Physical Design of VLSI Circuits and the Application of Genetic Algorithms*

Jens Lienig
Tanner Research, Inc.
2650 East Foothill Blvd.
Pasadena, CA 91107

Email: jens@ieee.org
Phone: (626) 792-3000
Fax : (626) 792-0300

Abstract

The task of VLSI physical design is to produce the layout of an integrated circuit. New performance requirements are becoming increasingly dominant in today's sub-micron regimes requiring new physical design algorithms. Genetic algorithms have been increasingly successful when applied in VLSI physical design in the last 10 years. Genetic algorithms for VLSI physical design are reviewed in general. In addition, a specific parallel genetic algorithm is presented for the routing problem in VLSI circuits.

1 Introduction

Electronic design automation is concerned with the design and production of VLSI systems. One of the important steps in creating a VLSI circuit is its physical design. The input to the physical design step is a logical representation of the system under design. The output of this step is the layout of a physical package that optimally or near-optimally realizes the logical representation. Physical design problems are generally combinatorial in nature.

What makes electronic design automation problems particularly difficult compared to traditional combinatorial optimization problems is that the

*This article appeared in *Evolutionary Algorithms in Engineering Applications*, Springer Verlag, Berlin, pp. 277-292, 1997.

number of elements that must be handled can be quite large – a circuit can be easily composed of over one million gates. For this reason, design automation practitioners have a strong tradition of quickly considering and adapting new and alternative solution techniques. For example, simulated annealing is an optimization technique that emulates the annealing of crystals. This combinatorial optimization method was first proposed in the literature in 1983 and by the following year, the major design automation conferences had multiple sessions on simulated annealing for design automation [35]. Early adoption was again repeated on neural networks [36] which simulate the organizing principles of nervous systems.

Although design automation did not immediately add genetic algorithms to its basic tool chest, genetic algorithms have been consistently used in the field for the last ten years.

The purpose of this chapter is to provide the reader with an up-to-date overview of genetic algorithm applications for the VLSI physical design process. In Section 2, below, we first briefly describe the VLSI physical design process. We present a systematic review of genetic algorithms that have been successfully applied to the physical design process in Section 3. In Section 4 we present a parallel genetic algorithm for the channel and switchbox routing problem in VLSI circuits. We conclude with a summary in Section 5.

2 Overview of VLSI Physical Design

The major steps in a typical VLSI design process are shown in Figure 1. In particular, the substeps of physical design are given.

The physical design phase is an important part of this process. Its input is generally a logical description of the circuit, often in the form of a netlist. The task of the physical design step is to produce a layout, an assignment of geometric coordinates to the circuit components, either in the plane or in a specified number of planar layers. The layout must satisfy the requirements of the fabrication technology (sufficient spacing between components of the circuit, and so on) and should minimize certain cost criteria (the lengths of the interconnections, etc.).

Due to its complexity, the physical layout problem is generally divided into subproblems which can be solved sequentially. These subproblems are still NP-hard, but they reduce the practical complexity to a manageable level. The physical design problem is usually decomposed into the following subproblems: *partitioning*, *placement*, *routing* and *compaction*.

Partitioning is the task of dividing a circuit into smaller parts in order to reduce the problem size. The circuit is often divided into portions that are implemented separately. The goal is to partition the circuit such that the sizes of the parts are within prescribed ranges and the complexity of the

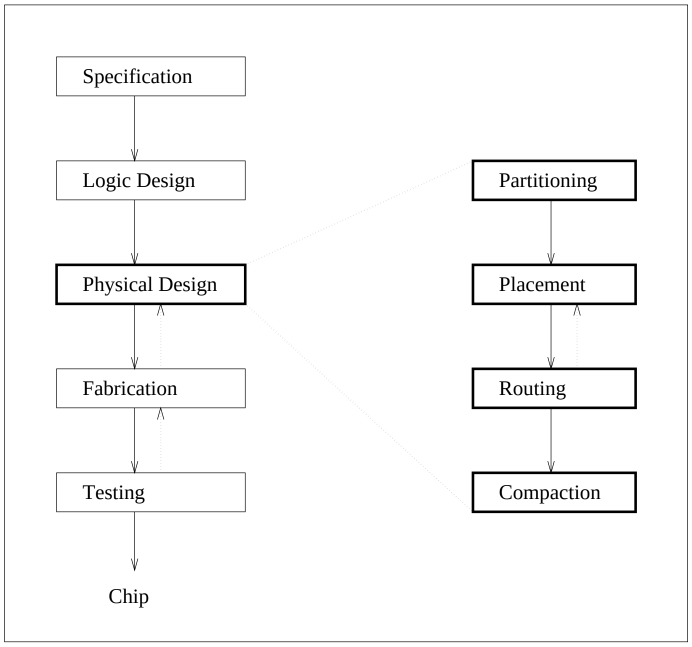


Figure 1: The design process of VLSI chips.

interconnections between these parts is minimized.

