Efficient VLSI Layouts for Homogeneous Product Networks

Antonio Fernández and Kemal Efe, *Member*, *IEEE*

Abstract—In this paper, we develop generalized methods to layout homogeneous product networks with any number of dimensions, and analyze their VLSI complexity by deriving upper and lower bounds on the area and maximum wire length.

In the literature, lower bounds are generally obtained by computing lower bounds on the bisection width or the crossing number of the network being laid out. In this paper, we define a new measure that we call "maximal congestion," that can be used to obtain both the bisection width and the crossing number, thereby unifying the two approaches. Upper bounds are traditionally obtained by constructing layouts based on separators or bifurcators. Both methods have the basic limitation that they are applicable only for graphs with bounded vertex degree. The separators approach generally yields good layouts when good separators can be found, but it is difficult to find a good separator for an arbitrary graph. The bifurcators approach is easier to apply, but it generally yields larger area and wire lengths. We show how to obtain "strong separators" as well as bifurcators for any homogeneous product network, as long as the factor graph has bounded vertex degree. We illustrate application of both methods to layout a number of interesting product networks.

Furthermore, we introduce a new layout method for product networks based on the combination of collinear layouts. This method is more powerful than the two methods above because it is applicable even when the factor graph has unbounded vertex degree. It also yields smaller area than the earlier methods. In fact, our method has led to the optimal area for all of the homogeneous product networks we considered in this paper with one exception, which is very close to optimal. In regards to wire lengths, the results obtained by our method turned out to be the best of the three methods for all the examples we considered, again subject to one (and the same) exception. We give an extensive variety of such examples.

Index Terms—Interconnection networks, product networks, VLSI, collinear layout, separator, bifurcator.

1 Introduction

The Cartesian product is a well-known operation defined on graphs.[1] When applied to interconnection networks, the Cartesian product operation combines a set of "factor" networks into a "product network." Several well-known networks are instances of product networks, including the grid, the torus, the hypercube [15], and the generalized hypercube [4]. Recently, we have observed a strong interest in product networks, since a variety of new networks based on the Cartesian product operation have been proposed. However, there is no paper which studies the VLSI layout complexity of product networks as a general class. This problem is important since VLSI layout cost ultimately determines the feasibility of an interconnection network. This paper makes an attempt to fill this gap by developing generalized approaches to layout homogeneous product networks.

We say that a product network is *homogeneous* if all its factor networks are isomorphic. Otherwise, the product network is said to be *heterogeneous*. All of the above mentioned examples of product networks are homogeneous and others have been recently introduced, like the product of de Bruijn networks proposed by Rosenberg [19], the product of shuffle-exchange networks proposed by Panwar and Patnaik [18], and the products of complete binary trees networks proposed by Efe and Fernández [7], [8]. Examples of recently proposed heterogeneous product networks are the hyper-de Bruijn network proposed by Ganesan and Pradhan [10], the hyper-Petersen network proposed by Das and Banerjee [6], and the banyan-hypercube network proposed by Youssef and Narahari [26].

Besides these works, the Cartesian product operation has been seen as a potential framework to unify the study of interconnection networks. This allows derivation of general results applicable to any product network. For example, Efe and Fernández [7] presented general results on the degree, distance, diameter, routing, embedding, and partitioning properties of product networks. Baumslag and Annexstein [1] developed generalized off-line permutation routing algorithms for product networks. El-Ghazawi and Youssef [9] developed fault tolerant routing algorithms.

The interest in product networks is due to their elegant mathematical structures, as well as their increased power and versatility. It is shown in [7], for example, that homogeneous product networks constructed from shuffle-exchange, de Bruijn, or complete binary tree graphs are computationally more powerful than their corresponding factor graphs. That is, they can emulate their like-size factor graph with a small constant slowdown, but they can also

1. We use "graph" and "network" interchangeably.

- *A. Fernández is with the Dipartimento de Arquitectura y Tecnoogia de Computadores, Escuela Universitaria de Informatica, Ctra. Valencia Km. 7, 28031 Madrid, Spain.*
- *K. Efe is with the Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, LA 70504-4330. E-mail: efe@cacs.usl.edu*

Manuscript received 31 May 1995; revised 28 Feb. 1997.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 105194.

emulate several other graphs with constant slowdown, which their like-size factor graphs cannot. These graphs are ideally suited for sorting, matrix multiplication, and graph theory problems as they can perform these computations with the same running time complexity as the hypercube, but they cost much less. In addition, heterogeneous product networks appear to combine the advantages of the different factor graphs they contain.

This paper develops generalized methods to obtain lower and upper bounds on the area and the wire length required by VLSI layouts of homogeneous product networks. These are the two most important parameters of a layout, since a large area implies low yield in the fabrication process and long wires imply large communication delays (see [24] for the technological details). Our emphasis on homogeneous product networks is not due to any limitation of the methods presented here. The obtained results may be easily extended to heterogeneous product networks. We comment on how this can be done in the conclusions section of this paper. The emphasis on homogeneous product networks in this paper is due to the fact that it simplifies the discussion.

The VLSI layout model assumed in this paper is the Thompson's model [23]. In this model, the VLSI layout area is seen as a two-dimensional grid into which the network has to be embedded with unit "congestion." The congestion of embedding a "guest" graph G into a "host" graph H is the maximum number of edges of G which must be mapped to any edge of H. In the study of VLSI complexity, the host graph is the two-dimensional grid and the graph representing the network is mapped into it.

The concept of edge congestion is important for the analyses presented in this paper. In particular, when the guest graph is the N-node directed complete graph and the host graph is an arbitrary graph containing N nodes, then we pay special attention to the corresponding congestion of the embedding. In [7], we coined the term "maximal congestion" for the first time in the literature to refer to the congestion of embedding a directed complete graph in an arbitrary graph. The maximal congestion of a graph measures the maximum required amount of congestion to map any graph onto it, because no graph has more edges than the directed complete graph. Maximal congestion is an intrinsic parameter valid for any connected graph, just like the crossing number, the chromatic number, etc., are intrinsic parameters of graphs.

We use upper bounds on the maximal congestion to derive lower bounds on the VLSI layout area and the maximum wire length for homogeneous product networks. We initially show how to obtain an upper bound on the maximal congestion of a homogeneous product graph from an upper bound on the maximal congestion of its factor graph. Then we apply Leighton's methods [14], [15] to obtain lower bounds on the bisection width and the crossing number of the product network. Direct application of results from [23] and [14] yields the desired lower bounds. The maximal congestion has been previously used as an intermediate value obtained in order to compute the bisection width or the crossing number of a graph. Here, we use it as the basic property to be known about the factor graph, because the bisection width or the crossing number of a factor graph do not give enough information to compute lower bounds on the bisection width or the crossing number of the product graph.

Subsequently, we obtain upper bounds on the VLSI layout complexity for product networks with bounded number of dimensions constructed from bounded-degree factor graphs. This is done by using two traditional frameworks: *separators* and *bifurcators*. We initially use a special kind of separators that we call *bisectors* (because they bisect the graph) and show that bisectors for a product network are easily obtained from bisectors of its factor graph. The direct application of results compiled in [24] and [14] allows us to obtain upper bounds on the area and the length of the wires.

Since it is not easy to obtain small separators for an arbitrary graph, we present a similar result for bifurcators. We show how to obtain a bifurcator for a product network, given a bifurcator for its factor network. We also consider some special cases which improve the upper bounds for some special graphs. These results allow us to derive the desired bounds from the results in [2].

Finally, we present a universally applicable method to obtain efficient layouts for product networks based on collinear layouts (layouts with all the nodes along a line) of the factor network. Briefly, we combine collinear layouts for the factor network to obtain a layout for their product. We first show how to obtain efficient collinear layouts for the factor networks and then we present an algorithm to combine them to obtain layouts for product networks with any number of dimensions.

We applied all of these approaches to several well-known networks, as well as others not previously introduced in the literature. The results show that the method based on collinear layouts yields optimal area in most cases (there was just one nonoptimal case among all the cases that we considered, but it was also very close to optimal). The separator approach has yielded optimal area layouts in more cases than the bifurcator, but it turned out to be applicable for only about half of the cases we considered. The bifurcator approach does not generate optimal area layouts as often as the others, but its layouts are larger only by a logarithmic factor of the size of the factor graph. For the maximum wire lengths, the method based on collinear layouts generates optimal upper bounds that match the lower bounds if the number of dimensions is considered bounded. In all cases but one, the wire lengths obtained by this method were shorter than the wire length obtained by using bisectors or bifurcators.

2 Definitions

We start by defining the class of product graphs considered in this paper.

