FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

CMOS Design and Implementation of a Reduced-KII Multiplexed Network

Rui Graça

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Professor Cândido Duarte

Co-Supervisor: Professor Vítor Grade Tavares

October 15, 2015

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES | **2014/2015**

A Dissertação intitulada

“CMOS Design and Implementation of a Reduced-KII Multiplexed Network”

foi aprovada em provas realizadas em 15-10-2015

o júri

[illegible]

Presidente Professor Doutor Diamantino Rui da Silva Freitas
Professor Associado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

[illegible]

Professor Doutor Paulo Mateus Mendes
Professor Associado do Departamento Eletrónica Industrial da Universidade do Minho

[illegible]

Professor Doutor Manuel Cândido Duarte dos Santos
Professor Auxiliar Convidado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

x [illegible]

Autor - Rui Pedro Zenhas Graça

Faculdade de Engenharia da Universidade do Porto

Abstract

The need for massively parallel computing strategies has increased over the last years, chiefly due to the increasing demand for real-time pattern recognition applications. Nature and several million years of evolution have provided human beings with a highly-parallel and fault tolerant mechanism for solving such problems - the brain. Hence, it is of practical interest to artificially mimic brain operation with the clear objective of embedding evermore "intelligence" into engineering applications. This idea is not new, and the concept of artificial neural networks is well established in the scientific community and more recently in industry. It should be noticed however that the relationship between most artificial neural networks and accurate physiological models is a rather loose bond.

Analog VLSI is a promising technology for implementing these massively parallel algorithms, favouring solutions with low-power and area consumption. However, massively parallel systems are generally massively interconnected, which poses big challenges for VLSI design.

In this work, the RKII network, a biologically realistic model of the olfactory bulb is studied and implemented in analog VLSI. Although it is intrinsically very different from generic feed-forward or recursive artificial neural networks, the basic processing cells are very similar in both cases, and as with the last, interconnections are an important issue.

The RKII network is a set of coupled oscillators, and it behaves as a content addressable memory, much like a Hopfield network. However, differently than the Hopfield network, trajectories in space state are limit cycles. This dynamic behavior allows quick state transition upon input change.

Early implementations of the model decreased the burden of interconnections by time multiplexing the system. These implementations served as a proof of concept. In this work, a system implementation based on earlier approaches, but with capacity for small practical applications, is studied and designed.

Resumo

A necessidade de formas de computação altamente paralelas tem vindo a crescer nos últimos anos, principalmente devido ao aumento da importância de aplicações de reconhecimento de padrões em tempo real. A natureza, juntamente com milhões de anos de evolução, forneceu ao ser humano um mecanismo de computação altamente paralelo e tolerante a falhas - o cérebro. A mimetização artificial do nosso cérebro é, portanto, uma boa base de partida para o desenvolvimento de aplicações dotadas de inteligência, com interesse em aplicações de engenharia. Esta ideia não é nova, e o conceito de rede neuronal artificial está bem estabelecido na comunidade científica e, mais recentemente, na indústria. É importante referir, no entanto, que a relação entre a generalidade das redes neuronais artificiais e modelos fisiológicos mais precisos é bastante vaga.

VLSI analógico é uma tecnologia com elevado potencial para a implementação destes algoritmos altamente paralelos, com baixo consumo de potência e com baixa ocupação de área. No entanto, sistemas altamente paralelos são geralmente altamente interligados, o que impõe grandes desafios em implementações VLSI.

Neste trabalho, a rede RKII, um modelo biologicamente realista do bolbo olfativo é estudado e implementado em VLSI analógico. Apesar de ser intrínsecamente diferente das redes neuronais artificiais, as células elementares de processamento são semelhantes em ambos os casos, e, tal como nas redes neuronais artificiais, a implementação das interconexões é uma questão importante a resolver.

A rede RKII é um conjunto de osciladores acoplados, que se comporta como uma memória endereçável por conteúdo, à semelhança da rede de Hopfield. No entanto, ao contrário do que acontece na rede de Hopfield, as trajetórias no espaço de estados são oscilatórias. Este comportamento dinâmico permite uma mudança rápida do estado perante a mudança da entrada.

Implementações anteriores do mesmo modelo usaram multiplexagem temporal para diminuir o número de interconexões físicas. Estas implementações foram usadas como prova de conceito. Neste trabalho, é estudada e proposta uma implementação de um sistema baseado nas abordagens anteriores, mas com capacidade para pequenas aplicações práticas.

Acknowledgements

I would like to thank my supervisors, Professor Cândido Duarte and Professor Vítor Tavares, for all the guidance and the support in this work. Moreover, I thank them for proposing me such an interesting work to develop and explore.

I thank Microelectronics Students' Group for all technical support, without which, this work would not be possible.

I thank my family, specially my parents and my brother, for all the moral support and for being always available to help me with everything.

I thank my friends for the encouragement and for the interest shown towards this work.

I specially thank Rita for the emotional support and for being always on my side.

Rui Graça

Contents

List of Figures

List of Tables

Acronyms

ASIC	application-specific integrated circuit
BIST	built-in self test
CAM	content-addressable memory
CNN	cellular neural network
CS	chip select
EEG	electroencephalogram
F&H	filter and hold
FPGA	field-programmable gate array
GPU	graphics processing unit
LMS	least-mean-square
MLP	multilayer perceptron
SCL	serial clock
SDI	serial data in
SDO	serial data out
SPI	serial peripheral interface
VLSI	very-large-scale integration
XOR	exclusive OR

Chapter 1

Introduction

Over the years, the traditional digital computer has proven to be an extremely powerful tool, performing many tasks much better and faster than humans. Operations such as the multiplication of large numbers, which take a considerable amount of time for a human being to complete, are performed in a few nanoseconds by digital computers. However, there are still operations in which our brain performs much better than any computer. It is able to learn from experience, and based on that, it is able to recognize data in extremely noisy environments. For instance, it is extremely easy for humans to identify a familiar face and to associate that face to a certain person in completely different situations. Even if the face is partially covered, seen from different angles or even if it has changed considerably, we are able to associate it to the right person. This kind of operations are extremely difficult to implement with intrinsically sequential algorithms on traditional computers. However, they are quite interesting from an engineering point of view. Pattern recognition and machine learning are fields that are becoming more and more important, since more efficient ways to learn from data, and to abstractly recognize data in noisy environment are demanded. Among important applications that would benefit from more efficient pattern recognition, we can highlight computer-aided medical diagnosis, the recognition of handwritten characters and speech recognition. All these tasks are relatively simple to humans, therefore research about computation in the brain offers engineers a starting point for the development of electronic machines based on how the brain operates, which may be able to solve problems as humans do.

Although appealing, the implementation of electronic devices based on biological evidence is not a simple task. The brain is massively interconnected, and interconnections are a huge problem in very-large-scale integration (VLSI) implementations. Hence, direct application of most models for computing machines are not scalable. In this work, an approach based on time multiplexing is followed [1]. This approach significantly reduces the burden of the interconnections at the cost of a more complex digital control structure.

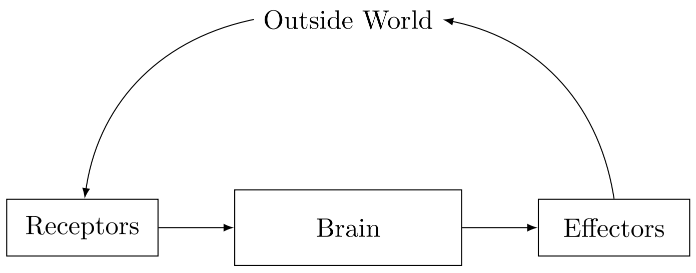


Figure 1.1: Interface between the brain and the outside world.

1.1 Modeling the Brain

It is reasonable to consider that modeling the brain is fundamental for conceiving engineering systems with similar functionalities. In fact, as stated by Haykin [2], “the brain is the living proof that fault tolerant parallel computing is not only physically possible, but also fast and powerful”. It is the central element of the nervous system and it is constituted by nervous cells, commonly known as neurons.

As depicted in Figure 1.1, the brain receives information from the outside world (or from within the body) through receptors and acts on the outside world through effectors. Both receptors and effectors are essential to the operation of the brain, since they provide the interface between physical external information and signals that can be processed by the brain. These signals, known as action potentials or spikes, are voltage pulses. Spikes propagate within a neuron through its axon, a relatively long line, which can be modelled as an RC transmission line [2]. Neurons communicate with each other by synapses, being the chemical synapse the most common interface. In a chemical synapse, the occurrence of patterns of action potentials causes the release of a chemical substance, the neurotransmitter, from the pre-synaptic neuron. This substance is received by a post-synaptic neuron, triggering a pattern of action potentials [3]. This kind of synapse is an electrical to chemical conversion followed by the reverse conversion. A synapse can either have excitatory or inhibitory effects over the post-synaptic neuron.

Although early studies tried to model the brain similarly to a digital computer [4], there are evidences that behavior and precise functions result from the organization of neural circuits, making the paradigm of the logical computer unsuitable for modeling the brain [3,5,6]. This gave rise to a new paradigm, known as connectionism [5], that defends that the way neurons are connected with each other is determinant to their function. In this way, networks of neurons are formed with information stored in the interconnections.

Based on the same principles as connectionists, and due to the large density of interconnection among neurons, another paradigm appeared, suggesting that neural activity is best explained observing masses of neurons, or neural sets. Under such assumption, the brain is considered a continuum and is modeled by continuous differential equations [6–8], whereas in the study of individual neurons, action potentials are used as the observable basis of study. Conversely, the study

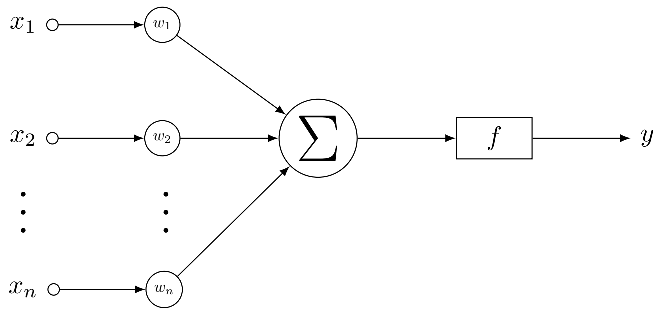


Figure 1.2: Artificial neuron.

of neural masses is based on the observation of electroencephalogram (EEG) waveforms [6].

Early studies suggested that the brain cortex had distinct functional regions [9], meaning that language was treated by a section of the brain, movement by a distinct section, smell by some other section, and so on. According to theory of mass action [10], however, this is not entirely true. In his work, Lashsley observed that the capability of a rat with a brain lesion to learn a task is much more dependant on the severity of the lesion than on its exact location. He concluded that the brain is highly redundant, since, under deprivation of a sensory capability, a function that would usually be associated to the region of the lesion can still be performed using some other region of the cortex.

The study of computation in the brain, commonly known as computational neuroscience [11], aims to model and to understand the mechanisms used by the brain to execute functions. The brain is generally modeled as a nonlinear dynamical system [12].

Concluding, the brain can be viewed as a massively parallel computer. It is highly redundant and, therefore, highly fault tolerant. Its operation, however, is radically different than the one of a digital computer.

1.2 Artificial Neural Networks

An artificial neural network is the earliest engineering system inspired on (loose) biological evidence. It consist of several nodes, known as artificial neurons, which perform parallel computation. A typical artificial neuron is depicted in Figure 1.2.

The concept appeared in the 1940s, with the first mathematical model of a neural network, proposed by McCulloch and Pitts [4]. After that, the field of neural networks became quite active. The goal was both to understand how the brain works at a level higher than synapses and action potentials, and to develop a machine able to perform the same kind of tasks that the brain is good at, exploring highly parallel computation approaches. A major step in the development of neural networks was the perceptron [5]. The perceptron is a single-layer feedforward neural network. This means that, besides the input layer, which receives the input signals from the outside (and does not perform any computation), there is only one other layer of neurons, known as the output

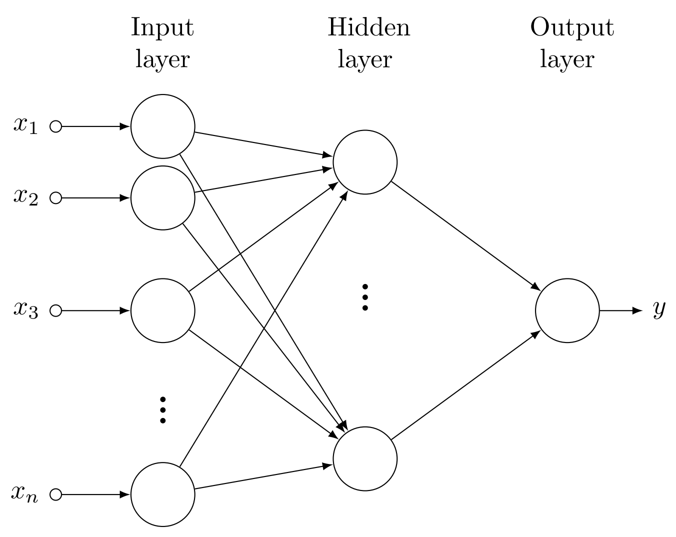


Figure 1.3: Feedforward neural network with one hidden layer.

layer. The function of a perceptron is controlled by properly adjusting the interconnection between the input and the output layers. Since it is feedforward, there are no connections neither between nodes in the same stage nor from the output stage to the input stage. As demonstrated later, the perceptron is very limited, in the way that many useful functions cannot be mapped, being the exclusive OR (XOR) a typical example of a function not mappable in the perceptron [13]. This fact significantly reduced the interest in artificial neural networks.

Later developments in the field introduced the concepts of hidden layers and recurrent networks, that significantly extend the range of problems solvable by neural networks, in relation to the perceptron. Hidden layers are layers between the input and the output stage of the network. Again, in feedforward networks, there are only connections between neurons from a layer to the following one. In this way, hidden layers do not receive inputs from the outside and their output is not an output of the network, therefore the name ‘hidden’. A simple neural network with one hidden layer may be found in Figure 1.3.

The inclusion of feedback loops in neural networks may bring interesting properties. Networks with feedback are usually known as recurrent neural networks. In such networks, the output of a stage may be the input of a previous stage. Recurrent neural networks may or may not have hidden layers. A common example of a recurrent network is the Hopfield network [14]. Hopfield networks are used as content-addressable memories (CAMs) (or associative memories) in which a learned pattern affected by noise at the input can be recovered at the output.

As referred before, the behavior of a neural network is controlled by the interconnection weights. The process of adjusting the weights to obtain a desired response is known as learning. The learning process may be either supervised or unsupervised, and may be either deterministic or stochastic.

In its genesis, artificial neural networks aimed to model the brain, in order to study how it operates at the level of learning and behavior. However, developments in this field significantly diverged from this goal, and generally, there are few similarities between artificial neural networks and more accurate models of the brain. However, there is still interest in the development of realistic electronic systems that mimic the brain operation. This field, known as neuromorphic engineering, is a highly multidisciplinary field, since it requires knowledge of neurosciences, electronic engineering and nonlinear dynamical systems.

Because neurons comprise a summation block, a nonlinear function, and interconnections that are weighted replicas of the output from other cells, and since they are intrinsically parallel, neural networks are clearly suited for an analog VLSI implementation. However, there are also digital implementations proposed in the literature, either using field-programmable gate arrays (FPGAs) or implemented on application-specific integrated circuits (ASICs). As already referred, both cases are subject to a big problem common to many other electronic applications - the interconnectivity. Biological neural networks are highly interconnected, and so are most artificial neural network models. In a totally connected feedforward network, the number of connections increases with the square of length of the input vector [15]. This is clearly unscalable, therefore, other approaches must be followed in order to overcome this problem.

1.3 Freeman Olfactory Neural System model

In contrast with many earlier approaches, Walter J. Freeman modelled the olfactory neural system not from the neuron level, but from a mesoscopic approach, based on the observation of EEG waveforms [6]. The model is defined hierarchically, with each level denoted by K, after Katchalsky, for his earlier work. The most basic cell is the KO, which includes a summation block, a linear second order low-pass filter and a sigmoid. A KI is formed by connecting two KO acting over each other with either positive weight (excitatory pair) or negative weight (inhibitory pair). The KII is composed by four KO. As shown in Figure 1.4a, two of the KO have positive (excitatory) outgoing weights, and the other two have negative (inhibitory) outgoing weights. By properly adjusting the weights, a KII behaves as an oscillator controlled by the input, in which a positive input causes the output to oscillate, and an input of 0 causes the output to vanish to 0. A KII network is formed by several KII interconnected. The highest level of the model is the KIII, and it is composed by blocks of lower levels.

The Reduced KII (RKII), shown in Figure 1.4b is a simpler cell, consisting only of two KO (one excitatory and one inhibitory), but has a behavior similar to the one of KII [6]. It is significantly simpler to use in practical applications than the KII, since it has fewer KO cells and fewer connections, which means that it has fewer parameters to adjust, and therefore it is simpler to study and train.

The KII network, and consequently also the RKII network, models the olfactory bulb, which plays a fundamental role in the identification of odorant stimuli [16]. Both the KII and the RKII networks behave as CAMs, much like Hopfield networks [17, 18]. However, differently than

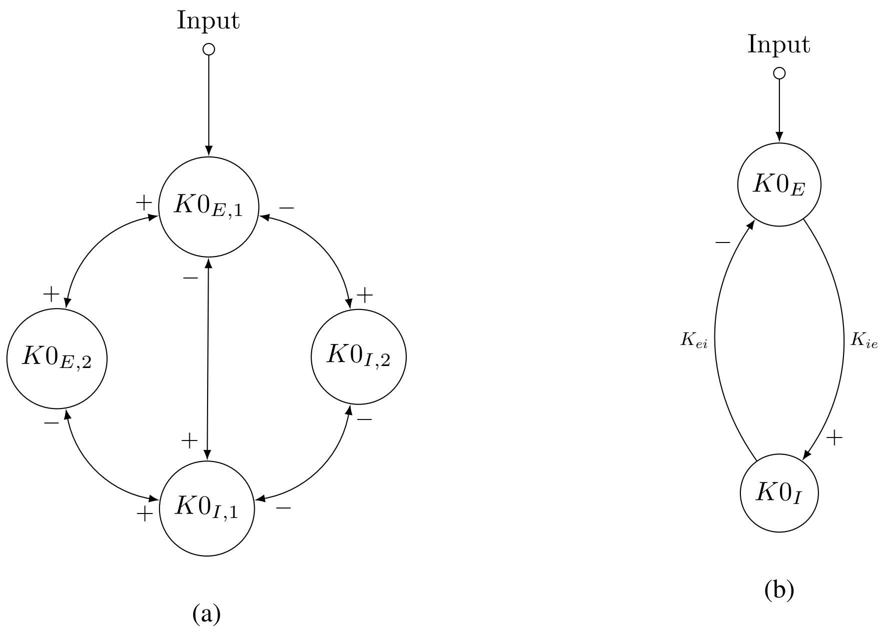


Figure 1.4: a) KII and b) RKII.

Hopfield networks, equilibrium points in the state space are unstable, resulting in oscillatory trajectories. Moreover, the KIII has chaotic basal dynamics. Both these characteristics result in that it requires less energy to change from one equilibrium point to another [16].

1.4 Objectives and Approach

In this work, a CMOS implementation of an RKII network is proposed. In order to mitigate the large number of interconnections, time multiplexing is used [1], so that the number of physical interconnections scales linearly with the number of cells, keeping the same number of logical interconnections. Moreover, time multiplexing enables the possibility of resource sharing, which, besides reducing the area and power consumption, mitigates possible problems due to mismatches. The cost of the multiplexed approach is that it requires digital control in order to properly select the active cell at the right time.

In order to further reduce mismatches, a built-in self test (BIST) circuit is implemented. This circuit, proposed by Duarte *et al.* [19], tests cells and searches for the group of cells that have the better match with each other and selects that group of cells to be active.

Departing from small networks with few cells already implemented in previous works [1, 18, 20–25], in this work a system with enough cells for small practical applications is proposed. Moreover, a control structure for the network and for the BIST is implemented. The result is a system implementation that will enable research about the application of RKII networks in pattern classification problems.

The implementation proposed uses a 0.35 µm CMOS technology with dual power supply of ± 1.65 V. The implementation is limited by area, up to a maximum of 5 mm^2.

1.5 Structure of the Document

After this brief introduction, in which some basic concepts and topics that serve as motivation for this work were presented, a deeper analysis of neural networks and their implementation is presented in Chapter 2, as well as a more detailed overview of Freeman's model and its application in pattern classification. In Chapter 3, the implementation proposed in this work is further explained and modelled, and the system architecture is presented. In Chapter 4, the design of the multiplexed network is explored, as well as its control structure. In Chapter 5 the BIST design is presented, as well as some considerations about the system implementation. At last, conclusions, and future work are presented in Chapter 6.

Chapter 2

Background and Related Work

In this chapter, the fundamentals of neural networks are explained and topics about possible hardware implementations are discussed. Although the origins of Freeman model are intrinsically different from typical neural networks, they have many similar applications and characteristics: the basic processing element (KO in Freeman model, neuron in artificial neural networks) are very similar. The interconnection problem is also a common issue to both the Freeman model and artificial neural networks. Therefore, it is interesting to study artificial neural networks and how the problems arising from hardware implementations were covered. Moreover, Freeman Olfactory System model is explained with more detail, focusing on aspects related with its application to pattern classification and with its implementation in hardware.

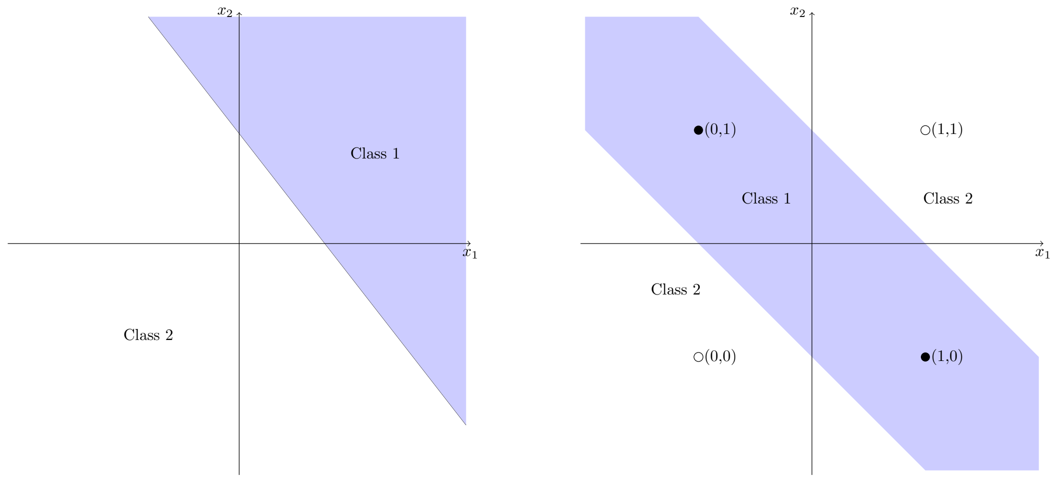
Section 2.1 focuses on the Perceptron and multilayer perceptron (MLP), Section 2.2 focuses on recursive neural networks and Section 2.3 on hardware implementation of artificial neural networks. In Section 2.4, Freeman model is presented in detail.

2.1 The Perceptron and Feedforward Neural Networks

The perceptron was an important step in the early developments of neural networks. It consists of a single layer network, without any kind of feedback. With the perceptron, Rosenblatt aimed to create a model for computation in the brain, following a connectionist approach. This approach substantially differs from earlier ones, that were significantly influenced by developments in digital computers and symbolic logic, in the way that it aimed to mimic the behavior of the brain at a lower level [5].

The perceptron is basically a linear combiner and a threshold function, which means that its inputs are weighted and summed, and the output of the perceptron is either high (+1) or low (-1) depending on the result of the weighted sum. Hence, the perceptron classifies inputs in one of two classes. A network may be formed using several perceptrons in parallel, each one trained independently for a given function.

In order to adapt the perceptron to a desired response to a set of stimuli, a learning rule is required. The perceptron learning rule is simply an adaptive filter that drives the interconnection



(a) (b)

Figure 2.1: a) shows a linearly separable problem, thus solvable by the perceptron, and b) shows a non-linearly separable problem (the XOR), not solvable by the perceptron.

weights to a set that ensures the desired response. This is done by setting the weights (and the threshold) to arbitrary small values. Being w_n the set of weights at iteration n, y_n the set of perceptron outputs for a training set x as input, using w_n as weight set, d the desired set of output for x and μ the step of the algorithm, w is updated by:

$$w_{n+1} = w_n + \mu \cdot (d - y) \tag{2.1}$$

This rule is guaranteed to converge if the problem in question is mappable to the perceptron. Otherwise, there is no set of weights that suits the problem. The threshold value can be adjusted by this algorithm by including an extra value in the weight set, with a corresponding value of 1 in x. However, as observed by Minsky and Papert [13], the set of problems mappable to the perceptron is quite limited. In order for a problem to be solvable by the perceptron, the two output classes must be linearly separable, which means that, in the input space, there must be a linear hyperplane that divides the space in two sections, with every input that maps to the same output lying in the same section, as in Figure 2.1a. As we can see in Figure 2.1b, the XOR is an example of a non-linearly separable problem. There is no such straight line that divides the space in the desired classes. This observation significantly reduced the interest, not only in the perceptron, but in neural networks in general.

Contrary to what Minsky and Papert [13] expected, however, multilayer neural networks may be constructed to map a much wider diversity of problems. For instance, a two layer network, as seen in Figure 2.2, may be used to map the XOR [2].

An example of a generic feedforward multilayer network was already showed in Figure 1.3. In a general case, it is composed by several layers, with every cell in each layer connected to every cell in the following layer and not directly connected to any other cell. Layers other than

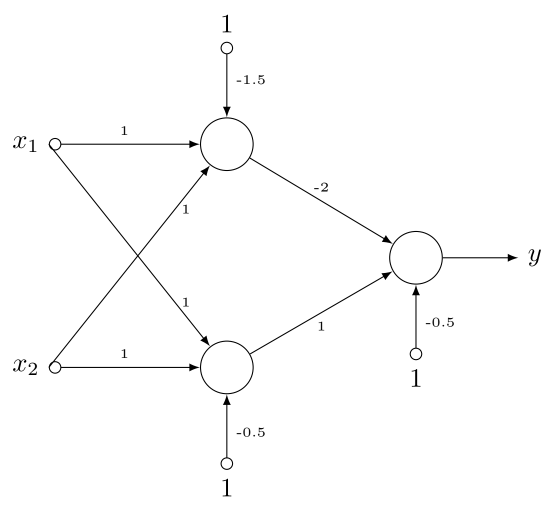


Figure 2.2: XOR mapped to a two layer network.

the input or output ones are known as hidden layers. Such networks are also known as MLPs [2]. Differently than the perceptron, MLPs usually use a sigmoid as the activation function and not a simple threshold. These functions are differentiable everywhere. A common example of a sigmoid function is the logistic function, defined by (2.2).

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.2}$$

A plot of the logistic function may be found in Figure 2.3.

The most common learning algorithm for MLPs is back-propagation, popularized by Rumelhart and McClelland in 1986 [26]. This algorithm is a generalization of the least-mean-square (LMS), which is also a popular algorithm for the perceptron [2]. It is an iterative algorithm, divided in two steps. In the first step, the response of the network for the actual set of weights is computed. In the second step, an adaptive filter is applied in order do minimize the error between the resultant response and the desired one. This is done by first computing the error for neurons in the output layer, and then iterating layer by layer, from the output to the input layer. This order is required, because the computation of the influence of a neuron to the error requires the computation of the error in subsequent layers. Due to the popularity of the algorithm, MLPs are frequently referred as back-propagation networks.

More recently, more efficient learning algorithms have been developed. This, along with the advance in computational power of general purpose computers and graphics processing units (GPUs), enabled networks with many hidden layers to be feasible [27–29]. Networks with several hidden layers are generally called Deep Neural Networks and they are now a preferred technology in pattern recognition [29].

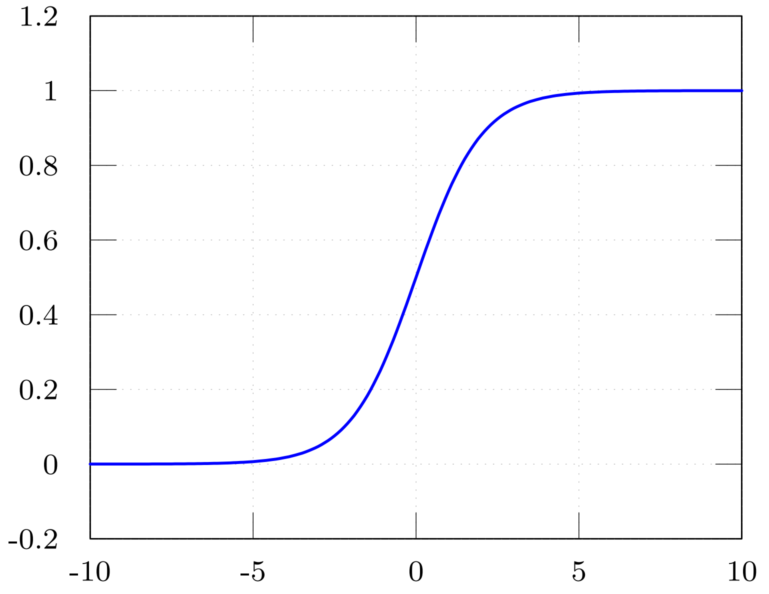


Figure 2.3: Logistic function.

2.2 Recurrent Neural Networks and Hopfield Networks

In the previous section, the networks referred were purely feedforward. From a biological perspective, this is clearly not the case in real neural networks [6], and from an engineering perspective, interesting applications may be achieved by the use of feedback. Neural networks that exhibit feedback are known as recurrent networks. The analysis of recurrent networks is substantially different from the one of feedforward networks, due to their inherent dynamical behaviour. The framework of nonlinear dynamical systems is, therefore, widely used for the analysis of recurrent networks.

As nonlinear systems, recurrent networks may exhibit many attractors in their state space. Computational properties of these networks are achieved by proper placement of their attractors, and learning consists in obtaining the desired placement [2].

The most common example of a recurrent neural network is the Hopfield model [14], popularized by Hopfield in 1982. The structure of an Hopfield network is depicted in Figure 2.4. Every neuron is connected to all other ones, but not to itself. It means that there is no self-feedback. An Hopfield network can either be discrete or continuous in time. Whereas the discrete time model uses a McCulloch-Pitts neuron, described before [14], neurons in the continuous model exhibit a low-pass linear dynamic response, cascaded with a sigmoid nonlinearity [30]. Attractors in a Hopfield network are fixed-points, meaning that it is globally stable (after a finite time, the system will converge to an equilibrium point).

It is quite intuitive that such system can be used as a CAM. A learned pattern should be stored as a state in the network state space. When a noisy and incomplete version of a stored pattern is

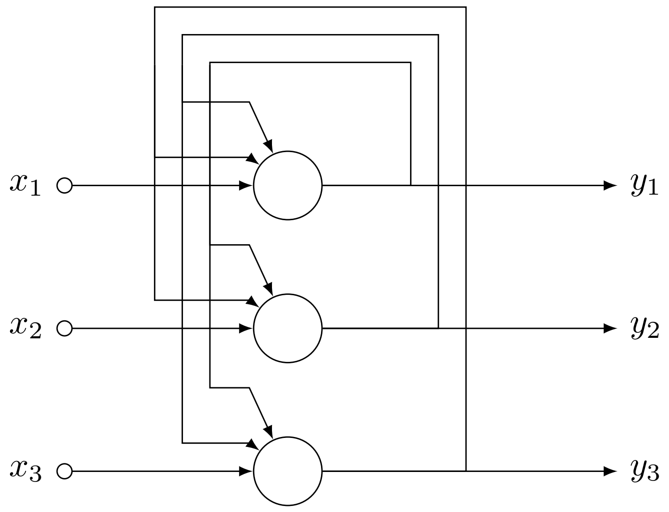


Figure 2.4: Hopfield network with three nodes.

presented at the input, the system state will converge to the stored pattern [14]. Learning consists, therefore, in properly selecting the interconnection weights so to force the desired patterns to be attractors in the state space.

As well as feedforward networks, recurrent networks may exhibit hidden layers. Examples of models of neural networks with hidden layers are the Elman networks [31] and Jordan networks [32].

2.3 VLSI Implementations of Neural Networks

Many results about neural networks were obtained by simulations in digital computers. However, in order to fully explore their parallelism, hardware implementations are desired.

Most neural networks models have an intrinsic analog behavior. Therefore, analog VLSI implementations can be achieved by direct application of those models. However, due to the large number of interconnections, such implementations would not likely be scalable. In order to obtain a scalable neural network, different approaches can be followed. An option is to design digital networks. In such networks, the burden of the interconnections can be reduced by the usage of time multiplexing and resource sharing, but they are not as efficient as their analog counterpart when it comes to the accumulation of the inputs and the multiplication by the weights [15]. Analog implementations are more efficient in terms of area and power, but are are much more sensitive to variations in the manufacturing process [33].

The usage of hybrid networks, such as the one proposed in this work, combines advantages from both analog and digital implementations. For instance, in the multiplexed RKII network [1], digital control is used to time-multiplex an analog network, thus allowing resource sharing. This

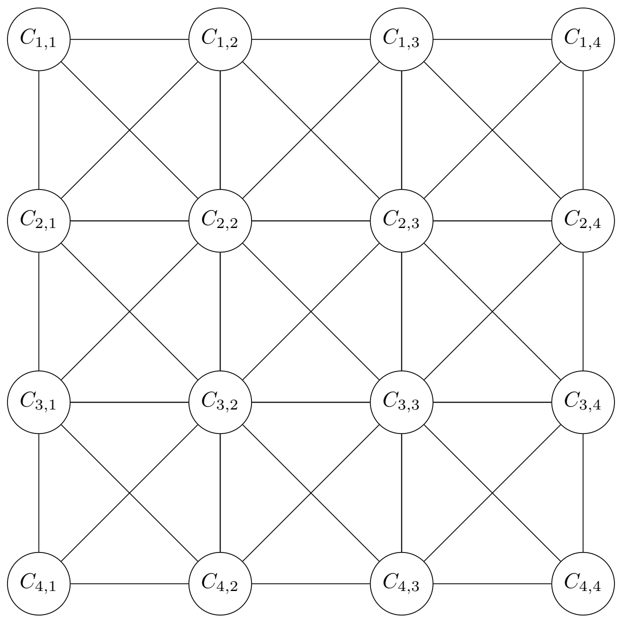


Figure 2.5: Cellular Neural Network.

allows to reduce the number of physical interconnections without affecting the number of network-level interconnections.

Other approach to reduce the burden of the interconnections is to reduce the number of interconnections in the network, and not only at the physical level. This approach is followed in cellular neural networks (CNNs) [34] [35]. Such network are inspired in cellular automata. Cells are disposed in a lattice and are connected only to their physically nearest neighbours, thus reducing the number of connections. Original CNNs were purely analog circuits. A CNN is depicted in Figure 2.5.

A comprehensive survey about hardware implementation of neural networks was presented by Misra and Saha [33]. All networks referred are radically different from Freeman's model, but their implementations face similar issues.

Despite its potential, analog hardware implementation of neural networks is still an highly unexplored field. Digital implementations are much more common nowadays. An example that has recently attracted some interest is the SpiNNaker [36], which is a massively parallel spiking neural network. It was created with the to enable real-time simulation of realistic spiking neural networks for neuroscience research. It decreases the burden of interconnection by using a central node for communication. Every node is physically connected only to the central node, and not to every node in the network. The network operation is event-oriented, and nodes communicate events via addressed packets.

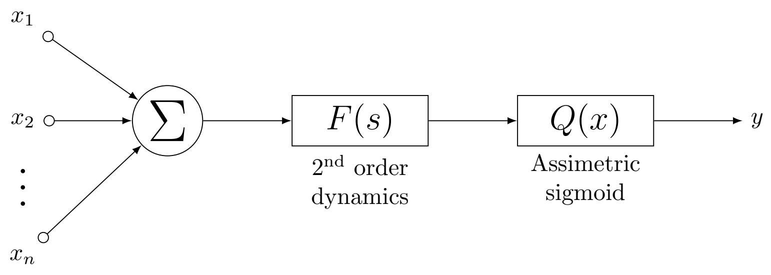


Figure 2.6: The KO.

2.4 Freeman Olfactory Neural System model

Freeman model for the neural olfactory system was introduced by Walter J. Freeman [6]. It has been developed and applied for several engineering purposes by Freeman and many other researchers.

Freeman approaches the brain at a level higher than single neurons and action potentials. This level of approach is called mesoscopic [37]. At this level, the brain is modeled using differential equations, recurring to the observation of EEG waveforms as the observable basis of study. Although accepting the role of neurons as the fundamental cells in computation in the brain, Freeman states that, due to the interrelation of many neurons in information transmission, the observation of single or few neurons is not an efficient way to explain the inherent cooperative behavior observed in neural activity. On the other hand, the observation of neural masses, treated as a continuum and not as individual cells, is much more insightful about this cooperative behavior [6].

The most basic building block of the model is the KO set, which models a set of neurons ranging from 10^3 to 10^8 with common input, common sign output, and no functional interconnections among them [6]. The KO, depicted in Figure 2.6, is similar to the cells in a continuous and deterministic Hopfield network [30]. It exhibits a summation node, a linear low-pass filter and a sigmoid nonlinearity. However, differently from the Hopfield network, the low-pass filter has a second order response, and the nonlinearity is an asymmetric sigmoid. The linear dynamics are given by (2.3), where a and b are the poles of the dynamic response. From biological evidence, $a = 220\,\text{rad/s}$ and $b = 720\,\text{rad/s}$ [6].

$$H(s) = \frac{a \cdot b}{(s+a)(s+b)} \tag{2.3}$$

The asymmetric sigmoid is depicted in Figure 2.7 and given by (2.4). Q_m is the asymptotic maximum of the sigmoid function, which is found by biological evidence and it is different from layer to layer. In Figure 2.7, $Q_m = 5$ is used, which is commonly used for the olfactory bulb [38–41].

$$Q(x) = \begin{cases} Q_m \cdot (1 - e^{-(e^x - 1)/Q_m}) & \text{, if } x > \ln(1 - Q_m \cdot \ln(1 + \frac{1}{Q_m})) \\ -1 & \text{, if } x < \ln(1 - Q_m \cdot \ln(1 + \frac{1}{Q_m})) \end{cases} \tag{2.4}$$

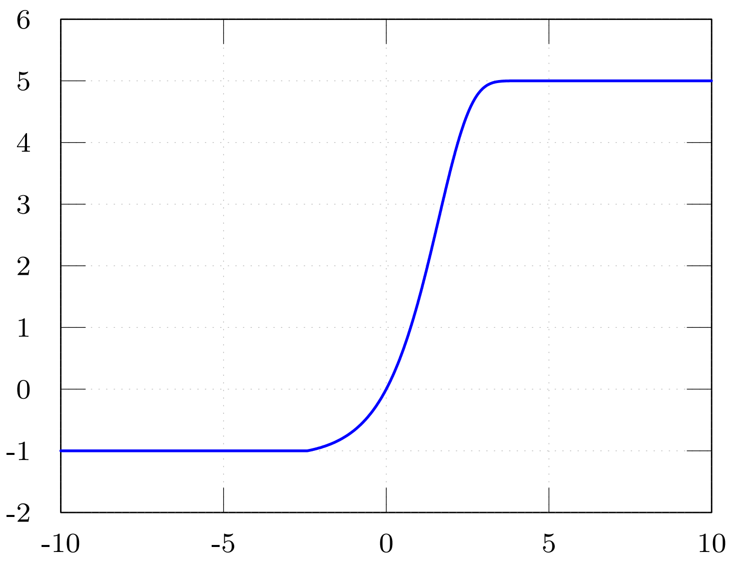


Figure 2.7: Nonlinearity in KO with $Q_m = 5$.

