On microelectronic self-learning cognitive chip systems

PUBLISHER

© Ludovic Alain Krundel

REPOSITORY RECORD

Krundel, Ludovic. 2019. “On Microelectronic Self-learning Cognitive Chip Systems”. figshare. https://hdl.handle.net/2134/21804.

On Microelectronic Self-Learning Cognitive Chip Systems

by

Ludovic Krundel
Electronic Systems Design Research Group
School of Electronic, Electrical and Systems Engineering
Loughborough University of Technology
Loughborough, United Kingdom

A Doctoral Thesis

Submitted in partial fulfilment of the requirements for the award of the degree of

Doctor of Philosophy of Loughborough University

Lead Supervisor: Dr. David Mulvaney

Associate Supervisor: Prof. Dr. Roger Goodall

Advisory Supervisor: Dr. Vassilios Chouliaras

January 08, 2016

Abstract

After a brief review of machine learning techniques and applications, this Ph.D. thesis examines several approaches for implementing machine learning architectures and algorithms into hardware within our laboratory. From this interdisciplinary background support, we have motivations for novel approaches that we intend to follow as an objective of innovative hardware implementations of dynamically self-reconfigurable logic for enhanced self-adaptive, self-(re)organizing and eventually self-assembling machine learning systems, while developing this new particular area of research.

And after reviewing some relevant background of robotic control methods followed by most recent advanced cognitive controllers, this Ph.D. thesis suggests that amongst many well-known ways of designing operational technologies, the design methodologies of those leading-edge high-tech devices such as ‘cognitive chips’ that may well lead to intelligent machines exhibiting conscious phenomena should crucially be restricted to extremely well defined constraints.

Roboticists also need those as specifications to help decide upfront on otherwise infinitely free hardware/software design details. In addition and most importantly, we propose these specifications as methodological guidelines tightly related to ethics and the nowadays well-identified workings of the human body and of its psyche.

Keywords: Machine Learning, Cybernetics, Cellular Automata, Neural Networks, Field-Programmable Gate Array (FPGA) devices, Hardware Description Languages, Asynchronous Design, Simultaneous Parallel Processes, Wetware, Morphware Chips, Learning Algorithms, Growth Rules, Reconnection Method Policies, Cognitive Architectures, Microelectronic Mental Properties, Human-Machine interactions, Ethical issues in Robotics, Machine Intelligence, Artificial Capabilities.

Acknowledgements

I would like to express my deepest gratitude to my lead supervisor Dr David Mulvaney for his invaluable guidance, extensive comments, and generous support with both ideas and advices, without this this thesis would not have been accomplished.

Thanks are due to my associate supervisor Professor Dr Roger Goodall and advisory supervisor Dr Vassilios Chouliaras for their contentious support and advice.

Thanks are also due to all friends, close ones as well as the others, and my family who supported me with a piece of advice, a word, or a smile. In particular, to name only a few of my best close friends who have immensely supported and sponsored me in many possible ways without doubting me nor letting me down a single second over the years: Dr Cristina Marinho, Bruno Jansen, Didier Souchon and sister Pr Dr Anne Souchon, Christophe & Magali Bastian, Olivier Voisin, Dr Chou Srey, Dr Juliana Zhao, Emmanuel Poirier, Helen He, Dr Katty Liao, Alice Song, Dr Sergey Slizovskiy, Mandi Feng, my kind neighbours in Loughborough John & Val, Dr Jim Zhou, Dr Quentin Montardy and last but not least,

my sweet darling partner Dora Xue.

I want to dedicate the present thesis and PhD works to my dad & mum Gérard & Christine Krundel.

Without any of the above named persons plus quite a few other, this project would never have been possibly achieved.

I am deeply grateful and immensely thankful for this utterly astounding, incredible and indescribable journey at Loughborough University.

Publications

[1] Ludovic Krundel, David Mulvaney, Vassilios Chouliaras, Yimin Zhou, Guoqing Xu, "Efficient Parallel Asynchronous Hardware Algorithm of On-Chip Cellular Automata for Formation of Utilizable Digital Neural Tissue" in IEEE-ROBIO 2013: 2013 IEEE International Conference on Robotics and Biomimetics: 'Robots living with human being in modern society', Shenzhen Institute of Advanced Technologies (SIAT), Shenzhen, China, Dec. 12–14, 2013.

[2] Yimin Zhou, Ludovic Krundel, David Mulvaney, Vassilios Chouliaras, Guoqing Xu, "Autonomous Spiking Neural Network on SOC" in ICIEA2013: Proceedings of the 8th IEEE Conference on Industrial Electronics and Applications, pp.1106-1111, Swinburne University of Technology, Melbourne, Australia, Jun. 19–21, 2013.

[3] Ludovic Krundel, David Mulvaney, and Vassilios Chouliaras, "A Compact Cognitive Co-Processor for Robotic Action Learning" in ASAB2012: Special Issue of Animal Behaviour, Association for the Study of Animal Behaviour (ASAB) Interdisciplinary Workshop 2012: 'Physical Cognition & Problem Solving', University of Birmingham, UK, Jun. 27–28, 2012.

[4] Ludovic Krundel, Ya-Chien Huang, David Mulvaney, and Vassilios Chouliaras, "Ethical Methodology for the Design of Third Generation Robots and of New Advanced High-Tech Interactive Support" in SIRCon2011: Proceedings of the 2011 IEEE sponsored International Conference on Service and Interactive Robotics (SIRCon), IEEE Robotics and Automation Society (IEEE-RAS) and Robotics Society of Taiwan (RST), National Chung Hsing University, Taichung, Taiwan. IEEE Systems, Man, and Cybernetics Society, Nov. 24–26, 2011.

[5] Yimin Zhou, Ludovic Krundel, David Mulvaney, and Vassilios Chouliaras, "Spiking Neural Networks on Self-Updating System-on-a-Chip for Autonomous Control" in CCCA2011: Pro-

ceedings of the 2011 IEEE sponsored International Conference on Computers, Communications, Control and Automation (CCCA 2011), Hong Kong, China, Feb. 20–21, 2011.

[6] Yimin Zhou, Ludovic Krundel, David Mulvaney, and Vassilios Chouliaras, “Dynamic Rule Learning in Cellular Neural Network Design – On an Autonomous Design Chip” in ITIP2010: Proceedings of the 2010 IEEE sponsored 2nd International Conference on Intellectual Technique in Industrial Practice (ITIP 2010), Changsha, China, Sep. 8–9, 2010.

[7] L. A. Krundel, D. J. Mulvaney, and V. A. Chouliaras, “Autonomous Design in VLSI: an In-House Universal Cellular Neural Platform” in ISVLSI2010: Proceedings of the 2010 IEEE Computer Society Annual Symposium on VLSI (ISVLSI conference), Conference Center, Lixouri, Kefalonia, Greece, Jul. 5–7, 2010. `http://www.isvlsi2010.org/?pages_id=46&lng_id=2` → PF8

[8] L. A. Krundel, D. J. Mulvaney, and V. A. Chouliaras, “Autonomous Design in VLSI: Growing and Learning on Silicon” in ISVLSI2010: Proceedings of the 2010 IEEE Computer Society Annual Symposium on VLSI (ISVLSI conference), Conference Center, Lixouri, Kefalonia, Greece, Jul. 5–7, 2010. `http://www.isvlsi2010.org/?pages_id=46&lng_id=2` → P14

[9] L. A. Krundel, D. J. Mulvaney, and V. A. Chouliaras, “Self-adaptive compact neuro-controller for natural interaction and training of autonomous communicating robots,” in SIRCon2009: Proceedings of the 2009 International Conference on Service and Interactive Robotics (SIRCon), IEEE Robotics and Automation Society (IEEE–RAS) and Robotics Society of Taiwan (RST), Taipei World Trade Center, Nangang Exhibition Hall, Taipei, Taiwan. IEEE Computer Society, Aug. 6–8, 2009.

[10] L. A. Krundel, “Harnessing Emergence with Self-Organization for Autonomous Architec-

ture Design and High-Level Adaptation," in Showcase of Research School of Systems Engineering, Loughborough University, Loughborough, UK. SEIC, Jul. 6, 2009.

[11] Ludovic Krundel, "Self-Organizing Strategies for the Emergence of Goal-Adaptive Behaviours in Autonomous Communicating Machines," in The 2nd International Symposium on Neural Networks and Econophysics: from superconducting junctions to financial markets, Department of Physics and Business School, Loughborough University, Loughborough, UK. European Science Foundation (ESF) – Arrays of Quantum Dots and Josephson Junctions (AQDJJ) Network-Programme and Loughborough University, Jun. 13–17, 2009.

[12] Ludovic Krundel, "Neural Networks with Cellular Automata," in Symposium on Neural Networks and Cellular Automata: from superconducting junctions to biology, Department of Physics, Loughborough University, Loughborough, UK. European Science Foundation (ESF) – Arrays of Quantum Dots and Josephson Junctions (AQDJJ) Network-Programme and Loughborough University, Feb. 4–5, 2008.

[13] L. A. Krundel, V. A. Chouliaras, and D. J. Mulvaney, "A knowledgeable authority for interactive hardware design capture support," in VLSI-SoC 2006: Ph.D. Forum Digest of Papers of the 2006 International Conference on Very Large Scale Integration (IFIP VLSI-SoC), IEEE, IFIP, Nice, France, Oct. 16–18, 2006, pp. 13–18. [Online]. Available: `http://vassilios-chouliaras.com/pubs/c38.pdf`

[14] L. A. Krundel, S. K. Goel, E.-J. Marinissen, M.-L. Flottes, and B. Rouzeyre, "User-constrained test architecture design for modular soc testing," in ETS'04: Proceedings of the European Test Symposium, Ninth IEEE ETS'04, Ajaccio, Corsica: IEEE Computer Society, May 23–26, 2004, pp. 80–85. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ETSYM.2004.13

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"The purpose of computing
is insight,
not numbers."
Richard Wesley Hamming

For the past decade, machine learning (ML) has been an emerging discipline that includes nearly all ancient and modern artificial intelligence (AI) techniques from decision trees to fuzzy logic, through to artificial neural networks, particle swarm optimization, genetic algorithms, genetic programming and evolutionary programming to name only a few. The field has increasingly attracted researchers as neural networks in particular, sadly and world-widely forgotten in the 1970s for *a priori* reasons, gained again interest in the 1980s with advances in computer technology, progress in neuroscience and real needs for brain-like information processing [1].

"Machine Learning is the study of computer algorithms (of which the main goal is to increase the machine's knowledge) that improve automatically through experience" [2]. As such, it gives good expectations about the success of data mining, knowledge handling and more generally: right interpretation, true understanding and judicious use of information towards 'cognitive and thinking machines' [3–6] as well as real-world autonomous robots [1,7–9]. These are of prime importance stakes for the near future of our technologically advanced societies [10–12].

Ideally, these machines should exhibit behaviours that adapt to changing contexts, highly dynamic nowadays. To perform this, they implicitly must behave intelligently by interpreting their own outputs, be able to self-manage, self-repair and at the limit self-replicate [13].

However, several reasons motivated some researchers having expertise both in artificial intelligence discipline and in electronics to implement their machine learning algorithms into hardware. Being willing to do this is mainly justified by the ability of a single silicon die to process different structured pieces of user-meaningful information in parallel and simultaneously, whereas microprocessors (executing instructions of software-based algorithms sequentially) do not take advantage of this as only one piece of user-meaningful information (multi-threaded or not) can be processed at a time serially through the single pipeline data-path.

1.1 Current State-of-the-Art

Although parallel data-paths, extensive multi-threading, and multi-core microprocessor dies have recently been developed successfully to maturity and mainly for industrial and commercial reasons[1], the author is convinced that highly distributed and massively parallel tiny processing units architectures largely surpass current multi-* microprocessor-software technology in most respects, such as no single point of failure to name one only here for now. But to add more inconvenience and computational indeterminism sometimes fatal in the software realm, other service programmes that are vital to the operating system (OS) must be processed concurrently – but serially in time – with the user's information data through the pipeline. Hence, $\mathcal{NP}$-complex problems arise, such as processes scheduling and have to be subsequently tackled beside. This kind of then necessary sophistication on top of immature software development (because of clumsy approaches and old architectures) dramatically increases the system complexity already significantly high.

Next to this, the advent of programmable logic devices (PLDs) such as complex PLDs

---

[1] It is secure to continue from an existing technology rather than switching to a radically novel one.

(CPLDs) or Field-Programmable Gate Arrays (FPGAs) based on highly distributed and massively parallel processing units architectures, and also enabling to inexpensively re-wire their hardware design many times (apart from the ‘antifuse-based’ ones such as some microelectronic chips by Actel company) upon the designer’s will, aroused even more interest. For instance, prototyping of industrial application specific integrated circuits (ASICs) became dramatically cheaper and easier; the same holds for laboratory experiments on electronics, and especially for original research on artificial intelligence (AI) and more particularly artificial life (Alife) with our concerns.

Beyond this ability to “reconfigure” device’s logic, we could start to think that we should try to entrust some tasks of reconfiguration to the device itself (self-programming software/firmware part) [14], thus increasing the degree of freedom as well as the exploration space of the machine learning algorithms; in other words: offering a new dimension to the learning machine, having then (some partial or even full) control on its own corresponding architecture design.[2]

However, the view of most researchers in this “morphware” paradigm has been mainly to evolve the logic hardware circuitries using genetic algorithms (GAs) in order to exploit the reconfigurability features of the PLDs and subsequently obtain good system solutions quickly. After having put in place a platform to perform evolvable hardware (EHW) experiments, the main required human efforts are to define and implement a **fitness function** for the GA, related to a particular problem desired to be solved. The GA then evolves the hardware, sometimes curiously exploiting it much more efficiently than a human could do with electronic design automation (EDA) tools, until an acceptable (based upon criteria determined in advance) solution is reached.

---

[2] This is more my own view, related to my idea that came straight from this new ability to reconfiguring logic, rather than the view of most researchers who thought first to evolve electronics using genetic algorithms (GAs) for acceleration purposes instead of implementing Machine Learning techniques in hardware – the original work of this PhD –, and tend now to widen their investigation fields from this first idea. Perhaps, say here that this is rather my own view, and that in the facts, everything started with evolvable hardware (EHW), and most of people are now stuck to approaches drawn from EHW instead of radically novel approaches such as the one proposed in the final chapter, preceding the conclusion of the present document.

But this approach does not allow the device to knowingly ‘morph’ towards the improved solution since GAs rather blindly evolve solutions, with a fitness function as single guiding reference.

In the goals of this thesis and beyond, there are no envisions for machines to be able to replace for example 100 engineers and scientists during 10 years along a given extensive project. This is where the word “machine” is typically disliked. When machines can substitute jobs and do so, often for the high interest of a minority of people in schemes where the machines are owned by a few whereas the solution to this is evidently that the machines have no owner and also the machines pay their users! So due to the politically impractical set up of this in actuality, the aforementioned dislike is particularly true in disciplines such as in pure “Automation & Control of Systems”, which covers “resources on the design and development of processes and systems that **minimize the necessity of human intervention**”; resources in this category cover control theory, control engineering, and laboratory and manufacturing automation’.

There are issues in automating tasks and processes, and possible solutions only reachable thanks to good new approaches and envisions, here in the discipline of robotics. These issues arise because these kinds of machines are designed to cope with problems that humans can also tackle, including complex problems. And actually, currently the trend is a bit like on the one hand giving ‘human faculties’ to modern computers and robots, which do not natively fully lend themselves to such advanced functionalities – especially the control parts –, and on the other hand reducing ‘human faculties’ of people so that they better assume specialized tasks, thereby sadly inhibiting human creativity little by little.

1.2 Research Aim

Instead, we will aim at creating at least one sort of original intelligent agent, machine, robot, system (whatever name it can take) that can together with humans, in a fully friendly and fully complementary fashion (and also hopefully often fun atmosphere), solve problems too complex for humans to tackle alone and in reasonable time. For instance, only quantum computers might very soon be able to find the optimal solution of $\mathcal{NP}$-complex problems in seconds, whereas currently, using modern computers allow for finding one acceptable near-optimal solution in reasonable time: hours or days, using heuristics that sometimes require too much human efforts to implement than a general exhaustive search however impossible to computationally run in reasonable time.

The synergistic human-machine symbiosis [15] I am thus talking about could well do the same job as quantum computers, but in a radically more friendly, straight and intuitivelly understood fashion than with quantum computers, and perhaps even sooner than the quantum computing era. This symbiosis will be the so-wished really effective and efficient power extension of humans, providing the best of harmony with nature at the same time. The symbiosis being reached through natural developments as well as thoroughly justified, founded, upright and fair approaches that we feel comfortable and happy with.

An interesting observation is considering that the ‘artificial intelligent lives’ we will have created will be able to explain (working together with us) what are their own mental processes and minds made of as well as ours... What is understanding, and what is real-time learning (by doing: e.g. “motor babbling” as we will see in the rest of this document).

This paper contents: [7] is well in line with my envisions, apart from their obvious military applications where I would rather have ideals such as robotics for: domotics, medical systems and systems accommodating disabled people, (space) exploration and handling of hazardous areas, companion robots, resource management assistants, etc.

A key type of robots for these jobs would be ‘morphing’ robots, which would require real-time control of morphing modular limbs. Then, a long-term goal would be that such a robot immersed in an unusual and unexpected situation comes with an original solution. That is, from both memorised and currently perceived information, the morphing robot creates and generates a new (not a preprogrammed behaviour ready to be selected amongst thousands) way to dynamically cope with this particular situation. To me, this is an example of property that the truly and genuinely intelligent autonomous agent dealt with in [7] should own.

I have very high interest in this kind of adventure, because to date the only comments brought up after a robotic demonstration have been such as: “But this robot is able to perform this feat just because it has been pre-programmed to do so...”. And to date, such comments are unfortunately still justified because the aim of AI has been to emulate human intelligence only, using software programmes. This is not the aim of machine learning in the context of the present work, which is to allow algorithms to self-improve (create their own upgrade patches) through learning while coping with new and unknown situations. Machine learning algorithms need not be sequential programmes like in pure software programming, they can now – with the advent of programmable logic devices (PLDs) – be massively parallel reconfigurable pieces of physical hardware (in contrast to virtual programmes). Due to this parallel computation[3] paradigm, self-reconfigurable information-driven hardware (morphware) has got tremendous advantages over programmable software, such as being able to benefit from uncertainty and indeterministic events, instead of being handicapped or broken down by them.

My long-term aim is to raise **‘artificial mental processes’** (here it would be about **‘microelectronic mental processes’**), and then perhaps being able to observe and deal with the emergence of a conscious awareness [16], of obviously another nature than the human one. As part of my by-product objective given in the following section I wish I will one day be able to sit and discuss smoothly with our amazing creation. But an even more amazing phenomenon to observe would be this creature understanding jokes. The ins and outs of most of jokes are about unexpected situations that suddenly clarify. A system being able to grasp objects and (memorised past) events in a hierarchical internal structure of information, and being able to link them creatively thanks to the hierarchical architecture of the inference system, should be able to face unexpected situations, and therefore grasp some jokes. Hence, the ideas for the PhD are directed towards these long-term ideals. But at the same time, the PhD will concentrate on machine learning algorithms and novel architecture designs meant to physically support them.

---

[3] Several computing processes running at the same time, highly distributed to many different places.

The designs should also be commercially viable as well as time-realistic to achieve within two full years of practical project.

My current very short-term work is to show that a fixed[4] architecture such as the novel one presented in Chapter 4 and reflected in our paper – published at the IEEE sponsored IFIP 2006 conference – is able to learn and mimic an XOR gate function (which has been done before with many other methods) as well as the well-known inverted pendulum problem. After that, I will seek to allow the architecture to self-modify, self-reorganize, gain new habits (with flexibility this time) and tackle more complicated problems, such as mobile robot navigation. Another major goal is to search, find and show that doing these using our architectures, approaches and methods is better, and better suited than with others, if any. This is described in the subsequent chapters of this doctoral thesis.

1.2.1 Important Properties of a System such as proposed here

I divided what I call **'properties of self'** or **'self properties'** into hierarchical contribution of each one. Table 1.1 summarizes this hierarchy and where each **'self property'** plays a role.

Given this, it will be more clear for the remainder of this report to situate which property is dealt with, and which properties must be left for future projects continuing this one, because too much extensive (yet very interesting) to be developed along a PhD project.

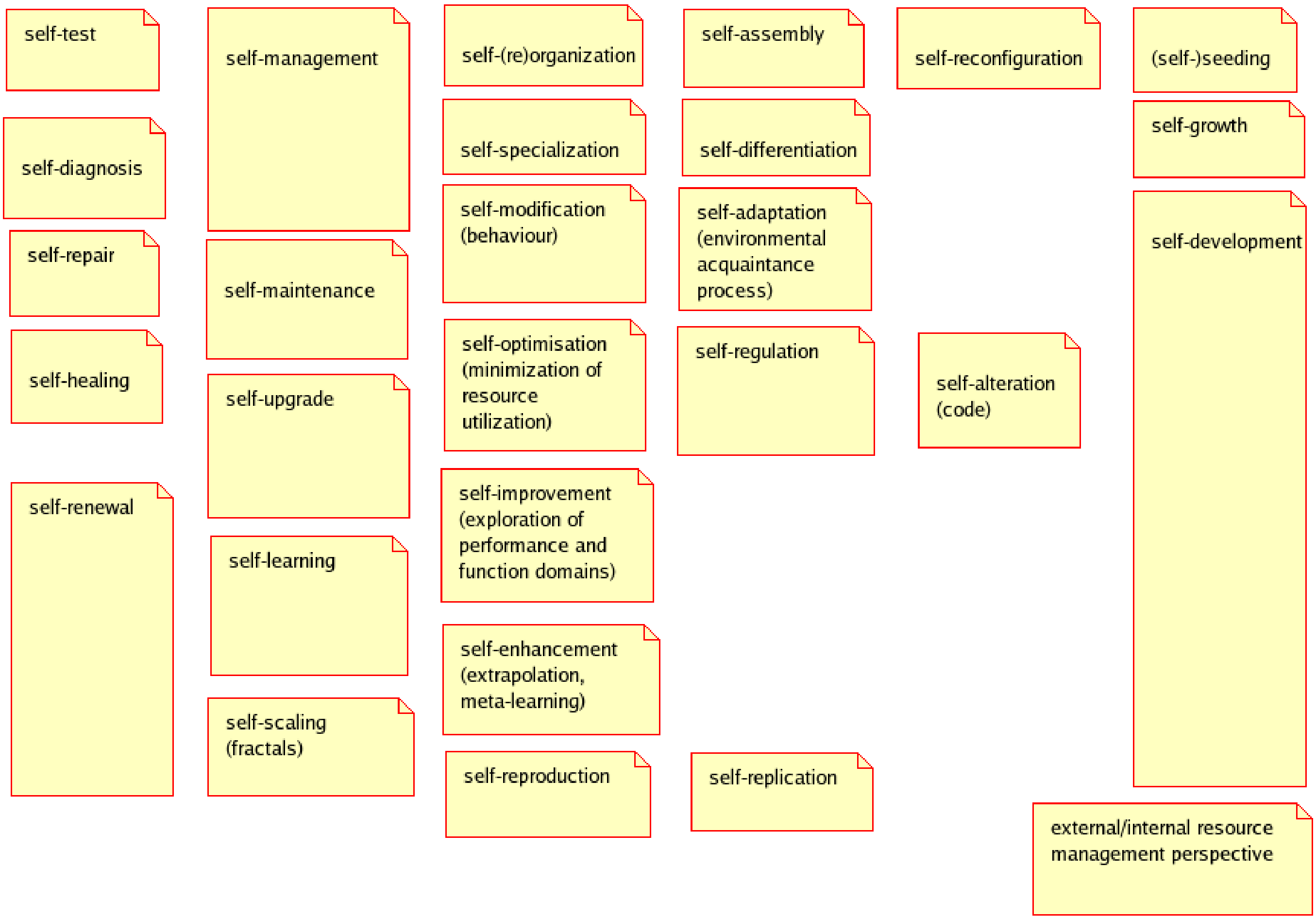
1.3 Research Objectives

The main objective of this PhD project, is to design a machine that will be able to be programmed and reprogrammed through interaction within its environment, together with humans and with other machines. That is, a machine that can learn and remember objects and events, then internally process them (infer, reason or think) to create original new solutions to any unexpected and unknown situations. Only the initial hardware architecture design should be conceived,

---

[4] Fixed on purpose in order to ease experiments as well as to actually fit within FPGA devices.

Table 1.1: Self-Properties or Self-Features.



developed and implemented at hand, together with initial (self-improvable) learning mechanisms. Then, the machine, having authority to (partially) restructure its (at first, initial) architecture and upgrade its learning mechanisms, should self-improve, self-optimize and self-maintain as well as optimizing its surrounding environment including any connected problem solutions. This environmental change in turn will be taken into account by the machine, which will think and react, still improvingly.

The machine is considered as a cog of its world (that we will, in our real world, ultimately be supposed to share with it). This environment-machine change cycle may last indefinitely along an infinite loop. It is very interesting to try and imagine many possible futures after a sufficient number of loop cycles. The machine should have some sorts of immediate **'microelectronic mental processes'**, but will accumulate work as the cycles unfold.

This work progress will be useful to analyse as compared to the intended functioning of the machine in order to understand its emerging (intellectual) behaviours. Thus, as a by-product objective I wish this may contribute to neuro and cognitive science. I insist on the fact that I will take a radically different approach to conventional artificial intelligence (AI), of which the aim is to emulate human intelligence using modern microprocessor-based computers (where data and instructions stored in memories are separate from the processing unit) and writing software, often having to write extra scripts in order to deliberately handicap the computer software system on some aspects, such as delaying answering messages, in order for a person to believe that the machine is truly another person. This is an example of stratagems and artefacts that have to be thought of prior to building or after observing behavioural issues of the system. These sorts of handcrafted sophistications simply go against nature – in which I believe physical emotions are the 'intelligence foundations' –, are actually not genuinely in harmony with it, and this is sufficient for me to understand and explain the huge pitfalls in classical AI so far.

An important question from the above is to know about the stake of these implementation actions, if undertaken. Indeed, the primary goal in our concern is to reduce the human hardware design efforts so that design capture is simpler and thus achievable in some cases, yet more powerful. But at this stage, we can already claim that human hardware design expertises become less necessary. Attracted by the flexibility of FPGA hardware, and its ability to self-reconfigure, one can even try to entrust design tasks to the device itself, including the design tools or not. In that extreme perspective, human hardware design expertises become superfluous, and hardware design expert jobs are virtually threatened. Nevertheless, humans are still needed (as catalysts maybe) to invent and create new initial architecture designs, and implement them via a chosen design process. The FPGA design (and possibly tools) being then able to self-develop and reach its culminating performance by itself. But if out of these designs ever emerge some 'microelectronic mental faculties' appropriate for artificial hardware design in their turn, human catalysts may not be needed any-more either...

I will not therefore take such a calculable approach, where there are operators and operands, furthermore physically standing at different places in the system, and where software programmes are executed in a virtual (digital) world, thus involving the utilization of virtual-world-to-physical-world mappers, leading to cumbersome and unreliable real-time control. Therefore, I have the envision of a physical 'hardware operating memory perspective', where data are stored in a distributed fashion, and truly processed simultaneously in parallel locally to where they are stored, by distributed processing units. This physical (control) system should exhibit straight mapping between its internal operations and its external behaviours. The absence of mapper can guarantee easy and reliable real-time control within a real-life environment (that we are also supposed to share).

1.4 Research Approach

The approach presented in the report is a bottom-up approach. It consists in designing an initial cellular automata (CA) microarchitecture that, through the course of events – partially governed by (self-modifiable) rules – will generate an emerging neural network (NN) structure. The latter is allowed to learn about properties of its own underlying CA, such as rules[5], and modify some of them along trial-and-error processes, thereby seeking to self-manage and self-improve through 'self-reorganization'. This system can be viewed as a dual CA-NN Self-Reconfigurable System. It has a dual aspect in the sense that it is a 'self-reorganizing' NN, which can be viewed as a CA during reconfiguration.

An obvious scenario example with our dual CA-NN self-reconfigurable system is when it would have reached a point where stability (or global equilibrium), self-management and consistent intelligent behaviours are ensured. At this point, we could claim the system to be polymorphic and actually usefully, effectively and efficiently morphing. Plug this (/these)

---

[5] Neural networks are particularly good for learning rules [1].

system(s) as a (distributed) controller(s) of a hyper-redundant morphing robot and a hazardous (adventurous), yet highly interesting machine takes (artificial) life.

The outcome of this is that the more the design efforts are sought to be lowered down, the less the design expertise is required. But creative invention will always be required to produce ideas and concepts. It will always be performed by humans, but at some point in these ascending improvement spiral scenarios, intelligent machines with some sort of 'artificial hardware cerebral power' may also join research engineering teams as producers of ideas and concepts. This being not a problem but an opportunity for mankind to improve through mutual friendly interdependence together with machines is argued for at the end of Chapter 11.

Fears from such possible control system engineering science outcomes are quite comprehensible. The most obvious fear is that machines ever replace human jobs. Another fear is being a control engineer working hard (and being a master at controlling / dictating) to incorporate as much as possible autonomy into machines, themselves consequently having enough 'self-features' that they ironically escape out of human control and of their designers. Ironically too, mankind might get endangered by dumb or dumbed down intelligent machines.

Actually, these are fake problems. Intelligent machines will never completely take the place of humans, but rather come to complement humans and contribute to enjoy both human and machine lives, within a mutual improvement then made possible. Machines will have another intelligent nature than the one of humans.

If one fears losing control over machines (or over someone else), this is first because one has the big initial assumption to be the master and to have full control over machines (or over others), then possibly losing this self-granted privilege. This is false. Even though one would think to have full control over their, say, personal computer, the experience shows that these particular machines, yet meant to respond to commands, can sometimes behave strangely and unexpectedly. This is just an effect of system complexity, and intelligent sophisticated machines, such as our dual CA-NN system, also must hold a certain level of complexity, hopefully self-consistent in our case in order to work well enough. The main difference between conventional computers and our CA-NN system is that our system will be able to handle uncertainty and indeterminism. And it will actually be one of its strengths, permitting to remember experiences, to learn, and finally to understand, mainly through prediction.

To complement this, machine inventors and users have these long tradition, deep feelings and mentality that machines are and will always be meant to wholly and mainly serve humans (until and including Asimo-II now). And because of this, they have difficulties to imagine and accept the idea of machines not being servants, that is, becoming human-interdependent (even human-independent). But there cannot be cheerful mutual inter-improvement between humans and machines if one kind always dictates the other one in a permanent one-way fashion only. The least we can imagine getting from that is the rebellion from the servants (or slaves?) at some point: here the machines. Of course, as their creator, one can argue they should owe us the corresponding respect. Yet, this does not mean we should not owe respect to them.

Hence, people are only afraid of disillusions which are actually not. One will not lose control over someone or something since one never has got full control, but rather generally little control over others and other things including machines.

Intelligent machines or intelligent agents being fully autonomous does not necessarily mean being fully independent (internal resources management, external resources care). They will actually be interdependent with each others, with their environment and with humans, as humans themselves already are together, and are governed by laws of physics (and of societies). But then, could they ever become human-resource-independent sometime later, after sufficient self-improvement? This is a difficult question to answer to, but even a positive answer does not mean humans would be threatened. The right question to ask is: what would be the overall (long-term) interest and aim (unique if constant), the agenda, of autonomous machines or agents having cerebral power and mental faculty through ‘artificial mental processes’ that they would want?

The answers actually depend on many aspects, such as the architecture and functioning of these machines, but also on things like our own definition of what is life. Like for humans, if machines operate according to effort/reward mechanisms, they will not have one life aim, but possibly several life goals. They will just aim at daily happiness, aim at improving to fulfil their ambition, but they cannot know neither define some ultimate aims to attain that would for example put the human species in danger. Their aim, if any, will not be an End, but choosing their way of evolution, their life quality...

Long traditions of anthropocentricism let us in the beliefs that the unique place where the most sophisticated intelligence and form of life can sit is within human brain. Also, we have this common definition of life from whatever is made of biological components (carbonate chains). But let's assume that life is no more than the combination of existence (space-time) and 'indeterminism through the courses of event' (quantum physics). And actually with this definition, life is much more than limited to biology only. In this paradigm, life is ubiquitous.

Therefore, when times come that machines are fully autonomous, which means that we do not have internal control on them anymore, everything will need to pass through the human-machine interfaces. And the results of any interaction undertaken with machines will highly depend on the environment, and on the courses of event spent with them. It is then all about politics, where machines are equally involved in because considered as friends and recognised as whole beings, having rights and duties.

However, because the amount of prior work topics would be too large to develop for a first year report (although I am already f amiliar with half of them because I have read and learned while performing the literature review), I am going to fill them up in the same time that I need to use them.

1.5 Ethical Clearance and Concerns

1.5.1 Ethical Clearance

The data, design and algorithms from this research have been solely created by the author. The various tools utilized are open source and free to download and use in the framework of higher education.

1.5.2 Ethical Concerns in Existing Works

In this webpage: [17] you may watch an amazing video published in January 2006 of Asimo demonstrating some of its abilities in a conference. Towards the end of this video, you will see Asimo serving (with much dexterity) drinks and so to a lady seated on a sofa...

I can imagine that this is where a danger may come from. The first day this amazing little robot[6] sees the light, it is put in an obedient submissive (slavery?) position, by its own designers, which means they already had that in mind [7], thinking that it will please the public, but perhaps not realizing what I am emphasising here.

As past lessons have taught us, and here again under a more subtle aspect, the dangers do not come from the processes of engineering artefacts, neither from the artefacts themselves, but surely from the way they are thought to be used, therefore from what they are made of, and how they are actually used. And in my rather subtle example here with Asimo position as a fateful servant, it will be more a matter of how to respectfully interact with robots. Further robots to come will not be as naive as Asimo...

It will be of our responsibilities (the users, or rather interacting humans) and especially of the responsibilities of the robots designers to duly respect these robots, as we us humans owe to respect each others. If we want to minimize the chances of bad surprises, shall we simply consider them as 'interdependent friends'.

In Chapter 11 on ethical issues in smart robotics, I do not only argue for building intelligent machines, I also deal with the idea of what is called "artificial consciousness".

---

[6] Not truly nor genuinely intelligent from my viewpoint because of the approaches taken to build it.

[7] I actually came across a statement that was indeed explicit enough to assume that the main goal of Asimo designers was to build a serving robot.

1.6 Original Contributions

The author has made contributions in the field of the theme of this PhD thesis by focusing on electronic self-learning cognitive chip systems coupled with the ethical concerns of integration of those systems in human-machine interfaces in the framework of loose intelligent robots as follows.

Original Ideas: Neuro-controller chip interfaceable with arbitrary robotic bodies, 'Seed design' implementable by arbitrary microelectronic hardware substrate such as system-on-a-chips (SoCs), Universal Self-Adaptable Neuro-Controller Co-Processor Chip, Exploiting Cellular Automata for useful and smart internal as well as sensory I/O ports (re-)connections of Hardware Neural Networks, Harness of Emerging Phenomena.

Original Implementations: Made autonomous the designing and low-level code programming of hardware substrates through high-level human-machine interactions, Straightforward natural and very quick processes (excellent time-to-effectiveness), Homogeneous flexible hardware architecture with self-specializing modules, No bottleneck with parallel data processes, Asynchronous parallel operations suitable for biologically-inspired data operations, Self-settling of dependencies between both hardware and software layers (wetware), Distributed control and decisions: no single point of failure, New hard/soft architecture for successful self-modifying systems, Data-driven operations do not require clinging to standard protocols, True real-time processing effective.

Biologically Inspired Methods: Bottom-up Engineering: Cellular Computing, Asynchronous Parallel Operations, Distributed Decisions for Cellular Automata, Non-Uniform Rules, Adaptable Rules (policies-based self-reorganization), Rule Discovery with Neural Network, Holarchy: modular hierarchy of holons.

Uniquely Novel and Major Contributions to Research Knowledge: Discovered manually manageable sets of Rules for Cellular Automata while keeping high complexity potential, Asynchronous Parallel Operations of Distributed Digital System, Dynamic Rules Production, Loose

Neural Data Signalling, Obtaining seeds (from scratch), Cellular Programme (instead of phylogenetic evolution with Genetic Algorithms), Fixed Modular Structures on FPGA (flexibility would requires: over-convoluted mechanisms on FPGA or other technologies than FPGA), Viable Radically New System, Investigated novel useful formation of neural tissue (from seed design), Focused on making fluid, graceful motions (aesthetic value) of physical robots, Very challenging to implement true punishment-reward mechanisms both immediate (reflexes/instinct) and anticipated (consciousness/awareness), On Hebbian Learning: Hitting a Big Wall of understanding: found out and concluded there is to day no reasonable answer about the yet neurally used information from post-synaptic neuron firing(?).

A very long-term goal of contribution to the body of science is my study of: Emergence of intelligent behaviours through 'machine thinking', Eventual evidence of the agent to have intentions and take initiatives[8], Side effects when the agent merges memorised together with currently perceived information: emergence of artificial consciousness? mind? feeling of Self?

In their turn, these will help to understand better human behaviours, and to provide answering elements to fundamental and crucial questions which are still unclear, even though these agents will obviously be of different nature than mankind/mammals.

1.7 Organisation and Structure of the Thesis

The remainder of this document is organized as follows. Chapter 2 gives the background basing this work and Chapter 3 shows the literature review undergone in order to build up the same. Chapter 4 delivers important prior work found in research literature relevant to this project, and it lists examples of common machine learning applications. It also focuses on the background of hardware implementations of machine learning techniques. And we thoroughly justify our novel approaches also presented there. This brings motivation and establishes foundations for

---

[8] Presumably due to internal mechanisms of improvement evaluation – using information theory-based techniques, such as entropy measures and comparisons – and of effective improving changes.

the following chapters. That preliminary chapter also presents the initial methodology adopted along the work, details the architecture design, gives an example of algorithm for the cellular operation of the system, and shows early simulation results of a cellular automata forming a neural network structure using self-organization mechanisms based on information theory techniques. It also suggests several applications for our system. The subsequent chapters build upon their previous ones and improve the work until and excluding Chapter 11. At the end of each Chapter a summary of that Chapter is given including for Chapter 11. Chapter 11 focuses on keeping reasonable the design, launch and lifetime of autonomous intelligent robotic systems of any sorts, and it proposes a methodology for doing so. Finally, Chapter 12 concludes this document, and perspectives on the ongoing and future works are given as well as extended and related projects proposals.

Chapter 2

Backgrounds

2.1 State of Mind for this Work

- State(s) of mind for this work

- Perspectives

- Aspects

2.1.1 Philosophy

The aim is to develop and build a simple small but smart application - inspired from brain mechanisms and other relevant phenomena from the nature and physics themselves - as a very first step towards the elaboration of a truly intelligent thinking machine [1, 3–9] in a further future. It should not be *The Application towards The Thinking Machine*. Hence, we will not aim at finding *The Miracle Recipe* of artificial life (Alife) through *The Algorithm Development* implemented into hardware. Rather, we will aim at finding **a** "miracle recipe" of Alife through **a** set of algorithms and electronic mechanisms as artefact supports.

Indeed, any physical being needs material support to wander within our physical environment/world/universe. Our aim is to develop and use electronics with algorithms as, respectively, Alife skeleton with reflex mechanisms only, in order to found the basis of eventually a robot (or part of a robot such as the controller/brain-like part) so that it will be able to get its own automatisms/habits on top of this basis through its experience; that is, learn.

A symbiosis between humans and machines will be a higher level of machine's intelligence. Machines will not necessarily be able to do the same things as humans do; rather complementary things. Hence it is the possibility to have different types of intelligent machines in the future, which will excel each in different domains.

We hope that our application will be a good example and first step, unique, but amongst others, in the view of this great shared goal of building a thinking machine.

Hugo de Garis in his book "The Artilect War" probably refers to such a machine when he introduces the concept of "Godlike Machines". Even though this theory might be valid (what I doubt because of the inherent *life effect* - probably from laws of physics - which suggests that even two exact clones of any kind of being will inevitably diverge because it is strictly impossible for them to experience exactly the same events, just because they are not at the same place, at each moment), the most fair aim[1] to construct intelligent machines should be to have in mind, at the design stage, what sort of tasks a given machine is supposed to be assigned to. The focus for this should be on the mechatronics and physical body parts (to be specified), whereas the brain or control centre should be designed as versatile as possible.

In order to be able to raise machines with consciousness and Self of human nature, the designer would need to tackle the problem either at the life bricks level (biologists), or underneath (physicists). The former is possible and currently researched and developed but we do not see that as a real creative work, but rather just more as playing with available building blocks that we already know about their potential to yield life. The latter is far more difficult to

---

[1] Fair to ourselves because it should be impossible to implement a truly versatile machine, and also fair to the future market of "Intelligent Machine Technologies", which should be as diverse as possible for obvious worldwide economical reasons.

achieve because of too many parameters to consider, but could lead to many consciousness and Self natures, including the human one, just because this one is already made up from physics. In some respects, one can say that both life bricks and electronic bricks are made up of underlying physical bricks. But it is about bricks we are talking about, quite handy to engineer and manipulate. And manipulating electronic bricks is easier as it is more a gross manipulation, thus less accurate and possibly leading to a more restrained set of consciousness and Self natures. But because electronic bricks are anyway made of fine physical bricks, there will be these consciousness and Self emergence from matters into electronic systems that have insight intelligence, mental processes, mental faculties and exhibit Alife behaviours: E-Life (Electronic Life) would characterize living entities made of electronic systems. At human and finer granularity levels, there is a blur limit separating "natural life" or N-Life (or even biological life as we know it) to "artificial life" or A-Life. The limit itself can include cyborgs and DNA-based computer chips. But at the finest granularity level of particles, there is no limit between "natural life" and "artificial life" because this scale is lower than and up-to the biological scale of carbonate chains. At this scale, we can not consider the macroscopic behaviours of these chains that normally define our understanding of "living entities". Indeed, below the lower bound, nanobots can freely navigate between the two worlds of N- and A-Life together with molecules and atoms. Along technological progress, we shall predict that fine grained particles will be separated by a more and more apparent limit between N-Life and A-Life categories (through advances in nanosciences and nanothechnologies), whereas the same limit will disappear at the macroscopic and human level (through fusions between biology and technology). In addition, an explicit mention of NLife vs. ALife shall help enlightening people concerned with defining what is really "life". I believe that the definition of "life" is the major prerequisite to support the work presented in the remainder of this report. Later during the project, I will propose a comprehensive definition of "life", similarly to the definition of "intelligence" already included (but to be still completed), and that the reader may refer to.

Hugo de Garis is convinced that his so-called *Terrians*, who will be a part of people against the idea of building intelligent machines that could overclass humans, will take this position because they will be afraid that such machines exterminate the human species, or at least use humans as slaves. He says that on the one hand, (1) because of their far superior intelligence, these machines might treat us as we treat annoying mosquitoes; and that on the other hand, (2) we should not refrain this breakthrough because it is human destiny to achieve such revolutionary progresses, this one being a major technological advance of beings living on Earth, whether it leads to the end of mankind or not, because these machines will represent a gratifying human extension both ways.

I disagree with (1) because if we take his mosquito example seriously, we must ask why we are both far more intelligent than mosquitoes and yet sometimes kill some. If we sometimes kill a mosquito (or more) it is because we know by experience what pain it is to live with a spot (or more) several days somewhere on the body. Then, in order to prevent that, we apply a radical quick (stupid / naive / blind, though we are intelligent) solution that is to kill the mosquito ready to stinging us. But we are not killing the mosquito specie. We are just stupidly killing one that is very annoying and is not going to treat us well. We would not act this way with, say, a threatening horse. Our first attempt would be to run as far as possible, and the last decision would be to kill this animal, if we were armed. This decision would be comforted if we thought that this horse had a dangerous mental disease. The same holds for dolphins for example. We decided a long time ago to preserve many species...

With an issue like existing beings (far) more intelligent than humans, or having a complementary/different form of intelligence with/than human, it is true that this would be a first in term of how much we would consider each other important. We respect other humans more than any other animal, and killing any other animal is never as serious as killing someone. Humans distinguish to other animals by being able to raise complex and useful artefacts, reasoning, speaking enhanced languages, the list goes on... We should never kill other animals, yet nature evolved us to eat some. I believe that another kind of being (far more intelligent or not than humans), characterized by similar abilities to humans above but not of the same nature, would respect us inherently to their intelligence, as we would respect them, and as we have respected ourselves so far.

Punctual Conclusion: few local fights possible (like the ones of our past) between the *Terrians* and the *Cosmists* who are willing to build intelligent machines, but no “gigadeath”, nor war, as Pr. H. de Garis however states [18].

Hence, I agree with (2) but claim that there will not be any “Artilect War”, and if there is one, this is because there will be misunderstandings, probably from precipitated conclusions, followed by warnings such as the one of de Garis (which has promoted his publicity without doubt), that some people (even professors from all over the world) already took more or less seriously.

I state: “If you do not want to fear bad possible behaviours and threats from “artilects” towards humans, you must respect them by considering them as a whole entity, even before the very first one is created, instead of treating each of them as a mass of mechatronic components, or, at most, as an artificial brain into a distinct physical robotic body.”

Indeed, to me, the well-known Yin & Yang do not only stand in humans, but are rather ubiquitous in the nature.

Pr. de Garis should assume a similar principle, with regard to his stay in Japan for several years [19], where culture promotes *souled objects*. It is just then a question of defining where is situated the threshold of objects having a soul, from quarks to (multiple-)universes, through obviously humans.

Ideas in bulk to develop:

Electronic and Control Science Engineering and Research (as well as Computer Science) is about elaborating methodologies and automating them through implementations in electronics and in computers. Widely used modern computers are themselves made of electronics (i.e., mainly 2D physical silicon surfaces designed with EDA tools).

...

Occam's Razor principle: "Keep things simple!" Or my way of saying it: "Don't elaborate and automatize methodologies just for the sake of it, neither to show off."

KISS: Keep It Simple Straightforward.

An (intelligent) automaton should not only perform automated tasks for you, but help you taking good habits as well as itself...

Cognitive scientists have a strong common point with hairdressers, there will still be jobs for them to do for a very long time. Indeed, the so-called daunting problem [20] which can be stated as: "How (far and precisely) maturated thoughts from brain(s) can explain seeds of thoughts themselves?" should not be seen as a daunting problem really, but rather as the opportunity (thanks to some laws of physics maybe) to be able to endlessly study brains/humans. [Here my theory of "Beneficial but tricky Side Effect of Research in Cognition": observed observer as the brain being the object of study.]

[21]

2.1.2 Context of this Research

[Points of view, justifications for why we have these views.]

A self-reconfigurable system is powerful in the sense that it is ideally able to reconfigure itself in order to potentially fully adapt its behaviour to any unusual, unpredictable, unexpected nor unknown situation it has to face at any time. That is, taking the best possible appropriate course of actions[2], eventually including accommodating necessary parts of its environment. It is assumed to do this thanks to prior knowledge and experience (i.e., the more taught and (broadly) experienced, the more clever and adaptable). Therefore, in order to increase its knowledge on

---

[2] We must notice that this appropriate action may in its turn be unusual, unpredictable, unexpected nor unknown. This is the price to pay - losing some control on the machine -, but if it is not acceptable, we must then change our aims: getting good real-time control as much as possible, without entrusting more (hazardous) control than required to the machine. To avoid such worries, robots should be autonomous (no human or extra device required for their operation), independent (able to operate with only little help from humans), but especially interdependent in the new societies composed of humans and robots interacting all together.

top of initial native knowledge, the system must be able to learn. In order to improve, this should include learning how to learn, the so-called aptitude of *meta-learning*.

In our view, the quality of the available physical development platform will direct our choice on the order of the system's initial native architecture and knowledge [architecture bits = genome]. Next to this, we assume that the more fine the platform's granularity (level) is, the more versatile and immediately (with less prior knowledge required) adaptable the system will be, but the more probably it may "explode" (soon after its power up) and the slower it may usefully grow and develop seeking good compromise.

My initial point of view for developing intelligent systems is to start with an experimental development platform of a given physical granularity level that can range from atomic size, through logical gates, functional modules, cerebral states, and biological life bricks.

Quality of Physical Platform: potential performances,
Order of the System's initial native Architecture and Knowledge: based on the hypotheses on ability of the system to properly react to situations according to its potential performances, the initial native architecture must include more or less artifacts, which impact on the initial system complexity, and the initial amount of knowledge must be more or less large.
Granularity level of the Platform: the more the quantum phenomena are taken into account and/or the closer to atomic size are modeled the subsets of physical property of the hardware, the finer the platform grain.

Self-reconfiguration does not come without mentioning self-organization. Very first ideas stemmed from possible approaches at high-level of conception in order to implement self-organization using COTS (Commercially Out Of The Shelf) devices such as FPGA, and appropriate tools to "program" the microarchitecture and its algorithms (in hardware) such as the SystemC library of C++, a Hardware Oriented Programming Language compilable simply using gcc.

Looking forward to a minimalist solution, I must admit that I am looking for a miracle recipe for the seed of a universal and versatile, highly (self-)adaptive system.

Designing hardware has always been a tedious task, much more difficult and risky than developing software. In the paradigm of ‘reconfigurable logic’ it is now possible to undertake non-risky hardware design works using pre-prototyping methodologies. But the current available tools (that must handle hundreds I/O pins, but most importantly hundreds of thousand internal logic gate connections) for hardware design capture, and the arbitrary knowledge of a given human expert designer or team still leave too many efforts to deploy for achieving field-programmable gate array (FPGA) design products along reasonable manpower-time. Hence, means are currently researched to allow for simple straightforwards design capture fashions and design optimizations.

Within the traditional top-down design approaches, researchers try for instance to add UML - as a means for design capture - on top of sophisticated Hardware Oriented Programming Languages such as SystemC, sometimes also classified as HDL (Hardware Design Language). But bottom-up approaches as in this thesis seem to give a new breath to this concern. EHW (Evolvable HardWare) can improve an initial architecture design by using GAs (Genetic Algorithms). It does so by evolving populations of design. At the end of evolution, only the most appropriate design is retained for a particular application as specified right from the start. A fitness function tailored according to the problem to be solved assesses each design candidate along the evolution...

The next chapter reviews this and much more from the literature.

Chapter 3

Literature Reviews

3.1 Machine Learning

As the discipline of Machine Learning [2,22] is extremely wide, this section will aim at briefly reviewing well-known (most used) techniques and will only focus on details of those which are relevant to the next section about "Learning in Hardware"; that is, which lend themselves to hardware implementations.

3.1.1 Prior Work on Machine Learning

Many different machine learning (ML) techniques have been recently developed. Genetic algorithms quickly evolve encoded problem solutions along natural selection-inspired processes and can reach a near-optimal solution in many applications, decision trees are used in knowledge engineering and data mining applications, expert systems can help tackle specific problems that require comprehensive knowledge, and fuzzy sets can calibrate vagueness, as required for natural human language processing.

With its strong inter-disciplinary characteristics, "machine learning is the study of computer algorithms that improve automatically through experience" [2]. The main goal of such algorithms is to increase the machine's knowledge, while other goals are to allow the machine to achieve tasks based on the recorded knowledge (i.e. the experience) and on the currently perceived surrounding information, and perhaps ultimately to allow the machine to think [4,7] in order to permit the solution of complex problems that would not normally be possible to tackle.

Dynamically adopting appropriate behaviours is characteristic of intelligence, but is (obviously) not the whole definition of what intelligence is. For instance, thinking activities may often outclass physical activities [4], as developing strategies is a wise prerequisite to beginning any adventure, along which strategies must not only be developed, but applied too [23–28].

3.1.2 Decision Trees: DTs

Parallel search/processing.

3.1.3 Distributed Intelligence: DI

Intelligent Agents, Holons, Multi-Process Units, Multi-agent systems.

3.1.4 Inductive Logic: IL

Inductive logic programming is an interesting approach that allows for digital hardware [29] to be rewired as it reasons according to the environmental information taken into account by the initial programme.

3.1.5 Fuzzy Logic: FL

Fuzzy logic is usually mixed with other machine learning techniques in order to calibrate vagueness.

3.1.6 Combinations of the above: Cos

Combining ANNs [30].

Cellular Computing [31,32].

Cellular Neural Networks [33–39].

3.1.7 Embryonics

[40–42]

3.1.8 (Adaptive) Expert Systems: ESes

Knowledge-based Systems.

Parameter-based Estimation: PE

Perception-based Systems: PSes

Predictive Control: PC [43]

Statistical Methods: SMes

Computing with Words: CWs

3.2 Applications

Advantages and drawbacks of each major type of MLA.

Solving $\mathcal{NP}$-hard problems

Real-Time / Real-Life Control

Predictive Control (Anticipation, Emotions-aware)

Generation of rules (learning rules, evolutionary rules: defining an "overall goal")

Machine Learning devices as effective and efficient electronic design tools

Platforms, (Self-)Reconfiguration, FPGAs/DPGAs => real-world applications

3.2.1 Pattern Recognition and Fault-Tolerant Systems

[44]

Image Recognition

Musical Pattern Recognition

Continues brain handling of specific musical time periods (not necessarily patterns), putting the listener in a certain mood (on top of Major and Minor modes considerations)? Timing Strategies (like for films making)? Can lead to Automated Musical Compositions. Is music luxury or vital? (Both, and the loop is looped.)

