Dynamic power management of MPSoC using networks of neural cliques

Bartosz Boguslawski

THÈSE / Télécom Bretagne
sous le sceau de l'Université européenne de Bretagne

pour obtenir le grade de Docteur de Télécom Bretagne
En accréditation conjointe avec l'Ecole Doctorale Sicma
Mention : Sciences et Technologies de l'Information et de la Communication

présentée par
Bartosz Boguslawski
préparée dans le département électronique
Laboratoire Labsticc

Dynamic power management of MPSoC using networks of neural cliques

Thèse soutenue le 5 novembre 2015
Devant le jury composé de :

Patrick Loumeau
Professeur, Télécom ParisTech / président

Michel Paindavoine
Professeur, Université de Bourgogne / rapporteur

Gilles Sassatelli
Directeur de Recherche, LIRMM – Montpellier / rapporteur

Frédéric Heitzmann
Ingénieur de Recherche, CEA Leti – Grenoble / examinateur

Vincent Gripon
Chargé de Recherche, Télécom Bretagne / examinateur

Fabrice Seguin
Maître de Conférences, Télécom Bretagne / examinateur

Claude Berrou
Professeur, Télécom Bretagne / directeur de thèse

N° d'ordre : 2015telb0370

Sous le sceau de l'Université européenne de Bretagne

Télécom Bretagne

En accréditation conjointe avec l'Ecole Doctorale Sicma

Dynamic power management of MPSoC using networks of neural cliques

---

Thèse de Doctorat

Mention : Sciences et Technologies de l'Information et de la Communication

Présentée par **Bartosz Boguslawski**

Département : Electronique

Laboratoire : Lab-STICC Pôle : CACS

Directeur de thèse : Claude Berrou

Soutenue le 5 novembre 2015

Jury :

M. Michel Paindavoine, Professeur, Université de Bourgogne (Rapporteur)
M. Gilles Sassatelli, Directeur de Recherche CNRS, LIRMM (Rapporteur)
M. Claude Berrou, Professeur, Télécom Bretagne (Directeur de thèse)
M. Patrick Loumeau, Professeur, Télécom ParisTech (Examinateur)
M. Frédéric Heitzmann, Ingénieur de Recherche, CEA Leti (Examinateur)
M. Fabrice Seguin, Maître de Conférences, Télécom Bretagne (Examinateur)
M. Vincent Gripon, Chargé de Recherche, Télécom Bretagne (Examinateur)

Abstract

Doctor of Philosophy - Télécom Bretagne

Dynamic power management of MPSoC using networks of neural cliques

by Bartosz Boguslawski

The challenge of combining high-performance and energy efficiency is clearly present in today's electronics applications. This objective can be reached with Multiprocessor System-on-Chip (MPSoC) platforms that provide high-level of adaptability, performance, reliability and energy efficiency. Nevertheless, they have to be accompanied with a power management decision unit that continuously adapts their operation. This leads us to neural networks that can provide comparable properties to those of the cerebral cortex. We use an associative memory that relies on neural cliques to store information and sparse activity to retrieve it. We analyze these networks in power management applications using real-world data. We show that in order to be compatible with real-world applications, the initially proposed model of networks of neural cliques has to be improved. We propose several methods to do so within which the most efficient relies on a proposed concept of twin neurons. Sparse activity in these networks opens opportunities for low-power implementations. We show that neural cliques are more energy-efficient than power management solutions relying on game theory or Content-Addressable Memory (CAM). Furthermore, we propose a novel 3D interconnect approach for high-performance neural cliques' implementation. We show important gains in terms of total interconnect length and power consumption.

Résumé

Docteur de Télécom Bretagne

Gestion dynamique de consommation de MPSoC par un réseau de cliques neurales

par Bartosz Boguslawski

Les applications microélectroniques d'aujourd'hui nécessitent de combiner haute performance et efficacité énergétique. Cet objectif est accessible grâce aux "Multiprocessor System-on-Chip" (MPSoC) qui fournissent un haut niveau d'adaptabilité, de performance, de fiabilité et d'efficacité énergétique. Néanmoins, ils doivent être couplés à des systèmes de décision et de gestion de la consommation qui adaptent en permanence leur mode de fonctionnement. Cela nous amène à considérer les réseaux de neurones, lesquels ont vocation à offrir des propriétés comparables à celles du cortex cérébral. Ils peuvent notamment servir comme mémoire associative, avec un processus de relecture très économe en énergie. Nous analysons ces réseaux dans des applications de gestion de l'alimentation, avec des données du monde réel. Nous montrons que pour être compatible avec des applications réalistes, le modèle de réseaux de cliques neurales initialement proposé doit être amélioré. Nous proposons plusieurs améliorations et présentons le concept de neurones jumeaux. L'activité parcimonieuse dans ces réseaux ouvre des opportunités pour des implémentations à faible consommation. Nous montrons que les cliques neurales sont plus efficaces en termes d'énergie que des solutions de gestion de l'alimentation reposant sur la théorie des jeux ou des "Content-Addressable Memory" (CAM). En outre, nous proposons une nouvelle approche d'interconnexion 3D pour l'implémentation physique de cliques neurales à haute performance. Nous montrons des gains importants en termes de longueur totale d'interconnexion et consommation d'énergie.

Acknowledgements

Remerciements

Je tiens tout d'abord à remercier mon directeur de thèse, Claude Berrou. Gardant toujours une vue d'ensemble sur ma thèse, il m'a appris à prendre du recul sur mes travaux. Ses contributions ne se limitant pas aux aspects purement techniques mais, comme il le dit si bien "Une thèse, c'est une formation complète", j'ai du apprendre à expliquer des concepts complexes de manière simple, discuter, poser des questions, approcher des problèmes... Et même manger des fruits de mer (en provenance de Bretagne, bien évidemment).

Aussi ai-je eu la chance d'avoir de très bons encadrants : Frédéric Heitzmann, Fabrice Seguin et Vincent Gripon. Tous ont passé un temps infini à répondre avec moi aux questions que nous nous sommes posées. Grâce à eux, je garderai de très bons souvenirs, pas seulement du bureau mais aussi au travers des autres activités que nous avons partagées. Fabrice m'a montré son nom inscrit sur la tour Eiffel (oui !). Fred m'a une fois demandé d'emporter mes baskets lors d'un déplacement professionel, et aujourd'hui je ne peux plus me passer de course à pied. Et Vincent m'a fait découvrir Montréal le soir et m'a fait goûter la poutine. Me rappeler de ces instants me donnera toujours le sourire.

J'adresse également un grand merci au chef du LISAN, Fabien Clermidy, pour son support, nos nombreuses discussions et l'environnement de travail très riche et motivant qui reigne au sein de son laboratoire. Je n'oublie pas non plus Rodolphe Héliot qui a été la première personne à me présenter le sujet qui est devenu ma thèse et qui a toujours cru en moi.

Merci aussi à tous les membres du jury, Michel Paindavoine, Gilles Sassatelli, Patrick Loumeau et bien sûr Claude, Fred, Fabrice et Vincent. Merci pour l'intérêt porté à mon travail et nos échanges lors de ma soutenance.

Je voudrais remercier les autres doctorants avec lesquels j'ai eu la chance de travailler, notamment Benoît Larras de Télécom Bretagne et Hossam Sarhan du CEA. C'était une très bonne expérience et un vrai plaisir.

Durant ma thèse, j'ai eu la chance de travailler en deux endroits - à Télécom Bretagne et au CEA Leti. Grâce à cela, j'ai rencontré des personnes formidables qui sont devenues de très bons collègues et mes amis. Ils sont nombreux et se reconnaîtront. Pour les pauses café, la grimpe, le ski - tous ces moments fantastiques que nous avons passés ensemble,

je ne trouve pas de mots pour exprimer à quel point je vous suis reconnaissant. Un simple merci ne suffit pas !

J'adresse enfin le merci le plus important à ma famille, notamment mes parents qui m'ont depuis toujours préparé pour cette aventure. Sans leur soutien, je n'aurais jamais été capable d'en arriver là. Merci à ma Rebecca pour son aide, sa présence et son amour.

"Words must be weighed, not counted."

Polish saying

To my parents...

Contents

List of Figures

List of Tables

Abbreviations

IoT	**I**nternet-**o**f-**T**hings
MPSoC	**M**ulti**p**rocessor **S**ystem-**o**n-**C**hip
PE	**P**rocessing **E**lement
CAM	**C**ontent-**A**ddressable **M**emory
NoC	**N**etwork **o**n **C**hip
NI	**N**etwork **I**nterface
GALS	**G**lobally **A**synchronous **L**ocally **S**ynchronous
VFI	**V**oltage/**F**requency **I**sland
DVFS	**D**ynamic **V**oltage and **F**requency **S**caling
FDSOI	**F**ully **D**epleted **S**ilicon **o**n **I**nsulator
FIFO	**F**irst **I**n, **F**irst **O**ut
LDM	**L**ocal **D**ecision **M**aker
SRAM	**S**tatic **R**andom **A**ccess **M**emory
MC	**M**emory **C**ell
SL	**S**earch**l**ine
ML	**M**atch**l**ine
HNN	**H**opfield **N**eural **N**etwork
SNN	**S**piking **N**eural **N**etwork
STDP	**S**pike-**T**iming-**D**ependent-**P**lasticity
DBN	**D**eep **B**elief **N**etwork
WTA	**W**inner **T**akes **A**ll
LsKO	**L**oser**s** **K**icked **O**ut
FSA	**F**irst **S**egment **A**nomaly
RO	**R**ing **O**scillator
PVT	**P**rocess **V**oltage and **T**emperature

TSV	**T**hrough-**S**ilicon-**V**IA
NV	**N**on-**V**olatile
PUF	**P**hysical **U**nclonable **F**unction
FPGA	**F**ield-**P**rogrammable **G**ate **Array**
NLP	**N**atural **L**anguage **P**rocessing

Symbols

f	clock frequency	Hz
V_{dd}	supply voltage	V
P	power	W
P_{dyn}	dynamic power	W
P_{stat}	static power	W
P_{short}	short circuit power	W
P_{switch}	switching power	W
I_{sc}	short circuit current	A
C	capacitance	F
α	switching activity factor	
I_{leak}	leakage current	A
E_{stat}	static energy	J
L	latency deadline	s
E_{dyn}	dynamic energy	J
X	time to process a task	s
N	number of clock cycles needed to finish a task	
T	clock period	s
E_{Hdyn}	dynamic energy in high-activity state	J
E_{Ldyn}	dynamic energy in low-activity state	J
E	total energy	J
V_{nom}	nominal supply voltage	V
T_{nom}	nominal clock period	s
E_{nom}	nominal energy	J
E_{dd}	dynamic energy per clock cycle at V_{dd}	J
T_{dd}	clock period at V_{dd}	s

γ	energy reduction per clock cycle in low-activity state
ν	number of VFIs in the MPSoC
s	game theory action
u	game theory outcome
z	number of matchlines
w	search word length
x	output word length
y	number of entries in RAM
g	integration function
a	activation function
q	number of signals at neuron's input
v	excitatory signal at neuron's input
ζ	inhibitory signal at neuron's input
σ	neuron's activation threshold
n	total number of neurons in neural network
M	number of messages
m	a message
w_{ij}	weight between neurons i and j
c	number of segments in a message, number of clusters
ℓ	segment's maximal value, cluster size
δ	minimal distance between two cliques
d	network density
c_e	number of segments erased in a message
P_c	probability of a correct message retrieval after one iteration
P_e	probability of a incorrect message retrieval after one iteration
p	probability of choosing value one on the first segment of *FSA* distribution
d'	density between the first cluster and any other one for any value on the first segment that is not one for *FSA* distribution
d'_1	density between the first cluster and any other one for the first segment equal one for *FSA* distribution
π	first segment equals one in a message drawn from *FSA* distribution

$\overline{\pi}$	first segment different from one in a message drawn from *FSA* distribution	
ϵ	first segment is erased in a message drawn from *FSA* distribution	
$\overline{\epsilon}$	first segment is not erased in a message drawn from *FSA* distribution	
$P_{e,\pi\epsilon}$	error probability after one iteration for $\pi\epsilon$ case of *FSA* distribution	
$P_{e,\pi\overline{\epsilon}}$	error probability after one iteration for $\pi\overline{\epsilon}$ case of *FSA* distribution	
$P_{e,\overline{\pi}\epsilon}$	error probability after one iteration for $\overline{\pi}\epsilon$ case of *FSA* distribution	
$P_{e,\overline{\pi\epsilon}}$	error probability after one iteration for $\overline{\pi\epsilon}$ case of *FSA* distribution	
n_1	number of fanals associated to value one in the first cluster for *FSA* distribution	
P_r	process	
T_{emp}	temperature	°C
$\hat{V}$	approximated voltage	V
$\hat{T}_{emp}$	approximated temperature	°C
$\overrightarrow{F}$	frequency vector	Hz
V_T	threshold voltage	V
h	length of a vector at the output of the transcoder in digital implementation of neural cliques	
k	fanal's index in digital implementation of neural cliques	
I_{UNIT}	unitary current	A
A_{f+s}	area occupied by one fanal circuit and all its synapses	μm^2
A_{fanal}	area occupied by fanal circuit	μm^2
$A_{synapse}$	area occupied by synapse circuit	μm^2
ψ_s	number of synapses connected to one fanal	
$dist_{(i,j)(i',j')}$	Manhattan distance between two fanals with coordinates (i,j) and (i',j')	μm
τ_{2D}	RC delay in a 2D circuit	s

$R_{per\ \mu m}$	resistance per unit length	Ω
$C_{per\ \mu m}$	capacitance per unit length	F
τ_{3D}	RC delay in a 3D circuit	s
R_{TSV}	resistance of a TSV	Ω
C_{TSV}	capacitance of a TSV	F
$dist_{total}$	total wire length	μm
τ_{max}	maximal RC delay	s
W	workload	
f_{glob}	global frequency	Hz
c_{in}	number of input clusters	
c_{out}	number of output clusters	
ψ	total number of connections	
p	number of transistors necessary to implement a connection	
κ	implementation complexity	
b	number of transistors necessary to implement a buffer	
t	number of transistors necessary to implement a fanal	

Introduction

Context and motivation

The increasing density of transistors per chip provides more and more computing power within a single chip. Computing systems become increasingly integrated and attractive for new, previously inaccessible application fields. They become ubiquitous and have to be mobile. Supporting the variety of applications and providing a sufficient autonomy means that these systems have to be efficient both in terms of speed and energy.

This is relevant for all the components of a complete system. Internet-of-Things (IoT) microcontrollers for wireless sensor nodes (Bol et al., 2013), on-chip interconnect network routing (Qi et al., 2010), memories (Noguchi et al., 2015), dedicated hardware accelerators (Kuo et al., 2009), wireless communication (Lu et al., 2013), sensors (Boukhayma et al., 2014), just to name a few - nowadays, they all have to provide high-performance and consume as little energy as possible.

To support this large variety of applications, multiple dedicated components are integrated on one chip and interconnected with a communication infrastructure. Depending on the application running, different components are used to provide the necessary performance. This type of architecture is called Multiprocessor System-on-Chip (MPSoC) and is present throughout this report.

Figure 1a shows an MPSoC-type mobile system. Dedicated computation units are represented with squares, the lines depict the communication infrastructure. We call the computation units Processing Elements (PEs). PEs can be general purpose processors but also hardware accelerators (*e.g.* audio/video coders/decoders), memories, peripherals (*e.g.* Bluetooth, USB) among others. In the classical design, illustrated in Figure 1b, all the PEs work at their maximal performance (marked in red). This guarantees that

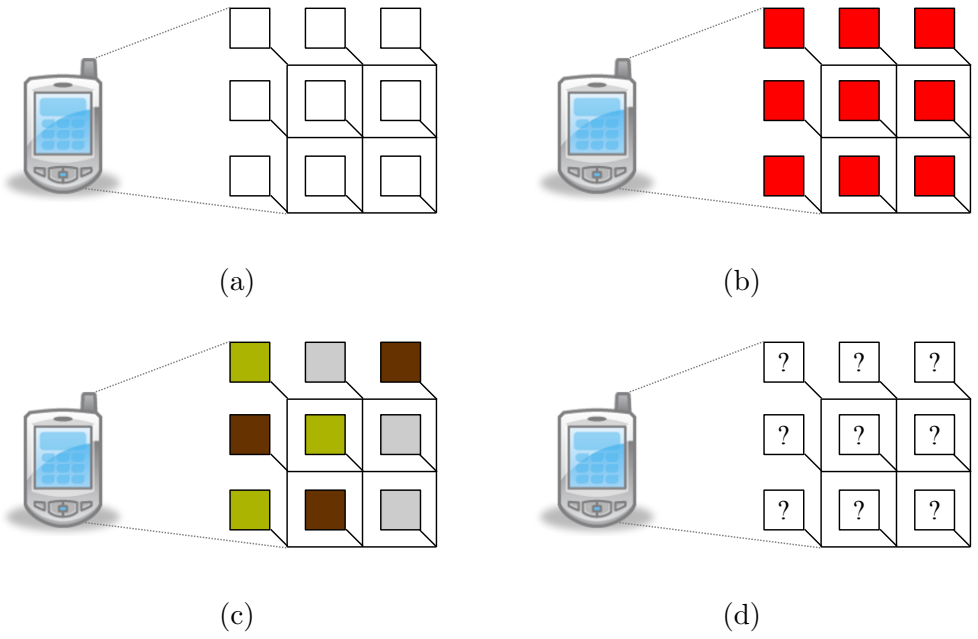


FIGURE 1: (a) MPSoC-type mobile system (b) Classical design for the most demanding application (c) Flexible PEs can adapt to the current requirements (d) How to find the optimal settings for each PE?

all the applications targeted for this platform are executed at the highest speed, and therefore all the time-related constraints are preserved. However, from the energetic point of view, this is the worst-case design. Since all the PEs work at their maximal capabilities, they consume a maximal amount of energy. Now, as mentioned before, the MPSoC platform supports a large group of applications, not all of them having the same performance requirements for all the PEs. In the given example, the MPSoC is embedded in a smartphone. Obviously, a smartphone may run a lot of different applications with very diverse computing requirements. The user does not need the whole device's computing power to write a simple text message. Moreover, not only the application may change, but the environment is dynamic as well. The processing of a signal from a base station changes with regard to the distance between the smartphone and the base station. When the phone is close to the base station, the signal processing tasks can be simplified. Another consequence of the classical design system from Figure 1b is excessive heating of the PEs. As they all work at the highest speed, they dissipate a maximal amount of power that is then converted into heat. If no special care is taken, this may lead to unexpected behavior or even irreversible system breakdown.

We can improve the classical design by adding a certain flexibility to the PEs. What if each PE had a knob that allows to slow it down or speed it up according to the

current needs (application type, environment, temperature-safe operation, ...)? When the application or its constraints are changed, when the smartphone moves closer to the base station, or some of its PEs are heating too much, one simply adapts the knobs. This situation is illustrated in Figure 1c. Different colors represent different positions of the PEs' knobs.

Even if it sounds quite easy, the control of the knobs is not an obvious task (Figure 1d). Given that PEs do not have the same characteristics, knowing that they depend on each other, what is the optimal position of each knob to globally consume the least energy without altering the application operation? Even more importantly, how to find the good solution for all the knobs fast enough to reconfigure the system in real-time? Furthermore, the decision process is complex, requires a significant computing power and can consume a significant amount of energy as well. Possibly the decision time and energy demand are too important compared to the potential gains. A fast and efficient decision system is needed, which is difficult to achieve with today's silicon technology.

One can see that in the context of MPSoCs, the aforementioned need for high-performance and low-energy consumption is clearly the issue in hand. Nonetheless, human brain is a good example of a performant and energy-efficient system. Although this organ consumes just 20W (Drubach, 1999), it surpasses the fastest supercomputer Tianhe-2 that consumes 17.8MW (Sengupta and Stemmler, 2014), in tasks such as decision making, classification, associations, information cross breeding or production of new one, functioning in presence of noise, uncertainty and errors (Whitworth, 2008, Rojas, 1996). The gap between the energy efficiency of the two shows that there is a lot of inspiration to be derived from the human brain to electronics.

Objective

The challenge of combining high-performance and energy efficiency raised above is really fundamental for this work. The objective is to apply a recently introduced type of neural networks based on neural cliques (Gripon and Berrou, 2011a) to an application requiring high speed and energy efficiency. This type of neural network showed a huge gain in performance compared to state-of-the-art neural networks (Gripon and Berrou, 2011a). It relies on an efficient way to store information and sparse activity to treat it. That

opens opportunities for low-power implementations. Since the abovementioned MPSoC architecture fits very well into this context, it is the main application targeted in this document. The objective is to explore whether networks of neural cliques can be used to propose a high-performance and energy efficient hardware solution for such a type of application.

Contribution

The contributions presented in this report are as follows:

- The analysis of networks of neural cliques in a context of real-world applications and more specifically in terms of real-world data.
- The proposal of a theoretical analysis and evaluation of multiple ways to adapt networks of neural cliques to real world-data. The validation on power and variability management applications. Adapting networks of neural cliques to real world-data is essential to use it in practically any real-world application.
- The proposal of a specific neural cliques' architecture for hardware MPSoC power management controller and validation on several test-cases. The proposed solution offers a very good compromise between decision speed, energy efficiency and decision optimality compared to state-of-the-art controllers.
- The exploration of numerous improvements of the neural cliques' hardware implementation improving its functionality and efficiency.

Report organization

The first chapter explains the MPSoC architecture in detail and elaborates on power management techniques. Later, the concept of using an associative memory for power management is presented for the first time. In the end of this chapter, a comparison of strong and weak points of each type of power management control is given.

The second chapter introduces biological and artificial neural networks. The goal is to outline the basic concepts of neural networks to ease the introduction of networks relying

on neural cliques. The rest of the chapter presents the principles of networks of neural cliques.

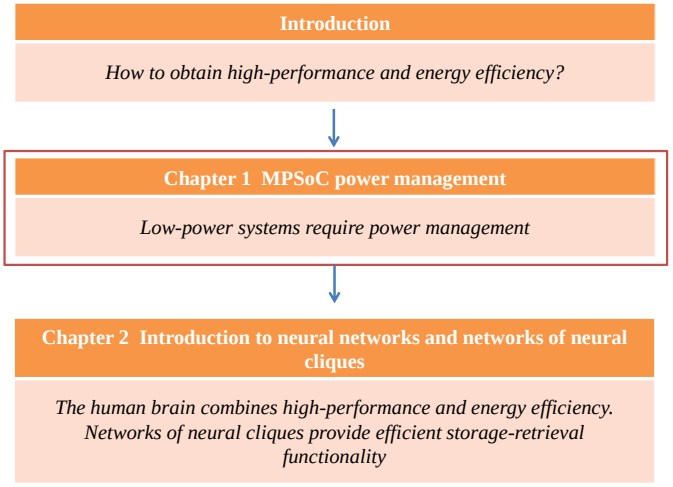
The next chapter attempts to use networks of neural cliques for the first time in a practical scenario with a real-world type of data. It is shown that real-world data impacts the functioning of these networks, and therefore it is essential to propose methods to counterbalance these effects if real-world applications are targeted. We propose, evaluate and compare multiple approaches. The most efficient method relies on a concept of twin neurons and is analyzed in depth both formally and by simulations. To assess the model adapted to real-world data, twin neurons are used as power and variability management controller to optimize power consumption of electronic circuits.

Later, chapter 4, proposes neural cliques' hardware structure for MPSoC power management application. It is then compared to a state-of-the-art method. A comparison with a Content-Addressable Memory (CAM)-based associative memory is also provided. Further, some additional points on hardware implementation, such as process variability robustness, synapses programming or efficient interconnect are discussed. We propose a compact interconnect approach using 3D technology to reduce interconnect delay and energy consumption.

Finally, detailed contributions and conclusions are given. The presented work leads to numerous perspectives that are discussed both in terms of implementation and applications in the closing part.

Chapter 1

MPSoC power management



1.1 Introduction

This chapter presents the MPSoC architecture and explains how it can be exploited to allow for dynamic power management. Then, state-of-the-art of power management decision units is outlined. A detailed power model is given and power optimization is formally defined. Later, functioning of CAM-SRAM associative memory is explained and it is shown how to use it for power management. The chapter is concluded with a comparison of different types of power management decision units.

1.2 MPSoC architecture

1.2.1 Generalities and definitions

Let us first discuss how MPSoC-type architectures emerged. We used to associate *high-performance computing* with scientific applications, *e.g.* medical imaging, bio-science, electrical simulations. Nowadays, the meaning of this term is changing. High-performance computing includes also embedded computing, multimedia, voice recognition, video compression/decompression, high-speed communication. These new applications imply real-time constraints which means that if some task is not finished in time, the whole system may fail. Furthermore, if a task is finished too early, the whole system may not necessarily benefit. This means that more flexibility is needed to fine-tune the system to time-related constraints.

Apart the need for a new level of flexibility, the second reason to search for new computing architectures is the increasing technological variability (Borkar, 2005). The transistor's dimensions continue to scale down which causes reliability issues. It is unlikely that all the parts of the chip are equally efficient. Technological improvements are not sufficient, new architectural approaches are indispensable. Consequently, shifting to parallelism instead of increasing the operation frequency is preferred. Replicating a computation unit multiple times introduces redundancy and allows for control and error correcting schemes. For example, computation results of one unit can be controlled and corrected if needed by another unit. Isolating faulty parts of the system is also possible. If, due to variability issues, a unit does not provide sufficient performance, its parameters may be tuned as well. Moreover, as the computation is distributed, it is advantageous in terms of radiation hardening as well.

The third reason is that modern mobile systems have to provide extremely low-power operation since they operate on batteries. This can be achieved using dedicated accelerators with specifically optimized instruction sets to offer the best performance/energy efficiency trade-off (Wolf et al., 2008). At the same time, multitasking requires multiple resources to allow for parallel processing. Embedding multiple types of units allows performing different functions concurrently.

Since the design cost and time-to-market metrics are important for business prosperity, new systems have to be designed with minimal effort. As a consequence, it is important

to obtain an easily scalable architecture that can be quickly adapted to the current market needs. It is therefore important to search for regular, distributed approaches to avoid the complete redesign of each new product.

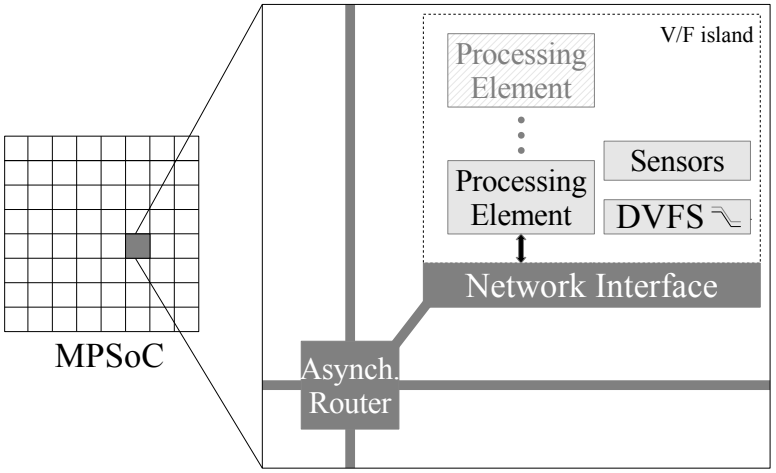


FIGURE 1.1: MPSoC with power management capability. Dynamic Voltage and Frequency Scaling (DVFS) allows to modify the power mode of the Voltage/Frequency Island (VFI). Each VFI groups a number of Processing Elements. Communication between VFIs is possible through the Network Interface and routers.

Knowing requirements for the new architecture, we proceed to the description of MPSoCs. This architecture responds very well to all the abovementioned points.

Figure 1.1 presents an MPSoC architecture. The tasks are executed on PEs. Depending on the type of PEs, two different categories of MPSoCs are distinguished:

- Heterogeneous MPSoCs which are composed of different types of PEs designed to perform their specific tasks (dedicated accelerators).
- Homogeneous MPSoCs which are composed of one type of PE instantiated multiple times where all the elements necessary for the targeted application are embedded inside the PE.

A comparison of both types of MPSoCs is given in (Jalier et al., 2010). In Table 1.1 we present a general comparison of heterogeneous and homogeneous MPSoC architectures. Thanks to dedicated PEs, heterogeneous MPSoC provide good energy efficiency and performance. Memory management is simplified thanks to the fact that a certain memory block is addressable only by certain PEs and consequently, at a given moment,

TABLE 1.1: Comparison of heterogeneous and homogeneous MPSoC architectures

	Heterogeneous	Homogeneous
Energy efficiency	+	-
Performance	+	-
Memory management	+	-
Flexibility	-	+
Scalability	-	+

it is easier to determine which data is manipulated by which PE. Contrary to heterogeneous MPSoCs, homogeneous MPSoCs are configurable after the fabrication and much easier to scale (by increasing the number of PEs) (Saponara and Fanucci, 2012). The choice between the two depends on the targeted group of applications. Generally speaking, homogeneous MPSoCs are well suited for data-parallel systems where many simpler tasks have to be performed on independent data streams. Heterogeneous MPSoCs target high-performance applications with low-power requirements.

1.2.2 Communication schemes for MPSoCs

To allow possibly large number of PEs to communicate, an efficient communication scheme is necessary. The traditional bus-based communication becomes the bottleneck since only one communication transaction at a time is allowed by the arbitration result. Consequently, the communication bandwidth of each PE is inversely proportional to the number of PEs in the MPSoC. A hierarchical communication structure with few buses could alleviate the limitations of single bus structure, yet it requires a bigger design effort, several communication protocols and application specific PEs' grouping (Tsai et al., 2012). That is why Network on Chip (NoC) is proposed as a new communication scheme for MPSoCs (Kumar et al., 2002). It consists of Network Interfaces (NIs), routers and links (*cf.* Figure 1.1). The NI provides an interface between the PEs and the router, converts data into network packets and implements asynchronous-synchronous interfaces (Beigné et al., 2005). Routers route data between the source and destination through the links. NoC is usually organized in a mesh topology due to its simple layout and easy routing resulting in reduced-complexity routers. NoC allows for Globally Asynchronous Locally Synchronous (GALS) systems which means that there are local areas which work within the same clock domain but globally the system is asynchronous. As a

result, the problem of clock skew is avoided and the power consumption related to clock propagation is reduced (Kumar et al., 2002). Furthermore, thanks to the distributed NoC architecture, routing decisions are made based on the local information at the routers. This allows for easier scaling.

1.2.3 Dividing MPSoC on Voltage/Frequency Islands

Having multiple PEs and a communication structure enabling GALS, it is possible to divide the MPSoC on a number of Voltage/Frequency Islands (VFIs). Each VFI works with its own clock frequency f and supply voltage V_{dd}. To simplify, the couple (f, V_{dd}) is called *power mode*. To be able to apply the power modes, specific *actuators* are necessary. For that reason, each VFI integrates Dynamic Voltage and Frequency Scaling (DVFS) actuator that sets a given power mode. Nowadays, DVFS actuators allow switching between two different power modes in time of the order of tens of nanoseconds (Kim et al., 2008, Truong et al., 2009, Beigné et al., 2009). The authors of (Kim et al., 2008) show different tradeoffs in DVFS actuator's design in terms of voltage scaling time, voltage variation of the load transient response and switching efficiency. It is shown that it is possible to switch between two voltages in 50ns. Small voltage variation of 5% is ensured. To guarantee stable operation, when changing the voltage to a higher level, low-to-high frequency transitions are allowed once the voltage settles. Similarly, when changing the voltage to a lower level, firstly the frequency is scaled down and then, followed by the voltage transition. Further, as stated in (Kim et al., 2008), to take advantage of the fast switching capabilities of DVFS actuators, fast algorithms have to be developed to provide DVFS actuators with power modes that have to be set.

Additionally, each VFI is equipped with a set of sensors to provide an information about the current working conditions, such as temperature, consumed power or other. The number of PEs within a single VFI can be traded to balance the cost of adding additional circuitry (NI, DVFS, sensors) and the flexibility of the MPSoC.

MPSoC provides the possibility to fine-tune the system to the current time constraints. Its distributed architecture, enabling parallel computation, responds well to variability issues and suits well multitask applications. Integrating dedicated accelerators allows for high-performance and energy-efficient computation. Its regular architecture reduces time-to-market and simplifies the scalability.

1.3 Power management on MPSoC

Equipping each VFI with the DVFS actuator allows for adapting (f, V_{dd}) to current requirements. In this section it is explained how this can reduce the power consumption of the MPSoC.

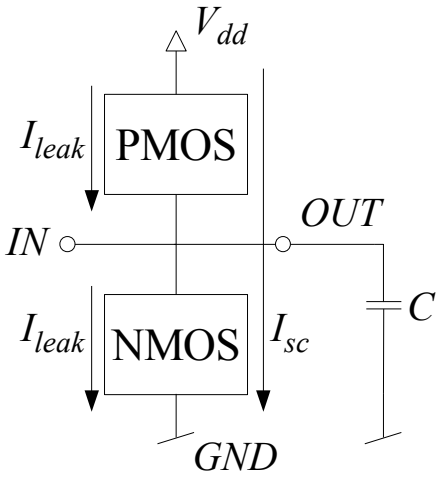


FIGURE 1.2: CMOS circuit and the used notation. Load capacitance C, short circuit current I_{sc} and leakage current I_{leak} are marked.

The power consumption of a CMOS integrated circuit consists of dynamic power P_{dyn} and static power P_{stat} and is given by (Shauly, 2012):

$$P = P_{dyn} + P_{stat} = (P_{short} + P_{switch}) + P_{stat} = I_{sc}V_{dd} + \alpha C V_{dd}^2 f + I_{leak}V_{dd} \quad (1.1)$$

The dynamic power P_{dyn} consists of short circuit power P_{short} and switching power P_{switch}. P_{short} is related to short circuit current I_{sc} which flows from V_{dd} to ground for a short time when both NMOS and PMOS transistors are active. P_{switch} is a result of charging and discharging the load capacitance C. The parameter α is called switching activity factor and its typical value is 0.2 for logic blocks designed for 65nm technology (Morifuji et al., 2006). Even when the transistors are not switching, there is some leakage current I_{leak} that contributes to the total power consumption through the static power P_{stat}. This is represented in Figure 1.2.

We can see that P_{dyn} can be directly reduced by lowering the clock frequency f when possible. Since the supply voltage V_{dd} required for stable operation is determined by f, it is also possible to reduce V_{dd} accordingly. Reducing V_{dd} allows for minimizing the

static power P_{stat} as well. Static power P_{stat} can be further reduced by keeping the chip temperature low (Deo, 2010), layout optimization techniques (Shauly, 2012) or technological improvements, *e.g.* Fully Depleted Silicon on Insulator (FDSOI) technology (Vitale et al., 2010).

Equation (1.1) shows how we can reduce the power consumption by using DVFS actuators to adapt (f, V_{dd}) to the application needs. We call the process of adapting the power modes *power management*. However, to exploit this opportunity, DVFS actuators have to be provided with a decision on which power modes have to be set.

1.4 State-of-the-art of power management decision units

The above sections show that the MPSoC architecture allows for a new level of performance, flexibility and energy consumption reduction opportunities. However, there is a need for an energy efficient and fast *decision unit* that provides DVFS actuators with power modes that are set to reduce the power consumption. This section reviews the state-of-the-art of these decision units for power management on MPSoC.

We divide the state-of-the-art decision units into two main categories:

- Low-level decision units which take the decision based on local information.
- High-level decision units which take the decision based on global information.

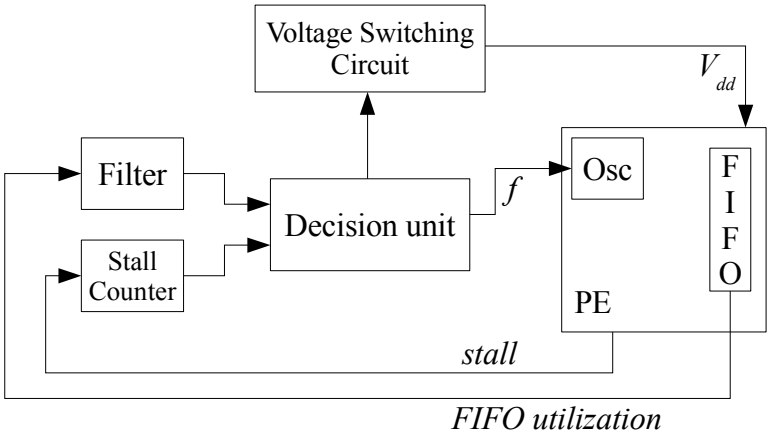


FIGURE 1.3: Low-level decision unit from (Truong et al., 2009). Decision is made based on two signals from the PE. Either *FIFO utilization* containing current fullness of the PE's input data buffer FIFO or *stall* asserted whenever the PE is inactive.

