INTEGRATION OF A LOCAL SEARCH OPERATOR INTO EVOLUTIONARY ALGORITHMS FOR VLSI–MODEL PARTITIONING

Reiner Haupt, Klaus Hering and Thomas Siedschlag
Universität Leipzig, Institut für Informatik,
Augustusplatz 10/11, 04109 Leipzig, Germany
Phone +49 - 341 - 9732284, Fax +49 - 341 - 9739348
email: {haupt,hering,siggi}@informatik.uni-leipzig.de

ABSTRACT: The application of *Evolutionary Algorithms* in hierarchical model partitioning for parallel system simulation in VLSI design processes has proven to be successful. Thereby, individuals embody partitions of hardware designs. On the basis of a formal model of parallel cycle simulation a fitness function is chosen combining load balancing and interprocessor communication aspects.
As supplement to the concept of superposition we introduce a *Local Search Operator* to achieve a fast decreasing fitness function during evolution. This operator is based on a modification of a classical iterative partitioning algorithm by FIDUCCIA-MATTHEYSES. Results are shown for the partitioning of two real processor models, representing the *PowerPC 604* and an *IBM S/390* processor.

1 INTRODUCTION

The background of the present paper is given by simulation processes for functional design verification [Spr89] 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. Time-extensive simulation processes are necessary for the logic design of whole microprocessor structures. Here 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 time spent for simulation processes can be drastically reduced by parallelization based on model partitioning.

According to the use of *parallelTEXSIM* [Döh96] as parallel version of the cycle-based simulator *TEXSIM*[1], basic elements for building partitions of a hardware model [HHV96] are the so-called (fan-in) *cones*. Cones contain all logical boxes which have the potential to influence at least one input of a corresponding cone-head (latch) during the simulation of one cycle. For each model M a cone set $Co(M)$ is given. Usually, cones can overlap each other. Partitions of M are introduced as partitions of $Co(M)$ in mathematical sense. A distribution of overlapping cones to different partition components leads to a replication of boxes of the corresponding model parts. Nets leaving one partition component and feeding another one imply interprocessor communication during parallel simulation.

The model partitioning problem can be formulated as a combinational optimization problem. In this context partitions are characterized by a complicated cost function [HHPV97] which models the *run-time* of one parallel simulation cycle of the corresponding hardware model parts. This cost function is to be minimized to reduce the expected parallel simulation time. As basis of the run-time estimation a formal model of parallel cycle simulation [Her96] is taken. In this context the partition-dependent run-time results from the maximum over component-related sums of three terms. Each partition component $\mathcal{C}$ (set of cones) is accompanied by two terms:

1. the time $t_B^{\mathcal{C}}$ spent for the evaluation of logical boxes and the update of latches, proportional to the number of boxes, and
2. the time $t_C^{\mathcal{C}}$ spent for communication pre- and post-processing of cut nets, proportional to the number of cut nets.

[1] copyright by IBM

Additionally, for all partition components a communication term t_{CC} is included. The time consumption of the underlying collective communication depends on the number of processors (model parts) and the maximum number of net values which have to be transferred between any pair of processors.

For an effective calculation of the run-time belonging to a partition of a model M two hypergraphs are deduced from M:

1. *Overlap hypergraph* $OHG(M) = (V_O, E_O)$

   The set V_O of vertices is identified with the cone set $Co(M)$. A cone set $\mathcal{V} \subseteq V_O$ forms a hyperedge belonging to E_O if the set $B(\mathcal{V})$ of boxes of M which are exactly covered by the cones of $\mathcal{V}$ is non-empty. Each hyperedge $\mathcal{V}$ is weighted with $|B(\mathcal{V})|$. $OHG(M)$ allows an effective calculation of the terms $t_B^{\mathcal{C}}$ for the run-time estimation.

2. *Communication hypergraph* $CHG(M) = (V_C, E_C)$

   V_C is also identified with $Co(M)$. An ordered pair (C_O, C_I) of two cone sets $C_O, C_I \subset V_C$ with $C_O = \{c_o\}$ and $C_O \cap C_I = \emptyset$ forms a directed hyperedge of E_C if there exists a net of M with the cone-head of c_o as source and with sinks in all cones of C_I but not in $V_C \setminus (C_I \cup C_O)$. Each hyperedge is weighted by the number of nets belonging to the corresponding hyperedge (C_O, C_I). $CHG(M)$ is used for the calculation of communication-related terms $t_C^{\mathcal{C}}$ and t_{CC}.

