Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

Work-Preserving Real-Time Emulation of Meshes on Butterfly Networks

by
Alf-Christian Achilles

The emulation of a guest network G on a host network H is work-preserving and real-time if the inefficiency, that is the ratio W_G/W_H of the amounts of work done in both networks, and the slowdown of the emulation are $O(1)$.

In this thesis we show that an infinite number of meshes can be emulated on a butterfly in a work-preserving real-time manner, despite the fact that any emulation of an $s \times s$-node mesh in a butterfly with load 1 has a dilation of $\Omega(\log s)$.

The recursive embedding of a mesh in a butterfly presented by Koch et al. (STOC 1989), which forms the basis for our work, is corrected and generalized by relaxing unnecessary constraints. An algorithm determining the parameter for each stage of the recursion is described and a rigorous analysis of the resulting emulation shows that it is work-preserving and real-time for an infinite number of meshes.

Data obtained from simulated embeddings suggests possible improvements to achieve a truly work-preserving emulation of the class of meshes on the class of butterflies.

WORK-PRESERVING REAL-TIME EMULATION OF MESHES ON BUTTERFLY NETWORKS

by
Alf-Christian Achilles

A Thesis
Submitted to the Faculty of the Graduate Division of the
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science
Department of Computer Science
August 1991

APPROVAL PAGE

Work-Preserving Real-Time Emulation of Meshes on Butterfly Networks

by
Alf-Christian Achilles

Dr. Bruce Parker, Thesis Advisor,
Assistant Professor of Computer Science,
Department of Computer and Information Science,
New Jersey Institute of Technology, Newark

BIOGRAPHICAL SKETCH

Author: Alf-Christian Achilles

Degree: Master of Science in Computer Science

Place of Birth: Wolfenbüttel, Germany

Undergraduate and Graduate Education:

- Master of Computer Science, New Jersey Institute of Technology, Newark, NJ, 1991
- Vordiplom in Computer Science, Technische Universität Braunschweig, Braunschweig, Germany, 1987

Major: Computer Science

Presentations and Publications:

Alf-Christian Achilles and Helfried Broer, "A Development System for Occam 2 Programs in Smalltalk/V", presented on the Annual GI (Gesellschaft für Informatik) Workshop of the Group 2.1.4 *Alternative Konzepte für Sprachen und Rechner*, Bonn, Germany, 1989.
Appeared also in *Transputer*, Markt und Technik, 1989, pp. 297–309, and *Design und Elektronik*, Vol. 21, 1989, pp 121–130.

To my parents, Albrecht and Helga Achilles,

and

to Jackie.

ACKNOWLEDGMENTS

I would like to express my gratitude to Dr. Bruce Parker, my thesis advisor, for his support and guidance throughout the preparation of my thesis and his many helpful comments.

Furthermore I would like to thank Bruce Maggs and Eric Schwabe for their assistance in providing me with the latest extended version of their unpublished paper.

Many thanks also to the German Fulbright Commission for giving and extending the support for my stay in the U.S. and the Institute for International Education in New York City, especially Nancy Mills, for shielding me from most of the red tape.

This research was partly funded by a Fulbright grant and by funds provided by the Computer Science Department of the New Jersey Institute of Technology.

Contents

List of Tables

List of Figures

List of Theorems

Chapter 1

Introduction

The importance of emulations of networks became evident when computer scientists became aware of the topological dependency of algorithms and programs constructed for multi-processor machines. It is necessary to know how feasible it is to make computations on another network topology without having to make major changes to the algorithm.

To emulate a network (*guest*) by another one (*host*) means that the same algorithm (or, in extreme cases, the same program) that was designed for the guest network is run without changes on the host network. The cost associated with this emulation determines the feasibility. The overhead of an emulation results from the fact that whereas messages between neighbors in the guest network were delivered in one step, they now might have to travel along a path consisting of many edges. Furthermore, if the number of nodes in the host network is smaller than in

the guest network, it is evident that the computational load on the nodes in the host network is higher than in the guest network. Assuming equal computational power and communication bandwidth in both the guest and the host network, the computation will then take longer than in the guest network.

An emulation is often completely described by an *embedding* of the guest network in the host network, that is, by a mapping of nodes and edges in the guest network to nodes and paths in the host network.

1.1 Motivation

In this section we present a rationale for studying efficient emulations of a two-dimensional mesh by a butterfly.

We assume that the technology available has fixed-connections, that is, each processor has a constant set of neighbors with whom it may communicate.

This is, of course, technology dependent. If we are considering VLSI where the processors and connections are embedded in silicon, then transforming a mesh topology to a butterfly is not possible. On the other hand, a set of transputers, with say four transputers per board, may be reconnected by manually reconnecting the jumper wires, a task which becomes quite tedious for more than a few processors, and even more so if we wish to reconnect statically, that is for each job that we wish to run on the network. On the other hand, retargetable lasers in an optical

network allow not only static but even dynamic reconfiguration [19]. Thus future technologies will remove this assumption. Our concern here is with those networks which cannot be statically reconfigured or for which it is not feasible to do so.

We now consider the base topology to support general parallel computing needs. The two principle topologies that have been commercially available are the two-dimensional mesh (e.g., the Xnet communication on the MasPar) and the hypercube (e.g., NCube).

There are many applications that can best exploit nearest-neighbor properties in each topology. Vision processing is usually done on a two-dimensional array and thus in many ways is best suited to a mesh (though not all — take for example the pyramid and mesh-bus architectures considered by others). There are also many numerical algorithms that are most appropriate for the mesh. On the other hand FFT is best suited for the hypercube, or specifically, a hypercube-related network called a butterfly.

The question is: having chosen a base topology between a mesh and a butterfly, which can best support applications best implemented on the other topology? Or put another way, how well can one topology emulate the other?

Koch, Leighton, Maggs, Rao, Rosenberg, and Schwabe in their paper "Work-preserving emulations of fixed-connection networks" [13, 14] consider the emulation of T_G steps of an N_G-node guest network G on an N_H-node host network H. They define a *work-preserving emulation* to be one where the time required by the

host, T_H, is $O(T_G N_G/N_H)$, because the time-processor products of both networks, $\Theta(T_H N_H)$, are to within a constant factor of each other. The slowdown $S(N_H)$ is T_H/T_G. As a special case, a work-preserving emulation is called *real-time* if $T_H = O(T_G)$, that is, the work-preserving emulation has constant slowdown.

Due to a congestion-based lower bound [14], any work-preserving emulation of a butterfly by a k-dimensional n-node mesh has a slowdown of at least $2^{\Omega(n^{1/k})}$. In fact, the ratio of the processor-time product of the mesh to the butterfly is at least $\Omega((N_H/\log_k N_G)^{1/(k+1)})$, where N_H is the number of mesh nodes and N_G is the number of butterfly nodes.

In contrast, Koch et al also exhibit a real-time emulation of a butterfly by a mesh.

We conclude, then, that a butterfly is a better base topology than a mesh.

N.B. This conclusion is quite technology dependent. This analysis assumes that the physical length of an interconnection does not matter, that communication between every pair of processors is equal in delay. This is despite the facts that

1. Butterflies have long wires [21].

2. Long wires in VLSI cause several problems, among them: increased delay, signal attenuation, and capacitive coupling [17]. The latter two problems can be dealt with by such devices as repeaters, but the first remains.

There are two ways to counter this argument. First, optical interconnections

have none of these problems [10]. Thus a fixed-connection butterfly is even more interesting as a basis for future networks. Second, there are additional tradeoffs which show that for a reasonable range of processors per unit area, word size, and message length, the butterfly performs at least as well as the mesh [20].

1.2 The Butterfly Network

A $n\,2^n$-node butterfly without wrap-around is a graph $G_B = (V_B, E_B)$ where

$$V_B = \{\langle i,j\rangle \,\big|\, 0 \le i < n, 0 \le j < 2^{n-1}\}$$

is the set of vertices in the butterfly graph. The value i is considered the *level* (or *row*) of the node in the butterfly and j denotes the *column*, which is usually given in binary digits. These interpretations become clear when looking at Figure 1.2.

E_B is the set of edges connecting the vertices. Two vertices are connected by a *forward* edge if they are in the same column on adjacent levels. A *cross* edge exists between nodes that are on adjacent levels i and $i+1$ and the binary represention of their column numbers differ in exactly the i-th bit.

$$E_B = \{(\langle i,j\rangle, \langle i',j'\rangle) \,\big|\, i' = i+1, j \text{ and } j' \text{ differ in the } i^{th} \text{ bit}\}$$

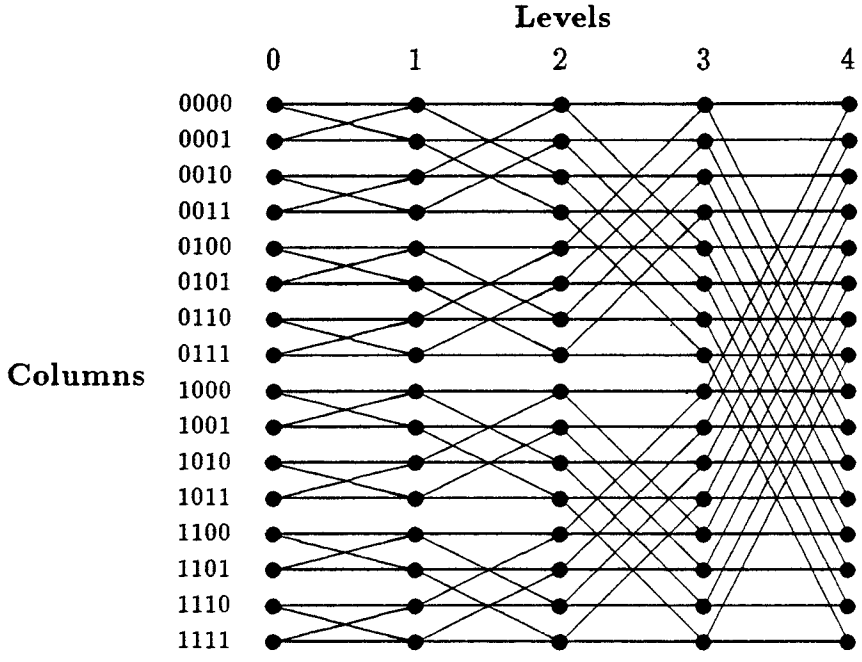

Figure 1.1: A butterfly network without wrap-around containing 5 levels (rows) and 2^4 columns.

1.3 Previous Work

Work-preserving emulations of networks on smaller networks of the same class have been studied by Fishburn and Finkel in their paper on quotient networks [11]. Meyer auf der Heide [18] presented efficient emulations among different models of parallel computers. Since the hypercube has been and still is a popular interconnection topology, most work done in the area of emulations (embeddings) of networks has been focussed mainly on emulations of various other topologies on

hypercubes. Bhatt and Ibsen [3] and Desphande and Jenevein [8] have worked on embeddings of trees in hypercubes. Efe [9] demonstrated an embedding of a mesh of trees in a hypercube. Embeddings of meshes on hypercubes have been shown by Chan [6, 5, 7] and Ho and Johnsson [12].

Bhatt et al. [2] describe optimal emulations by butterfly networks and Leighton et al. [15] and Koch et al. [13, 14] demonstrated the embedding of a mesh on a butterfly that forms the basis for this thesis.

The guest network does not necessarily have to be a physical network of processing elements, but most often is the internal structure of a computational problem. Many algorithms exhibit a structure that can be described as a graph such as trees (divide-and-conquer, depth-first-search) or meshes (numerical solutions of differential equations, some imaging problems). The problem of mapping these algorithms to a physical topology is called the *mapping problem* (Bokhari [4], Berman and Snyder [1]).

1.4 Emulations of Networks

Networks are described by graphs consisting of a number of vertices and edges connecting those vertices. Assume that we have two networks described by the

two graphs

$$H = (V_H, E_H) \quad \text{and} \quad G = (V_G, E_G)$$

where H describes the host network and G describes the guest network, that is, the network to be emulated on H. $N_H = |V_H|$ and $N_H = |V_H|$ are the number of nodes in the respective networks.

Emulations of networks can be characterized by certain measures which quantify the performance.

Definition 1.1 *Let T_G be the time (number of steps) on G to be emulated and T_H the time needed to emulate T_G steps on H.*

The amount of *work* being done therefore is $W_G = T_G N_G$ and $W_H = T_H N_H$.

Definition 1.2

1. *The* inefficiency *I of the emulation is defined as* $\frac{W_H}{W_G}$.
2. *The* slowdown *S of the emulation is* $\frac{T_H}{T_G}$.
3. *The* contraction *C of the emulation is* $\frac{N_G}{N_H}$.
   The blowup *B is the reciprocal of the contraction* $\frac{1}{C} = \frac{N_H}{N_G}$.

As Koch et al. [14] pointed out, inefficiency is the most important measure for emulations since its decribes the waste of resources. Therefore the aim of any emulation is to minimize inefficiency.

The embedding of a network consists of a mapping of nodes in G to nodes in H and a mapping of edges in G to paths in H. The following terms are characteristics of embeddings.

Definition 1.3

1. *The* load *of an embedding is the maximum number of nodes in G mapped to a node in H.*

2. *The* dilation *is the number of edges in the longest path in H onto which an edge in G has been mapped.*

3. *The* congestion *is the maximum number of paths in H, onto which edges in G have been mapped, that traverse the same edge.*

1.4.1 The Computational Model

In order to analyze emulations of networks, a model is needed to specify the capabilities of the guest and host networks and just what a single step in the emulation comprises. Koch et al. [14] presented the general concept of pebbling. Here we will only use a simplified version of this model, which is described in [13, 15] and which suffices for our purposes. The pebbling process is a way to keep track of the states of the nodes in the emulated network. For each such node v and for every time step t, where $0 \leq t \leq T_G$, a pebble is a pair (v,t) representing the

state of the node v at time t. At the beginning of the emulation all the pebbles $(v_0, 0), (v_1, 0), \ldots, (v_{N_G}, 0)$ exist.

An emulation is modeled as a pebbling process on a directed acyclic graph (DAG), Γ, where at each step of the emulation each node in the host network H can

1. Copy a pebble it contains.

2. Send a pebble to one of its neighbors in H.

3. Create a pebble (v, t) if it contains pebbles with labels $(v, t-1)$, $(v_1, t-1)$, $(v_2, t-1)$,..., $(v_k, t-1)$, where $v_1, v_2, \ldots, v_k$ are the neighbors of v in G.

The emulation stops if the host network has computed all pebbles of the form (v, T_G).

In practice, of course, the pebbles will not represent the complete state of a node in the guest network but only contain the information that is exchanged between nodes in G at time t.

An important aspect of pebbling is that a pebble (v, t) may have more than one instance which can result in redundant computation where more than one node in H compute identical pebbles. The idea of redundant computation is the basis for the embedding of meshes in butterflies presented in [13, 14] and further developed in this thesis.

1.4.2 Work-Preserving Emulations

In [13, 14, 15] the notion of work-preserving emulations of networks is introduced. The main argument is that slowdown and blowup are relevant characteristics for emulations but that their quality is best described by their efficiency. That is, it is most important to minimize inefficiency.

A work-preserving emulation is an emulation with inefficiency $I = O(1)$. A class of networks is said to have a work-preserving emulation on another class of networks with slowdown S if each network in the class has a work-preserving emulation on a network of the other class with slowdown S. When S is $O(1)$ we speak of a real-time emulation.

1.5 The Embedding of a Mesh in a Butterfly

Richard Koch, Tom Leighton, Bruce Maggs, Satish Rao, Arnold Rosenberg and Bruce Maggs [13, 14] describe a recursive embedding of a mesh in a butterfly which will serve as a basis for the work presented in this thesis.

