

A Parallel Genetic Algorithm for Performance-Driven VLSI Routing

Jens Lienig, *Member, IEEE*

Abstract—This paper presents a novel approach to solve the VLSI (very large scale integration) channel and switchbox routing problems. The approach is based on a parallel genetic algorithm (PGA) that runs on a distributed network of workstations. The algorithm optimizes both physical constraints (length of nets, number of vias) and crosstalk (delay due to coupled capacitance). The parallel approach is shown to consistently perform better than a sequential genetic algorithm when applied to these routing problems. An extensive investigation of the parameters of the algorithm yields routing results that are qualitatively better or as good as the best published results. In addition, the algorithm is able to significantly reduce the occurrence of crosstalk.

Index Terms—Channel routing, crosstalk, parallel computation, parallel genetic algorithm, punctuated equilibria, switchbox routing, VLSI physical design.

I. INTRODUCTION

INTERCONNECTION routing is one of the major tasks in the physical design of very large scale integration (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 submicron regimes [4]. Hence, new algorithms are needed to meet the severe topological and electrical constraints posed by current VLSI circuit design. *Performance-driven routing* addresses these performance-related routing constraints. In light of this trend, performance-driven routing has been the main focus of routing-related algorithm development in the last couple of years (e.g., [5], [10], [11], [14], [15]).

Channel and switchbox routing are the two most common routing problems in VLSI circuits. Examples of channel routing and switchbox routing problems are shown in Fig. 1.

One of the main challenges of the routing process of submicron regimes is *crosstalk*. Crosstalk results mainly from coupled capacitance between adjacent (parallel routed) interconnections. With further minimization in design, and thus further reduction of the distance between interconnections, crosstalk is becoming an important electrical constraint and it is going to be more so in the future [4], [28].

Another electrical constraint which is increasingly important is *electrical delay*. This is defined as the time it takes for signals to propagate through the circuit. As integrated circuit

Manuscript received September 16, 1996; revised January 29, 1997. This work was supported by the German Science Foundation (DFG).

The author is with Tanner Research, Inc., Pasadena, CA 91107 USA (e-mail: jens.lienig@tanner.com).

Publisher Item Identifier S 1089-778X(97)03304-3.

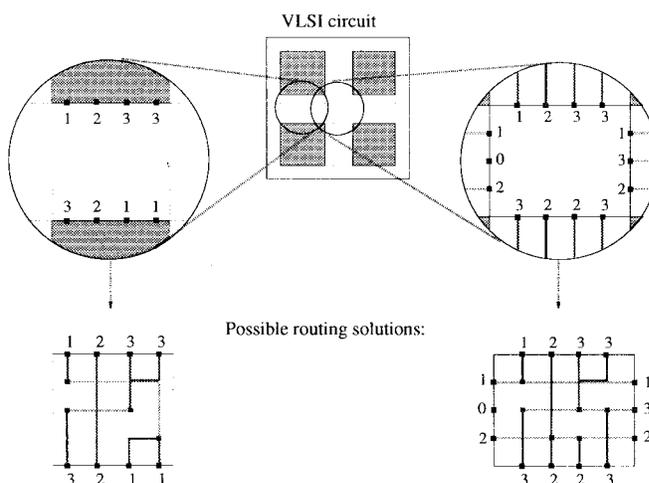


Fig. 1. The VLSI channel (left) and switchbox (right) routing problem and possible routing solutions.

features decrease, electrical delay is increasingly governed by the routing delay (rather than delay within the logic cells) and as a consequence needs to be considered in the routing process.

Our motivations to present an evolution-based algorithm for the detailed routing problem are threefold. First, many previously published detailed routing strategies only consider physical constraints, such as the netlength and the number of vias (defined in Section II). However, with further minimization in VLSI design, new electrical constraints 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 indicate that parallel genetic algorithms (PGA's) with isolated evolving subpopulations (that exchange individuals from time to time) may offer advantages over sequential approaches [2], [7], [9], [20], [23].

We present a PGA for detailed routing, called GAP (genetic algorithm with punctuated equilibria), that runs on a distributed network of workstations. 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.

We show that our parallel approach performs better than a sequential genetic algorithm when applied to the channel and switchbox routing problem. Furthermore, on many benchmark examples, the router produces better results than the best of those previously published. We examine the performance of GAP while varying important parameters of a PGA.

The contributions of this paper are:

- a formulation of a PGA that is capable of handling the VLSI routing problem with both topological *and* electrical constraints, in particular, a consideration of crosstalk minimization directly during the routing process;
- comparisons of the performance of our algorithm with previous routing strategies;
- comparisons of the solution quality of our parallel approach based on the punctuated equilibria model with a sequential genetic algorithm running under the same constraints;
- an investigation of the influence of various parallelization parameters on the routing results.

