Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

Regularity-constrained floorplanning for multi-core processors

Xi Chen [a,*], Jiang Hu [a], Ning Xu [b]

[a] Department of ECE, Texas A&M University, College Station, TX 77843, USA
[b] College of CST, Wuhan University of Technology, Wuhan 430070, China

ARTICLE INFO

Article history:
Received 22 February 2012
Received in revised form
22 March 2013
Accepted 26 May 2013
Available online 3 June 2013

Keywords:
VLSI
Physical design
Floorplanning
Multi-core processors
Regularity

ABSTRACT

Multi-core technology becomes a new engine that drives performance growth for both microprocessors and embedded computing. This trend requires chip floorplanners to consider regularity constraint since identical processing/memory cores are preferred to form an array in layout. In general, regularity facilitates modularity and therefore makes chip design planning easier. As chip core count keeps growing, pure manual floorplanning will be inefficient on the solution space exploration while conventional floorplanning algorithms do not address the regularity constraint for multi-core processors. In this work, we investigate how to enforce regularity constraint in a simulated annealing based floorplanner. We propose a simple and effective technique for encoding the regularity constraint in sequence-pairs. To the best of our knowledge, this is the first work on regularity-constrained floorplanning in the context of multi-core processor designs. Experimental comparisons with a semi-automatic method show that our approach yields an average of 12% less wirelength and mostly smaller area.

1. Introduction

When the Moore's law is near its end, continuing chip performance growth will inevitably rely on the improvement of system level integration. This is evidenced by the popularity of multi-core technology for both microprocessors and embedded processors. In a foreseeable future, current multi-core processors will advance to many-core processors, which allow hundreds of cores on a chip. This trend presents new challenges to the design and design automation technologies. This paper discusses floorplanning problem for multi-core and many-core processors and proposes an algorithmic solution to this problem.

Floorplanning is the first primary physical design step that decisively affects chip layout, on-chip communication, power, performance and various design concerns [1]. When the number of cores on a chip is small, the floorplanning can be managed by manual designs, especially for Chip Multiprocessors (CMP). For instance, a 4-core processor can be manually placed in a 2×2 array. When the core count exceeds one hundred, the options of floorplans increase dramatically. Then, it would be very difficult for manual design to quickly and thoroughly explore these options. Besides processing cores, a processor chip usually contains cache, I/O blocks and communication fabrics. Further, CMP technology will move from homogeneous to heterogeneous cores [2] like IBM Cell processor. These facts imply heterogeneous entities, which make manual floorplanning even more difficult. Therefore, there is a strong need for automatic floorplanning [3,4] techniques for many-core CMP designs.

Multi-core technology is also widely adopted in System-on-Chip (SoC) designs and leads to the so-called Multi-Processor SoC (MPSoC). SoC designs are often targeted to embedded computing and require much shorter design turn-around time than microprocessors. Although conventional floorplanning techniques are applicable to current MPSoC designs, there is a new problem as the system grows from multi-core to many-core. That is, if multiple identical cores are adopted, usually they are preferred to be placed in a regular array. If ever possible, regularity is desired in chip layout for the sake of design simplicity, modularity and easy management of physical resources.

The regularity issue is rarely considered in the conventional floorplanning. One similar case is analog circuit layout [5–10] where components are often placed in a symmetric fashion. One may want to fulfill the regularity constraint by enforcing the symmetry constraints. Even though symmetry and regularity are related, regularity is actually more complex than symmetry and often more difficult to achieve. A chip with m cores can be placed in a $p\times q$ array and there are often multiple ways for the factorization of $m=p\times q$, e.g., $m=30=1\times30=2\times15=3\times10=5\times6=6\times5\ldots30\times1$. Even for a specific factorization, symmetries to different axes need to be maintained to obtain a regular array. The work of Ref. [9] addressed the array-type constraint for the analog placement. However, there is a key difference between the array constraint in analog placement and regularity constraint in multi-core processor floorplanning. In analog placement, array blocks of the same device type are compacted together in order to reduce the effect of spatially-dependent variations. In multi-core

* Corresponding author. Tel.: +1 979 676 3418.
E-mail address: xichen07@gmail.com (X. Chen).

0167-9260/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.vlsi.2013.05.002

processor floorplanning, in contrast, non-array blocks can be placed between array blocks and one group of array blocks can be placed inside of another group of array blocks. By allowing such option, one may have an opportunity to further reduce interconnect delay and congestion. For example, placing the memory controller in the center of an array of processing cores conceivably causes less interconnect congestion than placing it at peripheral regions.

Nowadays, microprocessor design is very complex and needs to account for many other factors as well. One is the design turn-around time. A regular layout can alleviate design complexity. For example, the inter-core wire connection can repeat the same patterns if the cores are placed in a regular array. This will not only make the routing simpler (by repeating certain patterns) but also reduce verification such as LVS/DRC time. Moreover, an overall regular floorplan can make clock network construction easy. Clock skew is easier to be managed for a regular structure, e.g., one can directly use an H-Tree at the top level of clock network. Further, a regular floorplan also makes the power grid design and verification easier. For cases where wirelength is more important, designers have the freedom to choose non-regular floorplan. If some projects have very tight time-to-market, designers are entitled to choose regular floorplan. Our work is targeted to such scenarios.

