Fully Binary Neural Network Model and Optimized Hardware Architectures for Associative Memories

PHILIPPE COUSSY, CYRILLE CHAVET, HUGUES NONO WOUAFO,
and LAURA CONDE-CANENCIA, Université de Bretagne Sud, Lab-STICC

Brain processes information through a complex hierarchical associative memory organization that is distributed across a complex neural network. The GBNN associative memory model has recently been proposed as a new class of recurrent clustered neural network that presents higher efficiency than the classical models. In this article, we propose computational simplifications and architectural optimizations of the original GBNN. This work leads to significant complexity and area reduction without affecting neither memorizing nor retrieving performance. The obtained results open new perspectives in the design of neuromorphic hardware to support large-scale general-purpose neural algorithms.

Categories and Subject Descriptors: B.7.1 [**Integrated Circuits**]: Types and Design Styles—*Algorithms implemented in hardware*; C.1.3 [**Processor Architectures**]: Other Architecture Styles—*Neural nets*; I [**Computing Methodologies**]

General Terms: Design, Algorithms

Additional Key Words and Phrases: Neural network, sparse network, associative memory, neural cliques

ACM Reference Format:
Philippe Coussy, Cyrille Chavet, Hugues Nono Wouafo, and Laura Conde-Canencia. 2015. Fully binary neural network model and optimized hardware architectures for associative memories. ACM J. Emerg. Technol. Comput. Syst. 11, 4, Article 35 (April 2015), 23 pages.
DOI: http://dx.doi.org/10.1145/2629510

1. INTRODUCTION

Human brain is a powerful machine able to realize complex operations like abstracting, memorizing, feeling, reasoning, controlling, or solving problems. The enormous memory capacity of the brain, as well as its power efficiency and processing capability are the main features that interest the information technology community. In this research field, neuroscientists have explored several computational models of brain processing, providing the promise of practical applications in many domains (e.g., vision, navigation, motor control, decision-making, etc.). These developments have led to different approaches [Nageswaran et al. 2010] such as neuromorphic VLSI [Mead 1990; Furber and Temple 2007], neuro-biological systems [Jiping et al. 2001; Giotis et al. 2011; Theodorou and Valero-Cuevas 2010] and brain-inspired algorithms [FrostGorder 2008; Jhuang et al. 2008; Soo-Young 2007].

Levels of abstraction used in these research works range from biophysical up to theoretical models through neural-circuit, application-specific and generic models (see Nageswaran et al. [2010] for more details). Our concern is on generic models that rely

Authors' addresses: P. Coussy, C. Chavet, H. N. Wouafo, and L. Conde-Canencia, Lab-STICC Laboratory, Université de Bretagne Sud, Centre de recherche Christiaan Huygens, rue St Maude, 56100 Lorient, France; email: {philippe.coussy, cyrille.chavet, hugues.nono-wouafo, laura.conde-canencia}@univ-ubs.fr.

1550-4832/2015/04-ART35 $15.00
DOI: http://dx.doi.org/10.1145/2629510

on the fact that brain-circuits have a template architecture where similar circuits are replicated for processing and learning various sensory signals [Hawkins and Blakeslee 2004; Granger 2006]. Most of the generic models do not incorporate detailed neural mechanisms (i.e., spikes, oscillations$\cdots$) but rather consist of algorithmic considerations such as clustering or associative memory. Indeed, inside the brain, information processing is a multilevel process that mostly relies on pattern matching and sensory association rather than calculation and logic inference. Brain creates invariant representations from always changing inputs through a complex hierarchical associative memory organization [Kandel et al. 2013]. Unlike a conventional computer, the brain does not rely on programmed instructions, but rather on a complex process to learn, store and retrieve information [Sandberg 2003; Hawkins and Blakeslee 2004]. In this context, tasks like voice or face recognition can take a considerable advantage by using content-based access techniques offered by associative memories, compared to conventional computer systems in which such processing are complex to realize. Moreover, these associative memories are robust against input noise and have a practically constant retrieval time independent of the number of stored associations. These memories have been intensively studied in the past (e.g., Kohonen [1977], Palm [1980, 2013], and Willshaw [1971]) with many successful applications (e.g., Sudo et al. [2009], Perfetti and Ricci [2008], and Annovi et al. [2013]).

Abstract neural networks can be used to design neural associative memories [Palm 1980, 2013] and to perform learning and association function following the Hebb's Theory [Hebb 1949]. Associative memories can in turn be used as computational models for cortical circuits where the computation function allows learning and retrieving information. Several neural-network based models have been proposed to design associative memories like Hopfield networks (HNN) [Hopfield 1982], Boltzman machines [Ackley et al. 1985] or Kohonen maps [Kohonen 1977]. Those models create internal states allowing to exhibit dynamic temporal behavior and to process arbitrary sequences of inputs. Hopfield networks have been defined in order to guarantee that its dynamics will converge; they are very simple dynamic models that seem to offer an attractive learning capacity. However, efficiency of such models/representations collapses (to zero) as the amount of learned messages grows ad infinitum.

In this article, we introduce a new fully binary neural network model that significantly reduces the complexity of the original GBNN model, and that allows designing efficient hardware architectures without loss of performance. For that purpose, the arithmetical-integer semantic of the original GBNN has been transformed into a logical semantic. Further optimizations are proposed to tackle area through reduced memory complexity and serialized communications. The memory reduction halves the storage requirements (i.e., the memory footprint) and reduces the complexity of the learning process. The communication serialization in the neural network leads to clock frequency optimization and strong area reduction while guaranteeing the same functionality as in the original GBNN.

This article is organized as follows: Section 2 introduces the GBNN model. Section 3 introduces the new fully binary model that we propose. Section 4 compares the performance of the original GBNN and the proposed new model through simulation results. This section also compares hardware implementations of the original GBNN and the proposed new model in terms of area and clock frequencies through synthesis results on both FPGA and ASIC. Finally, Section 5 draws conclusion and future work.

2. GBNN: A SPARSE NEURAL NETWORK WITH LARGE LEARNING DIVERSITY

2.1. Principle

A GBNN is an abstract neural network model based on sparse clustered networks that can be used to design associative memories. The principle of GBNN is to gather sets of

neurons in clusters. Neurons (also called *fanals*) that belong to the same cluster cannot be connected to each other (this principle is called *sparcity*). However, any neuron of a given cluster can be connected to any neuron in any other cluster. More precisely, the network consists of N binary neurons arranged into C equally-partitioned clusters and each cluster includes $L = N/C$ neurons. Each cluster is associated through one of its neurons with a portion of an input message to be learned or retrieved. A message m of K bits is thus divided into C clusters each with L fanals and the length of the submessage associated with each cluster is $X = K/C = log_2(L)$. During the learning phase, the network memorizes that the set of activated neurons (i.e., the set of neurons that constitute the input message) are connected to each other and form a *clique*. Unlike Hopfield networks, GBNN uses binary weighted connections between neurons to record if a connection exists or not. Then, to memorize a connection that exists between one neuron i of cluster j, $n_{i,j}$ and one neuron k from cluster g $n_{k,g}$ (with $j \neq g$), each neuron stores locally the value "1" in its corresponding synaptic weight w (i.e., $w_{(i,j)(k,g)} =$ "1" in cluster j and $w_{(k,g)(i,j)} = 1$ in cluster g). All the weights are initialized to 0 that is, no message has been learnt before training. The retrieving process is based on a Scoring step and a Winner Takes All (WTA) step to detect which neuron, in a cluster associated to a missing part of the message, is the most "stimulated" one. Equation (1) defines the original scoring function used to compute the "score" of a neuron $n_{i,j}$ at time instance $t+1$, $s_{n_{i,j}}^{t+1}$. This score depends on all the values of the other neurons k from all the other clusters g (i.e., $n_{k,g}$) in the GBNN computed at time instant t (i.e., in the previous iteration of the network) and on the corresponding synaptic weights (i.e., value of $w_{(k,g)(i,j)}$) that have been stored during the learning process.

$$\forall i \in \left[0..L-1\right], j \in \left[0..C-1\right],$$
$$s_{n_{i,j}}^{t+1} \leftarrow \sum_{g=0 \text{ and } g\neq j}^{C-1} \sum_{k=0}^{L-1} w_{(k,g)(i,j)} * v^t\left(n_{k,g}\right). \tag{1}$$

The WTA Equation (2) allows defining in each cluster c_j the neuron $n_{i,j}$ or the group of neurons to activate, that is, the neuron or the group of neurons that achieves the maximum score S_{max}

$$S_{max,c_j}^{t+1} = \max_{0 \leq n_i \leq L-1} \left[s_{n_{i,j}}^{t+1}\right] \tag{2}$$
$$v^{t+1}(n_{i,j}) = \begin{cases} 1 & \text{if } s_{n_{i,j}}^{t+1} = S_{max,c_j}^{t+1} \\ 0 & \text{otherwise.} \end{cases}$$

The network converges in a few time instances, called *iterations*. At the end of the process, the clusters which originally had no selected neuron are provided with the selection of a neuron or a group of neurons. The answer of the network is then defined by the set of neurons that were chosen to represent each single cluster.

Figure 1(a) presents a GBNN neural network based on three clusters of three neurons. Let us consider that the network has learned three messages represented by cliques: $(n_{1,0}, n_{0,1}, n_{0,2})$, $(n_{2,0}, n_{1,1}, n_{0,2})$, and $(n_{2,0}, n_{2,1}, n_{0,2})$. These cliques are represented by using a binary synaptic-weight matrix (also called "*Interconnection matrix*") thanks to a classical representation of adjacency matrix as depicted in Figure 1(b) Each line of the matrix contains the weights storing the existence -or not- of connections between the neurons of a cluster and the neurons of other clusters. There is no element in the diagonal since the neurons of a given cluster cannot be connected between them. Note that this means that these connections are not represented in the synaptic weights matrix (i.e., the diagonal of the matrix is missing).

