Experiments in Parallel Evolutionary Partitioning

Hendrik Schulze, Reiner Haupt, Klaus Hering
Universität Leipzig, Institut für Informatik,
Augustusplatz 10-11, D-04109 Leipzig, Germany
email:{hendrik,haupt,hering}@informatik.uni-leipzig.de

Abstract

Parallelization of logic simulation based on a replicated worker principle allows significant acceleration of functional verification in microprocessor design. Corresponding hardware model partitioning itself is an interesting subject for parallelization. In this paper we consider the application of Parallel Evolutionary Algorithms in the framework of a 2-level partitioning scheme. Applying a multiple subpopulation approach that permits various communication strategies we have reached promising experimental partitioning results for IBM S/390 processor models with more than 2 million elements at register-transfer and gate level.

1 Introduction

Due to challenging technological capabilities the complexity of VLSI designs is growing rapidly. Therefore, the use of verification tools in all design phases is unavoidable. Simulation is a very important but time-extensive verification method for the logic design of microprocessor structures [Spr89]. Based on the message-passing paradigm, in [HHV96] we describe a parallelization approach concerning particular logic simulation processes for functional verification. Within a parallel simulation, several co-operating simulator instances run on a loosely-coupled system, each of them handling a part of a complex hardware model (replicated worker principle). The efficiency of the parallel simulation is essentially influenced by the corresponding model partitioning which is related to a NP-hard graph partitioning problem. Due to the complexity of processor models under consideration and a special cost function based on a formal model of parallel cycle simulation [Her96] we generally apply a 2-level partitioning strategy. Thereby, model partitioning appears as bottom-up clustering starting from possibly overlapping elementary model parts called cones. During a fast prepartitioning step cones are combined to super-cones. Prepartitioning represents a compromise between the advantage of reducing the problem complexity for the following partitioning step and a restriction of the solution space that lets one expect suboptimal partitioning results. Within our strategy prepartitioning establishes the precondition for the successful application of *Evolutionary Algorithms* [Rec73] in final partitioning.

In Chapter 2 we describe the application of Evolutionary Algorithms in the field of model partitioning. Especially, a complex partitioning example is considered and a short

retrospect concerning our previous work on Evolutionary Algorithms is provided. The next chapter outlines advantages of the *multiple subpopulation approach* which naturally leads to *Parallel Evolutionary Partitioning Algorithms* (PEPAs). These algorithms are subject of Chapter 4. Selected experimental results concerning migration strategies and a special asynchronous communication mechanism called *lazy communication* are presented. Finally, in Chapter 5 conclusions are given.

2 Evolutionary Partitioning Algorithms

Using Evolutionary Algorithms we apply principles of the biological evolution theory to the field of computer science to tackle combinational optimization problems. Our *Evolutionary Partitioning Algorithms* (EPAs) work over a set (population) of individuals, which are represented by their genetic code describing a hardware model partition. Each evolution cycle starts with recombination (crossover) where pairs of individuals produce a set of offspring by reproducing and mixing their genetic code fragments. Additionally, the genetic code is changed with a weak probability by a mutation operator. The new individuals are evaluated by a *fitness function* which estimates the *parallel simulation time* per clock-cycle for the corresponding partitioned processor model. The best individuals, resulting from the offspring and a part of the parent generation, are selected to establish the following generation. During the evolution process the population improves its average fitness and converges towards the global optimum.

We use a straightforward coding of individuals (see Figure 1), where each gene represents a super-cone coded by an integer as block index.

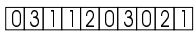

Figure 1: Representation of a partition with 10 super-cones and 4 blocks as an individual.

So, each individual has the same length (number of genes) and each gene has the same range of valid integer values, which guarantees the creation of valid individuals applying the recombination operator (see Figure 2).

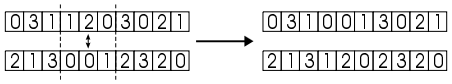

Figure 2: Recombination of two individuals exchanging 3 genes.