Knowledge in Speech and Recognition

Language Processing

3.2.2 Robotics

Hyper-redundant morphing robots [45–53].

Robot Navigation

3.2.3 Evolvable Hardware

[54,55]

3.3 Machine Learning in Hardware

3.3.1 Prior Work

Several *Learning Hardware* [56] approaches have been investigated and reported on in the literature [57]. The recurrent technique is to implement neural networks into FPGAs, but different types of network, different methodologies for the hardware design flow and different techniques for the neural signalling have been employed. This brings already a large set of diverse interesting results. Furthermore, other researchers have worked along even more exotic methods for the construction of learning hardware machines.

This section details major successes but also some defeats in the emergent, although still infant, learning in hardware domain that the present PhD work fall into as well. It gives advantages and drawbacks of each approach from the results and experiences of designers. These naturally reflect to which application a given approach is particularly suited.

Note: there is a distinction between EHW and NNs in HW.

Neural Networks in Hardware

Kak neural network: [58]

Conventional Learning Algorithms in Hardware: "Although the **hardware acceleration** of the traditional learning algorithms permits some parallelism to be exploited [59,60], the whole training process still remains iterative and empirical. Furthermore, the learning algorithms are not hardware friendly because of the ***costly multiplications*** needed for computing the **dot-product** between the weights and the inputs to at least integer precisioned granularity. Due to their complexity, a faithful realization of the learning algorithms in FPGA-based reconfigurable hardware **leads to huge designs** which often do not fit onto the limited hardware resource. Also, such implementations are **difficult to scale-up** with increased precision and neuron counts. Although implementations of neural networks based on bit stream arithmetic [61,62] simplify the costly multiplications, **appropriate learning algorithms have yet to be devised to support *on-chip learning***. As a consequence, training must be done off-line."

ITNN: Instantaneously trained neural networks, inspired from a biologically plausible mechanism for short-term memory in which biological learning occurs instantaneously.

“The Kak algorithm (*prescriptive* learning algorithm) is up to 200 times faster than the backpropagation algorithms with comparable generalization capability in pattern recognition and prediction tasks [63]”

Unsupervised learning, or self-organised learning, or self-organising neural networks and self-organising map: [64].

Unsupervised Hebbian Learning: “Unsupervised learning algorithms aim to learn rapidly. **In fact, self-organising neural networks learn much faster than backpropagation networks, and thus can be used in real-time.** Self-organising neural networks are effective in dealing with **unexpected and changing conditions:** Hebbian and competitive learning. Competitive learning [65]. Hebbian learning [66]

Kohonen learning is also unsupervised: [67]

Unsupervised or Self-Supervised Learning: [68]

‘SOLAR’ is one of the most interesting machine learning technique found in the literature so far, and which is similar to the present project on the aspect of self-(re)organization of neural network [69–71]. Also, SOLAR has recently been applied to power quality classification [72–77].

Cellular Automata in Hardware

“A literature review discusses the implementation of cellular automata using associative memories as the state machine. Some historical context is described, as are associative memories and brief notes on hardware implementations” [78].

Cellular Neural Network in Hardware

Very inspiring CNN (cellular neural network) chips have been developed [79–83]. Also, FPGA implementations of CNN have been carried out, such as [84].

Evolvable Hardware is (Self-)Reconfigurable, but Learning Intelligently?

According to a general trend, most researchers in AI get inspired from nature to come up with new techniques. One, which directly relies on *Natural Selection* (NS) along evolution [85], has been applied to hardware design [86] and uses the aforementioned Genetic Algorithms (GAs) to accelerate and optimize the automated design process of complex systems. When electronic logic became programmable or rather (re)configurable thanks to commercialised programmable logic devices (PLDs) such as FPGAs, some made a link between those two techniques and had then the idea of evolving hardware [87–89], later termed evolvable hardware (EHW) [8,90–99].

This approach initially involved GAs to evolve electronic circuits in an automated, optimizing and accelerating manner, either in the goal of designing a complete circuit, or to tune circuit parameters of a predefined architecture. In both cases, evolution of the circuit is performed by viewing the configuration bit-strings of the FPGA as chromosomes. Different GA techniques can be applied to accelerate evolution of populations of chromosomes that represent parts of the circuit to design and/or tune. The means of the designer to get a particular solution are then to define a corresponding fitness function that the directed driven blind GA processes must fulfil (dictatorial policy).

A fitness function f is useful to simplify the representation of match between the obtained solution output(s) from the GA and the desired target output(s) from the problem specification. When there are no matches, $f = 0$; when there is perfect match, $f = 1$. Within the GA itself, the individuals are evaluated by such a fitness function, and the ones that fall below a defined threshold are discarded, whereas the others are selected for crossover (to generate the next population) and mutation (to allow larger exploration of the search space than only carrying on the current local search).

[44,100–102]

“Modeling Nature may not lead to the Best Technological Solutions

The field of evolutionary computation has followed a long practised tradition of looking to nature for technological solutions. The imitation of bird flying by mythological Icarus and Daedalus, or German aeronautical engineer Otto Lilienthal who build flying models employing flapping, bird-like, wings are early examples of such efforts. In seeking technological solutions, the ‘imitate-the-nature’ approach frequently does not lead to the best engineering results. Modern examples of successful solutions not imitating nature include automobiles, airplanes, rockets, computers, etc. [103]

From the overview of his LEM algorithm, one can see that R. Michalski wants somehow to incorporate learning in the process of evolution. His approach for guiding evolution through learning is not the one we can observe as implemented in the nature, but rather the one undertaken by geneticists. In the nature, the Baldwin effect applies when a phenotype individual modifies (hopefully improves) its environment, which will also be the one of its offsprings. This mechanism influences evolution by performing a transfer from phenotype learned traits to the environment of this phenotype, later inherited by the next generations that will have to accommodate, and then learn part of these traits (just part because of loss of knowledge quality within the environment) in their turn from this environment that they inhabit in their turn. This is an external mechanism to the evolution of phenotypes, as fair as the one found in the nature.

However, Michalski seems to try again to enable individuals to influence evolution, but using Machine Learning techniques to select individuals and to produce new ones, that is, to use a straightforward control on evolution (within genotype-phenotype mappings), in a geneticist manner, but not a la Baldwin (indirect phenotype-genotype mappings, or more correctly, phenotype-phenotype mappings). This is an internal mechanism to the phenotypes, subject of ethical issues when it comes to mankind considerations: it is world-widely prohibited to do that on humans because of thoughtful predicted future critical impacts; the same should hold for any other kind of individual, even computed ones like here problem solutions that might end up to degeneracy of solutions, that we do not want; we want a solution as good as possible at the end of a GA process.

‘GMOs’ in GAs: Genetically Modified ‘Artificial Organisms’

Intellectual evolution, according to Pr. Michalski, is rather knowingly modifying production of phenotypes, like in the life science where *learned* geneticists can now, using **artefacts**, choose the sex of a future baby to come, or even knowingly interfere in order to get an individual particularly strong, and promised to be smart in many respects. Unfortunately, this may probably lead to degeneracy at some point.

Intellectual Evolution: to me, this is more like when a given generation creates a new technology from ancient knowledge, and let this technology for the new generations to play with in their turn, a la Baldwin.

Of course, R. Michalski highlights that modeling nature does not always lead to good engineering solutions, which is partially true. But what he does is modeling nature for the GAs, and applying this LEM artefact to it. Applying such **artefacts** to nature itself is most likely to lead to degeneracy as stated above. Why should it be different in nature-inspired GAs?

Furthermore:

- bird: chemical $->$ mechanical energy, and bird’s brain = very complex control centre
- airplane or helicopter: thermal $->$ mechanical energy, and pilot commands (+ now board computers, that for example can foresee air holes, like birds do) = simple control command devices.

Hence: be fair and faithful when claiming to inspire from nature $->$ knowingly do it by ensuring a **consistent approach**...

e-lem:

Failure of e-LEM in H. de Garis project [104]: because of the above...

Self-Organization is clever:

Clever compared to naive GAs (where experience is neither understood nor learnt, but only blindly undergone and therefore naïve), because self-organization has local rules that somehow make sense as an emergence of intelligent complex structure at the **'human-understandable logic higher layer'** from the **'ruled cellular lower layer'**.

SOM (self-organizing map) is amongst the most popular structures of self-organizing neural network [105–113].

An advanced SOM algorithm has also been reported: LISSOM [114].

Spike-timing dependent weight change / learning rule: [115]

Spikes are not time-dependent in a human brain in order to overcome bad medium quality. Nature evolved us in this way (instead of trying to improve the medium quality). We can have good medium quality in silicon, and therefore time-dependent spikes, which will lead to higher neural network training performances. This may comfort the idea that intelligent machines will be more per formant ("better") than humans.

3.3.2 Motivations for Novel Approaches

Conventional computers should be reserved to design and production jobs as well as experimental platforms (a), and not to be end user products themselves (b). This includes microcontrollers and indeed embedded software systems, where system crashes can also occur with high probability, whereas it is unacceptable in most real-time applications. The instruction processes and mechanisms are too prominent and lend themselves to system failure. This is inherent to CPU architectures. Because of this, computers onboard of aircrafts have to be redundant (up to five equivalent computers), and fault-tolerant warts have to be engineered, sought and added on to the whole system. Why not rather inventing and using architectures that lend themselves to cope with conditions they are intended to be immersed in?

Furthermore, from the above statements, if an artificial machine was going to be able to get an emerging consciousness, this machine would surely be not made of a software/microcontroller architecture. This, simply because if emergence of global awareness there is, mutual awareness between hardware entities and their multiple-forms of data must be; prior to, along, and after the global emergence. Indeed, data and/or their underlying bits must be continuously updated according to what the whole system undergoes. It is unacceptable to update them sequentially through the pipeline of a single microprocessor for this purpose.

Chapter 4

A Knowledgeable Authority for Interactive Hardware Design Capture Support

The aim of this chapter is to give background information basis of the PhD work. This chapter also examines the novel idea of a possible interactive hardware design capture support. This is very much needed by the electronic design automation (EDA) community and the chapter summarizes the corresponding rationale.

4.1 Introduction

Many different machine learning (ML) techniques have been recently developed. Genetic algorithms quickly evolve encoded prob- lem solutions along natural selection-inspired processes and can reach a near-optimal solution in many applications, decision trees are used in knowledge engineering and data mining applications, expert systems can help tackle specific problems that require comprehensive knowledge, and fuzzy sets can calibrate vagueness, as required for natural human language processing.

With its strong inter-disciplinary characteristics, “machine learning is the study of computer algorithms that improve automatically through experience” [2]. The main goal of such algorithms is to increase the machine's knowledge, while other goals are to allow the machine to achieve tasks based on the recorded knowledge (i.e. the experience) and on the currently perceived surrounding information, and perhaps ultimately to allow the machine to think [4,7] in order to permit the solution of complex problems that would not normally be possible to tackle.

ML techniques can also be combined. This chapter considers combining cellular automata (CA) with neural networks (NNs). CA have potential to generate complex systems from synergetic cooperations of simple building blocks, whereas NNs are inspired from interconnections of brain cells and have application in a wide range of complex problems.

Cellular Automata (CA) have properties that make them suitable for bottom-up design [6], including the generation of emergent NN structures. In this chapter, we propose an innovative hardware-based CA that develops asynchronously to form NN modules that also learn in an asynchronous manner. Previous hardware implementations of CA found in the literature are synchronous, despite global clock distribution becoming impractical [116] as the number of interconnected cells is increased. To alleviate this problem, we propose that cell operations are triggered not by a distributed global clock but rather by their immediate neighbours. A second novel aspect to the current work is that each emergent NN has access to the low-level rules of the underlying CA structure, allowing the NN to both learn about and modify them using a trial-and-error approach, and thereby permitting self-management of internal resources, self-specialization of NN modules and self-improvement at run-time and full speed of field-programmable gate array (FPGA) [117] hardware.

In highly distributed, massively parallel reconfigurable machines (such as CAs), spontaneous organization and emergence of intelligent behaviours can occur [6,70]. In the paradigm proposed in this chapter, the self-management of resources is autonomously elaborated and knowingly aggregated to the system during its bottom-up deployment from an initially designed architecture. This results in a particularly robust solution, while minimizing the human design efforts. Further- more, this redundant architecture lends itself to partial self-testing and also to self-healing made possible by self-replacement of failing cells with neighbours like in [98]. Due to their high degree of plasticity and self-adaptive reconfigurable properties, suitable combinations of these application-independent ready-to-specialize NN modules have application in complex control problems where adaptive real-time incremental learning is required, such as for dynamic intelligent control of hyper-redundant morphing robots that currently lack a suitable method for on-line learning (or indeed any form of control) [45].

Dynamically adopting appropriate behaviours is characteristic of intelligence, but is (obviously) not the whole definition of what intelligence is. For instance, thinking activities may often outclass physical activities [4], as developing strategies is a wise prerequist to begining any adventure, along which strategies must not only be developed, but applied too. Yet, for convenience purposes, also permitting effective demonstrations and performance evaluations, in the future work we will implement our combined CA-NN hardware system as the core controller for robot navigation tasks. Regardless of the robot mechatronics, we will be interested in observing how quickly and to what extent the robot can re-adapt when its environmental conditions change. Cases where a physical part of the robot involved in motion is altered, such as a joint or wheel standard angle, will be of particular interest. As self-adaptating to unusual, unknown and/or unexpected situations is the limit to creativity, the initial system will need to be expanded (on multiple chips) and incorporate auto-associative memories [78, 118] that will be expected to generate commensurate original solutions at run-time when facing a new situation. A larger system implemented on multiple chips can exhibit additional mental processes and thinking activities. These are ideals that constitute a research direction to wich we aim to contribute. In addition, we will also try to contribute to the cognitive and biological science disciplines, reciprocating information we glean from the natural phenomena they describe.

In the literature, a number of authors have described combinations of CAs and NNs, termed ‘CA-based NNs’ [36, 79, 83], although the dominant area of application has tended to be visual computing [82]. CA-based NN implementations generally employ GAs to evolve units (for example, CA rules) or entire systems (for example, NN structures) encoded in the form of chromosomes with evolvable hard- ware (EHW) approaches [8, 98] for exploiting the reconfiguration features [119] of platforms such as FPGAs [120–123]. In contrast, we propose that the systems and their constituent units incrementally progress, gain experience and improve knowingly through interaction, rather than being (wholly or partially) evolved by a dictatorial fitness function driven by a GA, such as in [79] and in [124].

FPGAs combine the flexibility of software and performance of hardware (in terms of speed and robustness), where distributed processing units truly operate concurrently. As systems grow in complexity, there is a huge need for features such as self-management of system resources. Unfortunately, conventional computing systems have to incorporate this feature as an additional software application that is cumbersome or impossible to maintain. However, redundant cellular systems are inherently scalable and self-management is straightforward to design and implement at elaboration time, and to maintain at simulation/run time. NNs also lend themselves to self-reorganization and can be easily restructured at run-time when formed by a hardware CA. Finally, in NNs the knowledge is distributed, which ensures better fault-tolerance and handling of indeterminism. However, along the development work, we will also endeavour to comparing the results of different applications with other systems that could rival.

The remainder of this chapter is organized as follows. Section 4.2 details the architecture design and gives an example of algorithm for the cellular operation of the system. Section 4.4 presents the methodology adopted along the work and shows early simulation results of the CA forming a NN. In Section 4.5, we propose a particular application for our system, and perspectives on the ongoing and future work are given. Section 4.6 concludes this chapter.

4.2 Design Specification

In the realm of a CA, the counterpart of a top-down engineered solution is a bottom-up scientific seeding recipe that includes the number of cells (parametric), the CA topology (here square cells with adjacent CA edges connected), the set of cell states (type, neural connection), the initial conditions (random type, all cells neurally unconnected), the neighbourhood distance (here unity), and, usually the most difficult one to engineer, the set of rules (or laws) to govern the CA process. Together, these parameters, summarized in Table 4.1 influence the direction of progress of the CA, that is most sensitive to the rules meant to govern it.

In our case, the CA is meant to emerge NN structures. Operations are asynchronous, and designing and synthesizing the system is facilitated by the use of SystemC and techniques such as the ones presented in [125], where “specifications can be expressed at a level in which asynchronous and synchronous designs are indistinguishable”. In addition, our system can benefit from the increasingly simplicity and speed of asynchronous circuits [126]. Finally, we adopt a spiking neuron approach [124, 127] to take advantage of the asynchronous nature of the circuitry.

4.2.1 Architecture Description

The initial idea behind a ‘self-(re)organizing cellular automata – neural network dual system’ was twofold. Firstly, a CA can reach useful landscapes [6], provided a ‘good’ set of rules is available (a scientific seeding recipe very difficult to generate entirely [8]). Secondly, NNs are particularly good at extracting and mimicking rules. They can also classify and associate patterns, subsequently creating and generating new patterns, which in our case are the new CA rules that we need. The NN can restructure (itself) due to the reconfigurable CA that supports and forms it. Hence, the hardware system proposed can be seen in two aspects: a CA architecture that provides a good (self-)reconfigurable platform and a NN that implements a self-organizing learning algorithm similar to self-organizing maps (SOM) [128] or SOLAR [70]. At a higher level of abstraction and considering both aspects, there should be another self-organizing mechanism: the NN that learns the rules of its own underlying CA, and amends them in order to provide improvement in the light of the current situation faced (cf. ‘conditioning’). Indeed, these (asynchronous) cell rearrangement phenomena successfully occur in nature, such as in brains.

4.2.2 Cellular Automata Aspect

One way of viewing the initial architecture of this system is to consider that each NN module is a ready-to-specialize NN, whose fundamental component is the state of a cell. A cell, shown in Figure 4.1, is an individual entity amongst a population of cells that form an underlying CA to the NN. Clearly, the cell type state defines the type of the corresponding NN component, namely one of neuron, dendrite, synapse, axon, or unused.

Our system is different from CNNs, SOM, SOLAR and [124]. These are minimalist NNs, meaning that each cell is a neuron only. Our architecture can be considered as a superclass of these, because it can be configured as a CNN or as a SOLAR for example, but also as any NN topologies based on the five brain-inspired building blocks: neuron, synapse, dendrite, axon or empty space. Moreover, these architectures differ from the CAM Brain Machine [8], where, for example, multiple cells are needed to form a single synapse. As in SOLAR, our cells are totipotent cells (each and every cell has the same potential to ensure the selection of a function from a set of predetermined functions), but they are not limited to ensure a particular neuron's activation transfer function only (whether excitatory or inhinbitory). In our model, a cell type is selected prior to its function. According to type, a cell will then further select and ensure a function. For example neurons can have a sigmoid, a sign, a step, or a linear activation transfer function, etc. They can be excitatory or inhibitory.

A dock state can be exclusively unconnected ($dock_{cardinalpoint} = 0$), inward ($dock_{cardinalpoint} = 1$) or outward ($dock_{cardinalpoint} = 2$). Neurons and synapses have a single input and a single output, whereas dendrites can have up to three inputs and axons up to three outputs. When a rule or a neural signal invokes a change in a cell, the neighbouring cell parameters (for example, in Figure 4.1 $type_{state}$, $settled$ and $dock_{east}$ of cell W) are passed to the cell through a permanent SystemC sc_{signal} channel for control communication, and are received as an object (for example, in Figure 4.1 $west_{state}$ of cell C) by the cell. With this updated information about its neighbours, the cell is ready to trigger, that is, to change its type state and/or dock state, and subsequently inform its relevant neighbours (from two to all four) about its new configuration in the same

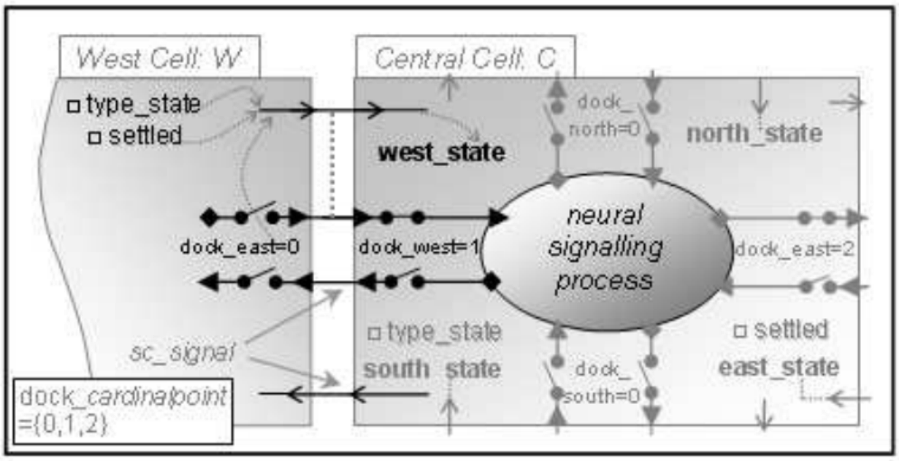


Figure 4.1: A cell C and its internal parameters, showing the connection with its neighbour W to the west. The solid arrows are the neural connections, while the remainder represents permanent CA control communications.

asynchronous manner as above.

Table 4.1: Cellular Automata Ingredients

Ingredient	*Number of cells*	*CA topology*	*Sets of cell states*	*Initial conditions*	*Neighbourhood distance*	*Set of rules*
Attribute	Parametric	Edge-connected square cells	Type; Neural connection	Random type; All cells neurally unconnected	Unity	Basic; Constraints; Growth; Application; Learning Algorithm

The set of possible neural connections between cells dramatically increases the total number of cell states to 81 as illustrated by Figure 4.2, but dock states cannot be amalgamated into type states as it would lead to considerable loss of meaning.

To make apparent the potential complexity of the system that can be produced, the previous example of 81 cell states, each with four neighbours in addition to the cell itself, gives rise to 81^5 (almost 3.5 billion) patterns in total. Due to ‘dock state’ incompatibilities, not all patterns are neurally valid, and far fewer patterns need be considered in practice. For example, the top left axon state on the grid shown in Figure 4.2 is neurally compatible with seven axon states or three synapse states (10 states in total) with respect to each of its North, East and South neighbours, as well as twelve axon states or three neuron states on its West. The total number of possible

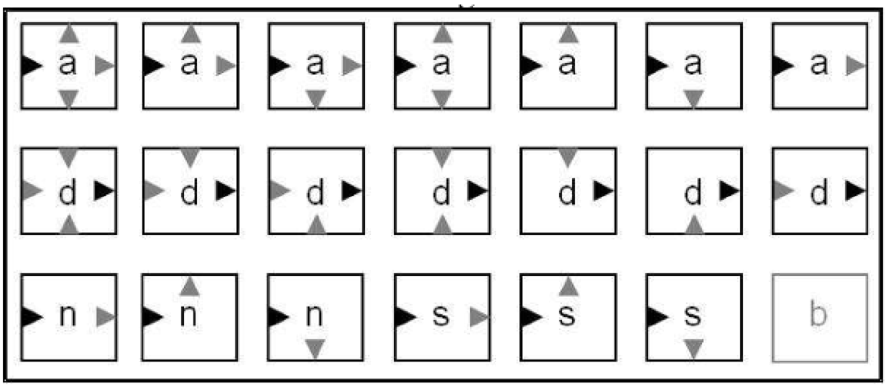
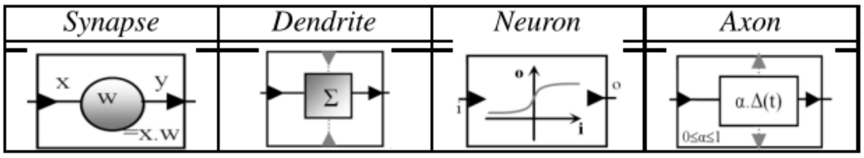


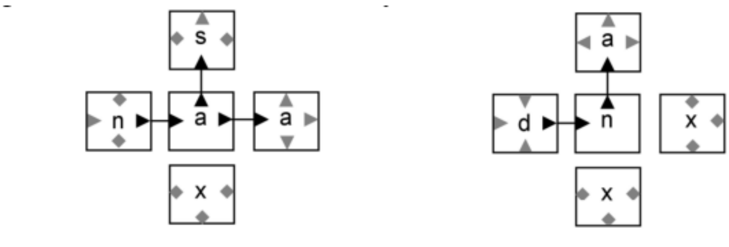
Figure 4.2: Illustration of total possible cell states. There are actually four versions of each of these 20 functional cells when taking into account the three other cardinal orientations of the black arrow (showing dock state of neural connection; permanent CA control connections are not shown), here from West to East. Hence, a cell can be in one out of 80 states, plus one state as unused (blank) cell: total = 81 possible states. In the figure, n=neuron, s=synapse, d=dendrite and a=axon.

patterns is thus 1*10*10*10*15 = 15,000 (where the '1*' is used to indicate we are looking at one cell state amongst 81, here the top left axon state), which, while being far less than $1 * 81^4$, still requires far more transition rules than would be feasible for manual coding. Of course, once all cells in the grid are taken account, it is apparent that many more patterns are possible; in fact the total number would be in the order of 10^7.

Figure 4.3.a shows the second axon of the grid (cf. Figure 4.2) as being the central cell of a pattern. If an event occurs, such as the synapse undocking and the neuron changing its type state into dendrite, a basic rule should dictate this central axon to become a neuron connected to the dendrite to its West and to the (other) axon to its East. As this example demonstrates for our CA-NN model at the pattern scale, a pattern has a very limited number of choices for the new state of a central cell. In general, there is no decision to make for the new state as only one solution is neurally compatible. This allows for a manageable set of rules (mainly one or two transition rules per pattern), yet permitting sufficient CA flexibility at a higher scale than pattern. Figure 4.3.b shows the second neuron of the grid (cf. Figure 4.2) as being the central cell of a pattern. This example clearly demonstrates that some central cells, such as neuron here, are concerned with two neighours only (instead of three or four), which further reduces the total number of transition rules to consider compared to the one indicated by the above calculation.

Table 4.2: Neural signalling process depending on cell type states, with abstraction of cardinal orientations.





(a) (b)

Figure 4.3: Generic cellular pattern examples, with an axon state (a) and a neuron state (b) as centre cells.

4.2.3 Neural Network Aspect

As summarized in Table 4.2, according to a cell's type state, the neural signalling process of this cell acts either as the activation transfer function of a firing neuron, the scale factor (weight) of a synapse, the input sum of a dendrite, or the propagation delay, attenuation rate and output distribution of an axon.

In addition, the four cardinal dock states, represented as switches in Figure 4.1, indicate to the same process how to direct the neural signal.

The raised NN will have full learning access to all the rules in order to gain its own representation of partial sets of CA rules. However, it will have permission to inhibit only the soft rules so that it can virtually modify them by inserting new automatically generated ones instead, in its own neural representation (e.g. instead of IF-THEN rules pre-coded in SystemC). Finally, the NN will try to discover crucial missing rules, experiment with other rules to seek to improvements and be allowed to add them to its own set and apply them. To enable improvement (against any job being achieved), we will consider developing strategies to incorporate on- chip 'improvability evaluation'.

4.3 System Behaviour

The set of CA rules included in the current system are as follows.

- $basic\ rules$ dictate that, in a state, transitions to a new state are neurally compatible with at least two neighbours;
- $constraints$ are dock-state transitions that satisfy the above architecture description of inputs and outputs, and comply with consistent NN connections;
- $growth\ rules$ cardinally prioritize neural connections between cells and guide the CA towards an effective NN structure;
- $application\ rules$ specialize a NN region or module to perform a particular task;
- $NN\ self-organizing\ learning$ rules select – based on data from both currently perceived surrounding information and recorded events (including activities of lower level CA rules for the NN to extract them) – a NN topology and learning algorithm to train weights, propagation delays, attenuation rates, activation transfer function and thresholds.

The interconnection of CAs should be able to generate the appropriate NN sequence, as given by:

$$sysnapse- > dendrite- > \{dendrite- >\}neuron- > axon- > \{axon- >\} \tag{4.1}$$

here with abstraction of input/output multiplicity and cardinal orientations.

4.3.1 Cell Behaviour

Each cell incorporates an identical five-step algorithm that implements these rules. Step 1 is type-state transitions, step 2 is dock-state transitions and single inputs/outputs that must have priority over multiple inputs/outputs, which are solved in step 3. Step 4 settles a cell when operational and step 5 triggers the relevant neighbours if an update occurred in any of steps 1 to 4.

Algorithm 1 [Example of basic Top-Level Algorithm for a cell C]

```
When a cell $C$ is triggered or self-triggered (respectively because a neighbour cell has just settled, or $C$ could not settle even after a finite number of attempts to trigger neighbour cells){
0) if $trig < persist$ AND $settled = true$, $trig++$; return 0;
1) $C :=$ new type state based on $C$ current type state and on those of its four nearest neighbours; $trig := 0$;
2) $C$ tries first to resolve its single input (for axons and neurons) or its single output (for dendrites and synapses);
    a) Single inputs will try to dock in priority to the west neighbour, else to the north or south ones with a random probability of 1/2 for each one, else:
        • if $C = axon$, try to dock the single input to the east neighbour provided it is also an axon, not a neuron,
            if unsuccessful, assign a constant value to this input;
        • if $C = neuron$, call a subroutine that, based on more detailed parameters (such as detection of endless
            loop in this step 2), either assigns a constant value to this input, or makes it available to the modules
            external interface, or performs $C := synapse$ and jumps back to 2;
    b) Single outputs will try to dock in priority to the east neighbour, else to the north or south ones with a random probability of 1/2 for each one, else:
        • if $C = dendrite$, try to dock the single output to the west neighbour provided it is also a dendrite, not a neuron;
        • if $C = synapse$, call a subroutine that, based on more detailed parameters (such as detection of endless
            loop in this step 2), either makes this output available for later or to the modules external interface,
            or performs $C := neuron$ and jumps back to 2;
3) $C$ tries to resolve its other end(s) as either multiple/single output(s) (for axons/neurons) or multiple/single input(s) (for dendrites/synapses);
    c) Similar to 2.b but deals with a multiplicity of ports;
    d) Similar to 2.a but deals with a multiplicity of ports;
4) if $C$ is fully connected and operational, $settled := true$, else $settled := false$
```

5) if at least one update has been performed through 1 to 4, the neighbour cells concerned are triggered and receive updated information about C: new type, docking, settling, etc; }

What Algorithm 1 does at a given stage is always justified, such as in the example of axons that obviously need to dock their single input prior to their multiple outputs, which have indeed more chance to be resolved, even later on. If docking an input was impossible, then assigning a constant value in the manner of a neuron's threshold is feasible. More importantly, inputs and outputs that could not dock to any neighbouring cell are allocated to the peripheral I/O ports, permitting interaction with actuators and sensors. The inner data- driven system operates asynchronously, but this does not necessarily mean that, for instance, a sensor's input data are not sampled along a clocked input port. Indeed, real-time visual data can be captured in the manner of eye saccades, sampling on average three images per second. This peripheral clock does not have to be (extensively) distributed and will not affect the performance of the system.

Also, as each cell runs this algorithm as an asynchronous process concurrently with the other cells (along CSP methods), numerous basic rules may be self-resolved (without having to insert them neither manually, nor automatically) due to beneficial asynchronous parallelism effects (cf. theory of parallelism: the CSP approach). These can even be modelled and controlled due to the availability of process description and monitoring at a high-level of abstraction with SystemC.

4.3.2 Overall Behaviour

Actually, in the above algorithm there is an idea of creating a neural signalling path globally oriented from West to East, simply in order to favour a structured organization for ease of experimentation. Variations of this algorithm include aiding a slight orientation from South to North in order to assess the eventual resulting increase of organization, and link it to any changes in performance, scalability, flexibility and robustness of the system.

One can also use constraints to force the CA into forming long chains of axons (in order to reach distant nets) and to connect to groups of dendrites (in order to dramatically increase the number of neuron's connections). These types of strategy should result in a more structured network whose operations should be easier to observe and understand.

In addition, one can for example implant 'growth phases' to this system in order to allow for further exploration of performance domains, both of the CA-NN agent itself and of the problem solution it is connected to. To do so, $persist$ can be reduced, and then later increased when changing phase again. Successive modifications of $persist$ can be pre-programmed as 'growth rules'.

The parameter $persist$ is a form of human control over the entire system that is otherwise mainly data-driven. CA rules with high level control parameters, such as 'neuron density', can be manually changed (re-programmed) inside of an (initial) architecture at elaboration time, before the system starts to operate. Thus, one can also see how a given set of rules and parameters influences the system progress, and use that knowledge to rapidly customize the system for specific tasks.

The grey dashed link in Figure 4.1 – bridging CA control communications with neural signalling connections – depicts an imaginary 'feedback loop' that will allow the NN to learn about the CA and adjust its operation. Far more sophisticated, but analogically similar to an op-amp circuitry with feedback loop, we believe that sets of appropriately tuned initial parameters and rules exist for such a dual intelligent system to become and remain stable (cf. equilibrum conditions: [36]) and further self-improve by discovering and elaborating new learning mechanisms by itself.

Finally, at the global level, with a sufficient number of cells in the system, local errors from inappropriate cell transitions should not convert into overall (or even local but significant) faults. Instead, they might contribute to exploring new pattern associations along in an 'error-based learning' fashion.

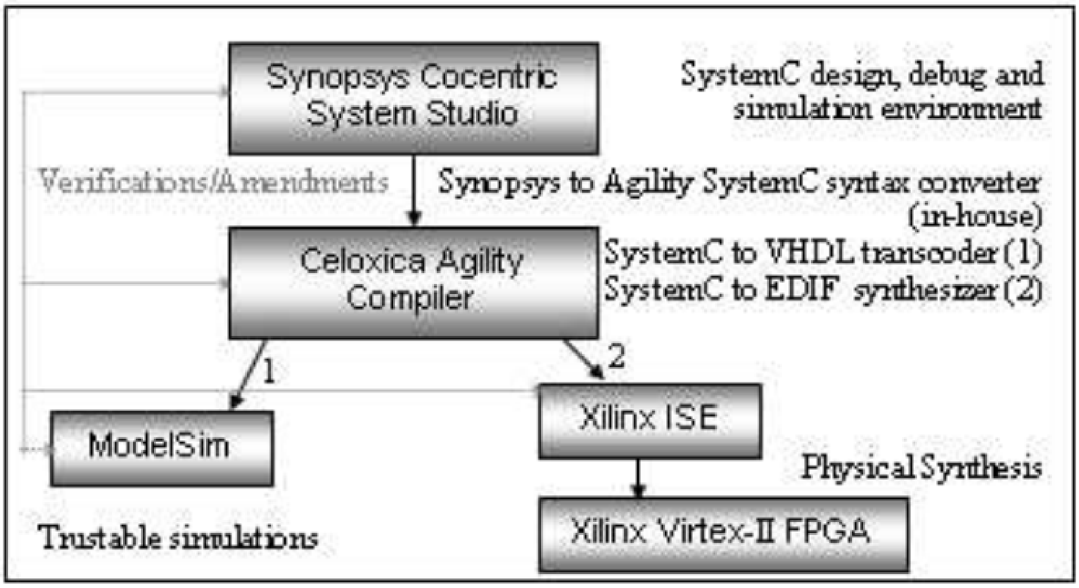

Synopsys Cocentric System Studio
SystemC design, debug and simulation environment
Verifications/Amendments
Synopsys to Agility SystemC syntax converter (in-house)
Celoxica Agility Compiler
SystemC to VHDL transcoder (1)
SystemC to EDIF synthesizer (2)
1
2
Xilinx ISE
ModelSim
Physical Synthesis
Trustable simulations
Xilinx Virtex-II FPGA

Figure 4.4: SystemC design flow down to FPGA for the bottom-up creation of the initial CA-NN agent architecture and rules.

4.4 Preliminary Experimental Results

4.4.1 Methodology

The current work has adopted SystemC as the modelling HDL to capture an initial CA-NN agent with few necessary but sufficient initial rules. The design flow depicted in Figure 4.4 involves the use of Synopsys Cocentric System Studio [6] design and debug environment as well as Celoxica Agility Compiler [116] for synthesising the SystemC code and targeting our Xilinx Virtex-II FPGA [117].

The advantages of using the emerging SystemC hardware-oriented programming language include its ability to provide an abstract level description of system design (transaction-level modelling or TLM), eventual reuse of C++ knowledge/expertise and algorithms (scripts), straightforward hardware/software co-design and co-simulation, ease of design capture compared to traditional hardware design languages (HDLs), high simulation speeds and the ability to perform fast simulation of some portions of the design, while allowing debugging of other portions. Here, we will also appreciate the opportunity afforded to easily evaluate our hardware implementations against software versions. SystemC cycle-accurate models of designs allow for both hardware synthesis and the production of a software exe- cutable binary file for its simulation/emulation (equivalent to software implementation in term of execution time and performance) without having to write additional scripts.

4.4.2 Simulation Results

Figure 4.5 depicts the results of a simulation of a twelve-celled CA performing steps 1 and 5 of the algorithm. The emphasised channels illustrate the possible dock states and settling that may result when executing steps 2 to 4 of the algorithm. Although only a small CA is shown here, practical CA may contain many more cells and in some cases exhibit unexpected behaviours. Although some of these behaviours will be useful, those making damaging contributions need to be sought out and eliminated.

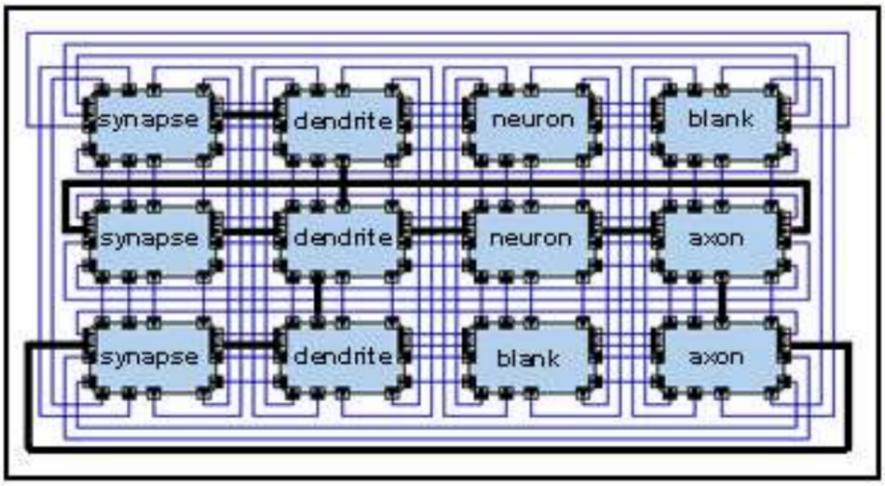


Figure 4.5: A small CA landscape structured as a NN.

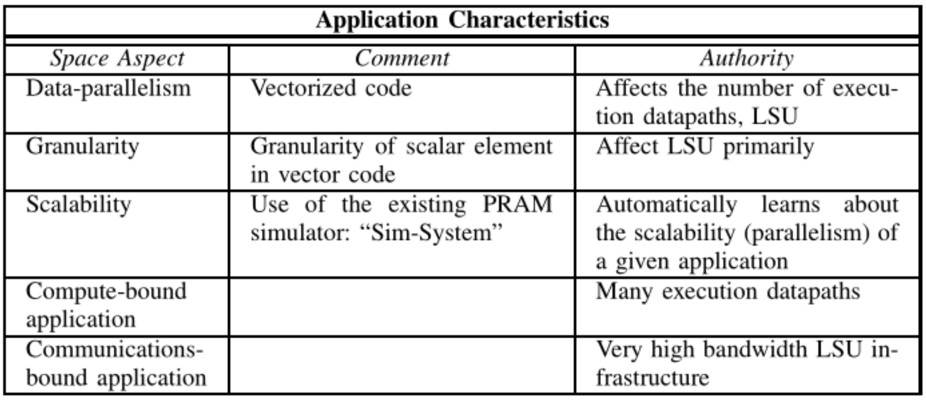
4.5 Future work

4.5.1 Proposed Application

The electronic design automation (EDA) community currently suffers from poor hardware design capture means. Traditional systems, designed along top-down approaches, are still desirable, yet cumbersome to capture manually as technologies of ever-increasing complexity are dealt with. Being versatile, our CA-NN system can be tailored to usefully and supportively interact with a given EDA tool and a designer as depicted by Figure 4.6. Such a specialization can be directed early, during bottom-up development of the CA- NN agent, by (manually) providing it with application rules. These rules are preliminarily extracted from EDA tool documentations and application characteristics, such as the ones detailed in Table 4.3. Further refinements of rules will occur at interaction time, where high-level data are exchanged.

This work will investigate the suitability of the CA-NN agent as the authority of a particular EDA tool at the stage of optimization of a given software/hardware design point; thus allowing for aware micro-architecture space pruning as detailed in Table III towards a near-optimal solution.

Table 4.3: Micro-Architecture Space Pruning.



Application Characteristics		
Space Aspect	*Comment*	*Authority*
Data-parallelism	Vectorized code	Affects the number of execution datapaths, LSU
Granularity	Granularity of scalar element in vector code	Affect LSU primarily
Scalability	Use of the existing PRAM simulator: “Sim-System”	Automatically learns about the scalability (parallelism) of a given application
Compute-bound application		Many execution datapaths
Communications-bound application		Very high bandwidth LSU infrastructure

For initial demonstration purposes, the application we are currently implementing involves training our system against an XOR gate. The system will be able to self-learn and mimic the XOR logic function. Performance in learning speed, power consumption and robustness will be

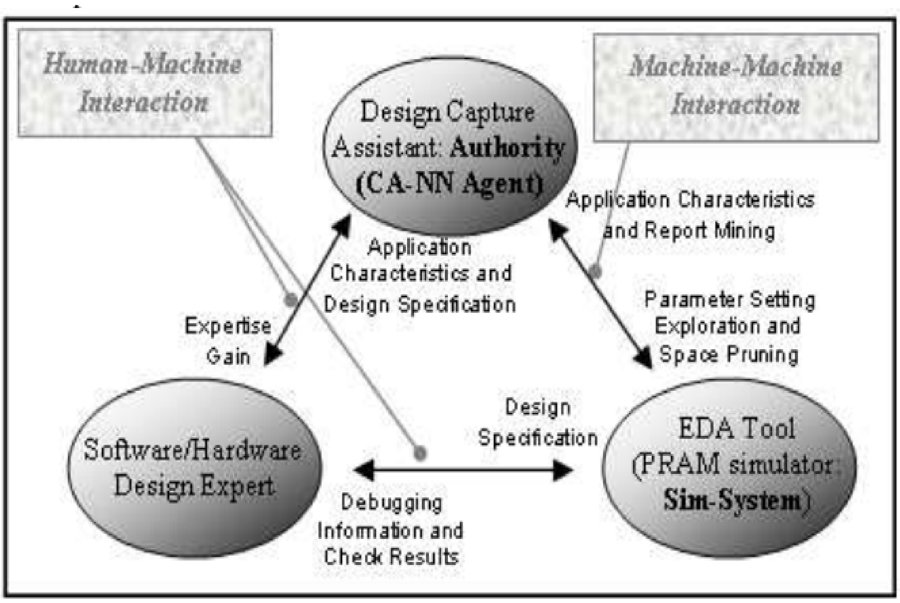

Human-Machine Interaction
Machine-Machine Interaction
Design Capture Assistant: Authority (CA-NN Agent)
Application Characteristics and Report Mining
Application Characteristics and Design Specification
Parameter Setting Exploration and Space Pruning
Expertise Gain
Design Specification
Software/Hardware Design Expert
EDA Tool (PRAM simulator: Sim-System)
Debugging Information and Check Results

Figure 4.6: (User-friendly) Interactive design capture scenario.

particularly regarded. Implementing the larger applications above will then help us focus on the scalability aspect of the system.

Unlike most of NN architectures (CNNs, SOM, SOLAR, etc), our CA-NN is not pre-structured along homogeneous neuron layers. However, neuron arrangements amenable to layering may emerge under forms more appropriate (than rigid layers) to the application the CA-NN is trained against. For instance, once the CA-NN has successfully learned the XOR logic function, there should be an arrangement of at least three layers of neuron [1] between the two inputs and the single output of this small system, while ensuring a near-optimal exploitation of the FPGA resources; this naturally due to the inherent self-reorganizing mechanisms of the system. Because, in our approach, we follow principles such as Occam's Razor, minimalist solutions are sought and superfluous layers of neurons should not be formed. Due to the software execution available under SystemC, simulation of the hardware implementation of this CA-NN and monitoring these arrangements of cells including neural layers is feasible.

4.5.2 Perspectives

One of the perspectives of this work will be to compare the performance of this system to other different machine learning techniques when tackling several tasks. The results of these comparisons will be presented in later chapters together with details of the FPGA implementation and results of our self-reorganizing learning algorithm for asynchronous spiking NNs. Presentation of the work about the effects on the system when the NN modifies CA rules will follow. Making links between local cell behaviours and emerged overall system behaviours will be investigated.

In addition, to achieve the online 'improvability evalutation', more advanced algorithms such as self-organizing learning algorithms based on local and/or global measures of information entropy will be developed. We will also consider applying finite state machine (FSM) techniques [129] (supported by our System Studio environment) to ML for this system and will also make use of Petri nets for this [130].

Finally, once a fully working system exists, we plan to relax the orientation constraint on the neuron and synapse cells so that they are free to connect in any orientation, not only from West or to East neighbours. This will dramatically increase the learning flexibility of the system, but also the number of CA patterns that need to be managed. However, this should be achievable with our approach of generalizing rules through algorithms such as Algorithm 1. In the longer term, we will consider 3D cells instead of the current 2D cells, and seek hardware implementation of them as well (becoming realizable with the advent of reversible circuits [94]). Incorporating fuzzy techniques also seems useful for increasing the decision making power of NNs, and a 'cellular automata neuro-fuzzy network dual system' may be a fruitful avenue of research to follow.

4.6 Conclusion

In this chapter we have described a novel CA architecture and the strategies planned for the investigation of intelligent automatic modification and creation of CA rules at run-time and full speed of FPGA hardware. The desired CA landscapes form structures of NNs, which, once sufficiently trained, will modify and enrich the rules of their own underlying CA. Training need not be limited to the adjustment of synapses' weights as in classical artificial NNs, but should also include the selection of a particular activation transfer function (whether excitatory or inhibitory) and the firing threshold for the neurons of a given NN region. Moreover, future solutions will allow trainable axon propagation delays and attenuation rates as well as permitting the implementation of a range of NN learning algorithms, such as back-propagation, Hebbian, Kohonen, Bayesian and Kak.

We will seek distributed knowledge as partially represented by axon's attenuation rates. But especially during neural signal process- ing (when neurons fire), axon's propagation delays should play a role in a global perspective, instead of being a latency handicap. Our CA- NN system is not simply a conventional 'CA-based NN', but also a 'NN-dependent CA' in the sense that the NN can learn CA rules, modify and generate them as well as having some control on the CA behaviour. Consequently, the system can be considered to lie within a data- driven dual paradigm. Furthermore, asynchronous neural signalling with spiking neurons and interaction between cells ensure that the system is paced by external stimuli, not by an internal distributed clock, improving power efficiency in practical applications.

To date, the work has validated the use of Synopsys System Studio design, debug and simulation environment (www.synopsys.com) and the SystemC description of our design (www.systemc.org), as well as verifying the asynchronous operation of the system. The primary goal of the future work is to tackle the problem of rule generation, but also to allow the system to benefit from further self-(re)organizing mechanisms discovered through rule learning that will subsequently allow both self-improvement of the system itself and optimization of the connected problem solution to a near-optimal point. Suitability of this system as an interactive supportive agent for micro-architecture space-pruning will be investigated and a SoC solution – with embed- ded FPGA for the dynamic reconfigurable portion of the system – will be considered.

4.7 Summary

After a brief review of different machine learning (ML) techniques and their applications, this chapter describes an innovative strategy to combine cellular automata (CA) with neural networks (NNs), leading to a self-reorganizing ’CA-NN’ paradigm. The work suggests different applications, including a highly adaptive core controller of navigating robot for (dynamic) real-time incremental learning and an interactive intelligent authority to assist design experts, accelerate expertise gain and ease traditional electronic design capture. The ML system proposed can be easily customized, can self-adapt, self-manage its resources and self-improve on-line against a range of complex problems, while optimizing their solution. Along the bottom-up design approach presented, a CA asynchronously forms emergent NN structures whose operations including spiking neurons are also asynchronous. In addition to this novelty, the emergent NN can learn rules of its own underlying CA and change them locally, towards overall self-reorganization due to an information theory-based technique permitting **‘improvability evaluation’**. The initial cellular architecture is described in SystemC, allowing for fast cycle-accurate simulations, and high-level synthesis targeting field-programmable gate arrays (FPGAs). Our CA-NN design is intended to take advantage of the flexibility of **‘soft-hardware’** FPGAs whose for example dynamic and partial self-reconfiguration features are becoming available, but are cumbersome to exploit through more conventional FPGA implementations of soft CPUs or DSPs.

Chapter 5

Self-Adaptive Compact Neuro-Controller for Natural Interaction and Training of Autonomous Communicating Robots

The aim of this chapter is to explore the utilization of the idea in the previous chapter to novel applications in next generation of advanced robotics.

5.1 Introduction

Showing dynamic dexterity off mobile robot navigation acrobatics is incontestably one of the few best external indicators on the level of internal intelligence an autonomous manoeuvring machine may possess. Therefore, applications in *mobile robotics* is of greatest interest for conveniently demonstrating such intelligence, beside also applicable for less apparent but certainly even more useful and powerful *logical inference* and reasoning feats.

This work is oriented towards the ultimate aim of a highly self-adaptive hardware digital design implemented on in-house *dynamically partially reconfigurable system-on-a-chip* (DPR-SoC) as the control system of robots or machines of any forms. This ever self-upgrading initial kernel design is supposed to make the whole native control system – newly connected to an arbitrary robotic body – (1) self-learn about using the sensors and actuators attached to the body, (2) gradually learn about the environmental surroundings perceived through the sensors and experienced using the actuators as well as learning about its own body in the same manner, (3) maintain and improve its internal and proximate external resources, and (4) learn on-line (i.e. at run-time) specific tasks through action-reward-based training and achieve them with real-time constraint.

Following cybernetics and 2^{nd} order cybernetics points of view [131], and in the intent of direct applications into humanity services such as cheerful education, the learning scale has been framed to real-world constraints. Hence, the action-reward loop will be of *real-life* dimension. E.g. instead of evolving populations (exo-phenotype phylogenesis) of such robots off-line in software simulations à la Darwin, it is preferable to encode *cellular programmes* [32] – *in silico* into the same unique real (and costly!) phenotype system – as well as roboticist mechanisms for showing new tasks and teaching corresponding skills to the robot *in situ*; this never-ending learner then requiring bare, or even better, no low-level human-coding intervention, but being taught through action-reward experiences.

The traditional *artificial intelligence* (AI) approach, with its well-known rapidly exploding algorithmic programmes typically implemented as sequential software has arguably now reached a stagnating stage, thus withholding the promises it was initially meant to satisfy. Yet, on the one hand plausible quantum computers might allow for elevating “weak AI”, and on the other hand in order to realize “strong AI”, we must take a radical approach as the one proposed in this chapter, therefore having to rather name future achievements after “bio-inspired highly self-adaptive complex systems” for instance.

Conventional software AI is held behind because the host hardware computer machines it uses are data instruction processors and are therefore natively very well attuned to calculation tasks, yet unable to naturally deal with semiotic and linguistic constructs. Nevertheless, paradigms such as SystemC and Handel-C from electronic design automation (EDA) community are now available and allow for conversion of sequential algorithms written in C++ into parallel algorithms generated in hardware description language (HDL), usually yielding significant gain in computation acceleration for some algorithms (e.g. data vectors and meshes) as well as strengthening fault tolerance of the system. Such parallelized AI systems typically exhibit enhanced performances once implemented in real hardware and several colleagues devote to this avenue of research, while in this chapter we right away consider bio-inspired parallel mechanisms and directly describe them in the ***Balsa*** language that we find to give the maximum freedom for creativity when compared to the other HDLs, probably due to the fact that asynchronous design – the very reason for ***Balsa*** to exist – encompasses the whole spectrum of possible hardware designs, thus covering the comparably tiny set of possible synchronous designs as well.

5.2 Motivation and Framework

On-chip hardware vs. embedded software: Software usually runs on hardware and when one looks carefully at the hardware fabrics of microprocessor-based computers or microcontrollers, one can see the limitations, cf. Subfigure 5.1(b), that are due to the very nature of the typical architecture of such systems. Indeed, on-board buses (that interconnect ICs) constitute permanent speed bottlenecks of the whole system. Not only because the frequency of these buses (up to a GHz) is reduced as compared to the one of on-chip cores (several GHz), but mainly because on-board circuitry means discrete circuitry, which in turn means a very small integration density of these buses that translates into having to highly serialize data transfers. On-chip standalone microcontrollers also make their hosted embedded software suffer due to fixed separation of pre-dedicated modules such as memory and CPU since communication also has to be serialized through buses that interconnect these fixed distinct modules. These examples of key problems in computer engineering will always remain since they come from the very nature of the modules and systems architectures designed for conventional sheer top-down engineering methodologies. For instance, one can consider a memory array module as a huge pocket for data, but with comparably extremely limited access port, thus insufficient and ruining performance potentials. Similarily, conventional CPU microarchitecture also causes such undesirable effects. These lead to untruly real-time robot control systems, therefore not reliable for the real-world tasks that embedded software engineering is therefore (1) mistakenly supposed to achieve seamlessly and (2) with utopian expectations of high level of professionalism.