Placement assigns the cells of the circuit to their geometrical locations on the chip. (A cell may be a single transistor, an adder or subcircuit, etc.) The objective of placement depends on the design style. In *standard cell design* (where all cells have the same width and are placed in rows) and in *macro cell design* (where cells have different sizes and are placed irregularly), the goal is to minimize the total layout area of the chip. In *gate-matrix design* (where all cells are placed in a matrix pattern) the objective of placement is to ensure routability and to minimize congestion of the interconnections between the cells.

Routing follows the placement phase. It determines the paths of the interconnections between the cells laid out during the placement procedure. The goal is to connect all pins that belong to the same net, subject to certain quality constraints (such as minimizing the lengths of interconnections, and so on) and routing constraints (interconnections must not short-circuit or cross one another, etc.).

Compaction is often the final step in the physical layout design. It transforms the symbolic layout (produced by the preceding steps) into a mask layout, the geometric mask features on the silicon. The objective of compaction is to minimize the size of the resulting circuit layout.

3 Application of Genetic Algorithms to VLSI Physical Design

3.1 Partitioning

To our knowledge, the first evolution-based algorithm for solving the partitioning problem was published in [37]. In contrast to previous heuristic algorithms that usually optimize on only one constraint, this approach is capable of handling both a number of constraints and a number of objectives. The algorithm, which does not include a crossover operator, yields multi-way partitions with fairly balanced sizes and a small number of pins for each part. The presented strategy has a reasonable execution speed that is similar to other published heuristic approaches.

In [21] and [22], different coding schemes for the problem of circuit partitioning are investigated to find the most suitable coding. The proposed genetic algorithm is specifically tailored for the partitioning of circuits with complex bit-slice components using a special two-step coding of partitions. The algorithm consists of a crossover and a mutation operator and a deterministic improvement strategy.

The genetic algorithm in [9] is based on a population structure that involves subpopulations which have their isolated evolution occasionally interrupted by inter-population communication. Although the investigated problems are from actual VLSI design efforts, comparisons with other approaches and runtimes are not presented.

A hybrid genetic algorithm for the ratio-cut partitioning problem is presented in [4]. Here the problem is formulated in terms of a hypergraph. The genetic encoding is a binary string, where each subset of the circuit's components has a corresponding location on the string. Before the genetic algorithm is executed, the ordering of these genes is determined by a depth-first search to improve the performance of the genetic algorithm. Traditional crossover and mutation operators are combined with a fast partitioning heuristic applied to each offspring as an improvement operator. The performance of the algorithm is compared with two other partitioning approaches, using benchmark data sets. Averaged over all benchmarks, the presented algorithm achieves better results than the other approaches, while having a similar amount of runtime for smaller graphs and less runtime for the largest graphs.

Another hybrid approach is published in [42]. It combines a simulated annealing method with a genetic algorithm. The main motivation for this approach is the parallelization of the simulated annealing strategy by replacing its single solution search process with a population-based approach using a genetic algorithm. The two benchmark results that are presented are not compared with other state-of-the-art approaches.

3.2 Placement

As mentioned in Section 2, the placement procedure is responsible for the assignment of the circuit's cells to their locations on the chip. According to the variation in size and location of these cells, placement algorithms can be divided into algorithms for standard cell design, macro cell design and gate-matrix design.

After the pioneering work of [6], further applications of genetic algorithms [39], [40] and evolution strategies [24], [25] for standard cell placement have been presented. These approaches produce high-quality placements of real-world VLSI circuits that can compete with sophisticated simulated annealing-based placement strategies. However, the published runtimes are not as competitive (up to 6 hours [24] and up to 12 hours [40]).

In [33], the runtime has been reduced significantly by using a parallel implementation of a genetic algorithm that runs on a distributed network of workstations. The total population is split over different processors and a migration mechanism is used to exchange genetic material between them. While the placement results are similar to a serial genetic algorithm, an almost linear speedup can be achieved with this method.

We next discuss three investigations that use genetic algorithms for macro cell placement [5], [11], [12]. The approach in [5] is based on a two-dimensional bitmap representation of the macro cell placement problem. Another representation scheme, a binary tree, is applied in [11]. In [12], a combination of a genetic algorithm with a simulated annealing-based optimization strategy is presented. The experimental results suggest that a mixed strategy performs better than a pure genetic algorithm for the macro cell placement problem. The results are better or comparable to previously published results of placement benchmarks. However, the runtime is not as competitive.

An application of a genetic algorithm for the placement of gate-matrix design has been published in [41]. The approach uses the Genesis package [19] as the basic genetic algorithm. This package is modified with a special algorithm for constructing permutations that considers only a small subset of the solution space. The results are compared with only one previously published algorithm. The runtime is in the order of minutes (up to 1 hour).

3.3 Routing

Routing is the process of connecting pins subject to a set of routing constraints. VLSI routing is usually divided into *global routing* (to assign nets into certain routing regions) and *detailed routing* (to assign nets to exact positions inside a routing region).

