Recent Directions in Netlist Partitioning: A Survey*

Charles J. Alpert and Andrew B. Kahng

UCLA Computer Science Department, Los Angeles, CA 90024-1596

Abstract

This survey describes research directions in netlist partitioning during the past two decades, in terms of both problem formulations and solution approaches. We discuss the traditional min-cut and ratio cut bipartitioning formulations along with multi-way extensions and newer problem formulations, e.g., constraint-driven partitioning (for FPGAs) and partitioning with module replication. Our discussion of solution approaches is divided into four major categories: **move-based** approaches, **geometric representations**, **combinatorial** formulations, and **clustering** approaches. Move-based algorithms iteratively explore the space of feasible solutions according to a *neighborhood operator*; such methods include greed, iterative exchange, simulated annealing, and evolutionary algorithms. Algorithms based on geometric representations embed the circuit netlist in some type of "geometry", e.g, a 1-dimensional linear ordering or a multi-dimensional vector space; the embeddings are commonly constructed using *spectral* methods. Combinatorial methods transform the partitioning problem into another type of optimization, e.g., based on network flows or mathematical programming. Finally, clustering algorithms merge the netlist modules into many small clusters; we discuss methods which combine clustering with existing algorithms (e.g., two-phase partitioning). The paper concludes with a discussion of benchmarking in the VLSI CAD partitioning literature and some perspectives on more promising directions for future work.

1 Introduction

The essence of netlist partitioning is to divide a system specification into clusters such that the number of intercluster connections is minimized. The partitioning task is ubiquitous to many subfields of VLSI CAD: any top-down hierarchical (i.e., "divide and conquer") approach to system design must rely on some underlying partitioning technique. There are several reasons why partitioning has recently emerged as a critical optimization in many phases of VLSI system synthesis, and why the past several years have seen so much research activity on the subject.

Above all, partitioning heuristics are used to address the increasing complexity of VLSI design. Systems with several million transistors are now common, presenting instance complexities that are unmanageable for existing logic-level and physical-level design tools. Partitioning divides a system

*Partial support for this work was provided by NSF Young Investigator Award MIP-9257982.

into smaller, more manageable components; the number of signals which pass between the components corresponds to the interactions between the design subproblems. In a top-down hierarchical design methodology, decisions made early in the system synthesis process (e.g., at the system and chip levels) will constrain succeeding decisions. Thus, the feasibility – not to mention the quality – of automatic placement, global routing and detailed routing depends on the quality of the partitioning solution. A bottom-up clustering may also be applied to decrease the size of the design, typically in cell- or gate-level layout. The current emphasis on a quick-turnaround ASIC design cycle reinforces the need for reliable and effective algorithms.

Partitioning heuristics also have a greater impact on system performance as designs become interconnect-dominated. In current submicron designs, wire delays tend to dominate gate delays [15]; the differences between on-chip and off-chip signal delays and the increasingly pin-limited nature of large chips make it desirable to minimize the number of signals traveling off a given chip. Larger die sizes imply that long on-chip global routes between function blocks will more noticeably affect system performance. Other considerations (e.g., design for testability, low-power design, etc.) also require partitioning algorithms to identify interconnect structure, albeit at more of a functional or communication-based level.

Finally, partitioning heuristics affect the layout area: wires between clusters at high levels of the hierarchy will tend to be longer than wires between clusters at lower levels, and total wirelength is directly proportional to layout area due to minimum wire spacing design rules. The traditional minimum-cut objective is natural for this application: if the layout area is divided into a dense uniform grid, total wirelength can be expressed in "grid" units or equivalently as the sum over all gridlines of the number of wires crossing each gridline. This view can also improve auto-routability since it suggests reducing the wire congestion in any given layout region.

