
PREFACE

This handbook for routing interconnects inside a VLSI chip provides mathematical models of important classes of wiring techniques for students interested in gaining insights in integrated circuits layout automation techniques and for practicing engineers working in the field of electronic design automation (EDA). This book presents a comprehensive review on VLSI routing techniques that was undertaken in early 1990's with a view to developing a generalized routing accelerator that could speed up routing chores for different styles of wiring techniques, namely, *maze routing* used widely for connecting different circuit blocks by finding the shortest path, *channel routing* used in connecting standard cells of uniform heights and variable widths arranged in the form of rows of cells, *switchbox routing* used in connecting surrounding multiple blocks of dissimilar aspect ratios within an enclosed routing area, and so on.

In 1988, when I started my academic career at the University of Michigan, I designed a new graduate-level course on computer-aided design, EECS 527: VLSI Layout Algorithms. The course was introduced to educate graduate students and spur doctoral research in the-then burgeoning field of computer-aided design (CAD) for integrated circuits (ICs) that propelled the exponential growth of integration density in VLSI chips, as postulated by Moore's Law. At that time, there was no suitable textbook on the subject to teach graduate students about the state-of-the-art layout algorithms that were key to design complex VLSI chips. Therefore, I combed through the literature on the subject and assembled the course materials in order to teach students systematically basic underlying mathematical techniques for circuit partitioning, floor-planning, cell placement, and routing. Subsequently, I engaged my own doctoral students to expand my lecture materials in the form of comprehensive

reviews.

For example, with the assistance of my doctoral student, Dr. K. Shahookar, who studied the Genetic Algorithm (GA) for VLSI cell placement techniques, I coauthored a 78-page review paper, which was published in *ACM Computing Surveys* in June 1991. After poring over nearly a hundred publications on placement algorithms for standard cells and macro-cells, I divided them into five main categories: (i) the placement by simulated annealing, (ii) the force-directed placement, (iii) the placement by min-cut graph algorithms, (iv) the placement by numerical optimization, and (v) the evolution-based placement. While the first two types of algorithms owe their origin to physical laws, the third and fourth are analytical techniques, and the fifth class of algorithms is derived from biological phenomena. The taxonomy of placement algorithms was created to study inherent parallelism of the different classes of algorithms. While designing the course, I realized that in order to push the mammoth potential of Moore's Law, the chip design phase must be accelerated several folds by harnessing the evolving computing platforms.

In the late 80's, the computing platforms for the VLSI design environment were rapidly transforming from mid-frame computers, namely, Digital Equipment Corporation Vax 11/780, Hewlett Packard HP 3000, and Wang Laboratories Wang VS, to the network of workstations, what is widely known as the NOW. This opportunity in hardware evolution warranted deeper insights into VLSI cell placement and routing (P&R) techniques so that sequential algorithms that used to run on standalone mid-frame computers could be rendered into parallel CAD algorithms for running efficiently on the NOW platform. Also, emergence of commercial parallel computers such as Intel hypercube and Sequent Computer System shared memory had further pushed the needs for developing parallel P&R algorithms.

In order to promote the NOW platform for EDA research, I started working with my students to develop imaginative distributed Genetic Algorithms (GAs) for partitioning, placement and floor-planning techniques needed in VLSI chip layout automation. My research group had at that time developed an EDA tool, named *Wolverines* for parallel implementation of standard cell placement algorithms on the NOW platform. Since workstations are connected by a local area network (LAN) that often deploys the Ethernet to connect different workstations, communication of packets

between two specific workstations generally require considerable time even when the Ethernet did not undergo collision of message packets. Because of the length of a LAN, two workstations located afar may locally sense and infer that the Ethernet is free and may launch packets asynchronously. In case, there is a collision of packets, all the senders must abandon transmission by backing off. Then they wait randomly within the range of time before attempting to transmit the packet. If a sender encounters the collision of packet again, it then waits randomly over a period of time that is twice longer than the previous time period. This exponential backing off protocol used in random-access LAN causes a severe constraint to run parallel routing algorithms because of their fine-granularity of parallelism in contrast with placement algorithms that do not require frequent communications in parallel mode of operation over the NOW.

Therefore, the vision I had at that time is to develop distributed version of Genetic Algorithms (GAs) that can accelerate the partitioning, floor-planning and placement of cells on the NOW without requiring any hardware augmentation of workstations. The main impediments we encountered at that time were that unlike simulated annealing and graph-based techniques, the GAs could not be directly applied to solve VLSI layout problems. Professor John Holland, who invented the Genetic Algorithm at the University of Michigan, had devised the GA to solve a plethora of theoretical and practical problems that quintessentially required *functional optimization*. The GA applies its biology-inspired operators such as crossover, mutation and inversion in the genotype or chromosomal representation of the problem. The fundamental premise in GA is that genetic codes of biological creatures encapsulate their physical characteristics. The crossover operator in the GA produces offspring by splicing fragments of two different chromosomes pertaining to the two parents. Therefore, the GA applies all its transformations by combining the features of chromosomes on the genotype of a problem. In order to obtain the solution of a problem, the genotype is mapped on to its phenotype or physical appearance. It is well known that the functional optimization problems work very well with GA type algorithms since there are no inconsistencies between the genotype and phenotype of a problem. In other words, the genetic codes always generate feasible solutions in functional optimization problems.

However, the EDA algorithms for placement is a *constrained combinatorial*