Their work may be divided in two parts: First they describe an embedding of a mesh in a butterfly with constant congestion. The embedding leaves a number of parameters open that control the characteristics of the emulation. Then they choose the values for these parameters and claim that the resulting emulation constitutes a work-preserving, real-time emulation of the class of meshes on the

class of butterflies. Here we will concentrate mainly on the description of the embedding, since the choice of the emulation parameters in [13, 14] does not allow a work-preserving emulation (see Section 1.5.4) and in the later chapters we will chose a new set of parameters for the embedding.

The embedding relies on a recursive scheme to emulate submeshes by subbutterflies and to connect the subbutterflies (and therewith the submeshes) to emulate the mesh. The following paragraphs will describe their work as briefly as possible.

Let $N = s^2$ be the number of nodes in the $s \times s$ mesh to be emulated. Since the emulation is recursive, let us denote the current stage[1] of the recursion with k ($0 \leq k \leq \omega$), where ω is the number of recursive stages needed to completely embed the $s \times s$ mesh.

Assume that we have an emulation of a s_{k-1}^2-node mesh on a $N_{k-1} = n_{k-1}2^{n_{k-1}}$-node butterfly. The objective is to emulate a s_k^2-node mesh on a $N_k = n_k 2^{n_k}$-node butterfly. Koch et al. require the parameter n_k to be a power of two.

1.5.1 Partitioning a Mesh into Submeshes

The s_k^2-node mesh will be composed of overlapping meshes of size s_{k-1}^2, where the width of the overlapping region is f_k. Figure 1.5.1 shows how the submeshes overlap. The set of nodes on the border of a mesh and of the nodes at distance

[1]Note that we have reversed the numbering of the stages with s_0 being the base case stage. Because of the constructive nature of the later chapters as opposed to the existentialistic approach in [13, 14] we consider this way of numbering the stages to be better suited.

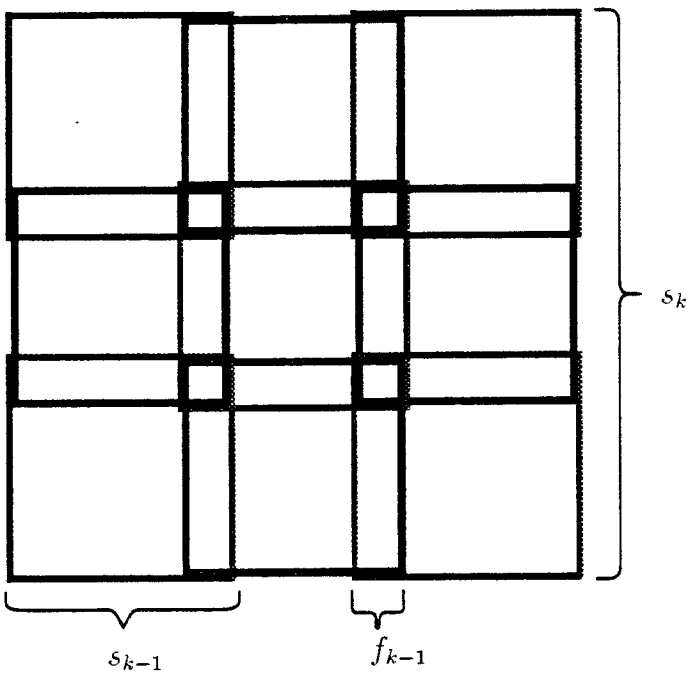

Figure 1.2: (adapted from [13]) A mesh is divided into overlapping submeshes.

f_{k-1} from the border will be called F_{k-1}. The nodes at distance f_{k-1} from the border of the submesh can be emulated for f_{k-1} steps until pebbles produced by a neigboring submesh are needed to proceed. The emulation of the whole mesh is therefore divided into periods of emulating f_{k-1} steps on the submeshes and subsequently sending the f_{k-1} pebbles produced by the nodes at distance f_{k-1} from the border to the neighboring submeshes. Once these pebbles arrive at the border nodes of the neighboring submeshes the emulation in these submeshes can resume.

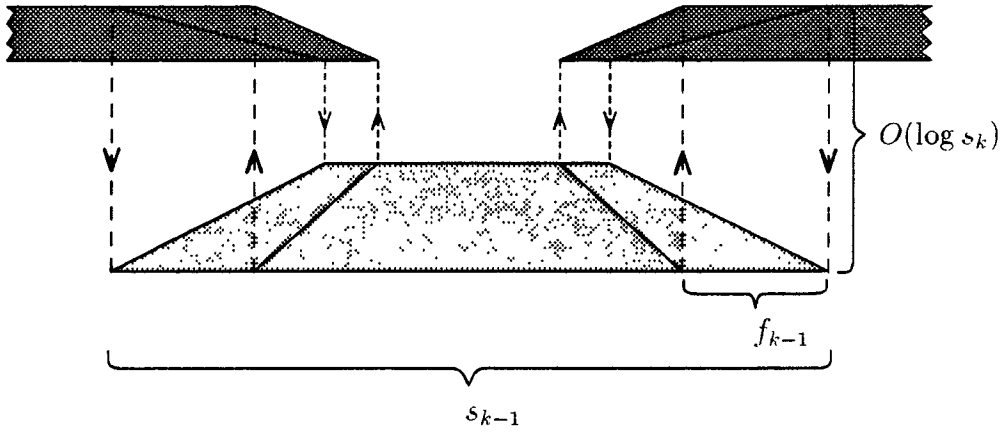

Figure 1.3: The overlap of meshes in one dimension. The border nodes of the overlapping region in each mesh either receive or send pebbles as indicated by the arrows.

The key idea here is to exploit the pipeline effect induced by the redundant emulation of the nodes in the border region (the nodes of the mesh from the border up to the distance f_{k-1} from the border) while not letting the additional resources (butterfly nodes) needed for this redundancy inhibit a constant blowup.

Each submesh will be emulated by a subbutterfly and the subbutterflies will be connected to allow pebbles to be sent between corresponding nodes in F_{k-1}.

1.5.2 Partitioning a Butterfly into Subbutterflies

Following is a description of how a $N_k = n_k 2^{n_k}$-node[2] butterfly is divided into $N_{k-1} = n_{k-1} 2^{n_{k-1}}$-node subbutterflies and how these subbutterflies are connected.

A butterfly is divided into subbutterflies by removing all edges between each level which is a multiple of n_{k-1} and the next higher level and considering the remaining connected components. A more formal description of the partition follows.

Let the bit string $c_{n_k-1} c_{n_k-2} \cdots c_0$ denote the column of a node in the $n_k 2^{n_k}$-node butterfly. A subbutterfly in a $n_k 2^{n_k}$-node butterfly consists of the set of nodes with the following property: Let α be a multiple of n_{k-1} (possibly zero); then all nodes of a subbutterfly share common values of $c_{n_k-1} \cdots c_{\alpha+n_{k-1}}$ and $c_{\alpha-1} \cdots c_0$. α will be used to characterize the level of a subbutterfly within the superbutterfly.

Since the subbutterflies emulating submeshes must be connected within the butterfly to send pebbles from one subbutterfly to another, some subbutterflies are not used to emulate a submesh but to create connections between the emulating subbutterflies. A subbutterfly will be used exclusively for creating such connections if there exists γ, a multiple of n_{k-1}, $\gamma > \alpha$ (where α is the value used to define the subbutterfly) and $c_{\gamma+\epsilon_{k-1}-1} \cdots c_\gamma = 10 \cdots 0$ for all nodes in the subbutterfly, or if $\alpha > 0$ and $c_{\epsilon_{k-1}-1} \cdots c_\gamma = 00 \cdots 0$. The parameter ϵ_{k-1} has to be chosen for each

[2]Note, that the number of nodes assumes a butterfly with wrap-around, an assumption which we will discuss in Section 1.5.4.

stage; its meaning and the constraints for its value will become obvious in the next section.

1.5.3 Choosing Paths to Connect Subbutterflies

Each subbutterfly that emulates a submesh has a node v in level zero for every node u in F_{k-1} of the submesh such that there exists a path from v to the butterfly node that emulates u along which pebbles can be send without slowing down the emulation of the submesh. These nodes in level zero, which we will call I/O ports, have the ϵ_{k-1}-bit string $10\cdots0$ as the least significant bits of their column and the next ϵ_{k-1} bits are the same for all I/O ports, but can be chosen arbitrarily.

To connect the corresponding nodes v in F_{k-1} of adjacent submeshes, paths have to chosen that connect the I/O port u_1 which, according to the recursive assumption, is connected to a subbutterfly node u emulating v, to the I/O port u_1' in another subbutterfly. The subbutterfly containing u_1 be characterized by α. As specified above, the bits $c_{\alpha+2\epsilon_{k-1}-1}\cdots c_{\alpha+\epsilon_{k-1}}$ for the I/O ports in the subbutterfly can be chosen arbitrarily and they will be set to the bit string $c_{\epsilon_{k-1}-1}\cdots c_0$. Let u_2 be the node in level zero of the butterfly whose column is obtained by changing the ϵ_{k-1} least significant bits to $10\cdots0$. There exists a path for every u_1 to its corresponding node u_2 in level zero of the butterfly. To connect u_1 and u_1' it is sufficient to connect the corresponding nodes u_2 and u_2' by a permutation on the columns of the subbutterfly consisting only of those nodes whose ϵ_{k-1} least

significant bits are $00\cdots0$. As Koch et al. have shown, none of these paths conflict with each other or with paths at other levels of the recursion, since the ϵ_{k-1}'s are distinct.

To satisfy the requirements for the next stage of the emulation, I/O ports in level zero of the butterfly must be provided for the nodes in F_k. Since $F_k \subseteq F_{k-1}$ we know that there exists already a path from each node u emulating a node v in F_k to a node u_2 in level zero of the butterfly. After choosing one of the u's for each v we can connect the corresponding u_2's with the I/O port by routing a permutation on the columns.

The butterfly must also have enough I/O ports in level zero to connnect the internal butterfly nodes emulating a mesh node in F_k to other butterflies. There are $8s_k$ nodes in F_k and the number of I/O ports is $2^{n_k-2\epsilon_k}$. Therefore the following condition must be satisfied:

$$\lg 8s_k < n_k - 2\epsilon_k \tag{1.1}$$

1.5.4 Failure of the Emulation as Proposed by Koch et al.

The choice of emulation parameters in [13, 14] — or the analysis thereof, respectively — has flaws which inhibit the emulation of the mesh in a work-preserving, real-time fashion. In this section we will discuss these errors and indicate how we corrected them.

Butterflies with Wrap-Around

The choice of parameters in [14] assumed that a butterfly with wrap-around would be used to emulate the mesh. In the analysis a butterfly is partitioned into butterflies with wrap-around with the argument that even though these butterflies would not really be butterflies with wrap-around (a butterfly with wrap-around is not a recurrent graph), a butterfly without wrap-around can emulate a butterfly with wrap-around in constant time in a work-preserving, real-time fashion and that one can therefore, for the purpose of analyzing the emulation, assume a partition into butterflies with wrap-around. However, this emulation of wrap-around butterfly on butterflies without wrap-around would be performed at each stage of the recursion, such that the constants governing the slowdown of these butterfly-butterfly emulations at each stage would have to be multiplied in order to get the overall effect on the mesh-butterfly emulation. Since the number of recursive stages grows with N towards infinity, the accumulated effect of the slowdown in all stages is not constant and therefore the mesh-butterfly emulation can not be be real-time.

In our analysis we will correct this and partition a butterfly into butterflies without wrap-around.

Inadequate Choice of ϵ_k

In this section we will show that the choice of emulation parameters as proposed in [13], specifically the choice of ϵ_k, inhibits an emulation with constant blowup.

The choice of emulation parameters in [13] sets

$$\epsilon_k = \frac{\log n_k}{30} \qquad \text{and} \qquad \omega = \lceil \log_{20} \lg N \rceil$$

In order to layout the communication paths in the proposed manner without letting the layout on one stage interfere with the layout on another stage, the size of the ϵ_k's must be strictly monotonically increasing from the lowest to the highest stage. Since ϵ_k is an integral parameter, it has to increase at least by one at each stage ($\epsilon_k \geq \epsilon_{k-1} + 1$). Thus the ϵ_ω of the final stage must be at least as great as the number of recursive stages $\omega : \epsilon_\omega \geq \omega$. When we apply the necessary condition in Equation 1.1 get obtain:

$$\begin{aligned} \frac{1}{30} \lg n_\omega &\geq \log_{20} \lg N \\ \lg n_\omega &\geq \Omega(\log \log N) \\ n_\omega &\geq \Omega(\log N) \qquad (1.2) \\ 2^{n_\omega} &\geq \Omega(N) \qquad (1.3) \end{aligned}$$

By multiplying the relations (1.2) and (1.3) we can infer the following relation between N and N_ω :

$$N_\omega = n_\omega 2^{n_\omega} \geq \Omega(\log N)\,\Omega(N)$$

$$N_\omega \geq \Omega(N \log N)$$

This leads to a blowup of $B = \frac{N_H}{N_G} = \frac{N_\omega}{N} \geq \log N$ which shows that the proposed choice of ϵ_k inhibits an emulation with constant blowup. Since the slowdown S of any emulation is at least 1 it follows that the inefficiency $I = S \cdot B$ is also not constant and therefore the emulation is not work-preserving.

The rationale for the choice of the value for ϵ_k in [14] is not clear. In our embedding we simply set ϵ_k to the largest possible value that still leaves enough I/O ports in level zero of the butterfly (see page 39 item 4). The larger the value of ϵ_k, the less subbutterflies of size $n_k 2^{n_k - 1}$ are only used to provide paths for the connection of emulating subbutterflies (see Lemma 2.5.1).

The Emulation is Work-Preserving Only for Some Meshes

Koch et al. require the parameter n_k to be a power of two. As we increase the size of the mesh to be emulated, the size of the emulating butterfly increases and therewith n_k. Let us take a look at all the meshes of size $N = 2^{n+1}2^{2^n}$, where n is odd (to ensure that the size is a square number). The smallest butterfly that could possibly emulate such a mesh with a load of 1 is the butterfly of size

$N_\omega = 2^{n+1}2^{2^{n+1}}$. The minimum blowup for meshes of these sizes therefore is

$$B \leq \frac{N_\omega}{N} = \frac{2^{n+1}2^{2^{n+1}}}{2^{n+1}2^{2^n}} = \frac{2^{2^{n+1}}}{2^{2^n}} = 2^{2^{n+1}-2^n} = 2^{2^n}$$

Therefore the blowup increases with n towards infinity for meshes of the chosen sizes. This means that there is an infinite number of meshes which can not be emulated with constant blowup and, strictly speaking, this emulation of the class of meshes on the class of butterflies is not work-preserving.

In our choice of the n_ks we remove this unecessary constraint. The embedding does not a priori require that the n_ks be powers of two, it is only necessary that that n_k is a multiple of n_{k-1} so that the butterfly can be completely partitioned into subbutterflies. However, as will be discussed in Section 3.3.4, 4.1 and 5.1.1, the question whether there is a work-preserving emulation of the class of meshes on the class of butterflies or not, could not be answered in the work presented and needs further investigation.

Chapter 2

The Emulation

In this thesis we will show that the recursive embedding scheme presented in [13, 14] allows for a work-preserving real-time emulation of a mesh on a butterfly given the right choice of the parameters n_k, s_k, f_k and ϵ_k for each stage of the recursion. Koch et al. attempted to show this in [14] but their analysis contains flaws inhibiting a work-preserving emulation. Furthermore they neglect to account for the integral nature of the parameters to be chosen and do not provide a base case for their induction. Their analysis also did not mention a serious restriction on the work-preserving quality of the emulation.

