Electronic Colloquium on Computational Complexity, Report No. 70 (2001)

Total Wire Length as a Salient Circuit Complexity Measure for Sensory Processing[1]

Robert A. Legenstein
Institute for Theoretical Computer Science
Technische Universität Graz, Austria
E-mail: legi@igi.tu-graz.ac.at

and

Wolfgang Maass
Institute for Theoretical Computer Science
Technische Universität Graz, Austria
E-mail: legi@igi.tu-graz.ac.at

Version: October 24, 2001

We introduce *total wire length* as salient complexity measure for analyzing the circuit complexity of sensory processing in biological neural systems and neuromorphic engineering. The new complexity measure is applied in this paper to two basic computational problems that arise in 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. In addition we derive general bounds for the total wire length of circuits in terms of traditional complexity measures.

Key Words: circuit complexity; total wire length; sensory processing;

1. INTRODUCTION

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 [10]). 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.

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.

[1]Research for this article was partially supported by the the Fonds zur Förderung der wissenschaftlichen Forschung (FWF), Austria, project P12153, and the NeuroCOLT project of the EC.

We will follow traditional circuit complexity theory in assuming that the underlying graph of each circuit is a directed graph without cycles. 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" [10]. This view has been adopted by quite a number of neuroscientists as a guiding principle for understanding cortical circuitry. In [11] the partition of the neocortex into areas, and in [4] and [6] the ocular dominance patterns of visual cortex were derived from this principle. The goal of this article 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 special functions of k inputs such as the function computed by a threshold gate[2] *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 [8]). In the relatively abstract context of this model we model such computational modules as "threshold 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*

[2] A threshold gate computes a Boolean function $T : \{0,1\}^k \to \{0,1\}$ of the form $T(x_1, \ldots, x_k) = 1 \Leftrightarrow \sum_{i=1}^{k} w_i x_i \geq w_0$.

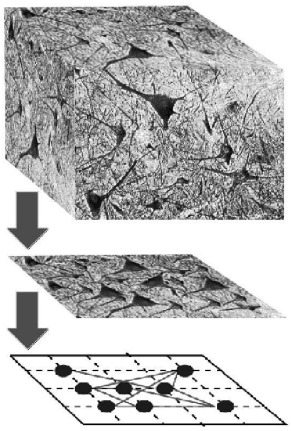

FIG. 1 The relationship between cortical circuitry and a simple mathematical model is explained by a projection onto a 2-dimensional plane.

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 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. The relationship between the new complexity measure total wire length and traditional circuit complexity measures is discussed in Section 3 of this article.

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 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 [7]), 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 [7]. 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 article 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 [1, 7, 13, 2].[3] Collectively they 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.

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 article 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 article. This suggests that the circuit designs that we consider in this article do not only satisfy the complexity requirements imposed by cortical circuitry, but can potentially also be implemented in VLSI.

[3]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.

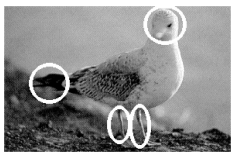

FIG. 2 Examples of some local features (marked), whose spatial arrangement is essential for recognizing an object.

We refer to Section 12.2 in [12] for the precise definition of the abstract model for VLSI-area to which the theorems in this article refer. One assumes there 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. The complexity measure induced by this model is the *area* of the smallest rectangle that encloses the circuit.

Since we consider in this article 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 [12] 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 [12] we always assume that all inputs are presented in parallel.

In this article 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 Section 2 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. It turns out that the same circuit design techniques that we introduce in this article also yield circuits that require relatively little area in the common abstract model for VLSI-area.

In Section 3 we derive general bounds for the total wire length of a circuit in terms of the number of gates and in terms of the VLSI area required by the circuit.