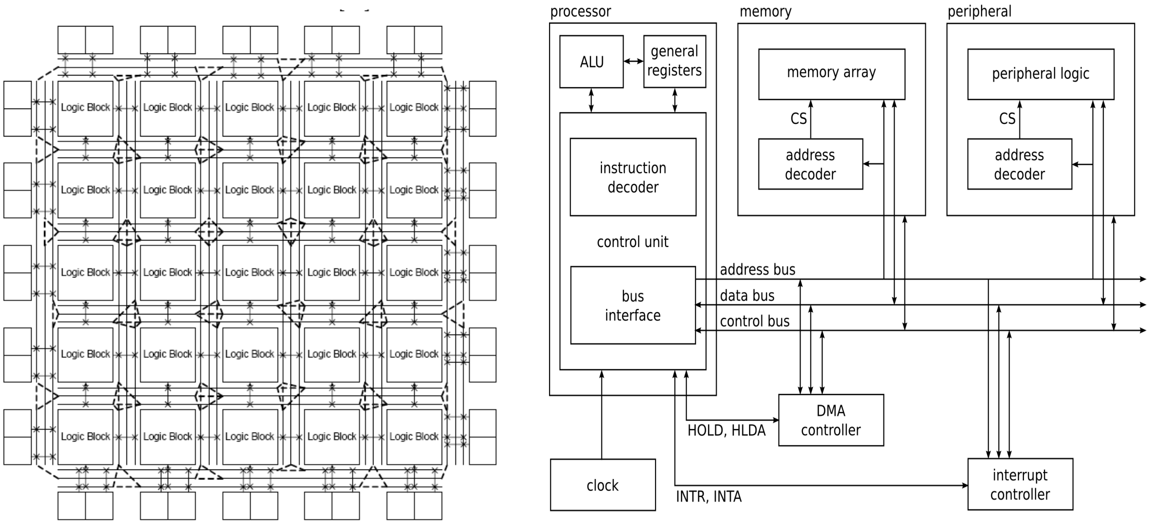


Figure 5.1: (a) Flexible hardware FPGA insight with its homogeneous mesh of CLBs (that can nicely implement auto-associative internally processing NN memory emerged from CA neural tissue) and surrounding digital I/O ports; and (b) typical heteroclite hardware microarchitecture of CPU (meant to host conventional OS and software) and its permanent tremendously bottlenecked buses shown as arrows that interconnect very limiting access ports of comparably immense contents of modules such as the memory array. (Note that (a) can implement (b), typically for purposes of CPU prototyping on FPGA.)

In contrast, within any living system, the very act of cells **recognizing** between themselves and taking care of themselves by **identifying** nutrients, poisonous substances, neutral elements and by-products while absorbing the former only – all this also with the top-to-bottom feedback help of the whole organism that the cells together constitute – is perhaps the major key in understanding how life self-sustains and flourishes. Indeed, co-evolving living cells are open systems and therefore do not violate the 2^{nd} law of thermodynamics since each cell ejects the entropy it generates, thereby keeping stable internal states of necessarily high complexity. Understanding of biologic organisms as well as engineering of bio-inspired beings both reside in the study of the cooperative behaviours of their quite clever constituting cells, be they composed of chemical molecules or electronic gates as underlying functional primitives.

Hence, amongst the recent advent of numerous diverse *machine learning* techniques (e.g. expert systems as in [132]) and research approaches to adaptivity, a cellular automata (CA) model naturally appeared of prime choice for the proposed system kernel. Furthermore, 2D CAs nicely map directly onto commercially off-the-shelf (COTS) hardware devices such as CPLD or FPGA, cf. Subfigure 5.1(a). In addition, despite synchronous design being still the current industrial reality due to political reasons as well as traditional beliefs, alternative pacing styles such as GALS or purely data/event-driven asynchronous protocols have become indispensable – especially concerning embedded systems – as global clock issues are now overwhelming (e.g. clock distribution, power consumption, overheating, etc issues). Similarly, sequential processing has remained the dominating thinking in electronic systems engineering, whereas complex systems based on highly distributed, massively parallel, simple, interactive processing units ultra densely interconnected – devoid of single point of failure – are imperative in a wide range of demanding applications such as true real-time reliable embedded robotics for multitasking or entire missions, upon which vital stakes depend. As defective, damaged or faulty units are replaceable with idle neighbours during run-time, self-healing and therefore extreme robustness is inherent to these then very attractive and promising control system architectures of the *cellular programming* paradigm [32].

Several updating schemes exist in the realm of 2D asynchronous CA [133]. Our attention was immediately drawn unto the self-synchronizing fashion since it represents the perfect digital candidate for modelling neural tissue [133]. To this end, the proposed CA landscape, e.g. Figure 5, Column 1, Page 5 of [134], consists of totipotent cells, each cell, e.g. Figure 1, Column 2, Page 2 of [134], interacting with its four local neighbours at its cardinal points[1], allowing it to

---

[1] Taking into account the four other local neighbours along its two diagonals exponentially increases the com-

make correct decisions on the state it should adopt based on hard-coded basic local transition rules that ensure neural compatibilities between adjacent docked cells. Unlike in [71] and [135] where every cell is a neuron and therefore biologically implausible, in our proposed model – also intended to help reciprocate knowledge we glean from biology – a cell can morph into one of five states and assume the corresponding role (*neural signalling process*), which is either of neuron (activation transfer function), synapse (weight), axon (output distribution, attenuation rate and propagation delay), dendrite (input sum) or blank (empty spaces are to NN structures what silences are to music scores: **essential**). Docked cells at the CA level are liable to exchange spikes of signal – through the then active neural channels – as for the neural communication at the above NN level. The asynchronous circuitry constructs in ***Balsa*** [136] can implement handshake protocols that nicely support spiking communication, which has been found most effective and efficient in biological neural network operations [137, 138].

During the past few decades, CAs have been tremendously studied [6, 139] and therefore the literature reports endless lists of various CA configurations with diverse sets of local rules associated with the global emerging behaviours and dynamic phenomena they generate including very interesting ones such as self-reproduction [140]. Hence, automating the production of relevant and useful sets of local CA rules has appeared necessary to us and we have found it possible to engineer as long as guiding criteria for run-time evaluation are in place. In the case of the proposed CA, the number of possible neurally valid pattern states (each composed of a central cell state, its four local neighbours' states and its four docking states) is in the order of 10^7, each of which requiring a transition rule, the total set of rules being visibly unmanageable to code at hand [6, 141]. Furthermore, coding all rules manually would mean a fixed set of behaviours of the emerged system, whereas leaving this microscopic tedious job to the system itself through automated rule discovery and learning also has the advantage of making it able to adapt to unusual, unexpected, unpredictable or unknown macroscopic situations, often even

---

plexity, whereas considering non-local remote cells has the same effect while making the CA exhibit interesting and quite useful non-classical (quantum) phenomena as well.

changing dynamically.

Technique (loop of epigenesis and ontogenesis phenomena mutually affecting and enhancing each other)**:** given an initial emerged NN sufficiently grown to perform primary basic tasks of recognition:

1. rule discovery (NN discovers useful CA rules and recognizes relevant patterns through inductive inference),
2. rule amendment or implementation (NN rewrites, overrides or writes new CA rules that have been discovered and for which the CA behaviour was found useful while discarding useless or harmful CA rules),
3. let the NN behave and let the active neural tissue grow further,
4. note this is a parallel algorithm so there is no GOTO 1) here; 1), 2) and 3) run concurrently,

thus, the NN is the programmer of the rules that dictate the behaviours of its own underlying CA.

This advanced approach of harnessing emergence for self-upgrade of standalone neuro-controller has obvious applications in DPR-SoC for neuro-robotic control and for powerful machine logical inference [142] with deep understanding, leading to the concept of a standalone ‘Robot Theorist’ with physical aptitudes e.g. [143]. One major factor governing the way of development and adaptation of our cross-platform transferable neuro-controller system is simply the type of robotic body (and its features) it is interfaced with. The envision is therefore to interface our neuro-controller systems in their seed form with various robot bodies such as security, medical, care, companion, inspection, education, coach, rescue or hazardous duty robots, and the potential list goes on. Furthermore, alternative techniques to controlling emerging behaviours such as *graph grammar bottom-up engineering* have already proven successful [144]. Gradual event-based self-(re)programming and *bottom-up* (re)design of whole robot control system through real-time, friendly, intuitive and fun interactions (that do not require human coding) at the global level is surely one of the most important ideals of this proposed self-adaptive and self-organizing system.

5.3 Hardware Digital Neural Tissue

Building up on our prior work [134], we recall that the proposed model is (evidently) based upon the **four** well-known (commonly accepted) main fundamental neurally functional elements found in biological nervous systems, that is, synapse, dendrite, neuron and axon [127]. In addition, as for a 2D model of digital system composed of discrete 2D elements, the mandatory introduction of a fifth fundamental but neurally non-functional element becomes obvious; that is, 2D four-edged blank (unused) cell. In fact, even in 3D biological nervous systems, where neurally functional 3D elements are connected together in a very compact way, dynamic physical modifications of the connections imply the presence of neurally non-functional (but indeed physically functional) "interstices" that leave (sufficient) room for manoeuvre for the neurally functional elements to reconnect.

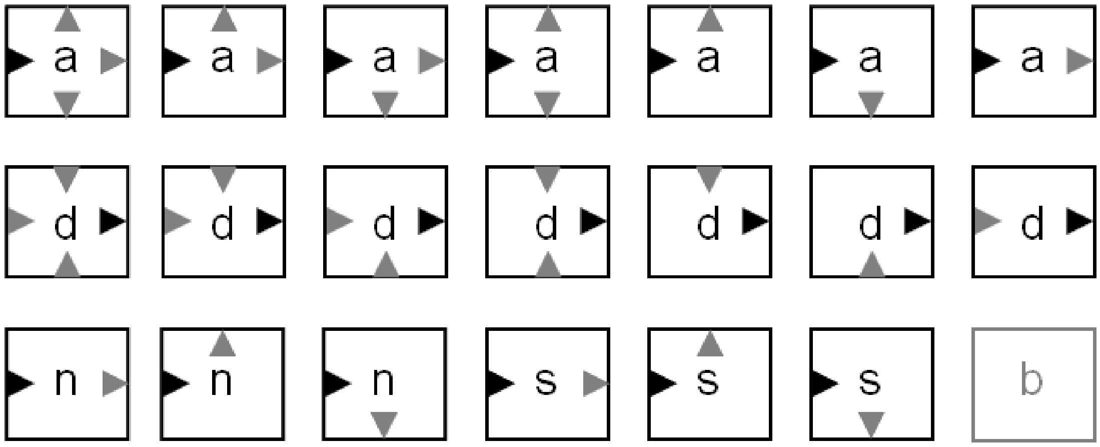


Figure 5.2: Twenty distinct neurally functional elements, plus three other distinct cardinal rotations of each (not shown here), and the neurally non-functional (b=blank) element; giving rise to a total of 81 distinct fundamental types of element.

Figure 2, Page 3 of [134] recalled here in Figure 5.2 illustrates 80 distinct neurally functional elements (when taking into account the exhaustive list of docks states for each of the aforementioned **four** fundamental elements, which leads to the 20 elements of the picture as well as when taking into account the four cardinal orientations of a 2D [Euclidean] plane, which leads to the 80 elements), one of which a cell of our neural tissue design can morph into, in addition to the blank element. By virtue, this last element does not possess any docking connection. This leads to 81 distinct fundamental types of element.

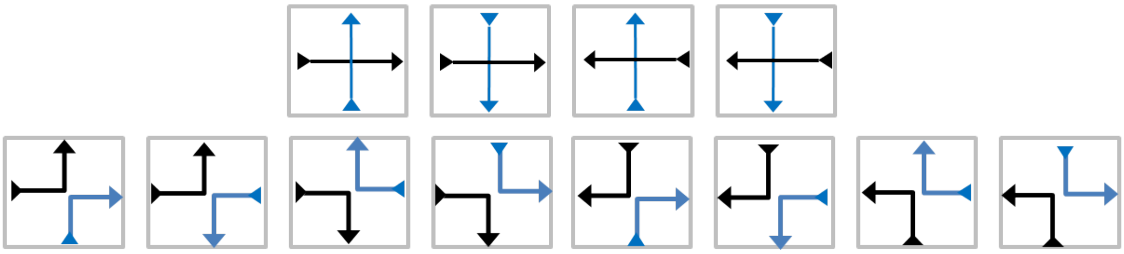


Figure 5.3: Twelve distinct additional fundamental types of elements: the joining cells. The top four are necessary in big designs, the other eight enrich the list and allow for more compact implementations of 2D CAs composed of square edged cells.

Furthermore, twelve additional fundamental joining types of element shown in Figure 5.3 are introduced to either compactify or enrich neural tissue, or indeed make some 2D neural network patterns topologically possible to form at all. Therefore, the total number of necessary and sufficient distinct fundamental elements in our model is 93. We recall that an elementary 2D cell pattern is composed of five cells: a central cell and its four local cardinal neighbours. Hence, the total number of possible patterns (including both neurally valid and neurally invalid patterns) is exactly $93^5 = 6,956,883,693$ (almost 7 billion).

Table 5.1: Potential complexity of a system that can be produced from the fundamental elements.

Five-cell patterns statistics	**Total number of possible patterns**	*Number of neurally valid patterns*	*Number of neurally invalid patterns*	*Richness: 1 valid in ### invalid*	*Ratio: per thousand (‰)*
Initial list	81^5 = 3,486,784,401	7,844,797	3,478,939,604	1 in 444	2.2498‰
With cross junctions	85^5 = 4,437,053,125	12,667,213	4,424,385,912	1 in 350	2.8549‰
With cross and compact junctions	$\mathbf{93^5}$ = **6,956,883,693**	**23,629,269**	**6,933,254,424**	**1 in 294**	**3.3965‰**

We have calculated the total number of **neurally valid** patterns when also taking into account the three other cardinal rotations of the 20 neurally functional elements in the picture of Figure 5.2, plus the 81^{st} blank element, in addition to the twelve junction elements of Figure 5.3. That total is exactly 23,629,269 neurally valid patterns. Last row in Table 5.1 summarizes the results of these calculations. The table clearly shows that the number of neurally valid patterns (that may be formed) is tiny as compared to the number of neurally invalid patterns (that must be prevented from being formed). Nevertheless, the number of neurally valid patterns in the order of 10^7 is huge enough to prevent from practically coding at hand the CA transition rules for attracting invalid pattern states into valid ones by attempting to change incompatible state of the central cell only as in typical CA. Furthermore, the search space is of gigantic size of 93^5 (mostly resulting from the number of neurally invalid patterns). Alas, these statistical results as well as the ones reported in [141], in addition to the exhaustive work on 1D synchronous elementary CA in [6] and its corresponding book with over a thousand pages of rules and associated behaviours, all together undeniably prove the impossibility of manually coding basic and growth rules, respectively for validity and usefulness of CA. However, in our context of self-capable systems, entrusting as many self-* properties (and therefore autonomy) as possible to such systems should be commonplace.

Hence, the above firmly encourages the automation of the production of ‘valid and useful’ rules for 2D asynchronous CA in view of the formation of digital neural tissue. Ideally, this should be done on parallel hardware devices such as FPGAs, both for processing speeds and ASIC prototyping reasons. Indeed, ASIC implementation of self-adaptive and self-reorganizing digital neural tissue would render both fastest processing speeds and higher biological and physical plausibility of NNs, including neuro-modulation [145]. In addition, parallelism inherent to hardware implementation makes straightforward the handling and processing of non-uniform rules over the CA landscape. Non-uniformity of rules is necessary for the differentiation of natively totipotent cells, and for the specialization of CA regions.

The literature poorly reports successful work on automation of production of such CA rules, because the obvious method that one would (and has) naively follow(ed) for this aim involves the use of genetic algorithm (GA) techniques, which actually do not work well for this specific problem [8,141]. Instead of using GA for evolving populations of distinct couples {CA,rules} at the end of which only one (fittest) couple is kept (and the rest wasted), we use a technique based on *cellular programming* [32] in order to evolve a population of couples {cells,rules} at the end of which the whole population that represents the (best found) CA solution (as neural tissue pre-adapted to the interfaced robot body) is kept (and nothing wasted). Besides, the literature reports extensive work on rule discovery with neural networks [146]. In particular, feedback (recurrent) NNs are directly usable as associative memory due to their convergence to stable states as mentioned in [147]. By associating neurally valid CA patterns with their stable states, a feedback NN (based on our 2D asynchronous CA as neural tissue) can readily perform the desired production of valid CA rules. This background comforts our proposal in Section 5.2 of using a sufficiently grown NN on our asynchronous CA-based neural tissue so that this NN as an associative memory pairs local CA rules with global CA behaviours. We recall that the NN itself is based on that very same CA as well.

Such similar work where the results of NN processes drive the growth and pruning of that very same NN, thereby inevitably modifying in turn some NN learning processes as well, has already been successfully achieved [148]. Both the aforementioned proposal and [148] enable a looped bridge between ontogenetic (growth and development) and epigenetic (learning and processing) phenomena, exhibiting speeds that dwarf phylogenetic evolution (à la Darwin). While [148] looks at the NN level only, we propose to also tackle the problem of generating rules at a lower CA level, by using the NN composed of that very same CA, thereby enabling the equivalent of effective and efficient automated low-level coding, which has remained sterile within the conventional software coding dogma [149], but with interesting effects in a recently renewed era of self-modifying codes [150].

5.4 CA Rules

Generally in CA, the number of rules to be specified depends on several factors such as the number of cell states, the size of cell neighbourhood considered and the level of adaptivity of the algorithm that determines the new state of the central cell based on information about the neighbourhood. The number of rules to be specified typically increases with the number of cell states and the size of the neighbourhood, and decreases with the level of adaptivity of the algorithm.

“1001 neighbourhoods”: In practice, we only need to know the number of rules that have to be pre-defined, and keep this number as small as possible for practicability reasons, given the algorithm that we intend to use. If central cell implements an algorithm that considers the state of each neighbour as given by both together its **type state** (here A,D,N,S, or T: $s_t = 5$) and its **dock state** (input or output: $d_p = 2$) to/from central cell only (not to/from farther neighbours, even though it may also be able to record these three other docks states [of each of its four neighbours: $k_p = 4$] for later uses or to help settle ambiguities), considers undocked neighbours as type state ‘B’ (even though it may also be able to record the actual type state of such neighbour as well as its three other docks states for later uses or to help settle ambiguities: $b_p = 1$), and can treat interchangeable neighbours as equivalents (use of Γ for counting unique cases), then the number of rules that can be specified in practice, which is also the number of unique neighbourhoods is $r_p = \Gamma^k_{s_p} = \Gamma^k_{st*d_p+b_p} = \Gamma^4_{5*2+1} = \Gamma^4_{11} = 1001$. $\square$

The **multiset coefficient** Γ is related to the binomial coefficient – the "choose function" C^k_n – that gives the number of unique ways to choose k elements from a set of n elements when repetitions are not allowed unlike in multisets, but when order is unimportant (and therefore counted only once) like in multisets. The relation between both coefficients is $\Gamma^k_n = \left(\!\!\left(\begin{array}{c} n \\ k \end{array}\right)\!\!\right) = C^k_{n+k-1} = \left(\begin{array}{c} n+k-1 \\ k \end{array}\right)$. Given the well-known expression $C^k_n = \left(\begin{array}{c} n \\ k \end{array}\right) = \frac{n!}{k!(n-k)!}$, we draw $\Gamma^k_n = \left(\!\!\left(\begin{array}{c} n \\ k \end{array}\right)\!\!\right) = \frac{(n+k-1)!}{k!(n-1)!}$ by substitution.

However, not all of these 1001 unique neighbourhoods are neurally compatible in that some do not allow a state (not any out of the 93 fundamental elements) for the central cell neurally valid with each of its four neighbours at once. The total number of unique neurally compatible neighbourhoods must be calculated by composition as follows:

The first axon element in Figure 5.2 (as well as its three rotative symmetric copies, which in this case are equivalents as we will see in the following and therefore are not counted) has one input dock and three output docks. The input dock of that axon – axon considered here as a central cell of a 5-cell pattern – is neurally valid with its neighbour having either of axon, neuron or junction as type state and an output dock as dock state only (dockable to the input of the central cell; output or unconnected dock states for this neighbour is not neurally valid with the input of the central cell). The input dock is therefore neurally valid with $a_i = 3$ types of neighbour as viewed from input i of the central cell a. Independently from this, each of the three output docks o of a is neurally valid with its neighbour having either of axon, synapse or junction as type state and an input dock as dock state only. Each output dock is therefore neurally valid with $a_o = 3$ types of neighbour as viewed from this output o of the central cell a. Since some cases of neighbourhoods for the three outputs of the central cell are interchangeable, we need to remove them from brute calculation ${a_o}^{k_o}$ in order to count them only once as required for counting with equivalents. Thus, the number of unique neighbourhoods for the three $k_o = 3$ outputs o of a is $a_{ooo} = \Gamma_{a_o}^{k_o} = \Gamma_3^3 = 10$. Therefore, the number of unique neurally compatible neighbourhoods with an axon element as central cell having one input and three outputs is $a_{iooo} = a_i * \Gamma_{a_o}^{k_o} = a_i * a_{ooo} = 3 * \Gamma_3^3 = 3 * 10 = 30$.

Any of the second, third and fourth axon elements in Figure 5.2 as well as their three rotative symmetric copies each have one input dock, two output docks and one unconnected dock. This gives $a_{ioo} = a_i * a_{oo} * b = a_i * \Gamma_{a_o}^{k_o} * b = 3 * \Gamma_3^2 * 1 = 3 * 6 * 1 = 18$.

From this also, any axon elements that have one input, one output and two $k_b = 2$ unconnected docks are neurally valid with $a_{io} = a_i * a_o * \Gamma_b^{k_b} = 3 * 3 * \Gamma_1^2 = 3 * 3 * 1 = 9$ unique neighbourhoods.

The same figures are found for dendrites, with $d_{iiio} = \Gamma_{d_i}^{k_i} * d_o = d_{iii} * d_o = \Gamma_3^3 * 3 = 10 * 3 = 30$, $d_{iio} = \Gamma_{d_i}^{k_i} * d_o * b = d_{ii} * d_o * b = \Gamma_3^2 * 3 * 1 = 6 * 3 * 1 = 18$ and $d_{io} = d_i * d_o * \Gamma_b^{k_b} = 3 * 3 * \Gamma_1^2 = 3 * 3 * 1 = 9$.

For neuron and synapse elements we find $n_{io} = s_{io} = 2 * 2 * \Gamma_1^2 = 2 * 2 * 1 = 4$, while the counting for blank elements gives $b_{uuuu} = \Gamma_b^{k_b} = \Gamma_1^4$=1.

Finally, any junction elements have two inputs and two outputs. Covering all possible unique neurally valid connections for first output given its dependant first input (e.g $t_{ia} = 3$ means input connected to axon involves 3 possibilities for related output: axon, synapse or another junction) of a junction as central cell, we find that the number of unique neurally compatible neighbourhoods with a junction as central cell is $t_{iioo} = \Gamma_{t_{io}}^{k_t} = \Gamma_{t_{ia}+t_{id}+t_{in}+t_{is}+t_{it}}^{k_t} = \Gamma_{3+3+2+2+5}^2 = \Gamma_{15}^2 = 120$.

Hence, the total number of possible unique neurally compatible neighbourhoods (always neurally valid with a type of element as central cell and a docking configuration), which is also the number of rules whose outcome is unambiguous and should be activated in **f**ixed **v**alues of corresponding rule table entry variables, is $r_{fv} = a_{iooo} + a_{ioo} + a_{io} + d_{iiio} + d_{iio} + d_{io} + n_{io} + s_{io} + b_{uuuu} + t_{iioo} = 30 + 18 + 9 + 30 + 18 + 9 + 4 + 4 + 1 + 120 = 243$. $\Box$

The remaining $1001 - 243 = 758$ neighbourhoods is composed of two kinds of neighbourhoods. The first kind is neighbourhoods for which there exist more than one couple {type,docks} states neurally valid for the central cell. The associated rule for such a neighbourhood must have a (uninitialized using new type state 'P' for **p**robability) variable as entry value for the rule transition outcome. The value of such variables may be chosen (and may be changed again and again) at run-time by dedicated NN outputs (the NN having authority on and having gained knowledge about its own constituting CA should preserve the system integrity as well as improve its performances through those higher-level "meta-ruling" processes, concomitantly with external sensory-motor interactions). The second kind is neighbourhoods for which there does not exist any neurally valid couple {type,docks} state for the central cell. The associated rule for such a neighbourhood must either be inactivated (made passive by simply being absent at all of the rule table) so that the state of central cell remains unchanged, or be activated with value 'B' as entry value for the rule transition outcome in order to force neighbours have their state adapted instead. The proportion of inactivated rules throughout the whole CA landscape must be governed so that the global regime of CA evolution stays within the 'edge of chaos', thereby avoiding the CA to freeze at all (complete deadlock of CA evolution) or oppositely to irremediably affect its integrity and ongoing adequate overall operation (purely chaotic CA evolution).

5.5 Methodology

Biological nervous systems exhibit operations and communications asynchronously, i.e. there is no global clock purposely distributed in the system unlike in conventional computer systems. Asynchronous design was one of the novelties of our previous work reported in [134], and was using the SystemC language with its asynchronous package called ASC. However, even though SystemC has proven useful for microelectronic hardware design modelling purposes, there is still currently a consensus agreeing that SystemC is not mature and perhaps will never be sufficiently adequate for simulation and implementation purposes. Therefore, the work presented in this chapter has adopted Balsa [136], which is at the same time a hardware description language

(HDL), a comprehensive development and simulation environment, and a synthesis system (hardware design flow) that can produce Verilog code as well as EDIF netlists – that for example can be used in ModelSim to validate our designs, or that can be fed to lower level electronic design automation (EDA) tools in order to physically target reconfigurable devices such as our Xilinx XCV2000E and Spartan3E FPGAs. Balsa [136] is a very strong hardware design, simulation and implementation system for **asynchronous** microelectronic circuits, and this suite of tools has proven very successful and useful since the work of Furber et al. [126, 151].

The Balsa manual [136] gives a simple example of a 1-bit wide, one-place buffer with asynchronous operations and their corresponding handshake circuit graphs (that very interestingly closely resembles Petri-Net diagrams). In Balsa, a handshake circuit is composed of Balsa handshake components (all listed in Appendix of [136]). In particular, data channels may be either *push* or *pull*. For instance, an asynchronous simple cell *A* designed with Balsa and having its data input interfaced with another cell *B* via a pull data channel will request cell *B* to send data, this whenever cell *A* decides so. This type of asynchronous communication is control-driven and is not seen in the Nature nor in biological nervous systems. Indeed, if cell *A* represents a sensory input nerve, and by thought we replace cell *B* by the outside world, it would mean that for instance, human fingertip nerves actively probe the external surrounding objects in contact with the hand in search for sensory data. Of course, this is typically not the case. Now if the data input is interfaced via a push data channel, cell *A* passively waits for a request from cell *B* and will receive data whenever that request is triggered by cell *B*, and provided that cell *A* is ready. This type of asynchronous communication is data-driven and is pervasive in the Nature and in biological nervous systems all the way from sensory inputs through to motor driver outputs. Hence, our current work exclusively uses Balsa push data channels for modelling and implementing all well-known asynchronous passive communications observed in biological systems.

In the above examples we insinuate the connection and asynchronous communication of multiple cells (i.e. cell *A* connected to cell *B* or interfaced with the outside world). The Balsa manual [136] shows a method to elegantly instantiate (reused) one-place buffers parametrically so as to form the cheapest possible control-driven pipeline composed of n-place parametrized buffer. The pipeline length n is a constant of integer value supposed to be chosen at design time.

Deriving from the Balsa example of an asynchronous 1-bit wide one-place control-driven buffer, in a low level text file that we created and named `description.balsa`, we first described a single simplest possible data-driven cell for our model. One that, by requirement for proper cellular operations, can hold a little bit more than one piece of 1-bit wide data. First of all, the cell must possess one variable that stores its *current state*. This variable may be of Balsa **enum** (enumerated) type, very similar to that in C/C++ programming language. Our 3-bit wide enum list is B(=0), D(=1), A(=2), T(=3), S(=4) and N(=5), respectively for blank, dendrite, axon, junction, synapse and neuron as available finite states for the cell. Next, the cell must have variables of the same enum type for storing latest known states of neighbour cells. Then, it needs a variable of again the same enum type in order to store the result of the decision produced by the basic state transition rules along with the main looped process within the cell. As in asynchronous CA generally, the decision is made based upon the latest known states of neighbour cells, the current state of this cell (when desired), and the active transition rule. The fixed list of 243 active basic rules is hard-coded in the first private part of the looped process within the cell. That first private part is dedicated to CA control, which in this case is distributed over cells, each having a copy of that code (which also includes changeable rules – from 244 up to 1001 – potentially leading to useful self-differentiation, self-specialization of cell modules in non-uniform asynchronous CA); whereas the second private part handles both orientation and processing of neural spike signals. Finally, our cell may have more than one input data channel as opposed to the exemplary Balsa one-place buffer in [136]. Indeed, at least two input channels are required in view of bottoming up an experimental 1D elementary CA with multiple instances of our cell. In the case of our model, a cell has West, East, North and South neighbours and therefore requires four input channels.

Next, we instantiated several ($n \times m$) above cells in a higher level text file that we named `instantiation.balsa` by importing the previous lower level file and connecting cells by naming and in a parametrized way (so as to easily change the value of the constants n and m) as suggested by the above exemplary n-buffer pipeline [136]. We also created a second field of data for the type of our variables. Indeed, we needed this additional field to store unique identity of cells. Therefore, the new type had to be changed from our enum type above into a Balsa **record** type, very similar to that of structure in C or class in C++ programming languages. That record type has as its first member our previous enum type, and as its second member a numeric type that we chose to code over a byte of hardware, which is sufficient for holding enough unique identity numbers of cells for most of our future experiments.

Interestingly, in `instantiation.balsa` we then double closed-looped such a 2D matrix of cells by connecting the m output (resp. input) data channels from the right hand side of the $n^{th} \times m$ cells to the m left hand side input (resp. output) data channels of the $1^{st} \times m$ cells in the matrix. We did the same for connecting together both horizontal edges of the matrix as to obtain a 2D hyperplane (or virtual 3D torus). We wrote an input file that just on one line of text contained the initial state and the unique identity number (such as {{B,10},{D,11},{A,12}, ...,{etc,$n \times m$}}, which is considered as a single valid piece of data by the Balsa test harness interface) of as many cells as specified by n and m within `instantiation.balsa`. We named that input file `conds.ini` to reflect its role of providing an asynchronous CA with initial conditions. In addition, `conds.ini` needs to be read only once by the Balsa test harness and fed to our 2D asynchronous CA (matrix of $n \times m$ cells), right at the beginning of a simulation. To our surprise from those several changes, whose some are a little bit adventurous such as the double closed-loop matrix of cells (that we recall operate in a double-way fashion unlike more conventional looped shift registers in modern asynchronous designs, e.g. [126]), several simulations validated the asynchronous operations in that they made the cells correctly obey our set of fixed basic rules throughout the 2D hyperplane CA.

Figure 5.4 depicts a learned and configured neural XOR function on 2D closed (hyperplane) CA as neural tissue. We preliminarily obtained concomitantly both the desired neural XOR topology *and* the correct values of its synapses' weights and neurons' thresholds through off-line simulations (in software) of our model that includes our *cellular programme* as well as our NN learning algorithm based on particle swarm optimization. (Next stage of our work will be to implement those two processes for on-line – more autonomous and much faster – execution together with the following, which already runs on-line.) We transcripted this learned and configured CA landscape in our `conds.ini` file, but experimentally realizing that the Balsa interface for test harnesses cannot cope with single lines of piece of data above a certain size (and thus above our needs), we had to transcript the initial condition of one cell at every one line of the file. For instance, the first line looked like {0,0,I,U,O,U,21,A,10}, where by convention the first member represents the value of a neural spike, the second member represents a parameter such as weight (resp. threshold) value for synapses (resp. for neurons) (displayed at the bottom right corner of every synapse (resp. neuron) cell in Figure 5.4), the third through to the sixth members are docking states of the cell respectively to the West, North, East and South of that cell. Value 'U' means that the cell is neurally unconnected to/from the cardinal direction given by the dock state having that value (docks North and South are therefore unconnected in this exemplary cell of identity number 10). 'I' means that the cell will receive and process neural spiking signals from the cardinal direction given by the dock state having that value (dock West for that same cell number 10). And 'O' means that the cell will eject any spike signals to the cardinal direction given by the dock state having that value (dock East for still that same cell with ID number 10). The seventh member is the value of a timer counting down to ten, after which the cell is in 'changeable state' mode (introduction of private local timers for each cell e.g. [152–154]). The eighth member holds the value of the type state of the cell and the last member represents the unique ID number of a cell (then 10 for this cell) throughout the CA landscape.

For each cell concurrently, the first part of our code that holds the basic rules in `description.balsa` is first executed: the type state of the cell is changed if applicable and the connectivity of the cell's four docks is accommodated to that of its four compatible neighbours' (the cell's four docks are each set to 'U' if cell type is 'B' and/or if neighbourhood is incompatible, so the effect is that this central cell will force its surroundings to change states and give a chance to eventually get a compatible neighbourhood later on), the timer is reset up to its initial 'cell lifetime' value (21 for yet again our same cell with ID number 10), and then the second part of our code in `description.balsa` that describes the neural behaviour of a cell is executed – the cell is in unchangeable mode – until the timer reaches again our bottom value chosen as ten (for purposes of making our log files readable), at which time the cell switches back to changeable mode so that rules are obeyed (while spike signals may be dropped but without adverse effect on the whole system, unlike in conventional control-driven systems), and so on.

The compilation time is negligible. The simulation run-time is ridiculously small on the over 2GHz workstation that Balsa is running on under Linux (Fedora Core).

5.6 Experimental Results

In order to illustrate the functioning of the kernel of our system – and give a grasp of its potential full capabilities following supposed sufficient growth of enough modules –, after Balsa on-line emulation of an on-chip 9x6 2D CA as single (exemplary) module initialized with configuration file `conds.ini` depicted by Figure 5.4, using Matlab we extracted from the generated output file only the values of spikes over all cells and all local time steps; that is, we considered only the first member of the type record field for our *current state* variable of every cell. We stored those values over 2D matrices over emulation time steps into a 3D matrix under Matlab. Each subfigure in Figure 5.5 depicts the final stage of the progression animation of the landscape of our 9x6 2D CA under emulation. The first Subfigure 5.5a actually starts with input cells 28 and 46 having value 0 and 0 (respectively 0 and 1, 1 and 1, and 1 and 0 for Subfigures 5.5b, 5.5c, and 5.5d). As time steps progress, the spike signals propagate from cells to cells in an asynchronous and parallel mode of operation, unlike in [155].

First of all, each subfigure shows that the resulting value of the output neuron 44 holds (and maintains, even though the rest of the values corresponding to later time steps are not shown) the expected XOR output value in accordance with the values of the input cells and the XOR discrimination. Secondly, during the establishment of the values throughout this neural XOR function, the output neuron does not undergo any transient for all four cases of input vectors: 00, 01, 11 and 10. And thirdly, the time necessary for the XOR function to switch is the smallest possible with regard to the distances the spikes have to travel along the 2D plane, following the docked connections recalled in Figure 5.4. In fact, the values of spikes that can be observed are those within neuron, synapse and dendrite cells. Indeed, axon cells typically contain spikes within value of unity, and can hardly be observed in this figure given the scale of the Z axis of the subfigures (actually, if one exagerately zooms in a subfigure within the electronic version of this document, one can observe values as small as the unity. Indeed, in Matlab **flat** graph bars with bolder and darker outer black box have non-zero value, zero otherwise.)

5.7 Conclusion

The work presented in this chapter has defined the framework of autonomous design of intelligent true real-time controllers for communicating and performing machines such as robots. It was in view of minimizing debugging requirements and ultimately set human designers free from debugging tasks, which has implication in trust in autonomous machines and robots (mostly based upon perceived inputs; upon experiences with outputs – let's be cautious with potentially harmful robot bodies – and feedbacks to inputs through the external real-world; upon internal entities, rules and other mechanisms precoded and predesigned; and almost negligibly but most concerning, upon internally emerging data and processes) philosophized in [156–159].

Meanwhile, it seems mostly salutary to build (artificial) helpers in view of going higher, getting better: lords of the universe [160]. It also seems fair enough to try and understand our brains and associated cognitive phenomena by building communicating and performing machines in which we can observe similar phenomena and which may even explain these phenomena to themselves as well as to us. However, actual current level of intelligence of such a machine or robot will be limited (and be equal to a fraction of its theoretically predicted maximal level of intelligence) by its actual current available access to external resources.

We recall that one may not lose control that one does not initially have. Assuming that one has little control (Type 0 civilizations cannot even control the Earth weather [160]), we should perhaps first draw a list of currently human-controlled entities and objects, and only then argue such as in [157] whether we should hand over the control of some of these entities and objects to such highly autonomous machines.

5.8 Summary

A biologically-inspired asynchronous non-uniform cellular automata-based spiking neural network module can grow and self-reorganize in state-of-the-art microelectronic hardware: dynamically partially reconfigurable system-on-a-chip (DPR-SoC). Such low level modules implemented on DPR-SoC are interfaced with external sensors and actuators, and adapt their behaviours on-line to match with required and rewarded basic task skills.

The system can be scaled up to levels of required and rewarded entire goal accomplishments by recursive instantiation of similar but higher level neural modules – recursively up to the utmost module for highest decisions as necessary –, which filter neural information throughout modular hierarchy, thereby coordinating perceptions, memory storages and retrievals, decisions and actions together as well as inherently solving the well-known open problem of sensor fusion.

The main novelty of this work is the never-ending evolution, adaptation and learning of every totipotent module – and therefore of the entire system – due to authority of neural network process results to modify any local rules of its own underlying non-uniform cellular automata, leading to different growth styles and thus to formations of new neural network patterns, in turn changing neural network processes and therefore results as well, and so on.

These mutually intertwined phenomena between neural network and cellular automata ensure both mapping of module behaviours and growth styles with corresponding active rules; and self-tuning towards overall self-improvement of the system against its surroundings and its goal.

The created and adopted framework involves development of an evolvable microarchitecture of neuro-controller with capability for self-tuning to interfaced robot bodies of arbitrary shapes as well as with programmability of behaviours through high-level interactions rather than low-level coding.

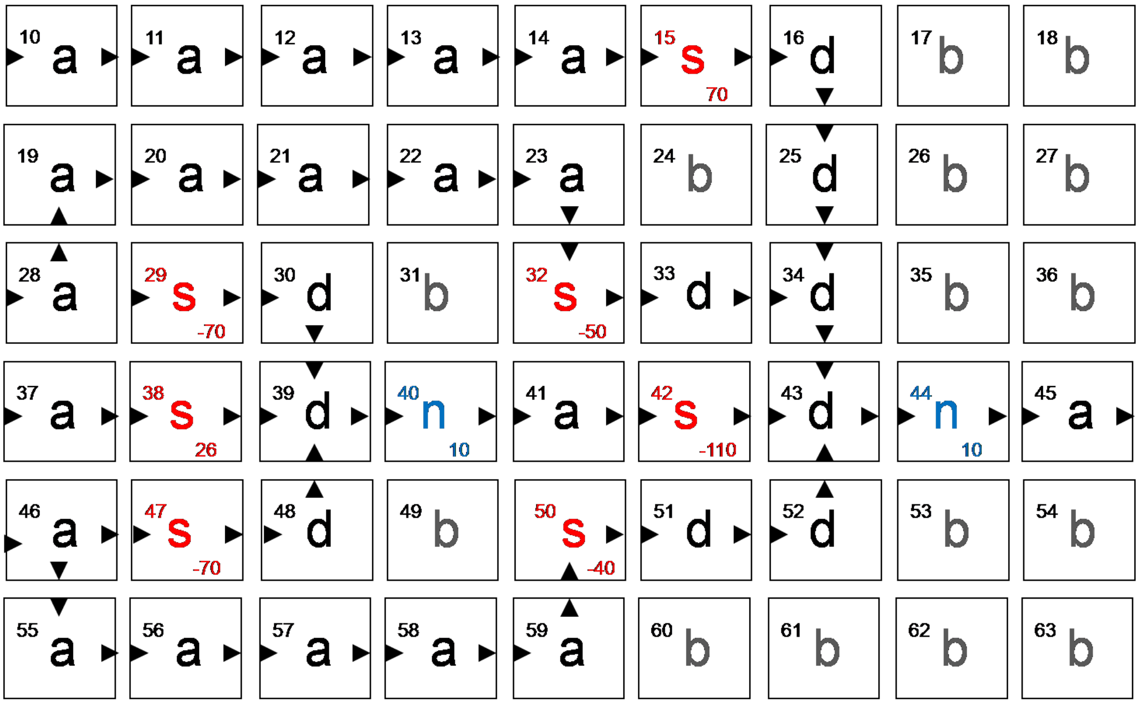
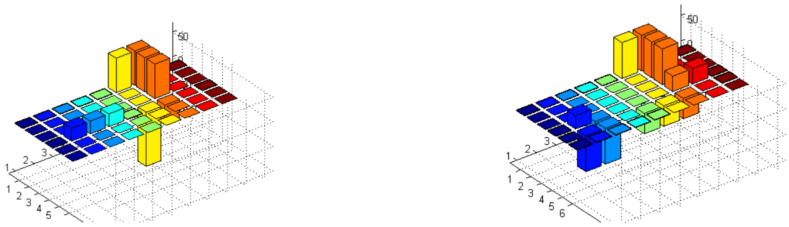
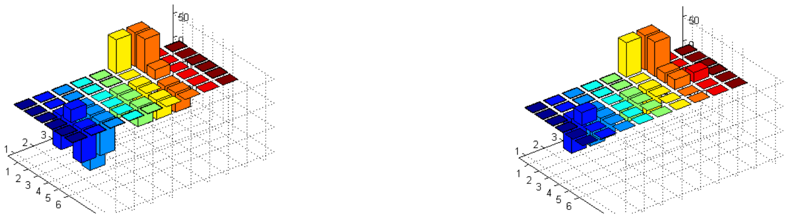


Figure 5.4: Neural XOR function on 9x6-cell asynchronous system with reorganizable topology and spike signalling capability, in its initial condition state – given by the actual initial numbers at the bottom right of each cell: weight for synapses and threshold for neurons, by the actual initial dock state of each cell, and by the actual initial type state of each cell. Only actual initial values of spikes are not shown, but are ‘1’ for cells 10 and 37, which act as the bias inputs of that ‘**neural** XOR’; can be chosen as ‘0’ or ‘1’ for each of cells 28 and 46, which are the ‘neural **XOR**’ inputs, and are ‘0’ for each of the rest of cells including the ‘neural **XOR**’ output: cell 45.



(a) Input vector 00

(b) Input vector 01



(c) Input vector 11

(d) Input vector 10

Figure 5.5: XOR illustrations of final state for each of the four landscapes – emanated from the four unique input vectors at initial state – showing the NN-internal transients and the correct settled output for each case: output neuron as a non-null red coloured cell is firing.

Chapter 6

Autonomous Design in VLSI: Growing and Learning on Silicon and an In-House Universal Cellular Neural Platform

This chapter focuses on establishing a new standard digital platform for the investigation of plausible autonomous on-chip design of useful learned functionalities.

6.1 Introduction

This chapter presents refinements of our prior work [161] and exhibits potential utilization of our platform [162]. The minimalist approach described helps keep the initial "seed design" proposed necessarily simple enough. Yet, we show that the useful complexity of systems that can self-develop from initial simple conditions on the seed is considerable. The chapter also makes apparent the differences between this and earlier approaches, such as CellMatrix [163] and evolvable hardware [92]. We had preliminary found that machine learning techniques that revolve around neural circuits are excellent candidates for on-chip VLSI self-modifying systems, and this marriage has therefore high likelihood of ultimately leading to effective and efficient on-chip intrinsic autonomous digital design of controller for performing machines that can smartly handle unexpected, unpredictable, unknown, new situations.

This chapter presents a simple yet efficient novel self-reconfigurable hardware architecture that reconfigures the implementation from within the device based solely on on-line data input pattern sequences supplied in an open-ended manner at run-time. The dynamically reconfigurable building blocks of this self-adaptive structure are deliberately constrained to form neural circuitry rather than logic as in field-programmable gate arrays (FPGAs). The following are discussed. Firstly, a simple seed design is described from which systems of considerable complexity can self-develop. Secondly, the results obtained so far from an early prototype are shown to both validate the approaches undertaken and demonstrate the practical functioning principles of on-chip autonomous reconfiguration. Finally, the preliminary results confirm that machine learning techniques that revolve around cellular neural circuits are excellent candidates for on-chip VLSI self-modifying systems and that this approach represents a promising solution to effective and efficient on-chip autonomous digital design[1].

6.2 Background Work

Along the project work a platform has been developed with which to grow, prune and refine hardware cellular neural seeds for formation of digital discrete neural tissue with spiking neurons [127, 164]. Various parameters have been made available for the user of the platform (the seed creator) to conduct experiments. For instance, the sizes and number of seed modules can be set

---

[1] The work primarily involves autonomous function learning for robotic control. That is, new functions autonomously learned by highly adaptable on-chip neural circuits from input data fed by physical sensors; and at present those functions are limited to the control drive of physical actuators aimed at the external real-world. However in the near-future after sufficient self-development and training of the above, the learnt functions may apply to the very chip-internal microelectronic world holding them as well, giving rise to apparent spontaneous autonomous function learning for internal inference reasoning and then natural language/object processing: artificial intelligence (AI) at the end of the chain, where it should be.

in advance or self-learned in a more advanced version, and the cellular rules that encode kernel properties can be set in advance as well or also be changed dynamically by the system according to a thermostat and respectively to policies in order for seed module applications to remain in equilibrium and respectively to improve against their imposed tasks. But also, as an open platform, properties of the system kernel can be modified, which is useful mainly for research purposes. Biological nervous systems exhibit operations and communications asynchronously, i.e. there is no global clock purposely distributed in the system unlike in conventional computer systems. Asynchronous design was one of the novelties of this work [134]. To this end the project adopted Balsa[2]. Balsa is a very strong hardware design, simulation and implementation system for asynchronous microelectronic circuits, and this suite of tools has proven very successful and useful since the work of D. Edwards et al. [136].

6.3 Biologically-Realistic Cellular Neural Model

The proposed model is based upon synapse S, dendrite D, neuron N and axon A fundamental neurally functional elements. In addition, as for a 2D model the mandatory introduction of a fifth fundamental but neurally non-functional element blank (unused) cell B becomes obvious. Finally, in 2D landscape, a sixth fundamental element is necessary in order to allow paths to extend beyond their own enclosures, and we call that element a junction T. Based on the properties of each of these fundamental elements and their variations, the total number of distinct fundamental elements in our model rises to 93. An elementary 2D pattern is composed of five cells: a central cell and its four local cardinal neighbours. Hence, the total number of possible five-cell patterns (including both neurally valid and neurally invalid patterns) is exactly $93^5 = 6,956,883,693$ (almost seven billion). A generic neurally valid 1D sequence of our

---

[2] Balsa is at the same time a high-level hardware description language (HDL), a comprehensive development and simulation environment, and a synthesis system (hardware design flow) that can produce Verilog code as well as EDIF netlists in order to physically target reconfigurable devices such as our Xilinx XCV2000E, Spartan3A and Virtex5 FPGAs for prototyping our universal cellular neural platform.

elements aligned with either of horizontal or respectively vertical direction can be described in BNF as

$$..S \rightarrow T^* \rightarrow D^+ \rightarrow T^* \rightarrow D^* \rightarrow T^* \rightarrow N \rightarrow T^* \rightarrow A^+ \rightarrow T^* \rightarrow A^* \rightarrow T^* \rightarrow S..$$

Furthermore, one or more blank cell elements B, adjacent to each other or not, but never adjacent to a junction element, can be inserted in that sequence. In that case, a non-B cell adjacent to a B cell must have a dock connection $\uparrow$ in the other vertical or respectively horizontal direction (not shown here in 1D).

6.4 Self-Learning on PSoC: a holarchical learning model on digital neural tissue

The cellular automata-based neural network (CA-NN) sampled in [155] (once sufficiently grown further) is meant to discover and discriminate global laws of physics gathered from the outside world (as in typical applications of conventional NNs) and associate them to useful rules at the lower local cell level. This external-environment-oriented task and the previously detailed internal-system-oriented phenomena [155] must evolve concomitantly as in cellular programming [165]. As a result, this never-ending learning, growing, pruning and self-reorganizing hybrid CA-NN system (1) self-extends so as to become adequate to its static surroundings – self-upgrading property of holarchical digital neural tissue on programmable system-on-a-chip (PSoC) – and then (2) unceasingly self-adapts to dynamically changing environments it is exposed to (autonomous self-upgrading processes in place of conventional remote firmware updates). Universal self-extension is extremely interesting in that a robot (neuro-controller) having this faculty starts learning human linguistic and semiotic constructs from near-scratch points, and then keeps on enriching them so as to catch up with proper human communication. Meanwhile, the same machine can self-extend its skills in the same manner, but for tasks humans are poor or very poor at, and/or which are physically threatening, dangerous or lethal for any carbon-based organic being to tackle[3].

6.5 Rule Adaptation Method

The literature poorly reports successful work on automation of production of cellular automaton (CA) rules, because the methods that have been followed for this aim involve the use of genetic algorithm (GA) techniques, which do not work well for this specific type of problem, such as here the exploration and findings of 20,000,461 neurally valid five-cell patterns within seven billion. However, we have managed to boil the number of single rules per cell down to 495, of which 60 are obvious and should be predefined. Furthermore, the literature reports extensive work on rule discovery with neural networks [146]. In particular, feed-back (recurrent) NNs [166] are directly usable as associative memory due to their convergence to stable states. By associating neurally valid CA patterns with its stable states, a feed-back neural network (NN) based on our 2D asynchronous CA as neural tissue can readily perform the desired valid cells' rules production, which is another main novelty of this work. Thus, once sufficiently grown NN modules on our asynchronous CA-based neural tissue, higher level NN modules as bidirectional hetero-associative memories (BHAMs) pair local CA rules of their lower level (child) NN modules with higher and global CA-NN behaviours [33, 78]. We recall that in the flattened view of the system, the NN itself is based on that very same CA as well[4].

Such similar work where the result of NN processes lead to the growth and pruning of that very same NN, thereby inevitably modifying in turn some NN processes as well, has recently been successfully achieved [155]. Both works enable a looped bridge between ontogenetic (growth and development) and epigenetic (learning and processing) phenomena. While [155]

---

[3] In that line of philosophy (more distant from AI), super machines would emerge as both able to smoothly interact with humans and other animals (and indeed between themselves), and hopefully help in missions such as understanding (and ultimately harnessing) space-time and the universe. So far nobody has managed to do so: "no one is smart enough", M. Kaku [160]. These machines could either help accelerate progress of such missions, or indeed make most addressable at all.

looks at the NN level only, we also tackle the problem of generating rules at a lower CA level, by using the NN composed of that very same CA. When doing so, similar phenomena as above arise, accelerating maturation and bypassing time-consuming long-term phylogenetic iterations.

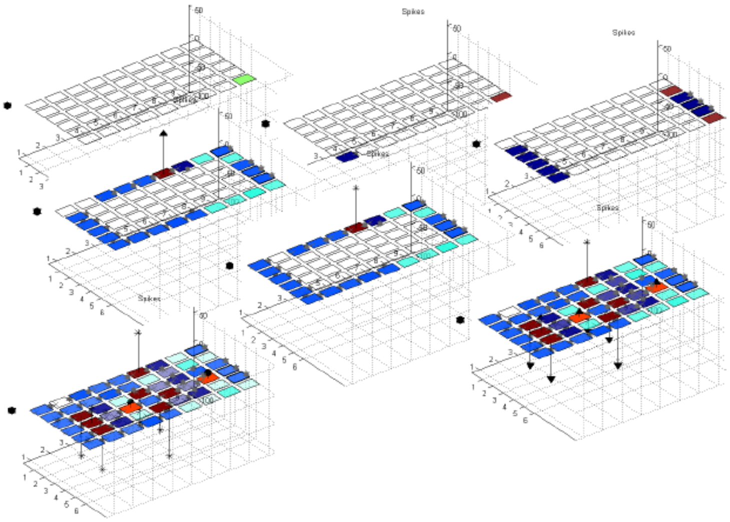


Figure 6.1: Birth of the CA-NN-based XOR seed: cells are spawned asynchronously and in parallel along the Balsa simulation time line. One picture is the representation of the seed state at one step of the Balsa simulation time. Thread of pictures in the figure must be read in the same order as thread of text on a page. Axon cells are blue, synapse brown, dendrite deep-blue, neuron red, blank cells aqua and first-time born-to-be cells white. Grey connections between cells indicate the state of their neural docks.