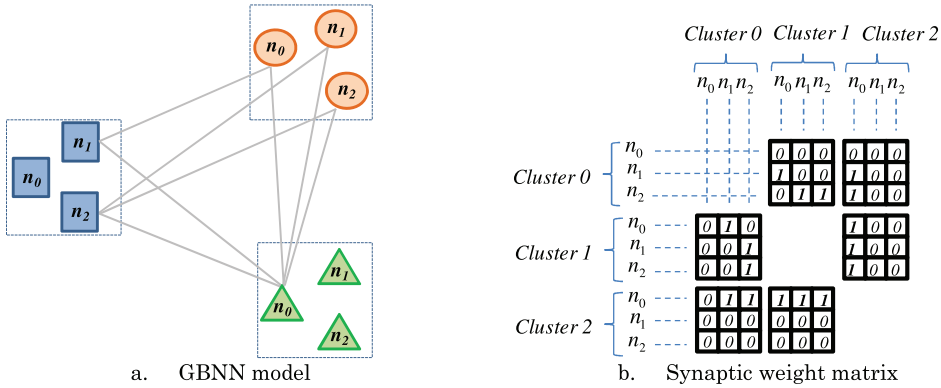


Fig. 1. Pedagogical example.

For example, the message $(n_{1,0}, n_{0,1}, n_{0,2})$ is memorized in the matrix through weights $w_{(1,0)(0,1)} = w_{(1,0)(0,2)} = w_{(0,1)(1,0)} = w_{(0,1)(0,2)} = w_{(0,2)(1,0)} = w_{(0,2)(0,1)} = 1$.

Next, if a partial message $(\_, n_{1,1}, n_{0,2})$ is presented to the network, where _ denotes a missing symbol (i.e., submessage associated to cluster *0* is missing), then the network must take a decision. The values of the known neurons are first activated (i.e., the values of the neurons associated to known submessages are set to 1) and the values of all the neurons are then broadcasted through the network. At the end of the scoring step, neurons $n_{1,0}$ and $n_{2,0}$ in cluster c_0 have a score of 1 and 2, respectively. Indeed, neuron $n_{2,0}$ receives two non-null values since it is linked to two active neurons (i.e., neurons $n_{1,1}$ and $n_{0,2}$) while neuron $n_{1,0}$ receives only one non null-value (from neuron $n_{0,2}$). Hence, at the end of this iteration, neuron $n_{2,0}$ will be selected as the activated neuron by the Winner Take All algorithm.

As shown in Gripon and Berrou [2012], the original scoring function presented in Eq. (1) may provide wrong answers. Indeed, a cluster with an ambiguous selection (i.e., several neurons selected) has potentially a more important impact on the scoring process than that of an unambiguous cluster. This drawback can be easily explained from our pedagogical example. Let us consider that the partial message $(\_, \_, n_{0,2})$ is presented to the network. At the end of the first retrieving iteration, neurons $n_{1,0}$, $n_{2,0}$, $n_{0,1}$, $n_{1,1}$, $n_{2,1}$ and neuron $n_{0,2}$ are selected for the next iteration. Then, during the second iteration neuron, $n_{2,0}$ receives three inputs, meaning that this neuron is linked to three activated neurons (i.e., neurons $n_{1,1}$ and $n_{2,1}$; neurons $n_{0,2}$) while neuron $n_{1,0}$ that belongs to the same cluster receives only two inputs. Hence, at the end of the second iteration, neuron $n_{2,0}$ is selected by the Winner Take All algorithm. This arbitrary choice is the consequence of the retrieving process and leads in this case to a wrong answer since there is no nonambiguous answer when the network is stimulated only with neuron $n_{0,2}$.

Gripon and Berrou [2012] propose to modify Eqs. (1) and (2), respectively, into Eqs. (3) to ensure that the score of a neuron will not be incremented if it receives two or more signals from the same cluster and (4) to ensure that the value obtained by the Winner Take All algorithm in a given cluster will never exceed C-1 when all the neurons in clusters corresponding to missing symbols are selected during the initial iteration.

$$\forall i \in [0..L-1], j \in [0..C-1],$$
$$s_{n_{i,j}}^{t+1} \leftarrow \sum_{g=0}^{C-1} \max_{0 \le k \le L-1} \left(v^t(n_{k,g}) * w_{(k,g)(i,j)} \right) \quad (3)$$

$$v_{(n_{i,j})}^{t+1} = \begin{cases} 1 & \text{if } s_{n_{i,j}}^{t+1} = C-1 \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

In Gripon and Berrou [2011, 2012], GBNN has been shown to be more efficient than the existing recurrent clustered-neural networks in terms of diversity (number of stored words), capacity (total number of stored bits), efficiency (ratio of the number of stored data to the number of total storage data) or error rate for the same memory used.

The network density refers to the ratio between the number of messages stored and the maximum number of messages that can be stored. A density close to 1 would lead to an overloaded network, which cannot retrieve learnt messages. On the contrary, a density close to 0 corresponds to a network that contains only a few cliques and that is thus able to retrieve messages even in the presence of strong erasures.

Considering that the messages to learn are uniformly distributed, the density of the network after the learning of M, M $\gg$ C messages, is close to

$$d = 1 - \left(1 - \frac{1}{L^2}\right)^M \approx \frac{M}{L^2} \, for M \ll L^2. \tag{5}$$

Given this density, the probability P_e that the correct message will be retrieved, if c_e out of the C clusters are not provided with information, is

$$P_e = 1 - (1 - d^{C-c_e})^{(L-1)c_e}. \tag{6}$$

Given this probability, the diversity $\mathrm{M_{max}}$ of the network, that is, the number of messages that the network is able to learn and retrieve is given by

$$M_{\max}(P_e) = \frac{\log\left(1 - \left(1 - (1 - \mathrm{P_e})^{\frac{1}{(\mathrm{L}-1)\mathrm{c_e}}}\right)^{\frac{1}{\mathrm{C}-\mathrm{c_e}}}\right)}{\log\left(1 - \frac{1}{\mathrm{L}^2}\right)}. \tag{7}$$

Finally, given the diversity, the capacity of the network is defined as

$$Cap = C.\log_2(L).M_{max}(P_e). \tag{8}$$

For instance, targeting an error rate of 10^{-2} one can learn up to 1.5×10^4 messages of length 64 ($C = 8$ and $L = 256$) using a GBNN, which corresponds to 9.6×10^5 bits learnt. Compared to Hopfield neural network with the same amount of memory used (this corresponds to 800 neurons and 50 learnt messages) and half its input bits erased, this represents an increase by a factor of 300-in diversity and 24-in capacity, for the same error probability. More generally, it has been demonstrated that the diversity increases as the square of the number of neurons for GBNN, whereas the law for a HNN is only sub-linear. More details and complete formula can be found in Gripon and Berrou [2011, 2012].

2.2. Hardware Architecture

Jarollahi et al. [2012] introduced a fully parallel hardware implementation of the GBNN proposed in Gripon and Berrou [2011] as an associative memory. The main blocks that compose a cluster are a Learning Module and a Decoding Module. The learning module stores the weights between every couple of neurons, that is, $L^{2^k*}(C-1)$ values since each neurons can be connected to $L^*(C-1)$ distant neurons and each cluster contains L neurons. The decoding module realizes the scoring and the WTA steps.

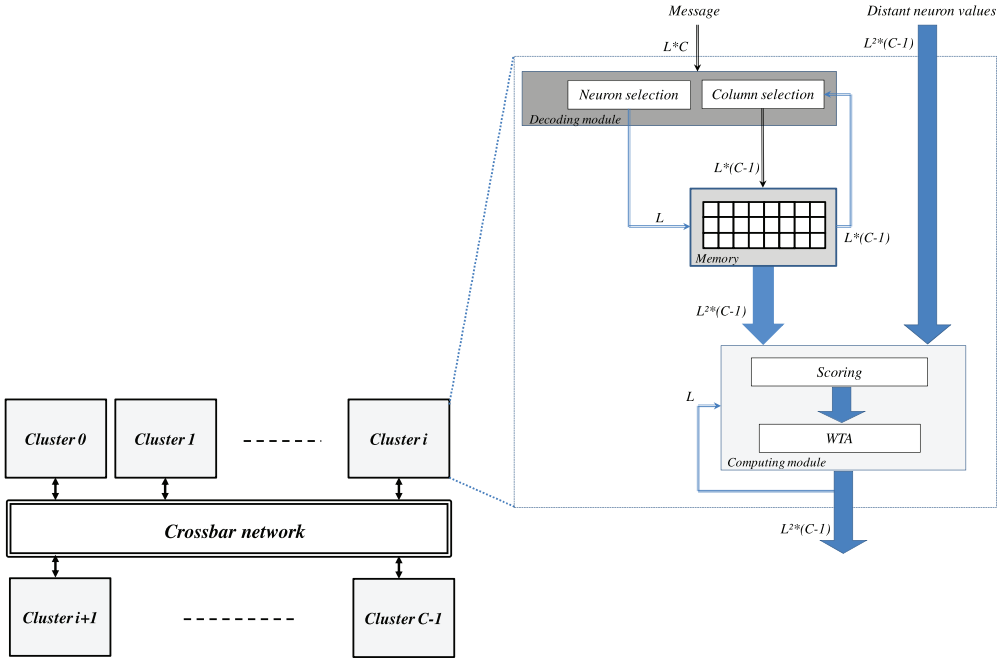


Fig. 2. GBNN architecture.

Figure 2 presents a simplified GBNN architecture with its main components: a set of clusters of neurons decomposed in three modules (i.e., a decoding module, a memory module storing the synaptic weights and a computing module to process the WTA algorithm) and a crossbar interconnection network dedicated to the interchanges of neurons values between the clusters.

During the learning process, each cluster receives the K-bit binary word to be stored in the associative memory. The decoding module first splits this input word in C subwords. Then, the subword corresponding to the local cluster is used to determine which neuron of this cluster is involved in the message, that is, which neuron must be activated (i.e., the selection of the line in the memory of the local cluster). The remaining subwords are used to determine which distant neurons (i.e., from distant clusters) must be connected with the locally activated neuron (i.e., column selection in the local cluster memory). Then, in order to store the clique, the memory is updated with the corresponding selected synaptic weights.

During the retrieving process, each local neuron receives the values of all distant neurons (i.e., $L^{2-k}*(C-1)$ binary values) which are used with the $L^{2^k}*(C-1)$ local synaptic weights to process the Scoring step as described in Eq. (1). Then, at the end of the scoring step, the WTA step elects a neuron, or a group of neurons as described in Eq. (2). Local neuron values are updated with this new information and broadcasted to all distant neurons of the GBNN.

Based on this architecture, a GBNN composed of $C = 8$ clusters, with $L = 16$ neurons each, has been implemented on an Altera Stratix IV FPGA. The authors found that logic dedicated to computations (scoring and WTA) and learning circuitry represents more than 2/3 of the total area, and the rest is for storage elements. The memory required to store the binary weights accounts for the major part of the total memory usage, that is, more than 90%. As we will show later, due to computation complexity and memory

footprint, the maximum size of the network that can fit into the target FPGA contains 16 clusters of 16 neurons each, that is, 256 neurons. In this architecture, at each cycle each neuron accesses to 240 weights and broadcasts its value to the 240 neurons which do not belong to its cluster.