The output of the linear dynamics in the KO is generally known as the KO state.

Several KO interconnected with common sign feedback form a KI set. A KI set can be either excitatory or inhibitory, depending of the sign of the connection weights. In such network, each KO is connected to every other in the network and there is no self-feedback.

A KII set is as depicted in Figure 1.4a. It consists of an excitatory KI and an inhibitory KI with dense interconnection [6]. When isolated, it behaves as an oscillator controlled by the input: an input of zero results in a zero output, and a positive input results in an oscillatory response. Several KII connected form a KII network, as depicted in Figure 2.8. In such network, inputs are applied in a excitatory KO of each KII, and there are connections both among excitatory KO of each KII and among inhibitory KO.

The higher layer of the model is named KIII. This structure, depicted in Figure 2.9 [22], contains several lower lever structures and models the entire olfactory neural system.

The KII can be simplified to a cell with only two KO. This cell is the reduced KII (RKII). As well as the KII, the RKII is an oscillator controlled by the input, and both cells result in similar responses at system level [6,21]. An RKII network is formed in a similar way as the KII network, as can be seen in Figure 2.10. The RKII is significantly simpler to use than the KII, since it has less connections and, therefore, less weights to adjust.

KIII simulations have shown that its behavior successfully mimics EEG waveforms and its statistical properties. EEG and KIII waveforms are similar both when an odorant stimuli is given and in the absence of stimulation. Moreover, with parameter variation, the KIII can be adjusted to mimic both the waveforms obtained during epileptic seizures and during deep anesthesia [39].

An interesting aspect of the KIII is its chaotic dynamics, which arise from dispersive delays

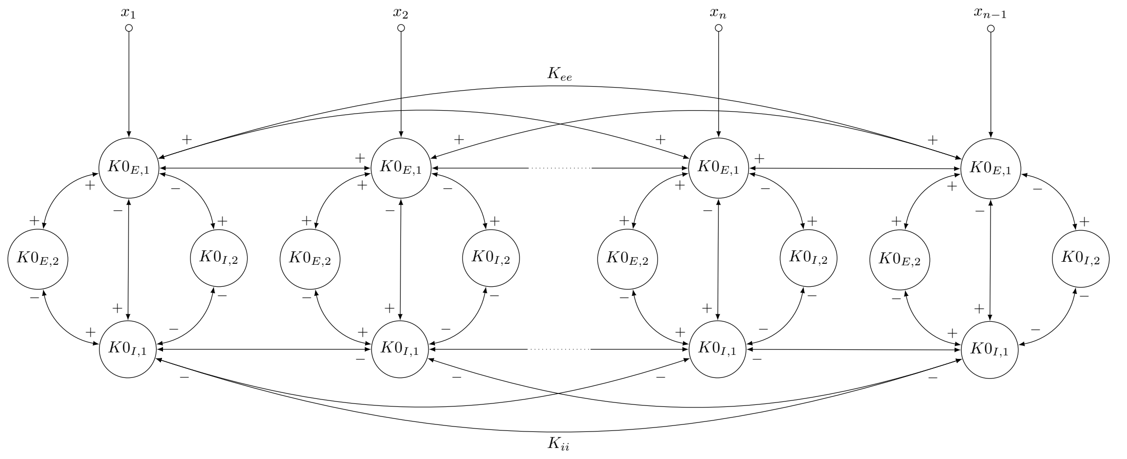


Figure 2.8: KII network.

on feedback loops. Evidence suggest that the KIII has the capacity to learn input patterns by adjusting the weights of excitatory to excitatory connections in the KII network that models the olfactory bulb [40, 42]. Every other interconnection weight is fixed. A learned pattern results in the formation of an attractor in the KIII state space. In this way, when a learned pattern is presented at the input, the KIII state converges to a near-limit cycle (low-dimension chaotic wing) that internally represents the input pattern [16,40]. After learning, a landscape of low-dimensional chaotic local basins of attraction is formed, with each basin corresponding to a learned pattern [43]. Moreover, the system has the capacity to successfully recognize a learned pattern if it is distorted or incomplete [42]. This properties are the ones of a CAM.

In the absence of an input, both the KIII and biological evidence have a high-dimensional chaotical behavior [39]. This basal chaotic state enables fast convergence to every learned state when an input is presented. When an unknown pattern is presented, a characteristic chaotic behavior arises, which suggests that the chaotic behavior of the KIII has an important role in learning [16,39,40]. These characteristics are considerably different from the ones found in common recursive neural networks such as the Hopfield model, where chaos is treated as undesirable [14,30]. In the KIII, chaos is not only an intrinsic characteristic, but it brings properties that are interesting for pattern classification, since it enables fast and ready state transition when the input changes.

Due to these interesting properties, the KIII has been tested for pattern classification application. Learning is done by adjusting excitatory to excitatory weights in the olfactory bulb, generally in a Hebbian way. Hebbian learning is a common learning rule, in which a synaptic weight is strengthened when the respective two neurons are activated by the same input [2]. The first learning rule proposed for the KIII was a modified Hebbian rule [17]. It is a binary rule - it sets a interconnection weight to a fixed high value if two inputs are active in any input pattern in the learning set and to a low value otherwise. Although quite simple and limited in the number of patterns than can be stored simultaneously, this rule has shown good results [17, 42]. A simple modification to this rule uses three weights instead of two [44]. If two KII are active simulta-

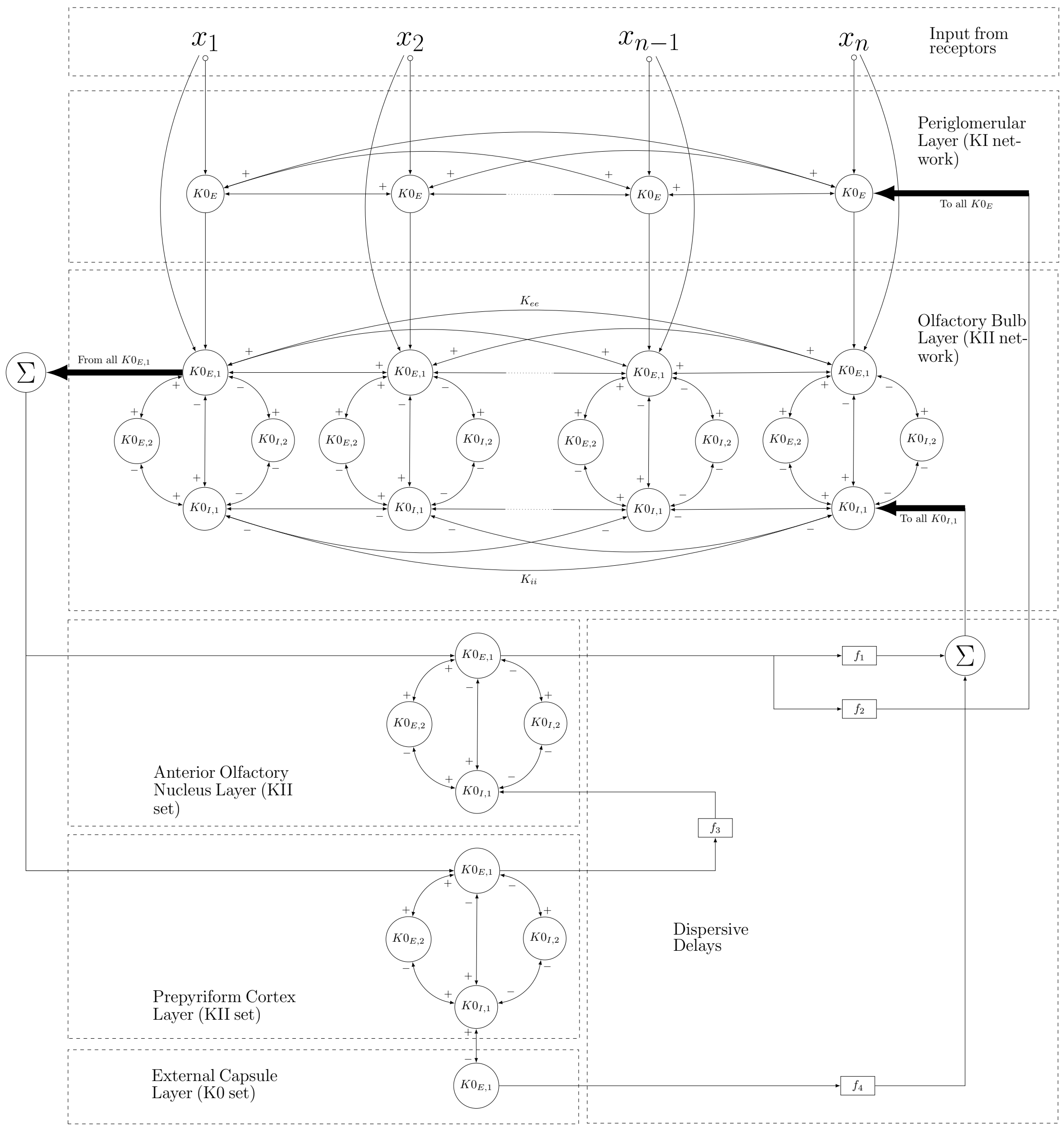


Figure 2.9: The KIII.

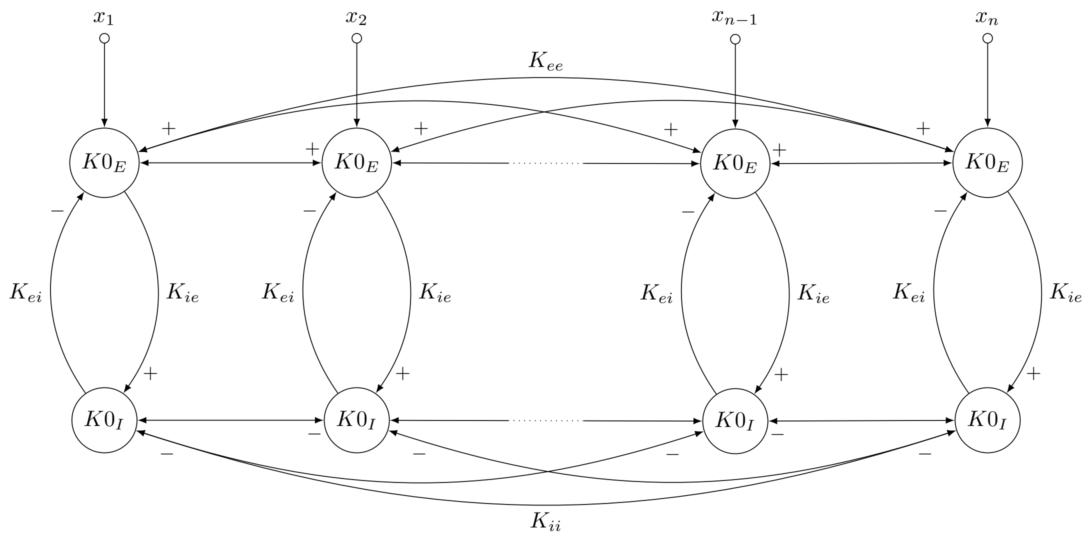


Figure 2.10: RKII network.

neously for only one input in the learning set, the weight between them is set to a high value, if they are never simultaneously active for any input they are set to a low value, and if they are simultaneously active for more than one input pattern, the weight is set to an intermediate value.

More recently, learning has been done by three processes simultaneously [43]: Hebbian, habituation and normalization. The Hebbian process may be either the binary process referred above, or a reinforcement process, in which the weights between two KII are linearly increased they are active for the same input pattern [45]. Habituation is the gradual decrease of the weights if they are not reinforced by Hebbian learning, and normalization is used to maintain stability.

Using these processes, the KIII has shown good performance in several pattern recognition applications. It has been used successfully for character classification [41, 45, 46], face recognition [47], and for classification of odorants [48, 49]. Moreover, it has shown positive results in time series prediction [50, 51].

An extension to the KIII model was proposed by Kozma *et al.* [52–54]. It includes an higher level in the hierarchy, the KIV, which models higher cognitive processes and intentionality. It is constituted by three interacting KIII. One KIII models sensory cortical functions, other KIII models the hippocampus, which is responsible for spatial and temporal orientation, and the other KIII models the midline forebrain, which is responsible for internal sensory functions. It has been used successfully as artificial brain of goal-oriented autonomous agents [52, 53, 55].

A major problem in the Freeman model is that, due to its nonlinear, and even chaotic, dynamic behavior, a mathematical analysis of its learning properties is far from straightforward. Although there already interesting results [17, 24, 40, 43, 45, 56–58], the dynamical analysis of a KIII or even a KII network is still an unsolved problem in many ways.

At the RKII set level, interesting results may be found by bifurcation analysis of an RKII set [59]. This analysis shows that, under the right choice of weights, the RKII is an oscillator controlled by the input. It has a single attractor, and, depending on the weights and the value of the input, the attractor can be either a point or a limit cycle. Naming m_1^* and g_1^* the state of the excitatory and inhibitory KO, respectively, the bifurcation occurs for [59] (K_{ei} is the inhibitory to excitatory connection weight, and K_{ie} the excitatory to inhibitory weight, as depicted in Figure 2.10):

$$|K_{ie} \cdot K_{ei}| = \frac{1}{Q'(m_1^*) \cdot Q'(g_1^*)} \cdot \frac{(a+b)^2}{a \cdot b} \tag{2.5}$$

If $|K_{ie} \cdot K_{ei}|$ is lower than this value, the attractor is stable, if its higher, the attractor is unstable. A stable attractor corresponds to convergent equilibrium point, and an unstable attractor corresponds to oscillation. For an input of 0, both m_1^* and g_1^* are 0, and $Q'(m_1^*) \cdot Q'(m_1^*) = 1$, therefore, the bifurcation point is simply [59]:

$$|K_{ie} \cdot K_{ei}| = \frac{(a+b)^2}{a \cdot b} \tag{2.6}$$

However, for inputs different than zero, the assessment must be done case by case. Based on this expressions, three different cases can be obtained by properly selecting the weights. If $|K_{ie} \cdot K_{ei}|$ is less than the bifurcation point for both zero and the value of the high input, there RKII will be always stable (there will be no oscillation). If it is lower than the bifurcation point for zero input but higher for the high input, the RKII will oscillate when the input is high, but oscillations will fade for an input of zero. At last, if $|K_{ie} \cdot K_{ei}|$ is higher than the bifurcation point for both zero and high input, it will oscillate for both cases. In this last case, however, oscillations will have different characteristics (such as phase and amplitude) in both cases. Although the Freeman model is based on operation when the attractor is stable for low input and oscillatory for high input, applications with oscillatory attractors for both cases also exhibit computational properties [60].

As seen in Figure 2.9, the olfactory bulb is modeled by a KII network. Its approximation by an RKII network is reasonable and often used [6,21]. The olfactory bulb plays an important role in odorant discrimination and in learning [16,61]. KII and RKII networks alone may also be used as CAMs [24,42,44,62], but with a significantly worse performance than the KIII [42].

Results presented for KIII performance in practical applications were obtained by computer simulations. Due to the system complexity, these simulations are highly computer intensive, making the simulation time quite long [45]. For this reason, real-time hardware implementations are demanded for more powerful applications. However, as in the general case for neural networks, the number of interconnections imposes a big problem in practical realizations of the Freeman model.

The first hardware implementation reported was a microprocessor-based RKII network with eight RKII sets [18]. The burden of interconnection was reduced by sampling and multiplexing the several RKII sets in the network.

Model discretization along with time multiplexing became the most common strategy used. It was used in the first analog VLSI implementation of a KII set [20], and later for the first analog VLSI implementation of an RKII network [21], constituted by four RKII sets. Relatively to the previous implementation, besides the size reduction, these implementations introduced the usage of filter and hold (F&H), which enables the large time constants intrinsic to the model to be practically achievable in VLSI. Later, this implementation was improved by sharing the sigmoid blocks among RKII sets [23], which decreases mismatches. This strategy was used to build a network with eight RKII sets [24].

In a radically different strategy, a modification was proposed to the model [25]. It replaced the sigmoid non-linearity by a integrate-and-fire block, transforming the KO in a spiking neuron. While this is not consistent with the biological origins of the Freeman model, the modified model dynamics are similar to the original one. Following this strategy, a single RKII set was tested on chip [25].

These implementations confirm that a discrete time VLSI implementation allows a real-time application of the Freeman model. However, in order to allow real-time usability for practical applications, an algorithm that uses the multiplexed RKII network in a KIII must be implemented, and an RKII network with enough RKII sets for practical application must be achieved.

Chapter 3

System Modeling and Architecture

The approach followed in this work is based on the strategy proposed by Tavares *et al.* [1], which decreases the burden of the interconnections between cells by the usage of time-multiplexing. Figure 3.1 shows the multiplexing architecture [19], which allows the sigmoid blocks to be shared by all the cells, hence mitigating problems due to mismatch that would likely occur if one block per cell was used. An important difference between the original and the multiplexed network model is that the latter operates in discrete time. Therefore, filters are also discrete time, and a delay was introduced in the feedback loop.

The price to pay for the multiplexed network is the need for a digital control structure, in order to properly activate the cells in their time slots. Multiplexing also enables the usage of F&H [63]. This technique consists in switching the filter *on* and *off* with a certain duty cycle, increasing the time constant of the filter by the inverse of the duty cycle. Therefore, large time constants are achieved with not prohibitively large resistors and capacitors, and with a low power consumption.

In order to further improve matching between cells, a BIST scheme [19] is also employed. This scheme selects the set of cells with the best matching (even if their parameters are not the closest to the designed ones) and selects that set to be active. Cells that significantly differ from this set are disabled. The BIST implements an offset test that checks if there are cells with significant offset, and a dynamic test that selects cells that have the closest oscillation amplitude in response to a sinusoidal input.

In this chapter, the proposed system architecture is presented. Firstly, the F&H technique is explained and modeled in Section 3.1. The operation of the multiplexed network is explained in Section 3.2. The BIST algorithm is presented in Section 3.3 and the final system architecture and the control structure are described in Section 3.4.

3.1 Filter and Hold

As stated in literature [6], the linear dynamics of the KO consists of a low pass filter with real valued poles at 220 rad/s and 720 rad/s. A realization of a filter with such low-frequency poles using common discrete time techniques such as switched-capacitor or switched-current is neither

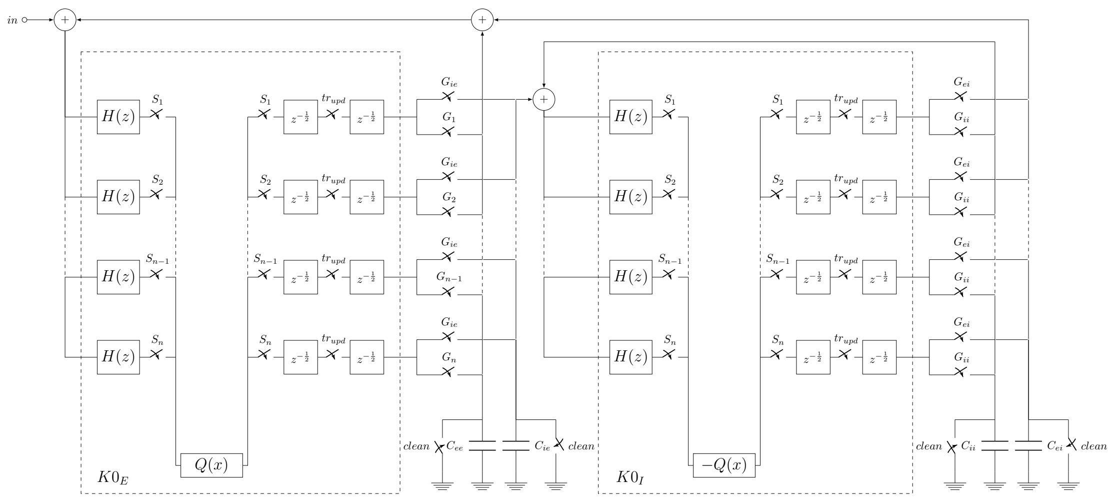


Figure 3.1: Multiplexed RKII network.

area nor power effective [64]. Filter and hold [63, 64] is a technique that consists of switching a filter with a time constant RC with a duty cycle α, as depicted in Figure 3.2. If the switch control signal Φ has a duty cycle of α, the time constant of the switched filter is multiplied by $\frac{1}{\alpha}$.

Assuming an input v_{in} sampled with period T and a filter control signal Φ with period T and duty cycle α, if the switch is closed at time nT, the voltage initially stored at the capacitor will be the one stored at the end of the previous clock round, $v_{out}[n-1]$. If the switch is kept closed for infinite time, the voltage stored at the capacitor converges to the current value of v_{in}, $v_{in}[n-1]$. Hence, the value of v_{out} up from the moment the switch is closed follows (3.1).

$$v_{out}(t) = v_{in}[n-1] + (v_{out}[n-1] - v_{in}[n-1])e^{-\frac{t}{RC}} \tag{3.1}$$

If the switch is opened at time α and remains closed until the end of the clock cycle, the voltage v_{out} at time T will be the same as at time αT, resulting in (3.2) [63].

$$v_{out}[n] = v_{out}(T) = v_{out}(\alpha T) = v_{in}[n-1] + (v_{out}[n-1] - v_{in}[n-1])e^{-\frac{\alpha T}{RC}} \tag{3.2}$$

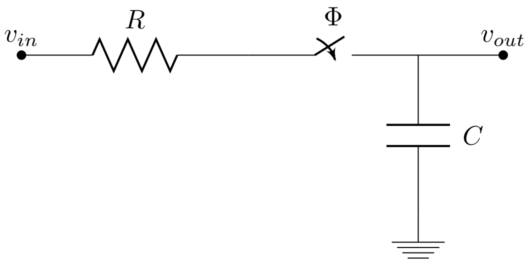


Figure 3.2: Simple Filter and Hold RC circuit.

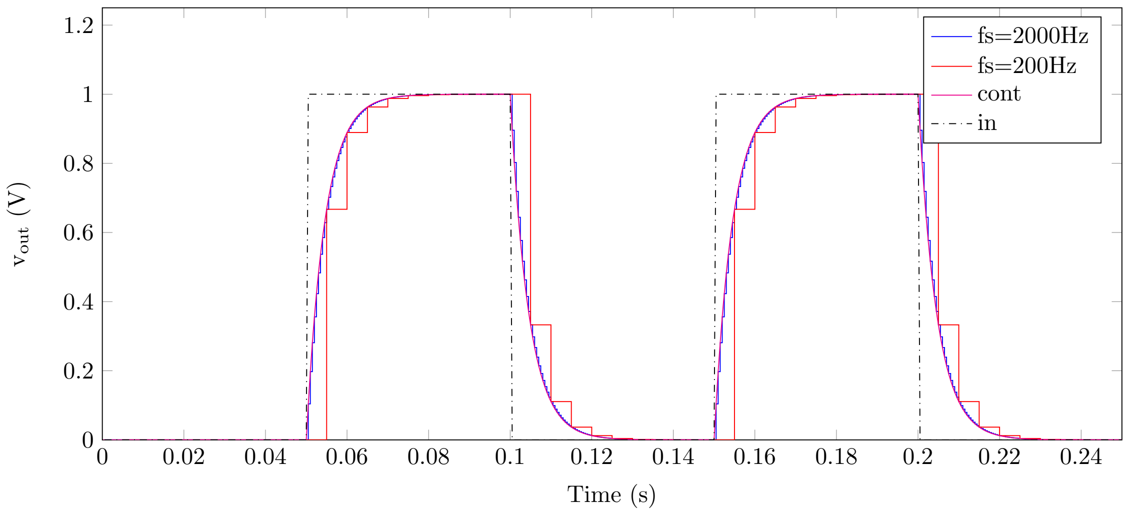


Figure 3.3: Single pole Filter and Hold model simulation.

The transfer function for the filter is, therefore, given by (3.3) [63].

$$H(z) = \frac{(1 - e^{-\frac{aT}{RC}}) \cdot z^{-1}}{1 - e^{-\frac{aT}{RC}} \cdot z^{-1}} \tag{3.3}$$

As desired, the time constant is increased by a factor of $\frac{1}{a}$, thus by scaling $R \cdot C$, high time constants are achievable from smaller values of R and/or C.

Figure 3.3 shows the output of an ideal F&H block, simulated in Matlab using (3.2), with $\frac{aT}{RC}$ =220 rad/s. As seen in the figure, the output of the filter is equal to a sampled version of the output of a continuous time RC filter with the time constant scaled by $\frac{1}{\alpha}$.

Another property of the F&H, also made evident by (3.3), is that the pole and zero positions relative to the input bandwidth is independent of the sampling frequency. As long as the filter duty cycle is kept constant, adjusting the sampling frequency affects time resolution but not the relative pole and zero positions [64].

The filter in the RKII has two real valued poles, one at 220 rad/s and one at 720 rad/s. In this work, the filter is implemented using a switched Gm-C filter, depicted in Figure 3.4 [63]. The transconductance amplifiers shown are equal and their transconductance is G_m. The difference equation for such filter was derived and is presented in (3.4). v_m is the voltage stored by C_1, τ_1 is the time constant of the first stage $\frac{C_1}{G_m}$ (not considering the F&H scaling factor), and, equivalently, τ_2 is $\frac{C_2}{G_m}$. a and b are, respectively, $e^{-\frac{\alpha T}{\tau_1}}$ and $e^{-\frac{\alpha T}{\tau_2}}$. Again, α is the F&H duty cycle and T is the sampling period.

$$\begin{cases} v_m[n] & = & v_{in}[n-1] \cdot (1-a) + v_m[n-1] \cdot a \\ v_{out}[n] & = & v_{in}[n-1] \cdot \frac{1}{\tau_1 - \tau_2} \cdot (\tau_1 \cdot (1-a) - \tau_2 \cdot (1-b)) \\ & & +v_{out}[n-1] \cdot b \\ & & +v_m[n-1] \cdot \frac{\tau_1}{\tau_1 - \tau_2} \cdot (a-b) \end{cases} \tag{3.4}$$

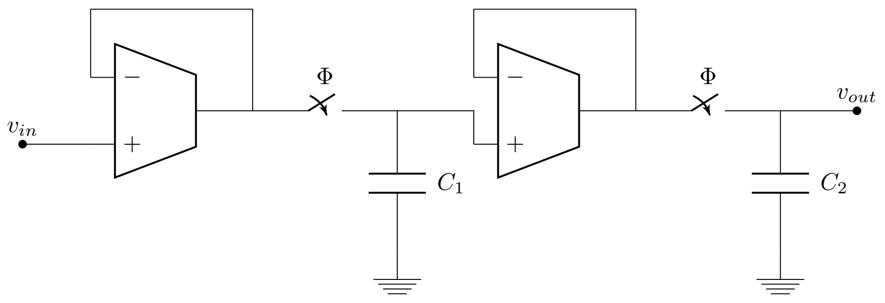


Figure 3.4: Switched Gm-C filter schematics.

The response for a filter with original poles at 220 krad/s and 720 krad/s and scaled by $\alpha = 0.001$ are shown in Figure 3.5 and compared with a continuous time response of a filter with poles at 220 rad/s and 720 rad/s. As we can see, the same properties observed in the single pole case are held, and both time constants are scaled by a factor of $\frac{1}{\alpha}$.

A filter with the same parameters as above was modelled in Cadence Virtuoso® with ideal elements and compared with a continuous time RC filter, also modelled in Virtuoso with ideal elements. Figure 3.6 shows the response of both filters to a square wave.

The only difference to the results obtained with the difference equation model is that the output changes from $v_{out}[n-1]$ to $v_{out}[n]$ at time $(n-1)T+\alpha$, because the filter is active at the beginning of the clock period. Therefore, the output changes its value from $t=(n-1)T$ to $t=(n-1)T+\alpha$ and holds this value until $t=nT$. Therefore, at $t=nT$, the value of the output $v_{out}[n]$ is still equal to the output of the continuous time filter. This occurs independently of the phase of the filter control signal.

The frequency response of F&H filters with the same poles and duty cycle as the cases above, and with sampling frequencies of 200 Hz and 2000 Hz were obtained in Virtuoso using a SpectreRF Periodic AC analysis and the result is shown in Figure 3.7. The figure also shows the frequency response of a continuous time filter with poles at 220 rad/s and 720 rad/s. F&H achieves an identical frequency response as the continuous time case up to near half the sampling frequency.

3.2 The Multiplexed RKII Network

The structure of the multiplexed network is presented in Figure 3.1. Each row inside dashed rectangles corresponds to a KO, being that the dashed rectangle on the left corresponds to excitatory KO cells, and the rectangle on the left corresponds to inhibitory KO cells. The switches at the end of each row together with the capacitors shared by all rows establish the interconnections. The output of the $z^{-\frac{1}{2}}$ memories is a current signal, and the interconnection weight is controlled by the fraction of time that the corresponding switch is conducting, according to (3.5), where C is the interconnection capacitance and Δt is the amount of time that is used for charging the capacitor in

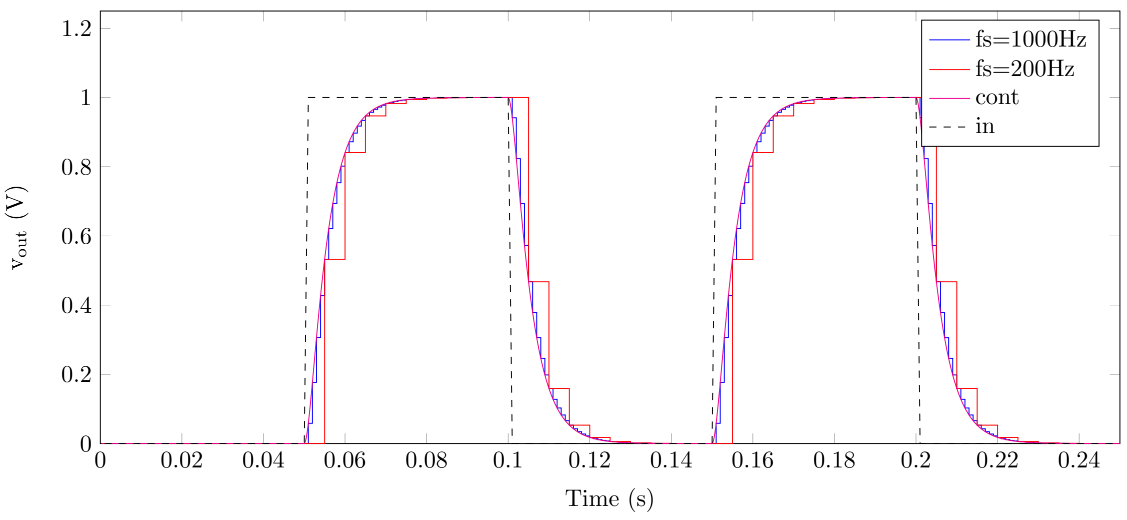


Figure 3.5: Two pole Filter and Hold model simulation.

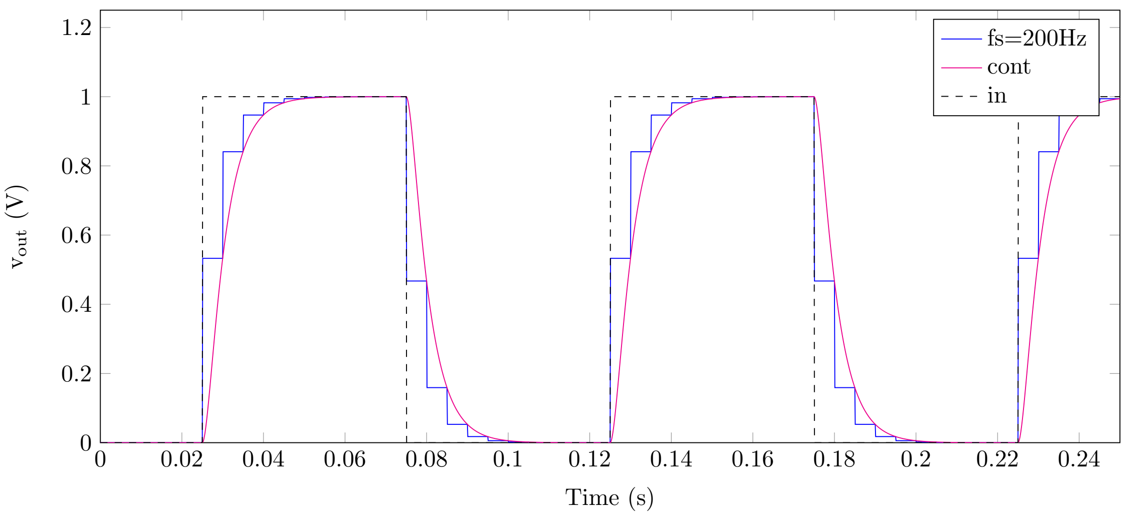


Figure 3.6: Two pole Filter and Hold Cadence Virtuoso simulation.

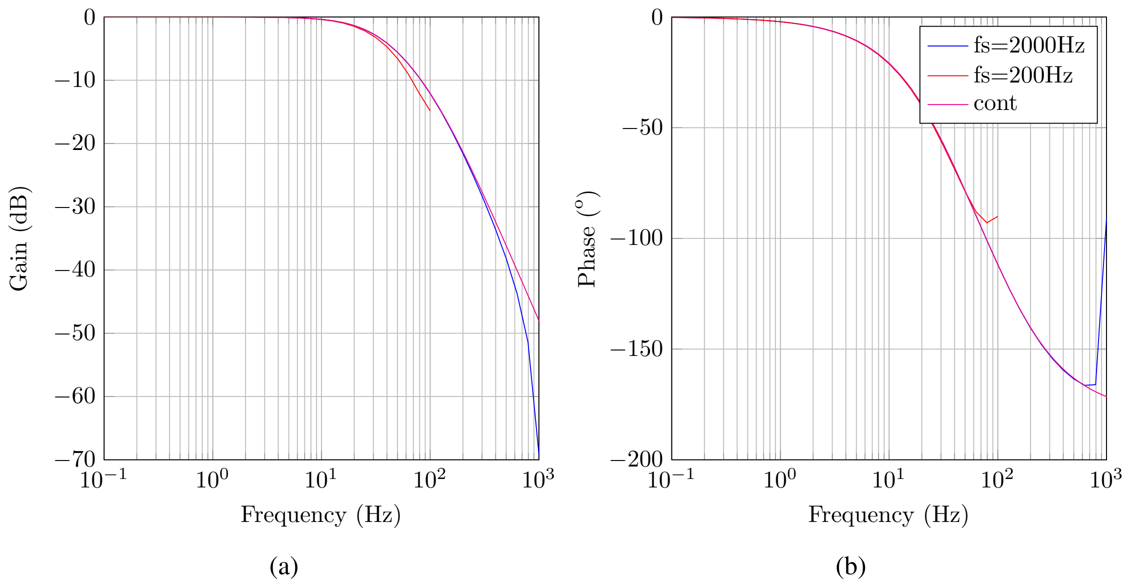


Figure 3.7: Filter and Hold frequency analysis and comparison with continuous time filter (a) magnitude resuponse, (b) phase response.

each slot.

$$K = \frac{\Delta t}{C} \quad [\mathrm{V/A}] \tag{3.5}$$

Each KO cell is controlled by a signal S_i. Time multiplexing is achieved by activating the S_i signals sequentially and separately for each i. A RKII is formed by two KO (one inhibitory and one excitatory) controlled by the same S_i signal. Moreover, the filter for each KO is also activated when the respective S_i signal is active, so that the input of the KO is sampled at the right time. The signal tr_{upd} ensures simultaneous update of the outputs of all cells. This is required for coherent operation of the network, since it ensures that in every round, every cell is using the output computed in previous round for every interconnection.

In the original algorithm [19], the interconnections operated as follows: in its own slot, the excitatory KO charges the capacitor responsible for the excitatory to inhibitory interconnection (C_{ie}) and the inhibitory KO charges the capacitor responsible for the inhibitory to excitatory interconnection (C_{ei}), therefore establishing a loop between the excitatory KO and its respective inhibitory KO (forming an RKII). In every other slot, a KO charges the capacitor responsible for connections between several RKII (either C_{ee} or C_{ii}). This is done simultaneously by every RKII cell other than the owner of the current time slot, with the amount of time in which the capacitors are being charged adjusted individually for each cell. Therefore, at the end of this phase, the voltage in the capacitor is a weighted sum of the outputs, and this will serve as the input for the owner of the slot. After each slot, charge in the capacitors must be cleared for correct operation in the following slot, requiring extra switches and control signals to short-circuit the capacitors to ground.

In this work, a slight modification is proposed to the algorithm: the number of interconnection capacitors is reduced from four to two - one for interconnection with the excitatory KO cells,

C_e, and one for interconnection with the inhibitory KO cells, C_i. C_e is charged by the output of the inhibitory KO in its own slot (inhibitory to excitatory connection) and by the output of the excitatory KO of the remaining cells outside their slots (excitatory to excitatory connection), therefore replacing both C_{ei} and C_{ee}. Likewise, C_i replaces C_{ie} and C_{ii}. Moreover, the input is added by current integration in C_e - a constant current is applied to C_e for a defined time interval, and the value of the input signal is given by $v_{in} = \frac{i_{in}\Delta t}{C_e}$. This modifications allow C_e to be directly the input of the filters in the excitatory side and C_i in the inhibitory side, overcoming the necessity of analog voltage adders that would be required otherwise.

The network operates as follows: after the activation of the cell clock, weights and the input are integrated in the capacitors, as described above. After, the voltage at the capacitors remains constant, and the filters of current RKII are activated, updating the state. The RKII state is the set of values comprised by the output of the filters (i.e., the state) of both KO. In this document, the state of the inhibitory KO is shortly referred to as inhibitory state. Likewise, the state of the excitatory KO is referred to as excitatory state. During the slot of an RKII cell, the output of its filters are connected to the respective sigmoid block and the output of the sigmoid is connected to its respective memory input (as depicted in Figure 3.1), causing the RKII outputs to be the memory inputs. At the falling edge of the cell clock, the memories store the values at their inputs. This is iterated for every RKII cell. Between the slots of two cells, the interconnection capacitors are cleaned, and between two rounds, the second memory stage of every KO is updated, so that every KO outputs the value computed in the previous round.

The computation of the interconnections in current mode, as proposed in [19], is a significant change relatively to previous implementations, and allows a scalable implementation of the network.

In the following subsections, the multiplexed network is modeled, and simulation results of these models are presented and compared with the original continuous time system.

3.2.1 RKII Set Simulation

In order to compare the discrete time model implemented in this work with the original Freeman model, both were implemented. The original was implemented in both MathWorks Simulink® and Virtuoso, and the discrete time RKII model was implemented in Virtuoso. Virtuoso models use ideal circuit elements and VerilogA blocks for the sigmoid and for control elements. The block diagram of the discrete Virtuoso model is the same as of that later implemented at transistor level, and the control signals are generated by a TCL script. The script reads a configuration file and generates the respective control signals. Moreover, a difference equation was derived for the discrete time system and implemented in Matlab.