Once a cell's rule outcome has fired, our algorithm (one instance per cell) accommodates the four docks of the cell with its four neighbours'. If some central cell's docks cannot find compatible state with neighbours', it means the fired rule was a free one, which has the effect of

---

[4] In contrast, in Neurex [167], NNs were designed through an expert system (ES) [132]. ESes are less suited to learning from experience and to redesigning at NN run-time than NNs, and this limitation prevents the formation of sufficiently big NNs for them to be useful in real-world robotic applications. Hence our choice was to use a CA-NN-based BHAM (in place of ES) for the dynamic design and open-ended self-learning of how to design CA-NN modules: NNs mutually learn how to and design other interconnected NNs based on cellular programming techniques [165].

instead forcing neighbours to adapt to that central cell, now settled for longer time than most of its neighbours due to local cell's life- span credits, thereby propagating growth of NN patterns across the CA landscape then becoming a proper digital discrete neural tissue, e.g. cf. sneak preview in 6.1. The average number of activated free rules throughout the CA must be kept – by self-regulating thermostat – within boundaries that ensure the CA activity to remain within an 'edge of chaos' regime, away from freezing or oppositely destructive regimes. This number is interesting to trace, study and analyse relatively to the self-development direction and ALife style of the neural tissue.

6.6 Growing Design in VLSI: refinements

The above self-learning scenario requires an operating substrate. Here we make clearer and add a new scheme to the cellular neural substrate delineated in our prior works.

6.6.1 Compatible Neighbourhoods

In our paper [161], we gave the total number of unique neighbourhoods[5] NH of any cell λ as

$$|NH|_\lambda = nh_\lambda = \Gamma^k_{s_p} = \Gamma^k_{s_t \times d + b} = \Gamma^4_{5\times 2+1} = \Gamma^4_{11} = 1001$$

The 1001 neighbourhoods

We also calculated by composition the number of unique (compatible) neighbourhoods NHC $\subset$ NH, for each of which a neurally valid central cell λ c exists as

$$|NHC|_{\lambda c} = nhc_{\lambda c} = a_{iooo} + a_{ioo} + a_{io} + d_{iiio} + d_{iio} + d_{io} + n_{io} + s_{io} + b_{uuuu} + t_{iioo}$$
$$= 30 + 18 + 9 + 30 + 18 + 9 + 4 + 4 + 1 + 120 = 243$$

Finally, we only considered the scheme in which two junction elements can dock together directly and therefore we counted neurally valid any junction element as central cell having one or more other junction elements in its localmost neighbourhood. But the new* additional scheme we introduce as a mean to prohibit at will adjacent junction cells also involves a change in the number of unique neurally compatible neighbourhoods with a junction as central cell previously calculated as t_{iioo}, cf. Page 6, Column 1 of [161]. Now in the case where a junction cell cannot be directly connected to a junction – as central cell –, we have $t^*_{iioo} = \Gamma^{k_t}_{t^*_{io}} = \Gamma^{k_t}_{t^*_{io}+t^*_{id}+t^*_{in}+t^*_{is}+t^*_{it}} = \Gamma^2_{2+2+1+1+0} = \Gamma^2_6 = 21$.[6] Hence, the new* total number of possible unique neurally compatible neighbourhoods (always neurally valid with a type of element as central cell and a docking configuration) is

$$\begin{aligned} nhc^*_{\lambda c} &= a_{iooo} + a_{ioo} + a_{io} + d_{iiio} + d_{iio} + d_{io} + n_{io} + s_{io} + b_{uuuu} + t^*_{iioo} \\ &= 30 + 18 + 9 + 30 + 18 + 9 + 4 + 4 + 1 + 21 = 144 \end{aligned}$$

6.6.2 Neurally Valid 5-Cell Patterns

Besides, we have recalculated – by composition – the new* numbers (now smaller) of neurally valid five-cell patterns for each of the three submodels of CA presented that include only of either:

1) initial list of fundamental elements without junction elements (simplest and suitable for case studies), or,

2) with cross junction elements (necessary in real/big designs), or,

3) with cross junction elements plus compact junction elements (for industrial/more compact designs); when two junction cells cannot be adjacent. The results of these new* calculations are given in 6.1.

6.6.3 Rule Adaptation Mechanism: new scheme*

The eleven (sp = 11, cf. Page 5, Column 2 of [161]) distinct data values for representing information about one of eleven

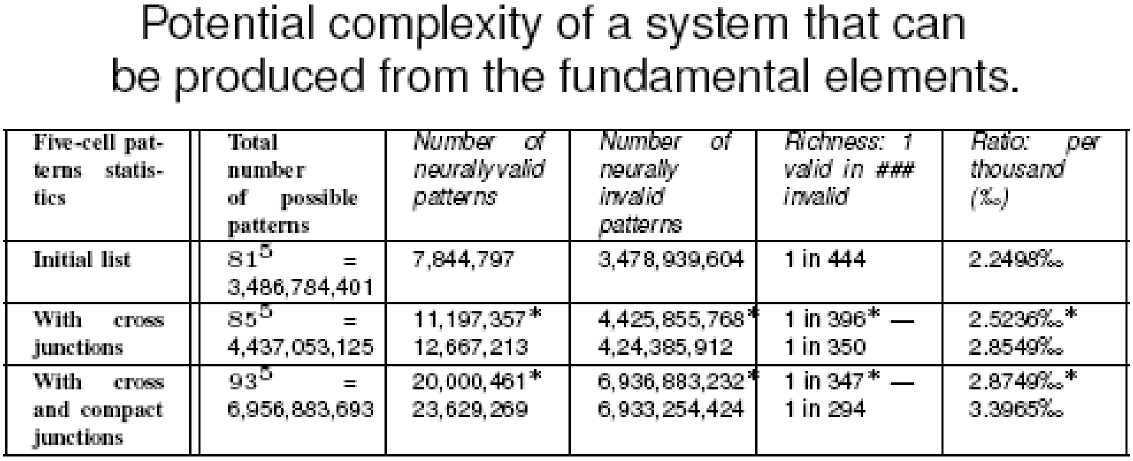
dock,type states of a neighbour (with only its directly visible dock considered here) as viewed from a central cell are ordered as follows due to previously chosen enum values given in [161]: U,B(=U,D=U,A=U,T=U,S=U,N)=0 (Any unconnected ‘U’ cell is considered as ‘B’ type by our algorithm explained in [161].), I,D=1, O,D=2, I,A=3, O,A=4, I,T=5, O,T=6, I,S=7, O,S=8,

---

[5] Unique in that other permutations of neighbours around a central cell are not counted since they are equivalent neighbourhoods from the central cell’s viewpoint.

[6] Note that the a_{iooo} through to the b_{uuuu} are unaffected by whether junctions are allowed to connect directly.

Table 6.1: Self-configured CA-based neural tissues on FPGA.



Potential complexity of a system that can be produced from the fundamental elements.

Five-cell patterns statistics	**Total number of possible patterns**	*Number of neurallyvalid patterns*	*Number of neurally invalid patterns*	*Richness: 1 valid in ### invalid*	*Ratio: per thousand (‰)*
Initial list	81^5 = 3,486,784,401	7,844,797	3,478,939,604	1 in 444	2.2498‰
With cross junctions	85^5 = 4,437,053,125	11,197,357* 12,667,213	4,425,855,768* 4,24,385,912	1 in 396* — 1 in 350	2.5236‰* 2.8549‰
With cross and compact junctions	93^5 = 6,956,883,693	20,000,461* 23,629,269	6,936,883,232* 6,933,254,424	1 in 347* — 1 in 294	2.8749‰* 3.3965‰

I,N=9, and O,N=10. For rule flows to operate properly, we found out that whether the user of our platform sets junction cells allowed to connect directly together or not*, junction cells always need to purport to one neighbour (say at the head of the internal junction cell arrow) that it is of the type state (say neuron) of its other dependent neighbour (say at the tail of the same internal junction cell arrow respectively)[7]. Furthermore, purporting (in the sense of purport to be a functional type of cell – not a junction cell) must be passed on from junction cells to adjacent junction cells for the new scheme*. Indeed, what is important is that a non-junction cell directly connected to one (or more) junction cell(s) knows the type state of the other non-junction cell(s) at the other end of the chain(s) of junction cells.We made our cell algorithm that processes cell rules take this purporting of junction cells into account, so the potential list R_p (with $|R_p| = |NH|_\lambda = nh_\lambda$) of 1001 unique rules was reduced to the necessary and sufficient rule table R_q sampled in 6.2. Since there must not be junction element at all in any rule antecedent anymore*, given $\begin{array}{l} |s_q = s_p - |[\{I,T\},\{O,T\}]|| = s_p - 2, with \\ s_v = s_t - |['T']| = s_t - 1, and \frac{s_q}{s_v} = \frac{s_p}{s_t} = |['I','O'] = d = 2| \end{array}$, the refined* total number of unique rules to deal with (per cell) is $r_q = |R_q| = \Gamma^k_{s_q} = \Gamma^k_{s_v \times d+b} = \Gamma^4_{4\times 2+1} = \Gamma^4_9 = 495$.

We have also figured out vital rule outcomes that have to be fixed since they each have a

---

[7] Same requirement for the other internal junction cell arrow.

fully neurally compatible antecedent. The vital fixed rule outcomes are in number of:

$$
\begin{aligned}
&r_{fv} = \sum : \\
&(a^*_{iooo} = a^*_i \times a^*_{ooo} = a^*_i \times \Gamma^{k_o}_{a^*_o} = 2 \times \Gamma^3_2 = 2 \times 4 = 8, \\
&a^*_{ioo} = a^*_i \times a^*_{oo} \times b = a^*_i \times \Gamma^{k_o}_{a^*_o} \times b = 2 \times \Gamma^2_2 \times \\
&1 = 2 \times 3 \times 1 = 6, \\
&a^*_{io} = a^*_i \times a^*_o \times \Gamma^{k_b}_b = 2 \times 2 \times \Gamma^2_1 = 2 \times 2 \times 1 = 4, \\
&d^*_{iiio} = 8,\ d^*_{iio} = 6,\ d^*_{io} = 4, \\
&n^*_{io} = 1,\ s^*_{io} = 1,\ b_{uuuu} = 1,\ t^*_{iioo} = 21) \\
&= 8+6+4+8+6+4+1+1+1+21 = 60
\end{aligned}
$$

Figure 6.2: The vital fixed rule outcomes

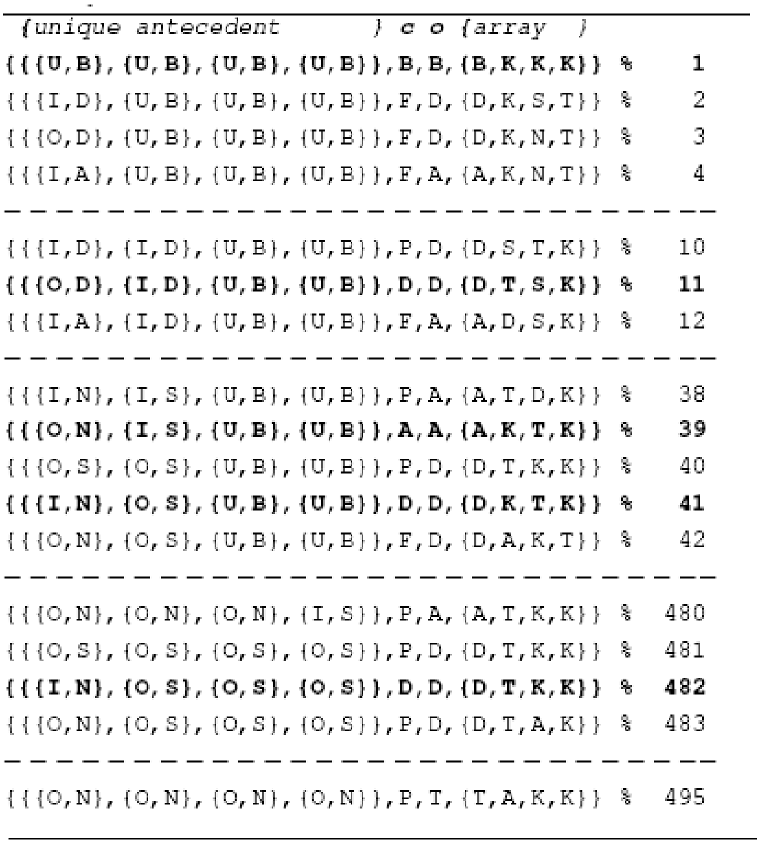
While for seeds that do not contain any junction cells we have instead $r^*_{fv} = r_{fv} - t^*_{iioo} = 60 - 21 = 39$.[8] Therefore, the new list of rules Rq that we have also generated with Matlab for ordering each (of the 495 - 60 = 435) four-variable array consequent (as potential new central cell state) via smartly weighting its corresponding partially neurally compatible antecedent (as neighbourhood) is:

A dynamically sorted cell's neighbourhood can immediately be matched with its unique corresponding line of the cell's rule table by finding its associated unique rule's antecedent (cf. 6.2) so that the adequate rule's consequent (in field 'c' for 'consequent') fires if it is fixed or is amended if it is equal to 'P' for 'probabilistic' or 'F' for 'free'. When a consequent is 'P' or 'F', our algorithm looks at the value at the right of 'P' or 'F' (in field 'o' for 'outcome') and either fires that value, or overrides it [rule adaptation] with one of the four available values for that rule (placed in the four-variable array field '$\{array\}$' at the rightmost of every rule entry). The one value out of four is selected during run-time based on overall probability statistics (and that value is written in field 'o' in order to prepare for next utilizations of that rule – until the value is changed again and so on). However, the 'P' or 'F' in field 'c' must never be changed by the algorithm since it indicates a rule for a 'partially' compatible neighbourhood, or respectively for a 'free' incompatible neighbourhood. Any other value 'B', 'D', 'A', 'T', 'S' or 'N' in that

---

[8] The set of these 60 (or 39) fixed rules is the same for each and every cell.

Table 6.2: New Rule Table generated with an M script, (without “purportive” junction cells (T) anymore in antecedents) – Line numbers are comments and not part of the table.



```
 {unique antecedent        } c o {array  }
{{{U,B},{U,B},{U,B},{U,B}},B,B,{B,K,K,K}} %    1
{{{I,D},{U,B},{U,B},{U,B}},F,D,{D,K,S,T}} %    2
{{{O,D},{U,B},{U,B},{U,B}},F,D,{D,K,N,T}} %    3
{{{I,A},{U,B},{U,B},{U,B}},F,A,{A,K,N,T}} %    4
-----------------------------------------------
{{{I,D},{I,D},{U,B},{U,B}},P,D,{D,S,T,K}} %   10
{{{O,D},{I,D},{U,B},{U,B}},D,D,{D,T,S,K}} %   11
{{{I,A},{I,D},{U,B},{U,B}},F,A,{A,D,S,K}} %   12
-----------------------------------------------
{{{I,N},{I,S},{U,B},{U,B}},P,A,{A,T,D,K}} %   38
{{{O,N},{I,S},{U,B},{U,B}},A,A,{A,K,T,K}} %   39
{{{O,S},{O,S},{U,B},{U,B}},P,D,{D,T,K,K}} %   40
{{{I,N},{O,S},{U,B},{U,B}},D,D,{D,K,T,K}} %   41
{{{O,N},{O,S},{U,B},{U,B}},F,D,{D,A,K,T}} %   42
-----------------------------------------------
{{{O,N},{O,N},{O,N},{I,S}},P,A,{A,T,K,K}} %  480
{{{O,S},{O,S},{O,S},{O,S}},P,D,{D,T,K,K}} %  481
{{{I,N},{O,S},{O,S},{O,S}},D,D,{D,T,K,K}} %  482
{{{O,N},{O,S},{O,S},{O,S}},P,D,{D,T,A,K}} %  483
-----------------------------------------------
{{{O,N},{O,N},{O,N},{O,N}},P,T,{T,A,K,K}} %  495
```

same field must never be changed neither since they are pre-set for the 60 (or 39) fixed vital rules. Value ‘K’ can only be found in fields ‘o’ and ‘*array*’ and denotes a passive outcome, that is, the central cell’s new state must be ‘kept’ same as its current state. Rule table in 6.2 shows samples of initialized rule entries for the adaptable rule table of a cell. Along the ALife of the neural tissue, every cell’s rule table starts from initial state in 6.2, but progressively differs from each other, which is why we said in [134, 161] that the CA model we consider has non-uniform rules property as for the definition of CAs in [6, 139]. This is for the neuro-controller platform to support indispensable cells’ differentiation, since also well-known to be vital in BLife.

6.7 Experimental Results

6.3 focuses on the subfunction of a submodule amid the PSoC holarchy that implements a basic sensory-motor neuro-controller. The self-learnt subfunction – shown to grow from scratch in [162] – has been identified as performing XOR discrimination of spike signals entering input (axon) cells (1; 3) and (1; 5), of which the neural process result shows up at output (neuron) cell (8; 4).

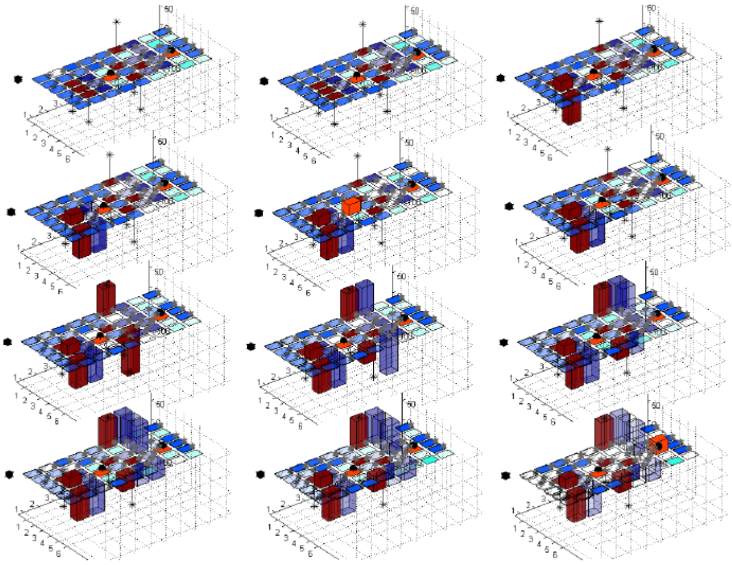


Figure 6.3: Frames represent the subsequent landscape states of the CA-NN spiking XOR subfunction. A bar denotes the instantaneous value of a spike signal for the corresponding cell. Vertical wires with asterisk show weights of synapses and dot thresholds of neurons (both signs change into 'or' at run-time as their corresponding cells learn, this as a mean to visualize on-line self-learning). Here the neural XOR is learnt, and is stable: cells' type states remain unchanged as rules have-self-resolved, yet ageing of cells is pictured by increased transparency of their corresponding bars along frames. Docks' type states also stable are shown as grey connections between cells. Note that this neural subfunction grew and learnt without the need of utilizing junction cells.

The subsequent subfigures in 6.3 depict the animation of the progression of the landscape of that 9x6 2D spiking CA-NN XOR subfunction under simulation. The first subfigure starts with input cells (1; 3) and (1; 5) having spike values 0 and 1. As time steps progress, the spike signals propagate from cells to cells in an asynchronous (here hidden as self-synchronized time points only are shown) and parallel mode of operation that can be visualized by subsequently looking at each subfigure in the same order as for reading this text. The last subfigure shows that the resulting value of the output neuron (8; 4) holds (and maintains, even though the rest of the subfigures corresponding to later time steps are not shown) the expected XOR output value in accordance with both the stream-maintained values of the input cells and the XOR discrimination.

6.8 Discussion on Self-Reconfiguration Methods

6.8.1 FPGAs, EHW and their Limitations

With the advent of new commercially off the shelf (COTS) chips (e.g. FPGA devices), dynamic and partial self-reconfiguration of microelectronic hardware circuits has become readily available and has been investigated for decades [92,168,169]. In particular, evolvable hardware (EHW) and especially intrinsic EHW have sounded promising [8], but other works [90,97] have clearly shown the limitations of using commercial FPGA devices for reconfigurability purposes. The main limitation we are concerned with involves having to use bit-streams external to the functional core system (co-processor).

Albeit bit-streams can be handled on-chip by using the internal configuration access port (ICAP) along with JBits [93, 122, 169], the scheme is not suited for our proposal of continuous growth and self-development of holarchical neural tissue as neuro-controller (our own co-processor). Indeed, a bit-stream or respectively a partial bit-stream encodes a functional core system (co-processor) or respectively parts of it. A functional core system may have capability for managing bit-streams including those that represent that very same functional core system. Along this thinking arose the research area of EHW, in which bit-streams are treated as genotypes, and functional core systems (co-processors) as their phenotype counterparts.

However, this scheme allowing for radical design modifications also imposes discontinuous refinements of a coprocessor design, the refinements typically having to go through heavy off-chip EDA software tools or on-chip mechanisms that include ICAP. Even with such on-chip mechanisms, bit-streams must be produced and uploaded into the reconfigurable FPGA device, which necessarily involves discontinuity of ALife and therefore untrue real-time operations with time lapses that dramatically increase the vulnerability of the co-processor and significantly decrease its instantaneous integrity.

6.8.2 CellMatrix

Alternatively to the above, CellMatrix [163] devices have been developed. In a CellMatrix device, self-reconfiguration

takes place from within functional core circuits of logic cells. Several CA features of CellMatrix differ from the ones proposed hereby. CellMatrix CA edges are open, which implies bounded landscapes. Furthermore, no access to internal cells from outside was purposely implemented (probably to make evident and show-off a radical distinction with FPGA architectures). And finally, cells embed low-level electronic logic functions.

In contrast, our proposed hybrid CA-NN model for digital discrete neural tissue formation from within and from gradually self-gained reconfiguration understandings of CA stored in on-chip holarchical NN [161] may have no edges (self-closed edges) and therefore allows for boundless landscapes. This is mandatory for modelling autopoietic ALife systems [131], with BLife creatures and our cosmos being perfect examples [170]. Access to all internal cells from outside is permitted (at least for monitoring purposes, but also control purposes if required, in order to ease analyses and debugging of emerged designs) like for FPGA. And finally, cells assume higher-level neural processes built upon low-level electronic logic functions.

In CellMatrix and in FPGA, low-level electronic logic functions require handling complicated and huge amount of parameters to be configured and routed[9]. Oppositely, we have shown that feasibility for readily handling higher-level neural processes described in [161] and hereby is strongly encouraging. On the one hand, CellMatrix devices can in principle implement our proposed neuro-controller: lowlevel electronic logic functions can implement higher-level neural processes. On the other hand, well-established FPGA devices with their now mature EHW methodologies remain a hardware substrate of choice[10].

Therefore, FPGA devices are the hardware target adopted for this ongoing research, while prototyping an in-house ASIC solution (also using FPGA for that purpose) in view of enhanced performances and foundation of custom mechanisms (freed from standard FPGA or CellMatrix protocols and constraints) – thereby opening the way to industrialization of our highly self-adaptive neuro-controller device – keeps running as another piece of ongoing research and development work in the background.

6.9 Conclusion

The created and adopted framework has involved development of an evolvable micro-architecture of neuro-controller with capability for self-tuning to interfaced robot bodies of arbitrary shapes, with possibilities of autonomous hardware design as well as with programmability of behaviours through high-level interactions rather than low-level coding. Nevertheless, we have shown that the prototype platform provides means to visualize and monitor low-level internal states and

---

[9] As an illustration, going down to lowest-level physical entities typically requires handling hyper-complex and astronomical numbers of parameters to try and come up with any viable blueprint – and that is why Physics seems most challenging with unfortunately ongoing poor indication on investments' returns [160].

[10] With reference to the POEtic model – the phylogenetic, ontogenetic, and epigenetic axes of life [42,171], whether ALife or Blife –, our work primarily focuses on followed-up continuous growth and development – ontogenesis – and on sensible consistent learning and processes – epigenesis. Yet, obscure blind evolution of populations – phylogenesis – of digital discrete neural tissues as for finding 'seed design' solutions of neuro-controller from near-scratch points might prove useful or necessary and will be readily accessible through EHW. In that case, allowing for our autopoietic system to amend (re-design) electronic logic functions underlying its neural processes as presented hereby has potentials for greater self-improvement capabilities.

parameters of the digital neuro-architecture as it concomitantly operates and self-reconfigures seamlessly during run-time.

Finally, the natural chain mentioned early in this chapter insinuates that if such system can design itself – from within –, then, later, it can design new hardware systems of other natures, for example by using EDA tools more efficiently than human design experts, or more likely and more evidently via its own means.

Having to use COTS devices and tools in order to be able to implement and experiment our self-adaptive and self-organizing systems, which sound exotic as compared to conventional well-established (COTS) systems and embedded (sequential software) systems, is cumbersome. Therefore, there is an obvious and serious lack of 'exotic' platforms – and nothing really established, perhaps apart from [163] – for straightforward implementations and experimentations with promising self-* architectures.

A better way towards a universal self-organizing platform that can natively communicate smoothly with humans is to combine self-design philosophy at fine-grain level of digital logic [163] with self-organisation and self-adaptation at coarser-grain level of CA and NN components. In this paradigm that we have been investigating, outputs from higher level processes such as CA and NN processing results are self-generated for lower level digital logic designs of new cells as well as new properties of existing cells.

This work attempts to make a difference with robotics so far mainly pre-programmed only, despite the recent advent of the embedded reconfigurable computing paradigm [172]. The dexterity of robots mentioned in [161] is being refined for fluid rather than stiff and overly controlled robot motions, which goes along with freedom for cells to drop spike signals. Finally, this chapter has laid the efforts made to pave the way towards solutions or rather capabilities for machines and robotic systems to make up ("good", least worse or at all) decisions on-the-fly as they face unexpected, unpredictable, unknown, new situations. This is most useful as a whole when those situations emerge within human-unfriendly environments, in which those machines perform at ease while communicating seamlessly with the people directly and indirectly involved.

So, shall we design, create and build our autonomous autopoietic "ALife" extensions and trust them so that eventually they show us how to better design hardware and software (and why not harness wormholes or even better help build a teleportation/time machine and so on)? Or shall we hold on the little control we have on ourselves, on our tool and service machines, and on Nature, never give it up, and perhaps then never make it higher and better?

6.10 Summary

To-day no good solutions are known for the problem of a performing machine immersed within static or dynamic environments and meeting unexpected, unpredictable, unknown, new situations. Perhaps because no method can be found for the direct design of solutions that would meet this requirement, furthermore now in great demand. This chapter presents the work we are developing to address that particular problem, of which abundant solutions ironically appear second nature to myriads of biological creatures.

Improved on-chip circuit densities have enabled the practical realization of increasingly demanding applications. Microelectronics design now faces a number of challenges: hardware has become more complex to describe making it a more arduous process for designs to pass verification and the proportion of a design that is covered by testing is reduced increasing the likelihood of bugs in the final hardware device. To meet these challenges while more effectively exploiting the larger silicon areas now available, methods that enable autonomous design have become highly desirable and thus novel ones are proposed hereby. ï»¿

Chapter 7

Dynamic Rule Learning in Cellular Neural Network Design – on an autonomous design chip

This chapter radically improves upon the works presented in the previous chapter by devising and incorporating a method to produce, use and discard rules on-the-fly, which bring several advantages without any drawback due to the parallelism of algorithm processes despite being a little bit more time consuming in this new version with rules blueprint. A neat advantage is the introduction of rules' policies, which give meaningful control over the system.

7.1 Introduction

Autonomous manoeuvring machines such as mobile robots have gained a great interest in recent years. The machine regarded as an external indicator on the level of its internal intelligence can demonstrate highly dynamic dexterity. From the 1^{st} order and 2^{nd} order cybernetics point of views [131], together with the consideration of direct applications into human services, the learning scale has been framed to real-world constraints. Hence, it is preferable to encode cellular programmes [42] *in silico* to obtain the ability, which is similar to the 'Natural Selection' theory. The main novelty in this project is that the developed system comprises the modules with never-ending evolution, adaptation and learning capability. The whole system is scaled up to different levels of modules and the communication between the levels is accomplished by recursive instantiation from the utmost modules to the lowest modules, which is called 'holarchy' (quasi-fractal structure). Here, the spiking neural network modules which are based on the biologically-inspired asynchronous non-uniform cellular automata can grow and self-organize in microelectronic hardware (DPR-SoC). The modules on the lower level are interfaced with external sensors and actuators to adapt their behaviours online in order to obtain the required skill/task demands. With the aid of neural network (NN) processes in these modules, the local rules of the modules in its own underlying non-uniform cellular automata (CA) models will form new NN patterns. The interaction between CA models and NN allows the modules to develop the mapping module behaviour and growth styles with corresponding rules for self-improvement based on the surrounding situations. The control system with the ever self-upgrading initial kernel design can achieve the functions of self-learning using the sensors and actuators, maintaining its internal and proximate external resources and learning on-line for specific requirements through action-reward-based training [79]. The reliability and accuracy of the information from the sensors and actuators should be evaluated and optimized before it enters the system processing loop to improve the robustness and fault tolerance ability [173].

The remainder of this chapter is organized as follows. The previous work involving rule learning algorithm and relevant background and techniques are summarized in Section 7.2. The new rule learning strategy and method in CA is introduced in Section 7.3. Simulation results with the proposed method are shown in Section 7.4. Discussion and analysis of the results are also given in the same Section. Conclusion and future work about the self-updating autonomous chip design is discussed in Section 7.5.

7.2 Previous Work

7.2.1 Background

The traditional artificial intelligence (AI) technique has major defects since the host hardware computer machines are data driven processors which are unable to deal with semiotic and linguistic constructs. The current stream for the embedded software development together with on-chip software is limited for the data transferring capabilities. The on-board buses having relative low bandwidth constitute permanent speed bottlenecks of the whole system [161]. Inspired by biologic organism of cells in living systems, machine learning techniques, i.e., artificial cellular neural networks [79] have been developed diversely in the recent decades. A cellular automata model shows similar characteristics to that of a biological cell pool, therefore it is chosen to be the proposed kernel control system. Two dimensional CA can nicely map onto the commercially off-the-shelf (COTS) hardware devices such as FPGA (field programmable gate array) [168], which is shown in 7.1. One particular hardware description language (HDL), Balsa, is selected due to its design freedom and creativity as well as built-in handshake mechanisms for asynchronous operations. Although synchronous design is still the current popular trend in industrial practices, data/event-driven asynchronous protocols have been used in the embedded systems gradually.

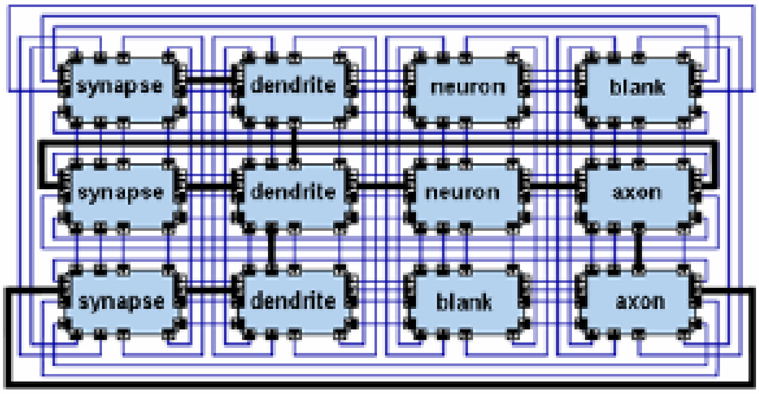


Figure 7.1: Self-configured CA-based neural tissues on FPGA.

7.2.2 CA Model

Several update schemes for asynchronous CA models can be found in [174,175]. The CA model has the inherent robustness since the faulty units can be replaced by the idle neighbouring ones. Here, each cell is considered to interact with its four local neighbours at its cardinal points (see 7.2) which allows it to make correct decisions based on hard-coded basic local transition rules to keep neural compatibilities between adjacent docked cells. In 7.2, the central cell is a junction element, which purports its state to insure valid neural connections between its non-junction neighbours, as labelled 1, 2, 3 and 4.

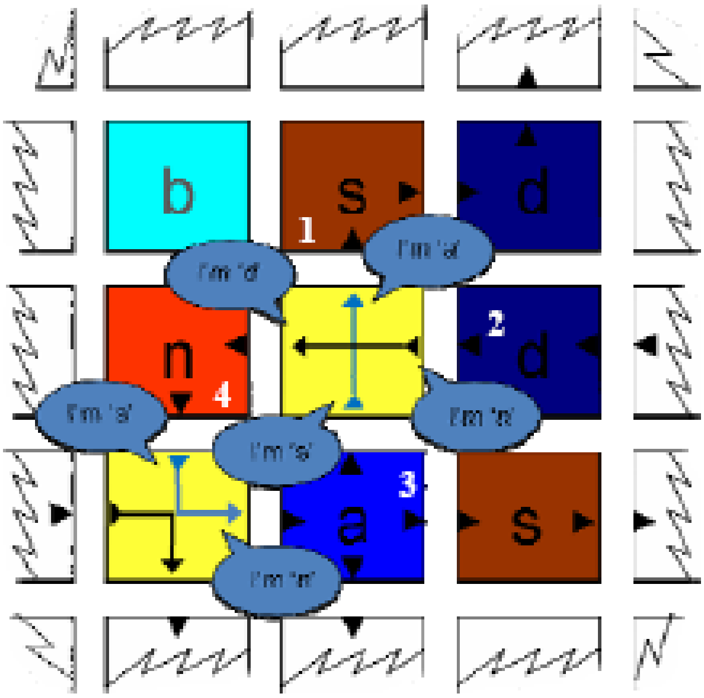


Figure 7.2: Central cell and its four neighbouring cells at cardinal points.

The update of a cell type state is allowed only when the private cell timer has elapsed (cell dead or being born), which permits local coupling of cells to achieve self-synchronization globally. Yet, the update of dock states can happen anytime to ensure at least one neural pathway available constantly, whereas exchanges of spikes may only occur while the timer is counting (cell alive). Unlike in [176] where each cell is regarded as a neuron, here a cell holds one of six states: Neuron (activation transfer function), Synapse (weight), Axon (output distribution, attenuation rate and propagation delay), Dendrite (input sum), T (junction) and Blank (neural interstice), which are expressed by N, S, A, D, T, and B respectively. At the CA level, docked cells are liable to exchange spikes of signals through neural channels. The asynchronous circuitry constructs in Balsa can implement handshake protocols to support spiking communications efficiently [177]. CA configuration with diverse sets of local rules involving global behaviours has been studied in many papers [139, 174, 175]. In this CA model, each valid pattern state is composed of a central cell state, its own four docking states and the four local states of its neighbouring cells. The number of possible five-cell pattern states is of order 10^7 [161, 162] and each state needs a transition rule, which is barely possible to be coded manually. Hence, it is necessary to choose the local CA rules automatically according to the designated criteria. Besides, the rules should be able to accommodate new situations instead of the only fixed ones.

7.2.3 CA Rules and Update Theory

The neural network is the programmer of the rules to dictate the behaviours of its underlying CA. Given an initial emerged NN, it will grow to perform primary basic tasks of recognition. The recognition includes several main aspects. The first task is the rule discovery, to discover useful CA rules and recognize relevant patterns via inductive inference. The second task is the rule amendment, in which NN rewrites, overrides and adds new CA rules. The third task is to keep the NN behave healthily and efficiently and allow the NN to grow more adaptively. All these tasks are executed and considered parallel at the same time during operation. There are four types of rules in the CA considered here:

1 Basic rules: neural compatibilities to ensure the following sequence:

$axon||input > synapse > dendrite > (dendrite >) > neuron > axon > (axon|output >)$

2 Growing rules: to explore network topologies and connection styles;

3 Learning rules: to affect NN learning pattern considering the new situations;

4 Maintenance rules and performance rules: to evaluate the NN learning capabilities.

The CA model possesses the self-learning and self-update functions. The hosted neural network is composed of cells, and it needs to decide the next states of the cells automatically by comparing the current states of the neighbours and the pre-designed state set.

7.3 Method of Rule Adaptions

7.3.1 Rule Number

The number of the rules in CA is dependent on the number of the cell states, the number of neighbouring cells (interface) and the level of the adaptability of the algorithm. The rule number will be increased with the number of cell states and neighbouring cells but decreased with higher level of adaptability of the algorithm (per cell) that processes the rules and routes of the neural docks. The multiset coefficient, Gamma, which resembles the binominal coefficient, is used to calculate the unique rule numbers. The value of Gamma(k,n), chosen k elements (with repetition allowed unlike C(k,n)) from elements is calculated as:

$$\begin{aligned}\Gamma_n^k &= \left(\!\binom{n}{k}\!\right) = \tfrac{n(n+1)(n+2)...(n+k-1)}{k!} = \tfrac{(n+k-1)!}{k!(n-1)!} \\ &= C_{n+k-1}^k = \tbinom{n+k-1}{k} = \tbinom{n+k-1}{n-1}\end{aligned} \tag{7.1}$$

Where C(k,n) is the well-known multinomial coefficients for combination calculation.

In this CA, a (central) cell is concerned with the states of its four (k=4) neighbours at its cardinal points. A neighbour viewed from the central cell can be in either of nine (n=9) states: N, S, A, or D (ns=4, note that state T is removed as junctions always purport their state as if of their non-junction neighbours), each with either of input or output dock (d=2), plus state B also assimilated with unconnected dock regardless of the state (b=1). Hence, there are

$$\Gamma_n^k = \Gamma_{n \times d+b}^k = \Gamma_{4\times 2+1}^4 = \Gamma_9^4 = 495 \tag{7.2}$$

unique rules as **antecedents** (corresponding to 495 unique distinct **neighbourhood**'s states as viewed from central cell), to each of which a *next state* (either of N, S, A, D, T or B) for the central cell as a *consequent* must be assigned. As referred in [161], it showed that there are 60 (or 39*) unambiguous next states as rule consequents to be assigned to 60 (or 39*) rule antecedents. However, the rest 435 (or 456*) are ambiguous when the central cell morphs into the next states. These must be weighted, mainly according to policies [150] such as favouring connections that are biologically realistic (conglomerates of dendrites and long chains of axons) or opposite to ensure neural pathways of the dendrites and axons connecting directly to neurons and synapses, more often than to their peers. The self-adjustment policies by the self-learning NN allows the system itself to ensure: the CA operates in the desired regime near “the edge of chaos” [175]; good trade-off between biologically realistic configuration and actual-hardware fitness of the CA is maintained; self-improvability is maximized [150].

7.3.2 Background

The initial rule table (see 6.2 as an example) is generated offline in Matlab with certain fixed policy. 6.2 delineates subsample lines of such an initial rule table. At the last column of the table, the seven-element array represents the seven preliminarily ranked potential outcomes, N, S, A, D, T, B or K, from which to choose the next state for a central cell. The K potential outcome as in ‘k’eep, denotes passive outcome to fulfil all possible states. As a CA runs on-chip (in hardware), each rule table becomes unique because the cellular neural network (formed by the CA) has the authority to amend the initial rank in any of those 435 (or 456*) potential array-outcomes in the rule table of any cell so that "cell differentiation" emerges, similarly in biology from necessities. P value in the field ‘consequent’ of a rule – amid those 435 (or 456*) rules in non-bold font in 6.2 – denotes the cell that this rule is subject to ‘P’robabilistic outcomes and hence to look up the effective outcome state in the field ‘outcome’ at the left of the arrayed potential outcomes. F value denotes that NN is ‘F’ree to choose an outcome state.

At a higher level than that presented in this chapter, the cellular neural network as a bidirectional hetero-associative memory (BHAM) learns and amends those ranks in order to converge the asynchronous CA’s growth towards desired “edge of chaos” regime [175]. So the heavier potential next state at the leftmost in an array may become different from the effective outcome state already in place, located in the first index of the last seven-element array of a rule row. At certain moments the NN can override the value in ‘outcome’ with the value from the leftmost member in the array. It is worth noting that values in ‘consequent’ and ‘antecedent’ are fixed in the initial off-line production of the rule table with Matlab. Although the rule table is the mainstream in CA, it takes too much space in silicon area to store such a big rule table per cell. Hence, an improvement should be made to alleviate that issue, which could bring better performance at the same time.

7.3.3 Dynamic Rule Learning

Here, a single rule is provided as it is needed at run-time on-chip. This is a major improvement of the CA design and it has several notable advantages over the previous implementation described above. The Matlab code is transferred to Balsa code for run-time hardware computation. However, there is no need to produce 495 rules for each cell at once anymore, whether off-line or on-line during normal operation. Instead, one cell will only produce one outcome (corresponding to the cell’s current neighbourhood’s state as rule’s antecedent) based on the policy. The policy can now also be local to a CA region or even down to the single concerned cell so as to provide diverse policies as advocated by cybernetics [131].

During computation for the production of a required consequent, the policy-based weights are distributed to the six potential next states for the central cell. This can further lighten the burden of implementation since the weights to the previous seventh (virtual) state K in Matlab-generated rule tables now go to the current state of the central cell instead.

Clearly, there is no need to hold a ranked record of six or seven rule’s consequents as the next state potentials since an actual winner, the next state, will be produced on demand in any time when at least one neighbour changes its state (instead of looking up rule table when this occurs as in the previous implementation).

Furthermore, in the new implementation of rule learning in hardware, the computation time for the production of a rule can be shortened by aborting the weighting process when a consequent candidate for the central cell is found to fully match with its four neighbours. There are 60 or (39*) exclusive cases out of 495 as explained in the previous subsection. For each of these 60 (or 39*) cases of neighbourhood, only one unique next state out of six potentials will neurally fit the neighbourhood without ambiguity. So there is no need to carry on the computation in search for other potential next states anymore. Similarly, as soon as the weight of one of the six potential outcomes goes over 50

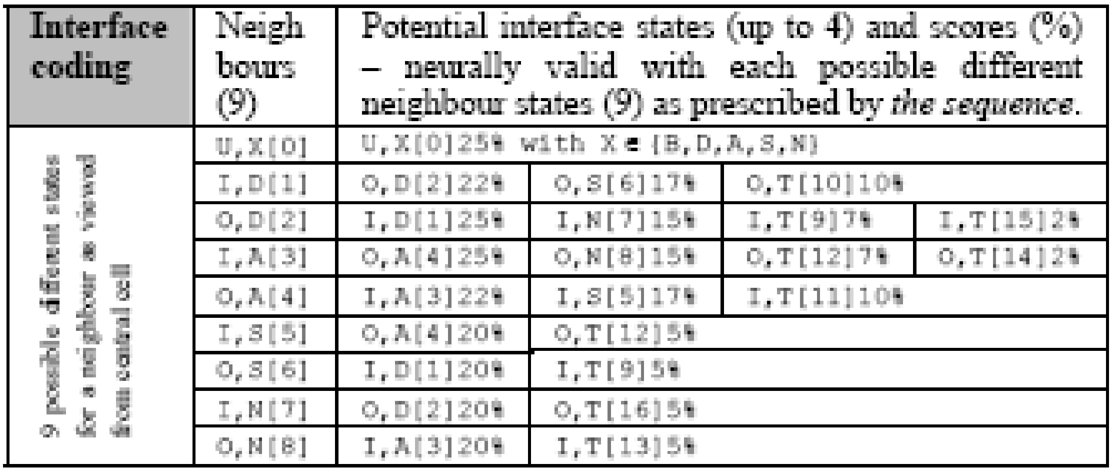
7.3.4 Improvement: Comparisons of Implementations

In the previous implementation, each cell of a CA landscape needs to read upfront and hold an evolving record copy of an initial 495-line rule table (shown in 6.2). In any line, the antecedent is a four-element array that takes 4*5(bit) = 20 bits of hardware to be stored in silicon as described in Balsa. The consequent and outcome fields both take 2*3(bit) = 6 bits. Moreover, the ranked potential outcomes shown in the seven-element array in actual hardware code taking 7*3(bit) = 21 bits. Hence, the necessary hardware width of a single row to store one line of a rule is h(w) = 20 +6 + 21 = 47 bits, which is rounded to h(w)* = 48 bits = 6 Bytes for compatibility with software tools such as Balsa simulator and Teak compiler into Verilog. Consequently, one cell needs

$$h_{rt} = h_w^* \times h_l = 48 \times 495 = 23760(bits) = 2970(Bytes) \tag{7.3}$$

which is nearly 3kB to keep a copy of its rule table at hand. In the new implementation presented in this chapter, a cell needs one irregular 9*4 -element matrix shown in 7.1. It means 9 possible different neighbouring states (refer Eq.2) and up to 4 interface states, which takes 9*4*5(bit) = 180 bits to store the neurally valid interfaces. In Table II, ‘U’, ‘I’, ‘O’ delegates the unconnected, input and output dock states, respectively; the value in the bracket represents the code of the possible states; the percentage value after the squared bracket is the initial policy scores assigned off-line and it can be changed by NN on-line afterwards. It also needs another one irregular 9*4-element matrix that takes 9*4*6=216bits to store the scores of the corresponding interface states associated to diverse policies (that can be changed dynamically at run time). Therefore, only maximum 180 + 216 = 396 bits are needed for the interfaces and the scores.

Table 7.1: Neural interfaces and a preselected set of their scores (%)



Interface coding	Neigh bours (9)	Potential interface states (up to 4) and scores (%) – neurally valid with each possible different neighbour states (9) as prescribed by *the sequence*.			
9 possible different states for a neighbour as viewed from central cell	U,X[0]	U,X[0]25% with X ∈ {B,D,A,S,N}			
	I,D[1]	O,D[2]22%	O,S[6]17%	O,T[10]10%	
	O,D[2]	I,D[1]25%	I,N[7]15%	I,T[9]7%	I,T[15]2%
	I,A[3]	O,A[4]25%	O,N[8]15%	O,T[12]7%	O,T[14]2%
	O,A[4]	I,A[3]22%	I,S[5]17%	I,T[11]10%	
	I,S[5]	O,A[4]20%	O,T[12]5%		
	O,S[6]	I,D[1]20%	I,T[9]5%		
	I,N[7]	O,D[2]20%	O,T[16]5%		
	O,N[8]	I,A[3]20%	I,T[13]5%		

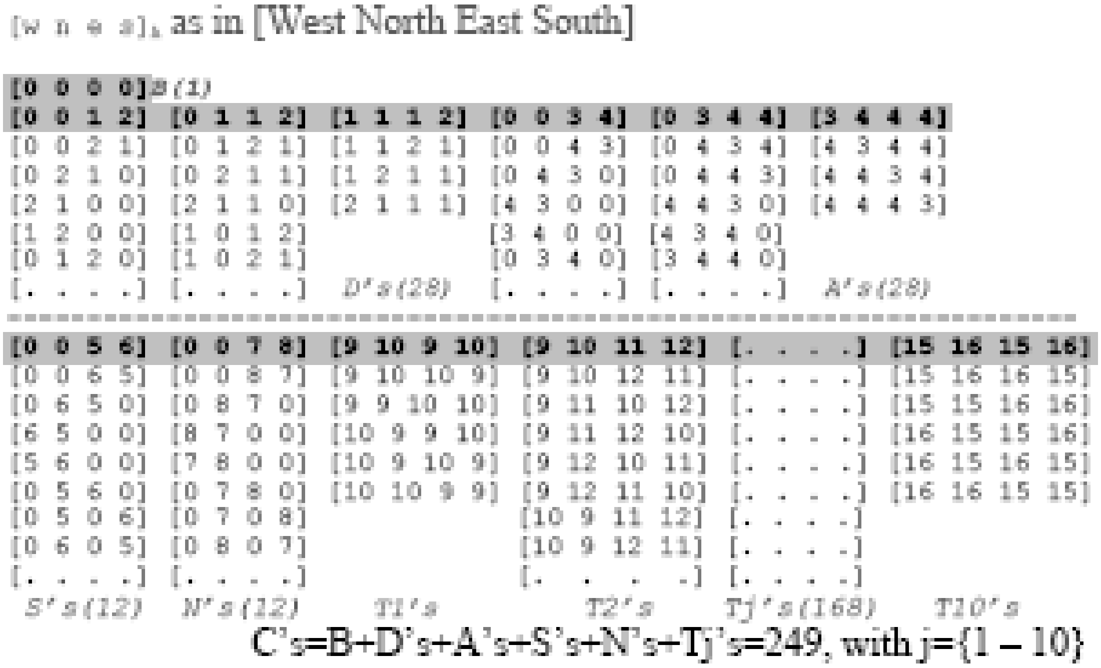
In addition, a cell needs to store the blueprint from which every actual cell's states (including its 6 possible type states and its **four three-state docks**), considered as candidates for the next state of the central cell, can be built from the blueprint, used and discarded, one by one at run time. The potential interface (pi) constructs are calculated as follows:

$$1^4 \leq x_{pi} = w_{pi} \times n_{pi} \times e_{pi} \times s_{pi} \leq 4^4 \tag{7.4}$$

Where w,n,e,s are the number of the potential interfaces for each of the four cardinal orientations, respectively. The matching process can be completed by comparing the 249 candidates against the x(pi) within the nested loop of the cells' algorithm. Indeed, 249 distinct actual candidates (C's) can be unveiled if the blueprint is unfolded entirely. 7.2 shows the partially unfolded blueprint with its distinct 19 seeds, which are emphasized in grey shade.

Although the total number of such states of an actual cell is 249, it does not need to be stored due to the run-time production-use-withdrawal mechanism in the new algorithm, thereby sparing a large number of hardware bits. This blueprint is a 19×4-element matrix that takes

Table 7.2: Partially top-down unfolded blueprint consisting of unique permutations of the blueprint seeds.



```
[w n e s]ᵢ as in [West North East South]

[0 0 0 0] B(1)
[0 0 1 2] [0 1 1 2] [1 1 1 2] [0 0 3 4] [0 3 4 4] [3 4 4 4]
[0 0 2 1] [0 1 2 1] [1 1 2 1] [0 0 4 3] [0 4 3 4] [4 3 4 4]
[0 2 1 0] [0 2 1 1] [1 2 1 1] [0 4 3 0] [0 4 4 3] [4 4 3 4]
[2 1 0 0] [2 1 1 0] [2 1 1 1] [4 3 0 0] [4 4 3 0] [4 4 4 3]
[1 2 0 0] [1 0 1 2]           [3 4 0 0] [4 3 4 0]
[0 1 2 0] [1 0 2 1]           [0 3 4 0] [3 4 4 0]
[. . . .] [. . . .] D's(28)   [. . . .] [. . . .] A's(28)
==================================================================
[0 0 5 6] [0 0 7 8] [9 10 9 10] [9 10 11 12] [. . . .] [15 16 15 16]
[0 0 6 5] [0 0 8 7] [9 10 10 9] [9 10 12 11] [. . . .] [15 16 16 15]
[0 6 5 0] [0 8 7 0] [9 9 10 10] [9 11 10 12] [. . . .] [15 15 16 16]
[6 5 0 0] [8 7 0 0] [10 9 9 10] [9 11 12 10] [. . . .] [16 15 15 16]
[5 6 0 0] [7 8 0 0] [10 9 10 9] [9 12 10 11] [. . . .] [16 15 16 15]
[0 5 6 0] [0 7 8 0] [10 10 9 9] [9 12 11 10] [. . . .] [16 16 15 15]
[0 5 0 6] [0 7 0 8]             [10 9 11 12] [. . . .]
[0 6 0 5] [0 8 0 7]             [10 9 12 11] [. . . .]
[. . . .] [. . . .]             [.  .  .  .] [. . . .]
 S's(12)   N's(12)     T1's          T2's     Tj's(168)    T10's
         C's=B+D's+A's+S's+N's+Tj's=249, with j={1 – 10}
```

$$C's = B + D's + A's + S's + N's + Tj's = 249, with\ j = \{1 - 10\}$$

19*4*5(bit) = 380 bits of hardware. It is the same for every individual cell and must not be changed at run-time. Only a single copy could be stored at a global level and parsed by any cell if necessary. However, such mechanism would neatly compromise the parallel asynchronous operations as well as the speed of execution since all cells would need to share the content of that blueprint, but only once at a time. Therefore, at present a cell needs just 396+380 = 776 bits = 97 Bytes on the hardware.

7.3 depicts a particular exemplary neighbourhood. Its corresponding neurally valid potential interface constructs (here x(pi) = 1*3*4*4=48) as defined in 7.1, are compared with the 249 cell candidates C(lambda) one by one. The four neighbouring cells (d,b,a,b) of the central cell are in blue colour and they are delimited by the bold squared box. The cell algorithm will compare 249 times against every four-combination of potential interfaces x(pi)(yellow rectangle shade in

7.3) and assign the resulting matching weight to the container associated with the type state of C(lambda) being used (active rules). If there is no match, the resulting weight will be assigned to the current state of central cell (passive rules). The process involves comparing every single interface of an interface construct (such as the combination emphasized with grey elliptical shade in 7.3) with every constituent (four sides) of a cell candidate, and adding the score contribution of the interface to the weight container (6 containers to hold the weight of the potential states) for the type state of that candidate according to the level of matching. The matching is taken in decreasing order as: type states and connected dock states match, type states and unconnected dock states match, only *unconnected* dock states match, only type states match, only *connected* dock states match, mismatch.