This means that 61440 (i.e., $C^*L^{2*}(C-1)$) binary weights must be accessed concurrently at each cycle and 61440 additional signals are required to exchange neurons' values leading to place and route issues.

2.3. Discussion

As described in the previous subsections, the GBNN learning process creates neural cliques, where each neuron in the clique belongs to a different cluster. GBNN model strongly enhances performance of associative memories compared to Hopfield networks. Unfortunately, straightforward GBNN implementation leads to complex hardware architectures whose area and timing performances do not scale well with the size of the networks. In other words, this means that complex networks can be designed on ASIC at a prohibitive cost or that only very small networks can be designed by using large and thus expensive FPGA. In the following section, we introduce optimizations of this model following three principal axes: memory, computation, and communication. Hence, in order to optimize the complexity of the retrieving process, the traditional GBNN model is first transformed into a full binary one which allows both simplifying scoring computation and removing WTA step and thus to reduce the area. Second, we propose to memorize only half of the synaptic weights which allows reducing the number of storage elements and the cost of the learning logic. Finally, to further reduce the area, we serialize the communications. All the enhancements proposed in this article aim at easing the process realized by the neurons and also optimizing hardware implementation while keeping the functionality and the performances (i.e., capacity, density...) of the original GBNN model identical. As it will be shown in the experiments, coupling these optimizations leads to the design of smaller hardware architectures for a given GBNN size which allow to scale-up and to implement larger associative memories.

3. PROPOSED SIMPLIFIED NEURAL NETWORK

3.1. Full Binary Computation

GBNN models proposed in Gripon and Berrou [2011, 2012] rely on binary connections between neurons of different clusters and arithmetical computations to calculate the values of neurons and to realize the winner-take-all algorithm (see formulas (1), (2) and (3), (4) in the previous section). We propose to replace all the arithmetical-integer computations by logical equations, that is, to define a full binary neural network model. This property, which has first been introduced in Chavet et al. [2012] and later partially used to design content addressable memories in Jarollahi et al. [2013], is detailed in this section. The proposed fully binary model allows removing both the WTA step and achieving the same performances as the enhanced GBNN model [Gripon and Berrou 2012].

During the learning process, each new message to learn leads to record a new neural clique. As in the original models, memorizing a clique in a fully binary model means storing the connections between neurons of the clique itself (i.e., the values "1" in the corresponding memorizing elements). However, the message retrieval described in Gripon and Berrou [2011] is modified to take advantage of the fact that a neuron can be active only if it potentially belongs to a clique, that is, is connected to at least one active distant neuron in each active cluster. A cluster is said to be active if it has at least one active neuron.

The main principle is the concept of the "unanimous vote": a neuron $n_{i,j}$ is active in a given cluster j (i.e., $\vartheta(n_{i,j}) =$ "1"), if at least one active neuron in each other active cluster (i.e., "distant active neurons"), indicates that it is connected with neuron $n_{i,j}$ (i.e., during the learning process a connection with $n_{i,j}$ has been recorded for each of these distant neurons).

This approach changes the design of the decoding module. Instead of performing a WTA based on integer values, it is possible to only use Boolean equations to compute the value of each neuron without loss of generality. Then, Eqs. (1) and (3) are not useful anymore and can be replaced by the Boolean equation (9). The unanimous vote is represented by using the conjunction of all the votes.

$$\vartheta^{t+1}_{(n_{i,j})} = \bigwedge_{k=0,k\neq i}^{C-1} \left(\left(\bigvee_{g=0}^{L-1} \vartheta^{t}_{(n_{k,g})} \wedge w_{(i,j)(k,g)} \right) \cdots \right). \tag{9}$$

In Eq. (9), $\vartheta^{t+1}_{(n_{i,j})}$ represents the value $\vartheta(n_{i,j})$. at time $t+1$ of neuron $n_{i,j}$. It is obtained as the logical conjunction of the value of all others neurons $\vartheta^{t}_{(n_{k,g})}$ connected to $n_{i,j}$ ($w_{(i,j)(k,g)}$) for all distant clusters ($k = [0..C-1]$ and $k \neq i$), if these neurons are active. Then, additional elements must be added: as long as a given distant cluster is not active (the corresponding part of the input message is missing), its participation to the vote needs to be neutralized in order to avoid undesired side effects. This is called *transparency* (cf. final Equation (10)).

$$\vartheta^{t+1}_{(n_{i,j})} = \bigwedge_{k=0,k\neq i}^{C-1} \left(\left(\bigvee_{g=0}^{L-1} \vartheta^{t}_{(n_{k,g})} \wedge w_{(i,j)(k,g)} \right) \vee \left(\overline{\bigvee_{g=0}^{L-1} \vartheta^{t}_{(n_{k,g})}} \right) \right). \tag{10}$$

This last constraint is expressed in (10) by the last term of the expression (i.e., $(\overline{\bigvee_{g=0}^{L-1} \vartheta^{t}_{(n_{k,g})}})$). It should be noted that all the sums and the products of the original formula have been removed. Furthermore, the WTA disappears explicitly (compare and select), because each neuron realizes it implicitly (vote).

As an example, let us suppose that the GBNN presented in Figure 1(a) is formalized as a fully binary neural network; the synaptic matrix (cf. Figure (b)) remains the same. If a partial message $(\_, n_{1,1}, n_{0,2})$ is presented to the network, the network must take a decision. The values of the known neurons are set to "1" and others remain "0". Then in an active cluster (e.g., cluster c_2), the transparency part of the equation will result in value "1", so inactive clusters (e.g., cluster c_0) will not impact the evaluation of active clusters:

$$\overline{\vartheta^{t0}_{(n_{0,0})} \vee \vartheta^{t0}_{(n_{1,0})} \vee \vartheta^{t0}_{(n_{2,0})}} = \overline{0 \vee 0 \vee 0} = 1.$$

In an inactive cluster (in our example, cluster c_0), the logical Equation (10) is able to determine which neuron must be activated. In this case, since all the distant clusters are active, then the corresponding transparency bit results in value "0".

Logical Equation (10) is applied for all the neurons of inactive cluster c_0 of our example.

Hence, we obtain:

$$\begin{aligned}
\vartheta^{t1}_{(n_{0,0})} &= \left(\left(\left(\vartheta^{t0}_{(n_{0,1})} \wedge w_{(0,0)(0,1)}\right) \vee \left(\vartheta^{t0}_{(n_{1,1})} \wedge w_{(0,0)(1,1)}\right) \vee \left(\vartheta^{t0}_{(n_{2,1})} \wedge w_{(0,0)(2,1)}\right)\right) \vee 0\right) \\
&\quad \wedge \left(\left(\left(\vartheta^{t0}_{(n_{0,2})} \wedge w_{(0,0)(0,2)}\right) \vee \left(\vartheta^{t0}_{(n_{1,2})} \wedge w_{(0,0)(1,2)}\right) \vee \left(\vartheta^{t0}_{(n_{2,2})} \wedge w_{(0,0)(2,2)}\right)\right) \vee 0\right) \\
&= \left(\left((0 \wedge 0) \vee (1 \wedge 0) \vee (0 \wedge 0)\right) \vee 0\right) \wedge \left(\left((1 \wedge 0) \vee (0 \wedge 0) \vee (0 \wedge 0)\right) \vee 0\right) \\
&= 0 \\
\vartheta^{t1}_{(n_{1,0})} &= \left(\left(\left(\vartheta^{t0}_{(n_{0,1})} \wedge w_{(1,0)(0,1)}\right) \vee \left(\vartheta^{t0}_{(n_{1,1})} \wedge w_{(1,0)(1,1)}\right) \vee \left(\vartheta^{t0}_{(n_{2,1})} \wedge w_{(1,0)(2,1)}\right)\right) \vee 0\right) \\
&\quad \wedge \left(\left(\left(\vartheta^{t0}_{(n_{0,2})} \wedge w_{(1,0)(0,2)}\right) \vee \left(\vartheta^{t0}_{(n_{1,2})} \wedge w_{(1,0)(1,2)}\right) \vee \left(\vartheta^{t0}_{(n_{2,2})} \wedge w_{(1,0)(2,2)}\right)\right) \vee 0\right) \\
&= \left(\left((0 \wedge 1) \vee (1 \wedge 0) \vee (0 \wedge 0)\right) \vee 0\right) \wedge \left(\left((1 \wedge 1) \vee (0 \wedge 0) \vee (0 \wedge 0)\right) \vee 0\right) \\
&= 0 \\
\vartheta^{t1}_{(n_{2,0})} &= \left(\left(\left(\vartheta^{t0}_{(n_{0,1})} \wedge w_{(2,0)(0,1)}\right) \vee \left(\vartheta^{t0}_{(n_{1,1})} \wedge w_{(2,0)(1,1)}\right) \vee \left(\vartheta^{t0}_{(n_{2,1})} \wedge w_{(2,0)(2,1)}\right)\right) \vee 0\right) \\
&\quad \wedge \left(\left(\left(\vartheta^{t0}_{(n_{0,2})} \wedge w_{(2,0)(0,2)}\right) \vee \left(\vartheta^{t0}_{(n_{1,2})} \wedge w_{(2,0)(1,2)}\right) \vee \left(\vartheta^{t0}_{(n_{2,2})} \wedge w_{(2,0)(2,2)}\right)\right) \vee 0\right) \\
&= \left(\left((0 \wedge 0) \vee (1 \wedge 1) \vee (0 \wedge 1)\right) \vee 0\right) \wedge \left(\left((1 \wedge 1) \vee (0 \wedge 0) \vee (0 \wedge 0)\right) \vee 0\right) \\
&= 1.
\end{aligned}$$

At the end of the first iteration, the values of neurons $n_{0,0}$ and $n_{1,0}$ are evaluated to be "0" and the value of neuron $n_{2,0}$ is evaluated to be "1". Neuron $n_{2,0}$ is thus selected and the network provides the right answer.