Figure 3.8 shows the response of the RKII with $K_{ei} = 1.3$ and $K_{ie} = 3.8$ when the input is a square wave with a period of 1 s and duty-cycle of 0.5 s, varying between 1 V and 0 V. As we can see in Figure 3.8a, when the input is 1 V, the excitatory state shows an oscillatory behavior, whereas oscillations fade when the input goes to 0 V. The same behavior is observed for the inhibitory state. It is possible to observe that the results obtained in Simulink and in Virtuoso with

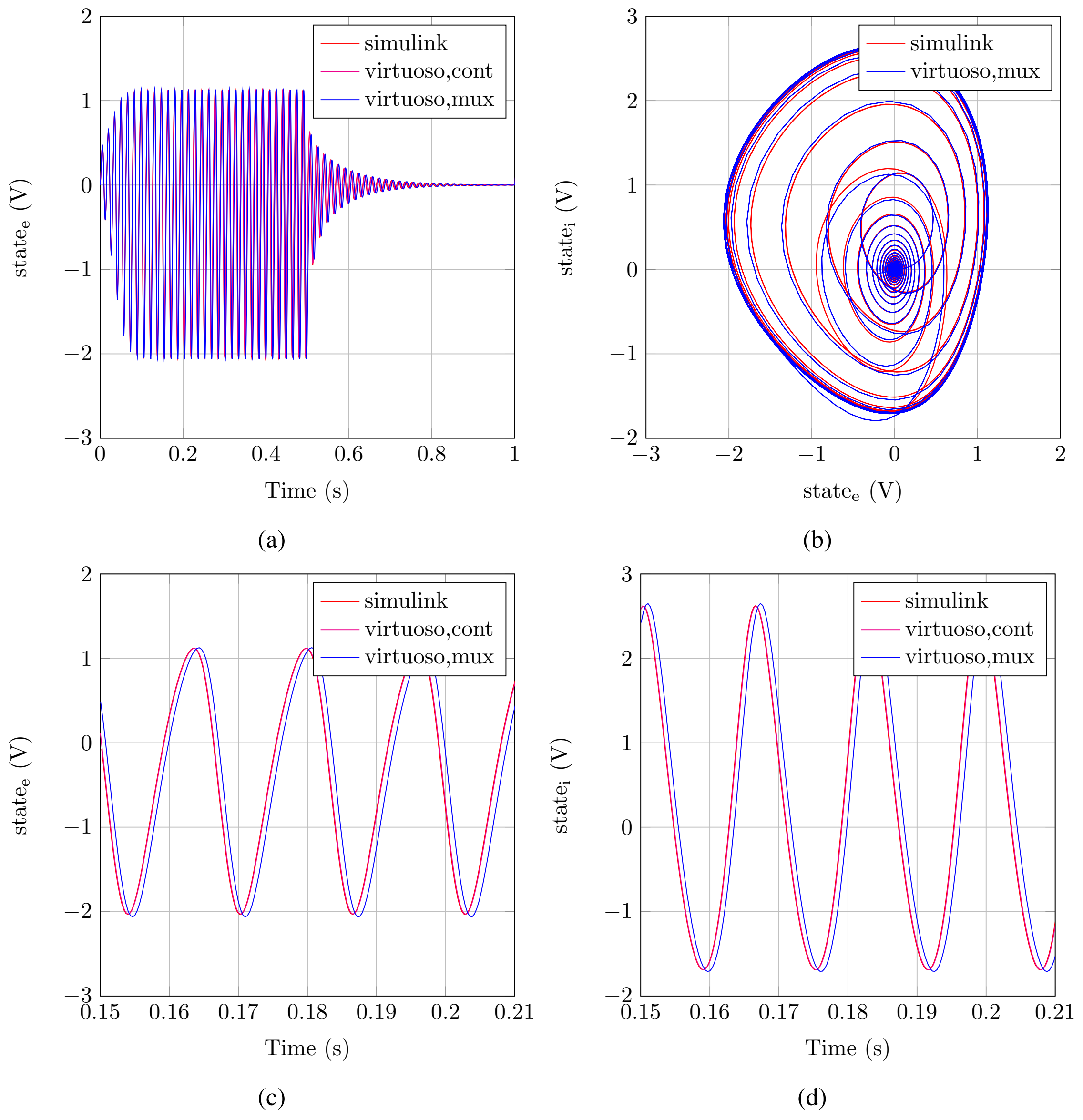
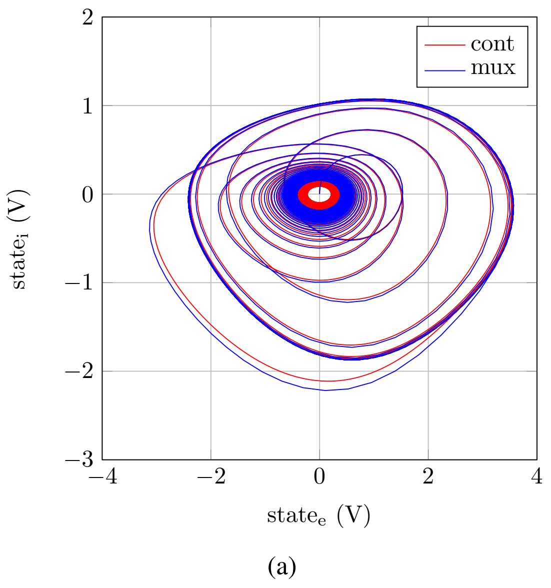


Figure 3.8: Plots for one cell simulation in Virtuoso, Simulink and in Virtuoso in a multiplexed network. (a) shows the time response of the excitatory state, (b) shows the phase portrait, (c) shows oscillations in the excitatory state when the input is high, and (d) shows oscillations in the inhibitory state when the input is high.

the original model are exactly the same, and the results for the multiplexed model are very similar to them. When the input is high, the frequency of oscillation is around 61.4 Hz. In Figure 3.8b, the phase portrait obtained for the continuous model and for the discrete time model is shown. When the input is high, both models converge to a similar limit cycle, and when the input is low, both models converge to (0,0). The oscillatory response of the excitatory and inhibitory state when the input is high is better observed in figures 3.8c and 3.8d. Oscillations are equal for both continuous models, and there is a small phase difference and a small difference in amplitude between the continuous and the discrete time models.



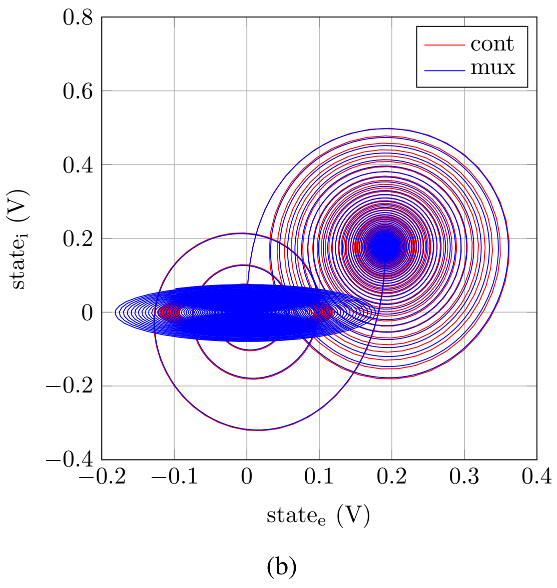


Figure 3.9: Phase portraits for a single cell with: (a) $K_{ie} = 5.6$ and $K_{ie} = 1$, and (b) $K_{ie} = 1$ and $K_{ei} = 4$.

3.2.1.1 Influence of the Connection Weights

In order to test how the weights influence the behavior of the models, and to compare this influence to the results predicted by bifurcation analysis [59], different values for the weights were tested. Figure 3.9 shows the phase portrait obtained for a square wave input with a high voltage of 1 V and a low voltage of 0 V. In Figure 3.9a, $K_{ie} = 5.6$ and $K_{ei} = 1$ were used, and in Figure 3.9b $K_{ie} = 1$ and $K_{ei} = 4$ were set instead. According to results obtained by Xu and Principe [59], the former case should result in unstable equilibrium points for both high and low values of the input, and the latter should result in stable equilibrium points for both high and low values of the input. This corresponds to the observed results. We have, thus, observed all three types of dynamic behavior of the RKII: two different point attractors for the two values of the input (Figure 3.9b), two limit cycle attractors (Figure 3.9a), and point attractor for low input but limit cycle for high input (Figure 3.8).

3.2.1.2 Influence of the Sampling Frequency

In previous simulations, the sampling frequency used for the discrete time model was 62.5 kHz, which is not feasible for the practical implementation in this work. System dynamics are dependent on the sampling frequency, and reducing the sampling frequency causes larger differences between the discretized and the original models [44].

However, similar behavior can be obtained by adjusting interconnection weights. In Figure 3.10, the response for a square wave input with high value of 1 V and low value of 0 V is presented. In figures 3.10a and 3.10b, we can observe that for $K_{ie} = 3.8$ and $K_{ie} = 1.3$ the attractor is still a limit cycle for a high value of the input, but for a low value it is a limit cycle, whereas it

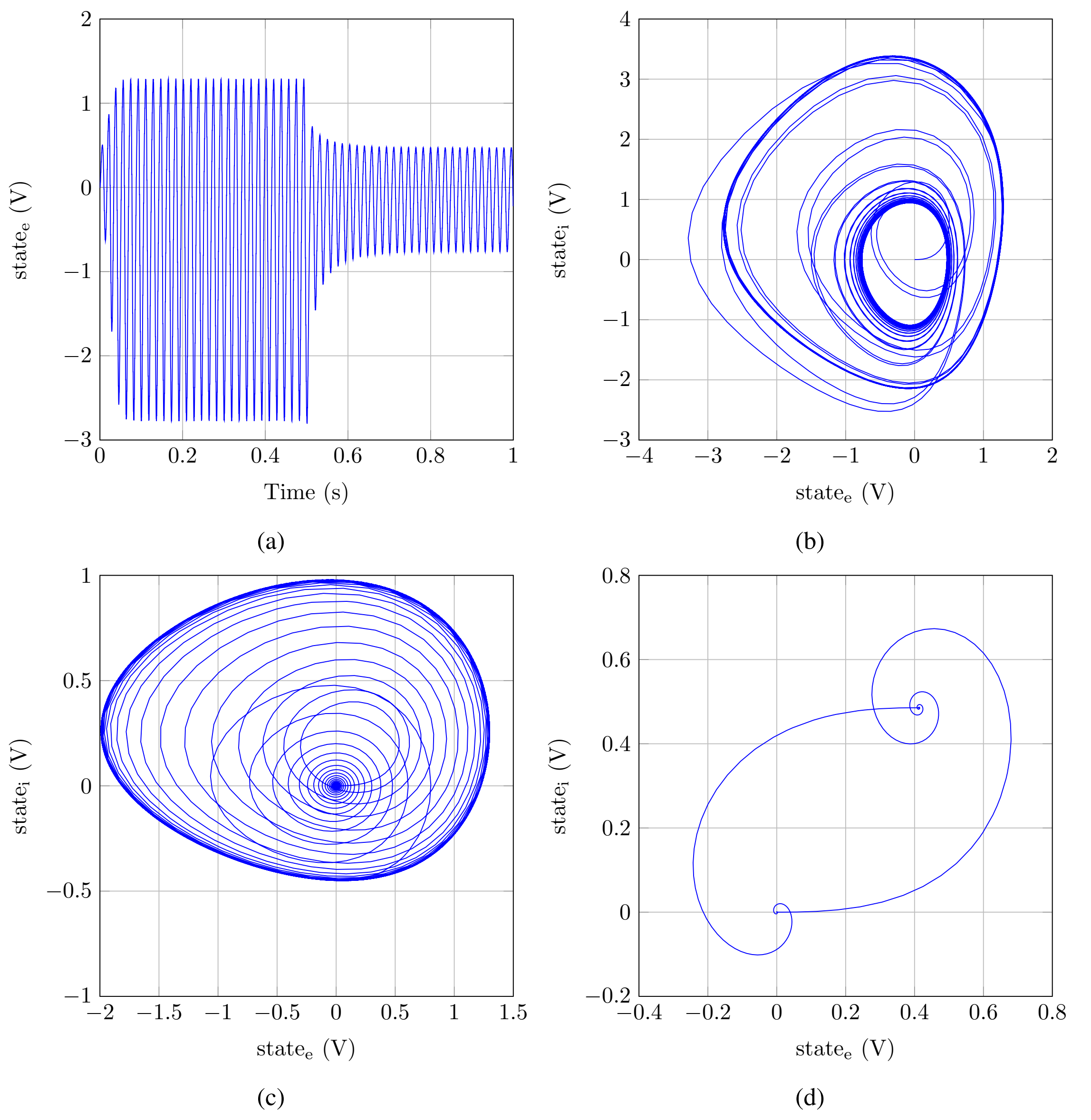


Figure 3.10: Dynamic response to a square wave for f_s =2 kHz (a) Time response when $K_{ie} = 3.8$ and $K_{ie} = 1.3$, (b) phase portrait for $K_{ie} = 3.8$ and $K_{ie} = 1.3$, (c) phase portrait for $K_{ie} = 1$ and $K_{ei} = 4$, and (d) phase portrait for $K_{ie} = 1$ and $K_{ei} = 1$.

was a point attractor in the continuous model. In Figure 3.10c we see that for $K_{ie} = 1$ and $K_{ie} = 4$, the system has now a limit cycle attractor for a high input, whereas it was a point attractor in the continuous model. For a low input, the equilibrium point is stable, like in the continuous model. In Figure 3.10d, we see a case in which there are one point attractor for each value of the input. For this case, $K_{ie} = 1$ and $K_{ie} = 1$ were used.

3.2.1.3 Difference equation model

In order to further validate the discrete time model, a difference equation was derived and simulated in Matlab. Figure 3.11 shows a diagram block of the discrete time RKII set. In the figure,

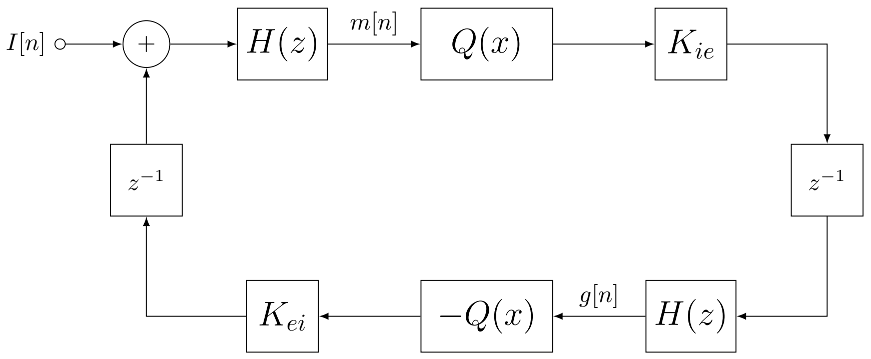


Figure 3.11: Discrete time RKII set.

$I[n]$, $m[n]$ and $g[n]$ are the input, the excitatory state and the inhibitory state, respectively. The difference equation is given in (3.6). $m'[n]$ and $g'[n]$ are intermediate states at the excitatory and inhibitory KO, respectively. Filter and Hold difference equations, (3.4), were used to derive (3.6).

$$\begin{cases} m[n] &= b \cdot m[n-1] \\ & +\frac{\tau_1}{\tau_1-\tau_2} \cdot (a-b) \cdot m'[n-1] \\ & +\frac{\tau_1 \cdot (1-a) - \tau_2 \cdot (1-b)}{\tau_1-\tau_2} \cdot (I[n] + Q(G_{ie} \cdot g[n-1])) \\ m'[n] &= a \cdot m'[n-1] \\ & +(1-a) \cdot (I[n] + Q(G_{ie} \cdot g[n-1])) \\ g[n] &= b \cdot g[n-1] \\ & +\frac{\tau_1}{\tau_1-\tau_2} \cdot (a-b) \cdot g'[n-1] \\ & +\frac{\tau_1 \cdot (1-a) - \tau_2 \cdot (1-b)}{\tau_1-\tau_2} \cdot (Q(G_{ei} \cdot m[n-1])) \\ g'[n] &= a \cdot g'[n-1] \\ & +(1-a) \cdot (Q(G_{ei} \cdot m[n-1])) \end{cases} \tag{3.6}$$

Figure 3.12 shows the comparison between the discrete time model simulated in Virtuoso and the result of the difference equations in (3.6). We can see that the results for both models are practically the same. However, the Matlab implementation of the difference equations is much less time consuming than the Virtuoso simulation.

3.2.2 RKII Network Simulation

In order to further compare the behavior of the ideal multiplexed network with the difference equations of the ideal discretized model, a network with 16 RKII sets was simulated. The multiplexed network was simulated in Virtuoso, with the same block diagram as referred in 3.2.1, with control signals generated using the same script.

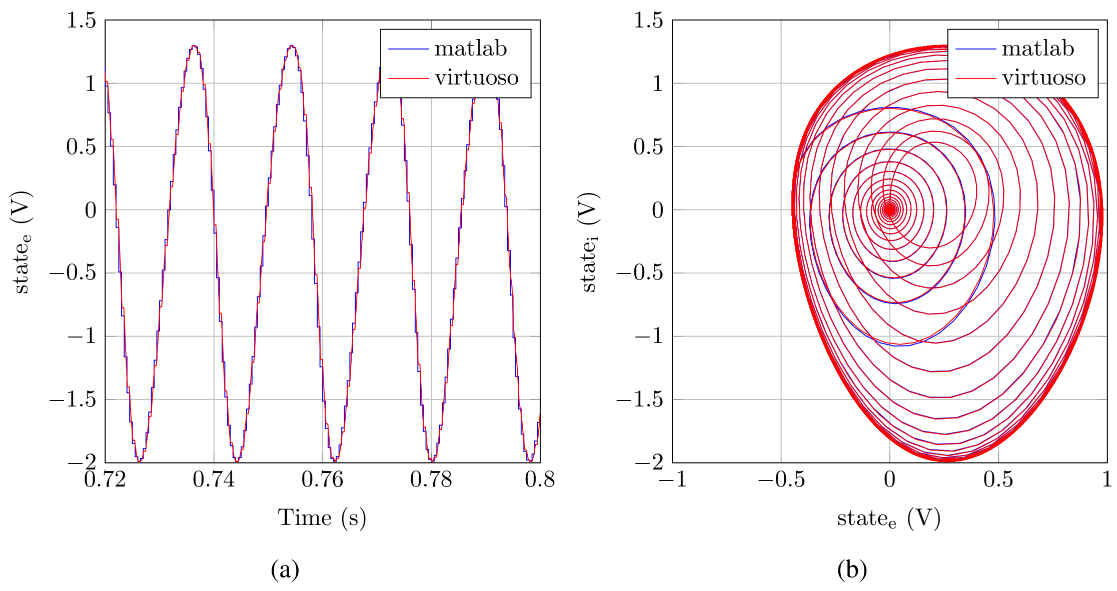


Figure 3.12: Comparison between the discrete time model in Virtuoso and the difference equations in (3.6) with $K_{ie} = 1$ and $K_{ie} = 4$. (a) Oscillations when the input is high, and (b) phase portrait.

The network was trained using a modified Hebbian rule [17] to store two patterns - [1 1 1 0 0 0 0 1 0 0 0 0 1 1 1 1] and [0 0 0 1 1 1 0 0 1 0 1 0 0 0 0]. The weights used were $K_{ie} = 2.2$, $K_{ei} = 1.8$, $K_{ii} = 0.007$, and K_{ee} is either 0.08 ($K_{ee,h}$) or 0.007 ($K_{ee,l}$), depending on training.

The first pattern was applied to the input, but with the third and fourth positions swapped - [1 1 0 1 0 0 0 1 0 0 0 0 1 1 1 1]. The excitatory state of all sets is shown in Figure 3.13. In the figure, blue curves were obtained directly from difference equations and red curves were obtained by simulation of the multiplexed network using ideal blocks in Virtuoso. We can see that both models match.

Regarding the behavior of the network, it can be observed that sets that are active in the first pattern oscillate with higher amplitude. The third set, that received a low input, oscillates with a similar amplitude as the remaining sets in the pattern, and with the same frequency but with a small phase error, as shown in figures 3.14a and 3.14c. Despite having a high input, the forth oscillates with less than half the amplitude of the sets of the first pattern and with different frequency. This may be observed in figures 3.14a and 3.14d. The remaining sets oscillate with much smaller amplitude and with different frequency, as shown in figures 3.14e and 3.14f.

Even with a distorted version of a learned pattern, RKII sets that are active in the learned pattern most similar to the input synchronize and oscillate with higher amplitude, while other sets are not able to synchronize. This simple example illustrates the behavior of the RKII network as a CAM.

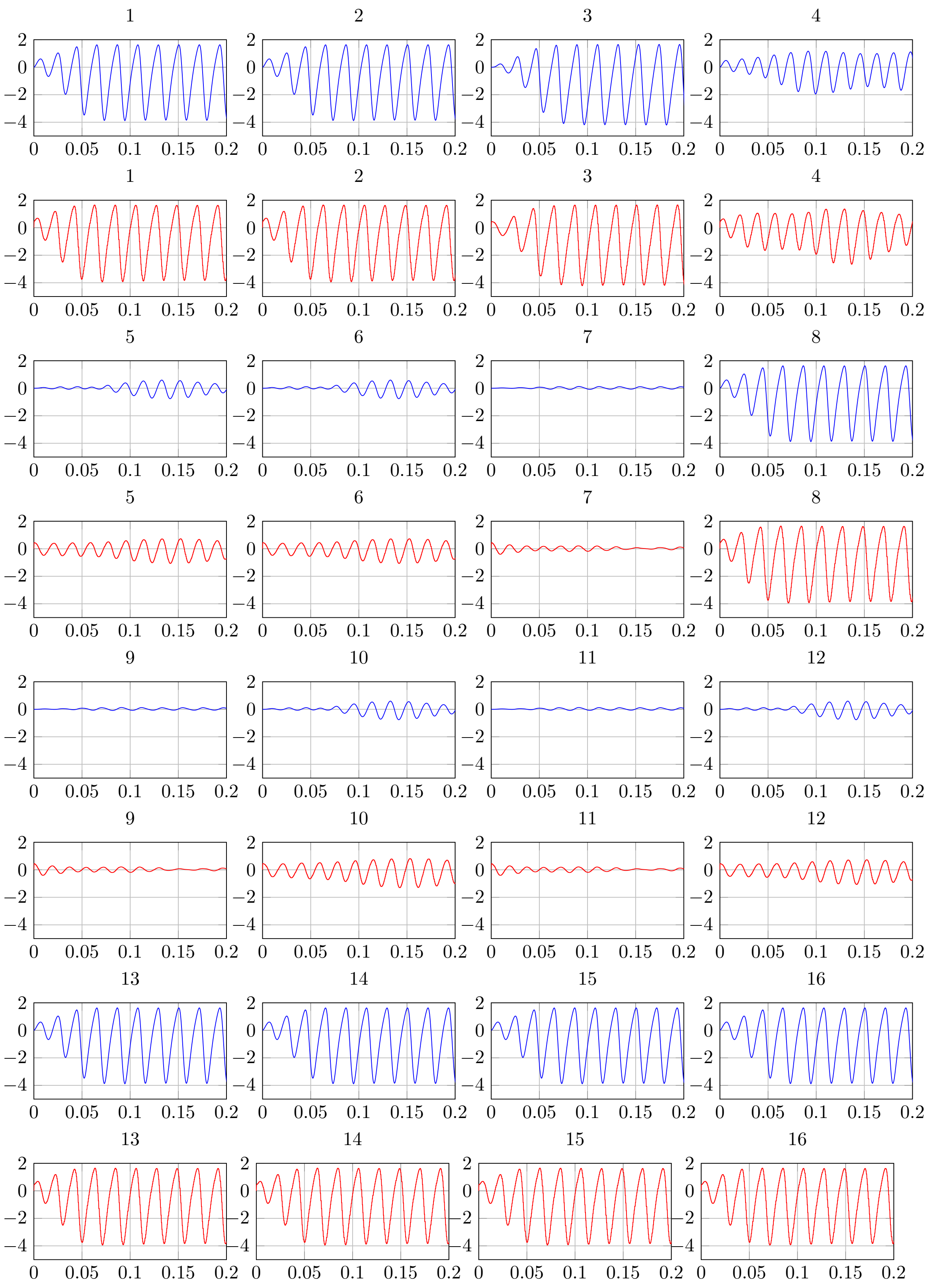
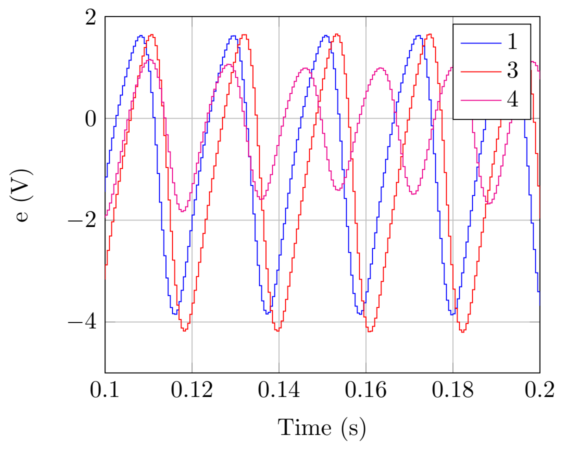
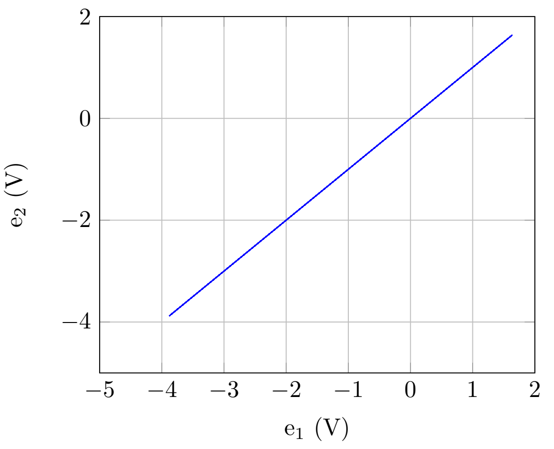


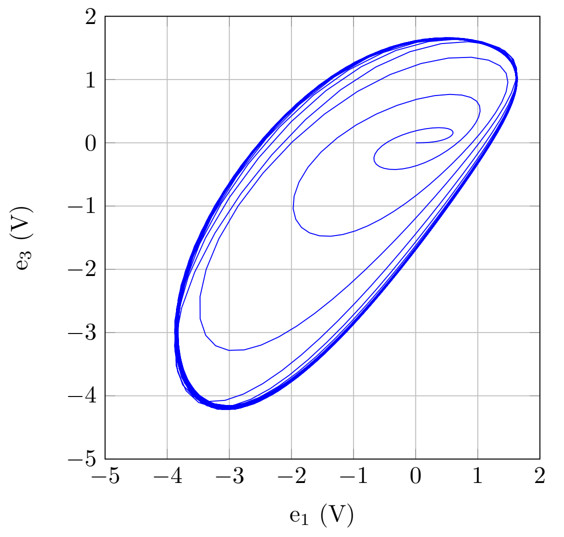
Figure 3.13: Excitatory state of every RKII set in a network of 16 sets. Blue curves were obtained directly from difference equations and red curves were obtained in Virtuoso.



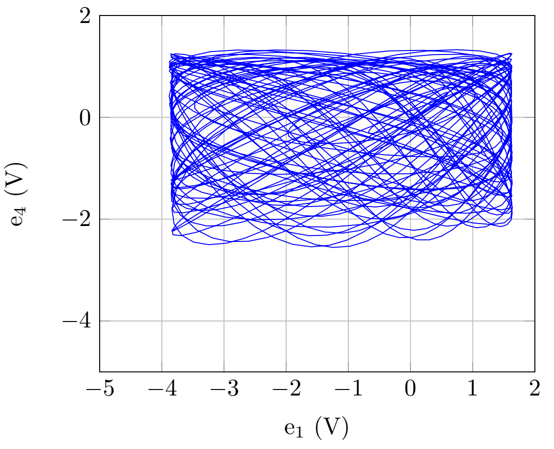
(a)



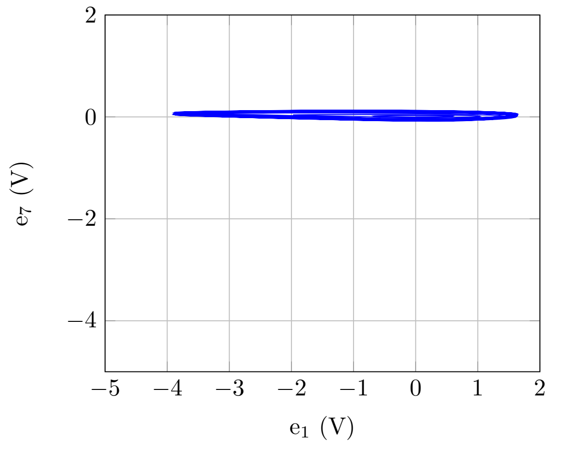
(b)



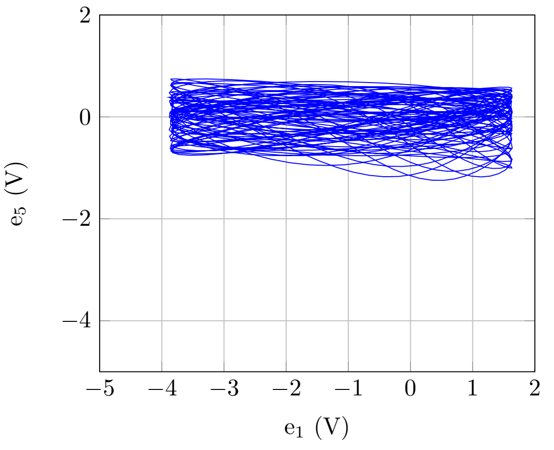
(c)



(d)



(e)



(f)

Figure 3.14: Simulation results of an RKII network with 16 sets. (a) oscillations in sets 1, 3 and 4 (b) state of set 1 versus state of set 2 (c) state of set 1 versus state of set 3 (d) state of set 1 versus state of set 4 (e) state of set 1 versus state of set 7 (f) state of set 1 versus state of set 5.

3.3 BIST

Because neural networks are redundant and fault tolerant, common BIST strategies may be ineffective [65]. A BIST scheme for RKII cells [19] is implemented in this work. It operates over a single RKII cell at each time and detects parametric differences. The BIST implements two tests: an offset test and a dynamic test. The same strategy is used in both cases and the output is a list of selected cells - L_{sel}.

The goal of the BIST is to select and build a list of cells with the closest possible parameters, ensuring optimal matching among cells. The algorithm that finds this list is based on gradient ascent. The algorithm is iterative, and in every iteration a list of selected cells is produced. $L_{sel}[n]$ is the list at iteration n. For the offset test, cell inputs are shorted to ground, and the feedback loop is opened - there is connection from the excitatory KO in the cell being tested to its respective inhibitory KO, but there is no other connection active. Initially, all cells are added to the selected list, and then, the average offset among cells in the list is computed. Next, the algorithm sequentially computes the offset of every cell in the network and compares its value to the average of the selected list. If it does not differs more than a given tolerance, the cell is added to the selected list at the current iteration, $L_{sel}[i]$, and cells excluded from $L_{sel}[i]$ signifies that their offset differs from the average more than a certain tolerance. This is iterated until it $L_{sel}[i] = L_{sel}[i-1]$. The dynamic test is similar, but a sinusoidal signal is applied to the cells input, and a peak detector is added in the path between the filter output and the sigmoid input of the excitatory KO, so that the variable under test is the filter output amplitude. Moreover, the dynamic test acts over the list selected by the offset test, meaning that every cell that was excluded from that list will never be considered for inclusion by the dynamic test. A checker block, based on a comparator, and peak detector are the only analog blocks required by the BIST. The checker output is used directly by the BIST digital control structure. The offset test is described in Algorithm 1 and the dynamic test in Algorithm 2.

```
$L_{sel}[0]$ = all cells;
i=0;
repeat
    i++;
    $L_{sel}[i]$ = { };
    compute average_offset for $L_{sel}$[i-1];
    for every cell in the network do
        compute cell_offset;
        if abs(cell_offset-average_offset) < tol then
            add cell to $L_{sel}[i]$;
        end
    end
until $L_{sel}[i] = L_{sel}[i-1]$;
```

Algorithm 1: BIST offset test algorithm.

```
$L_{sel}[0]$ = all cells selected by offset test;
i=0;
repeat
    i++;
    $L_{sel}[i]$ = { };
    compute average_amplitude for $L_{sel}$[i-1];
    for every cell selected by offset test do
        compute cell_amplitude;
        if abs(cell_amplitude-average_amplitude) < tol then
            add cell to $L_{sel}[i]$;
        end
    end
until $L_{sel}[i] = L_{sel}[i-1]$;
```

Algorithm 2: BIST dynamic test algorithm.

3.3.1 BIST Algorithm Simulation

The BIST algorithm was modeled and simulated in Matlab, using randomly generated values for offsets and amplitudes. Figure 3.15 shows the simulation results. Histograms show the value of the parameter being tested (either offset or amplitude) in the x-axis. The first seven histograms correspond to the offset test. Green bars are the selected cells at each iteration and white bars the remaining cells. The last five figures correspond to the dynamic test. Blue bars represent the selected cells at each iteration, green bars the cells selected by the offset test but not by the dynamic test, and white bars are the cells excluded by the offset test, that are not considered for the dynamic test.

3.4 Network implementation

Figure 3.16 shows the structure of the network implementation with the analog BIST blocks and with all the control signals displayed.

Control signals are generated by a control structure, which has a global and a distributed component. The global structure generates signals that are directly used by shared blocks, or processed by the distributed structures. Distributed structures are local to each RKII set, each generating specific control signals for its respective RKII set. This hierarchy is shown in Figure 3.17. It minimizes the number of interconnections, while keeping the local control structures simple. Global signals are listed and described in Table 3.1.

Apart from the listed control signals, local control structures receive the address of the currently active cell from the global structure. The control structure of each RKII set compares the received address with its own and adjusts its operation. The global filter signal is propagated to the local filter signal if the cell is the currently addressed one. The same happens to the cell clock, W_{ie} and W_{ei}, and the reverse happens to W_{ee} and W_{ii} - they are propagated only if the cell is not the currently addressed one.

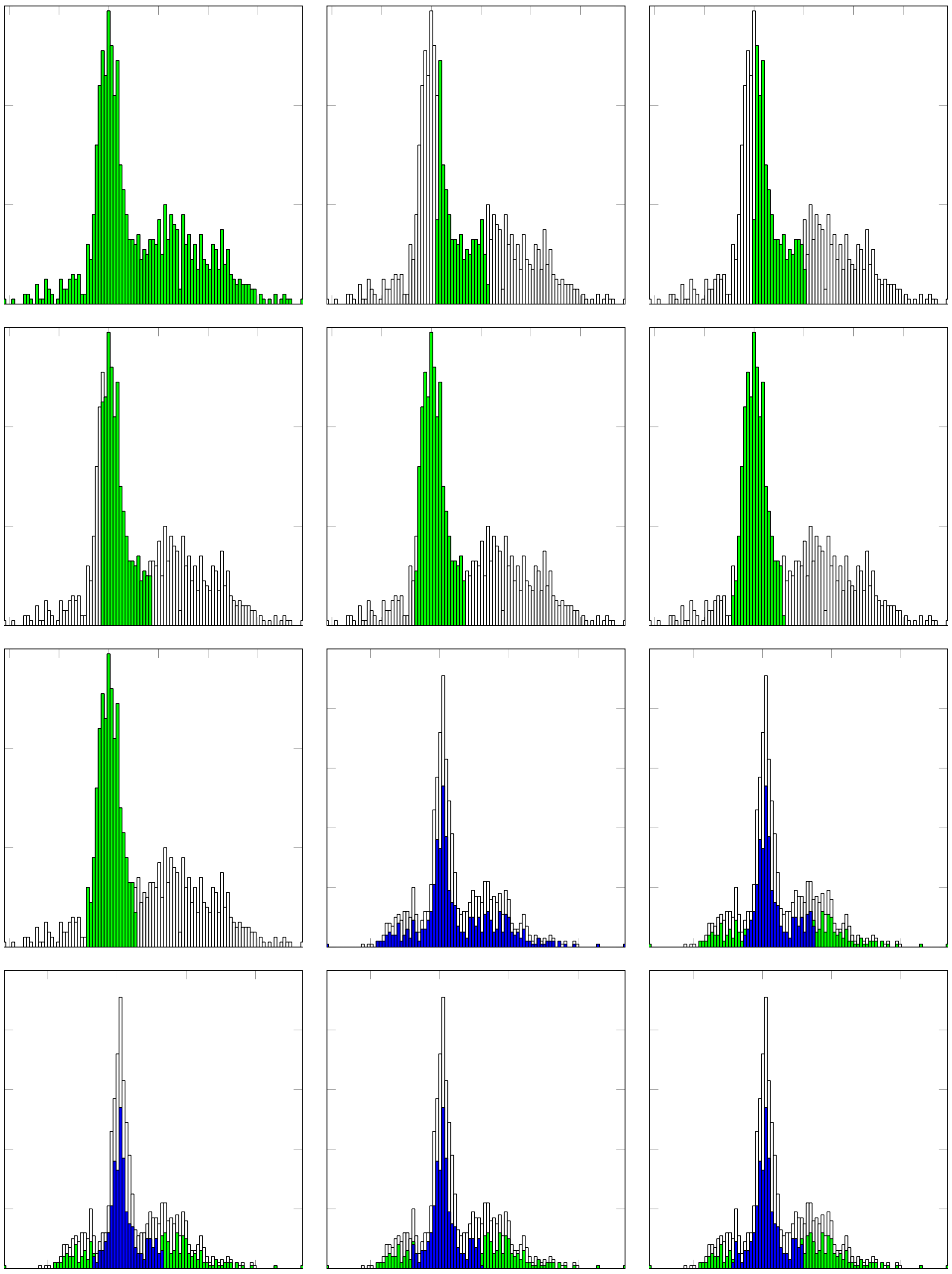


Figure 3.15: BIST model simulation.