To our knowledge, only one genetic algorithm for global routing has been reported [13]. The algorithm is based on a two-phase router. In the first phase, a genetic algorithm for the Steiner problem in a graph is used to generate a number of distinct, alternative routes for each net. Then, in a second phase, another genetic algorithm selects a specific route for each net (among the alternatives given from phase one), such that the overall layout area is minimized. The router is superior to TimberWolfMC [38], a state-of-the-art global router, with respect to solution quality, while being inferior with respect to runtime.

According to the position of the pins, detailed routing can be separated into *channel routing* (pins are only located on two parallel sides of the routing area) and *switchbox routing* (pins are placed on all four sides of the routing area).

Several papers have been published in which genetic algorithm-derived strategies are applied to the unrestrictive[1] channel routing problem [15], [17], [30], [32].

In [32], a rip-up-and-rerouter is presented which is based on a probabilistic rerouting of nets of one routing structure. However, the routing is done by a deterministic Lee algorithm [27] and main components of genetic algorithms, such as the crossover of different individuals, are not applied. Results are presented only for one channel routing benchmark. No runtime for this example is given.

The router in [15] combines the steepest descent method with features of genetic algorithms. The crossover operator is restricted to the exchange of entire nets and the mutation procedure performs only the creation of new individuals. The presented results are limited to simple VLSI problems, and no runtime remarks are made.

The genetic algorithm for channel routing published in [30] is based on a problem-specific representation scheme, i.e. individuals are coded in three-dimensional chromosomes with integer representation. The genetic operators are also specifically developed for the channel routing problem. The results are either qualitatively similar to or better than the best published results for channel routing benchmarks. The runtime of the algorithm (in the range of 1...50 minutes) is not as competitive.

The algorithms in [15], [32] are also applied to switchbox routing. While the router in [15] is not usable for large switchbox routing benchmarks, the algorithm in [32] can compete with other switchbox routing algorithms. However, the runtime is not given.

A different genetic algorithm for switchbox routing is presented in [31]. Similar to [30], the genotype is essentially a lattice corresponding the coor-

[1] Approaches for the restrictive channel routing problem (where all vertical net segments are located on one layer and all horizontal segments are placed on a second layer) cannot be applied to real-world VLSI channel routing problems and thus, won't be considered here.

dinate points of the layout. Crossover and mutation are performed in terms of interconnection segments. The algorithm assumes that the switchbox is extendable in both directions. Subsequently, these extensions are reduced with the goal to reach the fixed size of the switchbox. While more costly in runtime, on numerous benchmark examples the genetic algorithm produces solutions with equal or better routing characteristics (netlength, number of vias) than the previously best published results.

3.4 Compaction

As mentioned in Section 2, compaction transforms the symbolic layout to a mask layout with the goal of minimizing the size of the resulting circuit layout.

To the best of our knowledge, the only application of a genetic algorithm for compaction has been advanced by Fourman [14]. He describes two prototypes of genetic algorithms that perform compaction of a symbolic circuit layout. Although his results are limited to very simple layout structures, he does propose a new problem-specific representation for layout design that includes constraints of the compaction process.

4 A Parallel Genetic Algorithm for the VLSI Routing Problem

In the following, we present a parallel genetic algorithm to solve the VLSI channel and switchbox routing problems with the objective of satisfying crosstalk constraints for the nets. This approach is an extension of [30] in which a sequential genetic algorithm was applied to channel routing problems. In [28] we first introduced the parallel genetic approach and extensively investigated its main parameters.

4.1 Introductory Remarks

As mentioned earlier, interconnection routing is one of the major tasks in the physical design of VLSI circuits. Pins that belong to the same net are connected subject to a set of routing constraints. Channel and switchbox routing are the two most common routing problems in VLSI circuits. Simple examples of a channel routing problem and a switchbox routing problem are shown in Figure 2.

Our motives for developing a parallel genetic algorithm for the detailed routing problem have been threefold. First, almost all previously published detailed routing strategies only consider physical constraints, such as the

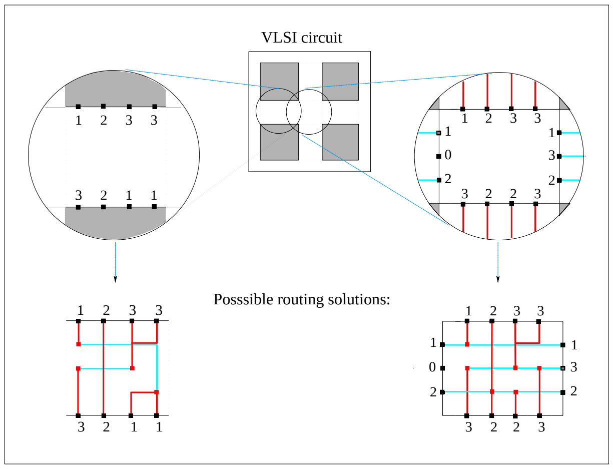


Figure 2: The VLSI channel (left) and switchbox (right) routing problem and possible routing solutions.