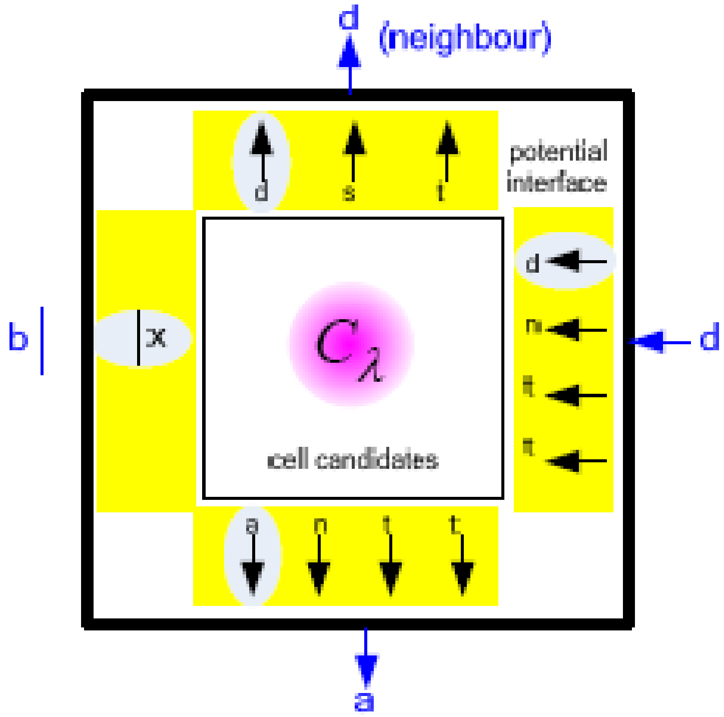


Figure 7.3: A neighbourhood of a central cell

7.4 gives the insight into the process of comparing C(lambda) with interface constructs of an exemplary neighbourhood. Here, the coding behind symbols of 7.4 are revealed and should be referred to 7.1 and 7.2. The potential interface yellow rectangle shade There is no significant difference of hardware usage in the two algorithms. In the old algorithm [161], a cell will look up its rule table and make a route for its four docks based on its current neighbours' states. In the new algorithm, however, the cell will rather produce rules' outcomes at run time by using the interfaces, their associated policy's scores, the blueprint for actual cell's states of candidates and the current central cell's state for contributing to increase the weight of passive rules (formerly the virtual state K) as well as routing its four docks, altogether based on current four neighbour's states and locally applicable policy.

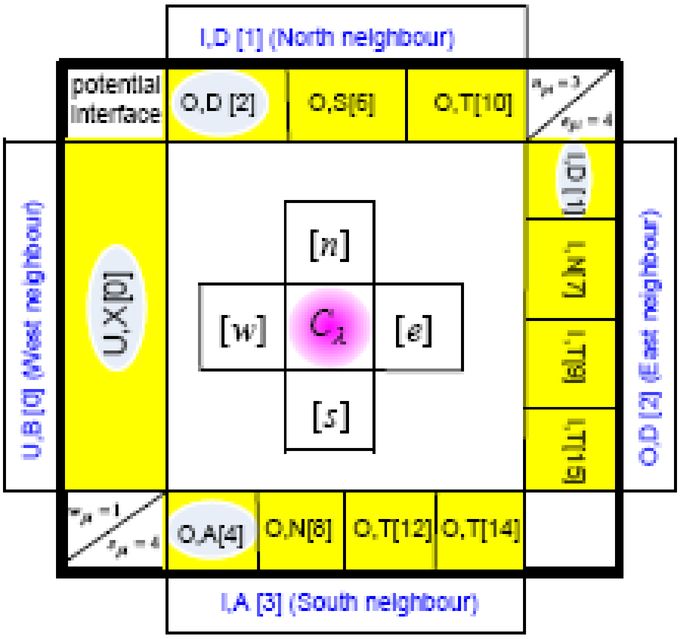


Figure 7.4: Insight into the process of comparing C(lambda) with interface constructs of an exemplary neighbourhood

7.4 Results Analysis

Previous simulations were limited to a CA landscape of about 9*6 totipotent cells (cf. the cellular neural XOR in [162] that includes only two neurons), beyond which both the Balsa simulator and synthesizer will fail due to a segmentation fault of the host memory. With the major improvement presented in this chapter, the latest experiments show that even with CA landscapes beyond 40*40 totipotent cells, both simulations and synthesis withstand normal operations. In addition, an excerpt from the latest experimental hardware simulations [176] shown in 7.4 has demonstrated that the simulator runs much more smoothly and the actual in-house DPR-SoC hardware operations run ten times faster, based on the previous cellular automata configuration as neural XOR function design [162].

A CA landscape is made up of more than 9*6 = 54 cells crashes experimentally at run-time or at synthesis time with 3kB of reserved hardware per cell as in the previous implementation, which is 54*3 kB=162 kB for the whole landscape. Hence, in the new implementation where the code occupies 97 Bytes per cell, calculation shows that it would take more than 162kB/97B=1,670 of these newly coded cells to crash the hardware simulator or the synthesizer.

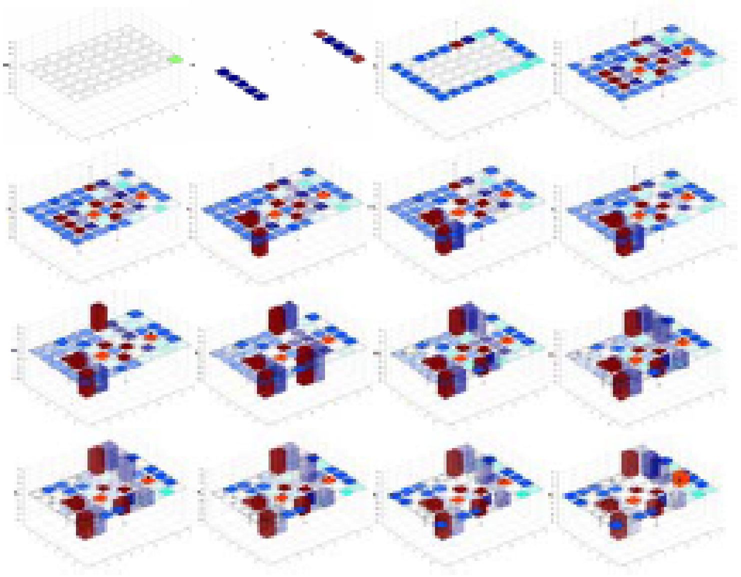


Figure 7.5: Self-synchronised hardware learning spiking network with new policy-based rule adaption algorithm: from birth to its mature forms.

That number of cells is largely sufficient in order to instantiate a single module with no more than 100 neurons (and the rest with up to 1,570 of the other types of cells) as required by conventional neural network learning algorithms. It is also sufficient enough to instantiate several (up to 16) clusters or modules, each of which with no more than 100 cells and using modified STDP (Spike Time-Dependent Plasticity) methods concomitantly with modified Hebbian laws for the novel action learning algorithms [177]. These two cases allow for hardware solutions of machine learning with applications in embedded systems for prediction and for action learning, respectively. The proposed future work is to apply the latter for the autonomous learning of walking robots, from motor babbling to fluidic motions.

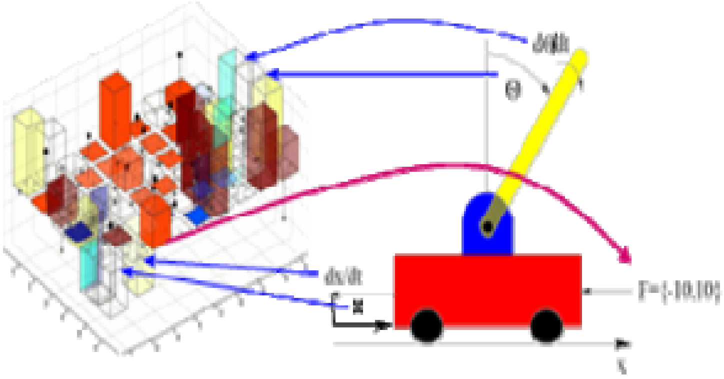


Figure 7.6: Self-action learning of the inverted pendulum.

7.5 Conclusion and Future Works

The developed universal self-adaptable neuro-controller chip can be applied in different applications, including aerospace, mobile phone, robotics, automobile vehicles, intelligent washing machines, etc. This neuro-controller chip would make the applied systems have faster response with increased accuracy, power efficiency, biological plausibility, reliability and robustness, and autonomy. In the next implementation of this chip design, the BHAM system will calculate the most optimal number of modules throughout the whole quasi-fractal cellular neural structure automatically against an arbitrary application such as the inverted pendulum problem depicted in 7.5. The cellular neural structure at the left hand side of 7.5 shows a single small, flat cellular neural fabric initialized with a random configuration. The dynamic adaptive rules will tend to make the structure form an effective cellular neural tissue, which in the future work will gradually cluster itself by adjusting the size of the modules so that the problem at hand is solved, and hence the performance will be improved and maintained at a most optimal level such as smooth pushes of the cart and fluidic motion of the stick. Results achieved for that problem will have direct impacts on the bigger problems of single-balled rolling robots and of walking biped robots.

7.6 Summary

There is a great demand on designing a highly self-adaptive digital hardware which can be implemented on in-house dynamically partially reconfigurable system-on-chip (DPR-SoC) as the kernel control system of robots or any different form of applied machines. Among the recent advents of numerous machine learning techniques, in this chapter a cellular automata (CA) model is selected in the proposed kernel control system. And a new algorithm to calculate the on-line outcome of the rules in hardware CA model is proposed. Simulation experiments demonstrate the efficiency of the developed algorithm.

Chapter 8

Spiking Neural Networks on Self-updating System-on-chip for Autonomous Control

This chapter introduces "spiking neurons" to the previous chapter works. This aids in the interfacing with commercially available pulse width modulation (PWM) driven servomotors as well as with biological nerve signals for biomedical applications such as chip implants.

8.1 Introduction

During the past two decades, artificial intelligence (AI) has been highly fragmented into numerous subfields in machine learning [166, 174]. AI technologies have encountered the problem in lack of communication between its subfields. Therefore, bottom-up design approach is applied in the learning system of a self-adaptive control system-on-a-chip (SoC) control system. A scalable cellular automata (CA) based neural processor is used in control systems to self-tune the learning capabilities [177]. This work is oriented towards the purpose of a highly self-adaptive hardware design implemented in programmable system-on-chip (PSoC) as the control system of those machines [176]. The applied machines/robots with this self-upgrading initial kernel design can learn its own status and surroundings by itself through collected information.

The remainder of this chapter is organized as follows. The previous work involving rule learning algorithm and relevant background are summarized in Section 8.2. The new rule learning strategy and method in CA is introduced in Section 8.3. Simulation results and discussions with the proposed method are shown in Section 8.4. Conclusion and future work about the self-updating autonomous chip design are given in Section 8.5.

8.2 Hardware Digital Neural Tissue Model

8.2.1 Background

Software usually runs on hardware, which has limitations due to the nature of typical architecture of the computer hardware systems. Also, CPU-based software systems require additional hardware to the main host hardware so that software failures are minimized. And the gaps between lower hardware layers through higher software layers make any embedded software system have a huge dimensional distance between its virtual (software) processes and the real sensors [173].

During the recent several decades, CA has been tremendously studied [174]. Various configurations with diverse sets of local rules have been associated to emerging collective behaviours and global dynamic phenomena [139]. Neural network (NN) has the property to find an optimal or near-optimal solution when facing a problem [79]. It can discover useful CA rules and recognize relevant patterns through inductive inference, which is called rule discovery.

The proposed system can not only be applied in navigation tasks but in self-extending useful applications. Event-driven self-programming and bottom-up design of the whole machine control system through real-time, intuitive interactions at the global level is one of the most important ideals of self-adaptive and self-organizing systems.

8.2.2 Cellular Autonoma on FPGA

After the initial manual top-down design of only one cellular unit, modular and hierarchical instantiation of cell copies recursively, the system can redesign, improve and adapt itself with no need of further manual intervention. The multi-sensor data fusion can be solved inherently by the system itself via the presented methodology. A bidirectional hetero-associative memory implemented as a NN on FPGA has the immense advantage of simultaneously processing information within itself through its dense interconnects [168].

Several updating schemes exist in the realm of 2D asynchronous CA design [174]. The proposed CA landscape, shown in 8.1, consists of totipotent cells. Each cell interacts with its four local neighbouring cells at its cardinal points, which allows correct decisions on the new state to be made. Docked cells at the CA level are liable to exchange spikes of signal through active neural channels, which are highlighted in black and bold line in 8.1.

A cell can morph into one of five states: neuron (activation transfer function), synapse (weight), axon (output distribution, attenuation rate and propagation delay), dendrite (input sum) and blank (idle). Each cell has a record of its four docking states and four local neighbours' states. Therefore, the neurally possible valid pattern states is of the order of 10^7, each of which requires a corresponding rule. Hence, automating selection of the relevant and useful sets of local rules is necessary and certain criteria have to be designed. Besides, coding all rules manually would limit the behaviours of the system.

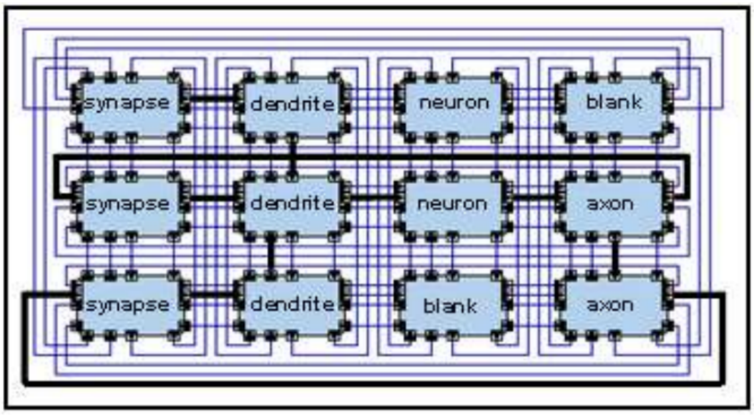


Figure 8.1: Self-configured CA-based neural tissues on FPGA.

8.3 Cellular Autonomous Rule

8.3.1 Hardware Digital Neural Tissue

Balsa, one of the hardware description languages, is used here as data-driven pattern formation for the proposed system [162]. First of all, the cell must possess one variable that stores its current state. This variable can be enum type, i.e., 0,1,2,3,4 to represent blank (B), dendrite (D), axon (A), synapse (S), neuron (N) states, respectively.

The proposed model is based on the four main functional elements, synapse, dendrite, neuron and axon. Here, the fifth junction element (T) is added in the model. An elementary 2D pattern is composed of five cells: a central cell and its four local cardinal neighbours. One of the four versions where a cell can morph into 2D elements is shown in 8.2.

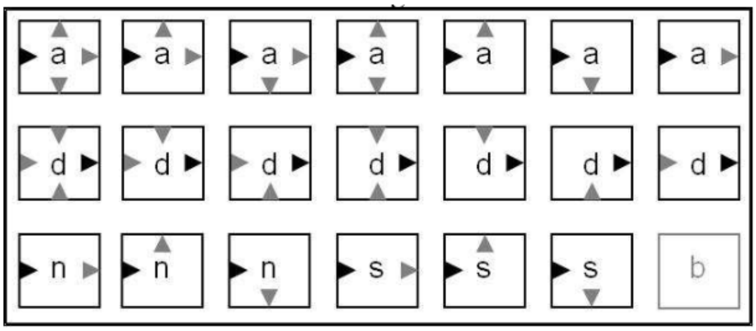


Figure 8.2: The total possible cell states.

The last blank element does not possess any docking connection. Therefore, the total number of distinct fundamental elements in the model is 81. The total number of possible 5-cell patterns is (almost 3.5 billion).

The decision of cellular autonomous rule is the key factor for the system operation. The number of the rules for the state update of a central cell is depended on several factors, such as the number of available cell states, the size of the neighbourhood and the level of adaptivity of the algorithm. It will increase with the rising number of cell states and the size of the neighbourhood but decrease with the level of the adaptivity of the algorithm [161].

The first axon in 8.2 (as well as its three rotated symmetric copies) has one input dock and three output docks. The input dock of that axon, considered here as a central cell of a 5-cell pattern, is neurally valid with $a_i = 3$ types of neighbour as viewed from input of the central cell a. Each of the three output docks, denoted as 'o' of a is neurally valid with its neighbour having either of axon, synapse or junction as type state and an input dock as dock state. Each output dock is also neurally valid with $a_o = 3$ types of neighbour. Since some cases of neighbourhoods for the three outputs of the central cell are interchangeable, some of them should be removed. Thus the number of unique neighbourhoods for the three $k_o = 3$ outputs of a is $a_{ooo} = \Gamma_{a_o}^{k_o} = \Gamma_3^3 = 10$ Therefore, the number of unique neurally compatible neighbourhoods with an axon element as central cell having one input and three outputs is

$$a_{iooo} = a_i \times \Gamma_{a_o}^{k_o} = 3 \times \Gamma_3^3 = 30 \tag{8.1}$$

where Γ is a multiset coefficient, denoting the binominal coefficient and it is used to calculate the unique rule numbers. The value ofΓ_n^k , chosen k elements (with repetition allowed unlike C_n^k from n elements is calculated as:

$$\begin{aligned} \Gamma_n^k = \left(\!\binom{n}{k}\!\right) &= \tfrac{n(n+1)(n+2)\cdots(n+k-1)}{k!} = \tfrac{(n+k-1)!}{k!(n-1)!} \\ &= C_{n+k-1}^k = \binom{n+k-1}{k} = \binom{n+k-1}{n-1} \end{aligned} \tag{8.2}$$

where C_n^k is the well-known multinomial coefficients for combination calculation. Since the unique neighbourhood conformations are counted only once regardless of the cardinal place of neighbours, the twelve axon elements above will be counted only once. The second axon from the left in the first row in 8.2 as a central cell has the unique neurally compatible neighbourhoods:

$$a_{ioo} = a_i \times \Gamma_{a_o}^{k_o} \times b = 3 \times \Gamma_3^2 \times 1 = 18 \tag{8.3}$$

where b denotes the unconnected dock state. The same theory is followed to calculate neural neighbourhood for dendrite as a central cell, i.e.,

$$d_{iiio} = \Gamma_{d_i}^{k_i} \times d_o = \Gamma_3^3 \times 3 = 10 \times 3 = 30 \tag{8.4}$$

For neuron and synapse elements, the unique neurally compatible neighbourhoods are:

$$n_{io} = s_{io} = 2 \times 2 \times \Gamma_1^2 = 2 \times 2 \times 1 = 4 \tag{8.5}$$

while the neighbourhood for the blank element is 1.

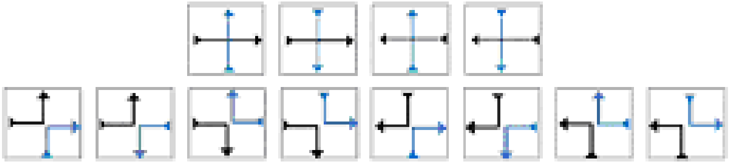


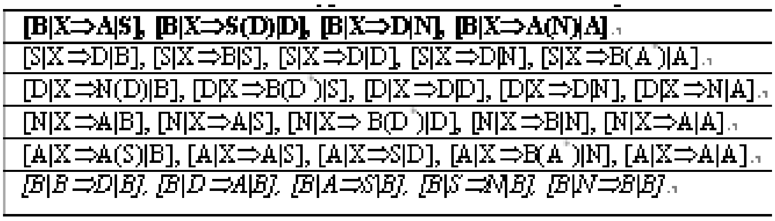
Figure 8.3: The junction element

Each junction element (T) in 8.3 has two inputs and two outputs. There are twelve junctions in total, and their neighbours will be counted only once due to their interchangeability between each other. The number of unique neurally compatible neighbourhoods with a junction as a central cell is 120. The detailed calculation is omitted here due to space constraints. Therefore, the total number of possible unique neurally compatible neighbourhoods is 243, which is always neurally valid with a type of element as a central cell and a docking configuration [161].

8.3.2 Methodology of Rule Update

The NN should discover and discriminate global laws of physics from outside and associate them to useful rules at the lower cell level [32]. As a result, this never-ending learning, growing, pruning and self-reorganizing CA-NN system can: (1) self-extend so as to become adequate to its static surroundings (self-upgrading property of neural tissue on PSoC) and then (2) continuously self-adapt to dynamically changing environments. As in asynchronous CA, the decision is made based upon the latest known states of neighbour cells, the current state of the cell, and the active transition rules. The set of the basic rules are listed in 8.4, in which all transition rules are also given. The list of rules in 8.4 is hard-coded in the first private part of the looped process within the cell, which is dedicated to CA control and distributed overall cells. The second private part both handles orientation of and processes neural spike signals.

In 2D CA, for neural tissue, the blank cell is considered. Hence, the set of basic rules on type state of element are meant to attract a central cell to be compatible with its two local neighbours. In 8.4, * represents inactive state and [ll...lr] depicts the states of the left (l) and right



[B|X⇒A|S], [B|X⇒S(D)|D], [B|X⇒D|N], [B|X⇒A(N)|A]
[S|X⇒D|B], [S|X⇒B|S], [S|X⇒D|D], [S|X⇒D|N], [S|X⇒B(A)|A]
[D|X⇒N(D)|B], [D|X⇒B(D)|S], [D|X⇒D|D], [D|X⇒D|N], [D|X⇒N|A]
[N|X⇒A|B], [N|X⇒A|S], [N|X⇒ B(D)|D], [N|X⇒B|N], [N|X⇒A|A]
[A|X⇒A(S)|B], [A|X⇒A|S], [A|X⇒S|D], [A|X⇒B(A)|N], [A|X⇒A|A]
[B|B⇒D|B], [B|D⇒A|B], [B|A⇒S|B], [B|S⇒N|B], [B|N⇒B|B]

Figure 8.4: Basic Rules on Type States of cell with two neighbours.

(r) neighbours. 'X' type state denotes 'no matter' and strictly from left to right connection for dock connection. Based on the original states, a new valid type state for a central cell can be determined. The only exception is for rules where both neighbours are the blank element 'B'. These rules are listed in the last row. In a 3-cell pattern, each cell has five possible states, and each state the direction of dock is always from left to right. If the last row in 8.4 is not developed into five different cases, only one case belonging to the family of the cases in the first row will be kept. Then it holds the five cases as the four other lines, which would bring the total number of cases to 25. The algorithm for a new state decision is a function to assign a new state to the central cell based on the 25 fixed rules corresponding to the 25 cases. Hence, the number of rules r_1 to be specified is calculated as

$$r_1 = s_1^k = 5^2 = 25 \tag{8.6}$$

where $s_{i(=1,2,3)}$ is the state and k is the size of neighbours. The number is exactly equal to the number of cases to be considered due to the use of an 'assign only' algorithm. If a more elaborated algorithm is used, fewer rules are required. Consider another modified scenario closer to the reality, in which dock connections are not strictly directed from left to right. Here, both neighbours can be either unconnected, as an input or an output to/from the central cell. In order to obtain the number of rules necessary to cover all cases of neighbourhood, new assumptions should be made for the determination of the new state for the central state. If the algorithm is an assignment function, the number of rules

$$r_2 = s_2^k = 13^2 = 169 \tag{8.7}$$

where the modified number of states s_2 for a neighbour is calculated based around the central cell diagram as:

$$s_2 = (s_1 - 1) \times d_2 + b = (5 - 1) \times 3 + 1 = 13 \tag{8.8}$$

while $(s_1 - 1)$ denotes the number of type states that can have docks (type B is excluded); $d_2$is the number of types of docks (input, output or unconnected) and b is the number of type states that cannot have docks.

However, if the algorithm is allowed to consider an unconnected neighbour of any type state (which have docks), the required number of rules becomes $r_3 = s_3^k = 9^2 = 81$.

If the algorithm can both deal with equivalents and unconnected neighbours as a single type state 'B' regardless of the actual type state of neighbours, the number of rules is $r_5 = \Gamma_{s_3}^k = \Gamma_9^2 = 45$. It can dramatically decrease the number of rules to specify down from greatest problem size of 169. Constrained by feasible coverage of rules, such algorithm must be sufficiently adaptive in order to compensate for non-predefined decisions on many rule outcomes.

Each of both previous algorithms require 169 and 81 rules to assign to an element among the 29 available elements with its type and 2-dock states to the central cell once it knows the neighbour surroundings. In both scenarios, assume that cells can also connect up and down and therefore can be partially or fully unconnected to/from left and right, which leads to

$$\underbrace{4}_{(A,D,N,S)} \times \underbrace{7}_{\substack{(2fullconnected, \\ 4partialconnected, 1unconnected)}} + \underbrace{1}_{(B)} = 29 \tag{8.9}$$

choices for the updated state of the central cell. However, each of both 'smarter' algorithms that require 81 and 45 rules first assign only a type state to central cell (no docks states yet) according to the pre-defined entry of the rule outcome given a neighbourhood. The algorithm can accommodate the docks types of the central cell to reflect those of the neighbours of the central cell which are compatible with. Note that once the algorithm has picked the rule outcome to form the look up table of rule entries, both operations on the central cell actually happen concurrently for the type state update and docks states accommodation.

8.4 Results and Analysis

8.5 illustrates the real-time animation of a 9×6 hardware asynchronous CA by showing twelve carefully chosen frames over ten minutes of run. The system starts from a random configuration and a random single stimuli shot (initial random condition at time 00min00sec), and without application of input stimuli during progression.

After four minutes of run, most cells have died out (completely transparent colours) while some others have reborn such as cell 5,1 from formerly neuron type (red-orange colour) to newly axon type (blue colour).

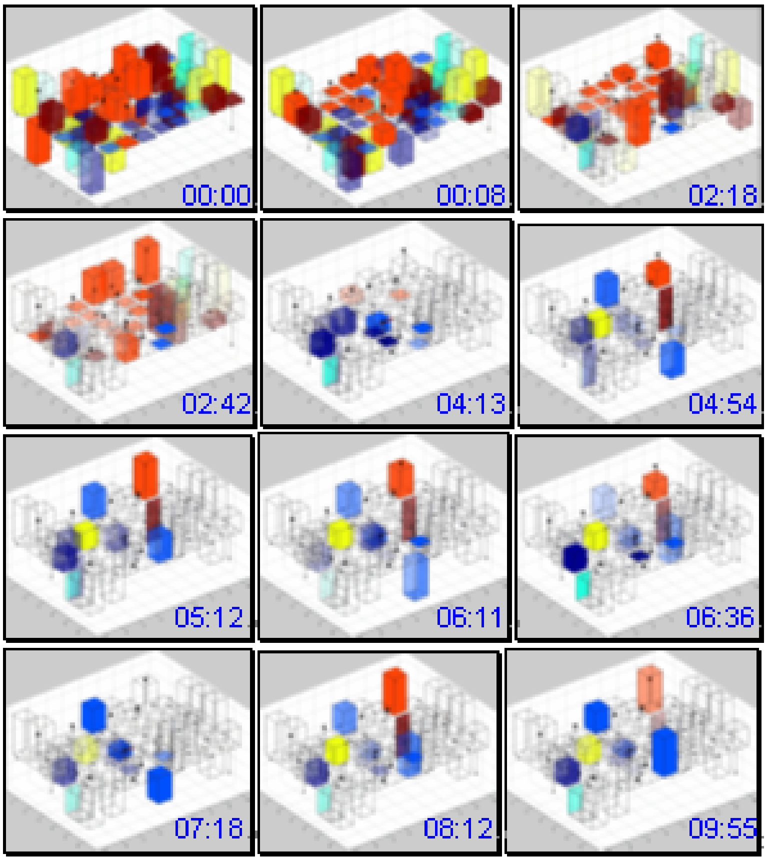


Figure 8.5: Progression of a hardware asynchronous CA as a NN fabric

The observed global behaviour is typical of asynchronous CA, for which initial conditions are unimportant, whereas rules drive the system. Here we can witness that it is the case even without persistent input stimuli, which shows the intrinsic characteristics of such a ruled system. The height of bars represents the value of a spike signal within cells. Over time, same cells have these values changed, which shows the proper processing of spike signals within the distributed algorithms of each cell as well as the effective global self-synchrony due to the coupling of asynchronous cells.

The persistence of living cells on active status until the end of the animation is directly related to the average ratio of active rules. Remaining effective living cells 1,4, 4,4 and 4,5 are dendrites in dark blue; 3,3 is junction in yellow; 5,1, 5,3, 5,6 and 6,5 are axons in blue; 8,1 is neuron in red-orange; and 8,2 is synapse in brown. Black blotches represent the value of neuron's threshold while stars represent the value of a synapse's weight. Darts laid on the horizontal plane show docking connectivity between neighbour cells.

In this small system of only 9×6 cells, the landscape is overly constrained (note that last row and last column are connected back to first row and first column respectively, to virtually form a torus), thus largely reducing the freedom of cells. This has the effect of preventing formation of many useful cell patterns, even though the rules are clearly obeyed and drive the formation of neurally valid cell patterns.

8.5 Conclusions and Future Works

In this chapter, a new method to update the rule for spiking neural network is studied. The simulation results and figures show the effectiveness of the proposed algorithm. The embedded machine with the learning algorithm can perform satisfactory responses/actions when encountering different novel situations. Future study will continue for the learning algorithm to enhance the performance of the neural network system.

8.6 Summary

The artificial intelligence (AI) technique has suffered in solving its computationally hard problems in recent years. In this chapter, a self-upgrading autonomous system is designed to tackle end-to-end AI-hard problems and to achieve self-adapting communication via modular and hierarchical extension from linguistic and semiotic constructs. A system-on-a-chip (SoC) self-adaptive control system can learn arbitrary shape of the robot body or machine parts. Simulation results have proved the effectiveness of learning abilities of the proposed autonomous system.

Chapter 9

Hardware Asynchronous Cellular Automata of Spiking Neural Networks on SoC for Autonomous Machines

This chapter improves further the work reported in the previous chapter and makes the implementation clearer.

9.1 Introduction

During the last two decades, the research of artificial intelligence (AI) [178] has been fragmented into numerous subfields with top-down design pattern that now revolves around machine learning (ML) [179]. This top-down approach has worked quite well in computer science for designing huge hardware and software systems, but AI research and technologies have become too much heteroclite with evident lack of communication between their subfields, thereby making it cumbersome or impossible to combine back together and solve AI.

Hence, the traditional AI approaches have now reached a stagnating stage. Practically, it is because the host hardware computer machines are unable to deal with semiotic and linguistic constructs. However, paradigms such as SystemC and Handel-C from electronic design automation (EDA) community are now available and allow for conversion of sequential algorithms written in C++ into parallel algorithms generated in hardware description language (HDL), usually yielding significant gain in computation acceleration as well as strengthening fault tolerance of the system.

However, in this chapter, a bottom-up approach is adopted to tackle end-to-end AI hard problems [180]. Inspired from physical and biological concepts and theories, the problem can be solved by highly self-adaptive control system-on-a-chip (SoC) to self-learn any interfaced (mobile) robot body of arbitrary shape.

This chapter delineates the potential complexity of the proposed control system and shows a novel system architecture. The control system will be able to self-manage its internal and external resources. Such an end-to-end problem represents a precious aim [181], hence, the more biologically plausible the model considered is, the more likely this aim is to be reached.

The reminder of this chapter is organized as follows. In Section 9.2, background of the learning methods in neural networks is introduced and surveyed. Methods of self-configuration are discussed in Section 9.3. The theory of cellular automata technique and related proposed projects based on CA rules are explained in Section 9.4. Results and future research direction are given in Section 9.5.

9.2 Background

Neural networks have the property of finding a solution configuration when facing problems, even with different solutions for the same problem. Each solution is guaranteed to reach an optimal or near-optimal point. The same property holds true for the proposed model, which includes cellular automata (CA), neural networks (NN) and fuzzy logic that can learn with hybrid competitive-cooperative swarm particles optimization methods. As explained in [173], CPU-based software systems are inherently lacking of robustness, which require additional hardware artifacts to minimize software failures. Furthermore, the gaps between lower and higher hardware layers make any embedded software system have a huge ‘dimensional distance’ between its virtual (software) processes and the actual sensors/actuators to deal with real-world tasks. The machine learning techniques based on interacting layers are delineated in 9.1.

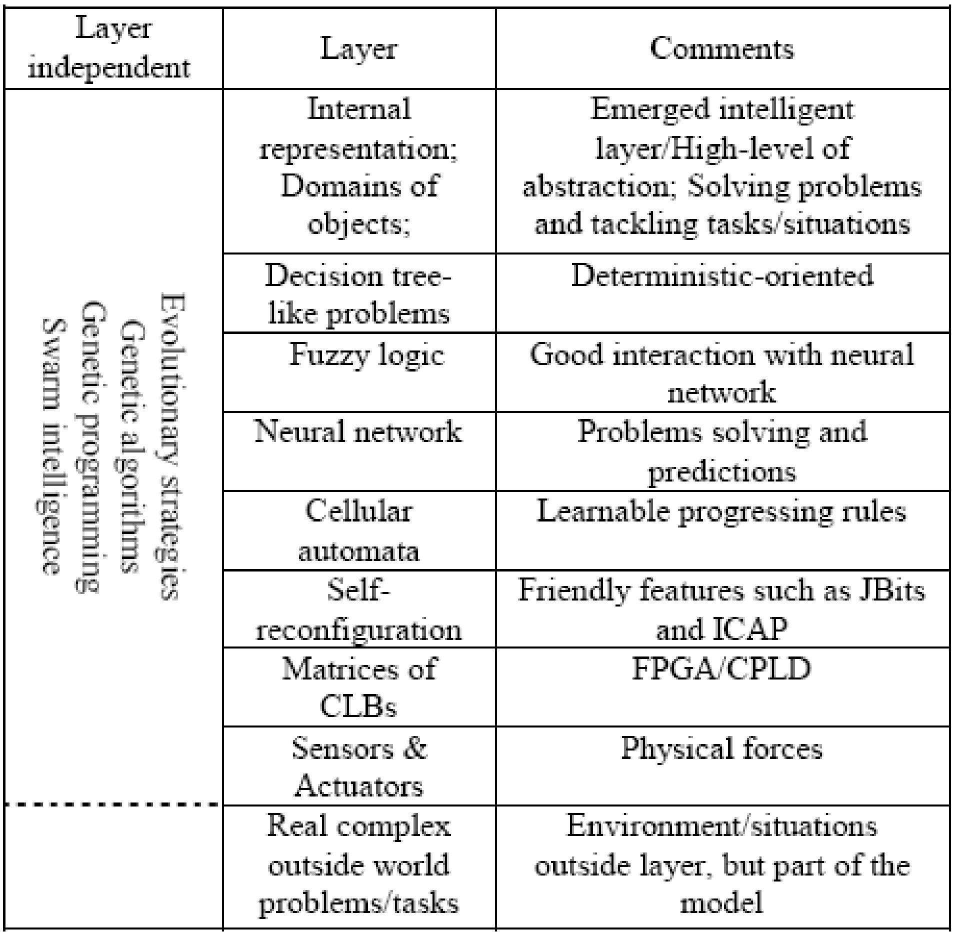
9.2.1 Neural Models

In artificial spiking neurons, the action of potential model typically relies on the simple version of Hodgkin-Huxley model, which takes into account basic biological ion channels and a model of passive membrane. However, amongst more advanced models, the Fitzhugy-Nagumo model considers interactions of neuron cells with surrounding biological tissues and cells as well as many ion channels.

Together with neuro-modulation phenomena, the Fitzhugh-Nagumo model is used here for hardware implementation to support connectionist idea and to build artificial nervous systems. From the cybernetic point of view, it is useful to combine reductionist with holistic approaches into holonic approach to handle mechanisms having self-recursive properties across different scale levels of hybrid neural network modules.

9.2.2 Learning Mechanism

Hebbian learning does not appear satisfying to use for biologically plausible artificial neural networks (NN) [178]. In biological NNs, synapses do not monitor the activity of their postsynaptic neurons to get information on how to update their weight. It is still not clear how much Biology knows about the processes involved in making synaptic connection strengths vary. Generally, it is assumed that a synapse involved in causing its postsynaptic neuron to fire will have its weight increased, whereas if it is not involved, and it will have its weight decreased. Since there are no feedback signals in biological NNs, a synapse cannot get the causal information from its postsynaptic neuron. However, in recurrent NNs, the information coming back to a



Layer independent	Layer	Comments
Evolutionary strategies Genetic algorithms Genetic programming Swarm intelligence	Internal representation; Domains of objects;	Emerged intelligent layer/High-level of abstraction; Solving problems and tackling tasks/situations
	Decision tree-like problems	Deterministic-oriented
	Fuzzy logic	Good interaction with neural network
	Neural network	Problems solving and predictions
	Cellular automata	Learnable progressing rules
	Self-reconfiguration	Friendly features such as JBits and ICAP
	Matrices of CLBs	FPGA/CPLD
	Sensors & Actuators	Physical forces
	Real complex outside world problems/tasks	Environment/situations outside layer, but part of the model

Figure 9.1: Machine learning techniques with interacting layers

synapse along a looped path is biologically realistic. In the meantime, this open question is answered, and an alternative mechanism is proposed to modify the connection between synapses to study the emerging phenomena. Besides, at a microscopic level, cells can be modified from global feedback. Therefore, the cells can interchange information mutually within the neural systems [182]. At a microscopic level of a nervous system, the weight of the cells such as synapses can be modified from global feedback, whereas the whole brain has its information modified from external stimuli as well as from within the brain due to internal processes. This observation may lead to a model at intermediary levels between synaptic connections and the whole brain, from which the information can be modified mutually, along with swarm effects.

9.3 Method of Self-Reconfiguration

9.3.1 On-Chip Hardware

With the advent of new commercially off the shelf (COTS) chips, i.e., FPGA (field-programmable gate array) devices, dynamic and partial self-reconfiguration of microelectronic hardware circuits has become readily available and has been investigated for decades [168]. In particular, Evolvable HardWare (EHW) sounds promising, but [176] have clearly shown the limitations of using commercial FPGA devices for reconfigurability purposes. One way to solve the main limitation is to use bit-streams within function core system (co-processor) for synthesis and place & route via the internal configuration access port (ICAP) along with JBits [168]. Together with the bottom-up design combined by parallel micro-mechanisms in hardware FPGA device, several features arise: 1) After initial manual top-down design of only one cellular unit, and then multiple instantiations of it on a two- dimensional (2D) cellular automata landscape, the system may redesign, improve and adapt itself without further manual intervention; 2) The system has the fault tolerance property; 3) Sensor fusion can be achieved by the system itself. Although bit-streams can be handled on-chip by using the ICAP with JBits, the scheme is not suitable for the purpose of **continuous growth** and self-development of our *holarchical* neural tissue as neuro-controller. Besides, the bit-streams must be produced and uploaded into the reconfigurable FPGA device, which is used to deal with discontinuous situation. Hence, it would increase the vulnerability of the co-processor and would decrease its instantaneous integrity. Alternatively, CellMatrix devices have recently been developed [163]. In a CellMatrix device, self-reconfiguration takes place from within functional core circuits of logical cells. There is no access to internal cells from outside and the cells are embedded in low-level electronic logical functions. In contrast, the proposed CA-NN model for digital discrete neural tissue formation from within and self-gained reconfiguration has no edges and allows for boundless landscapes. This is necessary for modelling autopoietic systems [180]. Access to all internal cells from outside is permitted, like for FPGA, for the purpose of monitoring and controlling the various phenomena of interest. In

CellMatrix and FPGA, low-level electronic logical functions require handling complicated and huge amount of parameters to be configured and routed. On the other hand, CellMatrix device can implement the proposed neurocontroller: low-level electronic logical functions can implement higher-level neural processes. Yet, the FPGA device with its mature EHW methodologies remains a hardware substrate of choice.

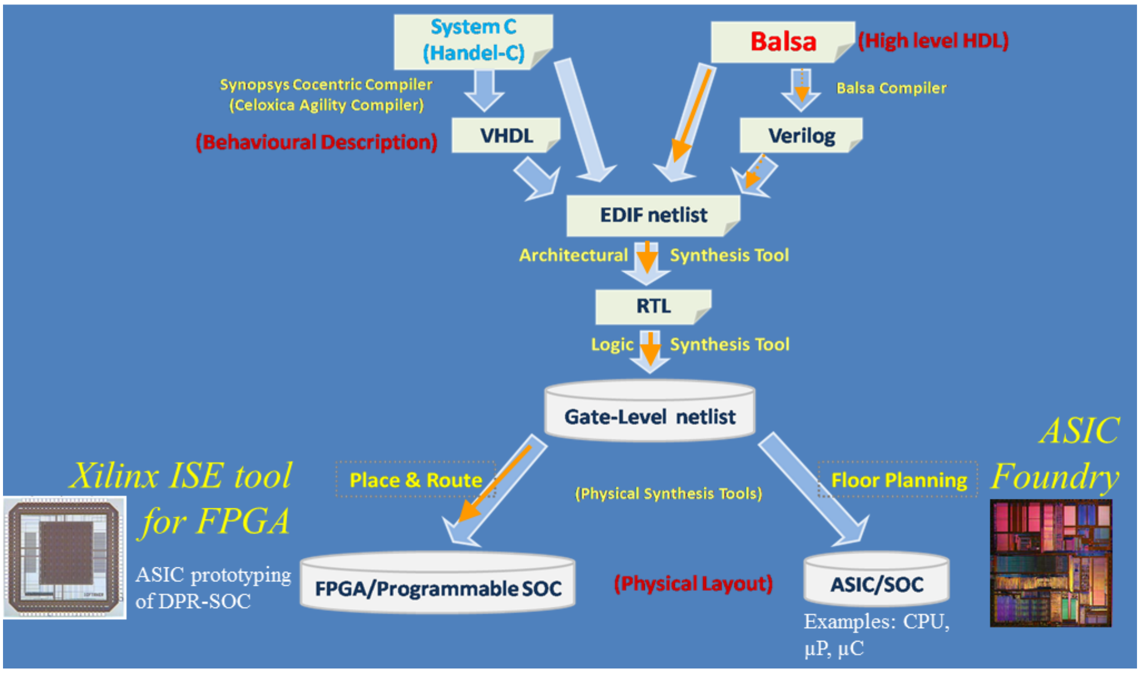


Figure 9.2: Diagram of the design flow utilised in this work, from hardware description in Balsa language of the sought wetware design down to implementation on FPGA silicon chip of the hence operational wetware.

With the reference to the POEtic model [176], the proposed work is primarily focused on follow-up of continuous growth and development of learning and processing. Yet, obscure blind evolution of digital discrete neural tissues as for finding “seed design” solutions of neuro-controller from near-scratch points might be proven useful or necessary through EHW. Meanwhile, in our current work, initial rules are elaborated manually and can be automatically tuned along with growth and learning. Therefore, FPGA devices are the hardware target for this ongoing research, while prototyping an in-house ASIC solution (also using FPGA) to enhance performance and foundation of custom mechanisms keeps running in the background as a long-term alternative.

9.3.2 Embedded Software

Software usually runs on hardware and it has limitations due to the typical architecture of the conventional computer systems. On-board buses (that interconnect ICs) are the reason for speed bottleneck of the whole system. The transferring frequency of these buses is only about 1GHz compared to the one of on-chip cores (nGHz). On-chip standalone microcontrollers also make their hosted embedded software suffer due to fixed separation of pre-dedicated modules such as memory and CPU since communication between them also has to be serialized through buses. Hence, among the recent advent of numerous diverse machine learning techniques, the cellular automata (CA) model appears to be a prime choice for the proposed system kernel. Two dimensional (2D) CA can map nicely onto commercially off-the-shelf (COTS) hardware devices such as FPGA. There are several updating schemes in 2D asynchronous CA, where self-synchronizing scheme is a good choice for modelling neural tissue [183]. To this end, the proposed CA landscape (seen in 9.3, consists of totipotent cells, and each cell interacts with its four local neighbours at its cardinal points. In-house policy-based CA rules allow the cell to make correct decisions on the state that it should adopt based on hard-coded local transition rules that ensure the compatibilities between adjacent docked cells. A cell can morph into one of six states: neuron (activation transfer function), synapse (weight), axon (output distribution), dendrite (input sum), junction (bridge), and blank (idle). Docked cells at the CA level are liable to exchange neural spikes of signals through active neural channels. The asynchronous circuitry constructs in Balsa, our chosen hardware description language (HDL), can implement handshake protocols to support spiking communication. Showing dynamic dexterity off mobile robot navigation acrobatics is incontestably one of the few best external indicators on the level of internal intelligence that an autonomous manoeuvring machine may possess. Therefore, mobile robotics has gained a great interest.

In particular, this work is oriented towards the ultimate aim of a highly self-adaptive hardware

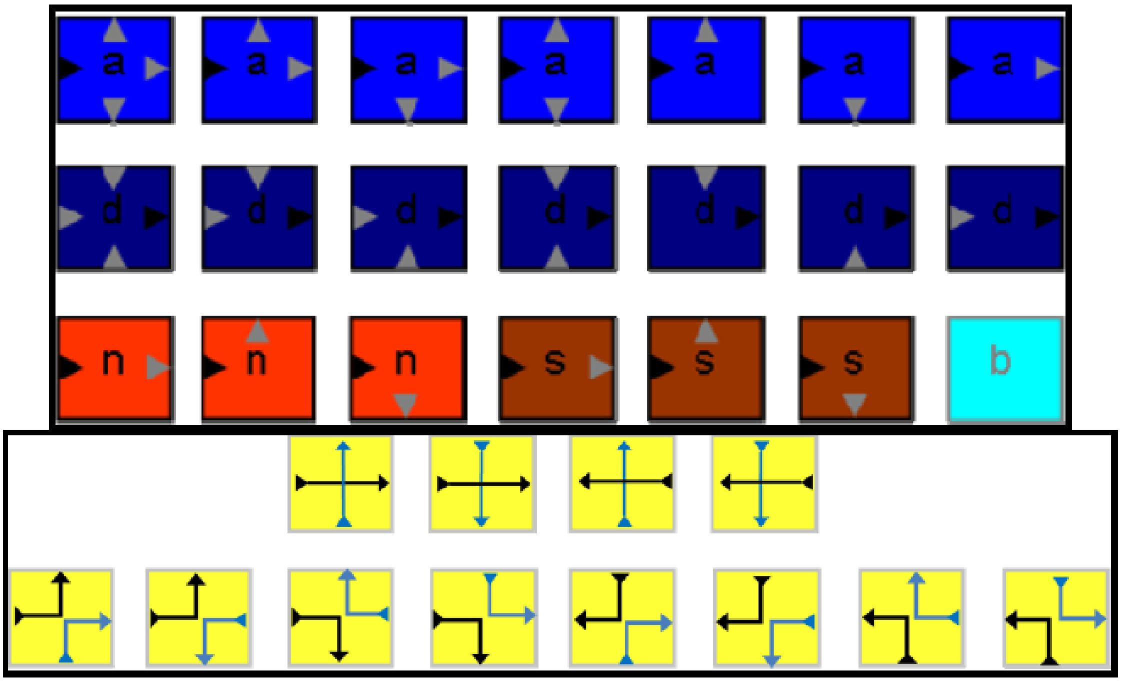


Figure 9.3: The 93 possible available states of a cell. There are three other rotations of the 20 role cell states shown, totalling 80 plus 1 blank cell states. The 12 available junction cell states in yellow colour are all shown.

digital design implemented in programmable system-on-chip (PSoC) as the control system of robots or machines of any forms. This ever self-upgrading initial kernel design is supposed to make the whole control system, which is connected to a random robotic body, to possess the functions: (1) self-learn using the sensors and actuators attached to the body; (2) gradually learn about the surroundings via action learning; (3) maintain and improve its internal and proximate external resources; (4) learn tasks online in an open-ended manner under supervision with real-time constraints. The work presented in this chapter has adopted Balsa, which is both a HDL and a synthesis system that can produce Verilog code. One bio-inspired parallel mechanism described in Balsa language is chosen for its asynchronous design. Balsa handshake protocols involve self-coupled cells, which translate the self-synchronized update scheme of asynchronous CA. Then, the cells can be coupled by using Kuramoto model (neural fabric/tissue [184]) so as to obtain a self-synchronized update scheme. The scheme can be fed to lower level electronic design automation tools in order to physically target reconfigurable devices such as Xilinx XCV2000E and Virtex5 FPGAs in this project. In Balsa, due to handshake circuit protocols, the data can be transferred mutually in two directions. For instance, a cell A designed with Balsa having its data input interfaced with another cell B can request cell B to send data at any time (cell A and B are any cells).

9.4 Cellular Automata

Following the 1^{st} and 2^{nd} cybernetics points of view, the learning scale has been framed to real world constraints, such as the robot running out of battery and requiring assistance. After the assistance, the robot is soft-reset and increases one life counter for this event. Accordingly, such occasions are programmed in the initial phase using minor human-coding intervention. Gradually, this never-ending learner will require bare or no human coding intervention anymore.

9.4.1 CA-based NNs

Within the CA-NN context, the cells or CA modules as depicted by 9.4 are primitive agents that perform transactions of standardized data structure records, which consist of two main parts:

(1) Control data: embed instructions to the cells; and,

(2) Spike signals: processed by agents according to their settled roles (as well as input/output events to/from external devices). This scheme has several implications:

1) Within a single CA (hyper-)plane (level-0 module of the holarchy), the control agents are distributed over the CA-NN and they are interfaced with external sensors and actuators;

2) Higher level modules constrain their low level modules recursively down to level-0 modules;

3) The agents execute consistent roles given correct fixed rules or free rules.

CA-based NN modules may self-develop by increasing self-knowledge so as to gain understanding on how to renew control and improve self-learning by means of amendable flexible and free rules, and so on. Furthermore, rules involving safety and security can ensure stability of the whole CA-NN based robotic neuro-controller. The proposed method is used to make the machine to perform tasks in dangerous places, which is useful in the real-world. The machine is equipped with highly self-adaptive neuro-controllers allowing communication with humans and bridging gaps between missions and available human reaches.

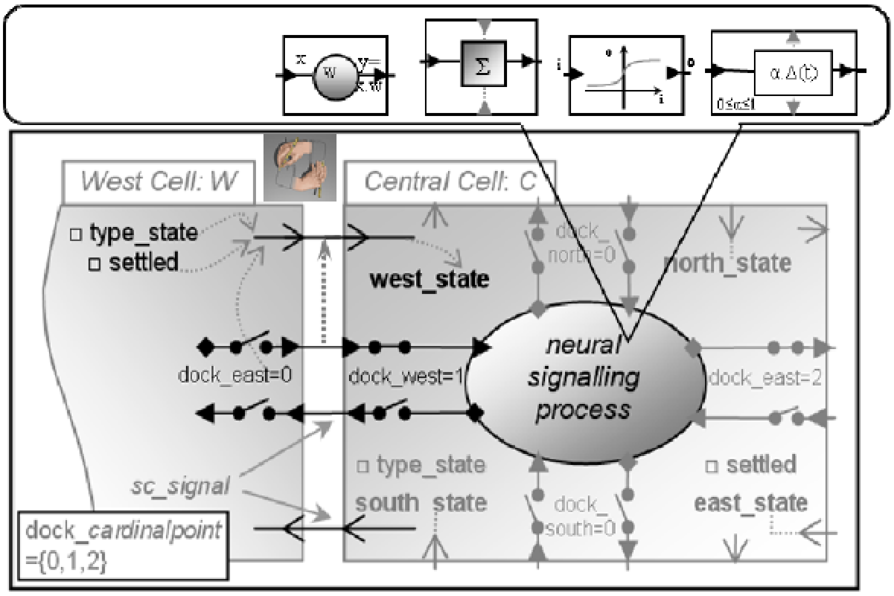


Figure 9.4: Design of a cell showing its four available role states at the top and its West neighbour on the left as well as connectivity: solid arrows represent spike signal pathways and the other arrows on the side represent control signals that indicate a cell's values such as its role/type state or dock state to its neighbours.

Harnessing emergence of self-organizing holonic units by holding a memory that associates behaviours of cells at a local level and a global level from the cell learning theory is considered. The auto-associative memory can be transferring among different levels based upon neural network structures [184]. However, to start with, a small separate supervised NN as hetero-associative memory should be used (in the beginning of the CA-NN growth) to record and retrieve required associations of CA rules with NN behaviours for each basic hybrid CA-NN module. Afterwards, the whole system will be reprogrammed recursively based on the new information in the whole levels of modules. The CA is used as a substrate for digital discrete neural tissue formation. Internal pattern formation is studied relatively to real-time sensed stimuli of external pattern sequences. It is desirable to empower the system with online learning and topology adaptation capabilities. Some relevant references can be found in [180]. About the former, it is non-trivial and poorly documented in the literature. However, the existing learning mechanisms, such as Hebbian learning, would suit the design in learning in spiking NNs. In [185], it reports studies on time-dependent post-synaptic conductivities through shifting registers, which is based on shifting neighbour state information from performing cell.

9.4.2 CA Rules

During the past several decades, CA have been tremendously studied [32] about the rules associated with global emerging behaviours. Take an example, in 9.5, the number of possible neurally valid pattern states (a central cell state, its four local neighbours' states and its four docking states) is of the order of 10^7 . Each state needs a transition rule so that it is necessary to automate/update the rules to react to the new situation. There are several basic rules:

(1) Basic Rules: sequence in explanatory 1D elementary CA $\{axon|input->\}synapse->dendrite->\{dendrite->\}neuron>axon->\{axon|output->\}$

(2) Growing rules: exploration of network topologies and connection styles;

(3) Learning rules: adapts online based on changing environmental conditions;

(4) Application n rules: formed with regard to mission or problem submitted;

(5) Maintenance rules and performance rules. The initial emerged NN sufficiently grows to perform primary basic tasks of recognition:

1) rule discovery: NN discovers useful CA rules and recognizes relevant patterns through inductive inference;

2) rule amendment or implementation: NN rewrites or add new CA rules.

