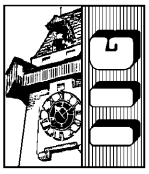
Doctoral Thesis

The Wire-Length Complexity of Neural Networks

Robert Albin Legenstein



Institute for Theoretical Computer Science
Technische Universität Graz,
Austria

TUG

Graz, Novemver 2001

Acknowledgments

I would like to thank the following people:

- Wolfgang Maass, my supervisor, for his inspiring ideas and all I learned from him in these years. I worked together with Wolfgang Maass on the topics described in Chapters 5–10.
- Helmut Pockberger, for his friendly willingness to be the second referee of this thesis.
- Gerhard Wöginger, for his trust in my work and his noble support.
- Thomas Natschläger, for general help and discussion.
- The whole "IGI-family", who made scientific life much more fun.
- My family and friends, for being my family and my friends, especially the one and only Katharina.

In addition I acknowledge support by the "Fonds zur Förderung der wissenschaftlichen Forschung (FWF), Austrian Science Fund", projects number X and X.

Abstract

The ability of our nervous system to rapidly process and react on the huge amount of sensory input data is grounded on its massively parallel architecture. Arguably, physical cost for communication, that is to say the space needed for wires, is the most severe bottleneck in biological as well as in artificial architectures of this type. In this thesis the complexity of wiring in biological and artificial neural networks, the implications of wiring constraints to models for brain circuits, and the implementation of wire-efficient circuit designs in hardware are studied.

We present a simple mathematical framework that allows us to study the wiring complexity of neural circuits in a formal and general manner. In this model, the complexity of a circuit is measured by the total length of wires needed to implement the circuit, a complexity measure that is one of the most salient ones if real-world constraints of implementations in hardware or "wetware" are considered. Several functions of biological relevance are studied in this context and efficient circuits are designed for them. It turns out that circuits and layouts we designed in the spirit of wire length minimization are also efficient in VLSI models and are therefore also of interest for hardware design.

Furthermore, we study the layout of general computational structures like tree computations. We give tight upper and lower bounds on the wire length of constrained tree layouts and show efficient layout strategies for prefix computations. These results are of interest in VLSI design as well as in biological circuits. Another chapter is concerned with the computational complexity of optimal layouts in VLSI circuits. It is shown that a layout problem that is fundamental to VLSI design is computationally intractable or NP-complete.

Zusammenfassung

Die enorme Leistungsfähigkeit unseres Nervensystems in der Verarbeitung von sensorischem Input ist in dessen massiv-parallelen Architektur fundiert. Solche Architekturen, biologische und künstliche, sind hauptsächlich beschränkt durch Kosten für Kommunikation, das heißt durch den für Kommunikationsverbindungen (*wires*) benötigten Raum. Diese Arbeit untersucht die Komplexität von biologischen und künstlichen neuronalen Netzwerken in Hinsicht auf ihre Verdrahtungskosten (*wire-Komplexität*), die Auswirkungen von Beschränkungen der wire-Komplexität auf Modelle biologischer neuronaler Schaltkreise und die Anwendung von in dieser Hinsicht effizienten Schaltkreisen in Hardware.

Wir führen ein einfaches theoretisches Rahmenmodell ein, das es uns erlaubt die wire-Komplexität von neuronalen Netzen in einer formalen und allgemeinen Form zu untersuchen. Die Komplexität eines Schaltkreises wird hierbei durch die in jedem Layout benötigte Gesammtverbindungslänge bestimmt. In diesem Kontext werden verschiedene biologisch relevante Funktionen untersucht und effiziente Schaltkreise zur Berechnung dieser Funktionen vorgeschlagen. Es stellt sich heraus daß solchermaßen entworfene Schaltkreise auch in VLSI-Modellen effizient sind und deshalb auch von Interesse für Implementierungen in Hardware sind.

Weiters werden Layouts für allgemeine Berechnungsstrukturen, etwa Baumstrukturen, untersucht. Wir bestimmen optimale Schranken für die wire-Komplexität von Baumlayouts mit Nebenbedingungen und zeigen effiziente layoutstrategien für Präfixberechnungen. Ein weiteres Kapitel beschäftigt sich mit der Rechenkomplexität von optimalen layouts in VLSI-Schaltkreisen. Es wird gezeigt daß ein wichtiges Layoutproblem NP-vollständig, also nicht attackierbar ist.

Contents

Chapter 1

Introduction and Overview

The ease with which humans are able to process and react on the permanent stream of data from millions of sensory input channels is amazing. For example, in the visual system of primates, the number of neurons that transmit information from the retina (via the thalamus) to the cortex is estimated to be in the order of 10^6. Difficult visual tasks like object recognition invariant under position, scale, rotation, deformation, etc., are performed permanently and with high precision and speed.

Understanding the computations that make such powerful behavior possible is one of the key challenges in science. This question is also of interest to hardware engineering, because there is a growing demand for time- and energy efficient hardware for sensory processing. In recent years, biological inspired hardware architectures have received growing attention in the field of neuromorphic engineering, see (Mead, 1989). Because the number of parallel inputs which such circuits have to handle is quite large, complexity issues are critical, both for circuits in hardware and "wetware".

This thesis is concerned with the investigation of complexity issues that reflect real world implementation constraints of biological and artificial neural circuits. In order to analyze these issues, a mathematical framework is established that allows us to address such questions in a systematic and general way. We base our framework on circuit complexity theory since it is established as a fundamental tool for analysing the complexity of computation in massively parallel architectures. The complexity of a circuit in circuit complexity theory is customarily measured as the number of computational elements (gates) in the circuit. However, most volume of neural tissue is dedicated not to computational elements (in this context cell bodies of neurons), but to connections between them (dendrites and axons). In billions of years of the development of efficient neural systems, evolution primarily optimized communication structures in order to build more and more complex and therefore more competitive systems. The hypothesis that brain circuits are designed such that wiring is economized is often referred to as

the "wiring economy principle" and its importance is pointed out by many authors (see Chapter 4). Therefore we propose a new complexity measure, the *total wire length*, which measures not only the number of connections but the total length of wires needed to implement some circuit. In the context of resources used by a brain circuit, this complexity measure is one of the most salient ones.

In addressing the issue of computation in the brain, not only the computational model and the complexity measure considered are important, but also the question of "what is computed". In the abstract framework of circuit complexity theory, this question is not of much interest. An infinite number of mathematical functions can be constructed and a lot of interesting mathematical functions are known. Therefore, most results in circuit complexity theory address functions like parity, sorting, or addition. However, in the context of biological neural computation, no simple answer can be given. What is the input of a cortical circuit, and what is its output? In this thesis, we propose "benchmark functions" that seem to be related to cortical function. We argue that functions we consider have to be somehow solved implicitly by cortical circuitry, although they might not be addressed explicitly.

Artificial computational devices are normally implemented in silicon in very large scale integration (VLSI). Because of the two-dimensional structure of VLSI circuits, wiring issues are even more important than in – basically three-dimensional – brain circuits. In the words of Carver Mead, this can be summarized as:"Economizing on wire is the single most important priority for both nerves and chips." (Mead, 1989). Therefore VLSI design strategies might be of help in understanding the structure of neural circuits and – the other way around – design properties of biological neural circuits can be of great help in finding efficient VLSI designs. In this thesis, we take a journey on the common ground of three different but related fields, neuroscience, circuit complexity theory, and VLSI theory. We believe that any of these fields is of great interest to any of the others, that new relevant relations can be found, and that new ideas and spirit can flow between the fields, gaining more insight into any of the disciplines. This thesis can be seen as a starting point in this direction. Questions relevant to any the fields and in the free space in between them are addressed.

We start the journey in Chapter 2 which presents the basics of brain function. Most issues are discussed on a rudimentary level. We go into more detail in topics related to the wiring of the brain. In particular, qualitative and quantitative aspects of neural arborizations and wiring aspects of the cerebral cortex are discussed more deeply.

Mathematical models of computational circuits are addressed in Chapter 3. Fundamental mathematical definitions and paradigms like graphs and layouts are given. Parallel computation is discussed with emphasis on biologically inspired circuit models and the basics of circuit complexity theory. Furthermore, the basics of VLSI, abstract VLSI models and ideas of analog VLSI are presented.

Issues concerning the wiring of the nervous system have been addressed by several authors. Although a variety of different viewpoints can be found in the literature, the field is still easy to overview. In Chapter 4 we summarizes the most important research – to the view of the author – on this topic. This chapter can be thought of being a motivation for the research done in this thesis, but it is also intended to give the reader a feeling of the state of the art of research in the field, to emphasize the remarkable different approach of this thesis, and to facilitate the comparison of different approaches.

In Chapter 5, the new complexity measure for parallel processing *total wire length* is introduced. We argue that this measure is among the most salient ones for sensory processing but is still simple enough to make mathematical analysis possible. Reasonable bounds on the total wire length of cortical circuits are discussed in Section 5.2. It turns out that cortical circuits are very restricted in their total wire length. The total wire length of a circuit is also of relevance in an implementation in hardware. In Section 5.3 we briefly discuss the relation of total wire length to classical VLSI performance measures and present the abstract model on which most VLSI analysis of subsequent chapters is based.

In Chapter 6 we discuss a simple one-dimensional sensory processing task. The discussed pattern recognition problem is fundamental to problems arising in biological tasks and in problems relevant for real-time VLSI applications. We give a circuit design with threshold gates that is efficient in the number of gates and its total wire length. We show that the number of gates used by this design is asymptotically the best possible for any threshold circuit. However, if in addition winner-take-all gates are employed, a much more efficient circuit can be developed. The circuit design we present combines a constant number of gates with linear total wire length, which is the best possible. This result shows that winner-take-all circuits can be very useful in circuits for sensory processing tasks.

While in Chapter 6 we merely dealt with one-dimensional pattern recognition problems, two-dimensional problems are of special interest because of two-dimensional input representations (maps) that can be found throughout the cerebral neocortex and because of the natural representation of many problem instances (*e. g.* visual input) in an two-dimensional array. In Chapter 7 we exhibit circuit design strategies for such problems that stay within realistic complexity bounds, having linear or almost linear total wire length. Furthermore, it turns out that such designs are also applicable for VLSI-implementations.

In the design of circuits, basic computational structures emerge that are useful in many concrete computations. One of the most basic VLSI layout strategies is the well known *H-tree*. We show in Chapter 8 that an extended version of the H-tree layout can be implemented efficiently, using only a constant multiple of the area and total wire length of the standard approach. Another computational structure that is often used is an *prefix computation*. An efficient implementation of a prefix computation with gates of limited fan-in is designed.

In classical circuit complexity theory, the circuit complexity of a function is normally expressed by its circuit size, *i. e.* the minimal number of gates of any circuit that computes the function. It is an interesting question how the total wire length of a function relates to this classical circuit complexity measure. In Chapter 9 we derive upper and lower bounds on the total wire length of a function in terms of its circuit size. In abstract VLSI models, the complexity of a function is normally expressed in terms of the area of an implementation of the function. The relation between total wire length and VLSI area is also discussed in Chapter 9.

Tree-like organizations are a fundamental structure in parallel processing, as we have already noted. In Chapter 10 we study the layout of trees where leaves are restricted to lie on a horizontal line. In contrast to previous results, we employ a model where trees of arbitrary fan-out m can be analyzed. An algorithm is presented that produces a layout with a total wire length that is significantly smaller than that of obvious layouts. Furthermore, we show that this algorithm produces layouts with optimal wire length.

The problem of routing in biological circuits is closely related to the problem of routing in VLSI models. While there is more freedom in cortical routing due to the three-dimensional structure of cortical circuits, the problem of finding optimal circuit layouts is even harder than in VLSI routing. Unfortunately, most VLSI routing problems are computationally very demanding. In Chapter 11 we analyze the computational complexity of a specific routing problem that is fundamental to VLSI design, the knock knee channel routing problem. The exact complexity of this problem was unknown so far. In this chapter, we show that one of the easiest versions of the problem is NP-complete (*i. e.* computationally intractable). This gives a tight characterization of channel routing in terms of its computational complexity.

Chapter 12 concludes the thesis with a discussion of the results and an outlook to future work.

Chapter 2

Fundamental Aspects of the Nervous System

The ability of the nervous system to rapidly process – and react on – the huge amount of sensory input-data is grounded on its massive parallel architecture. In this chapter we summarize these grounds of biological neural computation in describing the elementary concepts of the nervous system. However, a detailed review of these concepts is far off from scope of this thesis. Hence we concentrate on the topics most relevant for the following chapters and the interested reader is referred to (Purves et al., 1997; Bear et al., 1996; Kandel et al., 1991; Shepherd, 1994; Stuart et al., 1999; Abeles, 1998). The basic computational unit in the nervous system – the neuron – is introduced in Section 2.1. In Section 2.2, we discuss how information is communicated and processed in the brain. The way that communication is established in a physical sense is the topic of Section 2.3. Finally, several aspects of the cortex related to communication structure are discussed in Section 2.4.

2.1 Basic Computational Units in the Nervous System: The Neuron

The discovery that the brain is made out of discrete entities that communicate with one another by specialized contacts is primarily a result of the nineteenth-century pioneering studies of the Spanish neuroanatomist Santiago Ramon y Cajal. These elementary information processing units are called *nerve cells* or *neurons*[1] A schematic drawing of a typical cortical pyramidal cell is shown in Figure 2.1. These nerve cells are in most respects similar to other cells in the

[1]It should be noted that the nervous system not only consists of nerve cells but also of *supporting cells*. There are even several times more supporting cells in the nervous system than nerve cells but supporting cells are not primarily involved in electrical signaling.

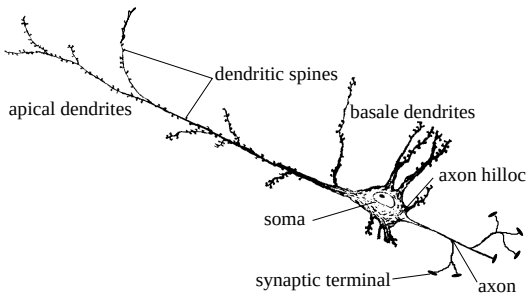


Figure 2.1: Schematic drawing of a cortical layer V pyramidal cell.

body. Each nerve cell has a *cell body* (or *soma*) that contains organelles essential to the function of all cells. What makes nerve cells different to all other cells is that nerve cells are highly specialized for intercellular communication. This specialization becomes most apparent in the fascinating geometries of neurons. The soma receives electrical signals from other cells via its *dendrites* which arise from the soma and are often highly branched (they are also called dendritic branches or dendritic processes). The dendrites (together with the soma) provide sites for synaptic contacts with other nerve cells. The information that is integrated at the soma is conducted to other cells via the axon which arises from the axon hillock and usually branches into several thousand arbors. Thus, the axon can be regarded as the portion of the neuron specialized for signal conduction. At the tip of such an arbor there may be a *synaptic terminal* which connects to the dendrite (or sometimes to the soma) of another neuron. These connections are called *synapses*.

2.2 Information Processing in Neurons: Spikes and Synapses

So far we merely dealt with morphological features of neurons. We mentioned that neurons are specialized for intercellular communication. But how is information communicated and processed by neurons? In the nervous system, the representation of information is based on electrical signals. Like in all other cells, there is an electrical potential difference that exists across the cell membrane, the so called *membrane potential*. The membrane potential arises from different ion-concentrations in the liquid inside and outside the cell. If there are no inputs impinging onto the neuron the membrane is at its *resting potential* which varies between -30 mV and -90 mV depending on the cell type.

If the input to a neuron rises the membrane potential at the axon hillock above a certain threshold an action potential or *spike* is initiated. A spike is a

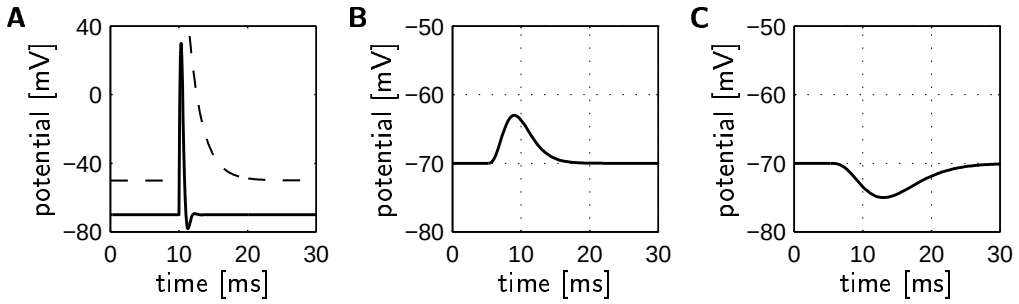


Figure 2.2: **A** Typical shape of an action potential. Typical shape of an EPSP (**B**) and an IPSP (**C**).

rapid positive change in the membrane potential (see Figure 2.2A). The action potential is conducted by the axon and passed on to the next cells in the pathway by means of synaptic transmission. The synaptic transmission produces a change in the membrane potential of the targeted (postsynaptic) cell, which is called a *postsynaptic potential* (PSP). The result of a postsynaptic potential is either an increase of the membrane potential (see Figure 2.2B) or a decrease of the membrane potential (see Figure 2.2C). In the former case it is called an *excitatory postsynaptic potential* (EPSP) and in the latter case an *inhibitory postsynaptic potential* (IPSP).

Historically, "synapses" is the plural for synopsis which means "connection". This term was firstly used by Foster and Sherrington (Foster, 1897). Synapses are believed to play a key role in this information processing. Although spikes are digital events (*i. e.* there is no information in the strength of spikes), the postsynaptic change in membrane voltage highly depends on the properties of the synaptic transmission. Thus, the function that is computed by a single neuron is believed to be highly determined by properties of its synaptic inputs[2]. Synaptic transmission is a highly complex, dynamic and stochastic process, see *e. g.* (Abbott et al., 1997; Dobrunz and Stevens, 1997; Maass and Zador, 1999)). However, in theory the properties of a synapse are often described by a single scalar, the *weight* $w_{i,j}$ (the indexes are chosen with respect to the neurons that are connected by this synapse). This weight can be regarded as a measure for the influence of the presynaptic neuron onto the postsynaptic neuron. Furthermore, the weight of a synapse may change over time, thus synapses are also regarded to be the basis of learning and adaptation in the nervous system. This means that the function of a neuron or a network of neurons can be adapted to its environment or learned – instead of being "programmed" into the network – by means of this synaptic

[2]It is also believed that many other factors – *e. g.* the summation of potentials via the dendritic tree – influence the function of a neuron. However, most theoretical models of neural computation rely merely on the power of synaptic transmission.

plasticity. This ability is one of the key features of the nervous system.

So far, this is all we need to know about the way that information is processed in nerve cells. It is obvious that this Section can merely be a very brief introduction into this topic and simplifications are unavoidable. The reader interested in this topic and in the mechanisms that underlie neuronal behavior are referred to the literature given at the beginning of this Chapter.

2.3 Communication Structure in the Nervous System: Dendrites and Axons

Since this thesis is concerned with the wiring of neural systems, it is helpful to gain insight in the way that neurons are connected in biological neural networks. As mentioned above, dendrites are the portion of the nerve cell that are specialized for receiving information. Most neurons have multiple dendrites, which are typically short and highly branched. However, there is high diversity in the geometries of neurons with respect to their dendrites. This diversity ranges from a small minority of neurons that lack dendrites at all up to neurons with dendritic arborizations of fascinating complexity, see Figure 2.3. Typically, the number of inputs received by a neuron in the human nervous system is in the order of 10^5, but within different cells the number of inputs ranges from 1 to about 10^6 with highly diverse branching patters.

The number of inputs a neuron receives is reflected in the complexity of its dendritic aborizations. The branched structure of dendrites results in an enormous enlarged cell surface which makes it possible that up to more than 10^6 synapses can contact a single neuron. For example, 97 % of the surface area of a motor neuron (excluding the axon) is dendritic (Ulfhake and Kellerth, 1981). A simple calculation shows that such area would result in a massively enlarged brain volume if this surface area would be supplied by the soma. The surface area of such dendrites is $370\,000\,\mu m^2$ while occupying a volume of $300\,000\,\mu m^3$. A sphere of the same surface area would occupy a volume of $20\,000\,000\,\mu m^3$. This increased surface area is indeed valuable for increased input sizes of neurons, since 80 % of the surface of proximal dendrites of motor neurons is covered with synapses (Kellerth et al., 1979).

Compared with axons, dendrites make relatively local connections. While axons may reach lengths of up to a meter, dendrites are rarely longer than 1–2 mm. This is illustrated by typical dimensions of two well known neuron types. The CA1 pyramidal cell in the rat has 5 basal dendrites with a *dendrite extent*[3] of the basal dendrites is 130 μm, and the total dendritic length of such cells is

[3]The dendrite extent is the average distance from the cell body to the tips of the longest dendrites

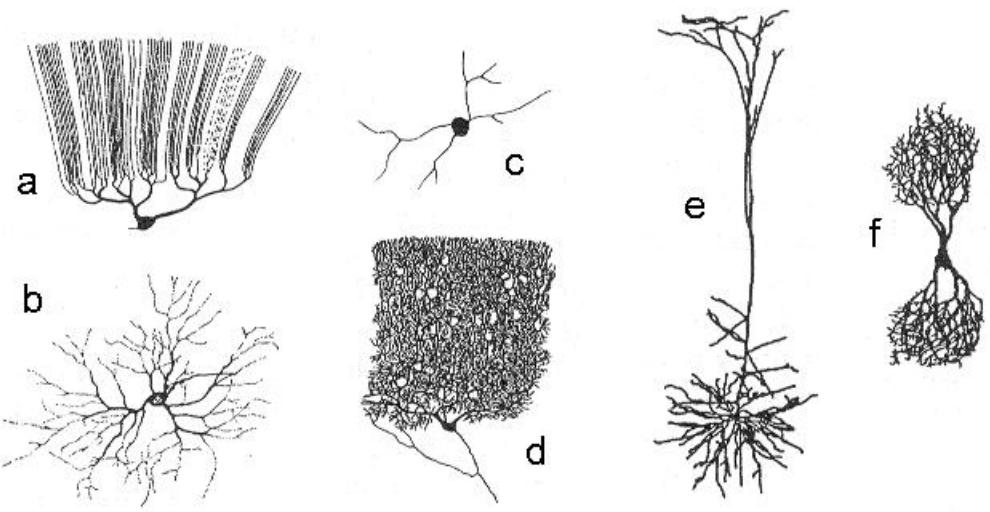


Figure 2.3: Diversity of dendritic geometries illustrated on six cell types. (a) Fish Purkinje Neuron. (b) Cat retinal ganglion neuron. (c) Salamander retinal amacrine neuron. (d) Human cerebellar Purkinje neuron. (e) Rat neocortical layer 5 pyramidal neuron. (f) Human nucleus of Burdach neuron. Modified from (Mel, 1994).

about 11 900 μm ,including stratum radiatum and lacunosum-moleculare (Bannister and Larkman, 1995). The Cerebellar Purkinje cell in the guinea pig has one dendrite with a dendrite extent of 200 μm and the total dendritic length of such cells is 9 100 μm(Rapp et al., 1994).

Dendrites of different neuron types have different characteristic branching patterns. These branching patters differ in several interesting ways. The way that dendrites extent into specific spatial domains is a primary contribution to the mode of connectivity of a neuron. In this way, particular portions of the dendritic arbor receives inputs from specific sources. A further differentiation is the extent to which the dendrite fills the spatial domain of its arborizations. The diversity ranges from dendrites with a small amount of specific targets (*selective aborizations*) to branches that occupy most of their domain (*space-filling aborizations*). In between these extremes are the medium space filling *sampling aborizations*. It can be argued that these different modes of dendritic branching is a indicator for very different neuronal function. On one extreme we have the selection of a single specific input an on the other one the integration of information from a hundred thousand neurons in a broad input area. If one is willing to speculate, one could compare it with two mathematical paradigms of algebraic computation with clear input-output relation and statistical analysis over distributions of inputs.

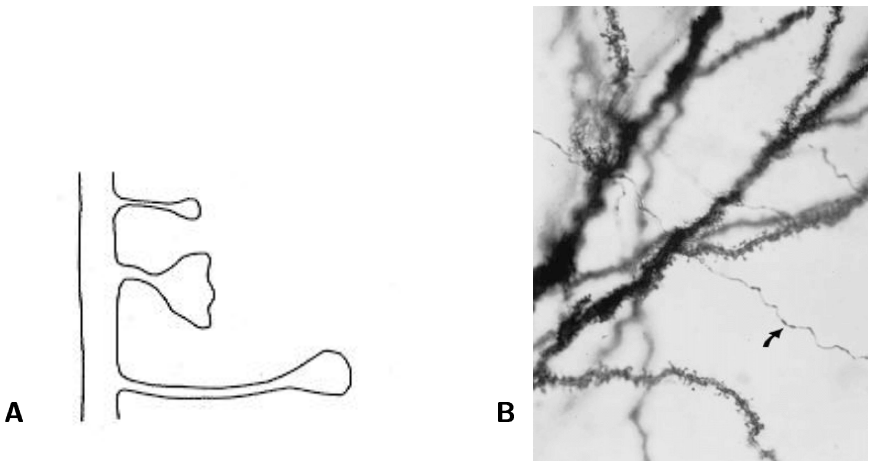


Figure 2.4: **A** Schematic drawing of pedunculated simple spines. A *thin spine* (top), *mushroom spine* (middle), and *gemmule* (bottom) is shown. **B** Dendrite of a pyramidal cell from the rat hippocampus area CA1 at high magnification. The cell was silver-impregnated using the Golgi method. Several spines can be seen on the dendrite. The non-spiny process (arrow) is an axon passing by. Modified from (Fiala and Harris, 1999).

Another vivid measure of the extent to which an arborization fills its spatial domain and also the complexity of an arborization is its *fractal dimension*. In addition to the well known dimensions of linear (dimension 1), planar (dimension 2) and solid objects (dimension 3), one can define fractal dimensions which are not natural numbers but fractions. Objects with fractal dimension fill a portion of the space in which they are embedded. On a limited scale, dendritic arbors can be considered to be fractal objects (Smith et al., 1989). The fractal dimension of selective arborizations is close to one, the fractal dimension of space-filling arborizations tends to the dimension of the region they occupy. Differences in fractal dimension relate to differences in connectivity. We give a simple example (Fiala and Harris, 1999). Retinal ganglion cells have sampling and arbors which are essentially planar and a fractal dimension of approximately 1.5 (Fernandez et al., 1994; Wingate et al., 1992). The arbor covers 25 000 μm^2 and receives 2 000 synapses (Sterling, 1990). The Purkinje cell has a space-filling planar arbor with a fractal dimension of about 1.8 in mammals. The arbor covers 50 000 μm^2 and receives 160 000 synapses (Harvey and Napper, 1991).

Synapses on dendrites may be made directly on the shaft (*shaft synapses*), but are often placed on special enlargements or protrusions. Such synaptic specializations are very diverse. The most common specializations are *simple spines*. Spines are protrusions from the dendrite that are often ending in a bulbous head attached to the dendrite by a narrow neck, see Figure Figure 2.4.

The essence of neural function is the generation of action potentials. Since den-

drites are normally the origin of most of the synaptic inputs to the neuron, their properties play an important role in the generation of action potentials. Action potentials are initiated at the axon hillock. The ability of an EPSP to depolarize the membrane potential toward the threshold of the neuron depends on the size of the EPSP and the extend to which it is attenuated as it propagates from the dendrite to the soma. Such attenuation can be dramatic for synaptic input on distal dendrites. Furthermore, the summation of EPSPs on the dendrite is sublinear and depends on the electrical properties and the sites of the synaptic inputs. For a detailed discussion on the topic of electrical properties of dendritic trees and dendritic information processing, see (Segev and London, 1999; Spruston et al., 1999) and the excellent review (Mel, 1994).

In summary, dendrites are morphologically diverse and commonly complex. They are the main determinant of neuronal connectivity and contribute significantly to neuronal function in cells and networks of cells.

Because axons may be much longer than dendrites, they are often thought to be the main contributor to wire length and communication in the nervous system. In the grey matter of the cerebral cortex (see below), axons and dendrites take up about the same amount of volume. However, only axons are sent outside the grey matter to communicate with distant cortical neurons. Neurons normally have a single axon, from which *axon collaterals* may branch orthogonally to the main stem. In humans, the majority of axons have a length of a few millimeters. However, since many axons are very long compared to dendrites, they are very important in signal conduction. The length of up to a meter requires a special mechanism for action potential conduction in the axon since otherwise the signal would not reach the tips of the axon. This problem is solved in the axon by special membrane properties such that the spike is not conducted but regenerated along the membrane. Thus, the axon membrane is a highly specialized structure for communicating action potentials over long distances. Moreover, axons are often *myelated* which means that certain glia cells wrap around the axon to isolate it and thus increase the speed of signal conduction. All this shows that nature uses great efforts to optimize communication pathways in the nervous system and often provides amazing solutions to the arising problems. More quantitative aspects of axons are described within the following section.

2.4 Aspects of the Cerebral Cortex

One of the most remarkable and prominent parts of the nervous system is the cerebral cortex, particularly its most recently developed part, the neocortex. Unlike other cortical structures, the neocortex is found only in mammals. Over the course of human evolution, the neocortex has expanded enormously. Usually, if the term cortex is left unqualified, it is usually intended to refer to the cerebral

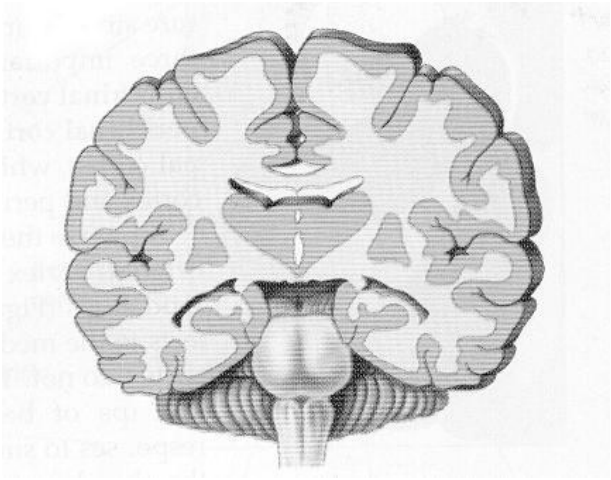


Figure 2.5: A coronal cut through the human brain. The cerebral cortex and its folded structure can be seen. The grey matter is right under the surface of the cerebrum (grey shaded region) and covers the white matter (white region). Modified from (Bear et al., 1996).

neocortex. The cerebral cortex is responsible for sensations, perceptions, voluntary movement, learning, speech, and cognition. Because of the central role of the neocortex in sensory processing, we will refer to cortical processing in several chapters. In this Section, we describe basic aspects of the cortex, with emphasis on wiring aspects. The will discuss the division of local and global communication in white and grey matter, cortical areas, and maps.

2.4.1 White and Grey Matter

The cortex consist of a sheet of *grey matter* at the surface area, about two millimeters thick. The grey matter appears grey because of the many cell bodies of neurons. Hence, the gray matter is a sheet consisting mostly of neurons, mainly pyramidal cells, less stellate cells and mariinotti cells. The pyramidal neurons of grey matter send and receive myelated axons in the volume below the it. These axons appear white to the human eye, hence this structure is called *white matter* (see Figure Figure 2.5).

Cortical neurons have dendrites and local axons that spread a fraction of a millimeter horizontally (*i. e.* parallel to the surface of the cortical sheet). Such connections are used only for local communication. All of the pyramidal cells and some of the spiny stellate cells send axons into the white matter (about 80% of cortical neurons). About 98.6% of these axons connect two cortical regions within the same cerebral hemisphere. There are also connections between the hemispheres (mostly via the corpus callosum), connections to and from deep

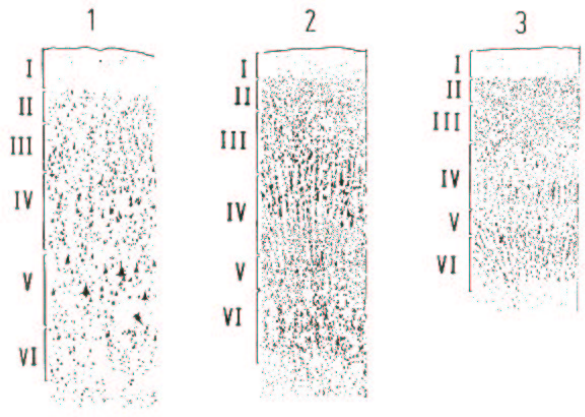


Figure 2.6: Three slices taken at different areas radial to the cortical surface. The slices are Nissl stained such that neuronal cell bodies can be seen. Several layers can be distinguished. It can be seen that the slices have diverse architectures. Modified from (Abeles, 1998).

nuclei and diffuse connections to and from areas the brainstem. Hence, white matter is dedicated to long distance information transfer between cortical regions. The folded structure of the cortex of humans is not merely a mechanical necessity to the fast growth of neocortex. The fibers of white matter would have to be much longer if the corex was not folded.

2.4.2 Cortical Areas

A common feature all over the cortex is the arrangement of neurons in horizontal layers, which have different types, sizes, and densities of cells, see Figure 2.6. The cortical slices shown in Figure 2.6 are from three different spatial locations of the cortex. One can see that the patterns of neural cell bodies vary remarkably. Such a pattern is called a cytoarchitecture. At the turn of the twentieth century, the cytoarchitecture was taken as a basis for parceling the cortex into different cortical areas by the German neuroanatomist Korbinian Brodmann. It was suspected that different areas also perform different cortical functions. At present, there is no doubt that this hypothesis holds. For example, area 17, called primary visual area or striate cortex is responsible for basic visual processing.

2.4.3 Cortical Maps

Neurons in primary visual cortex (V1) – as in other sensory areas – show receptive field properties. In V1, this means that the activation of such a neuron is highly correlated with stimulation of receptors in a particular local area of the retina.

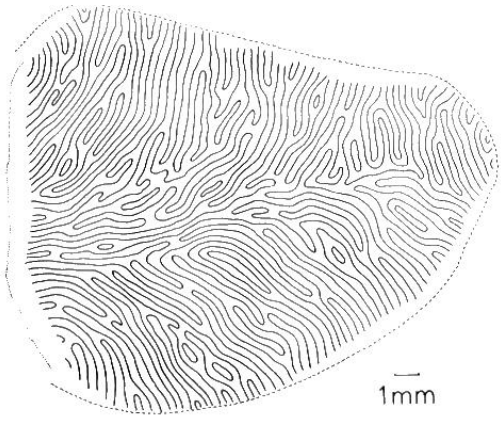


Figure 2.7: Ocular dominance patterns as described by Hubel and Wiesel. The figure shows area 17 on the left cerebral hemisphere. The black lines represent borders where ocular dominance shifts from one eye to the other one. Modified from (Abeles, 1998).

Therefore, for any local activation of retinal receptor cells, neurons in V1 exist that code this local stimulus. Therefore, one can ask how the layout of neurons in V1 corresponds to their receptive fields. Is there a map that places a neuron onto a specific position of the cortical sheet by means of its receptive field in some causal relationship? Furthermore, neurons of V1 are tuned to specific features of the stimulus such as orientation of the stimulus, color, direction of a moving stimulus, etc. How is such a high-dimensional parameter space mapped onto the basically two-dimensional cortical sheet? Such maps can be found throughout the cortex, thus the layout of neurons is highly correlated with its tuning properties and receptive field position. One such example are ocular dominance patterns. Some neurons in V1 differentiate their response to patterns of different eyes. There are neurons that respond better to right eye stimulation and others that respond better to left eye stimulation. These two types of neurons are not randomly scattered throughout V1, but concentrated in alternating stripes, the so called ocular dominance patterns (see Figure 2.7).

Another noteworthy layout are orientation columns. As mentioned above, many neurons in V1 exhibit orientation selectivity. Hubel and Wiesel showed that as a caricatured is advanced radially from one layer to the next, the preferred orientation remains the same for all selective cells encountered (Hubel and Wiesel, 1962). Furthermore, as an electrode passes tangentially through the cortex in a single layer, the preferred orientation shifts smoothly, such that all orientations can be found within a distance of about 1 millimeter. However also non orientation selective cells can be found. In layer III, there are such neurons that are highly color sensitive. These neurons are concentrated in circular regions, called blobs.

Hence, cortical maps seem to be a fundamental principle of cortical structure, at least in primary sensory areas. However, the role and significance of such maps to cortical computation remains controversial, see *e. g.* (Kaas, 1997; Weinberg, 1997). It seems like cortical maps are not fundamental to the processing itself, but it supports the physical implementation of processing in some way. One physical aspect, the economy of wiring, seems to be widely accepted to play an important role in this context. Since most communication in early processing stages has to be local (*e. g.* local smoothing, lateral inhibition), a cortical map can provide a layout with small wire length and thus might minimize physical costs of the computation. This aspect will be further discussed in Chapter 4 and specifically in Section 4.3.

Chapter 3

Preliminaries on Circuit Models and VLSI

In this chapter, the basic models of biology inspired circuits, basic circuit complexity issues, and the implementation of circuits in very large-scale integration (VLSI) are presented. Computational circuits can be conveniently described and represented as graphs. Therefore we review basic graph-theoretical definitions in Section 3.1.

The study of parallel processing with simple computational units has a long history. For a long time only digital units were considered which leads to a connection to Boolean logic. Therefore, these circuits are called *logic circuits*. Mechanical computers performing logic operations were already designed in the early 19th century by Babbage. In 1938, Claude Shannon analyzed logic circuits formally, which is considered to be the first formal work on this topic, see (Shannon, 1938). The complexity of computations in logic circuits – and therefore the complexity of Boolean Functions – have been investigated at least since Shannon's pioneering paper (Shannon, 1949). In these early years, no connections to biological circuits were established. With the work of McCulloch and Pitts in 1943, the first biology inspired mathematical model of computation was created, see (McCulloch and Pitts, 1943). The anatomy of nerve cells was taken as a blueprint for logical gates that are able to compute and learn functions. Gates, circuits and complexity issues are discussed in Section 3.2. For further reading on logic circuits and circuit complexity, (Wegener, 1987; Savage, 1998) is recommended. See (Valiant, 1994; Haykin, 1994; Siu et al., 1995) for further information on neural computation from the viewpoint of neuroscience, engineering, and theory respectively.

Logic circuits are the basis of modern computers. Therefore, their analysis is not only of theoretical interest, but is also important for physical implementations of artificial computing devices. The technology of such implementations is called VLSI. Due to the layout of circuits in two dimensions, communication cost is

a even more severe bottleneck in VLSI than in biological circuits. This fact is reflected in several models that were introduced to study the cost of VLSI-implementations. We recommend the textbook (Ullman, 1984) and Chapter 12 in (Savage, 1998) to the interested reader. In recent years, analog computation in VLSI circuits has achieved increasing attention. The technology of analog VLSI (aVLSI) is often used with inspirations from biological neural processing, mainly in early sensory stages. Analog VLSI with emphasis on neural systems is extensively studied in the influential textbook (Mead, 1989). Section 3.3.1 gives a brief introduction into technique of VLSI, aVLSI, and physical VLSI models.