We will correct the mistakes made in [14] and, in an attempt to make the emulation more practical, we will generalize the choice of parameters in order to allow for more design freedom, ridding the emulation of unnecessary irregularities and constraints. In Chapter 4 we will analyze the emulation in a more quantitative

way and investigate the results of simulations.

2.1 Recursive Assumption

At this point we think it is helpful to summarize the assumptions that allow the recursive embedding of the mesh into the butterfly. At each stage k the following must hold:

For the mesh: The f_k's and s_k's must be chosen such that $F_k \subseteq F_{k-1}$ for all k.

For the butterfly: The butterfly emulating the s_{k-1}^2-node mesh must adhere to the following contraints:

- For every butterfly node v emulating a mesh node in F_{k-1} there must be a butterfly node in level zero that can send pebbles to v with constant slowdown (constant congestion).
- The nodes in any column in the butterfly whose ϵ_{k-1} least significant bits are $00\ldots0$ are not participating in the emulation, where ϵ_{k-1}.

2.2 The Basic Idea Behind the Choice of Parameters

In the following sections of this chapter, we will present and analyze the emulation of a mesh on a butterfly. At this point we will give an overview of the features and an outline of the algorithm determining the emulation parameters.

The basic idea of the recursive embedding of meshes in butterflies is taken from [13, 14] which we will modify for more flexibility.

- We will take the base case into account and describe the constraints and degrees of freedom for the design of a base case emulation.

- The partitioning of butterflies into subbutterflies does not require a priori that the parameter n_k be a power of two, so that the only contraint we set up is that n_k be a multiple of n_{k-1}.

  Furthermore we will partition a butterfly into subbutterflies without wrap-around as the wrap-around butterfly is not a recurrent graph.

- The growth of the meshes from stage to stage can be controlled by a parameter g which we will call the *growth rate.*

- We will allow a constant congestion greater than 1 in order to allow for more flexibility in the design of the base case.

The general algorithm for finding the emulation parameters is the following. At each stage k we assume that we have an emulation of an $s_{k-1} \times s_{k-1}$ mesh on a $n_{k-1}2^{n_{k-1}-1}$-node butterfly.

1. Pick a minimum mesh size s'_k for the mesh to be emulated at stage k. If $s'_k \leq s_{k-1}$ then set all the parameters of this stage equal to the parameters of the last stage $k-1$ and continue with the next stage $k+1$. Else determine the number of meshes from stage $k-1$ that are needed to compose a mesh with at least s'_k nodes. Round s'_k to the next submesh size if necessary.

2. Determine the smallest butterfly of size $N_k = n_k 2^{n_k-1}$ with at least as many emulating subbutterflies as submeshes in the $s'_k \times s'_k$ mesh, where n_k must be an integral multiple of n_{k-1}.

3. Extend the $s'_k \times s'_k$ mesh by rows and columns of submeshes until there are not enough emulating subbutterflies left in the butterfly to extend it further. Let s_k be the number of nodes in one row of the extended mesh.

4. Set ϵ_k to the maximum integer such that there are still enough I/O ports for the $s_k \times s_k$ mesh in the N_k-node butterfly.

5. Let the width of the overlapping region f_k for this stage be the number of nodes along one dimension in the $s_{k-1} \times s_{k-1}$ mesh[1], so that when two $s_k \times s_k$

[1]In fact, one can set f_k to be any constant multiple of s_{k-1}.

meshes are overlapped the border lines will coincide with the border lines of the submeshes of which the $s_k \times s_k$ mesh is composed, and the condition $F_k \subseteq F_{k-1}$ is satisfied.

2.3 The Base Case

The base case of the recursive emulation is an emulation of a $s_0 \times s_0$ mesh on a $n_0 2^{n_0-1}$-node butterfly without wrap-around. Further parameters to be chosen for the base case are the width of the border region f_0 and the parameter ϵ_0. How this emulation is performed is of no importance. Only a few constraints, which form the recursive assumption for the embedding, have to be enforced:

- Koch et al. required that for *every* node v in F_0 there must be an I/O port in level zero of the butterfly. We will relax this constraint in order to allow for smaller-sized base case butterflies.

  Since the pebbles received and sent by the nodes in F_0 at the same side of the mesh all go along the same path to or from another subbutterfly, their flow can be multiplexed along the path, thus increasing the congestion c. As stated above this decreases the size of the smallest possible base case butterfly since the number of butterfly nodes in level zero, that function as I/O ports for any pebbles to or from the mesh emulated by the butterfly, decreases. For each side of the 2-dimensional mesh we now need $2s_0/c$ I/O

ports and therefore for the whole mesh $8s_0/c$ I/O ports are necessary. In the extreme case that $c = 2s_0$ there will be only one I/O port for each side of the mesh in the base case emulation through which all the pebble streams to and from the nodes in F_0 will be multiplexed.

As already described in Section 1.5.3, I/O ports are the nodes in level zero of the butterfly whose ϵ_0 least significant bits of the column are $00\ldots01$ and the bits in the positions ϵ_0 to $2\epsilon_0 - 1$ are the same for all I/O ports, but can be chosen arbitrarily. Therefore there are $2^{n_0-2\epsilon_k-1}$ I/O ports in the base case butterfly.

Since there must be at least $4 \cdot 2s_0/c$ I/O ports, the following condition must be satisfied for the base case emulation:

$$\log \frac{8s_0}{c} \leq n_0 - 2\epsilon_0 - 1 \tag{2.1}$$

- The butterfly nodes in the columns whose ϵ_0 least significant bits are all zeros must not participate in the emulation of the base case; they will be used for the connection of the base case butterflies in their superbutterfly. Note, that for ϵ_0 values of $1, 2, 3, \ldots$ only $\frac{1}{2}, \frac{3}{4}, \frac{7}{8}, \ldots$ of the nodes in the basecase can be used for the emulation of the base case mesh.

- $s_0 \geq 2$

  The size of the mesh emulated in the base case must be at least 2×2 because otherwise there would be no overlapping regions.

- $f_0 \leq \frac{1}{2} s_0$

  If the size of the overlapping region were any larger than half of the mesh width, the overlapping regions would overlap each other.

- $\epsilon_0 \geq 1$

  Obviously, 1 is the minimum for ϵ_0.

The base case butterfly has a minimum size which can be determined as follows: Since ϵ_0 must be at least 1 and at least 4 I/O ports are needed (this would mean a base case congestion of $c = 2s_0$) we can fill these values into Equation 2.1:

$$\begin{aligned} \lg \frac{8s_0}{c} &\leq n_0 - 2\epsilon_0 - 1 \\ \lg 4 &\leq n_0 - 2 - 1 \\ 5 &\leq n_0 \end{aligned}$$

A small base case butterfly has few I/O ports on level zero available compared to the number of nodes that can be used for emulation of mesh nodes, so that the congestion c will increase as more internal butterfly nodes are used for emulation of mesh nodes. Table 2.3 shows various base case configurations to illustrate this

Table 2.1: Various parameter configurations for the base case emulation.

Description	n_0	s_0	f_0	ϵ_k	Congestion
minimal base case	5	2	1	1	4
minimal base case with one-to-one mapping	5	6	≤ 3	1	12
minimal base case with one-to-one mapping and congestion of one	14	239	≤ 119	1	1
small base case	7	2	1	2	4
small base case with one-to-one mapping	7	18	≤ 9	2	36

behavior. A base case embedding that uses as many butterfly processors as possible and has a congestion of $c = 1$ is possible for values of $n_0 \geq 14$, because that is the minimal value that provides enough I/O ports for all the $8s_0$ nodes in F_0. That means that the base case butterfly would have at least $N_0 = 114,688$ nodes which would prove much too large as a starting point for the emulation. A congestion of $c = 1$ is therefore not feasible. Therefore one has to choose whether to accept a congestion greater than 1 or to waste resources by not assigning a mesh node to every available butterfly node in the base case.

Note also, that $\epsilon_0 = 1$ implies that at most a fourth of the base case butterflies will be used to emulate submeshes in stage 1, the rest will be used to route paths connecting these subbutterflies.

2.4 The Number of Submeshes and Subbutterflies

At this point we will show how to compute the number of submeshes in a mesh and the number of subbuterflies in a butterfly since the expressions for these two quantities are frequently used throughout this thesis.

The number of $s_{k-1} \times s_{k-1}$ submeshes at stage k is

$$\left(\frac{s_k - f_{k-1}}{s_{k-1} - f_{k-1}}\right)^2 .$$

This expression can easily be derived when looking at Figure 1.5.1 on page 13.

The number of N_k-node subbutterflies at stage k simply is the ratio of the number of nodes in the superbutterfly and the number of nodes in a subbutterfly:

$$\frac{N_k}{N_{k-1}} = \frac{n_k 2^{n_k - 1}}{n_{k-1} 2^{n_{k-1} - 1}} = \frac{n_k}{n_{k-1}} 2^{n_k - n_{k-1}}$$

2.5 The Number of Emulating Subbutterflies at each Stage

A subbutterfly will either be used to emulate a submesh or to provide communications paths to connect other emulating subbutterflies (and thus submeshes). In

this section we will investigate what the ratio between the number of emulating and connecting subbutterflies at each stage is and we will demonstrate a convenient lower bound for this ratio.

2.5.1 Combinatorial Analysis of the Division into Emulating and Connecting Subbutterflies

The following analysis is based on the definitions in Section 1.5.2.

The probability $p_E(\alpha, k)$ that a subbutterfly of size $n_{k-1}2^{n_{k-1}-1}$ characterized by α $(0 \leq \alpha < n_k/n_{k-1} - 1)$ at stage k is going to be used to emulate submeshes within the butterfly of size $n_k 2^{n_k-1}$ is

For $\alpha > 0$: $p_E(\alpha, k)$ is the probability that the bit string $0 \ldots 01$ of length ϵ_{k-1} does not appear in the column of the subbutterfly at any of the positions $(\alpha+1)n_{k-1}, (\alpha+2)n_{k-1} \ldots, n_k - n_{k-1}$ and that neither the bit string $0 \ldots 00$ nor the bit string $0 \ldots 01$ appear at position 0. Since these probabilities are independent, we can multiply them to obtain $p_E(\alpha, k)$:

$$p_E(\alpha, k) = \left(1 - 2^{-\epsilon_{k-1}}\right)^{n_k/n_{k-1}-\alpha-1} \left(1 - 2 \cdot 2^{-\epsilon_{k-1}}\right)$$

For $\alpha = 0$: $p_E(0, k)$ is the probability that the bit string $0 \ldots 01$ of length ϵ_{k-1} does not appear in the column of the subbutterfly at any of the positions

$n_{k-1}, 2n_{k-1} \ldots, n_k - n_{k-1}$

$$p_E(0,k) = \left(1 - 2^{-\epsilon_{k-1}}\right)^{n_k/n_{k-1}-1}$$

The general probability p_E for an arbitrary subbutterfly therefore is the sum of the probabilities for each row of subbutterflies divided by the number of rows:

$$\begin{aligned}
p_E(k) &= \frac{\sum_{\alpha=0}^{n_k/n_{k-1}-1} p_E(\alpha,k)}{\frac{n_k}{n_{k-1}}} \\
&= \frac{n_{k-1}}{n_k}\left(p_E(0,k) + \sum_{\alpha=1}^{n_k/n_{k-1}-1} p_E(\alpha,k)\right) \\
&= \frac{n_{k-1}}{n_k}\left(\left(1 - 2^{-\epsilon_{k-1}}\right)^{n_k/n_{k-1}-1} + \left(1 - 2\cdot 2^{-\epsilon_{k-1}}\right)\sum_{\alpha=0}^{n_k/n_{k-1}-2}\left(1 - 2^{-\epsilon_{k-1}}\right)^{\alpha}\right) \\
&= \frac{n_{k-1}}{n_k}\left(\left(1 - 2^{-\epsilon_{k-1}}\right)^{n_k/n_{k-1}-1} + \left(1 - 2\cdot 2^{-\epsilon_{k-1}}\right)\sum_{\alpha=0}^{n_k/n_{k-1}-2}\left(1 - 2^{-\epsilon_{k-1}}\right)^{\alpha}\right)
\end{aligned}$$

Since the sum of the geometric series $\sum_{i=0}^{n} a^i$ is $\frac{a^{n+1}-1}{a-1}$ the expression can be simplified to

$$\begin{aligned}
p_E(k) &= \frac{n_{k-1}}{n_k}\left(\left(1 - 2^{-\epsilon_{k-1}}\right)^{n_k/n_{k-1}-1} + \left(1 - 2\cdot 2^{-\epsilon_{k-1}}\right)\frac{(1 - 2^{-\epsilon_{k-1}})^{n_k/n_{k-1}-1} - 1}{(1 - 2^{-\epsilon_{k-1}}) - 1}\right) \\
&= \frac{n_{k-1}}{n_k}\left(\left(1 - 2^{-\epsilon_{k-1}}\right)^{n_k/n_{k-1}-1}(3 - 2^{\epsilon_{k-1}}) + 2^{\epsilon_{k-1}} - 2\right) \qquad (2.2)
\end{aligned}$$

Lemma 2.5.1 (Probability That A Subbutterfly Is Used For Emulation) *The probability $p_E(k)$ that a subbutterfly in a butterfly at stage k is used to emulate a submesh and not to connect emulating subbutterflies is*

$$p_E(k) = \frac{n_{k-1}}{n_k}\left(\left(1-2^{-\epsilon_{k-1}}\right)^{n_k/n_{k-1}-1}(3-2^{\epsilon_{k-1}})+2^{\epsilon_{k-1}}-2\right)$$

Proof: See previous paragraphs for the derivation of Equation 2.2. ∎

Definition 2.1 *Let $\pi_E(k)$ be the right hand term in the product defining $p_E(k)$:*

$$\pi_E(k) = \left(1-2^{-\epsilon_{k-1}}\right)^{n_k/n_{k-1}-1}(3-2^{\epsilon_{k-1}})+2^{\epsilon_{k-1}}-2$$

Therefore $p_E(k)$ can be rewritten as $\frac{n_{k-1}}{n_k}\pi_E(k)$.

In the following corollary we will show that $\pi_E(k)$ has a simple expression as a lower bound.

Corollary 2.1 (Lower Bound For $\pi_E(k)$)

$$\pi_E(k) > 2 - 3\cdot 2^{-\epsilon_{k-1}}$$

Proof: Using Lemma 2.5.1

$$\pi_E(k) = \left(1-2^{-\epsilon_{k-1}}\right)^{\frac{n_k}{n_{k-1}}-1}(3-2^{\epsilon_{k-1}})+2^{\epsilon_{k-1}}-2$$

and the fact that $\frac{n_k}{n_{k-1}} \geq 2$ we obtain the following relation:

$$\begin{aligned} \pi_E(k) &> \left(1 - 2^{-\epsilon_{k-1}}\right)\left(3 - 2^{\epsilon_{k-1}}\right) + 2^{\epsilon_{k-1}} - 2 \\ &= 2 - 3 \cdot 2^{-\epsilon_{k-1}} \end{aligned}$$

■

This lower bound leads to a lower bound of $\frac{n_{k-1}}{n_k}\left(2 - 3 \cdot 2^{-\epsilon_{k-1}}\right)$ for $p_E(k)$. Koch et al. [13, 14] have shown another lower bound for $p_E(k)$ which we will also use in our analysis of the emulation and which is presented in the following lemma.

