An Improved Mixture of Experts Approach for Model Partitioning in VLSI–Design Using Genetic Algorithms

K. Hering[†], R. Haupt[‡], Th. Villmann[*]

Institut für Informatik
Universität Leipzig, Augustusplatz 10-11
04109 Leipzig

Universität Leipzig / Institut für Informatik
Report Nr. 14 (1995)
ftp.uni-leipzig.de – directory: /pub/ifi/reports

[†] *email: khering@informatik.uni-leipzig.de*
[‡] *email: rhaupt@informatik.uni-leipzig.de*
[*] *email: villmann@informatik.uni-leipzig.d400.de*

Abstract

The partitioning of complex processor models on the gate and register-transfer level for parallel functional simulation based on the clock-cycle algorithm is considered. We introduce a hierarchical partitioning scheme combining various partitioning algorithms in the frame of a competing strategy. Melting together the different partitioning results within one level using superpositions we crossover to a mixture of experts one. This approach is improved applying genetic algorithms. We present two new partitioning algorithms (experts), the *Backward-Cone-Concentration* algorithm (n-BCC) and the *Minimum-Overlap Cone-Cluster* algorithm (MOCC), both of them taking cones as fundamental units for building partitions.

1 Introduction

Logic design for whole microprocessor structures is accompanied with time-extensive simulation processes. Within the design strategy outlined in [Spr89] the verification of functional (logical) behavior is strictly separated from the analysis of timing aspects. In this context the background of the present paper is given by simulation processes for functional design verification on gate and register-transfer level (logic simulation) where sequences of machine instructions or microcode are taken as test cases and underlying models comprise complex parts of processor structures. Under this assumptions the usage of cycle-based simulators is to be preferred, because they are significantly faster than event-driven simulators in this field of application. Nevertheless, the ratio between simulation runtime and simulated CPU time can reach values between 10^5 and 10^7.

TEXSIM[1] is a simulator of high performance for logic simulation of synchronous designs using the *clock-cycle algorithm* running on RS/6000 processors. Within a *TEXSIM* simulation model all elements representing combinational logic or global outputs, are statically rank ordered. These ranks are preceded by a special rank containing elements standing for clocked latches, registers and global inputs. Roughly spoken, the simulation of one cycle comprises the evaluation of all logical elements from lower to higher ranks followed by the propagation of values to the outputs of latches or registers. To achieve a significant reduction of running time for simulations we are working on the parallelization of *TEXSIM* for IBM's loosely coupled RS/6000 processor system *SP2*. Thereby a parallel *TEXSIM* simulation consists of several co-operating *TEXSIM* instances running on different processors over parts of the whole model. As a basic assumption, the process of the evaluation of combinational logic during the simulation of a cycle has been left unchanged for the parallel *TEXSIM* version. This directly influences model partitioning. Assigning a logical element to a model part requires the same assignment for all logical elements of the whole model which are able to contribute to a change of an input value of the considered element during one cycle. If special *fan-in cones* [SUM87, Man92, MTSDA93] are taken as basic building blocks for model partitioning, the demand mentioned above is fulfilled. A model partition is directly related to certain workloads of the processors involved in later parallel simulation and communication overhead between co-operating *TEXSIM* instances. In this way model partitioning has an essential impact on the speed-up possible due to parallelization. The amount of time acceptable for partitioning depends on the expected total duration of all simulation runs to be performed regarding to a

[1] copyright by IBM

corresponding model. Simulation processes we are dealing with are characterized by a large number of time-extensive runs concerning a given model. We investigate models of microprocessor structures the complexitivity of which will reach several million gate level elements in the near future.

In this paper, we introduce a *hierarchical partitioning strategy* for handling complex models. Starting with a formal *structural hardware model*, basic definitions concerning *cones* and *partitions* are given in section 2. In section 3 a *q-level partitioning scheme* is presented which allows to combine different partitioning algorithms at succeeding hierarchy levels. To overcome the lack of *a priori* knowledge in choosing algorithms suited for models at hand, within each hierarchy level the competition and mixture of different algorithms according to a so called *mixture of experts strategy* [JJ94] is proposed. In this framework we define *superpositions of partitions* to get the possibility of melting together results of various partitioning algorithms within one and the same hierarchy level. In a final step, an improved mixture of experts strategy using *genetic algorithms* is developed. Two new partitioning algorithms aiming at the generation of partitions with balanced workload are presented. First experimental results are given in section 4.

2 Definitions

In the following we define a structural model for the logic design on gate and register-transfer level. The underlying hardware is synchronous, i.e., asynchronous combinational feedback is not allowed. Basic components of the model are given by the sets listed below (Fig. 1):

- M_I ... global inputs,
- M_O ... global outputs,
- M_E ... logical elements,
- M_L ... storing elements,
- M_S ... signals.

M_E includes all elements which represent combinational logic within the hardware we want to simulate. Signals of M_S are to be interpreted as wires. The elements of the

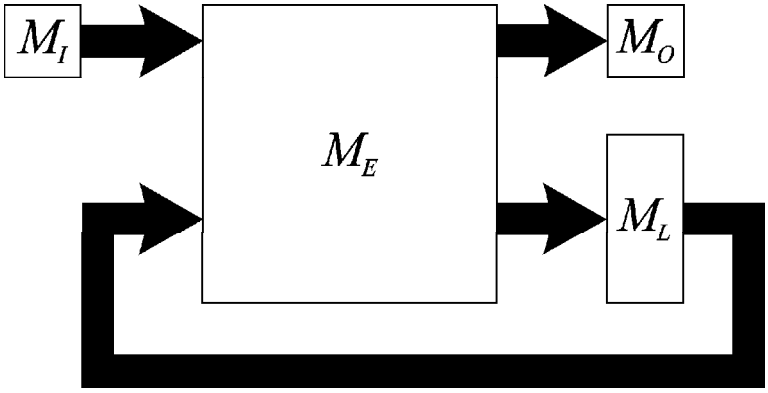

Figure 1: hardware model – basic structure

set M_L possess storing function and are cycle limiting in the sense of the *clock-cycle* algorithm. We concentrate the elements of all pairwise disjoint sets M_I, M_O, M_E and M_L (Fig. 1) to the set of boxes $M_B = M_I \cup M_O \cup M_E \cup M_L$. On the basis of these sets the hardware model can be taken as a directed bipartite graph. Therefore, we introduce the relation $M_{\mathcal{R}} \subseteq (M_B \times M_S) \cup (M_S \times M_B)$ describing the connections between boxes and signals. For any directed graph $G = (X, \mathcal{R})$ and $x \in X$ we have the sets of successors $\mathcal{N}_G^+(x) = \{y|(x,y) \in \mathcal{R}\}$ and predecessors $\mathcal{N}_G^-(x) = \{y|(y,x) \in \mathcal{R}\}$. Using these notations we are able to define the hardware model as:

