FLOORPLANNING CHALLENGES IN EARLY CHIP PLANNING

Jeonghee Shin, John A. Darringer, Guojie Luo[1], Merav Aharoni,
Alexey Y. Lvov, Gi-Joon Nam and Michael B. Healy
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598, USA
jeonshin@us.ibm.com

ABSTRACT

Early chip planning is becoming more critical as server system designers strive to explore a large design space with multiple cores and accelerators in an advanced silicon technology that includes 3D chip stacking. During early chip planning, designers search for the high-level design and layout that best satisfies a myriad of constraints and targets. In this paper, we discuss our experience in applying traditional floorplanning tools at this early stage and suggest how they might be adapted for early floorplanning.

I. INTRODUCTION

In the highly competitive server market, there is increasing pressure on system designers to consider more alternatives to meet performance demands with the reduced benefits from technology scaling. At the same time, there are growing concerns about power dissipation limits, temperature limits, current limits, voltage-drop limits and lifetime reliability – all influenced by the chip layout. In the initial stages of server chip design, designers consider a wide range of options for the overall design and try to estimate values for all the design metrics to decide which configuration is the "best." They typically use data from an existing design to estimate the size of the major blocks in the new technology, then make adjustments for planned changes in processor core performance and cache sizes, determine the I/O drivers needed for the required off-chip bandwidth and reconfigure the on-chip interconnect to handle any additional blocks. They also produce a high-level floorplan, using their experience to arrange the high-level blocks, to estimate chip size and consider other constraints. Since arranging the blocks is often a manual process done by an experienced designer, only a few configurations can be considered in detail. With 3D technology there are many additional issues, such as portioning function to the difference layers and considering the placement and impact of through-silicon vias (TSVs). This growing challenge has motivated the pursuit of more automation to enable many more alternatives to be explored with increased accuracy [1]. In this paper, we focus on automated floorplanning in the early stages of server chip design.

For the automation of early floorplanning, we first considered several commercial and academic tools. Note that most of these floorplanning tools were designed to be used much later in the design process, when physical design attributes are known. However, the netlists available at the early stage often include high-level blocks without interior details such as timing and pin locations. Instead, only block diagrams, constraints and power abstracts are available for placement, created for architectural configurations, based on a reference design. As a result, these tools need to be adapted to this very different task in order to generate good floorplans. In this paper, we describe our experience in applying the existing tools at the early stage of server design, and discuss several well-known techniques that can be adopted by the tools for early floorplanning.

In the rest of the paper, Section II and III discuss early floorplanning challenges and existing tools and methodologies. Section IV describes the techniques that are a good fit for early floorplanning and were tried in this paper. Section V discusses their results as well as our experience with existing tools. Section VI concludes the paper with future work.

II. EARLY FLOORPLANNING CHALLENGES

Floorplanning for chip integration is considered a mature area. While early floorplanning shares the basic challenges of the traditional floorplanning, it

[1] Guojie Luo is a PhD student at UCLA. This work was performed while he was an intern at the IBM research center.

978-1-4577-1617-1/11/$26.00 ©2011 IEEE

has the properties that are rarely dealt with in the traditional floorplanning:

- Handling high-level netlists without interior details that consist of a mix of super blocks (several square mm) and large or small blocks (a few square um) in hierarchical structures such as chiplets, units and macros.
- Satisfying constraints to place IO blocks at the chip's edge to ease package fan-out; proximity constraints to keep key components close for delay and routing; constraints to control area, aspect ratio, orientation, symmetry and modularity; and white space for global routing.
- Convenience and runtime scalability for fast yet "good" floorplan generation of many design alternatives and providing a simple interface to system designers, who are not chip integrators.

When applied at the very early stages of design, the traditional placement and floorplanning tools do not perform well. Placement tools are primarily optimized to place millions of tiny standard cells and not tuned for a small number of large blocks, typical in early floorplanning. While some floorplanning tools do handle large blocks, in our experience, it is not convenient to impose enough constraints and obtain the floorplans that are both optimized and legalized. These limitations force even experienced server chip integrators prefer a semi-manual floorplanning method with the aid of their own simple scripts. This manual method may work for a design based on a preceding design. But, it becomes quite inefficient for design space exploration, when a wide range of configurations need to be considered. This is where an automated floorplanning method is necessary.