netlength. However, with further minimization in VLSI design, new electrical constraints, such as crosstalk, are becoming dominant and need to be addressed. Second, today's typical computer-aided design environment consists of a number of workstations connected together by a high-speed local network. Although many VLSI routing systems make use of the network to share files or design databases, none of the known routing programs (evolution-based or deterministic algorithms) use this distributed computer resource to parallelize and speed up their work. Third, all published genetic algorithms that address the routing problem are sequential approaches, i.e., *one* population evolves by means of genetic operators. However, recent publications (e.g. [2],[26]) clearly indicate that parallel genetic algorithms with isolated evolving subpopulations (that exchange individuals from time to time) perform better than sequential approaches.

We present a parallel genetic algorithm for detailed routing, called GAP (Genetic Algorithm with Punctuated equilibria), that runs on a distributed network of workstations. Our approach considers routing quality characteristics such as the the netlength, the number of connections between layers, and the importance of crosstalk between neighboring interconnections. Due to variable weight factors, these routing objectives can be easily adjusted to the requirements of a given VLSI technology. Furthermore, on many benchmark examples, the router produces better results than the best of those previously published.

4.2 Problem Description

The VLSI routing problem is defined as follows. Consider a rectangular routing region with *pins* located on two parallel boundaries (*channel*) or four boundaries (*switchbox*) (see Figure 2). The pins that belong to the same net need to be connected subject to certain constraints and quality factors. The interconnections need to be made inside the boundaries of the routing region on a symbolic routing area consisting of horizontal *rows* and vertical *columns.*

We define a *segment* to be an uninterrupted horizontal or vertical part of a net. (Thus, any connection between two pins will consist of one or more net segments.) A connection between two net segments from different layers is called a *via.* The overall length of all segments of one net to connect its pins is defined as its *netlength.*

In sub-micron regimes, crosstalk results mainly from coupled capacitance between adjacent (parallel routed) interconnections. The shorter the length of these parallel routed segments, the better the performance of the circuit.

Thus, the following three factors (which are to be minimized) are used in this work to assess the quality of the routing:

- netlength,
- number of vias, and
- crosstalk.

4.3 Overview of the Parallel Genetic Algorithm

Different ways exist to parallelize a genetic algorithm [2]. However, most of these methods result only in a speed-up of the algorithm without qualitative improvements to the problem solutions. To gain better problem solutions, we use the theory of *punctuated equilibria* to design a parallel genetic algorithm [7],[10]. A genetic algorithm with punctuated equilibria is a parallel genetic algorithm in which independent *subpopulations* of individuals with their own *fitness functions* evolve in isolation except for an exchange of individuals (*migration*) when a state of equilibrium throughout all the subpopulations has been reached (see Figure 3). Previous research has shown genetic algorithms with such punctuated equilibria to have superior performance when compared to sequential genetic approaches [7],[9].

The parallel structure of our algorithm for the case of nine processors is shown in Figure 4. We assign a set of n individuals (problem solutions) to each of the N processors, for a *total population* size of $n \times N$. The set assigned to each processor, c, is its subpopulation, $\mathcal{P}_c$. The processors are connected by an interconnection network with a torus topology. Thus, each processor (subpopulation) has exactly four *neighbors.*

The genetic algorithm used by each processor and the main process that steers the parallel execution are presented in Figure 5. First, the main process

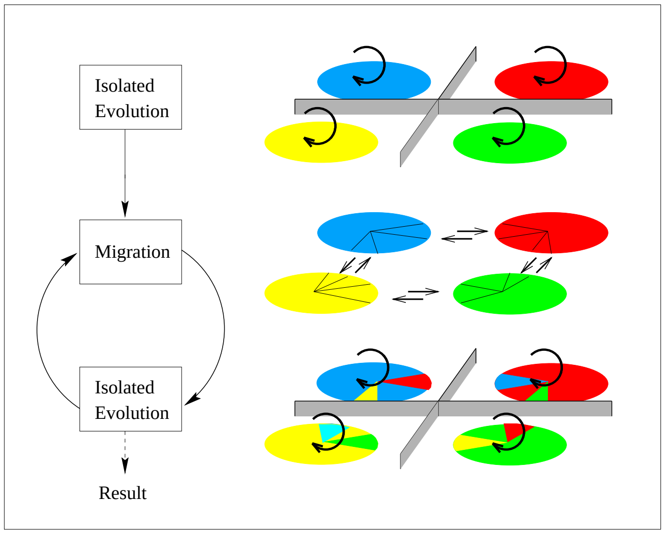


Figure 3: Punctuated equilibria model.

