Partitioning of VLSI Circuits and Systems

Frank M. Johannes

Institute of Electronic Design Automation
ECE Department, Technical University of Munich
Arcisstr. 21, D-80333 Munich, Germany
E-mail: Johannes@e-technik.tu-muenchen.de

ABSTRACT

Partitioning plays an increasingly important role in the design process of VLSI circuits and systems. There are partitioning problems to be solved on all levels of abstraction. The rapidly increasing size of the designs will make good partitioning tools even more essential in the future. As an introduction to the other papers in this session, this tutorial presentation discusses the numerous facets of partitioning.

I INTRODUCTION

Partitioning is a technique to divide a circuit or system into a collection of smaller parts (components). It is on the one hand a design task to break a large system into pieces to be implemented on separate interacting components and on the other hand it serves as an algorithmic method to solve difficult and complex combinatorial optimization problems as in logic or layout synthesis.

Partitioning has been an active area of research for at least a quarter of a century. The main reason that partitioning has become a central and sometimes critical design task today is the enormous increase of system complexity in the past and the expected further advances of microelectronic system design and fabrication. Soaring system complexities result from a combination of reasons:

Widely accepted powerful high-level synthesis tools allow the designers to automatically generate huge systems. By just changing a few lines of code in a functional specification the size of the resulting structural description (netlist) of a system can increase dramatically. Synthesis and simulation tools often cannot cope with the complexity of the entire system under development, and designers want to concentrate on critical parts of a system to speed-up the design cycle. Thus, the present state of design technology often requires a partitioning of the system.

Moreover, fabrication technology makes increasingly smaller feature sizes and augmented die dimensions possible, thus allowing a circuit to accommodate several million transistors. However, circuits are restricted in size and in the number of external connections. Thus, fabrication technology requires the partitioning of a system into components.

As a third reason, economical pressure yields larger systems, both to make production cheaper and to exploit the optimization potential of the complete system. The various parts of the system should be implemented in appropriate ways to achieve low-cost fabrication, optimal system performance, and easy adaptation to changing requirements, e.g.. Thus, profit can be received by partitioning a system optimally.

Partitioning applications exist on all levels of abstraction, specifically on the functional (behavioral) and on the structural (netlist) level. In the early stages of design, far-reaching decisions have to be made how to partition a design, often based on incomplete knowledge. In particular, it has to be decided whether to implement a component in various types of hardware or software to achieve an optimal size/performance trade-off. Because the granularity is low in this application, i.e. relatively few objects of moderate to high complexities, partitioning can possibly be done by human designers based on their experience. Since fully automatic partitioning is essential for fast iterations in the design cycle, considerable effort is made in academia as well as in industry to facilitate and improve the difficult decisions on functional level.

The components resulting from system partitioning are implemented by a team of designers or synthesized from a high-level description by using synthesis tools that generate a structural implementation. In case the hardware components are too complex for packaging because of area and i/o constraints they are further partitioned on structural level based on rather fine-grained objects (modules, cells). It has been observed [42] that structure synthesis tools, in general, do not generate a hierarchy which can be used directly for layout design if this hierarchy is deep. To give the layout synthesis tools the freedom they require to generate good results, the netlist has to be flattened out and repartitioned. The subject of partitioning on structural level is studied very well in the literature.

It is the intention of this paper to provide a concise review of the state of the art in the field of partitioning and also to serve as an introduction to the other papers of this session. To achieve this dual goal a subjective selection has been made from the abundance of partitioning methods published in the literature. This is of course necessary due to space limitations. But the choice has been facilitated very much by the availability of the profound and up-to-date survey published by Charles Alpert and Andrew Kahng [3] which includes a complete bibliography. Therefore, mainly the most recent publications have been included in the reference section of this paper. Previous reviews on partitioning have appeared in [11, 43]. Readers interested especially in algorithms are referred to [27]. Excellent evaluations of bipartitioning techniques have recently been published in [17, 54]. From the perspective of system specification and design the partitioning problem is addressed in [13], e.g..

The rest of the paper is organized as follows. In the next section different applications and the resulting partitioning problems are discussed. In Section III, an overview of basic solution techniques is provided. Finally, some remarks on possible future developments will be given.