DEFINITION 1. *The r-dimensional product graph, denoted as PG_r, of a graph G is the graph whose vertices comprise all the r-tuples $x = x_r x_{r-1} \ldots x_2 x_1$, such that every x_i, $1 \le i \le r$, is a vertex of G, and whose edges comprise all pairs of vertices (x, y), such that x and y differ in exactly one index position i and (x_i, y_i) is an edge of G.*

In this paper, the number of nodes of the factor graph G is denoted as N. Therefore, the number of nodes of the product graph PG_r is N^r. The diameter of G is denoted as d and its maximum vertex degree is denoted as Δ. The vertex degree of a node u of G is denoted as Δ_u.

We say that an edge (x, y) belongs to dimension i if the nodes incident to it differ only in the ith index position. A G-subgraph of PG_r is said to be a dimension-i subgraph if any two nodes in the subgraph differ only in the ith index position.

Next, we define several properties of networks used in this paper. We start by formally defining the maximal congestion of a network. An *embedding* of a guest graph G onto a host graph H is a one-to-one mapping of the vertices of G onto the vertices of H and a mapping of the edges of G into paths in H. The *congestion* of an embedding is the maximum number of paths (images of the edges of G) that traverse any edge of H.

DEFINITION 2. *The maximal congestion of a graph G is the smallest congestion of any embedding of the N-node directed complete graph onto G.*

In this paper, we are mainly interested in an upper bound on the maximal congestion of a graph. This can be easily obtained as long as there is a routing algorithm for the graph. Just send packets from every node to every other node in the graph and count the maximum number of packets that trace any edge according to the routing algorithm. For the purpose of upper bounds, the routing algorithm does not even need to be optimal, although better routing algorithms may yield better (smaller) upper bounds.

Previously, upper bounds on maximal congestion have been used to obtain lower bounds on the bisection width and crossing number of graphs. For instance, in [14], Leighton introduced a technique that uses upper bounds on the maximal congestion to obtain lower bounds on the crossing number of a graph. A similar technique has been used in [15] to compute lower bounds on the bisection width of a few well-known networks. However, to our knowledge, it was never explicitly defined as being a fundamental property of a graph. In this paper, we use "maximal congestion" as the fundamental property to be known about a factor graph in order to derive lower bounds on the VLSI complexity of its r-dimensional product.

DEFINITION 3. *The bisection width of a graph G is the minimum number of edges that have to be removed from G to obtain two disjoint subgraphs with the same number of nodes (within one).*

DEFINITION 4. *The crossing number of a graph G is the minimum number of edge crossings of any drawing of G in the plane.*

These last two properties of a graph have been traditionally used to obtain lower bounds on the area required by any layout of the graph. We continue by defining the class of separators used in this paper.

DEFINITION 5. *Let $f(x)$ be a monotonically nondecreasing function. A graph G has an $f(x)$-bisector either if it has only one node or if, by removing at most $f(N)$ of its edges, it can be divided into two subgraphs with the same number of nodes (within one), both with $f(x)$-bisectors.*

In general, separators need not bisect the graph at each stage. Our definition is more restrictive, for instance, than the definition of separator used by Leiserson [17]. However, Ullman [24] shows how to obtain a bisector (he calls it strong separator) from separators as defined by Leiserson. We will now define the concept of bifurcator.

DEFINITION 6. *A graph G has an F-bifurcator either if it has only one node or if, by removing at most F of its edges, it can be divided into two subgraphs, both with $F/\sqrt{2}$-bifurcators.*

The definition of bifurcator implies a way to iteratively partition G so that in the ith step of this partition process (with $i = 0$ initially) no more than $F/\sqrt{2}^i$ edges are cut in each partition. Special cases of graphs will be considered based on additional restrictions imposed in this partition process.

DEFINITION 7. *A graph G has an α-special bifurcator, $0 \le \alpha \le 1$, if it has an $O\left(max\left\{\sqrt{N}, N^{\alpha}\right\}\right)$-bifurcator such that no more than $O((N/2^i)^{\alpha})$ edges are cut in each partition at the ith step of the partition process, where $i = 0$ initially.*

Note that, when $\alpha = 1/2$, we have the definition of $\sqrt{N}$-bifurcator, but, for $\alpha \neq 1/2$, the partition process defined is more restrictive than the one implied in Definition 6. We now define a subclass of graphs with α-special bifurcators. This subclass was originally considered in [2].

DEFINITION 8. *A graph G has an α-type B bifurcator if it has an α-special bifurcator, where $\alpha > 1/2$.*

2.1 The Thompson's Grid Model

In this paper, we use the VLSI layout model defined by Thompson in [23]. In this model, the layout area is divided into square "tiles" of unit area, placed in a grid fashion. Each tile can hold either a section of wire, a node, or a wire crossing. The wires of the layout run either horizontally or vertically on this grid. If two wires enter the same tile they must have different directions and they cannot change direction in the tile.

Observe that, since a node is assigned to a tile, the nodes are not allowed to have more than four incident wires. When a node has a degree larger than four, Thompson proposed to model it with a set of adjacent tiles whose perimeter is at least the desired degree. Although the smallest area required to have a perimeter of Δ_u for a node u has $O(\Delta_u)$ tiles, it is much more realistic to assume that a node with vertex degree Δ_u will require area of at least $\Omega(\Delta_u) \times \Omega(\Delta_u)$. In this paper, we shall assume that any node u with vertex degree Δ_u is laid out as a rectangle with sides of length at least $\Omega(\Delta_u)$.

Under this model, the *wire area* of a layout is the number of tiles that hold either a section of a wire or a wire crossing. The *length of a wire* is the number of tiles traversed by the wire from its source node to its destination node. For technological reasons [24], the *layout area* is defined as the area of the smallest rectangle that contains all the allocated tiles of the layout. This value is fully described with the length and the width of this rectangle. In this paper, the

width of a layout is assumed to be the length of the shorter side of the rectangle and the *length* of the layout is the length of the longer side. We also assume that the rectangle is oriented in the grid with the longer side horizontally placed.

Fig. 1 shows an example of layout for K_5, the five-node complete graph, where all the nodes are placed in a horizontal line. This kind of layout is called a *collinear layout*. This layout has width 7, length 11, wire area 55, layout area 77, and maximum wire length 15.

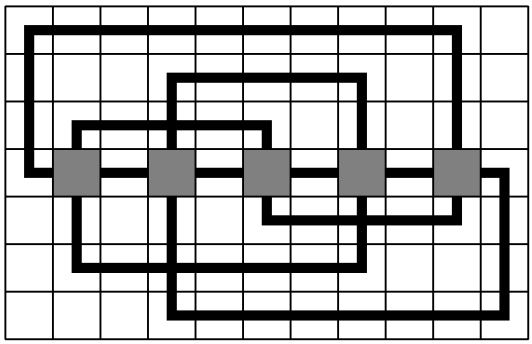

Fig. 1. Collinear layout for K_5.

2.2 Collinear Layouts

A VLSI layout is called collinear if all the nodes are placed along a straight line. We will use collinear layouts of the factor graph to generate layouts for the product graph. To be able to use them, we impose several restrictions to the collinear layouts. We assume that the nodes are aligned horizontally.

DEFINITION 9. *A collinear layout is* seminormal *if all the nodes in the layout are placed at the bottom rows of the layout, a node u occupies* Δ *rows and* Δ_u *columns, and all the wires are laid down above the row* Δ.

DEFINITION 10. *A collinear layout is* normal *if it is seminormal, all the nodes are adjacent, and all the wires are laid down as two vertical sections connected by a horizontal section.*

For these two classes of layouts we can define two new parameters.

DEFINITION 11. *The* wiring width *of a collinear layout is the number of rows used to route the wires in the layout.*

The value of the wiring width is always the width of the layout minus the maximum vertex degree Δ.

DEFINITION 12. *The* bandwidth *of a collinear layout is the maximum distance, in number of nodes, between any two connected nodes.*

The maximum wire length is closely related to the bandwidth of a layout as we discuss later. Fig. 2 presents a normal collinear layout for K_5 with wiring width 6 and bandwidth 4.

3 Lower Bounds

In this section, we obtain lower bounds on the layout area and maximum wire length required by any layout of PG_r.

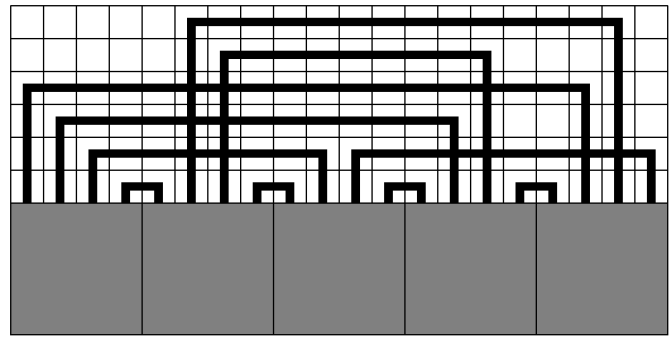

Fig. 2. Normal collinear layout for K_5.