3.2. Reduced Memory

In order to determine if a given neuron is active during the retrieving process, it is necessary to check if it is connected with already known active neurons in distant clusters. To do so, it is proposed in the original GBNN models [Gripon and Berrou 2011, 2012] that each cluster learns and stores for each of its neurons the synaptic weights (i.e., w in Eqs. (1) and (3)) corresponding to the connections a neuron has with all the neurons that belong to distant clusters. Hence, to learn that a given connection exists between two neurons $n_{i,j}$ and $n_{k,g}$ (i.e., neuron i from cluster j and neuron k from cluster g with $j \neq g$), two weights must be stored: first in cluster j where $n_{i,j}$ records it is connected to $n_{k,g}$ (i.e., $w_{(i,j)(k,g)} = 1$) and next in the cluster g where $n_{k,g}$ records it is connected to $n_{i,j}$ (i.e., $w_{(k,g)(i,j)} = 1$). However, these two weights are always identical since the corresponding connection is symmetrical. This property can be observed in Figure 1(b) where the upper-right blocks of the synaptic matrix are the transpose of the lower-left ones. It is thus not required to store twice the same values.

This optimization supposes that synaptic matrices are shared between clusters. However, storing only one synaptic weight per pair of neurons allows dividing the amount of memory by two without any performances loss. Besides, in order to store synaptic values into the memory, learning logic resources are necessary (decoders, accumulators, multiplexers$\cdots$). Memorizing half the information removes these computing resources and then further reduces the cost of the architecture. The optimized synaptic matrix will be referred as *triangular matrix* in the rest of the paper. Figure 3 illustrates for each cluster of our example the locations that weights have once the triangular matrix is used.

3.3. Serialized Communication

A fully parallel GBNN implementation, as proposed in Jarollahi et al. [2012], requires a huge amount of wires to connect all the neurons and a large amount of computing logic to process data concurrently. Indeed, in the GBNN model, a neuron $n_{i,j}$ from a given

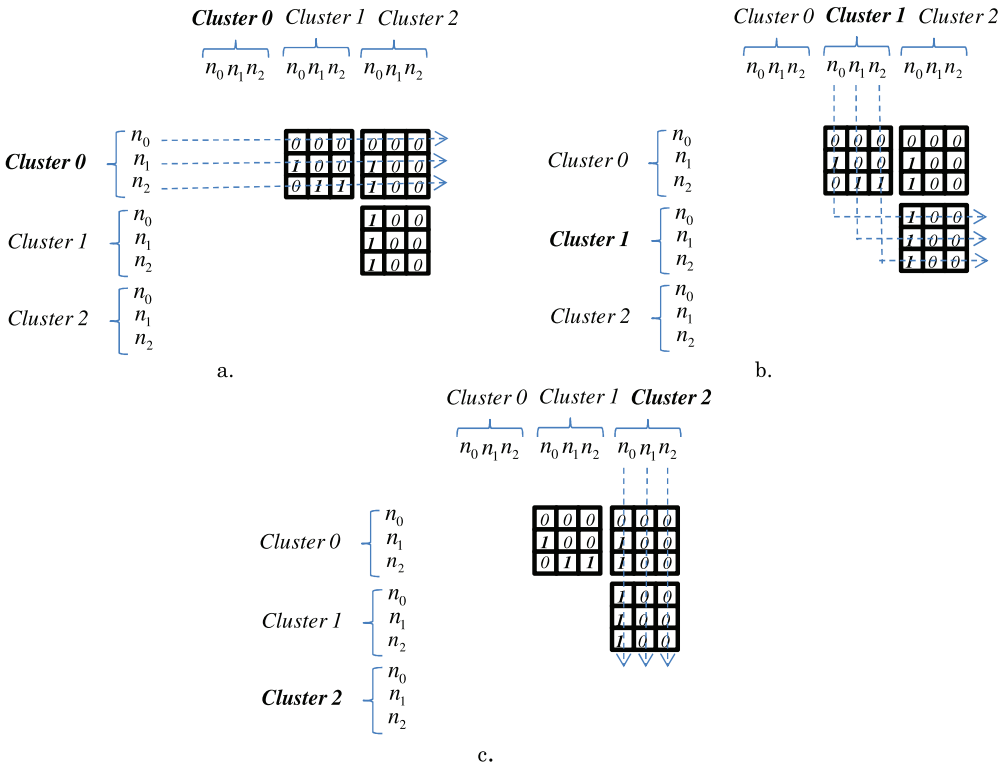


Fig. 3. Synaptic weight locations in triangular matrix.

cluster j must be connected to all other neurons of all distant clusters. So, each neuron $n_{i,j}$ is connected to $L^*(C-1)$ neurons. Since each cluster gathers L neurons, a cluster is connected to $L^{2*}(C-1)$ neurons. Finally, if we consider all the clusters, the total number of data exchanged and processed concurrently is $C^*L^{2*}(C-1)$. In order to reduce the complexity of both the interconnection network and the computing logic, data transfers must be serialized (i.e., at each cycle the amount of concurrently exchanged information must be reduced) and computations must be streamed (i.e., processing elements can be reused across cycles). Serialization leads to considerable area reduction that enables to scale-up complex GBNN architectures However, serialization of communications increases the latency of an iteration. Nevertheless, if serialization is smartly designed and implemented it can provide higher clock frequencies which finally can limit its impact on the timing performance (see the results in Section 4.3).

Serialization can be either cluster-based or neuron-based. In a cluster-based scheme, clusters take turns to broadcast the value of all their neurons. In a neuron-based scheme, the clusters broadcast concurrently the values of one of their neurons (i.e., the values of neurons that belong to the same cluster are successively broadcasted). In all the cases, a cluster can compute partially the values of its neurons while receiving the current values of distant neurons. These local neuron values are updated by sequentially using neuron values originating from distant clusters. Computations end when all distant neuron values have been received and local values are then defined. In the cluster-based scheme a single iteration will take C cycles to complete, while it will take L cycles in the neuron-based scheme.

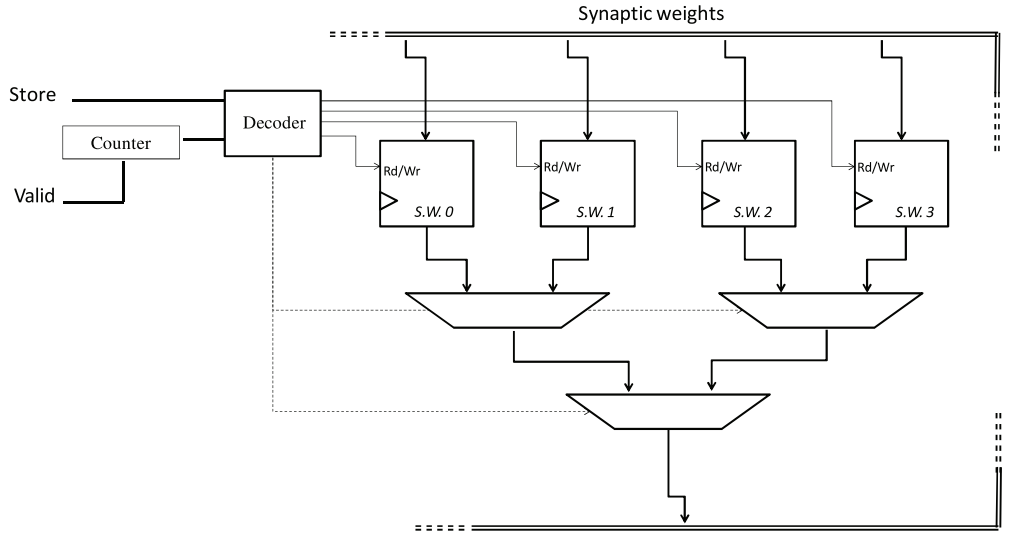


Fig. 4. Steering logic overhead.

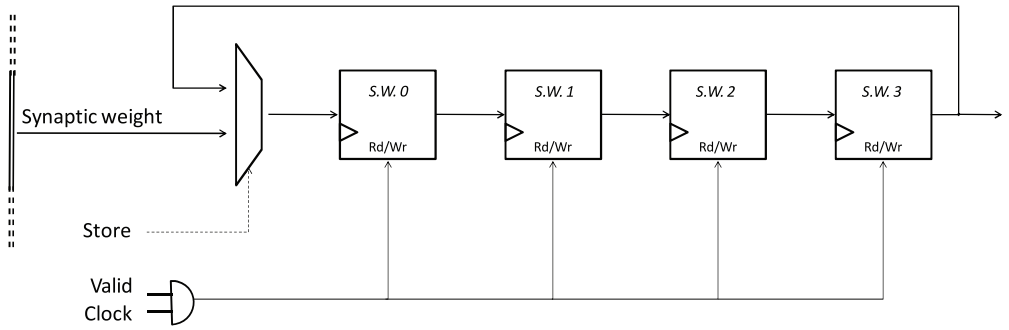


Fig. 5. Flip-flop ring.

Figure 4 depicts an architecture where four buffers storing synaptic weights (i.e., *s.w.*0, *s.w.*1, *s.w.*2, *s.w.*3) are used and where the required synaptic weight is selected thanks to a set of multiplexers during the retrieving process. During the learning step (respectively, retrieving) the input (resp. output) data (i.e., synaptic weights) are sequentially stored in (respectively, selected from) buffers whose order is given for example by a counter (the counter runs according to the *Valid* signal). As it will be shown in the experiments section, the drawback of this design comes from the additional multiplexers and associated control logic that lead to unacceptable area overhead.

In order to avoid this penalty, we propose a mechanism named *flip-flop ring*. Flip-flop ring is a *torus*-connected shift register (see Figure 5). The size of the ring equals the number of cycles needed to transfer information (i.e., C *or* L) depending on the selected serialization (i.e., cluster or neuron, respectively). Hence, groups of synaptic weights are no more stored in a set of independent registers but rather in flip-flop rings. The output of the ring is connected to the computing logic removing all additional circuitry.

In this architecture, four buffers are still used to store synaptic weights. However only one (2:1) multiplexer is required instead of the $L - 1$ (2:1) multiplexers of the previous architecture (see Figure 4). Hence, during the learning step (the step during

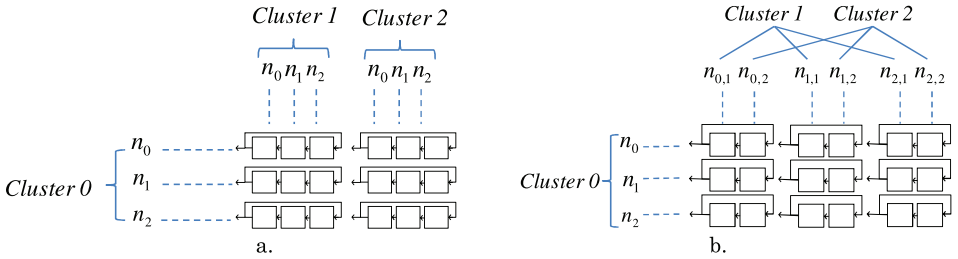


Fig. 6. Flip-flop ring for squared synaptic matrix and (a) neuron-based or (b) cluster-based serialization.

which cliques, that is, synaptic weights are stored in the memory), the input data is stored in the first buffer and the other data are shifted in the ring. During the retrieving step, data are shifted and output data is always read from the last buffer. After $L = 4$ cycles, the ring comes back in its initial state. The *Valid* signal is used to activate the ring.