3.1 Graphs and Layouts

Computational circuits can be conveniently described and represented as graphs. We give basic definitions on graphs, as described in (Savage, 1998).

Graphs A graph $G = (V, E)$ is a finite set V of *vertices* and a finite set of *edges* $E = \{(v_1, v_2) | v_1, v_2 \in V\}$. That is, $E \subseteq V \times V$. Vertices are also called *nodes*. A graph is *undirected* if for each edge $(v_1, v_2) \in E$ the edge (v_2, v_1) is also in E. A *directed* graph is a graph that is not undirected. The *in-degree* of a vertex in an undirected graph is the number of edges directed into it and its *out-degree* is the number of edges directed away from it. The *degree* of a vertex is the sum of its in-degree and out-degree. A *path* in a graph is a tuple of vertices $(v_1, v_2, \ldots, v_p)$ with the property that $(v_i v_{i+1})$ is in E for $1 \leq v_i \leq p-1$. A path $(v_1, v_2, \ldots, v_p)$ is a *cycle* if $v_1 = v_p$ and $p \geq 2$. The *length of a path* is the number of edges on the path. A *directed acyclic graph* is a directed graph that has no cycles. A *connected* graph is a graph such that for each pair nodes, $v_1, v_2 \in V$, there is a path connecting these two nodes.

Trees A *tree* is a connected acyclic undirected graph. A *rooted tree* consist of a tree T and a distinguished vertex r of T.[1] The vertex r is called the *root* of T. In this thesis, we will always refer to rooted trees if we use the term "tree". A rooted tree is *complete* if all paths from the root to a leaf have the same length. A tree is *binary* if all its internal nodes (non-leaf nodes) have out-degree two. It is common to regard a rooted tree T as a directed graph, with all edges oriented away from the root (Di Battista et al., 1999). If (u, v) is a directed edge in T, then u is the *parent* of v and v is a *child* of u. A *leaf* is a vertex with no children. If v is a vertex of T, then the *subtree* rooted at v consists of the subgraph induced by all vertices on paths directed away from v and has root v.

[1] In the literature, a rooted tree is often termed as a *planted* tree

The layout of graphs A *layout* of a graph is an embedding of the graph in d-dimensional space. Basically, a layout places each node onto some position in space. The definition of a layout discussed in the following is taken from (Fischer and Paterson, 1999).

For some graph $G = (V, E)$, the layout of G in d-dimensional space is a function $\lambda : V \to \mathbb{R}^d$. This function "places" each node onto some point of the space. For VLSI-layouts, d is naturally constrained to $d = 2$. The length of an edge $e = (v, w) \in E$ in the layout λ is the distance of the points $\lambda(v)$ and $\lambda(w)$ with respect to some norm $\|\ldots\|$:

$$length(e) =^{df} \|\lambda(v) - \lambda(w)\|.$$

A natural norm in layout problems is the L_2-norm, which gives the usual Euclidean distance:

$$\|(x_1, \ldots x_d)\|_{L_2} =^{df} \sqrt{\sum_{i=1}^{d} x_i^2}.$$

Since in VLSI-layouts, wires are often constrained to run rectilinearly, the L_1-norm might be better suited. The L_1-norm gives the "city block" metric:

$$\|(x_1, \ldots x_d)\|_{L_1} =^{df} \sum_{i=1}^{d} |x_i|.$$

3.2 Circuit Models

3.2.1 Logic gates

The gate-type introduced by McCulloch and Pitts is referred to as *McCulloch-Pitts neuron* or *threshold gate* (T-gate). A T-gate computes a Boolean function $T : \{0,1\}^k \to \{0,1\}$ of k variables $x_1, \ldots, x_k$ of the form

$$T(x_1, \ldots, x_k) = \begin{cases} 1 & \text{if} \quad \sum_{i=1}^{k} w_i x_i \geq \Theta \\ 0 & \text{otherwise.} \end{cases}$$

The coefficients $w_1, \ldots, w_k \in \mathbb{R}$ are called *weights* and $\Theta \in \mathbb{R}$ is the *threshold*. These parameters determine the function that is computed by the gate and can be learned by applying suitable learning algorithms. Although a drastic simplification, such gates capture some basic aspects of biological neurons. A weighted sum of the inputs is computed and compared to a threshold. If the sum exceeds the threshold, the gate is "active".

One may view circuits composed of T-gates as abstract models for computation on networks of spiking neurons, where the bit 1 is coded by the firing of a

neuron within a certain short time window, and 0 by the non-firing of this neuron within this time window, see *e. g.* (Valiant, 1994). However, T-gates differ from real neurons in several ways (Abeles, 1998; Maass, 1985). Synaptic input sums up linearly, complex dendritic summation and shunting inhibition is not possible (inhibitory synapses are modeled by negative weights). Several membrane properties of real neurons like the gradual recovery of the membrane potential towards resting potential and the relative refractory period are not included. Also, dynamic synaptic transmission such as facillitation and depression are not covered by the model. On the network level, since computation evolves in discrete steps, the input to neurons is assumed to appear as synchronized spikes (up to a few milliseconds). Some of these deficits can be addressed to minor knowledge of neuron physiology in the 40es. However, the model is relatively simple and hence well suited for formal analysis. Therefore, many interesting mathematical results on threshold circuits are known (see (Siu et al., 1995)).

The basic gates of logic circuits perform simple logic operations like a negation (NOT-gate), conjunction (AND-gate), or disjunction (OR-gate). Note that these three types of gates are threshold functions with fixed weights. The NOT-gate can be implemented by setting $w_1 = -1$ and $\Theta = 0$. AND and OR-gates of k inputs are realized by unit weights and threshold $\Theta = k$ and $\Theta = 1$ respectively. The technological advantage of these gates is that they can be implemented in VLSI very efficiently. On the logic level, they are the basic elements of most modern computers.

In this thesis, a third type of gate is considered. A *winner-take-all gate* (WTA-gate) $w_1, \ldots, w_k \in \mathbb{R}$ computes a Boolean function $T : \{0,1\}^k \to \{0,1\}^k$ of k variables $x_1, \ldots, x_k$ where the ith output is 1 if and only if $w_i x_i > w_j x_j$ for all $j \neq i$.[2] Note that the most strongly weighted non-zero input is switched to the output of the gate. This output is the winning output which "inhibits" other outputs to be activated.

The behavior of WTA-gates is motivated by massive lateral connections found in cortical circuits. It is conjectured that lateral inhibition plays an essential role in cortical circuits by allowing one neuron – the winning neuron – to prevent other neurons in its neighborhood to fire. Hence, a WTA-gate does not model single neuron behavior but the behavior of cortical microcircuits. The computational power of WTA-gates was studied in (Maass, 2000).

3.2.2 Feedforward circuits

Gates as introduced above are connected to *circuits* or *networks* to form a natural model for parallel computation. Computational circuits can be defined in several ways. We will follow the intuitively comprehensive definition of (Siu et al., 1995).

[2]Such gates are also termed "hard winner take all" in the literature.

Definition 3.2.1 (Feedforward Network) *A feedforward network can be modeled as a directed acyclic graph $G = (V, E)$ where V is the set of nodes and $E \subset V \times V$ is the set of edges. The nodes are partitioned into three sets:*

input nodes - *with no in-coming edges,*

internal nodes - *with both in-coming and out-going edges,*

output nodes - *with no out-going edges*

Each directed edge $(i, j) \in E$ represents a connection in the network. Each input node is associated with an input to the circuit, and each internal or output node, also called gate*, computes a function of its inputs. (The inputs to the j-th gate are the results of the functions computed at all gates i such that $(i, j) \in E$.)*

In this classical model, no recurrent connections are allowed. The computation progresses from input nodes to output nodes. The output computed by a gate i will be propagated along all its output edges. This output is used as input for the successor gates, *i. e.* all gates j such that $(i, j) \in E$. The output of the network is given by the output computed by output nodes. Hence, if all gates compute a Boolean function, then the circuit computes a multi-output Boolean function $\{0,1\}^{n_{in}} \to \{0,1\}^{n_{out}}$ with n_{in} being the number of input nodes and n_{out} being the number of output nodes.

Definition 3.2.2 (Fan-in/Fan-out) *The number of connections leading into a node is the* fan-in *of that node. Similarly, the number of connections leading out of a node is the* fan-out *of that node. The fan-in and fan-out of a circuit are the maximum fan-in and fan-out, respectively, among all gates in the circuit.*

In the theoretical model, the number of inputs to a gate as well as the number of outputs from a gate may be arbitrary large. In physical implementations, this is not the case. The number of inputs to a cerebral pyramidal neuron is roughly 10^4. Among the cells with largest input size is the cerebellum Purkinje cell with roughly 10^5 synapses on its elaborated dendrites. The number of inputs to gates on chips is much smaller.

Circuit complexity theory relies on two basic complexity measures that are defined below.

Definition 3.2.3 (Circuit Size/Depth) *The* size *of a circuit is the number of gates it contains. The* depth *of a gate is the maximum number of edges along any directed path from the input nodes to that gate. The depth of the circuit is the maximum depth of all gates.*

Normally, the depth of a circuit is interpreted as the time required for a parallel computation. The size of a circuit is normally interpreted as a measure of the required amount of hardware. These complexity measures are further discussed in Chapter 5.

Note that this circuit model is not designed for WTA-gates. WTA-gates compute a multi-output Boolean function. On may overcome this problem by using one node for each output of the WTA-gate and argue about implications in complexity of the circuit. WTA-gates are used in Chapter 6. The way we handle WTA-gates in this thesis is discussed at the introduction of our model in Chapter 5.

3.2.3 Asymptotic growth of functions

In theoretical computer science, costs of computations are normally not measured in the form of tight expressions with exact constants and factors but asymptotically bounded. Asymptotic bounds in circuit complexity theory only consider the order of cost-growth as the problem size (*i. e.* the number of inputs to a circuit) increases. This is very useful since it is often not possible to determine exact bounds. Furthermore, exact bounds are too sensitive to details of the model considered and hence are no gain compared to asymptotic bounds.

The notational conventions are as follows. We say that $f(n)$ is $O(g(n))$ or simply $f(n) = O(g(n))$, if there are positive constants c_0, c_1, and n_0 such that for all $n \geq n_0$, $f(n) \leq c_1 \cdot g(n) + c_0$. Thus $O(n)$ simply means: bounded by a function that is linear in n. We say that $f(n)$ is $\Omega(g(n))$ or simply $f(n) = \Omega(g(n))$, if there are positive constants c_0 and c_1 such that for an infinite number of values of n we have $f(n) \geq c_1 \cdot g(n) + c_0$. Note that the definition of $\Omega(g(n))$ bounds the function f from below even if $f(n)$ is greater than $g(n)$ only from time to time, but infinitely often. Often, this rigid definition is not needed and "$f(n) = \Omega(g(n))$" simply means $g(n) = O(f(n))$.[3]

Asymptotic bounds have turned out to be very useful in computer science. However, for practical purposes, they may be misleading. A logarithmic speed-up in computation time can for practical purposes often easily be outweighted by a larger constant factor or even an additive term. This fact has been pointed out by many authors. We will mention constant factors in our results whenever it is desirable and possible.

3.3 VLSI Technology and Models

Integrated circuits were invented in 1958 and 1959 by Jack Kilby and Robert Noyce. Components such as wires, transistors, resistors, and others are inte-

[3] See *e. g.* (Savage, 1998) or (Ullman, 1984).

grated on a small piece of semiconductor material. Such an implementation of an integrated circuit is called a *chip*. Increasing demand for powerful circuits and physical limitations on the size of chips led to exponential increase of elements placed on the same chip area. This integration of several millions of elements on a single chip is known as very large-scale integration (VLSI). As the number of elements on a chip grows, the fraction of area accounted for wiring computational units increases. Therefore, the layout of elements on the chip is crucial for the amount of resources needed. This fact has been taken into account by physical VLSI models.

3.3.1 VLSI models

VLSI circuits were intensively studied in the early eighties. Several models for VLSI circuits have been proposed. Most of them share many features with one another. The integration of circuit elements can be managed only up to sum spatial precision. Therefore, wires and transistors need to have some specific width and must be separated accordingly. In theory, this separations is identified with the *minimum feature size* λ. Furthermore, a constant number $\mu \geq 1$ of *layers* is postulated where wires may run. The *grid model* given in (Ullman, 1984) is in most aspects equivalent to the model given in (Savage, 1998), which is summarized in the following.

One assumes there that gates, memory cells, input- and output-ports and wires cover rectilinear areas with a width and separation of at least λ. Areas occupied by different gates, input- and output-ports are not allowed to intersect with one another. Areas occupied by wires may intersect with areas occupied by gates, input- and output-ports and also with other wires, but there is a constant bound μ on the number of wire areas to which a point of the plane may belong. Wires run rectilinear. The computation of such a circuit is the computation of a finite state machine. Gates realize functions $\{h : X^2 \to X\}$ on some alphabet X. The complexity measure induced by this model is the *area* of the smallest rectangle that encloses the circuit. The time needed for a computation depends on the transmission time model used. Man bounds are based on the *synchronous model*, where one unit of time is needed for one computational step.

3.3.2 VLSI complexity: the AT^2 bound

The complexity of a function in terms of its VLSI implementation is measured by the chip area A consumed and computation time T. The most prominent lower bounds have been obtained for a combination of these measures, the AT^2 measure. The first lower bound on this measure was derived by Thompson in 1979 (Thompson, 1979). He showed that the discrete Fourier transform in a suitable model needs $AT^2 = \Omega(n^2)$. Later, he generalized this result to sorting.

Lower AT^2 bounds employ a version of the planar separator theorem (Lipton and Tarjan, 1979). The idea of Thompson was generalized by Lipton and Sedgewick (Lipton and Sedgewick, 1981) and Yao (Yao, 1981), who independently developed the use of crossing sequences. Lipton and Sedgewick used fooling sets, showing that the circuit can be "fooled" (*i. e.* it produces the wrong output for some input pattern) if there is not enough communication on the chip. Yao employed communication complexity, introduced by himself in (Yao, 1979). Communication complexity is a somewhat more general view of fooling sets. The communication complexity of a function is the minimal number of bits, two communicating computing devices must exchange in order to compute the function. Communication complexity was also used for lower bounds in circuit complexity theory.

3.3.3 Layout of graphs for VLSI

Layout strategies for VLSI have been developed for several customary graphs. The layout of a graph can be analyzed in the more abstract and less detailed model given in Section 3.1.

The layout of trees was studied intensively. Mead and Rem introduced the H-tree layout, thus showing that a complete binary tree with n leaves can be implemented in $O(n)$ area (Mead and Rem, 1979). In this strategy, leaves are placed throughout the chip surface. We describe, use and study the H-tree layout in Chapter 7. If leaves have to be placed on the boundary of a convex region, the area of any layout is bounded by $\Omega(n \log n)$, which was shown in (Brent and Kung, 1982). More about the layout of trees can be found in Chapter 10.

A general purpose algorithm for several families of graphs, including planar graphs of degree four or less, works by partitioning graphs recursively into subgraphs with few connecting edges (Valiant, 1981; Leiserson, 1980; Floyd and Ullman, 1982). Hence this algorithm also relies on the power of planar separators. Since planar graphs have good separators (Lipton and Tarjan, 1979), all planar graphs can be implemented in $O(n \log^2 n)$ area. Using crossing numbers of graphs (the minimum number of wire crossings over all layouts) and this result, an upper bound on the area of an arbitrary graph with degree four or less related to its crossing number was derived in (Leighton 1981).

All these VLSI models concentrate on the area consumed by wires. This practical approach stands in contrast to most research in circuit complexity theory, since its complexity measures do not account for wiring.

3.3.4 Analog VLSI

While VLSI is based on digital computational elements (computing on binary values in discrete time steps), analog VLSI (aVLSI) relies on the power of ana-

log computation. Transistors are the basic elements of most VLSI circuits. In standard VLSI, transistors are used to form logic elements like AND-gates or registers. In this approach, transistors are used like binary elements with merely two specified states. In analog VLSI, transistors are used as analog elements. The book "Analog VLSI and Neural Systems" by Carver Mead (Mead, 1989) can be regarded as a seminal work in this field. Mead opens a new field by combining the technology of analog VLSI and neuroscience. Mead and his students and colleagues implemented many sensory tasks in analog VLSI in an biologically inspired manner. This leads to elegant and powerful solutions with much less effort and much more efficient than with digital computers.

This approach is not only of interest to VLSI design but also to the field of neural networks and the research on structure and function of the nervous system. Mead states that limitations of connectivity in the nervous system and in silicon technologies forces solutions into a very particular form. Therefore, solutions in the nervous system should be considered to be well suited for VLSI implementations. For example, one system designed in the book is the "silicon retina", an optical device that resembles many important features of the retina like the operation over an illumination range of many orders of magnitude. The circuit is designed such that wire length is kept very small.

Analog counterparts of the gates we introduced in Section 3.2 can be implemented very efficiently in analog VLSI. Threshold gates and winner-take-all gates can be implemented with an area that grows just linearly with the number of inputs to the gate, see (Mead, 1989; Lazzaro et al., 1989).

Chapter 4

Wiring Complexity in the Nervous System: A rush through the Literature

As mentioned earlier, the massive parallel structure of the central nervous system (CNS) leads to an increase in communication needs. Hence, the wiring of the parallel units (*i. e.* neurons) is crucial for the ressources occupied by the CNS. This chapter summarizes the most important research that was done so far on this topic. We start with the first mention of this topic about a century ago by Santiago Ramon y Cajal.

Althoug the problematic of wire length minimization in the nervous system has a prominent mentor with his statements, it was rarely studied in the following decades. Starting in the eighties of the past century, several authors payed attention to these issues. Maybe, they were influenced by VLSI layout-studies of the early eighties, since Mitchison wrote in (Mitchison, 1992): *"Certain features of corical structure - the mappings, stripes and blobs within areas, and areas themselves - are somewhat reminiscent of the layout of computer components, and suggest that the cortex may also be organized so as to economize on neural 'wiring'."* The comparison with relatively simple VLSI layouts seems somewhat ignorant to the amazing geometries and complexity of cortical circuits. However, as Mitchison points out, certain common features do exist and the problem of efficient wiring in the cortex seems to be one key factor of cortical architecture as it is in VLSI circuits. Others seemed to be influence by advances in parallel computing, see *e. g.* (Nelson and Bower, 1990). Thus, a variety of viewpoints and interdisciplinary research enriched the discussion on the topic of neuronal wiring. In the following, we will make a brief review of the most interesting findings in this area. One can divide the research on the basis of the brain scales it focuses, starting at the very global scale of brain-placement up to the scale of single neurons and layout of arborizations. Thus, recent research we will not be reviewed

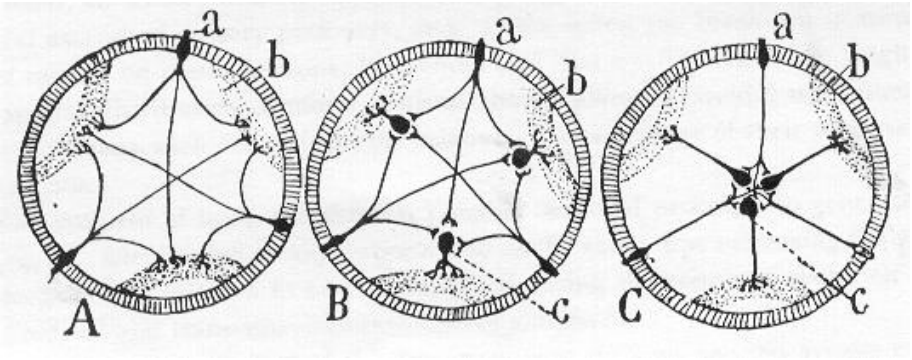


Figure 4.1: Three invertebrates with a simple communication structure. Three sensory inputs are connected to three muscle outputs. The diagram shows the conservation of neuronal material (wire length) by the concentration of neurons in central ganglia. Diagram taken from (Cajal, 1995).

chronicly, but divided by means of the brain-scale it considers.

4.1 Cajals' law of neuronal cytoplasm and transmission time conservation

The first one who mentioned the problem of wiring in the nerous system was Santiago Ramon y Cajal at the dawn of the 20th century, see (Cajal, 1995). Cajal picks up some thoughts by Herbert Spencer and aks why there is a coalescence of parts of the nervous system. For example, the double ganglion chain in less advanced invertebrates that stretches from one end of the body to the other, becomes a single chain in higher worms. Cajal postulates a law, the *law of neuronal cytoplasm and transmission time conservation.* He thus postulates that nature tends to minimize the length of aborizations to minimize the use of neuronal cytoplasm (*i. e.* neuronal substance), space used for computation, transmission time and thus reaction time. Cajal emphasizes his hypothesis with fascinating diagrams, we discuss three of them (see Figure 4.1). The pictures show three invertebrates in cross section. In Figure 4.1A, an idealized invertebrate without motor neurons is shown. Three sensory neurons (a) directly connect to three muscles (b). In Figure 4.1B, motor neurons (c) are introduced into the system. So far, there is no significant reduction in the wire length of the system. Figure 4.1C shows an invertebrate, such as a worm, with motor neurons (c) pulled together in a central ganglia. This arrangement achieves a significant reduction in the wire length of the system, since only six instead of nine long wires are needed for the same communication structure (all to all connection of three sensors to three muscles) as before. Cajal states that *"... the gradual concentration of initially*

isolated ganglion cells yields a significant reduction in length of the processes that link them together; in short, there is a net reduction of cytoplasmic transmission lines."

4.2 Global Scale Wiring: Brain, Cortex, Cortical Areas

In the early nineties, Cherniak considered the question of how neural components should be placed in the nervous system, such that wiring between the components is minimized, see (Cherniak, 1995). The problem of component-placement optimization (CPO) has received attention in VLSI design and can be simply stated as: given the interconnections among a set of components, find the layout of the components that minimizes total connection costs, for example, wire length. For the case of a single component, the problem is simple. For example, since the brain has more anterior connections than posterior ones in all vertebrates, Cherniak states that it shold be placed as anterior as possible. This is an explanaition for why the brain is placed in the head. The problem gets much more difficult if more components are involved. Indeed it can be shown that the problem becomes intractable for natural problem sizes (it is NP-hard). This computational difficulties not only complicate the research of CPO in the neural system, they are also a serious obstacle for nature in the search of the optimal wiring for the approximately 50 areas of the human cerebral cortex. Our universe is simply not old enough for an evolutionary exhaustive search for efficient solutions! Hence, most scientist are in favor of so-called 'quick and dirty' solutions that only approximate optimal solutions but can be carried out much faster. Cherniak overcomes this problem by stating a simple to verify *adjacency rule:"if components are interconnected, then they are positioned contiguously to each other, other things being equal"*. Althoug there was only incomplete information available on connections and contiguities available at that moment, Cherniak concludes from the available data that the examined systems *"departs strongly from random placement in favor of of the adjacency rule"*. The author also examines the neuronal layout of a species for which approximately complete neuronatomy exists, the *C. elegans*. After examing the ganglion layout of this species, which is due to its small size of merely 11 compontents computationally tracktable, Cherniak concludes: *"The actual ganglion layout of* C. elegans *in fact requires the least total length of connecting fiber of any of the millions possible layouts."*

Wiring of the cortex in a more quantitative way was subject to studies by Chklovskii and Stevens in the late nineties. In (Chklovskii and Stevens, 2000), Chklovskii and Stevens considered the number of synapses per neuron as a measure for the complexity of neural circuitry. They show that this complexity is maximized in the cortex at a fraction of wire volume to non-wire volume of 0.6.

The value of 60% of cortical volume being made up of wires is consistent with quantitative measurements, see (Braitenberg and Schüz, 1998). This is an interesting finding, since as the authors mention it shows that *"components of an actual cortical circuit cannot be rearranged in a way that needs more wire without eliminating synapses or reducing wire diameters."*

In the study of wiring in the neocortex, white matter is of special interest, since it consists mostly of bundles of axons running a long distance. Interestingly, as brains are becoming larger throughout different species, the volume of white matter increases diproportionally faster than the volume of gray matter. Allamn observed a power law relation with an exponent close to 4/3 between cortical gray matter volume and white mater volume in primates (Allamn, 1999). Based on these findings, Zhang and Sejnowski described and theoretically justified a scaling law between grey matter and white matter of cerebral cortex (Zhang and Sejnowski, 2000). Empirically, they show that over 59 mammalian species with a gray matter volume G and a white matter volume W that span more than 5 and 6 orders of magnitude, an exponent of 1.23 ± 0.01 can be observed. The authors give a theoretical explaination of this power law based on two assumptions. First they assume the basic uniformity of the neocortex (*i. e.* each piece of unit area cortex sends an receives about the same total cross-sectional area of long-distance connection fibers to and from other cortical regions), and second they assume that the average length of the long distance fibers is minimized by the global geometry of the cortex. This analysis predicts a power law $W \sim G^{4/3-0.1\pm0.02} = G^{1.23\pm0.02}$ with the small correction term accounting for the thickness of the cortex. Hence the theoretical prediction accurately matches the empirical obeservation.

The significance of cortical areas in efficient wiring was subject to investigations by Mitchison in (Mitchison, 1991; Mitchison, 1992). In (Mitchison, 1992), he shows the relevance of cortical areas to efficient wiring by proving that cortical volume would increas significantly if the areas were merged together. In this paper, Mitchison also considers finer levels of structure in the cortex, stripes, blobs and other types of patches within areas. But he does not come to a clear decission whether such structures offer advantages in wiring economy. Several authors were concerned with such questions, which is summarized in the next section.

4.3 Medium Scale Wiring: Cortical Maps, Stripes, Blobs, and Patches

As discussed in Section 2.4, in many cortical regions geometrical organization like retinotopic maps, stripes, blobs and other patchy organization can be observed. In this section, we summarize findings concerned with the influence of such organization on cortical wiring.

Nelson and Bower looked at the issue of brain maps from the perspective of parallel computer architectures (Nelson and Bower, 1990). In parallel multiprocessor machines, one of the factors that most influences the computational performance is how a specific computation is mapped onto the available processors. The efficiency of a particular parallel mapping depends on two key factors. First, the uniformity of load distribution on the processors, measured by the *load balance*. And second, the time overhead of communication between the processors, measured by the *communication overhead*. Different computational tasks give rise to different mappings, broadly categorized into three classes, based on the spatial structure of the load distribution of the problem onto the processors: First, *continuous maps*, where parameters are represented in as smooth and continous manner are optimal for computations with local interactions in problem space. Second, *scattered maps* shwo no apparent spatial structure. Such maps are near-optimal if there is no systematic structure in the pattern of interactions in problem space. And third, *patchy maps*, where spatial organization is intermediate, are good for computations with both, local and non-local interactions in problem space. Maps of each category can also be found in the mammalian central nervous system. The authors ask whether or nor efficiency considerations are reflected in the organization of brain maps. The analysis of the authors gives rise to the hypothesis that brain maps are "designed" in a computationally efficient manner. For example, in continuous maps - such as in the primary somatosensory cortex of a rat - connections are predominantly local, and suggested computations in this region are local ones, such as spatial filtering and local feature extraction. Note that this viewpoint works well with investigations with different approaches. It should be noted that communication overhead in parallel computers is not comparable with that of the nervous system, but as the authors mention, *"However, there is another aspect of communication overhead that may be more generally applicable, which concerns the space taken up by physical connections between processors."*

Durbin and Mitchison adressed the thematic by looking at axonal wiring in the cortex from the viewpoint of local optimization algorithms. In (Durbin and Mitchison, 1990), Durbin and Mitchison argue that cortical maps, such as those for ocular dominance, orientation or retinotopic position in primary visual cortex, could be understood as dimension-reducing mappings from many-dimensional parameter space to the two-dimensional surface of the visual cortex. The problem becomes aparent if one imagines that many feature parameters like position on retina, orientation or ocular dominance (parameter space) have to be maped onto the two-dimensional surface of the cortex. The authors claim that the goal of such mappings is to preserve neighborhood relations as far as possible. Together with the assumption that in primary visual cortex, mostly local operations are performed (*e. g.* inhibitory surrounds, sharpening of tunings), this leads to a minimization of neuronal wire length required. The way that this optimization

is performed is of particular interest. The basic assumption the authors make is that neurons develop their receptive field characteristics under the influence of incoming neural activity. Under such self-organizing algorithms, they simulate the development of cortical maps and come to similar patterns as have been observed in cortex. What is amazing on this study is that wire length minimization comes as a byproduct of a simple self-organizing algorithm. The minimization is not an explizit goal. Instead of mapping from parameter space to the cortex (such that nearby points in parameter space are maped to nearby points on the cortex), there is a mapping the other way around, namely a mapping from cortex surface to parameter space, hence attempting to put nearby cortical points close in parameter space. One has to be aware of the fact that the minimization of wire length may not happen explicitly, but arise naturally from other biological contstraints in the development of the cortex.

Questions related to wiring in cortical maps were also studied recently by Chklovskii and Koulakov (Chklovskii and Koulakov, 2000; Chklovskii, 2000a; Koulakov and Chklovskii, 2001). In (Chklovskii and Koulakov, 2000), the authors investigate the layout of ocular dominans patterns in mammalian primary visual area (V1). They distinguish between two types of neurons, those that are dominated by the left eye, and those that are dominated by the right eye. Depending on the species, these two types of neurons may be homogenous intermixed within V1 (*Salt and pepper* phase), they may be arranged in stripes (*Stripe* phase) or patches where one type builds blobs surrounded by the other type (*L-Patch* or *R-patch* phase). The authors propose a theory for ocular dominance patterns based on the premise that they are obtained by solving a wire length minimization problem. The parameters that determine the type of layout are the fraction of left-eye (respectively right-eye) dominated neurons to the total number of neurons and the N_S/N_O where N_S is the number of synapese a neuron receives from the same class of neurons and N_O is the number of synapses a neuron receives from the other class of neurons. Their analysis shows that within the three possible phases, if $N_S/N_O = 1$, the *Salt and pepper* phase is optimal. Otherwise, one of the other patterns is preferred, depending on the fraction of left-eye and right eye dominated neurons. If these fractions are about equal, then the *Stripe* phase is optimal. When this fraction drops below a critical value of about 0.4, a *patch* phase is prefered. This value of 0.4 is consistent with anatomical data from macaque and Cebus monkeys. It should be noted that this approach is different from Mitchison's in that it drops the requirement of retinotopic considerations.

A similar approach is taken in (Koulakov and Chklovskii, 2001). Instead of ocular dominance patterns, the authors study the layout of orientation preference patterns in mammalian visual cortex, again with the constraint of wire length minimization by the layout. Again, orientation preference patterns vary throughout different species and even in one animal. In such patterns, singularities such as pinwheels and fractures may appear. Previous models could not explain the

usefullness of singularities and suggested that they are merely due to imperfect development of the map. In a simple two dimensional model with neurons placed on a lattice, the optimal layout is computed using an algorithm based on the conventional Metropolis Monte-Carlo scheme. The geometry of the optimal layout depends on the connection function $c(\Theta)$, which gives the number of connections a neuron establishes with other neurons whose orientation preference differs by Θ. For a broad class of connection functions, resulting layouts show singularities similar to observed data, suggesting that singularities are indeed usefull in the minimization of wire length in intracortical circuits.

4.4 Local Scale Wiring: Single Cells

The diversity of neuronal architecture on the microscopic scale of single neurons suggests that nature also pays attention to the cost of wiring on this scale. Tree-like architectures of dendrites and axons which are predominantly present throughout all areas of the nervous system are optimal regarding many types of costs. It should be noted that especially in this microscopic regime, wire length is not the only cost one should pay attention to. The ultimate goal for nature seems to be the minimization of space required by wires. This can make significant differences in many theoretical setup, see below. It is not easy to draw a border between local scale of single neurons and network scale, since the network layout predefines the projections of a neuron and hence its geometry. However, in this section, research is presented that pays less or no attention to the network level and focusses on the level of single neurons.

In (Chklovskii, 2000b), the network level is described by topographic projections between two neuronal layers with different densities of neurons. The input layer (which sends axons) is thougt to lie above the output layer (which receives input from the axons of the input layer via its dendrite). Topographic projection means that each neuron of the input layer connects to the D (*Divergence*) nearest neurons in the output layer. Conversly, a neuron in the output layer receives input from C (*Convergence*) neurons of the input layer. The focus of the work lies in the diameter of axonal and dendritic arbors. Given a specific convergenc/divergenc ratio in the layers, what are optimal diameters such that wire length is minimized? The problem is illustrated in a one dimensional setup, shown in Figure 4.2. Figure 4.2a shows the wiring diagram for $D = 1$ and $C = 6$. In this small scale problem, it is most inportant to make use of synaptic connectinons between dendritic and axonal arbors, since this can significantly reduce the wire length of a layout as can be seen in Figure 4.2b. This layout has narrow axonal and wide dendritic arbors, which is optimal in this case and has much smaller wire length as the layout in 4.2c with wide axonal and small dendritic arbors. Chklovskii proposes the following rule: *" High divergence/convergence ratio favors wide axonal and narrow dendritic arbors while low divergence/convergenc ratio*

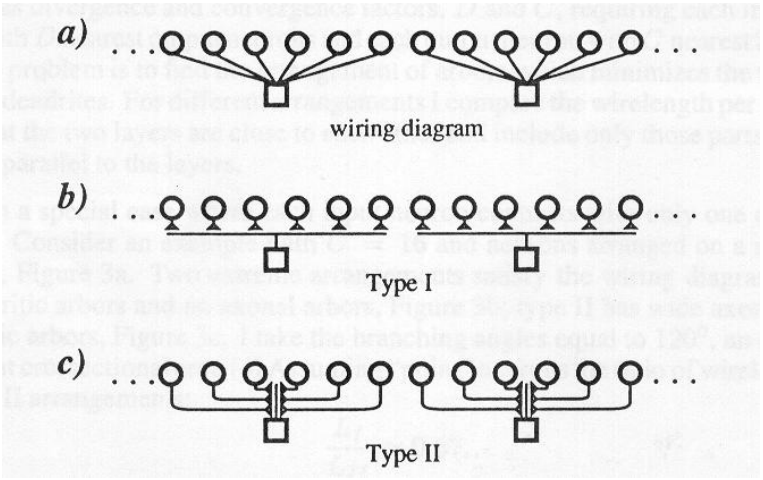


Figure 4.2: Two one dimensional layers with topographic connections and $C = 6$, $D = 1$. (a) Wiring diagram. (b) Type I layout with wide dendritic arbors. This layout is optimal. (c) Type II layout with wide axonal arbors.

favors narrow axonal arbors and wide dendritic arbors.", or in terms of neuronal densities: *" Sparser layer has wider arbors."*. This rule is quantified and derived from the wire length principle for the case of one dimensional and tow dimensional projections. The case that axons and dendrites have different cross-sectional areas h_a and h_d is also considered. The derived expressions are modified such that the total volume of connections is minimized. Let s_a and s_d be the axonal and dendritic arbor diameter respectively. The optimal ratio of dendritic and axonal arbor diameter is

$$\frac{s_d}{s_a} = \sqrt{\frac{Ch_a}{Dh_d}} \; .$$

In (Cherniak, 1992), Cherniak studies the layout of neuron arbors in a even more local version. Instead of looking at the whole arborizatin, he examines how local bifurcation junctions of arbors are realized. If wire length is optimized at this local level, junctions should form 120° branch angles. However, in dendrites and axons there is a tendency to an internal angle of about half of this. Cherniak suggests that this angle can be expained by different costs of trunk and branches. Cherniak assumes that not the wire length but the volume of the junction is locally minimized. Since branches usually have smaller cross-sectional area than the trunk, an angle smaller than 120° would reduce this cost. The mathematically derived predictions are in good agreement with the neuroanatomical data. Hence, Cherniak concludes: *"What is most strongly minimized at the individual junction level is total volume of the arborizations, rather than connections length, signal propagation speed, or surface area."*

Chapter 5

A Complexity Measure for Sensory Processing: Total Wire Length

The issue of circuit complexity is exhaustively studied in the context of theoretical computer science. Circuit complexity theory is established as a fundamental tool for the analysis of computation and learning in parallel systems. In recent years interest has grown in understanding the complexity of circuits for early sensory processing, both from the biological point of view and from the point of view of neuromorphic engineering (see (Mead, 1989)). There is growing demand for energy-efficient hardware for sensory processing, and complexity issues are critical since the number n of parallel inputs which such circuits have to handle is usually quite large (for example $n \geq 10^6$ in the case of many visual processing tasks). However classical circuit complexity theory provides little guidance for the design of efficient circuits for sensory processing tasks, both because its focus lies on a different set of computational problems, and because its traditional complexity measures are not tailored to those resources that are of primary interest in the analysis of neural circuits in biological organisms and neuromorphic engineering.

In Section 5.1, the basic model considered in this thesis is introduced. This model is analyzed with respect to cortical circuits in Section 5.2. We want to compare results on total wire length with results on VLSI-models and also get direct results on VLSI layout of sensory processing functions. Therefore, a basic – in some aspects expanded – model of VLSI layout is presented in Section 5.3. This chapter is partly based on (Legenstein and Maass, 2001a) and (Legenstein and Maass, 2001b).

5.1 Introducing Total Wire Length

The most frequently considered complexity measures in traditional circuit complexity theory are the number (and types) of gates, as well as the depth of a circuit. We will follow traditional circuit complexity theory in assuming that the underlying graph of each circuit is a directed graph without cycles (see Chapter 3.2). Neural circuits in "wetware", as well as most circuits in analog VLSI, contain in addition to feedforward connections also lateral and recurrent connections. This fact presents a serious obstacle for a direct mathematical analysis of such circuits. The standard mathematical approach is to model such circuits by larger feedforward circuits, where new "virtual gates" are introduced to represent the state of existing gates at later points in time. The depth of a circuit is defined as the length of the longest directed path in the underlying graph, and can also be interpreted as the computation time of the circuit. Most research had focused on the classification of functions that can be computed by circuits whose number of gates is bounded by a polynomial in the number n of input variables. This implicitly also provides a polynomial – although typically quite large – bound on the number of "wires" (defined as the edges in the underlying graph of the circuit), but no bound on the total length of these wires.

In contrast to these traditional complexity measures in circuit complexity theory, it has frequently been pointed out that "economizing on wire is the single most important priority for both nerves and chips" (Mead, 1989). This view has been adopted by quite a number of neuroscientists as a guiding principle for understanding cortical circuitry as we outlined in Chapter 4. The goal of this chapter is to make this principle also applicable to computational tasks. We introduce a simple method for estimating the total wire length required by a specific circuit design. Furthermore we show that it is feasible to use the minimization of total wire length as a guiding principle for the design of efficient algorithms and circuits for concrete computational problems.