II Types of Partitioning

In the first part of this section typical applications of partitioning will be discussed and classes of partitioning problems will be summarized in part two.

The emulation of systems and rapid prototyping based on arrays of interconnected field-programmable gate arrays (FPGAs) has become very popular. To obtain the shortest possible design cycle, automatic partitioning of a design is absolutely necessary. As the target architecture is fixed in most applications, i.e. the number of FPGA components, their types and their geometric arrangement are given, the partitioning task is to find a mapping of the system's objects to the FPGAs while satisfying constraints like FPGA area (number of logic blocks), FPGA pin count, or the delay of the critical paths. In this case, a decision problem has to be solved, i.e. a feasible solution is sought. If delay, e.g., is not a constraint but has to be minimized, a constrained optimization problem must be solved. Partitioning for multiple FPGAs was the application most frequently addressed in recent papers [5, 8, 18, 22, 25, 31, 26, 36, 40, 50, 51].

Multi-chip modules (MCMs) have been developed as an alternative packaging technology to conventional printed circuit boards to achieve higher packing densities, reduced signal delays, lower heat dissipation etc. To realize these goals and to explore more alternative solutions than a human designer can, it is necessary to apply automatic partitioning. If the chips have already been allocated, i.e. a set of system components has been found to implement the system's function, the partitioning task is similar to FPGA arrays. However, it may be desirable to search for optimal system partitionings first. The final allocation together with the partitioning can then be chosen from the set of solutions found.

The floorplanning problem is very well known from the literature and is mostly considered a genuine layout design problem. But floorplanning and partitioning are closely interrelated. Traditionally, with the partitioning of the system into blocks being given the floorplanning task is to determine the relative block positions as well as their sizes, aspect ratios and possibly their pin positions to optimize silicon area while fulfilling timing constraints. However, the optimal floorplan largely depends on the quality of the partitioning. Floorplanning is difficult since parts of the circuit may not be fully specified or implemented making accurate estimation of a block's parameters essential. With systems under design becoming larger the floorplanning problem has recently gained much interest as all major CAD vendors aim at offering powerful floorplanning tools.

Microelectronic systems typically consist of application specific hardware parts and programmable software parts, e.g. ASICs, custom or standard processors, and memories. Software is more flexible, can be developed easier, and is less expensive in terms of cost and development time. Hardware components provide higher performance at the expense of higher cost. Thus, a system designer will work towards a partitioning into hardware and software components that fulfills all performance constraints while using as little hardware as possible. This constrained optimization problem is a key problem of hardware/software codesign [23, 33].

The decision problem of partitioning digital signal processing (DSP) algorithms over multi-processor systems is hard to solve because of its many constraints. The programming of the signal processors is done by mapping a functional specification given by a signal flow graph onto a given network of processors. It is essential that this task is performed fully automatically [1] such that different architectures and solutions can be investigated quickly.

For the design of VLSI circuits, logic simulators are the design tools used most intensively. Due to the ever increasing complexity of digital circuits, logic simulation suffers from consuming vast amounts of computing time and memory. Parallel computer architectures and in particular workstation networks offer an ideal possibility to overcome these bottlenecks. To achieve high efficiency, the circuit to be simulated must be partitioned such that the amount of communication is adapted to the available resources and the simulation load is evenly distributed among the processors. Significant speed-ups can be observed with parallel simulation which may be viewed as a divide-and-conquer method. To avoid idle processors dynamic load balancing, i.e. updating the partitioning over time, is necessary (see [46, 9, 41], e.g.).

The divide-and-conquer paradigm is widely used for solving large problems to reduce their complexity. The problem is recursively (top-down) partitioned into smaller subproblems. This process continues until subproblems are small enough to be solved directly. The solutions are combined hierarchically which yields, in general, suboptimal solutions on the next higher level. A famous example for this successful solution strategy is the min-cut placement method in layout synthesis. Partitioning is applied recursively to the circuit's netlist thereby generating a hierarchical neighborhood (slicing) structure. This structure is then interpreted as a floorplan for chip assembly. In addition to reducing problem size, solution quality is improved and heavy wiring congestion avoided by minimizing the number of wires cut by the partitions.