Definition 1 *Let M_I, M_O, M_E, M_L, and M_S be pairwise disjoint and nonempty sets. M_B and $M_{\mathcal{R}}$ are defined as above. $M = (M_I, M_O, M_E, M_L, M_S, M_{\mathcal{R}})$ is called* ***hardware model*** *if the corresponding directed bipartite graph $G(M) = (M_B, M_S, M_{\mathcal{R}})$ [Len90, Spo95] satisfies the following conditions:*

1. $\left\{x|x \in M_B \cup M_S \wedge \mathcal{N}_{G(M)}^-(x) = \emptyset\right\} = M_I$,
2. $\left\{x|x \in M_B \cup M_S \wedge \mathcal{N}_{G(M)}^+(x) = \emptyset\right\} = M_O$,
3. *any directed cycle in $G(M)$ includes at least one element of M_L.*

M_I and M_O are the sets of all sources and sinks of $G(M)$, respectively. Condition 3 ensures the absence of directed cycles only including elements of $M_E \cup M_S$. This corresponds to the exclusion of asynchronous combinational feedbacks within the underlying hardware.

Due to our parallelization approach, cutting signals of M_S during a partitioning of M is only permitted at cycle-boundaries related to the clock-cycle algorithm mentioned above. Therefore, we are forced to define basic units for partitioning which are known as *cones* [SUM87, Man92, MTSDA93]. These units comprise elements of M_B with a limiting head element out of $M_O \cup M_L$. For further investigations of the partitioning problem we fix the following formal definitions with respect to an arbitrarily chosen hardware model M:

Definition 2 *The **fan-in cone** $co_I(x)$ of an element $x \in M_O \cup M_E \cup M_L$ is recursively defined by:*

1. $x \in co_I(x)$,
2. $y \in M_E \wedge \mathcal{N}^+_{G(M)}(\mathcal{N}^+_{G(M)}(y)) \cap co_I(x) \neq \emptyset \rightarrow y \in co_I(x)$.

Definition 3 *The **fan-out cone** $co_O(x)$ of an element $x \in M_I \cup M_E \cup M_L$ is recursively defined by:*

1. $x \in co_O(x)$,
2. $y \in M_E \wedge \mathcal{N}^-_{G(M)}(\mathcal{N}^-_{G(M)}(y)) \cap co_O(x) \neq \emptyset \rightarrow y \in co_O(x)$.

Now, let us take a *cone* $co(x)$ as a special fan-in cone $co_I(x)$ if $x \in M_O \cup M_L$. All the cones form the set $Co\,(M)$ as the set of basic units for the partitioning of M. An example of cones for the model shown in Fig. 1 is depicted in Fig. 2. The head element x of $co(x)$ is an element of the set $M_O \cup M_L$. The number of cones belonging to $Co(M)$ is $m_c = |Co(M)| = |M_L| + |M_O|$. For relevant models the ratio of the number of cones m_c to the number of boxes m_B is in the range of $m_c : m_B \approx 1 : 10$. A box $b \in M_E$ (logical element), from which directed paths (with all intermediate boxes being elements of M_E) to the heads of different cones $\hat{c}_i \in Co(M)$, $i = 1, \ldots, m$ exist, belongs to all of the m different cones $\hat{c}_i$: $b \in \bigcap_{i=1}^{m} \hat{c}_i$. These cones $\hat{c}_i$ are called to be *overlapping*. Considering the whole model, i.e. all cones c_i, we get:

$$\sum_{i=1}^{m_c} |c_i| \geq \left| \bigcup_{i=1}^{m_c} c_i \right| = |M_L| + |M_O| + |M_E| \ . \tag{2.1}$$

If we distribute overlapping cones to different processors we have to take into account the multiple evaluation of boxes in parallel simulations. On the other hand, communication between the processors is reduced to communication at the clock-cycle boundaries.

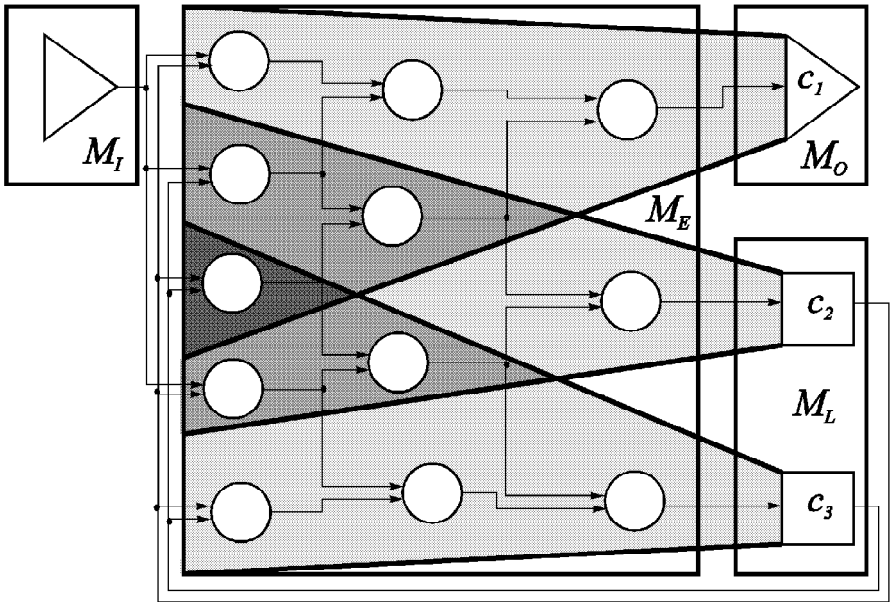

Figure 2: hardware model with cones (shaded)

Of course, replication in the evaluation of boxes is an additional restricting factor and has to be minimized. In the following, $\mathcal{C}$ always denotes nonempty subsets of $Co(M)$.

Definition 4 *The* ***box-related cone overlap degree*** $u : M_E \rightarrow \mathbb{N}$ *is defined by* $u(b) = |\{c \,|c \in Co(M) \wedge b \in c\}|$, *giving the number of cones which contain the box* $b \in M_E$.

Definition 5 *The* ***overlap region*** $ovr(\mathcal{C})$ *of a set of cones* $\mathcal{C}$ *is the set of boxes belonging to these and only these cones* $c \in \mathcal{C}$*:*

$$ovr(\mathcal{C}) = \left(\bigcap_{c \in \mathcal{C}} c\right) \setminus \left(\bigcup_{c' \in Co(M) \setminus \mathcal{C}} c'\right) . \tag{2.2}$$

Then, we can state:

1. All elements of the set $M_E \cup M_L \cup M_O$ are uniquely and disjoint distributable into overlap regions $ovr(\mathcal{C})$. Using $P^* = 2^{Co(M)} \setminus \{\emptyset\}$ we get:

$$|M_E| + |M_L| + |M_O| = \sum_{\mathcal{C} \in P^*} |ovr(\mathcal{C})| . \tag{2.3}$$

   In general, most of the overlap regions $ovr(\mathcal{C})$ are empty sets.