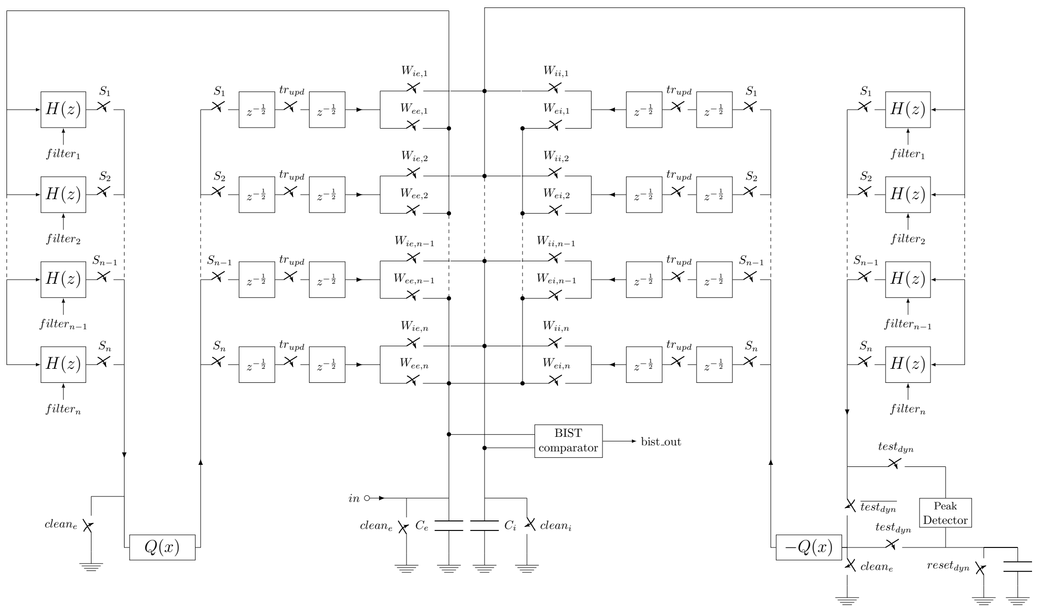
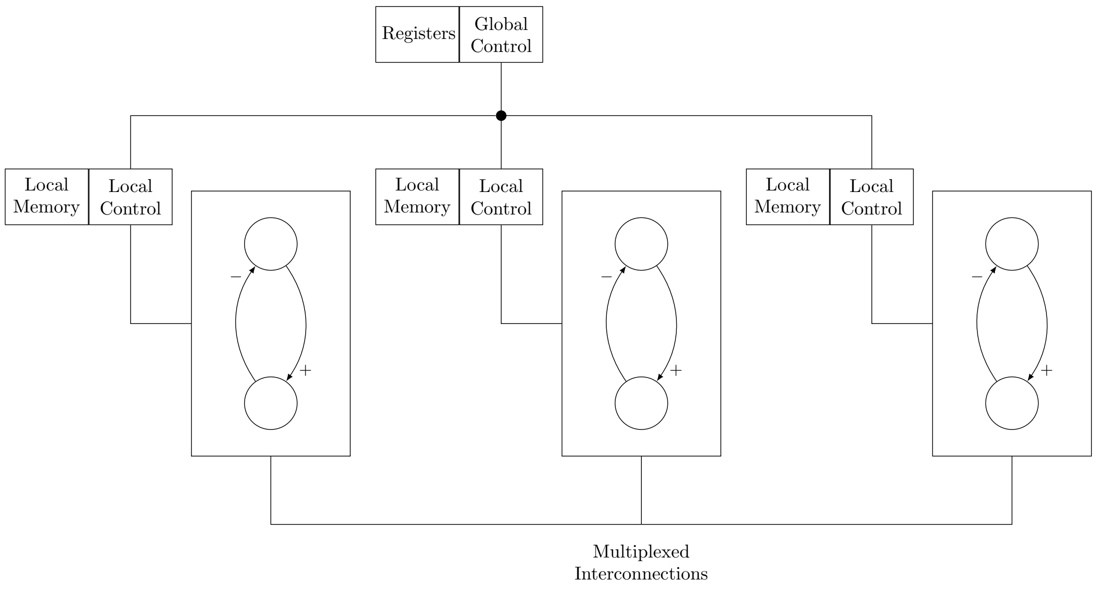


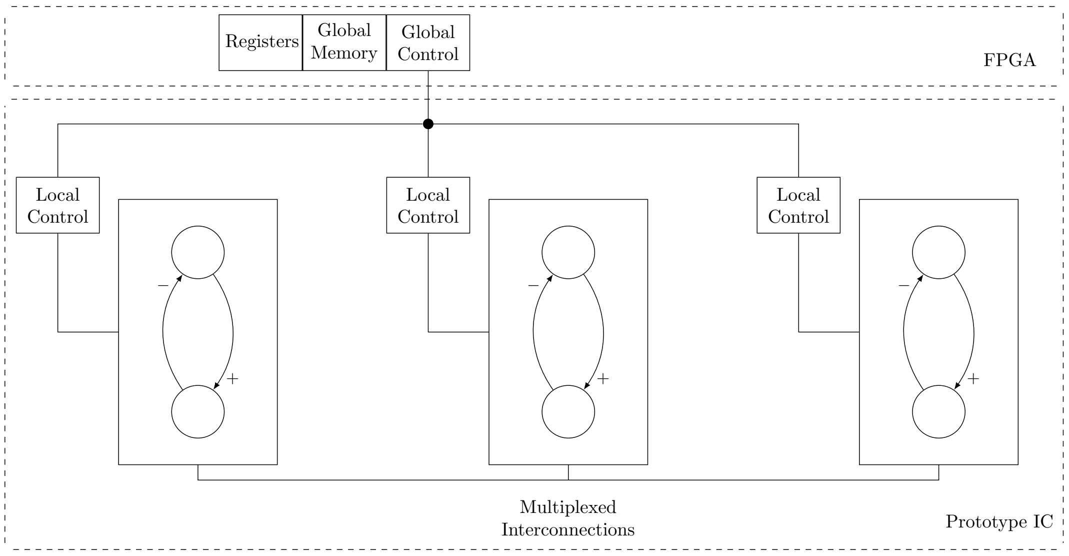
Figure 3.16: Multiplexed network implementation.

Table 3.1: Control signals in the multiplexed network.

Signal	Description
Used in local control	
S	Cell clock
W_{ee}	Excitatory to excitatory weight control
W_{ie}	Excitatory to inhibitory weight control
W_{ei}	Inhibitory to excitatory weight control
W_{ii}	Inhibitory to inhibitory weight control
$filter$	Activates the filter for the active cell
Directly applied analog blocks	
tr_{upd}	Updates all memories simultaneously
$clean_i$	Discharges C_i
$clean_e$	Discharges C_e
$test_{dyn}$	Activates the BIST dynamic test.
$reset_{dyn}$	Resets capacitor in BIST dynamic test.
Generated by analog blocks	
$bistresult$	Result of the BIST (either pass or fail).



(a)



(b)

Figure 3.17: Global and distributed control structure in the RKII network. (a) proposed architecture with distributed memories, and (b) proposed architecture for prototype, with global weight memory.

The weight signals (W_{ee}, W_{ie}, W_{ei}, and W_{ii}) control the integration time of interconnection capacitors, thus controlling the interconnection weight. W_{ie}, W_{ei}, and W_{ii} are fixed for all RKII sets, therefore they can be generated by the global control structure and its processing by each local control structure is just a logic AND with a simple condition (whether the local set is the active one or not). However, W_{ee} is adjusted by learning, and each RKII set has its own set of weights. This is implemented by propagating a global reference signal, which is scaled by local structures according to its respective weight.

The weights for all RKII sets form an N-1 by N matrix, being N the number of sets in the network. The first approach considered for the implementation of such matrix was to have a local memory with N-1 positions per RKII set. This memory would be accessed by its respective local control structure. This is the strategy shown in Figure 3.17a. However, area limitations forced this option to be discarded, and dictated memories to be implemented extra-chip in the prototype system implementation proposed in this work. With the usage of an external memory, the reading of weights by local control structures is done in a time division multiplexing scheme.

Moreover, in the prototype, global control is also implemented externally, in a FPGA. This allows more RKII sets to be inserted in the network, since area required by extra pads is smaller than the area that would be occupied by the control structure. Moreover, as the global control structure is reconfigurable, the whole system is much more flexible. This strategy is depicted in Figure 3.17b.

A final system implementation with global control on chip would have the same blocks as the prototype version, requiring only a communication interface for configuration.

The option for distributed memories versus a global memory accessed by time multiplexing, as in the prototype, would also be a decision to take into account in final system implementation. In this work, hardware description of both strategies was developed and simulated, but more emphasis is given to the implementation with a global memory, since it is the one used in the proposed prototype, and can also be used in the final implementation.

In order to configure the system, a serial interface, such as serial peripheral interface (SPI) would be required in a final version of the system. In a prototype version with a FPGA, serial interface would be an useful feature, but it is not fundamental for the system operation, as the control structure can be fully reconfigured. In this work, user interface for configuration purposes are implemented and simulated, but a complete serial interface was not.

3.4.1 Pins

Table 3.2 shows the list of the pins in the proposed final system implementation. Overall, there are 11 pins (excluding the power supply and ground connections), being four of them for the SPI interface, one for the analog input to the network, one for the input of an external clock that will be the system clock used by the control structure, one for the output of the cell clock and the remaining four are outputs for the state and output of the excitatory and inhibitory KO cells.

In the prototype implementation, nor SPI pins nor cell clock output are included. In this version, the system is not completely integrated on a chip, but divided between a chip and an

Table 3.2: List of pins in the final system implementation.

Direction	Pin	Description
Input	Input	Input to the network, multiplexed and shared by all active cells.
Input	Clock	External clock.
Input	CS	Chip Select, for SPI interface.
Input	SDI	Serial Data In, for SPI interface.
Input	SCL	Serial Clock, for SPI interface.
Output	SDO	Serial Data Out, for SPI interface.
Output	Output (e and i)	Two pins for output, taken at the interconnection capacitors.
Output	State (e and i)	Two pins for output of the state, one of the excitatory and one of the inhibitory side.
Output	Cell clock	Pin for synchronization. A low to high transition indicates the beginning of a time slot, and a high to low transition indicates the end of a slot. The input of the current active slot should be set right after the low to high transition, and the state/output may be read after the high to low transition.

FPGA. Therefore, it requires interface pads between the global control structure (FPGA) and local control structures (on chip). Moreover, pads for bias of all blocks are required.

3.4.2 Memories

The system contains two main memories: the selected cells memory and the weight memory. The former stores the cells that are chosen to be active. It is written by the BIST control structure, but can also be written externally. The size of this memory is $N \times \log_2(N)$, where N is the number of RKII sets in the network. This memory is a list, read sequentially in each cell clock cycle. An extra register is required to register the number of valid positions in the list. Besides, the weight memory has the size of $(N-1)\cdot N \times L_w$, where L_w is the number of bits of the weight value. All memories support random access from SPI for both read and write purposes.

In normal system operation, memories are accessed for reading only. In BIST mode, the selected cells list is written with the list of cells that passed the test.

3.4.3 Registers

Besides the memories, the system contains a set of registers. These registers may either be used for configuration purposes or to obtain information about the state of the network. They may be accessed by SPI and are used internally by the global control structure.

Time configuration registers are used to configure the time duration of control signals and time intervals important to the network operation. Other registers are used to control the operation mode of the network and the number of active cells.

Table 3.3: Blocks in the global control structure.

Block	Description
time_control	Generates global control signals for the multiplexed network. It is responsible for generating the cell clock, the filter control signal, the weight signals, the update signal and the signal for clean the interconnection capacitors.
select_cell	On cell clock active edge, reads bist selected cells memory in order to select the next active cell, outputting its address. Moreover, it generates a signals that indicates the beginning of a new round (when the number of slots processed in the current round is equal to the number of active cells).
bist_control	On BIST mode, it is responsible for controlling the operation of the network, obtaining control over the operation of the time_control block. Moreover, it generates control signals for switching the multiplexed network for the BIST operation.
selected_memory_control	Access control for the selected cells memory. This memory is accessed by the blocks bist_control (for random write), spi_controller (for sequential and random read and write) and select_cell (for sequential read). Moreover, this block controls the data bus used to transmit the cell address and the values of interconnection weights to local control structures.
weight_memory_control	Access control for the weight memory. This memory may be accessed by the spi_control (random access for read/write) and by the selected_memory_control block, that controls the data bus to interface with local control structures.
spi_controller	Controls SPI interface. It is responsible for accessing registers and memories, for both read and write purposes, based on orders from SPI commands.

3.4.4 Control structure and system architecture

The blocks present in the global control structure are listed in Table 3.3 and the blocks present in the local control structures are listed in Table 3.4.

Figure 3.18 shows the toplevel architecture of the system. In this figure, the Global Control block includes the time_control, the select_cell, and the weight memory and its access control, and the Selected Cells List block includes its access control block.

Digital interface with the outside is ensured exclusively by the spi_controller block. It receives the SPI signals (serial data in (SDI), serial clock (SCL) and chip select (CS)), interprets and executes commands, that are either read or write from registers or memories, and, depending on the command, retrieves the output to the outside in the serial data out (SDO) pad. It is connected to every register and to every memory in the system.

The generation of signals responsible for the operation of the multiplexed network is ensured

Table 3.4: Blocks in the local control structures.

Block	**Description**
local_control	Generates control signals for filter and switches based on time signals received from the time_control and the address from select_cell. Moreover, they receive a signal from the bist_control, that signals BIST operation.

by the time_control block. It generates the cell clock, the filter control signal, the control signals for cleaning the interconnection capacitors and for the update of analog memories, and the signals that control the charging of the interconnection capacitors. As well as for the filter control signal, the duration of these signals is important, since it controls the interconnection weights. The timing of these signals is achieved using counters based on the reference external clock, and therefore their durations are multiples of the clock period.

In order to select the next active cell, the selected cells memory is read sequentially. The selection of the next active cell is done by the select_cell block. It outputs the address of the new cell. Moreover, it signals the end of a round when the number of cells that were activated in the current round is equal to the number of cells in the *active cells* register. The signal that a new round has started is sent to the time_control block, so that it can activate the tr_{upd} signal.

The bist_control block is responsible for controlling the network when the BIST is running. When in BIST mode, it signals the time_control block that it should operate in BIST mode. Moreover, it generates all the control signals required to set the network to the correct test mode.

Although in normal operation capacitors are cleaned simultaneously, in the BIST mode one of the capacitors must store the average value, and therefore it must be controlled individually. The bist_control block is also responsible for controlling the interconnection capacitors individually when operating in BIST mode.

Because memories are accessed by several blocks, they require access control. For the selected cells memory, in normal operation, it is periodically accessed by the select_cell block. In BIST operation, the bist_control block accesses the memory as well, but it is easily ensured that these blocks do not access the memory at the same time. However, it is also accessed by the spi_controller. This requires more caution, since the SPI may desire to read the memory at any time. It must be ensured that there are no conflicts in memory access, and this is done by the selected_memory_control block. Since the SPI may operate at a lower rate than the system clock, it can tolerate small jitter and, therefore, wait a few clock cycles for the desired value to be retrieved from memory. The same applies to the weight memory.

The local control structures receive the global control signals and the active cell address. They must check if its local cell is the active one and generate local control signals according to that. In the final system implementation, they access the local memory in order to obtain the interconnection weights, and in the prototype the weight to be used in the next cycle is received in a shared bus, after the respective cell has been addressed for that purpose. The duration of the W_{ee} is signal obtained from the global $W_{ee,ref}$ signal, but scaled by the respective weight.

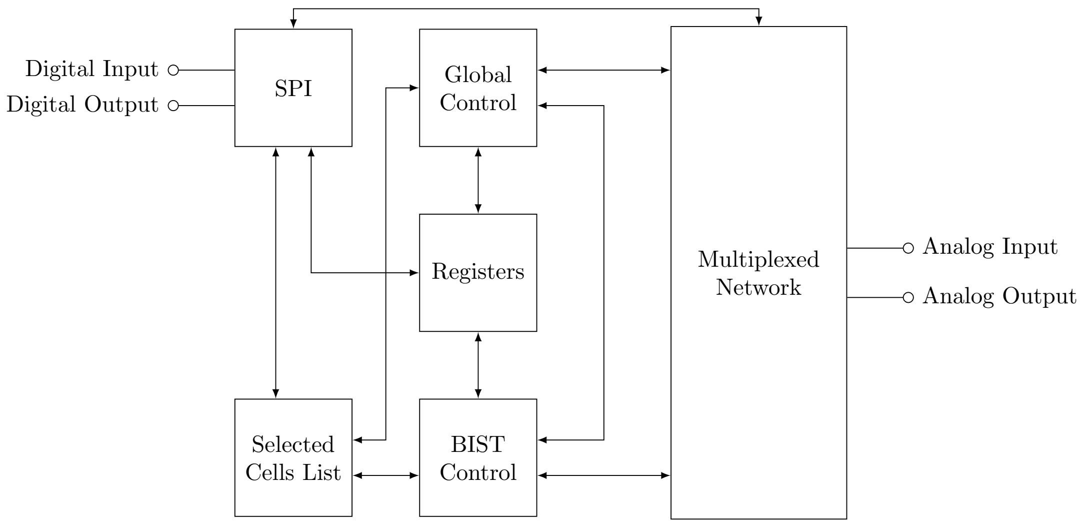
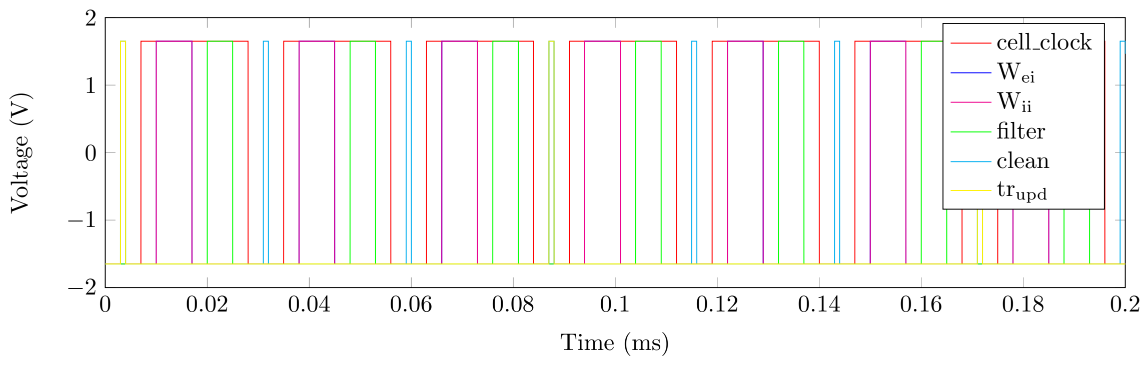


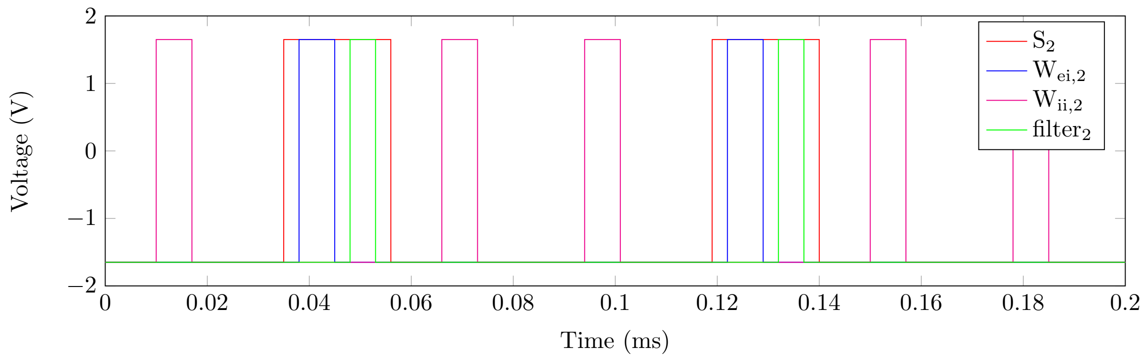
Figure 3.18: Toplevel block diagram of the system.

3.4.5 Time operation

Figure 3.19 shows the control signals for the multiplexed network when three cells are active. Figure 3.19a shows the global control signals, generated by the time_control block, and Figure 3.19b shows the control signals locally generated in cell 2. It can be seen that the local control structure in cell 2 generates the control signal S_2 in the second slot of each round only. Moreover, it generates the local W_{ei} signal at the same time as the global W_{ei}, but only in its own slot, and the W_{ii} at the same time as the as the global W_{ii}, but only in slots from other cells. This corresponds to the timing behavior explained before. After the weight computation, the filter control signal becomes active at the same time as the global $filter$ signal, but only inside the corresponding slot. Moreover, Figure 3.19a shows the timing of the $clean$ and tr_{upd} signals. $clean$ is activated after each slot, in order to clean the interconnection capacitors, and tr_{upd} is activated at the end of every round, in order to update the memories. Note that the activation of tr_{upd} is simultaneous to the activation of $clean$, but occurring only at the end of each round.



(a)



(b)

Figure 3.19: Control signals for the time operation of the multiplexed network. (a) Global signals, (b) Local signals in cell 2.

Chapter 4

Multiplexed RKII Network Design

In this chapter, the design of the analog blocks in the Multiplexed RKII network is described, along with their characterization by simulation results. Results presented were obtained by post-layout simulation, after R+C+CC extraction.

The blocks were designed and dimensioned using Synopsys HSPICE® as the main simulation environment. Cadence Spectre® was used for Monte Carlo simulations due to the absence of specific HSPICE models for the technology used. Full-custom layout was designed using the Cadence Virtuoso environment and checked for design rules (DRC) and equivalence to the desired schematics (LVS) using Mentor Graphics Calibre®. Calibre was also used for layout parasitics extraction.

Moreover, the design of the control structure is explained, and synthesis results are presented.

Each RKII set is composed of two KO sets - one excitatory and one inhibitory. Since the only difference between them is a change of signal in the sigmoid, and the sigmoids are shared, and therefore not included in the RKII cell block, the distributed components of both KO in an RKII are equal. Each one is composed by a filter and an analog memory. While the filter is responsible for ensuring the linear dynamics of the cell, the memory is used for storing the output of the cell, which is used for interconnection with other cells.

One important consideration that must be taken when designing the RKII cell is that the system will comprise as many cells as possible. As consequence, the cell should be small and have a low power consumption. Moreover, the quality of the system is dependent of the similarity among cells. Therefore, cells should be designed to be as independent of the fabrication process as possible.

4.1 Filter Design

The F&H technique was presented in Section 3.1 and the filter implemented in the KO was presented in Figure 3.4. It is built with two stages, each one with a transconductance (Gm) amplifier, a capacitor and switch. In Subsection 4.1.1, the Gm amplifier is presented, and in Subsection 4.1.2, the integration of Gm amplifiers, switches and capacitors to obtain the desired filter is discussed.

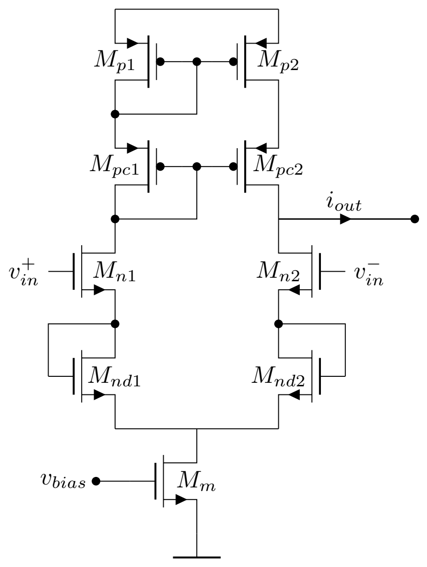


Figure 4.1: Transconductance amplifier topology.

4.1.1 Transconductance Amplifier Design

The time constant of each stage of the filter is given by $\frac{C}{\alpha K_m}$, where K_m is the transconductance of the Gm amplifier, C the capacitance in each stage, and α the F&H duty-cycle. In order to reduce the filter area, it is important to use the lowest possible value for C, therefore requiring the lowest possible value for K_m. In order to minimize K_m, the transconductance amplifier was designed to operate in weak inversion.

The topology used is shown in Figure 4.1. This is the same configuration used in previous implementations [21], with the difference that cascode transistors (M_{pc1} and M_{pc2}) were added in order to reduce the offset. Simulations shown that the introduction of the cascode transistors decreased the offset by one order of magnitude, without significant changes in the dynamic range and in K_m.

The emitter degeneration using diode-connected transistors (M_{nd1} and M_{nd2}) reduces K_m by a factor of two, and increases the dynamic range by the same factor [21]. This was also verified by simulation.

Other two critical aspects in the transconductance amplifier design are the susceptibility to mismatches and the area occupied. Mismatches cause different RKII sets to have different dynamics, which is undesired to proper function of the network, and the smaller the area occupied by the RKII set, the more sets can be included in the network. However, using smaller transistors causes the system to be more susceptible to mismatches and parameter variations. In order to find a good relation for this trade-off, Monte Carlo analysis with process variation and mismatch models was used.

Other aspect taken into account in the design was the linear dynamic range. Because the transconductance amplifier is used in a discrete time filter, it must be able to respond to steps in its

Table 4.1: Transconductance amplifier transistor dimensions.

M	M_{p1}	M_{p2}	M_{pc1}	M_{pc2}	M_{n1}	M_{n2}	M_{d1}	M_{d2}	M_m
W	$20\,\mu$m	$20\,\mu$m	$20\,\mu$m	$20\,\mu$m	$20\,\mu$m	$20\,\mu$m	$20\,\mu$m	$20\,\mu$m	$20\,\mu$m
L	$4\,\mu$m	$4\,\mu$m	$4\,\mu$m	$4\,\mu$m	$4\,\mu$m	$4\,\mu$m	$4\,\mu$m	$4\,\mu$m	$4\,\mu$m

input, and the maximum step it can respond to is given by the dynamic range. Therefore, the filter linearity is highly dependent on the transconductance amplifier dynamic range.

The dimensions used are shown in Table 4.1. The bias current was 100 nA. This value was chosen in order to achieve low power consumption, a low K_m, and a large enough slew-rate.

In order to reduce mismatches, interleaving and common centroid techniques were employed during layout. All PMOS transistors have two fingers and are interleaved to share their centroid. The same applies to all NMOS transistors but M_m. Moreover, dummy transistors were introduced at the boundaries, in order to further improve matching. The layout is shown in Appendix A.1.1.

In order to assess the performance of the transconductance amplifier, a static simulation was performed using the configuration shown in Figure 4.2. The input voltage and the output voltage were swept.

Simulation results are shown in Figure 4.3. In the figures, each colored curve represents simulation results for each process corner. The design goal was to achieve the minimum K_m possible, while keeping the offset current the lowest possible (1 nA was took as a maximum reference value) and the maximum positive and negative steps the largest possible (50 mV was taken as the minimum reference value). Moreover, the goal was to obtain good performance for these parameters for absolute values of the output voltage up to 250 mV, which is large enough considering the sigmoid dynamic range.

The values for K_m were measured as the maximum value of the derivative of the output current over the input voltage, and the step values were measured as the voltage range between $V_{in} = V_{out}$ and the value of $V_{in} - V_{out}$ for which the transconductance became less than 90 % of the maximum value of K_m.

Table 4.2 shows the results of a Monte Carlo analysis, using models for process variations and mismatches. Two hundred Monte Carlo points were used for the simulation.

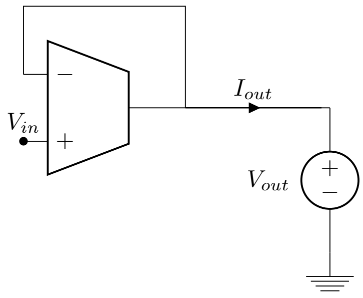
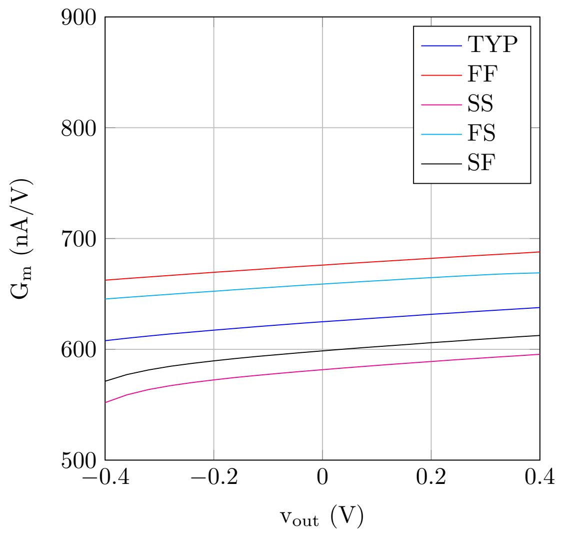
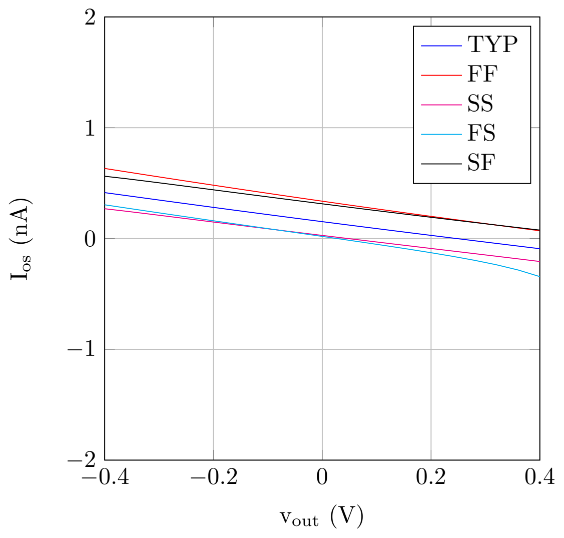


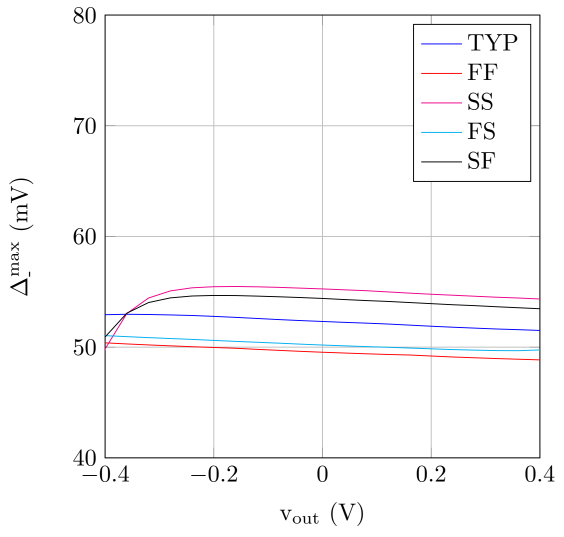
Figure 4.2: Transconductance amplifier static test.



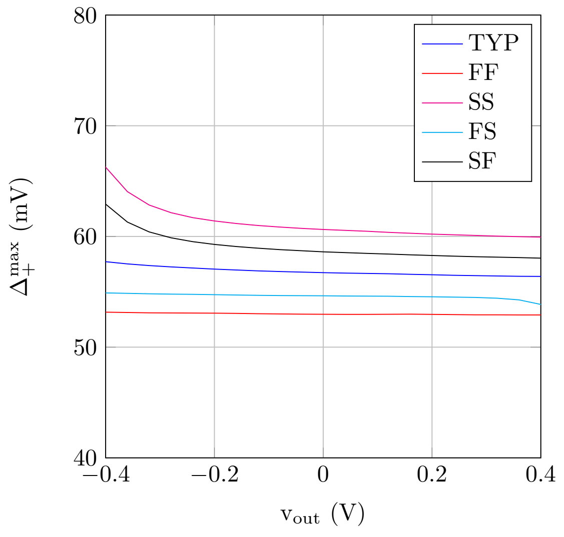
(a)



(b)



(c)



(d)

Figure 4.3: Transconductance amplifier static test results (a) K_m versus V_{out} (b) offset current versus V_{out}, (c) maximum negative step versus V_{out}, and (d) maximum positive step versus V_{out}.

Table 4.2: Transconductance amplifier Monte Carlo static analysis.

Parameter	K_m	I_{os}	Δ^-_{max}	Δ^+_{max}
Mean	622.9 nA/V	136.8 pA	52.43 mV	57.1 mV
Std. Dev.	13.69 nA/V	2.235 nA	1.849 mV	1.924 mV

Table 4.3: Capacitance values used in filters.

Pole (projected)	220 rad/s	720 rad/s
Pole (simulation)	228 rad/s	717 rad/s
Capacitance	2.8 pF	0.85 pF

4.1.2 Filter Integration

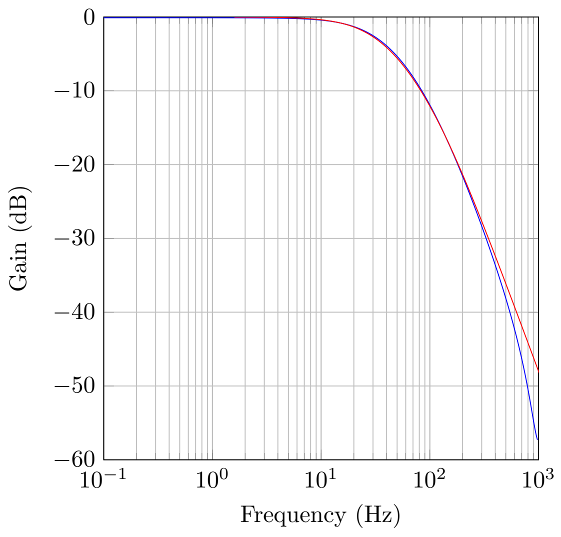
As observed in Section 3.1, each filter pole is given by the time constant of each filter stage, which is equal to $\frac{\alpha C}{K_m}$. Lowering the filter duty-cycle, α, decreases the value required for C, which reduces the area occupied. Furthermore, it allows more RKII sets to be multiplexed.

While the sampling frequency does not affect the filter time constant (assuming a constant duty cycle), it is one important aspect in the filter design. Decreasing the sampling frequency causes the RKII set to differ more from the original continuous time model. However, it allows more cells to be multiplexed, considering that the limitation is not imposed by the filter active time (which is proportional to the sampling period). Moreover, if the sampling frequency is higher, steps in the filter input/output are smaller, decreasing possible limitations caused by slew-rate and limited dynamic range. For the same duty-cycle, an increase in the sampling frequency is limited by the switching time, which may be a problem if the filter active time becomes too small. In this work, the design was made considering a target sampling frequency of 2 kHz, which is adequate, as seen in Chapter 3. The target duty-cycle considered was $\frac{1}{1000}$. While the design does not impose major limitations on the frequency to be used, capacitors and K_m are fixed by design, therefore causing the duty-cycle to be automatically fixed for the desired pole values.

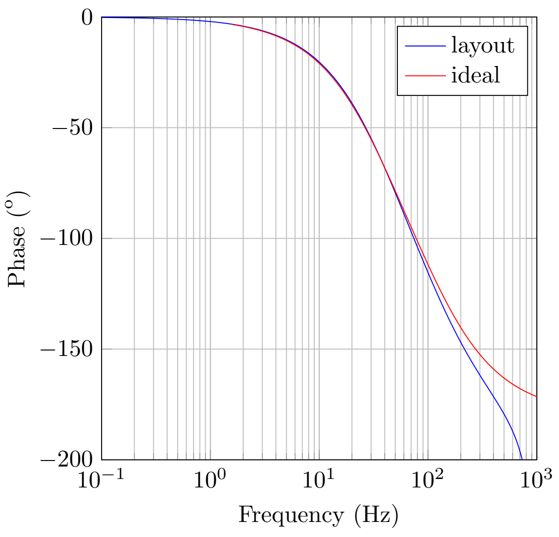
Table 4.3 shows the capacitance values used in the filter. Moreover, the table shows the pole frequencies obtained for a single stage F&H using only each capacitor individually. Simulations were done using HSPICERF Shooting Newton AC analysis, which is equivalent to SpectreRF Periodic AC analysis, referred in Section 3.1.

The order of the stages in the filter is not arbitrary. The lower frequency stage was used at the output, which has several advantages. The most significant one is a better and cleaner output signal, because spikes caused by clock transitions are better filtered. Moreover, it decreases possible limitations caused by slew-rate in the lower frequency stage. In this stage, the slew-rate is lower due to its higher capacitance. With the low frequency stage as the second in the filter, the signal applied to its input was already filtered by the first stage, thus reducing the input amplitude. Furthermore, the larger capacitance is at the filter output, reducing errors caused by charge sharing when the filter output is connected to the sigmoid input.

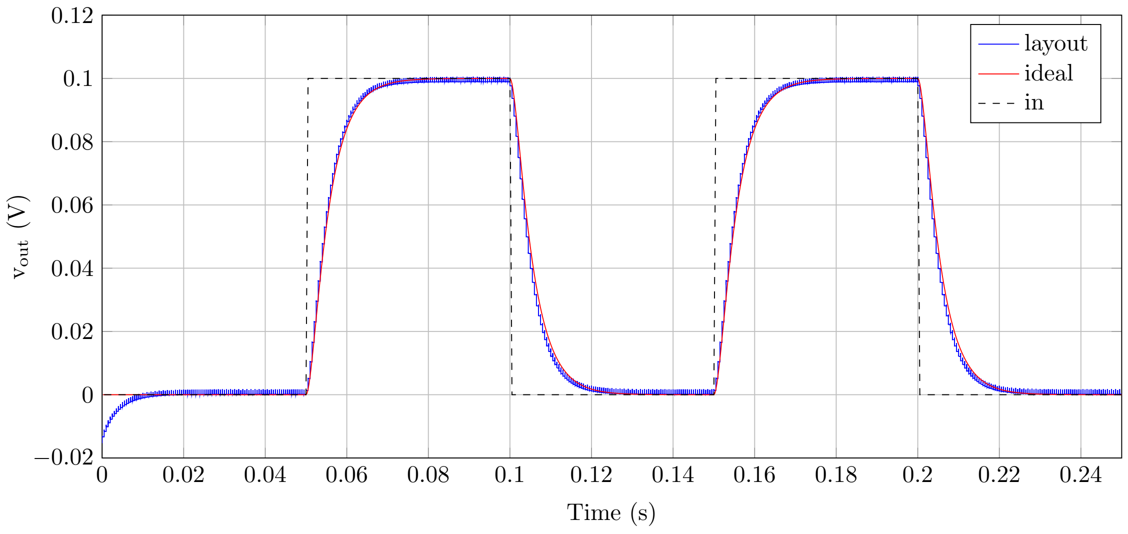
The switches used in the filter were implemented by CMOS transmission gates. A scheme with a dummy transmission gate to reduce clock-feedthrough [63] was tested and shown slightly effective in schematic simulations, but abandoned after post-layout simulations, because it showed worse results than the simple transmission gate, as it will be shown later. Filter layout is shown in Appendix A.1.2.



(a)



(b)



(c)

Figure 4.4: Post-layout filter simulatios. (a) magnitude frequency response, (b) phase frequency response, and (c) response to a square wave (with offset manually removed).

Figure 4.4 shows simulation results compared with the ideal filter response. Frequency response is shown in figures 4.4a and 4.4b. It was obtained by Shooting Newton AC analysis in HSPICERF. Figure 4.4c shows the response to a square wave. This response is affected by a significant offset, of about 15 mV. As it will be later described, this offset is caused by clock-feedthrough. In order to compare the simulation results with the ideal response, the offset was manually removed in Matlab, resulting in the curve shown in the figure. Both analysis confirm that F&H is successfully scaling up the time constants, like predicted by the theoretical model.

Clock-feedthrough is a significant problem in the configuration used, because asymmetries in the clock feedthrough edges are amplified, causing a voltage error. This is depicted in Figure 4.5a, which shows the voltage at the output of a single stage filter when the input is connected to ground.