Lemma 2.5.2 *The probability $p_E(k)$ that a subbutterfly in a butterfly at stage k is used to emulate a submesh and not to connect emulating subbutterflies is at least*

$$1 - \frac{n_k}{n_{k-1}} 2^{-\epsilon_{k-1}}$$

Proof: This lower bound can be derived by determining how many subbutterflies are only used to connect subbutterflies. Consider the set of subbutterflies characterized by α whose columns exhibit the ϵ_{k-1}-bit string $00\cdots1$ at any of the positions $\gamma + \epsilon_{k-1} - 1, \ldots, \gamma$ where γ ($\gamma \neq \alpha$) can be any multiple of n_{k-1}, or whose column's least significant ϵ_{k-1} bits are $00\cdots0$. Not all of these subbutterflies will be used to provide paths, but certainly all subbutterflies that will connect other subbutterflies are elements of this set.

The probability for the bit string to appear in one of the positions is $2^{-\epsilon_{k-1}}$. Since there are n_k/n_{k-1} events of the same likelihood (there are $n_k/n_{k-1} - 1$ possible values for γ) we simply multiply the probability with the number of events to obtain an upper bound on the overall probability for a subbutterfly to belong to the set mentioned above.

It therefore follows that the probability for a subbutterfly to belong to the complementary set, that is the set of butterflies that are used to emulate submeshes, is at least

$$1 - \frac{n_k}{n_{k-1}} 2^{-\epsilon_{k-1}} ,$$

which concludes the proof. ∎

2.5.2 The Number of Emulating Subbutterflies

Theorem 1 (The Number of Emulating Subbutterflies)

The number of butterflies that are used to emulate submeshes is

$$\frac{N_k}{N_{k-1}} p_E(k)$$

Proof: The overall number of subbutterflies $\frac{N_k}{N_{k-1}}$ multiplied by the probability $p_E(k)$ that a subbutterfly is used for emulation yields the number of emulating

subbutterflies. ∎

By combining Theorem 1, Lemma 2.5.1, and Corollary 2.1, we obtain a lower bound on the number of emulating subbutterflies as shown in Corollary 2.2.

Corollary 2.2 (The Minimum Number of Emulating Subbutterflies) *At least* $2^{n_k - n_{k-1}}(2 - 3 \cdot 2^{-\epsilon_{k-1}})$ *subbutterflies are used for emulation purposes at stage* k.

Proof:

$$\begin{aligned} \frac{N_k}{N_{k-1}} p_E(k) &= \frac{n_k 2^{n_k}}{n_{k-1} 2^{n_{k-1}}} \frac{n_{k-1}}{n_k} \pi_E(k) = 2^{n_k - n_{k-1}} \pi_E(k) \\ &> 2^{n_k - n_{k-1}}(2 - 3 \cdot 2^{-\epsilon_{k-1}}) \end{aligned}$$

∎

This lower bound will prove to be very convenient in our analysis, since the factor $(2 - 3 \cdot 2^{-\epsilon_{k-1}})$ has constant upper and lower bounds irrespective of the value of ϵ_{k-1}.

2.6 Choosing the Parameters for the Emulation

In this section it will be shown how the parameters for the assembly of butterflies from subbutterflies emulating submeshes are chosen and that the choice will satisfy

important constraints. In Section 2.2 we have already described the main ideas behind our choice of parameters.

Assume that we have a butterfly of size $n_{k-1}2^{n_{k-1}}$ that can emulate a $s_{k-1} \times s_{k-1}$ mesh. Associated with this emulation is a parameter ϵ_{k-1} describing the number of I/O ports of the mesh.

We want to find a butterfly of size $n_k 2^{n_k}$ that can emulate a mesh of size $s_{k-1}^g \times s_{k-1}^g$ where g is a parameter determining the growth of the meshes from stage to stage.

The parameters n_k and s_k will be determined as follows:

1. The first step is to determine a minimum, which we will name s'_k, for the size of the mesh at this stage in order to ensure a minimum growth of the mesh sizes from stage to stage. Let s'_k be the smallest integer that satisfies the following two conditions:

   (a) $s'_k \geq s^{g^{k-\omega}}$

   where g is a constant greater than 1 which we will call the *growth rate* of the emulation. This ensures the growth of the s_k's.

   (b) $s'_k \bmod (s_{k-1} - f_{k-1}) = f_{k-1}$.

   This ensures that the region F_k will also be in F_{k-1} and irregular submeshes are therefore avoided because of the corresponding choice of the f_k's later in this section.

If $s_k \leq s_{k-1}$ then all the parameters of this stage will be set equal to the last stage and one proceeds with the parameters of the next stage. In that case the current stage will be ignored.

2. Let n_k be the smallest integral multiple of n_{k-1} such that

$$\frac{N_k}{N_{k-1}} p_E(k) \geq \left(\frac{s'_k - f_{k-1}}{s_{k-1} - f_{k-1}} \right)^2$$

This condition ensures that the chosen butterfly contains at least as many emulating subbutterflies as submeshes.

Note, that n_k does *not* have to be a power of two as required in [13] which allows for a better adjustment of the size of the resulting butterfly to the size of the mesh to be emulated.

3. Let s_k be the greatest integer such that

(a) $s_k \bmod (s_{k-1} - f_{k-1}) = f_{k-1}$

This ensures that the mesh can be partitioned into overlapping submeshes without having to resort to irregular meshes as in [13].

(b)

$$\frac{N_k}{N_{k-1}} p_E(k) \geq \left(\frac{s_k - f_{k-1}}{s_{k-1} - f_{k-1}} \right)^2$$

This constraint ensures that the number of submeshes at most the num-

ber of emulating subbutterflies. This constraint is basically the same as in condition 2 except for the fact that n_k is constant and s_k is increased in order to maximally use the butterfly for emulation purposes.

4. $\epsilon_k = \left\lfloor \frac{n_k - 1 - \lg \frac{8s_k}{c}}{2} \right\rfloor$

   Choose the maximum ϵ_k that still leaves enough I/O ports. The larger the ϵ_k, the fewer communication butterflies there will be in the next stage.

5. $f_k = s_{k-1}$.

   This ensures that the boundary region coincides with the boundary of a submesh and therefore $F_k \subset F_{k-1}$.

This choice of parameters ensures that

1. The s_k's grow at a set rate.
2. We choose the smallest possible butterfly and
3. the butterfly characterized by n_k is maximally used for emulation purposes.

2.7 Validation of the Emulation Parameters

After having specified how the parameters governing the emulation are chosen for each stage, it is necessary to verify that the parameters always allow the embedding of subbutterflies (submeshes) in butterflies as described in [13].

We need to show that the overlapping regions in each mesh do not overlap each other, that is, that their width is at most half of the width of the mesh. The embedding requires that the parameter ϵ_k has a distinct, increasing value from stage to stage and it will be shown that this is true for the choice of parameters as given in Section 2.6.

2.7.1 The Size of the Overlapping Regions

Lemma 2.7.1 *The width of the mesh is at least twice as large as the width of the overlapping region in the mesh.*

$$\forall k : s_k \geq 2f_k$$

Proof: Proof by induction:

$k = 0$: The base case parameters satisfy this condition as set forth in the description of the base case.

$k - 1 \Rightarrow k$: Assuming that the relation holds for $k - 1$ we will show that it holds for k:

$$\begin{aligned} s_k &\geq \left(\sqrt{\frac{N_k}{N_{k-1}} p_E(k)} - 1\right)(s_{k-1} - f_{k-1}) + f_{k-1} \\ &= \left(\sqrt{2^{n_k - n_{k-1}} \pi_E(k)} - 1\right)(s_{k-1} - f_{k-1}) + f_{k-1} \end{aligned}$$

$$> \left(2^{\frac{n_{k-1}}{2}} - 1\right)(s_{k-1} - f_{k-1})$$

We now use the inductive assumption that $s_{k-1} \geq 2f_{k-1}$.

$$\begin{aligned} s_k &> \left(2^{\frac{n_{k-1}}{2}} - 1\right)(s_{k-1} - \frac{s_{k-1}}{2}) \\ &> \left(2^{\frac{n_{k-1}}{2}} - 1\right)\frac{s_{k-1}}{2} \end{aligned}$$

Since $n_k \geq n_0 \geq 7$ (see Section 2.3) and $f_k = s_{k-1}$ we proceed with

$$\begin{aligned} s_k &> \left(2^{\frac{7}{2}} - 1\right)\frac{f_k}{2} \\ &> 4\frac{f_k}{2} \\ &> 2f_k \end{aligned}$$

Conclusion: We can infer by induction that $s_k \geq 2f_k$ holds for all k.

■

2.7.2 The ϵ_ks are Distinct

The proof that the paths connecting subbutterflies at one stage do not interfere with the paths at another stage requires that the ϵ_k's be strictly monotonically increasing with respect to k, and thus distinct due to their integral nature. Therefore we have to show that this condition is satisfied when the parameters are chosen in

the proposed way. The value ϵ_k is defined as

$$\epsilon_k = \left\lfloor \frac{n_k - 1 - \lg \frac{8 s_k}{c}}{2} \right\rfloor$$

If we can show that

$$\begin{aligned} \frac{n_k - 1 - \lg \frac{8 s_k}{c}}{2} - \frac{n_{k-1} - 1 - \lg \frac{8 s_{k-1}}{c}}{2} &= \frac{n_k - n_{k-1} - \lg \frac{s_k}{s_{k-1}}}{2} \\ &> 1 , \end{aligned} \tag{2.3}$$

then it follows that $\epsilon_k - \epsilon_{k-1} \geq 1$, and the ϵ_k's increase from stage to stage. Since

$$\begin{aligned} s_k &< \sqrt{\frac{N_k}{N_{k-1}} p_E(k)}\,(s_{k-1} - f_{k-1}) + f_{k-1} \\ &= \sqrt{\frac{N_k}{N_{k-1}} p_E(k)}\,(s_{k-1} - f_{k-1}) + f_{k-1} \underbrace{\left(1 - \frac{N_k}{N_{k-1} p_E(k)}\right)}_{<1} \\ &< \sqrt{\frac{N_k}{N_{k-1}} p_E(k)} s_{k-1} < \sqrt{\frac{N_k}{N_{k-1}}} s_{k-1} , \end{aligned}$$

we can proceed by substituting s_k with $\sqrt{\frac{N_k}{N_{k-1}}} s_{k-1}$ in the term in Equation 2.3:

$$\begin{aligned} \frac{n_k - n_{k-1} - \lg \frac{s_k}{s_{k-1}}}{2} &> \frac{n_k - n_{k-1} - \lg \sqrt{\frac{N_k}{N_{k-1}}} - \lg \frac{s_{k-1}}{s_{k-1}}}{2} \\ &= \frac{n_k - n_{k-1} - \frac{\lg \frac{n_k}{n_{k-1}}}{2} - \frac{\lg 2^{n_k - n_{k-1}}}{2}}{2} \\ &= \frac{n_k - n_{k-1} - \lg \frac{n_k}{n_{k-1}}}{4} \end{aligned}$$

$$\begin{aligned} &\geq \frac{2n_{k-1} - n_{k-1} - \lg \frac{2n_{k-1}}{n_{k-1}}}{4} \\ &= \frac{n_{k-1} - \lg 2}{4} \\ &\geq \frac{n_0 - 1}{4} \\ &\geq \frac{5-1}{4} = 1 \end{aligned}$$

which gives us the desired result and ensures that the ϵ_k's will be distinct and increasing from stage to stage.

Chapter 3

Analysis of the Emulation

In this chapter we will analyze the emulation parameters with regard to the blowup and slowdown of the emulation. First we will make some observations that give us bounds for the number of stages and for the growth in the number of levels from stage to stage. To investigate the blowup we will present an expression for the number of nodes in the butterfly at each stage and show that this expression is within a constant factor of the number of nodes in the mesh.

After we know that the blowup is constant we can estimate the dilation of the embedding and show that the slowdown is constant as well.

Having shown that both the blowup and the slowdown are constant we conclude this chapter by inferring that the emulation is work-preserving and real-time.

3.1 The Number of Recursive Stages

To analyze the embedding we have to know how many recursive stages will be needed to completely embed the $s \times s$ mesh in a butterfly. In this section we will show an upper bound for the number of stages.

Definition 3.1 *Let ω be the number of recursive stages needed to emulate a mesh of size $N = s^2$ nodes.*

Theorem 2 (Upper Bound of the Number of Recursive Stages)

$$\omega \leq \left\lceil \log_g \log_{s_0} s \right\rceil$$

Proof: Since $s'_k \geq s^{g^{k-\omega}}$ and $s'_0 = s^{g^{-\omega}}$ it is clear that

$$\begin{aligned} s'_0 &= s^{-\left\lceil \log_g \log_{s_0} s \right\rceil} \leq s^{g^{-\log_g \log_{s_0} s}} \\ &= s^{\frac{1}{\log_{s_0} s}} = s^{\log_s s_0} = s_0 \end{aligned}$$

and therefore the mesh at stage $k = 0$ can be emulated by the base case butterfly and at most $\left\lceil \log_g \log_{s_0} s \right\rceil$ stages are necessary. ∎

Note, that as opposed to the approach by Koch et al. the number of stages in our analysis does not only depend on s, the width of the mesh, but also on the growth rate and the size of the mesh emulated in the base case emulation.

3.2 An Upper Bound of the Growth of n_k

In this section we will show that, after a constant number of stages, the number of levels in the butterfly at a stage is bounded by a constant multiple of the number of levels at the next lower stage.

The condition

$$1 \leq \frac{2^{n_{k-1}}}{8n_{k-1}^{g-1}} \tag{3.1}$$

will become true after a constant number of stages (depending on g, which is constant), since the parameter n_k is at least $n_0 2^k$. Using this relationship as a starting point we will now show that $n_k \leq (g+1)n_{k-1}$ after a constant number of stages. Let n_k be $(g+1)n_{k-1}$. Using the bound for $p_E(k)$ obtained in Corollary 2.1 we can derive

$$p_E(k) \geq \frac{n_{k-1}}{n_k}(2 - 3 \cdot 2^{-\epsilon_{k-1}}) = \frac{n_{k-1}}{(\lceil g \rceil + 1)n_{k-1}}(2 - 3 \cdot 2^{-\epsilon_{k-1}}) \geq \frac{1}{2(\lceil g \rceil + 1)}$$

and proceed to transform Equation 3.1 by replacing $\frac{1}{2(\lceil g \rceil + 1)}$ with $p_E(k)$:

$$1 \leq \frac{2^{n_{k-1}}}{8n_{k-1}^{g-1}} = \frac{(\lceil g \rceil + 1)2^{n_{k-1}}}{4n_{k-1}^{g-1}} \frac{1}{2(\lceil g \rceil + 1)} \leq \frac{(\lceil g \rceil + 1)2^{n_{k-1}}}{4n_{k-1}^{g-1}} p_E(k)$$

Since g is greater than 1 the expression $2^{(\lceil g \rceil - g)n_{k-1}+g-1}$ is also greater than one and we can proceed with

$$\begin{aligned} 1 &\leq \frac{(\lceil g \rceil + 1)2^{n_{k-1}}2^{(\lceil g \rceil - g)n_{k-1}+g-1}}{4n_{k-1}^{g-1}} p_E(k) \\ &= \frac{(\lceil g \rceil + 1)n_{k-1}2^{(\lceil g \rceil+1)n_{k-1}-1}}{4n_{k-1}^{g}2^{g(n_{k-1}-1)}} p_E(k) \end{aligned}$$

and therefore

$$\begin{aligned} (\lceil g \rceil + 1)n_{k-1}2^{(\lceil g \rceil+1)n_{k-1}-1} &\geq 4\left(n_{k-1}2^{n_{k-1}-1}\right)^g \frac{1}{p_E(k)} \\ N_k &\geq 4N_{k-1}^g \frac{1}{p_E(k)} \\ &\geq 4N_{k-1}^g \frac{1}{p_E(k)} \\ &= 4N_{k-1}N_{k-1}^{(g-1)} \frac{1}{p_E(k)} \\ &\geq 4N_{k-1}\left({s'_{k-1}}^2\right)^{(g-1)} \frac{1}{p_E(k)} \\ &= N_{k-1}4{s'_{k-1}}^{2(g-1)} \frac{1}{p_E(k)} \end{aligned}$$

Furthermore, because

$$s_{k-1} \geq s'_{k-1}\ ,\ s'_k = {s'_{k-1}}^g \text{ and } f_{k-1} \leq \frac{s_{k-1}}{2}\ ,$$

we can proceed with

$$\begin{aligned} N_k &\geq N_{k-1} \left(\frac{{s'_{k-1}}^g}{\frac{s'_{k-1}}{2}} \right)^2 \frac{1}{p_E(k)} \\ &\geq N_{k-1} \left(\frac{s'_k - f_{k-1}}{s_{k-1} - f_{k-1}} \right)^2 \frac{1}{p_E(k)} \end{aligned}$$

Lemma 3.2.1 *After a constant number of stages K the ratio $\frac{n_k}{n_{k-1}}$ is at most* $\lceil g \rceil + 1$.

$$\exists\, K : \forall\, k \geq K : n_k \leq (\lceil g \rceil + 1) n_{k-1}$$

Proof: As shown in the previous paragraphs, the following condition is satisfied for $n_k \leq (\lceil g \rceil + 1) n_{k-1}$ after a constant number of stages:

$$N_k \geq N_{k-1} \left(\frac{s'_k - f_{k-1}}{s_{k-1} - f_{k-1}} \right)^2 \frac{1}{p_E(k)}$$

n_k is defined to be the smallest integral multiple of n_{k-1} that fulfills this condition (see page 38 item 2). Since $(\lceil g \rceil + 1) n_{k-1}$ is a multiple of n_{k-1} and it satisfies the condition, it is clear that n_k can be at most $(\lceil g \rceil + 1) n_{k-1}$. ∎

3.3 The Size of the Butterfly

In this section we will show that the number of butterfly nodes is within a constant factor of the number of mesh nodes.

