TECHNISCHE
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK

Sonderforschungsbereich 342:
Methoden und Werkzeuge für die Nutzung
paralleler Rechnerarchitekturen

Partitioning VLSI-Circuits for Parallel Simulation on Transistor Level

Norbert Fröhlich, Rolf Schlagenhaft and Josef Fleischmann

TUM-I9719
SFB-Bericht Nr. 342/14/97 A
April 97

TUM–INFO–04-I9719-50/1.–FI

Alle Rechte vorbehalten
Nachdruck auch auszugsweise verboten

©1997 SFB 342 Methoden und Werkzeuge für
die Nutzung paralleler Architekturen

Anforderungen an: Prof. Dr. A. Bode
Sprecher SFB 342
Institut für Informatik
Technische Universität München
D-80290 München, Germany

Druck: Fakultät für Informatik der
Technischen Universität München

Partitioning VLSI-Circuits for Parallel Simulation on Transistor Level

Norbert Fröhlich, Rolf Schlagenhaft and Josef Fleischmann
Institute of Electronic Design Automation,
Technical University of Munich, 80290 Munich, Germany
Email: {froehlich,schlagenhaft,fleischmann}@regent.e-technik.tu-muenchen.de

Abstract

For parallel simulation of VLSI circuits on transistor level a sophisticated partitioning of the circuits into subcircuits is crucial. Each net connecting the subcircuits causes additional communication and computation effort. As the slave processors simulating the subcircuits advance synchronously in time, the computation effort for each subcircuit should be approximately the same. In this paper we present a new approach for partitioning VLSI circuits on transistor level yielding a low number of interconnects and balanced subcircuit sizes. Simulation of industrial circuits using this partitioning is up to 41% faster than with other known partitioning approaches for parallel analog simulation.

1 Introduction

While designing VLSI-circuits time critical parts have to be simulated on transistor level. At Siemens research laboratories, Munich, which we cooperate with, a major simulation application on transistor level is the validation of critical paths with more than 10,000 transistors. These critical paths are extracted from dynamic memories and digital circuits.

Due to the increasing size of VLSI circuits and the required accuracy and reliability of the simulation, the computation effort for circuit simulation is rising. To keep pace with this trend the applied methods for analog simulation like implicit numerical integration are traditionally vectorized and implemented for supercomputers. At Siemens' research laboratories, the analog simulator TITAN was developed. Vectorizing TITAN achieved average vectorization speedup factors of 12 compared to scalar computation on a Siemens S200 vector computer [1].

Nowadays the hardware environment is changing. The floating point performance of modern workstations approaches those of small vector computers. These workstations are typically connected by networks like Ethernet or FDDI. As these combined workstations are very powerful, it makes sense to utilize this power by running parallel software on the cluster [2]. But there is still one major problem using a workstation cluster for running parallel programs: The transmission performance in terms of bandwidth and latency of the network is still relatively low and limited by the underlying hardware and communication protocols, i.e. software. For example, the bandwidth of Ethernet is only 10 megabit per second.

Another interesting hardware platform is a multi-processor computer with distributed memory architecture and several processors communicating via message passing over a fast media. Such computers are much cheaper than vector computers and benefit from the raving progress going on in development of computers for big markets, because processors and other standard components of typical workstations are used. The major advantage of these parallel computers compared to a workstation cluster is a much better communication performance between the processors. But still, for good analog parallel simulation performance, a low number of nets interconnecting the subcircuits, which reduces the synchronization effort, is essential.

Parallel computing is promising for achieving speedups against sequential computing and for using the memories of several computers simultaneously, which may release from the need for an expensive special computer equipped with huge memory.

Therefore, the analog simulator TITAN, which was developed at Siemens' research laboratories, was parallelized by applying domain decomposition techniques [3]. These methods split the problem in the physical domain. Each domain is calculated by a separate process and the entire problem can be solved in parallel. The boundaries of the domains are synchronized by a master process running sequentially. In TITAN, numerical integration is performed by an optimized Newton method, the multi-level Newton method with latency [4]. For circuits with a highly parallel structure, as for example a ring oscillator a close to linear speedup of 8 on 8 processors is achieved. This demonstrates that there is only little overhead for parallel execution of the simulation.

During development of the parallel extensions to TITAN some test circuits have been partitioned by hand, but for practical use of the parallel simulation it is crucial to be able to partition the circuits automatically.