We initially present the following two lemmas, which allow us to obtain bounds on the bisection width and the crossing number of product networks. Both of these lemmas easily follow from Theorem 4 in [7], but we include their proofs here for convenience to the reader, since they were not stated in [7] in exactly the same form that they appear here.

LEMMA 1. *If the maximal congestion of G is k, then the maximal congestion of* PG_r *is, at most,* kN^{r-1}.

PROOF. We show a mapping of the edges of the N^r-node directed complete graph into paths of PG_r. We first map the nodes of the directed complete graph onto the nodes of PG_r one-to-one. Then, we map the directed edge from $x = x_r \cdots x_1$ to node $y = y_r \cdots y_1$ to the path

$$x \to y_r x_{r-1} \cdots x_1 \to \cdots \to y_r \cdots y_2 x_1 \to y.$$

The ith arrow represents the path in the corresponding G-subgraph from x_i to y_i for $i = 1, \cdots, r$. By definition of maximal congestion, these paths imply at most congestion k in the G-subgraph.

Let $(z_r \cdots z_i \cdots z_1, z_r \cdots z_i' \cdots z_1)$ be a dimension-i edge of PG_r. If this edge is traversed by a path from x to y as described, then the edge (z_i, z_i') can be traversed by, at most, k paths between two nodes of G. If this dimension-i edge is used for routing an edge of the directed complete graph, say from x to y, then y can differ from x in, at most, $r - 1$ symbol positions other than dimension i. Moreover, each differing symbol position can have N possible values. Therefore, an edge of G can be traversed by at most kN^{r-1} paths. □

LEMMA 2. *If the maximal congestion of G is k, then the bisection width of* PG_r *is at least* $\frac{N^{2r}-1}{2kN^{r-1}}$.

PROOF. The bisection width of the N^r-node directed complete graph is $N^{2r}/2$ if N is even and $(N^{2r} - 1)/2$ if N is odd. Since, from Lemma 1, we can embed it onto PG_r with congestion at most kN^{r-1}, the bisection width of PG_r has to be at least $\frac{N^{2r}/2}{kN^{r-1}} = N^{r+1}/2k$ if N is even, or $(N^{2r} - 1)/2kN^{r-1}$ if N is odd, because, otherwise, we

could bisect the embedded directed complete graph by removing fewer edges than its bisection width, which is a contradiction. Whether N is even or odd, the claimed value holds as a lower bound on the bisection width of PG_r. □

We note that it is not possible to obtain a result similar to Lemma 2 by just knowing the bisection width of G. In other words, we cannot say that "if G has bisection width B_1, then PG_r has bisection width at least B_2," because bisection of a product graph does not have to use individual bisections of its factor networks. In this sense, the maximal congestion carries more information than the bisection width.

LEMMA 3. *If G has E edges and its maximal congestion is k, then the crossing number of PG_r is at least*

$$\frac{\left(N^r-1\right)\left(N^r-2\right)\left(N^r-3\right)}{20k^2N^{r-2}}-\frac{rEN^{r-1}}{2}.$$

PROOF. Extending the results in [11], [12], [13], it was shown in [22] that, if an N-node graph G has E edges and the N-node complete graph can be embedded onto G with no edge having congestion more than k, then its crossing number is at least $\frac{N(N-1)(N-2)(N-3)}{20k^2}-\frac{E}{2}$. The condition on edge congestion specified here directly corresponds to the maximal congestion of G, thus we conclude that any N-node graph with E edges and maximal congestion of k has crossing number at least $\frac{N(N-1)(N-2)(N-3)}{20k^2}-\frac{E}{2}$.

We know from Lemma 1 that the maximal congestion of PG_r is at most kN^{r-1}. We also know from [7] that, if G has E edges, then PG_r has rEN^{r-1} edges. Using these, we obtain the claimed lower bound on the crossing number of PG_r. □

In [23], Thompson showed that the square of the bisection width is a lower bound (within a constant factor) on the wire area required by any layout of a graph. Similarly, Leighton [14] presented the crossing number as a lower bound on the wire area of any layout of a graph. Then, we can use any of the last two lemmas to prove the following theorem, which is one of the two main results of this section.

THEOREM 1. *If the maximal congestion of G is k, then the layout area of PG_r is at least $\Omega\left(\frac{N^{2(r+1)}}{k^2}\right)$.*

The second result of this section gives a lower bound on the length of the longest wire in any layout of a product graph.

THEOREM 2. *If the maximal congestion of G is k and its diameter is d, then the length of the longest wire in any layout of PG_r is at least $\Omega\left(\frac{N^{r+1}}{krd}\right)$.*

PROOF. Theorem 5-2 in [14] shows that any layout of a graph with diameter D and minimum layout area A has some wire of length at least $A^{1/2}/3D$. The diameter of PG_r is $D = rd$ [7] and, from Theorem 1, its layout area is at least $\Omega\left(\frac{N^{2(r+1)}}{k^2}\right)$. Therefore, we can conclude that any layout of PG_r has some wire of length at least $\frac{\Omega\left(N^{r+1}/k\right)}{3rd}=\Omega\left(\frac{N^{r+1}}{krd}\right)$. □

4 UPPER BOUNDS

In this section, we first present upper bounds obtained by traditional frameworks, namely bisectors and bifurcators. We show that, given a bisector or a bifurcator for the factor graph, we can obtain a bisector or a bifurcator for the product graph. Since these frameworks are only applicable to networks with bounded vertex degree, we will assume that the factor graph has bounded vertex degree and that the number of dimensions of the product network is also bounded. These assumptions are not very restrictive because all product networks obtained from fixed degree networks can grow without increasing the vertex degree.

Sherlekar and JáJá [20], [21] have investigated the use of separators and bifurcators to obtain efficient layouts for unbounded vertex degree graphs. However, the kinds of separators and bifurcators they use are so restrictive that it does not seem possible to obtain simple general results for product graphs by using them.

Subsequently, we present another approach that is universally applicable and does not have any restriction on the vertex degree or on the number of dimensions. As we mentioned, this method of obtaining efficient layouts for product networks is based on the existence of efficient collinear layouts for the factor networks. We show that it is always possible to find reasonably efficient collinear layouts for any network and present several techniques to do so.

4.1 Upper Bounds Based on Bisectors

The following theorem presents the basic result of this section.

THEOREM 3. *If G has an $f(x)$-bisector, then PG_r has an $O(x^{(r-1)/r}f(x^{1/r}))$-bisector.*

PROOF. We initially present the following lemma that shall be used in the proof.

LEMMA 4. *If G has an $f(x)$-bisector, then it has at most $O(Nf(N))$ edges.*

PROOF. Assume, for simplicity, that N is a power of two. By the definition of bisector, G can be divided into two subgraphs by removing no more than $f(N)$ edges. Then, we obtain two subgraphs with $N/2$ nodes each, which can be bisected by removing no more that $f(N/2)$ edges from each. After i bisections of this kind, we obtain 2^i subgraphs with $N/2^i$ nodes each, which, in turn, can be bisected by removing no more that $f(N/2^i)$ edges from each. After applying the bisection process log N times[2] we obtain N isolated nodes. The maximum number of edges cut in the whole process can be easily computed as,

$$f(N)+2f\left(N/2\right)+2^2f\left(N/2^2\right)+\cdots+2^{\log N-1}f\left(N/2^{\log N-1}\right)$$
$$=\sum_{i=0}^{\log N-1}2^if\left(N/2^i\right)=O\left(Nf(N)\right).$$

□

2. All logarithms are to the base two.

The proof of the theorem shows how to divide PG_r into isolated nodes by repeatedly applying bisections that respect the definition of $O(x^{(r-1)/r} f(x^{1/r}))$-bisector.

Initially, we show how to divide PG_r into 2^r disjoint subgraphs and, possibly, some isolated nodes. This process is done in r bisection steps, each of which cuts $O(N^{r-1} f(N))$ edges from its corresponding graph. Each of the obtained subgraphs is the r-dimensional (possibly heterogeneous) product of factor graphs with $\lfloor N/2 \rfloor$ nodes and $f(x)$-bisectors.

A partition process similar to the one applied to PG_r can then be applied to each of the subgraphs, to the subgraphs obtained from them, and so on, until all the nodes are isolated.

The basic partition process considers two cases, when N is even and when N is odd.

Case N even: By definition of bisector, each of the G-subgraphs in each dimension can be bisected by removing no more than $f(N)$ edges. We can initially consider only the G-subgraphs in dimension one. PG_r can be divided into two subgraphs with the same number of nodes in each by bisecting each of the dimension-one G-subgraphs. As there are N^{r-1} such subgraphs, we have cut no more than $N^{r-1} f(N)$ edges from the N^r-node graph.

Now we can take one of the two subgraphs of PG_r obtained and divide it into two subgraphs with same number of nodes by bisecting each of its dimension-two G-subgraphs. The number of edges cut this time is no more than $N^{r-1} f(N)/2$ from a graph with $N^r/2$ nodes.