2. GLOBAL PATTERN DETECTION IN 2-DIMENSIONAL MAPS

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 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 2), 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 $O(\ldots)$. A function λ of n is said to be $O(f(n))$ if there exist constants C_0, C_1 such that $\lambda(n) \leq C_1 \cdot f(n) + C_0$ for all $n \in N$. Thus $O(n)$ simply means: bounded by a function that is linear in n. Whenever needed we assume for simplicity that n is such that $\sqrt{n}, \log n$ etc. are natural numbers. 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.

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 indices 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

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

THEOREM 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 3) then it suffices to compute for each sub-square C_k the following data:

$$\begin{aligned} left(C_k) &:= \text{the x-coordinate of the leftmost location } i \text{ in } C_k \text{ with } a_i = 1 \\ right(C_k) &:= \text{the x-coordinate of the rightmost location } j \text{ in } C_k \text{ with } b_j = 1 \end{aligned}$$

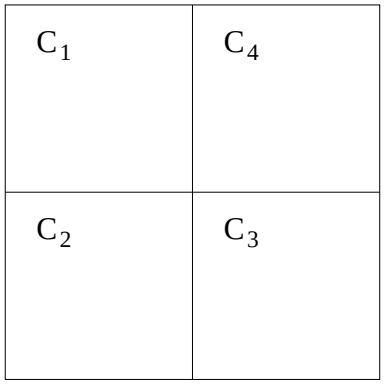

FIG. 3 The input area C is divided into four sub-squares C_k, which are numbered in a counterclockwise fashion.

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

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 $found(C)$ 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 3):

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

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.

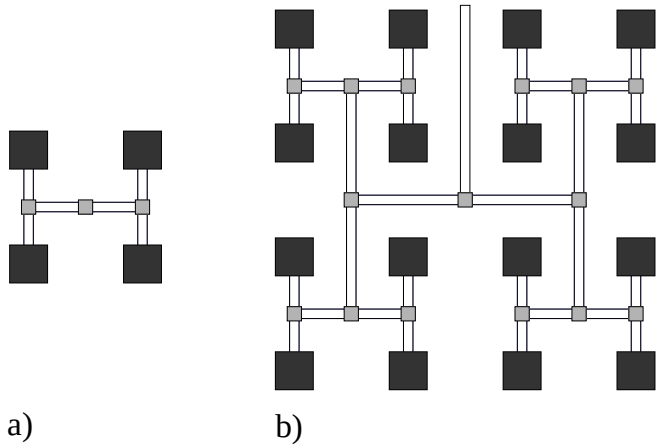

FIG. 4 The H-tree layout. Dark rectangles are leaves, light rectangles are inner nodes. a) H_1 is a tree layout for 4 leaves. b) H_2. H_{k+1} is constructed recursively by replacing the leaves of H_k with H-trees H_1 . (Figure taken from [12])

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. [12]), 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 4a 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 4b, 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 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.

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 1 shows that the extended H-tree stays within the claimed complexity bounds. The depth of an H-tree is $O(\log n)$, hence 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. ∎

LEMMA 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 5). Hence, the following recurrence holds:

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

The solution of Eq. (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

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

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:

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

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

$$C(H_k) \leq n(\frac{1}{4}C(H_1) + \frac{77}{12}s) \quad . \tag{3}$$

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 illustrated in Figure 5. We get a recursive formula which we iterate $k-1$ times:

$$\begin{aligned} TWL(H_k) &\leq 4TWL(H_{k-1}) + 2k \cdot e \cdot S(H_{k-1}) + 3d \cdot k^2 \qquad (4) \\ &\leq 4TWL(H_{k-1}) + k \cdot e(S(H_1) + 3c)2^{k-1} + 3d \cdot k^2 \\ TWL(H_k) &\leq 4^{k-1}TWL(H_1) + 2^{k-1}e(S(H_1) + 3c) \sum_{j=0}^{k-2} 2^j(k-j) \\ &\quad +3d \sum_{j=0}^{k-2} 4^j(k-j)^2 \quad . \qquad (5) \end{aligned}$$