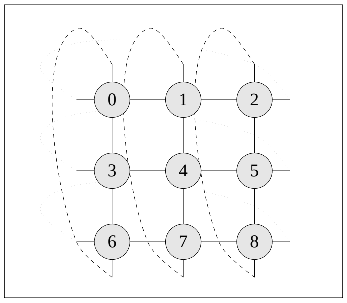
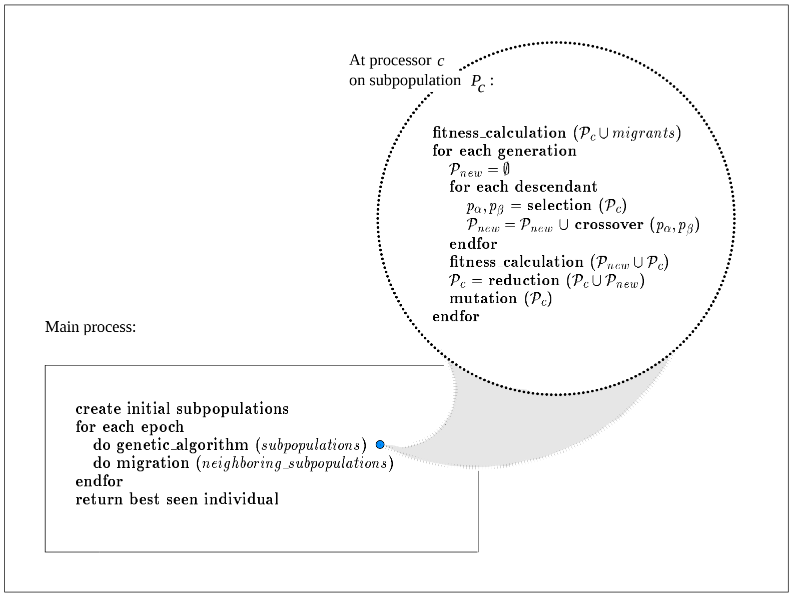


Figure 4: Neighborhood structure with nine subpopulations.



```
fitness_calculation ($\mathcal{P}_c \cup migrants$)
for each generation
  $\mathcal{P}_{new} = \emptyset$
  for each descendant
    $p_\alpha, p_\beta$ = selection ($\mathcal{P}_c$)
    $\mathcal{P}_{new} = \mathcal{P}_{new} \cup$ crossover ($p_\alpha, p_\beta$)
  endfor
  fitness_calculation ($\mathcal{P}_{new} \cup \mathcal{P}_c$)
  $\mathcal{P}_c$ = reduction ($\mathcal{P}_c \cup \mathcal{P}_{new}$)
  mutation ($\mathcal{P}_c$)
endfor
```

```
create initial subpopulations
for each epoch
  do genetic_algorithm ($subpopulations$)
  do migration ($neighboring\_subpopulations$)
endfor
return best seen individual
```

Figure 5: Algorithm overview.

creates an initial subpopulation at each processor. This initial subpopulation consists of randomly constructed (i.e., not optimized) routing solutions. They are designed by a random routing strategy which connects net points in an arbitrary order with randomly placed interconnections. (See [30] for a detailed description of our random routing strategy.) The main process consists of a given number of *epochs*. During an epoch, each processor, disjointly and in parallel, executes the sequential genetic algorithm on its subpopulation for a certain number of generations (*epoch length*). Afterwards, each subpopulations exchanges a specific number of individuals (*migrants*) with its four neighbors. The process continues with the separate evolution of each subpopulation during the next epoch. At the end of the process, the best individual that exists constitutes our final routing solution.

The following section briefly describes the genetic operators used by each processor to evolve its subpopulation.

4.4 Genetic Operators

Fitness Calculation The fitness $F(p)$ of each individual $p \in \mathcal{P}_c$ is calculated to assess the quality of its interconnections relative to the rest of the subpopulation $\mathcal{P}_c$. The following factors are taken into account (with different weights) when determining $F(p)$:

- overall netlength of p,
- number of vias of p, and
- the length of adjacent, parallel interconnections (crosstalk).

After the evaluation of $F(p)$ for all individuals of the subpopulation $\mathcal{P}_c$ these values are scaled linearly [18], in order to control the relative range of fitness in the subpopulation.

Selection Our selection strategy, which is responsible for choosing the mates for the crossover procedure, is stochastic sampling with replacement [18]. That means any individual $p_i \in \mathcal{P}_c$ is selected with a probability proportional to its fitness value.

Crossover During a crossover, two individuals are combined to create a descendant. Our crossover operator is a 1-point crossover operator [18] that gives high-quality routing parts of the mates an increased probability of being transferred intact to their descendant.

Crossover is performed in terms of wire segments. A randomly positioned line (crossline) perpendicular to the edges of the routing area divides this area into two sections, playing the role of the crosspoint. This line can be either horizontally or vertically placed. For example, interconnection segments *exclusively* on the upper side of the crossline are inherited from the first parent, and segments *exclusively* on the lower side of the crossline are inherited from the second parent. Segments intersecting the crossline are newly created within the descendant by means of our random routing strategy [30].