1.4.1 Low-level decision units

Low-level decision units are based on local microarchitectural control dedicated to provide DVFS actuators with the decision on power modes. The authors of (Kim et al., 2008) propose to exploit memory-bound intervals in the application to slow down the PE. Up to 60% energy savings are achieved for the presented benchmarks. Nevertheless, the effectiveness of such an approach depends directly on the number of memory-bound cycles. The authors show that for some benchmarks, their decision unit provides a limited power benefit compared to static configuration. Another type of low-level decision unit is proposed in (Truong et al., 2009). Its structure is depicted in Figure 1.3. The decision is made based on two signals from the PE: *FIFO utilization* and *stall*. *FIFO utilization* provides the information about the current fullness of the PE's input data buffer FIFO (First In, First Out). This signal is then averaged over time using a filter and fed into the decision unit. If the FIFO is often full the decision unit decides to speed up the PE. Otherwise the PE is slowed down. The *stall* signal serves as an alternative method to control the speed of the PE. This signal is asserted whenever the PE is inactive. This is caused by either reading an empty FIFO or writing to a full FIFO. The decision unit counts the number of cycles the PE stalls, and when it reaches a user defined threshold, the PE is slowed down accordingly. Since the decision is taken based on the local information, it is not guaranteed that it is globally optimal. Furthermore, adapting the thresholds for decision taking is left to the user. A decision unit relying on control automation theory is proposed in (Almeida et al., 2011). Once again, the decision is taken based only on local information. Moreover, it takes 100ms to reach the stable response of the decision unit. Such a time response does not allow taking advantage of the fast switching capabilities of DVFS actuators.

1.4.2 High-level decision units

High-level decision units have a global scope of the system and can provide a decision that is globally optimal or close to optimal. The operating system can serve as such a global decision unit (Zhuravlev et al., 2013). Based on the system state that resides in memory it adapts the settings of each VFI. However, in a system with distributed memory, the system state is distributed as well. This slows down the decision process and implies slower time response of the software. That is why hardware approaches are

explored. The work (Toprak-Deniz et al., 2007) proposes to use the principles of analog computation to obtain a decision unit minimizing energy consumption under timing constraints. The advantage of the analog computation is that it fully adapts to variations in process and temperature. Nonetheless, the decision unit's response time is 50μs, yet it controls only three PEs. Such a time response does not allow taking advantage of the fast switching capabilities of DVFS actuators. Moreover, no estimations are given about the decision unit's performance when the number of PEs scales. The authors of (Mansouri et al., 2010a) proposed a distributed high-level decision unit relying on a hybrid subgradient and consensus method. Although each VFI uses only local information from its neighborhood, the provided decision is close to optimal. Nevertheless, only theoretical study of the proposed algorithm is presented. The response time and hardware cost are not evaluated. Still, one iteration of the algorithm is estimated to take several ms, and tens of iterations are necessary for the algorithm to converge. Another approach is proposed in (Puschini et al., 2009). The proposed decision unit relies on game theory. In this case each VFI is a player that plays in such a way to maximize its own gain. This type of decision unit has a time response from few to hundreds of μs (at clock frequencies from 100 to 500MHz) for up to 100 VFIs (Puschini et al., 2008, Mansouri et al., 2010b).

Since the game theory decision unit is evaluated on hardware level and serves as one of the references for the decision unit proposed in this work, we describe it more in detail in Section 1.6. First however, we exploit the model of power dissipation in MPSoCs to formally define the energy optimization problem.

1.5 MPSoC power model and optimization formulation

We consider an MPSoC platform with an application mapped on it. As already mentioned before, the power consumption of a CMOS circuit consists of static and dynamic power. Here, we consider the static power P_{stat} constant, though in a more complex model, it can be a function of the supply voltage, temperature, and so forth. Consequently, the static energy E_{stat} depends only on the execution latency L when the application is running and equals:

$$E_{stat} = LP_{stat}. \tag{1.2}$$

Conversely, the dynamic energy E_{dyn} is a function of the circuit's activity. The task assigned to the VFI$_i$, is processed in time:

$$X_i = N_i T_i, \tag{1.3}$$

where N_i is the number of clock cycles needed to finish the computation and T_i is the clock period. During X_i the VFI consumes $E_{Hdyn,i}$. Until the application is launched again, the circuit is in low-activity state when classic low-power techniques, as clock gating (Shinde and Salankar, 2011), are used. Therefore, during $L - X_i$ it consumes $E_{Ldyn,i}$. The total energy consumed by the VFI$_i$ during the application execution is:

$$E_i = E_{Hdyn,i} + E_{Ldyn,i} + LP_{stat}. \tag{1.4}$$

The short circuit power P_{short} is comparatively insignificant to the switching power P_{switch} (Zhuravlev et al., 2013) and is not taken into account in the calculations. As a result, the dynamic power P_{dyn} becomes:

$$P_{dyn} = P_{switch} = \alpha C V_{dd}^2 f = \alpha C \frac{V_{dd}^2}{T}. \tag{1.5}$$

To obtain the characteristics of the MPSoC, a nominal supply voltage V_{nom} and nominal clock period T_{nom} are set. This gives the nominal dynamic energy E_{nom}:

$$E_{nom} = P_{dyn} T_{nom} = \alpha C V_{nom}^2. \tag{1.6}$$

Since αC is constant:

$$\alpha C = \frac{E_{nom}}{V_{nom}^2} = \frac{E_{dd}}{V_{dd}^2}. \tag{1.7}$$

Where E_{dd} is the dynamic energy at V_{dd}. Thus:

$$E_{dd} = E_{nom} \left(\frac{V_{dd}}{V_{nom}} \right)^2. \tag{1.8}$$

Since the frequency is proportional to the supply voltage, the dynamic energy consumption per clock cycle is:

$$E_{dd} = E_{nom} \left(\frac{T_{nom}}{T_{dd}} \right)^2, \tag{1.9}$$

where T_{dd} is the clock period at V_{dd}.

At this point, we can express the whole energy consumed by VFI_i working with the clock period T_i, with regard to its nominal parameters E_{nom}, T_{nom}:

$$E_i(T_i) = \left[N_i + \gamma_i \left(\frac{L}{T_i} - N_i\right)\right] \frac{E_{nom,i} T_{nom,i}^2}{T_i^2} + LP_{stat,i}, \tag{1.10}$$

where γ_i is the ratio of the energy consumed per clock cycle in the low-activity state to the energy consumed per clock cycle in the high-activity state, weighted by the time spent in each state:

$$\gamma_i = \frac{E_{L_{dyn,i}}/(L - X_i)}{E_{Hdyn,i}/X_i}. \tag{1.11}$$

Given the latency deadline L, and the number ν of VFIs, the optimization is formulated as follows:

$$\begin{aligned} &\text{minimize} && \sum_{i=1}^{\nu} E_i(T_i) \\ &\text{subject to} && \sum_{i=1}^{\nu} (X_i = N_i T_i) \leq L. \end{aligned} \tag{1.12}$$

Thus, the objective of the optimization is to minimize the total energy consumption of the system, considering local clock periods as state variables, and preserving the time constraint. An exemplary application is depicted in Figure 1.4. One has to adapt the clock period of each VFI in the application graph to balance energy consumption and processing time, accordingly to the global latency constraint L. The blocks do not necessarily have the same characteristics which is represented with their different sizes. The first block is set to a high-energy consuming power mode, and therefore its task is processed quickly. The second one works in a less-energy consuming power mode, whereas the last block is set to its slowest and most energy-efficient mode.

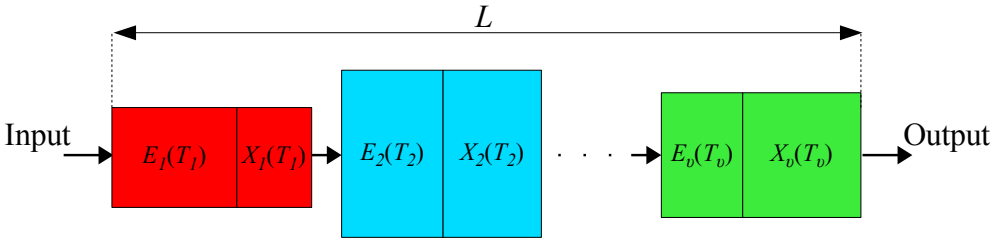


FIGURE 1.4: Example of an application. The total energy consumption has to be minimized, latency constraint L has to be preserved. VFIs have different characteristics represented with different sizes of the blocks. Each VFI consumes an energy E and processes its task in time X that depend on its clock period T.

In our energy model, for a given latency constraint L, global static power has a constant contribution. Therefore, it is discarded in the optimization process.

1.6 Game theory for power management on MPSoC

Game theory is firstly proposed in (Neumann and Morgenstern, 1944). A game is composed of several *players*. Each player has a number of possible *actions* s to play. Any player chooses the actions it plays such that it maximizes its own *outcome* u. The outcome of each player depends not only on its own actions but also on the actions played by other players. The game consists of multiple *game cycles*, to give the players a chance to improve their outcomes by choosing new actions, this time knowing the impact of the actions played by others on the global outcome. After a number of game cycles, when all the players cannot improve their outcomes anymore, they do not change their actions between two game cycles. Such a situation indicates that the game reached an *equilibrium*. In the context of MPSoC power management, the players do not cooperate between each other when taking the decision. This type of game is called *noncooperative*. A noncooperative game has at least one equilibrium, known as Nash equilibrium (Nash, 1951). Game theory is widely used in domains such as economy, sociology, biology, engineering, among others.

To obtain a power management control scheme, each VFI is associated with a player. This is illustrated in Figure 1.5. The actions s are different power modes (f, V_{dd}). After the game is finished, the power modes are set by DVFS actuators.

Depending on the test-case scenario, game theory-based decision unit allows for 38% energy consumption reduction (compared to a given energy budget) (Puschini et al., 2009) or 23% temperature reduction (Puschini et al., 2008). Several implementations are explored (Mansouri et al., 2010b). A software version of the decision unit is flexible and allows for algorithm's parameters tuning by updating the program. However, since it is implemented in a general purpose processor, its area is not optimized (0.122mm^2 per VFI, 4% of VFI's area) and it has a long time response (6.05μs for one game cycle). Hence, a hardware implementation designed for the 65nm ST technology is also explored. A Local Decision Maker (LDM) that realizes the player's functionality is integrated in each VFI. Thanks to hardware optimization, the area is reduced to 0.014mm^2 per VFI

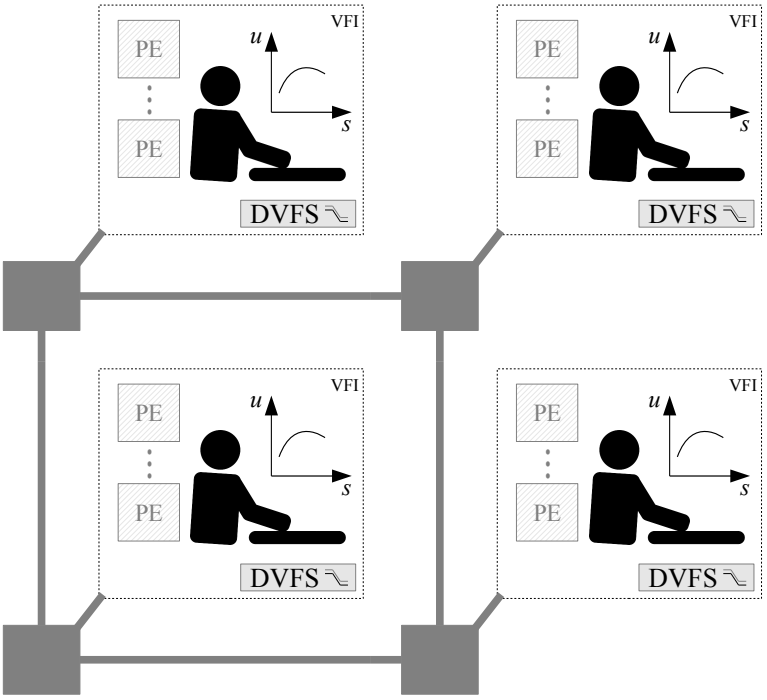


FIGURE 1.5: Game theory power management scheme. Each player tries to maximize its outcome u by playing actions s.

with a slight time response increase (7.52μs for one game cycle). Nevertheless, the authors claim that the hardware implementation scales much better compared to the software one.

Since high-level decision units do not allow for exploiting fast switching capabilities of DVFS actuators, another approach for a decision unit is introduced in the next section.

1.7 CAM-SRAM associative memory for power management on MPSoC

1.7.1 Generalities and definitions

In traditional indexed memories, data is addressed using a known pointer. The principle of associative memories is different: data retrieval is accomplished presenting a part (possibly small) of it. Thanks to the partial input, the rest of the information is recalled and consequently no address is needed. Associative memories are widely used in

practical applications, for instance databases (Lin et al., 1976), intrusion detection (Papadogiannakis et al., 2010), processing units' caches (Jouppi, 1990) or routers (Huang et al., 2001).

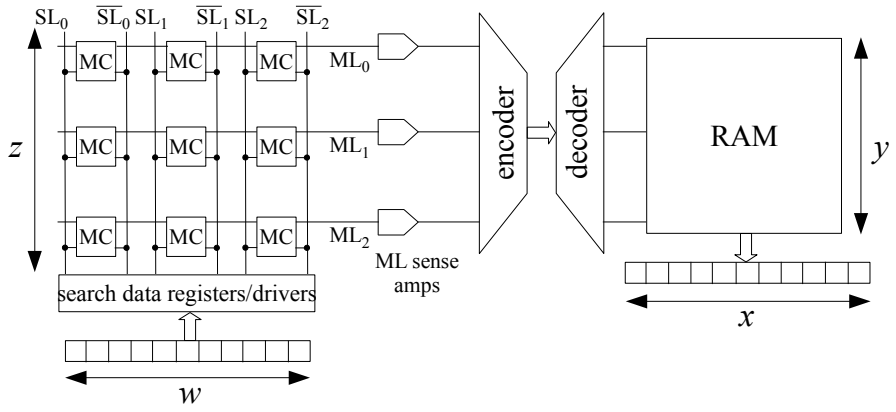


FIGURE 1.6: CAM-SRAM associative memory.

One of classical methods to implement an associative memory is Content-Addressable-Memory (CAM) and Random Access Memory (RAM) couple (Pagiamtzis and Sheikholeslami, 2006). A CAM consists of memory cells (MC), vertical searchlines (SL), horizontal matchlines (ML), and an encoder, Figure 1.6. At the beginning of the operation, all the z matchlines are precharged high corresponding to all the lines matching. The input to the system based on a CAM and RAM couple is the *search word* of length w. This search word is broadcast through the searchlines on the matrix of memory cells. It is then compared in parallel to all the memory cells. All the matchlines that have at least one bit that is different, are discharged. The state of the matchlines is fed into the encoder that outputs the encoded address of the location where the *output word* corresponding to the search word is stored. This address is then decoded and used to access the output word of length x stored in the RAM with y entries. Since low-power integrated circuits are targeted in this work, the Static RAM (SRAM) is chosen as the memory. The variables z and y are not necessarily the same. For example, there may be several search words leading to the same output word. That is why, in general case, we include two variables.

1.7.2 CAM-SRAM as a decision unit

As we have seen before, low-level decision units do not offer a sufficient quality of control. High-level decision units provide a decision that is close-to-optimal, yet they are more complex, energy consuming and slow compared to DVFS actuators. Can an associative memory be used as a decision unit that combines good decision quality, low complexity, energy consumption and fast time response?

The optimization (1.12) can be solved with an offline optimization system for a number of latency constraint L values. The solution is a set of associations between L and T_i values. To each latency value corresponds a set of clock periods T_i (or frequencies f_i) that have to be set by DVFS actuators to minimize the total energy consumption and preserve the latency L constraint.

CAM-SRAM associative memory can be used as power management decision unit that stores in CAM the L values coded on w bits and the corresponding T_i values coded on x bits all stored in one entry of SRAM. Then, during the MPSoC operation, L is used as the search word to find the output word with the T_i set subsequently by DVFS actuators.

Using associative memory simplifies the decision unit since no complex optimization is done during the MPSoC operation. If the application running on the MPSoC is well characterized and all the possible L values are known, we can find the solutions for all the possible cases. In the opposite case, a subset of L values is stored in CAM and the closest one is applied. This may result in a slightly less optimal decision and excessive energy consumption. However, the solutions stored in the associative memory are closer to optimal than, for example, obtained with the game theory decision unit. This is due to the fact that the solutions are found offline with complex mathematical software tools. Therefore, there is a trade-off in terms of the number of stored L values and decision quality. The question of the energy gains with regard to the number of stored L values is analyzed later in Subsection 4.4.3.3.

Since CAM operation is fully parallel, the response is obtained in a single clock cycle. However, CAM relies on a brute-force search, each stored entry is examined for a match. Search process is independent of the search word. Although evaluating the search in parallel speeds up the circuit's operation, it also means that the whole circuit

TABLE 1.2: Decision units state-of-the-art summary

	Low-level	High-level	CAM-SRAM -based	Our goal
Decision speed	+	-	+	+
Decision quality	-	+	+	+
Decision unit's energy efficiency	+	-	-	+

is switching for each search. Consequently, a large amount of power is consumed. The dissipated energy comes mostly from the matchlines that are all charged for each search and then discharged in case of a miss. Searchlines also represent an important part of the CAM's consumption that increases with the memory size (Agrawal and Sherwood, 2006). Additionally, we can imagine scenarios where other constraints are used as the search word depending on the system requirements. For CAMs the way the data is split between the search word and the output word is fixed. Ternary CAM (TCAM) expands CAM functionality allowing "don't care" bits matching both 0 and 1 values, thereby offering more flexibility. However, this comes at an additional cost as the memory cells must encode three possible states instead of the two in case of binary CAM. Consequently, the cost of parallel search within such a memory becomes even greater (Agrawal and Sherwood, 2006). We can see that the classical associative memory lacks energy efficiency combined with flexibility required in adaptive systems as MPSoCs.

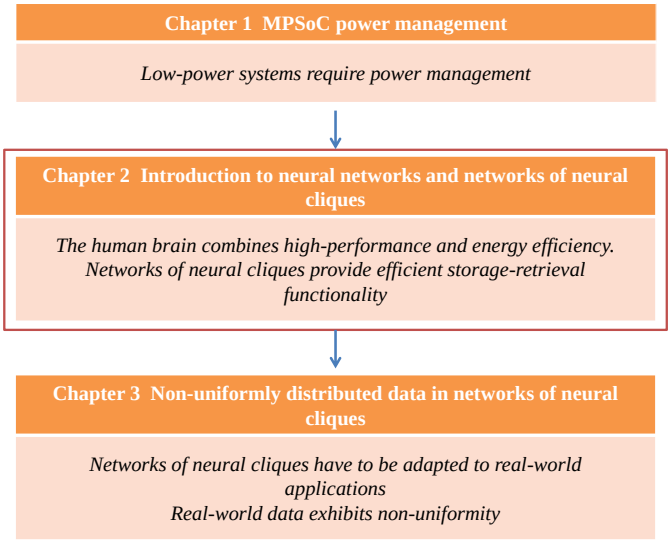
1.8 Conclusion

This chapter introduces the MPSoC architecture and explains how it allows for power management to obtain an energy-efficient operation under applicative constraints. Different types of power management decision units are outlined and three categories, namely low-level, high-level and CAM-SRAM associative memory-based decision units are distinguished. Each type of decision unit has its strong and weak points. Table 1.2 presents the summary of the state-of-the-art decision units. Thanks to their simplicity, low-level decision units allow for high decision speed and the controller itself is energy efficient. However, since they do not have the global scope of the system, they do not guarantee that the decision is globally optimal. This limitation can be addressed with high-level decision units that have the global scope of the system. Nevertheless, this comes with an additional complexity that results in low decision speed and high

decision unit's energy consumption. Thanks to its parallel operation, CAM-SRAM associative memory provides high decision speed. Since the decisions are obtained offline with complex mathematical tools, the decision is close to optimal. Yet, CAM relies on a brute-force approach that results in a high decision unit's energy consumption. Consequently, there is a need for a new approach that offers high decision speed, decision quality and is energy efficient. The chapter that follows, introduces networks of neural cliques that are later applied to obtain a new type of decision unit that replies to these requirements.

Chapter 2

Introduction to neural networks and networks of neural cliques



2.1 Introduction

The previous chapter concluded that associative memories could be a good approach to design a power management decision unit. However, classical methods rely on a brute-force approach that results in a high energy consumption. We propose to use another method. A long-established hypothesis is that some parts of the human brain rely on associations between concepts, *i.e.* a form of associative memory. Moreover, the human

brain is known to be extremely energy efficient. That is why we are interested in artificial models of neural networks. This chapter is devoted to biological and then, artificial neural networks. It is important to know the biological principles to understand how artificial models benefit from those findings. Even if some abstract models are loosely connected to biological equivalents, the fundamentals stay the same. After introducing some of the state-of-the-art architectures, networks of neural cliques are described in detail. Since this recently introduced model works as an associative memory and outperforms state-of-the-art solutions, it is used later in this work.

2.2 Biological neural networks

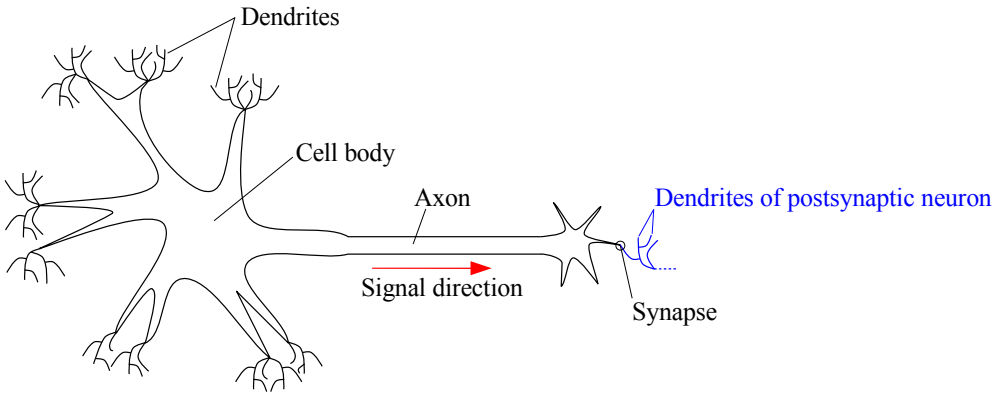


FIGURE 2.1: Biological neural network elements.

Biological neural networks essentially consist of neural cells or neurons and the contact points between the neurons, called synapses. A neuron is made of dendrites, cell body and axon. This is illustrated in Figure 2.1. The cell body contains the nucleus where all the information that the cell needs to grow, repair itself and perform its metabolism is stored. Furthermore, the cell body is filled with cytoplasm with chemicals needed for the cell to perform its functions. Dendrites that extend from the cell body resemble the branches of a tree. At the other end of the cell body is the axon that provides the way to connect to another neuron or other types of cells.

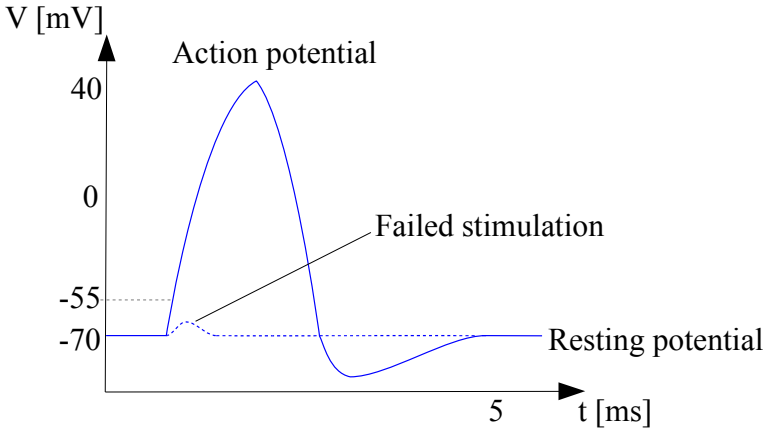


FIGURE 2.2: Biological neuron's potential over time.

Signals in biological neural networks are transmitted electrochemically which means that electrical signals (voltages) are caused by chemicals. A signal enters the cell body through dendrites and leaves the cell body through axon. The place where the signal

passes from one neuron to another, *i.e.* the contact point between the end of the axon and the dendrite, is the synapse. The signal arriving at the end of the axon releases tiny vesicles containing chemical neurotransmitters (Brown, 2001). At the other end of the synapse, the neurotransmitters attach to receptors of the neighboring neuron. These receptors influence the electrical response of the postsynaptic neuron, *i.e.* they modify its excitability. In other words, the postsynaptic neuron is made more (excitation) or less (inhibition) likely to continue to transfer a signal.

The chemicals (sodium, potassium) in the cell body and outside the cell have an electrical charge. The concentrations of the ions (electrically charged chemicals) inside and outside the cell result in a certain potential difference between the cell and its environment. Figure 2.2 shows how the neuron's potential changes over time. When the neuron does not transfer any signal, it maintains a resting potential which is equal to about -70mV. This means that the potential inside the neuron is 70mV less than outside. If the sum of the signals received at the synapses reaches a certain threshold, an action potential is released. The action potential is often called a *spike* or an *impulse*. The stimulus at the synapses causes some of the positively charged ions (sodium ions) to move inside the neuron and cause its depolarization. Consequently, the potential of the neuron increases. When it reaches around -55mV, the action potential is fired, *i.e.* the potential of the neuron abruptly increases. The behavior of all neurons is binary, if the threshold is not reached the action potential is not fired. Otherwise, the full action potential is released. This is known as the *all-or-none* principle. When the action potential is fired, some of the positively charged ions (potassium ions) leave the cell and reverse the depolarization (repolarization). The neuron potential reaches back -70mV. A temporal period follows where the potential reduction continues below -70mV (hyperpolarization). Gradually, the ion concentrations return to the resting state. Neurons need several milliseconds to react to a stimulus. This is rather slow compared to electronics which achieve switching times of a fraction of nanosecond.

The human neocortex accounts for 76% of the brain's volume and is responsible for such functions as conscious thought, language or spatial reasoning. There are roughly 20 billion neurons in the human neocortex (Pakkenberg et al., 2003) each neuron having between 1000 and 10000 synapses. The neurons are organized in cortical microcolumns (Johansson and Lansner, 2007). Each microcolumn contains between 80 and 120 neurons. Microcolumns form macrocolumns (also called hypercolumns). Each macrocolumn

contains from 50 to 100 microcolumns. Several macrocolumns are grouped to form a functional area of the brain. A long-established hypothesis is that neocortex operation relies on associations between concepts, *i.e.* it is a form of associative memory (Palm, 1982, Rolls and Treves, 1997). The authors of (Jones, 2010) and (Johansson and Lansner, 2007) state that the microcolumn is the most basic information processing unit in the brain, not a single neuron. This hypothesis is supported by the fact that the neurons within the microcolumn are reciprocally connected (Thomson and Bannister, 2003) and consequently, they are all driven by the same stimulus.

To summarize, biological neural networks consist of elementary units as input channels, a cell body and an output channel. The cell body performs a simple function as the input signals' summation and responds only if a certain condition occurs on its input channels. Biological neural networks have a distributed structure with no central control unit. Artificial neural networks rely on these principles as well.

2.3 Artificial neural networks

Artificial neural networks have been studied since the 40s. Some of the proposed artificial neural networks build on biological findings and therefore, have a certain degree of biological realism (or plausibility). Such theoretical models may help to understand the human brain and, for instance, allow developing new methods to treat neurological disorders. In other cases, the objective is to construct highly parallel information-processing systems, somewhat inspired from biological neural networks. Therefore, abstract models are proposed where biological plausibility is not the issue in hand. Such biologically inspired information-processing systems are more effective than classical algorithms in a certain group of applications.

Since artificial neural networks form a broad research field, we present a selection of a few models. Firstly, we introduce the historically important model (McCulloch-Pitts model). Next, we describe three models that have different objectives: 1) memorize (Hopfield Neural Networks), 2) model and compute (Spiking Neural Networks), 3) learn (Deep learning).

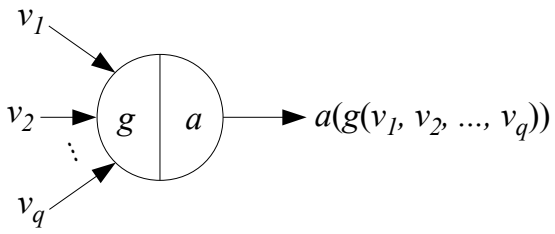


FIGURE 2.3: Generic neuron model.

2.3.1 McCulloch-Pitts model

Some of the earliest research comes from McCulloch and Pitts resulting in the proposal of McCulloch-Pitts networks (Pitts and McCulloch, 1947). It is assumed that the synapses are represented with connections between the neurons and that they are unidirectional, *i.e.* they transmit signals in a predetermined direction. For McCulloch-Pitts networks there is no limit on the number of connections going out from a neuron. This property is called the *unlimited fan-in*. Figure 2.3 depicts a generic neuron model. The operation of a neuron consists in two functions: an integration function g and an activation function a. The integration function g reduces the q signals v received at the input of the neuron to a single value. This single value is used as an argument to the activation function a that computes the output of the neuron. In McCulloch-Pitts networks the connections transmit only ones or zeros, that is they are unweighted. The connections are of *excitatory* or *inhibitory* type. To distinguish the type of connections, excitatory connections are labeled with v and inhibitory connections with ζ. The integration function g is the sum of the input signals:

$$g(v) = \sum_{i=1}^{q} v_i. \tag{2.1}$$

The activation function is the step function with threshold σ:

$$a(g(v), \zeta) = \begin{cases} 1 & \text{if } g(v) \geq \sigma \\ 0 & \text{if } g(v) < \sigma \vee \exists \zeta = 1. \end{cases} \tag{2.2}$$

The neuron's output equals one if there are no inhibitory signals at the input and the result of the integration function g is greater or equal to the threshold σ. Figure 2.4 shows a McCulloch-Pitts neuron with two excitatory inputs and one inhibitory input signal.

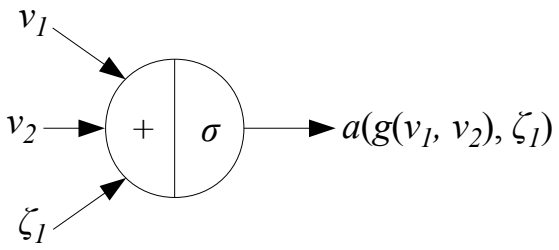


FIGURE 2.4: McCulloch-Pitts neuron.

McCulloch-Pitts neurons can be organized in networks to realize more complex models. More specifically, they can be organized in layers, so that the neurons in the second layer receive signals only from the outputs of the neurons in the first layer. For example, they can be used to compute logical functions. It is proven (Rojas, 1996) that a McCulloch-Pitts network of two layers can realize any logical function $F : \{0,1\}^q \rightarrow \{0,1\}$. Furthermore, weighted connections transmitting real values can be introduced. Also, the aforementioned absolute inhibition can be replaced with relative inhibition where connections are weighted with negative factors. It is shown that all this variations result in equivalent networks. The choice between different variants is a trade-off between the complexity of the network's topology and the complexity of its elements (Rojas, 1996).

2.3.2 Hopfield Neural Networks - memorize information

Since the neocortex operation relies on associations between concepts, artificial neural networks for associative memory are also proposed. The concept of associative memory is introduced in Section 1.7. The most prominent model in this category is proposed by Hopfield in (Hopfield, 1982). Throughout this document, we consider that neuro-inspired associative memories store messages (also called patterns) that they are later capable of retrieving given a sufficiently large part of their content. The message definition is detailed in the following section. Hopfield Neural Networks (HNNs) consist of neurons that are all interconnected with each other except being connected to itself. The connections are weighted and their values are restricted to integers. Each neuron in the network of n neurons has its index. Supposing that the set of messages to store contains M messages $m^1, m^2, ..., m^M$ the weight between neurons i and j is obtained as

follows:

$$w_{ij} = \begin{cases} \sum_{k=1}^{M} m_i^k m_j^k & \text{if } i \neq j \\ 0 & \text{otherwise.} \end{cases} \tag{2.3}$$

Therefore, the messages are projected onto the connection weights. HNNs can store messages of length n.

After adapting the weights, HNNs can be used to retrieve a previously stored message when only a part of it is known. The state of neurons represents the values in the message. Initially, all the neurons are set to zero. Then, the neurons corresponding to the known part of the message are set to their values. Classically, during the retrieval procedure, neurons in HNNs can present two states: -1 or 1. However, they can be modified to work with 0 and 1 states as well. As earlier, the inputs of neuron are denoted by v and the output (or state) of neuron is denoted by a. To retrieve a message, HNN is subject to several iterations in which the states of neurons are updated such that:

$$a_i = \begin{cases} 1 & \text{if } \sum_{j=1}^{n} w_{ij} v_j \geq 0 \\ -1 & \text{otherwise.} \end{cases} \tag{2.4}$$

When HNN converges to a stable point, that is the states of all the neurons do not change between two iterations, the retrieved message is obtained by reading the states of the neurons. The convergence analysis of HNNs is provided in (Rojas, 1996).

2.3.3 Spiking Neural Networks - model biological networks and compute

Spiking Neural Networks (SNN) form another family of artificial neural networks. The spikes represent the action potential from biological neurons. The main aim of SNNs is to realize neural computation. This implies that spikes are related to the quantities relevant to the computations. The main assumption underlying SNNs is that the behavior of neurons depends on the timing of spikes rather than their specific shape or amplitude (Gerstner and Kistler, 2002). Numerous experiments on animals show that the timing of spikes is a means for coding information (Bialek et al., 1991, Heiligenberg, 1991, Kuwabara and Suga, 1993). A few models to describe the behavior of SNN's neurons are proposed. The most biologically realistic is the Hodgkin-Huxley model (Hodgkin

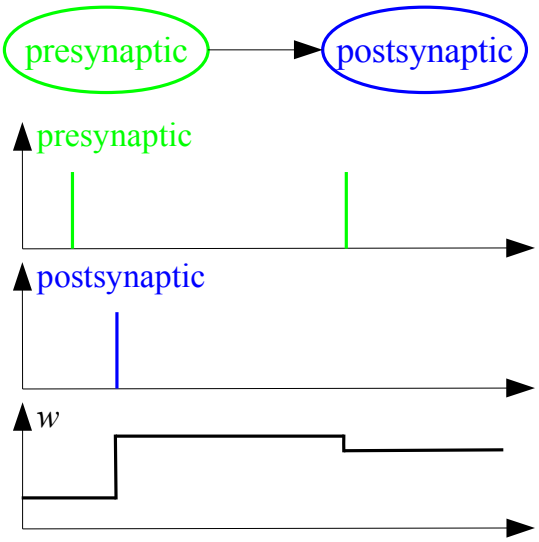


FIGURE 2.5: STDP weight modification.

and Huxley, 1952). Despite the fact that it compares well to data from biological experiments, its complexity entails difficulties in simulations of large networks. That is why several simplified models as Leaky-Integrate-and-Fire (Lapicque, 1907, Stein, 1965) or Izhikevich's (Izhikevich, 2003) neurons are proposed. The connection weights of SNNs are modified based on the coincidence of pre- and postsynaptic spikes. Spike-Timing-Dependent Plasticity (STDP) is a commonly used approach (Song et al., 2000). Figure 2.5 illustrates its principle. In this rule, the weight of a connection is increased if a presynaptic spike is followed by a postsynaptic spike. If a presynaptic spike fires after the postsynaptic spike the weight of a connection is decreased. The change in the weight depends exponentially on the difference in time domain between the spikes.

Comprehensive descriptions of SNNs are given in (Paugam-Moisy and Bohte, 2009, Vreeken, 2003, Ponulak and Kasinski, 2011, Gröning and Bohte, 2014). Neural computation relying on SNNs is used, for instance, to design hardware accelerators (Belhadj et al., 2014a). Such an architecture can be used in multiple computing applications thanks to the possibility to adapt the weights of the connections.

2.3.4 Deep learning - learn information

Currently, in many domains (e.g. vision) state-of-the-art methods for learning are based on *deep learning*. Deep learning allowes adapting weights of neural networks with multiple layers, typically ten hidden layers (hidden layer is a layer that is used neither as input nor output of the network). Networks made of multiple hidden layers prove to be more efficient than shallow networks (with a single hidden layer) (Bengio, 2009). Moreover, a network with a depth insufficient for the targeted task requires more neurons than a network with a depth matched to the problem. In addition, some studies show that the human brain processes information through multiple stages of transformation and representation, i.e. a type of deep architecture (Serre et al., 2007).

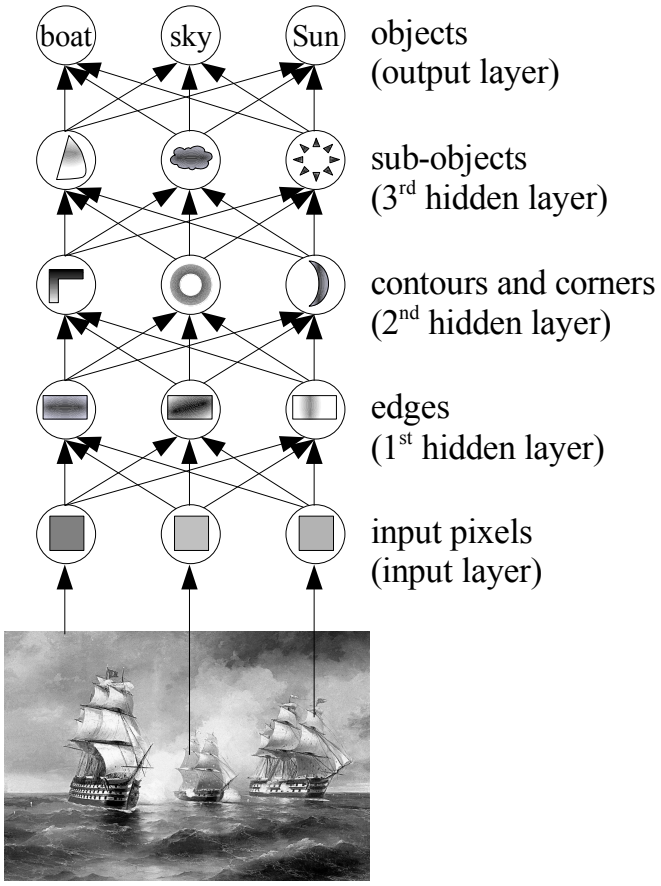


FIGURE 2.6: Illustration of deep learning.