We can continue this process, bisecting the obtained subgraphs along each dimension. When bisecting the subgraphs by dimension i, we are removing no more than $N^{r-1} f(N)/2^{i-1} = O(N^{r-1} f(N))$ edges from $N^r/2^{i-1} = \Theta(N^r)$-node graphs.

After bisecting the subgraphs by dimension r, we obtain 2^r disjoint subgraphs, each being the r-dimensional product of $N/2$-node graphs with $f(x)$-bisectors (because they are bisections of graphs with $f(x)$-bisectors).

Case N odd: The logic in this case is similar to the logic in the above case, but we must be careful because, by simply bisecting each subgraph along a dimension, we are not bisecting the whole graph. What we do in this case is breaking each G-subgraph in a given dimension into two subgraphs, with $(N-1)/2$ nodes each, and one isolated node. As the isolated nodes are connected between themselves by the other dimensions, we also cut these connections and distribute the so obtained isolated nodes evenly between the two large subgraphs.

We can initially take dimension one. By bisecting each dimension-one G-subgraph, we cut no more than $N^{r-1} f(N)$ edges and we obtain two subgraphs with $N^{r-1}(N-1)/2$ and $N^{r-1}(N+1)/2$ nodes, respectively. Clearly, PG_r has not been bisected. Now we can take the subgraph with the larger number of nodes and isolate one node along dimension one from each of the dimension-one subgraphs, the same node in each subgraph. Since we are assuming that G has bounded vertex degree, we can do so by cutting a bounded number of edges from each dimension-one subgraph. This leads to a total of $O(N^{r-1})$ edges cut.

Now we have two subgraphs with $N^{r-1}(N-1)/2$ nodes each, and an $(r-1)$-dimensional subgraph, isomorphic to PG_{r-1}. From Lemma 4, the factor graph G that generates the $(r-1)$-dimensional subgraph has no more than $O(Nf(N))$ edges. Therefore, we can isolate the nodes of this subgraph by removing, at most, $(r-1)N^{r-2}O(Nf(N)) = O(N^{r-1} f(N))$ edges.

As a result of the above process, we have two subgraphs with the same number of nodes and some isolated nodes. If we distribute the isolated nodes evenly between the two subgraphs, our bisection is done. The total number of edges cut has been $N^{r-1} f(N) + O(N^{r-1}) + O(N^{r-1} f(N)) = O(N^{r-1} f(N))$ from the initial N^r-node graph.

This process can be applied to each dimension, as in the case of N even. In each application, $O(N^{r-1} f(N))$ edges are cut from an $\Theta(N^r)$-node graph. After the graph has been bisected in this way along each dimension, we have 2^r disjoint r-dimensional subgraphs, each being the product of $(N-1)/2$-node graphs with $f(x)$-bisector, plus several isolated nodes distributed evenly between them.

We now have 2^r subgraphs of PG_r, each being the r-dimensional product of factor graphs, with $\lfloor N/2 \rfloor$ nodes and $f(x)$-bisectors. Note that, in the above described process, we only use the fact of PG_r having the same number of nodes along each dimension and the fact of each factor graph having an $f(x)$-bisector. Since the obtained subgraphs fulfill these requirements, the described process can be applied again to each of them. Subsequently, the subgraphs obtained from them will also fulfill the requirements, and the process can be applied to each of them, and so on, until all the nodes are isolated.

Since, in each bisection of the whole process, the number of edges cut does not exceed the limits imposed by the definition of $f(x)$-bisector for $f(x) = O(x^{(r-1)/r} f(x^{1/r}))$, the proof is complete. □

Once we obtain a bisector for the product network, we are ready to apply it to obtain bounds on the layout parameters. We can use Theorem 3.5 in [24], which derives upper bounds on the layout area of networks with a given bisector, to directly obtain the following result.

THEOREM 4. *If G has an $f(x)$-bisector, then PG_r can be laid out in a square of side $O(Nf(N)\log N)$ when $r = 2$, or side $O(N^{r-1} f(N))$ when $r > 2$.*

PROOF. Theorem 3.5 in [24] states that any m-node graph with a $g(x)$-bisector and bounded degree can be laid out in an area of side

$$O\left(max\left(\sqrt{m},\ \sum_{i=0}^{\log_4 m} 2^i g\left(m/4^i\right) \right) \right).$$

From Theorem 3, we have obtained that PG_r has an $O(x^{(r-1)/r}f(x^{1/r}))$-bisector. PG_r has N^r nodes and we can obtain the value of the summation as

$$\sum_{i=0}^{r\log_4 N} 2^i O\left(\left(N^r/4^i\right)^{(r-1)/r} f\left(N/4^{i/r}\right)\right) = O\left(f(N)\sum_{i=0}^{r\log_4 N} 2^i\left(\frac{N^r}{4^i}\right)^{(r-1)/r}\right),$$

since $f(x)$ is a monotonically nondecreasing function. The value of this last summation is $O(N\log N)$ when $r = 2$, or $O(N^{r-1})$ when $r > 2$ [24]. Therefore, the value of the first summation is $O(Nf(N)\log N)$ when $r = 2$, or $O(N^{r-1}f(N))$ when $r > 2$, and the claim follows. □

The most studied kind of bisectors has been $O(x^\alpha)$-bisectors, for bounded α. The results of Leiserson [16], [17] and Valiant [25] imply that any m-node graph with an $O(x^\alpha)$-bisector can be laid out in area $O(m)$ when $\alpha < 1/2$, in area $O(m\log^2 m)$ when $\alpha = 1/2$, or in area $O(m^{2\alpha})$ when $\alpha > 1/2$. Bhatt and Leiserson [3] have also shown that these upper bounds can be reached with maximum wire lengths of $O\left(\sqrt{m}/\log m\right)$, $O\left(\sqrt{m}\log m/\log\log m\right)$, or $O(m^\alpha)$, respectively. These results can be directly applied to product networks to obtain the next corollary (note, m is N^r here).

COROLLARY 1. *If G has an $O(x^\alpha)$-bisector, for bounded α, then PG_r can be laid out in an area of $O(N^2\log^2 N)$ with maximum wire length $O(N\log N/\log\log N)$ when $\alpha = 0$ and $r = 2$, or in an area of $O(N^{2(r+\alpha-1)})$ with maximum wire length $O(N^{r+\alpha-1})$ otherwise.*

4.2 Upper Bounds Based on Bifurcators

The following theorem and its corollary present the initial general results of this section. After these, we present additional results applicable to graphs with α-special bifurcators, which yield tighter bounds.

THEOREM 5. *If G has an F-bifurcator, then PG_r has an $N^{r-1}6\left(2+\sqrt{2}\right)F$-bifurcator.*

PROOF. From Theorem 6 in [2], we know that, if G has an F-bifurcator, then it has an $H = 6\left(2+\sqrt{2}\right)F$-bifurcator (balanced bifurcator) that bisects the graph at each partition. Then, after, at most, $\log N + 1$ partitions, G is transformed into N isolated nodes. Throughout the rest of the proof, we will denote $6\left(2+\sqrt{2}\right)F$ as H for brevity.

The proof is very similar to the proof of Theorem 3. We show that, given PG_r, we can obtain 2^r subgraphs, each being the r-dimensional (possibly heterogeneous) product of factor graphs with $H/\sqrt{2}$-bifurcators and, at most, $\lceil N/2\rceil$ nodes.

We initially consider dimension one. To partition PG_r, we can bisect each G-subgraph in this dimension, cutting no more than HN^{r-1} edges in total. Each dimension-one G-subgraph is so divided into an $\lfloor N/2\rfloor$-node and an $\lceil N/2\rceil$-node subgraphs. To follow the worst case, we take the larger of the two obtained subgraphs. We can partition this subgraph by dimension two by cutting at most $H\lceil N/2\rceil N^{r-2}$ edges. This value is smaller than $HN^{r-1}/\sqrt{2}$.

We can continue in this way, partitioning the subgraphs by each dimension. When dividing the largest subgraph by dimension i, we cut, at most, $H\lceil N/2\rceil^{i-1}N^{r-i}$ edges, that is, smaller than $HN^{r-1}/\sqrt{2}^{i-1}$. After dividing by dimension r, each subgraph obtained has, at most, $\lceil N/2\rceil$ nodes along each dimension.

If we start the process again, the next division will cut, at most, $H\lceil N/2\rceil^{r-1}/\sqrt{2}$ edges, that is smaller than $HN^{r-1}/\sqrt{2}^r$. Therefore, the process can be repeated without exceeding the number of edges allowed by the definition of bifurcator.

