efficiency of the loading process from the connection memory to the shift registers has been also estimated using using *Synopsys*®. Its value is 3.3pJ per clock cycle per neuron, which corresponds to 103pJ per neuron for the complete loading process, using 40 neurons out of the 128 implemented.

Once the shift register is loaded and the synapses are activated, the analog cluster takes 83ns (in the worst case) to converge and output the winning neuron index. During that time, the power consumption is very application-dependent since it varies depending on the number of active connections. For 1 active synapse, the measured dynamic power consumption overhead is 1.2μW. It corresponds to the successive mirroring of I_{UNIT} 4 times through the synapse and neuron circuits. Therefore, the unitary amount of energy needed to propagate an event from the synapse to the neurons' output, defined as a *synaptic event*, is 115fJ, including static biasing. As a reference, an artificial neuron in [7] consumes 26pJ per synaptic event in 28-nm CMOS process, while a neuron in [9] consumes 23.6pJ per synaptic event in 14-nm CMOS process. However, while both implementations in [7] and [9] have a higher computational power, their higher number of implemented neurons and denser connectivity make them less suited for embedded distributed applications. They are more adapted for integration as a whole in the aggregator, but they

do not address the issue of data exchange introduced in this work.

VI. Conclusion

This paper presents a distributed scheme for CBNNs to be integrated into multiple sensors in the same WSN. The proposed structure leads to a reduction of the number of transmitted bits, thus decreasing both the energy consumed per node, the inference latency and the complexity needed to process the complete data stream in the aggregator node. Besides, using the same structure in each node with the same output format anonymizes the transmissions, so that intercepting one data stream without knowing the context does not allow recovering its meaning. Distributing a CBNN also offers a resistance to connections being cut off because out-of-range sensor nodes. The proposed scheme is validated by the means of *Matlab®* simulations and a single-cluster CBNN ASIC dedicated to distributed computation and consuming 115fJ per neuron per synaptic event. Further developments will include the fabrication of the complete single-cluster ASIC, including the connection memory. The long-term objective of this work is to embed the fabricated chip in a wireless sensor node for the demonstration of AI distribution in the posture recognition application.

References

[1] S. Movassaghi, M. Abolhasan, J. Lipman, D. Smith, and A. Jamalipour, "Wireless Body Area Networks: A Survey," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1658–1686, Third 2014.

[2] R. LiKamWa, Y. Hou, Y. Gao, M. Polansky, and L. Zhong, "RedEye: Analog ConvNet Image Sensor Architecture for Continuous Mobile Vision," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 255–266.

[3] S. Lecoq, J. L. Bellego, A. Gonzalez, B. Larras, and A. Frappé, "Low-complexity feature extraction unit for "Wake-on-Feature" speech processing," in *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Dec 2018, pp. 677–680.

[4] B. Larras, C. Lahuec, F. Seguin, and M. Arzel, "Ultra-Low-Energy Mixed-Signal IC Implementing Encoded Neural Networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 11, pp. 1974–1985, 2016.

[5] B. Larras and A. Frappé, "Distributed Clique-Based Neural Networks for Data Fusion at the Edge," in *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2020, pp. 55–58.

[6] V. Gripon and C. Berrou, "Sparse Neural Networks with Large Learning Diversity," *Neural Networks, IEEE Transactions on*, vol. 22, no. 7, July 2011.

[7] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G. J. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha, "Truenorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, Oct 2015.

[8] E. Painkras, L. A. Plana, J. Garside, S. Temple, F. Galluppi, C. Patterson, D. R. Lester, A. D. Brown, and S. B. Furber, "SpiNNaker: A 1-W 18-Core System-on-Chip for Massively-Parallel Neural Network Simulation," *Solid-State Circuits, IEEE Journal of*, vol. 48, no. 8, pp. 1943–1953, Aug 2013.

[9] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. H. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, January 2018.

[10] B. Larras, B. Boguslawski, C. Lahuec, M. Arzel, F. Seguin, and F. Heitzmann, "Analog encoded neural network for power management in MPSoC," *Analog Integrated Circuits and Signal Processing*, vol. 81, no. 3, pp. 595–605, 2014.

[11] P. Chollet, R. Pallas, C. Lahuec, M. Arzel, and F. Seguin, "A sub-nJ CMOS ECG classifier for wireless smart sensor," in *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, July 2017, pp. 3840–3843.

[12] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A Scalable Multicore Architecture With Heterogeneous Memory Structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 1, pp. 106–122, 2018.

[13] C. Frenkel, J. Legat, and D. Bol, "MorphIC: A 65-nm 738k-Synapse/mm^2 Quad-Core Binary-Weight Digital Neuromorphic Processor With Stochastic Spike-Driven Online Learning," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 5, pp. 999–1010, 2019.

[14] B. Larras, P. Chollet, C. Lahuec, F. Seguin, and M. Arzel, "A Fully Flexible Circuit Implementation of Clique-Based Neural Networks in 65-nm CMOS," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 5, pp. 1704–1715, May 2019.

[15] "ISM330DLC Datasheet." [Online]. Available: https://www.st.com/resource/en/datasheet/ism330dlc.pdf

[16] H. Liu, Z. Sun, D. Tang, H. Huang, T. Kaneko, Z. Chen, W. Deng, R. Wu, and K. Okada, "A DPLL-Centric Bluetooth Low-Energy Transceiver With a 2.3-mW Interference-Tolerant Hybrid-Loop Receiver in 65-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 12, pp. 3672–3687, 2018.

Benoit Larras was born in Nancy, France in 1988. He received both his engineering degree and his Master in Telecommunications from IMT Atlantique in 2012. He received his Ph.D. in Electrical Engineering in 2015 from IMT Atlantique, Brest, France. He is now an associate professor at Yncréa Hauts-de-France, Lille, France, in the Electronics Team. His research topic is about analog/mixed-signal IC design and circuit implementation of neural networks and associative memories, in the context of "Near-sensor computing" and "Edge computing". He is the co-recipient of the Best Paper Award (2nd Honorary Mention) at the IEEE AICAS2020 conference.

Antoine Frappé (M'08-SM'17) graduated from the Institut Supérieur d'Électronique du Nord (ISEN), Lille, France in 2004. He received the M.Sc., Ph.D. and HDR (French highest academic degree) in electrical engineering from the University of Lille, France in 2004, 2007 and 2019, respectively. In 2007, he was with the Silicon Microelectronics group at the Institute of Electronics, Microelectronics, and Nanotechnologies (IEMN) in Villeneuve d'Ascq, France. He obtained a Fulbright grant in 2008 to pursue research in communication systems at the Berkeley Wireless Research Center (BWRC) at UC Berkeley, CA, USA. He is now Associate Professor at Yncréa Hauts-de-France, leading the Electronics Team. His research interests concern digital RF transmitters, high-speed converters, mixed-signal design for RF and mmW communication systems, energy-efficient integrated systems, event-driven and neuro-inspired circuits for embedded machine learning.