In the time interval shown, offset due to clock feedthrough is already at steady state, and the filter is active between 99 ms and 99.0005 ms. We can see that the voltage change at the capacitor when the switch starts conducting is different from the one when the switch stops conducting. This causes the transconductance amplifier to have a voltage difference at its inputs, therefore having non-zero current at its output. The voltage at the output converges to a value at which the voltage change in the capacitor during the filter active cycle exactly compensates the asymmetry in the clock feedthrough (as seen in Figure 4.5a). Naming ΔC_{ft} the asymmetry in clock feedthrough, the steady state offset must satisfy (4.1). Note that the output voltage change when the switch is off is not considered in this analysis, and V_{out} is assumed constant. Although neither is true, the approximation is reasonable.

$$\Delta C_{ft} + \frac{(V_{out} - V_{in})K_m}{C} \frac{\alpha}{f_{sample}} = 0 \tag{4.1}$$

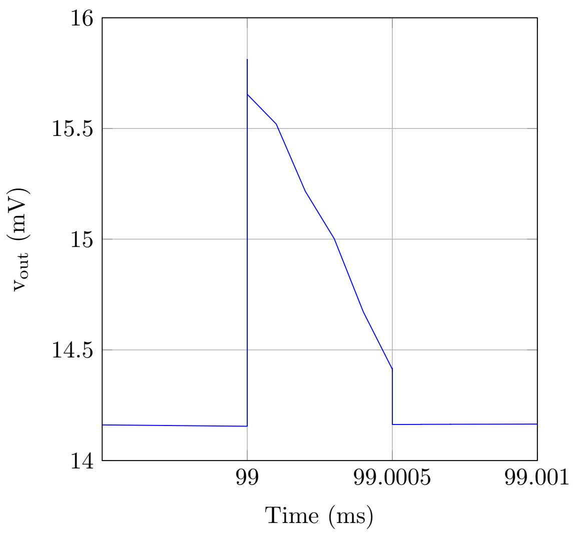
To evaluate the offset, we set V_{in} to zero. From that, the offset voltage is given by (4.2).

$$V_{os} = \frac{C \cdot f_{sample}}{\alpha \cdot K_m} \cdot \Delta C_{ft} \tag{4.2}$$

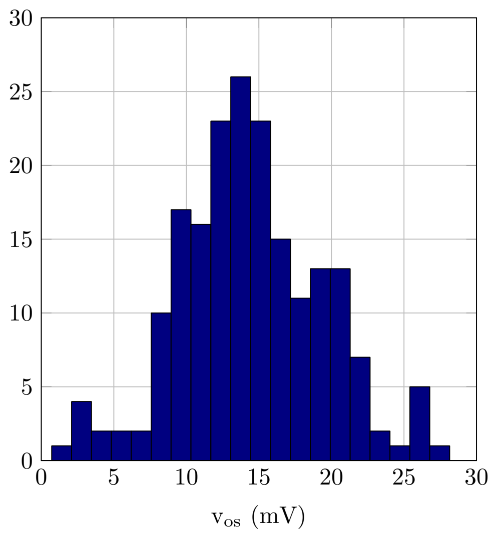
The asymmetry in clock feedthrough is being amplified by a factor of $\frac{C \cdot f_{sample}}{\alpha \cdot K_m}$. Analysing this factor, we note that $\frac{C}{\alpha \cdot K_m}$ is the filter time constant, fixed by the model. The only way to reduce the amplification factor is to reduce the sampling frequency. For the design parameters used, the amplification factor is around 2.7 for the higher frequency stage and around 9 for the lower frequency stage. It is important to note that the asymmetry in clock feedthrough is not constant for the input and output range, causing the error introduced to be nonlinear, and not a simple offset.

Figure 4.5b shows the histogram of the filter output offset voltage, obtained by Monte Carlo transient analysis. The mean value of the distribution is 14.9 mV and the standard deviation is 4.952 mV As we can see, the average offset is much larger than the one that would be obtained if the transconductance amplifier offset was the only offset source considered. However, the same does not happen to the standard deviation.

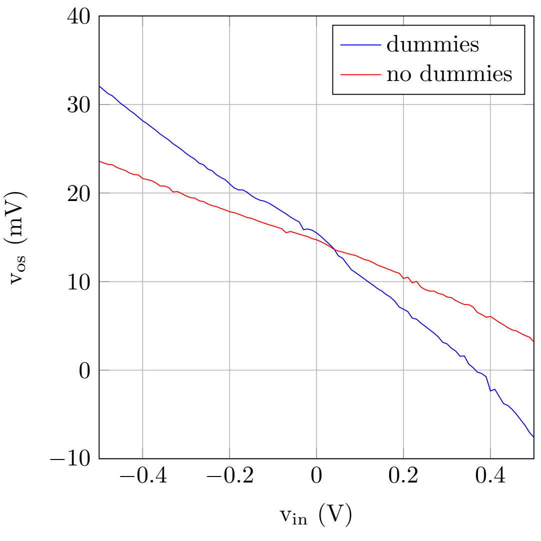
Earlier was referred that a scheme with a dummy transmission gate was considered for the switch, but later abandoned. The reason for using a dummy transmission gate was to reduce errors due to clock feedthrough. While this was successfully observed in simulations before layout, the introduction of parasitics in post layout simulation significantly increased clock feedthrough and decreased the apparent advantage in using dummy transmission gates. Post-layout analysis was performed using both filters by applying a constant signal at the filter input and observing the steady state value. Figure 4.5c shows the results of this analysis. The voltage error is considerably smaller in the configuration without a dummy transmission gate, therefore this option was preferred.



(a)



(b)



(c)

Figure 4.5: Filter offset analysis. (a) offset due to clock feedthrough, (b) filter offset histogram, obtained by Monte Carlo anaysis, and (c) offset versus input voltage.

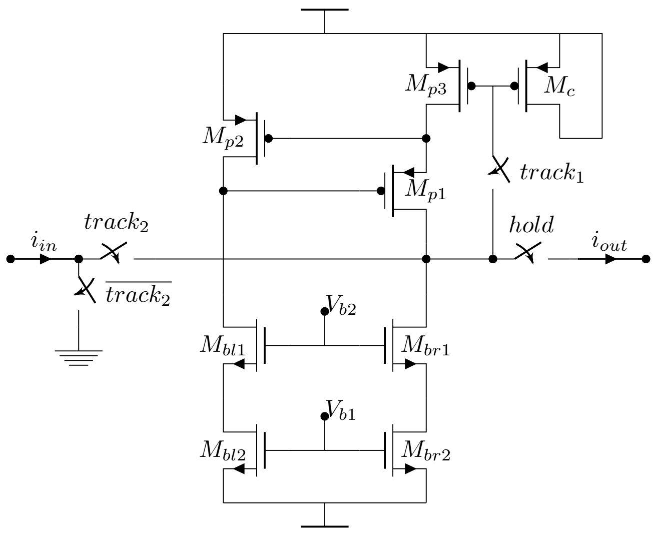


Figure 4.6: Delay block schematics.

4.2 Memory Design

The memory is implemented by two SI (switched current) $z^{-1/2}$ delay blocks [66], improved by a regulated cascode [67], as proposed in earlier RKII network implementations [19]. The schematics for the delay block is shown in Figure 4.6.

Both input and output are current signals. The circuit tracks the input when $track_1$ and $track_2$ are high and *hold* is low. Naming I_b the drain current in M_{br1} and M_{br2}, in tracking mode, the current in M_{p3} and M_{p1} is $I_b - i_{in}$, thus charging M_{p3} gate. M_c is a MOS capacitor, used in order to have a significant capacitance at M_{p3} gate. When the tracking signals becomes inactive and the hold signal active, charge stored at M_{p3} gate remains constant, thus keeping the output current equal to $-i_{in}$.

M_{p2} and M_{p3} form a regulated cascode [67]. They form a feedback loop that keeps the drain voltage of M_{p3} fixed, mitigating current errors caused by channel length modulation in in M_{p3}.

For proper functioning, $track_2$ must be disabled slightly later than $track_1$, otherwise the voltage in M_{p3} gate would change considerably in the transition from track to hold mode, causing significant current errors. The *hold* signal is simply a inverted version of $track_2$.

The transistor dimensions used are show in Table 4.4. The design goals were to minimize current offset and the rate of discharge of the voltage at M_{p3} gate, in order to minimize the current errors between currents used early in the round, soon after update, and currents used later in the round, soon before update. Moreover, effort was taken to keep minimum track time, power consumption and dimensions within reasonable values. Bias currents are 0.4 µA at M_{br1}/M_{br2} and 2 µA at M_{bl1}/M_{bl2}. The layout of a delay block is shown in Appendix A.1.3.

Table 4.5 shows the simulation results obtained for the delay block. Two tests were performed, one for obtaining the offset and the decay rate, other for assessing the settling time. The former was performed by applying a constant current at the input and measuring the output current when

Table 4.4: Delay block transistor dimensions.

M	M_{p1}	M_{p2}	M_{p3}	M_c	M_{bl1}	M_{bl2}	M_{br1}	M_{br2}
W	1 µm	1 µm	0.4 µm	16 µm	0.4 µm	0.4 µm	2 µm	2 µm
L	2 µm	2 µm	2 µm	14 µm	2 µm	2 µm	2 µm	2 µm

Table 4.5: Delay block simulation results.

Corner	Typical	FF	SS	FS	SF
Offset current	−0.89 nA	−0.95 nA	−0.74 nA	−0.77 nA	−1.10 nA
Decay rate	0.50 nA/ms	4.05 nA/ms	−3.07 nA/ms	−0.51 nA/ms	2.25 nA/ms
Settling Time	1.7 μs	1.6 μs	2.0 μs	1.9 μs	1.5 μs

Table 4.6: Delay block Monte Carlo analysis.

Measurement	Offset Current	Decay rate
Mean	−0.64 nA	−0.39 nA/ms
Std. Dev.	0.23 nA	0.55 nA/ms

the block changes from tracking to hold mode. Both the input current and the output voltage were swept. The offset current was measured in the beginning of hold, and the decay rate is the current difference per unit of time, measured by comparing the output current at the beginning of hold and after a considerable amount of time. It was observed that these values do not vary significantly nor with the output voltage nor with the input current within the range of interest. The test for the settling time was performed by applying an input step when in tracking mode and observing the time taken to the voltage at the gate of M_{p3} to achieve 99 % of its final value. Table 4.6 shows the results of a Monte Carlo analysis for the first test.

Results obtained are reasonable for the desired application. In order to achieve good results for offset and decay rate, M_c was chosen considerably large, which increased the settling time. The memory settling time will be the largest time slice in the multiplexed network operation. However, it will not limit the maximum number of multiplexed sets in the network, because they are already limited by the area available.

Table 4.7: Sigmoid block transistor dimensions.

M	M_1	M_2	M_3	M_4	M_{bl}	M_{bh}
W	10 µm	10 µm	10 µm	10 µm	10 µm	10 µm
L	2 µm	2 µm	2 µm	2 µm	2 µm	2 µm

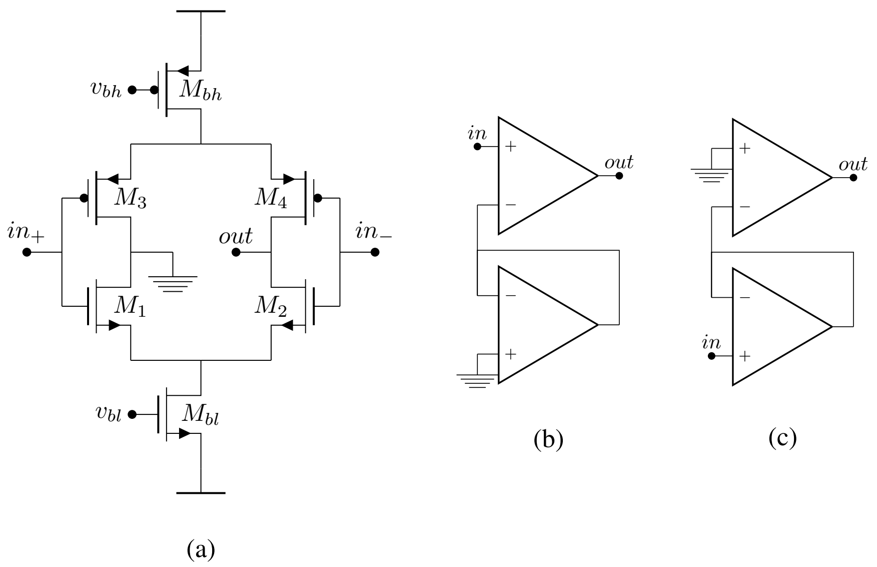


Figure 4.7: Sigmoid block. (a) fully complementary differential pair, (b) sigmoid block used in the excitatory KO, and (c) sigmoid block used in the inhibitory KO. .

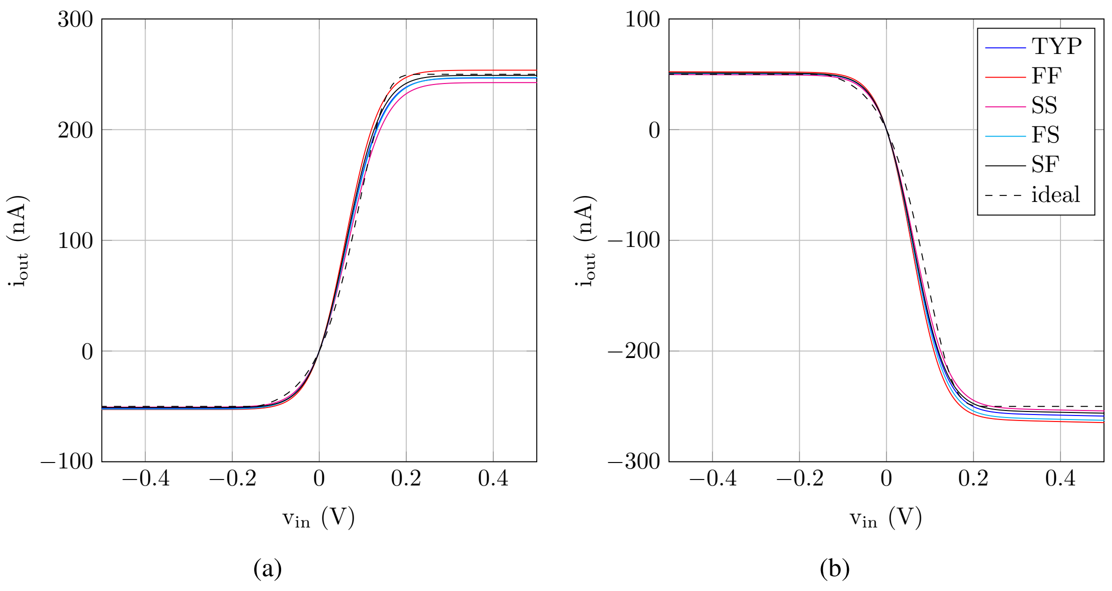


Figure 4.8: Post-layout sigmoid characteristic. (a) excitatory sigmoid, (b) inhibitory sigmoid characteristic.

4.3 Sigmoid Design

Differently than the activation function in most neural networks, the sigmoid in KO is asymmetric. In order to achieve such response, several approaches have been used in previous implementation. In earlier RKII implementations, the sigmoid was obtained using a transconductance amplifier with unbalanced active load, and with offset compensation [21]. In other implementation, a translinear current multiplier is used to unbalance the current mirror of a differential pair [23]. Both implementations are designed to operate in subthreshold and have limited control over saturation levels after design. In this work, a strategy with a fully complementary differential pair transconductance amplifier is used [68]. The schematics used are shown in Figure 4.7a. The saturation levels are programmed independently. The higher saturation level is the bias current of M_{bh}, and the lower saturation level is the bias current of M_{bl}. This configuration alone is affected by significant offset. However, the offset can be canceled using the configuration shown in Figure 4.7b [68], where each amplifier represents a fully complementary differential pair. A replica of the differential pair is used to generate a reference offset voltage, that is applied to the inverting input, cancelling its own input.

A simple modification of this scheme was used to obtain the inverted sigmoid used in the inhibitory KO. It is shown in Figure 4.7c. It obtains the inverse of the sigmoid by applying the input to the differential pair used to generate the reference offset voltage. As it is operating as voltage follower, the input will be summed to the offset and applied to the inverting input of the other differential pair, therefore obtaining an inverted version of the sigmoid obtained in Figure 4.7b, with offset equally cancelled.

The dimensions used in the sigmoid block are shown in Table 4.7. The sigmoid operates in the subthreshold region. The target output saturation currents are 250 nA and -50 nA in the excitatory sigmoid, and 50 nA and -250 nA in the inhibitory sigmoid. This values were chosen considerably larger than errors caused by offset and nonlinearities in other blocks, so that their effect is reduced, and low enough not to have an excessively large power consumption. Simulations shown that in this region the sigmoid characteristic does not change considerably with the transistor dimensions, so the design goal was to achieve good matching, in order to reduce the offset, and to have a low enough settling time. In order to achieve good matching, layout was done using common centroid and interleaving techniques. All transistors were designed with two fingers, and a common centroid is shared by each group of transistors in the two differential pairs. For instance, M_1 and M_2 of both pairs are interleaved and all have the same layout. The same happens for M_3 and M_4. The bias transistors of both pairs are also interleaved and share a common centroids. In this way, matching is optimized, in order to minimize the offset. Moreover, dummy transistors were added at the boundaries. Layout of the excitatory sigmoid is shown in Appendix A.1.6, and the layout of the inhibitory sigmoid is shown in Appendix A.1.7.

Figure 4.8 shows the sigmoid characteristic, obtained by static analysis. As we can see, it is similar to the one described in the model, shown dashed in the figures. In the codomain, 1 unit in the original model is equivalent to 50 nA in the implemented sigmoid, and in the domain 1 unit

Table 4.8: Excitatory sigmoid block Monte Carlo analysis.

Measurement	Offset Current	Low input saturation level	High input saturation level
Mean	0.34 nA	−108.8 mV	237.4 mV
Std. Dev.	5.9 nA	5.2 mV	6.0 mV

Table 4.9: Inhibitory sigmoid block Monte Carlo analysis.

Measurement	Offset Current	Low input saturation level	High input saturation level
Mean	0.15 nA	−108.9 mV	238.3 mV
Std. Dev.	5.0 nA	4.8 mV	5.9 mV

Table 4.10: Sigmoid block settling time.

Corner	Typical	FF	SS	FS	SF
Excitatory	0.51 µs	0.32 µs	0.55 µs	0.51 µs	0.41 µs
Inhibitory	1.47 µs	1.37 µs	1.51 µs	1.50 µs	1.48 µs

in the original model is around 60 mV in the implemented sigmoid. Tables 4.8 and 4.9 show the results of a Monte Carlo static analysis. Offset has a large standard deviation. However, we expect that it can be significantly reduced by layout techniques used, such as common centroid, which are not taken into account in simulation. The input saturation levels were measured as the input voltage that causes an output of 99 % of the saturation levels. An important aspect that must be taken into account is that sigmoid blocks are shared by every RKII set in the network, thus making parameter variations less important than in the case of the filter.

Table 4.10 shows the sigmoid settling time, obtained by applying a step to the sigmoid input and observing the time required for it to settle at 99 % of its final value. The settling time for the inhibitory sigmoid is significantly larger than the one for the excitatory sigmoid. This is easily explained by observing that in the excitatory case, the input is applied directly to the differential pair from which the output is taken, whereas in the inhibitory sigmoid it is not.

4.4 RKII network integration

In previous sections, the basic analog building blocks that constitute the RKII network were presented. In this section, the integration of these blocks to form a network is described.

4.4.1 Sigmoid and Memory integration

As observed in previous sections, the sigmoid and delay block used in memory have considerably large settling times. Moreover, they are cascaded in a KO set, so the settling time of the cascade of these blocks is an important parameter, since it will impose timing limitations to the system.

Table 4.11: Cascaded sigmoid and delay block settling time.

Corner	Typical	FF	SS	FS	SF
Excitatory	1.87 µs	1.70 µs	2.12 µs	2.06 µs	1.70 µs
Inhibitory	2.11 µs	1.81 µs	2.21 µs	2.28 µs	2.03 µs

In order to assess this settling time, a test was done by cascading a sigmoid and a delay block, applying a voltage step at the sigmoid input, and observing the time required for the voltage at the gate of M_{p3} in the delay block to settle at 99 % of its final value, while the delay block was in tracking mode. Results are shown in Table 4.11. The analysis was done for both the excitatory and the inhibitory sigmoid blocks.

The time between the end of filtering and the end of the cell clock must be large than the settling time to guaranty that the correct value is stored in memory.

4.4.2 Memory and weight capacitors integration

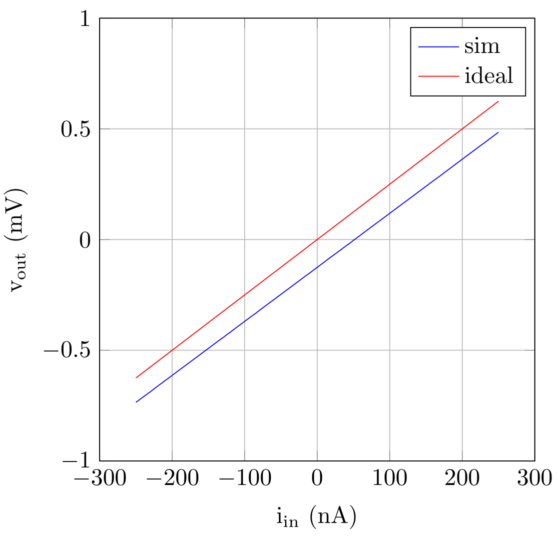
In order to compute interconnection weights precisely, the response of the switching from the delay block to the capacitor must be fast, so that a large range of time can used for current integration. A test was performed by connecting a 2 pF capacitor via a switch to the output of a delay block, charging the delay block for a defined current, driving the delay block to hold mode, and integrate the stored current for a defined time interval. The test was performed for time intervals between 5 ns and 5 µs, with correct performance for all cases.

Results are shown and compared with the ideal integration output in Figure 4.9. For lower integration time, the output is relatively more affected by offset, and curves show a slight decrease in their slope due to the intervention of parasitic capacitances in the integration. The change in slope is expected to be much larger when the system is fully integrated, because parasitic capacitances at the output node will be in parallel with the designed capacitor. However, this can be corrected by adjusting the integration time.

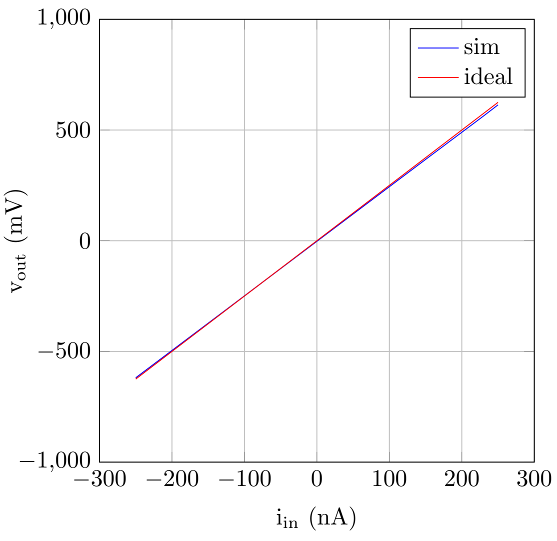
4.4.3 RKII network integration

In order to test the multiplexed RKII network operation, a layout containing three RKII sets fully connected was extracted, and simulated along with layout-extracted sigmoids. The control signals used were generated using the script referred in Chapter 3. The layout of a single RKII set is shown in Appendix A.1.4, and the layout of a network with three sets is shown in Appendix A.1.5

In the network simulated, K_{ii} and K_{ee} were set to 0, so that the coupling between sets could be tested. The sampling frequency used was 2 kHz and the filter duty cycle was $\frac{1}{1000}$. The capacitance used for the interconnection capacitors was 0.425 pF and the integration time for the interconnection weights was 2 µs for both excitatory to inhibitory (W_{ei}) and inhibitory to excitatory (W_{ie}). The inputs applied were current square waves with low current of 0 A and high current of 50 nA. The input current was integrated for 2 µs.



(a)



(b)

Figure 4.9: Memory and weight capacitor integration simulation results (a) integration time of 5 ns, and (b) integration time of 5 μs.

Figure 4.10 shows the simulation results. Inputs are not shown, but easily concluded from the waveforms, since oscillation occurs when a high input is present, and fades when a low input is present. From figures, we conclude that all sets work according to the model and that they are successfully decoupled from each other, since the state of one set does not influence states on other sets. Figures also show that the stable attractor is no longer (0,0), as in the model. This happens due to offset.

In order to assess whether RKII sets are working as expected, the sigmoid output current versus the excitatory state voltage was observed and is depicted in Figure 4.10e. As can be seen, results overlap almost exactly with the sigmoid characteristic. Furthermore, the voltage obtained in C_i after current integration was ploted versus the excitatory sigmoid output in the previous round. This is shown in Figure 4.10f. The curve obtained is expected to follow a straight line with slope of $K_{ie} = \frac{W_{ie}}{C_i}$, however, two phenomena are noticeable: the slope obtained is significantly different from the expected, and characteristic obtained is not exactly a straight line. The former is due to parasitic capacitances in the line used for connection between RKII sets and C_i. Since the line is quite long, its capacity is large, and it increases linearly with the number of RKII sets in the network. Hence, as the capacity increases, the integration time must increase to keep the interconnection weight constant. This will easily limit the number of cells that can be multiplexed for a given sampling frequency. A solution to this limitation will be proposed in Section 4.5. By the other hand, the fact that the characteristic is not a straight line occurs due to charge injection and other nonidealities caused by coupling due to parasitic capacitances. This, however, does not impose a major limitation to the network, since correct operation is achieved nonetheless.

The red curve in Figure 4.10f approximates the results obtained by linear regression. The slope of this curve (along with the sigmoid scaling factor) represents the interconnection weight. The

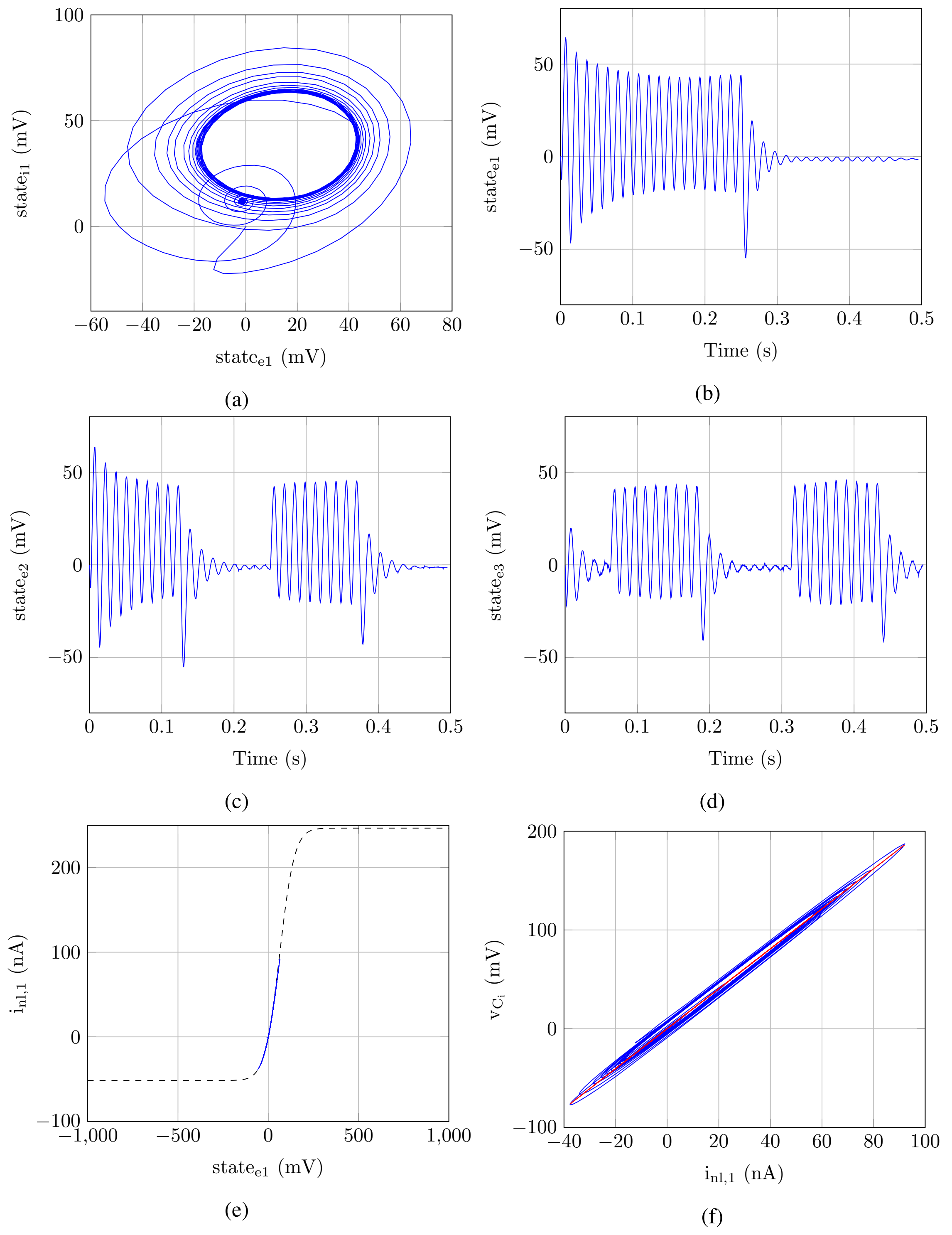


Figure 4.10: RKII network post-layout simulations. (a) phase portrait for RKII set 1, (b) excitatory state for RKII set 1, (c) excitatory state for RKII set 2, (d) excitatory state for RKII set 3, (e) sigmoid characteristic, and (f) interconnection voltage versus sigmoid output .

slope is given by $\frac{\Delta t}{C}$, where Δt is the integration time and C the capacitance. In this case, the slope is 2.02 mV/nA. As $\frac{\Delta t}{C}$ is 2 μs, the equivalent capacitance is 0.988 pF, which is 0.488 pF more than the desired.

In order to observe if the variation of the interconnection weights has the same effect as in the ideal model, the capacitance used was changed to 0.2 pF and 0.8 pF. The phase portrait for the former case is shown in Figure 4.11a and for the latter in Figure 4.11b. As can be observed, the former results in limit cycle attractors for both high and low input, and the latter results in point attractors for both high and low input.

Moreover, Figures 4.11c and 4.11d show the interconnection voltage versus the sigmoid output current in the previous round for both cases. For the first case, the slope is 2.86 mV/nA, resulting in a equivalent capacitance of 0.70 pF, and for the second case, the slope is 1.57 mV/nA, resulting in a equivalent capacitance of 1.28 pF. These results confirm that an equivalent parasitic capacitance of around 0.5 pF takes part in current integration. Furthermore, simulations shown that this capacitance scales linearly with the number of RKII sets in the network.

In order validate the operation of a fully operational network, 16 RKII sets were connected in layout and simulated. The simulation was similar to the one presented in Section 3.2. The same two patterns were learned, and interconnection integration times were W_{ii} = 0.02 µs, W_{ei} = 5 µs, W_{ie} = 5 µs, $W_{ee,h}$ = 0.4 µs, and $W_{ee,l}$ =0.02 µs. As in the simulation with the ideal network, the input applied was [1 1 0 1 0 0 0 1 0 0 0 0 1 1 1 1]. Low input was 0, and high input was 45 nA, integrated during 5 ns. Results are presented in Figures 4.12 and 4.13. It is clear that network operation is similar to the ideal model. The pattern [1 1 1 0 0 0 0 1 0 0 0 0 1 1 1 1] was successfully recognized. Even with switched inputs, set 3 successfully synchronized with the remaining active sets, while set 4 did not.

4.5 Improvements to the network implementation

As observed, parasitic capacitance increases the capacitance subject to current integration for interconnections, hence increasing the time required for integration, which limits the number of RKII sets that can be multiplexed in a network. This is a major limitation of the architecture previously presented. A simple way to overcome this problem is to use the integrator configuration shown in Figure 4.14.

Parasitic capacitances are connected to a virtual ground, hence significantly reducing their influence. Ideally, the result obtained after integration is the same as with the previous configuration, but with signal inversion. Observing the RKII set architecture, it is clear that signal inversion is not a problem: the excitatory KO is inverted, hence becoming ideally equal to an inhibitory KO, and the inhibitory KO becomes ideally equal to an excitatory KO. Thus, the consequence of the signal inversion is that the KO sets are switched. Moreover, as the input to the network is applied as a current signal, subject to current integration, the signal inversion in the integration process must be taken into account.

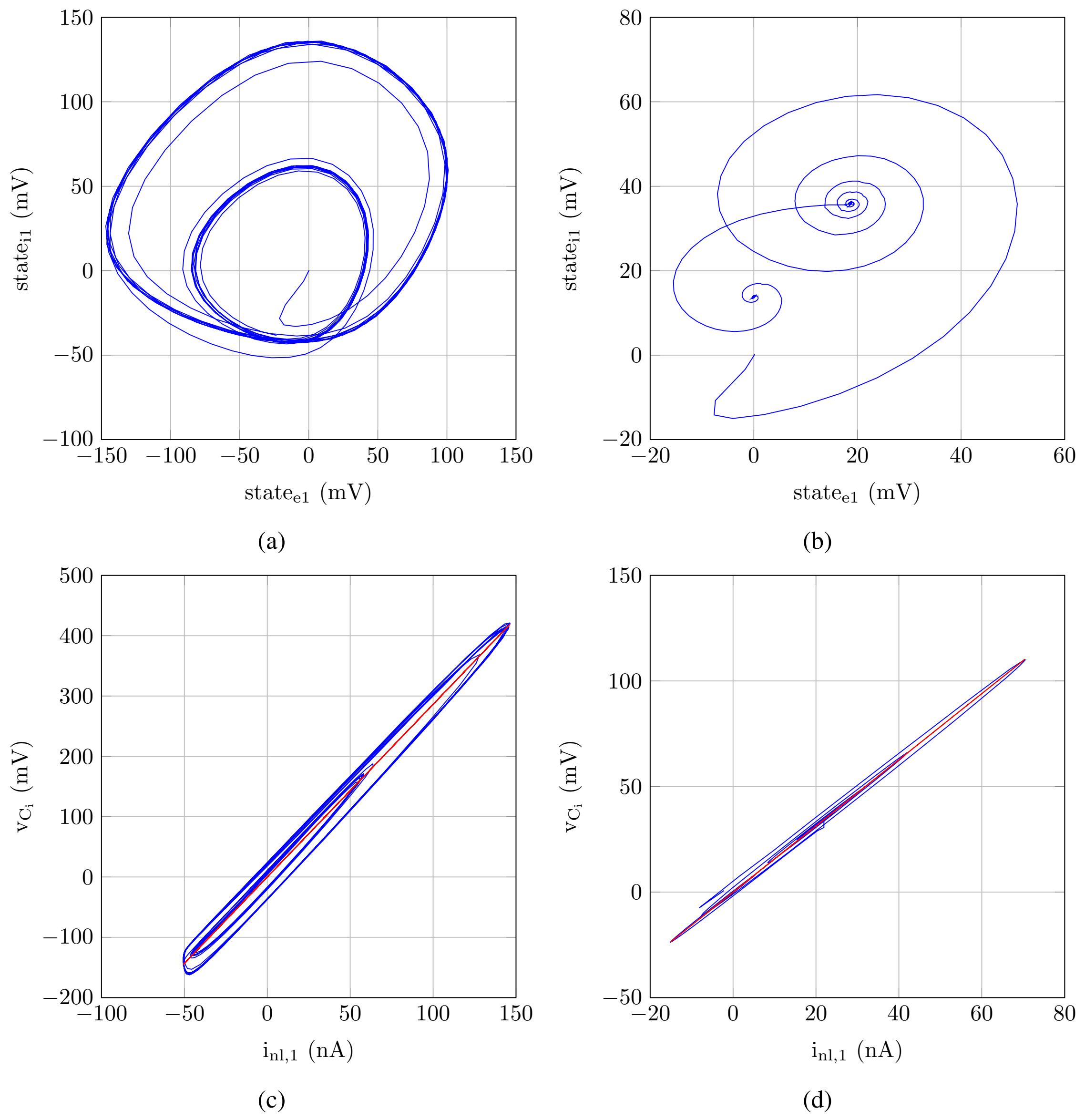


Figure 4.11: Influence of interconnection capacitance in post-layout simulation. (a) phase portrait for RKII for C =0.2 pF, (b) phase portrait for RKII for C =0.8 pF, (c) interconnection voltage versus sigmoid output for C =0.2 pF, and (d) interconnection voltage versus sigmoid output for C =0.8 pF .

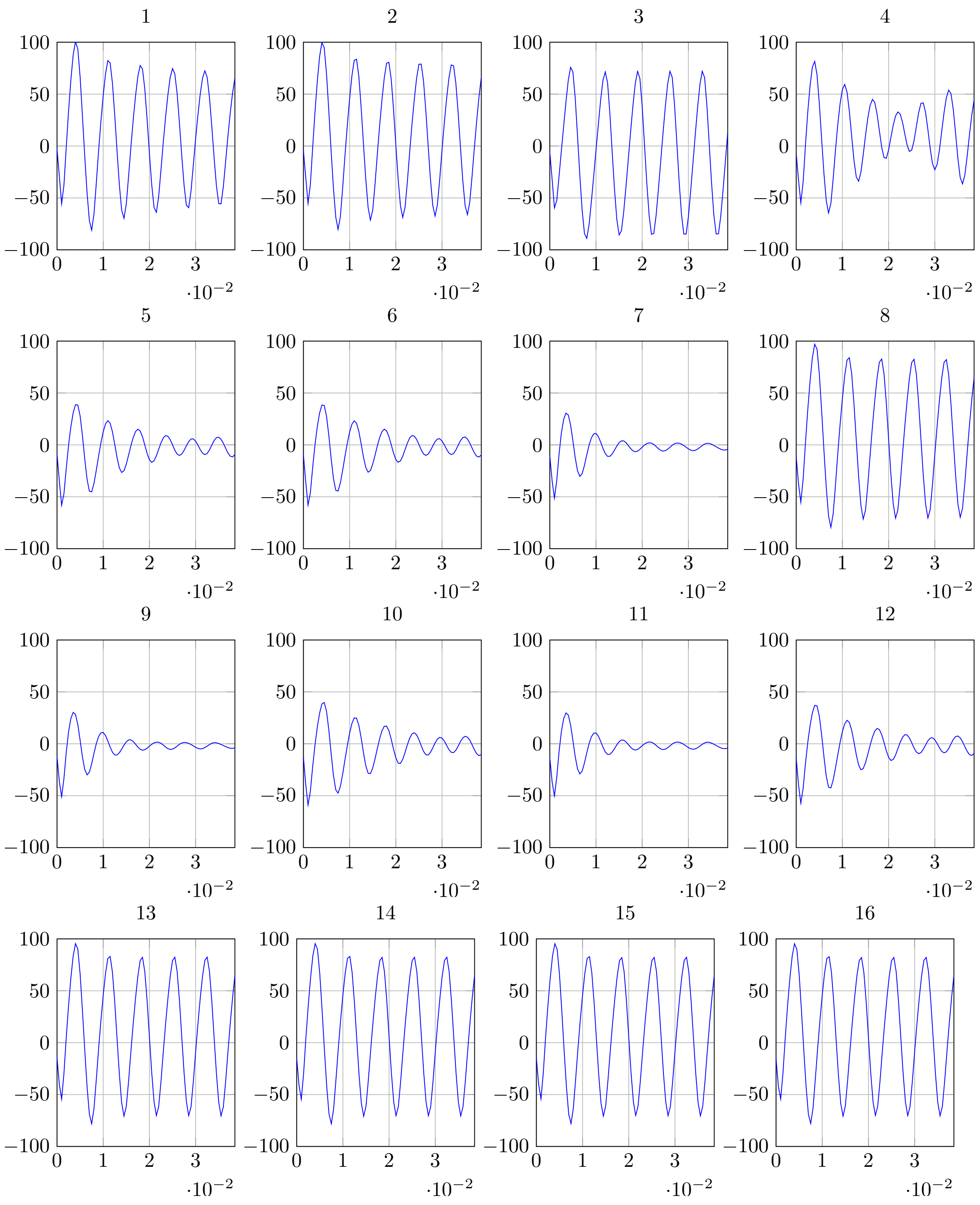
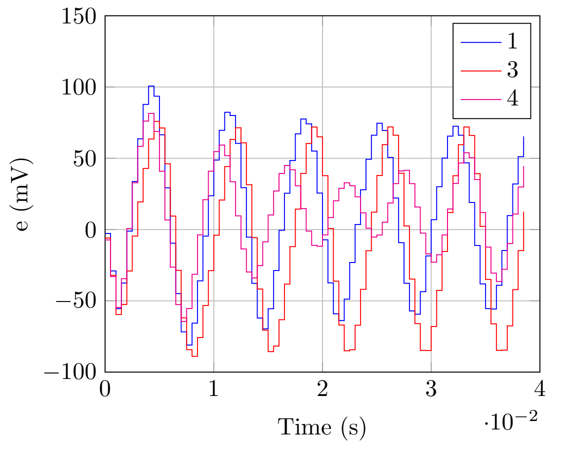
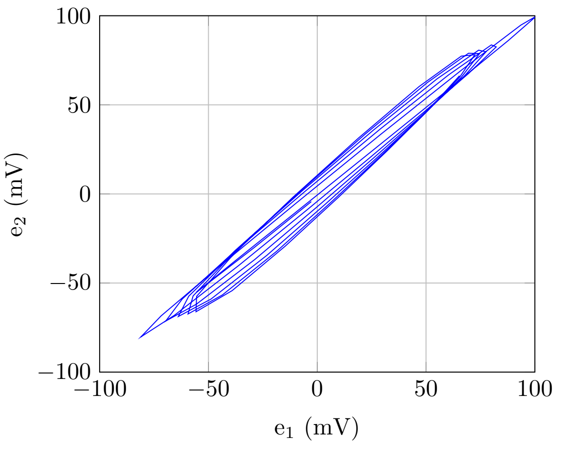


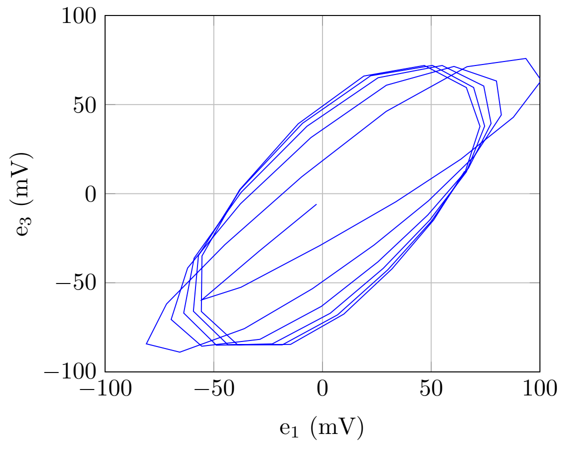
Figure 4.12: Excitatory state of every RKII set in a network of 16 sets, obtained by post-layout simulation.