(A more detailed description of our crossover operator is given in [30].)

Reduction Our reduction strategy simply chooses the $|\mathcal{P}_c|$ fittest individuals of $(\mathcal{P}_c \cup \mathcal{P}_{new})$ to survive as $\mathcal{P}_c$ into the next generation.

Mutation The mutation operator performs random modifications on an individual (to overcome local optima) by applying the random routing strategy [30] on randomly selected interconnections.

4.5 Experimental Results

Our algorithm, called GAP, has been implemented on a network of SPARC workstations (SunOS and Solaris systems). The parallel computation environment is provided by the Mentat system, an object-oriented parallel processing system [20].

4.5.1 Parameter Settings

The main parameters (see Section 4.3) were set to:

Individuals per subpopulation :	50
Descendants per subpopulation :	20
Number of subpopulations :	9
Number of epochs :	10
Epoch length (generations) :	50

Two randomly selected migrants were sent to each of the four neighbors in each epoch.

(See [28] for a detailed discussion of these parameters as well as a comparison with a sequential genetic approach.)

4.5.2 Comparison of GAP to Other Routing Algorithms

Any application of a genetic algorithm should focus on a comparison to solution techniques that have been acknowledged as effective by that application's community. Here we compare the results of GAP with the best known results of other algorithms for channel and switchbox routing benchmarks (see Table 1). The other routing algorithms do not consider crosstalk, and thus can only be compared with our routing results regarding netlength and number of vias.

We ran our algorithm 50 times per benchmark with different initialization of the random number generator. Table 1 presents the best-ever-seen results for all algorithms. We note that for GAP the best-ever-seen quality was achieved in at least 50 percent of the program executions.

It can be seen that our results are qualitatively similar to or better than the best known results from popular channel and switchbox routers published for these benchmarks. The layout of Burstein's difficult switchbox achieved with our algorithm is depicted in Figure 6.

4.5.3 Reduction of Crosstalk

By adjusting the value of the weight for crosstalk (see Section 4.4), our algorithm can also optimize the interconnections regarding crosstalk. The results presented in [29] show that an increase of the weight for crosstalk leads to significantly fewer parallel routed net segments. Hence, our router can construct solutions which contain minimal coupling capacitances between interconnections – an increasingly significant consideration in sub-micron VLSI design.

However, as discussed in [29], the minimization of crosstalk leads in general to an increase in both the netlength and the number of vias. It has to be decided by the user to which optimization goals he/she gives priority.

Bench-mark	Algorithm	Col-umns	Rows	Net-length	Vias	Time (sec)
Joo6_13	WEAVER[23]	18	7	167	29	312
	PACKER[16]	18	6	167	25	710
	SAR[1]	18	6	166	25	70
	GAP	18	6	164	22	172
Joo6_16	WEAVER[23]	11	7	121	21	220
	Monreale[15]	11	7	120	19	?
	GAP	11	6	115	15	207
Burstein's Difficult Switchbox	WEAVER[23]	23	15	531	41	1508
	BEAVER[a][8]	23	15	547	44	1
	PACKER[16]	23	15	546	45	56
	GAP	23	15	538	36	1831
Dense Switchbox	Silk[32]	16	17	516	29	?
	SAR[1]	16	17	519	31	150
	GAP	16	17	516	29	2380
Augmented Dense Switchbox	BEAVER[a][8]	16	18	529	31	1
	PACKER[16]	16	18	529	32	31
	SAR[1]	16	18	529	31	205
	GAP	16	18	529	29	2281

[a] BEAVER's number of vias has been adjusted.

Table 1: Comparison of GAP with the best-known results for some benchmark channels (upper half) and switchboxes (lower half).

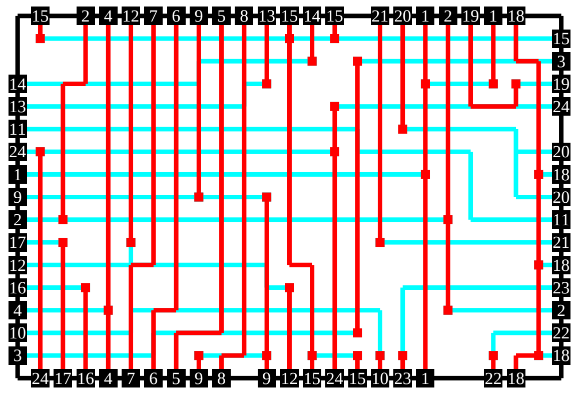


Figure 6: Our routing solution of Burstein's difficult switchbox.

