Proc. 5th European Congr. on Intell. Techn. and Soft Comp. (EUFIT'97), p.804-808, 1997.

HIERARCHICAL MODEL PARTITIONING FOR PARALLEL VLSI–SIMULATION USING EVOLUTIONARY ALGORITHMS IMPROVED BY SUPERPOSITIONS OF PARTITIONS

R. Haupt[†], K. Hering[†], U. Petri[†] and Th. Villmann[‡]
Universität Leipzig
[†]Institut für Informatik, Augustusplatz 10/11, 04109 Leipzig, Germany
[‡]Klinik für Psychotherapie und psychosomatische Medizin
Karl–Tauchnitz–Str. 25, 04107 Leipzig, Germany
Phone +49 341 - 9732284, Fax +49 341 - 9739348
email: {haupt,hering,petri,villmann}@informatik.uni-leipzig.de

ABSTRACT: Parallelization of VLSI-simulation exploiting model-inherent parallelism is a promising way to accelerate verification processes for whole processor designs. Thereby partitioning of hardware models influences the efficiency of following parallel simulations essentially. Based on a formal model of *Parallel Cycle Simulation* we introduce *partition valuation* combining communication and load balancing aspects.
We choose a 2-level hierarchical partitioning scheme providing a framework for a mixture of experts strategy. Considering a complete model of a *PowerPC 604* processor, we demonstrate that *Evolutionary Algorithms* can be applied successfully to our model partitioning problem on the second hierarchy level, supposing a reduced problem complexity after fast pre-partitioning on the first level. For the first time, we apply *superpositions* during execution of Evolutionary Algorithms, resulting in a faster decreasing fitness function and an acceleration of population handling.

1 MODEL PARTITIONING FOR VLSI–SIMULATION

We consider parallel logic simulation processes where several simulator instances co-operate over a loosely-coupled processor system. Each instance simulates a part of a complex hardware model. The corresponding model partitioning problem can be formulated as a combinational optimization problem. In this context partitions are characterized by costs (here: run-time of a corresponding parallel simulation). We take a formal model of parallel cycle simulation as basis of partition valuation.

1.1 MODEL CONSTRUCTION

A framework of concepts for the formal description of parallel cycle simulation combining structural and behavioural aspects is developed in [Her96]. At first a *Structural Hardware Model (SHM)* is introduced as bipartite graph $M = (M_B, M_S, M_\mathcal{R})$ where nodes (boxes) of $M_B = M_E \cup M_I \cup M_O \cup M_L$ represent design components as, for instance, logical gates (M_E), input and output pins ($M_I \cup M_O$) and latches (M_L). The elements of M_S stand for wires. For *SHM*s sequences of not closer specified actions bound to nodes of M_B are defined as *behaviour*. They are called *Sequential Cycle Simulation (SCS)*. With respect to partition valuation actions are considered as sources of simulation expense.
Our partitioning approach outlined in [HHV96] takes cones (subsets of M_B) as fundamental building blocks for partitions. Usually, cones can overlap each other. For each model M a cone set $Co(M)$ is given. Partitions of M are introduced as partitions of $Co(M)$ in mathematical sense. A *Parallel Structural Hardware Model (PSHM)* M^Ψ (with respect to a partition Ψ of a *SHM* M) embodies a set of *SHM*s $M_\Psi^\mathcal{C} = \left(M_B^\mathcal{C}, M_S^\mathcal{C}, M_\mathcal{R}^\mathcal{C}\right)$, each element determined by a cone set $\mathcal{C} \in \Psi$. Special sets $M_{I,L}^\mathcal{C} \subseteq M_I^\mathcal{C}$ and $M_{O,L}^\mathcal{C} \subseteq M_O^\mathcal{C}$ are devoted to communication between *PSHM* components which is restricted to cycle boundaries. *Extended Sequential Cycle Simulation (ESCS)* is introduced as *PSHM* component behaviour. Each *ESCS* consists of a leading sub-sequence of cycle-internal actions followed by a sub-sequence of communication related actions. The latter are split into three parts representing communication pre-processing, true communication and communication post-processing.

