DYNAMIC FITNESS FUNCTION FOR PARALLEL EVOLUTIONARY PARTITIONING ALGORITHMS IN THE CONTEXT OF PARALLEL LOGIC SIMULATION

Hendrik Schulze, Reiner Haupt and Klaus Hering
Universität Leipzig, Institut für Informatik,
Augustusplatz 10/11, D-04109 Leipzig Germany
Phone: +49-341-9732288, Fax: +49-341-9732209
email:{hendrik,haupt,hering}@informatik.uni-leipzig.de

ABSTRACT: Parallelization of logic simulation is a promising way to accelerate verification processes for whole processor designs. Partitioning of hardware models influences the efficiency of following parallel simulations essentially. Within the frame of a 2-level hierarchical partitioning scheme *Parallel Evolutionary Algorithms* are successfully applied using a *lazy communication* strategy. Introducing a *dynamic fitness function* we explicitly consider the different potential of balanced and unbalanced partitions for further improvement at the beginning of the evolution process. A cooling-off fitness term inside the dynamic fitness function yields an *implicit age of individuals*. Results are shown for the partitioning of a complete *IBM S/390* processor model.

KEYWORDS: evolutionary algorithms, partitioning, parallel programming, logic simulation, VLSI design

1 INTRODUCTION

The background of the present paper is given by simulation processes for functional design verification (Spruth 1989) 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. The time spent for extensive simulation processes which are necessary for the logic design of such systems can be drastically reduced by parallelization based on model partitioning. Parallel logic simulation processes where several simulator instances co-operate over a loosely-coupled processor system are realized by *parallelTEXSIM* (Döhler 1996) as parallel version of the cycle-based simulator *TEXSIM* [1]. Each instance simulates a part of a complex hardware model.
Basic elements for building partitions of a hardware model (Hering et al. 1996) 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 Ψ 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 (Haupt et al. 1997) which models the *run-time* t^{Ψ} of one parallel simulation cycle of the corresponding hardware model parts. This partition-dependent cost function t^{Ψ} 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 (PCS) (Hering 1996) is taken. According to this PCS model, the run-time t^{Ψ} is estimated by the sum of several periods which depend on the partition component Ψ_i, and a collective communication part which is the same for all components and which synchronizes the processors (partition components) in each simulation cycle. So, the run-time is determined by that processor (partition component) which consumes the biggest amount of simulation time for its component-dependent actions in one cycle.
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* (Hering et al. 1996). 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

[1] copyright by IBM

of super-cones is partitioned to the set of blocks using more complex algorithms. In Hering et al. (1996) we have successfully applied *Evolutionary Algorithms* (EAs) to the second hierarchy level. To achieve better partitioning results in shorter time, we have integrated the concept of *superposition* (Haupt et al. 1997) into the evolution strategy. Introducing a *local search operator* (Haupt et al. 1998) based on an adapted iterative partitioning algorithm of FIDUCCIA-MATTHEYSES (FM) (Fiduccia & Mattheyses 1982) we have enriched our EAs to a hybrid technique.
For our special partitioning aims we have extended the sequential EAs by the widely used *multiple subpopulation approach* (Fogel 1995, Mühlenbein et al. 1988) to the *Parallel Evolutionary Partitioning Algorithms* (PEPAs). Thereby the basic population P is divided into subpopulations P_i where the communication is realized as migration of individuals during the evolution. We have implemented various communication structures between subpopulations and different communication rates. Introducing the concept of *lazy communication* which avoids waiting phases for all subpopulations, the evolution process is accelerated without loss of quality. Experimental results concerning PEPAs are shown in Schulze et al. (1999).
For our PEPAs the *fitness function F* results from the underlying simulation background and is identified by the expected run-time t^{Ψ} which is to be minimized. Hence, this fitness function F only distinguishes between partitions (individuals) with different expected run-times. But, partitions with the same fitness function can be very different in genotype. On the one hand, if the expected simulation time of the dominating partition component is nearly equal to those of all the other components, a partition is balanced, on the other hand the nondominating components of a second partition with the same run-time can be almost empty. Hence, the second partition has a much higher potential for leading to better partitions in the following evolution process. Cones can be redistributed from the dominating component to the nondominating ones. Consequently, we have added a second generation-dependent fitness term F' to our fitness function F. This supplementary fitness term F' gives a measure for the sum of the expected simulation times of all partition components and, additionally, is cooled off. Hence, at the beginning of the evolution process the new combined *dynamic fitness function* F^{dyn} (Schulze 1998) prefers partitions which are unbalanced to have a higher potential for further improvement. For later generations the dynamic fitness function tends to our conventional fitness function F selecting the real best partition with lowest expected run-time for a corresponding model simulation. Experimental results show that better partitions are produced applying our new dynamic fitness function F^{dyn}.
In Section 2 the dynamic fitness function F^{dyn} is introduced and the additional fitness term F' is explained. Section 3 describes our Parallel Evolutionary Partitioning Algorithms based on a lazy communication scheme. Experimental results are discussed in Section 4. Finally, conclusions are given in Section 5.