5 Summary

We presented a systematic review of genetic algorithm investigations for the VLSI physical design process. These contributions are generally different than standard genetic algorithm investigations. One difference is that the genetic operators for physical design algorithms are typically very problem-specific. This specificity occurs because of the extreme importance of determining very high quality solutions – therefore expert information on the likely form of solutions is included as much as possible. Combinations of genetic algorithms with other optimization strategies are no longer an exception. Once the "high quality regions" are identified by a genetic algorithm, the application of local search routines are often the only way to ensure effective runtimes. Another difference is the concern for robustness. There exists a rich collection of design automation benchmarks (e.g., [34]) and for a solution method to be accepted, it must be demonstrated to work consistently well on those benchmarks.

We also presented a parallel genetic algorithm for the channel and switchbox routing problem. Our results are qualitatively similar to or better than the best known results from popular channel and switchbox routers. In addition, our algorithm is able to significantly reduce the occurance of crosstalk.

Genetic algorithms have a very large potential within physical design of VLSI circuits. The problems encountered in this field are extremely complex which is exactly the situation in which the performance of a genetic algorithm compares best to that of other methods. However, genetic-algorithm-based approaches are of practical interest to the VLSI community only if they are competitive with the acknowledged existing approaches with respect to performance and runtime. This chapter gave a review of the current situation in this field with the purpose of stimulating and guiding further applications of genetic algorithms.

References

[1] A. Acan and Z. Ünver, "Switchbox Routing by Simulated Annealing: SAR," in *Proc. IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 1985-1988, 1992.

[2] P. Adamidis, *Review of Parallel Genetic Algorithms*, Technical Report, Aristotle University of Thessaloniki, 199

[3] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, Reading, MA: Addison-Wesley, 1990.

[4] T. N. Bui and B. R. Moon, "A Fast and Stable Hybrid Genetic Algorithm for the Ratio-Cut Partitioning Problem on Hypergraphs", *Proc. of the ACM-IEEE Design Automation Conference*, pp. 664-669, 1994.

[5] H. Chan, P. Mazumder and K. Shahookar, "Macro-Cell and Module Placement by Genetic Adaptive Search with Bitmap-Represented Chromosome," *Integration, The VLSI Journal*, vol. 12, no. 1, pp. 49-77, Nov. 1991.

[6] J. P. Cohoon and W. D. Paris, "Genetic Placement," *IEEE Trans. on Computer-Aided Design*, vol. 6, no. 6, pp. 956-964, Nov. 1987.

[7] J. P. Cohoon, S. U. Hedge, W. N. Martin, and D. S. Richards, "Punctuated Equilibria: A Parallel Genetic Algorithm," *Proc. Second International Conference on Genetic Algorithms,* pp. 148-154, 1987.

[8] J. P. Cohoon and P. L. Heck, "BEAVER: A Computational-Geometry-Based Tool for Switchbox Routing," *IEEE Trans. on Computer-Aided Design*, vol. 7, no. 6, pp. 684-697, 1988.

[9] J. P. Cohoon, W. N. Martin, and D. S. Richards, "Genetic Algorithms and Punctuated Equilibria in VLSI," *Parallel Problem Solving from Nature,* H. P. Schwefel and R. Männer, eds., Lecture Notes in Computer Science, vol. 496, Berlin: Springer Verlag, pp. 134-144, 1991.

[10] N. Eldredge and S. J. Gould, "Punctuated Equilibria: An Alternative to Phyletic Gradualism," *Models of Paleobiology,* T. J. M. Schopf, ed., San Francisco, CA: Freeman, Cooper and Co., pp. 82-115, 1972.

[11] H. Esbensen, "A Genetic Algorithm for Macro Cell Placement," *Proc. of the European Design Automation Conference*, pp. 52-57, Sept. 1992.

[12] H. Esbensen and P. Mazumder, "SAGA: A Unification of the Genetic Algorithm with Simulated Annealing and its Application to Macro-Cell Placement," *Proc. of the 7th International Conference on VLSI Design*, pp. 211-214, Jan. 1994.

[13] H. Esbensen, "A Macro-Cell Global Router Based on Two Genetic Algorithms" *Proc. of the European Design Automation Conference*, pp. 428-433, Sept. 1994.

[14] M. P. Fourman, "Compaction of Symbolic Layout using Genetic Algorithms," *Proc. of the First International Conference on Genetic Algorithms*, pp. 141-153, 1985.

[15] M. Geraci, P. Orlando, F. Sorbello and G. Vasallo, "A Genetic Algorithm for the Routing of VLSI Circuits," *Euro Asic '91*, Parigi 27-31 Maggio, Los Alamitos, CA: IEEE Computer Society Press, pp. 218-223, 1991.

[16] S. H. Gerez and O. E. Herrmann, "Switchbox Routing by Stepwise Reshaping," *IEEE Trans. on Computer-Aided Design*, vol. 8, no. 12, pp. 1350-1361, 1989.

[17] N. Göckel, G. Pudelko, R. Drechsler, B. Becker, "A Hybrid Genetic Algorithm for the Channel Routing Problem," *Proceedings of the 1996 IEEE International Symposium on Circuits and Systems, ISCAS-96*, pp. 675-678, 1996.