For large VLSI-models the number of cones may be up to the range of $10^6 - 10^7$. Taking into consideration the complicated cost function, a direct partitioning of such a cone set to a set of blocks (model parts) in the range of 10^1, yielding an acceptable *speedup* for parallel simulation, is not practicable. Therefore, we have introduced a *hierarchical partitioning strategy* [HHV96]. Usually, a two-level scheme is applied. After fast prepartitioning, where the cone set is split into a set of super-cones with a cardinality in the range of $10^2 - 10^4$, on the second hierarchy level the set of super-cones is partitioned to the set of blocks using more complex algorithms. In [HHV96] we have successfully applied *Evolutionary Algorithms* (EAs) to the second hierarchy level for model partitioning. To achieve better partitioning results in shorter time, we have integrated the concept of *superposition* [HHPV97] into the evolution strategy.

In this paper a new genetic operator, the *local search operator* (LSO), is introduced to overcome the restricted applicability of superpositions. The application of the EAs within our hierarchical strategy is explained in Section 2. After developing the LSO in Section 3, different strategies for the integration of LSO into the EAs are compared for two microprocessor models in Section 4. Concluding remarks are given in section 5.

2 EVOLUTIONARY ALGORITHMS FOR VLSI MODEL PARTITIONING

On the first hierarchy level, prepartitioning of $Co(M)$ is realized by the STEP algorithm taking advantage of special information about the representation of the hardware models in the data base *DA_DB* [Zik92]. The components of prepartitions (super-cones) are used as basic set for the final partitioning in the second hierarchy level.

After reduction of the problem complexity, we can apply more expensive algorithms which directly use the run-time estimation for valuation of intermediate partitions or take advantage of the overlap and communication hypergraph. To introduce EAs in this level we need a set of initial partitions as start population. The usual way for producing a start population is a random assignment of block indices within a valid range to the set of super-cones. In [HHV96] we have developed the partitioning algorithm *MOCC* which constructs rather good initial partitions using our special data structures – OHG and CHG. There are parameters inherent in this algorithm which guarantee that the set of initial partitions shows a necessary heterogeneity in its partitions.

Our EA strategy is a mixture of Genetic Algorithms [Hol75] and Evolutionary Strategies [Rec73]. The *fitness* of an individual is identified by the expected run-time of a corresponding partition. We use a straightforward coding of individuals, each gene represents a super-cone and is coded as integer by the block index. The gene sequence results from the indexing of the underlying cones in the *DA_DB*. Generally, we use two-point crossover, inversion and mutation of the block index as genetic operators. For selection we apply our $[\mu * \lambda]$–strategy [HHV96] which starts with the $[\mu + \lambda]$–strategy and develops towards the $[\mu, \lambda]$–strategy [Sch81] during evolution always preserving the best individuals.

Considering the effect of parallelization on simulation, besides the pure reduction of simulation time, the amount of preparing data structures for parallel simulation has to be taken into account. In our case, the latter

essentially appears as amount of model partitioning. Therefore, our objective is to produce "good" partitions (concerning run-time of related parallel simulations) by handling a relatively low number of generations during the application of EAs.

Facing this objective, we have integrated the concept of *superposition* [HHPV97] into these algorithms. A superposition of a set of partitions is a partition again, embodying a fusion of the original partitions and allowing a new representation of them on the basis of the components of the superposition. The formation of superpositions within EAs together with building new representations of generations on the basis of these superpositions restricts the search space for partitions to accelerate the process of generation handling and to obtain good partitioning results faster. But, these results are coupled to special conditions imposed on the population. The superposition concept only works if the individuals are sufficiently fit and, additionally, genetically different from each other.

As supplement to the concept of superposition, we introduce a *local search operator* which does not restrict the search space and is able to guide individuals towards local minima of the fitness function.

3 INTEGRATION OF A LOCAL SEARCH OPERATOR

Partitioning algorithms are well-investigated for the min-cut graph-partitioning problem (see for instance [AK95]). The min-cut problem is formulated for a vertex- and/or edge-weighted graph; the objective is to find a partition of the vertex set with equally distributed sums of vertex weights and a minimum sum of cut edge weights related to the partition components. A very important and widely used class of partitioning algorithms has been developed starting from the move-based iterative algorithms initially designed by KERNIGHAN-LIN [KL70] and FIDUCCIA-MATTHEYSES (FM) [FM82]. Spoken in the language of EAs, these algorithms start from an individual (partition) and construct new individuals on the basis of neighborhood relations, carrying out series of simple changes in the gene structure of the corresponding individuals (move a super-cone or swap a pair of super-cones between two blocks). Such *local search algorithms* are very effective for finding local minima. Considering complex fitness functions with potentially many local minima (as, for instance, our partition-based function expressing expected run-time), the integration of local search algorithms as operator into the global working EAs should be very advantageous.

Because our partitioning problem based on the OHG and CHG is much more complex than the usual min-cut problem, we have developed a new local search algorithm outgoing from the original FM algorithm.