2. The *overlap degree* of a nonempty overlap region $ovr(\mathcal{C})$ is denoted by $u_{\mathcal{C}}$ for which

$$u_{\mathcal{C}} = |\mathcal{C}| \tag{2.4}$$

is valid.

3. The set of boxes of a cone c is uniquely decomposable into overlap regions $ovr(\mathcal{C})$, whereby $c \in \mathcal{C}$:

$$c = \bigcup_{(\mathcal{C} \in P^* \wedge c \in \mathcal{C})} ovr(\mathcal{C}) \text{ with } |c| = \sum_{(\mathcal{C} \in P^* \wedge c \in \mathcal{C})} |ovr(\mathcal{C})| \,. \tag{2.5}$$

The overlap structure allows a construction of a hypergraph identifying the nodes with cones and the hyperedges with cone sets $\mathcal{C}$ corresponding to nonempty overlap regions $ovr(\mathcal{C})$ with $u_{\mathcal{C}} > 1$.

Definition 6 *The* ***overlap hypergraph*** $GU = (V, E)$ *consists of a finite set of nodes* $V = Co(M)$ *and a set of hyperedges* $E \subseteq P^*$ *with* $\mathcal{C} \in E$ *if and only if* $u_{\mathcal{C}} > 1$ *and* $|ovr(\mathcal{C})| > 0$.

To map the whole structure of the overlap regions onto the overlap hypergraph we introduce weights for the nodes $\nu : V \rightarrow I\!N$ and for the hyperedges $\mu : E \rightarrow I\!N$. The weight ν for a node $c_i \in V$ is the number of boxes which are contained in the cone c_i, exclusively: $\nu_i = \nu(c_i) = |ovr(\{c_i\})|$. The weight μ for a hyperedge $\mathcal{C}_j \in E$ is the number of boxes which are contained in all cones $c_l \in \mathcal{C}_j$, exclusively: $\mu_j = \mu(\mathcal{C}_j) = |ovr(\mathcal{C}_j)|$. An overlap hypergraph $GU = (V, E)$ together with the weights ν and μ we call *weighted*.

There are other possibilities to define weighted overlap hypergraphs in a similar manner:

1. $GU' \equiv GU$ with a changed weighting function for the nodes $\nu'_i = \nu'(c_i) = |c_i|$ and $\mu' = \mu$.

2. MANJIKIAN introduced a *directed* hypergraph where each hyperedge gets exactly one head [Man92]. Whereas the node-weighting function ν is changed to ν'' the hyperedge-weighting function remains unchanged. To each weight ν_i the weight of all hyperedges whose heads are identical to c_i is added yielding ν''_i.

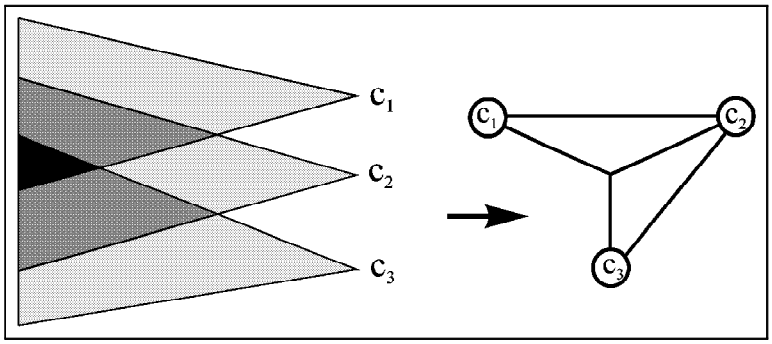

Figure 3: overlap hypergraph GU according Fig. 2

In Fig. 3 an example of an overlap hypergraph GU according to Def. 6 is represented coincident with Fig. 2.

In the following we want to introduce the terms – partitioning and partition – by means of two nonempty sets U and V.

Definition 7 *A **partitioning** of U related to V is a unique map $\Phi : U \rightarrow V$ assigning each element $u \in U$ to an element $v \in V$.*

Definition 8 *A **partition** Ψ_Φ of U related to the partitioning $\Phi : U \rightarrow V$ is given by $\Psi_\Phi = \{\Phi^{-1}(v) \mid v \in \text{cod } \Phi\}$, where* cod Φ *is the range of Φ.*

An element $v \in$ cod Φ represents the partition component $\Phi^{-1}(v)$ containing all elements $u \in U$ which are mapped onto v. Here, we identify U with the set of cones $Co(M)$ and V with the set of m_b blocks $\mathcal{B}$ representing processors.

Remark 1 *If the partitioning $\Phi : U \rightarrow V$ is not surjective, elements $v \in V$ exist which do not represent partition components. If it is not specified otherwise we consider surjective partitionings.*

For estimating the running time of parallel simulations several evaluation models are possible. Our approach bases on a unique time unit $\tau = 1$ for the evaluation of each box.[2] Under this assumption the *workload* W_i of a block B_i may be specified as

[2]Other assignments of evaluation times to special logical boxes are possible.

the sum of all boxes which are to be evaluated at the corresponding processor. With $\Phi^{-1}(B_i) = \left\{c_1^i, c_2^i, \ldots, c_{m_i}^i\right\}$ we have

$$W_i = \left| \bigcup_{j=1\ldots m_i} c_j^i \right|, \tag{2.6}$$

where each $c_j^i \in Co(M)$, which can be also expressed in terms of overlap regions:

$$W_i = \sum_{(\mathcal{C} \in P^* \wedge \mathcal{C} \cap \Phi^{-1}(B_i) \neq \emptyset)} |ovr(\mathcal{C})| . \tag{2.7}$$

For sequential simulation we have the sequential workload

$$W_{seq} = \sum_{\mathcal{C} \in P^*} |ovr(\mathcal{C})| = |M_E| + |M_L| + |M_O| \tag{2.8}$$

which is equal to the sum of boxes to be evaluated.