Mathematically, partitioning problems are mostly formalized by using graphs. Functional descriptions in hardware description languages can be modeled by *signal flow graphs*. The edges of this directed type of graph describe the signal flow between functional units or registers which are represented by the graph's nodes.

For modeling circuits as graphs on the structural level many alternatives have been published in the literature [27]. The modules of a circuit can be represented as a set of nodes. The signal nets interconnecting the modules can either be characterized by directed or undirected edges between pairs of nodes or by hyperedges connecting sets of nodes (hypergraph). Edges and/or nodes may be assigned weights, costs, or capacities. E.g., node weight can be used to characterize module area or edge weights model multiple connections between modules. Hyperedges can be mapped into sets of (binary) edges by using the model of a clique (complete subgraph) or a star (by adding additional nodes for the nets). Changing the modeling by weighting the clique's edges can improve the results, although it has been shown that perfect weightings cannot be achieved [27].

As a consequence of the differing applications various problem formulations for partitioning can be distinguished from an algorithmic point of view:

- Two-way partitioning or bipartitioning divides a graph into two non-empty subgraphs while minimizing the number or weight of the edges cut by the partition. To obtain balanced partitions which is often desirable constraints on the partition sizes are applied. The aims to minimize the number of cut edges and to obtain balanced partitions have been integrated in the minimum ratio cut objective function.

- Multi-way partitioning divides a graph into a prespecified number of subgraphs. The standard objective is to minimize the number of edges between all partitions while meeting constraints. Typical constraints are lower and upper size limits on the area and the pin count of the components. As a generalization of ratio-cut partitioning, various alternative objectives have been proposed for multi-way partitioning to combine cut size and the balance criterion in a single objective. Multi-way partitionings can be generated by recursively solving two-way partitioning problems.

For rapid prototyping using arrays of FPGAs the multi-way partitioning problem is formulated as a decision problem (see [8, 19], e.g.). A feasible solution is sought that satisfies all constraints. All such feasible solutions are equally good assuming they are routable.

- Performance-driven partitioning is targeted towards optimizing the system's performance rather than only minimizing the number of interconnections between components. Most recent papers aim at improving the timing [44, 24, 25, 26, 40, 51], but minimal power consumption has also been used as an objective [49]. In [26] knowledge about the logic function of the circuit has been used to improve performance.
- Layout-driven partitioning emphasizes geometric aspects. The partitioning is based on a placement or geometric ordering rather than merely on structure (see [35, 36, 37] and [14, 4, 2, 28], e.g.). This is motivated by the observation that a placement with minimized wire length will lead to a low probability for a net being cut. As system performance heavily depends on layout this type of partitioning inherently combines several partitioning objectives. It has been shown in [48] that partitioning and placement are indeed equivalent in an algorithmic sense.
- Partitioning with replication is useful to enhance system performance, in particular in prototyping systems with FPGAs where the available FPGA pins are the limiting resource. By eliminating the requirement that the components of a partition have to be disjunct, the number of component pins can be reduced and the system's performance improved by replicating parts of the netlist in different components. The additional area needed is usually available. This problem has gained much interest in the recent literature [22, 26, 30, 31, 40, 50, 51].

From an algorithmic standpoint, most partitioning problems are classified as hard problems, which means that in practice heuristics have to be used to obtain useful solutions.

III Solution Techniques

In this section an overview of attractive methods for solving partitioning problems will be given. More detailed information about the algorithms can be found in [3, 27].

Partitioning methods can be classified as being *constructive* or *iterative*. Constructive algorithms determine a partitioning from the graph describing the circuit or system, whereas iterative methods aim at improving the quality of an existing partitioning solution.

Partitioning algorithms can also be labeled *deterministic* or *probabilistic*. Deterministic programs generate the same solution each time they are started. Probabilistic methods result in differing solutions because they are based on random numbers.

Constructive partitioning approaches are mainly based on clustering, spectral or eigenvector methods, placement-based partitioning, mathematical programming, or network flow computations.