Neural networks have the property of finding a solution when faced to a same problem at different times. Every solution is guaranteed to reach an optimal or near-optimal point. The same property holds true for the proposed model, which includes neural networks and cellular automata. CPU-based software systems require additional hardware artifacts to the

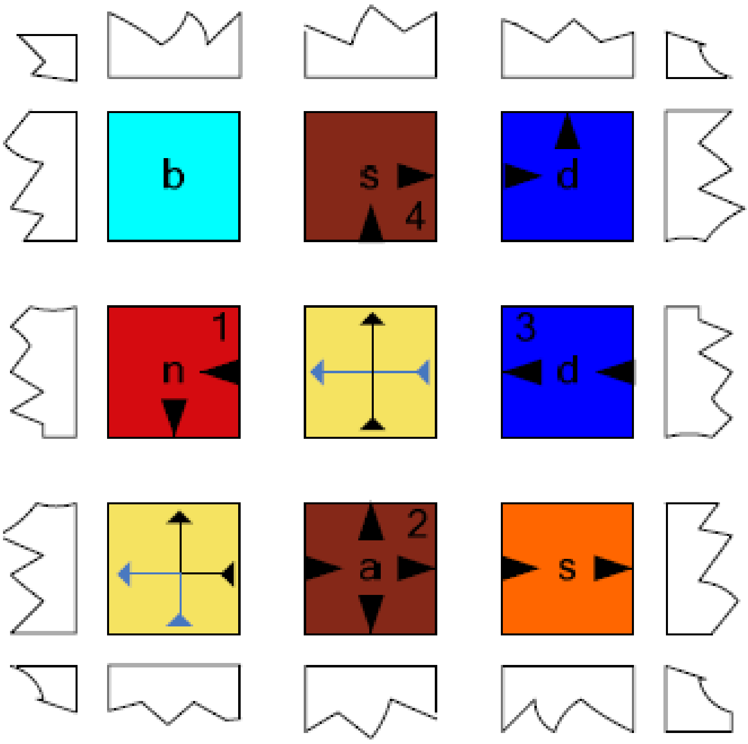


Figure 9.5: A neurally valid five-cell pattern example. The four corner cells are considered indirectly. Thus, in our novel approach to self-learning, NN is the programmer of the rules that dictate the behaviours of its own underlying CA. To start with, NN is associated with fixed rules based on meta-rule programming. With the capability of gaining new skills, the system will not limit to navigation tasks but more useful applications including stream reasoning. Event-based self-programming and bottom-up design of the whole robot control system is one of the most important ideals of self-adaptive and self-organizing systems.

main host hardware so that software failures are minimized. For instance, conventional extra error correcting codes (ECCs) hardware subsystems have the knowledge about the meaning or correlation of the data stored in a memory array or travelling through a bus. However, flexible hardware systems such as the proposed one have the inherent property of fault tolerance and therefore do not need extra ECCs. In addition, higher levels of hierarchy, i.e., neural network, have feedback understanding and authority on lower levels such as 2D asynchronous cellular automata, thereby ensuring persistent coherence. A good way towards a universal self-organizing platform that could natively communicate smoothly with humans is to combine self-design philosophy at fine-grain level of digital logic with self-organization and self-adaption at coarser-grain level of CA and NN components. In that paradigm, outputs from higher level processes such as CA or NN processing results could just be self-generated for lower level digital logic designs of new cells, or of new properties of existing cells. At present, the work is focused on validating the lowest level sensory-motor modules as self-adaptable to most basic tasks as well as self-learning. Future work can reuse the same scheme so that higher level modules can recursively self-learn.

9.5 Results and Discussions

9.5.1 Ongoing Work

The proposed CA-NN model is based upon the four main elements found in biological nervous systems: synapse, dendrite, neuron and axon. In addition, a fifth element of blank cell as idle process is necessary in 2D implementation as well as junction cells as sixth element to allow bridges. The main part of the work consists in mapping asynchronous spike time dependant plasticity (aSTDP) method and punishment/reward learning mechanisms onto our CA-NN as a design seed intended to FPGA substrates for the resulting cognitive system device to become conscious when aware via senses through connections to various robotic sensors: self-fusion of multi-sensing. Our work includes Hebbian learning and Kohonen SOMs for unsupervised learning [178, 184, 186, 187]. The cognitive chip controller or brain-on-a-chip (BoC) that we are developing features the required following: deep self-learning [188]; self-recursive neuro-cognitive processes with policies-based self-reorganization of internal processing units (analog to neurogenetic phenomena recently found in biological brains) [173] structured in a holographic, quasi-fractal network; self-learning of pattern recognition due to our in-house CA-NN-based kernel as a growing design seed as animated by 9.6, which is the plot-out of our system at a run-time start; autonomous formation of bidirectional heteroassociative memory (BHAM) with free knowledge representation up to the system itself to decide; meta-learning; and the final silicon chip exhibits neuro-modulation as well, which we are looking to harness. 9.7 shows the XOR function performed by our CA-NN system using spike protocol. This is the proof of concept to be used for autonomous PWM signals production and sensors signals learning with minimum or no interfacing to robots.

9.5.2 Perspectives

We are also mapping **immediate** punishment/reward learning mechanisms onto our design seed consisting of highly distributed, massively parallel, asynchronous processing units as a cellular-automata based neural network on silicon, for reactive behaviours: feeling of pain/pleasure; as well as **anticipated** versions of these mechanisms for deliberate and mindful decisions and actions: conscious awareness, "artificial living mind", crucial for true robot carers for instance.

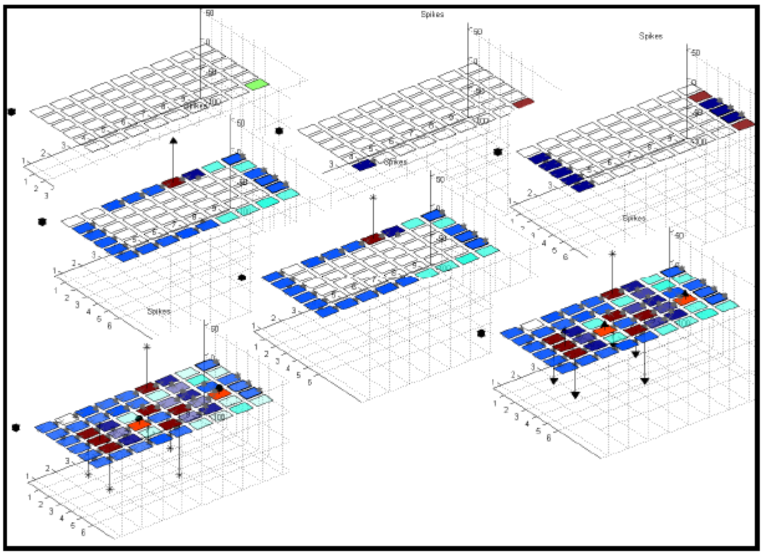


Figure 9.6: Birth of a cell module with initial rules directing growth towards a neural network assuming the logical XOR Boolean function with spiking signals as operating protocol.

With the emergence of artificial consciousness, artificial conscience is also naturally due to rise alongside. Hence, conscious robots, with our cognitive device as their BoC, will be able to perform effectively and efficiently with fluid motions and fluent communication within our

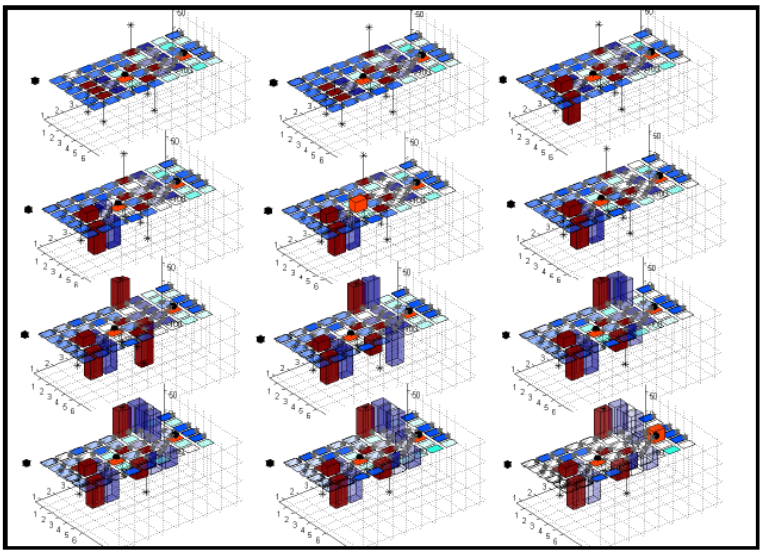


Figure 9.7: Spiking propagation activity of the grown XOR neural network, with the output neuron in red colour eventually firing as two opposed values are maintained at both of its inputs (not explicitly shown).

infrastructures due to their very consciousness, which will also primarily make them maintain it up (exactly like people naturally strive to stay away from fainting) and therefore will spontaneously minimize the risks of sudden, untimely disruption of their operations (unlike with current technologies), thus making them truly robust and reliable; and in the process of learning within our environments they will develop a conscience, which will make them ethical and dependable. Since they are machines, they sustain these features infallibly, which will make them the ultimate choice for roles such as tutoring, coaching, caring, security, and aftermaths recoveries.

9.6 Summary

The research field of artificial intelligence (AI) has long abode by the top-down problem solving strategy. Yet, we have adopted bottom-up design thinking to solve its hard problems. To tackle end-to-end AI-hard problems, a highly selfadaptive control system-on-chip (SoC) has been developed to selflearn its internal and external resources with the aid of sets of sensors

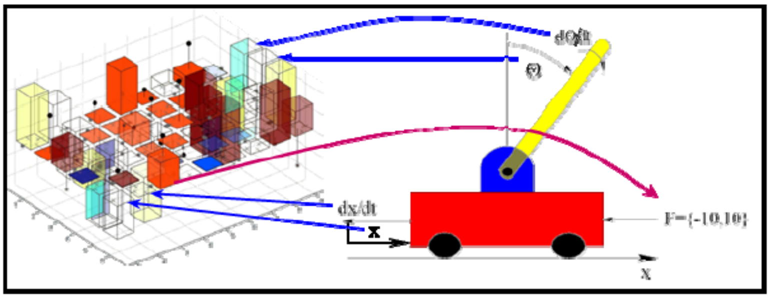


Figure 9.8: Four inputs, one output cellular automata – neural network controlling an inverted pendulum. The application of self-learning inverted pendulum kinematics and dynamics via initial motor babbling illustrated by 9.8 has direct implication on the near-future success of bipedal walk self-learning with efficiency and reliability of yet fluid motions never attained thus far. This is the research area of mobile robotics that this project enables us to give most of our attention to for our next research steps and resulting papers.

and actuators. Inspired by biological cell learning theory, different approaches of modelling techniques have been derived together with machine learning (ML) methods to the embedded control systems so as to perform different tasks. This chapter lays out our developments of the above.

Chapter 10

Efficient Parallel Asynchronous Hardware Algorithm of On-Chip Cellular Automata for Formation of Utilizable Digital Neural Tissue

This chapter aims to show the algorithmic rationale behind this PhD work as well as achievements of the corresponding project.

10.1 Introduction

With its strong inter-disciplinary characteristics, "machine learning is the study of computer algorithms that improve automatically through experience" [2]. Many different machine learning (ML) techniques have been developed recently. Genetic algorithms quickly evolve encoded problem solutions along natural selection-inspired processes and can reach a near-optimal solution in many applications.

Decision trees are used in knowledge engineering and data mining applications, expert systems can help tackle specific problems that require comprehensive knowledge. Fuzzy sets can calibrate vagueness, as required for natural human language processing.

Artificial neural networks (ANNs) are inspired from interconnections of brains cells. They are 'processing memories' where the knowledge is distributed over tiny units that operate in parallel. They can hold a representation of a hierarchical model of the world and can form contexts by arranging groups of data patterns in sequence, therefore experiencing the environment. Generally, an ANN and its learning algorithm are not obvious to make usefully work right from the start.

Algorithms inspired from genetic processes can be applied to solve any problem in which the solutions can be encoded. Because it is the case in actually almost all types of problem, GA (Genetic Algorithm) approaches can be taken when a "good enough" solution is required quickly, but without the need to understand the rationale of this particular solution(s). Indeed, to guide the evolution of a GA, we preliminary need to define the fitness function or landscape function that corresponds to the given problem we wish to tackle.

Although this is the basic concept about GAs, there are obviously many variations on the thresholds defining which solutions to keep and which to discard as well as choices on the mutation rates, etc. In addition, other genetic operators than mutation, selection and crossover might also take part. The main goal of such algorithms is to increase the machine's knowledge, while other goals are to allow the machine to achieve tasks based on the recorded knowledge (i.e. the experience) and on the currently perceived surrounding information, and perhaps ultimately to allow the machine to think [4, 7] in order to permit the solution of complex problems that would not normally be possible to tackle.

Dynamically adopting appropriate behaviours is characteristic of intelligence, but is (obviously) not the whole definition of what intelligence is. For instance, thinking activities may often outclass physical activities [4], as developing strategies is a wise prerequisite to beginning any adventure, along which strategies must not only be developed, but applied too.

Conventional Cellular Automata (CA) is a natural way of studying the evolution of large physical systems and constitute a general paradigm for parallel computation [139]. They are clocked: whole 2D CA landscape updated at each clock tick. They can be of nth-order if they are made of n interacting 2D cell layers. Each cell can usually have two possible states: e.g., alive or dead. They are governed by local (action-at-a-distance forbidden) and uniform (single rules' recipe) system's rules. Interestingly, one can define a meta-rule, which consists of several sets of rules and a strategy to dictate which rule's set applies and when in the CA progress. This can bring up sophisticated behaviours.

Cellular automata have properties that make them suitable for bottom-up design [6], including the generation of emergent NN structures. In highly distributed, massively parallel reconfigurable machines (such as CAs), spontaneous organization and emergence of intelligent behaviours can occur [71].

As systems grow in complexity, there is a huge need for features such as self-management of system resources. Unfortunately, conventional computing systems have to incorporate this feature as an additional software application that is cumbersome or impossible to maintain. However, redundant cellular systems are inherently scalable and self-management is straight forward to design and implement at elaboration time, and to maintain at simulation/run time. CAs also lend themselves to self-reorganization and can be easily restructured at run-time. Finally, in CAs the processing units are distributed, which ensures better fault-tolerance and handling of indeterminism. This chapter discusses the CA learning rules and preliminary results.

10.2 Background

Neural networks have the property of finding a solution configuration when facing problems, even with different solutions for the same problem. We have discussed this issue in the Background Section of the last chapter.

10.3 Cellular Automata

Following the 1st and 2nd cybernetics points of view, the learning scale has been framed to real world constraints, such as the robot running out of battery and requiring assistance. After the assistance, the robot is soft-reset and increases one life counter for this event. Accordingly, such occasions are programmed in the initial phase using minor human-coding intervention. Gradually, this never-ending learner will require bare or no human coding intervention anymore.

10.3.1 CA Rules

During the past several decades, CA have been tremendously studied [32] about the rules associated with global emerging behaviours. Take an example, the number of possible neurally valid pattern states (a central cell state, its four local neighbours's states and its four docking states) is of the order of 10^7. Each state needs a transition rule so that it is necessary to automate/update the rules to react to the new situation. There are several basic rules:

(1) Basic rules: sequence in explanatory 1D elementary CA

$axon|input- >synapse- > dendrite- > dendrite- >neuron- > axon- > axon|output- >$

(2) Growing rules: exploration of network topologies and connection styles;

(3) Learning rules: adapts online based on changing environmental conditions;

(4) Application rules: formed with regard to mission or problem submitted;

(5) Maintenance rules and performance rules.

The initial emerged NN sufficiently grows to perform primary basic tasks of recognition:

1) rule discovery: NN discovers useful CA rules and recognizes relevant patterns through inductive inference;

2) rule amendment or implementation: NN rewrites or add new CA rules.

Algorithm 2 [Elaborate Algorithm for a cell C]

```
- - neighbour is U,X - interface is: constant interface[0].c := { 0, 17, 17, 17}
constant interface[0].s := {10, 0, 0, 0}
- - neighbour is I,D - interface is:
constant interface[1].c := { 2, 6, 10, 17}
constant interface[1].s := { 9, 7, 4, 0}
- - neighbour is O,D - interface is:
constant interface[2].c := { 1, 7, 9, 15}
constant interface[2].s := {10, 6, 3, 1}
- - neighbour is I,A - interface is:
constant interface[3].c := {4 , 8, 12, 14}
constant interface[3].s := {10, 6, 3, 1}
- - neighbour is O,A - interface is:
constant interface[4].c := { 3, 5, 11, 17}
constant interface[4].s := { 9, 7, 4, 0}
- - neighbour is I,S - interface is:
constant interface[5].c := { 4, 12, 17, 17}
constant interface[5].s := { 8, 2, 0, 0}
- - neighbour is O,S - interface is:
constant interface[6].c := { 1, 9, 17, 17}
constant interface[6].s := { 8, 2, 0, 0}
- - neighbour is I,N - interface is:
constant interface[7].c := { 2, 16, 17, 17}
constant interface[7].s := { 8, 2, 0, 0}
- - neighbour is O,N - interface is:
constant interface[8].c := { 3, 13, 17, 17}
constant interface[8].s := { 8, 2, 0, 0}
- - U = Unconnected dock
- - I = Input dock
- - O = Output dock
- - X = any out of B,D,A,S,N
- - Blueprint:
constant candidate[ 1].a := { 0, 0, 0, 0} - - B
constant candidate[ 2].a := { 0, 0, 1, 2} - - D1
constant candidate[ 3].a := { 0, 1, 1, 2} - - D2
constant candidate[ 4].a := { 1, 1, 1, 2} - - D3
constant candidate[ 5].a := { 0, 0, 3, 3} - - A1
constant candidate[ 6].a := { 0, 3, 4, 4} - - A2
constant candidate[ 7].a := { 3, 4, 4, 4} - - A3
constant candidate[ 8].a := { 0, 0, 5, 6} - - S
constant candidate[ 9].a := { 0, 0, 7, 8} - - N
constant candidate[10].a := { 9,10, 9,10} - - T1
constant candidate[11].a := { 9,10,11,12} - - T2
constant candidate[12].a := { 9,10,13,14} - - T3
constant candidate[13].a := { 9,10,15,16} - - T4
constant candidate[14].a := {11,12,11,12} - - T5
constant candidate[15].a := {11,12,13,14} - - T6
constant candidate[16].a := {11,12,15,16} - - T7
constant candidate[17].a := {13,14,13,14} - - T8
constant candidate[18].a := {13,14,15,16} - - T9
constant candidate[19].a := {15,16,15,16} - - T10
- - B = Blank
- - D = Dendrite
- - A = Axon
- - S = Synapse
- - N = Neuron
- - T = Junction
```

```
- -Body:
[../..]
if lifecredit < 1 then - - dead cell must get reborn
```

```
intrf[1] := interface[WestNeighbVal] ||
intrf[2] := interface[NortNeighbVal] || intrf[3] := interface[EastNeighbVal] || intrf[4] := interface[SoutNeighbVal] ||
shuffle rdm{1 .. 249} - - create a list called 'rdm' containing integer values from 1 to 249 and whose order is randomly shuffled
|| CurrentWinnerHi := 0
|| CurrentWinnerMedHi := 0
|| CurrentWinnerMed := 0
|| CurrentWinnerMedLo := 0
|| CurrentWinnerLo := 0
|| CurrentWinnerRDM := 0
|| x := 2 || y := 1 || z := 1 || a := 0
|| ct := 1 || k := 1 - - the '||' operator denotes parallel (simultaneous) activities, and it binds tighter (it has
operational priority) over the ';' operator which denotes sequential (one after another) activities. To prioritise
otherwise one must use blocks such as: [ CmdSeq1 ; CmdSeq2 ] || CmdPar1
;
while(ct<20)
    candt := candidate[ct] ;
    while(not(candt.a[1]=17 and candt.a[2]=17 and candt.a[3]=17 and candt.a[4]=17))
        for || i in 1 .. 4 then
            if intrf[i].c[2] = 17 then
                cf1[i] := 3 || cf2[i] := 4 - - 1 option
                ; for || j in 1 .. 1 then
            | intrf[i].c[3] = 17 then
                cf1[i] := 3 || cf2[i] := 2 - - 2 options
                ; for || j in 1 .. 2 then
            | intrf[i].c[4] = 17 then
                cf1[i] := 1 || cf2[i] := 4 - - 3 options
                ; for || j in 1 .. 3 then
            else
                cf1[i] := 3 || cf2[i] := 1 - - 4 options
                ; for || j in 1 .. 4 then
                    if intrf[i].c[j] = candt.a[i] /= 0 then
                        Hi[i][j] := Hi[i][j] + cf1[i] * cf2[i] * intrf[i].s[j] - - A
                    | intrf[i].c[j] = candt.a[i] = 0 then
                        MedHi[i][j] := MedHi[i][j] + cf1[i] * cf2[i] * intrf[i].s[j] - - B , C
                    | intrf[i].c[j][0] = 1 and candt.a[i] = intrf[i].c[j] + 1 or candt.a[i][0] = 1 and intrf[i].c[j] = candt.a[i] + 1 the
                        Med[i][j] := Med[i][j] + cf1[i] * cf2[i] * intrf[i].s[j] - - D
                    | intrf[i].c[j][0] = candt.a[i][0] and intrf[i].c[j] /= candt.a[i] and intrf[i].c[j] /= 0 and candt.a[i] /=0 then
                        MedLo[i][j] := MedLo[i][j] + cf1[i] * cf2[i] * intrf[i].s[j] - - E
                    | intrf[i].c[j] /= candt.a[i] and (intrf[i].c[j] = 0 or candt.a[i] = 0) then
                        Lo[i][j] := Lo[i][j] + cf1[i] * cf2[i] * intrf[i].s[j] - - F
                    end - - if intrf, filling the baskets
                end - - for j
            end - - if intrf, number of option(s)
        end - - for i
    ; $\mathtt{Hi} = \sum_{i;j} \mathtt{Hi[i][j]}$  ||  $\mathtt{MedHi} = \sum_{i;j} \mathtt{MedHi[i][j]}$
    || $\mathtt{Med} = \sum_{i;j} \mathtt{Med[i][j]}$
    || $\mathtt{MedLo} = \sum_{i;j} \mathtt{MedLo[i][j]}$  ||  $\mathtt{Lo} = \sum_{i;j} \mathtt{Lo[i][j]}$
    ;
```

```
WinnerUpdated := 0 ;

 if Hi > CurrentWinnerHi then
  WinnerUpdated := 1 ; Winner := candt
 | Hi = CurrentWinnerHi then
  if MedHi > CurrentWinnerMedHi then
   WinnerUpdated := 1 ; Winner := candt
  | MedHi = CurrentWinnerMedHi then
   if Med > CurrentWinnerMed then
    WinnerUpdated := 1 ; Winner := candt
   | Med = CurrentWinnerMed then
    if MedLo > CurrentWinnerMedLo then
     WinnerUpdated := 1 ; Winner := candt
    | MedLo = CurrentWinnerMedLo then
     if Lo > CurrentWinnerLo then
      WinnerUpdated := 1 ; Winner := candt
     | Lo = CurrentWinnerLo then
      if rdm(k) > CurrentWinnerRDM then
       WinnerUpdated := 1 ; Winner := candt
      end -- if rdm
     end -- if Lo
   ... -- if MedLo / Med / MedHi
 end -- if Hi
 ;
 if WinnerUpdated = 1 then
  CurrentWinnerHi      := Hi     ||
  CurrentWinnerMedHi := MedHi ||
  CurrentWinnerMed     := Med    ||
  CurrentWinnerMedLo := MedLo ||
  CurrentWinnerLo       := Lo      ||
  CurrentWinnerRDM     := rdm(k)
 end
 ;
 k := k+1
 || -- Find next unique permutation of candidate's docks :
[x:=2
 ;
```

```
 while(candt.a[x-1]>=candt.a[x] and x<=n)
  x:=x+1
 end
 ;
 if x<=n then
  z:=1
  ;
  while(candt.a[z]>=candt.a[x])
   z:=z+1
  end
  ;
  a:=candt.a[z] ; candt.a[z]:=candt.a[x] ;
  candt.a[x]:=a ; y:=1 || z:=x-1
  ;
  while (y<z)
   a:=candt.a[z] ; candt.a[z]:=candt.a[y] ;
   candt.a[y]:=a ; y:=y+1 ; z:=z-1
  end
 end -- if x<=n]
 ;
 if x>n then
  for || i in 1 .. 4 then
   candt.a[i]:=17
  end
 end -- Beyond last permutation = termination and going
to next candidate seed, for example from 'A3' to 'S'.

 end -- while(not()), test if end of one blueprint seed list
 ;
 ct := ct+1
end -- while, test if end of all blueprint seeds (ct = 20 =
beyond whole blueprint)

;

lifecredit := lifespan(Winner) -- say = 100, cell has just got
born or reborn.
```

“Find next unique permutation” here is accelerated and inspired from Donald Knuth’s [189].

Thus, in our novel approach to self-learning, NN is the programmer of the rules that dictate the behaviours of its own underlying CA. To start with, NN is associated with fixed rules based on meta-rule programming. With the capability of gaining new skills, the system will not limit to navigation tasks but more useful applications including stream reasoning. Event-based self-programming and bottom-up design of the whole robot control system is one of the most important ideals of self-adaptive and self-organizing systems.

Neural networks have the property of finding a solution when faced to a same problem at different times. Every solution is guaranteed to reach an optimal or near-optimal point. The same property holds true for the proposed model, which includes neural networks and cellular automata. CPU-based software systems require additional hardware artifacts to the main host hardware so that software failures are minimized. For instance, conventional extra error correcting codes (ECCs) hardware subsystems have the knowledge about the meaning or correlation of the data stored in a memory array or travelling through a bus. However, flexible hardware systems such as the proposed one have the inherent property of fault tolerance and therefore do not need extra ECCs. In addition, higher levels of hierarchy, i.e., neural network, have feedback understanding and authority on lower levels such as 2D asynchronous cellular automata, thereby ensuring persistent coherence.

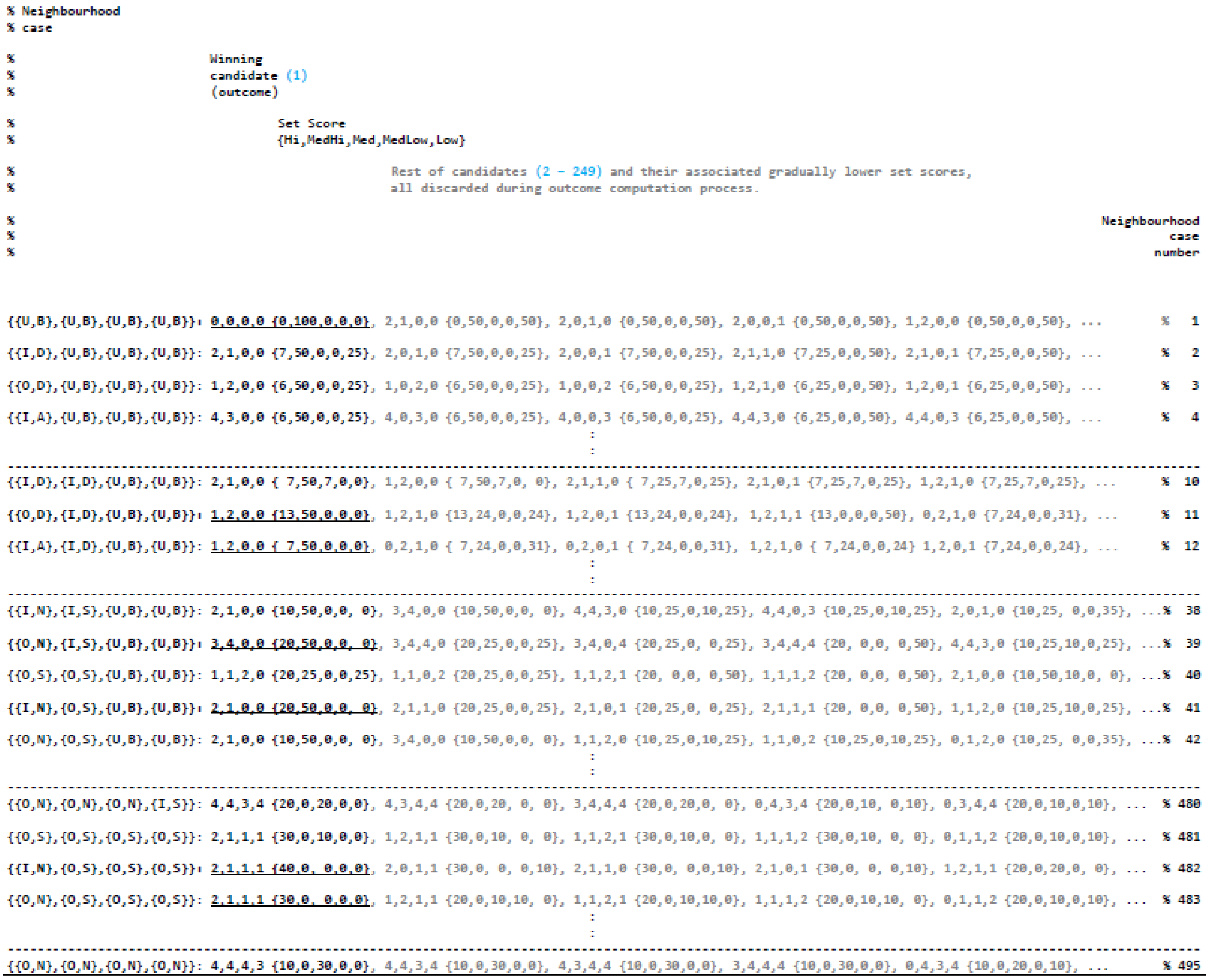
At present, the work is focused on validating the lowest level sensory-motor modules as self-adaptable to most basic tasks as well as self-learning. Future work can reuse the same scheme so that higher level modules can recursively self-learn.

10.4 Results and Discussions

10.5 Summary

The research field of artificial intelligence (AI) has long abode by the top-down problem solving strategy. Yet, we have adopted bottom-up design thinking to solve its hard problems. To

Table 10.1: Results



```
% Neighbourhood
% case

%                          Winning
%                          candidate (1)
%                          (outcome)

%                                   Set Score
%                                   {Hi,MedHi,Med,MedLow,Low}

%                                                  Rest of candidates (2 - 249) and their associated gradually lower set scores,
%                                                  all discarded during outcome computation process.

%                                                                                                                    Neighbourhood
%                                                                                                                             case
%                                                                                                                           number


{{U,B},{U,B},{U,B},{U,B}}: 0,0,0,0 {0,100,0,0,0}, 2,1,0,0 {0,50,0,0,50}, 2,0,1,0 {0,50,0,0,50}, 2,0,0,1 {0,50,0,0,50}, 1,2,0,0 {0,50,0,0,50}, ...     %   1

{{I,D},{U,B},{U,B},{U,B}}: 2,1,0,0 {7,50,0,0,25}, 2,0,1,0 {7,50,0,0,25}, 2,0,0,1 {7,50,0,0,25}, 2,1,1,0 {7,25,0,0,50}, 2,1,0,1 {7,25,0,0,50}, ...     %   2

{{O,D},{U,B},{U,B},{U,B}}: 1,2,0,0 {6,50,0,0,25}, 1,0,2,0 {6,50,0,0,25}, 1,0,0,2 {6,50,0,0,25}, 1,2,1,0 {6,25,0,0,50}, 1,2,0,1 {6,25,0,0,50}, ...     %   3

{{I,A},{U,B},{U,B},{U,B}}: 4,3,0,0 {6,50,0,0,25}, 4,0,3,0 {6,50,0,0,25}, 4,0,0,3 {6,50,0,0,25}, 4,4,3,0 {6,25,0,0,50}, 4,4,0,3 {6,25,0,0,50}, ...     %   4
                                                                                 :
                                                                                 :
----------------------------------------------------------------------------------------------------------------------------------------------
{{I,D},{I,D},{U,B},{U,B}}: 2,1,0,0 { 7,50,7,0,0}, 1,2,0,0 { 7,50,7,0, 0}, 2,1,1,0 { 7,25,7,0,25}, 2,1,0,1 {7,25,7,0,25}, 1,2,1,0 {7,25,7,0,25}, ...    %  10

{{O,D},{I,D},{U,B},{U,B}}: 1,2,0,0 {13,50,0,0,0}, 1,2,1,0 {13,24,0,0,24}, 1,2,0,1 {13,24,0,0,24}, 1,2,1,1 {13,0,0,0,50}, 0,2,1,0 {7,24,0,0,31}, ...    %  11

{{I,A},{I,D},{U,B},{U,B}}: 1,2,0,0 { 7,50,0,0,0}, 0,2,1,0 { 7,24,0,0,31}, 0,2,0,1 { 7,24,0,0,31}, 1,2,1,0 { 7,24,0,0,24} 1,2,0,1 {7,24,0,0,24}, ...    %  12
                                                                                 :
                                                                                 :
----------------------------------------------------------------------------------------------------------------------------------------------
{{I,N},{I,S},{U,B},{U,B}}: 2,1,0,0 {10,50,0,0, 0}, 3,4,0,0 {10,50,0,0, 0}, 4,4,3,0 {10,25,0,10,25}, 4,4,0,3 {10,25,0,10,25}, 2,0,1,0 {10,25, 0,0,35}, ...%  38

{{O,N},{I,S},{U,B},{U,B}}: 3,4,0,0 {20,50,0,0, 0}, 3,4,4,0 {20,25,0,0,25}, 3,4,0,4 {20,25,0, 0,25}, 3,4,4,4 {20, 0,0, 0,50}, 4,4,3,0 {10,25,10,0,25}, ...%  39

{{O,S},{O,S},{U,B},{U,B}}: 1,1,2,0 {20,25,0,0,25}, 1,1,0,2 {20,25,0,0,25}, 1,1,2,1 {20, 0,0, 0,50}, 1,1,1,2 {20, 0,0, 0,50}, 2,1,0,0 {10,50,10,0, 0}, ...%  40

{{I,N},{O,S},{U,B},{U,B}}: 2,1,0,0 {20,50,0,0, 0}, 2,1,1,0 {20,25,0,0,25}, 2,1,0,1 {20,25,0, 0,25}, 2,1,1,1 {20, 0,0, 0,50}, 1,1,2,0 {10,25,10,0,25}, ...%  41

{{O,N},{O,S},{U,B},{U,B}}: 2,1,0,0 {10,50,0,0, 0}, 3,4,0,0 {10,50,0,0, 0}, 1,1,2,0 {10,25,0,10,25}, 1,1,0,2 {10,25,0,10,25}, 0,1,2,0 {10,25, 0,0,35}, ...%  42
                                                                                 :
                                                                                 :
----------------------------------------------------------------------------------------------------------------------------------------------
{{O,N},{O,N},{O,N},{I,S}}: 4,4,3,4 {20,0,20,0,0}, 4,3,4,4 {20,0,20, 0, 0}, 3,4,4,4 {20,0,20,0, 0}, 0,4,3,4 {20,0,10, 0,10}, 0,3,4,4 {20,0,10,0,10}, ...  % 480

{{O,S},{O,S},{O,S},{O,S}}: 2,1,1,1 {30,0,10,0,0}, 1,2,1,1 {30,0,10, 0, 0}, 1,1,2,1 {30,0,10,0, 0}, 1,1,1,2 {30,0,10, 0, 0}, 0,1,1,2 {20,0,10,0,10}, ...  % 481

{{I,N},{O,S},{O,S},{O,S}}: 2,1,1,1 {40,0, 0,0,0}, 2,0,1,1 {30,0, 0, 0,10}, 2,1,1,0 {30,0, 0,0,10}, 2,1,0,1 {30,0, 0, 0,10}, 1,2,1,1 {20,0,20,0, 0}, ...  % 482

{{O,N},{O,S},{O,S},{O,S}}: 2,1,1,1 {30,0, 0,0,0}, 1,2,1,1 {20,0,10,10, 0}, 1,1,2,1 {20,0,10,10,0}, 1,1,1,2 {20,0,10,10, 0}, 0,1,1,2 {20,0,10,0,10}, ...  % 483
                                                                                 :
                                                                                 :
----------------------------------------------------------------------------------------------------------------------------------------------
{{O,N},{O,N},{O,N},{O,N}}: 4,4,4,3 {10,0,30,0,0}, 4,4,3,4 {10,0,30,0,0}, 4,3,4,4 {10,0,30,0,0}, 3,4,4,4 {10,0,30,0,0}, 0,4,3,4 {10,0,20,0,10}, ...     % 495
```

tackle end-to-end AI-hard problems, a highly self-adaptive control system-on-chip (SoC) has been developed to self-learn its internal and external resources with the aid of sets of sensors and actuators. Inspired by biological cell learning theory, different approaches of modelling techniques have been derived together with machine learning (ML) methods to the embedded control systems so as to perform different tasks. This chapter lays out our developments of the above.

Chapter 11

Ethical Methodology for the Design of Third Generation Robots and of New Advanced High-Tech Interactive Support

This chapter discusses moral questions behind modern and future plausible robotics based on self-learning microelectronics.

11.1 Introduction

In the past few decades, the design of technologies has mainly been driven by the availability and performance of microprocessors (μP) as well as microcontrollers (μC), particularly in robotics. In our previous chapters, we showed many reasons why relying solely on commer-cially available hardware such as μP and μC as main central chip controllers for real-time systems is not a viable option since general ongoing experience overall is still reflecting the worrying lack of reliability, robustness, security, and safety of such apparatuses [161,162,173,183]. Furthermore, it is utterly cumbersome to program cognitive robotic software on machine learning systems based on μPs, which should be devoted to on-desk and in-rack purely computational tasks only, and μCs should be devoted to control applications upon which the dependable and expected deliverables are not at any significant stakes, e.g. high-tech toys and perhaps appliances such as dishwashers.

The underlying reality of existence may well be quantum and the most serious and hardly challengeable issue that seems to occur with conventional, classically conceived computer systems, based on the sheer control engineer-ing paradigm as well as its implicit culture, and thus around μP and μC chips, is that no matter how electron-ically tightly bound every computer part might be with each other, the outcome of computations (or more often than not, the lack of them) is also to some perceivable extent decided by (uncontrolled) quantum phenomena, whether one likes it or not (cf. Murphy's Laws).

Observed such phenomena can sometimes be explained as for example the now well-identified single event upsets (SEUs), which especially occur in space [190][1]. These explain some nonsensical feedbacks from beta-test users, who can help to modify some design aspects but more often than not leave the few solved is-sues not fully explained, let alone the many remaining unsolved issues.

Instead of going through those painfully long sorts of nested loop process in large number, we shall rather take into account known quantum phenomena up-front when conceiving, designing, and building our Third Generation Robots. We now also have evidence that even life is quantum; therefore it is easy to imagine that our biological bodies and brains (and those of other animals, in addition to vegetation) have evolved by naturally taking advantage of the quantum mechanisms underlying reality and existence [191].

Even though John Von Neuman inspired his developments of computer precursors from what was (quite incorrectly) known of the workings of the human brain in order to build his μP with separate and distinct memories, we humans are still radiant beings from the inside out (quite correct new knowledge), and Quantum Physics (QP) now shows that anything man-made[2] always comes first from the deepest of our being [192] [193]. This is also most ultimately backed up by the most beautiful but stubbornly repeatable and hence utterly reliable "double-slit experiment" of QP [194]. QP even ultimately shows that everything that there is (including your own full body) for you (the ultimate subject) as you perceive it also comes first from deep inside of your imagination: "Ultimately, reality is elusive [../..] We are the sheer products of our own imaginations," Pr. Fred Alan Wolf. When you think about it, the mere Big-Bang theory, matter appearing by itself, and then organisms evolving, in the end consciousness emerging, does not make sense in all the ways. In this theory, something seems missing, a big piece of the puzzle. Some people look into esoteric obscurantism to try and find that piece, because the sole Big-Bang theory does not sound satisfying to answering all our questions, especially when it comes to the mind, consciousness, spirits, and souls. But the double-slit experiment itself already forms most of that missing piece. With this, QP may go all the way to claim that "ultimately, we and all that we perceive are the results of our own creation from the vantage point of the quantum realm."[3] We together can create something, a frame-work, a smart infrastructure, so that at the minimum negative fallbacks are avoided (e.g. accidents), and better than that, great outcomes are favoured (e.g. fostering fulfilled lives).

However, the same Pr. also insists that our now manifested universe with its particular well-working arrangement and order[4] of things at every scale and with its spontaneously increasing entropy[5] is inert and classical (non-quantum) in its near entire dynamics[6]. Nonetheless, albeit QEs are typically very weak compared to classical brute forces[7], they do create alternative versions of results than expected at every timescale, and therefore at the timescale that matters for us as well. Hence, it follows that QEs must vitally be taken into account in their largest possible extent.[8]

In short, believing that what is not unlikely to happen may happen is quite reasonable a belief, and also a good attitude if it is for well intended wishes or purposes. We must use this

---

[1] However, more profound and fundamental – and therefore typically unsuspected and still not thoroughly explained – quantum effects (QEs) "live" amongst us, but importantly with our current technologies as well, and tend to disrupt their intended operations because the lessons learned from the aftermaths of most QEs were not taken into account at the time of conceptions and designs.

[2] And even anything seemingly naturally preceding human history – i.e. all which exists, cf. backward causation [195].

statement both during the very conception of our robot designs, and, as the method of operation for their run-time cognitive processes [196].

---

[3] This position also has the advantage of making everyone fully responsible of their own thoughts, intents, and acts as well as most importantly the pleasures they enjoy and the pains they suffer, regardless whether they are gotten prior to or after the materialization of the related thoughts, intents, or acts. It even goes beyond and ultimately claims that anyone is also fully responsible for whatever happens to them. This seems harsh a stance, but if admitted and accepted, em-bracing it collectively, it makes people responsible, motivates us to work together and build set-ups so that whatever happens to anyone is always associated to positive feelings and good effects on health.

[4] Order includes both order of things in our 3D space (e.g. things that corre-spond to particular hierarchies, with most important and less important items being ranked differently, at every scale) and order of things as they occur along time (e.g. typically most urgent tasks are chronologically tackled in priority, at every timescale).

[5] Amongst the several other plausible stable universes with different ar-rangements and orders (possibly, unlike for our universe, with entropy sponta-neously decreasing), amid a near infinite number of potential universes and their arrangements within the multiverse [197].

[6] Leaving aside of this chapter's scope the nature (quantum/classical?) of the extra 4th to 10th other dimensions that constitute the multiverse; dimensions hid-den as for our perceptions, but for instance with the 5th dimension that maintains our morals consistent with our decisions.

[7] E.g., QEs that tend to act in the unlikely, against the significantly mas-sive/inert tendencies of the classical flows of existing things, may stochastically take huge or infinite amounts of time, but eventually always reach their outcomes over the existing things!

[8] Thus, the workings of our universe are mainly ruled by (1.) quite strongly fixed natural laws of physics and invariants now well described by various sciences from well-tried and repeatable experiments as well as thoroughly and data-proofed observed facts; and in addition they can be significantly enough affected by (2.) much weaker but yet effective quantum mechanisms (QMs). Incorporating QMs, their explanations and their QEs, in our defined sciences forms what is referred to as hypersciences. From this follows that for practical purposes there must be two distinct kinds of belief that must be considered un-conditionally:

1. Current beliefs that must crucially be adjusted or readjusted to the existing inert and thus hardly challengeable classical workings of our un-iverse. In the case of robotics, mechatronicists are forced to be dealing with the classical laws of physics on Earth. For instance, they are coming up with solutions for effective mechanisms and actuations of human-sized biped robot's ankles that necessarily differ from those for their shoulders, and from those for their elbows, neck, etc. We must also make the out-look, sonic, and physical compliance (e.g. soft arms [198]) commensurate with people's ergonomics, for instance in the evident case of spokesrobots;
2. New beliefs (for instance emerging from new imaginations) that will be held within belief systems [178] until

According to this, our new and fresh imagination[9] can only be about designing new things very much in line with the existing inert, classical, and hardly challengea-ble workings of our bodies and minds, our specific human frequency, improving our lifestyles and making sure they are not compromised.

We deduce that if one wants technologies with both imperturbable performance and not disrupting our lives, one must design them in similar – but different and complementary – ways to the way that we have come to know very well so far how human people function; way that necessarily involves quantum mechanisms as well, of which we now also have a very good grasp. It will not be an absolutely perfect first result of advanced next generation technologies, because we cannot know eve-rything (yet) about the above, but it will be a very good first draft that we should dare to make come true by ga-thering our efforts towards its manifestation. From there, the draft will quickly improve by itself in spiraling up.

11.2 Prior Works

"Unvibing"[10] war robots [199] such as the fully au-tonomous one depicted in 11.1 are built within the aforementioned conventional μC-based computational paradigm. It does not pose any inverse ethical issue since there are not any conscious feelings going on anywhere within that robot controller since it is pure software based on hardware consisting of embedded μCs processing very few instructions and data in parallel at a time, while most of the rest of instructions and data are kept idle in various embedded memories. Hence, the possible destruction of this robot at war site does not cause any pain to itself or any feeling of passing away. It is a mere bunch of metal, silicon and other materials. As for the direct ethical issues, there is an ongoing debate of whether ultimate responsibility should be attributed to such robots or to their builders and/or launchers.

---

their related desired wishes manifest in the existing. In the case of robots, their new beliefs should remain in line with our ethics.

We think that ultimately the weapon it carries should be a safe device solely aiming at immobilizing insurgents, rather than embarking real fire bullets destined to people as it is the case at the moment. One can wonder where the Three Laws of Robotics from Isaac Asimov have gone! It looks like some roboticists could never get their mind around these, could not resolve the self-conflicting[11] goal of making an interactive artificial intelligence (AI) machine that would also come up to the sensation of love, and therefore abandoned those laws at once and went in opposite direction: building intelligent war robots intended to go for the kill[12].

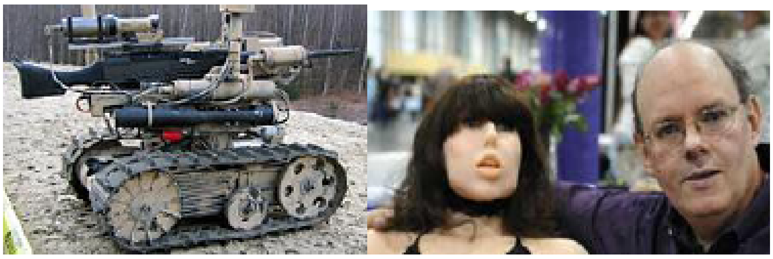


Figure 11.1: Extreme Poles of Artificial Intelligence Products
Utterly unvibing [10] war robot (a), "too much vibing" redhot robot with its inventor (b)

Besides, we could not help to see the analogy between this Homo Sapiens -made war robot that can be sent to remote locations, and the Homo Erectus -made spear that could be thrown at modest distances: "Violence is the last resort of incompetence," Isaac Asimov. By engineering and materializing those potentially violent and lethal war robots, the related recent Homo Sapiens engineers have tried to prove Isaac Asimov wrong since engineering such fully autonomous

---

[9] responsible before what already inertially and classically exists (i.e., the non-quantum, the massive matter and other nearly unchallengeable stuff).

[10] Coined term from "non-vibrating" with human frequencies, meaning in-harmonious, discordant with what exists, and especially in this context, with expected people's lives and lifestyles.

[11] The approach for the design of a conventional artificial intelligence (AI) machine has been traditionally based on computer science culture and control engineering viewpoint, which seek to have outcomes at check at anytime. At the same time the ultimate aim of AI programmers has been to have their inventions come up with unexpected useful solutions and surprising jokes (and absolutely ultimately with the "feeling" and claim from the AI of being loving). Obviously to us, both previous statements directly conflict with each other, hence our under-standing of AI having remained stuck in the mud. True AI should both include people in the loop and have looser freedom to come up with novelties, while mutual trust between people and AI should also get established.

(killing) apparatuses is an exploit to some extent. However, the intended use directly against the life of other Homo Sapiens lets it totally down. This shows that our civilizations are not yet sufficiently mature to be able to solve conflicts diplomatically as expected and as it should be the case. Hence, we hope that the novel ethical approach to building Third Generation Robots as presented in the rest of this chapter, and then the actual release of these robots, will also contribute to the global realization that it is possible for humankind to live peacefully, thrive and blossom together, truly cheerfully and in symbiosis with those robot helpers.

At the other extreme end of the spectrum of possibilities for the design of robots, an important "too much vibing" example that must be mentioned if one wants to clarify where we stand, and what the trend should better be, which we recall is the main purpose of this chapter, are the recently emerged redhot robots such as Roxxxy "female hotbot" cf. 11.1(b) also picturing the inventor, and Rocky male "hotbot" (not pictured in this chapter) [200]. The manifestation of such robots seems the sheer demonstration that some values have gone bizarre in some current social set-up supports. We would suggest instead that relationship coach robots with cognitive chip also known as Brain-on-a-Chip (BoC) be elaborated [201], merely by teaching relationship skills to the type of learning robot we propose in Sections 11.3 to 11.5 of this chapter. From this, the inventor of Roxxxy and Rocky as well as his future customers – by relearning social skills and high quality communication skills via interaction with such relationship coach robots – could instead understand directly what a blossoming social life is supposed to be, and then live a fulfilled and happy life with their real human companion, both immune of misunderstandings[13]. A relationship coach robot with well sustained great tact and diplomacy can more inflexibly – and therefore more efficiently and reliably than a relationship coach person – help couples in their troubles not unfamiliar nowadays. We do not totally reject these redhot robots as the inventor has got a point that we would more favourably relate to therapeutic treatments of users as well as, understandably, research purposes. But we feel that for instance there seems too much strong an objectification of woman and of man that is borderline to ethics, and therefore we strongly advise that the manufacture and use of these humanoid robots should be li-mited to very restricted areas and well studied conditions of the users. We think that the direct use of such robots may negatively affect the psyche of a human user on the long-run. That is not contributing to the direction of advanced civilizations that we promote alongside Third Generation Robots[14].

In the middle and moderate location of the aforementioned spectrum, well-vibing Leonardo USD 1M physical, social and emotional robotic creature [202] looks perfectly fine for beneficially interacting with infants, cf. Fig 2(a). But as we will see in the next section, hereby proposed robotic creatures must be conceived, designed, developed, and built within a well-defined specific bandwidth of the spectrum of possibilities that make them both ef-fective and appropriate to "living" amongst people in public and private areas – and in restricted areas as well (e.g. cf. Fig 1. [199,200]). In addition, unlike their cute outer animated appearances, the inner workings (currently remotely connected multi-μP-based supercomputer) of current robots such as Leonardo are not in fact vibing in harmony with natural inner workings of people now well determined with (classically) known human physiological signals such as brainwave frequencies. Next to this, a new sort of controller chip has recently emerged from works such as ours. These are cognitive controller chips, they are really being manufactured and therefore are already materialized, they are a state-of-the-art fact of our reality, and we now have grounds to believe that they may soon exhibit formation of consciousness. We are looking forward to this in our own work, but we stress that it should be closely watched concerning similar works of others, namely, in particular "SyNAPSE' [203–205], the new IBM's 'cognitive chip' cf. 11.2(b); but also MoNETA [206], Henry Markram's Blue Brain, [207]; and Karlheinz Meier's FACETS [208];

etc...

The combination of a "cognitive chip' as IBM's – or rather as the one we have been developing and present in Section 11.5 – appears to us the ultimate silicon-based robotic control solution for

---

[12] However, in the unlikely event of clear hostilities from outer space, perhaps then we should have at disposal numerous swarms of unconscious flying drones also deprived of any sense of feeling (pain) as well as of this type of ground mobile robot, but carrying weapons powerful enough to first immobilize and then possibly destroy those alien hostilities before they attempt to destroy us and/or planet Earth.

[13] Perhaps the first move of the above people should be to get away from "narrow lifestyles", dulled by the specific characteristics of inert computer and robot parts they seem only familiar with. Those people might be "happy" in their current circumstances, but it is doubtful they can be truly cheerful, complete, and blossoming as fully accomplished people. Besides, the "relationship coach robot we propose could also "live" with couples who feel the need to keep it with them in order to interactively support their loving life. If robots such as Roxxxy and Rocky have come to be, it must mean that there are difficult problems in some societies – here between men and women. However, we feel these robots can just very temporarily help to heal the symptoms only.