Finally, *Parallel Cycle Simulation (PCS)* is defined as behaviour of a whole *PSHM* on the basis of a model of parallel computation called *Communicating Processors (CP)*. A *PCS* embodies a sequence of action sets which can be interpreted as combination of *ESCS*s belonging to the components of the corresponding *PSHM.*

1.2 PARTITION VALUATION

Our objective of model partitioning is to *minimize* the partition-dependent *run-time* t^{Ψ} for a corresponding parallel simulation[1]. The run-time t^{Ψ} is estimated by the sum of several terms which belong to sequences of actions related to the time consumption for parallel simulation. According to our model of *PCS*, we distinguish three kinds of actions which influence t^{Ψ}:

1. Box evaluation and latch update actions are included for a partition component $\mathcal{C} \in \Psi$ by:
$$t_B^{\mathcal{C}} = t_B^M \cdot \left| M_E^{\mathcal{C}} \cup M_L^{\mathcal{C}} \right| \tag{1.1}$$
depending on the *SHM* M with an averaged time consumption t_B^M per single box.
2. For a partition component $\mathcal{C} \in \Psi$ the amount
$$t_{\text{comm}}^{\mathcal{C}} = t_{\text{pre\_comm}}^{\mathcal{C}} + t_{\text{post\_comm}}^{\mathcal{C}} = t_{\text{comm}}^{M} \cdot \left| M_{O,L}^{\mathcal{C}} \right| + t_{\text{comm}}^{M} \cdot \left| M_{I,L}^{\mathcal{C}} \right| \tag{1.2}$$
has to be taken into account concerning pre-processing and post-processing actions related to the box sets $M_{O,L}^{\mathcal{C}}$ and $M_{I,L}^{\mathcal{C}}$, respectively, whereby t_{comm}^{M} is a constant.
3. The *CP* model is substantiated with the strong synchronizing *mpc_index command* belonging to the *Message Passing Library* of the *AIX Parallel Environment.* This command initiates collective communication between the corresponding processors. The time consumption depends on the number of processors (number of partition components $|\Psi|$) and the maximum number of values which have to be transferred between any pair of processors: $\max_{\mathcal{C}_1,\mathcal{C}_2 \in \Psi} \left| M_{I,L}^{\mathcal{C}_1 \to \mathcal{C}_2} \right|$. In this context the mpc_index command requires the time
$$t_{\text{mpc\_index}}^{\Psi} = \max_{\mathcal{C}_1,\mathcal{C}_2 \in \Psi} \left| M_{I,L}^{\mathcal{C}_1 \to \mathcal{C}_2} \right| \cdot \left(t_a + t_b \cdot |\Psi| \right) \tag{1.3}$$
with $M_{I,L}^{\mathcal{C}_2} = \bigcup_{\mathcal{C}_1 \in \Psi} M_{I,L}^{\mathcal{C}_1 \to \mathcal{C}_2}$ and t_a and t_b as system-dependent constants.

Using (1.1), (1.2), and (1.3), the run-time t^{Ψ} is given by
$$t^{\Psi} = t_{\text{mpc\_index}}^{\Psi} + \max_{\mathcal{C} \in \Psi} \left(t_B^{\mathcal{C}} + t_{\text{comm}}^{\mathcal{C}} \right) . \tag{1.4}$$

The strong synchronization of the mpc_index command implies that the run-time t^{Ψ} is determined by the processor which consumes the biggest amount of time for box evaluation and latch updating together with the pre- and post-processing of the communication.

2 PARTITIONING STRATEGY

2.1 HIERARCHICAL PARTITIONING USING SUPERPOSITIONS OF PARTITIONS

Before presenting a description of our hierarchical partioning approach we have to introduce the concepts of partitioning and partition. In the following, we always assume that both U and V are non-empty sets.

Definition 1 *1.) A* ***partitioning*** *of U with respect to V is a unique map $\Phi : U \longrightarrow V$ assigning each element $u \in U$ to an element $v \in V$.*
2.) A ***partition*** *Ψ_Φ of U related to the partitioning $\Phi : U \longrightarrow V$ is given by $\Psi_\Phi = \left\{ \Phi^{-1}(v) \mid v \in \text{cod } \Phi \right\}$, where* cod Φ *is the range of Φ.*

[1]In the following, simulation time is generally related to one clock-cycle in the parallel simulation.