For an actual partition a non-marked super-cone is chosen for the move to another block according to the highest improvement of the run-time for the new partition or the lowest change for the worse. Each moved super-cone is marked and must not be moved again. A so-called pass has finished if all super-cones are marked. Then, that partition with the lowest run-time during the pass is selected for a new pass. All super-cones become unmarked and new passes are carried out up to no more improvement can be found. This intuitive but complex deterministic algorithm only works effectively if additional so-called gain structures are used which accelerate the calculation of run-time changes (gains) during a pass for the iteratively changed partitions. If the run-time for a given partition is known, the gain structure allows the calculation of the run-time of a changed partition by adding special values of this gain structure to the run-time of the starting partition. The gain structure depends on the actual partition and the graph which has to be partitioned.

Usually, for the min-cut problem, gain structures are based on one single graph. So, our special data structures, the two hypergraphs OHG and CHG, and the corresponding optimization function which expresses run-time, forced the construction of a new gain structure which almost exactly allows the calculation of run-time changes for modified partitions neglecting the communication-related term t_{CC}. The new gain structure, we have developed, significantly reduces the time necessary for applying our local search algorithm to a given partition. This fact allows an introduction of this modified FM algorithm as new complex *local search operator* (LSO) into the evolution strategy.

In addition to the usual EA procedure the LSO is placed between the application of the usual genetic operators and the selection step of the new μ parents for the following generation. In principle, the LSO can be applied to all λ children of a generation. But, practically, the LSO is used very sparsely to keep the balance between the global view of the EAs and the local one of the LSO.

In Section 4 we show the influence of the LSO considering two different hardware models which are partitioned into 2 and 8 blocks.

4 EXPERIMENTAL RESULTS

Expected parallel simulation run-times per cycle (fitness) of the best individuals of each generation without and with the use of the LSO are shown in Fig. 1 for the following two hardware models:

- *IBM S/390* processor: $|Co(M)| = 737\,606$, sequential run-time: 212.57 ms;
- *PowerPC 604* processor: $|Co(M)| = 46\,998$, sequential run-time: 36.63 ms.

The prepartitioning is realized by the STEP algorithm yielding a set of super-cones with 250 elements. The individuals of the initial population are represented by various MOCC partitions.

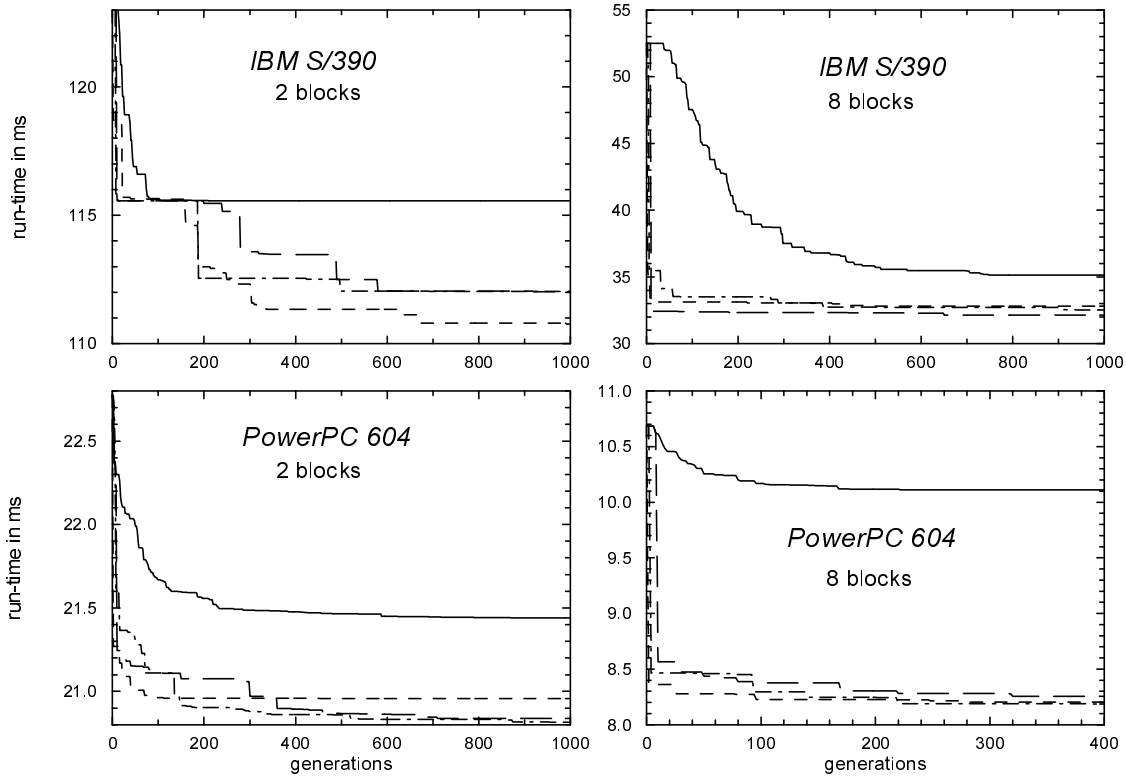

Figure 1: Predicted run-time corresponding to the best partition without using LSO (full line) and with using LSO: once each second generation (short-dashed line), 5 times each 10th generation (dashed line) and with an application probability of $p = 0.0055$ (dashed-dotted line).