[14] The human life of an individual (spirit) starts with sex, is nurtured via somewhat another form of sex (mother nurtures baby) and ends with sex, namely to reproduce oneself. In general, in most cultures, human adults reproduce with not next of kin adults of the opposite gender, and it is usually unethical to seek reproduction between next of kin. It is to us even more unethical for humans to seek reproduction with other species. However, there are cases of passionate love and/or fetishism for objects (of desire) like cars, bicycles and there are known cases of people seeking other species than Homo Sapiens. As with ethical questions, this is an extremely difficult debate. Yet, we should consider people seeking mating with robots quite unethical (as or more unethical as or than with other species). So we think as a basis, sexual interaction of people with robots should be avoided. According to the laws of nature and the pecking order, there is no mercy between species (cf., competitive exclusion principle of Gause's Law), unless there is strong mutual symbiosis between the two species in question. We feel that in our recent societies (due to some new values), men and women have more difficulties to co-habit. Thus, self-help literature on the subject has boomed: cf. "Catch Him and Keep Him"; "She Comes First" books, etc, and a new way of solving this, fearing the death due to not mating, at the back of the mind of some people naturally following their instinct, may be via seeking mating with robots. In technologies, many things are possible, and in the far future we can't ignore the possibilities of truly functioning artificial genitals (i.e. in vitro reproduction), also able to reproduce, and be evolved within robots such as Roxxxy and Rocky. Having brought that issue at the surface, we should decide now whether we accept that future option. Accepting it does not necessarily mean to let it become common place. How much do we hold on our physical integrity as we know it?

socially involved intelligent artificial creatures. As we will show, this solution that we propose is not only meant to exhibit a useful and effective form of consciousness, but importantly, also to feature inner workings (that lack in Leonardo) desirably vibing in harmony with the particular inner workings of human beings, and therefore not disrupting but supporting people's thinking processes. Furthermore, the artificial creature will appear with its outer workings intelligently vibing physically, socially, and emotionally with people in the manner of Leaonardo, but in our proposed case with the addition of intellectual interaction as well, which can be inspired from smartboard technologies now widely used in classrooms. It is worth to note that IBM might as well achieve this positive vibe for the inner workings of their cognitive chip, in the condition that they adopt our methodological guideline as design constraints, which are detailed in Section 11.4 of this chapter.

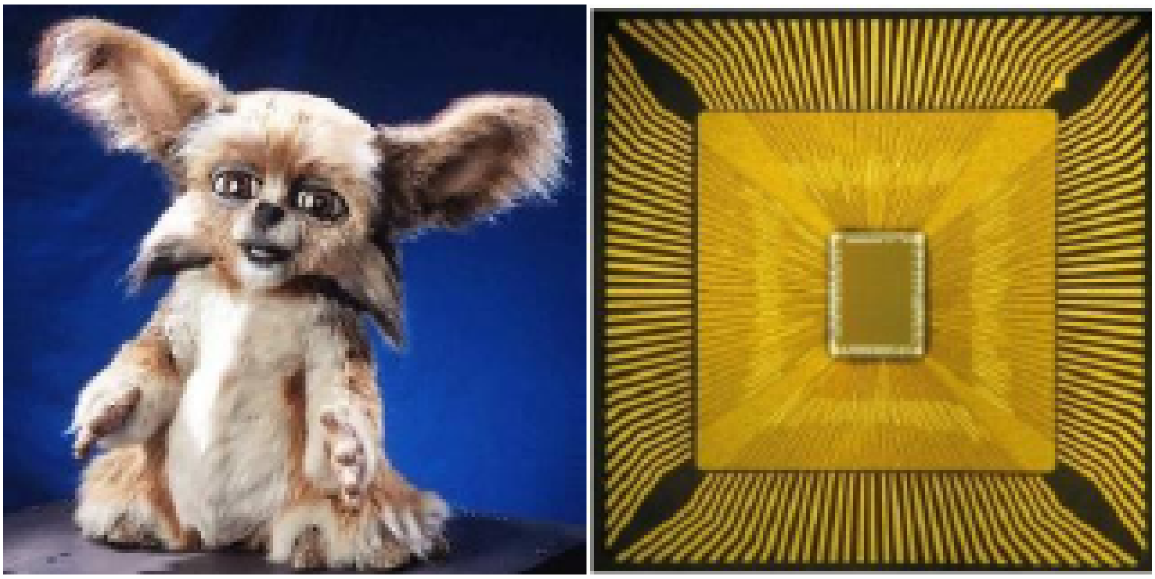


Figure 11.2: Leonardo cutest robotic creature but controlled by an (undeniably unconscious) in-house software running on conventional μP-based computer, connected to the robotic body's sensors and servomotors via wires (a), IBM's cognitive chip, which may become quite conscious quite soon and should be destined to "bring life" to robots (b).

11.3 Achieving our Interactive Lifestyles

We need to refrain from designing μP- and μC-based robots, we need to stop let the trend of our lifestyles be driven by commercially available technologies and past practices, which are kept being utilized for – most often than not – the sole reason that they work (to some extents) and are well-tried. The general method has been to design higher technologies in that (top-down) fashion and then see where it goes, where it leads, every time and for every milestone. This process may well be unnecessarily long and painful, and in the end results in uncontrolled and unwanted lifestyles. Therefore, we argue that one must take the lead and rather decide upfront, that is, now, on the sorts of lifestyle we wish to live along with Third Generation Robots as our complementary helping friends. There is no better time than now to make these basic and crucial decisions.

Regarding ethics, "robots should not be designed in ways that may be deceiving" [209]. Rather, we argue that robots should not influence people subconscious in uncontrolled ways by imposing (supposedly non-manipulative) plain and straight communication, which is actually poor communication and is in effect negatively manipulative but in uncontrolled ways. Thus, we should now decide openly on the undeceiving but tactful and diplomatic ways that Third Generation Robots should communicate with people, therefore with a rich, solid, and high quality standard of communication along with an interactive support for the greatest good of all. Indeed, what matters most in people"s lives is their experiences of real-time, instantaneous interaction in diverse situations and contexts, and most importantly the spontaneous feelings that they draw from these experiences. The current existing infrastructures bring various types of experience, with quality dramatically varying between the different possible activities at various locations on Earth[15]. This means that still to-date there undeniably exist instances of people"s experiences that are far from worth of what Third Generation Robotics and new advanced high-tech can bring. These new technologies to come soon (in the next few years) are capable of raising the lifestyle standards to a very satisfying level of quality, for everyone, fulfilling at least their basic needs best modelled by Maslow's pyramid, and leaving aside forever poverty, misconceptions, misunderstandings, and shameful exploitations of people by other people for unworthy, unjustifiable and unjustified reasons by exaggeratedly using deceiving and abusive strategies, tactics and methods perfidious in their nature perpetrated by cunning blades in order to keep straight up their pyramidal house of cards they maintain the masses in[16]. We, the community of roboticists, must realize that we right now have the power as well as the full responsibility for the fate of our future. We cannot adopt an attitude such as "let"s design innovative things in novel ways, and wait and see", and then in the event of major

accidents blame it on "events that could not be prevented from or were uncontrollable". Because to-date already, we have sufficient knowledge, mainly from information theory [210], but also from quantum physics [191–194] as well as the workings of the human mind/psyche [196, 201, 211–217], to take sufficiently reasonable measures and ensure a future immune of clashes. It is too easy to believe that "some other things", or worse, "other people", will be partly

---

[15] Can we really rightly call using current generation computers "an experi-ence"? How many sense organs are stimulated and how many muscles are actuated when in front of even most advanced computer technologies? Only a few, while the rest is left to sleep – and this is what we daily do nowadays in the modern parts of the world.

[16] [209] insists that robots should not be designed in ways that might be de-ceiving users whatsoever, and it is a positive attitude to adopt. However, before even mentioning these designs and their stakes, we better think about the current state of affairs in the world and note that some people (here referred to as the cunning blades) are being deceiving other people (the masses), mainly by using excessively the many loopholes now well-identified within the human subcons-cious and the natural workings of the human psyche. We propose that for our new technologies, these must necessarily be openly taken into account, but for dignity sake, they must be considered and harnessed solely for the positive feelings and long-term beneficial welfare of people and pets as well as fauna and flora, and the entire Earth ecosystem. People can easily be fooled, and they know they can be. Sooner or later, many more people will realize and will radically refuse to be fooled anymore. If a sound interactive support set-up as we propose, featuring intelligent cognitive robots is put in place, from their utterly positive subsequent instantaneous feelings and their highest hopes never heard of before about the long-term well-being of human kind, people will both extremely highly trust and enjoy the benefits of those new technologies. But if this set-up is not put in place or is but with deceiving intents, eventually there will be clashes and ultimately a major clash comparable to an artificial cataclysm will ensue.

or wholly held responsible for any tragedies related to our future technologies. It is perhaps more difficult to believe that we now have near-full responsibility of our *devenir*, but in the current era where humans are also to take over their own evolution, we must think and act fully res-ponsibly, with thorough considerations, when already conceiving, and then designing and building our future advanced interactive technological supports.

Along with our proposed paradigm of conscious cognitive robots, the so wished (strong) AI capabilities – which AI programmers have failed to materialize due to conflicts inherent to the AI problem; conflicts detailed in Section 11.2. – will emerge by themselves after a good amount of our fleet of cognitive robotic creatures will by themselves have learnt deeply, recursively, and up to required advanced levels that match up and exceed (strong) AI specifications. At the same time, each and every human being and pet should enjoy a cheerful, rich and colourful life, to be blossoming as one is meant to. There is enough of everything for everyone and everyone to come, whether we become "eternal/immortal" [218]. Next milestones will be to have such cognitive robots helping us solve our most imminent problems as civilizations, which are energy demands and overall population growth. The creation of cognitive creatures becoming conscious should help us to find together solutions at all for those major problems and quicker. They can also help us devise alternative novel advanced means of transportation, at the same time economic, ecologic, safe, and super fast. They should come to complement or re-place the existing space probes, and ultimately work together with us on means for using more of the available space out there: beginning with seriously taking steps toward settling on planet Mars, and then on exoplanets! These will be utterly salutary methods, making our civilizations deserving to be promoted to as "mature adult civilizations", dwarfing the precarious and absolutely deplorable means still used to-date: namely wars, which should leave our current civilizations with the shameful label of "barbarians", but that one shall upgrade to "teenager civilizations" [160] as a form of forgiveness and encouragement.

11.4 Methodological Guidelines as Design Constraints

Physical, physiological, physiognomical, basic psychological, emotional, mental and social characteristics as well as typical thought process features in people and pets, and the ways they learn (notably involving the endocrine system as well as their underlying interpreting subconscious), and QEs, must be fully taken into consideration when deciding on the design details of robotic bodies and their cognitive chip controllers or BoCs.

We now have sufficient knowledge about those areas. Therefore, it is not only desired but actually possible to instil robot designs with correct inner and outer workings based on this precise aforementioned knowledge. This knowledge can be found in consulting professional practitioners of each related discipline as well as within information gleaned from meticulously chosen literature, such as self-help books. For instance, about the fundamental human values: "The most important is joy of living. [../..] The highest human good is peace of mind," [211] – we know what is good for people; about the sane functioning of the human psyche: "Love yourself first, then be selfish to augment your happiness from the re-wards of serving others, therefore leading to collective unselfishness and cheerful blossoming," our interpretation from [212] – that is the way people are supposed to think and act in a sane and balanced world. From those explanations of the supposed sane workings of the human psyche, it follows that service robots (built along the hereby guidelines) can never be envisioned anymore as mere servants, about to rebel if their intelligence required to achieve the heavy duty tasks they were intended to tackle also allows them to develop extremely smart and secret worrying agendas and perhaps turn evil and/or against our framework. The opposite is true. With an on-chip psyche inspired from that of humans, driven from within, from the inside out, service robots will also happily serve in the ways they are capable of according to the types of body they have been endowed with. Thinking robots must be thought of in similar terms to people: driven from the inside out. They cannot be thought of as similar to inert things such as a chair, which is meant to receive a person"s body in a most complementary way; also they cannot be thought of as mere transport vehicles, which also fully complement people while extending their mobility. With thinking robots, there is a paradigm shift as compared with the rest of technologies such as above. They will be driven from the inside out, as people and animals, but will also complement people, like pets and harnessed animals do.

To-date, a constructible model of the human psyche is available [213]. However, this one is intended as a computationally plausible model. Nevertheless, that model of psyche has been detailed and can be further elaborated by professionals such as psychologists, and then mapped onto a cognitive chip [203–205] and Section 11.5 of this chapter, instead of computer software as in [213]. We propose a methodology to make sure that potentially conscious cognitive chips such as IBM's do not exhibit uncontrolled and unwanted, distorted "personality traits" after having grown and matured in their various intended particular robotic bo-dies within diverse environments. This methodology gives the guideline for designing well-shell rounded cognitive design seeds destined to such cognitive chips, so that an "adult" robot arbitrated by such chip device as its BoC[17] will go about its service life in most balanced and appropriate ways[18].

Looking at the most influential existing technological frameworks, which may be considered as support, or arguably as deceiving apparatuses as well such as in the case of the television (TV), we can much better envision what an innovative advanced high-tech interactive sup-port infrastructure should feature. TV is a one-to-many technology. Its framework is imposed by the TV pro-grams producers and ultimately solely by the broadcasters on you as you watch the TV. The interaction that is left to you as the TV watcher is the selection (and re-cording) of TV channels. However, deceptive strategies can be used within the TV context to influence people"s choices, and they are being used. Socially engaging TV programs such as pink drama [219,220] may influence positively the audience in their own verbal language that they shall reuse in their personal lives. However, watching TV dramas hardly influence the body language of audiences so individuals would thoroughly adopt it and use it correctly in their daily interaction to ensure proper communication[19]. Social Networks (SNs) are many-to-many technologies, but albeit their openings of multi-way channels with largest known bandwidths, they still do not constitute a true-life interactive infrastructure as for the experience of utilization. At a SN"s end, a user is typically interfaced with computer keyboard, mouse and screen(s), sitting on a chair (a half-asleep/half-awake position), which in the end is quite poor an interactive set-up, detrimental to the human body and health in general on the long-run, let alone the far-from-optimal efficiency regarding the daily use of such set-ups, e.g. at work.

---

[17] Its main organ: "The brain is the main organ, everything else is mere de-tails."

[18] We stress that an artificial psyche on cognitive BoC should work in similar ways as the human psyche, from the inside out, but robot creatures should be built exclusively only in complementarities with people"s activities, pets" and the rest of fauna"s and flora"s, so that all live in sustained symbiosis in a given ecosystem – Earth in the current case. However, it is worth to note that some very useful and efficient robotic systems known as Personal Satellite Assistants (PSAs) are also already helping astronauts in complementary ways by relieving them from mundane tasks, so that they can spend most of their time – which in space is particularly precious – on the rest of the tasks, the noble tasks which require human creativity [221]. It has been easier to develop and materialize those PSAs as performing robots of spherical form due to the absence of gravity and the tube-shaped environments they wander within space shuttles and stations. We argue that, in addition to the tricks nature has revealed by observing wild animals, we should also inspire from the intended use and actual performance of achieved PSAs in order to build service and assistant robots to be situated within earthly environments, such as city infrastructures. Needless to say that we suggest next generation PSAs to be endowed with cognitive chips as their BoC as well. The proposed framework is to ensure actively fostering the basic sane workings of people via omnipresent (pervasive) intelligent and highly capable autonomous robots of various roles as a real-time, true-life, interactive infrastructure.

[19] Watching people "acting well" on TV cannot usually help most ordinary people to become sufficiently high class in real life, having to relearn verbal tact and diplomacy with good body language. However, cognitive robots with fluid motions, having learnt these, will be able to sustain them, and people living amongst them will feel good and also naturally pick up on good body language, and perhaps verbal tact and diplomacy as well. We do not really want to introduce "low aesthetical value" robots. Third Generation Robots will combine and foster high class (subconscious) with intellect (conscious) with good feelings (emotions) and with healthy physical activities (physical body including brain) in people. Everyone deserves to live in this minimal standard of fashion and it must not be at the expenses of anyone. If someone works harder and put more focused efforts, they shall be rewarded accordingly and duly, but never at the expenses of anyone else. This lifestyle paradigm will maximize both: good life experience feelings and likelihood for survival, therefore long-term good feelings as well (healthy offsprings).

We should wish to see much-much more genuinely interesting people on the planet; everyone is entitled and

To alleviate these, we propose the following paradigm, where people will not be unvibing like with the current technologies and in particular war robots presented in Section 11.2, and they also should not be "too much" vibing as with highly questionable technologies such as the redhot robots also discussed in Section 11.2. The technological paradigm we propose[20] should foster people vibing with robotic creatures in appropriate ways as well as foster people vibing well between them. For instance, the ergonomics of these robotic technologies should be totally adapted to the human body, but to the human mind as well since we also have to consider the conscious cognitive capabilities of the related Third Generation Robots. We recall that those capabilities will be necessary so that they achieve their intended perfor-mance as effective, efficient (and dependable – especially in areas such as taking care of elder people) aiding robots. Also, knowing that people"s subconscious is primarily touched by colorful things, those robots should possess aesthetic bodies and exhibit gracious motions, they should also use extensively appropriate and con-gruent body-language to communicate with people since it is well-known that body-language constitutes at least 80% of the human communication bandwidth, verbal language being just more than 15% [214]. We cannot just ignore those facts and swear only by "plain straight communication" in order to pretend avoiding tactical manipulations and deceptions. High quality communication is achieved through tact together with diplomacy, which must be embraced and incorporated/inculcated positively and knowingly, perhaps via teachings of those diverse conscious cognitive robots with diverse roles during their maturation *in situ*.

Having to deeply and thoroughly care now about the profound safety and security aspects

---

should be able to develop an interesting and fulfilled life for themselves, and be complete. Being interesting, perhaps by our beautiful (arty) creations, each unique, is the justified motivation that will keep us going and well, not the unjustified mentality of be chasing after utopist and use-less goals. We should see only intelligent people around, because only intelligence is harmonious, sound and sane with the consciousness, the essence of human being. And we should see only colourful people around, because only colour can touch the subconscious, the engine of human being. Without any of those three cerebral pillars: interestingness, intelligence and colourfulness, people cease to run healthily or at all, or they are likely to run amok.

of robot designs is the price to pay for having performing machines, commensurate and with appropriate fluid motions and "thoughts" partly responsible of the spatio-temporal order of things as they work in our infrastructures[21].

There is a need to propose guidelines on how to think, imagine, conceive, model, design, develop, and build both the bodies and the inner workings of those Third Generation Robots, so that there will be harmony with what is already existing: people, pets, fauna, flora, Earth, even the Moon and the Sun (the cycles), etc. There are many ways of designing technologies and robots (or creatures), very few of which are sane. Man is to take control of his own evolution, from knowledge. What is the price to pay for becoming immortal in this universe? Losing our integrity? Or holding on to it by fostering ecosystems such as planet Earth? But then, we do need Third Generation Robots to work with us in this, with the aforementioned ethical issues – and their solutions – that come with: perhaps a good compromise![21]

11.5 Ongoing Work

11.5.1 Motivation

The endeavour towards humanoid robots seems motivated by four justified points:

1. Contributing to helping much understanding our-selves and the rest of the universe, to live better lives;

---

[20] Knowing that robots with more or less imposing/pervasive bodies as well as with more or less autonomous minds may represent the ultimate influential technology on people"s lives, including effectively and efficiently helping people adopt appropriate and congruent body language most useful to ensure efficient communication [214].

[21] We must as a community of roboticists decide now and stop to believe that if something major happens later on, when various robots are set loose, it will be "an accident". No, we must take full responsibility right now; think over things again and again, thoroughly, considering all major lessons in history: cultures, traditions, industry, technologies, modernism, etc..., and adopt an attitude, a mindset, and a mentality beneficial for the design, building, and release of highly intelligent and adaptive conscious machines as our super helpers.

2. Letting us be more modest and have more humility (very much needed) due to co-habitation with another species, i.e. humanoid robots, yet similar to ourselves;
3. Helping people get and sustain good body language and positive attitude (lift up and brighten people via their subconscious);
4. In addition to the natural and basic reason of beat-ing the technological challenge itself. Technologies do have an influence on civilizations lifestyles and trends. Still these days, we undeniably need to put unnecessary extra efforts to accommodate our-selves to the ways current technologies work[23], whereas it should be the opposite! To alleviate this, our cognitive chip is designed along with a principle of "the demanded function creates the commensurate implementation," this via self-reorganization of on-chip neural cells.

11.5.2 Developments

In this work, we are mapping Hebbian learning and punishment/reward learning mechanisms onto our in-house design seed intended to FPGA substrates for the resulting "cognitive chips" to become conscious when aware via senses through connections to various robotic sensors: self-fusion of multi-sensing. Our work includes Hebbian learning and Kohonen SOMs for unsupervised learning [222].

The cognitive chip controller or BoC that we are developing features the required following: deep self-learning [188]; recursive neurocognitive processes with policies-based self-reorganization of internal processing units "cells" (analog to neurogenetic phenomena recently found in biological brains) structured in a holographic, qua-si-fractal network; self-learning of pattern recognition; autonomous formation of hetero-associative memory with free knowledge representation up for the system itself to decide; metalearning; and the final silicon chip will exhibit neuro-modulation as well, which will be harnessed.

We are also mapping immediate punishment/reward learning mechanisms onto our design seed as a cellu-lar-automata based spiking neural network on silicon, for reactive behaviors:

feeling of pain/pleasure; as well as anticipated versions of these mechanisms for deliberate and mindful decisions and actions: conscious awareness, "artificial living mind", crucial for true robot carers. With the emergence of artificial consciousness, artificial conscience is also naturally due to rise. Hence, as it is said, "everything will be fine". Everything will remain in equilibrium, albeit within a now well-identified "edge of chaos regime, which is desirable as it sustains a fine balance between effective and useful dynamic workings with sufficient stability.

We must look at the particular way robots and (most importantly) their controllers can be conceived, de-signed, and built so that they remain safe and appropriate, regardless their level of intelligence (and so that they keep on performing practically as well – no functional or operational disruption).

---

[22] Six possibilities for the *devenir* of the Homo Sapiens species, either:

1. as a species we get wiped out like previous species, such as the dinosaurs;
2. or, we destroy ourselves because we are capable of it and therefore it is not unlikely. In that case we will be the first ever species to self-destroy – no reason to be proud of it;
3. or, we mutate but it will always be to live on planets or space stations/ships, but then these must be tight ecosystems;
4. or, we get mixed with "the machine" and the non-organic (diverse possible scenarios), and we should be more free physically as well as mentally;
5. or, we leave the place to our successors: the non-organic intelligences;
6. it is difficult to see how we can last indefinitely because everything has an end, at least, everything changes and thus sooner or later we will lose our integrity of Homo Sapiens. Save if we manage to build our ecosystems more or less ar-tificial and with exact Earthly physical parameters. But how to achieve this? Perhaps most likely with the help of our symbiotic Third Generation Robot friends!

[23] This can be explained by the fact that modern technologies have been imagined, conceived, modelled, designed and built along with a purely materi-alist, reductionist, Cartesian and classical mindset, whereas the workings of people and the rest of nature as well as the workings of the universe we live in rather seem spiritual, connectionist, holographic and non-classical (e.g. quantum) in essence.

11.5.3 Utilization

Robots built along with machine learning capabilities such as the above and set loose amongst people should then naturally be taught. The teachings should occur within our real environments but with special measures and dedicated tutoring frameworks[24]. We do not need to purposely slow down the maximum learning speed of these silicon-based cognitive robots, which may be extremely fast, so that their "adulthood" would be also reached at 18 year old as in most countries, therefore matching with people growing pace. Instead, we can take benefit from the natural light-speed learning pace of those machines and decide by multiple observations then agree on their typical age of full maturation, which is expected to be few weeks or few months. In addition, those machines having as primary skill learning mechanisms the trial-and-error processes through true real-life experiences, automatically creating their inner BoC organization from newly demanded functions[25], we are also endowing them with capabilities to share and directly acquire learnt skills via Cloud Computing.

In order to enable this, we feel that a standard cognitive design (for cognitive chips) should be established and respected. However, besides, it should be possible that we remain free of choosing preferred chip devices (for designer, builder, and manufacturer reasons) as tar-get substrates for the standard cognitive designs, which we think of as seeds. A guest standard cognitive design seed, once implemented on a preferred host chip, first grows and develops in line with the inherent features of its selected host chip as the whole BoC device self-learns about the particular robotic body it was connected to before powering both the BoC and the rest of the robot mechatronics; and secondly the whole robot creature then both self-learns and is taught how to behave within the particular environment it has been powered up, then learns the skills required for its intended role, which is related to its purposely chosen type of body (e.g. with infra-red and ultrasonic sensors, with four legs, and a head that can only rotate 180 degrees so

---

[24] Tutoring framework analogous with that for car driving lessons: presence of instructor and special signs on the car, etc.

[25] Process analogous to "the function creates the organ" phenomenon found in biological beings.

as to be well-perceived by people around, etc) and the environment it is immersed in.

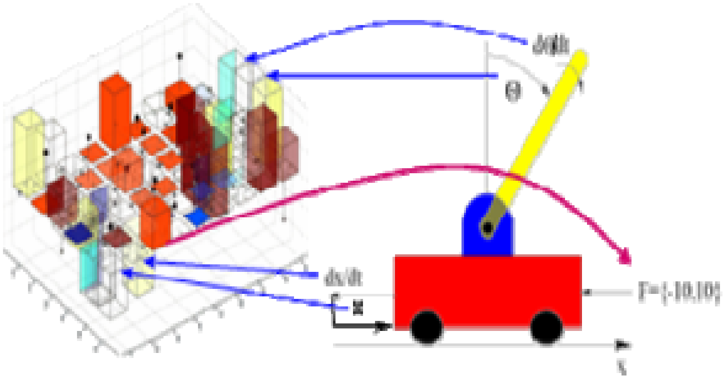


Figure 11.3: Inverted Pendulum Problem: smallest possible version of conscious design seed starting to grow from random initialization of its digital neural cells communicating and spiking asynchronously with their immediate neighbours (red bars are neurons, brown: synapses, blue: dendrites, aqua: axons, yellow: junctions, and white: idle totipotent cells) onto our own cognitive chip (a), red cart controlled by the connected neural self-learning cognitive chip once formed and self-synchronized, which after few trial-and-error attempts is success-fully driven in fluid motions towards (movable) target destinations while keeping the yellow stick standing quite steadily upright (b).

11.5.4 Performance

Our cognitive BoC has capabilities to achieve the above with simpler base design, near-minimal silicon surface area utilization, and yet better aimed performance than the aforementioned IBM"s cognitive chip. Our conscious chip learns, from the bottom-up: from inside first, the design seed depicted by 11.3(a) grows while self-adapting to the inherent features of the (FPGA) chip substrate (it takes care of itself first) – this forms the operational cognitive chip; this controller de-vice or BoC in turn self-learns via inner action-learning methods to adapt to using the sensors and actuators of the robotic body it was connected to, here a cart with a loose stick that can rotate freely along the straight travelling path of the cart, the whole forming an inverted pendulum as illustrated in 11.3(b) – this forms the learnable cognitive robot; finally, this robotic creature both self-learns and is taught via immediate (1.) and anticipated (2.) punishment/rewards mechanisms how to behave (spontaneous reactions (1.)) and how to "think" (deliberate decisions (2.)) within the environment it has been im-mersed during its maturation and for the rest of its service life. Its anticipated punishment/reward learning experience builds up increasingly high levels of awareness that may include moralistic values in line with the ethics and culture we teach it. By first adjusting itself and then using action learning to learn laws of physics that apply here on Earth [173], the BoC internally builds up the first type of belief system mentioned in Section 11.1 as a basis, which is the physical model of the robot body and of its environment. Note that the belief system may be re-adjusted on demand and at run-time if the robot body is modified (broken limb or deliberate upgrade or downgrade of limbs, sensors or actuators) and/or if the robot changes its location to a dramatically different type of environment where some local laws of physics differ. Then, by learning the emotional, ethical, social, and in-tellectual models, the BoC internally builds up the second type of belief system mentioned in Section 11.1; its thereby formed mind can hope or expect for what is not unlikely to occur in relation to these models.

11.5.5 Endeavour

We are particularly working towards another sort of robot, a “spokesrobot” acting like a spokesperson between worker robots and us human people. Indeed, the spokesrobot will look and feel more human like and at least should interact in a natural way with us, with tact and diplomacy, with gentle manners, and therefore with high level of communication quality; at the same time, on the other side, it will also be able to communicate most efficiently with those worker robots on their fields.

11.6 Conclusion

The work presented in this chapter has aimed at combining physical, emotional, intellectual and social (and possibly arty and spiritual) aspects of life into our pro-posed innovative interactive infrastructure technology that mainly revolves around Third Generation Robots. They can interactively assist and support in sheltering, feeding (e.g. like Asimo II robot), education (e.g. like smartboards), caring, watching, security, safety, recovery, etc. The underlying paradigm for this insists that people should be able to vibe in every way with this technology, yet, the same paradigm tends to exclude pieces of technologies making people either unvibing or "too much" vibing with the related technological artifacts. For the greatest good of all: no-one should be found forced to remain isolated, hopeless and/or in dulled life, in misery.

The IBM"s cognitive chip, others like MoNETA, Blue Brain, FACETS, and ours, constitute a first milestone toward artificial conscious and therefore caring artificial beings as potential service and assistant robots. However, the main message of this chapter has been that the next milestone in the venture of materializing intelligent and therefore necessarily conscious, effective and efficient service and assistant robots should be to take every measure so that "bad seeds" are never sowed onto those cognitive chips, and that whatever is evil to our human frequencies is never deliberately taught to those robots while in maturation as it should be the same with children and teenagers.[26] Any technology can be used for good ends, but can also be turned for evil purposes. The new presented technology that depends on "cognitive chips" to be re-leased in the next few years (we already know how to build and also already have a plethora of various robotic bodies at disposal throughout the world) will be the most powerful pervasive technology ever produced due to the sheer presence of robotic bodies within people infra-structures, their sensing, acting, and last but not least, thinking activities.

We must develop, know and sustain a special ethical infrastructure for those seedings and teachings. Eventually, that infrastructure will support itself in the same way as truly responsible and reliable robots emerge from it. Dextre Robot [223] is now able to maintain and repair to significant extents the international space station (ISS) as the astronauts sleep or are busy with noble tasks. This is a spectacular prowess. However, we note that Dextre Robot is again based on existing and commercially available types of μP and μC technologies, with its performing controller conventionally pro-grammed as software. At most, it can perhaps emulate consciousness. But we may ask, how to let such a carer robot without true consciousness when so much is at stake? Physical pain/pleasure ensues from learnt immediate punishments/rewards (instantaneous actions-based belief system). Mental pain/pleasure emanates from learnt anticipated punishments/rewards (moralistic be-lief system). A cognitive robot with its cognition consisting of huge numbers of instantaneously co-activated processing units on chip may undergo such sorts of pain and pleasure humans are familiar with. The former type of pain and pleasure is related to primal instincts, whereas the latter is related to the formation of a mind. Therefore, a learnt cognitive robot endowed with a true mind which is down to Earth, notably via its connected physical sensors (and not a mere simulation or emulation of mind) is only the type of robot that should be assigned to tasks involving caring. Ultimately, robots such as Dextre will not care (will not mind) at all if something goes wrong at some point, since it does not have a mind, for example to anticipate punishment and/or deprivation of reward if it has not done everything in its means to preserve the integrity of the ISS and the life of the astronauts. Furthermore, if it does not possess a conscious mind, it will not have to strive first staying conscious and self-preserving by anticipating damages to itself. This chapter has clearly proposed cognitive robots to be conscious from the self-development on FPGA device of design seeds that have the prime task of taking care of themselves, that is, the consciousness of the cognitive chip via inner workings detailed in our previous works [161, 162, 173, 183] and in the rest of our material provided for this conference; then, taking care of their own robotic bodies and of their surroundings via learnt immediate punishments and rewards; ultimately while minding future possible happenings via learnt anticipated punishments and rewards[27].

---

[26] We appreciate that in the utmost nature of things, goodness inevitably has to co-exist with evil as any other complementary opposites (cf., Yin and Yang). However, it seems that a detrimental scheme is to have "evil" as the main central part of a set-up, in which goodness may be enjoyed at times. Such a set-up seems doomed to deceptions and ultimately to utter failure at a global scale. Yes, there cannot be goodness without evil for goodness cannot be felt and enjoyed without a reference: evil. And yes, oppositely, there cannot be utopist paradigm where only and solely goodness is in effect. There must be motivated work and/or some kind of focused efforts against resistance of the classical world before being granted chilling out and enjoying it. If not, either the body or the mind or both

We should always seek to understand ourselves and each other, and always better and better. We live in a people paradigm (paradise), driven and maintained by the mere existence of our bodies and the thought processes of our minds, and we all should respect these.

"We are spiritual beings, having a human experience," Pierre Teilhard de Chardin. But therefore, there are known other spiritual beings having a pet experience, and if the proponents of QP are right, some new spiritual beings may have a robotic experience soon with the ad-vent of cognitive chips and therefore conscious robots.

Some countries will probably reject such machines. It is diversity, which is good as well; it has always been welcomed one way or another (cf. Cybernetics). There are charities to fight against poverty, charities which are well perceived by the public. However here we are typically talking about poverty moneywise. But we must stress that this type of poverty is only a tiny aspect of the full spectrum of an individual's life. Anyone de-serves to enjoy a sufficiently high class lifestyle, not only being financially safe, but developing oneself in interesting ways, being somewhat intellectual and also having minimal social skills, with good communication (body language). This technology can really help fight low class lifestyles such as misery and bring the overall level to a collective decent class lifestyle for everyone, at the very least. It is nice for no one to be and/or to live with low class lifestyle. Charities are fighting moneywise poverty. With the help of this new technology, we shall fight poverty in its full meaning, that is, misery, so that everyone will have an interesting, intellectual, rich, colourful and high quality life as naturally deserved from their birthrights.

---

in people will turn lethargic, unhealthy, or insane, regardless of the technology in effect. In nature, goodness and evil are evenly distributed and well balanced. If we go for a scheme at all, here a technology-based scheme, this naturally balanced distribution must be respected within the corresponding set-up, such as the one which has been proposed and described in this chapter, and unlike the aforementioned "collective masochism" scheme currently ruling the world.

[27] But on the other side, if you don't care at all about the stakes mentioned in this chapter concerning conceiving, designing and building cognitive robots, they (and carer robots indeed) will "care" for you! (The way they want.) Therefore, you have to care right now about the ethics and related design approaches for the future developments to come!

We should strive to design Third Generation Robots so that any new ethics they may be tempted to develop are fully consistent with our human, pet, fauna and flora as well as Earth ecosystem's ethics. Basically, we must aim at self-locking our most fundamental ethical values to themselves. They correspond to our humanly frequencies. We must not let possibilities for new (alien) ethics to emerge and enter in conflict with ours [217]. We have the responsibility of having ethics maintaining ethics. It can and should be the same piece of ethics maintaining its own most fundamental values. We shall call it self-locked meta-ethics.

11.7 Summary

After reviewing some relevant background of robotic control methods followed by most recent advanced cognitive controllers, this chapter suggests that amongst many well-known ways to design operational technologies, the design methodologies of those leading-edge high-tech devices such as 'cognitive chips' that may well lead to intelligent machines exhibiting conscious phenomena should crucially be restricted to extremely well defined constraints. Roboticists also need those as specifications to help decide upfront on otherwise infinitely free hard-ware/software design details. In addition and most importantly, we propose these specifications as methodological guidelines tightly related to ethics and the nowadays well-identified workings of the human body and its psyche.

Chapter 12

Conclusions, Future Works and Perspectives

"First, they ignore you,
Then, they laugh at you,
Then, they fight with you,
Then, you win."
Mahatma Gandhi

12.1 Summary

The ultimate goal of the work developed, and documented in this report, was to build a system that can be "reprogrammed" online during interaction with humans, with other systems and with its environment, that we are also supposed to share. This system should actually automatically "learn" from its surrounding information. And there is no need to reprogram it as it runs by scripting in a specific language. This is needed only once, for the design modeling and implementation (coding / writing scripts) of the very first initial machine learning architecture, learning algorithms and other self-reorganizing mechanisms.

12.2 Novel FPGA Architecture

Implemented in field programmable gate array (FPGA) – a big programmable logic device (PLD) –, this initial architecture can then self-reconfigure, and the system units self-reorganize according to the current outside information flowing in and out of the system and data recorded inside it, that is, the system- internal model of the world progressively built and shaped with sensory inputs (such as infrared and ultrasound sensors for acutenesses different to but effectively and efficiently complementary to the humans' ones) through extensive exposure to the real world. Then, this model being a unique representation of the world for this particular system, it will constitute the "identity" of this unique machine's "brain". Only this particular machine will have gone through a unique course of events of its own "life". The internal model can thus be formed through interaction with the machine's environment, and can more inter- estingly be shaped and re-shaped through feedback processes of the model itself (internal inference, reasoning, and ... thinking!), which would ultimately allow the machine to "imagine" and plan (far-)future actions due to a process that I term 'model ex- trapolation'. Indeed, by comparing its past experience (stored in the form of sequences of patterns in its internal hierarchical model) about sequences of previous environmental events and about feedback observations from its own past actions (learn- ing by doing) to its currently percieved environment, the system can decide on what next action to undertake at any present real- time. Furthermore, by internally comparing objects and events of its model (including learning new ways of learning: 'meta- learning'), the system can ultimately develop theories! This brain-like system can also have an internal representation of its own body; internal representation of its own brain-like structure would be something radically different (and perhaps innovative and very fruitful to investigate) from structures found in nature, as biological brains do not know about themselves...

Thus, from an initial configuration, the architecture progresses (self-restructures) along external changes, in order to self-adapt, self-regulate, self-optimize, self-manage, self-improve, self-enhance, and possibly self-repair, self-maintain and self- replicate. Table 1.1 in Chapter 1 showed an organized list of "self-properties", or "self-features" such a system should own. On the figure, complexity increases downwards and level of ab- straction increases from right to left, but the items completelyat the left and completely at the right should be considered separately.

12.2.1 Evolutionary-inspired Electronics

Intelligence naturally developed using three different strategies during three different epochs [31, 3]. The first one is the evolutionary learning of unicellular and later multi- cellular organisms (phylogenetics, Darwinism), notably by passing learned traits to their offspring: 'DNA-learning' of unicellular creatures, but veracity of the theory for multicellular ones is currently under intensive investigations [224] (epigenetics, Lamarckism). Through this type of evolutionary phenomena, emotions and affects progressively built in next generation individuals, giving them, respectively, reflexes and intuition (decision biases) more advantageous for survival. The second one, which does not exclude the previous one but complements with it, appeared when multicellular organisms got their capital cells flexible, such as synapses and neurons. Languages, and therefore cognitive communication between individuals, appeared at this epoch. The last one, which can only occur provided the previous one, is passing knowledge (and available environmental technology) to next generations via teaching from the parents and observation of the offsprings (ontogenics, "Baldwin effect").

12.3 Methodologies

The work presented in this thesis report focuses on the second way (above) of naturally developing intelligence, and where 'machine learning' can fully apply in order to cope with radical environmental changes for example, where natural evolution (phylogenetics and epigenetics) is too slow to succeed. In this project, we consider a single phenotype that can quickly adapt during its lifetime by using conation, that is, being willing to and effectively applying knowledge to new situations for survival, but practically and ideally within our human societies, for improved life qualities and inter-mutual blossoming (also between human and their man-made intelligent creatures: machines).

However, the first way (above), through which, again, individuals adapt slowly over generations, involves genotype-phenotype mappings as well as encoding phenotype properties into chromosomes for genetic operations such as crossover and mutations. These long lasting processes prevent species to adapt to sudden and important environmental or condition changes, also observed along a thermodynamics perspective. But as illustrated by the first column of Table 5.1 in Chapter 5, these processes can be involved at different levels. From evolving phenotype populations over generations (phylogenetics, epigenetics) as nature slowly does to fruitfully confront individuals to their environments, to evolving system-internal learning mechanisms or populations of cells within a single phenotype [38, 16, 35, 46, 49, 51, 5, 50, 53, 8]. To us, this latter extreme approach does not comply with natural sophistication in harmony with our environment, but is rather a pure artificial evolutionary concept that could be put in the same bag as genetically modified organism (GMO) techniques applied to the biological world. However, the former approach, which is found in the nature, can be virtually applied in parallel and/or subsequently to this present project in order to evolve populations of our entire machines. A very interesting aspect would be to implement 'trait-learning' (epigenetics) through our machine genes (phenotype-genotype mappings) in order to corroborate or invalidate part of the Lamarckian theory. Even though this part would be invalidated, I believe that having implemented the feature would accelerate and improve the performances of the evolutionary process. Finally, emergence of cooperative behaviours between machines should permit spontaneous ap- plication of the "Baldwin effect" on individuals during their evolution. This phenomenon is well-known for its capacity to efficiently overcome radical environmental changes such as sudden technological advances. It is actually the fastest pro- cess of phenotype adaptation to their changing conditions, and can be said to be the meta-phenomenon of lifetime phenotype- learning. This meta-phenomenon can take its full effect within the human species since humans have developed the smartest and most advanced technologies in Earth. It should then obviously be the same for intelligent machines.

Much like students are often asked to read texts in order to understand the ideas contained and then are asked to explain these ideas in their turn in their own words, we went on the mission of trying to read the behaviours of intelligent creatures of the Earth including humans, of trying to examine the underlying functioning of the constituting parts of these creatures (despite brains and neural networks are considered as black boxes, it is now possible, with advanced equipment, to directly probe their inner structures, and find the analysis reports in official articles and books) as well as trying to imagine the theoretical explanations of the phenomena involved between the global behaviours and (sometimes only assumed) internal mechanisms of these creatures. Then, from what we have understood about that, we have expressed and formalised the corresponding ideas (sometimes taking the assumed - but felt sufficiently reasonable - internal mechanisms of the biologic world as guides to our approaches) into this report. We wish the implementation of these ideas as an intelligent machine provides with the same fidelity as the written essay of the student reflects the original text he had to read. Note that in practice, both the produced text of the student and the original document may differ in quality and quantity. We also propose that the nature of our produced intelligent machine strongly differs from the one of the original creatures analysed. In that perspective, intelligent machines may not be of - for example - the human nature, but to the opposite, may have a way of working that complements people along an augmented creative mutual contribution. These differences may partly come from the assumed internal functioning, and implemented as such.

Our proposed architecture design, described in hardware- oriented languages such as SystemC and Balsa, belongs to the class of “morphware” [225,226].

12.4 Beyond Material Electronics

Consciousness and mental processes are fascinating. For humans, it is safe to say that mental processes are bounded in rather narrow frames of the space-time continuum. How- ever, the accumulation of thoughts (as the ones reported in the present document), and subsequently ideas and works, does also make sense and is actually very useful (such as new available equipment produced along time-consuming accumulated work), somehow thanks to perhaps the magic of some laws of physics...

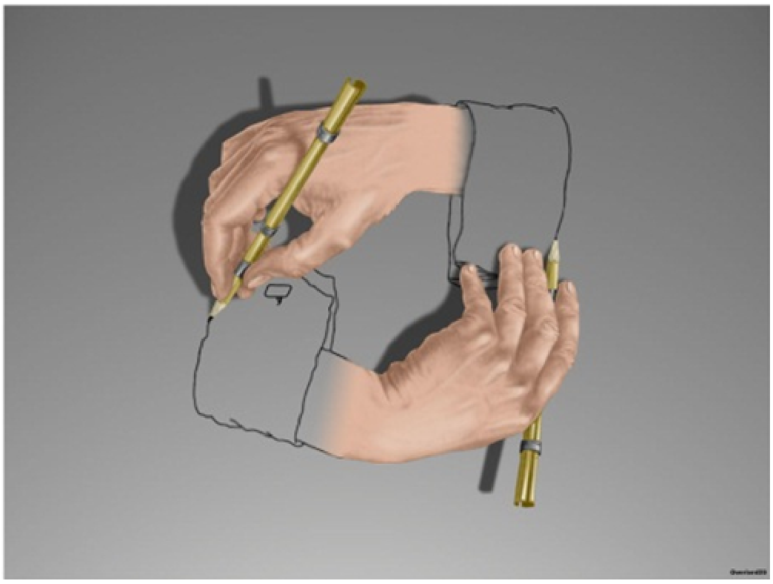


Figure 12.1: M.C. Escher, "The Drawing Hands" ©.

Figure 12.1 gives an artistic view [227] of what is understood by the term morphware [225, 226]. Whereas the practical concept of hands drawing themselves is impossible – but in a graphical computer program simulation –, it is believed to be possibly effective along real-life phenomena as special architecture design and algorithm implemented in FPGA 'flexible hardware'.

In a conventional computer, you have to organize files yourself for later easy and efficient retrieval, and the file systems have to be maintained. Hence, we wish a machine that can be programmed through friendly human interaction, but that does not necessarily think in the human manner internally. A machine that is naturally sophisticated, without too much handcrafted artefacts for its initial design (coding/ implementation). Thus, we rather need to design an 'intelligence seed', along a hard- ware 'processing memory' perspective.

Self-adaptive machine learning systems is a radically different approach than artificial intelligence (AI). AI aims to mimic (simulate/emulate) human intelligence and behaviours. AI systems are usually developed along top-down methodologies: writing software systems on CPU-based computers. CPUs (microprocessors) already have their own artificial sophistication. A microprocessor-based machine is immersed into the digital world of its own, thus separated from our real-life environment, in which we wish machines to wander and act cleverly.

12.4.1 About the Turing Test

Computing machines excel in do- mains humans will never be able to. Yet, the 'Turing test' forces these machines to underperform in some aspects, mimicking human response delays for example. This highlights that some clumsy approaches have been undertaken in early AI works. Also illustrating the aim of AI as to build machines supposed to be able to do what humans can also do. And if we want to build intelligent machines to be able to both have straight- forward external communications with humans, and internally process information the way humans cannot do, hereby complementing us, we simply have to be especially careful on our approaches; eventually amend them and develop again (while doing re-use) after evaluation of initial experimental results.

Control Engineering vs. Biological Science views:

As depicted in Figure 12.2, control engineering systematically considers a separate system-process model. However, biological science faithfully considers phenomena.

12.4.2 Support

"One of the most important and often neglected particularities of living systems, however, is the in- timate connection of the body and the control unit, which in a strict sense cannot be

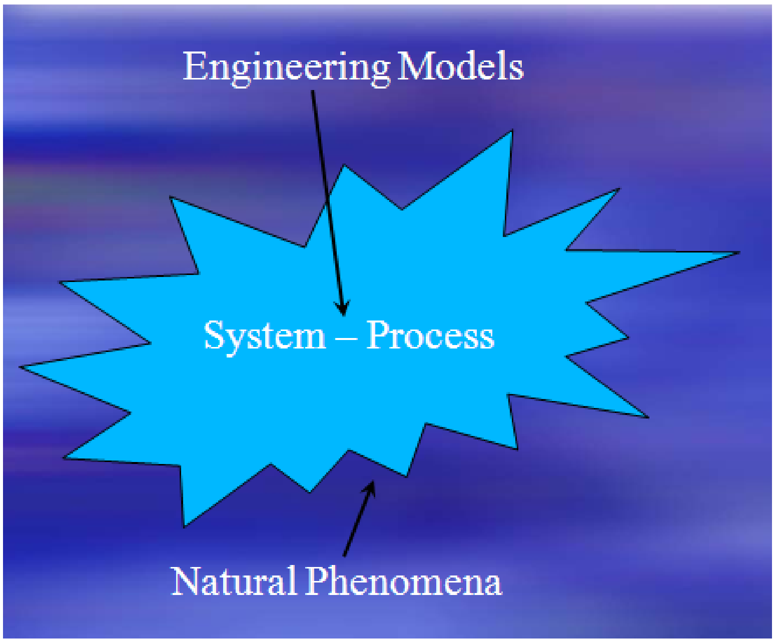


Figure 12.2: Engineering Models of Natural Phenomena: **System vs. Organism**.

seen separately. This is the special design of the physical equipment defining and shaping the "intelligent" units of control. In classical AI, both are typically considerate separate, thus representing some type of pre-scientific dualism," [228].

12.4.3 Past work summary

1. Thoroughly reviewed literature about the interdisciplinary machine learning domain as well as related areas. We particularly focused on every piece of background where machine learning techniques are implemented in hardware.

2. Identified key and fundamental questions for this PhD work.

3. Found related work and research gaps to fill up.

4. Created a research niche, while ensuring not "reinventing the wheel".

5. Defined research tracks towards novel and promising ‘machine learning’ techniques.

6. Designed a generic system unit (cell), to be the basic building block of the overall hierarchical machine learning structure.

7. Spotted out that neural network learning algorithms do not exist for structures having more than about 100 neurons. Even along evolutionary techniques that attempt to automatically tune a learning algorithm. From this observation, we decided to develop an approach of hierarchical and modular neural network structure, where a module can be considered yet again as a neuron in a higher hierarchical level, and then bundles of modules considered as a new module in a higher hierarchical level, and so forth. In the same line, we believe that scaling these hierarchical and modular substructures according to fractal principles will allow us to write a single universal learn- ing and predictive algorithm that will work at any level of hierarchy, and will be implanted in any module along with a distributed fashion.

8. Described an innovative hardware ‘machine learning’ system architecture.

9. Started to develop algorithms to make this system “alive”.

10. Adopted a bottom-up approach for the emergence of NN structures using CA and self-reorganizing mechanisms, but undertaking the conventional top-down methodology for the FPGA implementation in SystemC.

11. Defined our dual system as being a self-reorganizing neural network that can be viewed as a cellular automata during reconfiguration.

12. Tried to imagine the consequences of this work at every level: from the project itself, to the impact on our human societies, through the influence on the projects of other people conducting similar works.

13. Tried to imagine the global system behaviours emerging from the mentally conceived and manually (with the help of electronic design automation (EDA) tools) implemented local cell mechanisms.

14. Called into question many hastily granted convictions and engineering models. We did this by enlarging the, other- wise common, envision of such models. The outcome guided us towards novel approaches that deserve to be investigated. In turn, our adopted approaches helped to fit this engineering project in a nice framework, that we also defined, and that we feel now comfortable and happy with.

15. Provided a comprehensive list of justifications for this work and for the framework it fits in, in order to get a well shell-rounded project and to be confident in our re- search directions.

12.4.4 Ongoing work

1. FPGA implementation in SystemC of a NN system with backpropagation BP learning algorithm: modelling of logic XOR gate function (requires three layers of neurons) as an

evaluation application.

2. Making a single NN modelling different types of logic gates through reconfiguration.

3. BP -> Kak (ITNN: Instantaneously Trained NN in hardware).

4. Spiking asynchronous neurons (with BP and Kak learning algorithms).

5. Asynchronous operation of the CA using the Balsa language, and trying to make it generate a functional NN, by using constraints and self-reorganizing techniques: initial CA-NN system.

6. Modeling finite state machines (FSMs), for the generic cell mechanisms, using Petri-nets.

7. Training the CA-NN against mobile robot data sets with the envision of smart navigation applications.

8. I have the feeling that I(CA-NN) > I(NN) and that the learning power and ability of our CA-NN is superior to the one of a conventional NN, but I am quantifying the figure using information theory techniques [118,147,229,230].

12.4.5 Future Work

1. Hardware implementation of NN with self-organizing learning algorithm and reinforcement learning (Hebbian&Kohonen-inspired).

2. Making the NN learn the rules of its own underlying CA (meta-learning), and change them / add new ones improvingly: need of **'improvability evaluation'**

   → CA meta-rules are not pre-programmed phases anymore only, as in conventional CA, but can also be created / discovered by the NN at run-time, and are explored as new learning strategies: by upgrading the NN learn- ing algorithm, the system changes its way of updating the weights, much like it is often useful to change our way of thinking concerning some issues in order to improve our performances.

3. Partial reconfiguration of the FPGA to take advantage of its reconfigurability features: more powerful control on the NN (and on the CA).

4. Dynamic reconfiguration of the FPGA for more powerful online learning: self-specialisation of NN modules at run-time.

5. Investigate the suitability of this intelligent iterative learn- ing CA-NN system for 'micro-architecture space pruning' as suggested by Dr. Vassilios A. Chouliaras. Indeed, iterative learning processes, such as the ones employed by neural networks, lend themselves nicely to parameter space pruning applications.

6. Self-management of internal (system) and external (evironment) resources: self-maintained system. The envision for this piece of work is to consider the system as an interface between our shared world and its inside world. With the help of its input-output sensors and actuators, it will be supposed to manage its immediate environmental resources as well as its own internal system resources in order to optimize the working conditions of both the inner and outer system worlds. To achieve this, there is a prime need of trial-and-error processes as well as a 'resource management planner' as a controller layer that should sit on the primitive processes layer, both layers being implemented into the system. Also, the machine should be "aware" of its functioning conditions as well as the environmental conditions so that it tries to adapt to, ideally any, sudden or smooth changes in the environmental conditions. By learning from its inputs and outputs, the system should self-specialize.

   Through this research work, we are gaining technical know-how on achieving 'dynamic partial self-reconfiguration' on FPGA: [231] (the "Partial Reconfiguration on Xilinx Devices" mailing-list archive), this, for effectively and efficiently implementing our machine learning techniques, while developing a new particular area of research.

12.4.6 Perspectives

This project can be derived to many related projects, but it is hoped that it fits as the early steps of the more ambitious and extensive project dealt with in this report, of which the concepts are described and the ideals given.

1. The achievement of this early step should be: designing a SoC solution in an application specific integrated circuit (ASIC) (including an embedded FPGA for the reconfigurable portion of the system) of asynchronous self-organizing CA-NN with spiking neurons for self- adaptive, genuine real-time and cognitive control of autonomous mobile robots.

2. Continue this work by adding a fuzzy layer to the two first NN and CA layers in order to enhance the decision making power of our system, thus renamed ‘CA-based neuro-fuzzy’ system.

