Channel and Switchbox Routing with Minimized Crosstalk — A Parallel Genetic Algorithm Approach

Jens Lienig
Tanner Research
180 North Vinedo Avenue
Pasadena, CA 91107, U.S.A.
jens.lienig@tanner.com

Abstract

Reduction of crosstalk between interconnections becomes an important consideration in today's VLSI design. This paper presents a novel approach to solve the VLSI channel and switchbox routing problems with the objective of satisfying crosstalk constraints for the nets. The approach is based on a parallel genetic algorithm which runs on a distributed network of workstations. All our routing results are qualitatively better or as good as the best published results. In addition, our algorithm is able to significantly reduce the occurrence of crosstalk.

1. Introduction

Interconnection routing is one of the major tasks in the physical design of VLSI circuits. Pins that belong to the same net are connected together subject to a set of routing constraints. With new performance requirements for the design, routing constraints such as crosstalk between interconnections are becoming increasingly dominant in sub-micron regimes [3]. Hence, it is inevitable to develop routing tools for the reduction of crosstalk.

Crosstalk between two interconnections is proportional to their coupling capacitance. The coupling capacitance is proportional to the coupling length of the two interconnections (the total length of their overlapping segments) and inversely proportional to their separation. (Crosstalk between two interconnections also depends on the frequency of the signals in these wires. In order to simplify our presentation, we assume that the circuit operates at a fixed frequency.)

Algorithms for minimized crosstalk routing have been presented in [4],[8],[10],[11] and [22]. The solutions in [4],[8] and [22] are based on variable grid spacings to satisfy the crosstalk constraints. However, these solutions are difficult to implement on gridded VLSI routing problems. In [10] and [11], a conventional routing algorithm is first used to generate an initial routing solution with conventional objectives (e.g., channel height). The wire segments in the initial routing solution are then re-assigned to satisfy the crosstalk constraints and to minimize the total crosstalk in the nets.

We present a genetic algorithm approach for the channel and switchbox routing problem in VLSI circuits which directly minimizes crosstalk during the generation of the routing solutions. To our knowledge, this is the first approach which includes crosstalk considerations directly in a gridded VLSI routing process. Furthermore, our algorithm addresses the increased importance of the relationship between electrical delay and netlength by minimizing a nonlinear function of the lengths of the nets.

This paper is an extension of [18],[19] in which a sequential genetic algorithm was applied to channel routing problems. In [20] we first introduced the parallel genetic approach and extensively investigated its main parameters.

2. Problem formulation

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*). 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*.

With the advances in VLSI technology, the relationship between electrical delay and netlength becomes more important [3]. Thus, new measures of netlenghts are needed that reflect the electrical delay better than the commonly used minimization of the sum of the lengths of all nets. It can be assumed that in most cases the electrical delay of an interconnection is proportional to a quadratic function of its length. Thus, rather than minimizing the sum of the lengths of the nets, we minimize a nonlinear function of their individual lengths.

Crosstalk between two parallel routed net segments de-

0-8186-7755-4/96 $05.00 © 1996 IEEE

creases as their separating distance increases. Since it can be assumed that crosstalk between two non-adjacent net segments will be shielded by other nets between them, we simplify our computation by considering crosstalk only between adjacent net segments. We note that our algorithm can be easily extended to consider crosstalk between non-adjacent net segments as well.

Resulting from these considerations, three factors are used in this work to assess the quality of the routing:

- Netlength: We minimize a function which considers the length of each net with a quadratic growth. Thus, we put an increased "pressure" on the longest nets to be minimized.
- Number of vias: The number of vias should be as small as possible.
- Crosstalk: Crosstalk is expressed as the sum of the crosstalks in all nets, which in turn, is proportional to the length of parallel segments adjacent to each net. Thus, we minimize the overall sum of parallel, adjacent net segments for each net.

3. Description of the parallel genetic algorithm

3.1. Outline

Genetic algorithms carry out optimization by computations analogous to biological evolutionary processes [14]. A population of individuals representing different problem solutions is subjected to genetic operators, such as *selection*, *crossover*, and *mutation*. Using these operators, the individuals are steadily improved over many generations, and eventually the best individual resulting from this process is presented as the solution to the problem.

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 [5],[9]. A genetic algorithm with punctuated equilibria is a parallel genetic algorithm in which independent *subpopulations* of individuals with their own *fitness function* evolve in isolation, except for an exchange of individuals (*migration*) when a state of equilibrium throughout all the subpopulations has been reached (see Figure 1). Previous research has shown genetic algorithms with punctuated equilibria to have superior performance when compared to sequential genetic approaches [5],[7].