The core idea behind deep learning is introducing representations that are expressed by other, simpler representations. In other words, building complex concepts out of simple concepts resulting in a hierarchical structure. This allows a deep network to learn abstract information from raw data, for instance, a natural language description

of an image obtained from a set of pixels. To illustrate this, Figure 2.6 shows the stages of identifying objects on an image by a neural network made of multiple hidden layers. This complex problem is broken in series of simpler, increasingly abstract tasks. Firstly, the pixels are input to the network. Based on the pixels, we can identify edges by comparing the brightness of the adjacent pixels. The following layer, that is fed with the edges, finds the contours and corners which are basically collections of edges. Given the contours and corners, the next layer, finds specific collections of contours and corners and detects sub-objects. Based on the collections of sub-objects, objects present in the image are identified.

The main factor limiting the application of deep neural network architectures is adapting the parameters of the network, *i.e.* learning. The first successful report on deep learning comes from 2006 when Deep Belief Networks (DBN) are introduced (Hinton et al., 2006). The often used principle is to greedily learn each layer locally. Other methods exploiting similar idea are also proposed (Bengio et al., 2007, Ranzato et al., 2007, Weston and Ratle, 2008, Mobahi et al., 2009).

Artificial neural networks form a broad research field. Here, we give descriptions of one of the first introduced models (McCulloch-Pitts model), a neuro-inspired associative memory (HNN), a common model to perform neural calculations (SNN) and the most powerful, yet the most complex architecture (deep learning). The following section introduces a recently proposed neural network model on which builds the rest of this work.

2.4 Networks of neural cliques

As concluded in Section 1.8 an associative memory can be used as a power management decision unit. Consequently, we focus on approaches that realize an associative function. In practice one can distinguish two main families of associative memories, namely CAM-SRAM and neuro-inspired memories. A classical way to implement such a controller would be a CAM-SRAM associative memory. Although it allows for fast and close-to optimal decision, brute-force approach results in a high energy consumption. Moreover, it lacks the flexibility required in adaptive systems as MPSoCs.

In order to assess the performance of associative memories, several parameters can be introduced. A crucial one (Gripon and Rabbat, 2013) is termed memory efficiency and is defined as the best ratio of the total number of bits stored to the total number of bits used to store the memory itself, for a targeted performance. Note that this ratio is trivially one for indexed memories. Another important parameter is the computational complexity that depends on the operations needed to perform the retrieval. Again, this parameter makes usually no sense for traditional memories in which computational complexity is often considered to be of $\mathcal{O}(1)$. In terms of physical implementation, the efficiency impacts circuit's area, whereas the computational complexity dynamic power consumption.

Neuro-inspired associative memories combine lower complexity with higher flexibility, at the cost of reduced efficiency. We already know HNN model in which stored messages are projected onto the connection weights of a fully interconnected set of neurons. Nevertheless, when the size of this network is increased the efficiency decreases (Berrou and Gripon, 2010). This limits opportunities for large networks and is not satisfactory in terms of biological plausibility. Sparse networks originally proposed by Willshaw (Willshaw et al., 1969) use a small subset of connections to store each message, resulting in a much better efficiency (Palm, 2013). Further, works from (Salavati and Karbasi, 2012) are also known to allow for storing large number of messages.

Recently Gripon and Berrou proposed a new model (Gripon and Berrou, 2011a) that can actually be seen as a particular Willshaw network with cluster structure. This modification allows for efficient retrieval algorithm without diminishing performance. The network (Gripon and Berrou, 2011a) relies on a neural *clique* which is an assembly of neurons representing a piece of information stored in the neural network. The elementary part of information is called message and is associated with the clique. This model is able to store a large number of messages-cliques and retrieve them, even when a significant part of the input is erased. The simulations of the network working as a data structure (used to check if a given information is known by the network, can be used as an intrusion detection system) or an associative memory proved a huge gain in performance compared to HNN (when using comparable material) (Gripon and Berrou, 2011a). The fact that the network is able to retrieve messages with erasures on any position, or in the presence of noise, gives it an advantage over CAMs (and TCAMs that allow erasures on more positions than CAMs but still not all the positions). These

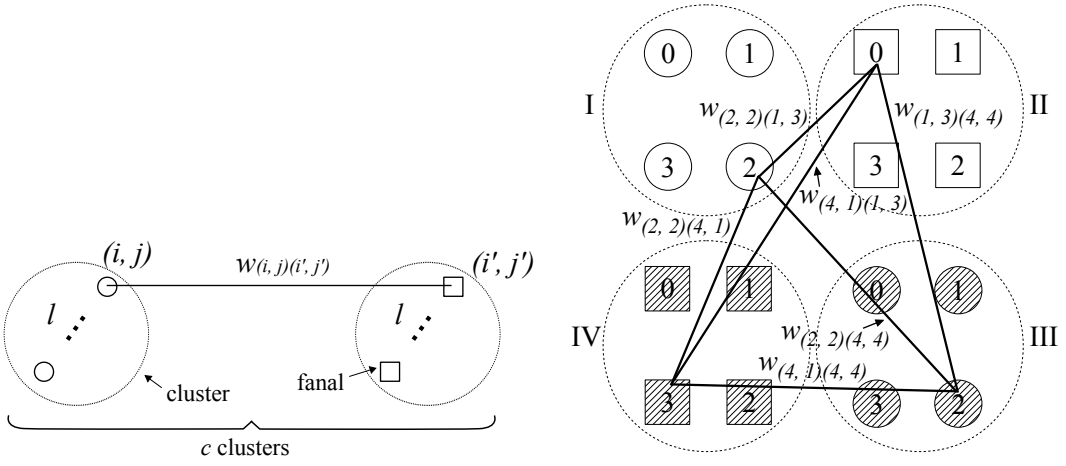
interesting properties originate in error correcting codes that underlie this network's principles (Gripon and Berrou, 2011b).

The rest of this chapter outlines the theory of networks (Gripon and Berrou, 2011a) that are called *networks of neural cliques* in the remaining part.

2.4.1 Message definition

Throughout this work, it is considered that associative memories store *messages* m that they are later capable of retrieving given a sufficiently large part of their content. In order to ease the readability of this document, and without loss of generality, it is considered that a message consists of c sub-messages or segments. Each segment is a value in range from 0 to $\ell - 1$. An exemplary message, for $c = 4$ and $\ell = 4$, is $m = \{2,\ 0,\ 2,\ 3\}$. It is sometimes convenient to think of messages as binary vectors of given length. A simple conversion of the above message gives $m = \{10,\ 00,\ 10,\ 11\}$.

2.4.2 Network structure



(a) (b)

FIGURE 2.7: (a) The network general structure and notation. Different shapes (circles, squares) represent fanals belonging to different clusters. (b) Exemplary network with $c = 4$, $\ell = 4$ and message $m = \{2,\ 0,\ 2,\ 3\}$ stored. The numbers inside each fanal correspond to its index.

In order to store messages, we use a network that consists of binary neurons and binary connections. The authors of (Gripon and Berrou, 2011a), use the term *fanal* (which means lantern or beacon) instead of neuron for two reasons: a) at a given moment,

in normal conditions, only one fanal within a group of them can be active and b) for biological inspirations, fanals do not represent neurons but microcolumns (Aliabadi et al., 2013). Figure 2.7a represents the general structure of the network and the notation. All the n fanals are organized in c disjoint groups called *clusters*. Fanals belonging to different clusters are represented with different shapes. Each cluster groups $\ell = n/c$ fanals. A node in the network is identified by its index (i, j), where i corresponds to the cluster number and j to the number of the fanal inside the cluster. The connections are allowed only between fanals belonging to different clusters, i.e. it is a multipartite graph. The connection between two fanals is denoted by a binary weight $w_{(i,j)(i',j')}$. Contrary to, for example, HNN, the connections do not possess different weights, the connection exists or not. Hence, the weight (or adjacency) matrix of such a network consists of values $\{0, 1\}$ where 1 indicates the connection between two fanals, and 0 the lack of the connection. Figure 2.7b shows an example of a network with $c = 4$ and $\ell = 4$.

2.4.3 Message storing procedure

To store a message m in the network, each of its c segments is associated with a distinct cluster, and more precisely with a unique fanal in its cluster (the one which index corresponds to the value of the segment). Then, this subset of fanals is fully interconnected forming a clique representing the message in the network. This term is also used in neurobiology to describe such groupings of neurons (Lin et al., 2006). When a new message shares the same connection as an already stored message, this connection remains unchanged. Therefore, the result of the storage procedure is independent of the order in which messages are presented to the network. As an example, in Figure 2.7b, message $m = \{2,\ 0,\ 2,\ 3\}$ is stored. Dividing the network into clusters, reduces the number of possible connections. Moreover, each message is stored using a small number of fanals. Such a message representation leads to a good minimal distance δ between cliques that is equal to:

$$\delta = 2(c - 1). \tag{2.5}$$

The minimal distance δ between two cliques stored in a network with $c = 4$ clusters equals six. This property allows for good distinction between cliques.

2.4.4 Message retrieval procedure

We call retrieval the process of retrieving a previously stored message when only part of its corresponding fanals is known. After the storing of all messages, the retrieval process is organized as follows. First, the known segments of the input message are used to stimulate appropriate fanals, i.e. the value on the given segment indicates which fanal should be chosen. These fanals are said to be active. After the initial stimulation, message passing phase comes next. The activated fanals send unitary signals to other clusters through all of their connections. Then, each of the fanals calculates the sum of the signals it received. However, as shown in (Gripon and Berrou, 2012), summing only the signals from distinct clusters slightly improves the performance. Within each cluster the fanal having the largest sum is chosen and its state becomes 1, i.e. it is made active. The other fanals inside the cluster present the state equal to 0. The rule according to which the active fanal inside the cluster is elected is called *Winner Takes All* (WTA). Such a rule is also present in biological neural networks (Lee et al., 1999, Coultrip et al., 1992). Other rules, such as Losers Kicked Out (LsKO) are also studied Aboudib et al. (2014). Here, we use the WTA rule. By reading the state of the fanals in the unknown clusters, one can identify the values on the missing segments. The whole process may be iterative, allowing the fanals to exchange information with each other, such that ambiguous clusters (those containing more than one active fanal) will hopefully be correctly retrieved. When more than one iteration is needed (input with significant noise or erasures resulting in non-unique fanal with the largest sum) an extra value is added to the score of the last winners in the next iteration. More details on adjusting this memory effect are given in (Gripon and Berrou, 2011a). Thanks to this iterative retrieval procedure the network converges step-by-step to the targeted previously stored message. In some cases though, it may happen that the output message is not correct, leading to nonzero error probability in the retrieval process.

As a result of the strong correlation brought by the connections of the clique embodying a message in the network, it is possible to retrieve the stored message based on partial or noisy information put into the network.

2.4.5 Density and error probability definitions

As previously stated, storing messages in networks corresponds to creating subgraphs (cliques) of interconnected fanals. When the number of stored messages increases, these subgraphs share an increasing number of connections. Therefore, distinguishing between messages is more difficult. As a logical consequence, there is an upper bound on the number of distinct messages we can store then retrieve for a targeted maximum error probability. The network density d is defined as a ratio of the established connections to all the possible ones. Therefore, the density is a parameter of first importance to assess the network's performance. A density close to 1 corresponds to an overloaded network. In this case the network is not able to retrieve stored messages correctly. For a network that stored M uniformly distributed messages expected density d is expressed by the following formula or its first-order approximation:

$$d = 1 - \left(1 - \frac{1}{\ell^2}\right)^M \approx \frac{M}{\ell^2} \text{ when } M \ll \ell^2. \tag{2.6}$$

The probability of correctly retrieving a message with c_e positions erased in a network constructed of c clusters is given by:

$$P_c = \left(1 - d^{c-c_e}\right)^{(\ell-1)c_e}. \tag{2.7}$$

This equation is valid for a single iteration. The probability of error increases with d. Note that in case of large density, simulations confirm that iterations improve the ability of the networks for retrieving messages correctly.

Figure 2.8 shows the evolution of the error retrieval rate for a network with $c = 8$ and $\ell = 256$. Half the clusters are not provided with information. This means that only four randomly chosen segments of a message are known, the remaining are erased. Hence, only four fanals are initially stimulated in four clusters. Figure 2.8 shows a theoretical curve for a single iteration and the network density. We see the interest of the iterative character of the retrieval procedure when comparing the one-iteration curve with the curve where four iterations are performed. The simulation shows that the network of $n = 2048$ fanals can store up to 15000 uniform messages of 64 bits each and retrieve them with a very high probability (error rate 0.029 for four iterations allowed) even when half the clusters are not provided with information. As claimed in (Gripon and

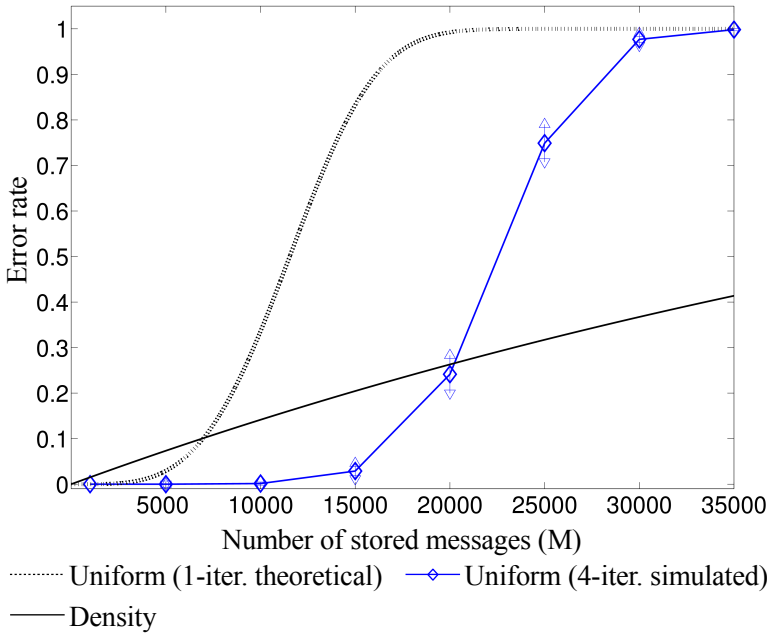


FIGURE 2.8: Evolution of the error rate with regard to the number of stored messages. The network composed of eight clusters of size 256, four randomly erased positions. For each point 100 networks are evaluated doing 100 tests per network. The arrows indicate standard deviations of the error rates.

Berrou, 2011a), compared to state-of-the-art and for the same amount of material used, this is unprecedented performance.

2.4.6 Neural cliques as associative memory

Neural cliques rely on a different principle than CAM-SRAM associative memory. The content of the input (known segments of the message) guides the retrieval process. The values present in the input indicate the fanals that are stimulated. Then, these fanals send signals only to the fanals that were connected to them during the storing procedure. As a result of the retrieval procedure, the fanals corresponding to the unknown segments are activated and the missing segments are retrieved. In terms of hardware implementation, all the parts of the circuit that are not related to the searched data are not activated. This is not the case for CAM-SRAM (described in Subsection 1.7.1) and therefore, neural cliques are a good candidate for low-power associative memories.

Interestingly, biological studies show that in the brain only 1-4% neurons are active at a given time (Attwell and Laughlin, 2001, Lennie, 2003). It seems that the brain developed similar type of sparse activity representation that allows for exceptional energy efficiency.

2.4.7 Network dimensioning guidelines

The dimensions of networks of neural cliques are defined by the number of clusters c and their size ℓ. The clusters do not necessarily have to be of the same size, *i.e.* we can have c_1 clusters of size ℓ_1, c_2 clusters of size ℓ_2 and so on.

The dimensions depend on the application and the designer's objectives. In some applications the objective is to store and retrieve the largest possible number of messages with the lowest error rate. In others, the main goal is to reduce the complexity of the network, even at the cost of retrieval performance. Most often a compromise between the two is necessary.

Given a number n of fanals, bigger clusters allow to store more messages. Since ℓ is larger, the connection density increases slower with the number of stored messages (*cf.* 2.6). On the contrary, if ℓ is fixed, increasing c does not allow to store more messages.

Given a length of messages to store, we can dimension the network in a number of ways. Namely, given messages of 64 bits, we can organize the network in eight clusters of 256 fanals. This results in $n = 2048$ fanals and 1835008 possible connections. It is shown in Section 2.4.5 that in such a network we can store 15000 messages and retrieve them with a very low error rate. However, if we do not want to store that many messages, and their length is still 64 bits, we can reduce ℓ to obtain lower complexity. For instance, a network with $\ell = 16$, $c = 16$ allows to store messages of 64 bits as well. Since the density grows faster as the messages are stored, the error rate increases faster as well. However, n equals now 256 and the number of possible connections is 30720.

Furthermore, if all the clusters are not of the same size, it is preferable to avoid large differences in their sizes. Large inequalities lead to high densities in small clusters and may hamper the retrieval process. In this case small clusters can be combined to form a bigger one. For instance, two clusters of four fanals that represent two bits in messages, can be combined to form a cluster of 16 fanals representing four bits.

Other aspects come into play when moving toward real-world applications where physical parameters are stored in networks. It is recommended to assign a cluster to each physical parameter. Otherwise, if the number of distinct values of one of the parameters is increased, the number of fanals in the cluster explodes. That is because all the possible combinations of parameters have to be represented. In case parameters are separated in distinct clusters, only one new fanal is necessary.

Further adjustments can be done when application characteristics are taken into account. Networks can be divided into clusters of different types, *i.e.* input clusters, output clusters, input/output clusters. This is applicable if the application characteristics are known beforehand, namely, when we know that a subset of segments in messages are always used as inputs, others are always used as outputs and some are used as inputs or outputs depending on the current application requirements. Fanals in input clusters are simplified by removing the WTA functionality, whereas fanals in output clusters are simplified by removing external stimulation functionality. Additionally, in some cases, it is possible to remove some connections. This may be the case for output clusters. If the stimuli received from the input clusters are enough to differentiate between fanals in output clusters, there is no need to keep the connections between the output clusters.

The following list summarizes the network dimensioning guidelines:

- Bigger clusters allow to store more messages.
- Increasing the number of clusters c does not allow to store more messages.
- The complexity can be traded for the maximal possible number of messages to store.
- It is preferable to avoid large differences in cluster's sizes.
- It is recommended to assign a cluster to each physical parameter.
- One should take advantage of application's characteristics to simplify the network.

2.5 Conclusion

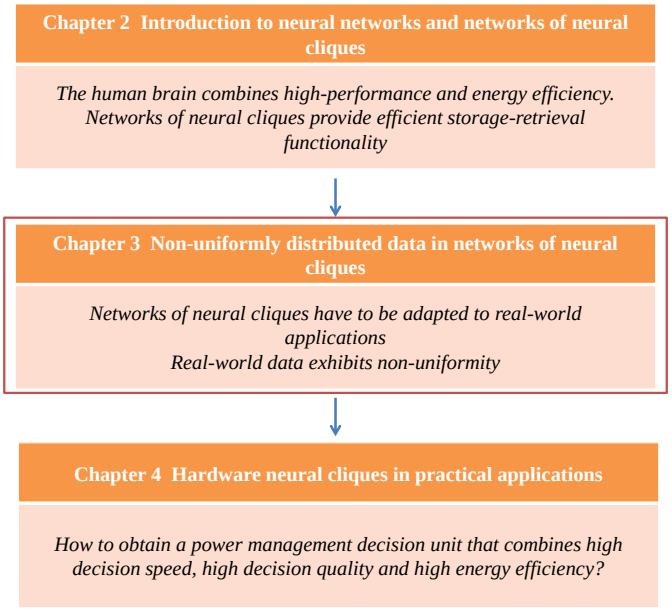
In this chapter basic principles of biological neural networks are introduced. Further, several artificial neural networks are described. Firstly, some of the architectures present

in state-of-the-art, then networks of neural cliques on which relies the present work are described. The essential definitions and notation used in subsequent chapters are given. Neural cliques associative memory shows unprecedented performance in terms of number of messages that can be stored and retrieved successfully. During the network's functioning there are always only a few parts that are active, *i.e.* the activity in the network is *sparse*. Similarly to the human brain, this is crucial for energy efficiency.

The notion of density is of special importance. As is it shown in the next chapter, density is crucial for obtaining good performances, especially in practical applications.

Chapter 3

Non-uniformly distributed data in networks of neural cliques



3.1 Introduction

The network as presented in (Gripon and Berrou, 2011a) is analyzed only for uniform i.i.d. (independent identically distributed) values among all the messages. In terms of the network construction this means that the number of connections going out from

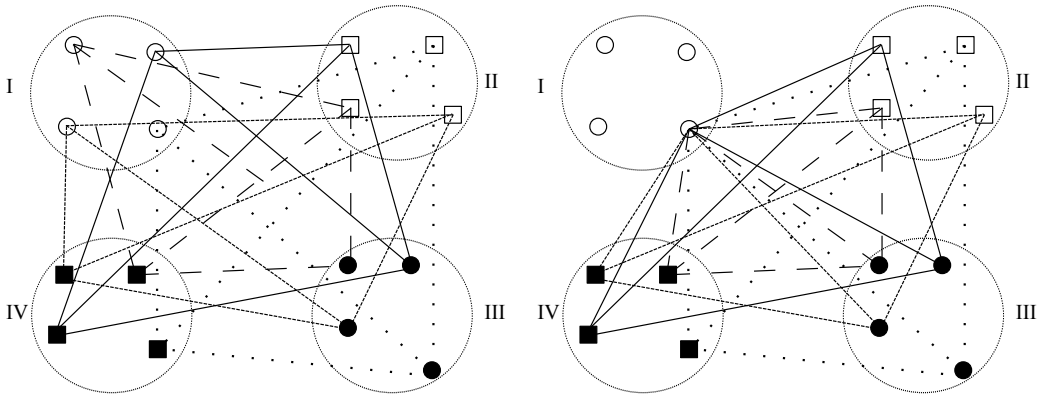
each node is uniformly distributed across the whole network. It is well known that non-uniformity of messages to store can lead to dramatic decrease in performance (Knoblauch et al., 2010). On the other hand, it is expected that real-world applications may contain highly correlated data.

In this chapter, the situation where non-uniform data is stored is analyzed and its influence on the network performance is explained. In order to approach the theoretical performances in real-world applications, the model needs to be improved. Further, we exploit the structures of these networks to introduce several techniques in order to efficiently store non-uniform data (Boguslawski et al., 2014). The most efficient method relies on a concept of twin neurons and is analyzed in depth both formally and by simulations (Boguslawski et al., 2015a). Later, networks of neural cliques are applied to dynamic power management applications. The networks are assessed in a practical context using real-world data from simulations and measurements. It is shown that with such a data standard networks become inoperative and one definitely needs to adapt the model. In order to approach the theoretical performance of the model, twin fanals are used when real-world data is stored. The results show that this solution approaches performances close to uniformly distributed data. Note that when considering the performances in terms of practical applications, not only error rate is relevant. Error rate is an important indicator of the networks functioning when theoretical data is used. As soon as neural cliques are applied to a practical application with real-world data, it is particularly interesting to asses their operation with applicative performance indicators that estimate the impact on the application. Moreover, putting all errors in one category is not precise enough. When physical parameters are stored in the networks, a large difference between a retrieved value and the correct one has more impact than a small one. This is taken into account in this chapter where networks' performance is assessed in terms of physical parameters.

3.2 Non-uniform distribution problem positioning

Figure 3.1 represents a case of a network made of four clusters. There are four fanals per cluster. Four messages are stored, each type of the line representing a different clique.

Figure 3.1a shows a network with uniformly distributed messages. Each fanal in each cluster has the same number of connections. However, for another set of messages stored in the network, some fanals can have much more connections than the others. This is illustrated in Figure 3.1b. In all of the clusters except cluster I, each node has the same number of connections. This means that on the segments II, III, IV of the messages each of the four possible values occurred. However, in the cluster I, only one of the fanals is always used, i.e. the value on the first segment is constant.



(a) (b)

FIGURE 3.1: (a) Network with uniformly distributed messages. (b) Network with non-uniformly distributed messages. Different shapes (filled or empty circles or squares) represent fanals belonging to different clusters, different types of lines represent connections belonging to distinct cliques. Clusters are numbered - they represent segments of messages.

This simple example depicts how the distribution of data stored in the network maps to the interconnection structure.

Figure 3.2 shows the evolution of the error retrieval rate for a larger network with $c = 8$ and $\ell = 256$. Half the clusters are not initially stimulated. The network of $n = 2048$ fanals can store up to 15000 uniform messages of 64 bits each and retrieve them with a very high probability (error rate 0.029 for four iterations allowed). However, when the messages are generated from the truncated Gaussian distribution (mean $\mu = 135$, standard deviation $\sigma = 25$), only 2000 messages can be stored (error rate 0.047). The

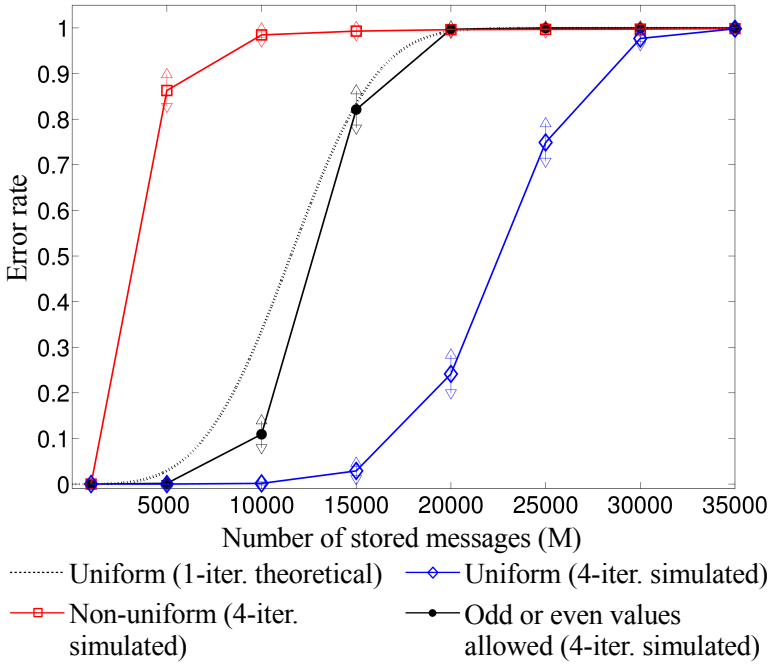


FIGURE 3.2: Evolution of the error rate with regard to the number of stored messages for different types of data distributions. The network composed of eight clusters of size 256, four randomly erased positions. For each point 100 networks are evaluated doing 100 tests per network. The arrows indicate standard deviations of the error rates.

truncated Gaussian distribution, contrary to the uniform one, implies that on a given segment of messages some values occur much more often than the others. Figure 3.2 presents also a curve (indicated with a dot) for a data with a specific correlation between the values within each message. Within each message either odd or even values are only allowed. This means that if on the first segment there is an odd value, one knows that all the other values are odd (e.g. $m = \{1\ 3\ 5\ 7\ 9\ 11\ 13\ 15\}$ in decimal). For this dataset the network can store 8000 messages and retrieve them with the same error probability as in the uniform case. The correlation in the stored data clearly shifts the curve toward lowest number of stored messages. In this example, the same network filled with normally distributed data, can store only 13% of the number of uniformly distributed messages.

As in most applications data cannot be considered uniform, it is of first importance to introduce techniques to counterbalance the effects of correlation on performance.

3.3 Strategies to store non-uniform data

In the following section we propose several strategies to store non-uniform data. They essentially all consist in adding spatial diversity to the networks.

3.3.1 Random clusters

The first of the strategies relies on adding to the existing network, clusters filled with random values drawn from a uniform distribution. These random values stored in *random clusters* play a role of a so-called *stamp*, providing the existing clique with additional information and supporting the message retrieval process. This way the influence of local high density areas caused by non-uniform data is lowered and eventually neutralized for small-enough density. Figure 3.3 illustrates the network from Figure 3.1 with one random cluster added and one message stored. When using this technique, each message comes with an additional segment (or several segments) which holds a value randomly generated from an arbitrary chosen range. Then, during the retrieval, only a subset of known non-random segments is used to initially stimulate the network. Through the established connections they stimulate all the other clusters, random clusters as well. Then, the latter stimulate the others and give additional support to the retrieval process.

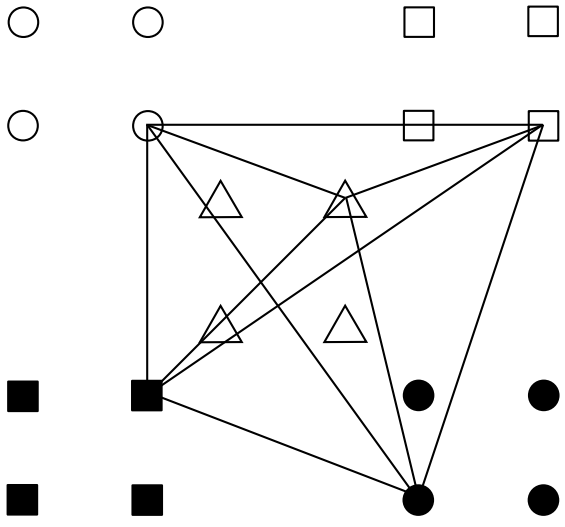


FIGURE 3.3: Network with one random cluster (represented by triangles) added.

The main advantage of this technique is the fact that no additional processing on the stored data is needed, the values that are stored in the network are directly used for the

initial stimulation. Thanks to the additional clusters, the necessary treatment is done by the network itself.

Similar technique is introduced in (Knoblauch et al., 2010). The authors propose adding to a feedforward neural network an additional intermediary layer filled with random patterns, to improve the performance of a single-layer model in case of non-random data. Since the two models have much in common, this technique is treated as a reference for the rest of the introduced strategies.

After explaining other techniques, it is presented how they improve the performance compared to adding random clusters.

3.3.2 Random bits

Another category of the proposed strategies relies on adding random uniformly generated bits to the existing data rather than whole random values stored in separate clusters. As a consequence, the structure of the network remains the same, only the size of clusters is modified since the range of values is expanded by a number of random bits. In other words, single fanal is no more associated with a given value. For instance, adding one random bit to a segment implies randomly choosing between two fanals associated with a given value.

Besides adding bits simply drawn from a uniform random distribution, we propose another approach to generate additional bits. To each value always the least used combination of bits (or one of the least used) is added. The rule is illustrated with Figure 3.4. For each of the values coded on eight bits a table is created where the number of occurrences of each additional bits' combination is stored. For instance, for the first value consisting of only zeros either 000 or 001 can be chosen. However, for value 00000001 only 001 can be added since 000 is already used once. This procedure continues for all the positions in all the messages.

As it is shown later, this technique offers much better performance compared to random clusters strategy when using comparable material. However, it requires changing the data by adding random bits before storing it in the network. This does not imply higher computational complexity compared to adding random clusters, since both methods

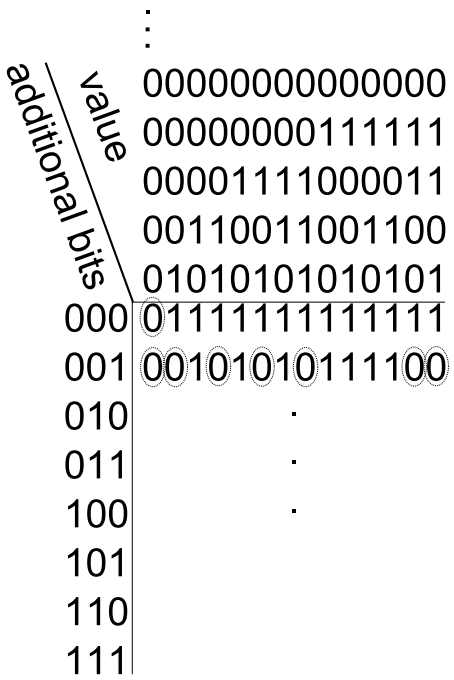


FIGURE 3.4: Adding least used combination of bits. The additional bits that are marked with circle can be added to the initial values.

require only random numbers generation. Nevertheless, manipulating the data before storing it, may be constraining in some applications.

3.3.3 Using compression codes

The last strategy proposed in this section is to apply algebraic compression codes. In this work Huffman lossless compression coding is applied. This technique allows to minimize the average number of coding symbols per message (Huffman, 1952). Note that for some datasets arithmetic coding may perform better than Huffman code, see (Bookstein and Klein, 1993) for comparison.

Huffman coding produces variable length codewords - the values that occur most frequently are coded on a small number of bits, whereas less frequent values occupy more space. One dictionary is constructed for each segment of all the messages. Such a procedure results in variable length segments, the most often occurring values being the shortest. Therefore, the sizes of the frequent values that break the uniformity are minimized. The free space obtained within each segment is filled with random uniformly generated bits. Now, the most often appearing values are associated with the largest

number of randomly chosen fanals. Decoding is possible thanks to the prefix-free property, that is a set of bits (codeword) representing a symbol is never a prefix of another codeword used for the same segment. For instance, a code consisting of {01, 11} is a prefix-free code. Strictly speaking, in order to decode the retrieved encoded message, each segment is compared bit-by-bit with its dictionary. When a codeword is met, one knows that these bits are useful, the remaining being the random ones.

The simulations show that using compression codes is the most effective from the already presented techniques. Thanks to the properties of the used coding technique, the frequent values are effectively redistributed across a group of fanals. Consequently, the influence of local high density areas is minimized. Compression codes, however, require additional operations related to coding and decoding. Since Huffman coding requires analyzing the whole data set before encoding the very first message, this may be a strong constraint in some applications. In this case a variation of Huffman coding such as Adaptive Huffman coding (Vitter, 1987) can be used. In this coding technique, the code is built as the data is transmitted, with no initial knowledge on the distribution.

3.3.4 Performance comparison

In this section performance of all the introduced strategies is evaluated. The simulations are performed for the same network as in Section 3.2 and the same non-uniform distribution.

Figure 3.5 shows the improvement in performance when using the proposed techniques. For random clusters technique, seven random clusters of 5000 fanals each are added. Compared to non-uniform case without using any technique the improvement is clear. The function is non-monotonic since the significant part of the network is filled with random values. To compare the used techniques, each of them is classified by the amount of material it uses. The used material is considered to be the number of possible connections. Number of connections is linearly related to the number of information bits stored in a software or hardware implementation. The rest of the techniques (random bits, least used bits, Huffman coding) use the closest additional material compared to the random clusters strategy. This is obtained for eight clusters of 4096 fanals which corresponds to four additional bits. These strategies offer much better performance using comparable material. Their limits are explored in the rest of this subsection.

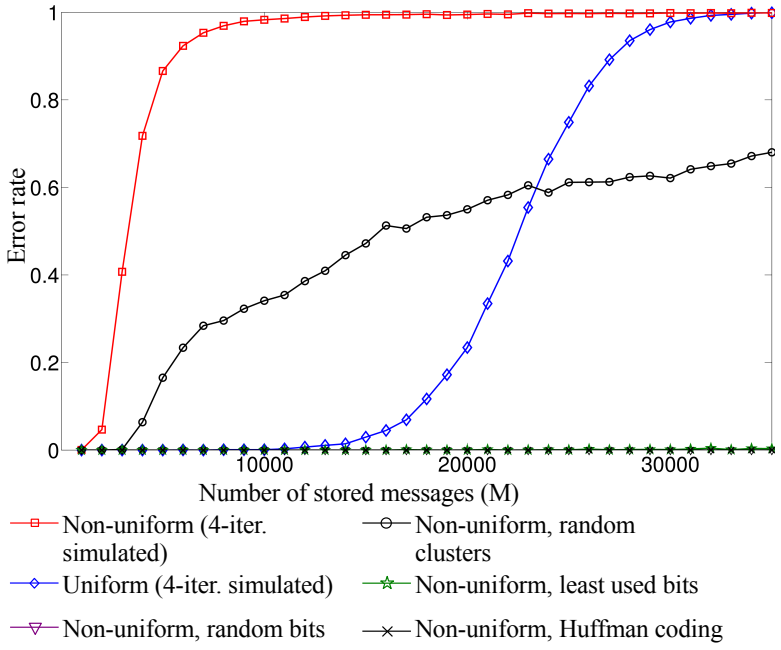


FIGURE 3.5: Evolution of the error rate with regard to the number of stored messages for different strategies. Four randomly erased positions. Network composed of eight clusters of size 256 for uniform and non-uniform curves. Seven random clusters of 5000 fanals added for random clusters strategy. Network composed of eight clusters of size 4096 for random bits, least used bits and Huffman coding. The material, in terms of maximal number of connections, used for each strategy is comparable

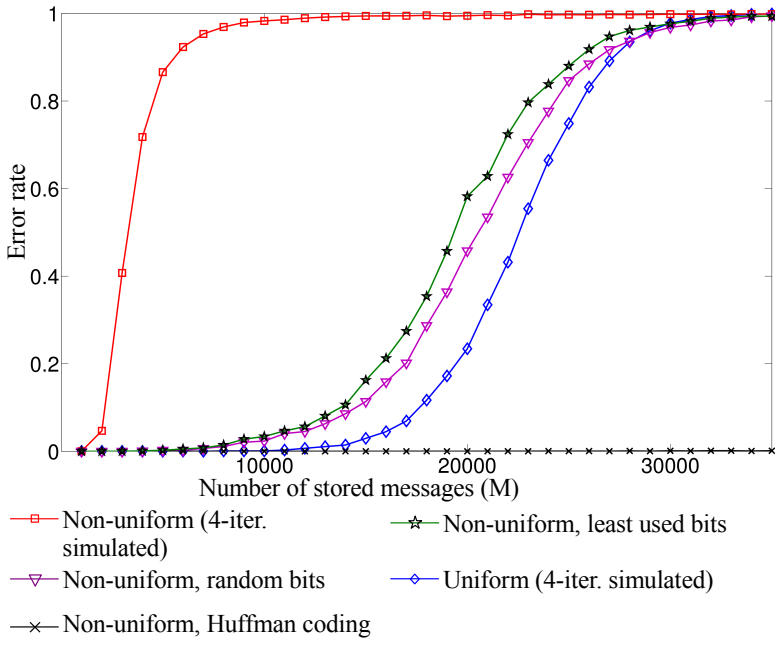


FIGURE 3.6: Evolution of the error rate with regard to the number of stored messages for different strategies with minimal material used to approach the performance close to uniform case. Four randomly erased positions. Network composed of eight clusters of size 256 for uniform and non-uniform curves. Eight clusters of size 1024 used for all the strategies.