Due to the complexity of processor models under consideration the EPAs are embedded in a *2-level partitioning strategy* [HHV96] in the context of our special partitioning problem. Combining cones as basic elements to super-cones at the first partitioning level (prepartitioning step) the problem complexity is drastically reduced but also the possibility to find acceptable solutions is restricted. At the first level, a fast and effective algorithm (STEP) is applied which makes use of special knowledge concerning the internal representation of

the VLSI models so that especially a lot of unfavourable partitioning solutions are excluded. Based on this prepartitioning the number of elements to be partitioned is enormously reduced, from up to millions of cones at the first level to hundreds or thousands of super-cones at the second level.

At the second level super-cones are merged to a final partition, the set of blocks. Our partitioning aim (minimization of the partition-bound parallel simulation time) is related to a complicated cost function of a partitioning applied to a hypergraph representing a combination of an overlap and communication hypergraph whose nodes are identified by the super-cones [HHPV97]. Because of this complicated cost function, conventional partitioning tools as for instance METIS [KK95] and PARTY [PD96] are not applicable for our special partitioning procedure. An intuitive way to get an acceptable partition at the second level is to produce a set of random partitions as initial population for the application of EPAs using the cost function as fitness function. But, the fitness landscape considered here has many local minima and is rather jagged. Our intention was to include special model knowledge using several parameterized initial partitioning algorithms like MOCC, nBCC and STEP (see [HHV96]) to prepare an initial population with many rather good but concerning its genetic variety very different individuals. Hence, the partition quality at the beginning of the evolution is already a higher one than using random partitions. Simultaneously, we have an almost homogeneous distribution of individuals inside of the search space which leads to a high improvement potential for evolution.

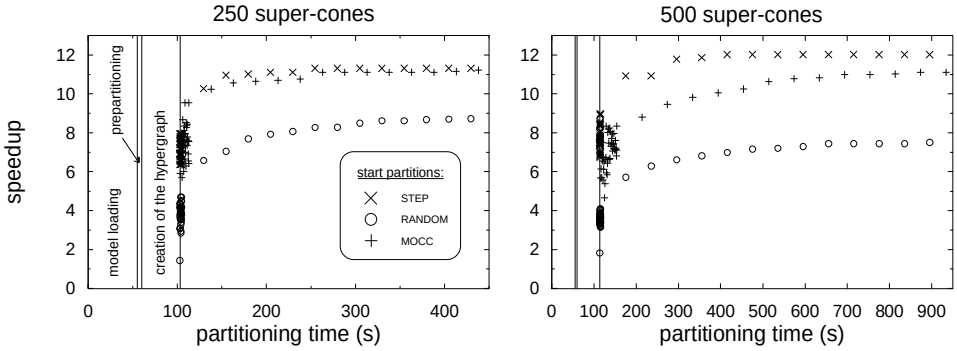

Figure 3: Complete partitioning procedure (model loading, prepartitioning, hypergraph creation, production of start population, EPAs) for 2 different numbers of super-cones and 3 different algorithms (STEP, RANDOM, MOCC) creating start partitions with 16 blocks at the second level. For all partitions of the initial population and the best individual per each 50th generation the speedup dependent on the partitioning time is shown.

In Figure 3 examples for a full partitioning procedure[1] in dependence on the time consumption are shown which include a constant phase comprising model loading, a short prepartitioning step and the creation of the hypergraph. The relative time slice for these three phases is reduced with an increasing number of super-cones. For the two diagrams of

[1]concerning an IBM S/390 processor model with 2,7 million elements on register - transfer and gate level

Figure 3 partition-bound speedup[2] values yielded by three different partitioning procedures are shown, each with 40 individuals using STEP, MOCC and RANDOM algorithms for the initial population, respectively. In comparison to the evolution phase, which consumes the major part of overall partitioning time, the production of the initial population is very fast (symbol cloud after hypergraph creation). For both diagrams, using knowledge-based algorithms (MOCC, STEP) the partitions of the initial populations are significantly better than those produced by RANDOM. Additionally, the partitions of initial populations based on MOCC and STEP provide an improvement potential by EPA application similar to that of randomly produced partitions. For all further experiments we use 250 super-cones, 16 blocks and the MOCC algorithms to generate the initial population. Furthermore, the processor model mentioned above is chosen in all cases.