The parallel structure of our algorithm for the case of nine processors is shown in Figure 2. 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

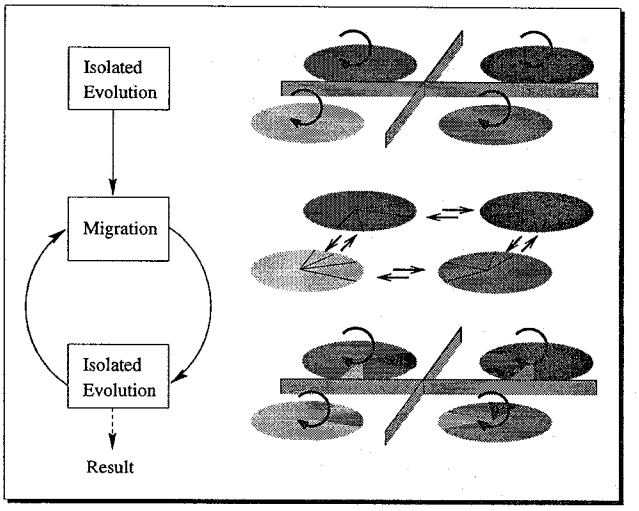

Figure 1: Punctuated equilibria model.

topology. Thus, each processor (subpopulation) has exactly four *neighbors*.

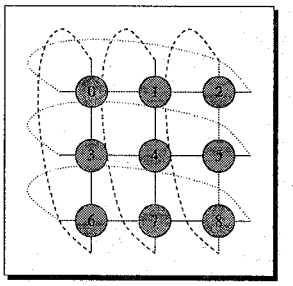

Figure 2: Neighborhood structure with nine subpopulations.

The genetic algorithm used by each processor and the main process that steers the parallel execution are presented in Figure 3. First, the main process 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 [18] for a detailed description of our random routing strategy.) The main process consists of max_epoch iterations, called *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 subpopulation 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 algorithm, the best individual that exists (or has existed) constitutes our final routing solution.

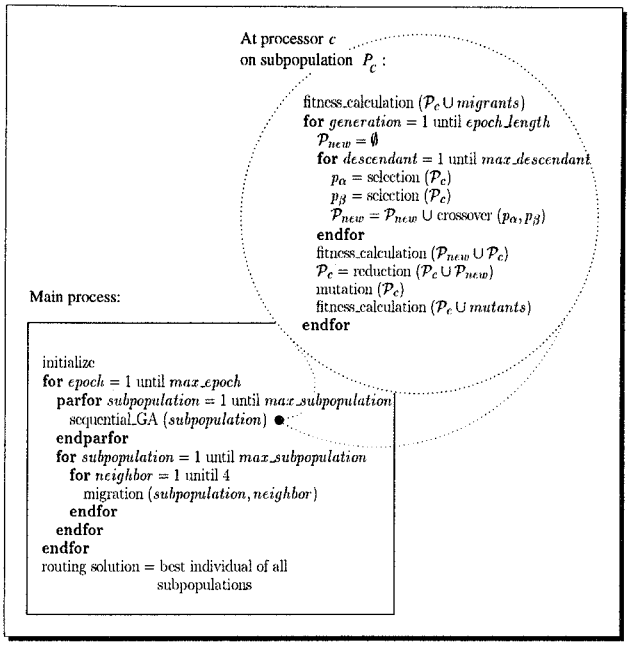

Figure 3: Algorithm overview.

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

3.2. Genetic operators

Fitness calculation The fitness $F(p_i)$ of each individual $p_i \in \mathcal{P}_c$ is calculated to assess the quality of its routing relative to the rest of the subpopulation $\mathcal{P}_c$. The selection of the mates for crossover and the selection of individuals which are transferred into the next generation are based on these fitness values.

First, a raw fitness function $F'(p_i)$ is calculated for each individual $p_i \in \mathcal{P}$ according to Equation 1.

$$F'(p_i) = \frac{1}{w_1 * l_p + w_2 * v_p + w_3 * p_p} \quad (1)$$

where l_p = netlength as the sum of a quadratic function of the length of each net of p_i,
v_p = number of vias of p_i, and
p_p = crosstalk, represented by the length of adjacent net segments (summarized over all nets) of p_i.

It is important to note that the variable weight factors $w_1, ..., w_3$ enable us to easily adjust routing quality characteristics, including the tolerance of crosstalk, to the requirements of a given VLSI technology.

The final fitness values $F(p_i)$ for all individuals of the subpopulation $\mathcal{P}_c$ are determined by linearly scaling $F'(p_i)$, as described in [14], 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 [14]. That means any individual $p_i \in \mathcal{P}_c$ is selected with a probability given by the following equation:

$$Prob\{p_i \text{ is selected}\} = \frac{F(p_i)}{\sum_{p \in \mathcal{P}_c} F(p)} \quad (2)$$

Crossover During a crossover, two individuals are combined to create a descendant. Our crossover operator is a 1-point crossover operator [14] 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. Interconnection segments *exclusively* on, say, 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 [18].