For parallel simulation a partitioning is required which yields balanced partition sizes and only few interconnects. Each net interconnecting the partitions causes communication between the slave processes and the synchronizing master process. The master process simulates the interconnection network between all partitions, i.e. subcircuits, and each additional interconnect net rises the computation effort for this time critical calculation. The slave processes simulating the subcircuits advance synchronously in time. Therefore, the whole simulation progress is determined by the slowest slave process. Maximum progress is achieved, if all slave processes have the same simulation load. This requires subcircuits of approximately the same size.

1.1 Previous Work

One of the first published algorithms for partitioning on transistor level is *Node Tearing* [5]. Further early approaches are some cluster algorithms as building *DC-Connected Components* and *Strongly Connected Components* or *Diagonal Dominance Norton Partitioning* [6]. There are also some approaches combining the early methods with the classical *Fiduccia-Mattheyses method* [7] and *hierarchical methods*, which exploit the subcircuit information contained in the circuit description [8, 9, 10]. Most of these approaches are only applicable on MOS-technologycircuits, others are highly focussed on the simulation tool they are developed for.

While developing TITAN a simple straightforward *Clustering Algorithm* (CLA) for partitioning was implemented. Starting from an input voltage source, adjacent elements are gathered until a specified partition size is reached. This approach is very fast but usually it has to be repeated several times with different parameters to obtain acceptable results.

In this paper we present our new concept for partitioning on transistor level. The basic idea is to use analytical placement methods [11, 12]. Applying the ratio cut objective, the placement result can be adapted for circuit partitioning [13, 14]. We are the first to apply this *Analytical Partitioning Method* (APM) to circuits on transistor level. It yields subcircuits with a small number of interconnect nodes and with well balanced sizes.

In the next section we give an outline of the interactions between our partitioning tool and the parallel simulator. Section 3 presents the partitioning approach in detail. In Section 4 we compare the partitioning results of the analytical partitioning method with the results of the simple straightforward clustering algorithm. Therefore, we present the simulation performance results for partitioning of industrial VLSI circuits with both methods. Finally, we give some hints on future work in Section 6.

2 The Parallel Analog Simulation System

The basic idea for partitioning is to use the excellent gate level partitioning system SEAPART [14], which is based on the analytical partitioning method, for splitting circuits on transistor level. SEAPART is designed for partitioning large circuits to be implemented in multi-chip architectures.

Figure 1 gives an overview of the parallel simulation system.

An electronic circuit on transistor level is typically described in SPICE [15], which is also the input format for TITAN. SEAPART uses the PROUD input format. In order to obtain subcircuits valid for simulation, we developed a converter tool ATLAS, which combines certain circuit elements into macro elements, as de-