2 DYNAMIC FITNESS FUNCTION

The fitness function F of an individual for our PEPAs is identified by the expected run-time t^{Ψ} of one parallel simulation cycle of the corresponding n hardware model parts depending on a special partition Ψ. The expected run-time t^{Ψ} of a partition Ψ is determined by the maximum of the simulation time t_i^{Ψ} for all partition components Ψ_i. For each component Ψ_i as subset of the basic cone set *Co(M)* a definite amount of expected simulation time for the corresponding model part is to be calculated (Haupt et al. 1997). Because of the synchronizing character of the collective communication action which is part of the simulation cycle, the run-time is determined by the most time-consuming partition component independent on the other partition components.
Both partitions in Fig. 1 have the same run-time and, consequently, are equally valuated in PEPAs by the fitness function F. The usual PEPAs cannot distinguish between partition 1 and 2 in Fig. 1. Comparing these two partitions, the block-dependent simulation time t_i^{Ψ} of partition 2 is much more homogeneously distributed than t_i^{Ψ} of partition 1. Contrary to partition 2, the probability, that for partition 1 a redistribution of cones from the partition component $\Psi_1^{(1)}$ to other components $\Psi_i^{(1)}$ with i >1 leads to a lower maximum of simulation time t_i^{Ψ}, is much higher. So, the potential of improving run-time t^{Ψ} during the further evolution process is much larger for partition 1 than for partition 2.

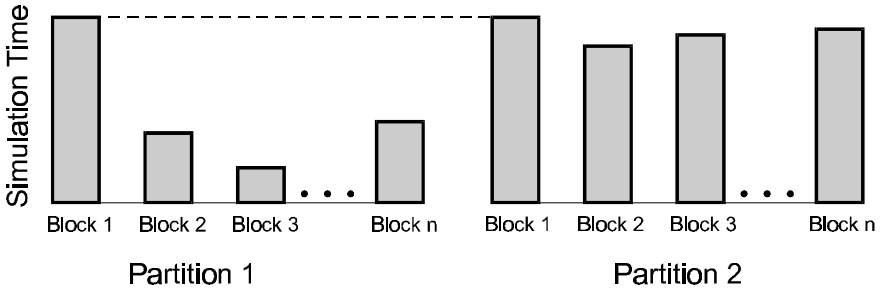

Figure 1: Block-dependent simulation time t_i^{Ψ} for two different partitions (individuals) with the same run-time t^{Ψ}.

Therefore, in addition to the conventional fitness function F

$$F = t^{\Psi} = \max_i t_i^{\Psi} \tag{2.1}$$

we have introduced a new term F'

$$F' = \sqrt{\sum_{i=1}^{n} \left(t_i^{\Psi}\right)^2} \quad . \tag{2.2}$$

Comparing two partitions with equal fitness F (Fig. 1) the additional fitness term F' is lower for a partition with more unbalanced components (partition 1). In contrast to the conventional fitness F, not only the worst partition component but all partition components influence the fitness F' of an individual. If both fitness terms, F and F', are integrated into the valuation of an individual, both minimal run-time and inhomogeneity of the block simulation time are considered for the selection step. In a later phase of the evolution process the conventional fitness term F should be the dominating term because, ultimately, the expected run-time for parallel simulation is the crucial criterion for choosing the best partition. Therefore, the dynamic fitness function F^{dyn} combines both fitness terms F and F' in such a manner that the new term F' is cooled off:

$$F^{dyn} = \frac{1}{10} \frac{g_{\max}}{g} F' + F \quad . \tag{2.3}$$

For our partitioning problem, F' is cooled off by the term $^1/_{10}\ g_{max}/g$ (g_{max} maximum number of generations, g current generation).