As in the proof of Theorem 3, we can apply the partition process just described to each of the 2^r subgraphs of PG_r obtained, to the subgraphs obtained from them, and so on, until all the nodes are isolated. By repeating this process, at most, $\log N + 1$ times, all the nodes in PG_r will be isolated, and the theorem follows. □

We recall here that Bhatt and Leighton [2] showed that, if G can be laid out in an area A, it has a $\sqrt{A}$-bifurcator. Thus, if G can be laid out in an area A, then PG_r has an $N^{r-1}6\left(2+\sqrt{2}\right)\sqrt{A}$-bifurcator. From Theorem 5, we can obtain bifurcator-based bounds for the area and maximum wire length by using the results from [2].

COROLLARY 2. *If G has an F-bifurcator, then PG_r can be laid out in an area of $O(N^{2(r-1)}F^2\log^2(N/F))$ with maximum wire length $O\left(N^{r-1}F\frac{\log(N/F)}{\log\log\left(N/6(2+\sqrt{2})F\right)}\right)$.*

The above theorem and corollary are universally applicable. However, as Bhatt and Leighton [2] noted, there are graphs with special characteristics which allow to improve the above bounds. This fact is reflected in the following results.

THEOREM 6. *If G has an α-special bifurcator, then PG_r has an $O(N^{r+\alpha-1})$-bifurcator. This is an $(r+\alpha-1)$-type B bifurcator if either $r > 2$ or $r = 2$ and $\alpha > 0$.*

PROOF. Let N be a power of two for simplicity. From Theorem 5 in [2], we know that G has a partition process where each partition in the ith partition step bisects the corresponding graph without cutting more than $6\sum_{s=i}^{p} O\left(\left(N/2^s\right)^\alpha\right)$ edges, where $0 \le i \le \log N$ and p is

the number of steps of the original partition process. This summation is a decreasing geometric series, that is, essentially on the order of its first term. Then, G has a partition process such that each partition in the ith partition step bisects the corresponding graph without cutting more than $6O((N/2^i)^\alpha) = O((N/2^i)^\alpha)$ edges.

The partition process is similar to the one presented in the proof of Theorem 5. We apply a basic process log N times, partitioning the graphs r times in each application. We will use i to count the applications of the basic process, i varying from 0 to log $N - 1$, and j to count the partitions within an application of the basic process, varying j from 0 to $r - 1$. The absolute count of partition steps for the complete graph is then $k = ir + j$.

In the ith application of the basic process, the size of the factor subgraphs that we are considering is $m = N/2^i$ and we cut at most $O((N/2^i)^\alpha)$ edges to partition this subgraph. Then, the kth absolute partition step cuts at most $O((N/2^i)^\alpha (N/2^i)^{r-j-1}(N/2^{i+1})^j)$ edges. We can write

$$\left(N/2^i\right)^\alpha \left(N/2^i\right)^{r-j-1}\left(N/2^{i+1}\right)^j = \frac{N^{r+\alpha-1}}{2^{k-i(1-\alpha)}}.$$

Since $k - i(1 - \alpha) \geq k/2$ for $r \geq 2$, we can conclude that PG_r has an $O(N^{r+\alpha-1})$-bifurcator. We still need to show in which cases this is an $(r + \alpha - 1)$-type B bifurcator. Note that

$$\frac{N^{r+\alpha-1}}{2^{k-i(1-\alpha)}} = \frac{\left(N^r\right)^{1-(1-\alpha)/r}}{\left(2^k\right)^{1-i(1-\alpha)/k}}.$$

Since $(1 - \alpha)/r \geq i(1 - \alpha)/k$, then $1 - i(1 - \alpha)/k \geq 1 - (1 - \alpha)/r$ and, therefore,

$$\frac{\left(N^r\right)^{1-(1-\alpha)/r}}{\left(2^k\right)^{1-i(1-\alpha)/k}} = O\left(\left(N^r/2^k\right)^{1-(1-\alpha)/r}\right),$$

where $1 - (1 - \alpha)/r > 1/2$ (i.e., we have a type B bifurcator) if either $r > 2$ or $r = 2$ and $\alpha > 0$. □

We can now apply the results of [2] combined with this theorem to obtain the following corollary.

COROLLARY 3. *If G has an α-special bifurcator, then PG_r can be laid out in an area of $O(N^2\log^2 N)$, with maximum wire length $O\left(N\frac{\log N}{\log\log N}\right)$, if $r = 2$ and $\alpha = 0$, or in an area of $O(N^{2(r+\alpha-1)})$ with maximum wire length $O(N^{r+\alpha-1})$ otherwise.*

4.3 Upper Bounds Based on Collinear Layouts

In this section, we present the collinear approach to obtain layouts for product networks. The collinear approach has several advantages:

1) It gives the optimal area layouts for all the cases we considered in Section 5 of this paper (with one exception), and wire lengths were quite close to optimal.
2) It depends on obtaining a collinear layout for the factor graph, which is much easier than obtaining good bisectors or bifurcators. The area of the layout only depends on the wiring width of the collinear layout. The maximum wire length depends also on the bandwidth of the collinear layout. Below, we present several ways to obtain normal collinear layouts with small wiring width for arbitrary graphs.
3) It is applicable for any graph regardless of the vertex degree, while the applicability of bisector or bifurcator–based approaches are limited to graphs with bounded degree.
4) The aspect ratio of layouts are always $O(1)$, which is a desirable characteristic for fabrication.

4.3.1 Methods for Obtaining Normal Collinear Layouts

In this section, we are interested in obtaining normal collinear layouts with small wiring width and small bandwidth for the factor graph G, because these are the properties that influence the cost/performance characteristics of the layout of the product graph that we will obtain. We can devise several methods to obtain normal collinear layouts for any graph.

First, observe that any graph G has a normal collinear layout of wiring width at most $\frac{1}{2}\sum_{u\in V}\Delta_u$, where V is the set of nodes of G, since this is the number of wires in the layout and each wire requires no more than one row.

Second, the problem of finding an efficient collinear layout is closely related to the class of problems known as "graph labeling" [5]. Given a way of labeling the nodes of an N-node graph with integer labels 1, ⋯, N (or, equivalently, given a way of placing the nodes of the graph on a line), the maximum distance between two connected nodes is the *bandwidth* of the labeling, while the maximum number of edges that cross a vertical line placed between any two nodes is the *cutwidth* of the labeling. Thus, if there is an embedding of a graph G onto the N-node linear array with bandwidth b and cutwidth c, it is trivial to obtain a normal collinear layout for G with wiring width c and longest edge of length $O(b\Delta + c)$. For an arbitrary graph, we can obtain a labeling which minimizes the bandwidth and the cutwidth by using dynamic programming algorithms or heuristics.

Third, it is shown in [17] how to construct normal collinear layouts for a graph G with an $f(x)$-separator. The layout has wiring width $O(f(N)\log N)$ in general, but, if $f(x) = \Omega(x^\alpha)$ for $\alpha > 0$, then the wiring width is $O(f(N))$. For graphs with F-bifurcators, we obtain a similar result in the following lemma.

LEMMA 5. *If G has an F-bifurcator, then it has a normal collinear layout with wiring width $O(F)$.*

PROOF. The construction of the layout is similar to the construction shown in [17] for separators. We use a divide-and-conquer process that divides the graph into two subgraphs, obtains a collinear layout for each, and reconnects the two layouts by adding, at most, as many new rows as the edges which were cut in the partition step. From the definition of bifurcator, the division process is applied at most 2 log F times before we obtain isolated nodes, and at step i at most $F/\sqrt{2^i}$ edges

are cut to divide the graph, for $0 \le i \le 2 \log F$. Then, the number of rows needed to route the edges of G are at most $\sum_{i=0}^{2\log F} F/\sqrt{2^i} = O(F)$. □

Finally, we present a general method to obtain a normal collinear layout from an arbitrary layout. The following result shows that there is always a seminormal collinear layout with small wiring width.

LEMMA 6. *If G has a layout of length l and with w, G also has a seminormal collinear layout of length $O(l + N\Delta)$ and wiring width $O(w)$.*

PROOF. We prove the lemma by showing how to transform the given layout into a seminormal collinear layout with the claimed dimensions. The transformation is illustrated in Fig. 3, where we show only the process for one node of the layout. The appearance of this node in the original layout is shown in Fig. 3a.

We first transform the nodes of the given layout by adding enough rows so that each node uses at least Δ_u rows where Δ_u is the vertex degree of the node u which is being transformed. The width of the resulting layout is at most $O(w)$. Fig. 3b shows the result of enlarging our example node to use two rows.

We subsequently create Δ new rows at the bottom of the layout. We will eventually move all the nodes in the layout to these new rows. No other rows are added in the rest of the transformation process, therefore, the wiring width of the new layout will be $O(w)$. At the bottom of Fig. 3c, the three new rows introduced for our example can be observed. We assumed, then, $\Delta = 3$.

Then, the following step is applied iteratively until all the nodes are in the created bottom rows and we have a seminormal collinear layout. The step searches from left to right for the first column with tiles assigned to nodes not yet moved. This column can have tiles from several nodes, if so we take one node arbitrarily.