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:

$$\begin{aligned} TWL(H_k) &\leq 4^{k-1}TWL(H_1) + \frac{3}{2}2^{2k-1}e(S(H_1) + 3c) + \frac{77}{36}d4^k \\ &\leq \frac{1}{4}n \cdot TWL(H_1) + \frac{3}{4}n \cdot e(S(H_1) + 3c) + \frac{77}{36}d \cdot n \\ TWL(H_k) &\leq n(\frac{1}{4}TWL(H_1) + \frac{3}{4}e(S(H_1) + 3c) + \frac{77}{36}d) = O(n) \quad . \qquad (6) \end{aligned}$$

∎

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 $a_{i^{(m)}}^{(m)} = 1$ 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 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*

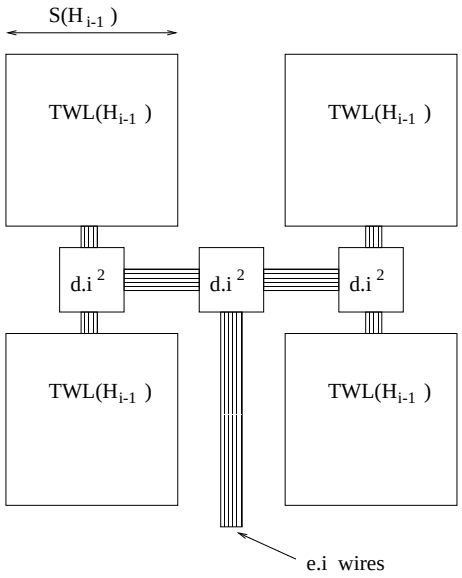

FIG. 5 The H-tree H_i has wires from four H-trees H_{i-1}, the wires of three inner nodes, and the wires of the busses. An inner node has a total wire length of $d \cdot i^2$, and a bus consists of $e \cdot i$ wires.

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 6. In Figure 6a, 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 6b 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 6b 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 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*: $x'_1 = x_1, x'_2 = x_1 \vee x_2, x'_3 = x_1 \vee x_2 \vee x_3, \ldots, x'_n = x_1 \vee x_2 \vee x_3 \vee \ldots \vee x_n$. Such a computation is called a *prefix computation.* There exist efficient circuits for such computations (see *e.g.* [12]). 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

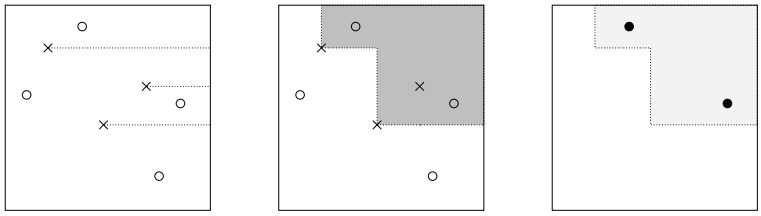

FIG. 6 Computing P_D with prefix circuits. Crosses mark locations where a feature a occurs, open circles mark locations where a features b is present. a) All locations that are in the same row and to the right of some feature $a = 1$ are marked as dotted lines. b) All locations that are to the right 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.

in Figure 6a. 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 6b). 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 6c, 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 2 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 2). 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_D^n 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 $P_D^{n,k}$ for some $k \geq 2$. We give a formal definition of $P_D^{n,k}$. 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 $P_D^{n,k}$:

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

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

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

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 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 1 holds:

COROLLARY 1. *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 $O(n \cdot (k \cdot \Delta \cdot \frac{\log n}{\log \Delta})^2)$.*

The proof of Theorem 2 relied on parallel prefix circuits. We show how a parallel prefix circuit can be implemented in our and the VLSI-model. 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 with inputs $x_1, \ldots, x_k$ computes the product $x_1 x_2 \ldots x_k$, for some fan-in k. Lemma 2 gives upper bounds on a circuit of such gates with maximal fan-in Δ that computes the prefixes $x_1, x_1 x_2, \ldots, x_1 x_2 \ldots x_n$. For simplicity, we assume that n is a power of Δ.