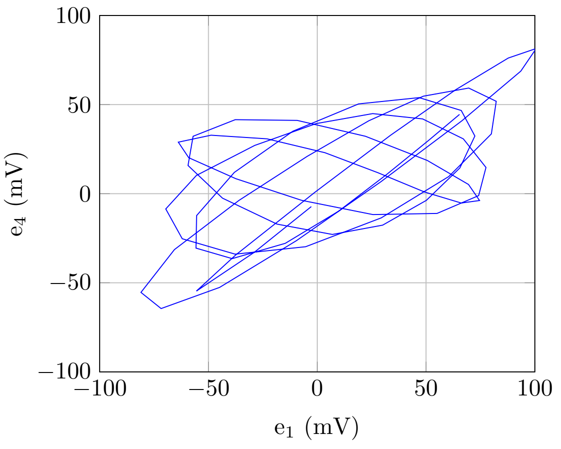
(a)



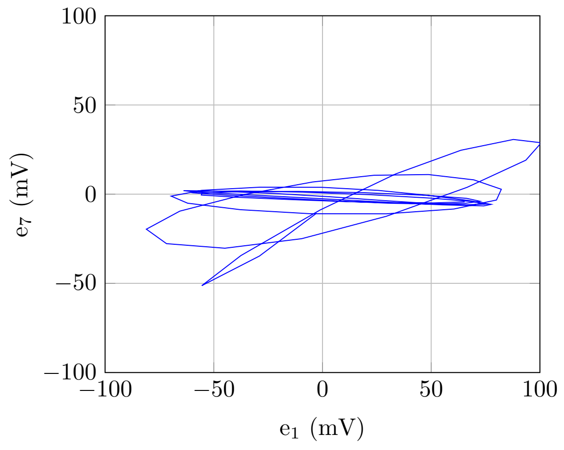
(b)



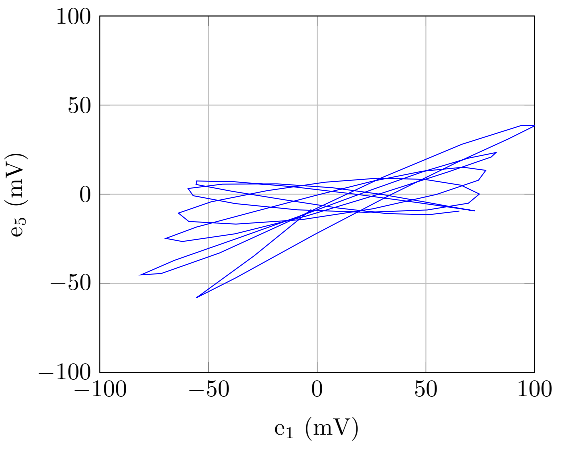
(c)



(d)



(e)



(f)

Figure 4.13: Post layout simulation results of an RKII network with 16 sets. (a) oscillations in sets 1, 3 and 4 (b) state of set 1 versus state of set 2 (c) state of set 1 versus state of set 3 (d) state of set 1 versus state of set 4 (e) state of set 1 versus state of set 7 (f) state of set 1 versus state of set 5.

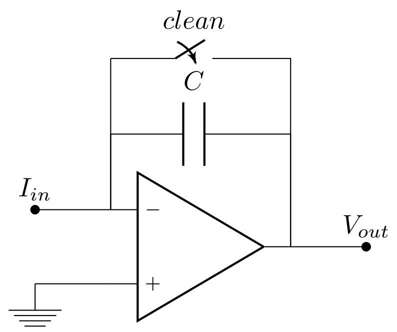


Figure 4.14: Integrator configuration.

In order to test this configuration, a network with three RKII sets was simulated, using the same control signals and parameters as in the previous example. The network simulated used parasitics extracted from layout of all blocks, but ideal interconnections between blocks. In order to simulate the parasitic capacitance that would be introduced by large lines in a practical layout, capacitors of 10 pF were introduced at the amplifier input and output. The curve of interconnection voltage versus sigmoid output current is shown in Figure 4.15. The capacitance used was 1 pF.

As expected, there is a signal inversion. Moreover, the slope obtained is $-1.99\,\mathrm{mV/nA}$. As Δt is $2\,\mu$s, the capacitance effectively used in integration is 1.00 pF. Hence, the effect of parasitic capacitances is successfully canceled.

Another modification to the network may be done observing that transconductance amplifiers in filters are used only when their respecting RKII set is the active one. Thus, it is possible to multiplex transconductance amplifiers, as depicted in Figure 4.16. Transconductance amplifiers would be shared by all RKII sets, and only capacitors would be distributed in the network. This modification would significantly reduce mismatches and area occupied. However, the lines used to connect capacitors and transconductance amplifiers would be long, thus having large parasitic capacitance.

Following the same strategy, the whole filter can be multiplexed with a small modification to the algorithm. This is depicted in Figure 4.17.

The whole filter is shared by all RKII sets. Every set stores its state (in four capacitors, as before), and before the filtering operation, the respective capacitor is connected to a buffer and pre-charges the filter capacitor. During filtering, the filter capacitor is connected to the buffer, and charges the RKII set capacitor to the filter output. In this way, the filter time constant is the same for all RKII sets, thus matching is significantly improved. Moreover, area is significantly decreased. Capacitors in RKII sets are not required to be linear, since they only store voltage and are not used in filtering. This allows MOSCAP to be used. The buffers are also shared by all RKII sets. Moreover, the same buffer may be shared for pre-charging and during filter, requiring only extra switching.

Although pre-charging requires a dedicated time slot, it does not necessarily means that the temporal limit to the number of sets that can be multiplexed is reduced by adopting this strategy.

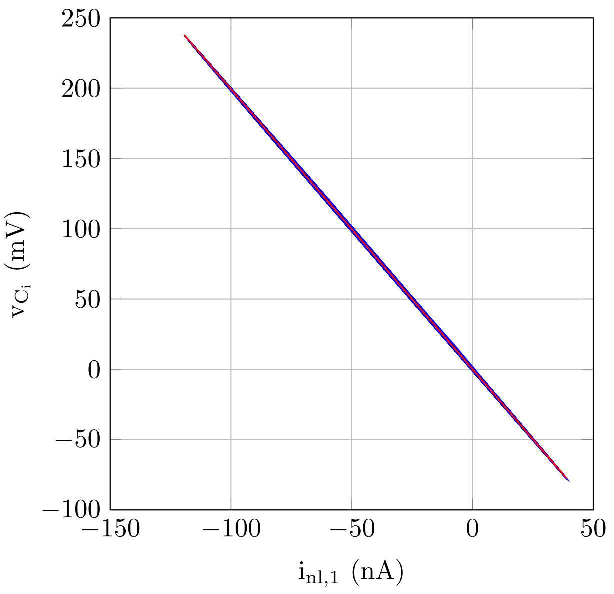


Figure 4.15: Interconnection voltage versus sigmoid output using the integrator in Figure 4.14.

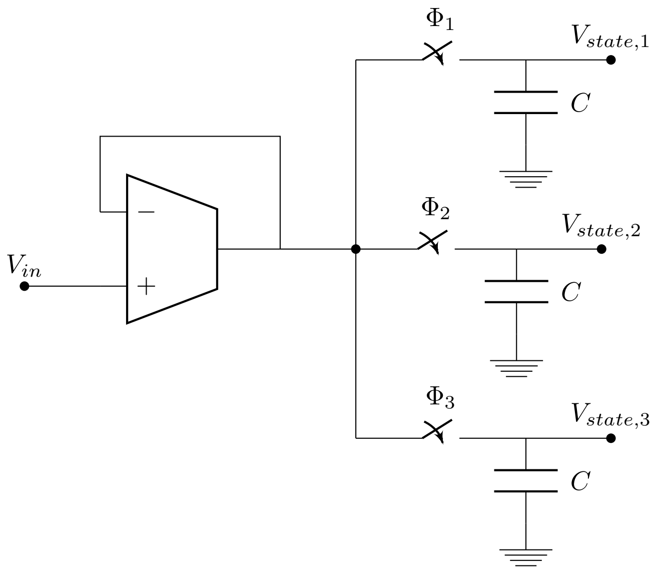


Figure 4.16: Multiplexed transconductance amplifiers.

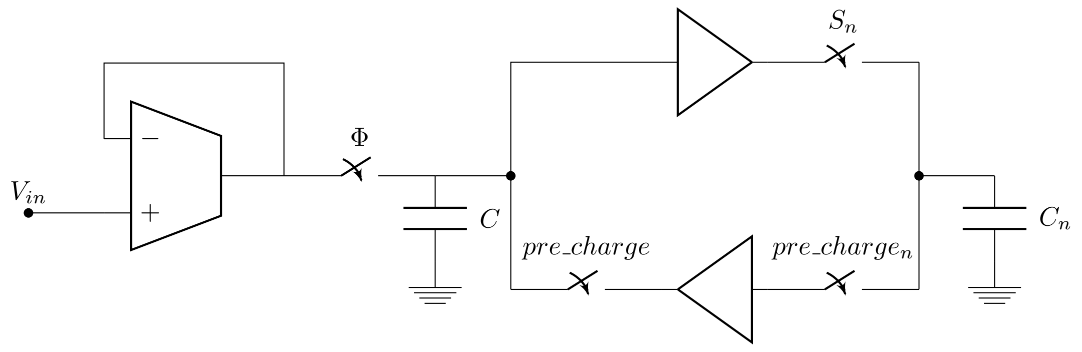


Figure 4.17: Multiplexed filter.

As the state is only required after current integration to obtain the RKII set input, pre-charging may occur concurrently with current integration. In fact, with proper switching, pre-charging can be done in the time frame between the end of the filter slot of the previous RKII set and the beginning of its respective filter slot. Thus, in principle, it will not impose a temporal limit.

A network of three sets using this strategy was simulated, and results were similar to the ones obtained before. Simulations used parasitics extracted from layout from all blocks, but ideal interconnections between blocks. The amplifier used as buffer will be presented in Section 5.1.2.

Simulation results are shown in Figure 4.18. Control signals and parameters were the same as in the simulations presented in Section 4.4, but a pre-charge signal with duration of 1 μs was added. The capacitance used in both interconnection capacitors was 1 pF.

A problem that arises from this configuration that would have to be studied and addressed in a practical implementation is charge sharing between the capacitors in each RKII set and the line parasitic capacitance. In order to reduce its effect, the capacitors should be kept large, but reducing the capacitance is desired in order to reduce area occupation. Nonetheless, this configuration is promising and worth studying, since it would significantly improve matching, area occupation, and power consumption.

4.6 Control Structure

Blocks that constitute the control structure were presented in Section 3.4.4. In this section, they are further described. Blocks were described at RTL level in Verilog and simulated in Mentor Graphics® Questasim. The control structure was implemented and tested in a Xilinx® Spartan-6 FPGA, using a Digilent® Atlys development board and Xilinx ISE Design Suite. Moreover, synthesis was also performed for the target technolog, using Synopsys® Design Compiler. Area and maximum frequency estimation after synthesis for the target technology is shown in Table 4.12. The synthesis assumed a network of 127 RKII sets, with addresses of 7 bits, weights of 7 bits, and registers of 9 bits.

In the FPGA, control structure was implemented using a 100 MHz clock.

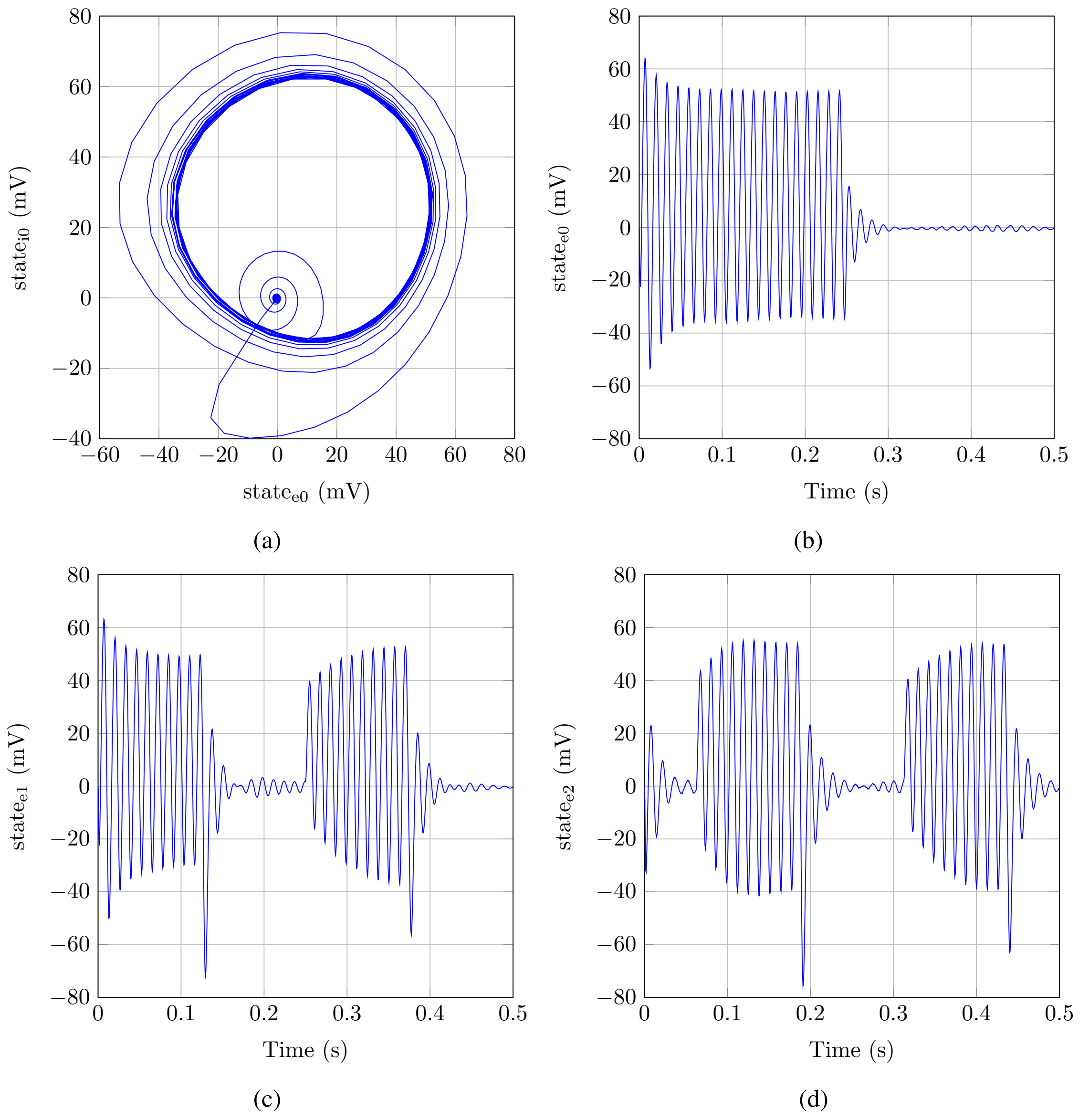


Figure 4.18: RKII network with multiplexed filter simulations. (a) phase portrait for RKII set 1, (b) excitatory state for RKII set 1, (c) excitatory state for RKII set 2, and (d) excitatory state for RKII set 3 .

Table 4.12: Control structure synthesis results.

Block	select_cell	selected_cells_list	time_control	weight_memory
Area	13 313 μm^2	426 321 μm^2	118 256 μm^2	62 990 μm^2
Max. Freq.	200.4 MHz	200.4 MHz	200 MHz	200.4 MHz

Block	local_control
Area	8856 μm^2
Max. Freq.	724.6 MHz

4.6.1 Control Structure Operation

Figure 4.19 shows the time diagram of the most important signals in the control structure, obtained by behavioral simulation. Besides the global control structure, this simulation result includes local control structures of four RKII sets, but only sets 1, 3 and 4 were in the selected cells list. The memory data bus is used for transmitting the active cell address and the scaling factor of W_{ee} to local control structures.

The time control block generates the time signals that control the multiplexed network: $cell\_clock$, tr_upd, $clean\_ce$ and $clean\_ci$; and signals that are processed by local control structures: $filter\_clk$, *wie*, *wei*, *wii* and wee_ref. The duration of these signals is stored in configurable registers, as well as the duration of intervals between signals, such as the time interval between the end of filtering and the end of $cell\_clock$.

At the end of every cell clock, the time control structure activates the *ask_new_cell* signal, that tells the select cell block to select the next active cell. The select cell block tracks the number of slots that were attributed in the current round, and when it is equal to the total number of active cells (which is also stored in a configurable register), it signals that the current slot is the last one in the round, activating the *new_round* signal.

The select cell block sets the $incr\_mem\_addr$ signal to increment the selected cells list memory position, and, in the last slot of the round, it sets the *reset_mem_addr* in order to go back to beginning of the list.

When the *new_round* signal is active in the end of a cell clock, the time control structure sets the tr_upd signal.

The wee_ref signal is controlled by two parameters, both stored in configurable registers: the duration of a pulse, and the number of pulses in each slot. The signal is active during the duration of a pulse, and then is disabled during one clock cycle, and iterates this behavior the desired number of times.

Local control structures sample the memory data bus in the beginning of $cell\_clock$, and verify whether they are the active cell or not. If they are, *wei*, *wie* and $filter\_clk$ are propagated to the local RKII set. Otherwise, only *wii* is propagated. The local signal *wee* is generated according with the weight received in the previous slot, based on the global signal wee_ref. A counter in the local control structure counts the number of pulses in wee_ref, and the local version of *wee* is active during a number of cycles equal to the value received from memory. For instance, in the example in the figure, *wee*0 is active during seven pulses in the slot of set 1, and 2 cycles in the slot of set 3. Moreover, local control structures generate signals *e_to_gnd* and *i_to_gnd*, that set a path from analog memories output to ground, so that current can flow. This signal is disabled when current is used for integration.

Weights are received in the memory data bus. As the active cell address is sampled by local control structures only in the beginning of $cell\_clock$, the remaining time will not affect the operation of the network. After the beginning of $cell\_clock$, the bus is used to distribute weights. Firstly, a cell is addressed in a clock cycle, and in the following clock cycle it receives the weight

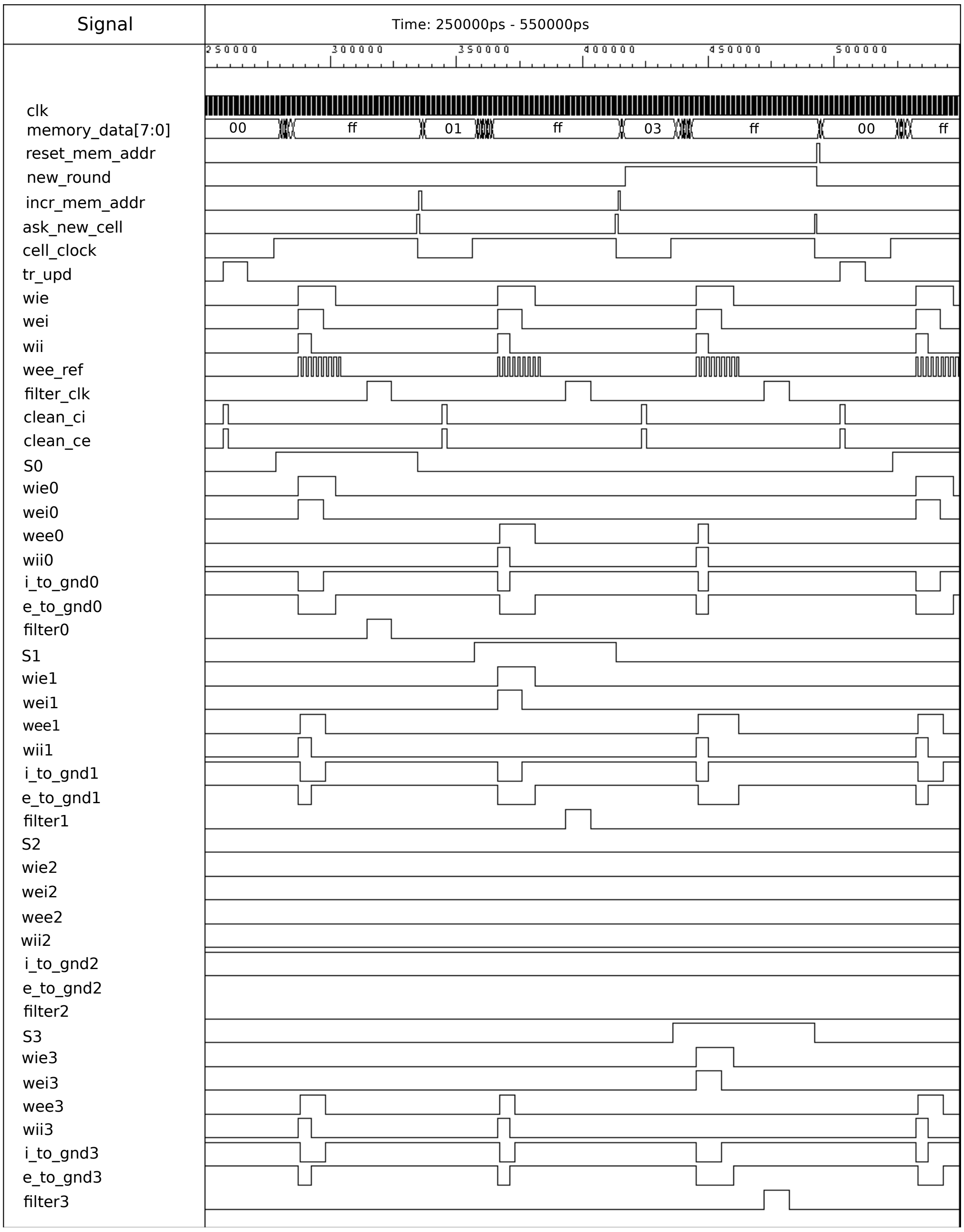


Figure 4.19: Control structure time diagram.

to be used in the following slot. This is iterated to all active cells. If a cell is not addressed, it sets itself to be inactive during the following slot, thus not propagating any signals to the network. This is what happens to set 2 in all slots in the figure. After the weight distribution, all positions in the bus are set to logic one. This value cannot be a valid address in the network, otherwise incorrect behavior could occur.

Chapter 5

BIST Design and System Integration

5.1 BIST Design

5.1.1 Checker Circuit Algorithm

In order to check if the RKII set under test should be added to the selected cells list, a comparison of the parameter under test with the average must be done. A cell passes the test if it does not differ from the average more than a given tolerance. Thus, the comparison tests if a value is within a variable window.

Window comparators are generally used in BIST applications for this purpose. However, most window comparators test if a variable is within a user-programmable window that is not relative to the input signal (i.e. if $v_{in} \in [V_{min}, V_{max}]$) [69–71]. For the desired application, the test required is to check if the difference between two variables is within a user-programmable window (i.e. if $v_{in} \in [v_{avg} - v_{tol}, v_{avg} + v_{tol}]$). This is not directly obtained by common window comparators, thus another approach is required.

An option considered was a static checker circuit, with bias-programmable tolerance [72]. The circuit was simulated, but it was considered unsatisfactory for the target application, due to its limited range of achievable tolerances. Moreover, user control over the tolerance through the bias current is sensitive to process variations and the tolerance value is not directly observable. Another problem is that common auto-zero techniques for offset cancellation would not be directly applicable, causing possibly unbearable mismatch problems.

The approach followed was to use a switched capacitor circuit, based on the one proposed in the original BIST implementation [19]. The original circuit is shown in Figure 5.1. It computes the result in two steps, one checks if the variable being tested is greater than the average plus tolerance, other if it is greater than the average minus the tolerance. The final result is computed digitally. The comparison with the average plus tolerance works as follows: first, the average, v_{avg}, is stored in C_i, then the signal Φ^+ is activated, switching the circuit to the one shown in Figure 5.1b. In this phase, C_{comp} is charged to v_{avg}. In the next phase, Φ^+ is disabled and Φ^+_{avg} is enabled, switching the circuit to the one shown in Figure 5.1c. In this phase, the voltage at v_b changes from 0 V to v_{tol}. As there is no current path available, the voltage change at the capacitor terminals is zero, thus

Table 5.1: Checker circuit control signals.

Control signal	ctr_0	ctr_1	ctr_2	ctr_3	ctr_4	ctr_5
Clock phase	-	-	$\Phi^+_{avg} \vee \Phi^-$	$\Phi^+ \vee \Phi^-_{avg}$	$\Phi^+_{avg} \vee \Phi^-$	$\Phi^+ \vee \Phi^-_{avg}$

the voltage at the comparator block input is $v_{avg} + v_{tol}$. The input voltage, v_{in}, is applied to C_e by current integration. By this process, the comparator inputs are v_{in} and $v_{avg} + v_{tol}$. The comparison with the average minus the tolerance is similar, and the voltages applied to the comparator input after Φ^+_{avg} are v_{in} and $v_{avg} - v_{tol}$.

Although this strategy shows to be effective conceptually, in practical terms it is vulnerable to errors caused by offset in amplifiers and in the comparator. Defining Δ_1 the offset in amplifier 1, Δ_2 that of amplifier 2, and Δ_c the offset of the comparator, the voltage v_a in Φ^+ is $v_{avg} + \Delta_2$, and in Φ^+_{avg} assumes $v_{avg} + \Delta_2 + v_{tol} + \Delta_1$. Hence, the overall comparator offset is $\Delta_c + \Delta_2 + \Delta_1$. In the comparison with the average minus the tolerance, it is $\Delta_c + \Delta_2 - \Delta_1$.

In order to minimize this source of error, an auto-zero technique was further employed to circumvent the offset problem. The circuit utilized for this purpose is shown in Figure 5.2. The relation between the control signals and the clock phases in the original circuit is shown in Table 5.1. The switched circuit in phase Φ^+ is shown in Figure 5.2b, and the switched circuit in phase Φ^+_{avg} is shown in Figure 5.2c. In Φ^+, C_{comp} is charged with Δ_1 at v_b, and with $v_{avg} + \Delta_2$ at v_a. At Φ^+_{avg}, voltage v_b changes from Δ_1 to $v_{tol} + \Delta_1$, thus changing the voltage v_a, at the comparator input, to $v_{avg} + v_{tol} + \Delta_2$. The influence of Δ_1 is thus cancelled. In Φ^+, the comparator is connected as a voltage follower. This is possible because the comparator used is a differential amplifier. In this configuration, its offset voltage, Δ_c is stored in C_e. In Φ^+_{avg}, C_e is connected to amplifier 2 input, and is charged by current integration, changing its voltage to $v_{in} + \Delta_c$. The output of the amplifier 2 is $v_{in} + \Delta_c + \Delta_2$. Therefore, the inputs of the comparator are $v_{avg} + v_{tol} + \Delta_2$ and $v_{in} + \Delta_c + \Delta_2$. Δ_2 is applied at both inputs, thus its effect is cancelled. Δ_c is also applied to the inverting input of the comparator, cancelling the comparator offset. This strategy makes the comparator operation invulnerable to offset in all blocks, with the penalty of having more switches and a longer comparison time, due to the amplifier cascaded at the input.

Using this scheme alone, significant error would occur due to charge sharing between C_{comp} and parasitic capacitances at the input of the comparator, which was previously connected to ground, and between C_e and parasitic capacitances at the input of amplifier 2. Two extra clock phases were added. During the first, ctr_0 alone is active, and the comparator noninverting input is charged with $v_{avg} + \Delta_2$, thus reducing the voltage difference that will be subject to sharing in Φ^+_{avg} or Φ^-_{avg} from $v_{avg} \pm v_{tol} + \Delta_2$ to $\pm v_{tol}$. During the other phase, ctr_1 alone is active, charging the amplifier 2 noninverting input to ground, thus reducing the voltage to difference to be shared in Φ^+_{avg} or Φ^-_{avg} from $v_{avg} - \Delta_c$ to $-\Delta_c$. This phases are included between Φ^+ and Φ^+_{avg}, and between Φ^- and Φ^-_{avg}. The downside of this operation is that more control signals are required and also the BIST control structure complexity is increased.

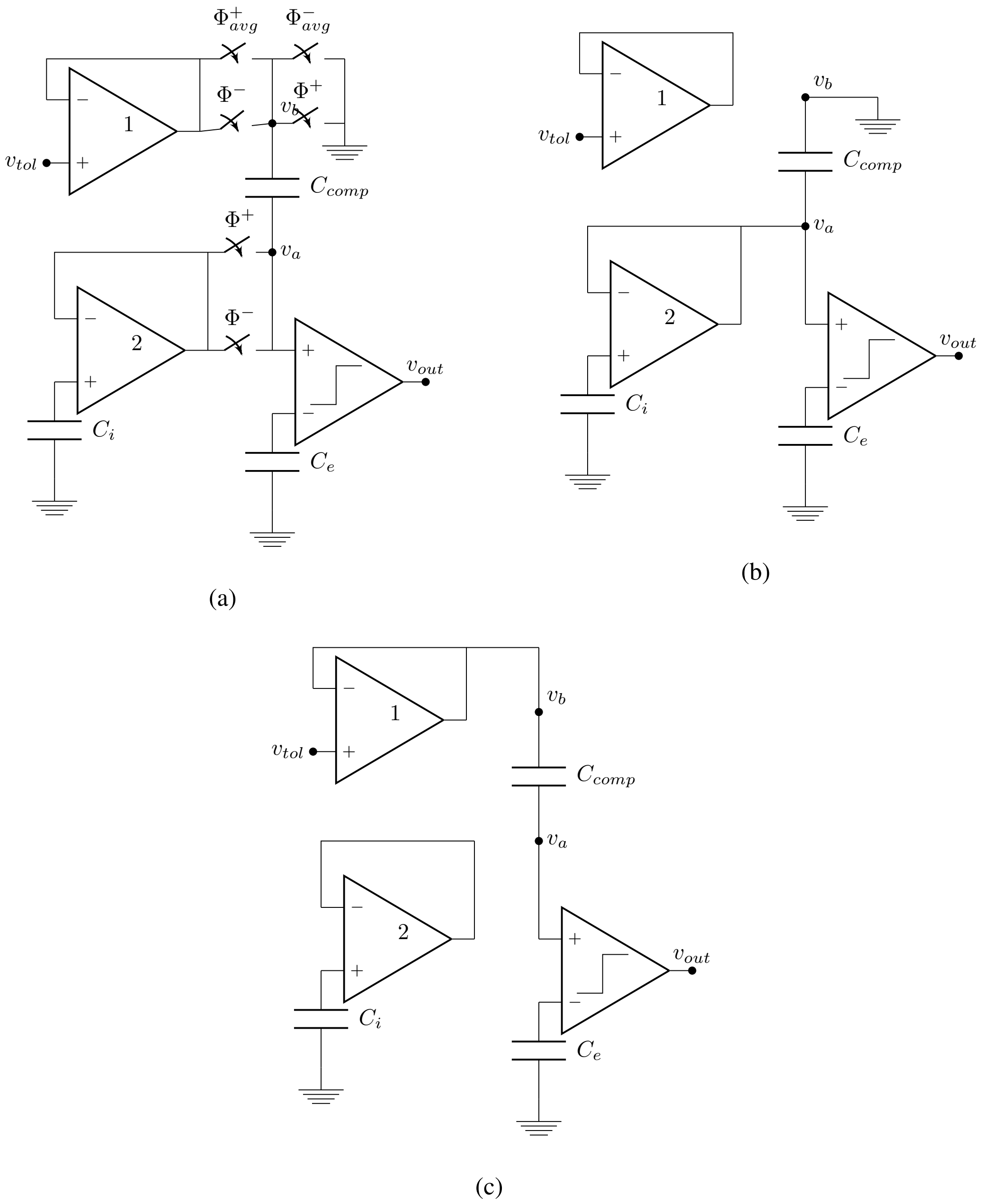


Figure 5.1: Original checker circuit used in the BIST. (a) whole circuit, (b) circuit switched in Φ^{+} phase, and (c) circuit switched in Φ^{+}_{avg} phase.

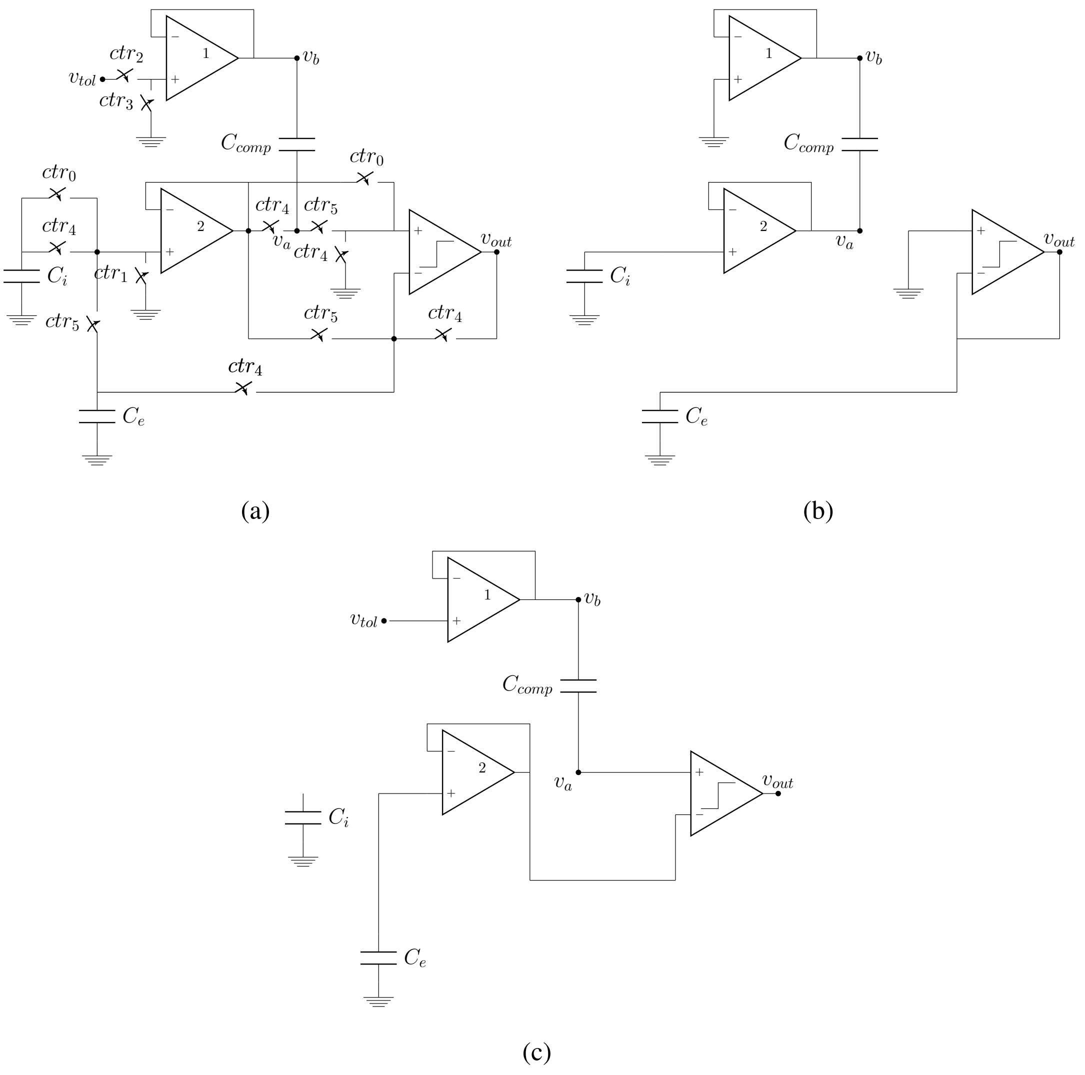


Figure 5.2: Checker circuit proposed in this work. (a) whole circuit, (b) circuit switched in Φ^+ phase, and (c) circuit switched in Φ^+_{avg} phase.

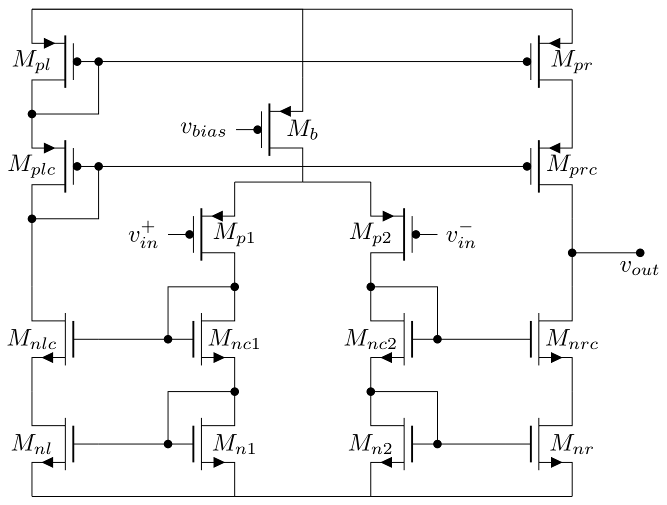


Figure 5.3: Amplifier block schematics.

5.1.2 Amplifier Block Design

All amplifiers used as voltage followers were implemented using the configuration shown in Figure 5.3. A cascode topology was used to increase the gain, in order to reduce voltage error that would affect the comparison precision. Moreover, a wide range configuration was also utilized for maximization of the dynamic range. Table 5.2 shows the final transistor sizes. The bias current was set to 5 μA in M_b, resulting in a total consumption of 10 μA. Layout is shown in Appendix A.2.1.

A static simulation was then performed in order to assess the voltage error when the amplifier is connected in unitary feedback, as it is in the checker algorithm. Figure 5.4a depicts the voltage error versus input voltage. It shows that, apart from a small offset (that is cancelled by the scheme presented before), the voltage error is within a range of 1 mV to absolute values of the input voltage up to above 1 V. In a Monte Carlo analysis, this same results were obtained.