The cooling procedure results in two consequences. On the one hand the term F' looses influence on F^{dyn} in the progression of evolution. On the other hand, F^{dyn} valuates the same individual at a later moment better than at the beginning of evolution (because of the cooling off term $^1/_{10}\ g_{max}/g$).

Each individual is valuated at the moment of its origin. If it is fit and survives the next generations, the fitness of this individual would be improved by the cooling-off fitness term F'. But our strategy is, not to change the fitness value of this surviving individual. So, a new individual with the same genotype as the old one gets a better fitness value than the old individual and can displace it. Hence, the cooling-off fitness term inevitably leads to an *implicit age of individuals*. In fact, good individuals survive only 4 to 6 generations because after this time new individuals are valuated with a better fitness, even if they have a worse run-time F. In this way it is possible to ensure, that old suboptimal individuals cannot restrict the search space and slow down the evolution process. But on the other hand, a continuity of evolution is preserved and a deterioration of the fitness is hardly possible. Thus, our dynamic fitness function yields a well balanced strategy between a too strong preference of suboptimal solutions and a too mighty exploration rate. In this context other strategies, like the elitist model or an explicit individual age implementation, become dispensable.

3 PARALLEL EVOLUTIONARY PARTITIONING ALGORITHMS

A *multiple subpopulation approach* is a very natural extension of the usual Evolutionary Partitioning Algorithms. Using the same multi-processor system where the corresponding parallel logic simulation will be done, up to n subpopulations can be created for PEPAs. The basic population P is divided into subpopulations P_i which exist independently from each other, but have more or less communication between each other during the evolution. We have implemented many different communication structures (ring, hypercube, star, all-to-all) and strategies (static or dynamic communication, cooling type, communication rate) to realize communication between the subpopulations.

Our Parallel Evolutionary Partitioning Algorithms have three advantages:

- good results can be found much faster,
- better results can be found, and
- evolutionary search converges more reliable.

Communication between subpopulations is realized as migration of the best individuals. Migrated individuals are not integrated into the new subpopulation, they only get the chance to translate their genetic code by proliferation. It has been shown that best results can be reached, if only 1%-10% of the individuals migrate in each first up to 5th generation and if the subpopulations are arranged in a ring topology. If communication is too strong, some suboptimal individuals can dominate all subpopulations and slow down the evolution.

Our PEPAs include a special kind of asynchronous communication, a *lazy communication*. In the case of synchronous communication, a heterogeneous workstation cluster or extrinsic processes on processors where subpopulations are handled can lead to waiting subpopulations. Lazy communication avoids such waiting periods, because no

subpopulation has to wait for another one. Using nonblocking send and receive functions of the MPI standard[2] it is possible to initiate communication and continue the evaluation without waiting for termination of the communication action. Before starting a new communication (sending individuals from one subpopulation to another) the success of the last communication to the same subpopulation is requested. If the last communication was not yet successful, the new communication action is rejected and evaluation is going on without sending individuals. In Fig. 2 synchronous and lazy communication are confronted for 2 subpopulations.