The choice of parameters described in section 2.6 leads to a maximal butterfly size of

$$N_k \leq N_{k-1} \left(\frac{s_k - f_{k-1}}{s_{k-1} - f_{k-1}} + 1 \right)^2 \frac{1}{p_E(k)} \tag{3.2}$$

This relation expresses the fact that if one more row and column of submeshes had been added to the mesh, it would not have fit into the butterfly. This recurrence relation can be unfolded as follows:

$$\begin{aligned}
N_k &\leq s_k^2 \frac{N_{k-1}}{s_{k-1}^2} \left(\frac{s_{k-1} - \frac{s_{k-1} f_{k-1}}{s_k}}{s_{k-1} - f_{k-1}} + \frac{s_{k-1}}{s_k} \right)^2 \frac{1}{p_E(k)} \\
N_k &\leq s_k^2 \frac{N_0}{s_0^2} \prod_{i=1}^{k} \left[\left(\frac{s_{i-1} - \frac{s_{i-1} f_{i-1}}{s_i}}{s_{i-1} - f_{i-1}} + \frac{s_{i-1}}{s_i} \right)^2 \frac{1}{p_E(i)} \right] \qquad (3.3) \\
N_k &\leq s_k^2 \underbrace{\frac{N_0}{s_0^2}}_{=O(1)} \underbrace{\prod_{i=1}^{k} \left(\frac{s_{i-1} - \frac{s_{i-1} f_{i-1}}{s_i}}{s_{i-1} - f_{i-1}} + \frac{s_{i-1}}{s_i} \right)^2}_{=P_1} \underbrace{\prod_{i=1}^{k} \frac{1}{p_E(i)}}_{=P_2} \qquad (3.4)
\end{aligned}$$

If the product $P_1 P_2$ is bounded by a constant, then N_k will be $O(s_k^2)$. P_1 can be interpreted as the contribution of the node redundancy due to the overlap of the submeshes to the blowup, whereas P_2 accounts for the waste of butterfly nodes due to the non-emulating subbutterflies. We will investigate P_1 and P_2 separately in Sections 3.3.1 and 3.3.2, respectively..

3.3.1 The Blowup Attributable to Redundancy

In this section we will show that the waste of butterfly nodes due to the redundancy caused by the overlap of submeshes is bounded by a constant ratio of the overall number of nodes.

The product P_1 as defined in Equation 3.4 is

$$P_1 = \prod_{i=1}^{k} \left(\frac{s_{i-1} - \frac{s_{i-1} f_{i-1}}{s_i}}{s_{i-1} - f_{i-1}} + \frac{s_{i-1}}{s_i} \right)^2 .$$

Note that a product $\prod_{n=1}^{\infty}(1 + a_n)$ converges if and only if the series $\sum_{n=1}^{\infty} a_n$ converges. We therefore write

$$P_1 = \left(\prod_{i=1}^{k} \left(1 + \frac{f_{k-1} - \frac{s_{k-1} f_{k-1}}{s_k}}{s_{k-1} - f_{k-1}} + \frac{s_{k-1}}{s_k} \right) \right)^2$$

and investigate the corresponding sum

$$\begin{aligned} \sum_{j=1}^{k} \left(\frac{f_{k-1} - \frac{s_{k-1} f_{k-1}}{s_k}}{s_{k-1} - f_{k-1}} + \frac{s_{k-1}}{s_k} \right) &= \sum_{j=1}^{k} \frac{f_{k-1} - \frac{s_{k-1} f_{k-1}}{s_k}}{s_{k-1} - f_{k-1}} + \sum_{j=1}^{k} \frac{s_{k-1}}{s_k} \\ &\leq \sum_{j=1}^{k} \frac{f_{k-1}}{s_{k-1} - f_{k-1}} + \sum_{j=1}^{k} \frac{s_{k-1}}{s_k} \end{aligned} \tag{3.5}$$

Lemma 3.3.1

$$\frac{s_k}{s_{k-1}} > \sqrt{2^{n_{k-1}-3}}$$

Proof:

$$\begin{aligned}
\frac{s_k}{s_{k-1}} &> \frac{\left(\sqrt{\frac{N_k}{N_{k-1}}p_E(k)}-1\right)(s_{k-1}-f_{k-1})+f_{k-1}}{s_{k-1}} \\
&> \frac{\left(\sqrt{2^{n_k-n_{k-1}}\pi_E(k)}-1\right)(s_{k-1}-f_{k-1})+f_{k-1}}{s_{k-1}} \\
&> \frac{\left(\sqrt{2^{n_k-n_{k-1}}}-1\right)(s_{k-1}-f_{k-1})+f_{k-1}}{s_{k-1}} \\
&= \frac{\sqrt{2^{n_k-n_{k-1}}}(s_{k-1}-f_{k-1})-s_{k-1}+2f_{k-1}}{s_{k-1}} \\
&> \frac{\sqrt{2^{n_k-n_{k-1}}}(s_{k-1}-f_{k-1})-s_{k-1}}{s_{k-1}}
\end{aligned}$$

Since $s_k \geq 2f_k$ (Lemma 2.7.1) and $n_k \geq 2n_{k-1}$ (see page 38) it follows that

$$\begin{aligned}
\frac{s_k}{s_{k-1}} &> \frac{\sqrt{2^{n_{k-1}}}\left(s_{k-1}-\frac{s_{k-1}}{2}\right)-s_{k-1}}{s_{k-1}} \\
&= \frac{\sqrt{2^{n_{k-1}}}}{2}-1 \\
&= \sqrt{2^{n_{k-1}-2}}-1 \\
&> \sqrt{2^{n_{k-1}-3}}
\end{aligned}$$

■

Lemma 3.3.2 *The sum* $\sum_{j=1}^{k}\frac{s_{k-1}}{s_k}$ *is bounded for all* k.

Proof: We will show that the sum is bounded by a geometric series which is, of course, convergent. With Lemma 3.3.1 we can bound the elements of the sum by

$$\frac{s_{k-1}}{s_k} < \sqrt{2^{3-n_{k-1}}}$$

Since the quotient of two successive elements of the bounding sum is at least

$$\frac{\sqrt{2^{3-n_{k-1}}}}{\sqrt{2^{3-n_k}}} = \sqrt{2^{n_k-n_{k-1}}} > \sqrt{2^{n_{k-1}}} > \sqrt{2^{n_0}} > \sqrt{2^5}$$

the sum $\sum_{j=1}^{k} \sqrt{2^{3-n_{k-1}}}$ converges and with it the bounded sum. ∎

Lemma 3.3.3 *The sum* $\sum_{j=0}^{k} \frac{f_k}{s_k - f_k}$ *is bounded for all* k.

Proof: Again we will show that the sum is bounded by a geometric series. With Lemma 3.3.1 we can bound the elements of the sum by

$$\frac{f_k}{s_k - f_k} = \frac{1}{\frac{s_k}{f_k} - 1} = \frac{1}{\frac{s_k}{s_{k-1}} - 1} < \frac{1}{\sqrt{2^{n_{k-1}-3}} - 1}$$

Since the quotient of two successive elements of the bounding sum is at least

$$\frac{\sqrt{2^{n_k-3}} - 1}{\sqrt{2^{n_{k-1}-3}} - 1} > \frac{\sqrt{2^{n_k-3}}}{\sqrt{2^{n_{k-1}-3}}} = \sqrt{2^{n_k-n_{k-1}}} > \sqrt{2^{n_{k-1}}} > \sqrt{2^{n_0}} > \sqrt{2^5}\,,$$

the sum $\sum_{j=1}^{k} \frac{1}{\sqrt{2^{n_{k-1}-3}} - 1}$ converges and with it the bounded sum. ∎

Lemma 3.3.4 *The product*

$$P_1 = \prod_{i=1}^{k} \left(\frac{s_{i-1} - \frac{s_{i-1} f_{i-1}}{s_i}}{s_{i-1} - f_{i-1}} + \frac{s_{i-1}}{s_i} \right)^2$$

is bounded by a constant for all k.

Proof: By combining the results of Lemma 3.3.2, Lemma 3.3.3, and Equation 3.5, we know that the sum

$$\sum_{j=1}^{k} \left(\frac{f_{k-1} - \frac{s_{k-1} f_{k-1}}{s_k}}{s_{k-1} - f_{k-1}} + \frac{s_{k-1}}{s_k} \right)$$

is bounded by a constant for all k.

Since a product $\prod_{n=1}^{\infty}(1 + a_n)$ converges if and only if the series $\sum_{n=1}^{\infty} a_n$ converges, we can infer that P_1 converges and therefore has a constant upper bound. ∎

3.3.2 The Blowup Attributable to Communication Sub-butterflies

In this section we will show that the product P_2 in Equation 3.4 is bounded by a constant.

$$P_2 = \prod_{j=1}^{k} \frac{1}{p_E(j)}$$

Recall that $\frac{n_k}{n_{k-1}} \leq \lceil g \rceil + 1$ for $k \geq K$ (Lemma 3.2.1). We will partition the product into a finite product consisting of the first K elements and the remaining product.

Lemma 3.3.5 *The product*

$$\prod_{j=K}^{k} \frac{1}{p_E(j)} ,$$

where K is the constant described in Lemma 3.2.1, is bounded by a constant for all k.

Proof: We will use the lower bound for $p_E(k)$ derived in Lemma 2.5.2:

$$p_E(j) \geq 1 - \frac{n_j}{n_{j-1}} 2^{-\epsilon_{j-1}}$$

Since $\frac{n_j}{n_{j-1}} \leq \lceil g \rceil + 1 \leq g + 2$ and $\epsilon_j \geq 2 + j$, we can bound $p_E(j)$ further by

$$p_E(j) \geq 1 - (g+2)2^{-j-1} .$$

Inserting this bound into the product yields

$$\prod_{j=K}^{k} \frac{1}{p_E(j)} \leq \prod_{j=K}^{k} \frac{1}{1-(g+2)2^{-j-1}} .$$

This product is bounded for all k if the sum

$$\sum_{j=K}^{k} \left(\frac{1}{1-(g+2)2^{-j-1}} - 1 \right) = \sum_{j=K}^{k} \frac{1}{\frac{2^{j+1}}{g+2} - 1}$$

is bounded. This can easily be verified by looking at the ratio of two successive elements of the sum, which is at least 2:

$$\begin{aligned} \frac{1}{\frac{2^{j+1}}{g+2} - 1} \cdot \left(\frac{2^{j+2}}{g+2} - 1 \right) &\geq \frac{1}{\frac{2^{j+1}}{g+2}} \cdot \frac{2^{j+2}}{g+2} \\ &= \frac{2^{j+2}}{2^{j+1}} \\ &= 2 \end{aligned}$$

■

This result allows us to bound the product P_2 in the followin lemma.

Lemma 3.3.6 *The product*

$$P_2 = \prod_{j=1}^{k} \frac{1}{p_E(j)}$$

is bounded by a constant for all k.

Proof: The product P_2 can be partitioned in the following way:

$$P_2 = \prod_{j=1}^{K-1} \frac{1}{p_E(j)} \prod_{j=K}^{k} \frac{1}{p_E(j)}$$

In Lemma 3.3.5 we have shown that the second product is bounded by a constant. The finite (first) product is also bounded and therefore we can conclude that P_2 is bounded by a constant. ∎

3.3.3 Constant Blowup of the Emulation

Theorem 3 (The Emulation Has a Constant Blowup)

The blowup of the emulation of the $s_k \times s_k$-node mesh on the $n_k 2^{n_k-1}$-node butterfly is $O(1)$.

Proof: Equation 3.4 shows that the number of nodes in the butterfly at stage k is at most

$$N_k \leq s_k^2 \underbrace{\frac{N_0}{s_0^2}}_{=O(1)} \cdot P_1 \cdot P_2$$

In Lemma 3.3.4 and Lemma 3.3.6 we have shown that both P_1 and P_2 are bounded by a constant for all k. Therefore N_k is within a constant factor of s_k^2, the number of nodes in the mesh at stage k and therewith $O(s_k^2)$ ∎

Since the emulation has a constant blowup at each stage of the recursion, this is also true for the last stage ω and we can conclude that the emulation of the $s_\omega \times s_\omega$ mesh on the N_ω-node butterfly has a constant blowup.

3.3.4 Caveat!

The result of the previous section (Theorem 3) shows that the $s_\omega \times s_\omega$ mesh is emulated with constant blowup. The mesh to be emulated, however, is a $s \times s$ mesh. Since $s_\omega \geq s$ we can emulate this mesh on the N_ω-node butterfly as well. Whether this emulation can be done with constant blowup will be investigated in this section, but we can already say that this is not necessarily the case.

For the emulation of the $s \times s$ mesh to be of constant blowup, the size s^2 of the mesh obviously has to be within a constant factor of the size s_ω^2 of the $s_\omega \times s_\omega$ mesh, since that mesh size is within a constant factor of the size of the butterfly.

However, no provisions have been made to ensure that s^2 is sufficiently close to s_ω^2. In the following paragraphs we will show that in the worst s^2 is indeed too small with respect to s_ω^2. Let s_z^2 denote the size of the $s \times s$ mesh in the worst case.

The $N_\omega = n_\omega 2^{n_\omega - 1}$-node butterfly consists of subbutterflies of the size $N_{\omega-1} = n_{\omega-1} 2^{n_{\omega-1} - 1}$ where n_ω is a multiple of $n_{\omega-1}$. Let m be $\frac{n_\omega}{n_{\omega-1}}$. We will now investigate the value of s_z^2.