Let u be the node we have chosen. Create Δ_u new columns on the left side of u. When creating these columns, do not stretch the wires incident to u across the columns just created. Fig. 3c presents the two new columns created in this step. Then, move u to the bottom rows, after resizing it to $\Delta_u \times \Delta$. Finally, use the newly created columns, as well as the rows originally allocated to u, to reroute the edges from the bottom rows. Since u had at least Δ_u rows and we have Δ_u columns, this rerouting can be done. Fig. 3d presents the final result for our example.

This ends the transformation. Note that the total number of added columns is $\sum_{u \in V} \Delta_u$, where V is the set of nodes of G, and, therefore, the length of the layout is $O(l + N\Delta)$. □

The above lemma shows that any layout can be transformed into a seminormal collinear layout with wiring width of the same order as the width of the original layout. While the transformation increases the length of the collinear layout, we will see that it is the width of the normal collinear layout which dominates the layout area complexity for the product graph. The collinear layout obtained can now be compressed to obtain a normal collinear layout with, at most, same wiring width. This is shown in the following lemma.

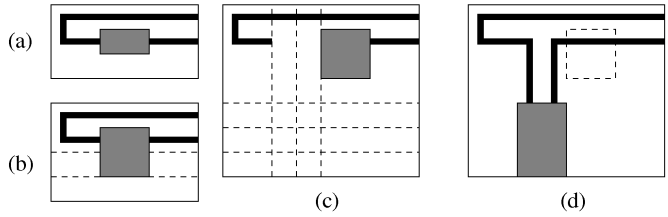

Fig. 3. Transformation of a compact layout into a collinear layout.

LEMMA 7. *If G has a seminormal collinear layout with wiring width w and bandwidth b, it also has a normal collinear layout with wiring width, at most, w and bandwidth b.*

PROOF. The original layout gives us a possible labeling (i.e., the order in which the nodes of G can be placed) to obtain the desired wiring width w. This is all we need for the purpose of obtaining the desired normal layout. We start by placing the N nodes touching each other along a straight line. The ith node in this line corresponds to the ith node in the seminormal collinear layout. We then connect these nodes by three-segment wires (two vertical and one horizontal), as required by the original layout.

Since there is a seminormal collinear layout of width w that uses the same node order, we can obtain a layout which has at most w rows used for wires. The bandwidth of the layout remains the same. □

Note that, in the above obtained layout, the length of the longest wire is at most $2w + b\Delta$, where b is the bandwidth of the layout.

We finish this section by presenting a lower bound on the wiring width of any normal collinear layout for arbitrary graphs.

THEOREM 7. *If the maximal congestion of G is k, then the wiring width for any normal collinear layout of G is at least $N^2/2k$.*

PROOF. Note first that any embedding of the N-node directed complete graph onto the N-node linear array requires congestion at least $N^2/2$. Since the N-node directed complete graph can be embedded onto the graph G with congestion k, it follows that any embedding of G onto the N-node linear array requires congestion at least $N^2/2k$, since, otherwise, we can obtain an embedding of the directed complete graph with congestion smaller than $N^2/2$.

Since the congestion of any embedding of G onto the linear array is a lower bound on the number of rows needed to route the edges of G, the result follows. □

4.3.2 The Layout Method for Product Graphs

The following theorem represents the main result of this section. The proof gives an algorithm to obtain the layout for a product graph from a normal collinear layout of its factor graph.

THEOREM 8. *If G has a normal collinear layout with wiring width w, then PG_r has a layout with square nodes of side $\Delta\lceil r/2\rceil$ placed regularly in $N^{\lceil r/2\rceil}$ columns of $N^{\lfloor r/2\rfloor}$ nodes each, where two adjacent columns of nodes are at distance $w\sum_{i=0}^{\lfloor r/2\rfloor-1} N^i$ and two adjacent rows of nodes are at distance $w\sum_{i=0}^{\lceil r/2\rceil-1} N^i$.*

PROOF. We show the iterative process that can be used to obtain the desired layout. The proof is illustrated in Fig. 4, which presents the construction of a layout for the three-dimensional product of two-node linear arrays, which is isomorphic to the three-dimensional hypercube. Fig. 4a presents a normal collinear layout for the two-node linear array.

Initially, we place the N^r nodes of PG_r in the layout as squares of side $\Delta\lceil r/2\rceil$ in a grid fashion with $N^{\lceil r/2\rceil}$ columns of nodes and $N^{\lfloor r/2\rfloor}$ rows of nodes. Each node touches its neighbor nodes in the layout. The size of the nodes will guarantee that there are enough connection points in each side of the node when needed. Fig. 4b presents this initial situation for our example graph.

For each row of nodes we apply the following iterative process. We start by creating w new rows above the row of nodes. The nodes in the row are divided in $N^{\lceil r/2\rceil-1}$ groups of N adjacent nodes each, and the nodes in each group are connected using the created rows with the wires laid down as specified by the normal collinear layout of G. This completes the connections for the first dimension of the product graph. We subsequently create wN new rows, divide the nodes in a row into $N^{\lceil r/2\rceil-2}$ groups of N^2 adjacent nodes each, and use the wN new rows in groups of w each to connect N nodes of the second dimension. These nodes are N nodes apart from one another.

In the ith iteration, we create wN^{i-1} new rows, divide the nodes in $N^{\lceil r/2\rceil-i}$ groups of N^i nodes each, and connect sets of N nodes in the ith dimension, each N^{i-1} nodes apart from one another.

This process is applied $\lceil r/2\rceil$ times for each row of nodes. The total number of wiring rows created is $w\sum_{i=0}^{\lceil r/2\rceil-1} N^i$. This is the distance between two rows of processors. Two adjacent processors in the same row are still touching each other. Fig. 4c presents the example layout after completion of the above process. To obtain this layout, we applied the iterative step twice.

The same iterative process can be applied to connect the columns. As a result, we find that the columns of processors are at distance $w\sum_{i=0}^{\lfloor r/2\rfloor-1} N^i$. This completes the proof. Fig. 4d shows the final layout obtained for our example graph. □

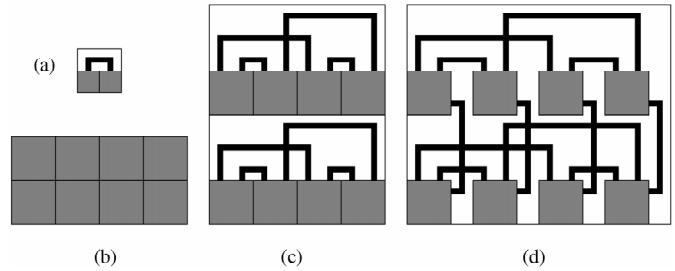

Fig. 4. Layout for the three-dimensional hypercube.

From this theorem, we can obtain bounds on the area and maximum wire length for the layout.

COROLLARY 4. *If G has a normal collinear layout with wiring width w and bandwidth b, then PG_r can be laid out in an area of dimensions $\Theta(w\,N^{r-1}) \times \Theta(w\,N^{r-1})$ with maximum wire length $\Theta(bwN^{r-2})$.*

PROOF. The length of the layout obtained from the above theorem along the horizontal dimension is

$$N^{\lceil r/2\rceil}\left(\Delta\lceil r/2\rceil + w\sum_{i=0}^{\lfloor r/2\rfloor-1} N^i\right).$$

Since $w \geq \Delta/2$, $\sum_{i=0}^{\lfloor r/2\rfloor-1} N^i = \Theta\left(N^{\lfloor r/2\rfloor-1}\right)$, and $N^{\lfloor r/2\rfloor-1} \geq \lceil r/2\rceil$ for $N \geq 2$ and $r \geq 2$, then this length is $\Theta(wN^{r-1})$. The length along the vertical dimension is

$$N^{\lfloor r/2\rfloor}\left(\Delta\lceil r/2\rceil + w\sum_{i=0}^{\lceil r/2\rceil-1} N^i\right) = \Theta\left(wN^{r-1}\right).$$

Similarly, the length of the longest edge is at most

$$2w\sum_{i=0}^{\lceil r/2\rceil-1} N^i + b\left(N^{\lceil r/2\rceil-1}\left(\Delta\lceil r/2\rceil + w\sum_{i=0}^{\lfloor r/2\rfloor-1} N^i\right)\right) = \Theta\left(bwN^{r-2}\right).$$

□

5 APPLICATION OF THE BOUNDS TO VARIOUS NETWORKS

In Tables 1 and 2, we have compiled the bounds obtained by applying the presented results to several networks. Table 1 presents the bounds on layout area, while Table 2 presents the bounds on maximum wire length. In this section, we will present how these bounds have been obtained.