III. EXISTING FLOORPLANNING TOOLS

In this section, we review several existing tools in more detail for the early floorplanning problem described above. All of the work on floorplanning and placement could not be mentioned here, but we believe that these are representative and frequently referenced ones.

A. Industry Tools

We began with commercial floorplanning tools, which include features for design planning, claimed to be similar to early floorplanning. For example, grouping constraints allow the users to specify the proximity of a set of blocks based on the designer's experience with "good" floorplans. The users may try to preserve absolute spacing between blocks for routing, if the blocks are heavily connected, or indicate a fixed location of certain blocks. However, commercial tools have difficulty handling a system-level netlist, consisting of large cores, tiny interfaces, as well as long and narrow buses. When placing the blocks inside a predefined region becomes difficult, the tools usually stop before optimizing the delay or the desired proximity of the blocks. As a result, manual modification to the generated floorplans is often necessary for reasonable quality.

We also applied IBM's placer [2], which is used primarily for standard cell placement, to some early floorplanning examples to study the limitations of placement tools for early floorplanning, such as

- Poor grouping and alignment of blocks
- Lack of recognition of bus-type interconnects
- Legalization failure, if area utilization is high
- Difficulty in incorporating other objective functions, such as power and temperature

Nonetheless, the IBM Placer did tend to find good wire length solutions - one important objective.

B. Academic Tools and Research

Academic floorplanning and placement research has covered a broad array of optimization styles. We reviewed the studies on mixed-sized placement for the early floorplanning problem since they generally include state-of-the-art floorplanning and placement techniques to place a few large blocks along with many standard cells [3]. For example, the most recent version of Capo [4] produces floorplans by using top-down partitioning on a fixed-size outline and simulated annealing based placement. mPL6 [5] is based on force-directed algorithms, where the floorplanning problem is modeled as a system of masses and springs. The latest version of NTUplace [6] also uses an analytical placement technique to consider pre-placed blocks and special density function control techniques. While the mixed-sized placement techniques are able to handle large blocks, it is not clear that they are suitable for early floorplanning, where typically many blocks are

extra large with the constraints described above. In Section V, we discuss our experience with Capo.

Simulated annealing is another well-known optimization algorithm that has been applied to the floorplanning problem in many incarnations. Since simulated annealing based tools produce legalized solutions and have few limits on the optimization objective, most work on physical planning for architecture evaluation and multi-objective optimization is based on simulated annealing [7, 8]. However, most floorplan representations used during annealing are based on block-packing, and generally not amenable to the reservation of white space that may be required. Further, the theoretical flexibility provided by the cost function is limited in practice due to the large number of times it must be evaluated, leading to significant run times even for cost functions with polynomial complexity. In addition, they often require fine tuning of various input parameters, which is inefficient for early chip planning to quickly evaluate many candidate designs.

IV. USEFUL ANALYTICAL TECHNIQUES FOR EARLY FLOORPLANNING

In this section, we discuss the analytical techniques that can be adopted by the traditional tools to address the early floorplanning problem described in Section II. Note that they can be replaced with any existing similar techniques but we chose them for quick implementation and evaluation by leveraging the tools available to us.

Physical planning of the new design often begins by scaling the macros composing system components such as cores, accelerators, and caches, for technology and architectural changes. Since the relative size and aspect ratio of the macros generally change from the preceding design, the original floorplan needs to be adjusted after scaling, to remove overlaps and unnecessary white space. Such incremental floorplanning can be performed well by migration or annealing algorithms that move blocks while maintaining their relative location.

Once system components are sized and shaped by the incremental floorplanning, they need to be placed by observing a set of constraints. As our attempts to impose the constraints with existing tools were not very successful, we developed a constraint-driven floorplanner based on IBM Haifa's CSP (Constraint Satisfaction Problem) solver [9] to explicitly specify the constraints. In addition, we developed ILP (Integer Linear Programming) [10] and NLP (Non-Linear Programming) [11] based floorplanners to exploit implicit and explicit hierarchy in the netlists, which is common in server chip design, and optimize floorplans for multiple objectives, which is becoming more important with advanced integration and technology scaling. We chose ILP and NLP over other approaches for relatively quick and easy implementation of analytical cost functions with heuristics.