Clustering is a bottom-up technique to determine strongly connected components of a graph. It can perform well when cluster sizes are small. To partition designs of millions of modules bottom-up clustering is often combined with top-down partitioning [8]. A formulation unifying both strategies has recently been published in [20]. Clustering has also been applied to optimize performance [24, 34, 49, 51, 53]. For parallel simulation and load balancing a hierarchical four-phase algorithm based on corolla clustering has been developed [46, 41]. A corolla is defined in [10] as a set of overlapping fanout regions of a circuit. Cone structures, i.e. sets of all nodes of a combinatorial block between a single output and the inputs that lead to that single output, have recently been applied to bottom-up FPGA partitioning for critical paths [5, 9]. In these approaches, the consideration of signal directions has resulted in improved partitioning solutions. The trade-off between run time and performance is explored in [39, 52]. In the next paper of this session a clustering approach is described where the graph's global connectivity information is derived from the clustering property of the eigenvector approach [29].

Next, mathematical programming and spectral methods will be addressed. These approaches are attractive because they keep the global view of the partitioning problem.

Mathematical programming techniques are used to optimize an objective function under inequality constraints. Quadratic programming [35], quadratic Boolean programming [45], linear and integer programming [24, 31, 23, 33] have been applied to partitioning problems.

Spectral methods have recently attracted much attention [14, 2, 4, 28, 6]. Based on the adjacency matrix of the graph the min-cut objective can be rewritten as an equation system. The eigenvector of its minimum non-zero eigenvalue can be interpreted as a linear placement or ordering of the graph's nodes. This ordering can be cut to yield a partitioning of the nodes. Many modifications of this basic method have been published in the literature including the use of more than one eigenvector. For multi-way partitioning it has been demonstrated that the more eigenvectors are used the better is the partitioning quality [4]. It is interesting to note that Rent's parameter and eigenvalues have properties in common [14].

Placement-based partitioning is closely related to spectral methods. These methods minimize a quadratic objective function. For placement, it has been shown that minimizing a linear objective yields improved results. In [35] this observation has been used to obtain better partitionings. Since the placement with a linear objective is derived from a eigenvector-based placement this method could also be classified as an iterative improvement method. This placement-based approach has also been extended to multi-way partitioning with application to FPGAs and MCMs [36, 37]. The next paper in this session will show how the linear objective could be integrated directly into eigenvector formulations [29].

With network flow methods directed signal flow can be used to improve system performance. Different types of network flow formulations have been proposed [20, 22, 28, 30, 31, 50, 53]. They all have in common that a graph model is devised from the directed netlist to determine a maximum flow which is equivalent to a minimum cut. These methods have shown to be particularly effective for solving partitioning problems with replication.

Probabilistic constructive algorithms have not been used frequently to solve partitioning problems. Recent examples are [18, 19, 53]. In [18, 19] a linear ordering is determined by randomly selecting nodes to start with. By using dynamic programming the ordering is split into clusters. The clustering approach in [53] is based on multi-commodity flows. A probabilistic flow-injection method is proposed to reduce the computational complexity of the flow-based algorithm.

Numerous deterministic iterative approaches have been published. These methods iteratively exchange nodes or interchange pairs of nodes to minimize the number of edges cut. For this reason, they are often collectively designated as min-cut algorithms. They differ significantly in the choice of the objective function used. An improved objective based on probabilistic gain computation is introduced in the last paper of this session [12]. Iterative improvement can be combined with clustering to reduce computational complexity. Recent advances are the following: Because deterministic iterative methods are sensitive to how the starting partitioning is chosen, a gradient method was proposed in [32] to overcome this disadvantage. The authors of [15] found that implementation choices play an important role and reported significant improvements in computing the gains. In [47] a min-cut partitioning method is proposed which is based on quadratic programming.

The probabilistic iterative improvement methods commonly used in design automation are simulated annealing and simulated evolution. Their most important advantage is that they can escape from local minima. A recent experimental evaluation [54] on two-way partitioning concluded that simulated annealing seems to offer little advantage in solution quality, but consumes a large amount of computing time. In [38] simulated annealing was applied to FPGA and MCM partitioning and computation time was reduced by clustering. A different randomized approach was proposed in [7]. This method can escape from local minima by using the concept of *alternate wires* comes from logic synthesis and is very similar to *redundancy addition and removal* known from automatic test pattern generation.