We propose the following abstract model for estimating the total wire length required for the neural implementation of an abstract circuit design (which is formally defined as a directed graph with nodes labeled by specific types of gates, or by input- or output variables):

Gates, input- and output-ports of a circuit are placed on different nodes of a 2-dimensional grid (with unit distance 1 between adjacent grid nodes). Connections between them are represented by (unidirectional) wires that run through the grid-plane in any way that the designer wants, in particular wires may cross and need not run rectilinearly (wires are thought of as running in the 3 dimensional space above the plane, without charge for vertical wire segments). We define the minimal value of the sum of all wire lengths that can be achieved by any such arrangement as the total wire length *of the circuit.*

We would like to make this model also applicable to cases where for $k > 2$ some

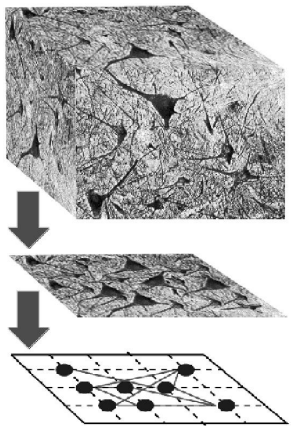


Figure 5.1: The relationship between cortical circuitry and a simple mathematical model is explained by a projection onto a 2-dimensional plane.

special functions of k inputs such as the function computed by a threshold gate or winner-take-all gate are computed by neural microcircuits or in analog VLSI by efficient subcircuits that employ a number of transistors, total wire length and area that are all linear in k, with a setting time that is independent of k (see (Lazzaro et al., 1989)). In the relatively abstract context of this model we model such computational modules as "threshold gates" of "winer-take-all gates" of k inputs, that take one unit of time for their computation like all the other gates, but which occupy each a set of k intersection points of the grid that are all connected by an undirected wire (whose length contributes to the total wire length) in some arbitrary fashion. Any one of these k nodes may be used to provide one of the k inputs or to extract one of the outputs of the function.

We will allow that a wire from a gate or input port may branch and provide input to several other gates. For reasonable bounds on the maximal fan-out (10^4 *in the case of neural circuits) this is realistic both for neural circuits and for VLSI.*

The length of a wire is defined by the Euclidean distance of the nodes the wire connects (L_2-norm). See also the definition of a layout in Chapter 3.1. The attractiveness of this model lies in its mathematical simplicity. Nevertheless it provides a useful criterion for judging whether an abstract circuit design is biologically realistic from the point of view of the total length of axonal and dendritic branches that it requires.

5.2 Total Wire Length and the Cortex

In the cortex, neurons do not occupy the nodes of a 2-dimensional grid, but a roughly 2 mm thick 3-dimensional sheet of "grey matter". However since there

exists a strikingly general bound on the order of 10^5 for the number of neurons under any mm^2 of cortical surface, the density of neurons in these circuits remains bounded if the circuits are projected onto a 2-dimensional plane running parallel to the cortical surface, (see Figure 5.1). This observation provides the justification for the assumption of our abstract model that the neurons are positioned at the nodes of a 2-dimensional grid. It also yields a biologically realistic estimate for the length of an edge between two nodes in this grid: $10^{-5/2}$ mm. Since we are considering just the 2-dimensional projection of a 3-dimensional neural circuit, we can estimate in this way only the contribution of all horizontal components of all connections. However since there exist quite good estimates for the total amount of dendritic and axonal wires under any mm^2 of cortical surface (8 km according to (Koch, 1999)), we know that also the horizontal component of all connections adds up to at most 8 km. This implies that the average cortical circuit with j neurons has an implementation in our simple 2-dimensional grid model where its total wire length is at most 25300 j grid units. The total length of axons and dendrites under any mm^2 of cortical surface is estimated to be on the order of 8 km (Koch, 1999). If one divides this number by the estimate 10^5 for the number of neurons under any mm^2, one arrives at an average wire length of 80 mm per neuron. Translated into our grid unit measure, this is equivalent to $80 \cdot 10^{5/2} = 25300$ grid units. The total bound of 25300 j grid units for the total wire length of cortical circuits with j neurons is likely to be an overestimate, since the preceding argument assumes that all of the 8 km of wires under a mm^2 of cortical surface can be used for horizontal connections. In this setup we arrive at a heuristic condition for any abstract circuit design with j neurons to be biologically realistic: it must have an implementation in our 2-dimensional grid model with a total wire length of at most $25300 \cdot j$ grid units.

In circuit complexity theory it is customary to express the total amount of resources used in terms of the number n of circuit inputs. For the sake of simplicity we denote in the formal results of this thesis the number of pixels by n, and the actual number of circuit inputs is some constant multiple of n. Several empirical studies provide estimates for the order of magnitude for the number n of inputs and the number of neurons in biological neural circuits for sensory processing, see (Abeles, 1998; Koch, 1999; Shepherd, 1998; Braitenberg and Schüz, 1998). In the following, we give a brief calculation on the number of inputs and processing units of the visual system of primates.

The number of neurons that transmit information from the retina (via the thalamus) to the cortex is estimated to be around 10^6 (all estimates given are for primates, and they only reflect the order of magnitude). The total number of neurons in the primary visual cortex of primates is estimated to be around 10^9, occupying an area of roughly 10^4 mm^2 of cortical surface. Since the total length of axonal and dendritic branches below one mm^2 of cortical surface is estimated to be at most 8 km, this yields an upper bound of 10^{11} mm for the

total wire length of primary visual cortex. Thus if one assumes for example that 100 separate circuits are implemented in primary visual cortex, each of them can use 10^7 neurons and a total wire length of 10^9 mm. Hence realistic bounds for the complexity of a single one of these circuits for visual pattern recognition with $n = 10^6$ inputs are $10^7 = n^{7/6}$ neurons, and a total wire length of $10^{11.5} < n^2$ grid units in the framework of our model.

The whole cortex receives sensory input from about 10^8 neurons. It processes this input with about 10^{10} neurons and less than 10^{12} mm total wire length. If one assumes that 10^3 separate circuits process this sensory information in parallel, each of them processing about 1/10th of the input, one arrives at $n = 10^7$ inputs for each circuit, and an average circuit can use on the order of n neurons and a total wire length of $10^{11.5} < n^2$ grid units in the sense of our model. The actual resources available for sensory processing are likely to be substantially smaller, since most cortical neurons and circuits are believed to have many other functions (for example related to memory, learning and attention) besides online sensory processing.

These calcuations suggest that only those circuit architectures for sensory processing are biologically realistic that can be implemented in our 2-dimensional grid with a number of gates that is almost linear in the number n of inputs, and a total wire length that is quadratic or subquadratic in n – with the additional requirement that the constant factor in front of the asymptotic complexity bound needs to have a value not larger than 1. Since most practically arising asymptotic bounds involve larger constant factors, one should focus on circuit architectures that can be implemented in our model with clearly subquadratic bounds for their total wire length.

In Chapters 6 and 7 we begin the investigation of circuits for basic pattern recognition tasks that can be implemented within biologically realistic bounds with regard to their number of gates and their total wire length. We show in Chapter 7 that two basic pattern recognition tasks can be solved under these severe complexity constraints, one of them even with a number of gates and a total wire length that are both linear in the number n of inputs. Obviously the algorithmic design and architecture of such circuits has to differ from previously proposed circuits for sensory processing.

5.3 Total Wire Length and VLSI

Our model for estimating the total wire length is easy to handle since one does not have to worry about how exactly the wires need to be routed in order to avoid interference. This laxness may be justified for modeling cortical circuits – since their 2 mm vertical dimension leaves a lot of room to route axons whose thickness lies in the μm range. But it is not a priori justified for estimating the actual total

wire length required by a VLSI-implementation of the same circuit, since currently available VLSI-technologies allow just a small number (typically less than 10) of horizontal layers in which wires can be routed. However it turns out that those circuit designs that we consider in this thesis require in the common abstract model for VLSI-area an area that is asymptotically as small as the total wire length that they require in the more abstract model introduced in this thesis. This suggests that the circuit designs that we consider in this thesis do not only satisfy the complexity requirements imposed by cortical circuitry, but can potentially also be implemented in VLSI.

5.3.1 An abstract VLSI model

The abstract model for VLSI to which the theorems of Chapters 6 and 7 refer is closely related to customary VLSI models discussed in Chapter 3.3.1. However, our approach differs from classical ones in some important aspects, which will be discussed in Section 5.3.2. In order to avoid confusion, we define our model in the following. Our model can be seen as a variation of the model given in (Savage, 1998) (see also Chapter 3.3.1).

Physical model: *One assumes that gates, input- and output-ports and wires cover rectilinear areas with a width and separation of at least λ. Areas occupied by different gates, input- and output-ports are not allowed to intersect with one another. Areas occupied by wires may intersect with areas occupied by gates, input- and output-ports and also with other wires, but there is a constant bound μ on the number of wire areas to which a point of the plane may belong. Wires run rectilinear, i. e. they run horizontaly and vertically.The complexity measure induced by this model is the* area *of the smallest rectangle that encloses the circuit.*

Since we consider in this thesis also circuits that involve gates with a large number of inputs such as threshold gates, we extend the model for VLSI-area by assuming that a threshold gate with k inputs can be implemented by $k+1$ gates (k of them for multiplying a binary input with a weight, one for comparing the weighted sum with the threshold) that are linearly connected by a wire. We follow (Savage, 1998; Ullman, 1984) in assuming that in the VLSI-model one unit of time is needed to transmit a bit along a wire (of any length), and also for each gate switching. However in contrast to (Savage, 1998) we always assume that all inputs are presented in parallel.

Computational Model: *The VLSI layout is based on a logic circuit C that computes the corresponding function. The gates and connections of the layout implement the graph of C. Hence, each gate g_i with fan-in $k_{in,i}$ computes the $k_{out,i}$-valued Boolean function $f_i : \{0,1\}^{k_{in,i}} \to \{0,1\}^{k_{out,i}}$ defined by the circuit C. Therefore, we consider layouts of very basic gates, like AND-gates, OR-gates, or*

threshold gates. For these gates, the computed function is always single-valued, and hence $k_{out} = 1$. We also consider winner-take-all gates for which the number of outputs k_{out} is equal to the number of inputs k_{in}. The computation of the layout is given by the computation of the circuit C and is therefore feedfoorward.

5.3.2 Discussion of the model

Several aspects of the given model are noteworthy to be discussed. Probably the most striking difference to classical models is that unbounded fan-in is allowed in our model. In the "grid model" discussed in (Ullman, 1984), the fan-in of gates is restricted to be constant. This restriction is necessary due to the computational model because the function that a single gate computes is not specified. Therefore, a gate having all inputs to the circuit as input could compute the whole function, which would obviously lead to silly results. We do not need this restricion because our gates are constrained to compute specific functions like a threshold function. Hence, our model is more closely related to circuit complexity theory. Ullman discusses the topic of constant fan-in and the question if complex functions can be implemented by gates of linear size[1]. For the gates we consider, such implementations are known (see Chapter 3.3.1). In VLSI-implentations, gates of very large fan-in are problematic because of several reasons (high input capacitance, increased settling time, implementation difficulties). However, a fan-in that grows logarithmic or with the square root of the problem size are reasonable. At results on our model, the fan-in of the circuit will be mentioned if not obvious.

The model given in (Savage, 1998) is based on gates that compute functions $\{h : X^2 \to X\}$ on some alphabet X. Hence, the fan-in of gates in this model is also bounded, but the circuit operates on an alphabet X rather than on binary values. This feature can be justified for implementations, where for example wires for integers travel together on the chip surface. Moreover, Savage bases the computational approach on finite state machines, where such a representation of values makes sence. Our basis are logical circuits, thus the binary representation is the most natural one. Savage notes that it is important to take care of this simplification if the set X is big. Furthermore, in contrast to (Savage, 1998), we do not emply memory cells in our model, since the use of memory cells is not included in the feedforward circuit model.

Another issue to be discussed is the time taken by a computation. We assume that one unit of time is needed to propagate a signal down a wire and to switch the gates. This transmission model is called the *synchoronous model*. For long wires, this model is not accurate, since the time used for transmission grows with the length of the wire. In the *transmission line model*, the delay of a wire is assumed to be linear with the length of that wire. However, for long wires, the

[1] See pages 30–34 in (Ullman, 1984).

situation is even worse. Since both the resistance and the capacitance of a wire grow linearly in its length, the product of them, which roughly determines the propagation time, grows quadratically. However, in practice – since integrated circuits are relatively small – propagation time does not dominate switching time. Furthermore, if there are a few very long wires, special measures can be taken to speed up propagation time. We will never explicitly talk about computation time of a circuit. The depth of a circuit is noted, which can be interpreted as the computation time.

For the purpose of this thesis, the model is well fitted. It gives reasonable bounds on VLSI-implementation and is tightly coupled to the other two essential fields we consider, circuit complexity theory and wiring complexity in biological neural system. The main difference to the model for estimating total wire length is the constant number of layers and hence the constant bound on crossings on a given area in the VLSI model. This reflects the limitations of VLSI emerging from its two-dimensional layout structure. However, t turns out that the same circuit design techniques that we introduce in Chapters 6 and 7 for the sake of efficient cortical processing of sensory data also yield circuits that require relatively little area in the abstract model for VLSI-area.

Chapter 6

Global Pattern Detection and 1-Dimensional Maps

In circuit complexity theory, circuits are analyzed that compute functions like sorting or addition. Such functions are more of mathematical interest than of interest from the viewpoint of sensory processing systems. It can be argued that scale- and translation-invariant pattern recognition tasks have to be solved in several sensory processing systems. One of the most basic pattern recognition problems is whether a certain local feature occurs in some linear array to the left of some other local feature. We construct in this chapter circuits that solve this problem with an asymptotically optimal number of threshold gates. Furthermore it is shown that much fewer threshold gates are needed if one employs in addition a small number of winner-take-all gates. In either case the circuits that are constructed have linear or almost linear total wire length, and are therefore not unrealistic from the point of view of physical implementations. This chapter is based on the publication (Legenstein and Maass, 2001c).

6.1 Introduction

Biological neural circuits can solve a number of complex pattern recognition tasks very fast, in 100 - 150 milliseconds, see (Thorpe et al., 1996). Since the computational units of neural circuits are relatively slow compared with a transistor, observation gives rise to some optimism regarding the possibility to build artificial circuits, for example analog VLSI chips, that solve complex real-world pattern recognition tasks in real-time. Classical circuit complexity theory is of little help in the search for such super-efficient circuit designs. Apparently there are two reasons for this. The complexity of circuits is usually analyzed in terms of their number of gates, and much of the existing work focuses on the derivation of polynomial upper bounds for the number of gates. But most circuits that appear

to be feasible from this point of view cannot be practically implemented, especially if the number n of input variables is very large (like for example in vision tasks where often $n \approx 10^6$). Furthermore even those circuit designs where one has been able to derive linear or almost linear upper bounds for the number of gates can usually not be implemented in VLSI because the required number of wires (= edges), or the required length of wires grows too fast with the number n of input variables. Therefore we focus in this chapter directly on the total wire length (the definition is given below) as the most salient complexity measure, the usually most restricted and hence arguably most relevant complexity measure for the practical implementation of an abstract circuit design.

Another obstacle for the application of classical circuit complexity theory to the design of efficient circuits for pattern recognition arises from the fact that most complexity studies focus on arithmetic and graph-theoretic problems, rather than on those computational tasks that typically arise in the context of pattern recognition. Both, in common machine vision approaches and in biological neural circuits for vision, the raw pixel image is first preprocessed by an array of local feature detectors (*e. g.* for the detection of edge segments, line segments, Gabor filters). Hence pattern recognition problems in vision typically require to find particular spatial arrangements of those local features, that are reported by local feature detectors. The local feature detectors are typically arranged in a one- or two- dimensional array that reflects the geometrical relationship between their receptive fields in the sensory space. In order to initiate a computational complexity analysis of algorithmic problems of this type we investigate in this chapter the arguably most simple problem of this type. We assure that there are two types of local feature detectors with binary output that are linearly arranged at n positions: detectors $a_0, \ldots, a_{n-1}$ for feature a and detectors $b_0, \ldots, b_{n-1}$ for feature b. The pattern recognition task is to decide whether feature a is reported at a location i to the left of some location j where feature b is reported. In other words, we analyze the circuit complexity of the Boolean function P^n_{LR} from $\{0,1\}^{2n}$ into $\{0,1\}$ with

$$P^n_{LR}(a_1, \ldots, a_n, b_1, \ldots, b_n) = \begin{cases} 1, & \text{if } \exists i, j (i < j \text{ and } a_i = b_j = 1) \\ 0, & \text{else .} \end{cases}$$

We investigate in this chapter circuits that compute P_{LR} with two types of gates that are both frequently discussed in models for neural computation: threshold gates and winner-take-all (WTA) gates. Both of these gates can be implemented very efficiently in analog VLSI, with an area that grows just linearly with the number k of inputs to the gate, see (Mead, 1989), and (Lazzaro et al., 1989). Recall that a threshold gate computes a Boolean function $T : \{0,1\}^k \to \{0,1\}$ of the form $T(x_1, \ldots, x_k) = 1 \Leftrightarrow \sum_{i=1}^k w_i x_i \geq w_0$. Also recall that a winner-take-all gate with weights $w_1, \ldots, w_k$ computes a Boolean function $W : \{0,1\}^k \to \{0,1\}^k$

where for input $x_1, \ldots, x_k$ the ith output is 1 if and only if $w_i x_i > w_j x_j$ for all $j \neq i$.

We employ the model described in Chapter 5.1 for estimating the total wire length required for the neural implementation of this function.

The attractiveness of this model lies in its mathematical simplicity, and in its generality. It provides a rough estimate for the cost of connectivity both in artificial (basically 2-dimensional) circuits and in neural circuits, where 2-dimensional wire crossing problems are apparently avoided (at least on a small scale) since dendritic and axonal branches are routed through 3-dimensional cortical tissue. We also give bounds on the complexity of our circuit designs in the common abstract model for VLSI. We use the abstract VLSI-model described in Chapter 5.3 to compare our results on total wire length with VLSI layouts.

We will show in Theorem 6.2.1 that P^n_{LR} can be computed by a circuit consisting of $O(\log n)$ threshold gates in depth 2, with a total wire length of $O(n \log n)$. Theorem 6.2.2 implies that no feedforward circuit can compute P^n_{LR} with fewer threshold gates. Finally it is shown in Theorem 6.3.1 that P^n_{LR} can be computed by a circuit of depth 2 consisting of two winner-take-all gates and one threshold gate, with total wire length $O(n)$. This result demonstrates that winner-take-all gates can in some contexts be computationally much more powerful than threshold gates, although they do not require much more area in analog VLSI (see (Maass, 2000) for some more general results in this direction).

6.2 1-Dimensional Pattern Detection with Threshold Gates

We start the analysis of this pattern recognition task by showing that P^n_{LR} can be computed very fast by a circuit consisting of $O(\log n)$ threshold gates. We also give bounds on the total wire length of this circuit and the area that it occupies in a VLSI layout. Then we show that the number of threshold gates employed by this circuit is asymptotically optimal.

Theorem 6.2.1 *P^n_{LR} can be computed by a feedforward circuit of depth 2, consisting of $2 \log n + 1$ threshold gates with total wire length $O(n \log n)$ and area $O(n \log n)$ in a VLSI layout.*

Proof: Denote with $\underline{a} = (a_0, \ldots, a_{n-1})$ and $\underline{b} = (b_0, \ldots, b_{n-1})$ the two vectors of input features. It will be convenient to denote the position l of the leftmost occurring feature a with $min(\underline{a})$ and the position r of the rightmost occurring feature b with $max(\underline{b})$. Note that these functions are not defined if there is no feature a respectively b present. The following precise definition eliminates this

ambiguity. We define

$$min(\underline{a}) = \begin{cases} min\{i|a_i = 1\}, & \text{if } \underline{a} \neq (0,\ldots,0) \\ n-1, & \text{otherwise.} \end{cases}$$

Furthermore we define

$$max(\underline{b}) = \begin{cases} max\{i|b_i = 1\}, & \text{if } \underline{b} \neq (0,\ldots,0) \\ 0, & \text{otherwise.} \end{cases}$$

Note that with this simple definition,

$$P^n_{LR}(\underline{a},\underline{b}) = 1 \Leftrightarrow min(\underline{a}) < max(\underline{b}). \tag{6.1}$$

We construct a threshold circuit which computes the binary encoding of $min(\underline{a})$ and $max(\underline{b})$ in its first layer. Let us call the function that maps $\underline{a}$ onto the binary representation of $min(\underline{a})$ *MinMux* and the function that maps $\underline{b}$ onto the binary representation of $max(\underline{b})$*MaxMux* respectively. The comparison of their outputs yields the desired output of P^n_{LR}.

For convenience, let $n = 2^k$ for some natural number k. The precise definitions of the functions *MinMux* and *MaxMux* are as follows.

$MinMux^n : \{0,1\}^n \to \{0,1\}^k$ is defined by

$$MinMux^n(\underline{a}) = \begin{cases} \text{binary encoding of } min\{i|a_i = 1\}, & \text{if } \exists i(a_i = 1) \\ \text{binary encoding of } n-1, & \text{otherwise .} \end{cases}$$

$MaxMux^n : \{0,1\}^n \to \{0,1\}^k$ is defined by

$$MaxMux^n(\underline{b}) = \begin{cases} \text{binary encoding of } max\{i|b_i = 1\}, & \text{if } \exists i(b_i = 1) \\ \text{binary encoding of } 0, & \text{otherwise .} \end{cases}$$

This comparison of the two $\log n$-bit binary numbers represented by $MinMux$ and $MaxMux$ can be carried out by an additional threshold gate with weights linear in n.

In the following, we construct a circuit consisting of $\log n$ threshold gates that computes *MinMux*. Note that, for any input assignment, setting $a_{n-1} = 1$ does not change the value of the function. We will use this trick to make sure that the output of the circuit is the binary encoding of $n-1$ if there is no feature a present.

Let m_j denote the j-th output bit of $MinMux^n$ $(0 \leq j \leq k-1)$, such that $min(\underline{a}) = \sum_{j=0}^{k-1} 2^j m_j$. The j-th bit of the binary encoding of some natural number x is 1 if $\lfloor \frac{x}{2^j} \rfloor \equiv 1 \mod 2$ and 0 otherwise.

This leads to the following threshold function for m_j:

$$m_j(a_0,\ldots,a_{n-1}) = \begin{cases} 1, & \text{if } \sum_{i=0}^{n-1} a_i 2^{n-i}(-1)^{1+(\lfloor \frac{i}{2^j} \rfloor \mod 2)} \geq 1 \\ 0, & \text{otherwise .} \end{cases}$$

Let $l = min(\underline{a})$ and suppose that $\lfloor \frac{x}{2^j} \rfloor \equiv 0 \mod 2$. It follows that

$$\sum_{i=0}^{n-1} a_i 2^{n-i} (-1)^{1+(\lfloor \frac{i}{2^j} \rfloor \mod 2)} \leq$$
$$2^{n-l}(-1) + \sum_{i=l+1}^{n-1} 2^{n-i} \leq$$
$$-2^{n-l} + \sum_{i=1}^{n-l-1} 2^i \leq -2 < 1$$

and the output of the threshold gate is 0. Suppose that $\lfloor \frac{x}{2^j} \rfloor \equiv 1 \mod 2$. It follows that

$$\sum_{i=0}^{n-1} a_i 2^{n-i} (-1)^{1+(\lfloor \frac{i}{2^j} \rfloor \mod 2)} \geq$$
$$2^{n-l}(+1) - \sum_{i=l+1}^{n-1} 2^{n-i} \geq$$
$$2^{n-l} - \sum_{i=1}^{n-l-1} 2^i \geq 1$$

and the output of the threshold gate is 1. Hence, m_j is the j-th bit of the binary representation of $min(\underline{a})$.

MaxMux can be constructed in a similar manner. Hence, each m_j can be computed by one threshold gate and the depth and size of the circuit given in Theorem 6.2.2 follow.

The VLSI-layout of the circuit for P^4_{LR} is shown in Figure 6.1a. We place the gates for $MinMux$ on rows beneath $a_0, \ldots, a_{n-1}$ and the gates for $MaxMux$ on rows beneath $b_0, \ldots, b_{n-1}$. Since the circuit consists of $\log n$ gates for $MinMux^n$ and $\log n$ gates for $MaxMux^n$ this occupies $O(\log n)$ rows. The comparison gate can be placed in the column between those gates. Hence, the layout of the circuit occupies $O(n \log n)$ area. A layout to estimate the total wire length is similar. The layout of the circuit for P^4_{LR} in is shown in Figure 6.1b. Simply replace a threshold gate of k inputs by k nodes that are connected by a common wire to sum up the inputs. This results in a wire length of $O(n)$ within each gate. The wire from an input port to its successor gates may spread and hence is $O(\log n)$ in length. The comparison gate has a total wire length of $O(\log n)$. Summing up those lengths, results in a total wire length of $O(n \log n)$.

■

The following lower bound result shows that the number of threshold gates used by the circuit of Theorem 6.2.1 is asymptotically optimal:

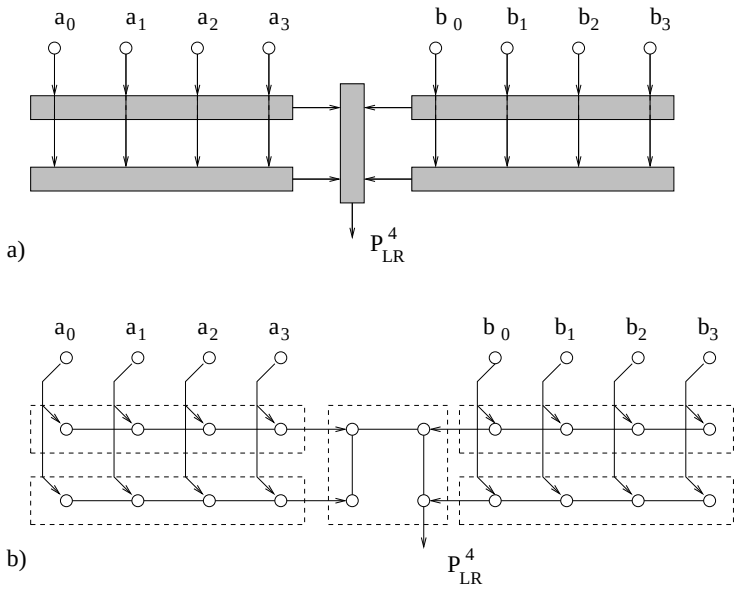


Figure 6.1: a) The VLSI-circuit layout for P_{LR}^4. The gates for $MinMux$ and $MaxMux$ are placed on rows beneath the inputs. The area used by this layout is $O(n \log n)$. b) A layout to estimate the total wire length of the circuit. A threshold gate of k inputs is represented by k nodes that are connected by a wire (wires without arrows). Such gates are indicated by a dashed rectangle. The total wire length is $O(n \log n)$.

Theorem 6.2.2 *Any feedforward circuit consisting of threshold gates needs to have at least* $\Omega(\log n)$ *gates for computing* P_{LR}^n.

We use the gate-elimination method to prove Theorem 6.2.2. The gate-elimination method was used widely in classic circuit complexity theory. It was used in the context of threshold circuits in a paper by Georg Schnitger and Bhaskar DasGupta (see (DasGupta et al., 1996)). In our case we have to exhibit some properties of P_{LR} that allow us to assign constants to inputs of a circuit S_n that computes P_{LR}^n, such that the circuit computes P_{LR} on the remaining non-constant variables. Furthermore, we use these properties to show that at least one threshold gate computes a constant after the assignment of constants to at most $\frac{63n}{64}$ of its input variables. We use this restriction to construct a circuit that computes $P_{LR}^{n/64}$ and has at least one gate less that S_n. Hence, the size of S_n is at least $S_{n/64} + 1$, which we use as an induction step. The induction hypothesis is that a circuit S_n that computes P_{LR}^n consists of at least $\lfloor \log_{64} n \rfloor$ threshold gates [1].

[1] $\lfloor x \rfloor$ denotes the floor of x, which is $\lfloor x \rfloor = max\{y \in \mathbb{N} \cup \{0\} | y \leq x\}$.

Proof: We will at first exhibit the three properties of P_{LR} that will be the basis for the proof. Then we will show, how to eliminate one threshold gate in a circuit computing P_{LR} by assigning constants to a fixed fraction of its inputs. Finally, we will use this gate-elimination to give an inductive prove of the lower bound.

The properties of P_{LR} given below are easy to verify.

property 1:

$$P^n_{LR}(a_0,\ldots,a_{i-1},0,a_{i+1},\ldots,a_{n-1},b_0,\ldots,b_{i-1},0,b_{i+1},\ldots,b_{n-1}) =$$
$$P^{n-1}_{LR}(a_0,\ldots a_{i-1},a_{i+1},\ldots,a_{n-1},b_0,\ldots,b_{i-1},b_{i+1},\ldots,b_{n-1})$$
$$\text{for all } i \in \{0,\ldots,n-1\} \quad .$$

property 2:

$$P^n_{LR}(0,\ldots,0,a_{k+1},\ldots,a_{n-1},1,\ldots,1,b_{k+1},\ldots,b_{n-1}) =$$
$$P^{n-k}_{LR}(a_{k+1},\ldots,a_{n-1},b_{k+1},\ldots,b_{n-1}) \quad \text{for all } k \in \{0,\ldots,n-2\} \quad .$$

property 3:

$$P^n_{LR}(a_0,\ldots,a_{n-1-k},1,\ldots,1,b_0,\ldots,b_{n-1-k},0,\ldots,0) =$$
$$P^{n-k}_{LR}(a_1,\ldots,a_{n-1-k},b_1,\ldots,b_{n-1-k}) \quad \text{for all } k \in \{1,\ldots,n-1\} \quad .$$

Let S_n be a threshold circuit computing P^n_{LR}. We show how to eliminate one gate in S_n by exploiting the properties of P_{LR} given above. We assume that n is a power of 64. If it is not, use property 1 to obtain a threshold circuit such that the number of non-constant inputs to the circuit is the next lower power of 64.

Let g be a gate in S_n which does not have an output of a gate as one of its inputs. Then g computes the function

$$g(\underline{a},\underline{b}) = \begin{cases} 1, & \text{if} \quad \sum_{i=0}^{n} u_i a_i + \sum_{i=0}^{n} v_i b_i \geq t \\ 0, & \text{else} \; . \end{cases}$$

First, we need all the weights for $\underline{a}$ to have same sign and all the weights for $\underline{b}$ to have same sign, where $sign(x) : \mathbb{R} \to \{-1,+1\}$ is $+1$ for all $x \in \mathbb{R}^+ \cup \{0\}$ and -1 otherwise. More formally, we want

$$\begin{aligned} sign(u_i) &= sign(u_j) \qquad \text{for all i,j} \\ sign(v_i) &= sign(v_j) \qquad \text{for all i,j} \; . \end{aligned}$$

This can be achieved by setting at most $3n/4$ variables in $\underline{a}$ and at most $3n/4$ variables in $\underline{b}$ to constant zero. By property 1, the circuit computes $P^{n/4}_{LR}$ on the remaining non-constant variables. We renumber the remaining $m = n/4$ variables in $\underline{a}$, the $n/4$ remaining variables in $\underline{b}$ (we preserve the order) and the corresponding weights. Let $\alpha_1 = \sum_{i=0}^{m/2-1} u_i$, $\alpha_2 = \sum_{i=m/2}^{m-1} u_i$, $\beta_1 = \sum_{i=0}^{m/2-1} v_i$ and $\beta_2 = \sum_{i=m/2}^{m-1} v_i$. We consider four cases:

case 1: $sign(u_i) = +1$, $sign(v_i) = -1$ for all $i = 0, \ldots, m-1$

case 1.1: $|\beta_1| > \alpha_2 - t$

We set $a_0 = \cdots = a_{m/2-1} = 0$ and $b_0 = \cdots = b_{m/2-1} = 1$. By property 2 of P_{LR}, the circuit computes $P_{LR}^{m/2}$ on the remaining non-constant variables. It follows that

$$-|\beta_1| + \sum_{i=m/2}^{m-1} |u_i| a_i - \sum_{i=m/2}^{m-1} |v_i| b_i < t - \alpha_2 + \alpha_2 - \sum_{i=m/2}^{m-1} |v_i| b_i < t \quad .$$

Hence $g(\underline{a}, \underline{b}) = 0$ for all possible values of $a_{m/2}, \ldots, a_{m-1}$ and $b_{m/2}, \ldots, b_{m-1}$.

case 1.2: $|\beta_1| \leq \alpha_2 - t$

We set $a_{m/2} = \cdots = a_{m-1} = 1$ and $b_{m/2} = \cdots = b_{m-1} = 0$. By property 3 of P_{LR}, the circuit computes $P_{LR}^{m/2}$ on the remaining non-constant variables. It follows that

$$\sum_{i=0}^{m/2-1} |u_i| a_i - \sum_{i=0}^{m/2-1} |v_i| b_i + \alpha_2 \geq \sum_{i=0}^{m/2-1} |u_i| a_i - |\beta_1| + |\beta_1| + t \geq t \quad .$$

Hence $g(\underline{a}, \underline{b}) = 1$ for all possible values of $a_0, \ldots, a_{m/2-1}$ and $b_0, \ldots, b_{m/2-1}$.

In case 1, there remain $2 \cdot m/2 = 2 \cdot n/8$ non-constant variables after the restriction.

case 2: $sign(u_i) = -1$, $sign(v_i) = +1$

We can treat this case in a similar manner as case 1.

case 3: $sign(u_i) = +1$, $sign(v_i) = +1$

case 3.1: $\beta_1 \geq t$

We set $a_0 = \cdots = a_{m/2-1} = 0$ and $b_0 = \cdots = b_{m/2-1} = 1$. By property 2 of P_{LR}, the circuit computes $P_{LR}^{m/2}$ on the remaining non-constant variables. Furthermore it follows that $g(\underline{a}, \underline{b}) = 1$ for all possible values of non-constant inputs.

case 3.2: $\alpha_2 \geq t$

We set $a_{m/2} = \cdots = a_{m-1} = 1$ and $b_{m/2} = \cdots = b_{m-1} = 0$. By property 3 of P_{LR}, the circuit computes $P_{LR}^{m/2}$ on the remaining non-constant variables. It follows that $g(\underline{a}, \underline{b}) = 1$ for all possible values of non-constant inputs.

After any of these restrictions, $2 \cdot n/8$ non-constant variables remain and the circuit computes $P_{LR}^{n/8}$. For the following restriction, we can assume $\beta_1 < t$ and $\alpha_2 < t$. In this case, the weights for the second half of the remaining inputs to g are small. So our aim will be to eliminate variables with large weights in g. Then, the sum of the remaining inputs to g will be too small to reach the threshold and the gate will output a constant for all possible values of non-constant inputs. In a first step, we set all inputs that contribute to α_1 and β_1 constant zero. The effect is that all weights of a's are small. We use another restriction to maintain small weights for non-constant b's. Then we set the inputs that have largest weights constant zero. We need to do this for at most 3/4 of the remaining variables to let the gate output zero for all possible values of non-constant inputs.

case 3.3 $\beta_1 < t$, $\alpha_2 < t$

We set $a_0 = \cdots = a_{m/2-1} = b_0 = \cdots = b_{m/2-1} = 0$. By property 1 of P_{LR}, the circuit computes $P_{LR}^{m/2}$ on the remaining non-constant variables. Let l=$m/2 = n/8$. Again, renumber the non-constant variables and corresponding weights of g, so that

$$g(\underline{a}, \underline{b}) = \begin{cases} 1, & \text{if} \quad \sum_{i=0}^{l-1} u_i a_i + \sum_{i=0}^{l-1} v_i b_i \geq t \\ 0, & \text{else .} \end{cases}$$

Let $\alpha_1' = \sum_{i=0}^{l/2-1} u_i$, $\alpha_2' = \sum_{i=l/2}^{l-1} u_i$, $\beta_1' = \sum_{i=0}^{l/2-1} v_i$ and $\beta_2' = \sum_{i=l/2}^{l-1} v_i$. Since $\alpha_1' + \alpha_2' = \alpha_2 < t$, we have $\alpha_1' < t$. If $\beta_1' \geq t$, case 3.1 applies and $\frac{l}{2} = n/16$ variables remain. Finally we consider weights such that $\alpha_1' < t$ and $\beta_1' < t$. In this case, we set $a_i = b_i = 0$ for $i = \frac{l}{2}, \ldots, l-1$ to eliminate the second half of the inputs (property 1 of P_{LR} applies). Then, by property 1 of P_{LR}, we set those $l/4$ remaining variables in $\underline{a}$ to zero that have maximal weights. We also eliminate those $l/8$ remaining variables in $\underline{b}$ with maximal weights. It follows that the overall sum of the remaining variables cannot reach t and $g(\underline{a}, \underline{b}) = 0$ for all possible values of non-constant inputs. There will remain at least $2\frac{l}{8} = 2\frac{n}{64}$ non-constant variables.

case 4: $sign(u_i) = -1$, $sign(v_i) = -1$

We can treat this case in a similar manner as case 3.

We have constructed a threshold circuit that has at least one gate less and computes $P_{LR}^{n/64}$.

We use this property of S_n to give an inductive proof of the lower bound. The inductive hypothesis is, that $size(S_n) \geq \lfloor \log_{64} n \rfloor$. Since we use the floor of $\log_{64} n$ in the bound, we can use induction on n for all n of the form $n = 64^m$ for some natural number m.

In the basis case, we have $n = 64$. Use property 1 to obtain a circuit that computes P^2_{LR}. Since, $P^2_{LR}(a_0, a_1, b_0, b_1) = a_0 \wedge b_1$, a circuit that computes P^{64}_{LR} consists of at least one threshold gate. Hence, the hypothesis holds for the induction basis. For the induction step, consider a threshold circuit S_n that computes P^n_{LR}. We show that, if the size of S_n is small, then we can construct a circuit$S_{n/64}$ with smaller size than possible. Suppose that $size(S_n) < \log_{64} n$. Construct a circuit $S_{n/64}$ that computes $P^{n/64}_{LR}$ by eliminating one gate in S_n. Then, $size(S_{n/64}) \leq size(S_n) - 1 < \log_{64} n - 1 = \log \frac{n}{64}$. This is a contradiction. Hence, $size(S_n) \geq \log_{64} n$.

■

6.3 1-Dimensional Pattern Recognition and Winner-Take-All

In analog VLSI the area occupied by a subcircuit that implements a winner-take-all gate is comparable to that for a threshold gate (see Chapter 3.3). Hence the next theorem demonstrates a drastic gain in efficiency if one employs modules for computing winner-take-all in addition to threshold gates. It combines the minimal possible computation time of 2 with a *linear* total wire length.

Theorem 6.3.1 *P^n_{LR} can be computed by a feedforward circuit of depth 2, consisting of two winner-take-all gates and one threshold gate, with total wire length and area $O(n)$.*

Proof: Denote with $\underline{a} = (a_0, \ldots, a_{n-1})$ and $\underline{b} = (b_0, \ldots, b_{n-1})$ the two vectors of input features. We construct a circuit that consists of two winner-take-all gates in the first layer and one threshold gate in the second layer. One winner-take-all gate marks the position of the leftmost occurring feature in $\underline{a}$ and the other winner-take-all gate marks the position of the rightmost occurring feature in $\underline{b}$. In the second layer, a single threshold gate with linear weights can compute the value of $P^n_{LR}(\underline{a}, \underline{b})$.