Throughout this work, we will use the term PGA to describe a genetic algorithm with multiple populations (population structures). Accordingly, “sequential genetic algorithm” indicates a genetic algorithm with a single population (panmictic). This usage is consistent with many previous papers. However, it is important to note that “parallel” and “sequential” refer to population structures, not the hardware on which the algorithms are implemented. In particular, the PGA could be simulated on a single processor platform (as any discrete parallel process can) and the sequential genetic algorithm could be executed on a multiprocessor platform.

II. 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*) (see Fig. 1). 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*. Two *layers* are available for routing in our model.

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 and is referred to as an *interconnection*. A connection between two net segments on different layers is called a *via*. The overall length of all segments of one net used to connect its pins is defined as its *netlength*.

With the advances in VLSI technology, the relationship between electrical delay and netlength becomes more important [4]. Thus, new measures of netlength are needed that reflect the electrical delay better than the commonly used minimization of the sum of the lengths of all nets. One such measure is accomplished by minimizing a nonlinear function

of the lengths of the nets. Under many technologies, a first-order approximation to the electrical delay is the product of the resistance and capacitance of the interconnection [10]. With a fixed width for the interconnection, this product is a quadratic function of the length of the interconnection. Thus, in most cases the electrical delay of an interconnection is proportional to a quadratic function of its length. Hence, rather than minimizing the sum of the lengths of the nets, we minimize a quadratic function of their individual lengths.

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.)

Crosstalk between two parallel routed net segments decreases as their separating distance increases. Since it can be assumed that crosstalk between two nonadjacent net segments will be shielded by other nets between them, we simplify the computation by considering crosstalk only between adjacent net segments. We note that the algorithm can be easily extended to consider crosstalk between nonadjacent net segments as well.

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

- **Netlength:** We minimize a function that considers the length of each net with a quadratic growth. Thus, an increased “pressure” is placed on the longest nets to be minimized, because these nets are mainly responsible for the delay in the routing.
- **Number of Vias:** The number of vias should be as small as possible.
- **Parallel Routed Net Segments:** 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.

Hence, as common in VLSI layout design, the routing problem belongs to the domain of multi-objective optimization (see [13] for a good introduction in this topic including different solution strategies with evolutionary algorithms). We use an objective function that is composed of terms which represent our three objectives combined with weight factors. Our goal is to minimize the sum of these terms by measuring the cost of the solution with respect to user-defined weights for each of the objectives.

III. PREVIOUS WORK

A. Minimum Crosstalk Routing

Algorithms for minimized crosstalk routing have been presented in [5], [11], [14], [15], and [28]. The solutions in [5], [11], and [28] are based on variable grid spacings to satisfy the crosstalk constraints. However, these solutions are difficult to implement on gridded VLSI routing problems.

In [14] and [15], 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 reassigned to satisfy the crosstalk constraints and to minimize the total crosstalk in the nets.

The above-mentioned strategies lead to routing solutions with significantly less crosstalk in the nets. However, it is important to note that the crosstalk minimization takes place *after* the routing procedure and thus is limited to a modification of the conventional routing solution.

B. Genetic Algorithms for the Routing Problem

Several papers have been published in which genetic-algorithm-derived strategies are applied to the routing problem of VLSI circuits [16], [17], [24]–[26], [29]–[32].

In [26], a rip-up-and-rerouter is presented which is based on a probabilistic rerouting of nets of one routing structure. However, the routing is accomplished by a deterministic routing algorithm, and main components of genetic algorithms, such as the crossover of different individuals, are not applied. Results are presented for channel and switchbox routing benchmarks. No runtimes for these examples are given.

The router in [16] 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 figures are shown.

The proposed algorithms in [29]–[32] are limited to the restrictive channel routing problem. Here, all vertical net segments are located on one layer, and all horizontal segments are placed on a second layer. This and other restrictions make these approaches unusable for real VLSI channel routing problems.

The genetic algorithm for channel routing published in [24] is based on a problem-specific representation scheme, i.e., individuals are coded in three-dimensional (3-D) 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 is not as competitive.

A genetic algorithm for switchbox routing is presented in [25]. Similar to [24], the genotype is essentially a lattice corresponding to the coordinate points of the layout. Crossover and mutation are performed in terms of interconnection segments. The algorithm assumes that the switchbox is expandable in both directions. Subsequently, these extensions are reduced with the goal to reach the fixed size of the switchbox. On numerous benchmark examples, the router produces results equal to or better than the previously best published results, while not being runtime competitive.