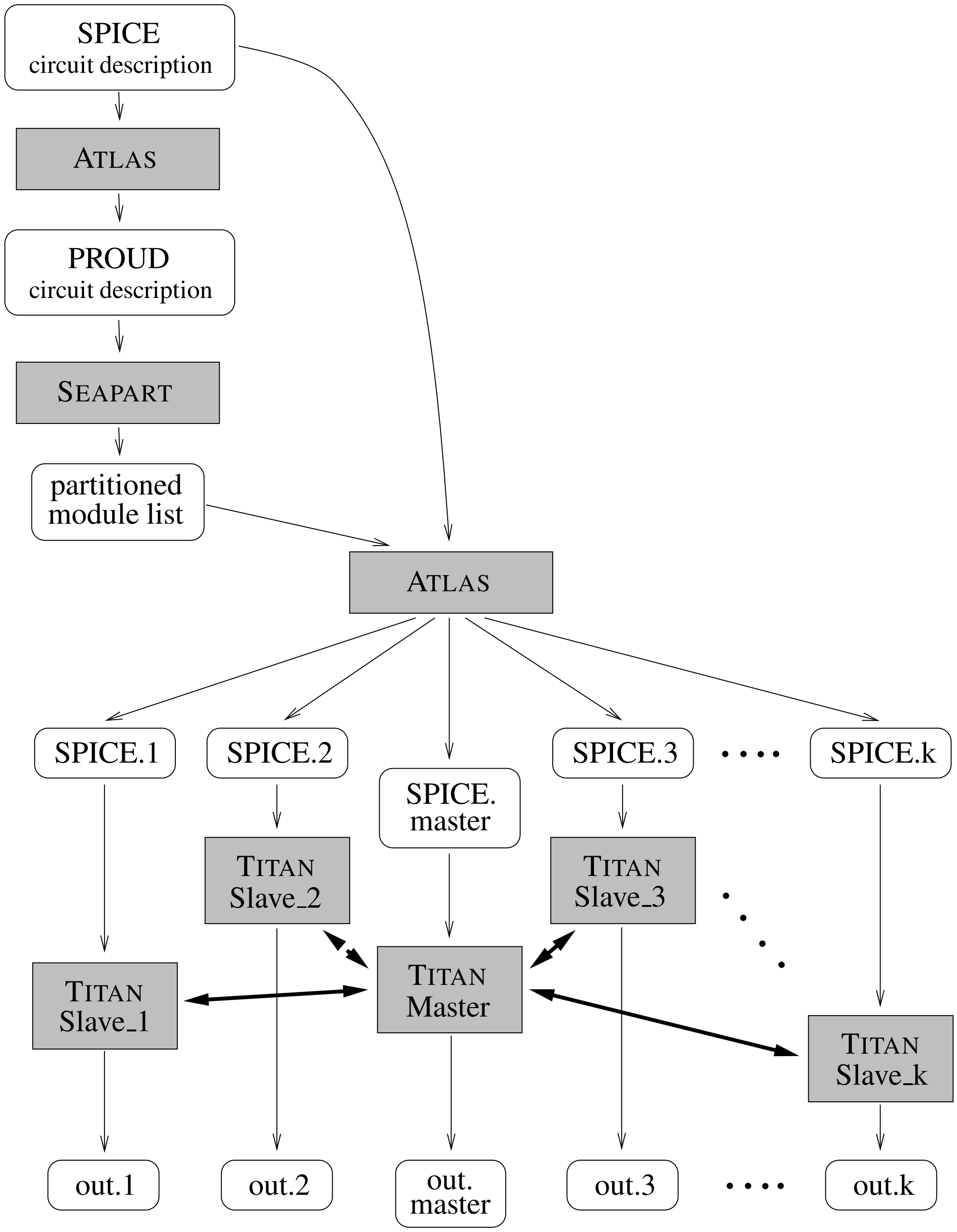

Figure 1: Outline of the Parallel Analog Simulation System

cribed in Section 3.1, and translates the SPICE input language into the PROUD input format of SEAPART. SEAPART reads this circuit description and creates k partitions, which are again converted by ATLAS into SPICE format. These netlists are input for the parallel TITAN. After running the parallel simulation, which is synchronized by the TITAN-master, the simulation output of every simulator is written to a single file. Merging these files yields the overall result.

3 Partitioning Approach for Circuit Simulation

In the following we describe the format conversion performed by ATLAS and the partitioning by SEAPART.

3.1 Circuit Conversion and Preparation for Partitioning

TITAN reads the circuit description in SPICE format, which allows complex constructs containing a numerous variety of elements like transistors, capacitors, inductors, driven sources, etc.. SEAPART operates on an undirected graph, which is created from the PROUD input format containing only modules and the connections between them. Hence, a conversion is necessary when applying SEAPART to a circuit description given in SPICE format. Since there are some restrictions for the parallel simulation resulting in nets not to be cut and some elements to be grouped together, a simple 1:1 conversion of elements to modules is not desirable. Within SEAPART it is not possible to prevent certain nets from being cut. Therefore, these nets need special consideration before starting partitioning with SEAPART.

Thus, we developed the converter tool ATLAS, which reads the SPICE circuit description, detects nets not to be cut, packs related elements into one module and writes the circuit description in PROUD format, as shown in Figure 1. After execution of SEAPART, ATLAS reads the partitioning result and the original SPICE netlist to create a SPICE netlist for each partition, which is required for a parallel circuit simulation.

3.1.1 Requirements of SEAPART

SEAPART has been developed for the design of multichip architectures. It partitions the set of modules to a number of equisized subsets and assigns each subset to a chip on the multichip architecture under consideration of fixed IO-pads on the multichip architecture border. Within ATLAS the computation effort for the simulation of a circuit element is interpreted as its corresponding physical module size. Thus, SEAPART's facility of creating balanced partitions in terms of area size can be exploited. The result is a set of subcircuits, each requiring approximately the same simulation time.