Let $\underline{a}' = (a'_0, \ldots, a'_{n-1}) = WTA(w_0 \cdot a_0, \ldots, w_{n-1} \cdot a_{n-1})$ denote the output vector of a winner-take-all gate with the inputs $a_0, \ldots, a_{n-1}$ weighted by integer weights $w_0, \ldots, w_{n-1}$. Set the weights of the winner-take-all gate such that:

$$\underline{a}' = WTA((n+1) \cdot a_0, n \cdot a_1, (n-1) \cdot a_2, \ldots, 2 \cdot a_{n-1}, 1).$$

If $\underline{a} = (0, \ldots, 0)$, a'_n wins (*i. e.* a'_n is the only non-zero output of the gate). Otherwise, a'_i wins if and only if $i = min\{j | a_j = 1\}$, for $0 \leq i \leq n-1$. Furthermore, set the weights of the second winner-take-all gate such that:

$$\underline{b}' = WTA(2 \cdot b_0, 3 \cdot b_1, 4 \cdot b_2, \ldots, (n+1) \cdot b_{n-1}, 1).$$

If $\underline{b} = (0, \ldots, 0)$, b'_n wins. Otherwise, b_i wins if and only if $i = max\{j | b_j = 1\}$, for $0 \leq i \leq n-1$. A simple threshold gate with $\underline{a'}$ and $\underline{b'}$ as its inputs can be used to compute the value of $P^n_{LR}(\underline{a}, \underline{b})$:

$$P^n_{LR} = \begin{cases} 1, & \text{if } \sum_{i=0}^{n-1} (b'_i \cdot (i+1) - a'_i \cdot (i+1)) - b'_n \cdot (n+1) - a'_n \cdot (n+1) \geq 1 \\ 0, & \text{otherwise .} \end{cases}$$

If there is no feature a present, a'_n wins and the gate outputs 0. The same holds for the case that no feature b is present. Otherwise, since there is exactly one a'_i and exactly one b'_j nonzero, if $a'_i = 1$ and $b'_j = 1$ and $i < j$, the weighted sum is above the threshold and the gate outputs 1. The sum is beyond the threshold for $i \geq j$ and the gate outputs 0.

Any gate can be implemented with linear wire length in our model. So the total wire length is $O(n)$. A similar VLSI layout uses linear area. ∎

In contrast to the threshold circuit of Theorem 6.2.1 just linear size integer weights are needed for this circuit. For implementations in VLSI or analog VLSI, this feature is a serious advantage, since weights of exponential precision are hard to implement.

6.4 Discussion

We have shown that the basic pattern recognition problem whether a certain local feature a occurs to the left of some other local feature b can be solved by circuits that require very little total wire length, and hence can potentially be implemented in analog VLSI. Furthermore it was shown that a circuit with $O(\log n)$ threshold gates can solve this problem, and that this number of threshold gates is asymptotically optimal. It was demonstrated that the same problem can be solved more efficiently if winner-take-all gates are employed in addition to a threshold gate. The resulting circuit is prefereable in several aspects. Only a constant number of gates is needed, which results in linear total wire length and area of the circuit. Furthermore, only linear weights are needed, which is a drastic gain compared to the exponential weights of the threshold circuit. This gives rise to the question which other concrete computational tasks can be carried out more efficiently by circuits that use winner-take-all gates besides (or instead of) threshold gates. The threshold circuit uses a total wire length of $O(n \log n)$ and no lower bound on the wire length of threshold circuits computing this problem could be stated. It would be interesting to find such a lower bound in our model or proof that an efficient threshold circuit with linear total wire length exists.

Chapter 7

Global Pattern Detection in 2-Dimensional Maps

While in the previous Chapter we merely dealt with one-dimensional pattern recognition problems, two-dimensional problems are of special interest because of two-dimensional input representations (maps) that can be found throughout the cerebral neocortex and because of the natural representation of many problem instances (*e. g.* visual input) in an two-dimensional array. In this chapter, the new complexity measure total wire length is applied to two basic computational problems that arise in two-dimensional translation- and scale-invariant pattern recognition, and hence appear to be useful as benchmark problems for sensory processing. We exhibit new circuit design strategies for these benchmark functions that can be implemented within realistic complexity bounds, in particular with linear or almost linear total wire length. This chapter is based on the publications (Legenstein and Maass, 2001a) and (Legenstein and Maass, 2001b).

7.1 Introduction

For many important sensory processing tasks – such as for visual or somatosensory input – the input variables are arranged in a 2-dimensional map whose structure reflects spatial relationship in the outside world. We assume that local feature

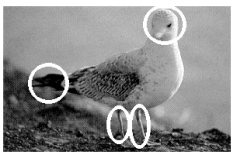


Figure 7.1: Examples of some local features (marked), whose spatial arrangement is essential for recognizing an object.

detectors are able to detect the presence of salient local features in their specific "receptive field", such as for example a center which emits higher (or lower) intensity than its immediate surrounding, or a high-intensity line segment in a certain direction, the end of a line, a junction of line segments, or even more complex local visual patterns like an eye or a nose. The ultimate computational goal is to detect specific *global spatial arrangements* of such local patterns (see Figure 7.1), such as the letter "T", or in the end also a human face, in a translation- and scale-invariant manner. We will use in the following the customary notation $O(\ldots)$, see Chapter 3.2.3. Whenever needed we assume for simplicity that n is such that $\sqrt{n}, \log n$ etc. are natural numbers[1]. The arrangement of the input variables on the grid will in general leave many nodes empty, which can be occupied by gates of the circuit.

7.2 Efficient Circuits for 2-Dimensional Pattern Recognition

We formalize such 2-dimensional global pattern detection problems by assuming that the input consists of arrays $\underline{a} = \langle a_1, \ldots, a_n \rangle, \underline{b} = \langle b_1, \ldots, b_n \rangle$, etc. of binary variables that are arranged on a 2-dimensional square grid. Each index i of an input variable can be thought of as representing a location within some corresponding square in the outside world. We assume that $a_i = 1$ if and only if feature a is detected at location i and that $b_i = 1$ if and only if feature b is detected at location i. In our formal model we reserve a sub-square within the 2-dimensional grid for each index i, where the input variables a_i, b_i, etc. are given on adjacent nodes of this grid. To make this more precise we assume that indexes i and j represent pairs $\langle i_1, i_2 \rangle, \langle j_1, j_2 \rangle$ of coordinates. Then "input location j is above and to the right of input location i" means: $i_1 < j_1$ and $i_2 < j_2$. The circuit complexity of variations of the function P_D^n where one or both of the "$<$" are replaced by "$\leq$" is the same. Since we assume that this spatial arrangement of input variables reflects spatial relations in the outside world, many salient examples for global pattern detection problems require the computation of functions such as

$$P_D^n(\underline{a}, \underline{b}) = \begin{cases} 1, & \text{if there exist } i \text{ and } j \text{ so that } a_i = b_j = 1 \text{ and input location } j \\ & \text{is above and to the right of input location } i \\ 0, & \text{else.} \end{cases}$$

This function is a generalization of P_{LR} – which was discussed in Chapter 6 – to two dimensions. We will show subsequently how the computation can be

[1] Throughout this Chapter, the logarithm is taken to the basis 2.

generalized to arbitrary dimensions with slightly worse complexity. Since, cortical maps map multi-dimensional input onto two dimensions, a rigorous analysis of the two-dimensional case is of interest. From the viewpoint of VLSI, this approach is also justified, since most available sensors (cameras, etc.) map onto at most 2-dimensional input space.

Theorem 7.2.1 *The function P_D^n can be computed – and witnesses i and j with $a_i = b_j = 1$ can be exhibited if they exist – by a circuit with total wire length $O(n)$, consisting of $O(n)$ Boolean gates of fan-in 2 (and fan-out 2) in depth $O(\log n \cdot \log\log n)$.*

The depth of the circuit can be reduced to $O(\log n)$ if one employs threshold gates with fan-in $\log n$. This can also be done with total wire length $O(n)$.

In the VLSI-model, this circuit uses $O(n)$ area.

Proof: This circuit design is based on a divide-and-conquer approach. On first sight it appears that such an approach is bound to fail for computing P_D^n, since there may exist for example just a single pair of witnesses i and j with the desired properties, but the chosen subdivision of the input area happens to assign i and j to *different* components of the subdivision. Hence the evaluation of P_D for each of the components is of little help for the evaluation of P_D^n for the full input area.

In order to make the divide-and-conquer approach feasible it is essential that one computes for each component of the subdivision more than just whether P_D holds for this component. If one divides iteratively each square into 4 sub-squares C_1, C_2, C_3, C_4, (see Figure 7.2) then it suffices to compute for each sub-square C_k the following data:

$$
\begin{aligned}
left(C_k) &:= \text{the x-coordinate of the leftmost location } i \text{ in } C_k \text{ with } a_i = 1 \\
right(C_k) &:= \text{the x-coordinate of the rightmost location } j \text{ in } C_k \text{ with } b_j = 1 \\
down(C_k) &:= \text{the y-coordinate of the lowest location } i \text{ in } C_k \text{ with } a_i = 1 \\
up(C_k) &:= \text{the y-coordinate of the highest location } j \text{ in } C_k \text{ with } b_j = 1 \\
found(C_k) &:= \begin{cases} 1, & \text{if } \ P_D \text{ applied to } C_k \text{ outputs } 1 \\ 0, & \text{else .} \end{cases}
\end{aligned}
$$

We assume that each of the first four functions assumes the value 0 on C_k if and only if there exists no location i or j in C_k with the desired property. Thus all coordinates are assumed to be numbers ≥ 1.

The essential property of these 5 functions is that $left(C)$, $right(C)$, $down(C)$, $up(C)$ and $found(C)$ can be computed from the values of these 5 functions for the 4 sub-squares C_1, C_2, C_3, C_4. This is obvious for $left(C)$, $right(C)$, $down(C)$, $up(C)$, requiring just comparisons of pairs of $(b+1)$-bit natural numbers if each C_k is responsible for a sub-square of the input-array of size $2^b \times 2^b$. The value of

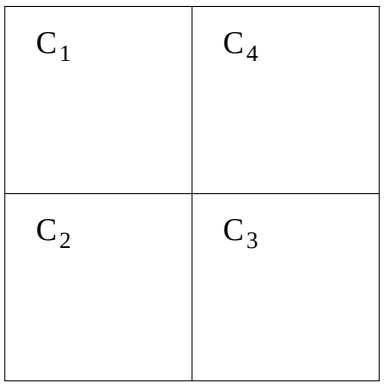


Figure 7.2: The input area C is divided into four sub-squares C_k, which are numbered in a counterclockwise fashion.

$found(C)$ can be computed in the following fashion, assuming that the components C_k that make up C are numbered in a counterclockwise fashion, starting with C_1 in the upper left quadrangle (see Figure 7.2):

$$\begin{aligned} found(C) = 1 \quad \Leftrightarrow \quad & \bigvee_{k=1}^{4} found(C_k) = 1 \vee \\ & 0 < down(C_1) < up(C_4) \vee \\ & 0 < down(C_2) < up(C_3) \vee \\ & (0 < down(C_2) \wedge 0 < up(C_4)) \vee \\ & 0 < left(C_2) < right(C_1) \vee \\ & 0 < left(C_3) < right(C_4) \quad . \end{aligned}$$

Obviously this algorithm makes use of the fact that the area is not subdivided in an *arbitrary* fashion into components, but in a way which is consistent with the map, *i.e.* with the spatial relationship of locations in the outside world. Or, with a variation of a well-known design philosophy of Carver Mead, one could say that *space represents itself* in this algorithm design.

The layout of a circuit for P_D^n with small total wire length is based on a variation of the well-known H-tree (see, e.g. (Savage, 1998)), which we will call an *extended H-tree*. An H-tree makes optimal use of area and wire length if the n inputs are allowed to be arranged as an $\sqrt{n} \times \sqrt{n}$ array on the plane. Figure 7.3a shows the H-tree H_1 with 4 darkly shaded leaves (inputs) and lightly shaded inner nodes of the binary tree. H_{k+1} can be constructed by replacing the leaves of H_k with H-trees H_1. Since H_1 is a tree with four leaves, H_k has 4^k leaves. In Figure 7.3b, each leaf of H_1 was replaced by an H-tree H_1.

The depth of a node v in an H-tree is the length of the shortest path from v to a leaf. Note that a recursive step in the construction of an H-tree adds depth 2

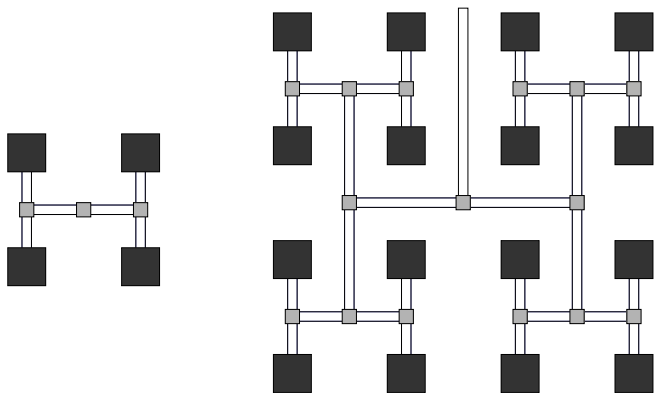


Figure 7.3: The H-tree layout. Dark rectangles are leaves, light rectangles are inner nodes. a) H_1 is a tree layout for 4 leaves. b) H_2. H_{k+1} is constructed recursively by replacing the leaves of H_k with H-trees H_1 . (Figure taken from (Savage, 1998))

to the graph. Hence, it will be more convenient to talk about *levels* rather than depth, where a node v is on level i if v is in depth $2i-1$ or in depth $2i$. So, the nodes in depth 1 and 2 are on level 1 (these are the nodes of the last recursive step in the construction of the H-tree), and the root of an H-tree H_k is on level k. Our layout will differ from the H-tree layout in a crucial point. Internal nodes of the H-tree are replaced by groups of several gates, and the connections between these groups consist of "buses" rather than of single wires. More precisely, each "node" on level i of an H-tree is a circuit with $O(i)$ gates and $O(i^2)$ total wire length and area with side length $O(i)$. Instead of a single edge in an H-tree one has a "bus" consisting of $O(i)$ wires if the bus connects a node on level i with a node on level i or $i+1$.

One has to be careful in talking about levels and nodes in an extended H-tree, since the circuit in a "node" might consist of several gates and might have even non constant depth. However each extended H-tree has an underlying H-tree and the levels are counted with regard to this underlying H-tree.

We now show how the extended H-tree can be used as a layout strategy for a circuit that implements the previously developed algorithm for solving P_D^n. The extended H-tree layout implements the structure of the algorithm by recursively dividing the input-area into four axis-parallel sub-squares. The computations needed in a node on level i of the H-tree can be carried out by a circuit of size $O(i)$ and $O(i^2)$ total wire length and area, which is placed at that node. The depth of a circuit at a node is $O(1)$ if threshold gates of fan-in $O(\log n)$ are used and $O(\log i)$ if Boolean gates of fan-in 2 are used. Lemma 8.1.1 in Section 8.1 shows that the extended H-tree stays within the claimed complexity bounds. The

depth of an H-tree is $O(\log n)$. Thus if the circuits at the nodes have depth $O(1)$, the extended H-tree has depth $O(\log n)$. If the circuits at the nodes have depth $O(\log i)$, the depth of the extended H-tree is $O(\log n \cdot \log\log n)$.

An extension to the circuit that reports a pair of witnesses is straight forward. ■

The linear total wire length of this circuit is up to a constant factor *optimal* for any circuit whose output depends on all of its n inputs. Note that most connections in this circuit are local, just like in a biological neural circuit. Thus, we see that minimizing total wire length tends to generate biology-like circuit structures.

However, the tree-like circuit structure results in considerable circuit-depth for large input-size. In biological neural systems, neural gates of large fan-in are used to implement shallow circuits, whereas the circuit design above is based on gates of fan-in 2 or $log(n)$ which is comparatively small. The next theorem shows that one can compute P_D^n faster (i.e. by a circuit with smaller depth) if one can afford a somewhat larger total wire length. This circuit construction, that is based on AND/OR gates of limited fan-in Δ, has the additional advantage that it can not just exhibit *some* pair $\langle i, j\rangle$ as witness for $P_D^n(\underline{a}, \underline{b}) = 1$ (provided such witness exists), but it can exhibit in addition *all* j that can be used as witness together with some i. This property allows us to "chain" the global pattern detection problem formalized through the function P_D^n, and to decide within the same complexity bound whether for any fixed number k of input vectors $\underline{a}^{(1)}, \ldots, \underline{a}^{(k)}$ from $\{0,1\}^n$ there exist locations $i^{(1)}, \ldots, i^{(k)}$ so that $a^{(m)}_{i^{(m)}} = 1$ for $m = 1, \ldots, k$ and location $i^{(m+1)}$ lies to the right and above location $i^{(m)}$ for $m = 1, \ldots, k-1$. In fact, one can also compute a k-tuple of witnesses $i^{(1)}, \ldots, i^{(k)}$ within the same complexity bounds, provided it exists. This circuit design is based on an efficient layout for prefix computations.

Theorem 7.2.2 *For any given n and $\Delta \in \{2, \ldots, \sqrt{n}\}$ one can compute the function P_D^n in depth $O(\frac{\log n}{\log \Delta})$ by a feedforward circuit consisting of $O(n)$ AND/OR gates of fan-in $\leq \Delta$, with total wire length $O(n \cdot \Delta \cdot \frac{\log n}{\log \Delta})$.*
In the VLSI-model, the circuit uses $O(n \cdot (\Delta \cdot \frac{\log n}{\log \Delta})^2)$ area.

Proof: The main idea in the construction of the circuit is illustrated in Figure 7.4. In Figure 7.4a, a two dimensional input-assignment for P_D^n is shown. Crosses mark locations where a feature a is present and open circles mark locations where a feature b occurs. Every feature b that is located in the shaded region in Figure 7.4b is located to the right and above of some present feature a. Hence, if there is some location j that is in the shaded region of Figure 7.4b and $b_j = 1$, then the value of $P_D^n(\underline{a}, \underline{b})$ is 1. We introduce indicator variables a'_j (j=1, ..., n), where $a'_j = 1$ if the location j is to the right and above to some location i with $a_i = 1$, and $a'_j = 0$ otherwise. (in Figure 3b, $a'_j = 1$, if j is a location in the shaded

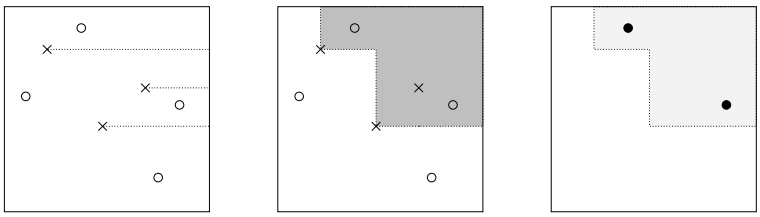


Figure 7.4: Computing P_D with prefix circuits. Crosses mark locations where a feature a occurs, open circles mark locations where a features b is present. a) All locations that are in the same row and to the right of some feature $a = 1$ are marked as dotted lines. b) All locations that are to the right and above of some feature $a = 1$ are shaded. c) All locations i with $b_i = 1$ in this area are marked with filled circles.

region). It follows that P_D^n has value 1 if there exists some location j such that $a'_j \wedge b_j = 1$.

Hence, the problem is reduced to the problem of computing the values of a'_j for all locations $j = 1, \ldots, n$. A straight-forward implementation would lead either to large depth or to large total wire length. In a one dimensional scenario, the problem would be equivalent to the following one. Suppose one has a one dimensional array of pixels $x_1, \ldots, x_n$. Then the equivalent problem to computing a'_j would be to compute the values of $x'_1, \ldots, x'_n$ where $x'_j = 1$ if and only if there is a $x_i = 1$ that is to the left of x_j. This is the problem of computing the *prefixes*: $x'_1 = x_1, x'_2 = x_1 \vee x_2, x'_3 = x_1 \vee x_2 \vee x_3, \ldots, x'_n = x_1 \vee x_2 \vee x_3 \vee \cdots \vee x_n$. Such a computation is called a *prefix computation*. There exist efficient circuits for such computations (see *e.g.* (Savage, 1998)). In the 2-dimensional case, we just need to apply these computations on all rows and columns. By applying the prefix computation on rows of $\underline{a}$, one can determine the locations in the input plane that are in the same row as some feature $a_i = 1$ and located to the right of a_i. This is illustrated in Figure 7.4a. Here, the horizontal lines in the input space represent locations where indicator variables have value 1 after that step. Let us call the outputs of the horizontal prefix circuits $\hat{a}_j$, where $j = 1, \ldots, n$ denotes locations in the same manner as the inputs are indexed. Then, a location j is in the right spatial relation to some feature $a_i = 1$ at location i, if it is above of some location k with $\hat{a}_k = 1$. Hence, we can successively apply the same prefix-operation on columns of these intermediate variables $\hat{a}_1, \ldots, \hat{a}_n$ to compute the correct value of all indicator variables (see Figure 7.4b). Now, $b'_i = a'_i \wedge b_i$ has value 1 if location i is in the right spatial relation with some present feature a and $b_i = 1$. (This is not exactly what we want, since this would also mark b-features that lie in the same row or column with some a-feature. However, we can also AND the b-feature with the marking-bit that is one pixel to the left and below it.) In Figure 7.4c, the locations l with $b'_l = 1$ are marked with filled circles.

Finally, an OR over all b'_i's outputs $P^n_D(\underline{a}, \underline{b})$ for all inputs $\underline{a}, \underline{b} \in \{0,1\}^n$.

Let $C(PREF^n)$, $depth(PREF^n)$ and $TWL(PREF^n)$ denote the size, depth and total wire length of a prefix circuit with n inputs. The circuit consists of prefix computations for every row and every column of features a ($2\sqrt{n}$ many), each consisting of OR gates only. Furthermore, n AND gates are used. Finally, there is one OR with inputs $b'_1, \ldots, b'_n$. This OR could be implemented also by a circuit of OR gates with smaller fan-in in order to reduce the total wire length. Hence, the circuit has size $2\sqrt{n}C(PREF^{\sqrt{n}}) + n + 1$ and depth $2depth(PREF^{\sqrt{n}}) + 2$.

In the following, we give upper bounds on total wire length and area for this circuit. Lemma 8.2.1 gives upper bounds on total wire length and area for an efficient prefix circuit consisting of gates with maximal fan-in Δ ($\Delta \in \{2, \ldots, \sqrt{n}\}$). There is a prefix computation of $\sqrt{n}$ inputs for each row of $\underline{a}$ in the input plane. We can place this prefix circuits in between the rows of inputs. Note that if these circuits would need too many rows, we had to place the input rows far away from each other which would influence the total wire length of the subsequent prefix circuits. But, since the prefix circuits use a constant number of rows in our model , the computations for rows and columns do not affect each other and the wire length used for this part of the computation is $O(\sqrt{n}\sqrt{n}\Delta\frac{\log n}{\log \Delta}) = O(n\Delta\frac{\log n}{\log \Delta})$. The AND gates that compute $b'_i = a'_i \wedge b_i$ need $O(n)$ total wire length all together. We implement the OR of $b'_1, \ldots, b'_n$ as a 2 dimensional tree of fan-in Δ. This influences the size and the depth of the circuit only by a constant factor. It can be shown that a 2 dimensional tree of fan-in Δ has total wire length $O(n\sqrt{\Delta})$. Hence, the circuit has $TWL = O\left(n \cdot \Delta \cdot \frac{\log n}{\log \Delta}\right)$, $depth = O\left(\frac{\log n}{\log \Delta}\right)$, and $size = O(n)$.

The situation is different in the VLSI-model. The crucial part of the layout are the prefix circuits. In the VLSI-model, these circuits have side-lengths $O(\Delta\frac{\log n}{\log \Delta})$ and $O(\sqrt{n})$ each (see proof of Lemma 8.2.1). Nevertheless, we layout these circuits in the same manner as above. Since we need one prefix circuit for every row and every column, the side length of the layout for the prefix circuits is $O(\sqrt{n}\Delta\frac{\log n}{\log \Delta})$. Hence, the circuit for P^n_D can be implemented within an area of $O\left(n \cdot \left(\Delta \cdot \frac{\log n}{\log \Delta}\right)^2\right)$. ∎

An advantage of this approach is that we computed *all* the witnesses in $\underline{b}$ for P_D. Hence we can use this information to compare these witnesses with some features $\underline{c}$. In other words, we can compute if there is some feature a beneath and to the left of some feature b which is beneath an to the left of some feature c and so on. Denote this function with $P^{n,k}_D$ for some $k \geq 2$. We give a formal definition of $P^{n,k}_D$. Consider $k \geq 2$ different feature types $\underline{a}^{(1)}, \ldots, \underline{a}^{(k)}$. We recursively define a function $W^{n,k} : \{0,1\}^{k \cdot n} \to \{0,1\}^n$ that outputs witnesses for $P^{n,k}_D$:

$$W^{n,2}(\underline{a}, \underline{b}) = (w_1, \ldots, w_n) \quad \text{, where}$$

$$w_j = \begin{cases} 1, & \text{if } b_j = 1 \text{ and there exist } i \text{ so that } a_i = 1 \text{ and input location } j \\ & \text{is above and to the right of input location } i \\ 0, & \text{else .} \end{cases}$$

$$W^{n,k}(\underline{a}^{(1)}, \ldots, \underline{a}^{(k)}) = W^{n,2}(W^{n,k-1}(\underline{a}^{(1)}, \ldots, \underline{a}^{(k-1)}), \underline{a}^{(k)}) \quad .$$

Recall that we computed $w_1, \ldots, w_n$ in the circuit for P_D^n and called these values $b'_1, \ldots, b'_n$ in the proof of Theorem 7.2.2. Hence, by the recursive definition of $W^{n,k}$, one just has to apply this circuit $k-1$ times to compute $W^{n,k}$. Now we can define $P_D^{n,2}(\underline{a}^{(1)}, \underline{a}^{(2)}) = P_D^n(\underline{a}^{(1)}, \underline{a}^{(2)})$ and $P_D^{n,k}$ for $k \geq 3$:

$$P_D^{n,k} = P_D^n(W^{n,k-1}(\underline{a}^{(1)}, \ldots, \underline{a}^{(k-1)}), \underline{a}^{(k)}) \quad .$$

Given this definition of $P_D^{n,k}$, Corollary 7.2.3 holds:

Corollary 7.2.3 *For any given n, $k \geq 2$ and $\Delta \in \{2, \ldots, \sqrt{n}\}$ one can compute the function $P_D^{n,k}$ in depth $O(k\frac{\log n}{\log \Delta})$ by a feedforward circuit consisting of $O(k \cdot n)$ AND/OR gates of fan-in $\leq \Delta$, with total wire length $O(k \cdot n \cdot \Delta \cdot \frac{\log n}{\log \Delta})$ and area $O(n \cdot (k \cdot \Delta \cdot \frac{\log n}{\log \Delta})^2)$.*

Another essential ingredient of translation- and scale-invariant global pattern recognition is the capability to detect whether a local feature c occurs in the middle between locations i and j where the local features a and b occur. This global pattern detection problem is formalized through the following function $P_I^n : \{0,1\}^{3n} \to \{0,1\}$:

If $\sum \underline{a} = \sum \underline{b} = 1$ then $P_I^n(\underline{a}, \underline{b}, \underline{c}) = 1$, if and only if there exist i, j, k so that input location k lies on the middle of the line between locations i and j, and $a_i = b_j = c_k = 1$.

This function P_I^n can be computed very fast by circuits with the least possible total wire length (up to a constant factor), using threshold gates of fan-in up to $\sqrt{n}$:

Theorem 7.2.4 *The function P_I^n can be computed – and witnesses can be exhibited – by a circuit with total wire length and area $O(n)$, consisting of $O(n)$ Boolean gates of fan-in 2 and $O(\sqrt{n})$ threshold gates of fan-in $\sqrt{n}$ in depth 7.*

Proof: We construct a circuit that projects the inputs $\underline{a}$ and $\underline{b}$ onto the horizontal and vertical axis of the input plane and computes the midpoints of these 1-dimensional projections (see Figure 7.5). This approach is not obvious since one can easily construct an example where there are two c-features that lie in the middle of the projections, but none of them lies in the middle of the occurring features in the 2-dimensional spatial constellation. One can handle this problem

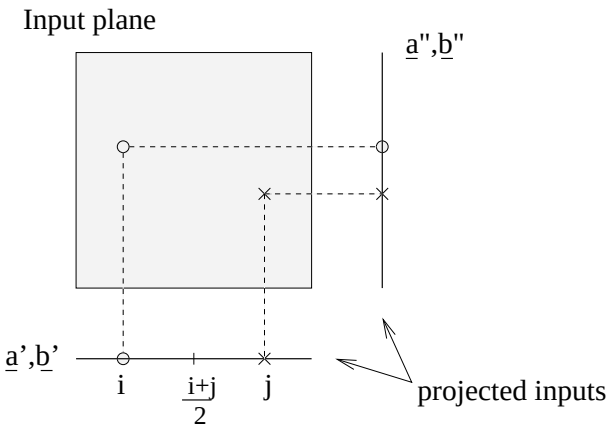


Figure 7.5: The projection of the input plane onto its horizontal and vertical axis. A circle represents an occurring feature a and a cross an occurring feature b. In the horizontal projection, features occur at locations i and j. We compute the midpoints of the projections and trace them back onto the input plane.

by tracing back the computed 1-dimensional midpoints to the 2-dimensional input plane and looking for a pixel where there is a horizontal as well as a vertical midpoint and a feature c present.

A more formal description of the circuit follows. To project the inputs onto one dimension we compute

$$a'_i = \bigvee_{\text{j is in the i-th column}} a_j \qquad b'_i = \bigvee_{\text{j is in the i-th column}} b_j \tag{7.1}$$

$$a''_i = \bigvee_{\text{j is in the i-th row}} a_j \qquad b''_i = \bigvee_{\text{j is in the i-th row}} b_j \quad . \tag{7.2}$$

These values are computed with a circuit of depth 1 consisting of $O(\sqrt{n})$ threshold gates of fan-in $\sqrt{n}$. The total wire length needed is $O(n)$.

Then we compute the vertical and horizontal midpoints of these projected inputs. For the horizontal midpoint, we define variables h_i $(1 \leq i \leq \sqrt{n})$, where the value of h_m is 1 if and only if m is in the middle of some i, j with $a'_i = b'_j = 1$. For the vertical midpoint, we define variables v_i $(1 \leq i \leq \sqrt{n})$ in a similar manner, where the value of v_m is 1 if and only if m is in the middle of some i, j with $a''_i = b''_j = 1$. Since the midpoint of i, j is $\frac{i+j}{2}$, we compute h_m by comparing

$\frac{i+j}{2}$ with m, where i is a location with $a'_i = 1$ and j is a location with $b_j = 1$:

$$g_m = \begin{cases} 1, & \text{if} \quad \sum_{i=\max\{1,2m-n\}}^{\min\{n,2m\}} i.a'_i + i.b'_i \geq 2m \\ 0, & \text{else .} \end{cases} \tag{7.3}$$

$$g'_m = \begin{cases} 1, & \text{if} \quad \sum_{i=\max\{1,2m-n\}}^{\min\{n,2m\}} i.a'_i + i.b'_i \leq 2m \\ 0, & \text{else .} \end{cases} \tag{7.4}$$

$$h_m = g_m \wedge g'_m \quad . \tag{7.5}$$

$h_1, \ldots, h_{\sqrt{n}}$ can be computed by a circuit of depth 2, consisting of $O(\sqrt{n})$ threshold gates of fan-in $\leq \sqrt{n}$ and $O(\sqrt{n})$ AND gates of fan-in 2. The total wire length needed is $O(n)$. Note that one can define a region where the c-feature may lie by changing the thresholds in Eqs. (7.3) and (7.4).

Finally, we need to trace back those values and compute the witnesses w_i. If i is a location in the x-th column and y-th row, then $w_i = h_x \wedge v_y \wedge c_i$. This can be implemented with $O(n)$ AND gates of fan-in 2, depth 2 and total wire length $O(n)$. It follows that $P_I^n(\underline{a}, \underline{b}, \underline{c}) = \bigvee_{i=1}^n w_i$ and this OR can be computed with $\sqrt{n}+1$ threshold-gates of fan-in $\sqrt{n}$, linear total wire length and depth 2. Hence, the total wire length of this circuit is bounded by $O(n)$ and the circuit has depth 7.

In the VLSI-model, we will need to model the threshold gates that project the inputs onto one dimension (Eqs. (7.1) and (7.2)) as $\sqrt{n}$ rectilinear parts of one input each on a common wire in order to be able to project onto the horizontal and vertical axis. Then the area needed is $O(n)$. To compute the midpoint in one dimension (see Eqs. (7.3) to (7.5)), we can use $2\sqrt{n}$ threshold gates of $\sqrt{n}$ area each and $\sqrt{n}$ AND gates of constant area. So the area needed to compute the midpoints is $O(n)$ and the final tracing back and witness-computation can be done with wires that need $O(n)$ area and $O(n)$ gates of constant area each. ∎

7.3 Discussion

In this chapter we have analyzed the total wire length required for solving two concrete computational problems that are inherent in many global pattern recognition tasks. It turns out that both of these problems can be solved by circuits whose total wire length is linear or almost linear. Furthermore these examples demonstrate that the design of circuits with small total wire length yields circuit architectures that differ significantly from those that arise if just the traditional circuit complexity measures (number of gates, depth) are minimized. We expect that in general the construction of circuits with small total wire length produces

circuit architectures that are less unrealistic from the point of physical implementation. In particular this strategy may help to "guess" circuit design strategies that are implemented in biological neural systems.

The input to our circuits can be interpreted as a computational map. Such maps occur in cortical sensory ares. Many authors pointed out the importance of cortical maps for efficient wiring (see Section 4.3. However no attempt was made so far to analyze the influence of cortical maps on specific computational problems. The design of our circuits suggest that the topological input organization is important for circuits of small total wire length. Note that the function P_D^n is not local. This might be an indicator that topographic representation is not only important for local computations, but also for global ones with some spatial structure. However, no lower bound on the total wire length for one of the functions computed on non-topographical representation could be given. Such a lower bound would be desirable to make further claims.

The complexity measure total wire length is somehow reminiscent to the complexity measure area in abstract VLSI-designs. However in contrast to VLSI-design, which are necessarily much more detailed, it is in general much easier to estimate the total wire length of a circuit architecture in the model that we have proposed in this chapter. Hence the new circuit complexity measure total wire length may represent a useful compromise between practical relevance and mathematical simplicity.

Chapter 8

The Total Wire Length of Basic Computational Structures

In the design of circuits, basic computational structures emerge that are useful in many concrete computations. Such structure are *e. g.* computations on trees (see Chapter 10 for an overview and references) or *prefix cicuits*. We already made use of a tree structure in Chapter 7. The *H-tree* (Mead and Rem, 1979) is an efficient layout for computations on trees on the two-dimensional plane that uses only linear area in the number of leaves. However, in Chapter 7 we need a somewhat more general structure. This structure is analyzed in Section 8.1. A prefix computation is also a general computational strategy that can be used in a variety of problems with some inherit structure. Prefix computations were first used by Ofman (Ofman, 1962) for an adder-circuit. We give an efficient layout for prefix circuits with gates of bounded fan-in Δ in Section 8.2. Both structures are analyzed in term of their total wire length and VLSI-area. Further issues of tree-layout are discussed in Chapter 10.

8.1 Extended H-tree Layout

An H-tree makes optimal use of area and wire length if the n inputs are allowed to be arranged as an $\sqrt{n} \times \sqrt{n}$ array on the plane. Figure 8.1a shows the H-tree H_1 with 4 darkly shaded leaves (inputs) and lightly shaded inner nodes of the binary tree. H_{k+1} can be constructed by replacing the leaves of H_k with H-trees H_1. Since H_1 is a tree with four leaves, H_k has 4^k leaves. In Figure 8.1b, each leaf of H_1 was replaced by an H-tree H_1.

The depth of a node v in an H-tree is the length of the shortest path from v to a leaf. Note that a recursive step in the construction of an H-tree adds depth 2 to the graph. Hence, it will be more convenient to talk about *levels* rather than depth, where a node v is on level i if v is in depth $2i-1$ or in depth $2i$. So, the

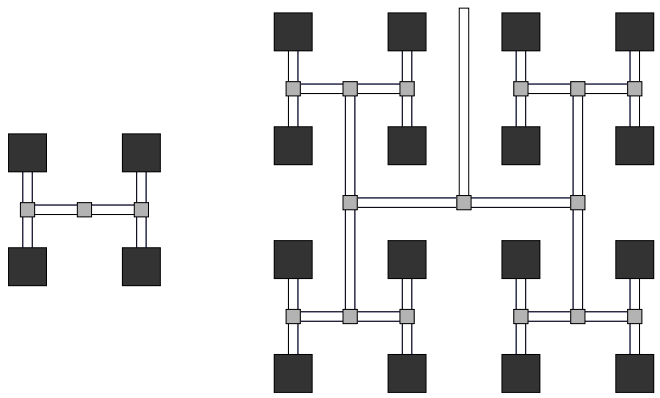


Figure 8.1: The H-tree layout. Dark rectangles are leaves, light rectangles are inner nodes. a) H_1 is a tree layout for 4 leaves. b) H_2. H_{k+1} is constructed recursively by replacing the leaves of H_k with H-trees H_1 . (Figure taken from (Savage, 1998))

nodes in depth 1 and 2 are on level 1 (these are the nodes of the last recursive step in the construction of the H-tree), and the root of an H-tree H_k is on level k. Our layout will differ from the H-tree layout in a crucial point. Internal nodes of the H-tree are replaced by groups of several gates, and the connections between these groups consist of "busses" rather than of single wires. More precisely, each "node" on level i of an H-tree is a circuit with $O(i)$ gates and $O(i^2)$ total wire length and area with side length $O(i)$. Instead of a single edge in an H-tree one has a "bus" consisting of $O(i)$ wires if the bus connects a node on level i with a node on level i or $i+1$.

This layout extends the capabilities of the H-tree since it allows a node with m inputs in its subtree to transfer $O(\log m)$ bits of information to its successor node. This is why we call this layout an *extended H-tree*. One has to be careful in talking about levels and nodes in an extended H-tree, since the circuit in a "node" might consist of several gates and might have even non constant depth. However each extended H-tree has an underlying H-tree and the levels are counted with regard to this underlying H-tree. The following lemma states that the extended H-tree of n leaves can also be embedded with linear total wire length and area.

Lemma 8.1.1 *The extended H-tree layout on n leaves can be implemented with $O(n)$ gates and total wire length.*
In the VLSI-model, the layout uses $O(n)$ area.