Figures 6(a) and 6(b) illustrate how flip-flop rings are used respectively for neuron-based and cluster-based serialization when considering squared synaptic weight matrix previously presented in Figure 1. From Figure 6(a), wherein only connections between cluster c_0 and distant clusters are shown, it can be observed that after i cycles of the retrieving process, all the clusters (see rows) have access to all the synaptic weights of the connections their local neurons share with the ith neuron of distant clusters. From Figure 6(b), wherein only connections between cluster c_0 and distant clusters are shown, it can be observed that after i cycles of the retrieving process, each cluster will have access to all the synaptic weights of the connections their local neurons share with all the neurons of the ith distant clusters.

Flip-flop rings can also be combined with triangular synaptic weight matrices. However, data must be mapped in such a way to avoid any additional complex steering logic. Indeed, triangular matrix is shared among clusters and clusters access two at a time to the same synaptic weights. In a neuron-based serialization, this behavior can be observed from Figure 3(a) that presents only weights shared by clusters c_0 and c_1. Let us consider the cycle during which all the clusters broadcast the value of their neuron n_0. Then, cluster c_0 accesses to all the weights that are related to distant neurons n_0, that is, the first column in Figure 3(a). Concurrently, cluster c_1 also accesses to all the weights that are related to distant neurons n_0, that is, the first line in Figure 3(a). Shifting data horizontally would be correct for c_0 but wrong for c_1 while shifting data vertically would be correct for c_1 but wrong for c_0.

The solution consists in mapping the data in flip-flop rings such that at each cycle each cluster of a given pair can access to the right data at the right time. In this way, all clusters can access to their data concurrently by reading always at the same location to avoid any additional steering logic. For that purpose, let us consider two L^*L 2D-arrays named *WEIGHT* and *RING*. WEIGHT represents the matrix of synaptic weights shared by two clusters and RING represents the set of L flip-flop rings of L cells each. Let us consider two indexes $N, M \in [0..L-1]$. Then, by applying Eq. (11), it is possible to compute where data must be mapped in flip-flop ring cells.

$$RING(M, N) = WEIGHT(M + N \bmod L, N) \tag{11}$$

This equation allows shifting by N rows the data of the Nth column of WEIGHT array to map them in the RING array. For the sake of simplicity, Figure 7(a) identifies by indexed letter x_a the weights shared between neurons of clusters c_0 and c_1. Figure 7(b) illustrates the mapping obtained by applying (11) in order to combine flip-flop rings and neuron-based serialization. For example, $RING(0,2) = WEIGHT(2,2) = x_8$. It can

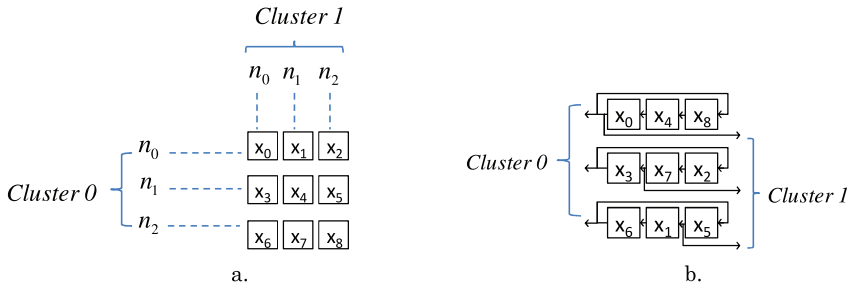


Fig. 7. Neuron-based serialization and triangular matrix: (a) synaptic weights and (b) flip-flop ring.

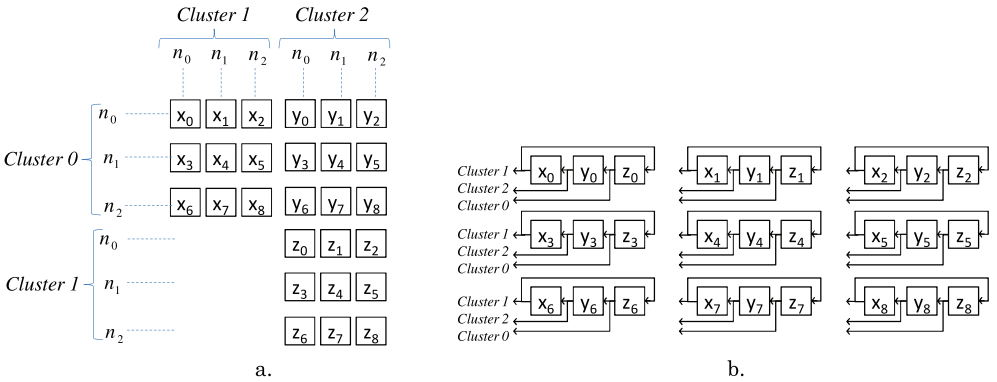


Fig. 8. Cluster-based serialization and triangular matrix: (a) synaptic weights and (b) flip-flop ring.

be observed that after i cycles of the retrieving process, clusters c_0 (respectively, c_1 will have access to all the synaptic weights of the connections its local neurons share with the ith neuron of cluster c_1 (respectively, c_0).

When considering cluster-based serialization, flip-flop rings must gather synaptic weights from different pairs of clusters. Figure 8(a) illustrates a triangular matrix where weights are identified by indexed letters x, y, and z. Figure 8(b) depicts the resulting set of flip-flop rings which allows, after i cycles of the retrieving process, that each cluster accesses to all the synaptic weights it shares with the neurons of the ith distant cluster.

4. EXPERIMENTS

The interest of the proposed optimizations is shown through several experiments where both original and proposed models, as well as the associated architectures are compared.

A first set of theoretical studies aims to show that our full binary model improves the performance of the original GBNN [Gripon and Berrou 2011] and reaches the same performance as the enhanced model proposed in Gripon and Berrou [2012].

The second set of experiments focuses on hardware design. The state of the art and the proposed optimized architectures are compared. Based on a STMicroelectronics 90 nm technology library, a wide range of networks is explored and architecture complexities are analyzed.

The third and last set of experiments compares the synthesis results based on a Stratix IV FPGA and HardCopy platforms from Altera for different network architectures.

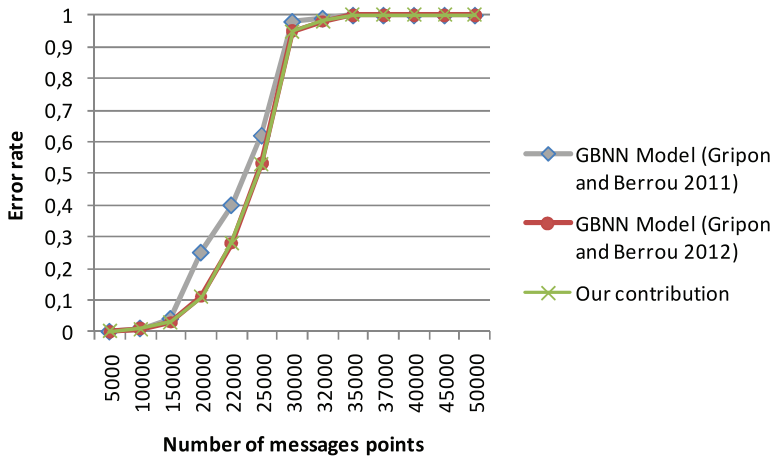


Fig. 9. Error-rate performance.

To allow for an easy reading of our charts, versioning concept has been used to name architectures. Hence:

—architectures based on the original GBNN model [Jarollahi et al. 2012] are referred as V0;
—architectures based on the fully binary model we proposed are referred to as V1.0;
—architectures based on binary model and triangular synaptic weight matrices are referred to as V1.1;
—architectures based on binary model and cluster-based serialization are referred as V1.2;
—architectures based on binary model and neuron-based serialization are referred as V1.3.

4.1. Retrieving Performance Analysis

In order to study and to compare retrieving performances of our fully binary model, we performed Matlab simulations. The messages to be learnt were randomly generated following a uniform discrete distribution. In this context, we derived the error retrieving rate in relation to the number of learnt messages when half the clusters have no information. These experiments have been performed with $C = 8$ clusters and $L = 256$ neurons (i.e., a total of 1024 neurons). Four retrieving iterations have been used as suggested in Gripon and Berrou [2011] to obtain the best performance. Simulations have been performed on a core i7 M620 2.7GHz, with 4GB RAM. Our results are compared with both the original GBNN model from Gripon and Berrou [2011] and from Gripon and Berrou [2012].