TABLE 3.1: The limiting number of messages that can be stored when using each strategy until the error rate reaches 0.1. Two additional bits used.

Technique	No. messages
Least used bits	14000
Random bits	15000
Huffman coding	117000
Uniform messages	220000

Since the random clusters technique show the smallest improvement in the retrieval, Figure 3.6 presents the other strategies with a minimal material to approach the performance close to the uniform case. When the random bits and the least used combination strategies are applied, two additional bits are enough to get close to the uniform data case. This means that the size ℓ of the clusters equals 1024 and the material used accounts for 4.9% of the material used for random clusters strategy. When three bits are added (not shown on the plot) the performance gets already much better than for the uniform messages, therefore two bits are the minimal necessary material needed. The curve for Huffman coding technique also uses two additional bits (two bits turned out to be the minimal material for the random bits, least used bits and Huffman coding techniques). In this case the gain in performance is significant, the error rate always stays close to zero. For the used data distribution the length of the messages sometimes exceeds 72 bits (8 clusters $\times$ (8 bits + 1 random bit)) and consequently, adding less than two bits is not possible. However, for different data sets minimizing the used material could still be feasible.

Figure 3.6 does not show the limiting number of messages for Huffman coding technique. Therefore, additional simulations for larger numbers of stored messages were carried out. Table 3.1 shows a comparison of the strategies when two additional bits were used (corresponds to Figure 3.6) and the error rate reaches 0.1. It also compares the results to the case when uniform messages are stored in the network of the same size ($c = 8$, $\ell = 1024$).

The analyses show clearly the importance of adapting the network to real-world data. We propose and evaluate several strategies to avoid performance degradation. One of them relies on supporting the cliques with additional signals coming from the added clusters. The principle of the other techniques is to spread the same values across a group of

fanals. In order to apply these recently introduced memories in practical applications, adapting the network to non-uniform messages is indispensable, as real-world data is not necessarily uniformly distributed.

Compared to the technique inspired by the similar network model (Knoblauch et al., 2010), other proposed strategies offer performance improvements. Especially, the strategy that uses Huffman coding offers great performance enhancements, since it minimizes the length of each message segment based on its distribution. However, Huffman coding requires constructing the dictionaries in order to code and decode messages.

The application determines which technique should be used. If one can afford transforming data before storing it in the network either random bits or compression codes should be used. Otherwise, one should use random clusters technique where only initial data is used when stimulating the network. Alternatively, when the used material is constraining, random bits and compression codes are more efficient. Nevertheless, Huffman coding implies additional cost of coding/decoding which means that in some cases random bits offer better performance/cost trade off.

3.4 Twin neurons for efficient real-world data distribution in networks of neural cliques

It is interesting to explore some self-adapting techniques built in the network architecture to overcome the limitations of Huffman coding. For instance, using several fanals to store the same symbol in a cluster may decrease the local density in the network, without the need to pre-analyze the whole set of messages. This is the core idea of the technique proposed in this section.

3.4.1 Introducing twin neurons

In order to avoid the additional cost related to coding and decoding and provide an online and flexible approach where the whole set of messages does not have to be known, a new strategy to store non-uniform messages is developed here.

The method relying on Huffman coding is not very satisfactory in terms of biological plausibility. Storing messages according to the procedure described in this subsection,

exploits the fact that neurons that are close to each other, receive inputs that are relatively close and become clones of each other called twins here. (Thomson and Bannister, 2003) confirms that in the brain there exist neurons that are reciprocally connected and consequently, driven by the same stimulus. Increasing the spatial diversity of the stored information, allows to reduce the density of the most stressed parts of the network.

The principle of the method is to introduce twin neurons, that is changing the role of neurons in response to data distribution. If a fanal is overloaded (has high local density) another fanal will be designated to represent the same piece of information. The pseudocode in Figure 3.7 outlines the new algorithm which relies on modifying the message storing procedure. Furthermore, an association table ($AssocTab$) connecting each value on each segment with a specific fanal is necessary. Initially this table is empty. Alternatively, instead of the association table, fanals representing the same value can be interconnected. When a fanal is initially stimulated it sends signals to all the other fanals in its cluster associated to the same value. Then, all the activated fanals start exchanging signals on the global (inter-cluster) level. Depending on the hardware/software implementation cost trade-off or biological plausibility aspects any of the solutions can be chosen. In this work, the association table is used.

Before storing messages, the fanal's connection limit ($ConnectionsLimit$) is chosen. After storing each message, the total number of outgoing connections of each fanal belonging to the newly created clique is controlled (line 19). If this number exceeds a formerly defined threshold, and there are unassociated fanals available in this cluster (line 21), a new fanal is associated to this value (line 22). Similarly to, for example, adding random bits that spread a given value over a number of fanals, the result is that the local density is limited. In case there is no additional fanal left, the association table remains unchanged, and for each value the last associated fanal is used. In order to retrieve a message based on partial input, all the fanals associated to the values on the known segments are stimulated.

As the result of this method, the network automatically adapts to the distribution of the stored data efficiently using the available material. Limiting local densities improves retrieval performance.

```
Input:
ℓ
ConnectionsLimit
Messages        \\ matrix; one message in each row, one segment in each column
SegVal          \\ value on the given segment
SegNum          \\ number of the given segment
AssocTab = [ ]  \\ empty cell matrix

Output:
AssocTab        \\ cell matrix; number of row indicates the value in the message,
                   number of column indicates the number of segment. Each cell
                   in the matrix holds the numbers of fanals associated to the
                   given value on the given segment

procedure TwinFanals(ConnectionsLimit)
  for each row in Messages do
    Store
    for each fanal used do
      if fan−out > ConnectionsLimit then      \\ fan − out - number of fanal's
                                                 outgoing connections
        if max(AssocTable(:, SegNum)) < ℓ then
          AssocTab(SegVal, SegNum)(end+1) = max(AssocTab(:, SegNum))+1
        end if
      end if
    end for
  end for
  return AssocTab
end procedure
```

FIGURE 3.7: Message storing procedure for twin fanals; pseudocode.

3.4.2 Theoretical analysis

For the presented curves the improved retrieval procedure is used where only the signals from distinct clusters are included in the sum calculated by each fanal.

Based on the probability of correctly retrieving a message (2.7) one can obtain the probability of error after one iteration of the retrieval procedure as:

$$P_e = 1 - P_c. \tag{3.1}$$

The error probability in case of non-uniform data is analyzed with the following non-uniform distribution. On the first segment the value one is chosen with probability p, any other value is chosen with the same probability that equals $\frac{1-p}{\ell-1}$. The rest of the

$c-1$ segments are drawn uniformly. To simplify the discussion, this distribution is called *FSA* (*First Segment Anomaly*).

As a consequence, the density between any two clusters associated to the last $c-1$ segments is unchanged. Between the first cluster and any other one, the density is:

$$d' = 1 - (1 - \frac{1-p}{\ell(\ell-1)})^M \ , \tag{3.2}$$

for any value on the first segment that is not one, and

$$d'_1 = 1 - (1 - \frac{p}{\ell})^M \ , \tag{3.3}$$

for the first segment equal one.

The error probability changes depending on whether the first segment equals one (to simplify the discussion we call this case π) or is different from one ($\overline{\pi}$) and whether the first segment is erased (ϵ) or not erased ($\overline{\epsilon}$). In the following part of this subsection, all the theoretical error probabilities are expressed with appropriate equations.

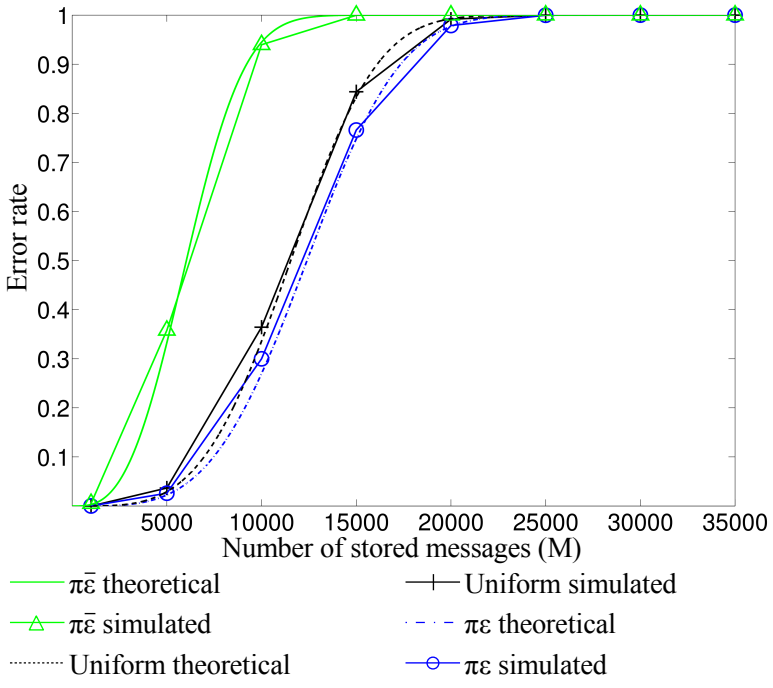


FIGURE 3.8: Error rate for *FSA* and uniform distributions, with network from (Gripon and Berrou, 2011a) used. First segment equal one (π). Four positions are randomly erased, results after one iteration. For each point 100 networks are evaluated doing 100 tests per network.

In case $\pi\epsilon$, the error probability after one iteration of the retrieval procedure is:

$$P_{e,\pi\epsilon} = 1 - \left(1 - d'^{c-c_e}\right)^{\ell-1} \cdot \left(1 - d^{c-c_e}\right)^{(c_e-1)(\ell-1)} . \tag{3.4}$$

In case $\pi\bar{\epsilon}$:

$$P_{e,\pi\bar{\epsilon}} = 1 - \left(1 - d'_1 \cdot d^{c-c_e-1}\right)^{c_e(\ell-1)} . \tag{3.5}$$

In case $\overline{\pi}\epsilon$:

$$P_{e,\overline{\pi}\epsilon} = \left(1 - d'^{c-c_e}\right)^{\ell-2} \cdot \left(1 - d^{c-c_e}\right)^{(c_e-1)(\ell-1)} \cdot \left(1 - d'^{c-c_e}\right) . \tag{3.6}$$

In case $\overline{\pi\epsilon}$:

$$P_{e,\overline{\pi\epsilon}} = 1 - \left(1 - d' \cdot d^{c-c_e-1}\right)^{c_e(\ell-1)} . \tag{3.7}$$

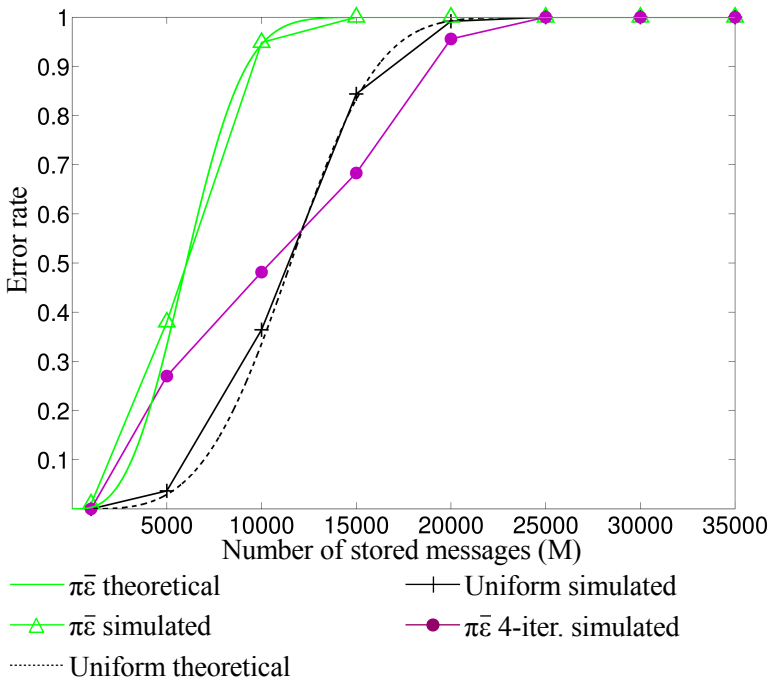


FIGURE 3.9: Error rate for *FSA* and uniform distributions, with compression codes or twin fanals used for *FSA*. First segment equal one (π). Four positions are randomly erased, results after one iteration unless otherwise stated. For each point 100 networks are evaluated doing 100 tests per network.

When using one of the techniques described in the paper (compression codes or twin fanals), the fanal corresponding to value one in the first cluster is duplicated when the density of its connections is large enough. Ideally, the density becomes constant and

thus equals $d' < d$. The number of fanals n_1 associated to value one in the first cluster is:

$$n_1 = \frac{log\left(1 - \frac{p}{\ell}\right)}{log\left(1 - \frac{1-p}{\ell(\ell-1)}\right)}. \tag{3.8}$$

As a result, in case π, $P_{e,\pi\epsilon}$ is unchanged and $P_{e,\pi\bar{\epsilon}}$ becomes:

$$P'_{e,\pi\bar{\epsilon}} = 1 - \left(1 - (1 - (1 - d')^{n_1}) \cdot d^{c-c_e-1}\right)^{c_e(\ell-1)}. \tag{3.9}$$

In case $\overline{\pi}$, $P_{e,\overline{\pi}\epsilon}$ is unchanged and $P_{e,\overline{\pi}\epsilon}$ becomes:

$$P'_{e,\overline{\pi}\epsilon} = 1 - \left(1 - d^{c-c_e}\right)^{(c_e-1)(\ell-1)} \cdot \left(1 - d'^{c-c_e}\right)^{n_1+\ell-2}. \tag{3.10}$$

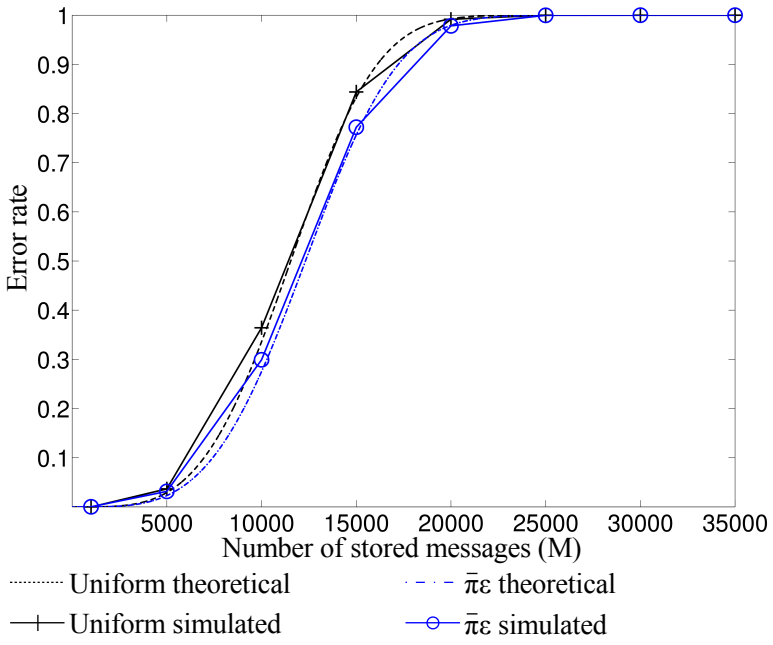


FIGURE 3.10: Error rate for *FSA* and uniform distributions, with network from (Gripon and Berrou, 2011a) used. First segment different from one ($\overline{\pi}$). Four positions are randomly erased, results after one iteration. For each point 100 networks are evaluated doing 100 tests per network.

In the following part of this subsection the theoretical equations are verified with simulations. The parameters chosen for the analysis are $c = 8$, $c_e = 4$, $\ell = 256$, $p = 0.5$. For the given parameters and distribution, $n_1 = 255$. Figure 3.8 illustrates the cases $\pi\epsilon$ and $\pi\bar{\epsilon}$. Both the equations (3.4), (3.5) and the simulation results are plotted. One can see that the simulations correlate well with the theoretical analysis. Additionally,

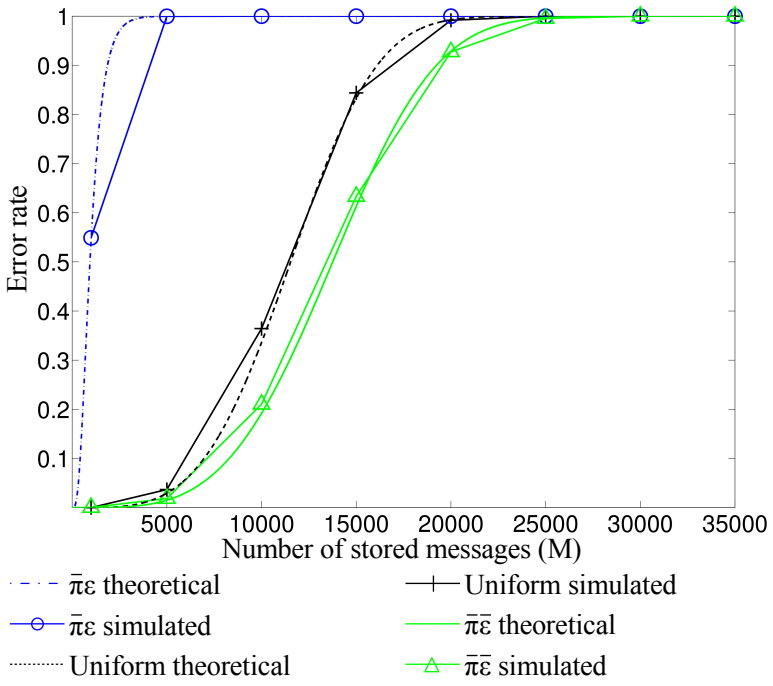


FIGURE 3.11: Error rate for *FSA* and uniform distributions, with compression codes or twin fanals used for *FSA*. First segment different from one ($\overline{\pi}$). Four positions are randomly erased, results after one iteration. For each point 100 networks are evaluated doing 100 tests per network.

the theoretical and simulated curves for uniformly distributed messages are given for comparison.

Figure 3.9 shows the curve for equation (3.9) for compression codes or twin fanals. As expected in this case, the performance after one iteration is the same whether compression codes or twin fanals are used or not. The additional curve for four iterations shows the performance improvement.

Figure 3.10 illustrates the cases $\overline{\pi}\epsilon$ and $\overline{\pi\epsilon}$. Equations (3.6), (3.7) correlate well with the simulation results. The equation (3.10) and the corresponding simulations are shown in Figure 3.11. One can notice significant performance improvement.

3.4.3 Performance comparison

Before comparing twin fanals to other techniques we propose another strategy based on Huffman coding that is further adapted to practical applications. Then, we proceed to performance comparison.

3.4.3.1 Comments on Huffman coding technique

The intrinsic characteristic of Huffman coding is the variable codeword's length. This parameter depends on the size and distribution of the dataset. It is possible that some of the codewords exceed the available cluster size ℓ. In this case, no random bits are added to this segment and the remaining bits are pushed to the next segment. Moreover, after adding random bits, the bits from all the segments are shuffled in such a way that even and odd bits from each of the segments are stored in separate clusters. This means that in each cluster the chosen fanal depends not only on the value on the same segment in the initial message but also on the other segments. This implies that erasures are allowed only on the messages after the coding. As pointed out in (Boguslawski et al., 2014), this characteristic makes this technique an effective solution for applications in which one can afford transforming data before storing it in the network. To simplify the discussion, in the present section this technique is called *Huffman shuffle*.

As a consequence of the abovementioned property, each time the known segments of a message are used to stimulate the network, only one fanal in each cluster is selected. In case of the strategy that relies on twin fanals, erasures are possible on the initial messages. This is important in terms of practical applications. Nevertheless, when retrieving messages, one does not know which fanal was used to create the clique representing the concerned message and all the fanals associated to the known segments are stimulated. If a unique fanal is stimulated in each cluster, the problem solved by the network is much easier and one can expect better performances.

To compare the techniques two cases can be considered: 1) messages are transformed before storing them in the network, 2) messages are not transformed before storing them in the network. In case of 1) the messages are modified such that they contain the number of fanal that was associated to a specific value in each message. Consequently, in all the strategies only a unique fanal is stimulated in each cluster. In case of 2) the association table is used to find the fanals to stimulate when a message is presented to the network. In this scenario, all the fanals associated to a value present in the message are stimulated. The first case, where erasures are allowed only on the transformed messages, can be applied to a very limited set of applications. Since the focus here is on applications and real-world data, the techniques are compared in the second scenario. If *Huffman shuffle* is applied in the second scenario, after all the described processing steps

TABLE 3.2: Comparison of Huffman shuffle and Huffman simple techniques

	Huffman shuffle	Huffman simple
Data transformed before storage	yes	no
Erasures allowed on initial messages (before coding)	no	yes
Multiple fanals initially stimulated	no	yes
Versatility	-	+

(adding random bits, pushing bits to the next segment, shuffling bits) one fanal can be associated to many different values. To eliminate the impact of the interdependencies between different segments and to adapt *Huffman shuffle* to the case 2), a new strategy is added to the comparison. The bits that exceed the available space are now simply cut instead of being pushed to the next segment. Furthermore, there are no bit-shuffle operations. This technique is called *Huffman simple* here. To provide a fair comparison, the non-uniform distribution is chosen so that the amount of codewords that exceed the space available in the cluster ($\ell = 1024$, 10 bits, the same size as before for Huffman coding) is negligible ($< 1\%$). This occurs for the truncated Gaussian distribution with $\mu = 135$, $\sigma = 65$. The characteristics of both techniques relying on Huffman coding are summarized in Table 3.2.

3.4.3.2 Comparison

Figure 3.12 shows the results when multiple fanals can be stimulated in each cluster. First note the performance of the network (Gripon and Berrou, 2011a) in case of the non-uniform distribution ($\ell = 256$, the same as in Figure 3.2). Then, several techniques are applied to the same network as in case of the former techniques ($\ell = 1024$). Random bits allow storing more messages, however they are outperformed by the other proposed strategies. Twin fanals present the best performance, *Huffman simple* being slightly less effective. As stated earlier Huffman shuffle is not adapted to stimulating multiple fanals. In fact, it reaches retrieval error rate equal to 0.97 when only 1000 messages are stored.

Figure 3.13 shows the connections limit values for each point of the curve for twin fanals. Each of this values was obtained by varying the connections limit and choosing the one that gave the lowest error rate. The connections limit parameter ($ConnectionsLimit$) can be connected to density d for a uniform distribution (2.6) and expressed as saturation

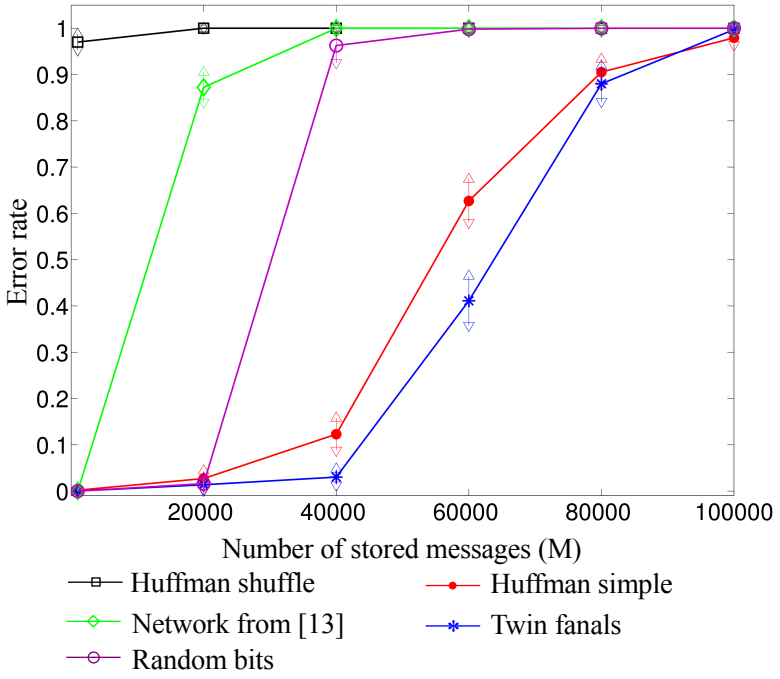


FIGURE 3.12: Performance comparison for different proposed strategies when multiples fanals are stimulated. Four positions are randomly erased. For each point 100 networks are evaluated doing 100 tests per network. Multiple fanals are stimulated in each cluster. The arrows indicate standard deviations of the error rates.

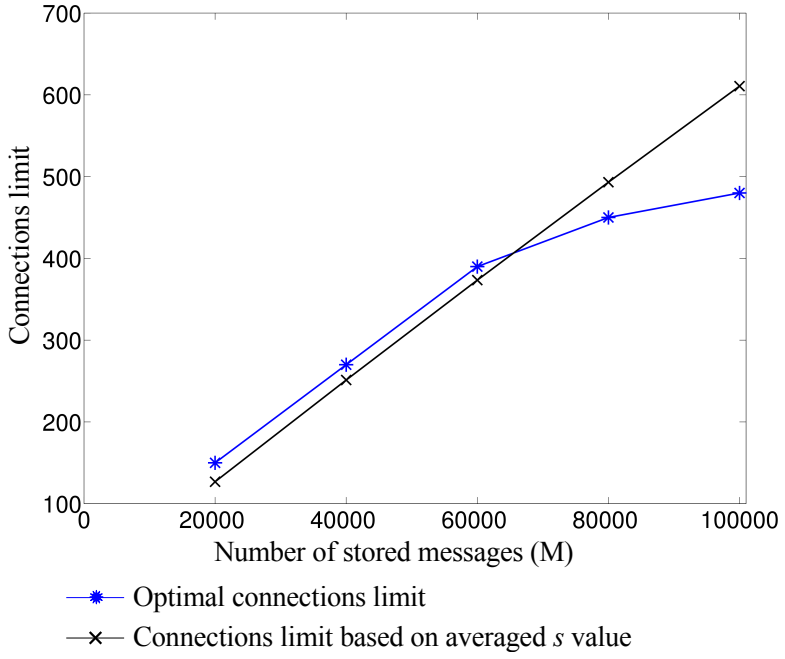


FIGURE 3.13: Optimal and predicted connections limit values with regard to the number of stored messages. For low numbers of stored messages any connections limit can be chosen.

s:

$$s = \frac{ConnectionsLimit}{(c-1)\ell d}. \tag{3.11}$$

$(c-1)\ell$ gives the maximal number of connections per fanal. By weighting this value with d one obtains the expected number of fanal's connections. Dividing $ConnectionsLimit$ by this number gives the saturation of the network.

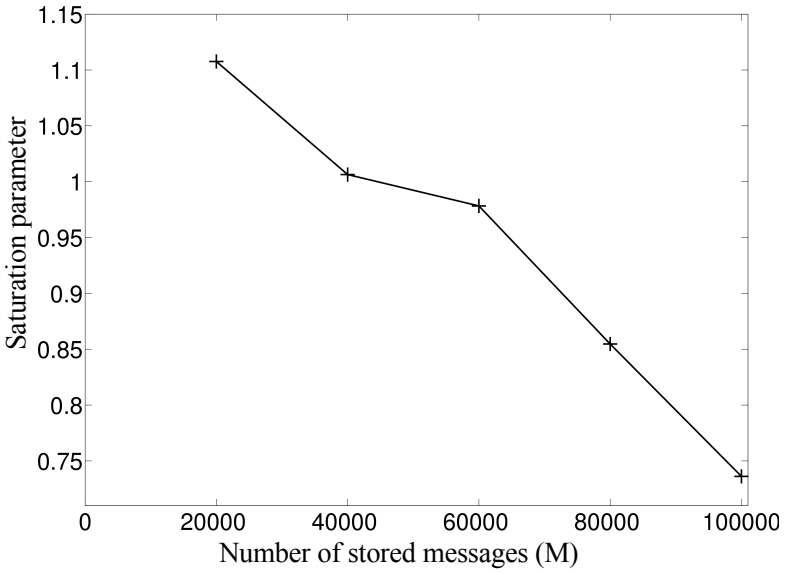


FIGURE 3.14: The saturation parameter used to predict connections limit value with regard to the number of stored messages. For low numbers of stored messages any connections limit can be chosen.

Based on the density which is a function of the number of stored messages M the approximate optimal value of the connections limit can be predicted. Figure 3.14 shows how the saturation depends on the number of stored messages based on the optimal connections limit values. Figure 3.13 shows also the connections limit values obtained from an averaged s value as $\overline{s}(c-1)\ell d$. The proximity of the optimal and predicted connections limit values in the range where the network is not overloaded, shows that the technique based on twin fanals is also easy to adjust to the stored data.

According to the obtained results, the strategy that relies on twin fanals presents the best performance. Moreover, it allows to avoid the coding/decoding overhead which is important in practical applications, especially in the ones presented in the following chapter, where the system's time response and low complexity are crucial.

3.4.4 Influence of distribution's standard deviation

As shown in the preceding sections, the distribution of the data stored in the network influences the retrieval performance. Here the standard deviation of the non-uniform data distribution is varied to show what is the impact on the performance of the network when using twin fanals. The experiments are done for truncated Gaussian distributions with different standard deviations. Additionally to the distribution chosen in the section IV.D with $\sigma = 65$, a distribution with $\sigma = 25$ and $\sigma = 100$ are used. Figure 3.15 shows how the retrieval process is impacted when the stored data is more non-uniform ($\sigma = 25$) and more uniform ($\sigma = 100$) than the previously chosen one. When comparing these curves to the results in Figure 3.12, one can see how the retrieval performance depends on the considered distribution's parameter.

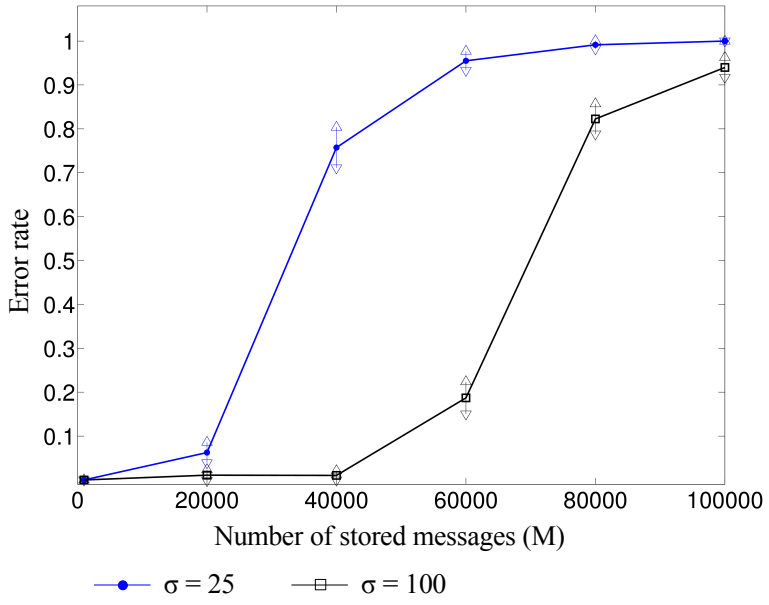


FIGURE 3.15: Error rate when using twin fanals for different values of σ of non-uniform distribution. Four positions are randomly erased. For each point 100 networks are evaluated doing 100 tests per network. The arrows indicate standard deviations of the error rates.

3.5 Real-world data in two practical applications

3.5.1 MPSoC power management for LTE receiver

3.5.1.1 LTE receiver implemented on MAGALI platform

In order to obtain data to store in the network, an applicative test case on MAGALI telecom chip is considered (Clermidy et al., 2009). MAGALI is a semi-heterogeneous MPSoC, i.e. there is a number of different PEs but each is reproduced several times. The available resources communicate through a NoC. Each PE can work with one of 256 frequencies in range between 20 and 790MHz (Clermidy et al., 2010). The application mapped on the platform is the LTE receiver. It consists of six tasks: TRX OFDM, Channel Estimation, Coefficient Interpolation, MIMO Decoding, RX Bit and Channel Decoding, Figure 3.16. The parameters of the application, necessary to calculate the energy (1.10), were characterized at $f_{nom,i} = 400$MHz and are summarized in Table 3.3. The five operating modes presented in the table differ in reachable data rates (Clermidy et al., 2009).

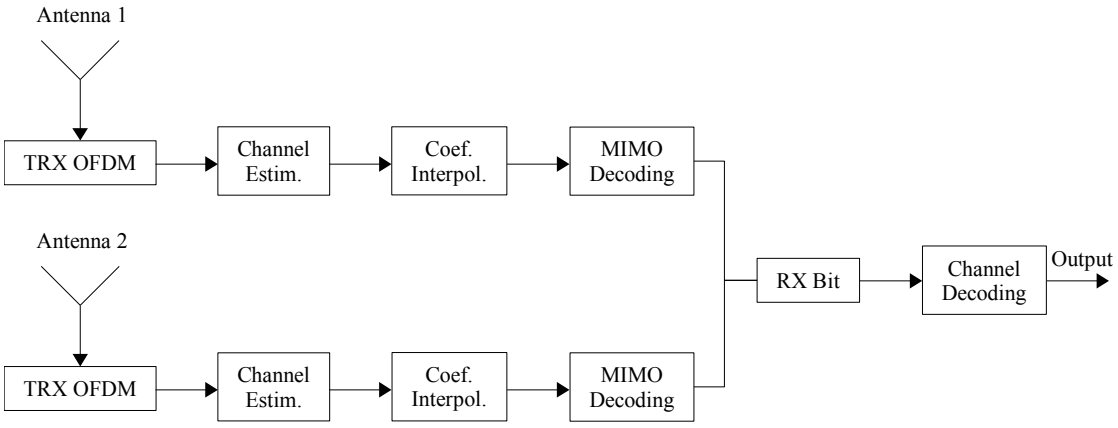


FIGURE 3.16: LTE receiver application graph. The memory buffers between the processing blocks are not presented.

3.5.1.2 Network of neural cliques used as power management unit

Since there are 256 frequencies available, each PE is associated a cluster of 256 fanals. The frequencies that are applied to the PEs depend on the latency constraint L and the operating mode. There is therefore a cluster of 5 fanals for modes and a cluster for

TABLE 3.3: LTE receiver application parameters (Mansouri et al., 2010a). Characterized at $f_{nom,i} = 400$MHz.

	TRX OFDM	Channel Estim.	Coef. Interpol.	MIMO Decoding	RX Bit	Channel Decoding
N_i [clk cycles]						
Mode 1	39200	91296	1668	12210	6445	$16{\cdot}10^3$
Mode 2			3336	24420	12890	$32{\cdot}10^3$
Mode 3			3336	24420	7855	$32{\cdot}10^3$
Mode 4			16680	122100	39275	$160{\cdot}10^3$
Mode 5			16680	122100	34665	$264{\cdot}10^3$
$E_{nom,i}$ [pJ/clk cycle]	47.67	180.55	198.94	198	93.84	247.74

TABLE 3.4: MPSoC power management results

Data type	Neural cliques network type	Error rate	No. L constraint violations
Uniform random	Standard	0	-
Real	Standard	0.71	87
Real	Twin fanals, conn. lim. = 6	0.07	8
Real	Twin fanals, conn. lim. = 10	0.03	3

L. L is ranging from 0.7 to 2ms. 51 latency values are considered in this application and consequently one cluster of 51 fanals is created. The impact of the L cluster's size on the power management efficiency is analyzed in subsection 4.4.3.3. Each time the system needs to be reconfigured, the L cluster and mode cluster are stimulated and the set of frequencies is retrieved from the network. This is illustrated in Figure 3.17.

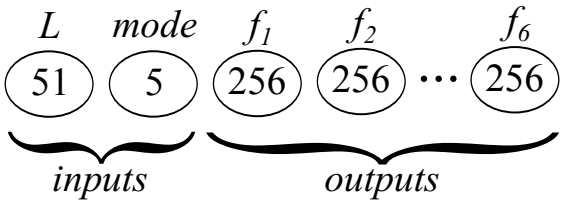


FIGURE 3.17: Network of neural cliques structure for LTE receiver power management. Latency L and operating mode are stimulated and the set of frequencies is retrieved from the network.

3.5.1.3 Simulation results

The results of the evaluated experiments are summarized in Table 3.4. Firstly, to test the network, a random uniformly distributed set of frequencies is stored. The network is simulated for each of the possible L values. The simulations show that in this case each of the stored frequencies is retrieved correctly.