Alternatively to EPAs as collective optimization methods, many single partitions can be produced and improved independently from another, followed by the choice of the best partition for a succeeding parallel simulation. For our purposes, we have developed iterative improvement algorithms such as TABU SEARCH and a FIDUCCIA-MATTHEYSES –like algorithm [Sie98] which handle single partitions. In contrast to EPAs, these algorithms strongly depend on the initial partition. Hence, if one performs a definite number of trials (containing an initial and improvement partitioning algorithm) no guaranty is given to get a final partition with acceptable quality. This main disadvantage is overcome using EPAs. Especially due to the effect of the recombination operator (see Figure 2), EPAs almost always provide a sufficiently good final partition if the initial population is big and genetically diverse enough.

To achieve better partitioning results in shorter time using EPAs, we have integrated the concept of *superposition* [HHPV97] into the evolution strategy. Introducing a *local search operator* [HHS98] based on an adaptation of an iterative partitioning algorithm of FIDUCCIA-MATTHEYSES we have enriched our EPAs to a hybrid technique. Both features work effectively if the fitness function is very complex with many local minima as it is in our case. Using a *dynamic fitness function* [SHH99] which includes an additional generation-dependent term favouring partitions with a higher potential of improvement especially at the beginning of the evolution, also better results are obtained.

In addition to that, a promising way to increase the probability to find a solution near the global optimum in a rich-structured fitness landscape is to broaden the population concerning the search space using a multiple subpopulation approach with an extended number of individuals.

3 Multiple Subpopulation Approach

The multiple subpopulation approach is based on the island model, a population genetic approach. Therein the individuals of one big population are distributed to n subpopulations. For each subpopulation a local evolution is computed. After a dedicated number of generations a few individuals can migrate to other subpopulations and translate their genetic code by proliferation. These migrating individuals are not integrated into the other subpopula-

[2]If not stated otherwise, throughout the paper with *speedup* the ratio between sequential simulation time (concerning the complete hardware model) and partition-bound parallel simulation time is meant (to express the partition quality).

tion but can leave some genetic improvements by creating offspring with an individual from the target subpopulation. Applying this approach it is possible to get better optimization results than with one big population. This can be explained by the fact that a suboptimal individual at early evolution time can dominate the complete population and force the other individuals to remain in its surrounding. In our approach the isolation of the subpopulations and selective migration evade the mentioned disadvantage. Using the multiple subpopulation approach acceptable results can be found in shorter time, better results are found and the evolution converges with more reliability.

4 Parallel Evolutionary Partitioning Algorithms

Local evolution processes within a multiple subpopulation approach give a natural way to parallelize Evolutionary Partitioning Algorithms, using the same loosely-coupled system as target hardware where after partitioning the corresponding parallel logic simulation will be done. The subpopulations can be distributed to the several nodes of the parallel hardware. Our implementation of the Parallel Evolutionary Partitioning Algorithms [Sch98] (PEPAs) realizes migration by message passing, using a MPI library.

Several subpopulation approaches differ especially in their migration strategies. The PEPA implementation includes data structures which facilitate a dynamic and highly effective communication and allow to adjust all parameters relevant for migration, e.g. the number of migrating individuals, the communication structure between the single subpopulations and the period between migrations. Every possible communication topology can be combined with any number of migrating individuals. In this context we have introduced a communication matrix, which is configured at program start. From this communication matrix a send and receive vector are extracted which control the communication behaviour.