3.1.2 Partitioning Constraints for Parallel Analog Simulation

For simulation, controlled sources need information about their controlling elements. One possibility to meet this requirement is to always assign the controlled source and its controlling element to the same partition. If the partitioning system assigns them to different partitions, the simulation system has to provide the controlled element's partition with information about the controlling element. An assignment to the same partition is also required for mutual inductors.

The nets between subcircuits, i.e. the interconnect nets, are cut while partitioning. Parallel analog simulators handle these nets as I/O nets for the related subcircuits and connect virtual voltage sources to these I/Os, as shown in Figure 2.

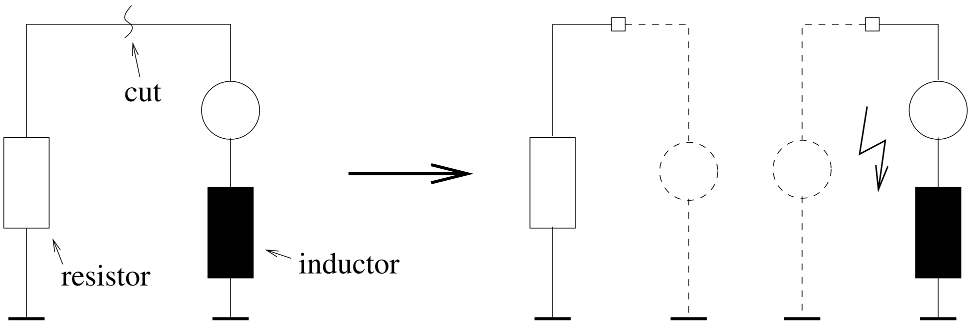

Figure 2: Cut Net

This is crucial, because loops in the circuitry containing only voltage sources and inductors cannot be simulated. It has to be ensured, that no paths from an I/O to ground consisting only of voltage sources and inductors are generated, because the added virtual voltage sources are also connected to ground and thus, a loop invalid for simulation will be created as shown on the right side of Figure 2.

The following sections describe the way we respect these partitioning constraints.

Coupled Modules

For voltage controlled sources the controlling voltage is expressed by the difference between potentials of nets. The SPICE description of the source contains the names of these nets. While parsing this source description, ATLAS generates a module additionally connected to these nets, as shown in Figure 3.

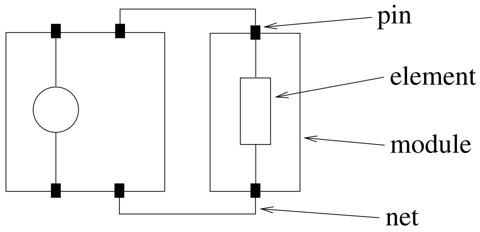

Figure 3: Voltage Controlled Source

The parallel simulator TITAN provides each subcircuit with status information about nets connected to module pins of this subcircuit. Therefore, the voltage controlled source is already informed about the potentials of its controlling nets and no special handling of these sources by ATLAS is required.

For current controlled sources the controlling current is flowing through a certain circuit element. The SPICE description of the source contains only the controlling element's name, but not its connected nets. This results in two independent modules, as shown in Figure 4.

If both modules are assigned to different subcircuits, they are not informed of each other's status. ATLAS creates *coupled modules* to avoid this. The controlling element is removed from the netlist and packed into the controlled module. Its

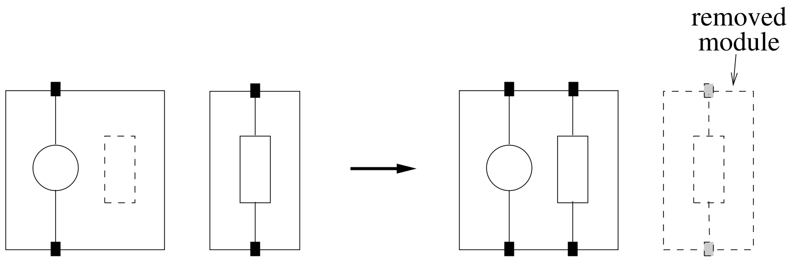

Figure 4: Coupling of Modules

pins become pins of the controlled module and the partitioning tool operates only on one module, which cannot be split. Mutual inductors are coupled in the same way.