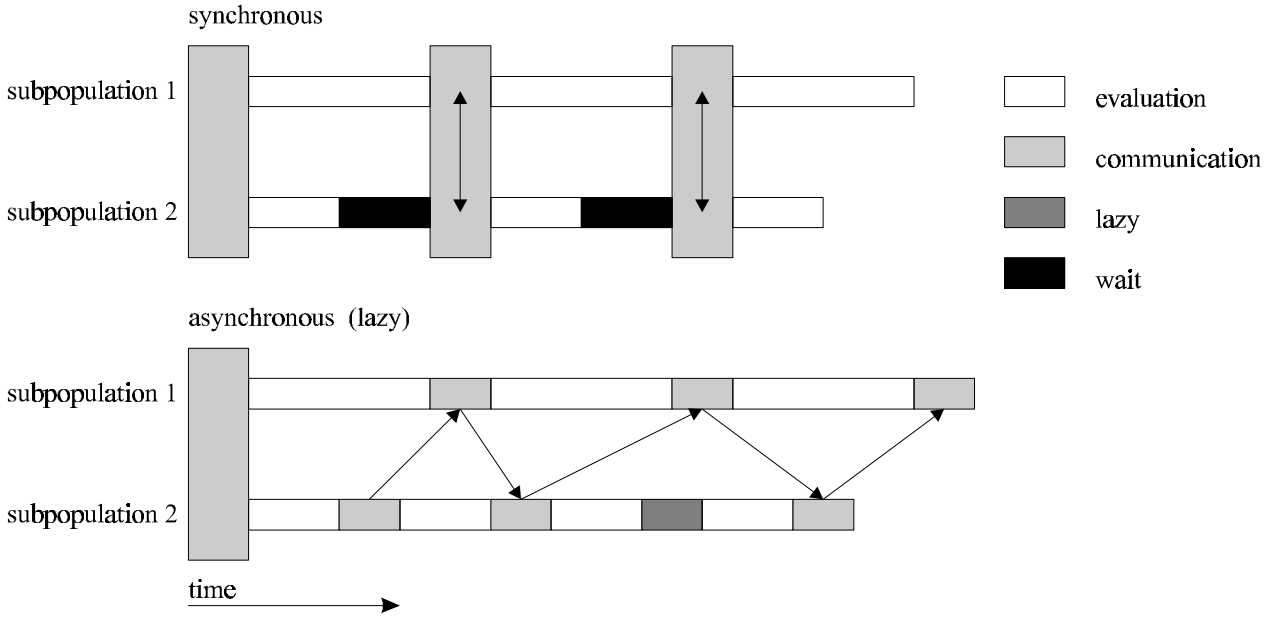

Figure 2: Different kinds of communication: synchronous versus asynchronous (lazy).

For the lazy communication, subpopulation 2 gives an example of a communication action where no individual is sent. Consequently, a few migrating individuals may be lost, but the evolution process is accelerated. Moreover, experiments show that the evolution results are not worsened.

4 EXPERIMENTAL RESULTS

Applying our 2-level hierarchical partitioning strategy to a hardware model representing an *IBM S/390* processor we have compared the static and dynamic fitness function as well as the sequential and parallel Evolutionary Partitioning Algorithms.

For the first level in the hierarchical strategy we use the simple but for this hardware model proper STEP[3] partitioning algorithm to partition the set of cones ($|Co(M)|$ = 737 606) to 250 super-cones. The individuals of the initial population are represented by various partitions created by MOCC algorithm (Hering et al. 1996). All following results are shown for a partitioning into 15 blocks (model parts).

The population of the sequential Evolutionary Algorithms comprises 40 individuals and 100 descendants using a mutation probability of 0.001 and 3 crossing points for crossover. In Fig. 3(a) the fitness (expected run-time t^{Ψ}) for the best individual (partition) of two populations is drawn where the evolution of one population valuated by the static fitness function F is compared to another population valuated by the dynamic fitness function F^{dyn}. For equal generation indices the expected run-time t^{Ψ} is improved up to 1.7 *ms* (8%) by the dynamic fitness function. Moreover, experimental results show, that only the combination of the dynamic fitness function with the implicit age approach causes the mentioned improvement. If any feature was used exclusively, no significant improvement could be reached.

Parallel Evolutionary Algorithms are tested using 8 subpopulations which are connected by a ring structure. Any single subpopulation contains 40 individuals. Communication can occur each 3rd generation if the lazy concept enables it. For every communication action 5 individuals are selected for migration to the other subpopulations. In Fig. 3(b) the fitness of the best individual is presented analogously to Fig. 3(a) but for the parallel version of the EPAs. The expected run-

[2] Message Passing Interface - international standard for development of parallel programs

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

time values t^{Ψ} of the best partition as result of the parallel EPAs are generally up to 1.5 *ms* better than those of the sequential EPAs. On the other hand, results with similar quality can be received in much shorter partitioning time using PEPAs. As in the sequential case, the introduction of the dynamic fitness function into the parallel EPAs leads to a further significant improvement of the expected run-time up to 1.2 *ms* (Fig. 3(b)).