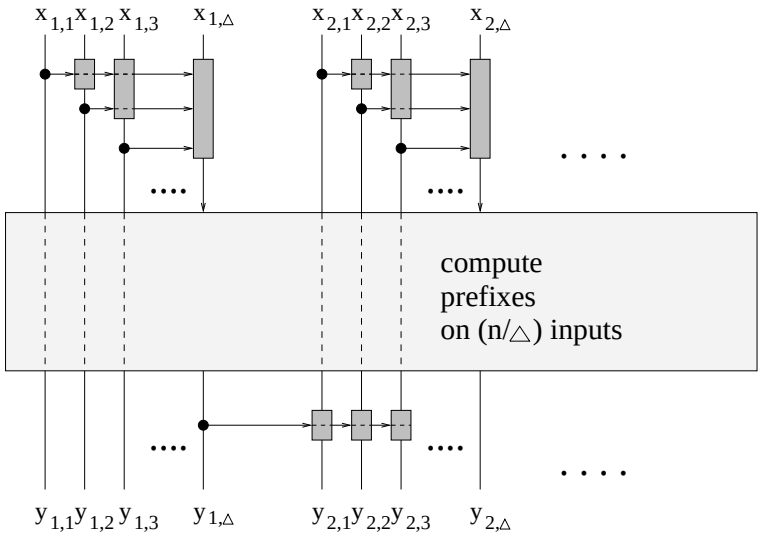

FIG. 7 Layout of an efficient prefix circuit with fan-in Δ. Dark shaded boxes are gates, the light shaded box is the recursive application of the circuit.

LEMMA 2. *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 7.

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 7. Figure 8 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:

$$C(PREF^n) \quad \leq \quad \frac{n}{\Delta}(\Delta - 1) + C(PREF^{\frac{n}{\Delta}}) + \frac{n}{\Delta}(\Delta - 1)$$

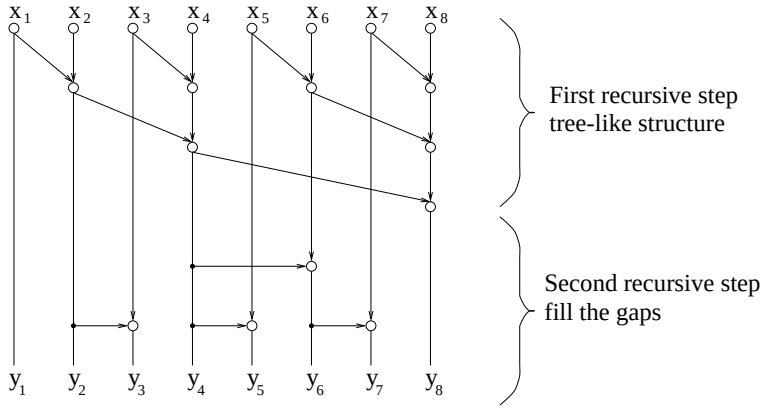

FIG. 8 The prefix circuit for $\Delta = 2$ and $n = 5$. It can be decomposed in a tree-structure and a post-processing. (Based on Figure 2.13 in [12].)

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

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.

$$\begin{aligned} S(PREF^1) &= 0 \\ S(PREF^n) &\leq (\Delta - 1) + S(PREF^{\frac{n}{\Delta}}) + 1 = \Delta\frac{\log n}{\log \Delta} \qquad (7) \\ area(PREF^n) &\leq nS(PREF^n) = n\Delta\frac{\log n}{\log \Delta} \quad . \qquad (8) \end{aligned}$$

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 7 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:

$$TWL(PREF^n) = O(n\Delta\frac{\log n}{\log \Delta}) \quad . \qquad (9)$$

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, 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 9. 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 9, the nodes are in the correct horizontal order. The root of the tree in the lower layout of Figure 9 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 9, 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) 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.