The need of at least one additional subbutterfly was the reason that the $(n_\omega - n_{\omega-1})2^{n_\omega - n_{\omega-1} - 1}$-node butterfly was not used for the emulation and we had to resort to the next larger butterfly N_ω. Therefore s^2 can be so small that to emulate the $s \times s$ mesh only one more subbutterfly is needed for the emulation of a submesh in addition the number of emulating subbutterflies already available in the next

smaller butterfly of size $(n_\omega - n_{\omega-1})2^{n_\omega - n_{\omega-1}-1}$.

We can therefore conclude that the value s_z^2 is at most the number of emulating subbutterflies in the $(n_\omega - n_{\omega-1})2^{n_\omega - n_{\omega-1}-1}$-node butterfly plus one multiplied by the size of the mesh emulated in each of these subbutterflies.

$$\begin{aligned} s_z^2 &\leq \left(\frac{(n_\omega - n_{\omega-1})2^{n_\omega - n_{\omega-1}-1}}{N_{\omega-1}} p_E(k) + 1 \right) s_{\omega-1} \\ &\leq \frac{(n_\omega - n_{\omega-1})2^{n_\omega - n_{\omega-1}-1}}{N_{\omega-1}} s_{\omega-1} \end{aligned}$$

Since $s_{\omega-1}^2 < l \cdot N_{\omega-1}$ for the load l (determined by the base case emulation), it follows that

$$\begin{aligned} s_z^2 &\leq \frac{(n_\omega - n_{\omega-1})2^{n_\omega - n_{\omega-1}}}{N_{\omega-1}} C N_{\omega-1} \\ &= C\,(n_\omega - n_{\omega-1})\,2^{n_\omega - n_{\omega-1}}\ . \end{aligned}$$

The blowup of the emulation of the s_z^2-node mesh on the N_ω-node butterfly is therefore

$$\begin{aligned} \frac{N_\omega}{s^2} &\geq \frac{N_\omega}{C(n_\omega - n_{\omega-1})2^{n_\omega - n_{\omega-1}}} \\ &= \frac{m\, n_{\omega-1} 2^{m\, n_{\omega-1}}}{C_1(m\, n_{\omega-1} - n_{\omega-1})2^{m\, n_{\omega-1} - n_{\omega-1}}} \\ &= \frac{m\, n_{\omega-1} 2^{m\, n_{\omega-1}}}{C_1(m-1)n_{\omega-1}2^{(m-1)n_{\omega-1}}} \\ &= \frac{m}{C_1(m-1)} 2^{n_{\omega-1}} \end{aligned}$$

Since the expression $2^{n_\omega - 1}$ increases towards infinity with increasing mesh sizes, it follows that the blowup of the emulation of the $s_z \times s_z$ mesh on the N_ω mesh is not constant.

Because the mesh size s_z^2 was defined as the smallest possible size for the $s \times s$ mesh as compared to the N_ω butterfly we can conclude the argument by stating that not all meshes can be emulated in a work-preserving manner.

3.4 The Slowdown of the Emulation

In this section we will bound the slowdown of the $s \times s$ mesh on the N_ω-node butterfly. In the first subsection we establish a bound on the growth of the parameter s_k from stage to stage. This result and the dilation of the embedding determined in the second subsection allows us to finally bound the slowdown of the emulation in the last part of this section.

3.4.1 A constant upper bound for the growth of s_k

To show an upper bound for the growth of s_k we need to observe two different cases. Lemma 3.4.1 will show the bound for the case that $n_k = 2N_{k-1}$, whereas Lemma 3.4.2 prepares Lemma 3.4.3 which shows the bound for the case that $n_k \geq$

$3n_{k-1}$. Theorem 4 then concludes this section by combining the results.

Lemma 3.4.1 *For $n_k = 2n_{k-1}$ the logarithm $\log_{s'_k} s_k$ has a constant upper bound.*

Proof: It is clear that $s_k'^2$ must be larger[1] than s_{k-1}^2. Furthermore, since the number of subbutterflies of size $N_{k-1} = n_{k-1}2^{n_{k-1}-1}$ in the butterfly of size $N_k = n_k 2^{n_k} = 2n_{k-1}2^{2n_{k-1}-1}$ is $\frac{N_k}{N_{k-1}} = 2 \cdot 2^{n_{k-1}}$ and each subbutterfly emulates a mesh of size s_{k-1}^2, it follows that s_k^2 is at most $2s_{k-1}^2 2^{n_{k-1}}$.

Using these bounds for s'_k and s_k, we can now derive the bound for the logarithm:

$$\begin{aligned}
\log_{s'_k} s_k &= \log_{s_k'^2} s_k^2 \\
&\leq \log_{s_{k-1}^2} \left(2s_{k-1}^2 2^{n_{k-1}}\right) \\
&= \log_{s_{k-1}^2} 4 + 1 + \log_{s_{k-1}^2} 2^{n_{k-1}-1} \\
&= \log_{s_{k-1}^2} 4 + 1 + \log_{s_{k-1}^2} n_{k-1}2^{n_{k-1}-1} - \underbrace{\log_{s_{k-1}^2} n_{k-1}}_{>0} \\
&< \log_{s_{k-1}^2} 4 + 1 + \log_{s_{k-1}^2} N_{k-1}
\end{aligned}$$

We know that the number of butterfly nodes in stage $k-1$ is within a constant factor of the number of mesh nodes (that is $N_{k-1} \leq A \cdot s_{k-1}^2$, A constant).

[1] This is true because otherwise the k-th stage would not have increased the size of the butterfly, and would have been identified with the $(k-1)$-st stage.

This yields

$$\begin{aligned}
\log_{s_k'} s_k &< \log_{s_{k-1}^2} 4 + 1 + \log_{s_{k-1}^2} A \cdot s_{k-1}^2 \\
&= \log_{s_{k-1}^2} 4 + 1 + \log_{s_{k-1}^2} A + 1 \\
&< \log_{s_0^2} 4 + 1 + \log_{s_0^2} A + 1 \\
&= \textit{constant}
\end{aligned}$$

■

Lemma 3.4.2 *For $n_k \geq 3n_{k-1}$ the following relation holds:*

$$\left(\sqrt{2^{n_k - 2n_{k-1}} \pi_E(k)}\,(s_{k-1} - f_{k-1}) + f_{k-1}\right)^2 > \sqrt{\frac{N_k}{N_{k-1}} p_E(k)}\,(s_{k-1} - f_{k-1}) + f_{k-1}$$

Proof: Since $n_k - 3n_{k-1} \geq 0$ and also $2 \lg s_{k-1} \geq 2$ for $s_{k-1} \geq 2$, it follows that

$$\begin{aligned}
n_k - 3n_{k-1} + 2 \lg s_{k-1} &\geq 2 = 2 \lg 2 \\
\lg \pi_E(k) + n_k - 3n_{k-1} + 2 \lg s_{k-1} - 2 \lg 2 &\geq 0 \\
n_k - 3n_{k-1} + \pi_E(k) + 2 \lg \frac{s_{k-1}}{2} &\geq 0 \qquad (3.6)
\end{aligned}$$

Lemma 2.7.1 says that $s_{k-1}/2 > f_{k-1}$. By adding s_{k-1} on both sides of the negated relation we get $s_{k-1}/2 < s_{k-1} - f_{k-1}$. Using this relation we can

develop Equation 3.6 further:

$$\begin{aligned} n_k - 3n_{k-1} + \lg \pi_E(k) + 2\lg(s_{k-1} - f_{k-1}) &\geq 0 \\ 2^{n_k - 3n_{k-1} + \lg \pi_E(k) + 2\lg(s_{k-1} - f_{k-1})} &\geq 1 \\ 2^{n_k - 3n_{k-1}} \pi_E(k) (s_{k-1} - f_{k-1})^2 &\geq 1 \\ \sqrt{2^{n_k - 3n_{k-1}}} \sqrt{\pi_E(k)} (s_{k-1} - f_{k-1}) &\geq 1 \end{aligned}$$

Since trivially $\sqrt{2^{n_{k-1}}} > \sqrt{2^{n_{k-1}}} - 2$ we can multiply these two relations and get

$$\begin{aligned} \sqrt{2^{n_{k-1}}} \sqrt{2^{n_k - 3n_{k-1}}} \sqrt{\pi_E(k)} (s_{k-1} - f_{k-1}) &> \sqrt{2^{n_{k-1}}} - 2 \\ \sqrt{2^{n_k - 2n_{k-1}}} \sqrt{\pi_E(k)} (s_{k-1} - f_{k-1}) + 2 &> \sqrt{2^{n_{k-1}}} \, . \end{aligned}$$

Multiply both sides with $\sqrt{2^{n_k - n_{k-1}} \pi_E(k)} (s_{k-1} - f_{k-1})$.

$$\begin{aligned} 2^{n_k - 2n_{k-1}} \pi_E(k) (s_{k-1} - f_{k-1})^2 + \underbrace{2\sqrt{2^{n_k - 2n_{k-1}}} \sqrt{\pi_E(k)} (s_{k-1} - f_{k-1})}_{>0} \\ > \sqrt{2^{n_k - n_{k-1}}} \sqrt{\pi_E(k)} (s_{k-1} - f_{k-1}) \end{aligned}$$

Multiply the marked term with f_{k-1} .

$$2^{n_k-2n_{k-1}}\pi_E(k)\,(s_{k-1}-f_{k-1})^2+2\sqrt{2^{n_k-2n_{k-1}}}\sqrt{\pi_E(k)}\,(s_{k-1}-f_{k-1})\,f_{k-1}$$
$$>\sqrt{2^{n_k-n_{k-1}}}\sqrt{\pi_E(k)}\,(s_{k-1}-f_{k-1})$$

Add $f_{k-1}^2 \geq f_{k-1}$.

$$2^{n_k-2n_{k-1}}\pi_E(k)\,(s_{k-1}-f_{k-1})^2+f_{k-1}^2+2\sqrt{2^{n_k-2n_{k-1}}}\sqrt{\pi_E(k)}\,(s_{k-1}-f_{k-1})\,f_{k-1}$$
$$>\sqrt{2^{n_k-n_{k-1}}}\sqrt{\pi_E(k)}\,(s_{k-1}-f_{k-1})+f_{k-1}$$

The left side is of the form $a^2+b^2+2ab=(a+b)^2$.

$$\left(\sqrt{2^{n_k-2n_{k-1}}\pi_E(k)}\,(s_{k-1}-f_{k-1})+f_{k-1}\right)^2$$
$$>\sqrt{2^{n_k-n_{k-1}}\frac{n_k}{n_{k-1}}p_E(k)}\,(s_{k-1}-f_{k-1})+f_{k-1}$$
$$>\sqrt{\frac{N_k}{N_{k-1}}p_E(k)}\,(s_{k-1}-f_{k-1})+f_{k-1}$$

■

Lemma 3.4.3 *For $n_k \geq 3n_{k-1}$ the logarithm $\log_{s'_k} s_k$ is bounded by 2.*

Proof: Since n_k has been chosen as the smallest multiple that can accommodate the mesh of size ${s'_k}^2$, it is clear that this mesh does not fit into the next smaller

butterfly of size $(n_k - n_{k-1})2^{n_k - n_{k-1}}$. This can be formulated as follows.

$$\begin{aligned} s_k'^2 &> \left(\sqrt{\frac{(n_k - n_{k-1})2^{n_k - n_{k-1}}}{N_{k-1}} p_E(k)} \, (s_{k-1} - f_{k-1}) + f_{k-1} \right)^2 \\ s_k' &> \sqrt{2^{n_k - 2n_{k-1}} \pi_E(k)} \, (s_{k-1} - f_{k-1}) + f_{k-1} \end{aligned}$$

Furthermore s_k has been chosen so as to maximally fill the butterfly of size N_k with emulating butterflies of size N_{k-1}.

$$\begin{aligned} s_k^2 &\leq \left(\sqrt{\frac{N_k}{N_{k-1}} p_E(k)} \, (s_{k-1} - f_{k-1}) + f_{k-1} \right)^2 \\ s_k &\leq \sqrt{\frac{N_k}{N_{k-1}} p_E(k)} \, (s_{k-1} - f_{k-1}) + f_{k-1} \end{aligned}$$

With Lemma 3.4.2 it follows that $\log_{s_k'} s_k < 2$. ∎

Theorem 4 (Upper Bound of the Mesh Growth)

For all k

$$s_k < s_k'^{\sigma}$$

where $\sigma > 1$ is a constant.

Proof: In Lemma 3.4.1 and Lemma 3.4.3 we show that the logarithm $\log_{s_k'} s_k$ is bounded by constants in the two cases that $n_k = 2n_{k-1}$ and $n_k \geq 3n_{k-1}$. We can conclude this proof by letting σ be the maximum of these two constants.

∎

This theorem is important as it allows us to establish relationships between the s_k's, which we will do in the following corollary.

Corollary 3.1

$$s_k < s_{k-1}^{\sigma g} \text{ and } s_k > s_{k-1}^{\frac{g}{\sigma}}$$

Proof: 1. $s_k < {s'_k}^{\sigma} = {s'_{k-1}}^{\sigma g} \leq s_{k-1}^{\sigma g}$

2. $s_k \geq s'_k = {s'_{k-1}}^{g} > s_{k-1}^{\frac{g}{\sigma}}$

■

Theorem 4 also lets us establish an interval for each s_k with respect to s, the width of the mesh to be emulated.

Corollary 3.2

$$s^{g^{k-\omega}} \leq s_k \leq s^{\frac{g^{k-\omega}}{\sigma}}$$

Proof: Since s'_k is defined as $s^{g^{k-\omega}}$ (see page 37), the claim follows with $s'_k \leq s_k$ and Theorem 4. ■

3.4.2 Dilation of the Embedding

In order to determine the slowdown of the emulation, we need to know the dilation of the embedding. The dilation is the maximum number of edges in the path along which pebbles are sent from one mesh node to its neighbor. It gives the maximum distance between two adjacent mesh nodes in the butterfly.

Definition 3.2 *Let d_k denote the dilation for corresponding mesh nodes in F_k in neighboring submeshes at stage k.*

To send a pebble from a node in F_k which resides in subbutterfly u to another node in the subbutterfly v emulating an adjacent mesh, we route the pebble from u to a node in level zero of the butterfly. This requires a maximum path of n_k edges. The pebble is then sent to another node in level zero by routing a permutation up and down all the n_k butterfly levels. This results in another maximum $2n_k$ edges being added to the path length. The final step of sending the pebble to the destination subbutterfly adds a maximum of n_k edges to the path length. Therefore the path needed to route a pebble from a submesh to another arbitrary submesh is $4n_k$.