Generally, in all investigated cases fitter individuals were found in shorter time if the LSO was included in the evolution process. The best results were reached if the LSO was applied with a probability of $p = 0.0055$ to the λ children of a generation (here we have $\lambda = 90$). In Fig. 1 2-block and 8-block partitions of our two hardware models are considered. Improvements of the run-time values from 3% (*PowerPC 604*, 2 blocks) up to 19% (*PowerPC 604*, 8 blocks) are possible. If the LSO is used more concentrated but with the same general probability (application of LSO only in selected generations), in some cases better results are possible (Fig.1: *IBM S/390*, 2 blocks – once each second generation; *IBM S/390*, 8 blocks – 5 times each 10th generation) than for an equally distributed application of the LSO. With a lower application probability of the LSO the effect of fitness reduction is smaller. If one further increases the probability the evolution process becomes too slow because of the complex time consuming character of the LSO and also the global view of the evolution process is reduced too strong.

Another way for the integration of the LSO into the evolution process is to apply it to the start individuals before execution of the EAs. The first generation becomes significantly fitter but the genetic diversity is reduced. So, it is much more complicated for the EAs to find still better individuals and the results strongly depend on the start individuals. On the other hand, if one appends an additional step of LSO application to

the final generation after breaking off the EAs, small and only small improvements are possible if individuals are near but not exactly in a local minimum.

5 CONCLUDING REMARKS

We have introduced a local search operator into Evolutionary Algorithms in the context of hierarchical model partitioning for parallel logic simulation. This operator is based on a modification of a classical iterative partitioning algorithm by FIDUCCIA-MATTHEYSES. According to our special partitioning problem we have developed a new gain structure which allows an effective application of this complex operator.

In experiments with two models of real processors, the use of this local search operator led to a significant reduction of the fitness (expected parallel simulation run-time) up to 19% in comparison to runs without use of the operator. Its application rate has to be chosen carefully to ensure the balance between the global view of the Evolutionary Algorithms and the local view of the search operator. In our experiments, an application rate of 0.55% has proven to be adequate.

ACKNOWLEDGMENT

THIS WORK WAS SUPPORTED BY DEUTSCHE FORSCHUNGSGEMEINSCHAFT (DFG) UNDER THE GRANT SP 487/1-2.

REFERENCES

[AK95] C. J. Alpert and A. B. Kahng. Recent Directions in Netlist Partitioning : A Survey. *INTEGRATION the VLSI Journal*, pages 1–81, 1995.

[Döh96] D. Döhler. *Entwurf und Implementierung eines parallelen Logiksimulators auf Basis von TEXSIM.* Diplomarbeit, Universität Leipzig, Fakultät für Mathematik und Informatik, 1996.

[FM82] C. M. Fiduccia and R. M. Mattheyses. A Linear-Time Heuristic for Improving Network Partitions. In *Proc. of the 19th Design Automation Conference, ACM/IEEE*, pages 175–181, 1982.

[Her96] K. Hering. Parallel Cycle Simulation. Technical Report 13(96), Department of Computer Science, University of Leipzig, Leipzig, Germany, 1996.

[HHPV97] R. Haupt, K. Hering, U. Petri, and T. Villmann. Hierarchical Model Partitioning for Parallel VLSI-Simulation Using Evolutionary Algorithms Improved by Superpositions of Partitions. In *Proc. of the 5th European Congress on Intelligent Techniques and Soft Computing (EUFIT'97), Volume 1*, pages 804 – 808. Verlag Mainz, 1997.

[HHV96] K. Hering, R. Haupt, and T. Villmann. Hierarchical Strategy of Model Partitioning for VLSI–Design Using an Improved Mixture of Experts Approach. In *Proc. of the Conference on Parallel and Distributed Simulation (PADS'96)*, pages 106–113. IEEE Computer Society Press, Los Alamitos, 1996.

[Hol75] J.H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[KL70] B. W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *Bell Systems Technical Journal*, 49(2):291–307, 1970.

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

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

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

[Zik92] D. S. Zike. Design Automation Data Base (DA-DB). 1992. IBM Austin.