A simple example of a crossover procedure is shown in Figure 4.

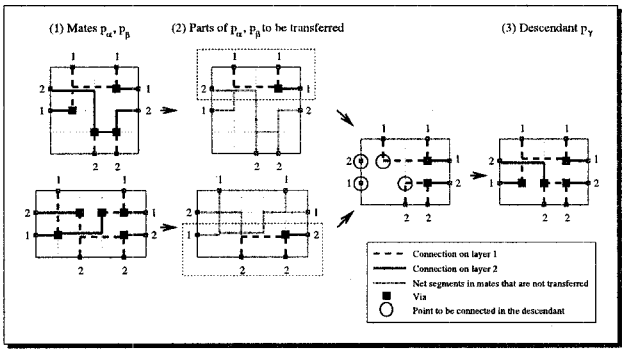

Figure 4: Crossover of mates p_α and p_β to create a descendant p_γ.

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 [18] on randomly selected interconnections.

4. 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 [15],[16]. The program, written in C++ and Fortran, is comprised of approximately 10,000 lines of source code. Our experimental results have been achieved with the machines running their normal daily loads in addition to our algorithm.

4.1. Parameter settings

The main parameters (see Section 3.1) were set to:

Individuals per subpopulation $\|\mathcal{P}_c\|$:	50
$max\_descendant$:	20
$max\_subpopulation$:	9
max_epoch :	10
$epoch\_length$:	50

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

(See [20] for a detailed discussion of these parameters as well as a comparison with a sequential genetic approach.)

4.2. Comparison of GAP to other routing algorithms

First, we compare the results of GAP with those 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. Hence, we kept the weight factor for crosstalk, w_3, at a low level (0.01). The other weight factors in Equation 1 are set to: w_1=1.0 and w_2=2.0.

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 5.

Benchmark	Algorithm	Columns	Rows	Netlength	Vias	Time (sec)
Burstein's Difficult Channel	PACKER[13]	12	4	82	10	87
	Monreale[12]	12	4	82	10	?
	GAP	12	4	82	8	16
Joo6_12	WEAVER[17]	12	4	79	14	134
	PACKER[13]	12	4	82	18	6
	GAP	12	4	79	14	23
Joo6_13	WEAVER[17]	18	7	167	29	312
	PACKER[13]	18	6	167	25	710
	SAR[1]	18	6	166	25	70
	GAP	18	6	164	22	172
Joo6_16	WEAVER[17]	11	8	131	23	220
	WEAVER[a][17]	11	7	121	21	216
	Monreale[12]	11	7	120	19	?
	GAP	11	6	115	15	207
Joo6_17	WEAVER[17]	11	9	166	19	325
	Silk[21]	11	9	166	18	?
	GAP	11	9	165	16	217
Pedagogical Switchbox	BEAVER[b][6]	15	16	396	38	1
	PACKER[13]	15	16	406	45	91
	SAR[1]	15	16	393	31	146
	GAP	15	16	394	29	682
Burstein's Difficult Switchbox	WEAVER[17]	23	15	531	41	1508
	BEAVER[b][6]	23	15	547	44	1
	PACKER[13]	23	15	546	45	56
	GAP	23	15	538	36	1831
Dense Switchbox	WEAVER[a][17]	16	17	517	31	1087
	Silk[21]	16	17	516	29	?
	SAR[1]	16	17	519	31	150
	GAP	16	17	516	29	2380
Augmented Dense Switchbox	BEAVER[b][6]	16	18	529	31	1
	PACKER[13]	16	18	529	32	31
	SAR[1]	16	18	529	31	205
	GAP	16	18	529	29	2281

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

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

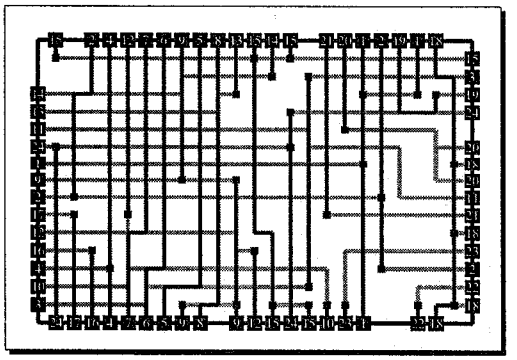

Figure 5: Our routing solution of Burstein's difficult switchbox ($w_1 = 1.0$, $w_2 = 2.0$, $w_3 = 0.01$).