3 Hierarchical Partitioning

3.1 Hierarchical Strategy and the Mixture of Experts Approach

Using the general Def. 7 of a partitioning problem we can describe the special case of mapping cones $c \in Co(M)$ onto a set $\mathcal{B}$ of blocks B_i (processors) by

$$\Phi : Co(M) \rightarrow \mathcal{B} \tag{3.1}$$

with $|\mathcal{B}| = m_b$ as the number of all blocks. The task is to find a partitioning which leads to a minimal running time for parallel simulation. To achieve this goal we have to consider quality functions taking into account several aspects such as interprocessor communication, workload balance and replication rate. For a certain quality function Ω one has to determine a partitioning $\Phi_{\text{opt}}^{\Omega}$ which optimizes Ω. Yet, such an optimization problem in general is a *NP*–complete one. Moreover, in the applications considered here the ratio between m_c as the number of cones and m_b may be up to the range of $10^5 - 10^6$. Therefore, we focus onto a hierarchical strategy which has been successfully applied to data extensive problems as color image processing [VMB+94], satellite remote sensoring [GS93, Vil95], robotics [RMS92] and time series prediction in high–dimensional data spaces [DEFH93, KMP95]. To gradually reduce the range of the problem we introduce a general q–level partitioning scheme according to Def. 7 by:

Definition 9 *A **q-level partitioning** of U with respect to V is defined by $\Phi_H : U \to V$ with*

$$\Phi_H = \Phi_q \circ \Phi_{q-1} \circ \ldots \circ \Phi_1 \tag{3.2}$$

where

$$\Phi_j : V_j \to V_{j+1} \tag{3.3}$$

and $V_1 = U$, $V_{q+1} = V$ and furthermore $|V_1| \geq |V_2| \geq \ldots \geq |V_{q+1}|$.

Clearly, in general Φ_H is only an approximation of $\Phi^{\Omega}_{\text{opt}}$. In dependence on the problem the number q of hierarchy levels has to be chosen. Up to now, in our application we use a 2–level scheme $\Phi_H = g \circ f$, i.e. $V_1 = U = Co\,(M)$, $V_{q+1} = V = \mathcal{B}$ and $V_2 = \mathcal{S}$:

$$\Phi_H : Co\,(M) \xrightarrow{f} \mathcal{S} \xrightarrow{g} \mathcal{B}\ . \tag{3.4}$$

Thereby, $\mathcal{S}$ is a set of elements S_l, the pre–images $s_l = f^{-1}\,(S_l)$ of which are called *super–cones*. We remark that super–cones are collections of usual cones.[3] In contrast to the determination of the cones the realizations of g and f are free. This allows an optimal adaptation. However, often an *a priori* optimal choice is not possible [Len90]. To overcome this difficulty we prefer in each level of the hierarchical scheme a strategy introduced in neurodynamics by JORDAN et al. [JJ94] which is called *mixture of experts*.

For a q–level scheme we consider several partitioning algorithms A^j_i, $i = 1 \ldots m_j$ corresponding to maps Φ^j_i and working in a parallel way in one hierarchical step j representing various partitioning heuristics. The resulting partitions $\Psi_{A^j_i}$ are compared with respect to a quality measure[4] and the β_j best of them will form the basis for the algorithms A^{j+1}_i of the next level which generate partitions $\Psi_{A^j_i, A^{j+1}_l}$. Thereby, the images of the super–cones of a partition $\Psi_{A^j_i}$ given by Φ^j_i are taken as the new minimal units. The final result of a q–level scheme is a partition $\Psi_{A^1_{i_1}, A^2_{i_2}, \ldots, A^q_{i_q}}$ the quality measure of which is the best in the last level. However, as yet this is only a simple strategy of competing experts.

By introducing the superposition Ψ^* of partitions Ψ_i of a certain level, which plays the role of a generating system for these partitions, we next extend the competing

[3] On the other hand, the cones themselves are sets of boxes which are elements of $\mathcal{M} = M_E \cup M_L \cup M_O$. Therefore, we can take the concentration of these as an initial 'partitioning' $\Phi_0 : \mathcal{M} \to Co\,(M)$. In this way we obtain $\Phi^*_H = g \circ f \circ \Phi_0$ as an extended 2–level scheme. The definition of the cones uniquely determines the map Φ_0. Yet, Φ_0 is not a partitioning in the sense of Def. 7.

[4] To obtain a quality measure which is applicable to each of these partitions $\Psi_{A^j_i}$ one has to define it on the basis of the set U from Def. 9, i.e., in our special case on the set of cones $Co\,(M)$.

approach to a mixture one. Then each super–cone of a partition Ψ_i is expressible in terms of $s_l^* \in \Psi^*$. To explain such a generating system we give the following definition:

Definition 10 *Let $\Pi = \{\Psi_1, \ldots, \Psi_k\}$ be a system of partitions of the set U. The elements of Ψ_i are denoted by s_i^j, $j = 1 \ldots n_i$. A set $\Psi^* = \{s_1^*, \ldots, s_m^*\}$ is called a* ***superposition*** *of Π if the following conditions are satisfied:*

1. *Ψ^* is a partition of U*
2. *Ψ^* is a generating system for each $\Psi_i \in \Pi$, i.e., for each $s_i^j \in \Psi_i$ ($i = 1 \ldots k$, $j = 1 \ldots n_i$) exist $s_{l_1}^*, \ldots, s_{l_r}^* \in \Psi^*$ such that $s_i^j = s_{l_1}^* \cup \ldots \cup s_{l_r}^*$.*

With Defs. 7 and 8 it follows that $\emptyset \notin \Psi^*$. Of course, the elements of U taken as single sets form a superposition U^* of Π. However, we want to have a superposition the granularity of which is rougher than the granularity of U^*, i.e., $|U^*| > |\Psi^*|$. Now we consider a special construction of superpositions:

Theorem 1 *Let $\Pi = \{\Psi_1, \ldots, \Psi_k\}$ be a system of partitions of the set U. The elements of Ψ_i are denoted by s_i^j, $j = 1 \ldots n_i$. Furthermore, let Ψ^* be given as*

$$\Psi^* = \left\{ s_{j_1 \ldots j_k}^* \mid s_{j_1 \ldots j_k}^* = \bigcap_{i=1 \ldots k} s_i^{j_i} \right\} \setminus \{\emptyset\} \tag{3.5}$$

with $j_i = 1 \ldots n_i$. Then Ψ^* is a superposition of Π.

The proof is shown in Appendix A.

Following the theorem, in most applications we are able to determine a superposition Ψ^* of the partitions $\Psi_1, \ldots, \Psi_k$ with $|U^*| > |\Psi^*|$, i.e. with a rougher granularity than this of U^*, by k–times intersections according to (3.5). Of course, in general we only have $|U^*| \geq |\Psi^*|$. The construction of the elements of Ψ^* depends on the structure of the partitions of the $\Psi_i \in \Pi$ which represent the different partitioning heuristics (realized by the corresponding algorithms). Hence, all used strategies influence the superposition, i.e., the expert knowledge of the algorithms is mixed in Ψ^*. We add a superposition according to Theorem 1 as a special partition to the β_j best of one hierarchical level j so that it may be used in the next level, too.