IV Future Directions

Due to the increasing requirements on partitioning tools further developments and improvements are very desirable. It has been observed in the past that even minor algorithmic modifications can be very effective.

From the application viewpoint, highly constrained performance-driven partitioning is attractive for research and accurate but efficient delay calculation remains an important issue. With the rapidly increasing computing power in mind enumerative methods will become more attractive.

On the higher levels of abstraction, applying logic synthesis methods, e.g. logic replication and retiming, seems to have great optimization potential. The estimation of system properties needs attention such that designers can examine potential partitioning solutions quickly at the highest abstraction level possible.

Benchmarking of partitioning approaches (as of other classes of design problems) urgently needs to be improved. In particular, huge problem instances are needed to demonstrate the power of solution methods. To generate such problems that resemble real problems a replication method has been proposed [16]. Another approach to solve this problem will be presented in this session [21].

Last but not least, synergy effects could result from coordinating the partitioning activities on different levels of abstraction and for different applications at the major conferences.

References

[1] E.H.L. Aarts, G. Essink, and E.A. de Kock. Recursive bipartitioning of signal flow graphs for programmable video signal processors. In *The European Design and Test Conference (EDTC)*. IEEE, 1996. Session 9A.

[2] Charles J. Alpert and Andrew B. Kahng. Multiway partitioning via geometric embeddings, orderings, and dynamic programming. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14:1342–1358, 1995.

[3] Charles J. Alpert and Andrew B. Kahng. Recent directions in netlist partitioning: a survey. *INTEGRATION, the VLSI journal*, 19:1–81, 1995.

[4] Charles J. Alpert and So-Zen Yao. Spectral partitioning: the more eigenvectors the better. In *Design Automation Conference (DAC)*, pages 195–200. ACM/IEEE, 1995.

[5] Daniel Brasen and Gabrièle Saucier. FPGA partitioning for critical paths. In *The European Design and Test Conference (EDTC)*, pages 99–103. IEEE, 1994.

[6] P.K. Chan, M.D.F. Schlag, and J.Y. Zien. Spectral based multi-way FPGA partitioning. In *Int'l Symposium on Field-Programmable Gate Arrays*, pages 133–139. ACM, 1995.

[7] David Ihsin Cheng, Chih-Chang Lin, and Malgorzata Marek-Sadowska. Circuit partitioning with logic perturbation. In *International Conference on Computer Aided Design (ICCAD)*, pages 650–655. IEEE/ACM, 1995.

[8] Nan-Chi Chou, Lung-Tien Liu, Chung-Kuan Cheng, Wei-Jin Dai, and Rodney Lindelof. Local ratio cut and set covering partitioning for huge logic emulation systems. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14:1085–1092, 1995.

[9] Jason Cong, Zheng Li, and Rajive Bagrodia. Acyclic multi-way partitioning of Boolean networks. In *Design Automation Conference (DAC)*, pages 670–675. ACM/IEEE, 1994.

[10] Sujit Dey, Franc Brglez, and Gershon Kedem. Corolla based circuit partitioning and resynthesis. In *Design Automation Conference (DAC)*, pages 706–712. ACM/IEEE, 1990.

[11] W.E. Donath. Logic partitioning. In B. Preas and M. Lorenzetti, editors, *Physical Design Automation in VLSI Systems*. Benjamin/Cummings, 1988.

[12] Shantanu Dutt and Wenyong Deng. A probability-based approach to VLSI circuit partitioning. In *Design Automation Conference (DAC)*. ACM/IEEE, 1996.

[13] Daniel D. Gajski and Frank Vahid. Specification and design of embedded hardware/software systems. *Design & Test of Computers*, pages 53–67, 1995.

[14] Lars Hagen, Andrew B. Kahng, Fadi L. Kurdahi, and Champaka Ramachandran. On the intrinsic Rent parameter and spectra-based partitioning methods. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13:27–37, 1994.