As shown in Theorem 3 the number of butterfly nodes is within a constant factor A of the number of mesh nodes for all but the last stage ω, that is,

$$\begin{aligned} N_k &\leq As_k^2 \\ n_k \lg n_k &\leq \lg\left(As_k^2\right) \\ n_k &\leq \lg\left(As_k^2\right) - \lg n_k \\ &= 2\lg s_k + \lg\frac{A}{n_k} \\ 4n_k &\leq 4\lg s_k + 4\lg\frac{A}{n_k}\,. \end{aligned} \tag{3.7}$$

Lemma 3.4.4 *The dilation d_k at stage k $(0 \leq k < \omega)$ is $O(\lg s_k)$.*

Proof: From relation 3.7 it follows that for $n_k > A$ the path length (dilation) is bounded by $4 \lg s_k$ and therefore $d_k = O(\lg s_k)$. ∎

We will show that this lemma also holds for the last stage ω by looking at the two cases that

$n_\omega / n_{\omega-1} = 2$: Since $s_\omega \leq s_{\omega-1}^g$ it follows that

$$\begin{aligned} D_\omega &\leq 4n_\omega = 8n_{\omega-1} \\ &= O(\lg s_{\omega-1}) \\ &\leq O(\lg s_\omega^{\frac{1}{g}}) \\ &= O(\frac{\lg s_\omega}{g}) = O(\lg s_\omega) \end{aligned}$$

$n_\omega / n_{\omega-1} > 2$: Since the next smaller butterfly of size $(n_\omega - n_{\omega-1})2^{n_\omega - n_{\omega-1}}$ could not acommodate the mesh of size $s_\omega \times s_\omega$ is it clear that

$$\begin{aligned} n_\omega - n_{\omega-1} &\leq O(\lg s_\omega) \\ n_\omega &\leq O(\lg s_\omega) + n_{\omega-1} \\ &= O(\lg s_\omega) + O(\lg s_{\omega-1}) \\ &= O(\lg s_\omega) \end{aligned}$$

Lemma 3.4.5 *The dilation d_k at stage k $(0 \leq k \leq \omega)$ is $O(\lg s_k)$.*

Proof: See Lemma 3.4.4 and the previous paragraph. ∎

3.4.3 Emulation Slowdown

Let T_k be the time to emulate f_k steps of a $s_k \times s_k$ mesh on a N_k-node butterfly. This emulation is divided into f_k/f_{k-1} phases: In each phase, we attempt to recursively emulate f_{k-1} steps of the $s_{k-1} \times s_{k-1}$ submeshes on the N_{k-1}-node subbutterflies.

Since the base case size has a constant upper bound the emulation of f_0 steps on the mesh of size $s_0 \times s_0$ is $O(f_0) = O(1)$. For all other stages each phase requires time $T_{k-1} + d_k$ for the recursive emulation and the delivery of pebbles along the paths of maximum length $d_k = O(\lg s_k)$.

$$\begin{aligned} T_0 &= O(f_0) = O(1) \\ T_k &= \frac{f_k}{f_{k-1}} (T_{k-1} + O(\lg s_k)) \end{aligned}$$

This recurrence can be unfolded as follows

$$\begin{aligned} T_\omega &= \sum_{k=1}^{\omega} \left(\prod_{j=k+1}^{\omega} \frac{f_k}{f_{k-1}} \right) O(\lg s_k) + T_0 \prod_{j=1}^{\omega} \frac{f_k}{f_{k-1}} \\ &= \sum_{k=1}^{\omega} \frac{f_\omega \cdot O(\lg s_k)}{f_{k-1}} + T_E(0) \frac{f_\omega}{f_0} \end{aligned}$$

$$
\begin{aligned}
&= f_\omega \left(\underbrace{\sum_{k=1}^{\omega} \frac{O(\lg s_k)}{f_k}}_{S_R} + \underbrace{\frac{T_\omega}{f_\omega}}_{S_B} \right) \\
&= T \times S_E
\end{aligned}
$$

where S_E is the slowdown of the emulation. S_E is the sum of the slowdown S_R yielded by the recursion and the slowdown S_B resulting from the emulation of the base case. The sizes of the base case butterfly and mesh have a constant upper bound (by definition), so that S_B is also constant. It remains to investigate S_R which will be the purpose of the following paragraphs.

$$
\begin{aligned}
S_R &= \sum_{k=1}^{\omega} \frac{O(\lg s_k)}{f_k} \\
&= O\left(\sum_{k=1}^{\omega} \frac{\lg s_k}{f_k} \right)
\end{aligned}
$$

Recall that $f_k = s_{k-1}$ (see page 39) and the bounds for s_k from Corollary 3.2.

$$
\begin{aligned}
S_R &\leq O\left(\sum_{k=1}^{\omega} \frac{\lg s_k}{s_{k-1}} \right) \\
&\leq O\left(\sum_{k=1}^{\omega} \frac{\lg s^{\frac{g^{k-\omega}}{\sigma}}}{s^{g^{k-1-\omega}}} \right) \\
&= O\left(\sum_{k=1}^{\omega} \frac{\frac{g^{k-\omega}}{\sigma} \lg s}{s^{g^{k-1-\omega}}} \right) \\
&= O\left(\frac{\lg s}{\sigma} \sum_{k=1}^{\omega} g^{k-\omega} s^{-g^{k-1-\omega}} \right)
\end{aligned}
$$

$$= O\left(\lg s \sum_{k=1}^{\omega} g^{k-\omega} s^{-g^{k-1-\omega}}\right)$$

In order to construct a constant upper bound for the slowdown we will investigate the integral bounding the sum:

Lemma 3.4.6 *The integral*

$$I = \lg s \int_0^{\omega} g^{k-\omega} s^{-g^{k-1-\omega}}\, dk$$

is bounded by a constant for all s.

Proof: First we will determine the indefinite integral: By defining $u = -g^{k-1-\omega}$ and deriving $\frac{du}{dk} = -g^{k-1-\omega} \ln g$, we can substitute dk.

$$\begin{aligned} I &= \lg s \int g^{k-\omega} s^{-g^{k-1-\omega}} \frac{du}{-g^{k-1-\omega} \ln g} \\ &= -\frac{g \lg e}{\ln g} \int s^u \ln s \, du \\ &= -\frac{g \lg e}{\ln g} \int s^u \, du \\ &= -\frac{g \lg e}{\ln g} s^u \\ &= -\frac{g \lg e}{\ln g} s^{-g^{k-1-\omega}} \end{aligned}$$

When evaluating the definite integral $[I]_0^\omega$ we get

$$\begin{aligned}
[I]_0^\omega &= -\frac{g \lg e}{\ln g}\left(s^{-g^{\omega-1-\omega}} - s^{-g^{0-1-\omega}}\right) \\
&= \frac{g \lg e}{\ln g}\left(s^{-g^{-1-\omega}} - s^{-g^{-1}}\right) \\
&\leq \frac{g \lg e}{\ln g} s^{-g^{-1-\omega}}
\end{aligned}$$

Theorem 2 says that $\omega \leq \left\lceil \log_g \log_{s_0} s \right\rceil$ and therefore $\omega \leq 1 + \log_g \log_{s_0} s$. Using this relation we can bound the integral:

$$\begin{aligned}
[I]_0^\omega &\leq \frac{g \lg e}{\ln g} s^{-g^{-1-(1+\log_g \log_{s_0} s)}} \\
&= \frac{g \lg e}{\ln g} s^{-g^{-2} g^{\log_g \log_s s_0}} \\
&= \frac{g \lg e}{\ln g} s^{-g^{-2} \log_s s_0} \\
&= \frac{g \lg e}{\ln g} s_0^{-g^{-2}} \\
&= constant
\end{aligned}$$

■

Theorem 5 (Constant Slowdown of the Emulation)

The slowdown S_E of the emulation of the mesh is of the order $O(1)$.

Proof: As shown in Lemma 3.4.6, the slowdown S_R of the recursive emulation (Equation 3.8) is of the order $O(1)$. The slowdown of the emulation in the

base case butterfly is also $O(1)$ because the base case mesh has a constant maximum size.

Since both S_R and S_B are bounded by a constant, it follows that $S_E = S_R + S_B = O(1)$. ■

Chapter 4

Simulation Results

In this chapter we will show the results of simulated embeddings of meshes in butterflies. The figures in this chapter are meant to visualize the behavior of the blowup as different parameters are varied. The slowdown depends much on how the emulation in the base case butterfly is performed and has not been investigated here. The following parameters can be varied:

Growth rate : The growth rate can be varied; in fact, one does not have to choose a single growth rate for an emulation. The analysis of the emulation only requires that the growth rate has a constant upper bound and a lower bound greater than 1.

Base case : The base case parameter configuration can be varied only subject to the constraint that the size of the base case butterfly and the load have a

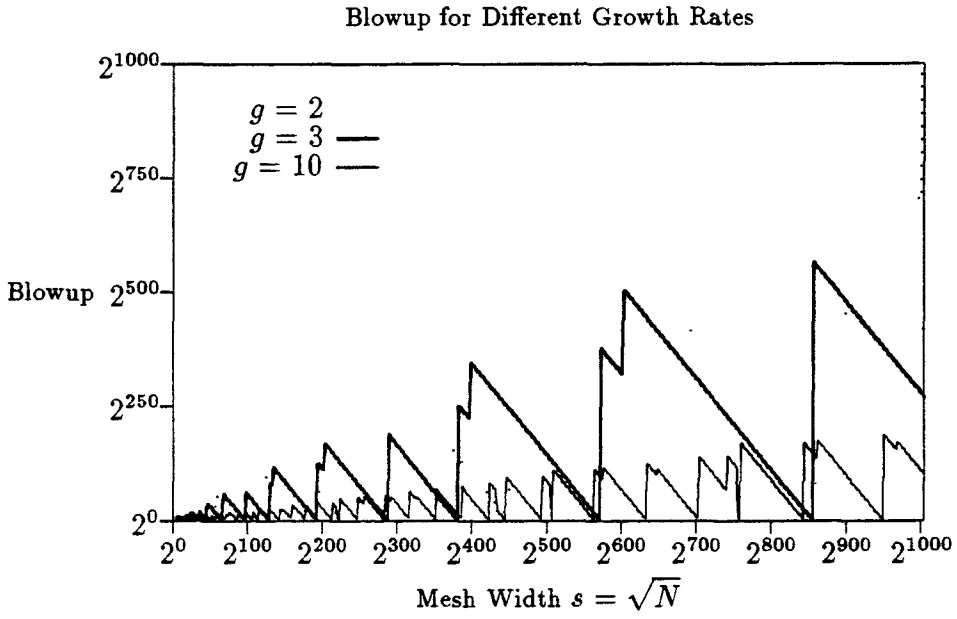

Figure 4.1: The ratio of butterfly nodes to mesh nodes for different growth rates.

constant upper bound. However, different from the variation of the growth rate, there are only a finite number of combinations to choose from, which therefore can not change the emulation characteristics except by a constant factor. For this reason, we did not simulate embeddings with different base case emulations.

4.1 The Behavior of the Emulation Blowup

As already shown in Section 3.3.4, the emulation does not have a constant blowup for meshes of *any* size. However, for each mesh there exists a larger mesh that can be emulated with constant blowup and therefore in a work-preserving manner.

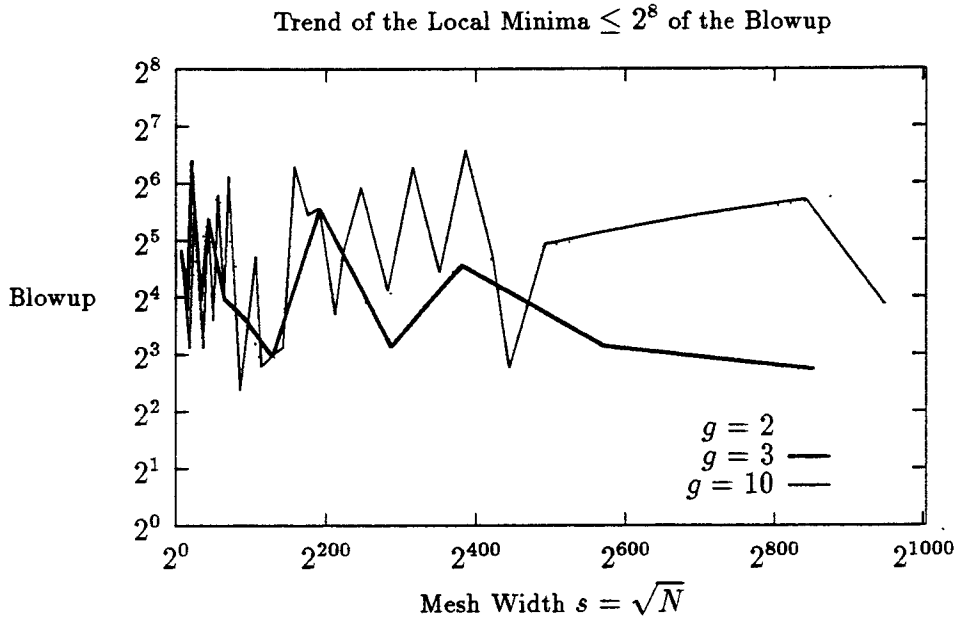

Figure 4.2: The minima of the blowup up to the value of 2^8 for different growth rates.

This behavior of the blowup is very obvious when looking at Figure 4.1[1] which plots the blowup against the mesh size. The mesh sizes that exhibit a minimum for the blowup are sizes of meshes that can be emulated using the corresponding growth rate with constant blowup and therefore in a work-preserving fashion. Figure 4.2 shows that the blowup values for these meshes are lower than 2^7 and sometimes even lower than $2^3 = 8$. Considering that the base case configuration used for the emulation does not even use all the butterfly nodes available for emulation of mesh nodes (the base case butterfly with $n_0 = 7$ and $\epsilon_0 = 2$ can accommodate a 18×18

[1]The data shown in Figures 4.1, 4.3 and 4.2 are based on an emulation with a base case parameter configuration of $n_0 = 7$, $s_0 = 15$, $f_0 = 3$ and $\epsilon_0 = 2$.

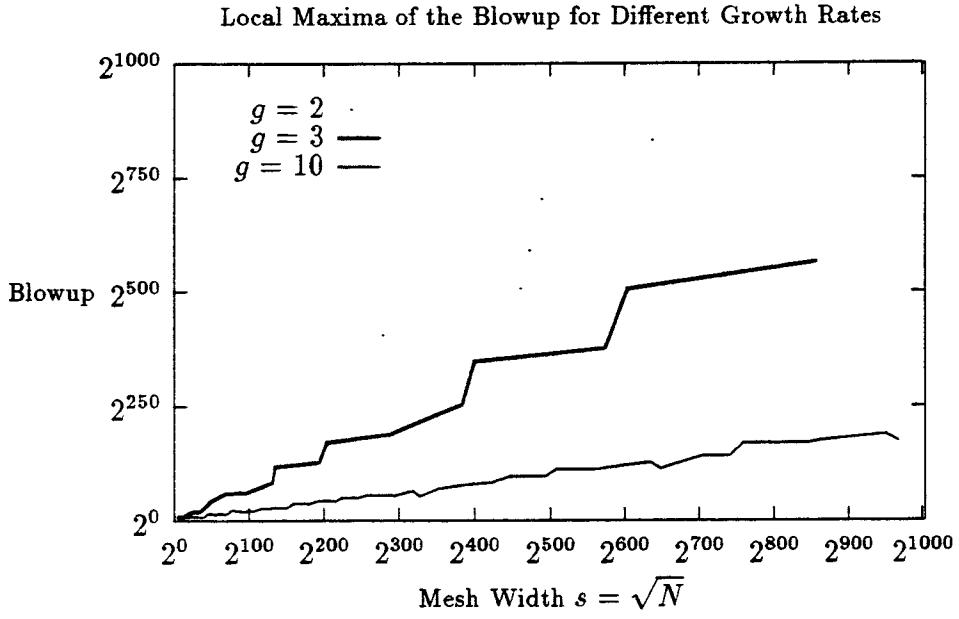

Figure 4.3: The worst case ratio of butterfly nodes to mesh nodes for different growth rates. The greater the growth rate, the smaller the peaks.

mesh with a load of 1) and that we could therefore improve the blowup by a factor of 0.7, these are satisfying results.

However, as shown in Section 3.3.4, only a few meshes sizes exhibit blowup minima and the maxima of the blowup are increasing towards infinity with the mesh size. Figure 4.3 shows the trend of the blowup peaks.

4.2 The Influence of the Growth Rate on the Emulation Blowup

As can be seen in Figure 4.1 and Figure 4.3, the growth rate g influences the blowup in the following ways:

1. The larger the growth rate, the smaller the worst-case blowup becomes.
2. As the growth rate increases, the distance between local minima becomes smaller.
3. Different growth rates exhibit blowup minima at different mesh sizes.