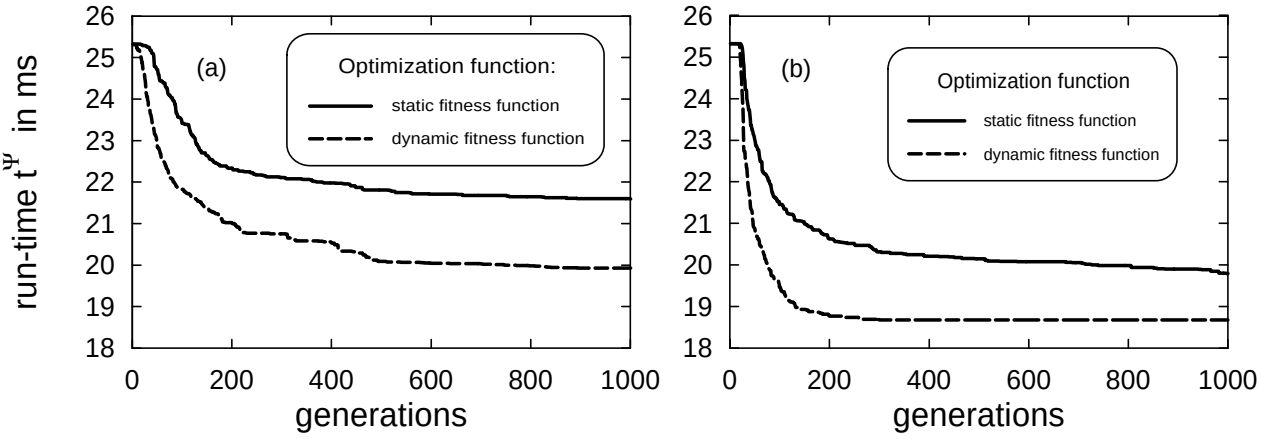

Figure 3: Expected run-time t^{Ψ} of the best partition for sequential (a) and parallel (b) Evolutionary Partitioning Algorithms.

Both, the application of the dynamic fitness function and of the parallel version of the EPAs drastically reduce the partitioning time spent for EPAs and, additionally, the expected run-time t^{Ψ} of the best partitions (individuals).

5 CONCLUSIONS

We have parallelized Evolutionary Algorithms in the context of hierarchical model partitioning for parallel logic VLSI-simulation. Our Parallel Evolutionary Partitioning Algorithms were enriched by a special type of asynchronous communication - lazy communication. In this way, the evolution process has been drastically accelerated and better expected run-time results for corresponding parallel simulations have been obtained. In addition to the usual static fitness function a dynamic term is applied. Especially for early generations, this leads to a better chance for unbalanced partitions to introduce their genetic code. The quality of the best partitions concerning the expected run-time could be enhanced again both in the sequential and the parallel case of Evolutionary Partitioning Algorithms.

ACKNOWLEDGMENT

THIS WORK WAS SUPPORTED BY DEUTSCHE FORSCHUNGSGEMEINSCHAFT (DFG) UNDER THE GRANT SP 487/1-3.

REFERENCES

Döhler, D., 1996, “Entwurf und Implementierung eines parallelen Logiksimulators auf Basis von TEXSIM”, Diplomarbeit, Universität Leipzig, Fakultät für Mathematik und Informatik, Germany.

Fiduccia, C.M.; Mattheyses, R.M., 1982, “A Linear-Time Heuristic for Improving Network Partitions”, Proc. of the 19th Design Automation Conference, ACM/IEEE, pp. 175 - 181.

Fogel, D.B., 1995, “Evolutionary Computation: Towards a New Philosophy of Machine Intelligence”, IEEE Press, Piscataway/NJ, USA.

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

Haupt, R.; Hering, K.; Siedschlag, T., 1998, “Integration of a Local Search Operator into Evolutionary Algorithms for VLSI-Model Partitioning”, Proc. of the 6th European Congress on Intelligent Techniques and Soft Computing (EUFIT'98), Volume 1, Verlag Mainz, Germany, pp. 377 - 381.

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

Hering, K.; Haupt, R.; Villmann, T., 1996, “Hierarchical Strategy of Model Partitioning for VLSI-Design Using an Improved Mixture of Experts Approach”, Proc. of the Conference on Parallel and Distributed Simulation (PADS'96), IEEE Computer Society Press, Los Alamitos, USA, pp. 106 - 113.

Mühlenbein, H.; Gorges-Schleuter, M.; Krämer, O., 1988, “Evolution Algorithm in Combinatorial Optimization”, Parallel Computing, (7), pp. 65 – 88.

Schulze, H., 1998, “Entwicklung, Untersuchung und Implementierung von Algorithmen für die Modellpartitionierungskomponente parallelMAP”, Diplomarbeit, Universität Leipzig, Fakultät für Mathematik und Informatik, Germany.

Schulze, H.; Haupt, R.; Hering, K., 1999, “Experiments in Parallel Evolutionary Partitioning”, Proc. of the Conference on Parallel Computing (ParCo’99), accepted for publication.

Spruth, W.G., 1989, “The Design of a Microprocessor”, Springer-Verlag, Berlin, Germany.