[15] Lars W. Hagen, Dennis J.-H. Huang, and Andrew B. Kahng. On implementation choices for iterative improvement partitioning algorithms. In *European Design Automation Conference (Euro-DAC)*, pages 144–149, 1995.

[16] Lars W. Hagen, Dennis J.-H. Huang, and Andrew B. Kahng. Quantified suboptimality of VLSI layout heuristics. In *Design Automation Conf. (DAC)*, pages 216–221. ACM/IEEE, 1995.

[17] Scott Hauck and Gaetano Borriello. An evaluation of bipartitioning techniques. In *Chapel Hill Conference on Advanced Research in VLSI*, pages 383–402, 1995.

[18] Scott Hauck and Gaetano Borriello. Logic partition orderings for multi-FPGA systems. In *International Symposium on Field-Programmable Gate Arrays*, pages 32–38. ACM, 1995.

[19] Dennis J.-H. Huang and Andrew B. Kahng. Multi-way system partitioning into single and multiple type FPGAs. In *International Symposium on Field-Programmable Gate Arrays*, pages 140–145. ACM, 1995.

[20] Dennis J.-H. Huang and Andrew B. Kahng. When clusters meet partitions: new density-based methods for circuit decomposition. In *The European Design and Test Conference (EDTC)*, pages 60–64. IEEE, 1995.

[21] Michael Hutton, J.P. Grossman, Jonathan Rose, and Derek Corneil. Characterization and parameterized random generation of digital circuits. In *Design Automation Conference (DAC)*. ACM/IEEE, 1996.

[22] L. James Hwang and Abbas El Gamal. Min-cut replication in partitioned networks. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14:96–106, 1995.

[23] I. Karkowski and R.H.J.M. Otten. An automatic hardware-software partitioner based on the possibilistic programming. In *The European Design and Test Conference (EDTC)*. IEEE, 1996. Session 9A.

[24] Yoshinori Katsura, Tetsushi Koide, Shin'ichi Wakabayashi, and Noriyoshi Yoshida. A new system partitioning method under performance and physical constraints for multi-chip modules. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 119–126. IFIP/ACM/IEEE, 1995.

[25] Chunghee Kim, Hyunchul Shin, and Younguk Yu. Performance-driven circuit partitioning for prototyping by using multiple FPGA chips. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 113–118. IFIP/ACM/IEEE, 1995.

[26] Roman Kužnar and Franc Brglez. PROP: a recursive paradigm for area-efficient and performance oriented partitioning of large FPGA netlists. In *International Conference on Computer Aided Design (ICCAD)*, pages 644–649. IEEE/ACM, 1995.

[27] T. Lengauer. *Combinatorial algorithms for integrated circuit layout*. Wiley-Teubner, 1990.

[28] J. Li, J. Lillis, and C.K. Cheng. Linear decomposition algorithm for VLSI design applications. In *International Conference on Computer Aided Design (ICCAD)*, pages 223–228. IEEE/ACM, 1995.

[29] Jianmin Li, John Lillis, Lung-Tien Liu, and Chung-Kuan Cheng. New spectral linear placement and clustering. In *Design Automation Conference (DAC)*. ACM/IEEE, 1996.

[30] Lung-Tien Liu, Ming-Ter Kuo, Chung-Kuan Cheng, and T.C. Hu. A replication cut for two-way partitioning. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14:623–630, 1995.

[31] Lung-Tien Liu, Ming-Ter Kuo, Chung-Kuan Cheng, and Te C. Hu. Performance-driven partitioning using a replication graph approach. In *Design Automation Conference (DAC)*, pages 206–210. ACM/IEEE, 1995.

[32] Lung-Tien Liu, Ming-Ter Kuo, Shih-Chen Huang, and Chung-Kuan Cheng. A gradient method on the initial partition of Fiduccia-Mattheyses algorithm. In *Int'l Conf. on Computer Aided Design (ICCAD)*, pages 229–234. IEEE/ACM, 1995.

[33] Ralf Niemann and Peter Marwedel. Hardware/software partitioning using integer programming. In *The European Design and Test Conference (EDTC)*. IEEE, 1996. Session 9A.