The details of the floorplanning techniques that were tried are described in the following subsections, and the floorplans generated are discussed in Section V, along with their pros and cons for early floorplanning.

A. Migration Algorithm based Floorplanning

We tried incremental floorplanning based on a design migration technique called MASH [12]. MASH has been primarily used for layout migration to new technologies, with the goal of retaining the good placement from the original design. The goal is achieved by adjusting the size and relative placement of the individual devices. MASH has been successfully applied to several memory arrays in server chips.

Layout migration is quite similar to incremental floorplanning, where the macros correspond to the devices in the layout. The MASH based floorplanner consists of four steps:

(1) Extract the floorplan of the preceding design and build a hierarchy graph in which nodes correspond to the macros.
(2) Use a scan line algorithm to detect relationships between any pair of macros and introduce weighted linear constraints on macro's coordinates and hard constraints to fix target macro size.
(3) Generate a new floorplan by minimizing the penalty function, which increases if the relationships are not retained, with a linear programming solver.
(4) Migrate the macros to the new design using MASH to maximize compaction with ground rules encoded as hard constraints.

In step (2), the relationships of interest include: A. two macros share a common boundary segment,

B. the macros are mutually visible, C. the macros share a common straight global wire, and D. two macros belong to a step-and-repeat array.

MASH is not recommended for whole chip floorplanning, because it may not be able to retain the relationships if the design is too large. In addition, run time grows with LP problem size. Nonetheless, the number of macros in our experiments ranges from 2 to 20K and the corresponding run times are 6 to 135 seconds, which is acceptable for system-level floorplanning. While the MASH algorithm is used for our study, other annealing methods adopted by existing tools [4, 5] can be also explored.

B. Constraint Solver based Floorplanning

CSP is a mathematical problem that consists of a set of variables and their constraints, and can be applied to a large class of problems. The CSP solvers have been used in placement problems [9], but not widely because the domains of possible values for locating a block are very large. This approach is more viable for early floorplanning because the number of blocks is relatively small and fewer pins are considered without exact assignment. Although the generated floorplans may need refinement, they can quickly suggest several options to the designer and help the user to define new constraints.

The constraint-driven floorplanner developed is based on IBM Haifa's CSP solver [9], with the floorplanning problem modeled as follows:

- Domains: for all variables, the initial domain is all possible positions on the chip.
- Variables: for every macro, a variable for its position on the chip (x and y coordinates).
- Constraints:
  - A global constraint to avoid overlaps
  - Maximum distance constraints to keep pairs of macros close for wiring
  - Minimum distance constraints to keep pairs of macros apart for power and temperature reasons
  - Specific location constraints for preferred locations, for example a central bus.
  - Overall minimization of area by an iterative scheme

In addition, we used a variable ordering based on the wiring of the blocks, starting from the central instances outward. We modified the value ordering so that values that failed are removed along with their surrounding values, so as to avoid thrashing.

C. Linear / Non-linear Programming based Floorplanning

The advantage of ILP or NLP based analytical modeling is that formulated objectives and constraints can be quickly implemented and evaluated with existing solvers. Our experiments started with basic floorplanning objectives and constraints that were formulated in linear functions and solved by CPLEX [10]. The half perimeter wire length (HPWL) is formulated as the sum of Manhattan distance between blocks. The non-overlap constraints are given with two binary variables for every pair of blocks as proposed in [13]. Since the formulation of the constraints described in Section II is not straightforward, we added some heuristics to the ILP based floorplanner. For instance, the proximity constraint is implemented with grouping. The chip-level netlist is partitioned into several groups composed of the blocks to place close. The groups can be further divided as needed, even though a 2-level hierarchy was sufficient for our examples as shown in Figure 1. In the top level, we consider these groups as soft blocks with a fixed area and a predefined range of aspect ratios, and the floorplan of these groups are solved by the ILP solver. Based on the top-level results, we keep the relative locations of these groups, and work on the bottom-level, where the blocks are confined inside their groups. In the bottom level, the binary variables for the non-overlap constraints only exist inside a group. This hierarchical approach may not produce the globally optimal solution, but does guarantee of optimality at each level.