Proof: We will not only derive asymptotic bounds, but also pay attention to the size of constant factors. To achieve this, we will use the recursive construction of the extended H-tree to derive recursive formulas on size, side-length and total

wire length of the layout. The nodes on level 1 play a special role in the circuit. There are $\frac{n}{4}$ extended H-trees H_1 on level 1 that compute, in parallel, the basic values for the subsequent "conquer steps". Let $S(H_1)$, $C(H_1)$ and $TWL(H_1)$ denote the side-length, size and total wire length of one such H_1-circuit.

We start the proof by deriving an upper bound on the side-length $S(H_k)$ of the extended H-tree H_k. We assume that the side-length of a node on level i is bounded by ci for some suitable constant c. The side-length of H_k is the sum of the side-lengths of two H-trees H_{k-1} and the side length of a node on level k (see Figure 8.2). Hence, the following recurrence holds:

$$S(H_k) = 2S(H_{k-1}) + ck \quad . \tag{8.1}$$

The solution of Eq. (8.1) yields the bound $S(H_k) \leq 2^{k-1}S(H_1) + \frac{3c}{2}2^k$. Since $n = 4^k$, we have $S(H_k) \leq \sqrt{n}(\frac{S(H_1)}{2} + \frac{3c}{2}) = O(\sqrt{n})$. The area of the layout is

$$area(H_k) = S^2(H_k) \leq \left(\frac{S(H_1)}{2} + \frac{3c}{2}\right)^2 n = O(n) \quad .$$

A similar recurrence holds for the number of gates $C(H_k)$ in the circuit for the H-tree H_k: Let the number of gates at a node on level i of the extended H-tree be bounded by si for a suitable constant s (recall that a recursive step in the H-tree layout adds 3 inner nodes). We get a recursive formula which we iterate $k-1$ times:

$$\begin{aligned} C(H_k) &\leq 4C(H_{k-1}) + 3 \cdot s \cdot k \\ &\leq 4^{k-1}C(H_1) + 3 \cdot s \sum_{j=0}^{k-2} 4^j(k-j) \quad . \end{aligned} \tag{8.2}$$

Since $\sum_{j=0}^{k-2} 4^j(k-j) \leq \frac{77}{36}4^k$ the solution of Eq. (8.2) is

$$C(H_k) \leq n(\frac{1}{4}C(H_1) + \frac{77}{12}s) \quad . \tag{8.3}$$

Now we use a similar argument to estimate the total wire length. The total wire length of the layout consists of the wire lengths at the inner nodes and the wire lengths of the "busses". Let d be a constant such that the total wire length of a node on level i is bounded by $d \cdot i^2$. Also, let the number of wires of a "bus" from a node on level i to a node on level i or $i+1$ be bounded by $e \cdot i$. The basis for the recursive calculation of the total wire length for an extended H-tree H_i is

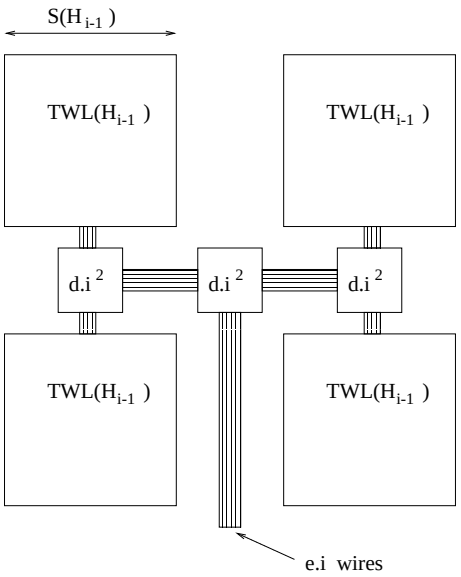


Figure 8.2: The H-tree H_i has wires from four H-trees H_{i-1}, the wires of three inner nodes, and the wires of the busses. An inner node has a total wire length of $d \cdot i^2$, and a bus consists of $e \cdot i$ wires.

illustrated in Figure 8.2. We get a recursive formula which we iterate $k-1$ times:

$$\begin{aligned} TWL(H_k) &\leq 4TWL(H_{k-1}) + 2k \cdot e \cdot S(H_{k-1}) + 3d \cdot k^2 \qquad (8.4) \\ &\leq 4TWL(H_{k-1}) + k \cdot e(S(H_1) + 3c)2^{k-1} + 3d \cdot k^2 \\ TWL(H_k) &\leq 4^{k-1}TWL(H_1) + 2^{k-1}e(S(H_1) + 3c)\sum_{j=0}^{k-2} 2^j(k-j) \\ &\quad +3d\sum_{j=0}^{k-2} 4^j(k-j)^2 \quad . \qquad (8.5) \end{aligned}$$

Since $\sum_{j=0}^{k-2} 2^j(k-j) \leq \frac{3}{2}2^k$ and $\sum_{j=0}^{k-2} 4^j(k-j)^2 \leq \frac{77}{108}4^k$ we get:

$$\begin{aligned} TWL(H_k) &\leq 4^{k-1}TWL(H_1) + \frac{3}{2}2^{2k-1}e(S(H_1) + 3c) + \frac{77}{36}d4^k \\ &\leq \frac{1}{4}n \cdot TWL(H_1) + \frac{3}{4}n \cdot e(S(H_1) + 3c) + \frac{77}{36}d \cdot n \\ TWL(H_k) &\leq n(\frac{1}{4}TWL(H_1) + \frac{3}{4}e(S(H_1) + 3c) + \frac{77}{36}d) = O(n) \quad . \qquad (8.6) \end{aligned}$$

■

8.2 The Layout of Prefix Circuits with Limited Fan-in

The proof of Theorem 7.2.2 relied on parallel prefix circuits. We show how a parallel prefix circuit can be implemented in our models with small total wire length and area. Consider a set X of elements with an associative binary operation. We denote the binary operation by juxtaposition of the elements in X. Suppose we have functional gates such that each gate with inputs $x_1, \ldots, x_k$ computes the product $x_1 x_2 \ldots x_k$, for some fan-in k. Lemma 8.2.1 gives upper bounds on a circuit of such gates with maximal fan-in Δ that computes the prefixes $x_1, x_1x_2, \ldots, x_1x_2 \ldots x_n$. For simplicity, we assume that n is a power of Δ.

Lemma 8.2.1 *If n inputs $x_1, \ldots, x_n$ are arranged on a row of a grid, then the prefixes $x_1, x_1x_2, \ldots, x_1x_2 \ldots x_n$ can be computed by a circuit with maximum fan-in $\Delta \in \{2, \ldots, n\}$, size $\leq 2n$ in depth $= 2\frac{\log n}{\log \Delta}$. In our model the circuit uses only a constant number of rows and the total wire length is $O(\frac{\log n}{\log \Delta} n\Delta)$. In the VLSI-model the circuit uses an area $\leq n\Delta\frac{\log n}{\log \Delta}$.*

Proof: We divide the inputs $x_1, \ldots, x_n$ into $\frac{n}{\Delta}$ consecutive subintervals and rename the inputs to $x_{1,1}, \ldots, x_{1,\Delta}, x_{2,1}, \ldots, x_{2,\Delta}, \ldots, x_{\frac{n}{\Delta},1}, \ldots, x_{\frac{n}{\Delta},\Delta}$. We denote the outputs of the circuit as $y_1, \ldots, y_n$ such that $y_i = x_1 \ldots x_i$. It will be convenient to divide the outputs into consecutive subintervals in the same manner as the inputs. Then, the outputs of the circuit can be written as $y_{1,1}, \ldots, y_{1,\Delta}, y_{2,1}, \ldots, y_{2,\Delta}, \ldots, y_{\frac{n}{\Delta},1}, \ldots, y_{\frac{n}{\Delta},\Delta}$ where $y_{i,j} = y_{(i-1)\Delta+j}$. These intervals for inputs and outputs are illustrated in Figure 8.3.

In a first step, we compute the prefixes for each group of inputs $x_{i,1}, \ldots, x_{i,\Delta}$, i.e. we compute $x'_{i,j} = x_{i,1}x_{i,2} \ldots x_{i,j}$ for $i = 1, \ldots, \frac{n}{\Delta}$ and $j = 1, \ldots, \Delta$. In a second step, we recursively apply the prefix computation on $x'_{1,\Delta}, x'_{2,\Delta}, \ldots, x'_{\frac{n}{\Delta},\Delta}$, gaining the prefixes $y_{i,\Delta} = x_1x_2 \ldots x_{i\cdot\Delta}$. In a third step, we finally fill up the gaps between those prefixes with $y_{i,j} = y_{(i-1),\Delta}x'_{i,j}$ for $i = 2, \ldots, \frac{n}{\Delta}$ and $j = 1, \ldots, \Delta-1$. The layout and structure of the circuit is shown in Figure 8.3. Figure 8.4 shows the whole circuit for $\Delta = 2, n = 8$.

Since the construction of the circuit and layout is recursive, we can give recursive formulas for size, depth, area and total wire length of the circuit. Let $PREF^n$ denote such a layout with n inputs. The circuit consists of $\frac{n}{\Delta}(\Delta - 1)$ gates in the first computation step, gates in the recursive step and $\frac{n}{\Delta}(\Delta-1)$ gates in the third computation step:

$$\begin{aligned} C(PREF^n) &\leq \frac{n}{\Delta}(\Delta - 1) + C(PREF^{\frac{n}{\Delta}}) + \frac{n}{\Delta}(\Delta - 1) \\ &= 2n - 2\frac{n}{\Delta} + C(PREF^{\frac{n}{\Delta}}) \quad . \end{aligned}$$

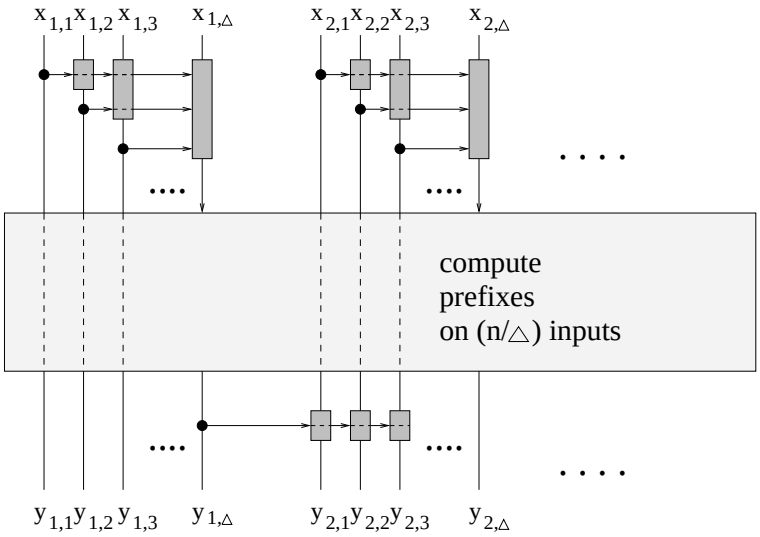


Figure 8.3: Layout of an efficient prefix circuit with fan-in Δ. Dark shaded boxes are gates, the light shaded box is the recursive application of the circuit.

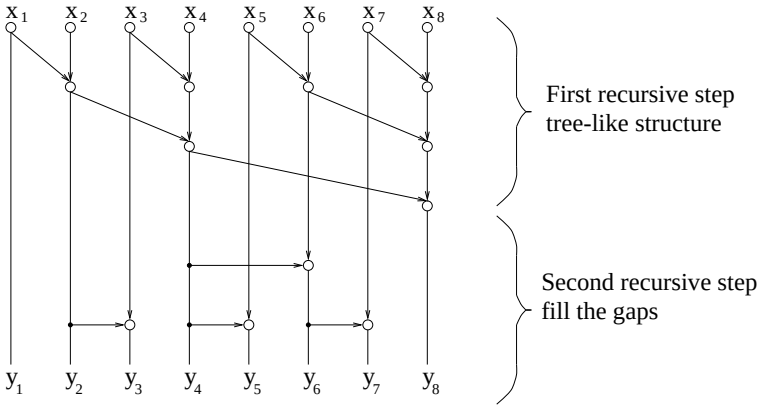


Figure 8.4: The prefix circuit for $\Delta = 2$ and $n = 5$. It can be decomposed in a tree-structure and a post-processing. (Based on Figure 2.13 in (Savage, 1998).)

The solution to this recurrence is $C(PREF^n) \leq 2n$, since $C(PREF^1) = 0$. Each recursive step adds depth 2 to the circuit depth:

$$depth(PREF^n) = 2 + depth(PREF^{\frac{n}{\Delta}}) \quad .$$

The solution to this recurrence is $depth(PREF^n) = 2\frac{\log n}{\log \Delta}$, since $depth(1) = 0$. To bound the occupied area in the VLSI-model, we compute the vertical side length $S(PREF^n)$ of the layout. Let $area(L)$ denote the area used by a layout

L.

$$\begin{aligned} S(PREF^1) &= 0 \\ S(PREF^n) &\leq (\Delta - 1) + S(PREF^{\frac{n}{\Delta}}) + 1 = \Delta \frac{\log n}{\log \Delta} \qquad (8.7) \\ area(PREF^n) &\leq nS(PREF^n) = n\Delta \frac{\log n}{\log \Delta} \quad . \qquad (8.8) \end{aligned}$$

Note that this area bound is derived for the VLSI-model. In our model, there is a better layout since we do not need space for wires. Nevertheless, Figure 8.3 gives an idea of a recursive formula for the total wire length of *horizontal* wires:

$$\begin{aligned} TWL(PREF^1) &= 0 \\ TWL(PREF^n) &\leq \frac{n}{\Delta}\Delta^2 + \Delta TWL(PREF^{\frac{n}{\Delta}}) + n \\ &= n\Delta + n + \Delta TWL(PREF^{\frac{n}{\Delta}}) \\ &= n(\Delta + 1)\frac{\log n}{\log \Delta} \quad . \end{aligned}$$

Since the circuit has logarithmic depth, *vertical* wires have a summed length of $O(n\Delta\frac{\log n}{\log \Delta})$ and the upper bound for total wire length is:

$$TWL(PREF^n) = O(n\Delta \frac{\log n}{\log \Delta}) \quad . \qquad (8.9)$$

The advantage of this circuit in our model is that one can implement it in area $O(n)$ without increasing the total wire length. As shown in Figure 8.4, the circuit implements a Δ-ary tree to compute larger and larger prefixes (first recursive step) and then fills up gaps in the computed prefixes (second recursive step). We will show that the tree can be implemented within two rows. This area efficient layout does not preserve the horizontal order of inner nodes. Since the horizontal order of some inner nodes is important for the subsequent computation, we will route their outputs to a more meaningful location.

The area-efficient implementation of a tree is illustrated in Figure 8.5. Consider a Δ-ary tree where an inner node is placed beneath the rightmost root of its subtrees. This layout has total wire length $O(n \log n)$. We show how to rearrange the inner nodes to achieve an area-efficient layout. We place an inner node beneath the leftmost leaf of its rightmost subtree. Note that this location is always free (also in our prefix circuit), and that the total wire length of this layout is bounded from above by the total wire length of the previous layout. But it uses just two rows on the grid.

In a prefix-circuit with this efficient tree layout, if the output of an inner node is needed for further computation, we will need to route it such that the horizontal ordering of the computed values is correct. In the upper layout of Figure 8.5, the

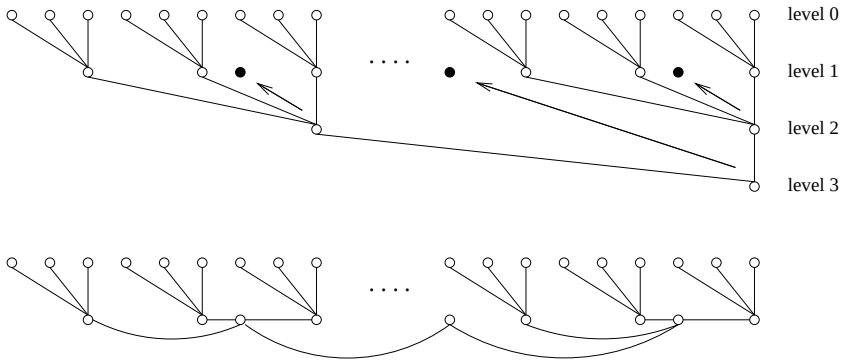


Figure 8.5: Tree layout on a grid for $\Delta = 3$. The upper layout shows a leveled arrangement of inner nodes. The arrows and filled circles indicate the rearrangement of inner nodes. We gain a layout which uses just two rows (lower layout).

nodes are in the correct horizontal order. The root of the tree in the lower layout of Figure 8.5 is horizontally displaced. So, we will need to route back some of the outputs of the displaced inner nodes. For simplicity, we assume that all displaced inner nodes are rooted back. As shown in Figure 8.5, we level the nodes of the tree such that leaves are in level 0, inner nodes which are incident to leaves are in level 1 and so on. More formally, a node v is in level i if the shortest path from v to some leaf has i edges. There are $\frac{n}{\Delta^i}$ nodes in level i. The horizontal displacement of a node in level i is Δ^{i-1} and nodes in level 1 are not displaced. Hence, the summed displacement of nodes is: $\sum_{i=2}^{\log_\Delta n} \Delta^{i-1} \frac{n}{\Delta^i} \leq \frac{n}{\Delta} \frac{\log n}{\log \Delta}$.

It remains to be shown that the computations in the second recursion step (see Figure 8.4) can be implemented in one row. Just observe that whenever there is a gate in this second recursive step, it computes an output of the circuit. Hence in the second recursive step, every output needs at most one gate and they can be arranged in one row. Also, since this does not further increase the total wire length, the vertical side length of the layout is constant.

∎

Note that in the model for total wire length estimation, the layout uses only a constant number of rows, whereas in the VLSI-model, a logarithmic number of rows is employed. This discrepance is due to the constraint of a constant number of layers in the VLSI-models, since VLSI-implementations are basically two-dimensional. In the biologically inspired model, more space for wires is available, since the cortex has three-dimensional characteristics.

Chapter 9

Relationship between Total Wire Length and other Circuit Complexity Measures

Since the total wire length was inspired by traditional circuit complexity theory, it makes sense to relate it to measures of the classical theory. In the previous chapters, the total wire length scaled linearly or almost linearly with the number of gates and inputs of the circuit. However, this is not always the case. As emphasized, special care must be taken to achieve a small total wire length for a function even if a circuit whose gates scale linearly with the number of inputs can easily be found. In Section 9.1 we show that the total wire length of a function scales at most quadratic with its circuit size and cannot be smaller than the circuit size. Section 9.2 shows that the total wire length of a function is asymptotically bounded from above by the area of the function in the common VLSI model. This chapter is based on a section in (Legenstein and Maass, 2001b).

9.1 Total Wire Length and Circuit Size

The most common complexity measure in traditional circuit complexity theory is the circuit size $C(f)$ of a Boolean function $f : \{0,1\}^n \to \{0,1\}$. $C(f)$ is the smallest number of gates in any feedforward circuit for f over some basis Ω. The basis Ω is normally indicated by writing $C_\Omega(f)$. We omit this index and assume that gates of the optimal circuits for $C(f)$ and $TWL(f)$ are drawn from the same basis Ω. We assume that f depends on each of its n variables. The relationship between the total wire length and the circuit size of a function is given by the following theorem.

Theorem 9.1.1 *The total wire length $TWL(f)$ of a function $f : \{0,1\}^n \to \{0,1\}$*

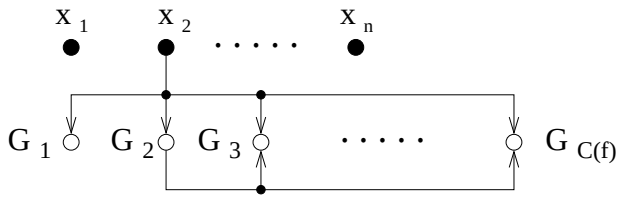


Figure 9.1: A layout for an arbitrary circuit whose total wire length can easily be estimated in terms of n and $C(f)$. Filled circles $x_1, \ldots, x_n$ are input ports and open circles $G_1, \ldots, G_{C(f)}$ are gates.

relates to its circuit size $C(f)$ in the following manner:

$$C(f) + n - 1 \leq TWL(f) \leq \frac{1}{2}C(f)(C(f) - 1) + n \max\{n, C(f)\}.$$

Proof: To show the first inequality, note that each input to the circuit as well as each gate of the circuit contributes to the output. Hence there is at least one edge from each input port to some gate and each gate except the output gate has fan-out at least one. Since gates and input ports are separated by unit distance, each such connection has at least unit length. The first inequality follows.

To show the second inequality, we construct a layout for some circuit C with circuit size $C(f)$. Since the circuit is feedforward, we can label the gates of C by $G_1, \ldots, G_{C(f)}$ such that G_i does not get input from gate G_j for all $1 \leq i < j \leq C(f)$. Arrange the gates on a row of the grid such that gate G_i is one unit to the left of G_{i+1} $(1 \leq i < C(f))$. In this arrangement all gates that receive input from some gate G_i are to the right of G_i (see Figure 9.1). Since outputs may spread, the wire length to connect G_i to all of its successors is at most $C(f) - i$. This results in a total wire length of $\frac{1}{2}C(f)(C(f) - 1)$ for connections between gates of the circuit. Furthermore, arrange the input ports of the circuit on the row one unit above the gates. In the worst case, each input port is connected to each gate. The wire length needed to connect one of the n input ports with all the gates is bounded by n if $n > C(f)$ and by $C(f)$ if $n < C(f)$. Hence, the total wire length needed to connect input ports to gates is at most $n \max\{n, C(f)\}$. This yields the second summand in the claimed upper bound for $TWL(f)$. ∎

9.2 Total Wire Length and Area

Another interesting question is how the total wire length of a function f relates to the area needed to implement f in VLSI. For the VLSI-model discussed in

Section 5.3 with gates of fan-in 2, we show that the total wire length is bounded by the area needed to compute f.

Theorem 9.2.1 *If the function $f : \{0,1\}^n \to \{0,1\}$ can be computed in a feed-forward manner in VLSI with μ layers, separation λ and area A, then the total wire length of f is bounded by*

$$TWL(f) = O(\frac{\mu}{\lambda^2}A).$$

Proof: We construct from a given VLSI-circuit for f a layout in our model for bounding its total wire length. We first superimpose a grid of grid-width $\lambda/2$ and area A over the VLSI-layout. Since gates, ports and wires have at lest width λ there is a grid-point in any gate and any two grid-points in connected gates can be connected by a grid-path that runs in the area of the gates and their connecting wire.

Consider a gate G with two inputs $I_{G,1}$, $I_{G,2}$ and output O_G. Define edges $(n_{I_{G,i}}, n'_{I_{G,i}})$ on the grid graph such that $n'_{I_{G,i}}$ is inside the area of the gate and $n_{I_{G,i}}$ is outside the area of the gate but inside the area of the i-th input wire $(i = 1, 2)$. Define edges (n_{O_G}, n'_{O_G}) for the output in a similar way (see Figure 9.2). Build a spanning tree on grid nodes and edges inside the area of G that connects $n'_{I_{G,1}}$, $n'_{I_{G,2}}$ and n'_{O_G}. We call this tree the *inner tree* of G. Consider the three paths of minimal length that connect two of these three nodes within the spanning tree (*i.e.* one path connects $n'_{I_{G,1}}$ with $n'_{I_{G,2}}$, one $n'_{I_{G,1}}$ with n'_{O_G}, and another path connects $n'_{I_{G,2}}$ with n'_{O_G}). There is exactly one node n_G that is contained in each of these three paths as the following observation shows.

Observation 9.2.2 *On a tree, denote a path without cycles from a node a to a node b by $P_{a,b}$. On a tree with three leafs A, B and C, there is exactly one node D that is contained in each of the paths $P_{A,B}$, $P_{A,C}$ and PB, C.*

Proof(Observation 9.2.2): Note that a path $P_{a,b}$ between nodes a, b of a tree is unique, since otherwise there would be a cycle in the tree. For a tree with 3 leafs A, B and C, let D be the last node on the paths $P_{A,B}$ and $P_{A,C}$ that is visited on both paths (this node exists, since each path is unique and both start from the same node A). Since $P_{A,D}$ and $P_{D,B}$ constitute $P_{A,B}$, the only node common to both is D (in $P_{A,B}$, each tree-node is visited at most once). The paths $P_{B,D}$ and $P_{D,C}$ constitute a path without cycles from B to C which uniquely defines $P_{B,C}$. This shows that D is visited in each of these 3 paths.

The only nodes that are common to $P_{A,B}$ and $P_{A,C}$ are the nodes in $P_{A,D}$. The only nodes that are common to $P_{A,B}$ and $P_{B,C}$ are the nodes in $P_{B,D}$. Hence, the only node that is common to $P_{A,B}$, $P_{A,C}$ and $P_{B,C}$ is D, which is the only node common to $P_{A,D}$ and $P_{B,D}$. ∎

In our construction, the gate G is mapped onto n_G and the inner tree is part of the connection-graph of the layout.

We treat an input-port G like a gate with $m \geq 1$ outputs and no input. Define edges $(n_{O_{G,i}}, n'_{O_{G,i}})$ in a similar way $(i = 1, \ldots, m)$. Build the inner tree that connects all the nodes $n'_{O_{G,1}}, \ldots, n'_{O_{G,m}}$ and define one of these nodes to be the mapping of G onto the grid. Output-ports can be treated in a similar way like gates with one input and no output. We will not distinguish between ports and gates in the following.

To connect a gate G with its successors $H_1, \ldots, H_m$, build a spanning tree on grid nodes and edges in the area of the VLSI-wires from G to its successors that connects n_{O_G} with the corresponding input-nodes $n_{I_{H_1}}, \ldots, n_{I_{H_m}}$. We refer to the spanning tree from n'_{O_G} to $n'_{I_{H_1}}, \ldots, n'_{I_{H_m}}$ as the *output-tree* of G (note that this definition involves nodes inside the gate-areas). The tree that connects a gate G with its successors $H_1, \ldots, H_m$ in the VLSI-Layout can therefore be mapped onto the output-tree of G together with parts of the inner trees such that n_G is connected to $n_{H_1}, \ldots, n_{H_m}$ (one may skip edges in the trees that are not needed for these connections). Hence, a gate G is connected to some gate H in the constructed layout, if and only if G is connected to H in the VLSI-circuit.

We show that for a given grid-edge, there are at most μ output-trees and at most one inner tree that contain this edge in the constructed layout. Consider an edge e of the grid-graph. Since wires in a layer are separated by at least λ and e has length $\lambda/2$, at most one VLSI-wire per layer intersects e (*i.e.* e is partly or fully in the area of this wire). Since there are μ VLSI-layers for wires, this shows that there are at most μ VLSI-wires that intersect e. Since, by construction, any edge in an output-tree intersects with a VLSI-wire, this shows that there are at most μ output-trees that contain e. Furthermore, since gates are separated by at least λ, at most one gate fully covers e (note that we tread inputs and output ports like gates and there is only one layer for gates and ports). Hence e is part of at most one inner tree. Since each tree uses e only once, e is used at most $\mu+1$ times in the whole constructed graph.

We can bound the number of edges in a grid-graph of area A and grid-width $\lambda/2$ by $O(A/\lambda^2)$. Since each grid edge is used at most $\mu+1$ times and grid edges have length 1 in our model for total wire length, the total wire length of the constructed layout is $O(\frac{\mu}{\lambda^2}A)$. ∎

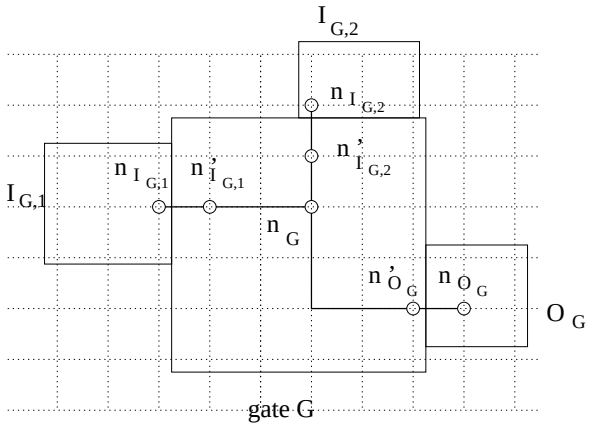


Figure 9.2: A mapping of a VLSI-gate onto our model. Unique nodes inside and outside the gate are defined for input and output. The nodes are connected by a (minimal) spanning tree. The gate is mapped onto the node n_G, the node that is common to any path that connects two leaves of the spanning tree.

Chapter 10

Optimizing the Layout of a Complete Tree

Tree-like organizations are a fundamental structure in parallel processing. The layout of trees has so far been studied mostly for trees of degree four or less. In this chapter, we give tight upper and lower bounds on the total wire length of complete m-ary trees ($m \geq 2$) on a two-dimensional grid if the leaves are constraint to lie on a grid line. For the case of binary trees, our construction results in a reduction of the total wire length of 33 % compared to the obvious "symmetric" layout. This chapter is based on the publication (Legenstein and Maass, 2001d).

10.1 Introduction

The importance of tree-like structures as an organizing principle for processing elements was stressed out by many authors, see *e. g.* (Bhatt and Leiserson 1982; Ullman, 1984; Savage, 1998). Therefore, the layout of trees in the context of VLSI-circuits has been studied intensively. Mead and Rem introduced the H-tree layout, thus showing that a complete binary tree with n leaves can be implemented in $O(n)$ area (Mead and Rem, 1979). This upper bound on area induces an upper bound on wire length. In this strategy, leaves are placed throughout the chip surface. The H-tree layout is described and studied in Chapter 7. If leaves have to be placed on the boundary of a convex region, the area of any layout is bounded by $\Omega(n \log n)$, which was shown by Brent and Kung (Brent and Kung, 1982). A generalization to noncomplete trees is due to Yao (Yao, 1981). The way how chips that implement "tree machines" (*i. e.* processing elements structured as trees) can be combined to larger trees, is investigated in (Bhatt and Leiserson 1982). Paterson, Ruzzo, and Snyder addressed the problem of minimizing the longest edge in a tree layout. It was shown in (Paterson et al., 1981) that in

any layout for a complete binary tree of n leaves, there is an edge of length $\Omega(\sqrt{n}/\log n)$. Furthermore a layout is given that achieves this bound. This result is extended to binary trees (*i. e.* also noncomplete binary trees) in (Ruzzo and Snyder, 1981).

The total wire length of tree-layouts was also discussed by Fisher and Paterson in (Fischer and Paterson, 1999). A preliminary version of this paper appeared in (Fischer and Paterson, 1980).The Authors give an efficient algorithm that produces a optimal layout for weighted trees under the L_1-norm if crossings are allowed.[1] Closed-form expressions on the cost of a tree layout for some special cases are given. Especially for the case of unit cost weights and a further constraint[2], their model is similar to ours if binary trees are considered. Our results for m-ary trees in the case of $m = 2$ are similar to those of Fisher and Paterson in this special case. The layout of m-ary trees on a grid was to our knowledge not yet considered in the literature. Since the degree of grid nodes is at most 4, merely trees of degree four or less were studied.

The layout of trees is also studied in the context of graph drawing. For a recent survey on graph drawing we refer to (Di Battista et al., 1999). However, the problem of minimizing the total wire length of trees has previously not been addressed in this context.

We assume here that all n leaves of the tree have to lie on adjacent intersection points on one horizontal row of the grid. This is a case of practical interest. For example, the leaves may present input/output ports, and such ports are normally placed on the border of a chip. Since ports are usually much larger than electrical elements, the total wire length is very sensitive to the horizontal layout of the tree. All intermediate nodes of the tree are to be placed on intersection points of the grid and edges of the tree are required to be realized by edge-disjoint paths along the grid lines. Since the vertical components of all wires contribute in any reasonable layout just a linear term to the total wire length, it suffices to focus on the horizontal components of the wires, and thus on the horizontal coordinates of the inner nodes of the tree on the grid. Intuitively one might think that a symmetric layout where the horizontal position of each node is as close as possible to the middle between the horizontal position of its children is optimal. However we show in this chapter that there exists another layout that reduces the total wire length by up to 33 %. Furthermore we show that this other layout strategy is optimal for the length of horizontal edges.

We will give a precise definition of the problem in Section 10.2. In Section 10.3

[1] In a weighted tree, each edge has a corresponding weight. The cost of and edge is its wire length multiplied with its weight.

[2] In their nomenclature, this constraint is that leaves are constrained in a *natural* order. This means that leaves of any two subtrees are on different intervals of the line they are arranged on. In our nomenclature this would mean that the subtrees induced by the two children of the root are non-overlaping (see below). Since we merely constrain the leaves to lie on a horizontal line, this is equivalent to our problem for the binary case.

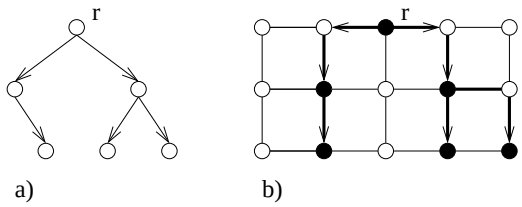


Figure 10.1: A graph G (a) and a layout L_G of G in the grid-graph (b). In (b), filled circles are nodes corresponding to nodes in G and bold lines are edges corresponding to edges in G. $HWL(L_G) = 3, VWL(L_G) = 5$ and $TWL(L_G) = 8$.

we will consider a layout strategy for binary trees. We give a simple argument which shows that the naive "symmetric" layout, which requires a total wire length of $\frac{n}{2}\log_2 n + O(n)$, is not optimal.The strategy presented uses a total wire length of $\frac{n}{3}\log_2 n + O(n)$. This upper bound is implicit in one of the results of (Fischer and Paterson, 1999). However, Section 10.3 may serve as a basis for Section 10.4, where an upper bound for the case of complete m-ary trees is given. This layout strategy yields a total wire length of $\frac{m-1}{m+1} n \log_m n + O(n)$. Finally it is shown in Section 10.5 that this layout strategy is optimal for any $m \geq 2$ in the sense that no other layout can achieve for any $m \geq 2$ a total wire length $a \cdot n \cdot \log_m n + O(n)$ with $a < \frac{m-1}{m+1}$. This lower bound argument is of some interest from the technical point of view since there exist many other layouts that require less wire length on some levels of the tree, but have to spend then more wire length on other levels of the tree. The lower bound argument has to take care of all these possible trade-offs, and hence cannot be carried out by a simple inductive proof.

10.2 The Layout Model

We consider a directed graph $G = (V, E)$, where V is the set of nodes and E is the set of edges, $E \subseteq V^2$. Since the graphs we consider are complete trees, we can level the nodes such that the root is in depth 0, all nodes that are incident to the root are in depth 1 and so on. More formally, the depth of a node v is the length of the shortest path from the root to v. Basic definitions for trees are given in Chapter 3.1.

We would like to define a layout of a graph $G = (V, E)$, i.e. we would like to map the graph onto the two dimensional plane. In our model, we consider a 2-dimensional grid with unit length in between neighboring grid-lines. We want to embed the graph into this grid graph such that each node $v \in V$ is mapped onto a node of the grid-graph by an injective function. Furthermore, the edges of the graph are mapped onto edge-disjoint paths of the grid-graph, connecting the corresponding nodes. Thus, a layout can be defined by a graph $L_G = (V', E')$ which is a subgraph of the grid-graph (see Figure 10.1).

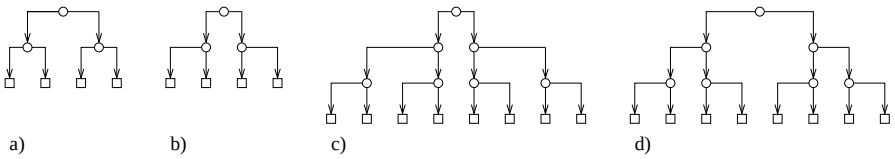


Figure 10.2: Different layouts for binary trees. Rectangles are leaves and circles are inner nodes. The layout in (a) is not optimal, since one can shift the children of the root towards it (b). A strategy with short wires from the root (c) and one that minimizes the wire-length of the subtrees (d).

The total wire length $TWL(L_G)$ of a layout L_G is the number of edges used in the grid-graph. We refer to the horizontal wire length $HWL(L_G)$ as the number of horizontal wires and the vertical wire length $VWL(L_G)$ as the number of vertical wires used in the grid-graph. For some layout L_G, it holds that $TWL(L_G) = HWL(L_G) + VWL(L_G)$.[3]

10.3 The Layout of Binary Trees

We investigate layouts for binary trees with respect to total wire length. In this section, we construct an algorithm which builds a tree of small total wire length. In Section 10.5, we show that the horizontal wire length of this layout is optimal. We will only consider full complete trees, *i. e.* trees of minimal depth. We assume that the leaves of the tree are given by consecutive nodes on a horizontal grid line. Then, one can build a binary tree by inserting inner nodes and edges into the graph. The common way of drawing trees is illustrated in Figure 10.2a. Each inner node is placed in the middle of its children, as far as this is possible in our grid model. The horizontal wire length of such a layout would be at least $\frac{n}{2}((\log_2 n) - 1) + 1$ as we show below.

Proof: The naive layout places each node horizontally in the middle of its children. We first assume that this is possible, although it is not always possible in the grid model. Then we correct the bound by the maximal error we made by this assumption. This maximal error is at most half a grid unit for each inner node of the tree and we assume that the horizontal wire length of the layout is reduced by placing nodes onto grid-nodes. This amounts to a reduction of at most $\frac{1}{2}(n-2)$ in the horizontal wire length. If we assume that we can always place a node in the middle of its children, the horizontal wire length of a naive layout T_n with n leaves can be described by the recursive formula $HWL(T_n) = \frac{n}{2} + 2HWL(T_{n/2})$ with the additional constraint that

[3]Note the differences to the model introduced in Chapter 5.1. First, the L_1-norm is used here instead of the Eucledian metric to determine the length of wires. Furthermore, in this model, wires may not spread on arbitrary positions and may no use edges twice.

$HWL(T_1) = 0$. This recursion evaluates to $\frac{n}{2}\log_2 n$. Together with the correction term this results in a horizontal wire length of at least $\frac{n}{2}((\log_2 n)-1)+1$. ∎

This layout is not optimal, since one can reduce the horizontal wire length by shifting these nodes towards the root. This is possible because the horizontal wire length of the subtree is independent of the placement of its root, as long as it stays in between its children (see Figure 10.2b). However, Figure 10.2b merely deals with a tree of depth 2. In trees of larger depth it is not so clear where to place the children of the root, since minimizing the length of wires from the root will increase the wire length of the subtrees. The layout in Figure 10.2c places the inner nodes in such a way that the wire length is optimized for the root wires, but not for the subtrees. In Figure 10.2d, the subtrees are optimized, while the wires needed to connect them to the root are longer. These effects cancel for a depth 3 tree, but for trees of depth 4 or higher, the latter layout (Figure 10.2d) guarantees a smaller total wire length.