[34] Rajmohan Rajaraman and D.F. Wong. Optimum clustering for delay minimization. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14:1490–1495, 1995.

[35] Bernhard M. Riess, Konrad Doll, and Frank M. Johannes. Partitioning very large circuits using analytical placement techniques. In *Design Automation Conference (DAC)*, pages 646–651. ACM/IEEE, 1994.

[36] Bernhard M. Riess, Heiko A. Giselbrecht, and Bernd Wurth. A new k-way partitioning approach for multiple types of FPGAs. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 313–318. IFIP/ACM/IEEE, 1995.

[37] Bernhard M. Riess and Andreas A. Schoene. A new layout design system for multichip modules. *International Journal of High Speed Electronics and Systems*, 6:509–538, 1995.

[38] K. Roy-Neogi and C. Sechen. Partitioning with performance optimization. In *Int'l Symposium on Field-Programmable Gate Arrays*, pages 146–152. ACM, 1995.

[39] Youssef G. Saab. A fast and robust network bisection algorithm. *Transactions on Computers*, 44:903–913, 1995.

[40] Prashant Sawkar and Donald Thomas. Multi-way partitioning for minimum delay for look-up table based FPGAs. In *Design Automation Conf. (DAC)*, pages 201–205. ACM/IEEE, 1995.

[41] R. Schlagenhaft, M. Ruhwandl, Ch. Sporrer, and H. Bauer. Dynamic load balancing of a multi-cluster simulator on a network of workstations. In *Workshop on Parallel and Distributed Simulation (PADS)*, pages 175–180, 1995.

[42] B. Schürmann, J. Altmeyer, and G. Zimmermann. Three-phase chip planning — an improved top-down chip planning strategy. In *International Conference on Computer Aided Design (ICCAD)*, pages 598–605. IEEE/ACM, 1992.

[43] N.A. Sherwani. *Algorithms for VLSI physical design automation*. Kluwer, Boston, 1993.

[44] Minshine Shih and Ernest S. Kuh. Quadratic Boolean programming for performance-driven system partitioning. In *Design Automation Conference (DAC)*, pages 761–765. ACM/IEEE, 1993.

[45] Minshine Shih and Ernest S. Kuh. Circuit partitioning under capacity and i/o constraints. In *Custom Integrated Circuits Conference*, pages 659–662. IEEE, 1994.

[46] Ch. Sporrer and H. Bauer. Corolla partitioning for distributed logic simulation of VLSI circuits. In *Workshop on Parallel and Distributed Simulation (PADS)*, pages 85–92, 1993.

[47] Katsunori Tani. A robust min-cut improvement algorithm based on dynamic look-ahead weighting. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 127–133. IFIP/ACM/IEEE, 1995.

[48] Ren-Song Tsay and Ernest Kuh. A unified approach to partitioning and placement. *Transactions on Circuits and Systems*, 38:521–533, 1991.

[49] Hirendu Vaishnav and Massoud Pedram. Delay optimal partitioning targeting low power VLSI circuits. In *International Conference on Computer Aided Design (ICCAD)*, pages 638–643. IEEE/ACM, 1995.

[50] Hannah Honghua Yang and D.F. Wong. New algorithms for min-cut replication in partitioned circuits. In *International Conference on Computer Aided Design (ICCAD)*, pages 216–222. IEEE/ACM, 1995.

[51] Honghua Yang and D.F. Wong. Circuit clustering for delay minimization under area and pin constraints. In *The European Design and Test Conference (EDTC)*, pages 65–70. IEEE, 1995.

[52] Ching-Wei Yeh. On the acceleration of flow-oriented circuit clustering. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14:1305–1308, 1995.

[53] Ching-Wei Yeh, Chung-Kuan Cheng, and Ting-Ting Y. Lin. Circuit clustering using a stochastic flow injection method. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14:154–162, 1995.

[54] Ching-Wei Yeh, Chung-Kuan Cheng, and Ting-Ting Y. Lin. Optimization by iterative improvement: an experimental evaluation on two-way partitioning. *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 14:145–153, 1995.