Avoiding Illegal Cuts, Meta Modules

Cutting an arbitrary net could produce a path from the cut to ground consisting only of *critical modules* such as voltage sources and inductors. To avoid this, we try to combine a critical module with an uncritical module; thus, the combined *meta module* is no longer critical. If a critical module has a net connected in series to an uncritical module, we restrict cutting the connection by marking this net critical. Next, all elements connected to a critical net are packed into a meta module and the connecting net is eliminated from the netlist as shown in Figure 5. Thus, the partitioning tool operates on a single module and cannot cut this critical net.

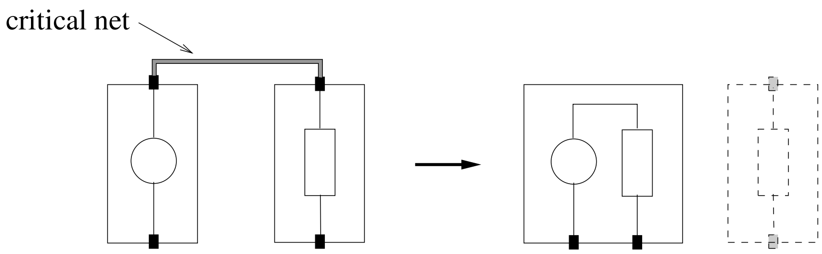

Figure 5: Building Meta Modules

The easiest way to avoid forbidden cuts would be to mark all nets connected to critical modules critical, but this would result in large meta modules. For fine granularity and well balanced partition sizes large meta modules have to be avoided.

The example in Figure 6 shows an an input voltage source connected to net A and B.

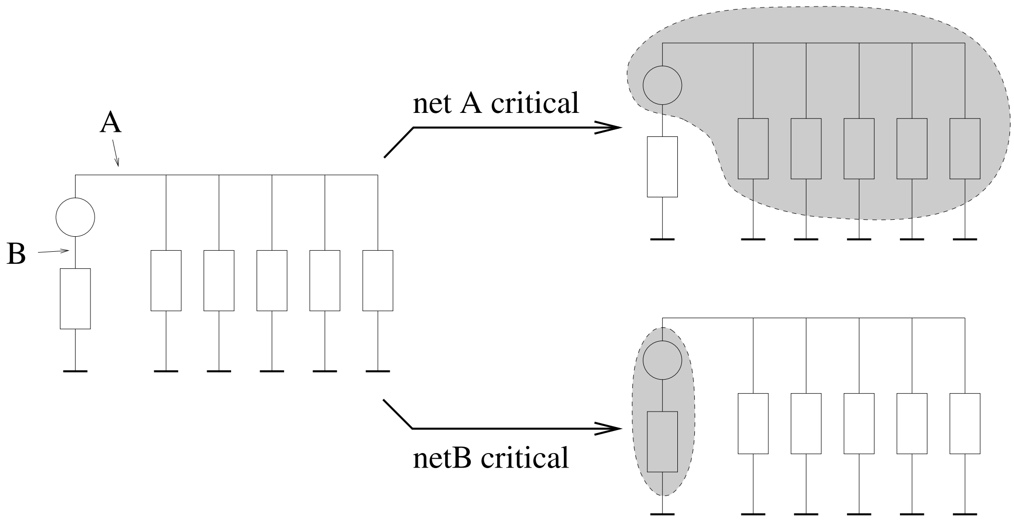

Figure 6: Example for Marking Critical Nets

Usually this source is connected to lots of elements, maybe at net A. If A is marked critical, all the connected elements are packed into one meta module and a large meta module is created. But if it is detected, that B is connected in series to an uncritical element, maybe a resistor, it is sufficient to mark net B critical and only a small meta module including the source and the resistor is created.

For small meta modules, but correct handling of the simulator restrictions it is crucial to find uncritical modules connected in series to a critical module and thus, to mark the right nets. Algorithm 1 examines each net of a critical module whether it is connected to another critical module. If there are nets not connected to other critical modules, the one with the least connections is taken and marked critical. Whereas the critical module is no longer regarded as critical, because in a later step it is combined with the selected net's uncritical modules to a meta module.

mark and count $(CritModCount)$ all critical modules
count for each net $(Net\_CritModCount)$ the connected critical modules
while $(CritModCount > 0) \&\& (CritModCount$ decreased in last iteration) **do**
  for each critical module **do**
    if there is one or more nets of the current module not connected to other critical modules ($Net\_CritModCount == 1$) **then**
      take the net with the least connected modules and mark it as critical
      change the current module's mark to uncritical
      decrease $CritModCount$
      decrease $Net\_CritModCount$ of all other nets connected to the current module
    end if
  end for
end while
if $CritModCount > 0$ **then**
  ERROR: circuit contains a critical loop even before partitioning
end if

Algorithm 1: Marking Critical Nets

Please note the difference between coupled modules and meta modules: In both cases elements are packed together into one module, but in case of coupled modules no nets are eliminated from the netlist, whereas meta modules are used to eliminate nets which are not to be cut.