Topology	Speedup after generation	
	250	500
Ring	**11,360**	**11,387**
dRing	11,198	11,277
Matrix	11,174	11,380
dMatrix	11,151	11,364
Star	10,879	11,287
dStar	11,033	11,341
AllToAll	10,680	11,166

Number of migrating individuals	Time period between migrations (counted in generations)				
	1	5	10	25	50
Speedup after generation 250					
1	10,972	10,821	10,761	10,596	10,528
5	**11,031**	10,859	10,726	10,631	10,598
10	10,941	10,989	10,907	10,619	10,576
20	10,952	10,958	10,961	10,828	10,695
Speedup after generation 500					
1	11,077	10,963	10,983	10,884	10,921
5	11,082	11,077	10,983	10,883	10,921
10	11,019	**11,339**	10,998	11,063	10,822
20	10,965	11,308	10,977	10,941	10,958

Table 1: Speedup related to special migration topologies, where dRing, dMatrix and dStar mean the double - connected topology version (left table). Speedup for selected migration strategies concerning migration frequency and number of migrating individuals (right table).

Based on our PEPA implementation we have examined a great variety of communication strategies. In the course of our experiments[3] the best partitioning results have been reached using a loosely connected communication structure (e.g. ring) and a static migration of 1%-30% of the individuals after every 1st up to 5th generation. Experimental results for examples with 8 subpopulations (each containing 40 individuals) are given in Table 1. Generally, if more individuals migrate they should do it in longer intervals. If communication is too strong, some suboptimal individuals can dominate all subpopulations and slow down the whole evolution. Compared to the huge expense for executing the fitness function (95% of the PEPA runtime) the communication overhead nearly disappears, a fact which is very favourable for parallel computing.

Our PEPAs include a special kind of asynchronous communication called *lazy communication*. Load influences and the heterogeneity of a workstation cluster can cause PEPAs to

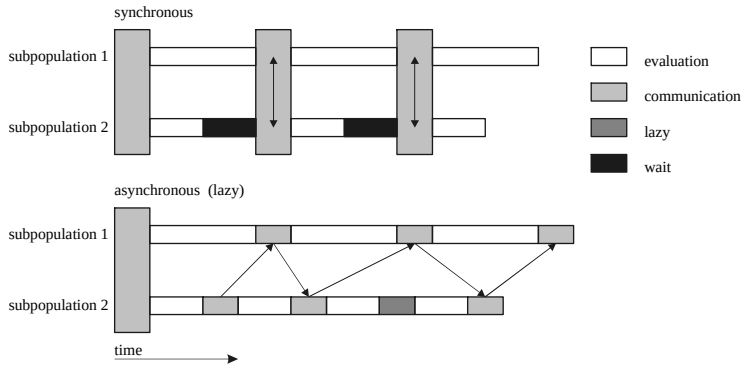

Figure 4: Synchronous vs. lazy communication.

wait at communication points. Lazy communication avoids such waiting periods by using nonblocking send and receive functions of the MPI-library. 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 has not yet been successful, the new communication is rejected and evolution is going on without sending individuals (cf. Figure 4).

communication loss	0%	5%	10%	20%	30%	40%	50%	60%	70%
average speedup	12,26	12,3	12,28	12,32	12,38	11,56	11,35	11,12	11,02

Table 2: Dependence of the partition quality on communication loss.

Experiments show that evolution results don't get worse, if up to 35% of all communication actions fail (see Table 2). The lazy communication approach allows to use heterogeneous workstation clusters efficiently without implementing dynamic load balancing algorithms (see Figure 5).

[3]About 8000 experiments have been done on an IBM SP2-Parallel Computer using up to 112 nodes. Every experiment has been done 16 times with a different initialization of the random number generator.

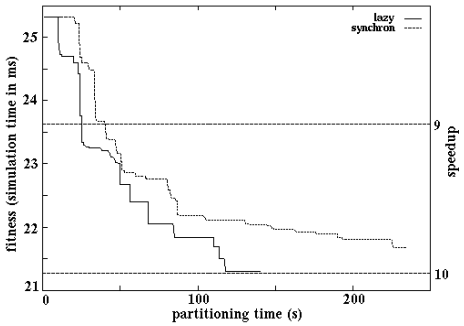

Figure 5: With 1 of 8 nodes slowed down by 50%, PEPAs on a double-connected ring topology using lazy communication reach the same fitness value as in the synchronous case in shorter time.