Then, the actual values of the frequencies are stored in the network and the same test procedure is applied. This time the error rate (the amount of messages in which at least one frequency was not retrieved correctly) clearly increases. As a consequence of the incorrectly retrieved frequencies, there are 87 out of 255 L constraints that are violated. Note that there may be more incorrectly retrieved frequencies that not necessarily need to lead to L constraint violation. Nevertheless, they result in non-optimal solutions which imply an excessive energy consumption.

In order to approach the optimal retrieval process of random uniform frequencies, twin fanals introduced in section 3.4 are applied. The structure of the network remains the same, the number of fanals does not change, only the message storing procedure is modified. The connection limit is set to 6. In this case, the error rate is reduced and there are eight constraint violations. The connection limit can be varied to find an optimal value. The minimal number of constraint violations was obtained for the connection limit equal to ten. The error rate and the number of constraint violations are slightly reduced.

3.5.2 Dynamic management of PVT variations

3.5.2.1 Introduction

As we already know, a solution to avoid worst-case design is to partition the system on a number of parts allowing each to work at its best operating point determined by applicative constraints. As the technology scales down and the supply voltage is reduced, the intra-die variability becomes an increasing challenge. The fabrication process becomes less reliable and causes static process (P_r) variations. Dynamic parameters as voltage (V) and temperature (T_{emp}) variations also have an increasing impact on the timing and power consumption. Consequently, the optimal operating point set by a power decision

unit, can be shifted by dynamic voltage drops and temperature variations. We can, however, integrate some sensors in our system that provide the information regarding current working conditions (V, T_{emp}). Based on that information, each part adjusts again its operating point to reduce the power consumption or preserve the functionality.

3.5.2.2 Multiprobe sensor for PVT variations

In order to provide the adjustment policy with current (V, T_{emp}) approximations, specific sensors have to be embedded in the system. Here, a low-cost digital sensor Multiprobe (Vincent et al., 2011) is considered. Multiprobe is made of seven Ring Oscillators (RO) that are designed to have different sensitivities to process, voltage and temperature (PVT) variations. These sensors, embedded in different parts of the circuit, experience the same static and dynamic variations as the parts they are placed close to. Therefore, after a calibration phase, based on the frequencies of the ROs, current approximations ($\hat{V}$, $\hat{T}_{emp}$) of (V, T_{emp}) can be found. Frequency of a RO depends in a complex way on the PVT parameters and it is impossible to obtain ($\hat{V}$, $\hat{T}_{emp}$) from a single frequency (Vincent et al., 2014). Since there are no models available, solving a set of equations in case of several ROs is mathematically infeasible.

An estimation method to obtain ($\hat{V}$, $\hat{T}_{emp}$) from the frequencies of Multiprobe is proposed in (Vincent et al., 2014). The proposed method relies on statistical tests of goodness-of-fit. First, the data from the sensors is specially adapted to the further treatment. Then, the statistical tests are evaluated, a brute-force search in the database obtained during the calibration phase is performed and a weighted mean of the closest found entries is calculated. This method provides satisfactory estimation precision yet is quite complex. Moreover, the estimation computed by a dedicated hardware block can be obtained in 25μs which is not fast enough to follow fast voltage variations (order of magnitude of few μs).

3.5.2.3 Network of neural cliques used as dynamic management unit

Networks of neural cliques can be used to provide ($\hat{V}$, $\hat{T}_{emp}$) pairs corresponding to frequencies of the ROs, Figure 3.18. The main focus here, is on the impact of the data

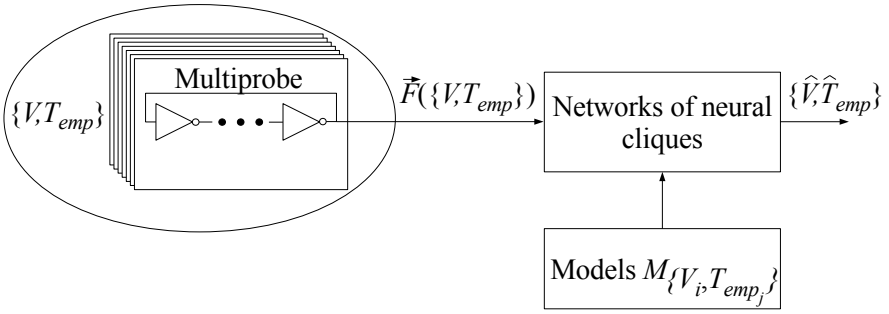


FIGURE 3.18: Overview of the variability management system.

distribution. However, as it is shown later, time response of hardware implementation of the networks is in the order of tens of ns which allows following fast voltage variations.

In order to obtain data to store in the network, electrical simulations of the Multiprobe sensor are performed with Eldo circuit simulator. The message set was obtained for V ranging from 0.7V to 1.3V with step of $\Delta V = 10$mV and T_{emp} ranging from -40°C to 120°C with step of $\Delta T = 10$°C. In this work only two ROs of the Multiprobe are used to retrieve ($\hat{V}$, $\hat{T}_{emp}$) couples. The simulations show that the impact of reducing the number of used frequencies is negligible on the accuracy of the retrieved ($\hat{V}$, $\hat{T}_{emp}$). Among the two kept ROs one is specifically designed to be more sensitive to temperature variations than the other. Due to the characteristics of the ROs, in the database containing 1037 models linking the frequencies with ($\hat{V}$, $\hat{T}_{emp}$) points, a number of models share the same frequencies leading to different ($\hat{V}$, $\hat{T}_{emp}$) couples. In such an ambiguous case both ($\hat{V}$, $\hat{T}_{emp}$) solutions are equitable and are output by the network. In a target system the network can be followed by an additional processing step which calculates a mean value of the obtained ($\hat{V}$, $\hat{T}_{emp}$) responses. Since the focus of this chapter is on exploring the impact of data distribution, these models are removed from the database. After this step there are 1018 models that are stored in the network.

3.5.2.4 Network of neural cliques dimensions

The ROs can output 256 and 512 (temperature-sensitive RO) different frequency values. Therefore, they are associated two clusters of 256 and 512 fanals respectively. In addition, there is a cluster to store T_{emp} values made of 17 fanals and a cluster of 61 fanals to store V values. Each time the ($\hat{V}$, $\hat{T}_{emp}$) are demanded by the variability management

TABLE 3.5: Variability management results

Data type	Neural cliques network type	Error rate	MAE_V	$MAE_{T_{emp}}$	σ_V	$\sigma_{T_{emp}}$
Uniform random	Standard	0.2	10.7	10.64	54.9	29.12
Real	Standard	0.54	4.3	24.1	9	37.62
Real	Twin fanals, conn. lim. = 3	0.49	3.6	22.95	7.9	37.21
Real	Twin fanals, conn. lim. = 3, increased size	0.24	1	2.03	3.2	5.44

system, the frequency clusters are stimulated and ($\hat{V}$, $\hat{T}_{emp}$) couple is obtained from the network.

3.5.2.5 Simulation results

The results are assessed in terms of the error rate (the amount of messages in which at least one of the ($\hat{V}$, $\hat{T}_{emp}$) parameters is not retrieved correctly) and mean absolute errors:

$$MAE_V = \frac{1}{M}\sum_{i=1}^{M}|\hat{V}_i - V_i|, \tag{3.12}$$

$$MAE_{T_{emp}} = \frac{1}{M}\sum_{i=1}^{M}|\hat{T}_{emp_i} - T_{emp_i}| \tag{3.13}$$

where M is the number of stored models. Additionally, the standard deviations associated to these errors are reported.

All the obtained results are summarized in Table 3.5. First, a random uniformly distributed set of frequencies is stored. The network is simulated for all the stored models. When the actual values of the frequencies are stored in the network and the same test is applied, the error rate increases significantly. The mean absolute error and standard deviation increased in case of temperature. In case of voltage the error and standard deviation decreased which together with the error rate means that there are more errors but they are smaller.

Then, as in the previous section, twin fanals are applied. The network's dimensions stay the same. As a consequence, the error rate is slightly reduced and the error and standard deviation of V are lower than in the former experiments. The temperature retrieval is

slightly improved as well compared to real data stored in non-adapted network. In order to improve the retrieval significantly, the network's dimensions are changed. When T cluster is increased to 500 fanals, the results are further improved. For both networks in which twin neurons were used, the connection limit was set to 3. This value proved to be the optimal one.

3.6 Conclusion

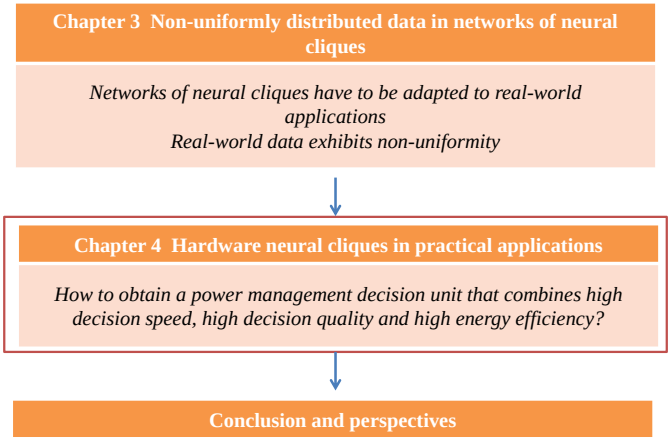
Networks of neural cliques are associative memories able to store and successfully retrieve a large number of messages. In this chapter, their performances are theoretically studied exploiting non-uniformly distributed information. When facing real-world applications, the information that is stored in the networks is not necessarily uniformly distributed. The solution based on Huffman compression coding offers great performance enhancements and allows efficient storage of non-uniform messages at the cost of high computational complexity. In some applications this can be a limiting factor.

Therefore, we propose a method with a limited complexity. Introducing twin fanals avoids complex coding and decoding phases and is biologically plausible. It allows erasures on initial messages without transforming data before the storage. Furthermore, there is no constraint in terms of data distribution parameters as it is the case with Huffman coding. The presented results show that thanks to spreading frequent values among a group of fanals, twin fanals offer better performances than Huffman coding-based technique.

To asses the proposed improvements, in the rest of the chapter, twin neurons are used with real-world data. It is shown that adapting the model (Gripon and Berrou, 2011a) is indispensable for using these networks in practical applications. Two test-cases in power management domain are exploited to obtain real-world data. When the networks presented in (Gripon and Berrou, 2011a) are applied to these applications, their performance is largely degraded leading to non-functional system. In order to approach the theoretical performance of the model, twin fanals are used when real-world data is stored. The results show that this solution approaches performances close to uniformly distributed data. Therefore, adapting the basic model is effective for using these networks in practical applications.

Chapter 4

Hardware neural cliques in practical applications



4.1 Introduction

In this chapter we propose a hardware neural cliques structure to design a power management decision unit. Both general studies and practical test-cases are included. Neural cliques decision unit is compared to high-level decision unit relying on game theory and CAM-SRAM associative memory. Comparison between our proposal and game theory reveals significant gains in terms of decision speed and energy consumption of the decision unit. Moreover, the quality of the decision is also higher, meaning that an MPSoC controlled by neural cliques consumes less energy as well. The comparison between

neural cliques and CAM-SRAM associative memory shows that neural cliques are less complex and more energy efficient. Later, some points on hardware implementation are discussed. In the end of this chapter, we propose a compact interconnect approach using 3D technology for interconnect delay and energy reduction.

4.2 Analog and digital ASIC implementation

This section presents ASIC implementations proposed in Larras et al. (2013a). The rest of the sections in this chapter present our proposals in terms of using these circuits as a power management decision unit. These include different dimensioning optimizations and hardware improvements.

4.2.1 Analog circuit

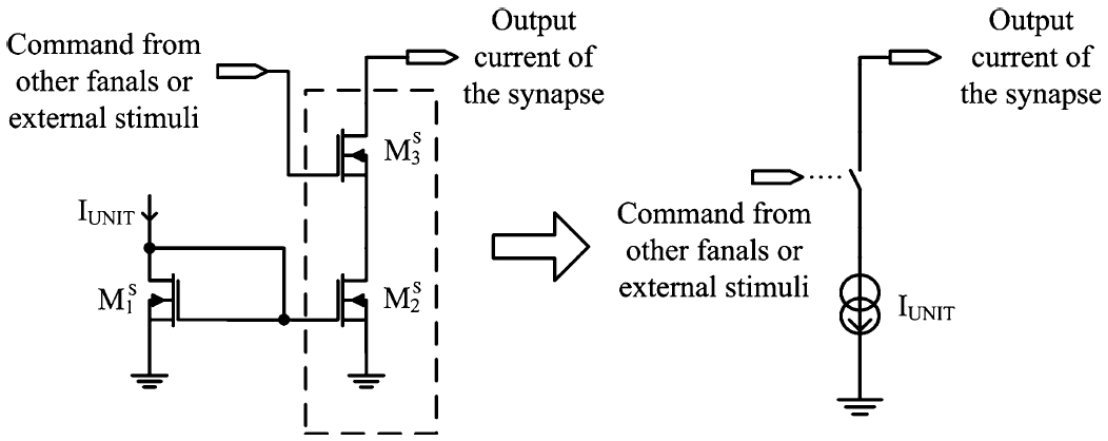


FIGURE 4.1: Schematic of the synapse circuit Larras et al. (2014).

The first analog circuit for networks of neural cliques is proposed in Larras et al. (2013a). The network circuit is built of two types of blocks: the fanal circuit and synapse circuit. The fanal circuit has two functions: 1) compute the sum of the received signals and 2) WTA functionality. The synapse circuit is transporting signals between fanals. The response of the fanal circuit is expressed in terms of voltage, yet the operations inside the fanal circuit are done with currents. To allow exchanging signals between fanals, the synapse circuit does a voltage-to-current conversion. The synapse circuit is shown in Figure 4.1. It consists of a current source switched by means of transistor M_3^S controlled by its gate voltage. The unitary current I_{UNIT} is mirrored with a current mirror M_1^S, M_2^S. This way the voltage coming from a fanal or an external (initial) stimulation is converted into a unitary current signal.

Figure 4.2 shows the fanal circuit in a cluster of four fanals. All the synapses of a fanal are connected to its input $A^k, k \in \{1, ..., \ell\}$. All the currents are summed at this node. That is how computing the sum of the received signals is realized. This sum is then

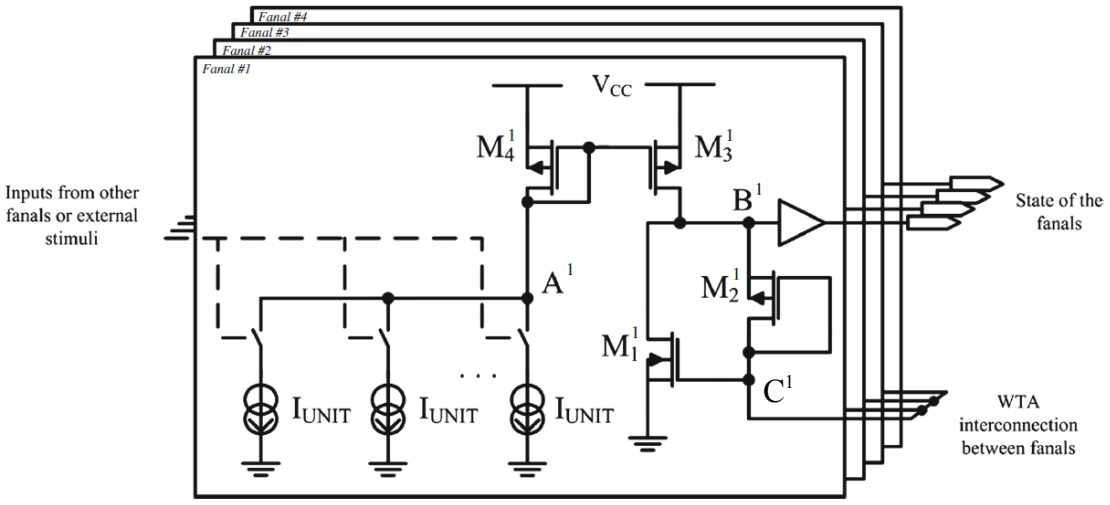


FIGURE 4.2: Schematic of a fanal in a cluster of size four Larras et al. (2014).

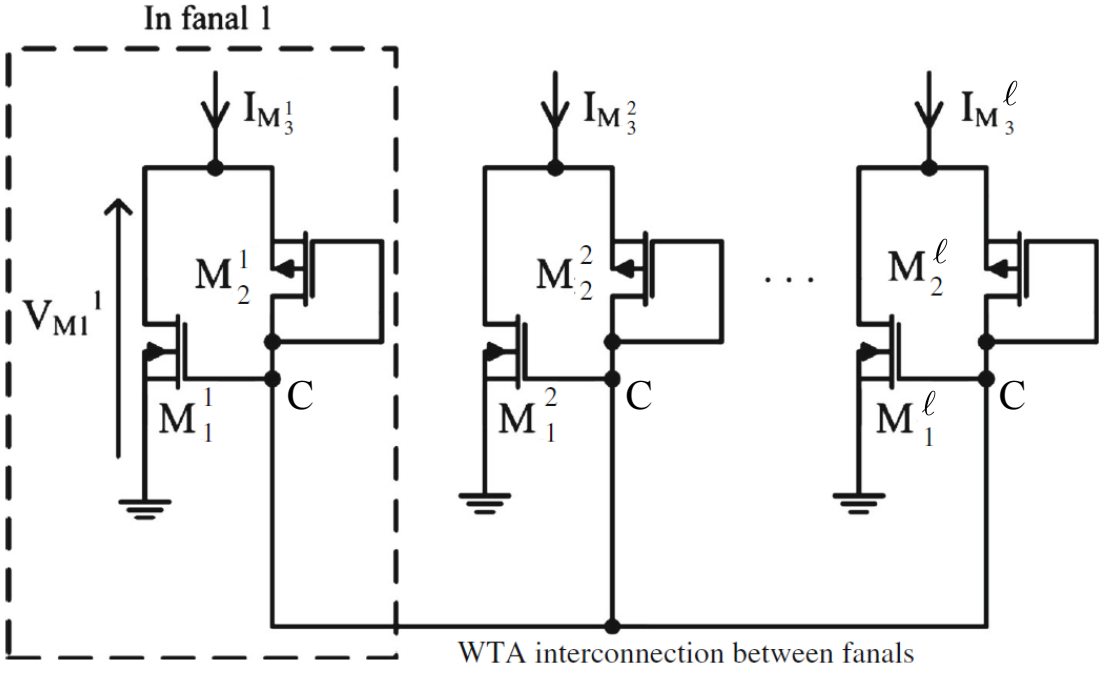


FIGURE 4.3: Schematic of the full WTA circuit Larras et al. (2014). Each two transistors are embedded in one fanal.

mirrored by a current mirror M_3^k, M_4^k. Transistors M_1^k, M_2^k incorporate a part of the WTA circuit that is formed together with all the other fanals in the cluster. The full circuit is obtained by connecting the gates of M_1^k of fanals in a cluster. This is illustrated in Figure 4.3. The gate and source potentials of all the transistors M_1^k are the same. If the current $I_{M_3^1}$ is the highest among the currents $I_{M_3^k}$, since all the transistors M_1^k operate at the same gate voltage V_C, the $V_{M_1^1}$ voltage is also the highest of all $V_{M_1^k}$ voltages. Moreover, since the V_C voltage is limited at $V_{M_1^k} - V_{T_{M_2^k}}$, the branch with the largest current defines the V_C value and sets all other M_2^k devices in the subthreshold region (since the $V_{M_1^k}$ voltage is lower than in the dominant branch).

Unlike in the theoretical model, in the analog implementation there are no distinct iterations in the retrieval process. Once stimulated, the circuit converges continuously to a solution. However, during the retrieval process it is possible to observe intermediate states that correspond to different iterations.

Electronic circuits are subject to PVT variations that cause variance in they performance. This is particularly true for analog circuits. The process variability in neural cliques analog circuits is discussed in Appendix A.

Programmable synapses are discussed in Appendix B.

4.2.2 Digital circuit

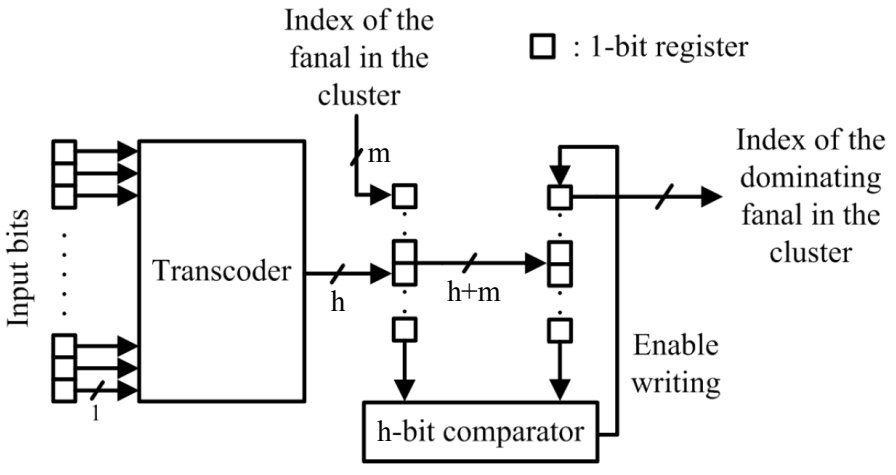


FIGURE 4.4: Schematic of the digital circuit implementing one cluster Larras et al. (2013a).

TABLE 4.1: Comparison of analog and digital implementations of networks of neural cliques Larras et al. (2013a)

	Digital	Analog	Ratio analog/digital
Surface area [μm^2]	39168	81732	$\times$2.1
Time response [ns]	208.44	37	$\div$5.63
Energy consumption per message [nJ]	16.2	0.0139	$\div$1165
Efficiency [kbit/s/μm^2]	25.48	68.77	$\times$2.7

The first dedicated digital circuit for networks of neural cliques is proposed in Larras et al. (2013a). It relies on a pipelined architecture allowing hardware reuse. Its structure is illustrated in Figure 4.4. One cluster is depicted here. At a given clock cycle, the set of input bits corresponds to the signals received through the synapses of the given fanal. Each input signal is stored in a one-bit register. A transcoder made of a chain of adders converts the inputs into an h-bit vector, where h is $\log_2$ of the number of inputs bits. The index m of the fanal is added to this vector. At each clock cycle, an h-bit comparator compares the vectors of the current fanal and the highest score so far. In the end of the process, the index of the fanal with the highest sum is output.

4.2.3 Comparison

A comparison of the analog and digital implementations is done in Larras et al. (2013a). A network of $c = 8$ clusters and $\ell = 26$ fanals in each cluster is simulated. Such a network is applied to store eight-letter words from Latin alphabet and retrieve them based on partial or noisy input. Both circuits are designed for the 1V supply ST CMOS 65nm technology. The implementation results are summarized in Table 4.1. The analog implementation is better in terms of time response and energy consumption. The overhead in the circuit's surface is acceptable when other gains are taken into account. This is expressed by the efficiency metric which combines the number of bits output by the network per second for an equivalent area.

4.3 Hardware 3D considerations

In this part, we propose a compact interconnect approach for networks of neural cliques using 3D technology (Boguslawski et al., 2015b). We present a general study to explore

the gains coming from 3D technology and we validate the proposed method on a power management test-case.

4.3.1 General introduction to 3D neural networks

Due to the number of elements that are interconnected, neural networks are wire-dominated systems. This entails challenges in hardware implementation - high latency, energy consumption and large memory requirements among others. One way to overcome these issues is to use a shared, multiplexed communication medium as bus or NoC. Nevertheless, this approach comes with performance reduction and strongly limits the application field. To obtain high-performance neural networks, physical connections are needed for all the synapses. That is why first neural networks in 3D technology are arising. 3D technology relies on stacking few dies one on another and communicating between them over Through-Silicon-VIAs (TSVs). This allows to overcome the limitations of 2D circuits and provide compact connections for all the synapses. In Belhadj et al. (2014b) a 3D SNN based accelerator is proposed. The authors report 52% energy savings and 64% bandwidth improvement. The authors of Clermidy et al. (2014) study a two-layer neural network for objects recognition in a video stream and demonstrate that the total connections' length is reduced three times. The work Joubert et al. (2012) shows that 3D technology can also be used to design spiking neurons (TSV-neuron) by exploiting the TSV's capacitance.

4.3.2 3D technology

As technology node advances, the impact of wiring interconnects parasitics, *i.e.* resistance and capacitance increases. 3D stacking technology affords an effective technique to reduce the wiring length. Different techniques to produce a 3D circuit exist. Here, we focus on 3D technology using TSVs as vertical interconnections. However, the proposed approach is valid using other 3D technologies.

TSVs are used to interconnect stacked dies. The pitch value of TSVs varies depending on the fabrication process. In Gutierrez et al. (2014) a TSV with a pitch of 1.2μm is presented. The main advantage of 3D technology is to replace the long horizontal wires, which introduce large parasitics, by vertical connections, *i.e.* TSVs, with lower

parasitics. Proper partitioning is needed to achieve such a goal. In the next sections, we show how to partition a 2D neural cliques network to create a 3D one.

4.3.3 3D neural cliques

The clique that represents a piece of information stored in the network contains more information than the necessary minimum Gripon and Berrou (2011b). That is why it allows the retrieval based on partial and/or noisy information. Consequently, the hardware implementation is wire-dominated. The long wires impact the energy consumption and time response since all the fanals in the clique have to exchange some signals between them. For that reason, it is interesting to organize fanals in 3D so that they create 3D cliques with shorter connections, and therefore lower energy consumption and time response.

The analog circuit described in 4.2.1 provides a high-performance implementation with physical connections for all the synapses. A significant wiring is necessary to interconnect all the parts of the network. This interconnect is optimized here using 3D technology.

4.3.4 Methodology

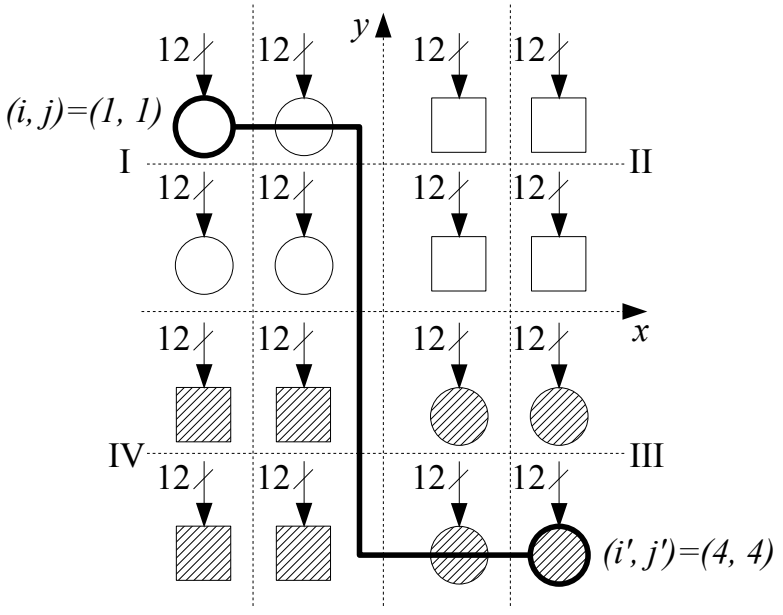


FIGURE 4.5: 2D network example. Different shapes (circles, squares) represent fanals belonging to different clusters. Each fanal has 12 synapses to connect to other fanals. The thick line represents the wire necessary to connect two fanals.

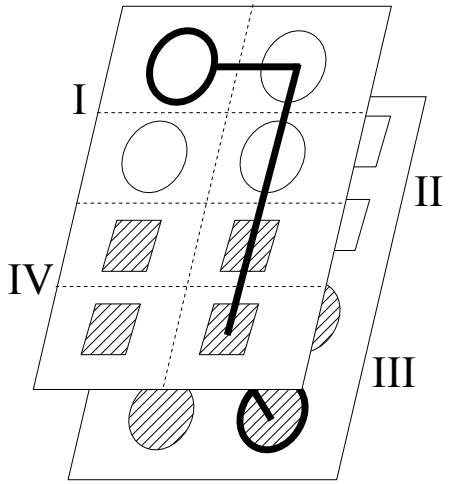


FIGURE 4.6: 3D network (folded network from Figure 4.5). Different shapes (circles, squares) represent fanals belonging to different clusters. The thick line represents the wire necessary to connect two fanals.

Since the clique is created by interconnecting the fanals from distinct clusters, the wires span all over the network. Moreover, the performance of the system depends on the longest wire in the clique since all the fanals activated in the retrieval process exchange the signals through their connections. Therefore, in such a wire-dominated structure, it is beneficial to reduce the length of the connections in the clique to reduce the delays and the energy consumption due to the signals exchanged between the fanals. This can be obtained by folding the network. Figure 4.5 shows an exemplary network. The network is made of four clusters of four fanals each. Fanals belonging to specific clusters are represented with different shapes. Each fanal has 12 synapses to provide all the possible connections. For the simplicity, the length of the wires is measured with the number of hops between the fanals in terms of the Manhattan distance. The Manhattan distance is obtained based on horizontal or vertical paths, as opposed to the diagonal distance. In the presented example the fanal $(i, j) = (1, 1)$ in the cluster I is connected with the fanal $(i', j') = (4, 4)$ in the cluster III. This connection represents the longest possible distance that equals six (the number of dotted lines that have to be crossed). Figure 4.6 shows the same network after folding. One can see that the clusters II and III are moved to another layer and put below the clusters I and IV. To realize the connection from Figure 4.5 a wire of length four is used. The connection to the layer below is ensured by the TSV. Depending on the size and arrangement of the clusters the folding can be done either in x or y direction.

4.3.5 Simulation model

To analyze the gains introduced by using 3D technology the lengths of all the possible connections have to be calculated.

First, the elementary distance between two fanals has to be calculated. This distance depends on the space occupied by the fanal and all its synapses. The area A_{f+s} occupied by one fanal and all its synapses is calculated as:

$$A_{f+s} = A_{fanal} + \psi_s A_{synapse} \tag{4.1}$$

where A_{fanal} is the area of the fanal, $A_{synapse}$ is the area of the synapse, ψ_s is the number of the synapses connected to one fanal and is calculated as:

$$\psi_s = (c-1)\ell. \tag{4.2}$$

In accordance with the aforementioned model each fanal can be connected to any fanal in a different cluster.

Second, the Manhattan distance $dist_{(i,j)(i',j')}$ between two fanals with coordinates (i,j) and (i',j') is (*cf.* Figure 4.5):

$$dist_{(i,j)(i',j')} = |i-i'|\sqrt{A_{f+s}} + |j-j'|\sqrt{A_{f+s}}. \tag{4.3}$$

The area A_{f+s} is approximated as a square and thus one hop distance is $\sqrt{A_{f+s}}$.

Knowing the distance, the unitary resistance and capacitance for the targeted technology, one can obtain the RC delay τ as:

$$\tau_{2D} = R_{per\ \mu m}\, C_{per\ \mu m}\, dist^2_{(i,j)(i',j')} \tag{4.4}$$

where $R_{per\ \mu m}$ and $C_{per\ \mu m}$ are the resistance and capacitance per unit length. In case of a 3D circuit, the resistance and capacitance of the TSV are added to the RC delay:

$$\tau_{3D} = \tau_{2D} + R_{TSV}\, C_{TSV}. \tag{4.5}$$

After the outline of methodology and simulation model, we proceed to exploring the gains resulting from the approach we propose.

4.3.6 General study results

To evaluate the proposed 3D architecture, the gains in terms of total wire length $dist_{total}$ and maximal RC delay τ_{max} obtained by using the 3D technology are explored for different configurations of the network. This includes scaling the number of clusters c and the cluster's size ℓ.

The results are based on physical implementation using 65nm technology and 3D technology with a TSV of resistance and capacitance equal to $R_{TSV} = 2\text{m}\Omega$ and $C_{TSV} = 5\text{fF}$. Using our technology specifications, each TSV is equivalent to horizontal wire length of $0.06\mu\text{m}$. The equivalent horizontal wire length of TSVs represents less than 1% of the total wire length in the 3D worst simulated case. These values have been included in the 3D results to take into account the impact of TSVs.

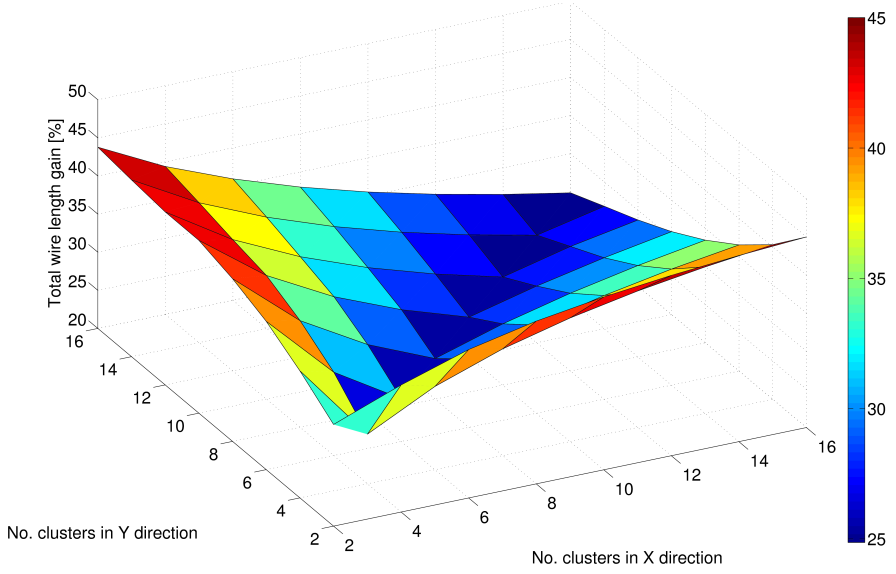


FIGURE 4.7: Total wire length gain compared to 2D in function of the number of clusters in each direction. Square cluster of size two by two is used. The numbers of clusters are given for 2D. For 3D the network is split in two equal parts in such a way to cut its longest dimension.

In the beginning, to facilitate the study, the size of the cluster ℓ is fixed to four (the cluster is square - two by two fanals) and the number of clusters c is scaled. Figure 4.7 shows the gain of the 3D technology in terms of total wire length $dist_{total}$ compared

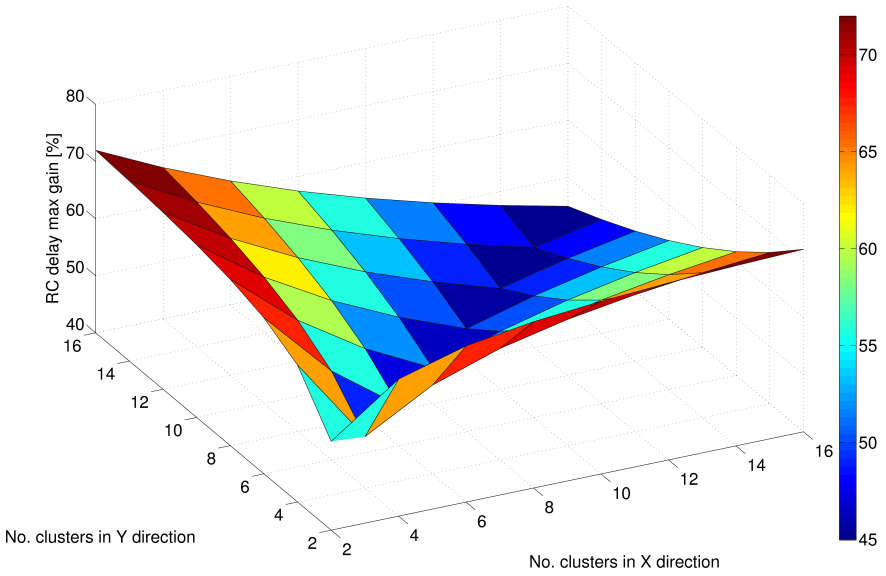


FIGURE 4.8: Maximal RC delay gain compared to 2D in function of the number of clusters in each direction. Square cluster of size two by two is used. The numbers of clusters are given for 2D. For 3D the network is split in two equal parts in such a way to cut its longest dimension.

to the conventional 2D circuit. The gains reach 45% when the network is strongly rectangular. For a given network size it is therefore better to organize the clusters in a rectangle. For instance, for c=16, the gain is 41% when the network is organized in eight by two clusters compared to 27% for four by four clusters. The square networks are clearly distinguished on the surface by their lower gains. Note that it is possible to use few rectangular networks to organize them in a square. Similar trends can be observed in Figure 4.8 that shows the gains in terms of maximal RC delay τ_{max}. In this case the maximal gains reach 72%. It is important to note that the time response of the neural cliques is determined by the longest path in the clique.

Figure 4.7 and Figure 4.8 show that by increasing the number of clusters equally in both x and y directions (square network), the 3D gain is higher for smaller numbers of clusters (*e.g.* for total wire length 33% for 2x2) than bigger ones (25% for 8x8, 24% for 16x16). The reason is that 3D cut partitioning is done in only one direction and consequently, the 3D gain is larger in that direction. Therefore, in case of increasing number of clusters equally in both directions, the effect of long interconnects will not be reduced in one of the two directions which will reduce the overall 3D gain.