Observations 1 and 2 are not very useful to solve the problem of non-work-preserving emulations for most mesh sizes. Observation 3, however, leads to the question whether there is a growth rate for every mesh size such that the blowup of the embedding of the mesh in the butterfly is minimal. Our analysis of the emulation requires that the growth rate have a constant upper and lower bound. Therefore the following question arises:

> *Does a constant range of growth rates exist such that for any mesh size there is a growth rate in this range which allows a constant blowup and therefore a work-preserving emulation ?*

Figure 4.2 displays how the blowup values for emulations of certain meshes behave when the growth rate is varied. Though the overall behavior is not very conclusive,

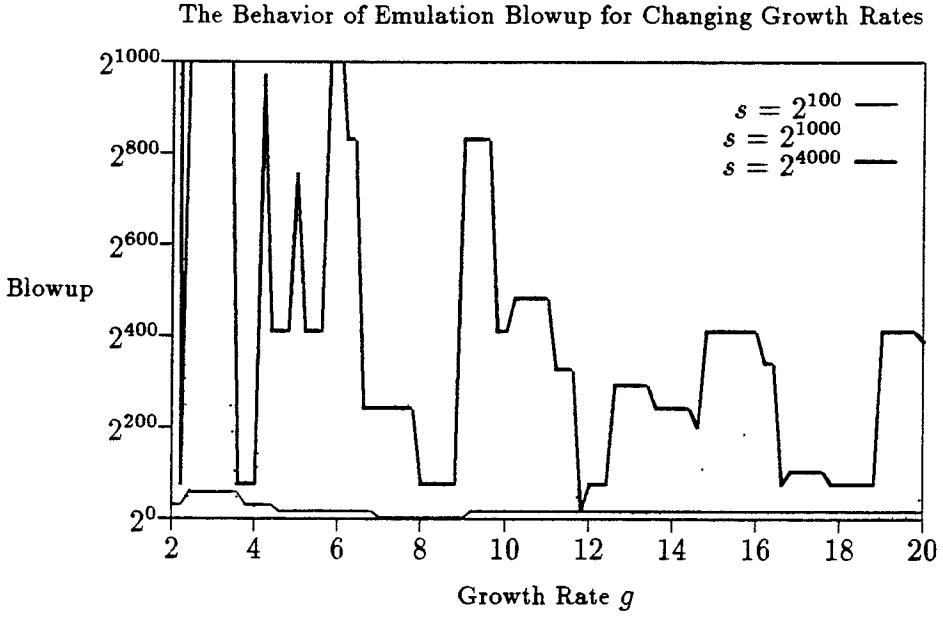

Figure 4.4: The change of the blowup of the emulation as the growth rate varies.

one can observe that the blowup has small minima for some growth rates.

To investigate these blowup minima, we show in Figure 4.2 the same data but only for small blowup values and we include more mesh sizes to increase the sample size. As one can see, for each of the investigated mesh sizes there is a growth rate in the range from 2 to 20 such that the blowup is at most 2^{20}. The quantity 2^{20} is certainly not an acceptable blowup value for practical purposes, but as the mesh sizes and therewith the number of stages in the emualation increases, one can expect that the growth rate becomes a better instrument for fine-tuning the size of the butterfly in the last stage (and therewith the blowup).

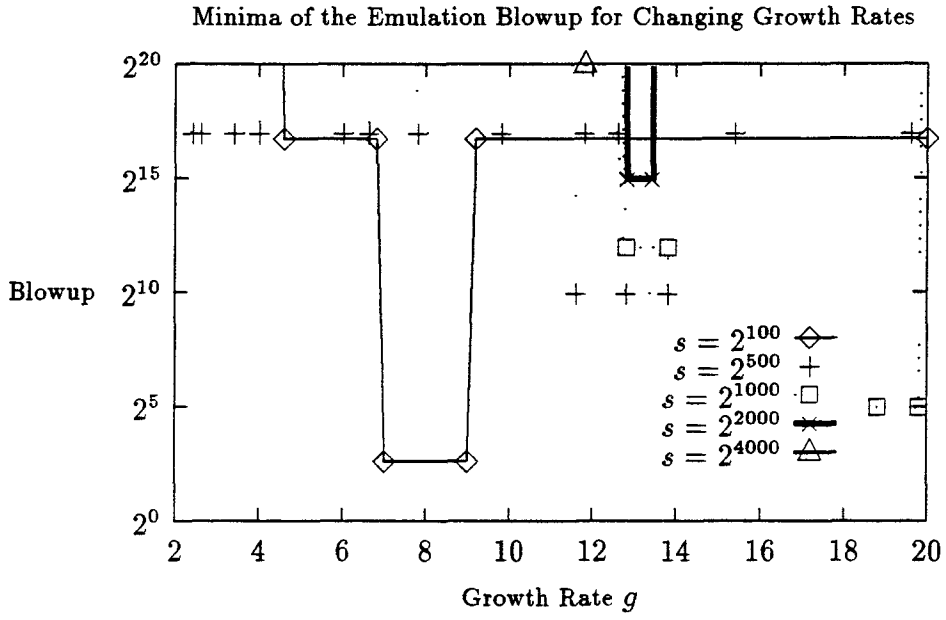

Figure 4.5: Blowup values $\leq 2^{20}$ as the growth rate changes. For each plotted mesh size exists a growth rate ≤ 20 such that the blowup is $\leq 2^{20}$.

The aforementioned question is very difficult to tackle analytically and could not be answered in this thesis. A more generalized version of this problem is presented in Section 5.1 as an open problem left for future research.

Chapter 5

Conclusion

5.1 Future Work

The work presented in this thesis does not answer the question whether a work-preserving emulation exists for *all* mesh sizes. Furthermore, the emulation can be generalized to include the emulation of higher-dimensional meshes. These issues are addressed here and should be investigated further.

5.1.1 Is There a Work-Preserving Emulation for Every Mesh?

In this thesis we did not show that there is a work-preserving emulation for *any* mesh size, but that there is a infinite sequence of meshes which can be emulated

by a butterfly with a constant blowup and constant slowdown.

Naturally, the question is now whether this is a consequence of our choice of the emulation parameters or if the recursive embedding strategy does not allow for the work-preserving emulation of any mesh (as was the case in [13, 14]).

Note, that the growth rate g does not have to be constant for all stages. One could chose a different growth rate g_k for each stage where $1 < g_{min} \leq g_k \leq g_{max}$, g_{min} and g_{max} constant. The analysis of the blowup and slowdown would not change except for a constant factor. The idea behind making the growth rate variable within certain bounds is that we want to increase the number of butterflies that can emulate a given mesh with constant blowup. Assuming a given base case, the emulation is completely charaterized by the sequence of the parameter n_k through the recursive stages, since all the other parameters can be derived from n_k and the parameters of the stage $k-1$. Let the vector

$$(n_0, n_1, n_2, \ldots, n_{\omega-1}, n_\omega)$$

be that sequence.

The variability of the growth rate would allow for more sequences of butterflies and makes the set of possible butterflies denser. The problem can be reduced to the question whether the increase of possible butterflies is enough to guarantee that there will always be a butterfly whose number of nodes $(n_\omega 2^{n_\omega - 1})$ is within a con-

stant factor of the mesh size such that a sequence of butterflies can be constructed that satisfies the constraints of the embedding.

Thus the problem of the existence of a truly work-preserving emulation of the class of meshes on the class of butterflies can be reformulated in the following way:

Problem: *Determine whether constants A, A_1, A_2 exist such that for all N there is a butterfly of size $n_\omega 2^{n_\omega - 1}$ such that there exists a sequence*

$$(n_0, n_1, n_2, \ldots, n_{\omega-1}, n_\omega)$$

such that the following conditions are satisfied:

1. *n_k is an integral multiple of n_{k-1} for all k*
2. $1 < A_1 \leq \log_{s_{k-1}} s_k \leq A_2$
3. $n_\omega 2^{n_\omega - 1} \leq A \cdot N$.

Even if the answer to this problem is positive, there is still a non-trivial optimization problem to be solved when trying to construct such a sequence for a given mesh.

5.1.2 Extending the Emulation to Related Networks

In this thesis we describe the embedding of a two-dimensional mesh into a butterfly. It is not very difficult to modify the embedding to include meshes of higher

dimensions. Furthermore one could change the target network to a close derivative of the butterfly, the multi-butterfly.

5.2 Conclusion

In this thesis we have presented an embedding of meshes in butterflies that allows a work-preserving emulation. Based on the work in [13, 14] we developed a recursive embedding of a $s \times s$ mesh in a butterfly, while correcting errors made in previous attempts and removing unnecessary constraints that prevented a work-preserving emulation. The recursive embedding consists of $O(\log \log s)$ stages in which a mesh is partitioned into overlapping submeshes which are emulated by subbutterflies. We studied the constraints imposed on the base case of the recursion and developed an algorithm which determines the parameter for each stage of the recursion.

In a rigorous analysis of the emulation we have shown that the blowup of the emulation is $O(1)$ for an infinite number of meshes and that the slowdown is also constant. Therefore we could conclude that the emulation is performed in a work-preserving real-time manner for an infinite number of meshes.

The introduction of the parameter g, the growth rate, which controls the growth of the meshes from one stage to another, proved to be an important step towards a generalization of the embedding and contributed to the emergence of a possible removal of the restriction on the work-preserving quality of the emulation.

We could show that for every mesh there exists a larger mesh that can be emulated with constant blowup. However since there are an infinite number of meshes for which we could not demonstrate a work-preserving emulation our results fall short of presenting a work-preserving emulation of the class of meshes on the class of butterflies.

This problem might have a possible solution, as the data obtained from simulations suggests, which would rely on a variation of the growth rate parameter introduced in this work. The analysis of the influence of a variable growth rate remains an interesting and demanding problem for future research.

Bibliography

[1] BERMAN, F., AND SNYDER, L. On mapping parallel algorithms into parallel architectures. *Journal of Parallel and Distributed Computing 4* (1987), 493–458.

[2] BHATT, S. N., CHUNG, F. R. K., HONG, J.-W., LEIGHTON, F., AND ROSENBERG, A. L. Optimal simulations by butterfly networks. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing* (May 1988), pp. 192–204.

[3] BHATT, S. N., AND IPSEN, I. C. F. How to embed trees in hypercubes. Tech. Rep. YALE/DCS/RR-43, Department of Computer Science, Yale University, 1985.

[4] BOKHARI, S. H. On the mapping problem. *IEEE Transactions on Computers C-30*, 3 (Mar. 1981), 207–214.

[5] CHAN, M. Y. Dilation-2 embeddings of grids into hypercubes. In *Proc. 1988 International Conference on Parallel Processing* (University Park, PA, 1988),

The Pennsylvania State University Press.

[6] CHAN, M. Y. The embedding of grids into optimal hypercubes. Tech. Rep. SPAA 89, Computer Science Department, University of Texas at Dallas, 1988.

[7] CHAN, M. Y. Embeddings of 3-dimensional grids into optimal hypercubes. In *Proc. Fourth Conference on Hypercubes, Concurrent Computers and Applications* (Mar. 1989).

[8] DESPHANDE, S. R., AND JENEVEIN, R. Scalability of binary trees on a hypercube. In *Proc. 1986 International Conference on Parallel Processing* (Silver Spring, MD, 1986), IEEE Computer Society, pp. 661–668.

[9] EFE, K. Embedding mesh of trees in the hypercube. *Journal of Parallel and Distributed Computing 11* (1991), 222–230.

[10] FEITELSON. *Optical Computing: A Survey for Computing Scientists.* MIT Press, 1980.

[11] FISHBURN, J. P., AND FINKEL, R. A. Quotient networks. *IEEE Transactions on Computers C-31*, 4 (Apr. 1982), 288–295.

[12] HO, C.-T., AND JOHNSSON, S. Embedding meshes in boolean cubes by graph decomposition. *Journal of Parallel and Distributed Computing 8* (1990), 325–339.

[13] KOCH, R. R., LEIGHTON, F. T., MAGGS, B., RAO, S. B., AND ROSENBERG, A. L. Work-preserving emulations of fixed-connection networks. In *Proceedings of the 21st Symposium on Theory of Computation* (May 1989), pp. 227–240. Extended abstract. For longer version see [14].

[14] KOCH, R. R., LEIGHTON, F. T., MAGGS, B., RAO, S. B., ROSENBERG, A. L., AND SCHWABE, E. J. Work-preserving emulations of fixed-connection networks. To appear in the Journal of the ACM. This is an improved version of [13].

[15] LEIGHTON, T., AND LEISERSON, C. Advanced Parallel and VLSI Computation. Research Seminar Series MIT/LCS/RSS 7, Massachusetts Institute of Technology, Boston, MA, Dec. 1989. Lecture notes of a course taught at the Massachusetts Institute of Technology.

[16] LEIGHTON, T., MAGGS, B., AND RAO, S. Universal packet routing algorithms. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science* (Oct. 1988), IEEE, pp. 256–271.

[17] MEAD, AND CONWAY. *Introduction to VLSI Systems.* Addison-Wesley, 1980.

[18] MEYER AUF DER HEIDE, F. Efficient simulations among several models of parallel computers. *SIAM Journal on Computing 15*, 1 (Feb. 1986), 106–119.

[19] MURDOCCA. *A Digital Design Methodology for Optical Computing.* MIT Press, 1990.

[20] PARKER, B. A VLSI comparison between the mesh and the butterfly. Work in progress.

[21] ULLMAN, J. *Computational Aspects of VLSI.* Computer Science Press, 1984.

Appendix A

Table of Used Symbols

Following is a table for reference purposes that lists all the symbols used in this thesis with their meanings, possible constraints on their values and a reference to a page where they were defined or discussed.

Symbol	Meaning	Constraints	Page
α	integer value characterizing in which levels of the butterfly a subbutterfly resides. All nodes of a butterfly characterized by α are in levels α to $\alpha + n_{k-1} - 1$ of the superbutterfly	$\alpha \bmod n_{k-1} = 0$ $0 \leq \alpha \leq \frac{n_k}{n_{k-1}} - 1$	15
c	Congestion. This is the maximum number of mesh nodes that send or receive their pebbles via the same I/O port.	$1 \leq c \leq 2s_0$	
D_k	dilation at stage k.		66
ϵ_k	defines the number of I/O ports relative to n_k at a given stage	$\epsilon_0 \geq 1$, $\epsilon_k \geq \epsilon_{k-1} + 1$	39
f_k	width of the overlapping region at stage k	$2f_k \leq s_k$	
F_k	the set of nodes on the border or at distance f_k from the border of the mesh at stage k		
g	*growth rate* of the mesh sizes from stage to stage.	$g > 1$	37
k	throughout the text k is used as an index in the range $[0, \omega(N)]$ to identify a stage of the recursive emulation, where $k = 0$ denotes the base case stage	$0 \leq k \leq \omega$	

(Continued on next page ...)

Symbol	Meaning	Constraints	Page
K	number of stages after which $n_k \leq (\lceil g \rceil + 1)n_{k-1}$	$K \geq 0$	48
n_k	number of levels of the butterfly at stage k	$n_0 \geq 5$, $n_k \bmod n_{k-1} = 0$	38
N_k	The number of nodes in the butterfly at stage k	$N_k = n_k 2^{n_k - 1}$	
N	The number of nodes in the square mesh to be emulated	$N = s^2$	
s	the size of one dimension of the mesh to be emulated		
s_k	number of columns (or rows) of the square mesh emulated at stage k		38
σ	a constant that is the upper bound on $\log_{s'_k} s_k$		
S_E	the slowdown of the emulation		
S_R	the slowdown of the emulation due to the recursive structure		
S_B	the slowdown of the emulation due to the base case		