Returning to our 2–level scheme, the usage of a superposition is only useful after the first partitioning level. If we assume that we have various algorithms A_i^1 realizing

the maps $f_i : Co(M) \rightarrow \mathcal{S}_i$ we obtain $\Psi_i = f_i^{-1}(\mathcal{S}_i)$ according to Def. 8. $\mathcal{S}_i$ are sets of the mappings of the super–cones determined by the partitions Ψ_i, respectively. In analogy, we introduce the abstract map $f^* : Co(M) \rightarrow \mathcal{S}^*$ where $\mathcal{S}^*$ is representing the super–cones $s_l^* \in \Psi^*$ and $\Psi^* = (f^*)^{-1}(\mathcal{S}^*)$. Then each element of the set system $\Sigma_f = \{\mathcal{S}_1, \ldots, \mathcal{S}_{\beta_1}, \mathcal{S}^*\}$ can be taken as a new system of basic units for partitioning in the second level.

Yet, the above mixture strategy is a very simple one. In the next section we will improve this strategy using genetic algorithms. Thereby, condition 2 of Def. 10 becomes important.

3.2 Improved Mixture of Experts Using Genetic Algorithms

In this part we extend the mixture approach introduced in section 3.1 using *genetic algorithms*. In genetic algorithms populations of individuals (parents) produce new individuals (children) in a manner which is inspired by biological evolution and biological reproduction. The individuals are chains of strings describing a set of parameters which are to be optimized.[5] For applying genetic algorithms to graph partitioning let us consider a partitioning map $\Phi : U \rightarrow V$ where U and V are discrete finite sets. Furthermore let us try to optimize Φ regarding to a certain quality function Ω (fitness function). In this context an individual j represents a certain partition, determined by a map Φ_j. The i-th component of a string is associated with the i–th element of U containing the mapping goal which is an element of V. Several authors applied genetic algorithms to the graph partitioning problem, for instance [MGSK88].

However, we will involve this approach into the above described hierarchical strategy. Here we focus onto our special 2–level scheme (3.4). Then genetic algorithms may be used as one of the parallel working algorithms in each hierarchical level. Yet, because of the large number of cones in $Co(M)$ the string of an individual representing a partition of $Co(M)$ is too long for mastering. Therefore, often in this first level the application of genetic algorithms is impossible. On the other hand, if taking genetic algorithms in the second level of the hierarchical scheme, they require a uniform set of basic elements. To serve this assumption the usage of the superposition specified in Def. 10 is necessary. We remark again that the superposition of all produced partitions simultaneously forms a *generating system* for it, i.e., each partition may be described in terms of this generating system. Hence, the superposition fulfills the conditions of

[5] For a more detailed introduction see for instance [Koz92] or [Gol89].

a uniform basis for genetic algorithms as demanded. Now the initial population for the genetic algorithm is the set of all partitions determined in the first level which are described by the elements of the superposition. We emphasize that the several algorithms implement various partitioning strategies the best of which *a priori* is unknown. Still more, in general a merged strategy will improve the result significantly. How we can involve such a real mixture strategy using genetic algorithms? The recombination by *crossing over* offers an elegant way to join different properties of two individuals (partitions) into new ones. The crossing over scheme may be interpreted as a more general exchanging than the simpler one in the algorithm of KERNIGHAN AND LIN [KL70]. However, we have to take into account a second argument, how much of the old individuals get the chance to the competing step for selection of the new population. Two contrary methods are well known: first we have the $[\mu, \lambda]$–scheme[6] where μ individuals produce λ children with $\mu < \lambda$ and only the μ best of the λ children form the new population; the second one is the $[\mu + \lambda]$–scheme where all $\mu + \lambda$ individuals are allowed for the selection process. Both schemes have advantages. In the second one the best solution is preserved. Yet, it tends to a stagnation into a local minimum. In the $[\mu, \lambda]$–scheme this property is weakened. On the other hand, good solutions may be lost here. Therefore, we introduce a new so called $[\mu * \lambda]$–scheme: Let $t = 0, 1, 2, \ldots$ be the number of generations which already have been generated. Then the value

$$\mu_t = \text{int}\left[(\mu - \mu_\alpha) \cdot \sigma(t)\right] + \mu_\alpha \tag{3.6}$$

determines that one has to take into consideration μ_t of the old individuals in addition to the λ new produced in the selection process. Thereby, $\text{int}[x]$ stands for the integer value of x. The function $\sigma(t)$ is of decreasing sigmoid type with $0 \leq \sigma(t) \leq 1$. μ_α describes a survival probability for the parent individuals. Then we have $\mu_0 = \mu$ and $\lim_{t \to \infty} \mu_t = \mu_\alpha$. In this way we combine the advantages of both the $[\mu, \lambda]$– and the $[\mu + \lambda]$–strategy. The whole procedure, which includes the generation of a superposition and following genetic algorithm, finally leads to the complete scheme of the improved hierarchical mixture strategy depicted in Tab. 1.

3.3 Special Experts

Our mixture of experts approach is a framework for applying several partitioning algorithms. There exists a broad variety of such experts. A survey of algorithms suitable for parallel logic simulation is given in [Spo95]. We distinguish direct and iterative

[6] Here we use the notions of RECHENBERG and SCHWEFEL [Rec73, Sch81].