For the same IBM S/390 processor model as used in Figure 3 PEPAs resulted in partitions which were about 10%-20% better (expressed in terms of the fitness function) than related EPA results. Compared to EPAs, generally it took PEPAs significantly less partitioning time to produce final partitions (cf. Figure 6).

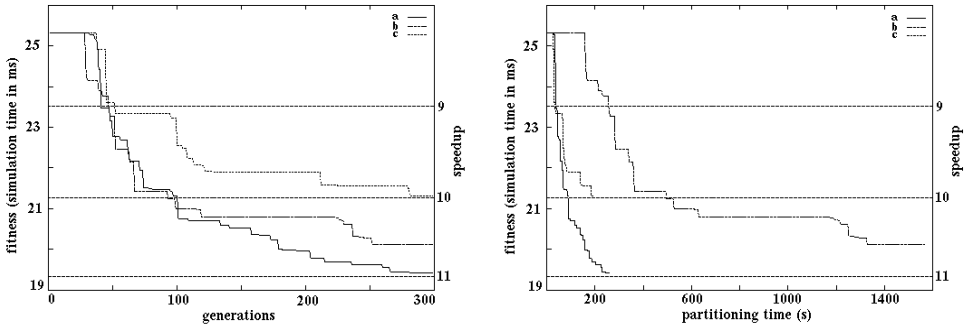

Figure 6: Partitioning results with a) PEPAs (8 subpopulations with 40 individuals and 100 children each) b) EPAs (320 individuals and 800 children) c) EPAs (40 individuals and 100 children) related to the number of generations and the partitioning time.

5 Conclusions

In the framework of a hierarchical strategy we have successfully applied Evolutionary Algorithms to the field of model partitioning for logic simulation in VLSI design. Based on a multiple subpopulation approach, in this paper we have presented Parallel Evolutionary Partitioning Algorithms taking advantage of dynamic communication structures and a special type of asynchronous (lazy) communication. Because of the small communication overhead and the potential of lazy communication these algorithms work very efficiently, even on a

heterogeneous workstation cluster. By means of an IBM S/390 processor model we have exemplified that the parallel approach leads to better partitioning results in significantly shorter time.

References

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

[HHS98] R. Haupt, K. Hering, and Th. Siedschlag. Integration of a Local Search Operator into Evolutionary Algorithms for VLSI-Model Partitioning. In *Proc. Of the 6th European Congress on Intelligent Techniques and Soft Computing (EUFIT'98), Volume 1*, pages 377 – 381. Verlag Mainz, 1998.

[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.

[KK95] G. Karypis and V. Kumar. MeTiS: Unstructured Graph Partitioning and Sparse Matrix Ordering System. 1995.

[PD96] R. Preis and R. Diekmann. The PARTY Partitioning-Library, User Guide - Version 1.1. Technical Report tr-rsfb-96-024, Univ. of Paderborn, Germany, 1996.

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

[Sch98] H. Schulze. *Entwicklung, Untersuchung und Implementierung von Algorithmen für die Modellpartitionierungskomponente parallelMAP.* Diplomarbeit, Univ. Leipzig, Inst. für Informatik, 1998.

[SHH99] H. Schulze, R. Haupt, and K. Hering. Dynamic Fitness Function for Parallel Evolutionary Partitioning Algorithms in the Context of Parallel Logic Simulation. In *7th European Congress on Intelligent Techniques and Soft Computing (EUFIT'99)*, 1999. accepted for publication.

[Sie98] T. Siedschlag. *Iterative Modellpartitionierungverfahren für die parallele Logiksimulation.* Diplomarbeit, Univ. Leipzig, Inst. für Informatik, 1998.

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