4.3. Crosstalk reduction

By adjusting the value of the weight w_3, our algorithm can also optimize the interconnections regarding crosstalk. Hence, our router can construct solutions which contain a minimal number of parallel, adjacent interconnections.

The length of all adjacent net segments of net i (i.e., the length of the segments that are routed adjacent to i) is denoted by the parameter $netcross(i)$. The parameter $maxcross(i)$ symbolizes the maximal tolerable crosstalk for net i by expressing the maximal tolerable length of adjacent segments of i. Thus, $netcross(i) > maxcross(i)$ represents a violation of the crosstalk constraint of net i and can be easily detected already during the routing process. The parameter $sumcross$ denotes the sum of $netcross(i)$ over all nets.

Table 2 presents the routing results that have been achieved by varying w_3. Since no maximum tolerable crosstalks in the nets were specified for the four benchmarks we used, $maxcross(i)$ was set to a value which we consider appropriate. The results show that an increase of w_3 leads to significant less parallel routed net segments ($sumcross$) and less violations of the individual crosstalk requirement ($netcross(i) \leq maxcross(i)$) of each net i. However, as can be seen in Table 2, 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	w_3=0.01			w_3=1.0			w_3=4.0		
	NV[a]	sc[b]	V[c]	NV[a]	sc[b]	V[c]	NV[a]	sc[b]	V[c]
Burstein[d]	82/10	52	3	84/11	46	2	94/15	42	0
Joo6_13	167/25	141	3	172/26	138	3	181/30	133	0
Joo6_16	120/19	132	4	122/20	130	3	128/21	125	0
Joo6_17	165/16	190	4	167/19	187	4	181/24	177	1

[a] Netlength/number of vias.
[b] $sumcross$.
[c] Number of constraint violations $netcross(i) \leq maxcross(i)$.
[d] Burstein's Difficult Channel.

Table 2: Comparison of the results by varying the weight factor for crosstalk, w_3, for three channels and one switchbox ($w_1 = 1.0$, $w_2 = 2.0$). The results per benchmark are averaged over five runs.

5. Summary

We presented a parallel genetic algorithm for two detailed routing problems in VLSI circuits. Our approach includes a new measure of the netlength to better reflect the electrical delay in sub-micron regimes. Importantly, our approach also optimizes the interconnections involving crosstalk by introducing crosstalk as a constraint in the fitness calculation. Hence, our router can construct solutions which contain a minimal number of parallel, adjacent interconnections – an increasingly significant consideration in sub-micron VLSI design.

Further work will concentrate on including the *netspecific* crosstalk constraint ($netcross(i) \leq maxcross(i)$) directly into the fitness calculation. Thus, instead of globally reducing the number of adjacent net segments, the algorithm will then focus only on critical nets.

References

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

[2] P. Adamidis, *Review of Parallel Genetic Algorithms*, Technical Report, Dept. of Electr. and Comp. Eng., Aristotle Univ. of Thessaloniki, 1994. Email: adamidis@eng.auth.gr.

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

[4] H.H. Chen and C.K. Wong, "Wiring and Crosstalk Avoidance in Multi-Chip Module Design", *Proc. Custom Integrated Circuits Conference*, pp. 28.6.1-28.6.4, 1992.

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

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

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

[8] W.M. Dai, R. Kong, J. Jue and M. Sato, "Rubber Band Routing and Dynamic Data Representation", Proc. International Conference on Computer-Aided Design, pp. 52-55, 1990.

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

[10] T. Gao and C.L. Liu, "Minimum Crosstalk Channel Routing", *Proc. International Conference on Computer-Aided Design*, pp. 692-696, 1993.

[11] T. Gao and C.L. Liu, "Minimum Crosstalk Switchbox Routing", *Proc. International Conference on Computer-Aided Design*, pp. 610-615, 1994.

[12] M. Geraci, P. Orlando, F. Sorbello and G. Vasallo, "A Genetic Algorithm for the Routing of VLSI Circuits", *Proc. Euro Asic '91*, pp. 218-223, 1991.

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

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

[15] A.S. Grimshaw, "Easy-to-Use Object-Oriented Parallel Processing with Mentat", *IEEE Computer*, vol. 26, no. 5, pp. 39-51, 1993.

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

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

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

[19] J. Lienig and K. Thulasiraman, "A New Genetic Algorithm for the Channel Routing Problem", *Proceedings of the Seventh International Conference on VLSI Design*, pp. 133-136, 1994.

[20] 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.

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

[22] A. Onozawa, K. Chaudhary, E.S. Kuh, "Performnace Driven Spacing Algorithms Using Attractive and Repulsive Constraints for Submicron LSI's", *IEEE Trans. Computer-Aided Design*, vol. 14, no. 6, pp. 707-719, 1995.