To keep things simple, we assume that the number of leaves is $n = 2^k$ for some natural number $k \geq 0$. In the following, we describe easy rules that define a tree-layout as given in Figure 10.2d. The underlying graph of the layout is constructed in the common way: Insert the root r into the set of nodes V. Recursively build the tree for the $\frac{n}{2}$ leftmost leaves and the tree for the $\frac{n}{2}$ rightmost leaves. Then, connect the roots of these trees with r. Since the connectivity of the tree is given, we can now describe the layout of the vertices and edges. Figure 10.2d indicates that there are two ways to place a node. Either above its leftmost or above its rightmost child. This reflects the principle of minimizing the length to the parent without increasing the wire length of its subtree. The rules can be described as follows (see Figure 10.3):

Let u be a node to be placed on the grid. Let T_u denote the subtree which is rooted at u. Let v be the parent of u.

- If T_u is the leftmost subtree with respect to v (i.e. its leaves are to the left of all other leaves in T_v), place u above its rightmost child.
- If T_u is the rightmost subtree with respect to v (i.e. its leaves are to the right of all other leaves in T_v), place u above its leftmost child.

It is straight-forward to design an algorithm that constructs such a tree-layout. Such an algorithm would start to place nodes that are incident to leaves, since their placement is independent of the placement of other nodes. After these nodes are placed, it places nodes that are incident with them and so on. We call this algorithm *Greed*.The horizontal wire length of a layout produced by Greed is given in the following claim.

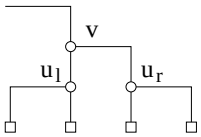


Figure 10.3: u_l is the root of the leftmost subtree with respect to v. Therefore, u_l is placed above its rightmost child. u_r is the root of the rightmost subtree with respect to v. Therefore, u_r is placed above its leftmost child.

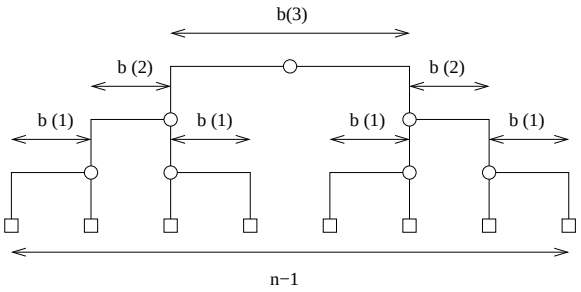


Figure 10.4: A binary tree layout. Rectangles are leaves and circles are inner nodes. The *root width* of this tree is $b(3) = 3$. But since the subtrees are constructed in the same manner, also $b(2)$ and $b(1)$ appear within this tree.

Claim 10.3.1 *For some $k \geq 0$, the* Greed*-algorithm produces a layout for a complete binary tree of $n = 2^k$ leaves – that are placed on consecutive grid nodes of a horizontal grid-line – with horizontal wire length of $\frac{1}{3}n\log_2 n + \frac{1}{9}(n + (-1)^{(\log_2 n)+1})$.*

Proof: Let T_n be a binary tree layout for $n = 2^k$ leaves constructed by Greed. We define the *root width* $b(k)$ to be the distance in between the children of the root (*i. e.* nodes of depth 1 in the tree). Figure 10.4 illustrates the root width of a tree constructed by Greed. Since the subtrees of T_n are constructed in the same manner, $b(1), \ldots, b(k)$ can be found within subtrees of T_n. To be more precise, $b(i)$ appears 2^{k-i} times (see Figure 10.4), and the horizontal wire length of T_n is the sum of these root widths:

$$HWL(T_n) = \sum_{i=1}^{k} 2^{k-i} b(i) \tag{10.1}$$

A recursive formula for $b(k)$ can be stated. Consider the line segment l that is defined by the leftmost and rightmost leaves as endpoints. l is $n - 1 = 2^k - 1$ in length. Projecting the wires from the root to its children onto l results in an interval of size $b(k)$ on this line. The root is connected to two trees of root-width $b(k-1)$. Because of the placement of nodes, the projections of the root-widths of

the tree and its left and right subtrees are directly adjoining and non-overlapping (see Figure 10.4). Projecting the root widths of leftmost and rightmost subtrees of depth i for $i = 1, \ldots, k$ onto l results in adjoining and non-overlapping intervals that fully cover l. Hence, for any $k \geq 1$ it holds that

$$b(k) = (2^k - 1) - 2\sum_{i=1}^{k-1} b(i) \tag{10.2}$$

and $b(0) = 0$. We show by induction on $k \geq 0$ that the following holds:

$$b(k) = \frac{1}{3}(2^k + (-1)^{k+1}). \tag{10.3}$$

Obviously, the induction hypothesis, Equation 10.3, is true for $k = 0$. Using Equation 10.2, we show the inductive step:

$$\begin{aligned}
b(k+1) &= (2^{k+1} - 1) - 2\sum_{i=1}^{k} b(i) = (2^{k+1} - 1) - \frac{2}{3}\sum_{i=1}^{k}(2^i + (-1)^{i+1}) \\
&= (2^{k+1} - 1) - \frac{2}{3}\Big(\sum_{i=1}^{k} 2^i + \sum_{i=1}^{k}(-1)^{i+1}\Big) \\
&= (2^{k+1} - 1) - \frac{2}{3}\Big(2^{k+1} - 2 + \frac{1}{2}((-1)^{k+1} + 1))\Big) \\
&= \frac{1}{3}2^{k+1} - \frac{1}{3}(-1)^{k+1} = \frac{1}{3}(2^{k+1} + (-1)^{(k+1)+1}).
\end{aligned}$$

We can now sum up the root widths that occur in our tree layout with Equation 10.1 to compute the horizontal wire length of the layout.

$$\begin{aligned}
HWL(T_n) &= \sum_{i=1}^{k} 2^{k-i} b(i) = \sum_{i=1}^{k} 2^{k-i}\frac{1}{3}\big(2^i + (-1)^{i+1}\big) \\
&= \frac{1}{3}\Big(\sum_{i=1}^{k} 2^k + \sum_{i=1}^{k} 2^{k-i}(-1)^{i+1}\Big) = \frac{1}{3}\Big(k2^k + \sum_{j=0}^{k-1} 2^j(-1)^{k-1+j}\Big)
\end{aligned}$$

We eliminate the remaining alternating sum with the formula

$$\sum_{j=0}^{k} 2^j(-1)^{k+j} = \frac{2^{k+1} + (-1)^k}{3}. \tag{10.4}$$

Since $2^k = n$ and $k = \log_2 n$, the horizontal wire length of this layout evaluates to

$$HWL(T_m^n) = \frac{1}{3}k2^k + \frac{1}{9}\Big(2^k + (-1)^{k+1}\Big) = \frac{1}{3}n\log_2 n + \frac{1}{9}\Big(n + (-1)^{(\log_2 n)+1}\Big).$$

■

In the Section 10.5, we will show that this bound is optimal even if we do not restrict inner nodes to lie on grid-points. As far as the vertical wire length is concerned, there is one vertical wire for each node except for the root, which results in a vertical wire length of $VWL(T_n) = 2(n-1)$. Hence, the total wire length of T_n is $TWL(T_n) = \frac{1}{3}n\log_2 n + \frac{1}{9}(n + (-1)^{(\log_2 n)+1}) + 2(n-1) = \frac{n}{3}\log_2 n + O(n)$.

10.4 Upper Bounds for Complete Trees

The simplicity of the tree construction in Section 10.3 is instructive. However, with a somewhat more elaborate model, one can generalize this construction to complete m-ary trees for any $m \geq 2$. We face the problem that for any $m \geq 4$, there is no way to use edge-disjoint paths to connect nodes with their children (since the degree of inner nodes of the tree is more than 4). Hence, we have to extend our model. Consider a computational circuit C. This circuit has an underlying directed graph $G = \{V, E\}$ where V is viewed as the set of gates and input-ports of the circuit and E defines connections between them. In a physical implementation of C, if some gate g is connected to m children, the edges from g to its children may be implemented by a wire that starts at g and spreads later on to serve as input for the children. Such a connection strategy is reasonable at least as long as m is constant, *i. e.* does not grow with the size of the circuit.

A convenient way to define such a layout is to insert nodes into V that serve as routing points for the edges. Hence we define a *routing graph* $G' = (V', E')$ where $V \subseteq V'$. Nodes in $V' - V$ are called *routing nodes*. Furthermore, each edge $(v_i, v_j) \in E$ is mapped onto a path in G' that starts with v_i and ends with v_j and any other node on the path is a routing node. Since the graph G is underlying some computational circuit, we need to be careful about routing nodes. As the routing nodes do not represent computational units, it does not make sense for routing nodes to have an in-degree larger than one[4]. To get a layout L_G, we can map the nodes of G' onto nodes of the grid-graph by some function $pos : V' \to \mathbb{N}^2$ and furthermore map the edges of G' onto edge-disjoint paths of the grid-graph. The horizontal, vertical, and total wire length can be defined as in Section 10.2.

Within this model, the principle of this layout given in Section 10.3 is also applicable for m-ary trees where m is larger than 2. Figure 10.5 shows a layout for a 3-ary tree of depth 3. In the following, we give simple rules on how to place inner nodes on the grid that define a tree-layout with small total wire length[5].

[4] The in-degree of a node v in a graph G is defined as the number of edges (u, w)in G with $w = v$. Similarly, the out-degree v is defined as the number of edges (u, w)in G with $u = v$.

[5] To keep things simple, we assume that the number of leaves is $n = m^k$ for some natural number $k \geq 0$.

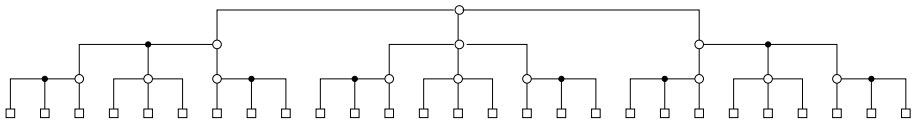


Figure 10.5: A layout for a 3-ary tree of depth 3. Rectangles are leaves, open circles are inner nodes, and filled circles are routing nodes. Note that all edges are directed from nodes of smaller depth to nodes of larger depth.

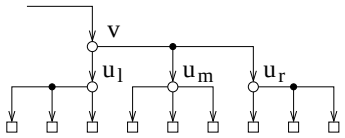


Figure 10.6: u_l is the root of the leftmost subtree with respect to v. Therefore, u_l is placed above its rightmost child. u_r is the root of the rightmost subtree with respect to v. Therefore, u_r is placed above its leftmost child. u_m may be placed above any of its children.

Let u be a node to be placed on the plane. Let T_u denote the subtree with root u and v denote the parent of u (see Figure 10.6).

- If T_u is the leftmost subtree with respect to v (i.e. its leaves are to the left of all other leaves in T_v), place u above its rightmost child.
- If T_u is the rightmost subtree with respect to v (i.e. its leaves are to the right of all other leaves in T_v), place u above its leftmost child.
- Otherwise, u may be placed above any of its children.

Again, it is straight-forward to design an algorithm $Greed_m$ that constructs such a tree-layout. $Greed_m$ start to place nodes that are incident to leaves, since their placement is independent of the placement of other nodes. After these nodes are placed, it places nodes that are incident with them and so on. The horizontal wire length of a layout produced by $Greed_m$ can be calculated in the same manner as in Section 10.3.

Theorem 10.4.1 *If, for some $k \geq 0$, $m \geq 2$, the $m = 2^k$ leaves of a complete m-ary tree are placed on consecutive grid nodes of a horizontal grid-line, then there exists a layout in the grid model with horizontal wire length of $\frac{m-1}{m+1} n \log_m n + \frac{m-1}{(m+1)^2}(n + (-1)^{(\log_m n)+1})$.*

Proof: Let $T_m^{m^k}$ be a m-ary tree layout constructed by $Greed_m$. We define the *root width* $b_m(k)$ to be the maximum over all horizontal distances in between pairs

of nodes that are incident with the root (*i. e.* nodes of depth 1 in the tree). Using the same argument as in Section 10.3, the horizontal wire length of $T_m^{m^k}$ is

$$HWL(T_m^{m^k}) = \sum_{i=1}^{k} m^{k-i} b_m(i) \tag{10.5}$$

The root width $b_m(k)$ of a layout by $Greed_m$ for a tree $T_m^{m^k}$ is given by the recursive equation

$$b_m(k) = (m^k - 1) - 2\sum_{i=1}^{k} b_m(i) \tag{10.6}$$

for any $k \geq 1$ and $b(0) = 0$. As in Section 10.3, one can show that

$$b_m(k) = \frac{m-1}{m+1}\left(m^k + (-1)^{k+1}\right). \tag{10.7}$$

Again, simple algebra is used to compute the horizontal wire length of the layout by Equation 10.5.

$$\begin{aligned} HWL(T_m^{m^k}) &= \sum_{i=1}^{k} m^{k-i} b_m(i) \\ &= \sum_{i=1}^{k} m^{k-i} \frac{m-1}{m+1}\left(m^i + (-1)^{i+1}\right) \\ &= \frac{m-1}{m+1}\left(\sum_{i=1}^{k} m^k + \sum_{i=1}^{k} m^{k-i}(-1)^{i+1}\right) \\ &= \frac{m-1}{m+1}\left(km^k + \sum_{j=0}^{k-1} m^j (-1)^{k-1+j}\right). \end{aligned}$$

We eliminate the alternating sum with the formula

$$\sum_{j=0}^{k} m^j (-1)^{k+j} = \frac{m^{k+1} + (-1)^k}{m+1}. \tag{10.8}$$

Since $m^k = n$ and $k = \log_m n$, the horizontal wire length of this layout evaluates to

$$\begin{aligned} HWL(T_m^n) &= \frac{m-1}{m+1} km^k + \frac{m-1}{(m+1)^2}\left(m^k + (-1)^{k+1}\right) \\ &= \frac{m-1}{m+1} n \log_m n + \frac{m-1}{(m+1)^2}\left(n + (-1)^{(\log_m n)+1}\right). \end{aligned} \tag{10.9}$$

■

Note that this upper bound is a generalization of Claim 10.3.1. If we set $m = 2$, we get exactly the bound given for binary trees. In the following Section, we show that this general bound is optimal even if we do not restrict inner nodes to lie on grid-points. The vertical wire length of this layout is obvious. There is one vertical wire for each node except for the root, which results in a vertical wire length of $VWL(T_m^n) = \frac{mn-1}{m-1} - 1$. Hence, the total wire length of T_m^n is $TWL(T_m^n) = \frac{m-1}{m+1} n \log_m n + \frac{m-1}{(m+1)^2}\left(n + (-1)^{(\log_m n)+1}\right) + \frac{mn-1}{m-1} - 1 = \frac{m-1}{m+1} n \log_m n + O(n)$.

10.5 Lower Bounds for Complete Trees

We show that the bound given in Theorem 10.4.1 for the horizontal wire length of a tree layout is tight. We relax the grid-model such that except for the leaves, which are placed on successive nodes on a horizontal grid line as before, nodes may be placed anywhere on the two-dimensional plane. The wire length of an edge (u, v) in the routing graph is the Euclidean distance between u and v on the plane. Since we merely consider the horizontal components of wires, we can assume that all nodes lie on the horizontal line defined by the leaves of the tree. Hence, the position of a node $v \in V'$ is defined by a function $xpos : V' \to \mathbb{R}$ which maps v onto its x-coordinate. The horizontal wire length of an edge (u, v) is $|xpos(u) - xpos(v)|$ (note that the graph involves also routing nodes).

Before we give the proof, we make another assumptions about a layout for a complete tree that minimizes the horizontal wire length. For this assumption, we need the following definition. Consider two nonempty, disjunct sets of nodes $S_1, S_2 \in V$ of some layout L. We say that S_1 is *horizontal non-overlapping* with S_2 if some vertical line separates the nodes of S_1 and S_2, *i. e.* for each pair of nodes $v_1 \in S_1$ and $v_2 \in S_2$, it holds that $xpos(v_1) < xpos(v_2)$ or for each pair $v_1 \in S_1$ and $v_2 \in S_2$, it holds that $xpos(v_1) > xpos(v_2)$. If this is not the case, we say that S_1 is *horizontal overlapping* with S_2. Consider a layout for a m-ary complete tree T. The root of the tree is connected to m subtrees $T_1, \ldots, T_m$. We say that T is horizontal overlapping, if for some $i, j \in \{1, \ldots, m\}$ with $i \neq j$, the set of leaves of T_i is horizontal overlapping with the set of leaves of T_j. Otherwise, we say that T is horizontal non-overlapping. The following lemma states that without loss of generality, we can assume that a tree layout with minimal horizontal wire length is horizontal non-overlapping.

Lemma 10.5.1 *For any layout T of a complete tree in the model given above with horizontal wire length $HWL(T)$, there exists a layout T' where T' is horizontally non-overlapping and the horizontal wire length of T' is smaller or equal to $HWL(T)$.*

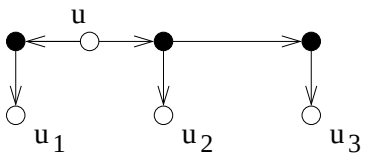


Figure 10.7: Filled circles are routing nodes. Given the positions of the nodes (open circles), this is an optimal way to connect u to its children with regard to the horizontal wire length. The nodes u_1, u_2, u_3 are on the same position as their corresponding routing nodes. The vertical displacement of nodes in the figure is for clarity.

Proof: We construct a horizontal non-overlapping layout T' out of T with smaller or equal horizontal wire length. This is done by constructing a layout where no paths cross. In the following, we define an operation on a layout that achieves this goal if applied correctly. Furthermore, we construct a layout R_2 out of T to show on which nodes to apply the operation.

One can assume that all nodes of T are on the horizontal line defined by the leaves. If this is not the case, shrink the layout to this line and consider this layout. Let r be the root of T.In the first step of the construction of T', we delete all routing nodes out of T and for each node $v \in V - \{r\}$ we place a routing node s_v on the same position as v. We connect each routing node s_v with its corresponding node v. Then for each node u and its children $u_1, \ldots, u_m$, connect u to the corresponding routing nodes with minimal total wire length (*i. e.* chain the routing nodes as shown in Figure 10.7). Note that the positions of nodes in V are not altered and the way to connect nodes with their children is optimal, hence the wire length of the layout does not increase by this manipulation.

We are now defining an operation on the layout that does not increase the horizontal wire length of the layout. We call such an operation a *flip*-operation. Consider two inner nodes u and v and the sets of their children S_u and S_v. Assume that u is to the left of v or placed on the same location as v. The operation connects u to the m leftmost nodes in the set of their children $S_u \cup S_v$ and v to the m rightmost nodes in $S_u \cup S_v$. This operation does not increase the horizontal wire length of the resulting layout. We formalize this idea: Suppose that $posx(u) \leq posx(v)$. Chose disjunct subsets $S'_u, S'_v \in S_u \cup S_v$ of size m each with the condition that for any pair of nodes $u' \in S'_u$ and $v' \in S'_v$ it holds that $xpos(u') \leq xpos(v')$. The flip-operation $flip_{u,v}$ deletes all connections in between u and its children and v and its children and connects u to the nodes in S'_u and v to S'_v in an optimal way. This will not increase the horizontal wire length of the layout.

We will use this operations to construct a horizontal non-overlapping tree out of T'. In order to figure out on which nodes to do that, we construct two other layouts R_1 and R_2 out of T.

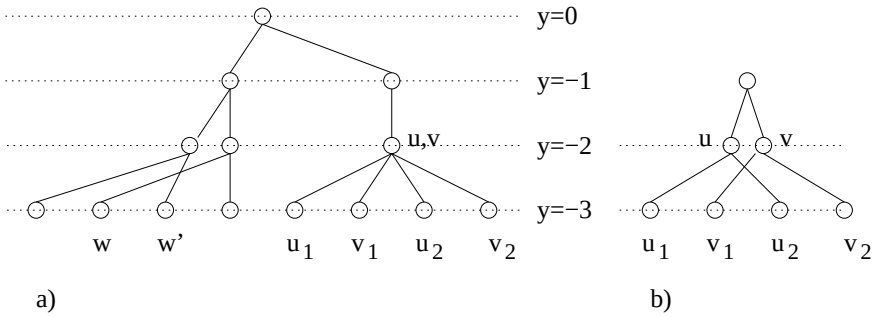


Figure 10.8: a) A stretched layout. Nodes of depth i are placed at y-coordinate $-i$. b) We separate nodes that are placed on the same position by a small amount. A detail of the layout in (a) is shown.

```
Let d be sufficiently small.
FOR all nodes u,v with pos(u)=pos(v) DO
  Set xpos(u):=xpos(u)-d/2
  Set xpos(v):=xpos(v)+d/2
  d := d/2
DONE
```

Figure 10.9: An algorithm that produces a layout R_2 out of R_1.

We construct R_1 out of T such that we delete routing nodes and connect graph nodes by direct edges. Furthermore, all nodes of depth i are placed on a horizontal line at y-coordinate $-i$ and the x-coordinates are unchanged. Such a stretched layout is given in Figure 10.8a.

In order to handle nodes that are on the same position on the plane , we construct another layout R_2 out of R_1 by the algorithm given in Figure 10.9. We do that by displacing nodes of same position by a small amount d in the x-direction, such that no two nodes are at the same x-position after the transformation. Let ϵ be the minimum horizontal distance greater than 0 between nodes of the layout. d must be chosen sufficiently small, so that if a node v is to the left of u in R_1, then v is to the left of u in R_2. This can be guaranteed by choosing $d = \epsilon/2$. In R_2, paths can only cross by edge-crossings.

Figure 10.8b shows a detail of the transformed version of the layout of Figure 10.8a. In the layout R_2, there are no two nodes that are placed on the same position. All edge crossings in R_2 are due to crossings that were in R_1 or or due to nodes that were at the same location in R_1. Suppose that there are nodes u, v such that edges (u, u') and (v, v') cross in R_2. A flip operation $flip_{u,v}$ on T' and the corresponding change in the connectivity in R_2 can be used to eliminate all crossings within edges from u and v to their children. Hence, all the crossings in

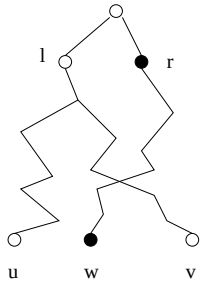


Figure 10.10: A tree layout R_2 where l and r are children of the root. If R_2 is horizontally overlapping, then a path from l crosses with a path from r.

R_2 can be eliminated without increasing the horizontal wire length in T'.

We show that after crossings were eliminated in R_2, R_2 is horizontal non-overlapping. Suppose that R_2 is horizontal overlapping (see Figure 10.10). There exist children of the root l, r with $xpos(l) < xpos(r)$ in R_2 such that for some leaves u, v of the subtree defined by l and w of the subtree defined by r it holds that $xpos(u) < xpos(w) < xpos(v)$. There are paths from l to u, from l to v and from r to w that do not visit the root of R_2. Since $xpos(u) < xpos(w) < xpos(v)$, the path from r to w crosses one of the other paths. Since no two nodes share the same location on the plane, edges cross. Hence, if there are no crossings in R_2, then R_2 is horizontal non-overlapping. T' is also horizontal non-overlapping, since its underlying graph has the same connectivity and the leaves are placed in the same order on the horizontal line. ∎

Theorem 10.5.2 *For integers $k \geq 1$ and $m \geq 2$, any layout of a complete tree with $n = m^k$ leaves that lie on a horizontal line with unit distance in between neighboring leaves has horizontal wire length of at least* $\frac{m-1}{m+1} n \log_m n + \frac{m-1}{(m+1)^2}(n + (-1)^{(\log_m n)+1})$.

Proof: We prove the theorem by induction on k for all n of the form $n = m^k$. To avoid cumbersome notation, we introduce the shortcut $f_m(n) = \frac{m-1}{m+1} n \log_m n + \frac{m-1}{(m+1)^2}(n + (-1)^{(\log_m n)+1})$ to denote the bound given above. We will prove a stronger result that implies the theorem. Since the m^k leaves are placed on a horizontal line, one can define the midpoint p_m of the leaves, which is the point that minimizes the maximal distance to a leaf. Let r be the root of the tree. The *deviation* x of the root is defined by $x = xpos(w) - xpos(p_m)$. The wire length of the tree layout will increase if the absolute value of the deviation is too big. On the other hand, we can give a region around the midpoint where no increase in wire-length occurs. This *bound region* for a tree of n leaves is defined by an interval of length $b_m(n) = \frac{m-1}{2(m+1)}(n + (-1)^{(\log_2 n)+1})$ to both sides of the

midpoint. If the root is placed within this interval, the wire length needed is independent of the deviation. If the root is placed outside this interval, the wire length increases linearly with the deviation. Hence, we define the *limited deviation* to be $\overline{x} = \max\{0, |x| - b_m(n)\}$. The horizontal wire length of the complete tree is at least $f_m(n) + \overline{x}$.

Observation 10.5.3 *Consider a layout for a m-ary tree with n leaves, deviation x and limited deviation $\overline{x}$. Then the following holds:*

a) $\overline{x} + x \geq -b_m(n)$,

b) $\overline{x} - x \geq -b_m(n)$,

c) $b_m(mn) + b_m(n) = \frac{n(m-1)}{2}$, *and*

d) $mf_m(n) = f_m(mn) - (m-1)n + 2b(n)$.

Proof(Observation 10.5.3): Observations 10.5.3a,b are easy to verify. Suppose that $|x| \leq b_m(n)$. Then $\overline{x} \pm x = \pm x \geq -b_m(n)$. Suppose that $|x| > b_m(n)$. Then $\overline{x} \pm x = |x| - b_m(n) \pm x \geq -b_m(n)$. Observation 10.5.3c is proven by $b_m(mn) + b_m(n) = \frac{m-1}{2(m+1)}(mn + (-1)^{\log_m n}) + \frac{m-1}{2(m+1)}(n + (-1)^{\log_m n+1}) = \frac{m-1}{2(m+1)}(mn + n) = \frac{n(m-1)}{2}$. Finally, we show Observation 10.5.3d:

$$\begin{aligned}
mf_m(n) &= \frac{m-1}{m+1}(mn)\log_m n + \frac{m-1}{(m+1)^2}(mn) + \frac{m-1}{(m+1)^2}(-1)^{(\log_m n)+1} \\
&= \frac{m-1}{m+1}(mn)\log_m (mn) - \frac{m-1}{m+1}(mn) + \frac{m-1}{(m+1)^2}(mn) \\
&\quad + \frac{m-1}{(m+1)^2}m(-1)^{(\log_m n)+1} \\
&= \frac{m-1}{m+1}(mn)\log_m (mn) + \frac{m-1}{(m+1)^2}(mn) + \frac{m-1}{(m+1)^2}(-1)^{(\log_m mn)+1} \\
&\quad - \frac{m-1}{m+1}(mn) + \frac{m-1}{(m+1)^2}(m+1)(-1)^{(\log_m n)+1} \\
&= f(mn) - \frac{m-1}{m+1}(mn) + \frac{m-1}{m+1}(-1)^{(\log_m n)+1} \\
&= f(mn) + 2b_m(n) - 2b_m(n) - \frac{m-1}{m+1}(mn) + \frac{m-1}{m+1}(-1)^{(\log_m n)+1} \\
&= f(mn) + 2b_m(n) - \frac{m-1}{m+1}(n + (-1)^{(\log_m n)+1}) - \frac{m-1}{m+1}(mn) \\
&\quad + \frac{m-1}{m+1}(-1)^{(\log_m n)+1} \\
&= f(mn) + 2b_m(n) - n(m+1)\frac{m-1}{m+1} \\
&= f(mn) - (m-1)n + 2b_m(n).
\end{aligned}$$

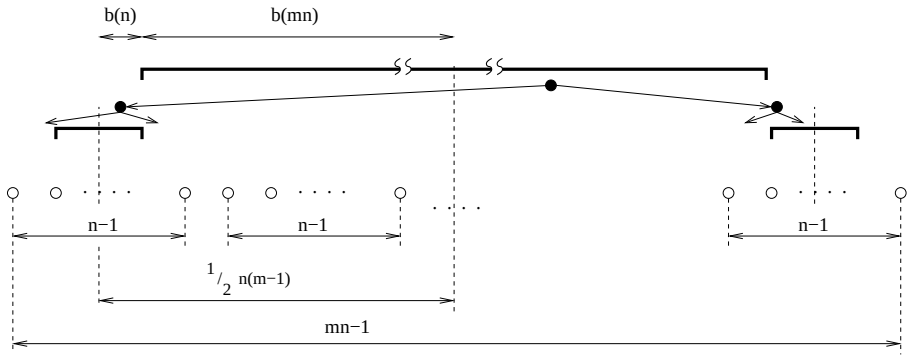


Figure 10.11: Bound regions of a tree. Filled circles are the root and its children and open circles are leaves of a tree. Bold lines indicate the bounded regions of the tree and its subtrees. These regions are directly adjoining.

∎

Observation 10.5.3c proves that the bound region $b(mn)$ of a tree of mn leaves is directly adjoining the bound region of its leftmost (respectively rightmost) subtree of n leaves as shown in Figure 10.11. This becomes clear if one takes into account that the distance between the midpoint of a tree with mn leaves and the midpoint of its leftmost (respectively rightmost) subtree is $\frac{n(m-1)}{2} = b_m(mn) + b_m(n)$ (as shown above we can assume that the leaves of the leftmost subtree are the n leftmost leaves of the tree). Now we are ready to give the inductive proof. Our hypothesis is that any layout for a complete tree of $n = m^k$ leaves that lie on a row of grid-points needs a horizontal wire length of at least $f(n) + \overline{x}$, where $\overline{x}$ is the limited deviation of the root. In the case of $n = 1$, the minimal wire length needed for a tree is the wire needed to route the only leaf to the position of the root, which is $|x|$. Our bound evaluates to $f(1) + \overline{x} = \overline{x} = \max\{0, |x| - 0\} = |x|$. For the induction step, we consider an arbitrary layout T for mn leaves. The root r of T is connected to m subtrees $T_1, \ldots, T_m$ of n leaves each.

Because we can assume that the tree is horizontal non-overlapping, there is a leftmost and a rightmost subtree, where leaves of the leftmost (respectively rightmost) subtree are the n leftmost (respectively rightmost) leaves of T. Let T_L be the leftmost subtree, r_L be its root, T_R be the rightmost subtree and r_R be its root. Furthermore let x_L denote the displacement of r_L , x_R denote the displacement of r_R, and x denote the displacement of r. Because of symmetry, we can assume without loss of generality that the root r is placed to the right of the midpoint. The situation is illustrated in Figure 10.12. We consider two cases, whether r is displaced by at most $b_m(mn)$ or by more than $b_m(mn)$.

In the first case we have $x \leq b_m(mn)$ and therefore $\overline{x} = 0$. The horizontal wire length of the tree is the sum of the wire lengths of the subtrees and the wires to

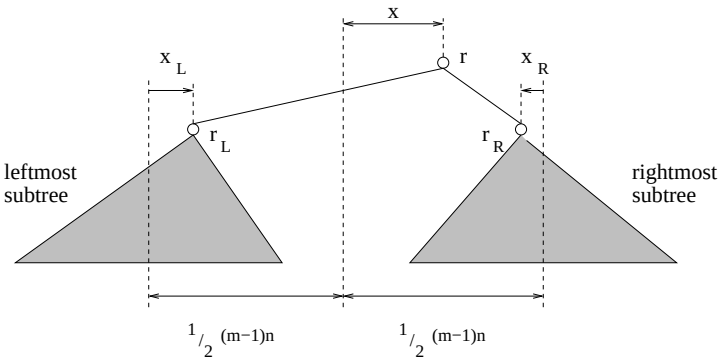


Figure 10.12: Inductive step: A m-ary tree with mn leaves. Only the leftmost and the rightmost subtree is shown. The root r is displaced by x from the midpoint. The r_L is displaced by x_L from its midpoint and r_R is displaced by x_R from its midpoint.

connect these trees. At least, there is a wire from the root to r_L and r_R.

$$\begin{aligned} HWL(T_{mn}) &\geq \sum_{i=1}^{m} HWL(T_i) + (m-1)n - x_L + x_R \\ &\geq mf_m(n) + \overline{x}_L + \overline{x}_R + (m-1)n - x_L + x_R \\ &\geq f_m(mn) - (m-1)n + 2b_m(n) + (m-1)n + \overline{x}_L - x_L + \overline{x}_R + x_R \\ &\geq f_m(mn) + 2b_m(n) - b_m(n) - b_m(n) \\ &\geq f_m(mn) \geq f(mn) + \overline{x}. \end{aligned}$$

In the second case, we have $x > b_m(mn)$ and therefore $\overline{x} > 0$. Suppose that $xpos(r) \leq xpos(r_R)$ (*i. e.* r is not to the right of r_R, see Figure 10.13a). Since $\overline{x} > 0$, it follows that $b_m(n) + x_r \geq \overline{x}$ (this is also shown in Figure 10.13a). Hence,

$$\begin{aligned} HWL(T_{mn}) &\geq mf_m(n) + \overline{x}_L + \overline{x}_R + (m-1)n - x_L + x_R \\ &\geq f_m(mn) + 2b_m(n) + \overline{x}_L - x_L + \overline{x}_R + x_R \\ &\geq f_m(mn) + b_m(n) + \overline{x}_R + x_R \\ &\geq f_m(mn) + b_m(n) + x_R \geq f_m(mn) + \overline{x}. \end{aligned}$$

Suppose that r is to the right of r_R. In Figure 10.13b, it is illustrated that the wire needed to connect r_R to r is $\overline{x} - (b_m(n) + x_R)$ long. The wire needed to connect r_L and r_R is as before $n(m-1) - x_L + x_R$ in length. Hence,

$$\begin{aligned} HWL(T_{mn}) &\geq \sum_{i=1}^{m} HWL(T_i) + n(m-1) - x_L + x_R + \overline{x} - (b_m(n) + x_R) \\ &\geq f_m(mn) + 2b_m(n) + \overline{x}_L - x_L + \overline{x}_R + x_R + \overline{x} - b_m(n) - x_R \\ &\geq f_m(mn) + b_m(n) + \overline{x}_L - x_L + \overline{x}_R + \overline{x} \geq f_m(mn) + \overline{x}. \end{aligned}$$

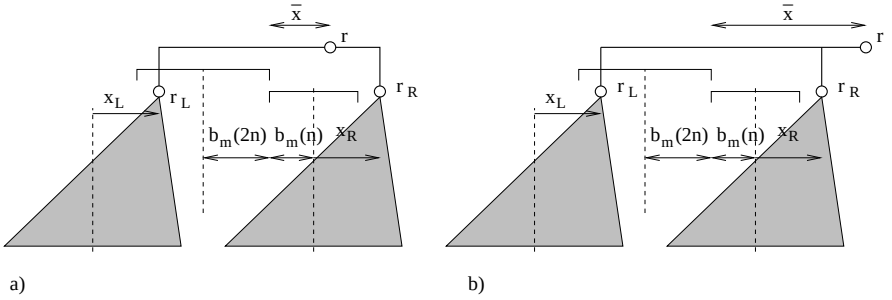


Figure 10.13: The root r is out of its bounded region. a) The root is not to the right of r_R. One can see that $b(n) + x_r \geq \overline{x}$. b) The root is to the right of r_R. The wire needed to connect r_R to r is $\overline{x} - (b(n) + x_R)$.

■

Hence, the layout in Section 10.4 is optimal with respect to the horizontal wire length. Furthermore, this shows that no layout for an m-ary tree can achieve for any $m \geq 2$ a total wire length $a \cdot n \cdot \log_m n + O(n)$ with $a < \frac{m-1}{m+1}$.

10.6 Discussion

We have exhibited in this chapter layouts of complete m-ary trees on a grid that are optimal with regard to their horizontal total wire length. Neither the construction of the optimal layout, nor the proof of its optimality, are obvious. One may interpret this as an indication that the construction of circuits with small total wire length does not only produce circuit architectures that are more interesting from the point of view of physical implementations, but also that this new circuit complexity measure gives rise to a number of interesting new theoretical problems.

Chapter 11

On the Complexity of Routing in VLSI-Circuits

As discussed in the previous chapters, the problem of routing in biological circuits is closely related to the problem of routing in VLSI models. Most problems related to routing are known to be computationally intractable. A basic and rigurously studied routing model is the *channel routing* model. The exact complexity of one problem in channel routing, the *knock-knee channel routing problem* was not known for a long time. In this Chapter, this gap is closed by showing that the "easiest" variation of the problem not known to be solvable in polynomial time is NP-complete. This chapter is based on the publication (Legenstein, 2001).

11.1 Introduction

The first lesson one has to learn if one is concerned with VLSI routing is that optimal routing is extremely difficult. Most routing problems are known to be "very hard", namely NP-complete. For NP-complete problems, no sequential algorithm is known that has execution time polynomial in the number of inputs. Althoug not proved, it is believed that any algorithm for an NP-complete problem needs super-polynomial execution time. For the model of parallel processing, this implies that circuits computing such a function have super-polynomial size. Such problems are said to be "intractable", since an optimal solution for even medium since problems is computationally too demanding. In this chapter, we are concerned with a problem arising in VLSI design, the *cannel routing problem*. We show that cannel routing with 3-terminal nets is NP-complete, thereby improving a result from (Sarrafzadeh, 1987). In this chapter, basic knowledge of proof techniques for NP-complete problems (reduction) is assumed to be familiar to the reader. For an introduction to NP-completeness, see (Garey et al., 1979).

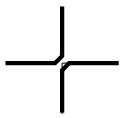


Figure 11.1: A knock-knee. Two nets bend at a grid vertex.

11.2 The Channel Routing Problem

The channel routing problem arises in the design process of VLSI circuits. A channel is a rectangular grid with top and bottom boundaries. Terminals are grid points located on the upper or lower boundary of the grid, and must be connected via wires. A k-terminal net is a set of k such terminals. The channel-routing problem can be described as follows: For a set of nets, find a set of edge-disjoint subgraphs of the grid connecting the terminals of each net, while minimizing the number of horizontal lines (tracks). Often, the number of terminals of the nets is restricted. The routing models mostly considered in the literature are Knock-knee routing (see, e.g. (Kuchem et al., 1996)) and Manhattan routing (see, e.g. (Middendorf, 1996)). A knock-knee is shown in Figure 11.1. At a knock-knee, two nets bend at a grid-vertex. Such a routing is allowed in the knock-knee model, but not in the Manhattan model. This paper is concerned with knock-knee routing. Middendorf showed that Manhattan channel-routing is NP-complete for 2-terminal nets even if all nets are single sided (*i. e.* both terminals of a net are either on the top or on the bottom boundary) and the bottom nets have density one (Middendorf, 1996). The proof of Middendorf also inspired the proof of the result in this paper. Hence Manhattan routing is harder than knock-knee routing (unless $P = NP$), since it is well known that knock-knee routing is solvable in polynomial time if only 2-terminal nets are involved (Frank, 82), (Formann et al., 1993). On the other hand, it was shown that knock-knee channel routing with 5-terminal nets is NP-complete (Sarrafzadeh, 1987), too. This paper closes the gap left open by these results by showing that knock-knee channel-routing is NP-complete for 3-terminal nets.

We start by introducing some notation and another channel routing problem in Section 11.3. In Section 11.4, it is shown that this problem is NP-complete. This result can be used directly to show the main result of the paper.