∎

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 3. *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.*

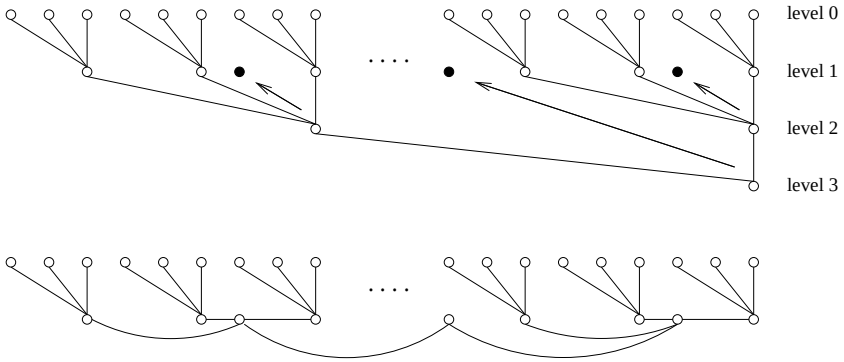

FIG. 9 Tree layout on a grid for $\Delta = 3$. The upper layout shows a leveled arrangement of inner nodes. The arrows and filled circles indicate the rearrangement of inner nodes. We gain a layout which uses just two rows (lower layout).

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 10). 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 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

$$a'_i = \bigvee_{\text{j is in the i-th column}} a_j \qquad b'_i = \bigvee_{\text{j is in the i-th column}} b_j \tag{10}$$

$$a''_i = \bigvee_{\text{j is in the i-th row}} a_j \qquad b''_i = \bigvee_{\text{j is in the i-th row}} b_j \quad . \tag{11}$$

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 $b_j = 1$:

$$g_m = \begin{cases} 1, & \text{if } \sum_{i=\max\{1,2m-n\}}^{\min\{n,2m\}} i.a'_i + i.b'_i \geq 2m \\ 0, & \text{else .} \end{cases} \tag{12}$$

$$g'_m = \begin{cases} 1, & \text{if } \sum_{i=\max\{1,2m-n\}}^{\min\{n,2m\}} i.a'_i + i.b'_i \leq 2m \\ 0, & \text{else .} \end{cases} \tag{13}$$

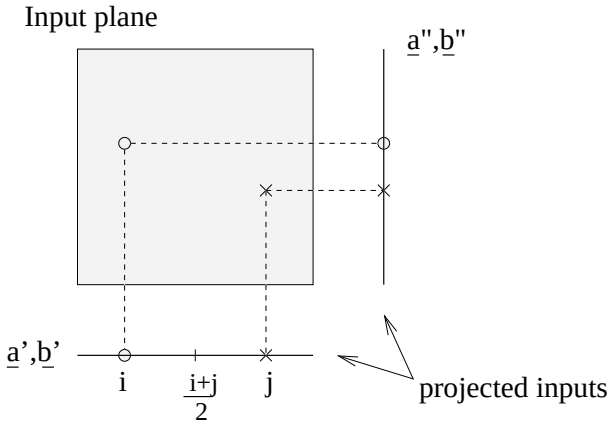

FIG. 10 The projection of the input plane onto its horizontal and vertical axis. A circle represents an occurring feature a and a cross an occurring feature b. In the horizontal projection, features occur at locations i and j. We compute the midpoints of the projections and trace them back onto the input plane.

$$h_m \quad = \quad g_m \wedge g'_m \quad . \tag{14}$$

$h_1, \ldots, h_{\sqrt{n}}$ 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. (12) and (13).