Figure 4.9a shows the total wire length gain. The cluster size is kept the same as in Figure 4.7 and Figure 4.8. The red curve shows the evolution of the total wire length gain

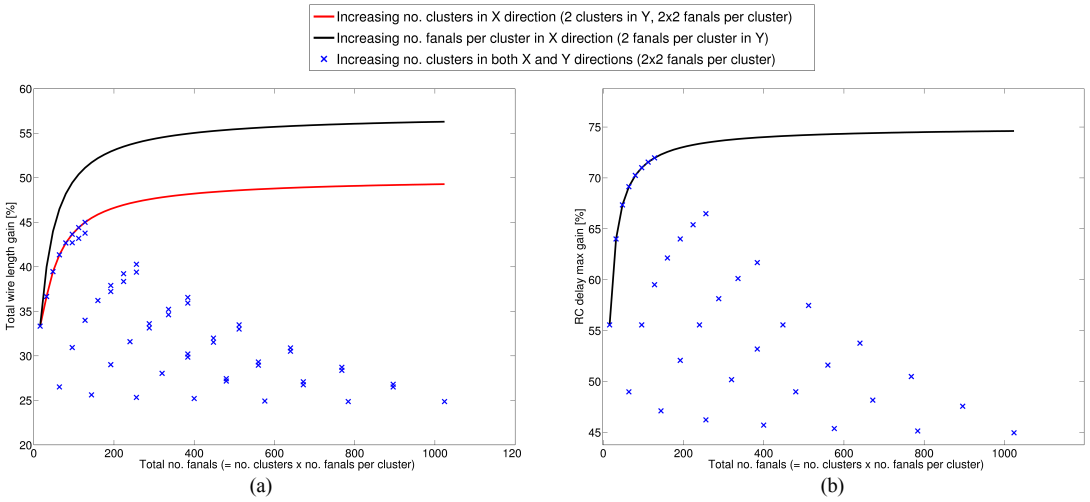


FIGURE 4.9: (a) Total wire length gain (b) maximal RC delay gain in function of the total number of fanals n for different network dimensions.

when the number of clusters in one direction is fixed to two and the network is scaled in another direction. The gain increases quickly for smaller numbers of fanals, then it saturates. It reaches 90% of the maximal value for n=112 fanals which corresponds to c=28 clusters. Blue crosses show the gains when the number of clusters is scaled in both directions, resulting in a network that is less rectangular. The obtained gains are never bigger than when scaling in only one direction. Therefore, it is more beneficial to scale the network in one direction. The black curve presents the gain when c is fixed to four (two by two clusters) and the number of fanals in the cluster ℓ is scaled. Now the gains are higher than in the former case for the same total number of fanals. Again, the same trend is observed. The gain increases quickly for smaller numbers of fanals, then it saturates. 90% of the maximal gain is reached for n=128 fanals which corresponds to ℓ=32. Comparing these two curves leads to the conclusion that for a given total number of fanals n from the 3D point of view it is more beneficial to have bigger clusters. This is consistent with Gripon and Berrou (2011a) where authors state that from the storage capacity point of view for a given total number of fanals n it is better to have bigger clusters since the density of the connections established in the network grows slower. This means that optimizing the gains coming from the theoretical model and hardware 3D implementation is not contradictory.

Figure 4.9b shows the similar analysis for the gains in terms of maximal RC delay τ_{max}. The gains, reaching 74%, are bigger than for total wire length. There is no difference

TABLE 4.2: Total wire length gain in percentage compared to 2D for a given number of clusters in x and y direction

y \ x	2	4	6	8	10	12	14	16
2	33	37	40	41	43	44	44	45
4	37	27	31	34	36	38	39	40
6	40	31	26	29	32	34	35	37
8	41	34	29	25	28	30	32	34
10	43	36	32	28	25	27	29	31
12	44	38	34	30	27	25	27	29
14	44	39	35	32	29	27	25	27
16	45	40	37	34	31	29	27	25

TABLE 4.3: Maximal RC delay gain in percentage compared to 2D for a given number of clusters in x and y direction

y \ x	2	4	6	8	10	12	14	16
2	56	64	67	69	70	71	72	72
4	64	49	56	60	62	64	65	67
6	67	56	47	52	56	58	60	62
8	69	60	52	46	50	53	56	58
10	70	62	56	50	46	49	52	54
12	71	64	58	53	49	45	48	51
14	72	65	60	56	52	48	45	48
16	72	67	62	58	54	51	48	45

between the maximal RC delay when increasing the number of clusters c or the number of fanals per cluster ℓ because in both cases the length of the longest connection is the same. Similarly, it is beneficial to scale the network only in one direction.

To give more insight in the gains represented by the blue crosses (when the number of clusters is increased in both x and y directions), Table 4.2 shows the gains obtained for total wire length (*cf.* Figure 4.9a). The table gives the number of clusters in each direction (x or y) and the corresponding gain compared to 2D. For instance, when x=2 and the clusters are added only in y direction, the gain compared to 2D increases from 33 to 45%. If a network of 16 clusters is considered, for two clusters in x direction and eight clusters in y direction, one obtains 41% gain whereas for four clusters in x and y direction one obtains only 27%. This shows once again, that it is more beneficial to

organize the clusters in a rectangle rather than in a square. Similar analysis is shown in Table 4.3 for the maximum RC delay.

Additionally, the power of the interconnects is directly proportional to the total wire length $dist_{total}$:

$$P_{interconnects} \propto dist_{total}. \tag{4.6}$$

Consequently, in case of reducing the total wire length $dist_{total}$ by 55%, as shown in Figure 4.9a, the power of the interconnects is reduced by the same percentage.

4.3.7 Case study simulation results

In this section, a real-world test-case is used to obtain the dimensions of the neural cliques and explore the gains of using 3D technology. In the considered application the time response of neural cliques is of first importance. Therefore, high-performance communication structure is essential.

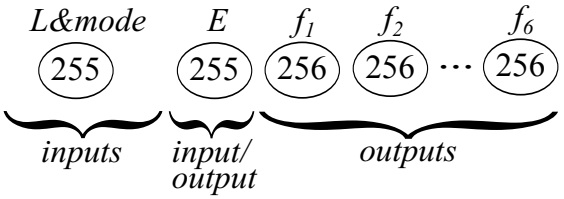


FIGURE 4.10: Network of neural cliques structure for 3D case study. Latency L and operating mode combinations are stored in one cluster and used to stimulate the network. One cluster of 255 fanals is created to store all the corresponding energies. It can be used either as input or output. The set of frequencies is retrieved from the network.

The considered test-case is a neural cliques-based power management decision unit for MAGALI MPSoC already introduced in Subsection 3.5.1.1. We recall that six VFIs are used in the application, each VFI can choose from 256 frequencies. The speed of each VFI is determined by a global latency constraint and an operating mode offering different data rates. The global latency constraint has 51 possible values, there are 5 different operating modes. Additionally, in each message a global estimation of the energy consumed by all the VFIs is included. Thanks to that, when the energy consumption is the main constraint (*e.g.* low battery level), the maximum affordable energy is used as the input to the network and the frequencies and the corresponding latency are retrieved. Otherwise, the estimation of the consumed energy is retrieved from the network based on the latency and operating mode. This kind of flexible controller is of high interest in low-power

TABLE 4.4: The gains obtained for 3D neural cliques used as power management controller. The results are normalized to 2D circuit

	2D	3D		Gain	
		Case 1	Case 2	Case 1	Case 2
Total wire length	1	0.65	0.83	35%	17%
RC delay max	1	0.43	0.69	57%	31%

systems. Consequently, the network is made of six clusters of 256 fanals to store all the possible frequency values, one cluster of 255 fanals to store all the possible latency and operating mode values (51 latencies times 5 operating modes), and one cluster of 255 fanals to store all the corresponding energies. The structure of the network is illustrated in Figure 4.10. For 2D, the network is organized in four clusters in x direction, two clusters in y direction. In each cluster 16 fanals are placed in each direction. For 3D, there are two possible cases: 1) network is cut in x direction (two clusters in each direction on each die), 2) network is cut in y direction (four clusters in x direction and one cluster in y direction on each die).

The gains in terms of total wire length and RC maximum delay compared to the conventional 2D circuit are summarized in Table 4.4. Case 1 gives better results. In this application, using 3D technology allows for 35% total wire length reduction and consequently, the same reduction in terms of the power of the interconnects. Furthermore, the maximum RC delay is reduced by 57%.

As TSV insertion adds additional area for creating the vertical connections, area overhead analysis should be done. In the test-case, the number of used TSVs is 1048576. The TSV pitch is 1.2μm Gutierrez et al. (2014), consequently TSVs occupy an area of 1.5mm^2 which represents 5% of the total chip area. Advanced 3D technologies with smaller pitch can be used to reduce the area overhead. For example, 3D sequential integration Batude et al. (2014) can reduce TSVs area overhead to 1.4%.

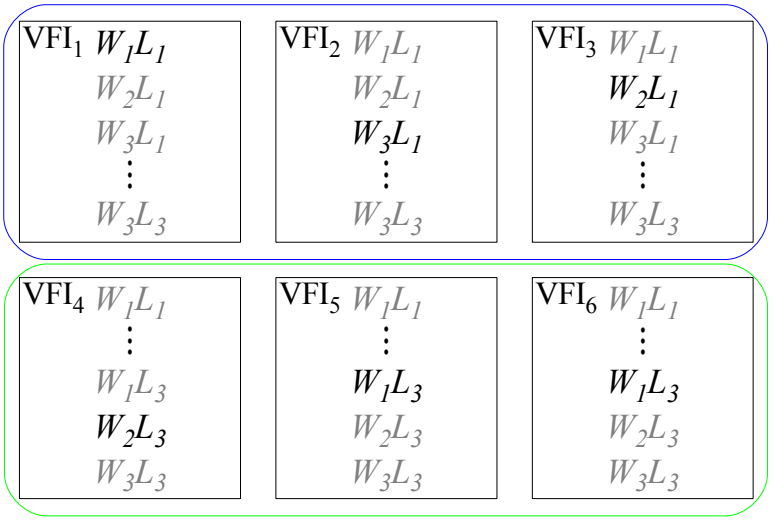


FIGURE 4.11: Homogeneous MPSoC with six VFIs and two applications mapped. Each VFI can process three different workloads W and there are three possible latency constraints L.

4.4 MPSoC power management: comparison with game theory decision unit

4.4.1 Generic neural cliques structure

The generic neural cliques structure has to support either homogeneous or heterogeneous MPSoC architecture. Homogeneous architecture, where generic PEs are exploited, can execute different types of tasks differing in workload W. Workload can be expressed as current fullness of the PE's FIFO or by a number of clock cycles N necessary to compute a task. Different applications can be mapped on such a generic platform which implies that VFIs can be organized in different configurations with different latency constraints L and workloads W. Figure 4.11 shows an MPSoC with six VFIs. Each VFI can process three different workloads and there are three possible latency constraints. Any application can be mapped on the MPSoC. Here, two applications are mapped. The first application (marked in blue) is processed by three VFIs. All the computations have to be finished within an L_1 global latency constraint. VFI_1 has to process a workload W_1, VFI_2 W_3, VFI_3 W_2. The second application (marked in green) is processed by other three VFIs. All the computations have to be finished within an L_3 global latency constraint. VFI_4 has to process a workload W_2, VFI_5 W_1, VFI_6 W_1.

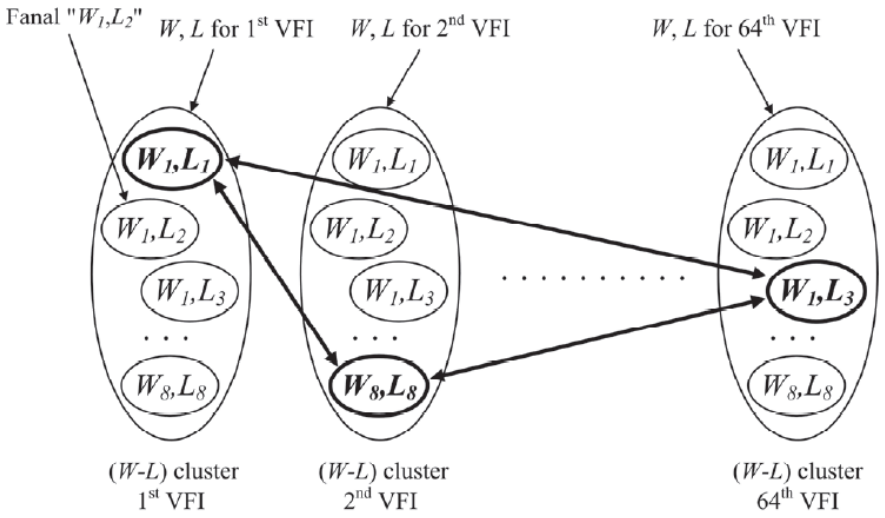


FIGURE 4.12: $(W - L)$ clusters for generic power management. There are 64 VFIs, eight workloads W, eight latencies L possible for each VFI.

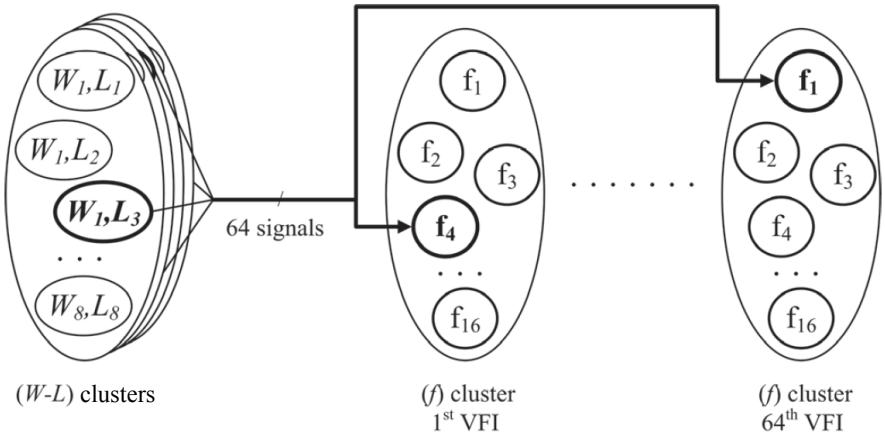


FIGURE 4.13: (f) clusters for generic power management. There are 64 VFIs, 16 frequencies f possible for each VFI.

The information about the possible workloads, latencies and power modes to be set has to be stored in the network. In this generic study, we consider an MPSoC with 64 VFIs (Larras et al., 2013b) that is bigger than in the introductory example and is also considered in (Puschini et al., 2008). The workload W and latency L are quantified on three bits each and coupled together as one value written on six bits. As in Beigné et al. (2009) the DVFS actuator is able to adjust the necessary voltage upon the given frequency. The frequency f is stored on four bits giving 16 possible frequency settings. A set of workloads and latencies represents different situations in the MPSoC. For a given situation, a frequency for each VFI ensuring the latency constraint is obtained form an offline optimization system and is stored in networks of neural cliques. Two

different sizes of clusters are used. Workloads W and latencies L are stored in $(W-L)$ clusters of $\ell_1 = 64$ fanals. These can be externally stimulated to input the current W and L of each VFI to the network. Frequencies f are stored in (f) clusters of $\ell_2 = 16$ fanals. One cluster of each type is associated with each VFI, and therefore $c_1 = c_2 = 64$. The $(W-L)$ clusters are depicted in Figure 4.12. The (f) clusters are stimulated by the $(W-L)$ clusters, as illustrated in Figure 4.13. There is no need for connections between fanals in (f) clusters since 64 stimuli they receive are enough to discriminate the right fanal.

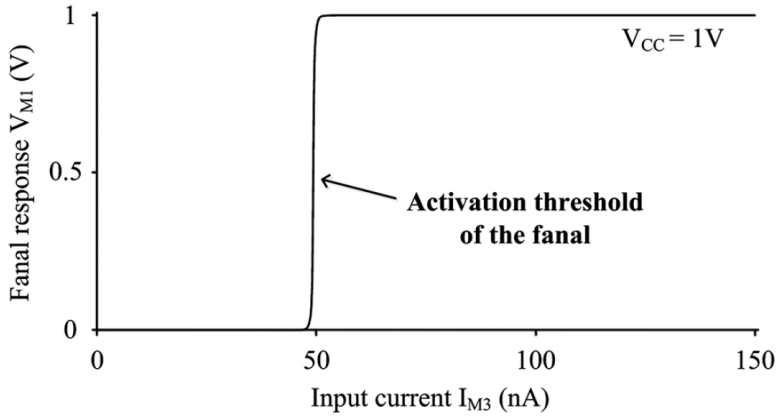


FIGURE 4.14: Fanal's response with respect to input current at node A^k.

Deep submicron technologies are prone to leakage currents. This implies that even if switch M_3^S in synapse, Figure 4.1, is off, a tiny current still adds up at node A^k. This infers a limitation in the number of synapses a fanal can have in order to avoid self-activation of the fanal. From simulations, the leakage current in M_3^S is 0.07nA. As shown in Figure 4.14, activating a fanal requires an input current at node A^k equal to 49nA. Thus, the number of synapses per fanal is limited to 700 to avoid self-activation when the fanal is not stimulated. This allows to store ten cliques that involve each fanal and consequently, assuming uniform distribution in $(W-L)$ clusters, 640 configurations.

The time response to obtain a result in the (f) clusters is divided in three steps. First, the stimulation time lasts from the moment the $(W-L)$ clusters are stimulated until the first $(W-L)$ fanals are activated. Then, signals are exchanged within the $(W-L)$ clusters and they stimulate fanals in the (f) clusters. The fanals activated in the (f) clusters are used by DVFS actuators to select a new VFI frequency.

4.4.2 General comparison with game theory decision unit

The generic neural cliques implementation is compared to game theory decision unit based on the results given in (Puschini et al., 2008) and implementation presented in (Mansouri et al., 2010b). The implementation results are referred to the same VFI (Jalier et al., 2010) as in (Mansouri et al., 2010b). Both game theory implementation and VFI are designed for the same 1V supply ST 65nm technology as the neural cliques implementation.

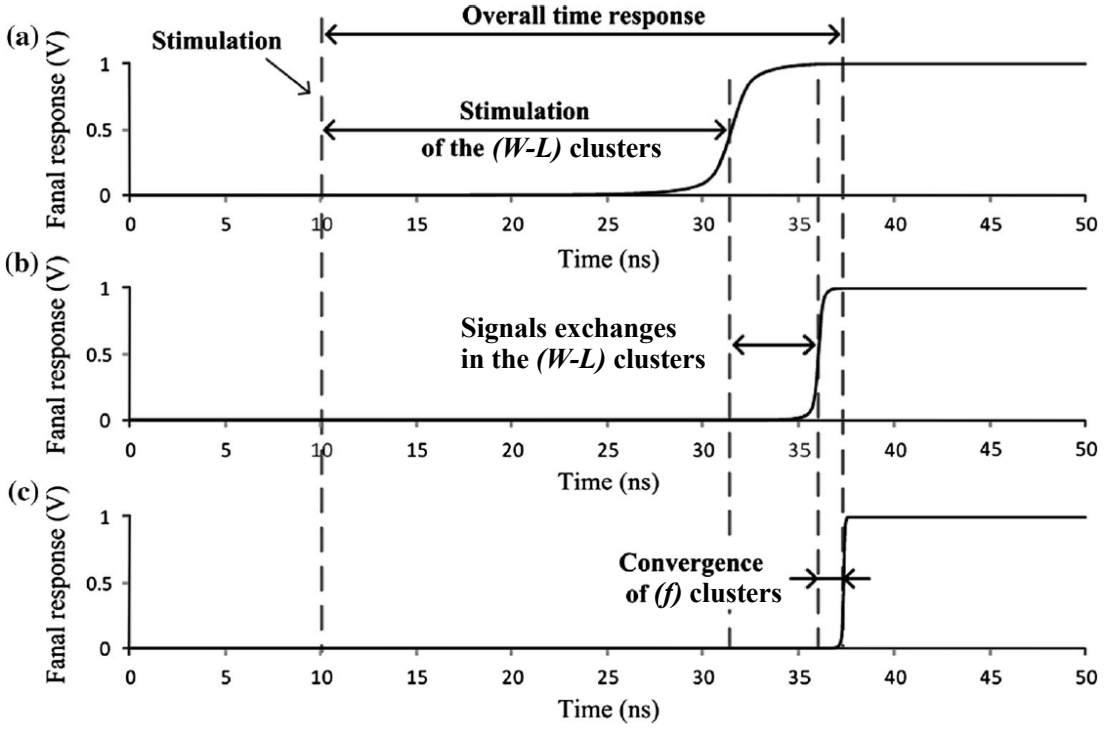


FIGURE 4.15: Three time steps composing the overall time response of the network to obtain a result in the (f) clusters. (a) $(W-L)$ clusters stimulation, (b) Signals exchanges in the $(W-L)$ clusters, (c) Convergence of (f) clusters.

Since clusters work in parallel and are identical, the time response of the whole network is equal to the time response of one cluster. This time response is a function of the number of synapses, i.e. the number of loads connected to node A^k. More capacitance at node A^k implies a slower voltage-to-current conversion and thus an increased time response. Adding synapses also means that more clusters are added. As explained in the end of the last section, the time response to obtain a result in the (f) clusters is divided in three steps. These three phases are represented in Figure 4.15, the stimulation occurs at 10ns of simulation time. The stimulation time is the longest of the three, because a fanal

TABLE 4.5: Comparison between neural cliques and game theory decision unit.

	Mansouri et al. (2010b)	This work
Number of VFIs	64	64
Energy per search [nJ]	68.6	0.01
Surface area [μm^2]	14000	92000
Time response [μs]	120	0.027

is stimulated only by the external stimuli. The two other periods are shorter due to the fact that other fanals also contribute to the stimulation. Consequently, the capacitances in the corresponding parts of the network are charged by a higher current causing the voltage at node B^k to reach faster the threshold of the logic buffer in Figure 4.2.

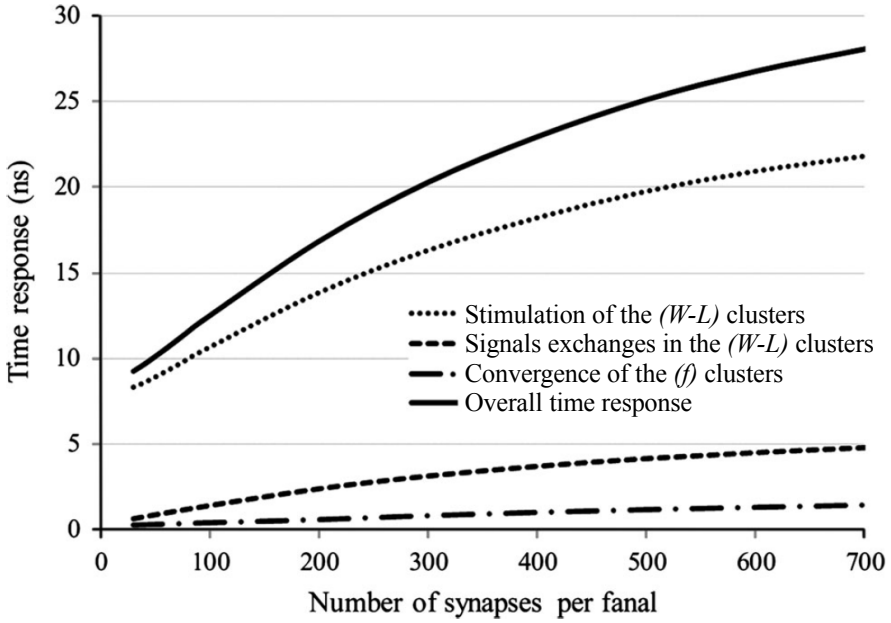


FIGURE 4.16: Time response of a cluster with regard to the number of synapses per fanal during different steps of retrieval process.

The influence of the number of synapses per fanal on each step of the retrieval process is shown in Figure 4.16. We observe that the time response of a cluster, and therefore the whole network with 630 synapses per fanal is 27ns. The unitary current I_{UNIT} is chosen according to the number of signals that can arrive at fanal's synapses Larras et al. (2014). To adapt a frequency f to new working conditions, the circuit consumes 10.25pJ per VFI, Table 4.5. This represents 0.4% of the VFI's energy consumption. A $(W-L)$ cluster occupies 73475μm^2 and an (f) cluster 18369μm^2. The per VFI area is the sum of a $(W-L)$ cluster and an (f) cluster and represents 3% of the VFI's area.

TABLE 4.6: MC-CDMA TX application parameters (Lattard et al., 2007). Characterized at $f_{nom,i} = 200$MHz.

	Encod	Bit inter	Mapp	FHT	OFDM
N_i [clk cycles]	576	560	560	24496	115248
$E_{nom,i}$ [pJ/clk cycle]	77.6	183.3	183.2	251.5	1516.5

The game theory occupies smaller area. However, this is acceptable considering the huge gains in terms of time response (4500 faster) and energy consumption (6800 smaller).

4.4.3 MPSoC power management for MC-CDMA transmitter

After the generic neural cliques implementation study, a test-case is exploited to compare the gains resulting from power management by neural cliques decision unit to energy gains from power management relying on game theory.

4.4.3.1 MC-CDMA transmitter implemented on FAUST platform

To compare energy gains the same applicative test-case as in (Puschini et al., 2009) is used. The considered MPSoC is a system for 4G telecom applications called FAUST (Lattard et al., 2007). It is a heterogeneous MPSoC integrating with an asynchronous NoC providing high-throughput communication. An advanced telecommunication application MC-CDMA Matrice, configured to work in QPSK-LQ specification, is mapped on the MPSoC. It consists of five tasks shown in Figure 4.17: encoder, bit interleaving, mapping, FHT, and OFDM modulator. Each task is assigned to a PE. As in (Puschini et al., 2009), the DVFS circuit is able to set 32 frequencies between 50 and 300MHz. The parameters of the application, necessary to calculate the energy (1.10), are summarized in Table 4.6. The values regarding energy $E_{nom,i}$ are measured at $f_{nom,i} = 200$MHz. When using clock gating, a power reduction of 80% is observed. Under the assumption made in (Puschini et al., 2009), the static power in the FAUST chip is considered to be the tenth of the dynamic power.

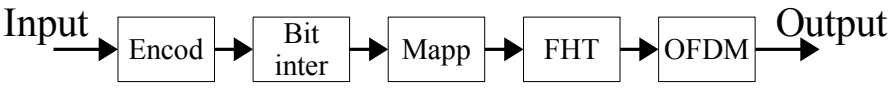


FIGURE 4.17: MC-CDMA TX application graph (Lattard et al., 2007).

4.4.3.2 Network of neural cliques used as power management unit

The network used in the test-case is a particular case of the generic neural cliques structure presented in Section 4.4.1. Further optimized structure of neural cliques is used in the subsequent section.

As a consequence of using dedicated hardware blocks, the workload expressed by a number of clock cycles N necessary to compute a task is no longer a relevant type of parameter input to the network. The workload is fixed for all the VFIs and equal to N_i given in Table 4.6. Therefore, only the latency is considered as a variable input. Moreover, in (Puschini et al., 2009) all the VFIs are organized in a chain that has to output data in a certain time implying that all the VFIs have the same latency constraint. Therefore, to each PE corresponds a VFI associated with an L and an f cluster. Consequently, the cliques in the L clusters group the same fanal in each cluster. This corresponds to having four synapses per fanal. To keep the same cluster size as in generic structure, L is quantized on six bits and f is quantized on four bits. L ranges from 475μs to 1ms. There are therefore 64 stored configurations in the network. Behavioral simulations show that all the frequencies f are correctly retrieved based on the initial stimulation.

As shown in Figure 4.16, the network implemented in 65nm technology has a time response of 9ns. The FAUST MPSoC is implemented in 130nm technology, and therefore we estimate the time response of neural cliques implemented in the same technology. Parasitic capacitors in 130nm are 3.5 times larger than in 65nm (Larras et al., 2014). Consequently, the time response of the network implemented in 130nm is estimated to be 32ns. This is still much less than game theory decision unit in 65nm which takes 105μs to converge (Puschini et al., 2008, Mansouri et al., 2010b)

4.4.3.3 Energy gains

First, from a chosen budget the energy savings brought in by neural cliques can be compared to the method consisting in setting a global frequency to all the VFIs. Knowing the total number of clock cycles needed to finish all the tasks, and the latency constraint,

the global frequency is computed as follows:

$$f_{glob} = \frac{\sum_{i=1}^{\nu} N_i}{L}. \tag{4.7}$$

Considering that the 64 possible latency values between 475μs and 1ms are uniformly distributed, the average consumed energy per application run is 281μJ. Using neural cliques with the 64 L values stored instead of the globally adjusted frequency allows saving on average 40μJ. This value accounts for 14% energy savings.

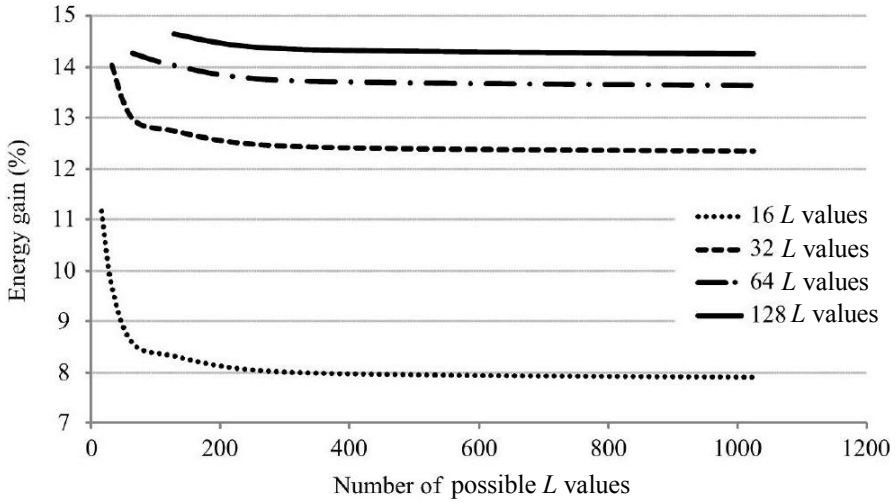


FIGURE 4.18: Energy gain compared to global frequency with regard to total number of possible latency L values. Different curves represent gains for different numbers of stored latency values.

The number of stored L values may be lower than the total number of possible values. The gap between the number of stored and possible L values results in suboptimal frequency settings. This reduces the energy gain. There is thus a compromise between energy gain and neural cliques complexity. Therefore, it is interesting to estimate the energy savings with respect to the number of L values stored in the network. Figure 4.18 depicts the energy gain with regard to the global frequency setting method with regard to the number of latency values L requested by the system for different cluster sizes. Based on the number of requested L values and the desired energy gain, one can dimension the network properly.

Second, the energy savings brought in by neural cliques compared to game theory are assessed. As reported in (Puschini et al., 2008), the controller based on game theory offers an average of 89% optimization quality, *i.e.* the distance between the computed

solution and the optimal frequency, with a maximum at 93%. An advantage of neural cliques compared to game theory is that they can store solutions closer to optimal since they can be obtained by solving 1.12 using mathematical software tools. The comparison is done with the example presented in (Puschini et al., 2009) which considers a latency $L = 850\mu$s and an energy budget of 280μJ. Both game theory and the neural cliques decision units provide one frequency per task. These frequencies are chosen among the 32 possible values the DVFS actuator can provide for game theory, while 16 in case of neural cliques. Only taking the energy saved into account once either game theory or neural cliques have converged to a solution, simulations show that both decision units yield similar energy savings, 38% and 40%, respectively. This result can be improved if neural cliques decision unit allows choosing among the 32 frequencies the DVFS actuator can provide instead of only half. In this case, the energy savings go up to 46%. This is achieved at the expense of a surface area per VFI which increases by 33% while the time response goes up by 1ns.

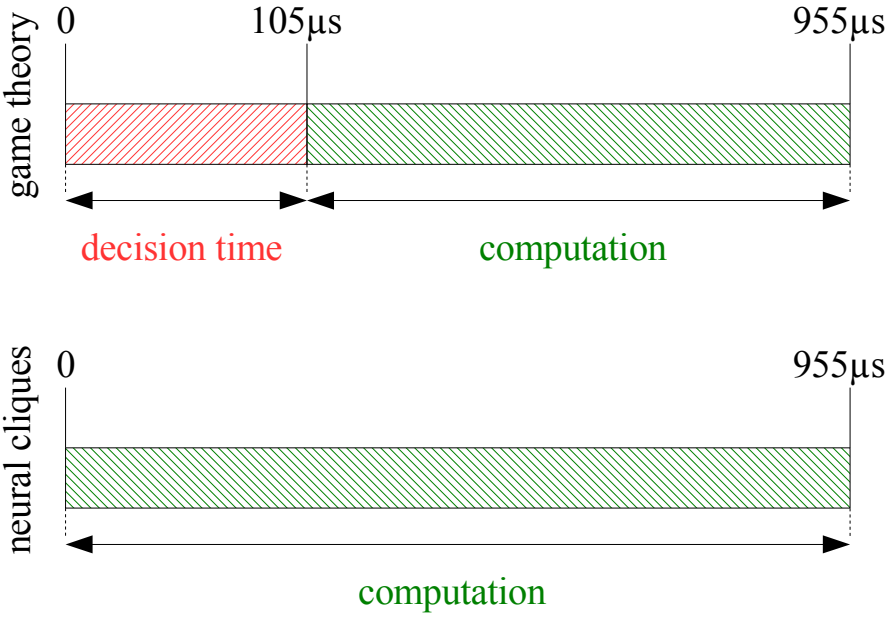


FIGURE 4.19: Breakdown of decision time and computation time in case of game theory and neural cliques decision units. The global latency constraint given by the higher level is 955μs.

However, these results do not account for the energy consumed by the VFIs while the power management systems are converging. The game theory takes 105μs to converge (Puschini et al., 2008, Mansouri et al., 2010b). This duration should be taken into account by the higher system layers when sending the requests relating to the task

completion time. It implies that the latency is 955μs with 850μs devoted to actually performing the task. Since the convergence time of neural cliques circuit is negligible compared to that of game theory, 955μs is also the time assigned for computing the tasks if the neural network is used. This described situation is depicted in Figure 4.19. Thanks to the longer time assigned to computing the tasks, VFIs use less energy. This yields a 58% energy savings, resulting both from the faster convergence and more optimal solutions stored in the network. This is achieved with only 16 frequencies considered available in the DVFS actuator. When 32 frequencies are available, neural cliques save 60% of the energy budget. The slight improvement is due to the fact that neural cliques decision unit outputs a frequency vector not far from optimal. Furthermore, faster convergence also implies more efficient design. Since the amount of data waiting for the computation is smaller, memory buffers in the system can be reduced. These design improvements are not taken into account in the presented gains.

The decision unit that we propose in this section consumes 6800 less energy and has 4500 smaller time response than the decision unit that relies on game theory. The practical test-case shows that it allows to obtain energy savings of 60% compared to 38% when game theory decision unit is applied. The following section presents a comparison with a decision unit relying on CAM-SRAM associative memory.

4.5 MPSoC power management: comparison with CAM-SRAM associative memory

In this section, neural cliques are compared to a CAM-SRAM associative memory (Boguslawski et al., 2015c,d). Here, we consider the neural network used as a classical CAM associative memory for applications in which the subsets of input and output segments are known beforehand. Consequently, there is no need for the connections between all the clusters. Only the connections leading from input clusters (the clusters associated to input segments - the known segments) to output clusters (the clusters associated to output segments - the unknown segments) are kept. This allows reducing the number of connections and simplifies the implementation.

Firstly, the complexity of neural cliques and CAM is compared in terms of transistor count. The introduced complexity metric accounts for the complexity of the cells without taking into account the interconnections. Since in CAM all the memory cells are interconnected and in neural cliques only certain nodes, the interconnection in CAM is more complex. The introduced complexity metric should not be confused with the surface of the circuit. One can have a very complex circuit occupying a small area. It is also a question of whether the circuit is analog or digital. Contrary to analog circuits, digital ones often use minimum-sized transitors and occupy less area. However, analog circuits usually consume less energy.

4.5.1 Neural cliques-based associative memory - implementation complexity

The hardware complexity of the networks of neural cliques depends on their dimensions and functionality. The dimensions are defined by the number of fanals and the way they are divided into clusters. The functionality is determined by programmable or non-programmable connections and the type of nodes (buffer or fanal). The network is divided into input clusters corresponding to known segments of messages and output clusters corresponding to unknown segments of messages. The nodes in input clusters are made from simple buffers, whereas the nodes in the output clusters are the fanals with WTA functionality. Assume that the network consists of c_{in} input clusters of different sizes, c_{out} output clusters of different sizes, c_1 being the number of clusters of size ℓ_1, c_2 being the number of clusters of size ℓ_2, and so on. Then, the total number of connections ψ that can be established in the network is calculated as follows:

$$\psi = \sum_{k \in c_{in}} c_k \ell_k \sum_{j \in c_{out}} c_j \ell_j \tag{4.8}$$

where k denotes the clusters of the same size only among the input clusters and j denotes the clusters of the same size only among the output clusters. The total number of connections ψ grows linearly with the number of fanals n.

The network is a regular structure that consists of fanals and connections, each of them representing a constant complexity in its hardware equivalent. Therefore, the total complexity of a given network is a function of its dimensions, complexity of each

element and its functionality. The complexity is expressed by means of the total number of transistors necessary to implement the network using the circuits proposed in Larras et al. (2013a). A programmable connection is composed of four transistors, whereas a non-programmable connection uses two transistors so in the following equation, $p = 4$ or 2. The complexity κ of the network is given by:

$$\kappa = b \sum_{k \in C_{in}} c_k \ell_k + t \sum_{j \in C_{out}} c_j \ell_j + p\psi. \tag{4.9}$$

b represents the number of transistors necessary to construct a simple buffer and equals four. These buffers are used to replace the fanals in the input clusters. The first sum represents the number of fanals in the input clusters. t represents the number of transistors necessary to construct the fanal and equals eight. The second sum represents the fanals in the output clusters so it is multiplied by t.