[18] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning,* Reading, MA: Addison-Wesley, 1989.

[19] J. J. Grefenstette and N. N. Schraudolph, *A User's Guide to GENESIS 1.2 UCSC*, CSE Dept., University of California, San Diego, 1987.

[20] Homepage: "http://www.cs.virginia.edu/~mentat/".

[21] M. Hulin, "Analysis of Schema Distributions," *Proc. of the Fourth International Conference on Genetic Algorithms*, pp. 204-209, 1991.

[22] M. Hulin, "Circuit Partitioning with Genetic Algorithms Using a Coding Scheme to Preserve the Structure of a Circuit," *Parallel Problem Solving from Nature,* H. P. Schwefel and R. Männer, eds., Lecture Notes in Computer Science, vol. 496, Berlin: Springer Verlag, pp. 75-79, 1991.

[23] R. Joobbani, *An Artificial Intelligence Approach to VLSI Routing*, Boston, MA: Kluwer Academic Publishers, 1986.

[24] R. M. Kling and P. Banerjee, "ESP: Placement by Simulated Evolution," *IEEE Trans. on Computer-Aided Design*, vol. 8, no. 3, pp. 245-256, March 1989.

[25] R. M. Kling and P. Banerjee, "Optimization by Simulated Evolution with Applications to Standard Cell Placement," *Proc. of the 27th ACM-IEEE Design Automation Conference*, pp. 20-25, 1990.

[26] B. Kröger, *Parallel Genetic Algorithms for Solving the Two-Dimensional Bin Packing Problem* (in German), Ph.D. Thesis, University of Osnabrück, 1993.

[27] C. Y. Lee, "An Algorithm for Path Connections and its Applications," *IRE-Trans. on Electronic Computers*, pp. 346-365, 1961.

[28] J. Lienig, "A Parallel Genetic Algorithm for Two Detailed Routing Problems", *Proceedings of the 1996 IEEE International Symposium on Circuits and Systems, ISCAS-96*, pp. 508-511, 1996.

[29] J. Lienig, "Channel and Switchbox Routing with Minimized Crosstalk – A Parallel Genetic Approach", to appear in: *Proceedings of the 10th International Conference on VLSI Design*, Jan. 1997.

[30] J. Lienig and K. Thulasiraman, "A Genetic Algorithm for Channel Routing in VLSI Circuits," *Evolutionary Computation*, vol. 1, no. 4, pp. 293-311, 1994.

[31] J. Lienig and K. Thulasiraman, "GASBOR: A Genetic Algorithm for Switchbox Routing in Integrated Circuits," *Progress in Evolutionary Computation,* X. Yao, ed., Lecture Notes in Artificial Intelligence, vol. 956, Berlin: Springer Verlag, pp. 187-200, 1995.

[32] Y.-L. Lin, Y.-C. Hsu and F.-S. Tsai, "SILK: A Simulated Evolution Router," *IEEE Trans. on Computer-Aided Design*, vol. 8, no. 10, pp. 1108-1114, Oct. 1989.

[33] S. Mohan and P. Mazumder, "Wolverines: Standard Cell Placement on a Network of Workstations," *IEEE Trans. on Computer-Aided Design*, vol. 12, no. 9, pp. 1312-1326, Sept. 1993.

[34] B. T. Preas, "Benchmarks for Cell-based Layout Systems," *Proc. of the ACM-IEEE Design Automation Conference*, pp. 319-320, 1987.

[35] *Proc. of the ACM-IEEE Design Automation Conference*, 1984.

[36] *Proc. of the ACM-IEEE Design Automation Conference*, 1987.

[37] Y. Saab and V. Rao, "An Evolution-Based Approach to Partitioning ASIC Systems," *Proc. of the ACM-IEEE Design Automation Conference*, pp. 767-770, 1989.

[38] C. Sechen, *VLSI Placement and Global Routing Using Simulated Annealing*, Boston, MA: Kluwer Academic Publishers, 1988.

[39] K. Shahookar and P. Mazumder, "GASP - A Genetic Algorithm for Standard Cell Placement," *Proc. of the European Design Automation Conference*, pp. 660-664, 1990.

[40] K. Shahookar and P. Mazumder, "A Genetic Approach to Standard Cell Placement using Meta-Genetic Parameter Optimization", *IEEE Trans. on Computer-Aided Design*, vol. 9, no. 5, pp. 500-511, May 1990.

[41] K. Shahookar, W. Khamisani, P. Mazumder and S. M. Reddy, "Genetic Beam Search for Gate Matrix Layout," *Proc. of the 6th International Conference on VLSI Design*, pp. 208-213, Jan. 1993.

[42] J. M. Varanelli and J. P. Cohoon, "Population-Oriented Simulated Annealing: A Genetic/Thermodynamic Hybrid Approach to Optimization," *Proc. of the Sixth International Conference on Genetic Algorithms*, pp. 174-181, 1995.