3.1.3 Creating Partial Circuit Descriptions

After partitioning, ATLAS uses the partitioning result and the original SPICE circuit description to create subcircuit descriptions in SPICE format.

This requires the correct unpacking of coupled modules and meta modules. Cut nets are detected and noted as I/Os for the subcircuits. As the subcircuit simulations are synchronized by the master process, a master description with special references to the cut nets is created. The original SPICE file contains the circuit description as well as several control instructions for the simulator, e.g. initial voltages for certain nets or commands for recording simulation output. After partitioning, control instructions which hold for the entire circuit are written in every subcircuit, whereas instructions related to a special net or an element have to be assigned to the subcircuit, which contains this net or element.

3.2 Analytical Partitioning Method implemented in SEAPART

For partitioning, SEAPART divides a circuit iteratively until each subcircuit, i.e. partition, fits into a single chip of the multichip architecture [14], as shown in Figure 7.

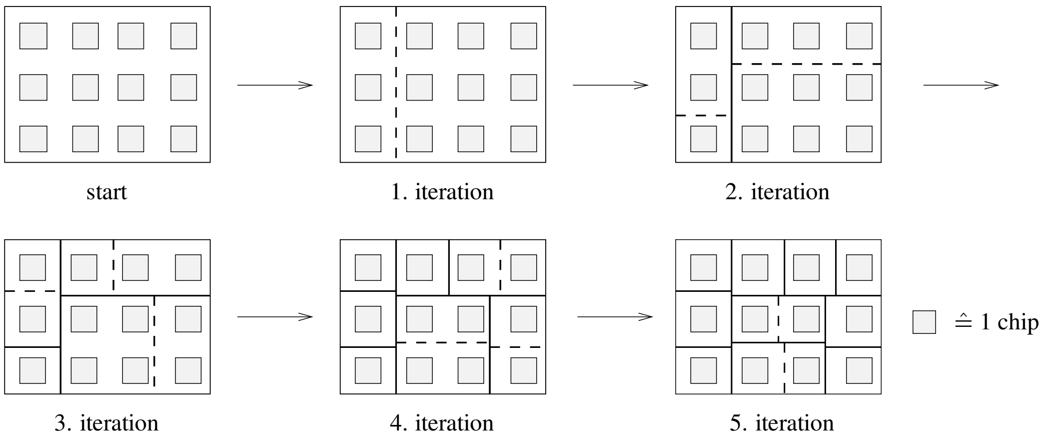

Figure 7: Partitioning Iterations

In each iteration, all partitions that are larger than the target partition size are divided into two parts. The target partition size is the total module area of the entire circuit divided by the desired number of partitions. In each iteration the following three steps are performed:

- calculation of a 2-dimensional embedding

- initial partitioning using the ratio cut objective
- improving the partitioning by moving modules between all partitions

Minimizing a linearized objective function in x- and in y-direction yields a 2-dimensional embedding of the modules with a minimal length of the connections between them. A constraint for this calculation is, that the center of gravity regarding the area of each partition's modules is placed in the middle of this partition. In each iteration, a new embedding is calculated for the complete set of modules. Modules of different partitions can influence each other, although they have already been assigned to certain partitions. This maintains the global view during the entire partitioning process.

Next, each partition which is larger than the target partition size is initially partitioned by a rectangular cut, as demonstrated by the dashed lines in Figure 7. Solid lines represent cuts of former iterations. The initial cut is based on the 2-dimensional embedding and uses an adapted ratio cut objective yielding well balanced partitions in terms of module area.

Finally, the number of cut nets of the current partitioning is reduced by moving modules over partition borders applying a new multi-way ratio cut method with a problem-specific objective function. Module moving is applied in each iteration between all partitions not only between those, which have been divided in the current iteration.

These three steps, calculation of the embedding, initial partitioning, and moving of modules, are done in each iteration, until the number of desired partitions is obtained.