4.5.2 CAM-based associative memory - implementation complexity

For the CAM, as in (4.9), the unit of κ is the number of transistors necessary to implement the circuit. The standard 10-T NOR-type CAM cell is used which serves both search word storage and comparison functions Pagiamtzis and Sheikholeslami (2006). Since 10 transistors are used to store one bit and assuming that w and z represent the dimensions of the CAM (Figure 1.6), the complexity of the CAM is calculated as:

$$\kappa_{CAM} = 10wz. \tag{4.10}$$

To construct the memory in Figure 1.6, the RAM is needed. Since low-power integrated circuits are targeted here, the SRAM is chosen as the memory. The conventional 6-T SRAM cell is used Do et al. (2011a). Knowing that six transistors are used to store one bit and assuming that y and x represent the dimensions of the SRAM (Figure 1.6), the complexity of the CAM and SRAM-based memory is calculated as follows:

$$\kappa_{CAM+SRAM} = 10wz + 6yx. \tag{4.11}$$

To keep the analysis sufficiently general, only the number of transistors used to construct the memory matrix storing a given number of information bits is taken into account. In

case of the CAM or SRAM, the peripherals needed to control the memory matrix depend strongly on the memory organization and design objectives. The overhead related to search data registers, drivers, matchline precharge circuitry, sense amplifiers, control circuitry for CAM and row decoder, input/output registers, controller for SRAM are not considered. Thus, only the storage complexity, not control complexity, is taken into account. For the network, the storage and processing are combined in one structure and are not separable.

The sizes of the transistors used in the network and in the CAM/SRAM are not the same, their working principle is also different. As it will be shown later, in case of the neural cliques the complexity has a stronger impact on the area, whereas in case of the CAM and SRAM it impacts more the energy consumption.

4.5.3 Implementation complexity comparison

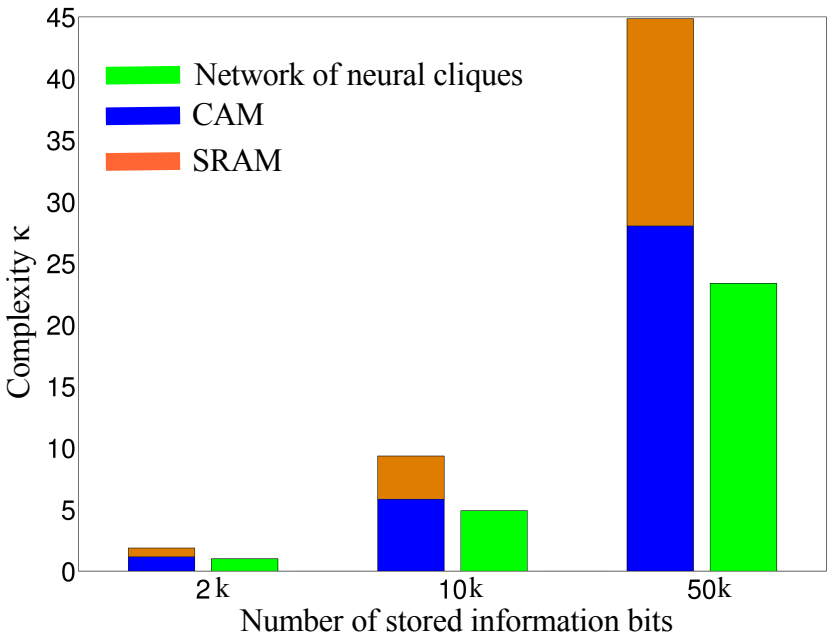


FIGURE 4.20: Implementation complexity in function of information bits stored (xy, cf. Figure 1.6) normalized to the minimal value for the network of neural cliques. Programmable connections are used for neural cliques.

Figure 4.20 presents the comparison of CAM-based memory (Figure 1.6) and neural cliques for variable number of stored information bits (e.g. associations between 256 different values of 8 bit long words give 2k bits - xy, cf. Figure 1.6). Programmable

connections are used in neural cliques. The results are normalized to the minimal κ of the network. The figure shows the contribution of each part of the associative memory on the overall complexity. First, one can see that the complexity κ_{CAM} is always greater and increases faster than the complexity of the network of neural cliques. It also has a major contribution to the complexity of the CAM-based associative memory (for equal CAM and SRAM sizes). As the memory size increases, the gap between CAM-based associative memory and neural cliques increases.

4.5.4 LTE receiver implemented on MAGALI platform

For the presented comparisons two cases are considered:
a) neural cliques are used to replace the CAM in Figure 1.6 and provide the associations between the search word and the address for the SRAM,
b) neural cliques are used to replace the entire CAM and SRAM couple and provide the associations between the search word and the output word.

The MPSoC platform under consideration is the MAGALI MPSoC Clermidy et al. (2010) already introduced in Subsection 3.5.1.1. Six types of processors are used in the test-case, each processor can choose from 256 clock frequencies. The speed of each processor is determined by a global latency constraint and an operating mode offering different data rates Clermidy et al. (2009). The global latency constraint has 51 possible values, there are five different operating modes. Therefore, the associative memory has to perform the associations between a search word that has 255 possible values (all the possible combinations between the global latency constraint and the operating mode) and six output words each having 256 possible values.

4.5.4.1 Dimensions of CAM and SRAM

To ensure the associations necessary in the test-case, the sizes of CAM and SRAM are $w = 8$, $z = 256$ (one additional replica line for timing control Pagiamtzis and Sheikholeslami (2006)), and $y = z = 256$, $x = 48$ (Figure 1.6).

4.5.4.2 Dimensions of neural cliques

Two comparisons are considered:
a) neural cliques are used to replace the CAM: in this case the network consists of one input cluster of 255 fanals and four output clusters of four fanals,
b) neural cliques are used to replace the entire CAM and SRAM couple: the network consists of one input cluster of 255 fanals and 24 output clusters of four fanals.
Indeed, the 256 output values are represented with an eight-bit encoded word. Each cluster represents two bits showed by the state of its four fanals.

4.5.4.3 Simulation results

Firstly, the comparison between neural cliques and CAM is made. Thus, neural cliques are used to associate the search word with the address used by the SRAM. The obtained results are compared with several CAM references, Table 4.7. The energy consumption of the CAM of the size necessary in the test-case is estimated based on the appropriate references. One can see that networks of neural cliques are more efficient than CAMs in terms of energy consumption per search which is related to the different operation principle described in Section II. Additionally, in Table 4.7, the conventional figure of merit (FOM) is presented. The network occupies an area in between the two CAMs for which the area is given. However, in case of CAMs only the cell matrix is taken into account. Since there is no CAM of the specific size used in the presented test-case available in literature, the control overhead is hard to estimate. The search delay of CAMs is shorter than for the network. Nevertheless, in the considered test-case of power management in MPSoCs, there is no need to provide the search as fast as CAMs, especially that it comes at the additional cost of power consumption.

TABLE 4.7: Comparison between neural cliques and CAM. Supply voltage 1V, Technology 65nm, $w = 8$, $z = 256$, cf. Figure 1.6

	Agarwal et al. (2006) Dual-supply	Do et al. (2013) Conventional	Do et al. (2013) Parity bit power gated	Arsovski and Wistort (2006) Self-referenced sensing	Do et al. (2014) Automated background checking scheme	This work
Energy per search [pJ]	18.4	3.8	1.6	2.0	1.6	1.2
FOM [fJ/bit/search]	9	1.85	0.76	0.99	0.77	0.59
Surface area [μm^2]	N/A	N/A	N/A	4710 (matrix only)	7782 (matrix only)	5511
Search delay [ns]	0.4	1.3	0.8	0.6	1.2	15.6
Simulated(S)/ Measured(M)	S	S	S	M	M	S

TABLE 4.8: Comparison between neural cliques and CAM and SRAM couple. Technology 65nm, $w = 8$, $z = 256$, $y = 256$, $x = 48$, cf. Figure 1.6

	Do et al. (2014) (CAM) + ST SRAM	This work
Energy per search [pJ]	13.6	7.0
Surface area [μm^2]	22791	26948
Search delay [ns]	(matrices only) 2.3	15.6
Supply (SRAM)	1.1V	1V
Sim.(S)/Meas.(M)/ Estim.(E)	M (CAM) E (SRAM)	S
Complexity	94208 (eq. (4.11))	17468 (eq. (4.9))

Secondly, neural cliques are compared to the entire CAM and SRAM couple. Thus now, neural cliques provide the associations between the search word and the output word. The results are shown in Table 4.8. For this comparison the most energy-efficient CAM from Table 4.7 for which all the data is available is coupled with an SRAM. Based on the SRAM from STMicroelectronics with y that corresponds to the test-case parameters and $x = 18$ and 32, the parameters of the SRAM with $x = 48$ are estimated. One can see that the energy consumption of the network in the considered test-case is 48% smaller than that of the CAM and SRAM. The energy consumption of the network is smaller even compared to the ST SRAM with $x = 18$ (7.51pJ). This comes at the cost of slightly larger area of the network, acceptable taking into account the energy reduction, and longer search time. The complexity κ (4.9) of the network is smaller than $\kappa_{CAM+SRAM}$ (4.11). Due to its dominating size compared to the CAM, the SRAM accounts for the major part (73728 transistors, 78%) of the complexity (4.11).

Both types of associative memory are evaluated in a practical scenario and the obtained results show that for a group of applications where the energy consumption is the main constraint, neural cliques can be an interesting alternative to typically used CAM designs. Furthermore, the general analysis shows that when the memory size increases, the gap between the implementation complexity of CAM and neural cliques becomes larger as well. The comparison between neural cliques and the entire CAM-based memory (with SRAM) reveals even greater energy savings (48%) which is consistent with the bigger gap in their implementation complexity.

4.6 Conclusion

We propose a hardware neural cliques structure to design a power management decision unit. Comparing to game theory decision unit, the decision unit that we propose consumes 6800 less energy and has 4500 smaller time response than the decision unit that relies on game theory. The practical test-case shows that it allows to obtain energy savings of 60% compared to 38% when game theory decision unit is applied. Comparing to CAM-SRAM decision unit, the decision unit we propose offers 48% energy savings and is less complex. Moreover, as the associative memory size increases, the gap in complexity of both types of memory increases. We discuss the impact of variability on the analog implementation of neural cliques and we envision three methods to improve the reliability of these circuits. Further, we estimate the cost of programming the synapses and we propose a novel 3D interconnect approach for high-performance neural cliques' implementation. A general study shows up to 55% reduction in terms of total interconnect length and interconnect power consumption, and 74% reduction of the maximal interconnect delay. The proposed approach is validated with a power management applicative test-case. We demonstrate that, in this scenario, the 3D architecture reduces interconnect length and power by 35% and the maximal delay by 57%, compared to 2D. We also propose to use clusters either as input or output depending on current requirements. For instance, we propose to use a cluster storing energy that can be used either as input when a maximal affordable energy is input to the decision unit or as output to obtain an estimation of the energy consumed by the VFIs.

Conclusion and perspectives

Contribution and conclusion

The challenge of combining high-performance and energy efficiency is clearly present in today's applications. This objective can be reached with MPSoC platforms that provide high-level of adaptability, performance, reliability and energy efficiency. Nevertheless, they have to be accompanied with a decision unit that continuously adapts their operation. This leads us to neural networks that benefit from biological findings on the human brain and allow for an energy efficient associative memory. Associative memory replaces energy-costly optimization with energy-efficient memorization. Since no address is used to retrieve stored data, it allows to retrieve information based on different types of inputs (such as latency or energy constraint) depending on dynamic conditions.

We used an associative memory that relies on neural cliques to store information and sparse activity to retrieve it. This opens opportunities for low-power implementations. Networks of neural cliques are shown to surpass state-of-the-art associative memories yet it is indispensable to adapt them to real-world applications.

We therefore analyzed these networks in practical applications using real-world data obtained from measurements and simulations. We showed that in order to be applied to real-world applications, the initially proposed model of networks of neural cliques has to be improved. We proposed several methods to do so within which the most efficient relies on a concept of twin neurons. We analyzed twin neurons both formally and by simulations and we applied them to power and variability management, showing gains from the application point of view. We then proposed a specific neural cliques' architecture for hardware MPSoC power management decision unit. We compared our proposal to other decision units as game theory-based one or CAM-SRAM associative

memory. Compared to game theory, we showed that our decision unit consumes 6800 less energy and reacts 4500 times faster. It also offers higher energy savings (60% compared to 38%) due to usage of advanced, offline optimization methods and better reactivity. Compared to CAM-SRAM associative memory, we showed that our decision unit is less complex and consumes 48% less energy. We further proposed techniques to improve the reliability of the used neural cliques' circuits and we estimated the cost of programming the synapses of the network. Finally, we proposed a novel 3D interconnect approach for high-performance neural cliques' implementation. We showed that we obtain up to 55% reduction of total interconnect length and interconnect power consumption, and 74% reduction of the maximal interconnect delay.

The goal of providing a power management decision unit that combines the following:

- decision speed matching fast switching capabilities of DVFS actuators,
- decision close to optimal,
- energy efficiency

was met.

However, the contribution of this work is more general. The methods proposed for adapting neural cliques to real-world applications are appropriate for other applications than power management. Moreover, several strategies were proposed in Chapter 3, each having its advantages, which allows choosing the one suiting best a particular application. We also proposed an energy efficient associative memory that is an interesting alternative to typically used CAM designs.

Figure 4.21 depicts the full structure of the document together with the contribution positioned within its structure.

Perspectives

The work presented herein opens a great number of perspectives both on implementation and application levels. The latter includes research applications but also applications that are commonly present in industry and could benefit from neural cliques and the findings of this work.

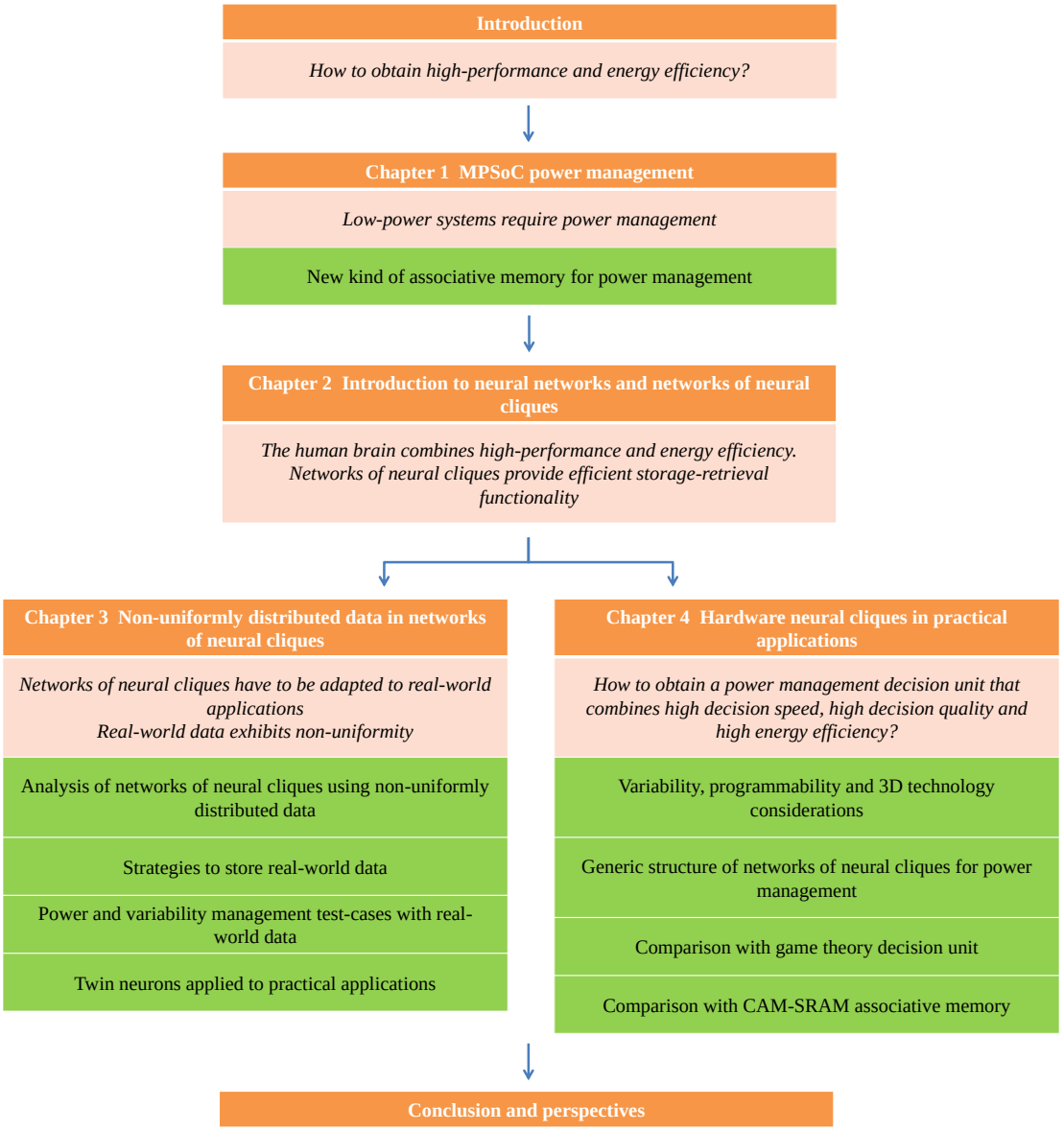


FIGURE 4.21: The structure of the document with contribution. The green boxes represent the contribution of this work and position it within the structure of the document.

Implementation

Several perspectives are open in terms of networks of neural cliques' implementation.

Analog circuits could be implemented in FDSOI technology to overcome their limitations. FDSOI allows for leakage reduction thanks to the buried oxide addition. This could increase the maximal number of synapses that can be connected to one fanal to avoid self-activation and consequently, improve the scalability of the hardware implementation. Further, with FDSOI technology, lower supply voltage is used which further

reduces the energy consumption. Additionally, by using body biasing one can envision dynamically adapting the circuit to trade speed for energy consumption according to the current requirements. In FDSOI, dopant usage is greatly reduced thus limiting the process variability. The characteristic of each transistor is closer to the average avoiding mismatch issues.

A mixed analog-digital implementation should be explored. Analog circuits for neural cliques have a number of advantages such as facilitated summation of signals through current summation or parallel WTA execution. However, mixed design would allow for better system integration and scalability which is limited in analog circuits by leakage currents.

Non-volatile (NV) feature is of great interest in low-power applications, for instance, energy-autonomous sensing nodes. Programming the synapses by using non-volatile memory, or even replacing synapses with such a memory, is one of the possibilities to be explored. We already have low-power neural cliques, adding NV, allowing to store their state without supply, would open a way to exploiting them in new IoT applications.

Miniaturization leads to lower voltages in circuits designed for modern technologies. This decreases the power consumption but also decreases the resolution in voltage domain. At the same time, this miniaturization allows for higher frequencies which increases the resolution in time domain. Consequently, it is interesting to search for time-based circuits instead of voltage-based ones. In terms of neural cliques' implementation, there is a link to be done with time-based circuits for SNNs.

Additionally, the asynchronous nature of activity in networks of neural cliques and their tolerance to unreliability of the used technology (Leduc-Primeau et al., 2014) encourages us to implement them with an asynchronous circuit exploiting approximate computing techniques (Han and Orshansky, 2013). The characteristics of neural cliques and these two design techniques seem to be extremely correlated and could result in a completely new level of performance.

Finally, the study of 3D neural cliques presented in Subsection 4.3 shows large gains and motivates us to expand it on more than two stacked dies (Solomon, 2012, Pavlidis et al., 2008).

Applications

Similarly, many perspectives are open in terms of applications of networks of neural cliques.

Physical Unclonable Functions (PUFs) emerge as a dominant security method for identification and anti-counterfeiting measures (Suh and Devadas, 2007). A classical PUF relies on exploiting differences in delays of paths with the same layout lengths. Several approaches are proposed but the principle idea is that a set of challenge-response pairs creates a unique signature of each circuit due to variability. We could take advantage of large storage capacity of networks of neural cliques and take advantage of variability to obtain a strong PUF. Moreover, there is no need to divide the messages into challenge-response pairs since any segment can be used as input or output to neural cliques. This creates even stronger circuit signature. Neural cliques-based PUF seems to be a very interesting area to explore.

Several companies, Analog Computing Solutions among others, propose a signal processing close to sensor for low-power sensor nodes. Such a processing allows to identify type of data that requires special treatment that implies waking up additional blocks. It is important to wake up these blocks only if it is really necessary, to save as much battery power as possible. Some of the solutions use neural networks to do this preprocessing. Neural cliques, apart their energy efficiency resulting from activating only the fanals implied in the retrieval procedure, can bring tolerance to noise. This is very important in close-to-sensor processing where data is inherently noisy. Additionally, a small part of data can be sufficient to do the necessary processing.

A whole group of applications where storage and error correction are necessary is open to neural cliques. This is because the clique is a codeword of a good error-correcting code. We envision here such applications as Field-Programmable Gate Array (FPGA) configuration memory protection for radiation immunity and, more generally, radiation immune processing/storage, reliable memory made of unreliable materials, error-correcting capabilities for embedded memories.

In the implementation perspectives we mention designing circuits for neural cliques with approximate computing principles. We can turn it the other way round: do approximate computing with neural cliques. When assessing neural cliques' performance we often talk

about error rates. This is the principle of approximate computing: tolerate loss of quality or optimality, accept approximations where it does not degrade results severely - live with errors. Recently, a new concept known as speculative computing emerges. The core idea is to anticipate and perform some tasks that may not be actually needed. This allows for faster result when it is needed, otherwise the obtained result is discarded. Neural cliques could be used for such tasks. Thanks to their fast time response, the decision on doing speculative computing can be postponed in time, increasing the probability that the obtained result is relevant for the application.

Neural cliques can be also seen as a multi-purpose accelerator that replaces time- and energy-consuming computations. Moreover, it can be used in many ways. Any segment can be used as input or output. We can therefore, obtain results based on some arguments but also obtain the input arguments when the result is known which may be of interest in some applications.

Very interesting recent works combine learning and remembering. Companies such as Facebook or Google work on architectures where there is a neural network on one side and a separate module on the other side that is used as memory. As discussed in Subsection 2.3.4, currently the most powerful technique for learning is deep learning. This type of neural network is augmented with an associative memory, just as neural cliques. Deep learning augmented with associative memories is used for Natural Language Processing (NLP) and works very well for question answering and language translation. Facebook AI Research already published Memory Networks (Weston et al., 2015). A similar concept called Neural Turing Machines is proposed by Google DeepMind (Graves et al., 2014). Since neural cliques outperform state-of-the-art associative memories, their application in deep learning augmented with associative memories should definitely be explored.

Appendix A

Process variability in neural cliques analog circuits

Electronic circuits are subject to PVT variations that cause variance in they performance. This is particularly true for analog circuits due to, for instance, mismatch which is the differential performance of two or more devices on a single integrated circuit Drennan and McAndrew (2003). This is also the case for the implementation used in this work. It is shown in Larras et al. (2013a) that for the network presented in Subsection 4.2.3, the time response of the analog circuit changes due to PVT variations. The results given in Table 4.1 are obtained for typical conditions, *i.e.* $V_{dd} = 1$V and $T_{emp} = 27$°C. The same simulations are carried out with voltage and temperature variations. For $V_{dd} = 0.9$V and $T_{emp} = 80$°C (slow corner), the time response is 53ns, whereas for $V_{dd} = 1.1$V and $T_{emp} = 0$°C (fast corner), the time response is 64ns. Still, both results are in favor of the analog circuit (the time response equals 208ns for the digital circuit).

Nevertheless, PVT variations in analog circuits can lead to incorrect results of the message retrieval procedure. According to our simulations, the most sensible part of analog circuit implementing neural cliques is the current mirror M_1^S, M_2^S in the synapse circuit (Figure 4.1). The mismatch between these two transistors impacts the current that is output by the synapse and the signal is not propagated correctly in the network. To harden these circuits few methods can be applied. Firstly, the unitary current I_{UNIT} can be increased to reduce the impact of the value added or substracted due to the variation. Secondly, the sizes of the transistors M_1^S, M_2^S can be increased to reduce the impact of

differences between their sizes. These methods increase the reliability yet they have a certain impact on power consumption and area respectively. Monte Carlo simulations show also that in some cases, due to variation, fanal fails to activate through an initial stimulation. This means that from the very beginning of the retrieval process, a known segment of message is not input to the network. Consequently, it is less likely to correctly retrieve the missing segments. To avoid such a case, the external stimulation should be done at the fanal's output instead of its input. Instead of applying high voltage state to an external synapse, high voltage state is directly applied to fanal's output.

Appendix B

Programming the synapses

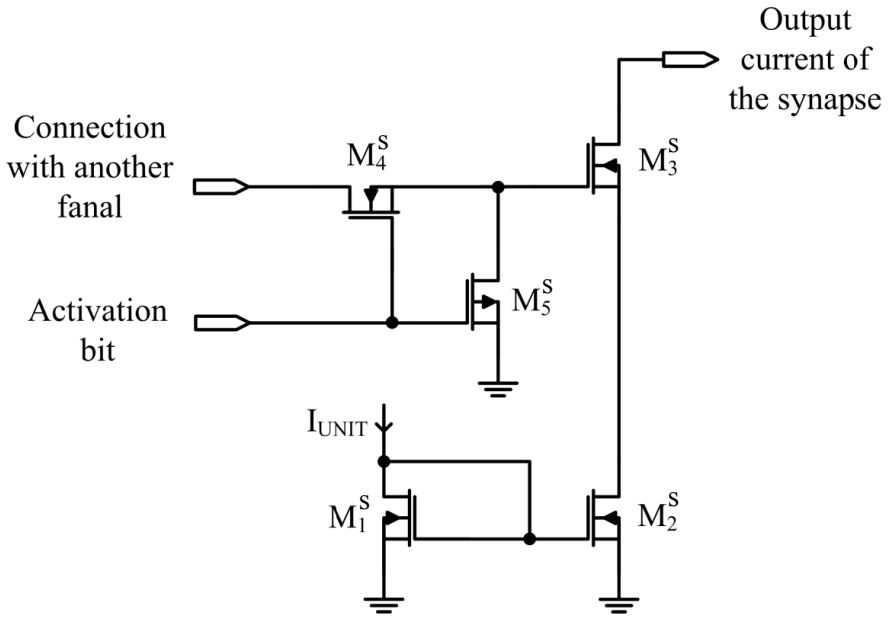


FIGURE B.1: Schematic of the programmable synapse circuit.

To allow programming the synapse two transistors are added to the initial synapse circuit shown in Figure 4.1. A programmable synapse circuit is presented in Figure B.1. If the activation bit is high, M_4^S conducts and M_5^S does not conduct. This results in transmission of the signal from another fanal (either low or high signal). If the activation bit is low, M_5^S conducts and no signal is transmitted through the synapse.

The activation bits for the fanals have to be stored in a memory. One bit in memory is sufficient to store an activation bit for a connection between two fanals. The cost of

TABLE B.1: SRAM memory for connection activation bits. Results estimated based on Do et al. (2011b). The area is estimated based on memory cell area with no control circuitry.

	Conventional Do et al. (2011b)	Proposed in Do et al. (2011b)
Energy/bit/read [fJ]	0.39	0.12
Surface area/bit [μm^2]	0.776	0.834
Read delay [ns]	2	1.43
Supply (SRAM)	1V	1V

storing connections activation bits can be estimated based on references on SRAM memories for the same technology. Two memory designs for 65nm technology are presented in Do et al. (2011b). Table B.1 presents the result obtained based on that reference. Two designs are included, a conventional and an improved one. Depending on design objectives, area can be traded for energy consumption and delay.

List of Publications

- Benoit Larras, Bartosz Boguslawski, Cyril Lahuec, Matthieu Arzel, Fabrice Seguin, and Frédéric Heitzmann. Analog encoded neural network for power management in MPSoC. In *Proceedings of the 11th International IEEE New Circuits and Systems Conference (NEWCAS)*, pages 1–4, Paris, France, 2013. Second Best Student Paper Award. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6573589.

- Bartosz Boguslawski, Vincent Gripon, Fabrice Seguin, and Frédéric Heitzmann. Huffman coding for storing non-uniformly distributed messages in networks of neural cliques. In *Proceedings of the Twenty-Eighth Conference on Artificial Intelligence*, pages 262–268, Quebec City, Canada, July 2014. http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/download/8160/8429.

- Benoit Larras, Bartosz Boguslawski, Cyril Lahuec, Matthieu Arzel, Fabrice Seguin, and Frédéric Heitzmann. Analog encoded neural network for power management in MPSoC. *Analog Integrated Circuits and Signal Processing Journal*, Springer, 81 (3):595–605, 2014.
http://link.springer.com/article/10.1007%2Fs10470-014-0420-z#page-1.

- Bartosz Boguslawski, Frédéric Heitzmann, Benoit Larras and Fabrice Seguin. Energy efficient associative memory based on neural cliques. In *Design Automation Conference (DAC) Work-in-Progress Session*, San Francisco, USA, 2015.

- Bartosz Boguslawski, Hossam Sarhan, Frédéric Heitzmann, Fabrice Seguin, Sebastien Thuries, Olivier Billoint, and Fabien Clermidy. Compact interconnect approach for networks of neural cliques using 3D technology. In *Proceedings of*

the IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), pages 116–121, Daejeon, South Korea, 2015. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7314402.

- Bartosz Boguslawski, Vincent Gripon, Fabrice Seguin, and Frédéric Heitzmann. Twin neurons for efficient real-world data distribution in networks of neural cliques. Applications in power management in electronic circuits. *IEEE Transactions on Neural Networks and Learning Systems, Special Issue on Neurodynamic Systems for Optimization and Applications*, PP(99):1–13 2015. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7302579.

- Bartosz Boguslawski, Frédéric Heitzmann, Benoit Larras and Fabrice Seguin. Energy efficient associative memory based on neural cliques. *IEEE Transactions on Circuits and Systems II Journal*, PP(99):1–5 2015. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7347365.

Bibliography

Vincent Gripon and Claude Berrou. Sparse neural networks with large learning diversity. *IEEE Transactions on Neural Networks*, 22(7):1087–1096, July 2011a. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5784337&tag=1.

Benoît Larras, Cyril Lahuec, Matthieu Arzel, and Fabrice Seguin. Analog implementation of encoded neural networks. In *ISCAS 2013 : IEEE International Symposium on Circuits and Systems*, pages 1612 – 1615, 2013a. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6572170.

David Bol, Julien De Vos, Cédric Hocquet, François Botman, François Durvaux, Sarah Boyd, Denis Flandre, and Jean-Didier Legat. Sleepwalker: A 25-MHz 0.4-V sub-mm^2 7-μW/MHz microcontroller in 65-nm LP/GP CMOS for low-carbon wireless sensor nodes. *J. Solid-State Circuits*, 48(1):20–32, 2013. URL http://dx.doi.org/10.1109/JSSC.2012.2218067.

Shubo Qi, Minxuan Zhang, Jinwen Li, Tianlei Zhao, Chengyi Zhang, and Shaoqing Li. A high performance router with dynamic buffer allocation for on-chip interconnect networks. In *28th International Conference on Computer Design, ICCD 2010, 3-6 October 2010, Amsterdam, The Netherlands, Proceedings*, pages 462–467, 2010. URL http://dx.doi.org/10.1109/ICCD.2010.5647657.

Hiroki Noguchi, Kazutaka Ikegami, Keiichi Kushida, Keiko Abe, Shogo Itai, Satoshi Takaya, Naoharu Shimomura, Junichi Ito, Atsushi Kawasumi, Hiroyuki Hara, and Shinobu Fujita. 7.5 A 3.3ns-access-time 71.2μW/MHz 1Mb embedded STT-MRAM using physically eliminated read-disturb scheme and normally-off memory architecture. In *2015 IEEE International Solid-State Circuits Conference, ISSCC 2015, Digest of Technical Papers, San Francisco, CA, USA, February 22-26, 2015*, pages 1–3, 2015. URL http://dx.doi.org/10.1109/ISSCC.2015.7062963.

Huang-Chih Kuo, Jian-Wen Chen, and Youn-Long Lin. A high-performance low-power H.264/AVC video decoder accelerator for embedded systems. In Andy D. Pimentel and Naehyuck Chang, editors, *ESTImedia*, pages 1–8. IEEE, 2009. ISBN 978-1-4244-5170-8. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5336823.

Xiyu Lu, Xinlei Chen, Yong Li, Depeng Jin, Lieguang Zeng, and Habib F. Rashvand. Zebraban: a heterogeneous high-performance energy efficient wireless body sensor network. *IET Wireless Sensor Systems*, 3(4):247–254, 2013. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6655805.

A. Boukhayma, Jean-Pierre Rostaing, A. Mollard, Fabrice Guellec, Michele Benetti, G. Ducournau, J.-F. Lampin, Antoine Dupret, C. Enz, Michaël Tchagaspanian, and J.-A. Nicolas. A 533pW NEP 31x31 pixel THz image sensor based on in-pixel demodulation. In *ESSCIRC 2014 - 40th European Solid State Circuits Conference, Venice Lido, Italy, September 22-26, 2014*, pages 303–306, 2014. URL http://dx.doi.org/10.1109/ESSCIRC.2014.6942082.

Daniel Drubach. *The Brain Explained*. Prentice Hall, 1999.

Biswa Sengupta and Martin B. Stemmler. Power consumption during neuronal computation. *Proceedings of the IEEE*, 102(5):738–750, 2014. URL http://dx.doi.org/10.1109/JPROC.2014.2307755.

Brian Whitworth. Some implications of comparing brain and computer processing. In *HICSS*, page 38. IEEE Computer Society, 2008. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4438742&tag=1.

Raúl Rojas. *Neural Networks: A Systematic Introduction*. Springer-Verlag New York, Inc., New York, NY, USA, 1996. ISBN 3-540-60505-3. URL http://page.mi.fu-berlin.de/rojas/neural/neuron.pdf.

Shekhar Borkar. Designing reliable systems from unreliable components: The challenges of transistor variability and degradation. *IEEE Micro*, 25(6):10–16, November 2005. ISSN 0272-1732. URL http://dx.doi.org/10.1109/MM.2005.110.

W. Wolf, A. A. Jerraya, and G. Martin. Multiprocessor system-on-chip (mpsoc) technology. *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, 27(10):1701–1713, October 2008. ISSN 0278-0070. URL http://dx.doi.org/10.1109/TCAD.2008.923415.

Camille Jalier, Didier Lattard, Ahmed Amine Jerraya, Gilles Sassatelli, Pascal Benoit, and Lionel Torres. Heterogeneous vs homogeneous MPSoC approaches for a mobile LTE modem. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '10, pages 184–189, 3001 Leuven, Belgium, Belgium, 2010. European Design and Automation Association. ISBN 978-3-9810801-6-2. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5457213.

Sergio Saponara and Luca Fanucci. Homogeneous and heterogeneous mpsoc architectures with network-on-chip connectivity for low-power and real-time multimedia signal processing. *VLSI Design*, 2012, 2012. URL http://www.hindawi.com/journals/vlsi/2012/450302/.

Wen-Chung Tsai, Ying-Cherng Lan, Yu Hen Hu, and Sao-Jie Chen. Networks on chips: Structure and design methodologies. *J. Electrical and Computer Engineering*, 2012, 2012. URL http://www.hindawi.com/journals/jece/2012/509465/.

Shasi Kumar, Axel Jantsch, Juha-Pekka Soininen, Martti Forsell, Mikael Millberg, Johny Öberg, Kari Tiensyrjä, and Ahmed Hemani. A network on chip architecture and design methodology. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, ISVLSI '02, pages 105–112, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1486-3. URL http://ieeexplore.ieee.org/xpl/abstractAuthors.jsp?arnumber=1016885&tag=1.

Edith Beigné, Fabien Clermidy, Pascal Vivet, Alain Clouard, and Marc Renaudin. An asynchronous noc architecture providing low latency service and its multi-level design framework. In *ASYNC*, pages 54–63. IEEE Computer Society, 2005. ISBN 0-7695-2305-6. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1402046.

Wonyoung Kim, Meeta Sharma Gupta, Gu-Yeon Wei, and David Brooks. System level analysis of fast, per-core DVFS using on-chip switching regulators. In *HPCA*, pages 123–134. IEEE Computer Society, 2008. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4658633&tag=1.

D.N. Truong, W.H. Cheng, T. Mohsenin, Zhiyi Yu, A.T. Jacobson, G. Landge, M.J. Meeuwsen, C. Watnik, A.T. Tran, Zhibin Xiao, E.W. Work, J.W. Webb, P.V. Mejia,

and B.M. Baas. A 167-processor computational platform in 65 nm CMOS. *Solid-State Circuits, IEEE Journal of*, 44(4):1130–1144, April 2009. ISSN 0018-9200. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4804961.

E. Beigné, F. Clermidy, H. Lhermet, S. Miermont, Y. Thonnart, X. Tran, A. Valentian, D. Varreau, P. Vivet, X. Popon, and H. Lebreton. An asynchronous power aware and adaptive NoC based circuit. *Solid-State Circuits, IEEE Journal of*, 44(4):1167–1177, 2009. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4804962.

Eitan N. Shauly. CMOS leakage and power reduction in transistors and circuits: Process and layout considerations. *Journal of Low Power Electronics and Applications*, 2(1): 1–29, 2012. ISSN 2079-9268. URL http://www.mdpi.com/2079-9268/2/1/1.

Eiji Morifuji, Takeshi Yoshida, Masahiko Kanda, Satoshi Matsuda, Seiji Yamada, and Fumitomo Matsuoka. Supply and threshold-Voltage trends for scaled logic and SRAM MOSFETs. *Electron Devices, IEEE Transactions on*, 53(6):1427–1432, June 2006. ISSN 0018-9383. URL http://dx.doi.org/10.1109/ted.2006.874752.