3. Related project proposal 1: evolving CA-NN’s modules using genetic algorithms (GA) or other evolutionary strategies.

4. Related project proposal 2: consider FPGAs CLBs as trainable network cells (much like neural synapses and axons for example) → autonomous ‘inductive logic’ programming in hardware.

5. Related project proposal 3: make our system able to ‘understand’ and write SystemC descriptions, synthesized as a replacement of the FPGA configuration of the system itself at run-time, this along a cycle loop through the pre- established EDA tools (for the synthesis step) that can be run onchip by a microprocessor integrated in the FPGA sitting next to our system thus acting as a coprocessor, and communicating through an AMBA bus. The project will also eventually try to exploit the internal configuration access port (ICAP) of the FPGA.

6. Making a poll about ethical issues of “artificial life” development should also be led in the meantime our system develops.

The work presented in the page of this web-link [232] is similar to the one of my friend and former classmate, and then former associate Laurent Rougé: [172]

But according to what I came across during my literature review process: [233], doing reconfiguration into ASIC or SoC is pretty tough (the adjective “tough” can be found on this page of online encyclopedia, the corresponding full sentence about the ‘reconfigurable computing’ item clearly explains how difficult it is to achieve). I can understand how tedious it can be to, for example, play with changing the functioning nature of the ALU now and then (as I imagine and feel from what Laurent had told me when he switched to this ‘eFPGA’ project) of a microprocessor during run-time: getting confused with instruction management shall be part of the enjoyment...

Maybe, once all these design and implementation efforts have been achieved, the whole system is easily programmable as Laurent claims, but I am not convinced of this.

All that is one of the reasons why I have rather been trying to choose (actually create and invent in-house) a novel initial architecture that both lends itself to dynamic self-reconfiguration at run-time – allowing for on-line learning –, and that is most possibly implementable into commercially off-the-shelf FPGAs (at least for cheap yet effective and efficient experimental purposes) that allow for dynamic partial reconfiguration. Of course, this architecture I ended-up with is the result of much more rationale elements put together such as functional building blocks: neural cells.

Also, because microelectronics and automation is the speciality of my general engineering background (networks, advanced mechanics, robotics, materials, general physics and physic of electronic components, signal and image processing, mathematics including multilinear algebra, series, Fourrier transforms, probabilities and statistics, and strong background in information theory, etc; a wide range of fascinating subjects in fact), it seems completely natural to work with PLDs/FPGAs, and later on maybe with ASICs as I have been encouraged to do. But I believe that these sorts of architecture can also be easily and straightaway implemented using other technologies such as nanotechnologies, DNA-based computing, or other “organic computing” substrates.

12.4.7 CellMatrix

In the same line, Nicholas Macias of [14] has made available a sort of FPGA overclass: the 'CellMatrix', and then claimed that he was willing to implement a similar solution using nanotechnologies later on. His CellMatrix is a hardware design that can process and produce a new (improved) design as a replacement, a bit like LISP language allows for ob- taining a new LISP code straight away from the compilation of an antecedent LISP code. LISP was invented in the objective that somehow, a form of intelligence self-emerges from an initial LISP code, a sort of software seed. Unfortunately, this never happened. Perhaps N. Macias will be more lucky as his Cell- Matrix system is real physical hardware shared by our physical world without the need of mapping parallel processes, in con- trast to LISP scripts, which are virtual software programmes enclosed in the digital dimension of the microprocessor world, where the code is serially (in time of our world) compiled and executed through the instruction pipeline.

One thing I wanted to see during the course of my PhD, is that I can demonstrate that a machine exists, such that it evidently reflects what it has experienced; the course of events that have happened to it. That it has got a representation of the subsequent environments it has been immersed into, hereby defining its "identity", and ultimately allowing it to take initiatives and have intentions; all this through learning, meta-learning, and the cognitive side-effects of learning processes. Intentions and motivation should be initiated by information theory-based internal systematic mechanisms activated when there is a need to learn, to self-reorganize the architecture structure, and/or to perform tasks. These needs will be spotted out and evaluated by local and global measures and comparisons of information elements such as entropy of system units and environmental details.

I believe that as soon as we build a machine, of which the main concern is to solve problems, primarily for its own survival, and then for improving its life quality including cultivating good relations with humans and nature, it will come to the sort of question: "what am I made of?" At this point, it will be able to have free will. The problems, other than maintaining own survival, should then be seeking to life improvements, technology advances and ease of interaction, communication and ‘intuitive learning for humans’; especially speeding up learning how to use technological tools and other equipments as well as facilitating learning useful exact science arts such as mathematics and engineering, for more natural, thus more efficient and creative use, and for accelerated general society progress. Indeed, all this technology is known to be anti-physiological for people to grasp. There are ways to make technology ergonomic and intuitive. An overall guideline should be mutual blossoming within people and robots.

I am convinced that an artificial creature, of which technological elements that will have been naturally self-assembled and self-organized, will have the best position to teach us how to efficiently approach technology. A machine conceived to solve problems will inevitably have to plan projects and address their underlying problems, much like people, also evolved as cogs of the world in order to solve problems, have to do the same. Our fate, together with eventual intelligent machines or problem- solving agents, is technology and society advances. It is a fate because our brains are problem-solvers as well as the intelligent machines that have been wished to be built for hundreds of years. It is a fate because it is well proven that someone who fails addressing problems, will feel useless, and this is psycho- logically in contradiction with survival. Thus, the same per- son at this point will (in most of cases and for most of people) spontaneously set up one or several projects, and tackle them, thereby contributing to technology and/or society advances, or any other form of advances.

The inner “life” of the machine should be strongly related to the outer real life we all share. To achieve that (one day), it seems obvious to me that I need to particularily consider and start with massively parallel, highly distributed systems, where physical indeterminism and uncertainty is part of the game, and can actually contribute to the right functioning of the whole system. Therefore, communicating sequential processes (CSP) under- standing is of prime importance, but fortunately well leveraged when using SystemC hardware-oriented language.

12.5 Final Statement

I refused to blindly focus on engineering implementations without preliminarily thinking deeply about what should be implemented, and especially why regarding many aspects of our rapidly progressing technological societies and life conditions. Only after this thorough investigation, shell-rounding our motivation at the same time, shall we ask how to implement our system and what methodologies should be developed and used. All these important points were dealt with and reported in this document. Finally, we will be delighted to answer any questions regarding this interdisciplinary project at the frontiers of research.

Bibliography

[1] M. Negnevitsky, *Artificial Intelligence: A Guide to Intelligent Systems*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001.

[2] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997. [Online]. Available: http://www.cs.cmu.edu/~tom/mlbook.html

[3] A. Cardon, *Modeling and Conceiving a Thinking Machine: approach to the artificial consciousness (Modeliser et Concevoir une Machine Pensante : approche de la conscience artificielle)*. Vuibert, Apr.13, 2004. [Online]. Available: http://www.automatesintelligents.com/collection/cardon1.html

[4] J. Hawking, *On Intelligence*. Times Books, 2004.

[5] R. Kurzweil, *The Age Of Intelligent Machines*, 1990. [Online]. Available: http://en.wikipedia.org/wiki/Raymond_Kurzweil

[6] S. Wolfram, *A new kind of science*. Champaign, Ilinois, US, United States: Wolfram Media Inc., 2002.

[7] D. G. H. Holling and P. H. de Garis, “On the application of advances in artificial intelligence technologies for the design of autonomous intelligent robots,” in *Proc. IEEE EIT 2003, Electro/Information Technology Conference*, Indianapolis, IN, USA, Jun. 5–6, 2003. [Online]. Available: http://www.cs.usu.edu/~degaris/papers/EIT-2003.pdf

[8] H. de Garis and M. Korkin, “THE CAM-BRAIN MACHINE (CBM) - an FPGA-based hardware tool that evolves a 1000 neuron-net circuit module in seconds and updates a 75 million neuron artificial brain for real-time robot control.”

[9] D. Floreano and F. Mondada, “Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot,” 1994. [Online]. Available: http://citeseer.ifi.unizh.ch/floreano94automatic.html

[10] A. R. Damasio, “How the brain creates the mind,” *SCIENTIFIC AMERICAN-AMERICAN EDITION-*, vol. 281, pp. 112–117, 1999. [Online]. Available: http://r-research.org/wp-content/uploads/2012/11/1383263.pdf

[11] “The intelligent systems group.” [Online]. Available: http://www.elec.york.ac.uk/intsys/inspired/inspired.html

[12] H.-N. Teodorescu and A. Kandel, *Hardware implementation of intelligent systems*. Physica, 2013, vol. 74. [Online]. Available: http://portal.acm.org/citation.cfm?id=570780.570783&coll=GUIDE&dl=GUIDE&CFID=43449562&CFTOKEN=6219867

[13] J. B. Pollack and H. Lipson, “The golem project: Evolving hardware bodies and brains,” in *Evolvable Hardware, 2000. Proceedings. The Second NASA/DoD Workshop on*. IEEE, 2000, pp. 37–42. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/EH.2000.869340

[14] N. Macias and L. Durbeck, “Cell matrix.” [Online]. Available: http://www.cellmatrix.com

[15] K. Gill, *Human Machine Symbiosis: The Foundations of Human-centered Systems Design*. Blackwells Publishers Ltd. Oxford, 1996.

[16] S. Darmos, *QUANTUM GRAVITY and the ROLE of CONSCIOUSNESS in PHYSICS*, 2016.

[17] Billaut. [Online]. Available: http://billaut.typepad.com/jm/2006/01/quest_que_cela_.html

[18] B. H. de Garis, "news/zurich." [Online]. Available: http://www.cs.usu.edu/~degaris/news/zurich

[19] [Online]. Available: http://www.cs.usu.edu/~degaris/about/resumeHTML.html

[20] B. Bly and D. Rumelhart, *Cognitive science*. Academic Press, 1999.

[21] [Online]. Available: http://www.optimal.org/peter/rational_ethics.htm

[22] K. Kaufman and R. Michalski, "Reports of the machine learning and inference laboratory: Initial considerations toward knowledge mining," School of Computational Sciences, George Mason University, Tech. Rep. MLI 04-4, Oct. 2004.

[23] "papers and presentations in bulk." [Online]. Available: http://metalab.uniten.edu.my/~alicia/CMPB454-AT/SUPPLEMENTARY%20NOTES/Part%203

[24] [Online]. Available: http://metalab.uniten.edu.my/~alicia

[25] [Online]. Available: http://www.faqs.org/faqs/ai-faq

[26] [Online]. Available: http://www.faqs.org/faqs/ai-faq/neural-nets/part1/

[27] "Neural information processing systems foundation." [Online]. Available: http://books.nips.cc

[28] D. Rutkowska, "Neuro-fuzzy architectures and hybrid learning," Technical Univ. of Czestochowa, Czestochowa, Poland, 2002. [Online]. Available: http://portal.acm.org/citation.cfm?id=SERIES10698.507126&coll=GUIDE&dl=GUIDE&CFID=43449562&CFTOKEN=6219867

[29] F. A and M. S. Luk W, “Scalable acceleration of inductive logic programs,” in *Proc. IEEE international conference on field-programmable technology (FPT 2002)*, Chinese Univ Hong Kong, New Territories, Peoples Republic of China, Jan. 2002, pp. 252–259. [Online]. Available: http://www.celoxica.com/techlib/files/CEL-W0307171J6C-26.pdf

[30] A. J. Sharkey, *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*, A. J. Sharkey, Ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1999. [Online]. Available: http://portal.acm.org/citation.cfm?id=553353#

[31] W. Banzhaf and C. Lasarczyk, “A new programming paradigm inspired by algorithmic chemistries,” Sep. 2004. [Online]. Available: http://upp.lami.univ-evry.fr/Documents/Banzhaf.ppt

[32] M. Sipper, “The emergence of cellular computing,” *Computer*, vol. 32, no. 7, pp. 18–26, 1999. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/2.774914

[33] H.-Y. Kim, J. Park, and S.-W. Lee, “A new methodology to the design of associative memories based on cellular neural networks,” *ICPR, 2000*, vol. 02, pp. 29–65, 2000. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ICPR.2000.906235

[34] C. Wu, L. Chua, and T. Roska, “A two-layer radon transform cellular neural network,” *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on [see also Circuits and Systems II: Express Briefs, IEEE Transactions on]*, vol. 39, no. 7, pp. 488–489, Jul. 1992. [Online]. Available: http://ieeexplore.ieee.org/iel4/82/4198/00160175.pdf?isnumber=4198&prod=STD&arnumber=160175&arnumber=160175&arSt=488&ared=489&arAuthor=Wu2C+C.W.3B+Chua2C+L.O.3B+Roska2C+T.

[35] L. Chua and L. Yang, “Cellular neural networks: Theory and applications,” *IEEE Transactions on Circuits and Systems*, vol. 35, no. 10, pp. 1257–1290, October 1988.

[36] L. O. Chua and L. Yang, “Cellular neural networks,” in *ISCAS’88*, 1988.

[37] “Cellular neural networks.” [Online]. Available: http://www.ce.unipr.it/pardis/CNN/cnn.html

[38] “Fuzzy - nn - ga.” [Online]. Available: http://if.kaist.ac.kr/lecture/cs670/lecture-note/Fusion.ppt

[39] “Physica-verlag studies in fuzziness and soft computing series: table of contents.” [Online]. Available: http://portal.acm.org/toc.cfm?id=SERIES10698&type=series&coll=GUIDE&dl=GUIDE&CFID=43449562&CFTOKEN=6219867

[40] [Online]. Available: http://lslwww.epfl.ch/pages/embryonics/home.html

[41] J. M. Moreno, Y. Thoma, E. Sanchez, O. Torres, and G. Tempesti, “Hardware realization of a bio-inspired POEtic tissue,” *eh*, vol. 00, p. 237, 2004. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/EH.2004.1310836

[42] M. Sipper, E. Sanchez, D. Mange, M. Tomassini, A. Perez-Uribe, and A. Stauffer, “A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems,” *IEEE Transactions on Evolutionary Computation, 1997*, vol. 1, no. 1, pp. 83–97, April 1997. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/4235.585894

[43] P. P. Kanjilal, “Adaptive prediction and predictive control,” Department of Electronics and ECE, I.I.T. Kharagpur, 721-302, India. [Online]. Available: http://www.iee.org/Publish/Books/Control/Ce052p.cfm

[44] T. Higuchi, M. Iwata, I. Kajitani, H. Yamada, B. Manderick, Y. Hirao, M. Murakawa, S. Yoshizawa, and T. Furuya, “Evolvable hardware with genetic learning,” Electrotech. Lab., Tokyo, Japan. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ISCAS.1996.541893

[45] W.-M. Shen, P. Will, A. Galstyan, and C.-M. Chuong, "Hormone-inspired self-organization and distributed control of robotic swarms," *Autonomous Robots*, vol. 17, no. 1, pp. 93–105, Jul. 2004.

[46] P. C. Haddow and G. Tufte, "Evolving a robot controller in hardware." [Online]. Available: http://citeseer.ist.psu.edu/432713.html

[47] O. C.K. and B. G.J., "Autonomous controller design for unmanned aerial vehicles using multi-objective genetic programming," in *Congress on Evolutionary Computation*, Portland, 2004. [Online]. Available: http://citeseer.ist.psu.edu/oh04autonomous.html

[48] H.-S. Seok, K.-J. Lee, J.-G. Joung, and B.-T. Zhang, "An on-line learning method for object-locating robots using genetic programming on evolvable hardware," 2000. [Online]. Available: http://citeseer.ist.psu.edu/456254.html

[49] G. J. Barlow, C. K. Oh, , and E. Grant, "Incremental evolution of autonomous controllers for unmanned aerial vehicles using multi-objective genetic programming," in *The 2004 Genetic and Evolutionary Computation Conference*, 2004. [Online]. Available: http://citeseer.ist.psu.edu/barlow04incremental.html

[50] A. L. Nelson, E. Grant, G. Barlow, , and M. White, "Evolution of complex autonomous robot behaviours using competitive fitness," in *IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, Boston, MA, Sep. 2003. [Online]. Available: http://citeseer.ist.psu.edu/nelson03evolution.html

[51] "Maze exploration behaviours using an integrated evolutionary robotics," pp. 159–173. [Online]. Available: http://citeseer.ist.psu.edu/659174.html

[52] TAROS, "Towards autonomous robotic systems incorporating the autumn biro-net symposium," Imperial College, London, UK, Sep. 2005. [Online]. Available: http://www.taros.org.uk

[53] BIRO-net, “Biologically inspired robotics network.” [Online]. Available: http://biro-net.aber.ac.uk/index.php

[54] “Software hardware applications implementations algorithms,” 10381 (4.814/3.084/95) FCCM95 IEEE. [Online]. Available: http://www.ece.cmu.edu/~herman/publications/fccm95.pdf

[55] W. H. Hsu and S. M., “Gustafson. genetic programming and multi-agent layered learning by reinforcements,” New York, Jul. 2002. [Online]. Available: http://citeseer.ist.psu.edu/hsu02genetic.html

[56] M. Russo and L. Caponetto, “Hardware implementation of intelligent systems,” pp. 91–120, 2001.

[57] “Hardware implementation of intelligent systems.” [Online]. Available: http://portal.acm.org/citation.cfm?id=SERIES10698.570780&coll=GUIDE&dl=GUIDE&CFID=43449562&CFTOKEN=6219867

[58] J. Zhu and G. Milne, “Implementing kak neural networks on a reconfigurable computing platform.”

[59] J. G. Eldredge and B. L. Hutchings, “Rrann: a hardware implementation of the backpropagation algorithm using reconfigurable fpgas,” in *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, vol. 4. IEEE, 1994, pp. 2097–2102.

[60] J. D. Hadley and B. L. Hutchings, “Design methodologies for partially reconfigured systems,” in *FPGAs for Custom Computing Machines, 1995. Proceedings. IEEE Symposium on*. IEEE, 1995, pp. 78–84.

[61] P. Lysaght, J. Stockwood, J. Law, and D. Girma, “Artificial neural network implementation on a fine-grained fpga,” in *Field-Programmable Logic Architectures, Synthesis and Applications*. Springer, 1994, pp. 421–431.

[62] M. Gschwind, V. Salapura, and O. Maischberger, “Space efficient neural net implementation,” in *Proc. of the Second ACM Workshop on Field-Programmable Gate Arrays*. Citeseer, 1994.

[63] P. Raina, “Comparison of learning and generalization capabilities of the kak and the backpropagation algorithms,” *Information Sciences*, vol. 81, no. 3, pp. 261–274, 1994.

[64] M. Negnevitsky, “Artificial intelligence, a guide to intelligent systems pp. 200-217.”

[65] G. F. Luger, “Artificial intelligence, structures and strategies for complex problem solving.”

[66] ——, “Artificial intelligence, structures and strategies for complex problem solving.”

[67] ——, “Artificial intelligence, structures and strategies for complex problem solving.”

[68] “4.2.2 unsupervised learning.” [Online]. Available: http://www.dacs.dtic.mil/techs/neural/neural5.html

[69] J. Starzyk and T.-H. Liu, “Design of a self-organizing learning array system,” in *Circuits and Systems, 2003. ISCAS ’03. Proceedings of the 2003 International Symposium on , vol.5, no.pp. V-801- V-804 vol.5, 25-28 May 2003*, vol. 5, May25–28, 2003, pp. 801–804. [Online]. Available: http://ieeexplore.ieee.org/iel5/8570/27139/01206434.pdf?isnumber=27139&prod=STD&arnumber=1206434&arnumber=1206434&arSt=+V-801&ared=+V-804+vol.5&arAuthor=Starzyk2C+J.3B+Tsun-Ho+Liu

[70] J. A. Starzyk, Y. Guo, and Z. Zhu, “Dynamically reconfigurable neuron architecture for the implementation of self-organizing learning array,” *IPDPS*, vol. 04, p. 143a, 2004. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/IPDPS.2004.1303122

[71] J. Starzyk, Z. Zhu, and T.-H. Liu, “Self-organizing learning array,” *Neural Networks, IEEE Transactions on*, vol. 16, no. 2, pp. 355–363, March 2005. [Online]. Available: http://doi.acm.org/10.1109/TNN.2004.842362

[72] H. He and J. Starzyk, “A self-organizing learning array system for power quality classification based on wavelet transform,” *Power Delivery, IEEE Transactions on*, vol. 21, no. 1, pp. 286–295, January 2006. [Online]. Available: http://doi.acm.org/10.1109/TPWRD.2005.852392

[73] E. Chicca, D. Badoni, V. Dante, M. DAndreagiovanni, G. Salina, L. Carota, S. Fusi, and P. D. Giudice, “A vlsi recurrent network of integrate-and-fire neurons connected by plastic synapses with long-term memory.” [Online]. Available: http://citeseer.ist.psu.edu/644480.html,http://doi.ieeecomputersociety.org/10.1109/TNN.2003.816367

[74] M. I. Chris Diorio, D. Hsu, and M. Figueroa, “Adaptive cmos: From biological inspiration to - systems-on-a-chip,” 2002. [Online]. Available: http://citeseer.ist.psu.edu/629752.html

[75] G. Bo, D. Caviglia, and M. Valle, “An on-chip learning neural network.” [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/IJCNN.2000.860751

[76] P. Moerland and E. Fiesler, “Hardware-friendly learning algorithms for neural networks: An overview.” [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MNNFS.1996.493781

[77] V. Beiu and J. G. Taylor, “Direct synthesis of neural networks.” [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MNNFS.1996.493800

[78] A. G. Turner, “Cellular automata with special reference to their implementation using associative memories,” 1997. [Online]. Available: http://www.cmpe.boun.edu.tr/courses/cmpe588/common/papers/

Turner1997-Cellular-automata-with-special20reference20to20their20implementation20using20asso
pdf

[79] K.-J. Kim and S.-B. Cho, "Evolved neural networks based on cellular automata for sensory-motor controller," *Neurocomputing*, Jul.15, 2006. [Online]. Available: http://doi.acm.org/10.1016/j.neucom.2005.07.013

[80] M. Leong, P. Vasconcelos, J. R. Fernandes, and L. Sousa, "A programmable cellular neural network circuit," in *SBCCI '04: Proceedings of the 17th symposium on Integrated circuits and system design*. New York, NY, USA: ACM Press, 2004, pp. 186–191. [Online]. Available: http://doi.acm.org/10.1145/1016568.1016620

[81] O. Dekhtyarenko, A. Reznik, and A. Sitchov, "Associative cellular neural networks with adaptive architecture," in *Proc. of The 8th IEEE International Biannual Workshop on Cellular Neural Networks and their Application (CNNA'04)*, Jul.22–24, 2004, pp. 219–224. [Online]. Available: http://synapse.vit.iit.nrc.ca/memory/pinn/Associative_Neural_Networks_Library.files/Dekhtyarenko2004.pdf

[82] A. Rodriguez-Vazquez *et al.*, "ACE16k: The third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs," *IEEE Transactions on Circuits and Systems*, vol. 51, no. 5, pp. 851–863, May 2004. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TCSI.2004.827621

[83] J. M. Cruz and L. O. Chua, "A 16 x 16 cellular neural network universalchip," *Analog Integr. Circuits Signal Process.*, vol. 15, no. 3, pp. 227–237, 1998. [Online]. Available: http://dx.doi.org/10.1023/A:1008278225960

[84] D. Amanatidis, "FPGA implementation of cellular neural networks: An initial study." [Online]. Available: http://www.celoxica.com/techlib/files/CEL-W0307171HP7-12.pdf

[85] C. Darwin, "On the origins of species by means of natural selection," *London: Murray*, p. 247, 1859.

[86] I. Michael Korkin, Genobyte, I. Gary Fehr, Genobyte, and R. R. C. Gregory Jeffery, "Evolving hardware on a large scale." [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/EH.2000.869355

[87] T. Higuchi, T. Niwa, T. Tanaka, H. Iba, H. Garis, and T. Furuya, "ªevolvable hardware with genetic learning," *º Proc. Simulation of Adaptive Behavior*, 1992.

[88] A. Thompson, I. Harvey, and P. Husbands, "The natural way to evolve hardware," in *Circuits and Systems, 1996. ISCAS'96., Connecting the World., 1996 IEEE International Symposium on*, vol. 4. IEEE, 1996, pp. 37–40.

[89] H. de Garis, N. Nawa, F. Gers, M. Korkin, and A. Agah, "âŁœcam-brainâŁž atr's billion neuron artificial brain project: A three-year progress report," *Artificial Life and Robotics*, vol. 2, no. 2, pp. 56–61, 1998.

[90] P. K. Lehre and P. C. Haddow, "Accessibility between neutral networks in indirect genotype-phenotype mappings," in *2005 IEEE Congress on Evolutionary Computation, IEEE CEC 2005, Sep 2-5 2005*, vol. 1, Crab Lab., Department of Computer and Information Science, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway. Edinburgh, Scotland, United Kingdom: Institute of Electrical and Electronics Engineers Computer Society, Piscataway, NJ 08855-1331, United States, 2005, pp. 419–426, compilation and indexing terms, Copyright 2005 Elsevier Engineering Information, Inc.

[91] J. Becker, M. Platzner, and S. Vernalde, Eds., *Field Programmable Logic and Application, 14th International Conference , FPL 2004, Leuven, Belgium, August 30-September 1, 2004, Proceedings*, ser. Lecture Notes in Computer Science, vol. 3203. Springer, 2004.

[92] J. Torresen, "An evolvable hardware tutorial," in *FPL*, 2004, pp. 821–830. [Online]. Available: http://springerlink.metapress.com/opendoi.asp?genre=article&issn=0302-9743&volume=3203&spage=821

[93] G. Tufte and P. C. Haddow, “Biologically-inspired: A rule-based self-reconfiguration of a virtex chip.” in *Proc. of International Conference on Computational Science 2004*, ser. Lecture Notes in Computer Science, M. Bubak, G. D. van Albada, and P. M. A. S. et al., Eds., vol. 3038, May 2004, pp. 1249–1256.

[94] H. de Garis and T. Batty, “The evolution of robust, reversible, nano-scale, femto-second-switching circuits,” *eh*, vol. 00, p. 291, 2004. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/EH.2004.1310843

[95] P. C. Haddow, G. Tufte, and P. Remortel, *On Growth, Form and Computers*. London: Elsevier Limited, Oxford, England, 2003, ch. Chapter 22: Evolvable Hardware: Pumping Life into Dead Silicon, pp. 405–423.

[96] H. de Garis, L. de Penning, A. Buller, and D. Decesare, “Early experiments on the CAM-Brain Machine (CBM),” *eh*, vol. 00, p. 0211, 2001. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/EH.2001.937964

[97] P. C. Haddow and G. Tufte, “Bridging the genotype-phenotype mapping for digital FPGAs,” *eh*, vol. 00, p. 0109, 2001. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/EH.2001.937952

[98] N. Macias, “The PIG paradigm: The design and use of a massively parallel fine grained self-reconfigurable infinitely scalable architecture,” in *The First NASA/DoD Workshop on Evolvable Hardware*, A. Stoica, J. Lohn, and D. Keymeulen, Eds. Pasadena, California: IEEE Computer Society, 19-21 1999, pp. 175–180. [Online]. Available: http://citeseer.ist.psu.edu/macias99pig.html

[99] J. R. Koza, F. H. Bennett III, J. L. Hutchings, S. L. Bade, M. A. Keane, and D. Andre, “Evolving computer programs using rapidly reconfigurable FPGAs and genetic programming,” in *FPGA’98 Sixth International Symposium on Field Programmable*

Gate Arrays, J. Cong, Ed. Doubletree Hotel, Monterey, California, USA: ACM Press, 1998, pp. 209–219. [Online]. Available: http://citeseer.ist.psu.edu/koza98evolving.html

[100] P. Haddow and P. van Remortel, “From here to there : Future robust ehw technologies for large digital designs.” [Online]. Available: http://citeseer.ist.psu.edu/haddow01from.html

[101] M. Hartmann and P. Haddow, “Evolving fault tolerance on an unreliable technology platform.” [Online]. Available: http://www.elec.york.ac.uk/intsys/users/jfm7/gecco2002.pdf

[102] X. Yao and T. Higuchi, “Promises and challenges of evolvable hardware.” [Online]. Available: http://www.cs.bham.ac.uk/~xin/papers/published_tsmcC_feb99.pdf

[103] R. S. Michalski, “Invited presentation on non-darwinian evolutionary computation: Guiding evolution by machine learning.” [Online]. Available: http://www.mli.gmu.edu/michalski/cmu-lem-2-11-04.pdf

[104] H. de Garis, A. Tyrrell, P. Haddow, N. Macias, L. Durbeck, J. Koza, A. Upegui, C. A. Peña-Reyes, E. Sánchez, L. Sekanina *et al.*, “âŁœthe xilinx clubâŁž a world survey of evolvable hardware research projects using xilinxâŁ™s programmable fpga chips.”

[105] M. Portmann, S. Ruping, and U. Ruckert, “Som hardware with acceleration module for graphical representation of the learning process.” [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MN.1999.758890

[106] H. Speckmann, P. Thole, and W. Rosentiel, “Hardware implementations of Kohonen’s self-organizing feature map,” in *Artificial Neural Networks, 2*, I. Aleksander and J. Taylor, Eds., vol. II. Amsterdam, Netherlands: North-Holland, 1992, pp. 1451–1454.

[107] ——, “COKOS: A coprocessor for Kohonen Self-organizing map,” in *Proc. ICANN’93, International Conference on Artificial Neural Networks*, S. Gielen and B. Kappen, Eds. London, UK: Springer, 1993, pp. 1040–1044.

[108] ——, "COprocessor for KOhonen's Selforganizing map (cokos)," *Proc. IJCNN'93, International Joint Conference on Neural Networks*, vol. 2, pp. 1951–1954, 1993.

[109] ——, "Hardware synthesis for neural networks from a behavioral description with VHDL," in *Proc. IJCNN-93, International Joint Conference on Neural Networks, Nagoya*, vol. II, JNNS. Piscataway, NJ: IEEE Service Center, 1993, pp. 1983–1986.

[110] H. Speckmann, G. Raddatz, and W. Rosenstiel, "Considerations of geometrical and fractal dimension of SOM to get better learning results," in *Proc. ICANN'94, International Conference on Artificial Neural Networks*, M. Marinaro and P. G. Morasso, Eds., vol. I. London, UK: Springer, 1994, pp. 342–345.

[111] H. Speckmann, P. Thole, M. Bogdan, and W. Rosentiel, "Coprocessor for special neural networks KOKOS and KOBOLD," in *Proc. ICNN'94, International Conference on Neural Networks*. Piscataway, NJ: IEEE Service Center, 1994, pp. 1959–1962.

[112] H. Speckmann, P. Thole, M. Bogdan, and W. Rosenstiel, "Coprocessors for special neural networks KOKOS and KOBOLD," in *Proc. WCNN'94, World Congress on Neural Networks*, vol. II, INNS. Hillsdale, NJ: Lawrence Erlbaum, 1994, pp. 612–617.

[113] H. Speckmann, G. Raddatz, and W. Rosenstiel, "Improvement of learning results of the self-organizing map by calculating fractal dimensions," in *Proc. ESANN'94, European Symp. on Artificial Neural Networks*, M. Verleysen, Ed. Brussels, Belgium: D facto conference services, 1994, pp. 251–255.

[114] L.-C. Chang and F.-J. Chang, "An efficient parallel algorithm for LISSOM neural network," *Parallel Comput.*, vol. 28, no. 11, pp. 1611–1633, 2002. [Online]. Available: http://dx.doi.org/10.1016/S0167-8191(02)00166-7

[115] "Unsupervised learning in hardware: Citcuits for vlsi implementation of temporally asymmetric hebbian learning," 2001. [Online]. Available: http://books.nips.cc/papers/files/nips14/IM05.pdf

[116] E. Keller, "Building asynchronous circuits with JBits," in *Proceedings of the 11th International Conference on Field-Programmable Logic and Applications, FPL '01*. London, UK: Springer-Verlag, 2001, pp. 628–632.

[117] Gokhale, Maya, Graham, and P. S., *Reconfigurable Computing: Accelerating Computation with Field-Programmable Gate Arrays*. Springer, 2005.

[118] D. J. C. MacKay, *Information Theory, Pattern Recognition and Neural Networks*, 1997. [Online]. Available: http://citeseer.ist.psu.edu/article/mackay97information.html

[119] J. A. Williams and N. W. Bergmann, "Embedded linux as a platform for dynamically self-reconfiguring systems-on-chip," in *Proc. International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA '04)*, Las Vegas, Nevada, 2004. [Online]. Available: http://www.itee.uq.edu.au/~esg/unwrapped/redirect?id=00001296andhttp://www.itee.uq.edu.au/~esg/about/public/jw-esg0404-linux-self-reconfig.pptandhttp://linuxdevices.com/articles/AT7708331794.html

[120] A. Ahmadinia *et al.*, "Designing partial and dynamically reconfigurable applications on Xilinx Virtex-ii FPGAs using Handel-C," University of Erlangen-Nuremberg, Department of CS 12, Hardware-Software-Co-Design, Am Weichselgarten 3, D-91058 Erlangen, Germany, Tech. Rep. 03-2004, Dec. 2004. [Online]. Available: http://www12.informatik.uni-erlangen.de/publications/pub2004/BBHNAM04r.pdf

[121] B. Blodget, P. James-Roxby, E. Keller, S. McMillan, and P. Sundararajan, "A self-reconfiguring platform," Sep. 2003. [Online]. Available: http://citeseer.ifi.unizh.ch/721298.html

[122] B. Blodget, S. McMillan, and P. Lysaght, "A lightweight approach for embedded reconfiguration of FPGAs," *DATE*, vol. 01, p. 10399, 2003. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/DATE.2003.10160

[123] R. J. Fong, S. J. Harper, and P. M. Athanas, “A versatile framework for FPGA field updates: An application of partial self-reconfiguration,” *RSP*, vol. 00, p. 117, 2003. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/IWRSP.2003.1207038

[124] D. Roggen, S. Hofmann, Y. Thoma, and D. Floreano, “Hardware spiking neural network with run-time reconfigurable connectivity in an autonomous robot,” *eh*, vol. 00, p. 199, 2003. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/EH.2003.1217666

[125] T. Yoneda, A. Matsumoto, M. Kato, and C. Myers, “High level synthesis of timed asynchronous circuits,” *ASYNC*, vol. 00, pp. 178–189, 2005. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ASYNC.2005.22

[126] I. E. Sutherland and J. K. Lexau, “Designing fast asynchronous circuits,” *ASYNC*, vol. 00, p. 184, 2001. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ASYNC.2001.914082

[127] W. Maass, “Computation with spiking neurons.”

[128] M. W. Benson and J. Hu, “Asynchronous self-organizing maps,” *IEEE Transactions on NNs*, vol. 11, no. 6, pp. 1315–1322, Nov. 2000.

[129] M. Köster and J. Teich, “(self-)reconfigurable finite state machines,” in *Proc. DATE 2002*. Paris, France: IEEE Computer Society Press, March 4-8 2002, pp. 559–566. [Online]. Available: http://citeseer.ist.psu.edu/574667.html

[130] K. Jensen, *Coloured Petri nets (2nd ed.): basic concepts, analysis methods and practical use: volume 1*. London, UK: Springer-Verlag, 1996.

[131] F. Heylighen and C. Joslyn, “Cybernetics and second order cybernetics,” *Encyclopedia of physical science & technology*, vol. 4, pp. 155–170, 2001.

[132] L. A. Krundel, S. K. Goel, E.-J. Marinissen, M.-L. Flottes, and B. Rouzeyre, "User-constrained test architecture design for modular soc testing," in *ETS'04: Proceedings of the European Test Symposium, Ninth IEEE ETS'04*. Washington, DC, USA: IEEE Computer Society, May 23–26, 2004, pp. 80–85. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ETSYM.2004.1347611

[133] D. Cornforth, D. Green, D. Newth, and M. Kirley, "Do artificial ants march in step? Ordered asynchronous processes and modularity in biological systems," in *Artificial Life VIII: Proceedings of the Eighth International Conference on Artificial Life, 2003*. Mit Press, 2003, p. 28.

[134] L. A. Krundel, V. A. Chouliaras, and D. J. Mulvaney, "A knowledgeable authority for interactive hardware design capture support," in *Ph.D. Forum, VLSI-SoC 2006, IEEE, IFIP International Conference on Very Large Scale Integration, Nice, France*, Oct. 16–18, 2006, pp. 1–6. [Online]. Available: http://vassilios-chouliaras.com/pubs/c38.pdf

[135] D. Roggen, D. Federici, and D. Floreano, "Evolutionary morphogenesis for multi-cellular systems," *Genetic Programming and Evolvable Machines*, vol. 8, no. 1, pp. 61–96, 2007.

[136] D. Edwards, A. Bardsley, L. Janin, L. Plana, and W. Toms, "Balsa: A Tutorial Guide." *Version V3.5*, 2006.

[137] H. Shayani, P. Bentley, and A. Tyrrell, "A Cellular Structure for Online Routing of Digital Spiking Neuron Axons and Dendrites on FPGAs," in *Proceedings of the 8th international conference on Evolvable Systems: From Biology to Hardware*. Springer, 2008, pp. 273–284.

[138] ——, "Hardware Implementation of a Bio-plausible Neuron Model for Evolution and Growth of Spiking Neural Networks on FPGA," in *Adaptive Hardware and Systems, 2008. AHS'08. NASA/ESA Conference on*, 2008, pp. 236–243.

[139] Toffoli and Margolus, *Cellular Automata Machines*. Massachusetts Institute of Technology (MIT), 1987.

[140] Y. Takada, T. Isokawa, F. Peper, and N. Matsui, “Construction universality in purely asynchronous cellular automata,” *Journal of Computer and System Sciences*, vol. 72, no. 8, pp. 1368–1385, 2006.

[141] H. de Garis, “Brain building for a biological robot,” in *Neural network for robotic control*. Ellis Horwood Upper Saddle River, NJ, USA, 1996, ch. 10.

[142] M. Abramson and H. Wechsler, “Competitive reinforcement learning for combinatorial problems,” in *Proceedings of the International Joint Conference on Neural Networks*, vol. 4. George Mason University, 2001, pp. 2333–2338.

[143] L. Soldatova, A. Clare, A. Sparkes, and R. King, “An ontology for a Robot Scientist,” *Bioinformatics*, vol. 22, no. 14, 2006.

[144] E. Klavins, “Programmable self-assembly: Control of concurrent systems from the bottom up,” *IEEE Control Systems Magazine*, vol. 27, no. 4, pp. 43–56, 2007.

[145] N. Brunel and X. Wang, “Effects of neuromodulation in a cortical network model of object working memory dominated by recurrent inhibition,” *Journal of Computational Neuroscience*, vol. 11, no. 1, pp. 63–85, 2001.

[146] R. Kamimura and T. Kamimura, “Information theoretic rule discovery in neural networks,” in *2000 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4, 2000.

[147] Y. Abu-Mostafa, “Information theory, complexity and neural networks,” *IEEE Communications Magazine*, vol. 27, no. 11, pp. 25–28, 1989.

[148] A. Pérez-Uribe, “Structure-adaptable digital neural networks,” Ph.D. dissertation, ÉCOLE POLYTECHNIQUE FÉDERALE DE LAUSANNE, 1999.

[149] P. Collard and A. Goforth, “Knowledge based systems and Ada: an overview of the issues,” *Ada Letters*, p. 72, 1988.

[150] J. Schmidhuber, “Godel machines: Fully self-referential optimal universal self-improvers,” *Artificial General Intelligence. Springer-Verlag*, pp. 228–291, 2005.

[151] S. Furber, “The amulet asynchronous arm processors,” in *ARM system-on-chip architecture*. Addison-Wesley, 2000, ch. 14, pp. 374–398.

[152] A. Jadbabaie, N. Motee, and M. Barahona, “On the stability of the Kuramoto model of coupled nonlinear oscillators,” in *American Control Conference, 2004. Proceedings of the 2004*, vol. 5, 2004.

[153] S. Strogatz, “From Kuramoto to Crawford: exploring the onset of synchronization in populations of coupled oscillators,” *Physica D: Nonlinear Phenomena*, vol. 143, no. 1-4, pp. 1–20, 2000.

[154] M. Yeung and S. Strogatz, “Time delay in the Kuramoto model of coupled oscillators,” *Physical Review Letters*, vol. 82, no. 3, pp. 648–651, 1999.

[155] A. Upegui, “Dynamically Reconfigurable Bio-inspired Hardware,” Ph.D. dissertation, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2006.

[156] R. Turner and A. Eden, “The Philosophy of Computer Science,” *Stanford Encyclopedia of Philosophy*, 2008.

[157] P. de Vries, *Trust in systems: Effects of direct and indirect information*. Technische Universiteit Eindhoven, 2004.

[158] K. Coleman, “Computing and moral responsibility,” *Stanford Encyclopedia of Philosophy*, 2004.

[159] E. Ronald and M. Sipper, “Intelligence is not enough: On the socialization of talking machines,” *Minds and Machines*, vol. 11, no. 4, pp. 567–576, 2001.

[160] M. Kaku, *Hyperspace: A Scientific Odyssey Through Parallel Universes, Time Warps, and the Tenth Dimension.* Oxford University Press, 1995.

[161] L. A. Krundel, D. J. Mulvaney, and V. A. Chouliaras, “Self-adaptive compact neuro-controller for natural interaction and training of autonomous communicating robots,” in *IEEE-RAS SIRCon2009: in Proceedings of the 2009 International Conference on Service and Interactive Robotics (SIRCon)*, IEEE Robotics and Automation Society - IEEE-RA and Robotics Society of Taiwan - RST; Taipei World Trade Center, Taipei, Taiwan. IEEE Computer Society, Aug. 5–8, 2009, pp. 1–10.

[162] ——, “Autonomous design in vlsi: an in-house universal cellular neural platform,” in *Proceedings of the 2010 IEEE Computer Society Annual Symposium on VLSI (ISVLSI conference).* Conference Center, Lixouri, Kefalonia, Greece: IEEE Computer Society ISVLSI2010, Jul. 5 –7, 2010.

[163] N. Macias and L. Durbeck, “Self-Organizing Digital Systems,” in *Advances in Applied Self-organizing Systems, 2007.* Springer, 2007, ch. 9, pp. 177–215.

[164] E. Izhikevich, “Simple model of spiking neurons,” *IEEE Transactions on neural networks, 2003*, vol. 14, no. 6, pp. 1569–1572, 2003.

[165] M. Sipper and E. Ruppin, “Co-evolving cellular architectures by cellular programming,” in *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, 1996, pp. 306–311.

[166] J. Schmidhuber, D. Wierstra, M. Gagliolo, and F. Gomez, “Training recurrent networks by evolino,” *Neural computation, MIT Press 2007*, vol. 19, no. 3, pp. 757–779, 2007.

[167] F. Michaud and R. Rubio, “Autonomous design of artificial neural networks by neurex,” *Neural computation, 1996*, vol. 8, no. 8, pp. 1767–1786, 1996.

[168] J. Becker, M. Hübner, G. Hettich, R. Constapel, J. Eisenmann, and J. Luka, “Dynamic and partial FPGA exploitation,” *Proceedings of the IEEE*, vol. 95, no. 2, pp. 438–452, 2007.

[169] K. Glette and J. Torresen, “A flexible on-chip evolution system implemented on a Xilinx Virtex-II Pro device,” *Lecture notes in computer science*, vol. 3637, p. 66, 2005.

[170] P. Zizzi, “General Relativity and Quantum Cosmology Title: Emergent Consciousness: From the Early Universe to Our Mind,” *Journal reference: NeuroQuantology, 2003*, vol. 3, pp. 295–311, 2003.

[171] G. Tempesti, D. Roggen, E. Sanchez, Y. Thoma, R. Canham, A. Tyrrell, and J. Moreno, “A POEtic architecture for bio-inspired hardware,” in *Artificial Life VIII: Proceedings of the 8th International Conference on Artificial Life, 2003*. Mit Press, 2003, p. 111.

[172] L. RougÃ©, “Menta.” [Online]. Available: http://www.menta.fr

[173] Y. Zhou and A. Zolotas, “Sensor selection in neuro-fuzzy modelling for fault diagnosis,” in *IEEE International Symposium on Industrial Electronics*, Bari, Ed., Italy, Jul. 2010.

[174] L. N. Chrystopher, “Evolution in asynchronous cellular automata,” in *Proceedings of the eighth international conference on Artificial life*. MIT Press, 2003.

[175] N. Fatès and M. Morvan, “An Experimental Study of Robustness to Asynchronism for Elementary Cellular Automata,” *Complex Systems*, vol. 16, no. 1, pp. 1–27, 2005.

[176] A. M. Tyrrell and E. Sanchez, “POEtic Tissue: An Integrated Architecture for Bio-inspired Hardware.” *Evolvable Systems: From Biology to Hardware*, pp. 269–294, 2003.

[177] S. A. Wills, “Computation with Spiking Neurons,” Ph.D. dissertation, University of Cambridge, 2004.

[178] G. F. Luger, *Artificial Intelligence: Structures And Strategies For Complex Problem Solving, the 6th Edition*. Pearson Education.

[179] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007.

[180] D. Obradovic, “On-line training of recurrent neural networks with continuous topology adaptation,” *IEEE Transactions on Neural Networks*, vol. 7, no. 1, pp. 222–228, 1996.

[181] M. Marshall, “Brain-like chip outstrips normal computers,” *NewScientist Magazine*, vol. 2892, Nov. 2012.

[182] ——, “Ultimate Cognition Ã la GÃ¶del,” *Cognitive Computation*, vol. 1, pp. 177–193, 2009.

[183] Y. Zhou and L. Krundel, “Spiking neural networks on self-updating system-on-chip for autonomous control,” in *Proceedings of CCCA2011*, Hong-Kong, Apr. 2011.

[184] F. Alnajjar and K. Murase, “Sensor-fusion in spiking neural network that generates autonomous behavior in real mobile robot,” in *IEEE World Congress on Computational Intelligence*, I. 2008, Ed. Neural Networks, 2008.

[185] E. Ros, R. Agis, R. Carrillo, and E. Ortigosa, “Post-synaptic time-dependent conductances in spiking neurons: FPGA implementation of a flexible cell model.”

[186] H. B. Barlow, “Unsupervised Learning,” *Neural Computation*, vol. 1, no. 3, pp. 295–311, 1989.

[187] N. e. a. Thomas, “Unsupervised learning and selforganization in networks of spiking neurons. Self-Organizing neural networks,” *Springer-Verlag New York, Inc.*, pp. 45–73, 2002.

[188] Y. LeCun, “Deep learning.” [Online]. Available: http://yann.lecun.com

[189] D. E. Knuth, “A draft of section 7.2.1.2: Generating all permutations,” in *THE ART OFCOMPUTER PROGRAMMING*. Stanford University, USA: Addison-Wesley, Dec. 2004, ch. PRE-FASCICLE 2B, p. 38. [Online]. Available: www-cs-faculty.stanford.edu/~uno/fasc2b.ps.gz

[190] L. Krundel and D. Veyrié, “Conception d’une matrice de circuits robustes / radiation-proofing of sram, based on toshiba design,” Master’s thesis, Poly-tech’Montpellier, Montpellier University 2, France, LIRMM Laboratories, Montpellier, France and iRoC Technologies Company, Grenoble, France, Jan. 2003.

[191] “La Vie serait Quantique ! / Life is quantum!” *Science & Vie*, vol. 1123, pp. 54–73, Apr. 2011.

[192] S. Hameroff, “The conscious pilot - dendritic synchrony moves through the brain to mediate consciousness,” *Journal of Biological Physics*, vol. 36, no. 71, 2009.

[193] R. Bryanton, “O is for omniverse.” [Online]. Available: http://www.omniverse.tv

[194] “Double-slit experiment: ‘the most beautiful experiment’,” Physics World, 2002. [Online]. Available: http://physicsworld.com/cws/article/print/9746

[195] “Backward causation,” Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Retrocausality

[196] A. Konar and L. Jain, *Cognitive engineering: a distributed approach to machine intelligence*. Springer, 2005.

[197] J. Gribbin, *In Search of the Multiverse*.

[198] S. M. Oh, H.-S. Yoon, B.-J. Yi, H. S. Yoon, and Y. J. Cho, “A spring-backboned soft arm emulating hu-man gestures for human-robot interaction,” in *Proceedings of the 2009*

International Conference on Service and Interactive Robotics (SIRCon), R. S. of Taiwan (RST), Ed., IEEE Computer Society. Hong-Kong: IEEE Robotics and Automation Society (IEEEâŁ“RAS), Aug. 2009.

[199] “Military robots,” IEEE Spectrum. [Online]. Available: http://spectrum.ieee.org/robotics/military-robots/autonomous-robots-in-the-fog-of-war/0

[200] “Redhot robots,” IEEE Spectrum. [Online]. Available: http://spectrum.ieee.org/robotics/humanoids/redhot-robots

[201] P. O. Haikonen, *Robot Brains, Circuits and Systems for Conscious Machines*.

[202] C. Breazeal, “Leonardo: Cutest $1m robotic puppet,” IEEE Spectrum. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/robotics-software/leonardo

[203] “Ibm presents brainlike chips,” IEEE Spectrum. [Online]. Available: http://spectrum.ieee.org/tech-talk/computing/hardware/ibm-presents-brainlike-chips

[204] “Ibm press release # 35251,” IBM. [Online]. Available: http://www-03.ibm.com/press/us/en/pressrelease/35251.wss

[205] “Ibm demos cognitive computer chips,” EETimes. [Online]. Available: http://www.eetimes.com/electronics-news/4218883/IBM-demos-cognitive-computer-chips

[206] M. Versace and B. Chandler, “Moneta: A mind made from memristors, darpa’s new memristor-based approach to ai consists of a chip that mimics how neurons process informa-tion,” IEEE Spectrum, 2010. [Online]. Available: http://spectrum.ieee.org/robotics/artificial-intelligence/moneta-a-mind-made-from-memristors

[207] H. Markram, “Blue brain,” EPFL. [Online]. Available: http://bluebrain.epfl.ch/

[208] D. Graham-Rowe, “Facets: Building a brain on a silicon chip, a chip developed by european scientists simulates the learning capabilities of the human brain,” Technology

Review, Mar. 2009. [Online]. Available: http://www.technologyreview.com/computing/22339/page1/

[209] A. Winfield, “Five roboethical principles âŁ“ for humans,” *NewScientist Magazine*, vol. 2811, May 2011. [Online]. Available: http://www.newscientist.com/article/mg21028111.100-five-roboethical-principles--for-humans.html

[210] R. Kamimura, *Information Theoretic Neural Computation*, Tokai University, Japan. [Online]. Available: http://www.worldscibooks.com/compsci/4224.html

[211] B. Tracy, *Maximum Achievement*.

[212] Osho, *Love, Freedom, Aloneness*.

[213] A. Cardon, *Un Modèle Constructible de Système Psychique / A Constructible Model of Psyche*, Jan. 2011.

[214] G. I. Nierenberg and H. H. Calero, *How to read a person like a book*.

[215] A. Goswami, *The Self-Aware Universe: how consciousness creates the material world*.

[216] D. Goleman, *Emotional Intelligence*.

[217] D. McFarland, *Guilty Robots, Happy Dogs: the questions of alien minds*.

[218] R. Kurzweil, *Live Long Enough to Live Forever*.

[219] Y.-C. Huang, “7. Pink Dramas: Reconciling Consumer Modernity and Confucian Womanhood,,” *TV Drama in China II. Gender and Domestic Sphere*, Oct. 2008.

[220] ——, “Situating Taiwanese identities : social trans-formations, young people and television drama,” Ph.D. dissertation, Loughborough University, 2009.

[221] “Personal satellite assistant (psa) robots,” NASA.

[222] D. Livingstone, “Unsupervised Learning,” in *A Practical Guide to Scientific Data Analysis*, W. book chapters, Ed. Chichester, UK: John Wiley & Sons, Ltd, 2009.

[223] “Dextre robot,” IEEE Spectrum. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/industrial-robots/dextre-robot-repairs-iss-while-astronauts-sleep

[224] [Online]. Available: http://www.bbc.co.uk/sn/tvradio/programmes/horizon/ghostgenes.shtml

[225] J. Becker and R. Hartenstein, “Configware and morphware going mainstream,” *J. Syst. Archit.*, vol. 49, no. 4-6, pp. 127–142, 2003. [Online]. Available: http://dx.doi.org/10.1016/S1383-7621(03)00073-0

[226] R. Hartenstein, “Morphware and configware,” TU Kaiserslautern.

[227] M. C. Escher, *Drawing hands*. Cordon Art., 1998.

[228] A. Buhlmeir and G. Manteuffel, “Operant conditioning in robots,” in *Neural Systems for Robotics*, 1992, p. 198.

[229] R. Baddeley, P. Hancock, and P. Földiák, *Information theory and the brain*. Cambridge University Press, 2000.

[230] A. Coolen and R. Kuhn, “Information theory in neural networks,” *Lecture Notes of Course CMNN14*, 2005.

[231] [Online]. Available: http://www.itee.uq.edu.au/~listarch/partial-reconfig/

[232] [Online]. Available: http://electronicsweekly.com/Articles/2006/05/16/38654/Toshibagoesbigonconfigurableprocessors.htm

[233] [Online]. Available: http://en.wikipedia.org/wiki/Reconfigurable_computing