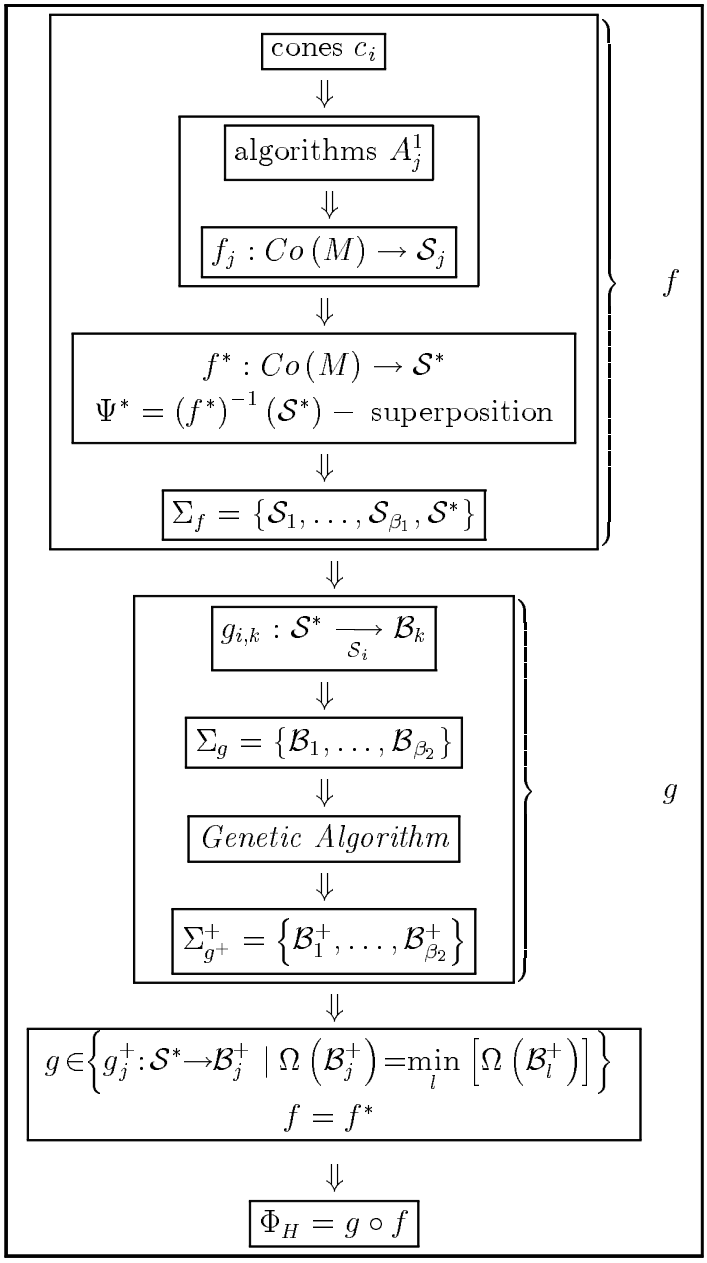

Table 1: Scheme of the improved mixture of experts strategy using genetic algorithms

partitioning algorithms. Direct partitioning algorithms construct a single partition resulting from basic units without building intermediate partitions. Iterative algorithms require an initial partition which is improved according to different quality functions. The algorithm by MUELLER-THUNS et al. [MTSDA93] is a direct one using cones as basic units. Other ones assign boxes to blocks in a special way [Hil81, SB93]. Well known iterative heuristics are those of KERNIGHAN & LIN [KL70] and its improvement by FIDUCCIA & MATTHEYSES [FM82] and SANCHIS [San89].

We have developed two new direct algorithms on the basis of cones aiming at balanced workload and minimal replication, the *Backward-Cone-Concentration* algorithm (*n-BCC*) and the *Minimum-Overlap Cone-Cluster* algorithm (*MOCC*).

The basic idea of *n-BCC* consists in iteratively assigning sets of cones to blocks with preferred choice of n cones overlapping each other using the box-related cone overlap degree u of Def. 4:

1. At first in the codomain of u a value n^* with smallest distance to n is fixed. All boxes in M_E are assumed to be unmarked.

2. Within all unmarked boxes out of $u^{-1}(n^*)$, a box e is chosen and the *fan-out cone* $co_O(e)$ (see Def. 3) starting from e is searched to find the head elements of the n^* cones covering e. These n^* cones are assigned to a block possessing the lowest number of cones for the moment and all boxes of the selected cones become marked. If there is a remaining unmarked box $e \in u^{-1}(n^*)$, then step 2 is repeated.

3. If there exists $n' \in \mathrm{cod}\ u$ with $u^{-1}(n')$ containing an unmarked box, then such an n' is taken as new n^* and one has to continue with step 2. Otherwise, the algorithm terminates.

Contrary to *MOCC* explained below, with *n-BCC* there is no explicit use of knowledge concerning the number of boxes in overlap regions or cones. First of all, *n-BCC* has been designed for application at the first level of our hierarchical strategy.

MOCC successively constructs a partition using the specifics of the corresponding hardware model in form of the overlap hypergraph *GU* according to Def. 6. With this algorithm the objective is to achieve partitions with blocks containing hypergraph nodes (cones) connected among one another with high-weighted hyperedges:

1. Initially, out of the m_c cones m_b ones with the highest numbers of boxes are assigned to the m_b blocks.

2. Taking block $B_i \in \mathcal{B}$ with the lowest number of boxes, we are looking for that overlap region $ovr(\mathcal{C}^*)$ of B_i with the maximum of the product of the number of boxes $|ovr(\mathcal{C}^*)|$ and the number of cones which are not assigned to B_i : $|\mathcal{C}^* \setminus \Phi^{-1}(B_i)|$.

3. Now we include the set of these up to now not assigned cones $\mathcal{C}^* \setminus \Phi^{-1}(B_i)$ of the selected overlap region $ovr(\mathcal{C}^*)$ to the block B_i.

4. If free cones exist yet, we proceed with step 2. Otherwise the partition is complete and the algorithm terminates.

MOCC aims at a minimum of multiple evaluation of boxes on different processors keeping a balanced workload corresponding to the resulting partition. If two-stage partitioning is necessary the complex structure of *GU* implies preferably applying *MOCC* to the second level of the hierarchical partitioning scheme.

4 Experimental Results and Conclusions

Finally, we present a special application of the improved mixture of experts strategy (Tab. 1) for a specific hardware model M representing a processor structure with $|M_E| = 16\,398$ boxes. This model was provided by *IBM* for a first testing phase of our strategy.

For the initial hierarchical level we use a set of n-BCC algorithms A_k^1 with β_1 parameters n and a fixed number of super-cones m_s. The crossing to the second level requires the production of an initial population Σ_g for the genetic algorithm to be applied. Generally, each S_i resulting from the first level, allows the production of many individuals $\mathcal{B}_k$ in the second level $g_{i.k} : S^* \xrightarrow[S_i]{} \mathcal{B}_k$, using the elements of S^* as new basic units and keeping such units together in one block of $\mathcal{B}_k$ which correspond to one and the same super-cone belonging to S_i. Here, we restrict the number of created initial individuals $\mathcal{B}_k$ to one for each S_i ($\beta_2 = \beta_1$). For the evaluation of individuals within the genetic algorithm and for choosing the final (best) partition described by Φ_H, a quality function Ω with $\Omega = \max_{1 \le i \le m_b} W_i / W_{seq}$ (maximum workload) is taken (see (2.6), (2.8)).