Sasmita Deo. Power consumption calculation of AP-DCD algorithm using FPGA platform. In *ReConFig*, pages 388–393. IEEE Computer Society, 2010. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5695337.

Steven A. Vitale, Peter W. Wyatt, Nisha Checka, Jakub Kedzierski, and Craig L. Keast. FDSOI process technology for subthreshold-operation ultralow-power electronics. *Proceedings of the IEEE*, 98(2):333–342, 2010. URL http://ieeexplore.ieee.org/xpl/abstractAuthors.jsp?arnumber=5395759.

Gabriel Marchesan Almeida, Rémi Busseuil, Luciano Ost, Florent Bruguier, Gilles Sassatelli, Pascal Benoit, Lionel Torres, and Michel Robert. PI and PID regulation approaches for performance-constrained adaptive multiprocessor system-on-chip. *Embedded Systems Letters*, 3(3):77–80, 2011. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6008624&tag=1.

Sergey Zhuravlev, Juan Carlos Saez, Sergey Blagodurov, Alexandra Fedorova, and Manuel Prieto. Survey of energy-cognizant scheduling techniques. *IEEE Trans. Parallel Distrib. Syst.*, 24(7):1447–1464, 2013. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6127864&tag=1.

Zeynep Toprak-Deniz, Yusuf Leblebici, and Eric Vittoz. On-line global energy optimization in multi-core systems using principles of analog computation. *IEEE Journal of Solid-State Circuits*, 42(7):1593–1606, July 2007. ISSN 0018-9200. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4261002.

Imen Mansouri, Fabien Clermidy, Pascal Benoit, and Lionel Torres. A run-time distributed cooperative approach to optimize power consumption in mpsocs. In *SoCC*, pages 25–30. IEEE, 2010a. ISBN 978-1-4244-6682-5. URL http://ieeexplore.ieee.org/xpl/abstractCitations.jsp?arnumber=5784664&tag=1.

Diego Puschini, Fabien Clermidy, Pascal Benoit, Gilles Sassatelli, and Lionel Torres. Dynamic and distributed frequency assignment for energy and latency constrained MP-SoC. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '09, pages 1564–1567, Leuven, Belgium, 2009. European Design and Automation Association. ISBN 978-3-9810801-5-5. URL http://ieeexplore.ieee.org/xpl/abstractCitations.jsp?arnumber=5090912.

Diego Puschini, Fabien Clermidy, Pascal Benoit, Gilles Sassatelli, and Lionel Torres. A game-theoretic approach for run-time distributed optimization on MP-SoC. *Int. J. Reconfig. Comp.*, 2008, 2008. doi: 10.1155/2008/403086. URL http://dx.doi.org/10.1155/2008/403086.

Imen Mansouri, Camille Jalier, Fabien Clermidy, Pascal Benoit, and Lionel Torres. Implementation analysis of a dynamic energy management approach inspired by game-theory. In *IEEE Computer Society Annual Symposium on VLSI, ISVLSI*, pages 422–427, 2010b. URL http://dx.doi.org/10.1109/ISVLSI.2010.61.

Jitesh Shinde and S.S. Salankar. Clock gating - a power optimizing technique for VLSI circuits. In *Annual IEEE India Conference (INDICON)*, pages 1–4, 2011. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6139440.

John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944. ISBN 0691119937.

J.F. Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951. URL http://www.cs.upc.edu/~ia/nash51.pdf.

Chyuan Shiun Lin, Diane C. P. Smith, and John Miles Smith. The design of a rotating associative memory for relational database applications. *ACM Trans. Database Syst.*,

1(1):53–65, March 1976. ISSN 0362-5915. URL http://dl.acm.org/citation.cfm?id=320447.

Antonis Papadogiannakis, Michalis Polychronakis, and Evangelos P. Markatos. Improving the accuracy of network intrusion detection systems under load using selective packet discarding. In *Proceedings of the Third European Workshop on System Security*, EUROSEC '10, pages 15–21, New York, NY, USA, 2010. ISBN 978-1-4503-0059-9. URL https://www.ics.forth.gr/dcs/Activities/papers/discarding.eurosec10.pdf.

Norman P. Jouppi. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. In *Proceedings of the 17th annual international symposium on Computer Architecture*, ISCA '90, pages 364–373, New York, NY, USA, 1990. ISBN 0-89791-366-3. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=134547.

N.F. Huang, W.E. Chen, C.Y. Lou, and J.M. Chen. Design of multi-field IPv6 packet classifiers using ternary CAMs. In *Proc. IEEE GLOBECOM*, volume 3, pages 1877–1881, 2001. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=965900.

Kostas Pagiamtzis and Ali Sheikholeslami. Content-addressable memory (CAM) circuits and architectures: A tutorial and survey. *IEEE Journal of Solid-State Circuits*, 41(3): 712–727, March 2006. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1599540.

Banit Agrawal and Timothy Sherwood. Modeling TCAM power for next generation network devices. In *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 120–129, Austin, TX, March 2006. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1620796&tag=1.

A.G. Brown. *Nerve Cells and Nervous Systems. An Introduction to Neuroscience*. Springer-Verlag London, 2001. URL http://www.springer.com/us/book/9783540760900.

Bente Pakkenberg, Dorte Pelvig, Lisbeth Marner, Mads J. Bundgaard, Hans Jørgen G. Gundersen, Jens R. Nyengaard, and Lisbeth Regeur. Aging and the human neocortex. *Experimental Gerontology*, 38(1-2):95 – 99, 2003. ISSN 0531-5565. URL http://

www.sciencedirect.com/science/article/pii/S0531556502001511. Proceedings of the 6th International Symposium on the Neurobiology and Neuroendocrinology of Aging.

Christopher Johansson and Anders Lansner. Towards cortex sized artificial neural systems. *Neural Netw.*, 20(1):48–61, January 2007. ISSN 0893-6080. URL http://dx.doi.org/10.1016/j.neunet.2006.05.029.

Günther Palm. *Neural Assemblies, an Alternative Approach to Artificial Intelligence*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982. ISBN 0387113665. URL http://www.springer.com/us/book/9783642817946.

Edmund T. Rolls and Alessandro Treves. *Neural Networks and Brain Function*. Oxford University Press, New York, USA, 1997. ISBN 9780198524328. URL http://www.oxfordscholarship.com/view/10.1093/acprof:oso/9780198524328.001.0001/acprof-9780198524328.

Edward. G. Jones. Microcolumns in the cerebral cortex. *Proc. National Academy of Sciences (PNAS)*, 10(97):5019–5021, 2010. URL http://www.pnas.org/content/97/10/5019.full.

Alex M. Thomson and A. Peter Bannister. Interlaminar connections in the neocortex. *Cerebral Cortex*, 13(1):5–14, January 2003. URL http://cercor.oxfordjournals.org/content/13/1/5.long.

Walter H. Pitts and Warren S. McCulloch. How we know universals: The perception of auditory and visual forms. *Bulletin of Mathematical Biophysics*, 9:127–147, 1947. URL http://link.springer.com/article/10.1007%2FBF02478291.

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Nat. Acad. Sci*, 79(8):2554–2558, April 1982. URL http://cns.upf.edu/jclub/hopfield82.pdf.

Wulfram Gerstner and Werner M. Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002. URL http://icwww.epfl.ch/~gerstner/BUCH.html.

William Bialek, Fred Rieke, Rob R. de Ruytervan Steveninck, and David Warland. Reading a neural code. *Science*, 252(5014):1854–1857, 1991. URL http://www.ncbi.nlm.nih.gov/pubmed/2063199.

W. Heiligenberg. *Neural Nets in Electric Fish*. MIT Press, 1991. URL https://mitpress.mit.edu/index.php?q=books/neural-nets-electric-fish.

N. Kuwabara and N. Suga. Delay lines and amplitude selectivity are created in subthalamic auditory nuclei: the brachium of the inferior colliculus of the mustached bat. *Journal of Neurophysiology*, 69(5):1713–1724, 1993. URL http://jn.physiology.org/content/69/5/1713.

A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117 (4):500–544, August 1952. ISSN 0022-3751. URL http://jp.physoc.org/content/117/4/500.abstract.

L. Lapicque. Recherches quantitatives sur l'excitation electrique des nerfs traitée comme une polarisation. *Journal de Physiologie et Pathologie General*, 9:620–635, 1907. URL http://homepages.inf.ed.ac.uk/mvanross/reprints/lapicque_trans.pdf.

R. B. Stein. A Theoretical analysis of Neuronal Variability. *Biophysical Journal*, 5:173–194, March 1965. ISSN 0006-3495. URL http://view.ncbi.nlm.nih.gov/pubmed/14268952.

Eugene M. Izhikevich. Simple model of spiking neurons. *IEEE Trans. Neural Networks*, pages 1569–1572, 2003. URL http://www.izhikevich.org/publications/spikes.pdf.

Sen Song, Kenneth D. Miller, and L. F. Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3(9), 2000. URL http://www.nature.com/neuro/journal/v3/n9/pdf/nn0900_919.pdf.

Hélène Paugam-Moisy and Sander M. Bohte. *Handbook of Natural Computing*, chapter Computing with Spiking Neuron Networks, pages 1–47. Springer-Verlag, September 2009. URL http://homepages.cwi.nl/~sbohte/publication/paugam_moisy_bohte_SNNChapter.pdf.

Jilles Vreeken. Spiking neural networks, an introduction, 2003. URL https://people.mmci.uni-saarland.de/~jilles/pubs/2002/spiking_neural_networks_an_introduction-vreeken.pdf.

Filip Ponulak and Andrzej Kasinski. Introduction to spiking neural networks: Information processing, learning and applications. *Acta Neurobiologiae Experimentalis*, 71(4), 2011. URL http://www.ane.pl/pdf/7146.pdf.

André Gröning and Sander M. Bohte. Spiking neural networks: Principles and challenges. In *Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 1–10, 2014. URL http://homepages.cwi.nl/~sbohte/publication/es2014-13Gruning.pdf.

Bilel Belhadj, Alexandre Valentian, Pascal Vivet, Marc Duranton, Liqiang He, and Olivier Temam. The improbable but highly appropriate marriage of 3d stacking and neuromorphic accelerators. In *2014 International Conference on Compilers, Architecture and Synthesis for Embedded Systems, CASES 2014, Uttar Pradesh, India, October 12-17, 2014*, pages 1:1–1:9, 2014a. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6972454.

Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009. ISSN 1935-8237. URL http://dx.doi.org/10.1561/2200000006.

Thomas Serre, Gabriel Kreiman, Minjoon Kouh, Charles Cadieu, Ulf Knoblich, and Tomaso Poggio. A quantitative theory of immediate visual recognition. *Progress in Brain Research*, pages 33–56, 2007. URL http://www.sciencedirect.com/science/article/pii/S0079612306650048.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006. ISSN 0899-7667. URL http://dx.doi.org/10.1162/neco.2006.18.7.1527.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, Université De Montréal, and Montréal Québec. Greedy layer-wise training of deep networks. In *In NIPS*. MIT Press, 2007. URL http://papers.nips.cc/paper/3048-greedy-layer-wise-training-of-deep-networks.pdf.

Marc'Aurelio Ranzato, Y-Lan Boureau, Sumit Chopra, and Yann LeCun. A unified energy-based framework for unsupervised learning. In Marina Meila and Xiaotong Shen, editors, *AISTATS*, volume 2 of *JMLR Proceedings*, pages 371–379. JMLR.org, 2007. URL http://yann.lecun.com/exdb/publis/pdf/ranzato-unsup-07.pdf.

Jason Weston and Frédéric Ratle. Deep learning via semi-supervised embedding. In *International Conference on Machine Learning*, 2008. URL http://link.springer.com/chapter/10.1007%2F978-3-642-35289-8_34#page-1.

Hossein Mobahi, Ronan Collobert, and Jason Weston. Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 737–744, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.148.5771.

Vincent Gripon and Michael Rabbat. Maximum likelihood associative memories. In *Proceedings of Information Theory Workshop*, pages 1–5, September 2013. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6691310.

C. Berrou and V. Gripon. Coded hopfield networks. In *Turbo Codes and Iterative Information Processing (ISTC), 2010 6th International Symposium on*, pages 1–5, Sept 2010. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5613860.

D. J. Willshaw, O. P. Buneman, and Longuet H. C. Higgins. Non-holographic associative memory. *Nature*, 222:960–962, 1969. URL http://www.nature.com/nature/journal/v222/n5197/abs/222960a0.html.

Günther Palm. Neural associative memories and sparse coding. *Neural Netw.*, 37: 165–171, January 2013. ISSN 0893-6080. URL http://www.sciencedirect.com/science/article/pii/S0893608012002298.

Amir Hesam Salavati and Amin Karbasi. Multi-level error-resilient neural networks. In *ISIT*, pages 1064–1068. IEEE, 2012. ISBN 978-1-4673-2580-6. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6283014.

Vincent Gripon and Claude Berrou. A simple and efficient way to store many messages using neural cliques. In *Proceedings of IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain*, pages 54–58, Paris, France, April

2011b. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5952106.

Behrooz Kamary Aliabadi, Claude Berrou, Vincent Gripon, and Xiaoran Jiang. Storing sparse messages in networks of neural cliques. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):980 – 989, 2013. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6658945.

Longnian Lin, Remus Osan, and Joe Z. Tsien. Organizing principles of real-time memory encoding: Neural clique assemblies and universal neural codes. *Trends Neurosci.*, 29 (1):48–57, January 2006. URL http://www.sciencedirect.com/science/article/pii/S0166223605003012.

Vincent Gripon and Claude Berrou. Nearly-optimal associative memories based on distributed constant weight codes. In *Proceedings of Information Theory and Applications Workshop*, pages 269–273, San Diego, CA, USA, February 2012. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6181790.

D. K. Lee, L. Itti, C. Koch, and J. Braun. Attention activates winner-take-all competition among visual filters. *Nature Neuroscience*, 2(4):375–81, Apr 1999. URL http://www.nature.com/neuro/journal/v2/n4/full/nn0499_375.html.

Robert Coultrip, Richard Granger, and Gary Lynch. Original contribution: A cortical model of winner-take-all competition via lateral inhibition. *Neural Netw.*, 5(1):47–54, January 1992. ISSN 0893-6080. URL http://www.sciencedirect.com/science/article/pii/S0893608005800061.

Ala Aboudib, Vincent Gripon, and Xiaoran Jiang. A study of retrieval algorithms of sparse messages in networks of neural cliques. In *COGNITIVE'14*, pages 140 – 146, 2014. URL http://www.thinkmind.org/download.php?articleid=cognitive_2014_6_30_40094.

D. Attwell and S. B. Laughlin. An energy budget for signaling in the grey matter of the brain. *J Cereb Blood Flow Metab*, 21(10):1133–45, 2001. URL http://www.nature.com/jcbfm/journal/v21/n10/full/9591146a.html.

Peter Lennie. The cost of cortical computation. *Current Biology*, 13(6): 493–497, 2003. URL http://www.sciencedirect.com/science/article/pii/S0960982203001350.

Andreas Knoblauch, Günther Palm, and Friedrich T. Sommer. Memory capacities for synaptic and structural plasticity. *Neural Comput.*, 22(2):289–341, February 2010. ISSN 0899-7667. URL http://www.mitpressjournals.org/doi/abs/10.1162/neco.2009.08-07-588#.VXVLV7yUONM.

Bartosz Boguslawski, Vincent Gripon, Fabrice Seguin, and Frédéric Heitzmann. Huffman coding for storing non-uniformly distributed messages in networks of neural cliques. In *Proceedings of the Twenty-Eighth Conference on Artificial Intelligence*, pages 262–268, Quebec City, Canada, July 2014. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/download/8160/8429.

Bartosz Boguslawski, Vincent Gripon, Fabrice Seguin, and Frédéric Heitzmann. Twin neurons for efficient real-world data distribution in networks of neural cliques. Applications in power management in electronic circuits. *IEEE Transactions on Neural Networks and Learning Systems, Special Issue on Neurodynamic Systems for Optimization and Applications*, PP(99):1–13, 2015a. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7302579.

David A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the Institute of Radio Engineers*, 40(9):1098–1101, September 1952. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4051119.

A. Bookstein and S.T. Klein. Is Huffman coding dead? *Computing*, 50:279–296, 1993. URL http://link.springer.com/article/10.1007%2FBF02243872.

Jeffrey Scott Vitter. Design and analysis of dynamic Huffman codes. *J. ACM*, 34(4):825–845, October 1987. ISSN 0004-5411. doi: 10.1145/31846.42227. URL http://doi.acm.org/10.1145/31846.42227.

Fabien Clermidy, Romain Lemaire, Xavier Popon, Dimitri Ktenas, and Yvain Thonnart. An open and reconfigurable platform for 4G telecommunication: Concepts and application. In *DSD*, pages 449–456. IEEE Computer Society, 2009. ISBN 978-0-7695-3782-5. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5350210.

Fabien Clermidy, Christian Bernard, Romain Lemaire, Jérôme Martin, Ivan Miro Panades, Yvain Thonnart, Pascal Vivet, and Norbert Wehn. A 477mW NoC-based digital baseband for MIMO 4G SDR. In *ISSCC*, pages 278–279. IEEE, 2010. ISBN

978-1-4244-6033-5. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5433920.

L. Vincent, E. Beigné, L. Alacoque, S. Lesecq, C. Bour, and P. Maurine. A fully integrated 32 nm multiprobe for dynamic PVT measurements within complex digital SoC. In *2nd European Workshop on CMOS Variability, VARI'11*, Grenoble, France, 2011. URL https://hal.archives-ouvertes.fr/hal-01067989.

Lionel Vincent, Edith Beigné, Suzanne Lesecq, Julien Mottin, David Coriat, and Philippe Maurine. Dynamic variability monitoring using statistical tests for energy efficient adaptive architectures. *IEEE Trans. on Circuits and Systems*, 61-I(6): 1741–1754, 2014. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6728703.

Benoit Larras, Bartosz Boguslawski, Cyril Lahuec, Matthieu Arzel, Fabrice Seguin, and Frédéric Heitzmann. Analog encoded neural network for power management in MPSoC. *Analog Integrated Circuits and Signal Processing*, 81(3):595–605, 2014. ISSN 0925-1030. URL http://link.springer.com/article/10.1007%2Fs10470-014-0420-z#page-1.

Bartosz Boguslawski, Hossam Sarhan, Frédéric Heitzmann, Fabrice Seguin, Sebastien Thuries, Olivier Billoint, and Fabien Clermidy. Compact interconnect approach for networks of neural cliques using 3D technology. In *Proc. IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 116–121, 2015b. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7314402.

Bilel Belhadj, Alexandre Valentian, Pascal Vivet, Marc Duranton, Liqiang He, and Olivier Temam. The improbable but highly appropriate marriage of 3D stacking and neuromorphic accelerators. In *Proc. CASES'14*, pages 1–9, 2014b. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6972454.

Fabien Clermidy et al. Advanced technologies for brain-inspired computing. In *Proc. ASP-DAC'14*, 2014. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6742951.

Antoine Joubert, Marc Duranton, Bilel Belhadj, Olivier Temam, and Rodolphe Heliot. Capacitance of TSVs in 3D stacked chips a problem? Not for neuromorphic systems.

In *Proc. DAC'12, WACI session*, pages 1260–1261, 2012. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6241670.

Anthony Gutierrez et al. Integrated 3D-stacked server designs for increasing physical density of key-value stores. In *Proc. ASPLOS*, pages 485–498. ACM, 2014. URL https://homes.cs.washington.edu/~luisceze/publications/3D-asplos_2014.pdf.

P. Batude et al. 3D sequential integration opportunities and technology optimization. In *IEEE IITC/AMC*, pages 373–376, May 2014. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6831837.

Benoit Larras, Bartosz Boguslawski, Cyril Lahuec, Matthieu Arzel, Fabrice Seguin, and Frédéric Heitzmann. Analog encoded neural network for power management in MP-SoC. In *Proceedings of the 11th International IEEE New Circuits and Systems Conference*, NEWCAS '13, pages 1–4, 2013b. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6573589.

D. Lattard, E. Beigne, C. Bernard, C. Bour, F. Clermidy, Y. Durand, J. Durupt, D. Varreau, P. Vivet, P. Penard, A. Bouttier, and F. Berens. A telecom baseband circuit based on an asynchronous network-on-chip. In *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, pages 258–601, Feb 2007. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4242363.

Bartosz Boguslawski, Frédéric Heitzmann, Benoit Larras, and Fabrice Seguin. Energy efficient associative memory based on neural cliques. *IEEE Transactions on Circuits and Systems II*, PP(99):1–5, 2015c. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7347365.

Bartosz Boguslawski, Frédéric Heitzmann, Benoit Larras, and Fabrice Seguin. Energy efficient associative memory based on neural cliques. In *Design Automation Conference (DAC) Work-in-Progress Session*, 2015d.

Anh-Tuan Do, Jeremy Yung Shern Low, Joshua Yung Lih Low, Zhi-Hui Kong, Xiaoliang Tan, and Kiat Seng Yeo. An 8T differential SRAM with improved noise margin for bit-interleaving in 65 nm CMOS. *IEEE Transactions on Circuits and Systems I:*

Regular Papers, 58-I(6):1252–1263, 2011a. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5701786.

Amit Agarwal, Steven K. Hsu, Himanshu Kaul, Mark A. Anders, and Ram K. Krishnamurthy. A dual-supply 4GHz 13fj/bit/search 64x128b CAM in 65nm CMOS. In *Proc. ESSCIRC'06*, pages 303–306, September 2006. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4099764.

Anh-Tuan Do, Shoushun Chen, Zhi-Hui Kong, and Kiat Seng Yeo. A high speed low power CAM with a parity bit and power-gated ML sensing. *IEEE Trans. VLSI Syst.*, 21(1):151–156, 2013. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6135529.

Igor Arsovski and Reid Wistort. Self-referenced sense amplifier for across-chip-variation immune sensing in high-performance content-addressable memories. In *Proc. CICC'06*, pages 453–456, September 2006. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4115000.

Anh-Tuan Do, Chun Yin, Kavitha Velayudhan, Zhao Chuan Lee, Kiat Seng Yeo, and Tony Tae-Hyoung Kim. 0.77 fj/bit/search content addressable memory using small match line swing and automated background checking scheme for variation tolerance. *IEEE Journal of Solid-State Circuits*, 49(7):1487–1498, 2014. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6805234.

F. Leduc-Primeau, V. Gripon, M.G. Rabbat, and W.J. Gross. Cluster-based associative memories built from unreliable storage. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 8370–8374, May 2014. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6855234.

Jie Han and M. Orshansky. Approximate computing: An emerging paradigm for energy-efficient design. In *Test Symposium (ETS), 2013 18th IEEE European*, pages 1–6, May 2013. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6569370.

N. Solomon. Three dimensional integrated circuits and methods of fabrication, March 13 2012. URL https://www.google.com/patents/US8136071. US Patent 8,136,071.

Vasilis F. Pavlidis, Ioannis Savidis, and Eby G. Friedman. Clock distribution networks for 3-D integrated circuits. In *Custom Integrated Circuits Conference, CICC*

2008. IEEE, pages 651–654. IEEE, 2008. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4672170.

G.E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, pages 9–14, June 2007. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4261134.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *International Conference on Learning Representations (ICLR)*, 2015. URL http://arxiv.org/abs/1410.3916.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing machines. *CoRR*, abs/1410.5401, 2014. URL http://arxiv.org/abs/1410.5401.

P.G. Drennan and C.C. McAndrew. Understanding MOSFET mismatch for analog design. *Solid-State Circuits, IEEE Journal of*, 38(3):450–456, Mar 2003. ISSN 0018-9200. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1183852.

Anh-Tuan Do, Jeremy Yung Shern Low, Joshua Yung Lih Low, Zhi-Hui Kong, Xiaoliang Tan, and Kiat Seng Yeo. An 8T differential SRAM with improved noise margin for bit-interleaving in 65 nm CMOS. *IEEE Trans. on Circuits and Systems*, 58-I(6): 1252–1263, 2011b. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5701786.

Résumé

La densité croissante de transistors fournit de plus en plus de puissance de calcul dans une seule puce. Les systèmes informatiques deviennent de plus en plus intégrés et plus attrayants pour de nouveaux domaines d'application, auparavant inaccessibles. Ils sont devenus omniprésents et doivent également être mobiles avec des contraintes fortes sur la consommation d'énergie.

Supporter une diversité d'applications tout en fournissant une autonomie suffisante nécessite que ces systèmes doivent être efficaces à la fois en termes de vitesse et d'énergie.

Ceci est important pour tous les composants d'un système, par exemple les micro-contrôleurs dans les nœuds de capteurs sans fil pour l'Internet des objets (IdO ou IoT pour Internet of Things) (Bol et al., 2013), les routeurs pour des réseaux sur puce (Qi et al., 2010), les mémoires (Noguchi et al., 2015), les accélérateurs matériels dédiés (Kuo et al., 2009), pour ne nommer que quelques-uns. Aujourd'hui, tous ces composants doivent être de haute performance et consommer le moins d'énergie possible.

Pour permettre la mise en œuvre de cette grande variété d'applications, de multiples composants dédiés sont intégrés sur une seule puce et interconnectés avec une infrastructure appropriée de communication. En fonction de l'application en cours d'exécution, différents composants (Processing Elements - PEs) sont utilisés pour fournir les performances nécessaires. Ce type d'architecture est appelé MultiProcessor System-on-Chip (MPSoC).

Une certaine flexibilité est donnée à chaque PE, permettant de le ralentir ou de l'accélérer en fonction des besoins courants (type d'application, conditions environnementales, température, ...). Lorsque l'application ou ses contraintes sont modifiées ou une partie des PEs monte trop en température, la vitesse est adaptée.

Un MPSoC doit donc offrir la possibilité d'adapter les tâches aux contraintes de temps réelles. Son architecture distribuée, procurant un haut degré de parallélisme, répond bien au problème de la variabilité et des applications multitâches. Intégrer des accélérateurs dédiés permet également un calcul à hautes performance et efficacité énergétique. De plus, son architecture régulière réduit le temps de conception et simplifie le passage à l'échelle.

Le contrôle de PEs n'est pas une tâche évidente. Étant donné que les PEs n'ont pas les mêmes caractéristiques et qu'ils dépendent tous les uns des autres, une question importante est de connaître la vitesse optimale de chaque PE pour consommer le moins d'énergie à l'échelle globale sans altérer le fonctionnement de l'application. Plus crucial encore, il s'agit de trouver assez rapidement la bonne solution pour chacun des PEs pour pouvoir reconfigurer le système en temps réel.

Ce processus de décision est complexe, nécessite une puissance de calcul importante et peut consommer une quantité non négligeable d'énergie. Le temps de la décision et l'énergie nécessaire peuvent être alors trop importants par rapport aux gains potentiels. Un principe de décision rapide et efficace est indispensable, ce qui est difficile à implémenter avec la technologie disponible aujourd'hui.

Dans le contexte des MPSoCs, la nécessité d'une haute performance avec une faible consommation d'énergie est clairement une affaire de première importance. Par comparaison, le cerveau humain est un bon exemple de système extrêmement performant tout en restant économe en énergie. Bien que cet organe ne consomme que 20W au plus fort de son travail (Drubach, 1999), il surpasse le supercalculateur actuellement le plus rapide (Tianhe-2 qui consomme 17.8 MW (Sengupta and Stemmler, 2014)), dans des tâches de haut niveau telles que la prise de décision, la classification, les associations, les croisements ou la production d'informations, fonctionnant même en présence de bruit, d'incertitudes et d'erreurs (Whitworth, 2008, Rojas, 1996). L'écart entre le rendement énergétique des MPSoCs et du cerveau humain montre qu'il n'est pas impossible d'imaginer des solutions, lesquelles seraient donc neuro-inspirées, pour améliorer la qualité des circuits électroniques.

Le défi de combiner haute performance et efficacité énergétique élevée est vraiment fondamental dans notre travail. L'objectif précis de cette thèse est d'appliquer le principe

d'un type de réseaux de neurones récemment introduit qui repose sur des cliques neurales (Gripon and Berrou, 2011a). Ce nouveau réseau de neurones offre un large gain de capacité de mémorisation et de performance par rapport aux réseaux de neurones conventionnels. Il s'appuie sur le principe de parcimonie pour stocker et retrouver efficacement des informations quantifiées et ouvre des perspectives nouvelles pour des implémentations de faible puissance. L'architecture des MPSoCs se prête très bien à l'idée d'utiliser des réseaux de neurones pour contrôler leur consommation. Notre objectif est d'explorer la possibilité pour les réseaux de cliques neurales d'être utilisés à cette fin.

Nous avons donc analysé ces réseaux dans des applications pratiques à l'aide de données du monde réel obtenues à partir de mesures et de simulations. Nous avons montré que, pour être utilisés dans des applications réalistes, le modèle initialement proposé de réseaux de cliques neurales doit être amélioré. Le réseau tel qu'initialement introduit avait été analysé et évalué uniquement pour des messages indépendants et identiquement distribués (i.i.d.) (Gripon and Berrou, 2011a). En termes de connectivité, cela signifie que le nombre de connexions sortant de chaque nœud est uniformément réparti sur l'ensemble du réseau. Il est bien connu que la non-uniformité de messages à mémoriser dans des structures connexionnistes peut conduire à une diminution importante des performances (Knoblauch et al., 2010). Les applications réelles que nous ambitionnons de traiter peuvent contenir des données hautement corrélées. Nous analysons donc des situations dans lesquelles des données non uniformes sont stockées et nous expliquons leur influence sur la performance du réseau. Afin d'approcher les performances théoriques dans des applications du monde réel et non plus seulement sur des données i.i.d., le modèle doit être adapté. Nous exploitons donc les structures de ces réseaux pour introduire plusieurs techniques afin de stocker efficacement des données non-uniformes (Boguslawski et al., 2014). La méthode la plus efficace repose sur le concept de neurones jumeaux que nous avons introduit et fait l'objet d'une analyse mathématique validée par des simulations (Boguslawski et al., 2015a).

Ensuite, concrètement, les réseaux de cliques neurales sont utilisés à des fins de gestion dynamique de l'alimentation des circuits électroniques. Les réseaux sont évalués dans ce contexte pratique en utilisant des données du monde réel obtenues à partir de simulations et de mesures. Avec ces données, il est montré que les réseaux standards deviennent inefficaces et qu'il est absolument nécessaire d'adapter le modèle. Les résultats obtenus

avec les neurones jumeaux montrent que cette solution offre des performances proches de celles qui sont obtenues dans le cas de données uniformément distribuées. Le taux d'erreur dans la récupération des messages est un indicateur important des réseaux de neurones utilisés en tant que mémoires associatives. Dès que les cliques neurales sont utilisées avec des données du monde réel, il est particulièrement intéressant d'évaluer leur performance en utilisant des paramètres applicatifs qui estiment l'impact d'erreurs sur l'application. En outre, mettre toutes les erreurs dans une catégorie n'est pas assez précis. Lorsque les paramètres physiques sont stockés dans les réseaux, un grand écart entre une valeur récupérée et la valeur correcte a plus d'impact sur l'application qu'un petit écart. Ceci est pris en compte dans ce travail où la performance de réseaux est évaluée en termes de paramètres physiques dans un contexte applicatif.

Nous avons ensuite proposé une architecture de cliques neurales spécifique pour un organe de décision dans la gestion de l'alimentation de MPSoCs. Nous avons comparé notre proposition à d'autres types d'organes de décision basés sur la théorie des jeux ou sur une mémoire associative CAM-SRAM (Content-Addressable Memory). Par rapport à une solution s'appuyant sur la théorie des jeux, nous avons montré que notre unité de décision consomme 6800 moins d'énergie et réagit 4500 fois plus rapidement (Larras et al., 2013b). Elle offre également des économies d'énergie plus élevées (60% par rapport à 38%) grâce à l'utilisation de méthodes d'optimisation avancées, appliquées au moment de la conception et à une meilleure réactivité (Larras et al., 2014). Par rapport à la mémoire associative CAM-SRAM, nous avons montré que notre unité de décision est moins complexe et consomme 48% moins d'énergie (Boguslawski et al., 2015c). Nous avons en outre proposé des techniques pour améliorer la fiabilité des circuits de cliques neurales et nous avons estimé le coût de la programmation des synapses du réseau.

La clique matérialisant un message particulier stocké dans le réseau contient bien plus d'informations que nécessaire (Gripon and Berrou, 2011b). Grâce à cette redondance, la récupération de ce message est possible à partir d'informations partielles et/ou bruitées. Cette redondance étant portée par des connexions, la complexité de l'implémentation matérielle est dominée par les fils. Les longs fils impactent la consommation d'énergie et le temps de réponse puisque tous les neurones (sommets) de la clique doivent échanger des signaux entre eux. Pour cette raison, il est intéressant d'organiser les neurones en 3D afin de pouvoir créer des cliques 3D avec des connexions plus courtes et d'obtenir un temps de réponse et une consommation d'énergie plus faibles.

Le circuit analogique décrit dans la section 4.2.1 est une implémentation de haute performance avec des connexions physiques pour toutes les synapses. Un câblage important est toutefois nécessaire pour l'interconnexion de toutes les parties du réseau. Cette interconnexion a été optimisée grâce à une nouvelle approche d'interconnexion 3D pour la mise en œuvre efficace des cliques neurales. Nous avons pu obtenir jusqu'à 55% de réduction de la longueur totale de l'interconnexion et de la consommation d'énergie liée à l'interconnexion, et 74% de réduction du temps maximal de propagation par l'interconnexion (Boguslawski et al., 2015b).

Toutefois, les contributions de ce travail sont plus générales que la seule application aux MPSoCs. Les méthodes proposées pour adapter les cliques neurales à des applications du monde réel peuvent convenir à d'autres enjeux où le concept de mémoire associative est utile. En outre, plusieurs stratégies de conception ont été proposées dans le chapitre 3, chacune ayant ses avantages, ce qui permet de choisir celle convenant le mieux à une application particulière. Le type de mémoire associative que nous avons proposé est efficace en termes d'énergie et est une alternative intéressante aux CAM conventionnelles.

Le travail présenté ici ouvre un grand nombre de perspectives au niveau des implémentations et des applications. Les cliques neurales, avec les améliorations que nous avons proposées et validées dans notre travail de thèse, peuvent trouver des applications nombreuses en dehors du seul cas de figure de la gestion de l'alimentation dans les circuits intégrés.

Résumé

Les applications microélectroniques d'aujourd'hui nécessitent de combiner haute performance et efficacité énergétique. Cet objectif est accessible grâce aux «Multiprocessor System-on-Chip» (MPSoC) qui fournissent un haut niveau d'adaptabilité, de performance, de fiabilité et d'efficacité énergétique. Néanmoins, ils doivent être couplés à des systèmes de décision et de gestion de la consommation qui adaptent en permanence leur mode de fonctionnement. Cela nous amène à considérer les réseaux de neurones, lesquels ont vocation à offrir des propriétés comparables à celles du cortex cérébral. Ils peuvent notamment servir comme mémoire associative, avec un processus de relecture très économe en énergie. Nous analysons ces réseaux dans des applications de gestion de l'alimentation, avec des données du monde réel. Nous montrons que pour être compatible avec des applications réalistes, le modèle de réseaux de cliques neurales initialement proposé doit être amélioré. Nous proposons plusieurs améliorations et présentons le concept de neurones jumeaux. L'activité parcimonieuse dans ces réseaux ouvre des opportunités pour des implémentations à faible consommation. Nous montrons que les cliques neurales sont plus efficaces en termes d'énergie que des solutions de gestion de l'alimentation reposant sur la théorie des jeux ou des «Content-Addressable Memory» (CAM). En outre, nous proposons une nouvelle approche d'interconnexion 3D pour l'implémentation physique de cliques neurales à haute performance. Nous montrons des gains importants en termes de longueur totale d'interconnexion et consommation d'énergie.

Mots-clés : Réseaux de neurones, Intelligence artificielle, Microélectronique, Multiprocesseurs, Systèmes sur puce, Mémoires associatives, Circuits intégrés tridimensionnels, Circuits électroniques

Abstract

The challenge of combining high-performance and energy efficiency is clearly present in today's electronics applications. This objective can be reached with Multiprocessor System-on-Chip (MPSoC) platforms that provide high-level of adaptability, performance, reliability and energy efficiency. Nevertheless, they have to be accompanied with a power management decision unit that continuously adapts their operation. This leads us to neural networks that can provide comparable properties to those of the cerebral cortex. We use an associative memory that relies on neural cliques to store information and sparse activity to retrieve it. We analyze these networks in power management applications using real-world data. We show that in order to be compatible with real-world applications, the initially proposed model of networks of neural cliques has to be improved. We propose several methods to do so within which the most efficient relies on a proposed concept of twin neurons. Sparse activity in these networks opens opportunities for low-power implementations. We show that neural cliques are more energy-efficient than power management solutions relying on game theory or Content-Addressable Memory (CAM). Furthermore, we propose a novel 3D interconnect approach for high-performance neural cliques' implementation. We show important gains in terms of total interconnect length and power consumption.

Keywords : Keywords: Neural networks, Artificial intelligence, Microelectronics, Multiprocessors, Systems on a chip, Associative storage, Three-dimensional integrated circuits, Electronic circuits

TELECOM Bretagne

n° d'ordre : 2015telb0370

Télécom Bretagne
Technopôle Brest-Iroise - CS 83818 - 29238 Brest Cedex 3
Tél : + 33(0) 29 00 11 11 - Fax : + 33(0) 29 00 10 00