One limit of the ILP approach is poor runtime scalability. The hierarchical approach reduced the problem size, but it still is not sufficient for larger designs. For large designs, we transformed the discrete linear functions to non-linear formulas similar to density constraints in [5] and use IPOPT [11] as a solver. The NLP based floorplanning is generally poor at legalization, thus an extra step may be needed to remove overlaps. In addition to HPWL, we developed additional objective functions for power density and thermal costs.

Table 1. Summary of the techniques described in Section III for early floorplanning.
+(+) : (very) efficient, +/- : neutral, -(-) : (very) inefficient.

	MASH	CSP	ILP/NLP
	Best for incremental floorplanning	Better for constraint-driven floorplanning	Better for multi-objective, hierarchical floorplanning
Constraints	- -	++	+/-
Legalization	++	+/-	+/-
Multi-objectives	+/-	+/-	++
Hierarchy	++	+	+
Scalability	-	-	+/-
Incremental Change	++	-	-

V. EARLY FLOORPLANNING EXAMPLES

In this section, we apply the tools introduced in Section III and IV to some of the floorplans produced for an abstract design derived from an IBM high-performance network processor chip, Wire Speed Power™ Processor [14] with most top-level units except the IO blocks. The original chip size is 16.7mm x 24.5mm in the IBM 45nm technology, and scaled to a 22nm technology for early chip planning. Figure 1 illustrates a logical view of the example design netlist, where four chiplets with processor cores and caches (EX), accelerators (EA, EB, EC, ED) and controllers (EM, EN) are connected to the central bus (PB) via bus interface (PBI). In addition to the data paths shown in the figure, the netlist includes control bits and pervasive signals. Figure 2 shows the resulting floorplans.

For the commercial floorplanner, we used grouping and spacing parameters to force the main functional blocks to be placed with the bus interface. However, as shown in Figure 2(a), the generated floorplan is far from ideal in terms of area, and even some blocks are far away from the bus or bus interface. Capo tries packing the blocks to optimize HPWL and area, when the preferred proximity between the blocks and the bus is not explicitly specified. The quality of the results might improve with more tuning of user parameters. Even for the simple example, the results confirm the difficulty of imposing constraints, including proximity. The physical distance between the main functional blocks and their bus interface, or between the interface bocks and the bus has direct impact on chip performance, as well as global wire congestion. The HPWL optimization, which is typical for many existing tools, fails to consider this.

In the CSP based floorplanner, the constraints are specified and well satisfied, as shown in Figure 2(c). However, this approach is not the best at area optimization and the chip size is larger than that of the Capo floorplan shown in Figure 2(b). For the ILP or NLP based approaches, we used heuristics to capture the proximity by exploiting the explicit and implicit hierarchy in the netlist, and did floorplanning in the top-level first and lower levels next. In addition, area and HPWL are optimized at each level. Figure 2(d) shows that the ILP based floorplanner performs well in the lower levels, while the top-level floorplan needs improvement on area. The ILP based floorplanner slows down significantly as the number of blocks in the netlist increases, while the NLP based floorplanner scales well, but often requires overlaps to be fixed.

While we discussed the different floorplanning approaches with a simple example, similar observations were made with real server chip designs. In fact, the limitations discussed above are often amplified in floorplanning 3D chips, which have more constraints and objectives to consider. To attack the early floorplanning challenges, we need a better solution.

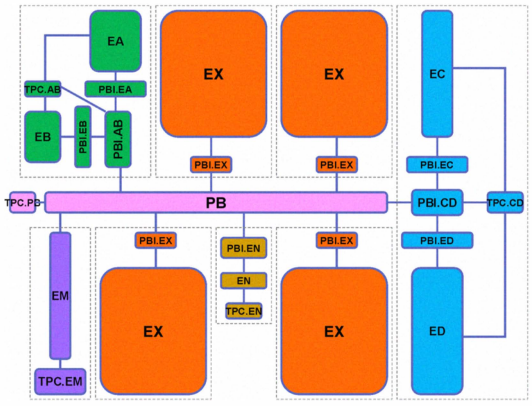

Figure 1. Logical view of the floorplan example.