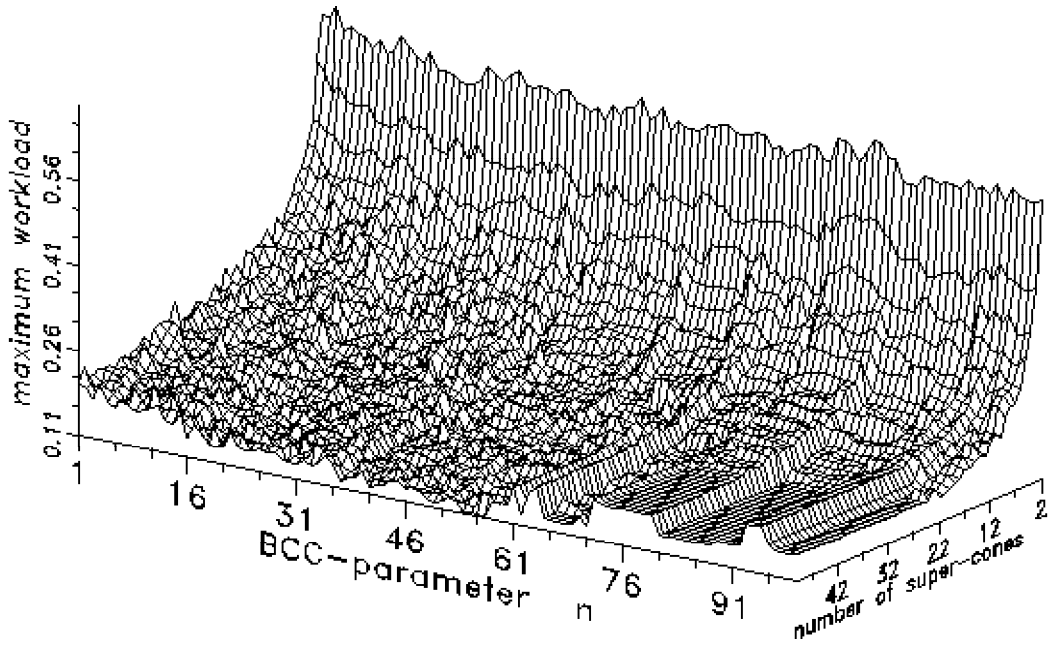

Figure 4: Quality function Ω concerning the maximum workload $\max_{1 \leq i \leq m_s} W_i / W_{seq}$ for the partitions resulting from the n-BCC for various values of m_s and parameters n.

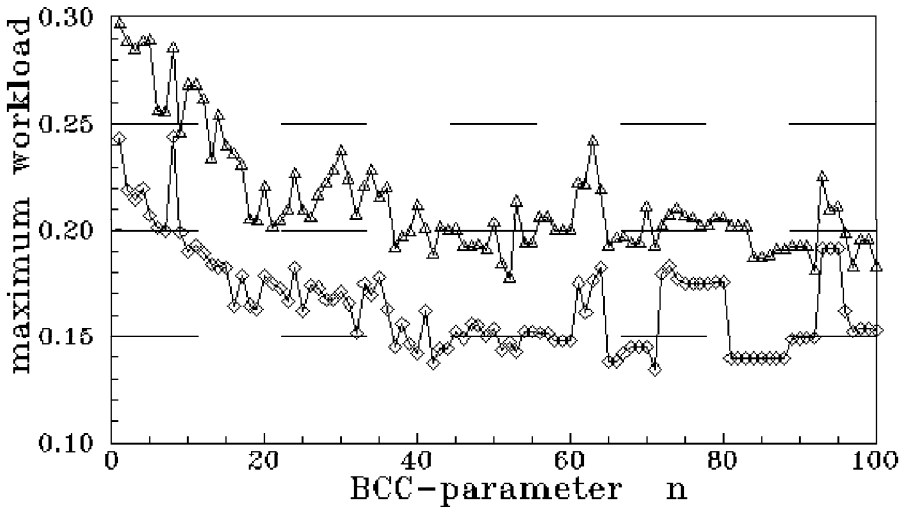

Figure 5: Maximum workload of partitions generated by n-BCC algorithms for $m_{s_1} = 16$ ($\triangle$) and $m_{s_2} = 32$ ($\diamond$) with respect to the parameter n.

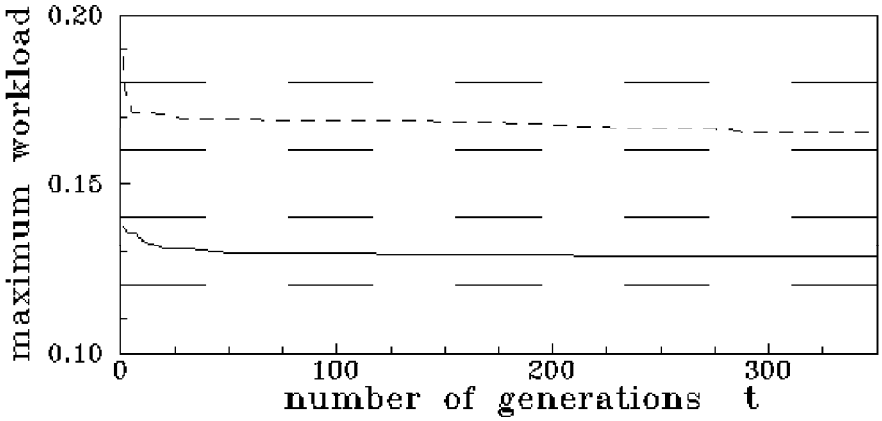

Figure 6: Maximum workload of the best partition generated by a genetic algorithm for $m_{s_1} = 16$ (short dashed) and $m_{s_2} = 32$ (straight line) with respect to time (number of generations).

In Fig. 4 the quality function Ω, applied to the partitioning results of the first hierarchical level of n-BCC for various $m_s = 2 \ldots 50$ with the parameters $n = 1 \ldots 100$, is shown.

In our exemplary application of the hierarchical scheme we consider only a fixed number m_s of super–cones, i.e., the set of possible n–BCC algorithms in the first level is restricted to such ones, which generate m_s super–cones. However the parameter n varies. The initial population for the genetic algorithm is chosen out of corresponding partitioning results randomly. The fitness (*maximum workload*) of all n–BCC experts with respect to m_{s_i} is depicted in Fig. 5 extracted from Fig. 4. The evaluation in time (number of generations) of the fitness of the best individuum of the population is shown in Fig. 6. In both cases the genetic algorithm generates partitions the maximum workload of which is better than the best ones of the initial n–BCC algorithms.

These first results show that a mixture of the *a priori* chosen strategies represented by the various n–BCC instances leads to improved partitions. The successful application of the genetic algorithms to the mixture strategy of partitioning algorithms was demonstrated using the idea of superposition of partitions.

A Appendix

At first for the proof of Theorem 1 we show the following lemma:

Lemma 1 *For $s_i^* \in \Psi^*$ and $s_j^* \in \Psi^*$ with $i \neq j$ holds: $s_i^* \cap s_j^* = \emptyset$, i.e. the elements of Ψ^* are pairwise disjoint.*

Proof of the lemma:

Let $s^*_{i_1 \ldots i_k} \in \Psi^*$ and $s^*_{j_1 \ldots j_k} \in \Psi^*$ be given with $(i_1, \ldots, i_k) \neq (j_1, \ldots, j_k)$. Then l exists such that $i_l \neq j_l$. Further we have $s^*_{i_1 \ldots i_k} = s_1^{i_1} \cap \ldots \cap s_l^{i_l} \cap \ldots \cap s_k^{i_k}$ and $s^*_{j_1 \ldots j_k} = s_1^{j_1} \cap \ldots \cap s_l^{j_l} \cap \ldots \cap s_k^{j_k}$. Then we obtain for the intersection $s^*_{i_1 \ldots i_k} \cap s^*_{j_1 \ldots j_k}$ the relation

$$s^*_{i_1 \ldots i_k} \cap s^*_{j_1 \ldots j_k} = s_1^{i_1} \cap s_1^{j_1} \cap \ldots \cap s_l^{i_l} \cap s_l^{j_l} \cap \ldots \cap s_k^{j_k} \cap s_k^{j_k} \tag{A.1}$$