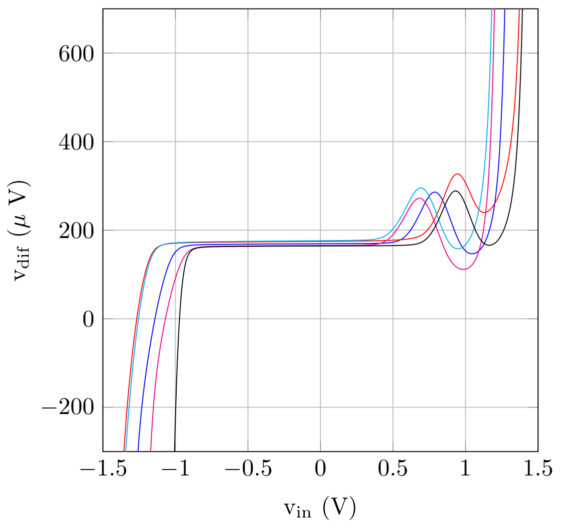
A compensation capacitor was added in order to increase the phase margin. Figure 5.4b shows the value of the phase margin versus the capacitance used. The value was set to 0.8 pF, because it showed a good trade of between phase margin and area.

Figure 5.4c represents the amplifier settling time versus input voltage. Settling time was measured as the time required for the output to achieve 99 % of the final voltage when a step is applied to the amplifier input. The amplifier tested was already compensated with the 0.8 pF capacitor. As it can be seen, performance degrades for higher positive inputs. In principle this is not a major problem, since both the amplitude value and average offset found by Monte Carlo analysis are positive, and the signals will be inverted by the inhibitory sigmoid before the comparison. Hence, values at the amplifier input are expected to be either negative or small enough.

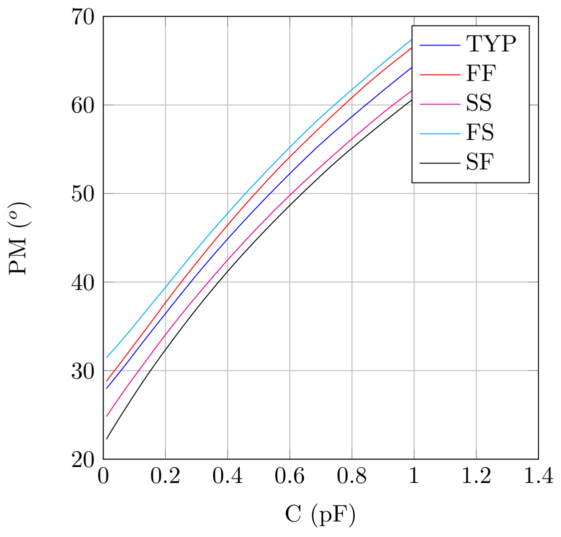
Table 5.2: Amplifier transistor dimensions.

M	M_{pl}	M_{plc}	M_{pr}	M_{prc}	M_{p1}	M_{p2}	M_b
W	$32\,\mu$m	$32\,\mu$m	$32\,\mu$m	$32\,\mu$m	$32\,\mu$m	$32\,\mu$m	$32\,\mu$m
L	$2\,\mu$m	$2\,\mu$m	$2\,\mu$m	$2\,\mu$m	$2\,\mu$m	$2\,\mu$m	$2\,\mu$m

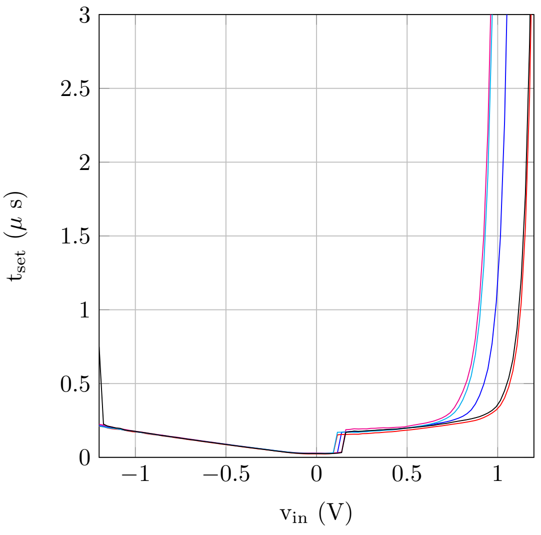
M	M_{nl}	M_{nlc}	M_{nr}	M_{nrc}	M_{n1}	M_{n2}	M_{nc1}	M_{nc2}
W	$32\,\mu$m	$32\,\mu$m	$32\,\mu$m	$32\,\mu$m	$8\,\mu$m	$8\,\mu$m	$16\,\mu$m	$16\,\mu$m
L	$2\,\mu$m	$2\,\mu$m	$2\,\mu$m	$2\,\mu$m	$4\,\mu$m	$4\,\mu$m	$2\,\mu$m	$2\,\mu$m



(a)



(b)



(c)

Figure 5.4: Amplifier simulation results. (a) voltage error versus input voltage, and (b) phase margin versus compensation capacitance (c) settling time versus input voltage.

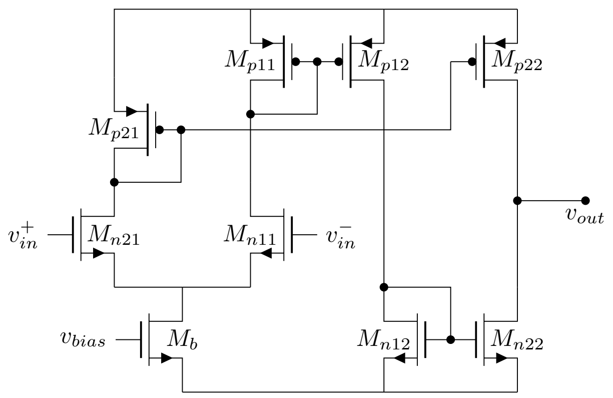


Figure 5.5: Comparator block schematics

5.1.3 Comparator Block Design

An operational amplifier was used as comparator. The configuration assumes the same wide range arrangement, as previous amplifiers, but without cascode, as shown in Figure 5.5 [73]. The decision behind this choice took into consideration the gain in linear region and the operating speed. These aspects are important for achieving high precision in the comparison, in order to enable small tolerances in the checker circuit. A complementary self-biased differential amplifier with very wide common mode range [74] was also simulated and considered for the comparator block, as proposed in the original BIST implementation [19], but its gain was considerably smaller than that under consideration, degrading the precision achieved by the comparison. The final transistor sizes can be checked in Table 5.3. The bias current used is $1\,\mu$A. Layout is shown in Appendix A.2.2.

Figure 5.6a shows the results obtained for the comparator gain under a static test. A high gain in the linear region of the amplifier means that the width of the linear region is lower, which is important to achieve high precision. Moreover, with higher gains, small difference voltages are more amplified, which allows them to be regenerated in following stages. In the present case, gain linearity is not of a major concern because the block will be solely applied for comparator operations. Figure 5.6b shows the offset voltage versus the reference input voltage applied to the comparator. The offset was measured as the input difference that sets the comparator output to 0 V. A constant offset is desired, independently of its value, because it can be canceled by auto-zero procedures, as described before. Results show that the offset is considerably constant in a range from -1.1 V to 1.1 V.

Table 5.3: Comparator transistor dimensions

M	M_{p11}	M_{p21}	M_{p12}	M_{p22}	M_{n11}	M_{n21}	M_{n12}	M_{n22}	M_b
W	$2\,\mu$m	$2\,\mu$m	$16\,\mu$m	$16\,\mu$m	$4\,\mu$m	$4\,\mu$m	$2\,\mu$m	$2\,\mu$m	$2\,\mu$m
L	$2\,\mu$m	$2\,\mu$m	$2\,\mu$m	$2\,\mu$m	$2\,\mu$m	$2\,\mu$m	$2\,\mu$m	$2\,\mu$m	$2\,\mu$m

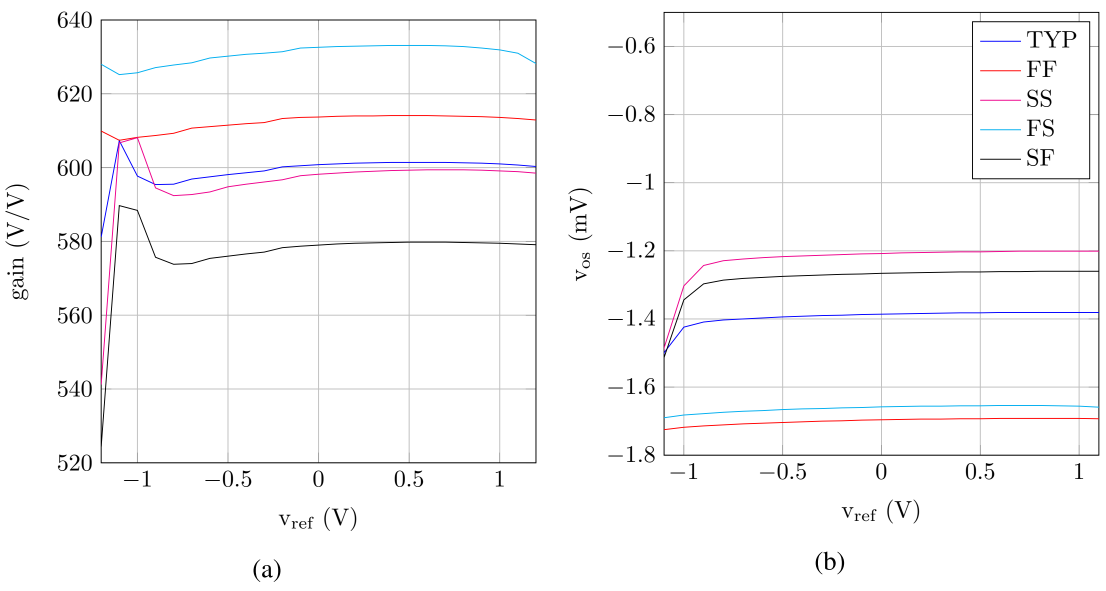


Figure 5.6: Comparator block simulations. (a) gain versus reference input, and (b) offset versus reference input voltage

5.1.4 Checker Circuit Integration

All blocks described so far were integrated in layout. Moreover, a regenerator circuit formed by a comparator block with an input connected to ground and a single CMOS inverter with the dimensions shown in Table 5.4 were also implemented. The regenerator circuit speeds up the comparison time and increases the precision of the comparison. The final circuit with the regenerator is shown in Figure 5.7. All blocks were integrated in layout, and post-layout simulation was performed. The final layout is shown in Appendix A.2.3.

Figure 5.8a and Figure 5.8b show the comparison offset obtained using the checker algorithm. This was measured by sweeping v_{avg} and v_{in} and observing the difference value for which the output toggles its value. The output value was sampled 2 μs after the settling of v_{in}. As observed, the offset of the amplifiers and comparator blocks was successfully canceled, and the gain of the comparator was enough for its output to be successfully regenerated by the following blocks.

In order to further validate the scheme, the amplifiers and the first-stage comparator block were (manually) significantly mismatched, resulting in offsets Δ_c of 13.7 mV, Δ_1 of 8.4 mV and Δ_2 of −7.0 mV. The same test was performed to this circuit, with results shown in figures 5.8c and 5.8d. Again, all offsets were successfully canceled, and a precision of tens to hundreds of μV is achievable.

Table 5.4: Comparator inverter transistor dimensions

M	M_p	M_n
W	8 μm	2 μm
L	2 μm	2 μm

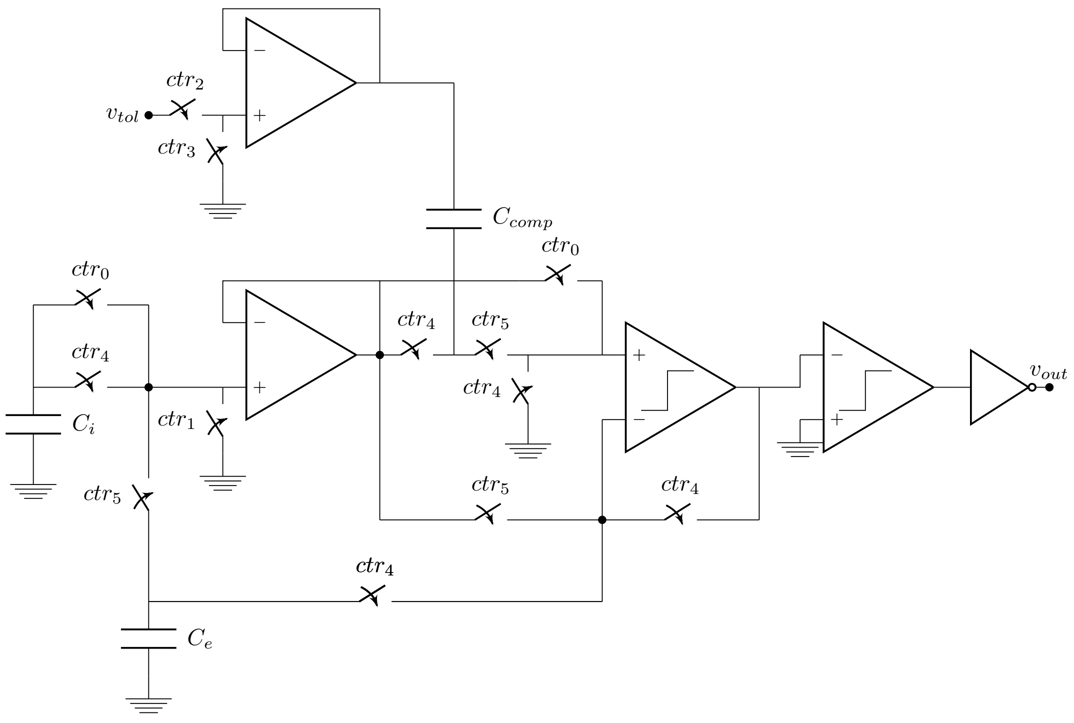


Figure 5.7: Checker circuit with regenerator

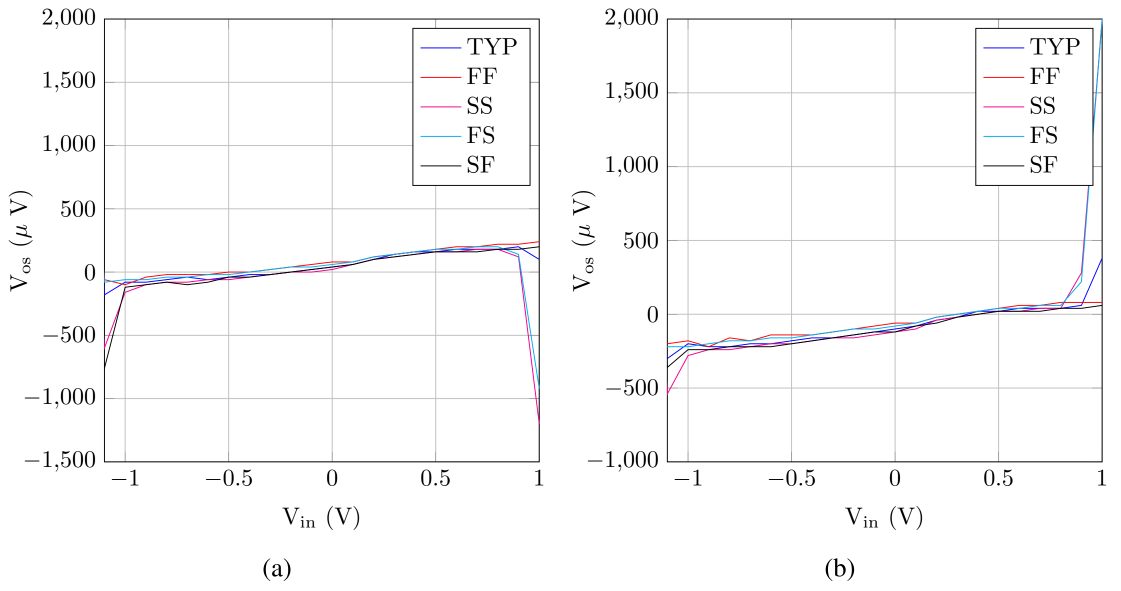
Results presented for all blocks also show that the lower bound of the checker range is imposed by the comparator, and the upper bound is imposed by the amplifier. Moreover, bounds found in the fully integrated circuit match with bounds found in isolated blocks.

5.1.5 Peak Detector Block Design

A conventional peak detector [75] was used, as proposed in the original BIST implementation [19]. Figure 5.9 shows the schematics of the peak detector block.

When the diode-connected transistor (M_d) is conducting, the voltage in C_{pd} is equal to the input voltage by effect of feedback, and when the diode is not conducting, the output voltage stored in C_{pd} is kept constant, as there is no path for it to discharge. If the input voltage is lower than the output voltage, the amplifier output will be low, thus the diode will not conduct. Otherwise, the output of the comparator will be high and the diode will conduct. In this way, the maximum value presented at the block input will be kept at its output until reset, which is done by switching the capacitor to ground.

C_{pd} was set to 1.14 pF, in a trade-off to maximize the capacity of the block to charge quickly, so that it can respond to higher frequency inputs, and to minimize the decay in the output when the diode is not conducting.



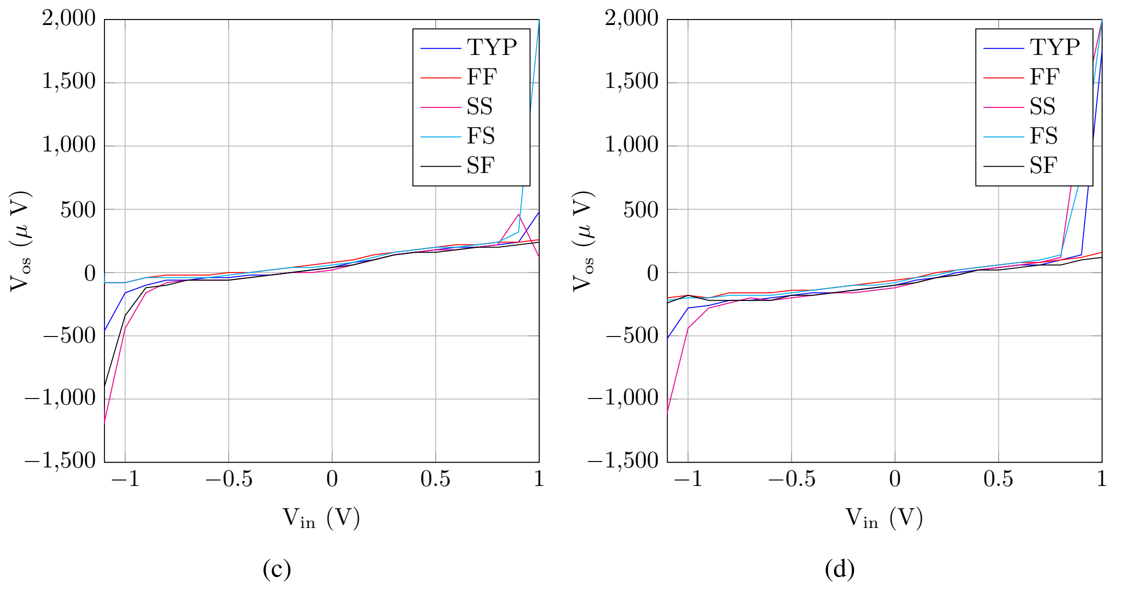


Figure 5.8: Checker algorithm simulations. (a) offset versus reference input in the average minus the tolerance test, (b) offset versus reference input in the average plus tolerance (c) offset versus reference input in the average minus the tolerance with mismatched blocks test, and (d) offset versus reference input in the average plus tolerance with mismatched blocks.

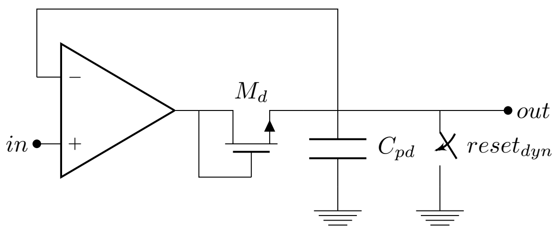


Figure 5.9: Peak detector block

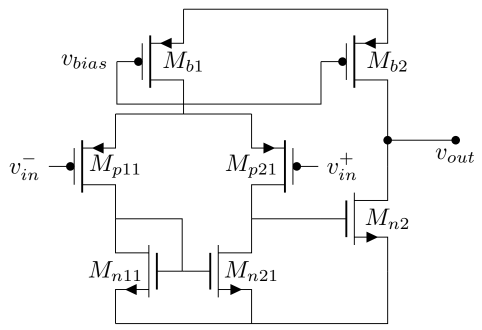


Figure 5.10: Operational amplifier used in peak detector

5.1.6 Peak Detector Amplifier Design

A two stage amplifier was used in the peak detector [75]. The configuration used is shown in Figure 5.10. This configuration was preferred to the one already designed for the amplifier in the checker block because simulations using that amplifier showed poor performance due to coupling between the input and the output.

The transistor dimensions used for this block are shown in Table 5.5. The bias current used is 1 μA in M_{b1} and 10 μA in M_{b2}, resulting in a total consumption of 11 μA. Layout is shown in Appendix A.2.4.

5.1.7 Peak Detector Integration

The peak detector layout is shown in Appendix A.2.5. It was tested by applying a sinusoidal input and observing the peak value stored. Figure 5.11 shows the error value obtained between the measured output of the peak detector circuit and the actual amplitude of the sinusoidal signal. As seen in the figure, there is a small offset, but it is constant. This does not introduce any problem to the system operation, since its only consequence is that an equal offset is added to every cell under test, hence having no effect on the final result of the BIST operation.

5.1.8 BIST Control Structure

The BIST control structure controls the network operation when in BIST mode. It switches the proper paths for both offset and dynamic test. Moreover, it controls the switches of the checker and samples the comparison result at the right time. As the remaining blocks of the control structure, the BIST control structure was designed in Verilog at RTL level, and simulated in Questasim.

Table 5.5: Peak Detector amplifier transistor dimensions

M	M_{p11}	M_{p12}	M_{n11}	M_{n12}	M_{n2}	M_b1	M_{b2}
W	2 μm	2 μm	2 μm	2 μm	20 μm	2 μm	20 μm
L	2 μm	2 μm	2 μm	2 μm	2 μm	2 μm	2 μm

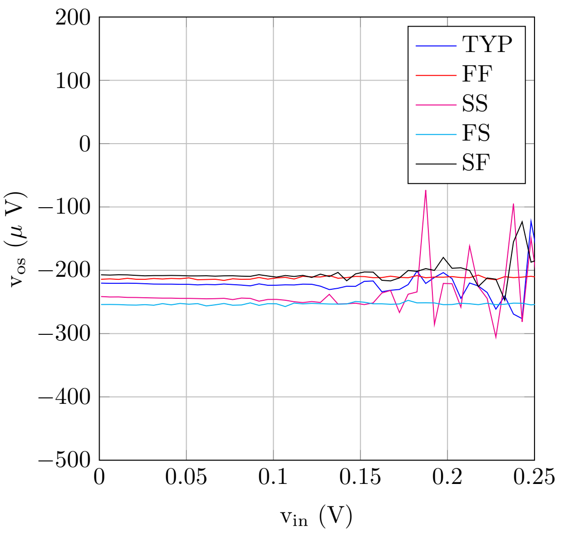


Figure 5.11: Peak detector offset voltage.

The BIST control structure keeps a memory of dimensions $N \times 1$, which is a list of cells selected in current round. The test ends when the list is unchanged in one round, and cells that are active in that list are the cells that passed the test.

In both tests, the parameter being tested (either the offset or an oscillation amplitude) in a cell is compared with the average of that parameter in every cells in a list. Both the output of the current cell and the average are voltage signals, obtained by current integration. In order to compute the average, the current of each cell is integrated during a defined amount of time, and for the output of a single cell, integration time is scaled by a factor given by the number of cells used to compute the average.

During the BIST operation, the BIST structure controls the selected cells list, and at the end of the test, the register that contains the number of active cells is set to the number of cells that passed the test, and the list is then filled with cells that passed the test.

At the beginning of BIST operation, the selected cells test is filled with the address of every cell in the network. Afterwards, the BIST structure starts an iterative process: every cell in the list is addressed and the output of that cell is integrated to the respective capacitor. As the *clean* signal of that capacitor is not activated after every slots, the outputs of every cell are summed. After that, the list of active cells is restored to a list filled with every cell in the network, and the output of every cell is integrated. Now, integration time is scaled by a factor given by the number of cells selected in the previous step. Values in both capacitors are checked in order to assess if they differ more than the tolerance, and the checker result is sampled by the BIST structure. If the difference is lower than the average, the cell is added to the list, otherwise it is not. This is iterated for every cell in the selected cells list. A register is set to zero at the beginning of every round of the algorithm, and set to one if a cell changes its activity state in the list in a round. Convergence occurs when that register is zero at the end of a round. The algorithm is performed first for the offset test, and afterwards for the dynamic test. In the dynamic test, only cells that passed the

Table 5.6: Area occupation and current consumption of analog blocks.

Block	RKII set	Sigmoid	Amplifier	Peak Detector	Checker Circuit
Area	$21\,896\,\mu\text{m}^2$	$1017.1\,\mu\text{m}^2$	$2866.1\,\mu\text{m}^2$	$1589.5\,\mu\text{m}^2$	$13\,808\,\mu\text{m}^2$
Current	$10\,\mu$A	$0.6\,\mu$A	$10\,\mu$A	$11\,\mu$A	$54\,\mu$A

offset test are considered, thus a list of cells that passed the offset test is kept. As the list of cells in the current round, this is a $N \times 1$ memory. The test finishes when both offset and dynamic test are over.

The BIST control structure was synthesised for the target technology. Synthesis predicted an area occupation of $169\,606\,\mu\text{m}^2$ and a maximum frequency of 205 MHz.

In order to test the BIST control structure, distributions were generated randomly in Matlab for the offset and for the dynamic test. The Verilog testbench developed reads the distributions from a file and generates the ideal output of the comparison. To do so, the testbench computes the average of the parameter being tested in every round of the algorithm, and, in every slot, it observes the difference between the average and the parameter for the cell that is active in that slot and it compares this value with the tolerance (which is a parameter of the testbench). It generates the result of the comparison, which is used by the BIST control structure. The operation of the algorithm was compared with the operation of the same algorithm in Matlab, presented in Section 3.3, and results obtained were the same.

5.2 System Integration

After the study and design of all blocks in the system, integration can be studied in order to assess the required pads and the number of RKII sets that can be included in a chip. As it was observed, implementation of the global control structure and memories on chip reduces the number of RKII sets that can be multiplexed, and its implementation on a reconfigurable platform such as an FPGA is a good idea for a prototype version of the system, since it allows easy and costless reconfiguration.

Table 5.6 shows area occupation and current consumption of the analog blocks. Moreover, synthesis of the local control structure estimated a power consumption of $421.7\,\mu$W. The area of a pad in the target technology is $33\,450\,\mu\text{m}^2$.

The following pads are required for the operation of the prototype version:

Analog power:

- $V_{dd,a}$
- $V_{ss,a}$
- gnd

Digital power:

- $V_{dd,d}$
- $V_{ss,d}$

Analog bias:

- V_{bl,sig_e}
- V_{bh,sig_e}
- V_{bl,sig_i}
- V_{bh,sig_i}
- $V_{b1,mem}$
- $V_{b2,mem}$
- $V_{b,gm}$
- $V_{b,pd}$
- $V_{b,amp}$
- $V_{b,comp}$

Network Control

- $cell\_clock$
- $filter\_clock$
- wie
- wei
- wee_ref
- wii
- tr_upd
- $clean\_ce$
- $clean\_ci$

Digital Control

- $reset$
- clk
- $data\_bus$

BIST Control

- dyn_test
- dyn_reset
- v_{tol}
- $v_{out,checker}$
- ctr_0
- ctr_1
- ctr_2
- ctr_3
- ctr_4
- ctr_5

Analog input

- i_{in}

Analog output

- $state_e$
- $state_i$
- v_{C_e}
- v_{C_i}

The $data\_bus$ length is given by the maximum number of bits between the cell address and the weight. For the system synthesised presented in Chapter 4, 7 bits were used, thus 7 pads are required. The number of pads could be reduced, hence decreasing area occupation, by introducing in the chip a decoder, and encoding the BIST control signals (ctr_0 to ctr_5) with 3 bits. Moreover, the $data\_bus$ could be serialized.

In the system implementation proposed, 48 pads are required. Moreover, two sigmoids are used (one inhibitory and one excitatory), two amplifiers are used in the integrator for the interconnections, and the capacitors (0.75 μF) used in the interconnection occupy 789 μm^2 each. As the area is limited to 5 mm^2, a limit of 111 RKII sets is estimated by the area of the several blocks. The real limit will be lower, due to interconnections.

Based on the results presented, the main sources of power consumption are the local control structures.

Considering time operation, with a sampling frequency of 2 kHz, considering a network of 111 RKII sets, 4.5 μs is available per slot. Considering a settling time of 2.3 μs for the sigmoid and memory, a filter duty cycle of 0.5 μs, and 0.1 μs for interconnection capacitor resetting and other time intervals, around 1.6 μs are available to current integration, which is enough for proper operation.

Chapter 6

Conclusions and Future Work

Biologically inspired algorithms have become quite popular over the last years, outperforming sequential digital computers in tasks such as pattern recognition. In this work, a system implementation for a multiplexed RKII network has been studied and proposed. The RKII network models the olfactory bulb and is part of a higher level structure, the KIII, that models the olfactory neural system.

As the usability of the KIII for practical problems has already been proved, a real-time analog implementation will allow to further extend its study, which is limited by simulation time when using digital computers.

The implementation proposed this work mitigates the problem of interconnection scalability by using time multiplexing and current integration.

Previous implementations of RKII networks were feasible with only a few RKII sets. In this work, a scalable implementation is studied and proposed. Results show that around hundred RKII sets can be included in a small chip of 5 mm^2 using a 350 nm technology.

Relatively to the original multiplexed RKII network algorithm, the number of capacitors used for interconnections was reduced from four to two. This eliminated the need for voltage adders. Moreover, an integrator configuration using an operational amplifier is used for interconnections, which mitigates the problem caused by large parasitic capacitances introduced by long lines. A scheme where the filter is fully multiplexed is also proposed. This scheme is expected to significantly reduce mismatches and area occupied by an RKII set, allowing more RKII sets to be included in a network.

Due to uncertainties and nonidealities of the fabrication process, mismatches will certainly occur in the network. For this reason, a BIST scheme was also implemented. Relatively to the original scheme, the checker algorithm was improved in order to cancel offset.

The operation of both the RKII network and the BIST is based on analog blocks, but both require digital control structures to operate. Both analog and digital blocks were implemented. Full-custom layout was designed to analog blocks, and digital blocks were described in Verilog at RTL level, synthesised and simulated. The control structure of the RKII network was implemented successfully in an FPGA.

6.1 Future work

In this work, the layout of an RKII cell (and its connection to a network) was fully designed, as well as the layout of the sigmoids, the checker circuit and the peak detector. The next step is to integrate these blocks in layout, as well as the pads, capacitors and local control structures. After this integration, the system is ready to production. Testing the system in hardware would be important to completely validate the implementation of the network and the BIST.

A system implementation with the fully multiplexed filter proposed is also an important step in future work, since area and mismatches are significantly reduced.

The RKII network is part of a higher level structure, the KIII. The KIII is shown to outperform a KII network in pattern recognition, and the chaotic behavior of the KIII has interesting properties. Hence, it would be highly interesting to include the RKII network implementation in a higher level implementation of a KIII.

Using real-time implementations of the RKII network, such as the one proposed in this work, properties of the Freeman model can be further studied, as well as its application to engineering problems. Real-time hardware implementations of the model are an enabling technology for this study, because model simulations in sequential digital computers are computationally heavy, and thus time consuming.

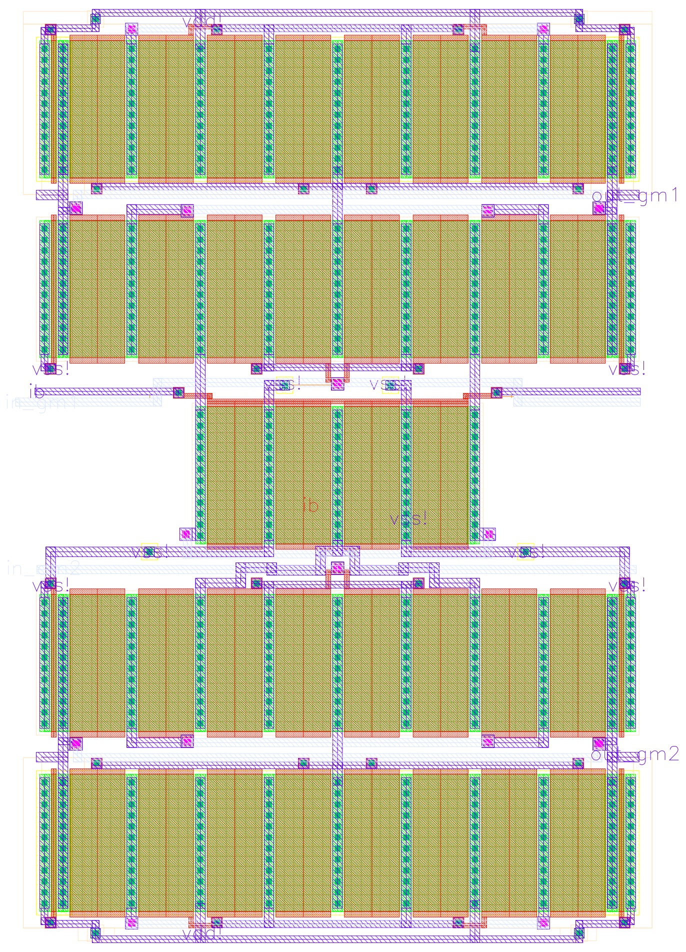
Appendix A

Layout Plots

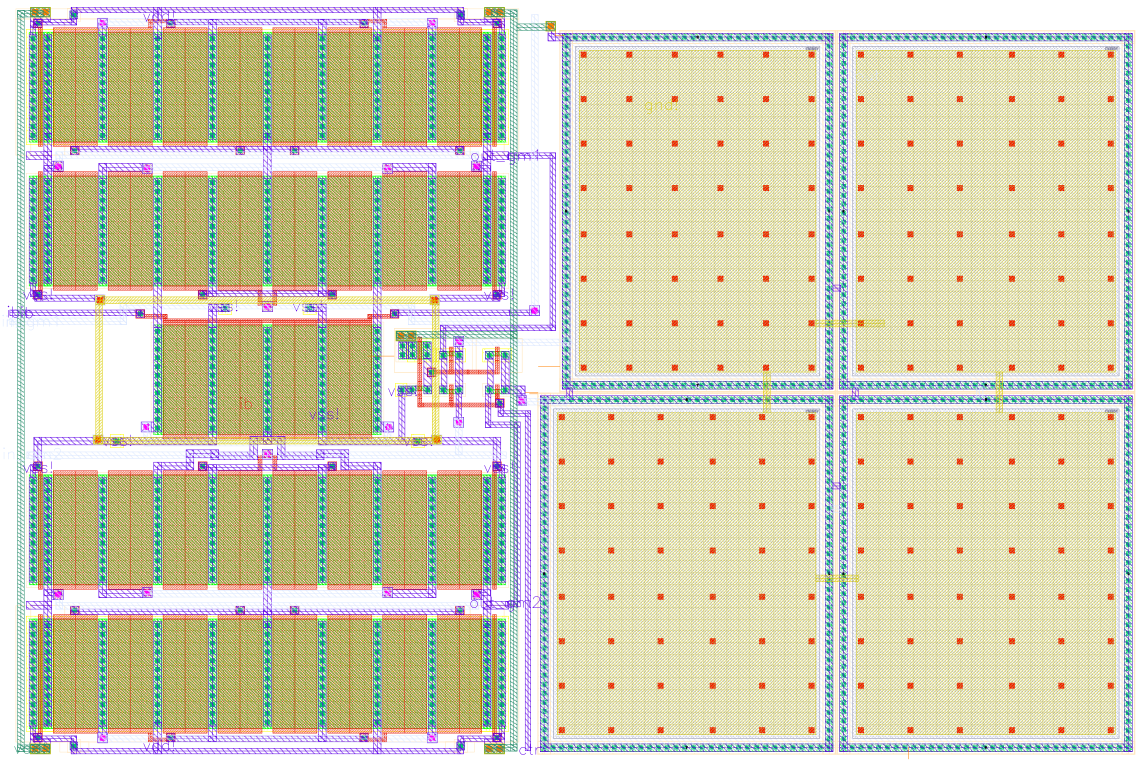
A.1 RKII Network

A.1.1 Transconductance amplifier

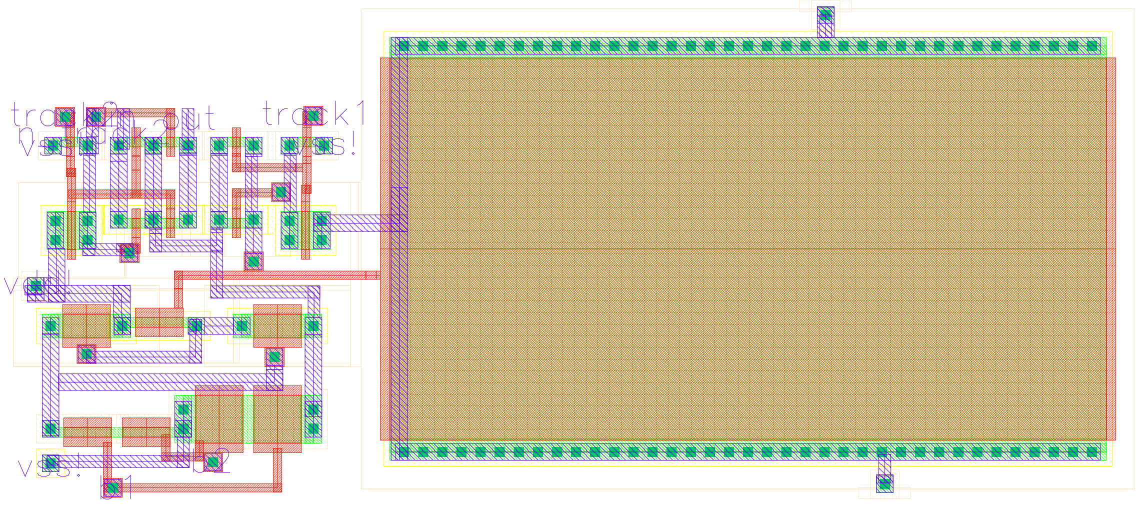
Note that two transconductance amplifiers are together in layout.