The second column in both tables presents the lower bounds obtained by direct application of Theorems 1 and 2. We first obtain an upper bound on the maximal congestion for each factor network. The maximal congestions of the linear array and the complete binary tree are easily found to be $O(N^2)$. The analysis of lower bounds on the bisection width in [15] implied that the maximal congestions of the shuffle-exchange, de Bruijn, butterfly, and cube-connected cycles (CCC) networks are $O(N \log N)$. The bounds for the hypercube are obtained from the fact of being the product of the two-node linear array. It can be also seen that the maximal congestion of K_N is $O(1)$ (actually, two).

TABLE 1
BOUNDS ON THE LAYOUT AREA OBTAINED BY APPLICATION OF THE PRESENTED METHODS

Factor network	Lower bounds	Upper bound for the area of product network				
		Bisector ($r = 2$)	Bisector ($r > 2$)	Bifurcator ($r = 2$)	Bifurcator ($r > 2$)	Collinear
Linear array	$\Omega(N^{2(r-1)})$	$O(N^2 \log^2 N)$	$O(N^{2(r-1)})$ *	$O(N^2 \log^2 N)$	$O(N^{2(r-1)})$ *	$O(N^{2(r-1)})$ *
C. binary tree	$\Omega(N^{2(r-1)})$	$O(N^2 \log^2 N)$ *	$O(N^{2(r-1)})$ *	$O(N^2 \log^2 N)$ *	$O(N^{2(r-1)})$ *	$O(N^{2(r-1)} \log^2 N)$
Shuffle-Exchange	$\Omega(\frac{N^{2r}}{\log^2 N})$	UN		$O(N^{2r} \frac{\log^2 \log N}{\log^2 N})$		$O(N^{2r}/\log^2 N)$ *
de Bruijn	$\Omega(\frac{N^{2r}}{\log^2 N})$	UN		$O(N^{2r} \frac{\log^2 \log N}{\log^2 N})$		$O(N^{2r}/\log^2 N)$ *
Butterfly	$\Omega(\frac{N^{2r}}{\log^2 N})$	$O(N^4)$	$O(\frac{N^{2r}}{\log^2 N})$ *	$O(N^{2r})$		$O(N^{2r}/\log^2 N)$ *
CCC	$\Omega(\frac{N^{2r}}{\log^2 N})$	$O(N^4)$	$O(\frac{N^{2r}}{\log^2 N})$ *	$O(N^{2r})$		$O(N^{2r}/\log^2 N)$ *
Hypercube	$\Omega(2^{2(r-1)})$	N.A.		N.A.		$O(2^{2(r-1)})$ *
K_N	$\Omega(N^{2(r+1)})$	N.A.		N.A.		$O(N^{2(r+1)})$ *

UN stands for "unknown," N.A. stands for "not applicable." The upper bounds marked with "" are optimal.*

TABLE 2
BOUNDS ON THE WIRE LENGTH OBTAINED BY APPLICATION OF THE PRESENTED METHODS

Factor network	Lower bounds	Upper bound for the wire length of product network				
		Bisector ($r = 2$)	Bisector ($r > 2$)	Bifurcator ($r = 2$)	Bifurcator ($r > 2$)	Collinear
Linear array	$\Omega(N^{r-2}/r)$	$O(N \frac{\log N}{\log \log N})$	$O(N^{r-1})$	$O(N \frac{\log N}{\log \log N})$	$O(N^{r-1})$	$O(N^{r-2})$ *
C. binary tree	$\Omega(\frac{N^{r-1}}{r \log N})$	$O(N \frac{\log N}{\log \log N})$	$O(N^{r-1})$	$O(N \frac{\log N}{\log \log N})$	$O(N^{r-1})$	$O(N^{r-1} \log N)$
Shuffle-Exchange	$\Omega(\frac{N^r}{r \log^2 N})$	UN		$O(N^r \frac{\log \log N}{\log N \log \log \log N})$		$O(N^r/\log N)$
de Bruijn	$\Omega(\frac{N^r}{r \log^2 N})$	UN		$O(N^r \frac{\log \log N}{\log N \log \log \log N})$		$O(N^r/\log N)$
Butterfly	$\Omega(\frac{N^r}{r \log^2 N})$	$O(N^r)$		$O(N^r)$		$O(N^r/\log^2 N)$ *
CCC	$\Omega(\frac{N^r}{r \log^2 N})$	$O(N^r)$		$O(N^r)$		$O(N^r/\log^2 N)$ *
Hypercube	$\Omega(2^{r-2}/r)$	N.A.		N.A.		$O(2^{r-2})$ *
K_N	$\Omega(N^{r+1}/r)$	N.A.		N.A.		$O(N^{r+1})$ *

UN stands for "unknown," N.A. stands for "not applicable." The upper bounds marked with "" are optimal if r is bounded.*

The third and fourth columns of the tables present upper bounds obtained from bisectors of the factor graphs. Again, it is easy to observe that the linear array and the complete binary tree have $O(1)$-bisectors. By applying Corollary 1 directly, the presented bounds are obtained. There are not, as far as we know, tight bisectors for the shuffle-exchange and the de Bruijn graphs. Thus, we present the corresponding bounds as unknown. We can easily show that the $n \log n$-node butterfly can be bisected by removing $O(n)$ edges, resulting in two butterflies with one less rank and several isolated nodes. Therefore, we conclude that the butterfly has an $O(x/\log x)$-bisector. Similarly, it can be shown that the cube-connected cycles has an $O(x/\log x)$-bisector. To obtain the bounds on wire length we use $x/\log x = x^{\alpha}$ for some $\alpha > 0$ and, therefore, $\alpha > 0$ in Corollary 1 for both networks. Since the hypercube can only grow by increasing the number of dimensions, it is considered as a network with unbounded number of dimensions, and the bisector approach cannot be applied to it. Similarly, this approach cannot be applied to the product of complete graphs, since K_N has not bounded vertex degree.

The fifth and sixth columns contain the bounds obtained from bifurcators of the factor networks. The linear array and the complete binary tree have zero-special bifurcators. The value of the bifurcators for the shuffle-exchange and de Bruijn networks are obtained from known layouts of area $O(N^2/\log^2 N)$ [14] that implies the existence of $O(N/\log N)$-bifurcators for these networks [2]. It is easy to see that the butterfly and the cube-connected cycles have one-special bifurcators. We then apply Corollaries 2 and 3 to obtain the bounds on layout area and maximum wire length for all these networks. Again, the hypercube and the product of complete graphs are not considered.

The last column of the tables present the upper bounds obtained from collinear layouts for the factor networks. If the nodes of the linear array are laid down in a line, we obtain a collinear layout with wiring width one and bandwidth one. The complete binary tree has a collinear layout with wiring width $O(\log N)$ and bandwidth $O(N)$, which can be obtained by just labeling the nodes in in-order. To obtain normal collinear layouts for the shuffle-exchange and de Bruijn graphs, we can apply Lemmas 6 and 7 to their optimal $O(N/\log N) \times O(N/\log N)$ area layouts [14] to obtain normal collinear layouts with wiring width $O(N/\log N)$, hence, the bounds on the layout area in Table 1. We do not know the bandwidth of these layouts to obtain a bound on the wire length. But, we use the wiring width and bandwidth of a collinear layout presented in [23] for the shuffle-exchange graph, which has wiring width $O(N/\log^{1/2} N)$ and bandwidth $O(N/\log^{1/2} N)$. Note, then, that the maximum wire length bounds presented in Table 2 for these networks may not be achievable with the optimal area layout presented in Table 1.

The normal collinear layout obtained by placing the ranks of the butterfly in order, one after the other, has wiring width $O(N/\log N)$ and bandwidth $O(N/\log N)$. A similar approach can be used for the cube-connected cycles to obtain the same bounds. The hypercube has, as factor network, the two-node linear array, which is laid out with wiring width one and bandwidth one (see Fig. 4a). Also, it is possible to obtain a normal collinear layout for K_N with wiring width $O(N^2)$.

6 Conclusions and Comparison of Results

In this paper, we have investigated bounds on the area and maximum wire length of layouts for homogeneous product networks. We have obtained lower bounds, based on the maximal congestion of the factor network, and upper bounds, based on the existence of bisectors, bifurcators, or an efficient normal collinear layout for the factor graph. A comparison of the area bounds for some product networks is given in Table 1.

The proposed method based on collinear layouts seems to generate layouts with optimum area in most of the cases. Only for products of complete binary trees, the layout area is not minimum, and it is not possible to reach an optimal area layout for this network using this method, since we would need a normal collinear layout for the complete binary tree, with $O(1)$ wiring width. In fact, the layout obtained for the product of complete binary trees is also area optimal for two dimensions, since this network has the mesh of trees as a subgraph, which requires area $\Omega(N^2 \log^2 N)$ for two dimensions [8]. The layouts obtained by using bisectors (when applicable) are also quite area efficient, since they have optimal area for more than two dimensions in the studied cases. The layouts obtained by using bifurcators are not always area optimal, but are off by only a polylogarithmic factor of N.