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. (10) and (11)) 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. (12) to (14)), 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. ∎

3. RELATIONSHIP BETWEEN TOTAL WIRE LENGTH AND OTHER CIRCUIT COMPLEXITY MEASURES

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

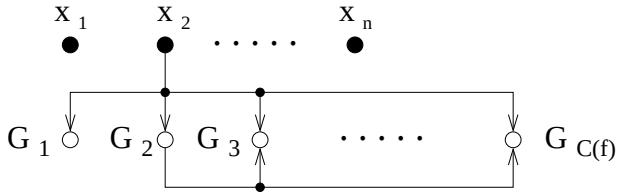

FIG. 11 A layout for an arbitrary circuit whose total wire length can easily be estimated in terms of n and $C(f)$. Filled circles $x_1, \ldots, x_n$ are input ports and open circles $G_1, \ldots, G_{C(f)}$ are gates.

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 lemma.

THEOREM 4. *The total wire length $TWL(f)$ of a function $f : \{0,1\}^n \to \{0,1\}$ 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 11). 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 $TWL(f)$. ∎

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

THEOREM 5. *If the function $f : \{0,1\}^n \to \{0,1\}$ can be computed in a feedforward 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 12). 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 paths connect $n'_{I_{G,1}}$ 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[4]. 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,

[4] On a tree, denote a path without cycles from a node a to a node b by $P_{a,b}$. Note that this path 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 $P_{B,D}$.

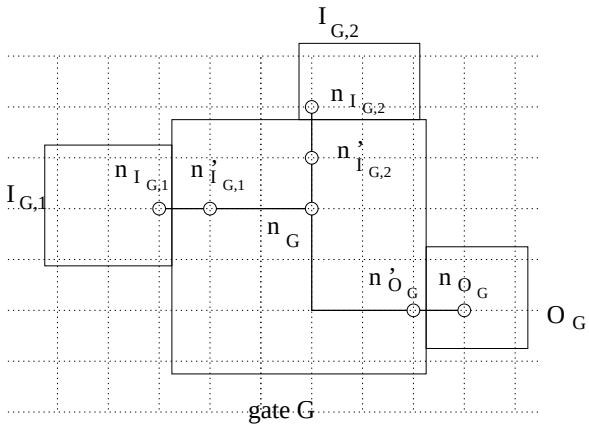

FIG. 12 A mapping of a VLSI-gate onto our model. Unique nodes inside and outside the gate are defined for input and output. The nodes are connected by a (minimal) spanning tree. The gate is mapped onto the node n_G, the node that is common to any path that connects two leaves of the spanning tree.

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 $O(\frac{\mu}{\lambda^2}A)$. ∎

4. DISCUSSION

We have introduced a new complexity measure, total wire length, that provides a useful criterion for judging whether a proposed circuit design is realistic from the point of view of a possible physical implementation in hardware or wetware. In particular we have shown that well-known empirical data from neurobiology suggest that biological neural circuits that solve global pattern recognition tasks have a total wire length that scales up subquadratically with the number of input variables. The relevance of the total wire length of cortical circuits had previously been emphasized by numerous neuroscientists, from Cajal (see for example p. 14 in [3]) to [5].

In Section 2 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 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. We also show that the new complexity measure total wire length is related 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 article. Hence the new circuit complexity measure total wire length may represent a useful compromise between practical relevance and mathematical simplicity.

REFERENCES

[1] M. Abeles, "Corticonics: Neural Circuits of the Cerebral Cortex", Cambridge Univ. Press, 1999.

[2] V. Braitenberg and A. Schüz, "Cortex: Statistics and Geometry of Neuronal Connectivity", 2nd ed., Springer Verlag, 1998.

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

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

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

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

[7] C. Koch, *Biophysics of Computation*, Oxford Univ. Press, 1999

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

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

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

[11] G. Mitchison, Axonal trees and cortical architecture, *Trends in Neuroscience*, 15(4)(1992), 22-26.

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

[13] G. M. Shepherd, *The Synaptic Organization of the Brain.* 2nd ed., Oxford Univ. Press, 1998