The combined approach allows to exploit the advantages of both analytical and iterative techniques. The analytical technique retains the global view during the entire partitioning process and provides a good initial solution. Starting from this initial solution, the iterative improvement procedure yields significant improvements in acceptable computation time.

Compared to other partitioning tools SEAPART's results are excellent in terms of low number of cut nets and well balanced partitions, which both are crucial for efficient parallel simulation on transistor level.

4 Experimental Results

The performance of the new partitioning approach on transistor level is evaluated on large VLSI circuits. Table 1 shows current large industrial circuits. Each circuit's size is characterized by the number of MOSFETs contained. *Industry1* is the critical path of a Dual Port RAM, whereas *Industry2* is the critical path of a 16 Megabit DRAM design.

Circuit	# MOSFETs	# nets
Industry1	18,000	4,220
Industry2	34,800	18,161

Table 1: Circuits

All experiments are performed by a SGI Power Challenge (8 $\times$ R8000/90MHz). For the simulations the parallel circuit simulator TITAN [1] of Siemens' research laboratories is used. The data communication between the CPUs is performed by the public domain software package *Parallel Virtual Machine* PVM [16].

The main objective for the partitioning is to achieve good speedup for parallel simulation. A low number of interconnect nets reduces the communication between the slaves and the master. Furthermore, the master process, which solves the equations describing the interconnect system, is accelerated. Well balanced partitions yield equal load for the slave processes. Therefore, the number of interconnect nets and the balance of the partitions indicate the achievable speedup. Table 2 compares the performance of the two partitioning methods, the clustering algorithm CLA and the analytical algorithm APM. The table contains the real CPU-time for the simulation, speedups and the number of interconnect nets.

The analytical algorithm is a global method, and thus has to spend additional CPU-time to solve a linear equation system while calculating the 2–dimensional embedding, whereas the clustering algorithm runs very fast because it exploits the natural subcircuit hierarchy of the circuit, which reduces the problem size. On the other hand, especially for eight partitions the analytical algorithm outperforms the

Circuit		1 Proc.	4 Proc.		8 Proc.	
			CLA	APM	CLA	APM
Industry1	real simulation CPU-time (hour:min:sec)	2:28:12	0:50:03	0:48:01	0:28:37	0:27:13
	speedup	–	2.96	3.09	5.18	5.45
	# interconnect nets	–	343	294	371	361
Industry2	real simulation CPU-time (hour:min:sec)	12:31:36	5:47:26	4:51:33	4:17:15	2:31:50
	speedup	–	2.16	2.58	2.92	4.95
	# interconnect nets	–	301	215	415	341

Table 2: Partitioning Results for the Clustering Algorithm (CLA) and the Analytical Method (APM)

clustering approach. It obtains a low number of interconnect nets and increases the performance of parallel simulation. Circuit *Industry1* in Table 2 shows good results for the CLA method but APM is slightly better. Moreover APM achieves significantly better performance than CLA for *Industry2* and reduces the simulation time by about 41%.

Furthermore, while developing the conversion tool to adapt the analytical method for partitioning on transistor level a small circuit with lots of controlled sources was created for testing purposes. The APM method handles this circuit without problems, whereas the CLA method cannot even create a partitioning valid for simulation.

5 Conclusions

In this paper we have presented a new approach for partitioning large electronic circuits on transistor level. Essential requirements for parallel simulation are a low number of interconnect nets and well balanced partitions. Few interconnect nets yield a low synchronization effort at the master process, which is very time critical. Therefore, we developed a conversion tool to apply an excellent analytical partitioning method for the first time on transistor level. This method yields few interconnect nets and equal sized subcircuits.

The efficiency of our new partitioning approach is demonstrated on industrial VLSI circuits. Compared to other known methods parallel simulation time is reduced up to 41%.

6 Future Work

For parallel simulation on gate level another partitioning method was developed at our institute. This method is a hierarchical clustering approach, which combines two well known clustering algorithms in a way that the disadvantages of both algorithms are eliminated and the advantages are added [17]. In terms of interconnect nets and balanced partition sizes comparisons of partitionings on gate level showed results of about the same quality as the analytical method. Thus, as it is promising to apply also the hierarchical clustering approach for partitioning on transistor level, we already started working on this.