In Table 2, we compare the results of maximum wire length. If the number of dimensions r is constant, the collinear method obtains bounds that match the lower bounds. In most cases, it gave better bounds than the other two approaches. The only exception we have is the product of complete binary trees. When applicable, the use of bisectors seems to give the same maximum wire lengths as the use of bifurcators.

The above analyses suggest the method based on collinear layouts as a very useful and powerful approach to the layout problem for homogeneous product networks. More research may help in finding normal collinear layouts with small wiring width and small bandwidth for a variety of other factor graphs.

If the product network is heterogeneous and all the factor graphs have equal number of nodes, it is not difficult to derive bounds similar to those presented by using the main results of this paper, without referring to the detailed discussion in their proofs. One way is to just consider the worst case. For instance, the lower bounds presented in Theorems 1 and 2 are still valid if we define k as the maximum of the maximal congestions of the factor graphs. Similarly, if $f(x)$ is the largest asymptotic complexity bisector of all the factor graphs, then the product graph has an $O(x^{(r-1)/r} f(x^{1/r}))$-bisector. The results for bifurcators and collinear layouts can be generalized in a similar way.

Additional detail may arise when different factor graphs have different numbers of nodes for different dimensions, or when they have different maximal congestions. Being more careful and giving exact bounds (upper or lower) in such cases is not more difficult, but discussion gets rather tedious. In fact, we had to deal with such cases during the proofs of Theorems 3 and 5. Even though we started with a homogeneous product network, after the first bisection, the product graphs obtained are neither homogeneous nor do they contain the same number of nodes at each dimension. Focusing on homogeneous product networks allowed the tedious part of discussion to be limited within the proofs and not in the mainstream discussions of the paper. The key point to realize is that we should consider the smallest bisection that can be found at each step of bisecting the product network.

Acknowledgment

A. Fernández's contribution to this work was made during the time in which he was at the University of Southwestern Louisiana.

References

[1] M. Baumslag and F. Annexstein, "A Unified Framework for Off-Line Permutation Routing in Parallel Networks," *Math. Systems Theory*, vol. 24, no. 4, pp. 233-251, 1991.

[2] S.N. Bhatt and F.T. Leighton, "A Framework for Solving VLSI Graph Layout Problems," *J. Computer and System Sciences*, vol. 28, pp. 300-343, 1984.

[3] S.N. Bhatt and C.E. Leiserson, "Minimizing Wire Delay in VLSI Layouts," MIT VLSI memo 82-86, 1982.

[4] L. Bhuyan and D.P. Agrawal, "Generalized Hypercubes and Hyperbus Structures for a Computer Network," *IEEE Trans. Computers*, vol. 33, no. 4, pp. 323-333, Apr. 1984.

[5] F.R.K. Chung, "Labelings of Graphs," *Selected Topics in Graph Theory 3*, L.W. Beineke and R.J. Wilson, eds., pp. 151-168. Academic Press, 1988.

[6] S.K. Das and A.K. Banerjee, "Hyper Petersen Networks: Yet Another Hypercube-Like Topology," *Proc. Fourth Symp. Frontiers of Massively Parallel Computation*, pp. 270-277, McLean, Va., Oct. 1992.

[7] K. Efe and A. Fernández, "Products of Networks with Logarithmic Diameter and Fixed Degree," *IEEE Trans. Parallel and Distributed Systems*, vol. 6, no. 9, pp. 963-975, Sept. 1995.

[8] K. Efe and A. Fernández, "Mesh Connected Trees: A Bridge between Grids and Meshes of Trees," *IEEE Trans. Parallel and Distributed Systems*, vol. 7, no. 12, pp. 1,283-1,293, Dec. 1996.

[9] T. El-Ghazawi and A. Youssef, "A General Framework for Developing Adaptive Fault-Tolerant Routing Algorithms," *IEEE Trans. Reliability*, vol. 42, no. 2, pp. 250-258, June 1993.

[10] E. Ganesan and D.K. Pradhan, "The Hyper-deBruijn Networks: Scalable Versatile Architecture," *IEEE Trans. Parallel and Distributed Systems*, vol. 4, no. 9, pp. 962-978, Sept. 1993.

[11] P.C. Kainen, "A Lower Bound for Crossing Numbers of Graphs with Applications to K_n, $K_{p,q}$, and $Q(d)$," *J. Combinatorial Theory*, vol. 12, pp. 287-298, 1972.

[12] D.J. Kleitman, "The Crossing Number of $K_{5,n}$," *J. Combinatorial Theory*, vol. 9, pp. 315-323, 1971.

[13] F.T. Leighton, "New Lower Bound Techniques for VLSI," *Proc. 22nd Ann. Symp. Foundations of Computer Science*, pp. 1-12, 1981.

[14] F.T. Leighton, *Complexity Issues in VLSI*. Cambridge, Mass.: MIT Press, 1983.

[15] F.T.. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, and Hypercubes*. San Mateo, Calif.: Morgan Kaufmann, 1992.

[16] C.E. Leiserson, "Area-Efficient Graph Layout (for VLSI)," *Proc. 21st Ann. Symp. Foundations of Computer Science*, pp. 270-281, Oct. 1980.

[17] C.E. Leiserson, "Area-Efficient VLSI Computation," PhD thesis, Carnegie-Mellon Univ., 1981. MIT Press, 1983.

[18] R.B. Panwar and L.M. Patnaik, "Solution of Linear Equations on Shuffle-Exchange and Modified Shuffle Exchange Networks," *Proc. 26th Allerton Conf.*, pp. 1,116-1,125, 1988.

[19] A.L. Rosenberg, "Product-Shuffle Networks: Toward Reconciling Shuffles and Butterflies," *Discrete Applied Mathematics*, vol. 37/38, pp. 465-488, July 1992.

[20] D.D. Sherlekar and J. JáJá, "Layouts of Graphs of Arbitrary Degree," *Proc. 25th Ann. Allerton Conf.*, Sept. 1987.

[21] D.D. Sherlekar and J. JáJá, "Balanced Graph Dissections and Layouts for Hierarchical VLSI Layout Design," Technical Report CSE-TR-22-89, Dept. of Electrical Eng. and Computer Science, Univ. of Michigan, Ann Arbor, 1989.
[22] O. Sýkora and I. Vrto, "On the Crossing Number of the Hypercube and the Cube Connected Cycles," *Proc. 17th Int'l Workshop Graph-Theoretic Concepts in Computer Science, WG '91*, G. Schmidt and R. Berghammer, eds., vol. 570, *Lecture Notes in Computer Science*, pp. 214-218. Fischbachau, Germany: Springer-Verlag, June 1991.
[23] C.D. Thompson, "A Complexity Theory for VLSI," PhD thesis, Carnegie-Mellon Univ., Aug. 1980.
[24] J.D. Ullman, *Computational Aspects of VLSI*. Rockville, Md.: Computer Science Press, 1984.
[25] L.G. Valiant, "Universality Considerations in VLSI Circuits," *IEEE Trans. Computers*, vol. 30, no. 2, pp. 135-140, Feb. 1981.
[26] A.S. Youssef and B. Narahari, "The Banyan-Hypercube Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 1, pp. 160-169, 1990.

Antonio Fernández received the degree of Diplomado en Informática in March 1988 and the degree of Licenciado en Informática in July 1991 from the Universidad Politécnica de Madrid. He received the MS degree in computer science in the fall of 1992 and the PhD degree in computer science in the fall of 1994 from the University of Southwestern Louisiana, supported by a Fulbright Scholarship.

Dr. Fernández is an associate professor in the Departamento de Arquitectura y Tecnología de Computadores at the Universidad Politécnica de Madrid, where he has been a member of the faculty since 1988. He is currently on leave as a postdoctoral researcher at the MIT Laboratory for Computer Science, supported by a grant from the Spanish Ministry of Education. His research interests include parallel architectures and algorithms, interconnection networks, distributed systems, and data communication.

Kemal Efe received the BSc degree in electronic engineering from Istanbul Technical University, the MS degree in computer science from the University of California at Los Angeles, and the PhD degree in computer science from the University of Leeds.

Dr. Efe is currently an associate professor of computer science at the Center for Advanced Computer Studies, University of Southwestern Louisiana. He was previously a member of the faculty of the Computer Science Department at the University of Missouri-Columbia. His research interests are in parallel and distributed computing, in which he has made many significant contributions. His expertise includes parallel algorithms and architectures, interconnection networks, distributed operating systems, distributed algorithms, performance evaluation, and VLSI complexity models. Dr. Efe has served on the technical committees of many international conferences and has given invited talks in the U.S., Europe, and Japan.

In 1995, Dr. Efe received the "Certificate of Recognition" from NASA for his research contributions. He is a member of the ACM and the IEEE.