According to Def. 10 of Ψ^* we get: $s_l^{i_l} \in \Psi_l$ and $s_l^{j_l} \in \Psi_l$. Because Ψ_l is a partition of U, $s_l^{i_l} \cap s_l^{j_l} = \emptyset$ is valid. $\square$

Now we are able to prove Theorem 1:

Proof of the theorem:

(I) Ψ^* is a generating system:

Let $s_i^j \in \Psi_i$ be arbitrarily chosen. We construct the sets S_l with $l \neq i$ according to the following rule: if $s_i^j \cap s_l^{j'} = \tilde{s}_l^{j'}$ with $s_l^{j'} \in \Psi_l$ and $\tilde{s}_l^{j'} \neq \emptyset$ holds then $\tilde{s}_l^{j'} \in S_l$. According to this construction, for each l the relation $\bigcup\limits_{j'} \tilde{s}_l^{j'} = s_i^j$ is valid. We consider the set $S^* = \left\{ s^*_{j'_1 \ldots j'_k} \mid s^*_{j'_1 \ldots j'_k} = \bigcap\limits_{\substack{l=1 \ldots k \\ l \neq i}} \tilde{s}_l^{j'_l} \;;\; \tilde{s}_l^{j'_l} \in S_l \right\} \setminus \{\emptyset\}$. Because of the definition of the $\tilde{s}_l^{j'_l}$ as intersections with $s_i^j \in \Psi_i$ we have $s^*_{j'_1 \ldots j'_k} \in \Psi^*$ and furthermore, $\bigcup\limits_{s^*_{j'_1 \ldots j'_k} \in S^*} s^*_{j'_1 \ldots j'_k} \subseteq s_i^j$. It still remains to show $\bigcup\limits_{s^*_{j'_1 \ldots j'_k} \in S^*} s^*_{j'_1 \ldots j'_k} \supseteq s_i^j$: Now we take an arbitrary but fixed $u \in s_i^j$. Then for each S_l one and only one $\tilde{s}_l^{j_l^*}$ exists such that $u \in \tilde{s}_l^{j_l^*}$, i.e., $u \in \bigcap\limits_{\substack{l=1 \ldots k \\ l \neq i}} \tilde{s}_l^{j_l^*}$ with $\bigcap\limits_{\substack{l=1 \ldots k \\ l \neq i}} \tilde{s}_l^{j_l^*} \in S^*$.

(II) Ψ^* is a partition of U:

According to (I) Ψ^* is a generating system for the $s_i^j \in \Psi_i$. Furthermore, it is assumed, that all Ψ_i are partitions of U themselves. Then one can find for each $u \in U$

a $s^*_{j^*} \in \Psi^*$ such that $u \in s^*_{j^*}$. Regarding to Lemma 1 all elements of Ψ^* are pairwise disjoint. This completes the proof of the theorem.□

References

[DEFH93] R. Der, H. Englisch, M. Funke, and M. Herrmann. Time Series Prediction Using Hierarchical Self-Organized Feature Maps. *Neural Network World*, (3):699–703, 1993.

[FM82] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proc. 19th Design Automata Conference, ACM/IEEE*, pages 175–181, 1982.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Lerning*. Addison Wesley, Reading, 1989.

[GS93] Markus H. Gross and F. Seibert. Visualization of multidimensional image data sets using a neural network. *Visual Computer*, 10:145–159, 1993.

[Hil81] W. Hilberg. Partitionierung mit Hilfe der Verbindungsmatrix. *ntz Archiv*, 3(3):57–62, 1981.

[JJ94] M. I. Jordan and R. A. Jacobs. Hierarchical Mixture of Experts and the EM Algorithm. In P. Morasso, editor, *Proc. ICANN'94*, pages 479–486. Springer, 1994.

[KL70] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*, 49(2):291–307, 1970.

[KMP95] J. Kohlmorgen, K.-R. Müller, and K. Pawelzik. Improving Short-Term Prediction with Competing Experts. In *Proc. ICANN'95*, Paris 1995, 1995.

[Koz92] J. R. Koza. *Genetic Programming*. MIT Press, 1992.

[Len90] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. Teubner-Verlag Stuttgart and JOHN WILEY & SONS, 1990.

[Man92] N. Manjikian. High performance parallel logic simulation on a network of workstations. Technical Report CCNG T-220, Department of Electrical and Computer Engineering and Computer Communications Network Group, University of Waterloo, 1992.

[MGSK88] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer. Evolution Algorithm in Combinatorial Optimization. *Parallel Computing*, (7):65–88, 1988.

[MTSDA93] R. B. Mueller-Thuns, D. G. Saab, R. F. Damiano, and J. A. Abraham. VLSI Logic and Fault Simulation on General Purpose Parallel Computers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12:446–460, 1993.

[Rec73] I. Rechenberg. *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Information.* Fromman Verlag Freiburg (Germany), 1973.

[RMS92] Helge Ritter, Thomas Martinetz, and Klaus Schulten. *Neural Computation and Self-Organizing Maps: An Introduction.* Addison-Wesley, Reading, MA, 1992.

[San89] L. A. Sanchis. Multiple-way network partitioning. *IEEE Transactions on Computers*, 38(1):62–81, 1989.

[SB93] C. Sporrer and H. Bauer. Corolla partitioning for distributed logic simulation of VLSI-circuits. In *Proc. Of the Workshop on Parallel and Distributed Simulation (PADS)*, pages 85–92, 1993.

[Sch81] H.-P. Schwefel. *Numerical Optimization of Computer Models.* Whiley and Sons, 1981.

[Spo95] C. Sporrer. *Verfahren zur Schaltungspartitionierung für die parallele Logiksimulation.* Verlag Shaker Aachen, 1995.

[Spr89] W. G. Spruth. *The Design of a Microprocessor.* Springer Berlin, Heidelberg, 1989.

[SUM87] S. P. Smith, B. Underwood, and M. R. Mercer. An analysis of several approaches to circuit partitioning for parallel logic simulation. In *Proceedings IEEE International Conference on Computer Design (ICCD)*, pages 664–667, 1987.

[Vil95] Th. Villmann. *Topologieerhaltung in selbstorgansierenden neuronalen Merkmalskarten.* Diss., eingereicht, Universität Leipzig, 1995.

[VMB+94] A. Verikas, K. Malmquist, M. Bachauskene, L. Bergman, and K. Nielson. Hierarchical Neural Nets for Color Classification. In P. Morasso, editor, *Proc. ICANN'94*, pages 847–851. Springer, 1994.