All of these considerations motivate the development of netlist partitioning tools that identify interconnection and communication structure in a given system design. Indeed, one of the five-year predictions issued by participants at the 1991 CANDE workshop [142] was that stand-alone partitioning for every phase of system synthesis would comprise the next major class of CAD tools to emerge in the marketplace. That this prediction has come to pass is evident from the various new tools and startup companies which focus almost exclusively on partitioning.[1]

Today, leading applications of partitioning include:

- **General design packaging.** Logic must often be partitioned into clusters, subject to constraints

[1] Two notable examples are: High Level Design Systems, whose "advanced design planner" performs design planning at the floorplan level, and ACEO, whose "SoftWire" tool performs communication-based partitioning of multiple-FPGA systems. Partitioning is also at the heart of new tools for system-level design optimization and mapping of designs onto emulation or prototyping architectures composed of multiple field-programmable gate arrays (FPGAs).

on cluster area as well as possible I/O bounds. This problem is known as *design packaging* and is still the canonical partitioning application; it arises not only in chip floorplanning and placement, but also at all other levels of the system design.[2] The design packaging problem also arises whenever technology improves and existing designs must be repackaged onto higher-capacity modules ("technology migration"). Note that the problem is usually associated with large cluster sizes, with few constraints on the internal structure of the clusters. When a finite library of available module types is specified, the optimization is more along the lines of "covering" or "technology mapping".

- **Netlist-level partitioning in HDL-based synthesis.** Synthesis tools have emerged which reduce the design cycle by automatically mapping a high-level functional description to a gate- or cell-level netlist. However, even with increased maturity of such high-level synthesis tools, netlist partitioning remains central to the success of the design procedure. This is essentially because writing HDL code – as opposed to performing layout – can abstract away the physical layout effects of design choices (for example, a few lines of HDL code that specify a register file or crossbar connection can correspond to a large portion of the final layout area). As a result, the block decomposition of the functional (software) description does not necessarily map well into a decomposition of the physical layout. Hence, in contrast to previous building-block methodologies which yielded a small number of function blocks that could be optimally hand-partitioned, HDL-based synthesis virtually requires the physical design methodology to shift from working with a small number of building blocks to working with large, flattened design representations [53]. Partitioning of flattened inputs is also necessary for such applications as the design of "precursor systems" (i.e., finding the packaging tradeoffs that correspond to optimum cost-performance points at early stages in the product life cycle).

- **Estimation for design optimization.** Accurate estimation of layout area and wireability has always been a critical element of high-level synthesis and floorplanning. Now, such estimates are becoming critical to higher-level searches over the system design space. Predictive models often combine analysis of the netlist partitioning structure with analysis of the output characteristics of placement and routing algorithms, in order to yield estimates of wiring requirements and system performance. This use of system partitioning hierarchies is increasingly prominent as "design optimization" and "electronic *system* design automation" capture the attention of CAD users and vendors.

- **System emulation and rapid prototyping (FPGA partitioning).** Many logic emulation

[2]Despite the existence of partitioning applications throughout the system design cycle, calls for papers of major CAD conferences still implicitly classify partitioning within placement and/or floorplanning. Interestingly, research activity in "pure partitioning" arguably exceeds that in either of these mainstay categories.

systems and rapid system prototyping methodologies (e.g., those from Quickturn or Zycad/Inca) use partitioning tools to map complex circuit designs onto hundreds or even thousands of interconnected FPGAs. Typically, such partitioning instances are challenging because the timing, area, and I/O resource utilizations must satisfy hard device-specific constraints. Furthermore, the partitioning optimization is affected by the discrete nature of system resources – e.g., interconnect delay in routing segments, layout area in configurable logic blocks (CLBs), or individual FPGA chips in a multiple-FPGA system – all of which have large "quanta".

- **Hardware simulation and test.** A good partitioning will minimize the number of inter-block signals that must be multiplexed onto the bus architecture of a hardware simulator or mapped to the global interconnect architecture of a hardware emulator. Reducing the number of inputs to a block often reduces the number of test vectors needed to exercise the logic.