Currently the SPICE circuit description is totally flattened by ATLAS, which results in lots of modules to be partitioned. Large VLSI circuits are always designed hierarchically using macro techniques. Thus, there are subcircuits, i.e. natural clusters, in the circuits. We are interested whether it is better to respect the natural clustering or to flatten the circuit description for leaving the most possible degree of freedom for the partitioning tool.

References

[1] U. Feldmann, U. Wever, Q. Zheng, R. Schultz, and H. Wriedt. Algorithms for modern circuit simulation. *Archiv für Elektronik und Übertragungstechnik AEÜ*, 46:274–285, 1992.

[2] M. Schneider, U. Wever, and Q. Zheng. Parallel Harmonic Balance. *IFIP Transactions A-42 VLSI'93*, pages 251–260, 1993. Grenoble.

[3] U. Kleis, O. Wallat, U. Wever, and Q. Zheng. Domain Decomposition Methods for Circuit Simulation. In *ACM/SCS/IEEE Workshop on Parallel and Distributed Simulation (PADS)*, pages 183–186, 1994.

[4] U. Wever and Q. Zheng. Parallel Transient Analysis for Circuit Simulation. In *Proceedings of the 29 Annual Hawaii International Conference on System Sciences*, volume 1, pages 442–447, 1996.

[5] Alberto Sangiovanni-Vincentelli, Li-Kuan Chen, and Leon O. Chua. An Efficient Heuristic Cluster Algorithm for Tearing Large-Scale Networks. *IEEE Transactions on Circuits and Systems CAS*, CAS-24(12):709–717, Dec. 1977.

[6] P. Debefve, F. Odeh, and A.E. Ruehli. Waveform Techniques. In A.E. Ruehli, editor, *Circuit Analysis, Simulation and Design, Part 2*, volume 3 of *Advances in CAD for VLSI*, chapter 8, pages 41–127. North-Holland, 1985.

[7] C.M. Fiduccia and R.M. Mattheyses. A Linear-Time Heuristic for Improving Network Partitions. In *ACM/IEEE Design Automation Conference (DAC)*, volume 19, pages 175–181, 1982.

[8] Paul Cox, Richard Burch, and Berton Epler. Circuit Partitioning for Parallel Processing. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 186–189, 1986.

[9] W. John, W. Rissiek, and K.L. Paap. Circuit Partitioning for Waveform Relaxation. In *European Conference on Design Automation (EDAC)*, pages 149–153, Feb 1991. Amsterdam.

[10] T. Kage, F. Kawafuji, and J. Niitsuma. A Circuit Partitioning Approach for Parallel Circuit Simulation. *IEICE Transactions on Fundamentals*, E77-A(3):461–466, 1994.

[11] Georg Sigl, Konrad Doll, and Frank M. Johannes. Analytical Placement: A Linear or a Quadratic Objective Function? In *ACM/IEEE Design Automation Conference (DAC)*, pages 427–432, San Francisco, 1991.

[12] L. Hagen and A.B. Kahng. Improving the Quadratic Objective Function in Module Placement. In *Fith Annual IEEE International ASIC Conference and Exhibit*, pages 42–45, 1992.

[13] L. Hagen and A. B. Kahng. Fast Spectral Methods for Ratio Cut Partitioning and Clustering. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 10–13, 1991.

[14] Bernhard M. Riess, Konrad Doll, and Frank M. Johannes. Partitioning Very Large Circuits Using Analytical Placement Techniques. In *ACM/IEEE Design Automation Conference (DAC)*, pages 646–651, June 1994.

[15] L. Nagel. *SPICE2: A computer program to simulate semiconductor circuits*. Ph. D. dissertation, Univ. of California, Berkeley, 1975.

[16] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. PVM 3.0 User's Guide and Reference Manual. *Oak Ridge National Laboratory*, 1993. Tenesse.

[17] Christian Sporrer and Herbert Bauer. Corolla Partitioning for Distributed Logic Simulation of VLSI-Circuits. In *ACM/SCS/IEEE Workshop on Parallel and Distributed Simulation (PADS)*, pages 85–92, 1993.

[18] N.B.G. Rabbat, A.L. Sangiovanni-Vincentelli, and H.Y. Hsieh. A multilevel Newton algorithm with macromodelling and latency for the analysis of large-scale nonlinear circuits in the time domain. *IEEE Transactions on Circuits and Systems CAS*, pages 733–741, 1979.