An element $v \in \text{cod } \Phi$ 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 $\mathcal{B}$ of m_b blocks each of them assigned to a single processor for parallel simulation. Then our task is to find a partitioning

$$\Phi_{\text{opt}} : Co(M) \to \mathcal{B} \tag{2.1}$$

with minimum run-time $t^{\Psi_{\Phi_{\text{opt}}}}$ according to (1.4).
One important aspect of partitioning in the context of VLSI–simulation is the extremely large number of objects which have to be mapped. Hence, a non-trivial one–step partitioning is often not feasible because of the extensive computation time and memory allocation costs. Therefore, we focus on a *hierarchical partitioning strategy* [HHV96].
In this approach we split the general partitioning scheme (2.1) into a q–level one [HHV96], which reduces the complexity in each level. Here, we use a 2–level scheme:

$$\Phi_H : Co(M) \xrightarrow{\Phi_1} \mathcal{S} \xrightarrow{\Phi_2} \mathcal{B} \tag{2.2}$$

with $\Phi_H = \Phi_2 \circ \Phi_1$. Thereby, $\mathcal{S}$ is a set of elements S_l, the pre–images $s_l = \Phi_1^{-1}(S_l)$ of which are called *super–cones.* Of course, in general Φ_H is only an approximation of Φ_{opt} in (2.1). However, one can handle each sub–partitioning Φ_j $(j = 1, 2)$ separately. This allows the application of various partitioning algorithms for each Φ_j. Yet, an *a priori* optimal choice of Φ_j often is impossible. To overcome this difficulty we prefer a strategy introduced in neurodynamics by Jordan et al. [JJ94], the so–called *mixture of experts approach,* which we have applied to the hierarchical partitioning strategy [HHV96]. In the scheme (2.2) we consider several partitioning algorithms A_i^j, $i = 1 \dots m_j$ corresponding to maps Φ_i^j in one hierarchical step j working independently. The several heuristics of the A_i^j influence the structure of the resulting partitions $\Psi_{A_i^j}$. To mix the different properties (expert knowledge) of the $\Psi_{A_i^j}$ we build a superposition of them. In general we define:

Definition 2 *Let $\Pi = \{\Psi_1, \dots, \Psi_k\}$ be a system of partitions of the set U. The elements of Ψ_i are denoted by s_i^j, $j = 1 \dots n_i$. $\tilde{\Psi}_\Pi = \{\tilde{s}_1, \dots, \tilde{s}_m\}$ is called a* ***superposition*** *of Π if and only if:*

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

Def. 1 yields $\emptyset \notin \tilde{\Psi}_\Pi$. We consider the following special construction of superpositions:

Theorem 3 *Let $\Pi = \{\Psi_1, \dots, \Psi_k\}$ be a system of partitions of the set U. The elements of Ψ_i are denoted by s_i^j, $j = 1 \dots n_i$. Furthermore, let Ψ_Π^* be given as*

$$\Psi_\Pi^* = \left\{ s^*_{j_1 \dots j_k} \mid s^*_{j_1 \dots j_k} = \bigcap_{i=1 \dots k} s_i^{j_i} \right\} \setminus \{\emptyset\} \tag{2.3}$$

with $j_i = 1 \dots n_i$. Then Ψ_Π^* is a superposition of Π. Furthermore, for all superpositions $\hat{\Psi}$ of Π with $\hat{\Psi} \neq \Psi_\Pi^*$ the relation $\left|\hat{\Psi}\right| > |\Psi_\Pi^*|$ is valid, i.e. Ψ_Π^* has the ***maximum granularity.***

Proof: The proof of the theorem is shown in [HHV96].
The so-called *maximum superposition* Ψ_Π^* of Π takes the different properties of the Ψ_i into account. Keeping together two elements of U within one component of Ψ_Π^* is to be interpreted as "collective decision" of all algorithms involved; the "individual decision" of one algorithm placing two elements of U into different partition components yields an assignment of these elements to different components of Ψ_Π^*, too. Due to Def.2 Ψ_Π^* is a generating system for all Ψ_i. These aspects are the reasons for integrating the maximum superposition into our hierarchical strategy.
In our 2–level scheme (2.2), the superposition is built after the first partitioning level. If we assume that we have *various* algorithms A_i^1 realizing the different maps $\Phi_i^1 : Co(M) \to \mathcal{S}_i$ we obtain $\Psi_i = \left(\Phi_i^1\right)^{-1}(\mathcal{S}_i)$ in agreement with Def. 1. Now we specify Π as the set of all Ψ_i and Ψ_Π^* as its superposition according to (2.3). In analogy to the maps Φ_i^1, we introduce the abstract map

$$\Phi_1^* : Co(M) \to \mathcal{S}^* \tag{2.4}$$

where $\mathcal{S}^*$ is representing the set of super–cones $s_l^* \in \Psi_\Pi^*$ and $\Psi_\Pi^* = (\Phi_1^*)^{-1}(\mathcal{S}^*)$. Then the set $\mathcal{S}^*$ can be taken as a new system of basic units for partitioning in the second level.