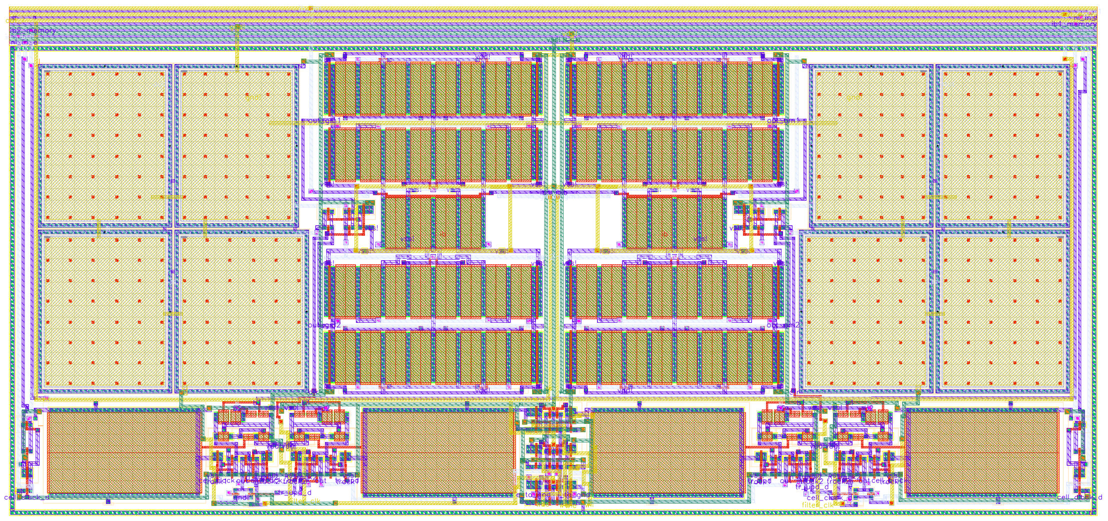
A.1.2 Filter



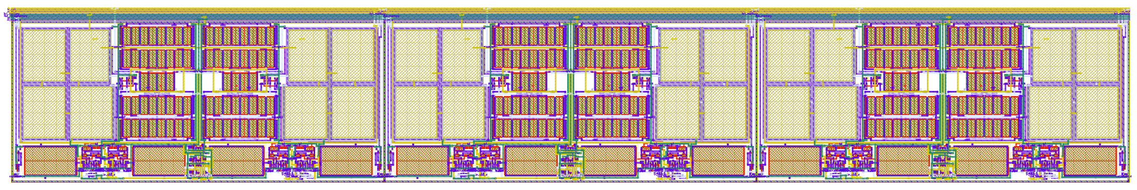
A.1.3 Delay block



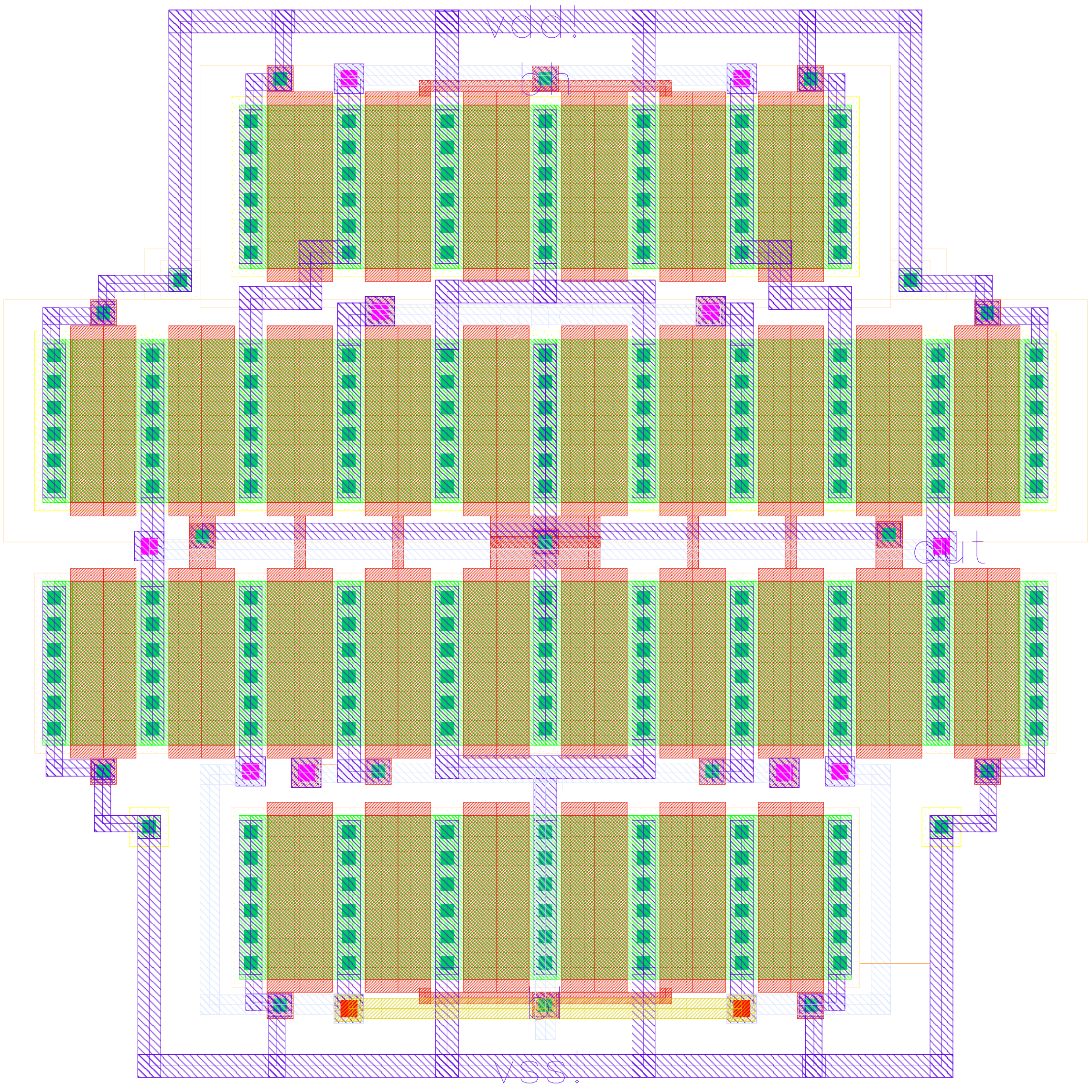
A.1.4 RKII set



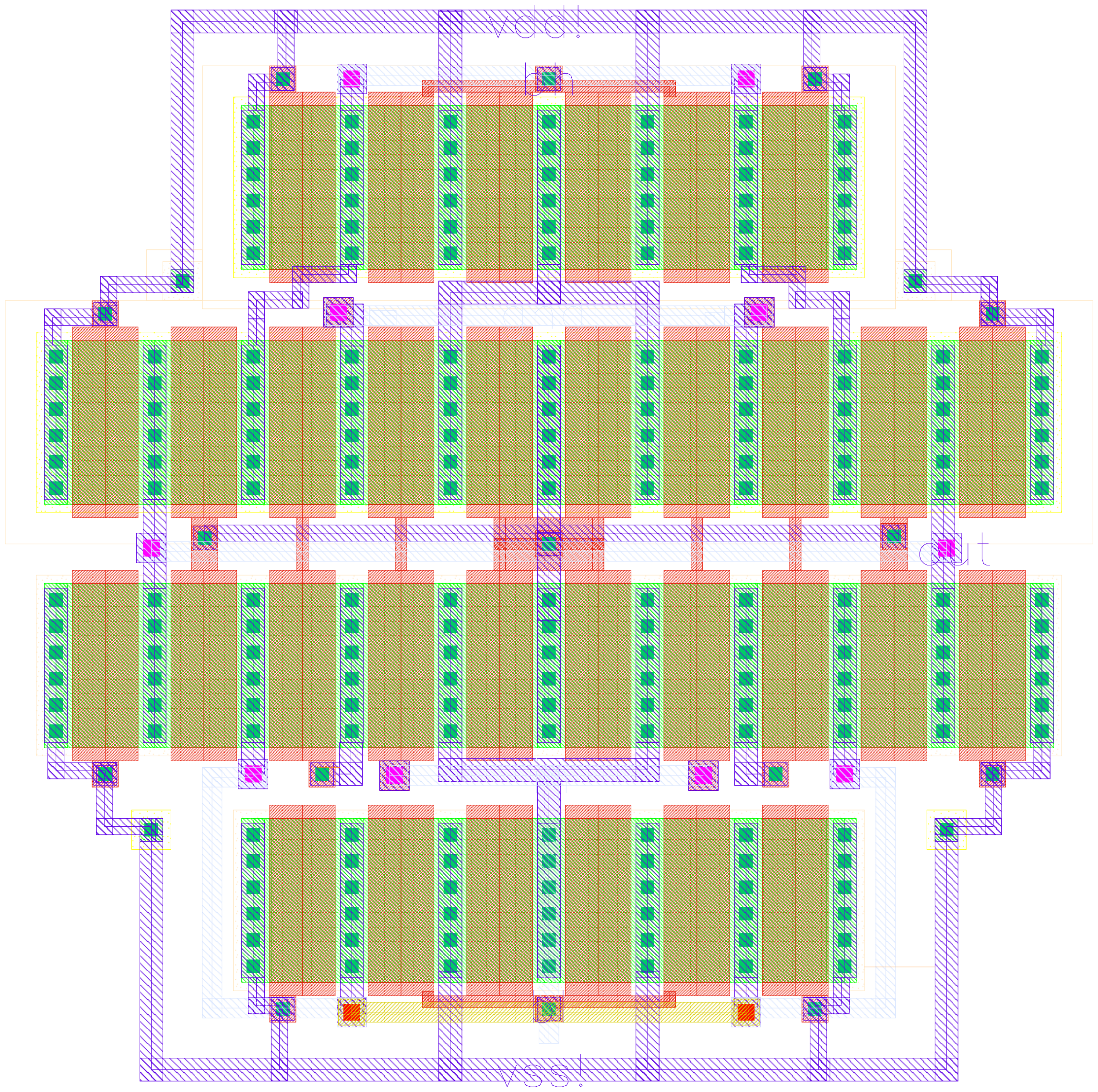
A.1.5 RKII network with three sets



A.1.6 Excitatory sigmoid

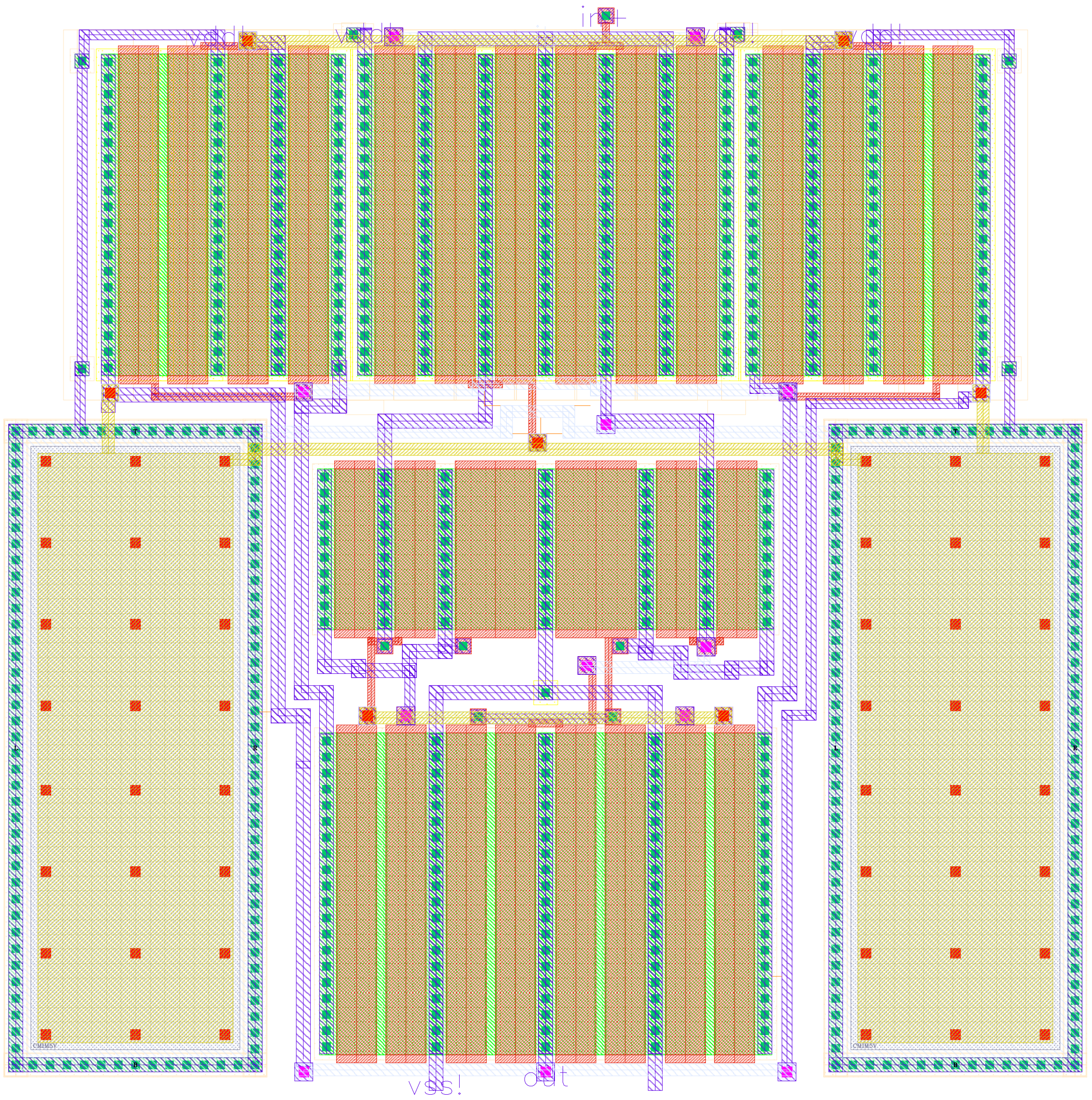


A.1.7 Inhibitory sigmoid

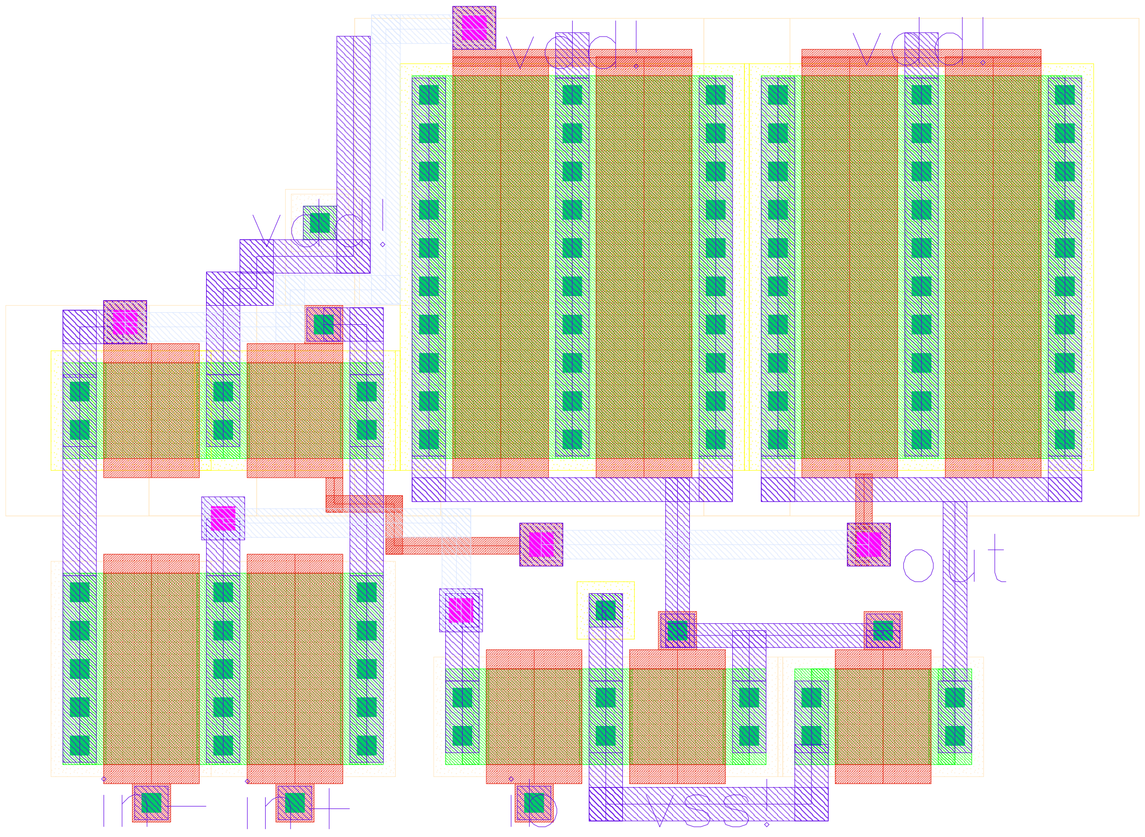


A.2 BIST

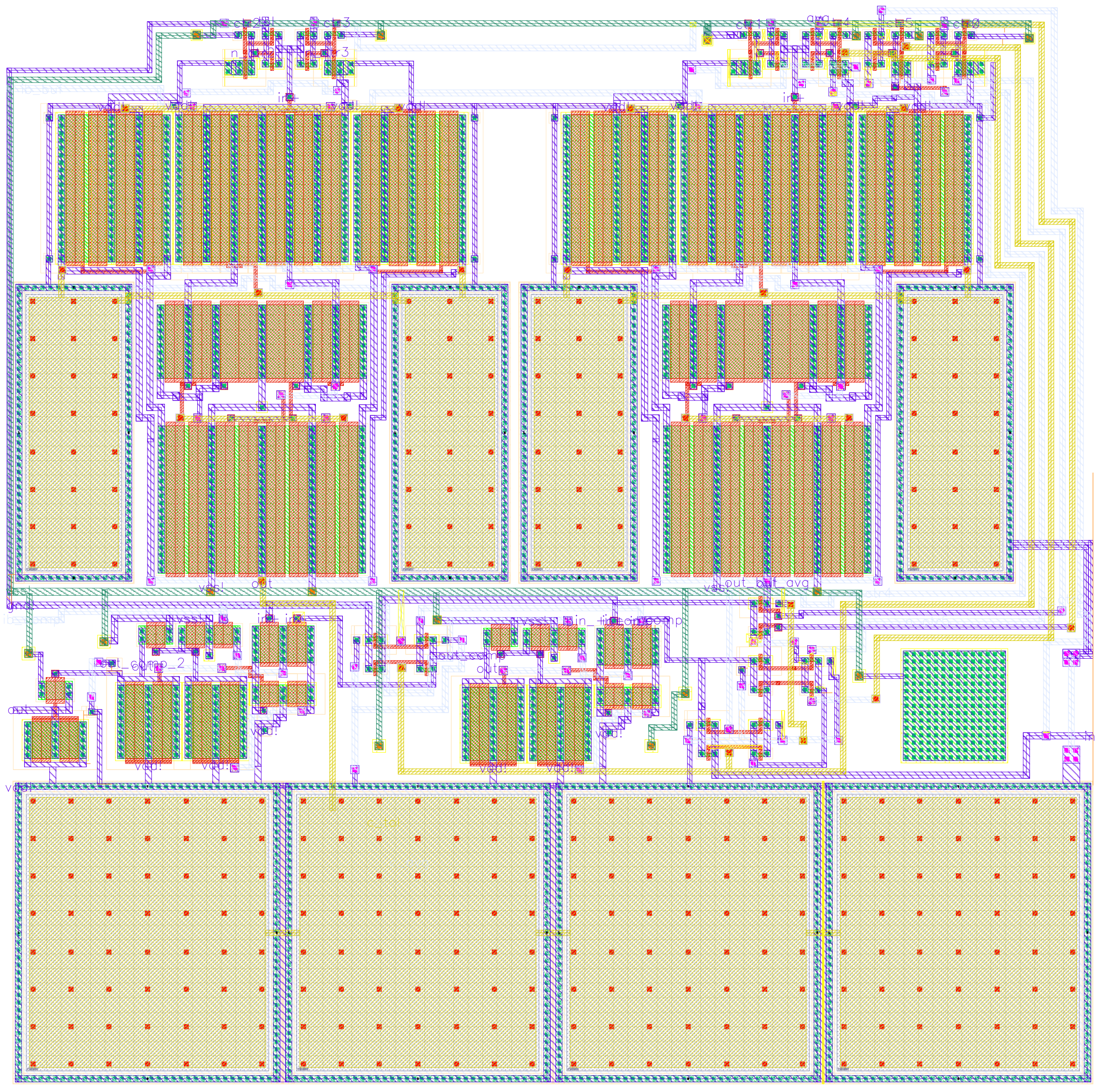
A.2.1 Amplifier



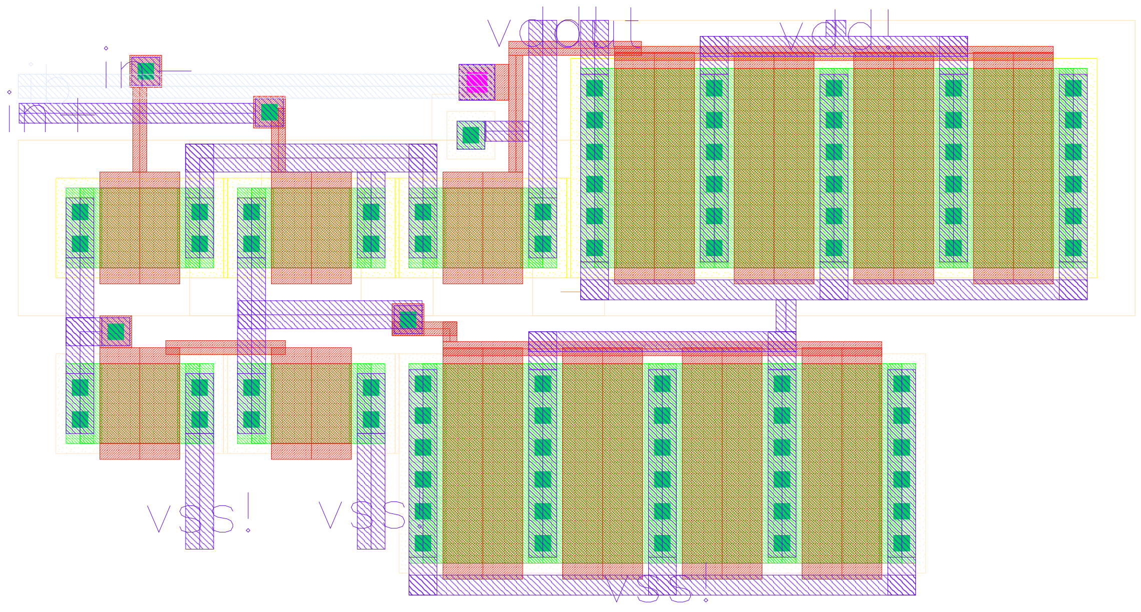
A.2.2 Comparator



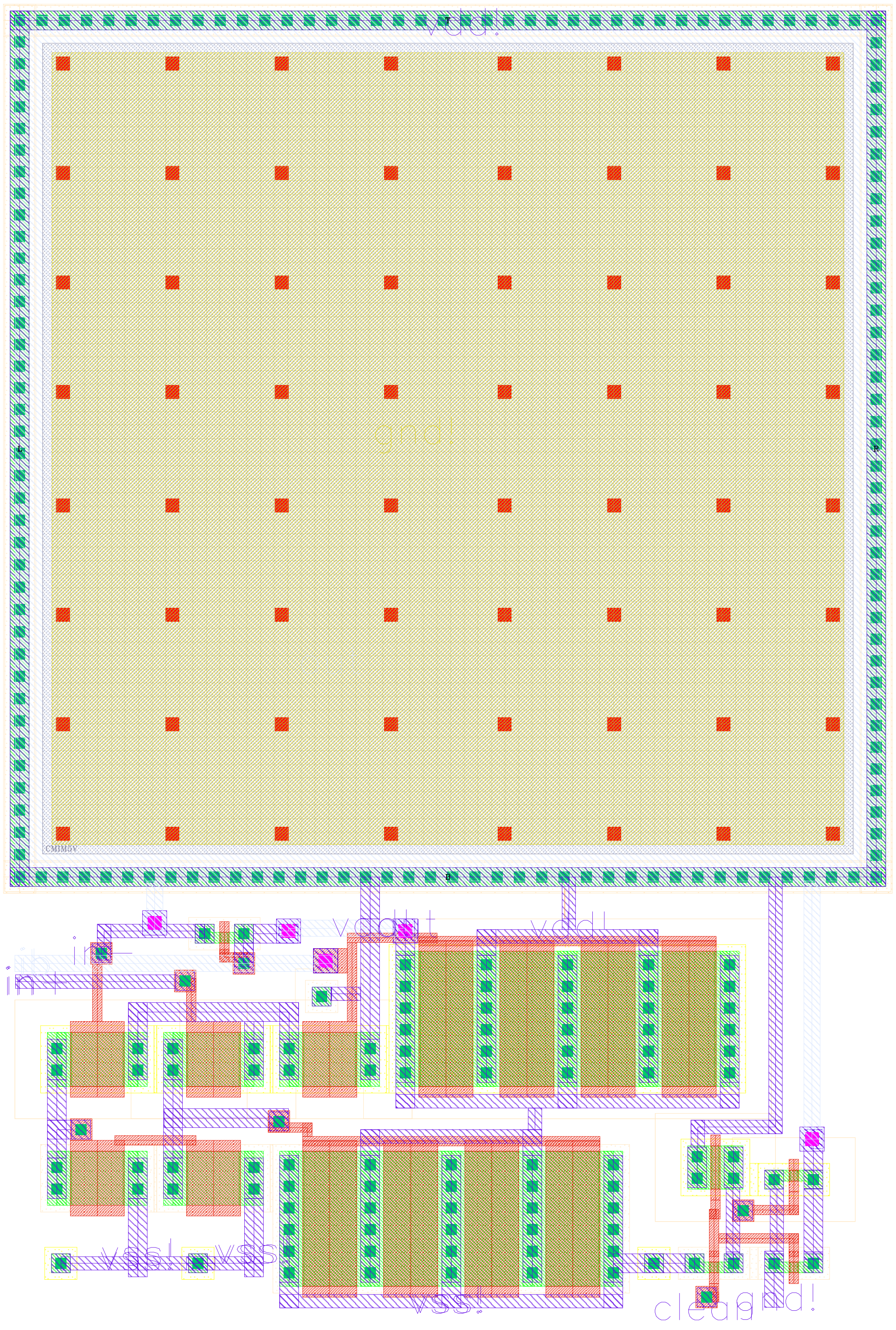
A.2.3 Checker circuit



A.2.4 Peak Detector amplifier



A.2.5 Peak Detector



References

[1] V. Tavares, S. Tabarce, J. Principe, and P. de Oliveira, "Freeman olfactory cortex model: A multiplexed KII network implementation," *Analog Integrated Circuits and Signal Processing*, vol. 50, no. 3, pp. 251–259, 2007.

[2] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999.

[3] E. Kandel, *Principles of Neural Science*, 5th ed., ser. Principles of Neural Science. McGraw-Hill Education, 2013.

[4] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[5] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, 1958.

[6] W. J. Freeman, *Mass Action in the Nervous System.* Academic Press, Oct. 1975.

[7] R. L. Beurle, "Properties of a mass of cells capable of regenerating pulses," *Philosophical Transactions of the Royal Society of London Series B Biological Sciences (1934-1990)*, vol. 240, 1956.

[8] J. Griffith, "A field theory of neural nets: I: Derivation of field equations," *The bulletin of mathematical biophysics*, vol. 25, no. 1, pp. 111–120, 1963.

[9] P. Broca, "Remarque sur le siege de la faculté du language articulé, suivie d'une observation d'aphémie (perte de la parole)," *Bulletin de la société anatomique de Paris*, vol. 36, pp. 330–356, 1861.

[10] K. S. Lashley, "In search of the engram." in *Physiological mechanisms in animal behavior. (Society's Symposium IV.).* Oxford, England: Academic Press, 1950, pp. 454–482.

[11] T. Sejnowski, C. Koch, and P. Churchland, "Computational neuroscience," *Science*, vol. 241, no. 4871, pp. 1299–1306, 1988.

[12] E. M. Izhikevich, *Dynamical systems in neuroscience: the geometry of excitability and bursting*. MIT press, 2007.

[13] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

[14] J. J. Hopfield, "Neurocomputing: Foundations of research," J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1982, ch. Neural Networks and Physical Systems with Emergent Collective Computational Abilities, pp. 457–464.

[15] R. Schalkoff, *Artificial neural networks*, ser. McGraw-Hill series in computer science: Artificial intelligence. McGraw-Hill, 1997.

[16] C. A. Skarda and W. J. Freeman, "How brains make chaos in order to make sense of the world," *Behavioral and Brain Sciences*, vol. 10, no. 2, p. 161, 1987.

[17] W. J. Freeman, Y. Yao, and B. Burke, "Central pattern generating and recognizing in olfactory bulb: a correlation learning rule," *Neural Networks*, vol. 1, no. 4, pp. 277–288, 1988.

[18] J. Eisenberg, W. J. Freeman, and B. Burke, "Hardware architecture of a neural network model simulating pattern recognition by the olfactory bulb," *Neural Networks*, vol. 2, no. 4, pp. 315–325, Jun. 1989.

[19] C. Duarte, H. Cavadas, P. Coke, L. Malheiro, V. Tavares, and P. Guedes de Oliveira, "BIST design for analog cell matching," in *17th IEEE European Test Symposium (ETS)*, May 2012, pp. 1–6.

[20] V. Tavares, J. Principe, and J. Harris, "A silicon olfactory bulb oscillator," in *IEEE International Symposium on Circuits and Systems, 2000. Proceedings*, vol. 5, 2000, pp. 397–400 vol.5.

[21] V. G. Tavares, "Design and implementation of a biologically realistic olfactory cortex model," Ph.D. dissertation, University of Florida, Gainesville, May 2001.

[22] J. Principe, V. Tavares, J. Harris, and W. Freeman, "Design and implementation of a biologically realistic olfactory cortex in analog VLSI," *Proceedings of the IEEE*, vol. 89, no. 7, pp. 1030–1051, Jul 2001.

[23] D. Xu, L. Deng, J. Harris, and J. Principe, "Design of a reduced KII set and network in analog VLSI," in *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03*, vol. 5, May 2003, pp. V–837–V–840 vol.5.

[24] D. Xu, "Dynamical analysis, applications, and analog implementation of a biologically realistic olfactory system model," Ph.D. dissertation, University of Florida, Gainesville, May 2005.

[25] T. A. Holz and J. G. Harris, "Towards a spiking VLSI implementation of Freeman's olfactory model," in *Proceedings of the 2004 11th IEEE International Conference on Electronics, Circuits and Systems, 2004*. IEEE, 2004, pp. 199–202.

[26] D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986.

[27] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, Nov 2012.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[29] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2014.

[30] J. J. Hopfield, "Neurocomputing: Foundations of research," J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, ch. Neurons with Graded Response Have Collective Computational Properties Like Those of Two-state Neurons, pp. 577–583.

[31] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[32] M. I. Jordan, "Serial order: A parallel distributed processing approach," *Advances in psychology*, vol. 121, pp. 471–495, 1997.

[33] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, no. 1–3, pp. 239 – 255, 2010, artificial Brains.

[34] L. Chua and L. Yang, "Cellular neural networks: theory," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 10, pp. 1257–1272, Oct 1988.

[35] ——, "Cellular neural networks: applications," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 10, pp. 1273–1290, Oct 1988.

[36] E. Painkras, L. Plana, J. Garside, S. Temple, F. Galluppi, C. Patterson, D. Lester, A. Brown, and S. Furber, "SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 8, pp. 1943–1953, Aug 2013.

[37] W. J. Freeman, "Mesoscopic neurodynamics: From neuron to brain," *Journal of Physiology-Paris*, vol. 94, 2000.

[38] W. Freeman, "Nonlinear gain mediating cortical stimulus-response relations," *Biological Cybernetics*, vol. 33, no. 4, pp. 237–247, 1979.

[39] ——, "Simulation of chaotic eeg patterns with a dynamic model of the olfactory system," *Biological Cybernetics*, vol. 56, no. 2-3, pp. 139–150, 1987.

[40] Y. Yao and W. J. Freeman, "Model of biological pattern recognition with spatially chaotic dynamics," *Neural Networks*, vol. 3, no. 2, pp. 153 – 170, 1990.

[41] G. Li, J. Zhang, and W. J. Freeman, "Mandarin digital speech recognition based on a chaotic neural network and fuzzy c-means clustering," in *IEEE International Fuzzy Systems Conference, 2007.* IEEE, 2007, pp. 1–5.

[42] Y. Yao, W. J. Freeman, B. Burke, and Q. Yang, "Pattern recognition by a distributed neural network: An industrial application," *Neural Networks*, vol. 4, no. 1, pp. 103–121, 1991.

[43] R. Kozma and W. J. Freeman, "Chaotic resonance — methods and applications for robust classification of noisy and variable patterns," *International Journal of Bifurcation and Chaos*, vol. 11, no. 06, pp. 1607–1629, 2001.

[44] M. D. Skowronski, "Biologically inspired noise-robust speech recognition for both man and machine," Ph.D. dissertation, University of Florida, 2004.

[45] X. Li, G. Li, L. Wang, and W. j. Freeman, “A study on a bionic pattern classifier based on olfactory neural system,” *International Journal of Bifurcation and Chaos*, vol. 16, no. 08, pp. 2425–2434, 2006.

[46] M. Obayashi, S. Koga, L.-B. Feng, T. Kuremoto, and K. Kobayashi, “Handwriting character classification using Freeman’s olfactory KIII model,” *Artificial Life and Robotics*, vol. 17, no. 2, pp. 227–232, 2012.

[47] G. Li, J. Zhang, Y. Wang, and W. J. Freeman, “Face recognition using a neural network simulating olfactory systems,” in *Lecture Notes in Computer Science*, 2006, pp. 93–97.

[48] X. Yang, J. Fu, Z. Lou, L. Wang, G. Li, and W. J. Freeman, “Tea classification based on artificial olfaction using bionic olfactory neural network,” in *Proceedings of the Third International Conference on Advnaces in Neural Networks - Volume Part II*, ser. ISNN’06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 343–348.

[49] J. Fu, G. Li, Y. Qin, and W. J. Freeman, “A pattern recognition method for electronic noses based on an olfactory neural network,” *Sensors and Actuators B: Chemical*, vol. 125, no. 2, pp. 489–497, 2007.

[50] H. Li and R. Kozma, “A dynamic neural network method for time series prediction using the KIII model,” in *Proceedings of the 2003 International Joint Conference on Neural Networks*, 2003, pp. 347–352.

[51] I. Beliaev and R. Kozma, “Time series prediction using chaotic neural networks on the CATS benchmark,” *Neurocomputing*, vol. 70, no. 13–15, pp. 2426 – 2439, 2007, selected papers from the 3rd International Conference on Development and Learning (ICDL 2004)Time series prediction competition: the CATS benchmark3rd International Conference on Development and Learning.

[52] R. Kozma, W. J. Freeman, and P. Erdi, “The KIV model—nonlinear spatio-temporal dynamics of the primordial vertebrate forebrain,” *Neurocomputing*, vol. 52–54, no. 0, pp. 819 – 826, 2003, computational Neuroscience: Trends in Research 2003.

[53] R. Kozma, “Intentional systems: Review of neurodynamics, modeling, and robotics implementations,” *Physics of Life Reviews*, vol. 05, no. 01, pp. 1–21, 2007.

[54] R. Kozma and W. J. Freeman, “The KIV model of intentional dynamics and decision making,” *Neural Networks*, vol. 22, no. 3, pp. 277 – 285, 2009, goal-Directed Neural Systems.

[55] R. Kozma, T. Huntsberger, H. Aghazarian, E. Tunstel, R. Ilin, and W. J. Freeman, “Intentional control for planetary rover SRR,” *Advanced Robotics*, vol. 22, no. 12, pp. 1309–1327, 2008.

[56] R. Kozma, M. Puljic, P. Balister, B. Bollobás, and W. J. Freeman, “Phase transitions in the neuropercolation model of neural populations with mixed local and non-local interactions,” *Biological Cybernetics*, vol. 92, no. 6, pp. 367–379, 2005.

[57] J. Zhang, G. Li, and W. Freeman, “Analysis of bionic olfactory neural networks based on small-world networks view,” in *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, June 2008, pp. 1537–1539.

[58] R. Kozma and W. Freeman, “Modeling cortical singularities during the cognitive cycle using random graph theory,” in *Advances in Cognitive Neurodynamics (IV)*, ser. Advances in Cognitive Neurodynamics, H. Liljenström, Ed. Springer Netherlands, 2015, pp. 137–142.

[59] D. Xu and J. Principe, “Dynamical analysis of neural oscillators in an olfactory cortex model,” *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1053–1062, Sept 2004.

[60] D. Xu, J. Principe, and J. Harris, “Logic computation using coupled neural oscillators,” in *Proceedings of the 2004 International Symposium on Circuits and Systems, 2004. ISCAS ’04*, vol. 5, May 2004, pp. V–788–V–791 Vol.5.

[61] R. Vassar, S. K. Chao, R. Sitcheran, J. M. Nun~ez, L. B. Vosshall, and R. Axel, “Topographic organization of sensory projections to the olfactory bulb,” *Cell*, vol. 79, no. 6, pp. 981 – 991, 1994.

[62] M. Ozturk, D. Xu, and J. Principe, “Modified Freeman model: a stability analysis and application to pattern recognition,” in *IEEE International Joint Conference on Neural Networks, 2004. Proceedings*, vol. 4, July 2004, pp. 3207–3212 vol.4.

[63] V. G. Tavares, J. Principe, and J. Harris, “F&H filter: A novel ultra-low power discrete time filter,” *Electronics Letters*, vol. 35, no. 15, pp. 1226–1227, 1999.

[64] V. G. Tavares, C. Duarte, P. Guedes de Oliveira, and J. C. Príncipe, “Filter & hold: a mixed continuous-/discrete-time technique for time-constant scaling,” *International Journal of Circuit Theory and Applications*, 2014.

[65] F. Warkowski, J. Leenstra, J. Nijhuis, and L. Spaanenburg, “Issues in the test of artificial neural networks,” in *IEEE International Conference on Computer Design: VLSI in Computers and Processors, 1989. ICCD ’89. Proceedings*, Oct 1989, pp. 487–490.

[66] C. Toumazou and J. Hughes, “Regulated cascode switched-current memory cell,” *Electronics Letters*, vol. 26, no. 5, pp. 303–305, March 1990.

[67] E. Säckinger and W. Guggenühl, “A high-swing, high-impedance mos cascode circuit,” *IEEE Journal of Solid-State Circuits*, vol. 25, no. 1, pp. 289–298, Feb 1990.

[68] S. Tabarce, V. Tavares, and P. de Oliveira, “Programmable analogue vlsi implementation for asymmetric sigmoid neural activation function and its derivative,” *Electronics Letters*, vol. 41, no. 15, pp. 863–864, July 2005.

[69] J. Franca, “Analogue-digital window comparator with highly flexible programmability,” *Electronics Letters*, vol. 27, no. 22, pp. 2063–2064, Oct 1991.

[70] V. Kolarik, M. Lubaszewski, and B. Courtois, “Designing self-exercising analogue checkers,” in *VLSI Test Symposium, 1994. Proceedings., 12th IEEE*, Apr 1994, pp. 252–257.

[71] Y. Zhang and M. Wong, “Self-testable full range window comparator,” in *TENCON 2004. 2004 IEEE Region 10 Conference*, vol. D, Nov 2004, pp. 262–265 Vol. 4.

[72] R. Xiao, A. Laknaur, and H. Wang, “A fully programmable analog window comparator,” in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, May 2007, pp. 3872–3875.

[73] P. E. Allen and D. R. Holberg, *CMOS analog circuit design.* Taylor & Francis US, 2002.

[74] M. Bazes, “Two novel fully complementary self-biased CMOS differential amplifiers,” *IEEE Journal of Solid-State Circuits*, vol. 26, no. 2, pp. 165–168, Feb 1991.

[75] H.-C. Chow and I.-H. Wang, “High performance automatic gain control circuit using a S/H peak-detector for ASK receiver,” in *9th International Conference on Electronics, Circuits and Systems*, vol. 2, 2002, pp. 429–432 vol.2.