11.3 Preliminaries

In contrast to a channel described in Section 11.2, a channel with a right boundary is a rectilinear grid that has boundaries at three sides, the top boundary, the bottom boundary and the right boundary. The horizontal grid lines between top

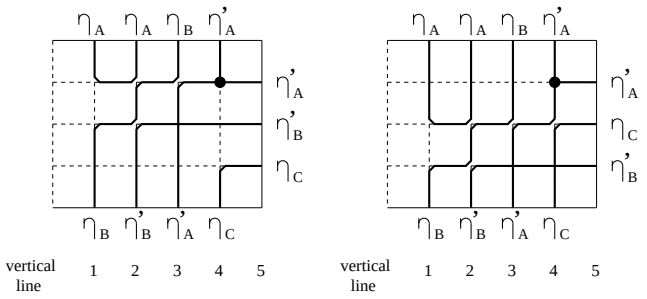


Figure 11.2: Two possible routings for supernets $\mathcal{N}$ as given in Example 11.3.1.

and bottom boundaries are called tracks. They are numbered $1, \ldots, k$ from the top track to the bottom track. The vertical grid lines are numbered from left to right, with the right boundary at the vertical line p.

A *terminal* is defined by a grid point on the boundary. No two terminals can be on the same grid point. In this model, terminals on the right boundary are movable in the vertical direction, *i. e.* if a terminal is specified to lie on the right boundary, the horizontal position can be chosen freely. We write t_i for a terminal that lies on the i-th vertical line and the top boundary, b_i for a terminal that lies on the i-th vertical line and the bottom boundary (for $i \in \{1, \ldots, p-1\}$ and r for a terminal that lies on the right boundary.

A *n-terminal net* η is a n-tuple of different terminals $\eta = (a_1, \ldots, a_n)$ where a_i is on a vertical line smaller or equal to the vertical line of a_{i+1} for $i = 1, \ldots, n-1$ (*i. e.* $a_i \in \{t_{tr_i}, b_{tr_i}, r\}$, $1 \leq tr_i \leq tr_{i+1} \leq p-1$, and $a_i \neq r$ for $i < n$). We say that a net $\eta = (a_1, \ldots, a_n)$ has its first (respectively last) terminal at vertical line l if a_1 (respectively a_n) is a terminal at vertical line l. We will often use the term "net" for a n-terminal net and omit the prefix. All nets considered in this paper are at most 3-terminal nets.

A *supernet* N is a set of nets where at most one net η in the set has a terminal at the right boundary. We say that a supernet N terminates on the right boundary if and only if there exists a net in N that has a terminal on the right boundary. A supernet N is a n-terminal supernet if all its nets are at most n-terminal nets.

Let $\mathcal{N}$ be a set of supernets for a channel with k tracks and a right boundary at vertical line p. A *routing* for $\mathcal{N}$ is an arrangement of routing paths in the channel for all the nets contained in the supernets in $\mathcal{N}$ with respect to the knock-knee model.

Example 11.3.1 *A channel with three tracks and a right boundary at vertical line 5 is given. Consider the set $\mathcal{N} = \{N_A, N_B, N_C\}$ of supernets where $N_A = \{\eta_A, \eta'_A\}$ with $\eta_A = \{t_1, t_2\}$ and $\eta'_A = \{b_3, t_4, r\}$, $N_B = \{\eta_B, \eta'_B\}$ with $\eta_B = \{b_1, t_3\}$ and $\eta'_B = \{b_2, r\}$, and $N_C = \{\eta_C\}$ with $\eta_C = \{b_4, r\}$. In Figure 11.2, two possible*

routings for $\mathcal{N}$ in this channel are shown.

We can now formulate our problem.

Definition 11.3.1 (knock-knee channel routing with right boundary)

Instance *Given a triple $I = (k, p, \mathcal{N})$ with integers k,p and a set $\mathcal{N} = \{N_1, N_2, \ldots, N_k\}$ of k 3-terminal supernets for a channel with k tracks and a right boundary at column p.*

Question *Is there a routing for I?*

We refer to this problem as KKRB.

The segment between two vertical lines i and $i+1$ is called *column* $i^{\rightarrow}$ or $(i+1)^{\leftarrow}$. For an instance of KKRB, the *density* of a column $i^{\rightarrow}$ (local density) is the number of nets with at least one terminal to the left (including vertical line i) and at least one terminal to the right (including vertical line $i+1$) of column $i^{\rightarrow}$. The density d of the instance (global density) is the maximum of all local densities.

For some routing R, we say that the net η is on track i in column $j^{\rightarrow}$ if track i is used by net η in column $j \rightarrow$ in R. Note that a net may use several tracks in a column. We say that a supernet N is on track i in column $j^{\rightarrow}$ if some net $\eta \in N$ is routed on track i in column $j^{\rightarrow}$. Furthermore, we say that a net η changes its track at vertical line l if η is on some track i in column $l \leftarrow$ and η is on some track $j \neq i$ in column $l \rightarrow$. We give some easy but useful observations on knock-knee channel-routings.

Observation 11.3.1 *Consider a knock-knee channel routing with k tracks. Let l be a vertical line in this channel (see Figure 11.3). If columns $l \leftarrow$ and $l \rightarrow$ have density k,*

a and no net has its last terminal at l, then no net changes its track at l.

b a net η_1 has a top terminal, and a net $\eta_2 \neq \eta_1$ has a bottom terminal at l and neither η_1 nor η_2 has a last terminal at l, then η_1 is routed at a track t_1 in columns $l \leftarrow$ and $l \rightarrow$ and η_2 is routed at a track t_2 in columns $l \leftarrow$ and $l \rightarrow$ with $t_1 < t_2$.

Proof: To show Observation 11.3.1a, suppose that a net η is on some track t_1 in column $l \leftarrow$ and on some track t_2 in column $l \rightarrow$ with $t_1 \neq t_2$ (see Figure 11.3a). Suppose that $t_2 > t_1$. Because both columns have full density, η uses exactly one horizontal grid edge in each of these columns. Hence, η uses the vertical grid edges between t_1 and t_2. Hence, all nets that are below t_2 in column $l \leftarrow$, are on tracks below t_2 in column $l \rightarrow$. Denote with η' the net that is on track t_2 in

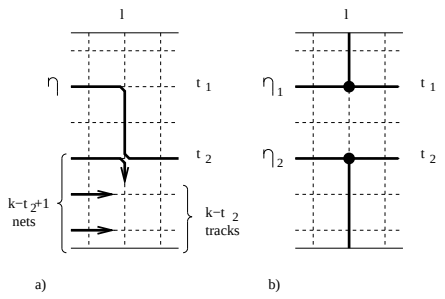


Figure 11.3: Three situations in a channel with full density around l. a) The situation of Observation 1. The given routing of η leads to a contradiction. b) The situation of Observation 3a. η_1 is routed above η_2 at columns $l \leftarrow$ and $l \rightarrow$ in any routing.

column $l \leftarrow$. If η has a bottom terminal at vertical line l, there is no way to route η'. If η has no bottom terminal at l, η' is routed at some track below t_2. Then there are $k - t_2 + 1$ nets routed below t_2, but only $k - t_2$ tracks below t_2. This leads to a contradiction. The proof for $t_2 < t_1$ is similar.

The situation of Observation 11.3.1b is shown in Figure 11.3b. In column $l \leftarrow$, η_1 is on track t_1 and η_2 is on track t_2. Because of the full density, both nets use exactly one horizontal grid edge in $l \leftarrow$ and exactly one in $l \rightarrow$. Because of Observation 11.3.1a, these nets do not change their tracks at l. Hence, the vertical grid edges from t_1 to the top boundary and from t_2 to the bottom boundary are used by η_1 and η_2, and it follow that $t_1 < t_2$. ∎

Let R be a routing for an instance $I = (k, p, \{N_1, \ldots, N_k\})$ of KKRB. For some column c, let N^c be the set of nets that are routed in column c by R *i. e.* $N^c = \{\eta | \eta \in \bigcup_{i=1}^{k} N_i$ and η is routed in column c by $R\}$. On each column c of I with full density, we define a function $tr_R^c : N^c \rightarrow \{1, \ldots, k\}$, such that for some $\eta \in N^c$, $tr_R^c(\eta)$ is the index of the track where η is routed in column c of routing R. We omit the column in the superscript if c is the last column of I (*i. e.* the column to the left of the right boundary of I). The function is defined in a similar manner for supernets of I. For some column c, let $\mathcal{N}^c$ be the set of supernets that are routed in column c by R. On each column c of I with full density, we define a function $tr_R^c : \mathcal{N}^c \rightarrow \{1, \ldots, k\}$, such that for some $N \in N^c$, $tr_R^c(N) = min\{tr_R^c(\eta) | \eta \in N$and η routed in column c by $R\}$. We take the minimum of because in our definition, a supernet could have several nets with terminals to both sides of a given column. However, we will avoid such situations in this chapter.

For some routing R of an instance I of KKRB we say that a net η makes a *detour* in column $p \rightarrow$ if *eta* uses at least two horizontal grid-edges in column $p \rightarrow$ and either η has no terminal to the right of $p \rightarrow$ (including vertical line $p + 1$)

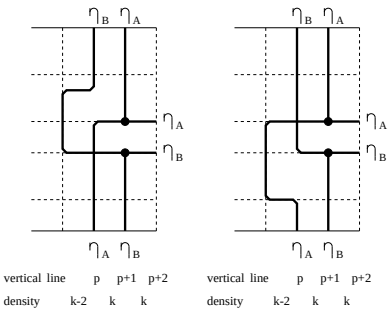


Figure 11.4: A channel portion described in Observation 11.3.2. Exactly one of the nets η_A, η_B makes a detour to the left at vertical line p. In the left figure η_B makes this detour, in the right figure η_A makes this detour.

or no terminal to the left of $p \rightarrow$ (including vertical line p). In the former case we also say that η makes a detour to the right at vertical line p and in the latter case we also say that η makes a detour to the left at vertical line $p+1$. In the left figure of Figure 11.4 η_B makes a detour to the left at vertical line p.

Observation 11.3.2 *Consider an instance $I = (k, q, \mathcal{N})$ of KKRB with two nets of the following form: For $3 \leq p \leq q-2$ and terminals a, b at vertical lines to the right of $p+1$, let $\eta_A = (b_p, t_{p+1}, a)$ and $\eta_B = (t_p, b_{p+1}, b)$. Furthermore let the density of $p \leftarrow$ be $k-2$ and the densities of $p \rightarrow$ and $(p+1) \rightarrow$ be k (see Figure 11.4). For any routing R for I it holds that*

a $tr_R^{(p+1)\rightarrow}(\eta_B) > tr_R^{(p+1)\rightarrow}(\eta_A)$,

b at vertical line p, exactly one of the nets η_A, η_B makes a detour to the left and no other net makes a detour in column $p \leftarrow$, and

Proof: From Observation 11.3.1b it follows that $tr_R p \rightarrow(\eta_A) = tr_R(p+1) \rightarrow(\eta_A) < tr_R(p+1) \rightarrow(\eta_B) = tr_R p \rightarrow(\eta_B)$ which implies Observation 11.3.2a. Since the density of $p \rightarrow$ is k, each of these nets uses exactly one horizontal grid-edge in this column and none makes a detour to the right. Suppose that none makes a detour to the left. Then the vertical grid-edges between $tr_R p \rightarrow(\eta_A)$ and $tr_R p \rightarrow(\eta_B)$ are used by both nets which contradicts the definition of a layout. Hence, at least one of the nets makes a detour to the left. Since the density at column $p \leftarrow$ is $k-2$ and this net uses two horizontal gird-edges at this column, k horizontal grid-edges are used and no other net in I makes a detour at column $p \leftarrow$ which shows Observation 11.3.2b. ∎

Let $I = (k, p, \mathcal{N})$ be an instance of KKRB. An extension of I is an instance $I' = (k, q, \mathcal{N}')$ with $q > p$ and $\mathcal{N}' = \{N_1', N_2', \ldots, N_k'\}$ such that for all $i \in \{1, \ldots, k\}$, the following holds:

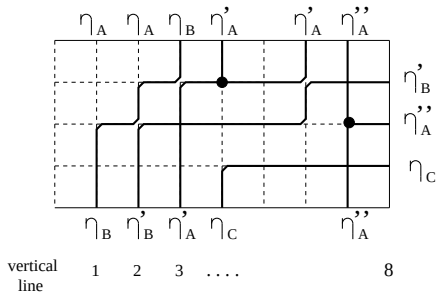


Figure 11.5: A routing for extension I' of Example 11.3.2.

1. For all nets of the form $(a_1, \ldots, a_n)$ without a terminal on the right boundary (*i. e.* $a_n \neq r$), we have $(a_1, \ldots, a_n) \in N_i \Rightarrow (a_1, \ldots, a_n) \in N_i'$.

2. If N_i contains a net of the form $(a_1, \ldots, a_{n-1}, r)$, then N_i' contains a net of the form $(a_1, \ldots, a_{n-1}, b)$ where b is a terminal on any of the three boundaries.

To avoid cumbersome notation, we will denote the set of supernets for an instance I of KKRB and an extension I' of I with the same character. The same will be done for the corresponding supernets and the corresponding nets contained in the supernets.

Example 11.3.2 *Consider an instance $I = \{3, 5, \mathcal{N}\}$ of KKRB where $\mathcal{N}$ is identical to the one defined in Example 11.3.1. Let $I' = \{3, 8, \mathcal{N}\}$ where $\mathcal{N} = \{N_A, N_B, N_C\}$, N_B and N_C are defined as in Example 11.3.1, and $N_A = \{\eta_A, \eta_A', \eta_A''\}$ with $\eta_A = \{t_1, t_2\}, \eta_A' = \{b_3, t_4, t_6\}$, and $\eta_A'' = \{t_7, b_7, r\}$. I' is an extension of I. In Figure 11.5, a routing for I' is shown.*

With the help of extensions, we will force specific properties for the routing of some supernets, e.g. that a supernet is routed above another one in any routing. We will want that in any routing, specific nets do not change their track within the portion of the channel added by an extension. This leads to the following definition. An extension $I' = (k, q, \mathcal{N}')$ of an instance $I = (k, p, \mathcal{N})$ is *$\mathcal{M}$-safe* for I, $\mathcal{M} \subseteq \mathcal{N}$ if for every routing for I' and each $N \in \mathcal{M}$ the supernet N is on track i in column $q^{\leftarrow}$ if and only if N is on track i in column $p^{\leftarrow}$.

Consider an instance I of KKRB with supernets A and B that terminate on the right boundary. The following lemma states that there exists an extension I' that enforces B to be routed below A on the right boundary.

Lemma 11.3.3 *Let $I = (k, p, \mathcal{N})$ be an instance of KKRB where all k supernets in $\mathcal{N}$ terminate on the right boundary. Then, for each two different supernets $A, B \in \mathcal{N}$, there exists an extension $I' = (k, p + 4, \mathcal{N})$ of I such that:*

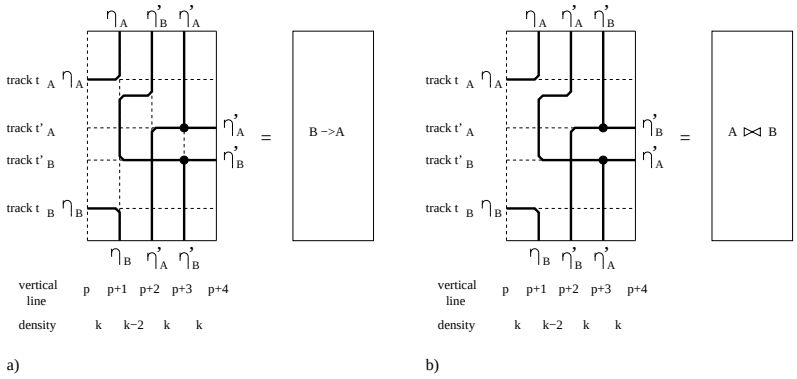


Figure 11.6: A layout for the supernets $A = \{\eta_A, \eta'_A\}$ and $B = \{\eta_B, \eta'_B\}$ in two extensions. Note that A is routed on a track above B at column $\leftarrow$ in any routing. a) In this extension, A is routed above B on the right boundary. b) In this extension, A is routed below B on the right boundary. Note that in (b), simply the net η'_A is exchanged with the net η'_B.

1. *all supernets in I' terminate on the right boundary,*

2. *there exist a routing R' for I' if and only if there exists a routing R for I with $tr_R(B) > tr_R(A)$.*

3. *If there exists a routing R' for I', then R' has the following properties:*

   (a) $tr_{R'}(B) > tr_{R'}(A)$

   (b) $tr_{R'}(B) \leq tr^{p\leftarrow}_{R'}(B)$

   (c) $tr_{R'}(A) \geq tr^{p\leftarrow}_{R'}(A)$.

4. *If in some routing R' for I' $tr_{R'}(A) = tr^{p\leftarrow}_{R'}(A)$ and $tr_{R'}(B) = tr^{p\leftarrow}_{R'}(B)$, then for each supernet $N \in \mathcal{N}$ it holds that $tr_{R'}(N) = tr^{p\leftarrow}_{R'}(N)$.*

Proof: Let $\mathcal{N}$ of I' be such that the supernets in $\mathcal{N} - \{B, A\}$ are not modified (note that supernets that terminate on the right boundary of I now terminate on the right boundary of I'). The supernets B and A are modified as shown in Figure 11.6a: The right boundary terminal net (a_1, a_2, r) in B is replaced by a net $\eta_B = (a_1, a_2, b_{p+1})$. Furthermore we add a 3-terminal net $\eta'_B = (t_{p+2}, b_{p+3}, r)$ to B. The right boundary terminal net (a_1, a_2, r) in A is replaced by a net $\eta_A = (a_1, a_2, t_{p+1})$. Furthermore we add a 3-terminal net $\eta'_A = (b_{p+2}, t_{p+3}, r)$ to A. By construction, Condition 1 holds.

The layout of Figure 11.6a proves that there is a layout for I' if the Condition 2 is met. Furthermore, this routing satisfies Condition 3. The equalities can be achieved by setting $t_A = t'_A$ and $t_B = t'_B$. Note that there is no terminal at

vertical line p by definition of an instance of KKRB. Hence, no net changes its track at p.

We show that such a layout exists only if Conditions 2 and 3 are met. Clearly, there is no routing for I' if there is no routing for I. Denote the track of η_A in column $p \rightarrow$ with t_A and the track of η_B in column $p \rightarrow$ with t_B. Furthermore, denote the track of η'_A in column $(p+3) \rightarrow$ with t'_A and the track of η'_B in column $(p+3) \rightarrow$ with t'_B. The local densities of the instance are given in Figure 11.6a. By Observation 11.3.1b, $t'_A < t'_B$ follows. Hence Condition 3a holds for any routing for I'. Furthermore, $tr_{R'}^{(p+2)\rightarrow}(\eta'_A) = t'_A < t'_B = tr_{R'}^{(p+2)\rightarrow}(\eta'_B)$ for any routing for R' for I' by Observation 11.3.1b.

By Observation 11.3.2, a net $\eta_{left} \in \{\eta'_A, \eta'_B\}$ makes a detour to the left at vertical line $p+1$ and neither η_A nor η_B makes a detour to the right at vertical line p. Furthermore by the full density in column $p \rightarrow$, η_{left} uses all vertical grid-edges between its horizontal grid-edges in $(p+1) \rightarrow$ at vertical line $p+1$. It follows that in any routing it holds that $t_A \leq t'_A < t'_B \leq t_B$. This shows Conditions 2 and 3.

Consider a routing R' with $tr_{R'}(A) = tr_{R'}^{p\leftarrow}(A)$ and $tr_{R'}(B) = tr_{R'}^{p\leftarrow}(B)$. Proposition 4 holds for these supernets by definition. Suppose that η'_B makes the detour at vertical line $p+2$. Then η'_A makes no detour, is on track t_A in column $(p+2) \rightarrow$ and uses all vertical grid-edges below t_A at vertical line $p+2$. Furthermore η'_B uses all vertical grid-edges above t_A, a horizontal grid-edges at t_A and t_B in $(p+2) \leftarrow$ and all vertical grid-edges between. Hence all vertical grid-edges are used at vertical lines $p+1$ and $p+2$ and no net changes its track at these vertical lines. Furthermore no net changes its track at vertical line $p+3$ by Observation 11.3.1a. ∎

Remark 11.3.4 *From Conditions 3 and 4 it follows that if $tr_R(B) = tr_R(A)+1$ in any routing R for I, then I' is $\mathcal{N}$-safe for I. In other words, if one understands the extension merely as some part of the channel, then under these conditions, for all routings R and supernets $N \in \mathcal{N}$ it holds that $tr_R^{(p+4)\leftarrow}(N) = tr_R^{p\leftarrow}(N)$.*

The following lemma is very similar to Lemma 11.3.3. The difference in the extension is, that we enforce that A and B change their order within the extension, *i. e.* A is routed below B after the extension.

Lemma 11.3.5 *Let $I = (k, p, \mathcal{N})$ be an instance of KKRB where all k supernets in $\mathcal{N}$ terminate on the right boundary. Then, for each two different supernets $A, B \in \mathcal{N}$, there exists an extension $I' = (k, p+4, \mathcal{N})$ of I such that:*

1. *all supernets in I' terminate on the right boundary,*

2. *there exist a routing R' for I' if and only if there exists a routing R for I with $tr_R(B) > tr_R(A)$.*

3. *If there exists a routing R' for I', then R' has the following properties:*

   (a) $tr_{R'}(B) < tr_{R'}(A)$

   (b) $tr_{R'}(A) \leq tr^{p\leftarrow}_{R'}(B)$

   (c) $tr_{R'}(B) \geq tr^{p\leftarrow}_{R'}(A)$.

4. *If in some routing R' for I', $tr_{R'}(A) = tr^{p\leftarrow}_{R'}(B)$ and , $tr_{R'}(B) = tr^{p\leftarrow}_{R'}(A)$ then for each supernet $N \in \mathcal{N} - \{A, B, \}$ it holds that $tr_{R'}(N) = tr^{p\leftarrow}_{R'}(N)$.*

Proof: Let $\mathcal{N}$ of I' be such that the supernets in $\mathcal{N} - \{B, A\}$ are not modified. The supernets B and A are modified as shown in Figure 11.6b: The right boundary terminal net (a_1, a_2, r) in B is replaced by a net $\eta_B = (a_1, a_2, b_{p+1})$. Furthermore we add a 3-terminal net $\eta'_B = (b_{p+2}, t_{p+3}, r)$ to B. The right boundary terminal net (a_1, a_2, r) in A is replaced by a net $\eta_A = (a_1, a_2, t_{p+1})$. Furthermore we add a 3-terminal net $\eta'_A = (t_{p+2}, b_{p+3}, r)$ to A. By construction, Condition 1 holds. Note that with respect to the extension of Lemma 11.3.3, we simply exchanged the nets η'_B and η'_A. Simply by exchanging η'_B with η'_A in the proof of Lemma 11.3.3, Conditions 2 and 3 can be shown. This is also true for Proposition 4 as long as the nets in $\mathcal{N} - \{A, B\}$ are concerned. Note that in this case, A and B exchange their tracks within the extension which follows easily from the previous conditions on any routing R' for I'. ∎

Remark 11.3.6 *From Conditions 3 and 4 it follows that if $tr_R(B) = tr_R(A) + 1$ in any routing R for I, then I' is $\mathcal{N} - \{A, B\}$-safe for I.*

As a consequence of Lemma 11.3.3, we can enforce a particular order of supernets on the right boundary.

Lemma 11.3.7 *For each k, there is an instance $I = (k, p, \mathcal{N})$ of KKRB such that:*

1. *all k supernets terminate on the right boundary, and*

2. *In every routing for I, supernet N_i terminates on track i on the right boundary.*

Proof: Define an instance $I_0 = (k, k+1, \mathcal{N})$ of KKRB where $N_i = \{(t_i, r)\}$. Any order of the supernets on the right boundary can be routed in this instance. Now we extend I_0 several times using the extension of Lemma 11.3.3. We will use $k-1$ extension steps, where the ith step extends I_{i-1} to I_i $(i = 1, \ldots, k-1)$.

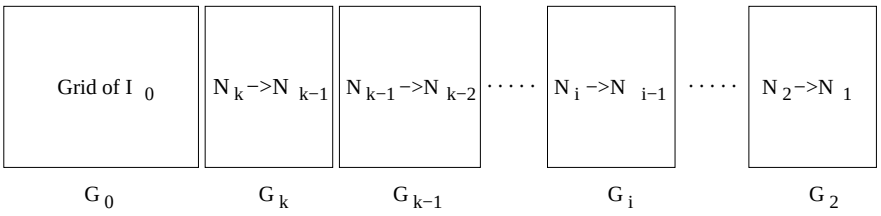


Figure 11.7: An instance I_{k-1} of KKRB where N_i terminates on track i in any routing. This is done with the extensions of Lemma 11.3.3. We divide the grid of I_{k-1} into k regions.

In the ith step, we enforce that N_{k-i+1} is below (higher track-index) N_{k-i} on the right boundary in any routing. For convenience, we divide the grid of I_{k-1} into k regions called G_0 and $G_2, \ldots, G_k$ (see Figure 11.7). G_0 is the grid defined by I_0, and G_i is the portion that was added by the $(k+1-i)$-th extension[1]. We show that in I_{k-1} (the last extension in our construction), N_i is at track i in the first column of G_i.

We show by induction on i that the supernet N_i is on a track with index i or higher in the first column of G_i. This serves as the induction hypothesis. Clearly, N_1 cannot be on a track with a lower index than 1 which proves the induction basis. Suppose that for some $i \in \{2, \ldots, k\}$, N_i is on some track $t_0 < i$ in the first column of G_i (and therefore in the last column of G_{i-1}). It follows by Lemma 11.3.3 that in the last column of G_i, N_{i-1} is on a track $t_1 < t_0 \leq i-1$. Hence, N_{i-1} is on a track $t_1 < i-1$ in the first column of G_{i-1}, which is a contradiction.

We show by induction on the extensions and by the properties given in Lemma 11.3.3 that for $i = 2, \ldots, k$, in any routing, N_i is on a track with an index lower than or equal to i in the first column of G_i. Clearly, N_k cannot be on a track with index larger than k in the first column of G_k which is our induction basis. Note that the basis is at k and we conclude from i to $i+1$ in the induction step. Suppose the N_i is on a track $t_0 > i$ in the first column of G_i. In the last column of G_{i+1}, N_i is on track t_0. It follows from Lemma 11.3.3 that N_{i+1} is on a track $t_1 > t_0 \geq i+1$ in the first column of G_{i+1} which is a contradiction.

Now, consider a region G_i with $i \in \{2, \ldots, k-1\}$. In the first column of this region, N_i is routed at track i and in the first column of G_{i+1}, N_{i+1} is routed at track $i+1$. By Lemma 11.3.3, G_{i+1} is also routed at track $i+1$ in the first column of G_i. By Remark 11.3.4, the track of N in the first column of G_i is the same as the track of N in the last column of G_i for all nets $N \in \mathcal{N}$. It follows that N_i is on track i on the right boundary of I. ∎

[1] G_i spans the columns $(6k+1-5i) \rightarrow$ to $(6k-5(i-1)) \rightarrow$ for $i = 2, \ldots, k$.

We will need another type of extension. In Lemma 11.3.7, we used the extension of Lemma 11.3.3 to enforce a particular order on the right boundary of the channel. But Lemma 11.3.3 cannot be used to enforce a supernet to be above another one without an influence on the other supernets in the channel. We use a trick to get a similar result. For each net N_i, introduce another net $\overline{N}_i$, which we will call the *shadownet* of N_i. Typically, $\overline{N}_i$ will be on a neighboring track of N_i. We say that a supernet $\overline{N}$ is a shadownet of N in an instance I of KKRB, if in every routing R for I, it holds that $tr_R(\overline{N}) = tr_R(N) + 1$.

We say that an instance I of KKRB is (A, B)-*mutable* for two supernets A, B, if for any routing R for I there is a routing R' for I, such that A and B change their tracks with respect to R on the right boundary. More formally, an instance $I = (k, p, \mathcal{N})$ of KKRB is (A, B)-*mutable* for two supernets $A, B \in \mathcal{N}$, if the following holds:

- If R is a routing for I, then there exists a routing R' for I such that $tr_{R'}(A) = tr_R(B)$ and $tr_{R'}(B) = tr_R(A)$.

Lemma 11.3.8 *Let $I = (k, p, \mathcal{N})$ be an instance of KKRB where all k supernets in $\mathcal{N}$ terminate on the right boundary. Then, for each four different nets $A, \overline{A}, B, \overline{B} \in \mathcal{N}$ where $\overline{A}$ is a shadownet of A and $\overline{B}$ is a shadownet of B in I, there exists an extension $I' = (k, p + 11, \mathcal{N})$ of I such that:*

1. *all supernets in I' terminate on the right boundary,*
2. *a routing exists for I' if and only if a routing R exists for I with $tr_R(A) < tr_R(B)$,*
3. *I' is (A, B)-mutable,*
4. *I' is $\mathcal{N} - \{A, \overline{A}, B, \overline{B}\}$-safe for I, and*
5. *$\overline{A}$ is a shadownet of A in I' and $\overline{B}$ is a shadownet of B in I'.*

Proof: The supernets A, $\overline{A}$,B and $\overline{B}$ are extended as shown in Figure 11.8. The nets η_A and η'_A belong to supernet A, the nets η_B and η'_B belong to supernet B, the nets $\eta_{\overline{A}}$ and $\eta'_{\overline{A}}$ belong to supernet $\overline{A}$, and the nets $\eta_{\overline{B}}$, $\eta'_{\overline{B}}$ belong to supernet $\overline{B}$. Other supernets are not altered. Condition 1 is true by construction. First we extend I by exchanging the tracks of B and $\overline{B}$. Since these nets are shadownets, this step is $\mathcal{N} - \{B, \overline{B}\}$-safe for I and the right boundary is at vertical line $q = p + 4$ after this extension. Clearly, the relative order of A and B does not change by this extension.

In order to simplify the notation, we will introduce the following abbreviations: For a routing R' of I' and a supernet $N \in \{A, B, \overline{A}, \overline{B}\}$ we write t_N for $tr^{q\rightarrow}_{R'}(N)$ and t'_N for $tr_{R'}(N)$. Note that in this notational convention, t_N corresponds to

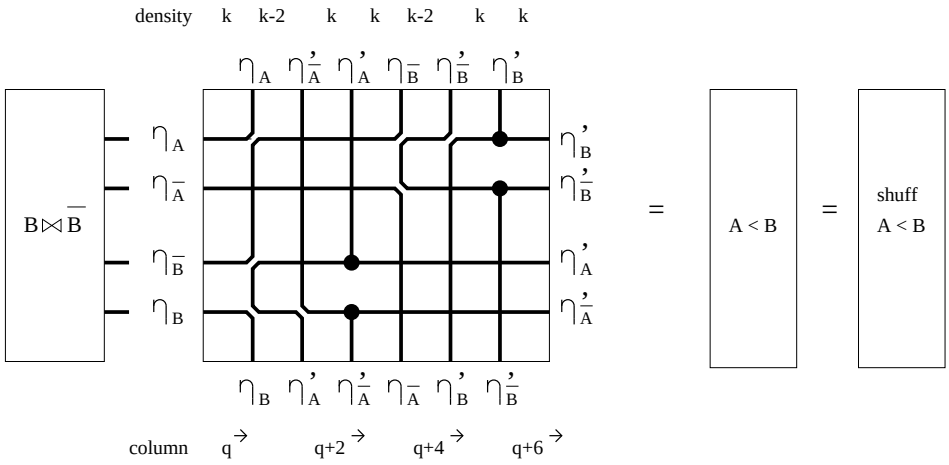


Figure 11.8: The shuffle-check extension and a possible routing.

the track of η_N at column $q \rightarrow$ and t'_N corresponds to the track of η'_N at column $(q+6) \rightarrow$. First, consider the columns $(q+3) \rightarrow, \ldots, (q+6) \rightarrow$. This part of the channel is very similar to the extension of Lemma 11.3.3. We can show that

- $t'_{\overline{B}} > t'_B$,
- there is no routing R' such that $tr_{R'}^{(q+3)\rightarrow}(\overline{B}) > tr_{R'}^{(q+3)\rightarrow}(\overline{A})$, and
- if in any routing R', $tr_{R'}^{(q+3)\rightarrow}(\overline{A}) = tr_{R'}^{(q+3)\rightarrow}(\overline{B}) + 1$, then no net except $\eta_{\overline{A}}$, η'_B and $\eta'_{\overline{B}}$ changes its track in these columns.

This can be easily shown by replacing the names of the respective nets in the proof of Lemma 11.3.3.

Now consider the columns $q \rightarrow, \ldots, (q+3) \rightarrow$. This is also a similar channel-portion to the one in Lemma 11.3.3. So, $t_B > t_A$ in any routing for I'. There exists a routing under this condition, as shown in Figure 11.8. This shows Condition 2. No net can change its track at vertical line $q+3$ by Observation 11.3.1a. Hence, in any routing $\eta_{\overline{B}}$ is on a track above $\eta_{\overline{A}}$ at column $(q+2) \rightarrow$. Furthermore, by Observation 11.3.2, one of the nets of $\eta'_{\overline{A}}$ and η'_A makes a detour to the left at vertical line $q+2$, and this detour uses exactly two horizontal grid-edges at column $(q+1) \rightarrow$. We denote the upper horizontal grid-edge with t_u and the lower one with t_l. Because of the full density at column $q \rightarrow$, this detour uses the vertical grid-edges between t_u and t_l. By Observation 11.3.2b, neither η_A nor η_B makes a detour at vertical line $q+1$. Hence, $t_A \leq t_u < t_l \leq t_B$ holds for any routing.

Claim 11.3.1 *In any routing for I', it holds that $t_u \in \{t_A, t_{\overline{B}}\}$ and $t_l \in \{t_B, t_{\overline{A}}\}$.*

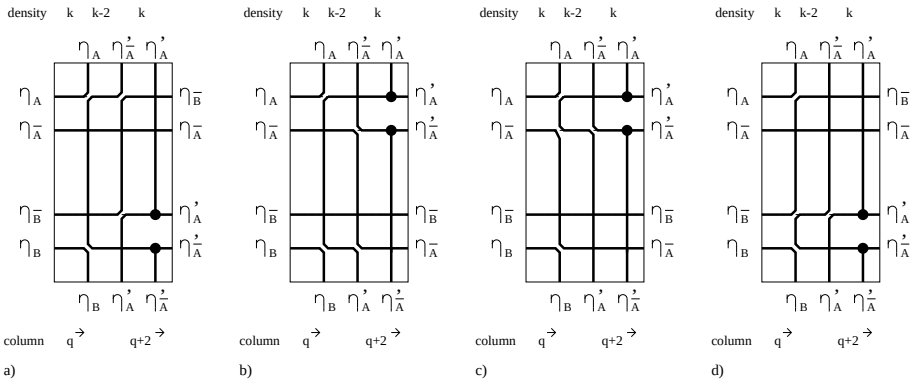


Figure 11.9: Four possible routings in columns $q \rightarrow, \ldots, (q+2) \rightarrow$ of the shuffle-check extension. We distinct three cases depending on the tracks that the detour uses at column $(q+1) \rightarrow$. a) Tracks t_A and t_B are used and $\eta'_{bar\,A}$ makes the detour. b) The same tracks as in (a) are used and η'_A makes the detour. c) Tracks t_A and $t_{bar\,A}$ are used. d) Tracks t_B and $t_{bar\,B}$ are used.

Proof(Claim 11.3.1): Suppose that $t_{\overline{A}} \leq t_u < t_{\overline{B}}$. Since the vertical grid-edges between t_u and t_l are used by the detour, $\eta_{\overline{A}}$ is on a track above t_u and $\eta_{\overline{B}}$ is on a track below $t_l$at column $(q+1) \rightarrow$. Furthermore, $\eta'_{\overline{A}}$ uses the vertical grid-edges above t_u at vertical line $q+2$ (no more detour is possible for $\eta'_{\overline{A}}$). Hence, $\eta_{\overline{A}}$ is above $\eta_{\overline{B}}$ in column $(q+2) \rightarrow$ which leads to a contradiction since there is no routing for this case. Hence, only the tracks t_A and $t_{\overline{B}}$ remain for t_u. A similar argument shows that $t_l \in \{t_B, t_{\overline{A}}\}$. ∎

By Claim 11.3.1, there are three cases to consider (by definition $t_u < t_l$):

Case 1: $t_u = t_A$ **and** $t_l = t_B$**.** Suppose that η'_A makes the detour at vertical line $q+2$ (see Figure 11.9a). No net of $\eta_{\overline{A}}$ and $\eta_{\overline{B}}$ changes its track at vertical line $q+1$. Since at vertical line $q+2$, $\eta_{\overline{A}}$ cannot change to a track below $\eta_{\overline{B}}$, $\eta_{\overline{B}}$ changes to track t_A in order to be above $\eta_{\overline{A}}$ at column $(q+2) \rightarrow$. Since η'_A is routed above $\eta'_{\overline{A}}$ at columns $(q+2) \rightarrow$, $(q+3) \rightarrow$, and the only possible horizontal grid-edge in this column above $\eta'_{\overline{A}}$ is at track $t_{\overline{B}}$, η'_A is routed on track $t_{\overline{B}}$ at columns $(q+2) \rightarrow$, $(q+3) \rightarrow$. Hence, at vertical line $q+3$, η'_A is on track $t_{\overline{B}}$ and $\eta'_{\overline{A}}$ is on track t_B. Furthermore, at this column, $\eta_{\overline{B}}$ is on track t_A and $\eta_{\overline{A}}$ is on track $t_{\overline{A}}$. It follows that in this case, $t'_B = t_A$, $t'_{\overline{B}} = t_{\overline{A}}$, $t'_A = t_{\overline{B}}$ and $t'_{\overline{A}} = t_B$ and the extension is safe for all nets without a terminal in the extended area.

Suppose that $\eta'_{\overline{A}}$ makes the detour at vertical line $q+2$ (see Figure 11.9b). By similar arguments, one can prove the tracks of the nets at column $(q+3) \rightarrow$ as shown in Figure 11.9b. It follows that in this case, $t'_B = t_{\overline{B}}$, $t'_{\overline{B}} = t_B$, $t'_A = t_A$ and $t'_{\overline{A}} = t_{\overline{A}}$ and the extension is safe for all nets without a terminal in the extended area.

Case 2: $t_u = t_A$ **and** $t_l = t_{\overline{B}}$**.** η'_A uses all vertical grid-edges below t'_A and hence no net can change its track at vertical line $q+2$ (see Figure 11.9c). $\eta_{\overline{B}}$ cannot change to a track above $\eta_{\overline{A}}$ at vertical line $q+1$. Hence $\eta_{\overline{A}}$ changes to track t_B at $q+1$. Hence, at vertical line $q+3$, η'_A is on track $t_{\overline{B}}$ and $\eta'_{\overline{A}}$ is on track t_B. Furthermore, at this column, $\eta_{\overline{B}}$ is on track t_A and $\eta_{\overline{A}}$ is on track $t_{\overline{A}}$. It follows that in this case, $t'_B = t_A$, $t'_{\overline{B}} = t_{\overline{A}}$, $t'_A = t_{\overline{B}}$ and $t'_{\overline{A}} = t_B$ and the extension is safe for all nets without a terminal in the extended area.