In this paper, we present our work on floorplanning with regularity constraint, which is oriented toward multi-core processor designs. Our floorplanner is a simulated annealing algorithm using sequence-pair representation. Our key contributions are on how to encode the regularity constraint in sequence-pair and how to achieve the regularity in packing procedure. To the best of our knowledge, this is the first work studying regularity-constrained floorplanning for multi-core processors. We compared our approach with a naïve method that manually tries multiple placements for array blocks, each of which is followed by conventional floorplanning for non-array blocks while the array blocks are fixed. The experimental results indicate that our approach can achieve an average of 12% less wirelength than the semi-automatic method. At the same time, our approach usually leads to smaller area.

2. Previous work with symmetry constraints

Among previous works on floorplanning, those for analog integrated circuits have the closest problem formulation as our regularity-constrained floorplanning work. Since an analog circuit typically has a small number of elements, its placement is often equivalent to the floorplanning of a digital integrated circuit. In analog circuit designs, one important requirement is to place blocks or devices symmetrically with respect to one common axis so as to improve the tolerance to common-mode noise [10].

There are a large amount of analog circuit layout works [5–16] focusing on the symmetry constraints. In Ref. [5], TCG-S is used as placement representation to do analog layout under symmetry constraint. In Ref. [6], in the context of Silicon on Insulator (SOI), mismatch analysis for analog layout is proposed and tested. In Ref. [7], ad symmetry-constrained analog block placement methods proposed. This work is based on a typical floorplanning approach–simulated annealing with sequence-pair representation. The symmetry constraint is described through sequence-pairs. For a sequence-pair (α, β), α_A^{-1} denotes the position of block A in sequence α. Consider a group of blocks G that must be placed symmetrically around a vertical axis. A sequence-pair (α, β) is symmetric-feasible for G if for any two blocks A and B in G

$$\alpha_A^{-1} < \alpha_B^{-1} \Leftrightarrow \beta_{\sigma(B)}^{-1} < \beta_{\sigma(A)}^{-1} \tag{1}$$

where $\sigma(A)$ is the block symmetric to A. However, it is pointed out in Ref. [12] that condition (1) is sufficient but not necessary. More recently, the work of Refs. [8,13] presented another sequence-pair based approach for simultaneously satisfying symmetry and centroid constraints. Another method based on B*-Tree is proposed in Ref. [14] for handling both 1-D and 2-D symmetry constraints. The other symmetry-constrained analog placement work [15] uses O-Tree representation. In Ref. [16], a symmetry-aware placement work is proposed based on Transitive Closure Graphs (TCG) data structure.

To certain extent, regularity constraint can be treated as an extension to the symmetry constraints. However, the extension is not trivial as the number of implicit symmetry constraints embedded in a regularity constraint can be quite large. More specifically, every spatially contiguous subset of an array group needs to follow its own symmetry. Fig. 1 is a simple example and it has four blocks that need to satisfy the regularity array constraint in (a) and symmetry constraint in (b). In (a), all spatially contiguous subsets {1,2}, {2,3}, {3,4}, {1,2,3} and {2,3,4} need to satisfy their own symmetry constraints. Hence, the regularity array constraint implies significantly more symmetry constraints than the case of analog circuit layout like in (b).

In Ref. [9], the array-type constraint is considered for analog circuit placement. In order to mitigate the effect of PVT (Process, Voltage and Temperature) variations, which are usually spatially correlated, the work of Ref. [9] packs array blocks of the same type right next to each other. In multi-core processor designs, however, the problem granularity level and concerns are different. By allowing non-array blocks to be embedded between array blocks, chip interconnects performance and cost can be reduced. Fig. 2 shows an industrial design where non-array blocks SIU and CCX are placed in between the 2×4 array of L2T blocks and the L2T blocks are placed in between the 2×8 L2D blocks.

3. Floorplanning with regularity constraint

3.1. Problem formulation

The input to floorplanning includes a set of n blocks, each with area A_i where $i=1, 2, \ldots, n$, a set of l nets $N_1, N_2, \ldots, N_l$ among the n blocks, a set of k array groups $G_1, G_2, \ldots, G_k$. Each *array group* is a subset of the given blocks that must be placed in a regular array. If a block is in an array group, it is called an *array block*; otherwise, it is called a *non-array block*. The problem is to construct a floorplan F that satisfies non-overlapping and the regularity constraint, and

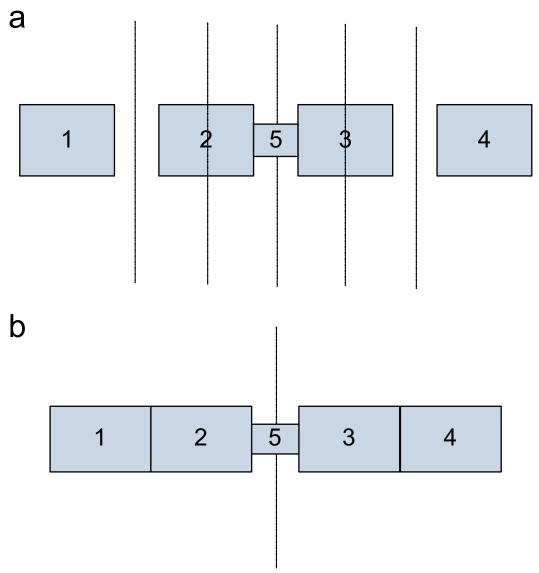

Fig. 1. Comparison between regularity and symmetry constraint: (a) regularity constraint for digital processors and (b) symmetry constraint for analog circuit.