Figure 9 shows the remaining error rates in the "retrieved" messages depending on the number of learnt messages. As in Gripon and Berrou [2012], the total number of input messages used to stimulate the GBNN has been set to 2000. Each point in these curves represents the ratio between the number of erroneous retrieved messages and the total number of input messages. It can be observed that the GBNN model defined in Gripon and Berrou [2012] has better performance than the original GBNN model [Gripon and Berrou 2011]. Indeed, up to 20,000 message points the best networks are able to achieve 0% "retrieving" errors, while the performance of the original GBNN model start to be deprecated with 15,000 message points. Moreover, our proposed contribution natively achieves exactly the same performance as in Gripon and Berrou

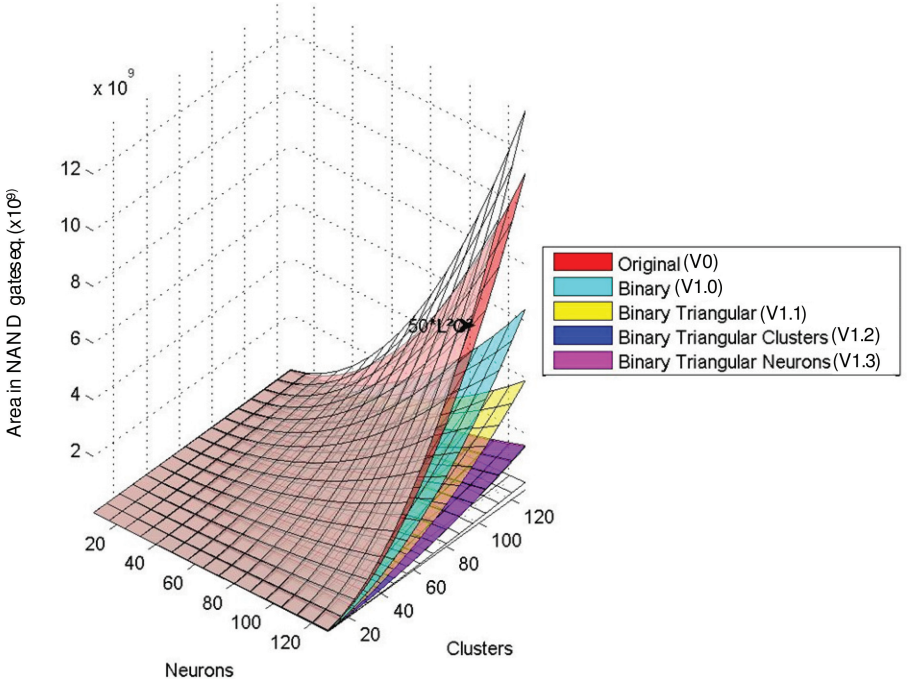


Fig. 10. Area for neural networks up to $C = 128$ and $L = 128$ (in NAND gate equation).

[2012] since the curves perfectly superimpose in Figure 9. This confirms that our fully binary model does not introduce any performance loss to the arithmetical-integer GBNN models.

4.2. Complexity Analysis

The second set of experiments explores a wide range of architectures in order to study the complexity breakdown offered by the optimizations we propose. The results are given in terms of NAND gate equation (based on STMicroelectronics 90-nm cell library) and the V0 architecture is used as baseline.

Figure 10 shows a first set of results in which both the number of clusters and the number of neurons are varied (i.e., C and L range from 1 up to 128). It should be noticed that Figure 10 includes an upper-bound surface (i.e., $(50^*L^{2*}C^2)$ in order to observe the limits of the design space more easily. Analysis of these results shows that all the proposed architectures are always greatly smaller than reference V0 architectures. Indeed, while simply using the fully binary model reduces the total area of the V1.0 architecture down to half the reference architecture, the combination of all our proposed enhancements pushes area optimization further: V1.1 reduces architecture complexity close to 1/3 of V0 (i.e., 70% of area reduction), while V1.2 and V1.3 reduce it to 1/6 (i.e., 83% of area reduction).

It should be noted that in Figure 10 the area of V1.2 and V1.3 are superimposed. However, if larger networks are explored (e.g., up to $C = L = 512$), it can be observed that the trends are different, and the cluster serial approach often reaches lower costs (cf. Figure 11).

The total resource costs presented in Figures 10 and 11 each decomposes into three parts: decoding, memorizing, and computing resources. These terms correspond to the three optimization axes we propose to design GBNN-based associative memories: full

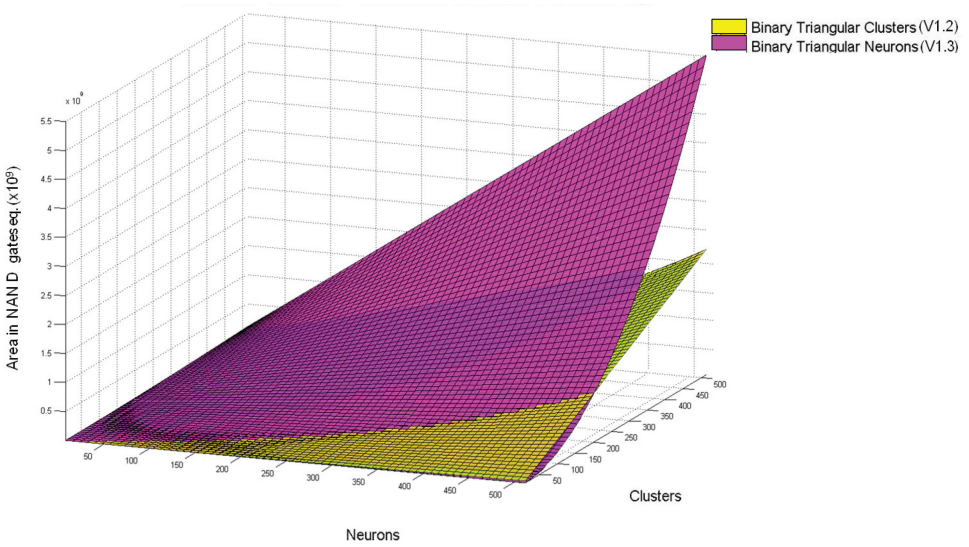
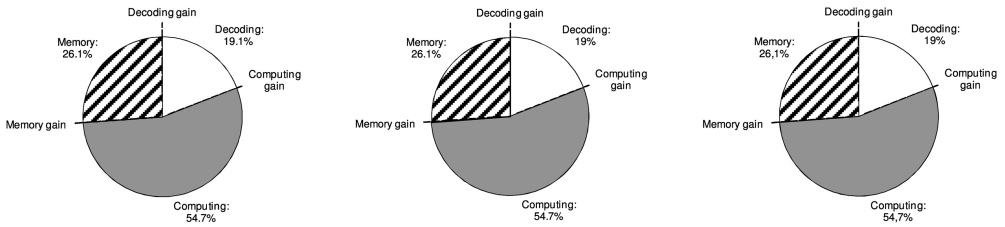


Fig. 11. Area for neural networks up to $C = 512$ and $L = 512$ (in NAND gate equation) for architecture V1.2 and V1.3.



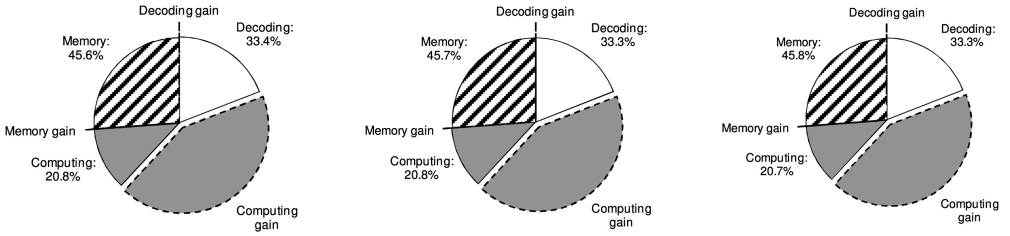
a. C= 256 L=64 b. C= 128 L=128 c. C= 64 L=256

Fig. 12. Repartition of resources in V0 architectures.

binary computation (cf. Computation in figures), memory reduction (cf. Memory and Decoding in figures) and serialized communication (cf. Decoding and Computation in figures). Full binary computation strongly reduces the cost of the computation module, memory reduction limits the cost of both the memory and the decoding modules, and, finally, serialization optimizes the computation and the decoding modules. Figures 12, 13, 14, 15, and 16 show the repartition of these three contributions for three configurations of the network namely 256 clusters/64 neurons, 128 clusters/128 neurons and 64 clusters/256 neurons. Memorizing resources are represented in hatched, decoding resources in white and computing resources in dark grey. Hence, Figure 12 depicts the repartition of resources in V0 and it can be seen that area mainly depends on the computing resources (54.7% of the total cost in average).

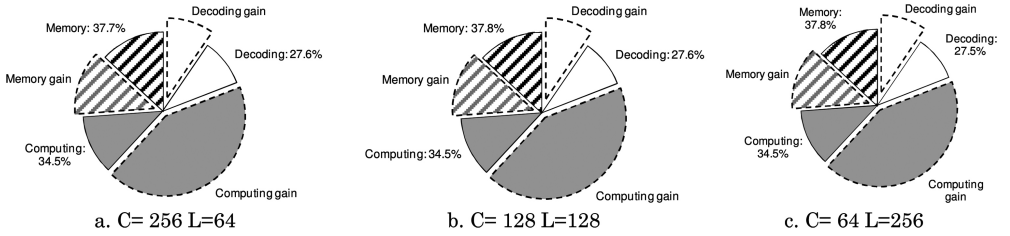
Figure 13 shows the repartition of resources in V1.0 and also mentions through dotted line partitions the savings we obtained against baseline results. It can be observed that the fully binary model allows greatly reducing the cost of computing resources. Computing resources roughly account for 20% of the total area in V1.0 (against 54.7% in average for V0) and almost half of the architecture is now composed by memorizing resources.

Figure 14 illustrates the repartition of resources in V1.1 architecture. Coupling triangular synaptic weight matrix to fully-binary model allows further reducing total



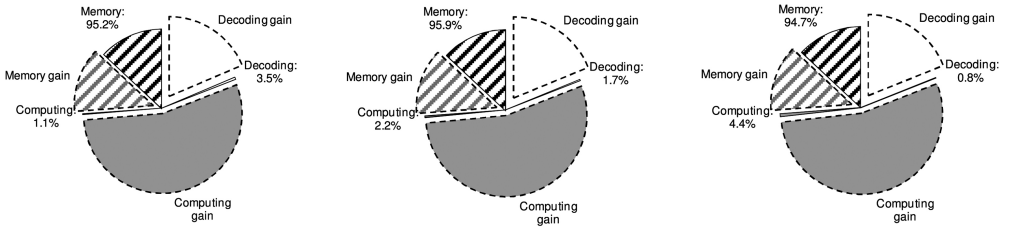
a. C= 256 L=64 b. C= 128 L=128 c. C= 64 L=256

Fig. 13. Repartition of resources in V1.0 architectures.



a. C= 256 L=64 b. C= 128 L=128 c. C= 64 L=256

Fig. 14. Repartition of resources in V1.1 architectures.



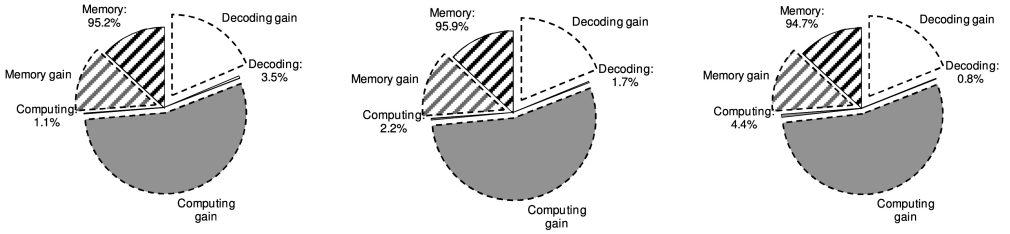
a. C= 256 L=64 b. C= 128 L=128 c. C= 64 L=256

Fig. 15. Repartition of resources in V1.2 architectures.

area. Hence, besides the savings obtained by using a fully binary model (i.e., V1.0), costs of decoding resources and memory have been reduced by almost 50%. In average, learning resources now account for 27% of the total area instead of 33.4% in V1.0 while memory resources account for 37.8% in average instead of 45.7% in V1.0.