Case 3: $t_u = t_{\overline{A}}$ **and** $t_l = t_B$**.** $\eta'_{\overline{A}}$ uses all vertical grid-edges above t'_B and hence no net can change its track at vertical line $q+2$ (see Figure 11.9d). $\eta_{\overline{A}}$ cannot change to a track below $\eta_{\overline{B}}$ at vertical line $q+1$. Hence $\eta_{\overline{B}}$ changes to track t_A at $q+1$. Hence, the supernets are on tracks given at Figure 11.9d. It follows that in this case, $t'_B = t_{\overline{B}}$, $t'_{\overline{B}} = t_B$, $t'_A = t_A$ and $t'_{\overline{A}} = t_{\overline{A}}$ and the extension is safe for all nets without a terminal in the extended area.

■

We call this extension a shuffle-check extension. We use this extension in two different ways. We can use Condition 2 of the lemma to enforce that A is routed above B in column $p \leftarrow$ of I'. In this case we need to take care of Condition 3 since as a side-effect of the extension, supernets A and B together with their shadownets may change their tracks. In this case we denote a shuffle-check extension for supernets A and B with $A < B$.

On the other hand, we can use Condition 3 to make two routings for A and B possible. Namely that A and B stay on their track or that A exchanges its track with B. In this case we need to take care that there is a routing such that A is routed above B in column $p \leftarrow$. In this case we indicate the aim of the extension by labeling the symbol for the check-shuffle extension with "shuff" as shown in Figure 11.8.

Note that in any of the extensions introduced, nets with a right boundary are at most 3-terminal nets. Furthermore, if we introduce a last terminal in a net in an extension, no other terminal is introduced for that net in this extension. Hence, by starting with a net of the type described in Lemma 11.3.7 and extending this net with one of the extensions introduced in this Section, at most 3-terminal nets are used in the resulting instance.

11.4 The Main Theorem

We show that KKRB is NP-complete by reducing a known NP-complete problem to it. This result will be used to prove the complexity of knock-knee channel routing with 3-terminal nets.

Theorem 11.4.1 *KKRB is NP-complete.*

Proof: A nondeterministic algorithm can guess a routing and check if it is valid. If yes, output "yes", otherwise output "no". This can be done in polynomial time with the number of tracks and columns. So, the problem is in NP. To prove the completeness for NP, we reduce the *exactly-one-in-three* 3SAT problem to KKRB. Let a set $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of clauses, each of size 3 over a set $\Sigma = \{v_1, v_2, \ldots, v_n\}$ of variables, be an instance of exactly-one-in-three 3SAT. Without loss of generality, we can assume that no clause contains a negated literal (this restriction is known to be NP-complete (Garey et al., 1979)). The exactly-one-in-three 3SAT problem asks whether there is a truth assignment of the variables in Σ such that each clause in $\mathcal{C}$ contains exactly one true literal.

The idea of the proof is as follows. We begin by constructing an instance of KKRB. We divide the tracks of the channel into five consecutive groups $G_1, \ldots, G_5$. Tracks in G_i are above tracks in G_{i+1} for $i \in \{1, \ldots, 4\}$. For each clause $C_l = \{v_h, v_i, v_j\}$, we introduce three supernets V_h^l, V_i^l, and V_j^l that terminate on tracks in group G_2 on the right boundary in every routing. Furthermore, for each variable v_i we introduce two supernets H_i and L_i that terminate on tracks in group G_3 on the right boundary in every routing. Then we extend our instance and enforce that for each variable v_i, either H_i changes to a track in group G_1 or L_i changes to a track in G_5. This will give us a truth assignment for the variables. Furthermore, we force all supernets of the form V_i^l to change to a track of group G_4 if and only if for the corresponding variable v_i the supernet L_i is on a track in group G_5. In addition we require that exactly one of the three supernets V_h^l, V_i^l, and V_j^l for a clause C_l will change to a track in group G_4. Thus, there will be a routing if and only if there exists a truth assignment for the variables satisfying $\mathcal{C}$.

We formalize these ideas. Let $k = 8n + 10m$ be the number of tracks. Recall that n is the number of variables and m is the number of clauses in the reduced problem. We divide the channel into the five groups $G_1 = \{\text{track } i | i \in \{1, \ldots, 2n\}\}$, $G_2 = \{\text{track } i | i \in \{2n+1, \ldots, 2n+6m\}\}$, $G_3 = \{\text{track } i | i \in \{2n+6m+1, \ldots, 6n+6m\}\}$, $G_4 = \{\text{track } i | i \in \{6n+6m+1, \ldots, 6n+10m\}\}$, and $G_5 = \{\text{track } i | i \in \{6n+10m+1, \ldots, 8n+10m\}\}$. We construct an instance $I_0 = (k, p_0, \mathcal{N})$ of KKRB that sorts the supernets based on Lemma 11.3.7 as follows:

1. For each variable v_i, $i \in \{1, \ldots, n\}$, there is a set $\mathcal{V}_i$ consisting of 8 supernets $\mathcal{V}_i = \{A_i, \overline{A}_i, B_i, \overline{B}_i, L_i, \overline{L}_i, H_i, \overline{H}_i\}$
2. For each clause $\mathcal{C}_l = \{v_h, v_i, v_j\}$, $l \in \{1, \ldots, m\}$, there is a set $\mathcal{C}_l$ of 10 supernets $\mathcal{C}_l = \{V_h^l, \overline{V}_h^l, V_i^l, \overline{V}_i^l, V_j^l, \overline{V}_j^l, X_l, \overline{X}_l, Y_l, \overline{Y}_l\}$

Now we set $\mathcal{N} = \bigcup_{i \in \{1,\ldots,n\}} \mathcal{V}_i \cup \bigcup_{l \in \{1,\ldots,m\}} \mathcal{C}_l$. $I_0 = (k, p_0, \mathcal{N})$ is constructed such that there exists a routing for I_0 if and only if the supernets in $\mathcal{N}$ terminate

with a net on the right boundary in the following ways:

1. For each variable v_i, $i \in \{1, \ldots, n\}$, the terminals on the right boundary for the supernets are as follows:
   (a) A_i, $\overline{A}_i$ are in this order on neighboring tracks in G_1 (more precisely, they are on tracks $2i-1$ and $2i$).
   (b) L_i, $\overline{L}_i$, H_i, $\overline{H}_i$ are in this order on neighboring tracks in G_3 (more precisely, they are on tracks $2n+6m+4i-3$, $2n+6m+4i-2$, $2n+6m+4i-1$ and $2n+6m+4i$).
   (c) B_i, $\overline{B}_i$ are in this order on neighboring tracks in G_5 (more precisely, they are on tracks $6n+10m+2i-1$ and $6n+10m+2i$).
2. For each Clause $\mathcal{C}_l = \{v_h, v_i, v_j\}$, $l \in \{1, \ldots, m\}$, $h < i < j$, the terminals on the right boundary for the supernets are as follows:
   (a) $V_h^l, \overline{V}_h^l, V_i^l, \overline{V}_i^l, V_j^l, \overline{V}_j^l$ are in this order on neighboring tracks in G_2 (more precisely, they are on tracks $2n+6l-5$, $2n+6l-4$, $2n+6l-3$, $2n+6l-2$, $2n+6l-1$ and $2n+6l$)
   (b) X_l, $\overline{X}_l$, Y_l, $\overline{Y}_l$ are in this order on neighboring tracks in G_4 (more precisely, they are on tracks $6n+6m+4l-3$, $6n+6m+4l-2$, $6n+6m+4l-1$ and $6n+6m+4l$).

We extend our instance I_0 step by step, in such a manner that we can fix a truth assignment for the variables in Σ. One extension step is performed for each variable. Let $I_i = (k, p_i, \mathcal{N})$ be the extended instance after the ith extension. The effect of the ith extension is that there is a routing for the extended instance I_i if and only if either supernet L_i and H_i terminate on tracks in G_1 and G_3 on the right boundary, or L_i and H_i terminate on a tracks in G_3 and G_5 on the right boundary. In the first case, we assign $false$ to variable v_i. In the latter case, we assign *true* to variable v_i. The extension I_i consists of four sub-extensions $I_{i,1}$ to $I_{i,4}$ and is shown in Figure 11.10 (shadownets are not shown). For notational simplicity, we denote the portion of the channel added by extension $I_{i,j}$ with $D_{i,j}$ ($i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, 4\}$), as shown in Figure 11.10.

Claim 11.4.1 *For all $i \in \{1, \ldots, n\}$ it holds that:*

1. *Extension I_i is $(\mathcal{N} - \mathcal{V}_i)$-safe for I_{i-1}.*
2. *All supernets of I_i terminate on the right boundary of I_i.*
3. *I_i is (L_i, H_i)-mutable and $\overline{L}_i$, $\overline{H}_i$ are shadownets of L_i and H_i respectively.*
4. *In every routing for I_i, either*

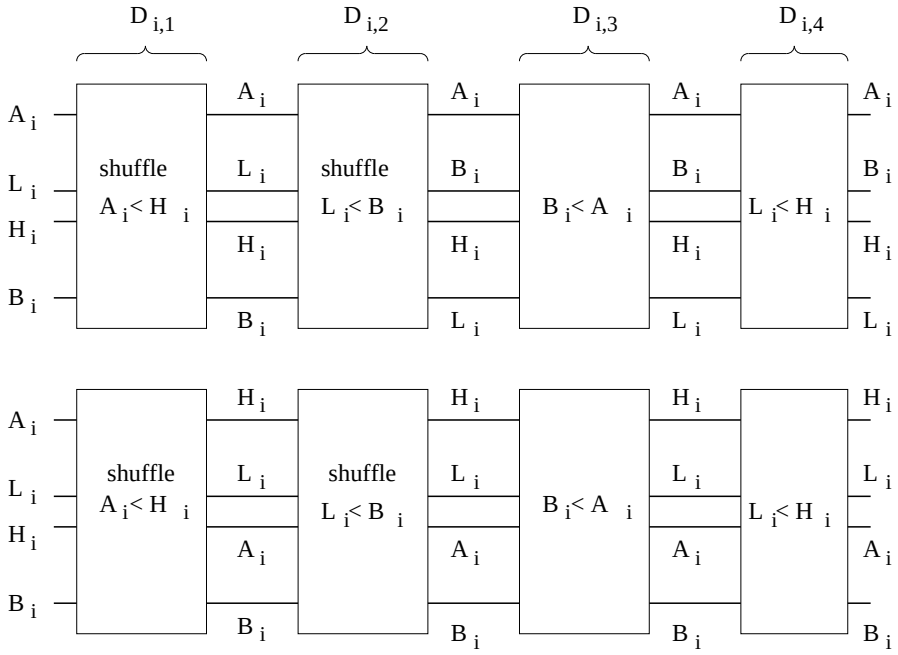


Figure 11.10: An extension that is used to produce a truth assignment for variable v_i and two possible routings (shadownets are not shown).

(a) exactly one supernet of $\{L_i, H_i\}$ terminates on a track in G_1 and exactly one of them terminates on a track in G_3 on the right boundary, or

(b) exactly one supernet of $\{L_i, H_i\}$ terminates on a track in G_3 and exactly one of them terminates on a track in G_5 on the right boundary.

5. For all cases in Condition 4, there exists such a routing.

Proof(Claim 11.4.1): Since any sub-extension $I_{i,j}$ of I_i is $\mathcal{N} - \mathcal{V}_i$-safe for $I_{i,j-1}$ respectively I_{i-1}, I_i is $\mathcal{N} - \mathcal{V}_i$-safe for I_{i-1}. By construction, all supernets of I_i terminate with a 2- or 3-terminal net on the right boundary of I_i. By construction of $D_{i,4}$, I_i is (L_i, H_i)-mutable and $\overline{L_i}$, $\overline{H_i}$ are shadownets of L_i and H_i respectively.

By construction of I_{i-1}, the arrangements of the supernets in the last column of I_{i-1} is as given in Figure 11.10. A_i is on a track in G_1, L_i is on a track above H_i in G_3, and B_i is on a track in G_5.

By Lemma 11.3.8, $I_{i,1}$ is $\mathcal{N} - \{A_i, \overline{A_i}, H_i, \overline{H_i}\}$-safe for I_{i-1}, and we have to consider two types of routings: whether A_i changes its track with H_i within $D_{i,1}$. Suppose that A_i does not change its track with H_i within $D_{i,1}$ (upper figure in Figure 11.10). Then, there exists no routing such that L_i stays on its track within $D_{i,2}$ (otherwise L_i would be above H_i in the first column of $G_{i,4}$ and hence there

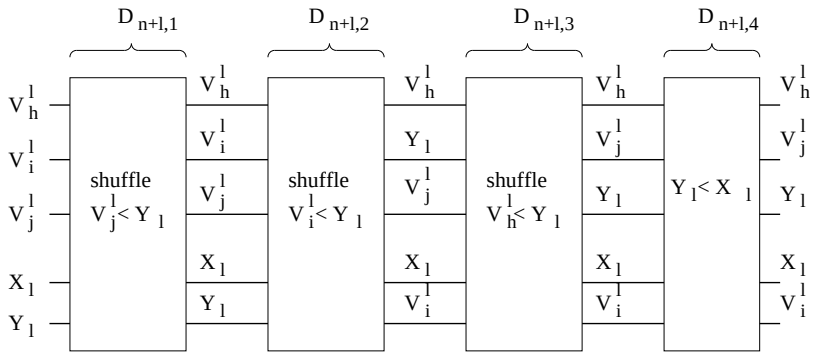


Figure 11.11: An extension that forces that exactly one supernet out of V_h^l, V_i^l, and V_j^l to a track in G_4. The interpretation is, that this variable is true.

is no routing for I_i by Lemma 11.3.8). Hence L_i change its track with B_i within $D_{i,3}$ and in this case, L_i and H_i are on tracks in $G_3 \cup G_5$ in the first column of $D_{i,4}$ and H_i is on a track above L_i. By definition of extension $I_{i,4}$, this holds for the right boundary of I_i.

Suppose that A_i changes its track with H_i within $D_{i,1}$ (lower figure in Figure 11.10). Then, if B_i changes its track with L_i within $D_{i,2}$, B_i is above A_i in the first column of $D_{i,4}$. It follows by Lemma 11.3.8, that there is no such routing for I_i. Hence, A_i does not change its track with L_i in the second extension, and L_i, H_i are on tracks of $G_3 \cup G_5$ on the right boundary of I_i. ■

Now, we extend our instance I_n step by step in such a way that we can choose exactly one true variable in each clause. One extension step is performed for each clause. Let I_{n+l} be the instance after the lth extension step. The effect of the lth extension step will be that there is a routing for I_{n+l} if and only if exactly one of the three supernets V_h^l, V_i^l, V_j^l changes to a track in G_4. The extension I_{n+l} is given in Figure 11.11 (shadownets are not shown). I_{n+l} consists of four sub-extensions $I_{n+l,1}, \ldots, I_{n+l,4}$, and we denote the portion of the cannel added by extension $I_{n+l,j}$ with $D_{n+l,j}$ ($l \in \{1, \ldots, m\}, j \in \{1, \ldots, 4\}$).

Claim 11.4.2 *For each clause $C_l = \{v_h, v_i, v_j\} \in \mathcal{C}$, the following holds:*

1. *Extension I_{n+l} is $\mathcal{N} - \mathcal{C}_i$-safe for I_{n+l-1}.*
2. *All supernets of I_{n+l} terminate on the right boundary of I_i.*
3. *In each routing for I_{n+l} exactly one of the three supernets V_h^l, V_i^l, or V_j^l terminates on a track in G_4, and the other two terminate on a track in G_2 on the right boundary. Furthermore in I_{n+l}, $\overline{V}_h^l, \overline{V}_i^l$, and $\overline{V}_j^l$ are shadownets of V_h^l, V_i^l and V_j^l respectively .*

```
FOR i = 1 TO n
    FOR all l with v_i ∈ C_l
        check V_i^l < L_i
        check H_i < V_i^l
    END
    check H_i < L_i
END
```

Algorithm 1: The algorithm to extend $I_n + m$ to I. We check if all V_i^l of clauses C_l with variable v_i are routed between H_i and L_i.

4. For all cases in Condition 4, there exists such a routing.

Proof(Claim 11.4.2): Since any sub-extension $I_{n+l,j}$ of I_{n+l} is $\{\mathcal{N} - \mathcal{C}_l\}$-safe for $I_{n+l,j-1}$ respectively I_{n+l-1}, I_{n+l} is $\mathcal{N} - \mathcal{V}_i$-safe for I_{i-1}. By construction, all supernets of I_i terminate on the right boundary of I_i and shadownets terminate beside their respective supernets on the right boundary. By construction of I_{n+l-1}, the arrangements of the supernets in the last column of I_{n+l-1} is as given in Figure 11.11. V_h^l, V_i^l, and V_j^l are on a track in G_2, X_l and Y_l are in this order on tracks in G_4. We show that in any routing for I_{n+l}, Y_l changes its track to a track in G_2 within $D_{n+l,1}$, $D_{n+l,2}$, or $D_{n+l,3}$. Suppose that Y_l does not change its track in one of these channel portions. Then, Y_l is below X_l in the first column of $D_{n+l,4}$. And by Lemma 11.3.8, there is no such routing for I_{n+l}.

Now, suppose that Y_l changes its track with V_h^l within $D_{n+l,1}$. Then, V_h^l is on a track in G_4 and V_i^l and V_j^l are on tracks in G_2 in the first column of $D_{n+l,2}$. By construction, V_h^l is on a track in G_4 and V_i^l and V_j^l are on tracks in G_2 on the right boundary of I_{n+l} in any such routing. Furthermore there is a routing for all the following shuffle-checks and Y_l is above X_l. Hence, there is a routing in this case.

Suppose that Y_l does not changes its track with V_h^l within $D_{n+l,1}$. Then V_h^l is on a track in G_2 on the right boundary of I_{n+l}, and either V_i^l or V_j^l is on a track in G_4. The proof that such routings exist is similar to the previous paragraph. ∎

To finish our construction for Theorem 11.4.1, we extend the instance $I_{n+m} = \{k, p, \mathcal{N}\}$ to the instance $I = \{k, q, \mathcal{N}\}$ in the following way. For each variable v_i check if all supernets V_i^l of clauses C_l with $v_i \in C_l$ are on tracks between H_i and V_i. We can do that by successively applying the extension of Lemma 11.3.8 such that $V_i^l < L_i$ and $H_i < V_i^l$. The exact way to do that is shown in Algorithm 1.

Claim 11.4.3 *A routing R' for I exists if and only if a routing R for I_{n+m} exists such that for all $i \in \{1,\ldots,n\}$ and each $l \in \{1,\ldots,m\}$ with $v_i \in C_l$, V_i^l is on a track between the tracks of H_i and L_i on the right boundary.*

Proof(Claim 11.4.3): We first show that there is a routing if the given condition is met. In this routing, for each $i \in \{1,\ldots,n\}$ there is some $h_i \in \{1,3\}$ such that H_i is on a track in G_{h_i}, V_i^l is on a track in G_{h_i+1} for all l with $v_i \in C_l$, and L_i is on a track in G_{h_i+2}. This routing is such that in each of the checks $V_i^l < L_i$ and $H_i < V_i^l$ the supernets do not change their tracks. Hence there is a routing in any of these checks. There is also a routing for the last check, $H_i < L_i$.

Now we show that there is no routing if the condition is not met. Suppose that there is a routing R for I such that there exists some $l \in \{1,\ldots,m\}$ with $v_i \in C_l$ and V_i^l is not between H_i and L_i. We consider the first such supernet that is checked in the channel (the leftmost check. This supernet has minimal superscript l for a given i). Suppose that L_i, H_i are on tracks of G_1, G_3 at column $p \leftarrow$. Then, since V_i^l is the first net in the checking of v_i that is not between G_1 and G_3, L_i and H_i are on tracks of $G_1 \cup G_2 \cup G_3$ and V_i^l is on a track in G_4. The following extension $V_i^l < L_i$ demands that L_i routed below V_i^l which leads to a contradiction. Hence, such a routing does not exist.

Suppose that L_i, H_i are on tracks of G_3, G_5 at column $p \leftarrow$. Then, L_i, H_i are on tracks of $G_3 \cup G_4 \cup G_5$ before the check and V_i^l is on a track of G_2 (see Figure 11.12). If V_i^l does not change its track with L_i at the first check $V_i^l < L_i$, there is no routing for the second check $H_i < V_i^l$. Hence, after the second check, L_i is on a track of G_2 and H_i is on a track of $G_3 \cup G_4 \cup G_5$. If C_l is the last clause that contains v_i, by Algorithm 1, a check $H_i < L_i$ follows which leads to a contradiction since such a routing does not exist. Suppose that there is another clause C_j with $v_i \in C_j$ that is checked right after the considered extension. V_i^j is routed on a track above L_i because of the check $V_i^j < L_i$. Hence, V_i^j is on a track in G_2 after this check, but the following $H_i < V_i^j$ demands that V_i^j is routed on a track below H_i which leads to a contradiction. Hence, such a routing does not exist. ∎

It remains to show that there exists a routing for I if and only if there is a $\mathcal{C}$-satisfying truth assignment for the variables in Σ such that there is exactly one true literal in each clause.

Suppose that such a truth-assignment exists. Consider the tracks of H_i, L_i on the right boundary of I_{n+m}. By Claim 11.4.1 and 11.4.21 we can find a routing for L_i, H_i such that H_i is on a track in G_1 and L_i is on a track in G_3 for all i with $v_i = false$ in the truth-assignment and furthermore H_i is on a track in G_3 and L_i is on a track in G_5 for all i with $v_i = true$ in the truth-assignment. Since in each clause exactly one of the variables is true, by Claim 11.4.2 we can find a routing such that for all $i \in \{1,\ldots,n\}$ and all l with $v_i \in C_l$ V_i^l is at a track in

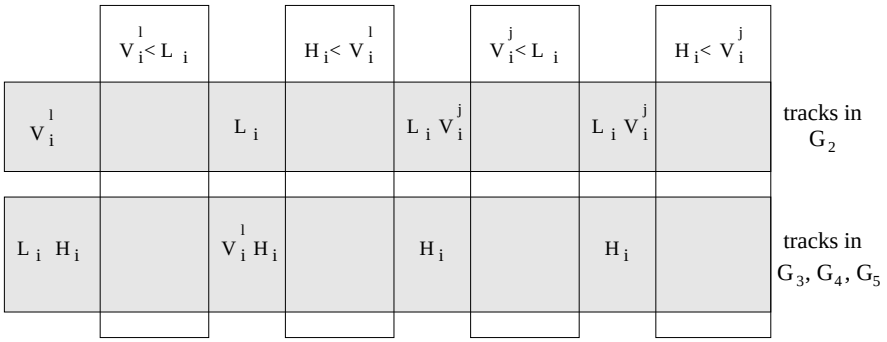


Figure 11.12: No routing exists if the truth-assignment of the global variable (done by L_i and H_i) does not match the truth assignment of the variables in the clauses (V_i^l). A single check is shown for the case that L_i, H_i are on tracks of $G_3 \cup G_4 \cup G_5$ and V_i^l is on a track in G_2.

G_2 if $v_i = false$ and at a track in G_4 if $v_i = true$ in the truth-assignment. By Claim 11.4.3 there is a routing for I.

Suppose that there exists a routing R for $I = (k, p, \mathcal{N})$. The routing remains valid for the portion of I_{n+m}. Denote this routing of I_{n+m} with R_{m+n}. By Claim 11.4.1 and 11.4.21, H_i and L_i are either at tracks of $G_1 \cup G_3$ or at tracks of $G_3 \cup G_5$ on the right boundary of I_{n+m} in R_{m+n}. In the first case set $v_i = false$, in the latter set $v_i = true$. Consider this truth-assignment for variables in Σ. By Claim 11.4.3, for all l with $v_i \in C_l$ the net V_i^l is between H_i and L_i on the right boundary of I_{n+m} in R_{m+n}. By Claim 11.4.2, for each l, exactly one supernet of V_h^l, V_i^l, V_j^l is on a track in G_4 and the other supernets are on tracks in G_2 on the right boundary of I_{n+m} in R_{m+n}. Hence, exactly one variable in each clause is true in this truth-assignment. ∎

Theorem 11.4.2 *The general knock-knee channel-routing problem with 3-terminal nets is NP-complete.*

Proof: It is easy to reduce KKRB with 3-terminal nets to the general knock-knee channel-routing problem . Let $I' = (k, p, \mathcal{N})$ be an instance of KKRB. We construct an instance I of the knock-knee channel-routing problem the following way: Replace the i-th net of the form $(a_1, \ldots, a_n, r)$ with the net $(a_1, \ldots, a_n, t_{p+i})$ (there are k such nets). Then, let instance I be the union of all the nets of the supernets in $\mathcal{N}$. Clearly, there is a routing for I if and only if there is a routing for I'. ∎

11.5 Discussion

The main result of this chapter is that Knock-knee channel-routing is NP-complete even if at most 3-terminal nets are involved. Polynomial time algorithm are known for Knock-knee channel routing with 2-terminal nets. Hence, this result gives a sharp boundary of intractability for channel-routing in the knock-knee mode.

The fact that most routing problems in VLSI are intractable has also implications to biological circuits. The optimal layout of millions of neurons is not tractable, even for an evolutionary process lasting for billions of years REF. Hence, it is not likely that optimal layout solutions can be found in biological circuits, reasonable good approximations are much more plausible.

Chapter 12

Conclusions and Outlook

Bibliography

Abbott, L., Varela, J., Sen, K., and S.B., N. (1997). Synaptic depression and cortical gain control. *Science*, 275:220–4.

Abeles, M. (1998). *Corticonics: Neural Circuits of the Cerebral Cortex.* Cambridge Univ. Press.

Allman, J. M. (1999). *Evolving Brains.* Freeman (New York).

Bannister, N. J. and Larkman, A. U. (1995). Dendritic morphology of CA1 pyramidal neurones from the rat hippocampus. *Journal of Comparative Neurology*, 360:150–160.

Di Battista, G., Eades, P., Tamassia, R., Tollis, I. G. (1999). *Graph Drawing: Algorithms For The Visualization of Graphs.* Prentice-Hall (New Jersey, USA)

Bear, M. F., Connors, B. W., Paradiso, M. A. (1996). *Neuroscience: Exploring the Brain* Williams & Wilkins, Baltimore, USA.

Bhatt, S. N. and Leiserson, C. E. (1982). How to assemble tree machines. *Proceedings of the Fourteenth Annual ACM Symposium on the Theory of Computing*, 77–84.

Braitenberg, V., Schüz, A. (1998). *Cortex: Statistics and Geometry of Neuronal Connectivity.* Springer Verlag, 2nd ed..

Brent, R. P. and Kung, H.-T. (1980). On the area of binary tree layouts. *Information Processing Letters*, 11(1):46–48.

Cajal, S. R. (1995). *Histology of the Nervous System*, volumes 1 and 2, Oxford University Press (New York).

Cherniak, C. (1992) Local optimization of neuron arbors. *Biological Cybernetics*, 6:503–510.

Cherniak, C. (1995) Neural component placement. *Trends in Neuroscience*, 18:522–527.

Chklovskii, D. B., and Koulakov, A. A. (2000). A wire length minimization approach to ocular dominance patterns in mammalian visual cortex. *Physica A*, 284(1-4):318–334.

Chklovskii, D. B. (2000). Binocular disparity can explain the orientation of ocular dominance stripes in primate primary visual area (V1). *Vision Research*, 40(13):1765–1773.

Chklovskii, D. B. (2000). Optimal sizes of dendritic and axonal arbors. *Advances in Neural Information Processing Systems*, vol. 12, MIT Press, 108–114.

Chklovskii, D. B., and Stevens, C. F. (2000). Wiring optimization in the brain. *Advances in Neural Information Processing Systems*, vol. 12, MIT Press, 103–107.

DasGupta, B., Schnitger G. (1996). Analog versus Discrete Neural Networks. *Neural Computation*, 8:819–842.

Dobrunz, L. and Stevens, C. (1997). Heterogeneity of release probability, facilitation ad depletion at central synapses. *Neuron*, 18:995–1008.

Durbin, R and Mitchison, G. A. (1990). A dimension reduction framework for understanding cortical maps. *Nature*, 343:644–647.

Fernandez, E., Eldred, W. D., Ammermuller, J., Block, A., Von Bloh, W., and Kolb, H. (1994). Complexity and scaling properties of amacrine, ganglion, horizontal, and bipolar cells in the turtle retina. *Jornal of Comparative Neurology*, 347:397–408.

Fiala, J. C. and Harris, K. M. (1999). Dendrite structure. In *Dendrites*, pp. 1–67. (G. Stuart, N. Spruston, M. Häusser, editors). Oxford Univ. Press.

Fisher, M. J. and Paterson, M. S. (1980). Optimal tree layout. *Proceedings of the Twelth Annual ACM Symposium on the Theory of Computing*, 177–189.

Fisher, M. J. and Paterson, M. S. (1999). Optimal layout of edge-weighted forests. *Discrete Applied Mathematics*, 90:135–159.

Floyd, R. W. and Ullman, J. D. (1982). The compilation of regular expressions into integrated circuits. *Journal of the ACM*, 29(2):603–622.

Foster, M. (1897). *A textbook of physiology* MacMillan (New York), 17th ed..

Frank, A. (1982). Disjoint paths in a rectilinear grid. *Combinatorica*, vol. 2(4):361–371.

Formann, D., Wagner, D., Wagner, F. (1993). Routing through a dense channel with minimum total wire length. *Journal of Algorithms*, vol. 15(2):267–283.

Garey, M. R., Johnson, D. S. (1979). *Computers and Intractability.* W. H. Freeman, San Francisco.

Harvey, R. J. and Napper, R. M. A. (1991). Quantitative studies on the mammalian cerebellum. *Progress in Neurobiology*, 36:437–463.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation.* Macmillan/IEEE Press.

Hubel, D. H., and Wiesel, T. N., (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160:106–154.

Kaas, J. H. (1997). Topographic Maps are Fundamental to Sensory Processing. *Brain Research Bulletin*, 44(2):107–112.

Kandel, E. R., Schwartz, J. H., and Jessel, T. M. (1991). *Principles of Neural Science.* Prentice-Hall, third edition.

Kellerth, J. O., Berthold, C. H., and Conradi, S. (1979) Electron microscopic studies of serially sectioned cat spinal alpha-motoneurons. *Journal of Comparative Neurology*, 184:755–767.

Koch, C. (1999). *Biophysics of Computation: Information Processing in Single Neurons.* Oxford University Press.

Koulakov, A. A. and Chklovskii, D. B. (2001). Orientation Preference Patterns in Mammalian Visual Cortex: A Wire Length Minimization Approach. *Neuron*, 29:519–527.

Kuchem, R., Wagner, D., Wagner, F., (1996). Optimizing Area for Three-Layer Knock-Knee Channel Routing. *Algorithmica*, vol. 15, 495-519.

Lazzaro, J., Ryckebusch, S., Mahowald, M. A., Mead, C. A. (1989). Winner-take-all networks of $O(n)$ complexity. *Advances in Neural Information Processing Systems*, vol. 1, Morgan Kaufmann (San Mateo), 703–711.

Legenstein, R. A. (2001). On the Complexity of Knock-Knee Channel Routing with 3-Terminal Nets. Submitted for publication.

Legenstein, R. A. and Maass, W. (2001a). Foundations for a circuit complexity theory of sensory processing. *Advances in Neural Information Processing Systems*, 13:259–265.

Legenstein, R. A. and Maass, W. (2001b). Total Wire Length as a Salient Circuit Complexity Measure for Sensory Processing. Submitted for publication.

Legenstein, R. A. and Maass, W. (2001c). Neural Circuits for Pattern Recognition with Small Total wire Length. Submitted for publication.

Legenstein, R. A. and Maass, W. (2001d). Optimizing the Layout of a Complete Tree. Submitted for publication.

Leighton, F. T. (1981). New lower bound techniques for VLSI. *Proceedings of the Twenty-Second Annual IEEE Symposium on Foundations of Computer Science*, 1–12.

Leiserson, C. E. (1980). Area efficient graph algorithms (for VLSI). *Proceedings of the Twenty-First Annual IEEE Symposium on Foundations of Computer Science*, 270–281.

Lipton, R. J. and Sedgewick, R. (1981). Lower bounds for VLSI. *Proceedings of the Thirteenth Annual ACM Symposium on the Theory of Computing*, 300–307.

Lipton, R. J. and Tarjan, R. E. (1979). A planar separator theorem. *SIAM J. Applied Mathematics*, 36(2):177–189.

Maass, W. (2000). On the computational power of winner-take-all. *Neural Computation*, 12(11):2519–2536.

Maass, W. (1985). Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 10:1659–1671.

Maass, W. and Zador, A. (1999). Dynamic stochastic synapses as computational units. *Neural Computation*, 11:903–917.

McCulloch W. S. and Pitts, E. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

Mead, C., and Rem M. (1979). Cost and performance of VLSI computing structures. *IEEE J. Solid State Circuits* SC-14(1979):455–462.

Mead, C. (1989). *Analog VLSI and Neural Systems*. Addison-Wesley (Reading, MA, USA).

Mel, B. W. (1994). Information processing in dendritic trees. *Neural Computation*, 6:1031–1085.

Middendorf, M. (1996). Manhattan Channel Routing is NP-Complete under Truly Restricted Settings. *Chicago Journal of Theoretical Computer Science*, vol. 1996, MIT Press, Article 6.

Mitchison, G. (1991). Neuronal branching patterns and the economy of cortical wiring. *Proceedings of the Royal Society of London, Series B. Biological Sciences*, 245:151–158.

Mitchison, G. (1992). Axonal trees and cortical architecture. *Trends in Neuroscience*, 15(4):22–26.

Nelson, M. E. and Bower, J. M. (1990). Brains, Maps and parallel computers. *Trends in Neurosciences*, 13(10):403–408.

Ofman, Y. (1962). On the Algorithmic Complexity of Discrete Functions. *Dokl. Akad. Nauk SSSR* (Soviet Math. Dokl.) 145(1962):48–51 (in Russian); English translation in *Soviet Math. Dokl.* 7(1963):589–591.

Paterson, M. S., Ruzzo, W. L., and Snyder, L. (1981). Bounds on minimax edge length for complete binary trees. *Proceedings of the Thirteenth Annual ACM Symposium on the Theory of Computing*, 293–299.

Purves, D., Augustine, G. J., Fitzpatrick, D., Katz, L. C., LaMantia, A. S., McNamara, J. O. editors (1997). *Neuroscience* Sinauer Associates, (Sunderland, MA, USA).

Rapp, M., Segev, I., Yarom, Y. (1994). Physiology, morphology and detailed passive models of guinea-pig celebral Purkinje cells. Journal of Physiology, 474:108–118.

Ruzzo, W. L. and Snyder, L. (1981). Minimum edge length planar embeddings of trees. in Kung, H.-T., Sproull, R., and Steele, G. (eds.): *VLSI Systems and Computations*. Computer Science Press (Rockville, Md., USA). pp. 119–123.

Sarrafzadeh, M. (1987). Channel-routing problem in the knock-knee mode is NP-complete. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, vol. 6(4):503–506.

Savage, J. E. (1998). *Models of Computation: Exploring the Power of Computing.* Addison-Wesley (Reading, MA, USA).

Segev, I. and London, M. (1999). A theoretical view of passive and active dendrites. In *Dendrites*, pp. 205–269. (G. Stuart, N. Spruston, M. Häusser, editors). Oxford Univ. Press.

Shannon, C. E. (1938). A Symbolic Analysis of Relay and Switching Circuits. *Transactions of the AIEE*, 57:713–723.

Shannon, C. E. (1949). The Synthesis of Two-Terminal Switching Circuits. *Bell System Technical Journal*, 28:59–98.

Shepherd, G. M. (1994). *Neurobiology.* Oxford University Press.

Shepherd, G. M. (1998). *The Synaptic Organization of the Brain.* Oxford Univ. Press, 2nd ed.

Siu, K.-Y., Roychowdhury, V., and Kailath, T. (1995). *Discrete Neural Computation: A Theoretical Foundation.* Prentice-Hall (New Jersey, USA).

Smith, T. G., Marks, W. B., Lange, G. D., Sheriff, W. H., and Neale, E. A. (1989). A fractal analysis of cell images. *Journal of Neuroscience Methods*, 27:173–180.

Spruston, N., Stuart, G., Häusser, M. (1999). Dendritic integration. In *Dendrites*, pp. 231–270. (G. Stuart, N. Spruston, M. Häusser, editors). Oxford Univ. Press.

Sterling, P. (1990). Retina. In *The synaptic organization of the brain*, pp. 170–215. (G. M. Shepherd, editor). Oxford Univ. Press, 3rd ed.

Stuart, G., Spruston, N., Häusser, M., editors (1999). *Dendrites*, pp. 231–270. Oxford Univ. Press.

Thompson, C. D. (1979). Area-time complexity for VLSI. *Proceedings of the Eleventh Annual ACM Symposium on the Theory of Computing*, 81–88.

Thorpe, S., Fize, D., Marlot, C. (1996). Speed of processing in the human visual system. *Nature*, 381:520–522.

Ulfhake, B. and Kellerth, J. O. (1981). A quantitative light microscopic study of the denrdrites of cat spinal alpha-motorneurons after intracellular staining with horseradish peroxidase. *Journal of Comparative Neurology*, 202:571–583.

Ullman, J. D. (1984). *Computational Aspects of VLSI.* Compuer Science Press (Rockville, Md., USA).

Valiant, L. G. (1981). Universality considerations in VLSI circuits. *IEEE Transactions on computers*, C-30(2):135–140.

Valiant, L. G. (1994). *Circuits of the Mind.* Oxford Univ. Press (New York).

Weinberg, R. J. (1997). Are Topographic Maps Fundamental to Sensory Processing. *Brain Research Bulletin*, 44(2):113–116.

Wegener, I. (1987). *The Complexity of Boolean Functions*, Wiley-Teubner, Stuttgart and New York.

Wingate, R. J., Fitzgibbon, T., and Thompson, I. D. (1992). Lucifer yellow, retrograde tracers, and fractral analysis characterize adult ferret retinal ganglion cells. *Journal of Comparative Neurology*, 323:449–474.

Yao, A. C. (1979) Some complexity questions related to distributed computing. *Proceedings of the Eleventh Annual ACM Symposium on the Theory of Computing*, 209–213.

Yao, A. C. (1981) The entropic limitations of VLSI computations. *Proceedings of the Thirteenth Annual ACM Symposium on the Theory of Computing*, 308–311.

Zhang, K. and Sejnowski, T. J. (2000). A universal scaling law between gray matter and white matter of cerebral cortex. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 97(10):5621–5626.