In this work, we survey the major research directions in netlist partitioning and establish a taxonomy of existing works based on the underlying solution methodology. While we have tried to make this survey both complete and self-contained, we emphasize more recent problem formulations and solution approaches, possibly at the expense of methods that have been treated in previous surveys.[3] Section 2 develops notation, discusses various graph and hypergraph representations of the circuit netlist, and formulates basic variants of the partitioning problem. These problem formulations include bipartitioning, multi-way partitioning, constraint-driven partitioning, and partitioning with replication. Sections 3 through 6 survey four major categories of partitioning approaches:

- Section 3 discusses **move-based** approaches: we classify an algorithm in this category if it explores the solution space by moving from one solution to another. Greedy and iterative exchange [117] approaches are most common – these always try to make the best move, but can easily can trapped in local minima. To avoid this behavior, many other strategies have been proposed including stochastic hill-climbing (simulated annealing), evolutionary algorithms, and the multi-start strategy. We discuss these approaches along with many adaptations that have used these methods to address more complex formulations.

- Section 4 discusses methods that construct a **geometric representation** of the partitioning problem via such constructions as a 1-dimensional linear ordering or a multi-dimensional vector space. Such a representation offers possibilities for geometric approaches to solve problems that are intractable for general graphs. *Spectral* methods are commonly used to construct geometric representations, due to their ability to capture global netlist information.

[3] Limited surveys are given in the textbooks [149] and [171]; the former contains a more personal perspective on early works, by W. Donath. The book by Lengauer [133] is noteworthy, especially for its complete development of combinatorial algorithms (network flow, multicommodity flow, etc.).

- Section 5 discusses **combinatorial** approaches; we loosely classify an approach under this category if the partitioning problem can be transformed into some other "classic" type of optimization, e.g., maximum flow, mathematical programming, graph labeling, or set covering. These approaches are promising since complex formulations that include timing, module preassignment, replication, and other hard constraints can often readily be expressed in terms of a mathematical program or flow network. In addition, changing user requirements for solution quality and runtime, as well as improved computing platforms, have made such approaches more practical.
- Section 6 treats **clustering**-based approaches, which traditionally consist of bottom-up approaches that merge netlist modules into small clusters. We augment this class to include methods which use a clustering solution within another algorithm, such as two-phase partitioning or placement. Clustering-based approaches have received much recent attention since they are viewed as the most promising method for tackling the increasing problem sizes in VLSI CAD.

Section 7 concludes with a discussion of benchmarking practice, as well as a brief list of perspectives on future research in the field.

2 Partitioning Formulations

In this section, we describe the major variant formulations of the partitioning problem. We defer the description of several less well-studied variants – e.g., retiming formulations – to the discussions of their respective underlying solution strategies.

2.1 Preliminaries

Given a set of n netlist modules $V = \{v_1, v_2, \ldots, v_n\}$, the purpose of partitioning is to assign the modules to a specified number k of clusters satisfying prescribed properties.[4]

Definition: A *k-way partitioning* $P^k = \{C_1, C_2, \ldots, C_k\}$ consists of k *clusters* (subsets of V), $C_1, C_2, \ldots, C_k$, such that $C_1 \cup C_2 \cup \ldots C_k = V$. If $k = 2$, we refer to P^2 as a *bipartitioning*.

The objective to be optimized is denoted by $F(P^k)$, i.e., the objective is a function of the partitioning solution. We generally make the traditional assumption that the clusters are mutually disjoint; note however that *replication* formulations permit a module to be a member of more than one cluster.

[4]Many works distinguish between the partitioning problem where k is small (e.g., $k \leq 15$) and the clustering problem where k is large (e.g., $k = \Theta(n)$). Such works may refer to what we call as cluster as a "partition"; we choose not to make this distinction, since our multi-way partitioning formulations are independent of the relative size of k. For reasons of clarity, we will generally refer to P^k as a "partitioning" for small k and as a "clustering" for large k.