In Figures 15 and 16, respectively, present the repartition of resources in V1.2 and V1.3 architectures. Results show impressive breakdown compared to V1.0 and V1.1 and even more compared to V0. This is the demonstration that serialization is able, not only to reduce the complexity of the interconnection network, but also to have a great impact on the total area through a limited amount of computing resources. Moreover, by coupling the proposed optimizations, memory resources represent about 95% of the total area meaning that neural network is now mainly composed of memory elements.

It can also be observed that if the number of clusters is greater that the number of neurons, in V1.2 architecture (i.e., binary cluster serial) the proportion of computing resources is lower than in V1.3 (i.e., binary neuron serial). On the contrary, if the number of clusters is lower than the number of neurons, in V1.2 architecture the proportion of computing resources is greater than in V1.3.



a. C= 256 L=64 b. C= 128 L=128 c. C= 64 L=256

Fig. 16. Repartition of resources in V1.3 architectures.

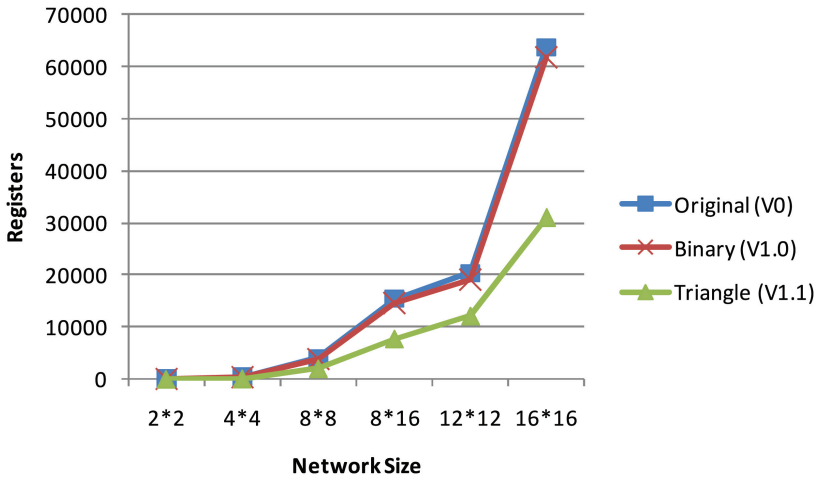


Fig. 17. FPGA register usage.

4.3. Synthesis Results

We have synthesized several architectures in order to compare our fully binary model, memory optimization and communication serialization to state of the art implementations of GBNN. Hardware architectures have been described in VHDL, synthesized on FPGA and ASIC technologies, validated and finally characterized in area and timing performances. Their error-rates have been evaluated with the same input vectors than for the theoretical analysis and they are strictly identical to the results presented in Figure 9.

4.3.1. FPGA Target. First, we have synthesized on a Stratix IV FPGA platform a set of architectures. To allow fair comparisons, all the syntheses have been done on the same target device than the one used in Gripon et al. [2012], that is, Stratix IV EP4SGX230KF40C2 FPGA. Different sizes of networks have been explored ranging from C*L = 2*2 up to C*L = 16*16. The test case C*L = 8*16 is the configuration used in Gripon et al. [2012]. We scaled the network complexity up to 256 neurons because baseline design V0 reaches the limits of the target FPGA for that size.

Figure 17 shows the scaling of registers usage while varying the number of neurons and clusters. It can be observed that the fully binary model V1.0 slightly requires less resources than the reference design V0 while triangular synaptic weight matrix V1.1 allows reaching a breakdown of 50% for all the configurations.

Figure 18 presents synthesis results in terms of LUT and highlights the interest of the proposed optimizations. Compared to reference design V0, in average area

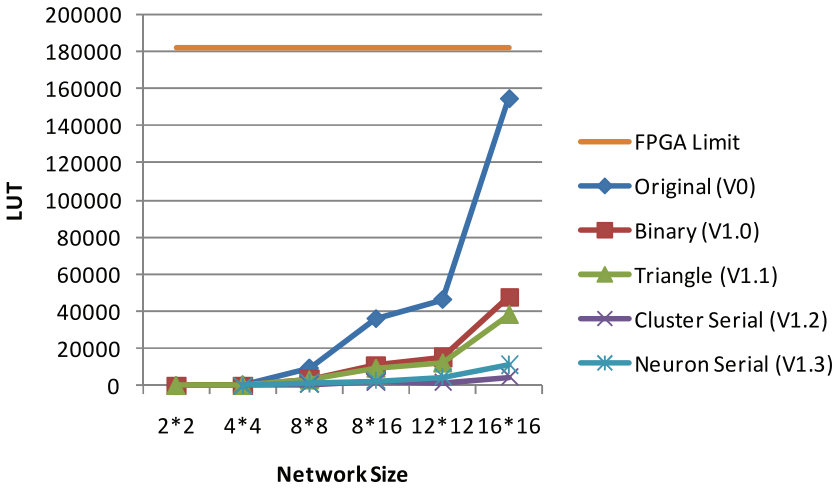


Fig. 18. FPGA LUT usage.

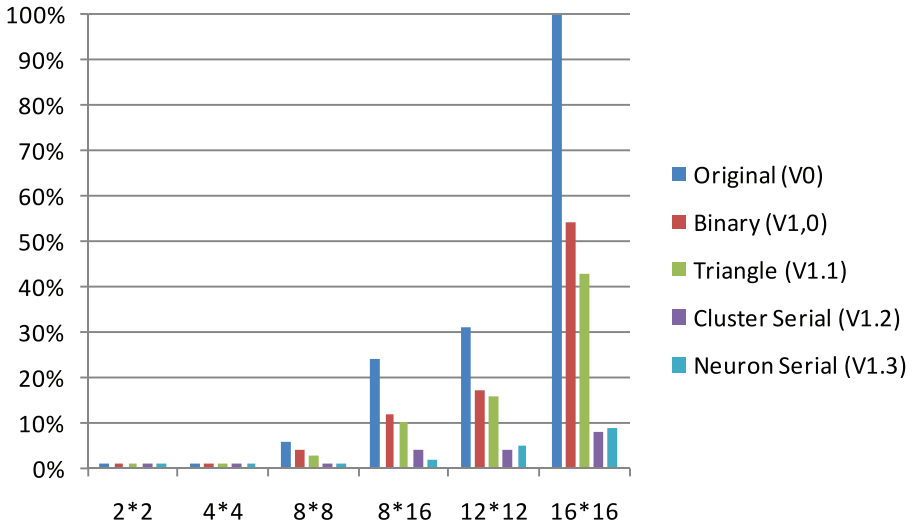


Fig. 19. FPGA resource usage.

reductions range from 62% for V1.0 up to 86% for V1.2. The larger is the network, the more important the reductions are. When considering a 16*16 configuration, area savings reach 76% and 92% for architectures V1.0 and V1.2, respectively.

Figure 19 sums up these results by presenting the total percentage usage of the FPGA ressources and confirms the general trend: the more the network grows, the more the gap between the proposed architectures and baseline increases. While V0 quickly reaches 100% of the targeted FPGA, our designs uses at most 8% of FPGA resources.

Figure 20 depicts evolution of maximum achievable clock frequencies. It can be observed that the clock frequencies of architectures decrease rather rapidly for the first smallest network complexities while performances fall slower for larger networks. Reference designs V0 decrease faster than the other architectures. When considering

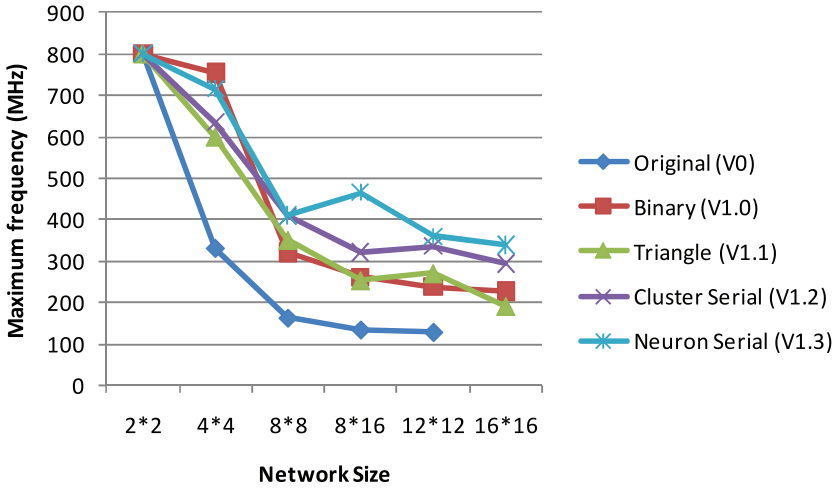


Fig. 20. Clock frequencies (FPGA).

the largest neural network, V1.3 gets the best performance over all the architectures with a clock frequency of 338.64 MHz. The better timing performances of serialized architectures allow counterbalancing to a certain extent their sequential behaviors.

Let us define the delay of a retrieving iteration as the number of cycles of one iteration divided by the clock frequency of the architecture. Then, the ratio of the delays of the reference design V0 and of our serialized approaches V1.2 and V1.3 can be computed. Experimental results show that this ratio is much lower than C (respectively, L) in case of cluster-based scheme (respectively, neuron-based scheme). As an example, for the most complex V0 architecture, that is, 12*12 network, our designs V1.2 and V1.3 should have been 12 times slower. However, ratio of the delays is equal to 4.5 for V1.2 and 4.2 for V1.3. This means that our architectures are respectively 2.6 and 2.8 times faster than expected. These results are obtained thanks to the proposed optimizations used to design the serialized architectures that allows to reach higher clock frequencies as well as reducing the area. However, it should be mentioned that our architectures have not been particularly designed to optimize timing performance, but rather area. As a consequence, even better performance could be obtained by considering this new design objective. Moreover, it can be observed that V0 has no result for a 16*16 network since this design cannot be "placed and routed" by the sysnthesis tools (all the FPGA ressources are already used). Hence, serialisation allows scaling up and implementing large hardware associative memories with quite good timing performance.