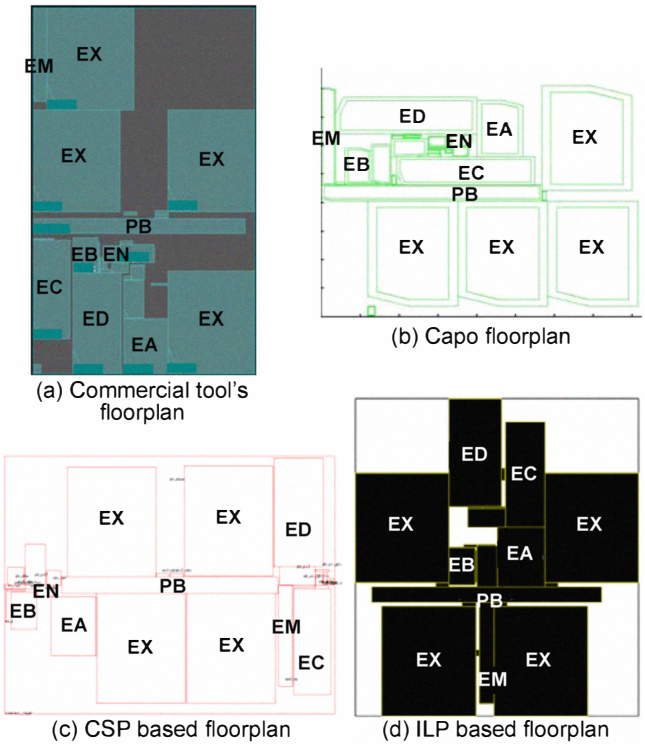

Figure 2. Floorplanning results on the example illustrated in Figure1.

VI. CONCLUSION

In this paper, we addressed the need of automated floorplanning methods to place high-level blocks with the constraints in the very early stages of server design, which differs from the traditional floorplanning of millions of diverse sized bocks during the later stage of chip integration. The automation of early floorplanning is essential for efficient design space exploration, but there exist no "good" automated tools yet. To address this issue, we reviewed several existing floorplanning tools and described our experience in using them for early floorplanning. We also tried and evaluated a few techniques that can improve the existing tools for the early floorplanning problems. The preliminary results were promising, but more research is required on this important problem.

REFERENCES

1. J. Shin, et al., "Early Chip Planning Cockpit", Design, Automation and Test in Europe, 2011.
2. N. Viswanathan, et al., "RQL: Global Placement via Relaxed Quadratic Spreading and Linearization", Design Automation Conference, 2007.
3. S. N. Adya, et al., "Unification of partitioning, placement and floorplanning," International Conference on Computer Aided Design, 2004.
4. UMICH Physical Design Tools, http://vlsicad.eecs.umich.edu/BK/PDtools.
5. T. F. Chan, et al., "mPL6: enhanced multilevel mixed-size placement", International Symposium on Physical design, 2006.
6. NTUplace: A VLSI Placement Tool, http://eda.ee.ntu.edu.tw/w04/download/ntuplace.php.
7. J. Cong, et al., "Microarchitecture evaluation with physical planning," Design Automation Conference, 2003.
8. M. B. Healy, et al., "Multi-objective microarchitectural floorplanning for 2D and 3D ICs," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 26, No. 1, pp. 38-52, 2007.
9. B. Dubrov, et al., "Pin Assignment using Stochastic Local Search Constraint Programming", International Conference on Priniciples and Practice of Constraint Programming, 2009.
10. IBM ILOG CPLEX Optimizer, http://www-01.ibm.com/software/integration/optimization/cplex-optimizer.
11. Interior Point OPTimizer, https://projects.coin-or.org/Ipopt.
12. F.L. Heng, et al., "A VLSI Artwork Legalization Technique Based on a New Criteria of Minimum Layout Perturbation," International Symposium on Physical Design, 1997.
13. S. Sutanthavibul, et al., "An analytical approach to floorplan design and optimization," Design Automation Conference, 1991.
14. C. Johnson et al., "A Wire-Speed PowerTM Processor: 2.3GHz 45nm SOI with 16 Cores and 64 Threads," ISSCC, 2010.