2.2 EVOLUTIONARY ALGORITHMS IN THE CONTEXT OF HIERARCHICAL PARTITIONING

In this chapter we extend the simple mixture approach introduced in section 2.1 using *Evolutionary Algorithms* (EAs). Thereby, condition 2 of Def. 2 becomes important.
In EAs populations of individuals (parents) produce new individuals (children) in a manner which is inspired by biological evolution and reproduction. The individuals are strings describing a set of parameters which are to be optimized.[2] Applying EAs to graph partitioning we consider a partitioning map $\Phi : U \to V$. One has to optimize Φ regarding to a certain quality function Ω (fitness function) which is chosen here as run-time t^{Ψ} (1.4). In this context an individual i represents a certain partition, determined by a map Φ_i. The l-th component of the string is associated with the l–th element of U containing the mapping goal which is an element of V. Several authors have applied EAs to graph partitioning, for instance [MGSK88].
We involve this approach into the second level of the above described hierarchical strategy (2.2) taking $\mathcal{S}^*$ from (2.4) as the set of basic units.[3] The initial population is built by several partitioning algorithms A_i^2. Here, in addition to the usual scheme (2.2), in the second level we insert for certain time steps $t_k \in T$ the possibility of further superpositions of the actual partitions (described by the individuals). Then all individuals are expressed in terms of the new basic units and the next EA–period takes place. This approach induces a further reduction of the search space and a lower time amount for producing new populations.
As evolution strategy we prefer a new so–called $[\mu * \lambda]$–scheme, explained in [HHV96], which balances both the $[\mu, \lambda]$– and the $[\mu + \lambda]$–strategy (in the notation of [Sch81]).

3 APPLICATION TO MODELS OF REAL PROCESSORS

Here, we give an example of applying our 2-level hierarchical partitioning strategy to a model representing a PowerPC 604 processor. This processor model is characterized by $|M_B| = 330\,031$ boxes. The set which has to be partitioned consists of $|Co(M)| = 46\,998$ elements. For run-time prediction we assume the following constants: $t_B^M = 111$ ns, $t_{comm}^M = 204$ ns, $t_a = 6$ ns, and $t_b = 6$ ns, respectively. Sequential simulation requires a run-time per cycle of $t = 36.63$ ms.

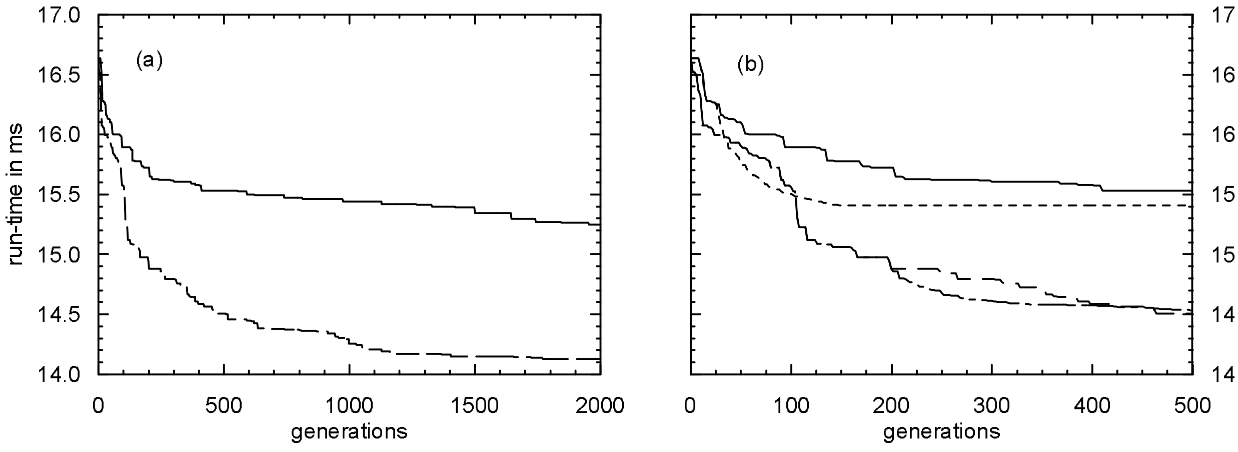

Figure 1: Predicted run-time t^{Ψ} corresponding to the best partition: (a) without (full line) and with (dashed line) superposition at the beginning of the EAs; (b) same as in (a), and additionally, with superposition only after each 25 generations (short-dashed line) and with superposition at the beginning of the EAs and after each 200 generations (dashed-dotted line).