4.3.2. ASIC Target. ASIC technology has been targeted by using Altera HardCopy platform that maps all the resources used in the architecture onto HCELLS instead of LUT and registers. Figure 21 presents synthesis results in terms of HCELLS. Compared to reference design V0, area reductions range from 33% for V1.0 and up to 57% for V1.2, in average. Since reductions evolve according to the network complexity, area savings reach 56% and 87% for architectures V1.0 and V1.2, respectively, when considering a 16*16 configuration.

Figure 22 presents the evolution of maximum achievable clock frequencies. The maximum frequency of V0 still decreases faster than the other ones like V1.2 which while being better, follows the same trend. Apart from configurations 8*8 and 16*16, architectures V1.3, V1.0, and V1.1 often provide equivalent performance.

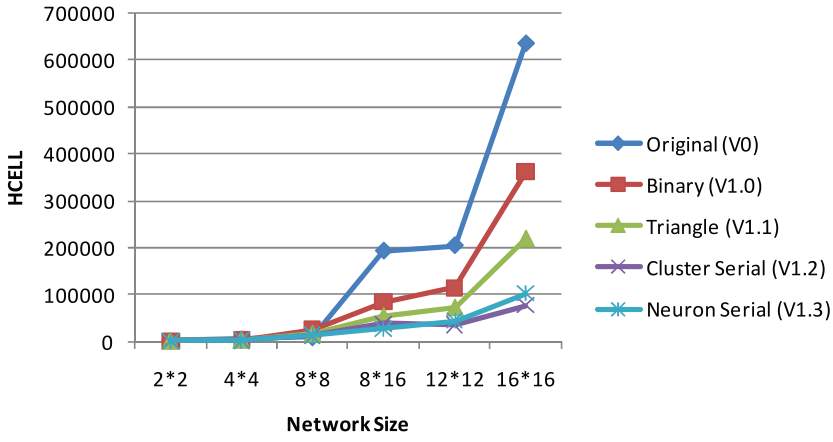


Fig. 21. Area results (HCELLS).

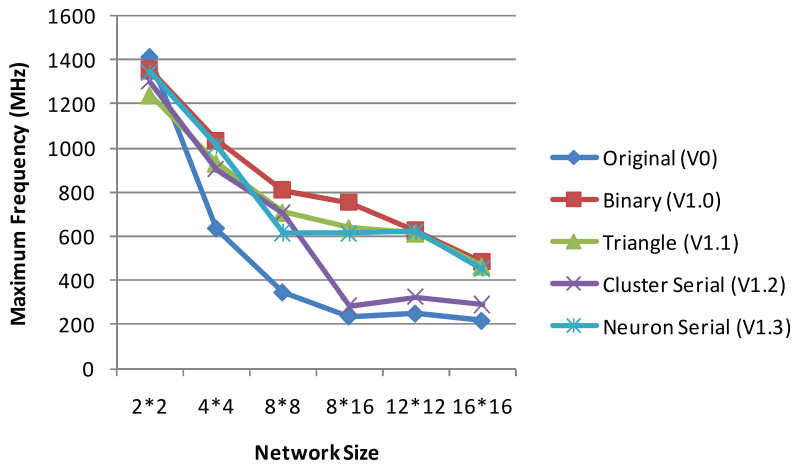


Fig. 22. Clock frequencies (ASIC).

All the synthesis results verify the trends provided by the complexity analysis and thus confirm the results our optimizations provide. In this case, if we compute the delay (as defined in the previous subsection) of one iteration for the serialized architectures it can be observed that as it has been shown targeting FPGA plateform, the impact of serialization is soften by the clock frequencies of the generated architectures. As an example, for the 16*16 network, this ratio is equal to 9.1 for V1.2 and 4.7 for V1.3, instead of 16. This results are still promising since, once again, our architectures have not been particularly designed to optimize timing performance.

5. CONCLUSION

In this article, we have presented optimizations to reduce the complexity of the original GBNN model and to allow for efficient hardware design. First, we have defined a fully binary neural network model that allows reducing computing resources. Then, we have proposed to store half of the information compared to the original GBNN, thus reducing the cost of storage elements and the complexity of the learning process. Finally, we introduced serial communications between neurons to allow further optimizing the

architecture complexity. We have shown that combining these optimizations is not straightforward. However, experimental results show impressive area breakdown and better clock frequencies compared to baseline architectures without any loss of neural network performances (capacity, efficiency, error-rate, etc.).

REFERENCES

L. F. Abbott and S. B. Nelson. 2000. Synaptic plasticity: Taming the beast. *Nature Neurosci.* 3. 6 pages.

D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. 1985. A learning algorithm for Boltzmann machines. *Cognit. Sci.* 9, 1, 147–169.

A. Annovi, G. Broccolo, A. Ciocci, et al. 2013. Associative memory for L1 track triggering in LHC environment. *IEEE Trans. Nucl. Sci.* 60, 5, part 2, 3627–3632.

R. Brette, M. Rudolph, T. Carnevale, et al. 2007. Simulation of networks of spiking neurons: A review of tools and strategies. *J. Computat. Neurosci.* 23, 349–398.

C. Chavet, P. Coussy, and N. Charpentier. 2012. Architecture de réseau de neurone, procédé d'obtention et programmes correspondants. Patent n° 1261155.

P. Frost Gorder. 2008. Computer vision, inspired by the human brain. *IEEE J. Comput. Sci. Eng.* 10, 2. DOI:10.1109/MCSE.2008.53

S. Furber and S. Temple. 2007. Neural systems engineering. *J. Roy. Soc. Inter.* 4.

T. Giotis, M. A. Christodoulou, and Y. Boutalis. 2011. Identification of combination therapy models using a neuro fuzzy identification scheme. In *Proceedings of the 19th Mediterranean Conference on Control & Automation*, 1283–1288.

R. Granger. 2006. Essential circuits of cognition: The brain's basic operations, architecture, and representations. In *AI at 50: The Future of Artificial Intelligence.*

V. Gripon and C. Berrou. 2011. Sparse neural networks with large learning diversity. *IEEE Trans. Neural Netw.* 22, 7, 1087–1096.

V. Gripon and C. Berrou. 2012. Nearly-optimal associative memories based on distributed constant weight codes. In *Proceedings of Information Theory and Applications Workshop*. 269–273.

V. Gripon, V. Skachek, W. J. Gross, and M. Rabbat. 2012. Random clique codes. In *Proceedings of the 7th International Symposium on Turbo Codes and Iterative Information Processing*. 121–125.

J. Hawkins and S. Blakeslee. 2004. *On Intelligence*. Times Books.

D. O. Hebb. 1949. *The Organization of Behavior*. Wiley, New York.

R. Hecht-Nielsen. 2007. *Confabulation Theory.*Springer-Verlag, Heidelberg.

J. Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA* 79, 8, 2554–2558.

A. Jarollahi, N. Onizawa, V. Gripon, and W. J. Gross. 2012. Architecture and implementation of an associative memory using sparse clustered networks. In *Proceedings of the IEEE International Symposium on Circuits and Systems*.

A. Jarollahi, V. Gripon, N. Onizawa, and W. J. Gross. 2013. A low-power content-addressable-memory based on clustered-sparse-networks. In *Proceedings of the 24th International Conference on Application-Specific Systems, Architectures and Processors*.

H. Jhuang, T. Serre, L. Wolf, and T. Poggio. 2008. Biologically inspired system for action recognition. In *Proceedings of the 11th IEEE International Conference on Computer Vision*, 1–8.

H. Jiping, M. G. Maltenfort, W. Qingjun, and T. M. Hamm. 2001. Learning from biological systems: Modeling neural control. *IEEE J. Cont. Syst.* 21, 4. DOI:10.1109/37.939944

E. Kandel, J. Schwartz, T. Jessell, S. Siegelbaum, and A. J. Hudspeth. 2013. *Principles of Neural Science* 5th Ed. McGraw-Hill Ryerson.

T. Kohonen. 1977. *Associative Memory*. Springer, New York.

J. E. Laird and Y. Wang. 2007. The importance of action history in decision making and reinforcement learning. In *Proceedings of the 8th International Conference on Cognitive Modeling*.

C. Mead. 1990. Neuromorphic electronic systems. *Proc. IEEE* 78, 1629–1636.

NAE Grand Challenges for Engineering. 2013. www.engineeringchallenges.org.

J. M. Nageswaran, M. Richert, N. Dutt, and J. L. Krichmar. 2010. Towards reverse engineering the brain: Modeling abstractions and simulation frameworks. In *Proceedings of the IEEE/IFIP International Conference on VLSI and System-on-Chip*.

G. Palm. 1980. On associative memories. *Biol. Cybernet.* 36, 19–31.

G. Palm. 2013. Neural associative memories and sparse coding. *Neural Netw.* Springer.

R. Perfetti and E. Ricci. 2008. Recurrent correlation associative memories: A feature space perspective. *IEEE Trans. Neural Netw.* 19, 2, 333–345.

Q. Qiu, Q. Wu, M. Bishop, R. E. Pino, and R. W. Linderman. 2013. A parallel neuromorphic text recognition system and its implementation on a heterogeneous high-performance computing cluster. *IEEE Trans. Comput.* 62, 5, 886–899.

A. Sandberg. 2003. Bayesian attractor neural network models of memory. Ph.D. dissertation. Stockholm University.

H. Simon. 2010. *Cognitive Science: Relationship of AI to Psychology and Neuroscience*. AAAI.

L. Soo-Young. 2007. Artificial brain based on brain-inspired algorithms for human-like intelligent functions. In *Proceedings of the IEEE International Conference on Integration Technology*.

A. Sudo, A. Sato, and O. Hasegawa. 2009. Associative memory for online learning in noisy environments using self-organizing incremental neural network. *IEEE Neural Netw.* 20, 6, 964–972.

E. Theordorou and F. J. Valero-Cuevas. 2010. Optimality in neuromuscular systems. In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 4510–4516.

D. Willshaw. 1971. Models of distributed associative memory. Ph.D. dissertation, University of Edinburgh.

L. Wiskott and T. J. Sejnowski. 2002. Slow feature analysis: Unsupervised learning of invariances. *J. Neural Computat.* 14, 4, 55 pages.

Received August 2013; revised November 2013 and January 2014; accepted April 2014