For the first level in the hierarchical strategy we use the simple but in this case proper STEP partitioning algorithm[4]. Thus, we have a partitioning that maps the cone set to a set of 2000 super-cones which is taken as $\mathcal{S}^*$ from (2.4) for the second level. The previously developed MOCC partitioning algorithms [HHV96] yield a

[2]For a more detailed introduction see for instance [Mic96].

[3]In general, EAs may be used in each hierarchical level. Yet, the large number $|Co(M)|$ would require too extensive computation time and memory allocation costs.

[4]STEP breaks the cone list into pieces of equal length using the interior order of cone indices.

set Π of initial partitions (with $|\mathcal{B}| = 4$ in (2.2)), which forms the initial population of the following EAs. The individuals of the initial population are already of rather good quality concerning their fitness function. On the other hand, all initial partitions are mutually different. These two features are important for a successful application of the superposition to the usual EAs.
In Fig.1 different strategies of applying the concept of superposition to the EAs are compared. If one has initial partitions produced by the MOCC algorithms, as in the example described here, a reduction of the search space for the genetic operations after an immediate superposition for $T = T_0 = \{0\}$ occurs (see section 2.2). This leads to a remarkable improvement of the best partitions and an acceleration of the EAs (Fig.1(a)). These are two essential qualities which make EAs applicable for VLSI-partitioning with such a complex fitness function t^Ψ (1.4). Of course, due to search space reduction the global minimum of the fitness function t^Ψ can be lost. But, usually one wants to get a good but not a perfect partition in short time for parallel VLSI-simulation. To achieve this aim, it can be useful to apply further superpositions after time steps $\hat{T} = \{t_1, t_{2,\ldots}\}$ with $t_i > 0$. In Fig.1(b), it is shown that an additional acceleration of the decrease of the run-time can be achieved by applying superpositions with respect to $T = \hat{T} \cup T_0$ or $T = \hat{T}$. Especially, the superpositions at time steps from $\hat{T}$ essentially reduce the time consumption for the EAs but also further restrict the search space and, hence, possibly the quality of the final best partition.

4 CONCLUDING REMARKS

The strategy of hierarchical partitioning enables a succesful application of EAs to the problem of model partitioning for parallel VLSI-simulation. We have incorporated the concept of superposition of partitions into the EA approach. It was shown that for sufficient good initial partitions which are mutually different from each other, a superposition of them reduces the search space in such a way that the decrease of the fitness is accelerated. Furthermore, the EAs itself work faster because of the reduced number of elements which are to be mapped and the simplified calculation of the run-time (fitness function). On the other hand, the global optimum of the fitness function may be excluded from the search space, but this is already possible due to the introduction of the hierarchical partitioning scheme. Superpositions can be applied for the initial population but also during the evolution strategy.
Generally, the concept of superposition can be used for EAs with a complex fitness function, where good initial individuals exist and a fast sub–optimal solution of the problem is required.

ACKNOWLEDGEMENT
THIS WORK WAS SUPPORTED BY DEUTSCHE FORSCHUNGSGEMEINSCHAFT (DFG) UNDER THE GRANT SP 487/1-2. THE AUTHORS WOULD LIKE TO THANK D. DÖHLER, H. HENNNINGS, R. REILEIN AND TH. SIEDSCHLAG FOR GRATEFUL SUPPORT DURING THE TEST PHASE AND PROGRAMMING WORK.

REFERENCES

[Her96] K. Hering. Parallel cycle simulation. Technical Report 13, University of Leipzig / Inst. of Informatics, Germany, 1996.

[HHV96] K. Hering, R. Haupt, and Th. Villmann. Hierarchical Strategy of Model Partitioning for VLSI–Design Using an Improved Mixture of Experts Approach. In *Proc. Of the Conference on Parallel and Distribute Simulation (PADS)*, pages 106–113. IEEE Computer Society Press, Los Alamitos, 1996.

[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.

[Mic96] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer–Verlag Berlin Heidelberg New York, third, revised and extended edition, 1996.

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

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