In [17], a genetic algorithm for the channel routing problem is presented that includes a rip-up-and-reroute strategy. The initial population is created with a shortest-path algorithm combined with random decision making. The published re-

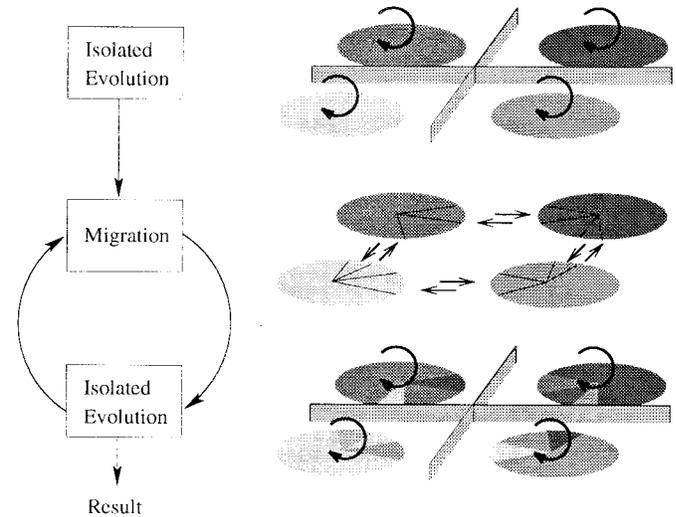


Fig. 2. Punctuated equilibria model with four subpopulations. Subpopulations evolve in isolation (“Isolated Evolution”), periodically interrupted by a limited exchange of individuals (“Migration”).

sults are equal to the ones in [24] while obtaining shorter runtimes.

Please note that the mentioned genetic algorithms for VLSI routing have two characteristics. First, they are sequential approaches despite the fact that PGA’s have been shown to lead generally to better results (e.g., in [9], [20], and [23]). Second, they consider only netlength and the number of vias as optimization goals but not electrical constraints such as crosstalk.

IV. DESCRIPTION OF GAP

A. Outline

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 designed a PGA inspired by concepts from the theory of *punctuated equilibria* [7], [12]. A genetic algorithm with punctuated equilibria is a PGA 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 Fig. 2).¹ Previous research has shown genetic algorithms with such punctuated equilibria to often have better performance when compared to sequential genetic approaches applied to the same domain [7], [9], [23].

The parallel structure of our algorithm for the case of nine processors is shown in Fig. 3. 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*.

¹This form of PGA has also come to be called “the island model.”

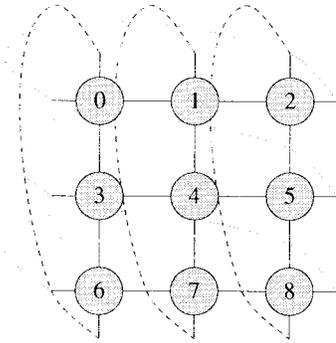


Fig. 3. Neighborhood structure of nine subpopulations. The subpopulations are arranged in a torus.

The genetic algorithm used by each processor and the main process that steers the parallel execution are presented in Fig. 4. 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 [24] 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. Please note that we exchange the individuals themselves, i.e., the migrants are removed from one subpopulation and added to another. Hence, the size of the subpopulations remains the same after migration and the assimilation of migrants is simply a fitness recalculation (*fitness_calculation*($\mathcal{P}_c \cup \text{migrants}$)).

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 (or has existed) constitutes our final routing solution.

The following section briefly describes some specific characteristics of the sequential genetic algorithm used by each processor to evolve its subpopulation.

B. Characteristics of GAP

1) *Genetic Representation*: The genetic encoding of the routing problem is based on the problem-specific representation scheme presented in [24]. Here the layout is coded in a 3-D lattice-like chromosome with the cells representing different coordinate points of the routing solution. The value of a cell indicates which net is routed at this coordinate point in the routing solution. A negative cell value indicates a fixed assignment (e.g., a pin) and zero indicates that the area is unused (see [24] for a more detailed description of our representation scheme).

We chose this 3-D encoding scheme with integer representation after numerous experiments with other genetic encoding schemes. For example, parts of the routing structure with near-optimal routing paths (termed as good “routing islands”)

are often scattered over the chromosome instead of being represented in one compact building block when binary or integer *string* representations are used. Our 3-D encoding scheme ensures that good “routing islands” in the routing structure are preserved as compact high-fitness building blocks in the chromosome. Consequently, these building blocks have a high probability of being transferred intact and recombined with other high-quality building blocks in offspring solutions. Furthermore, this encoding scheme enables a simple monitoring of the routing constraints directly in the chromosome.

2) *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 parents for crossover and the selection of individuals which are transferred into the next generation are based on these fitness values.

A raw fitness function $F'(p_i)$ is calculated for each individual $p_i \in \mathcal{P}_c$ according to

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

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 ;
- p_p length of adjacent net segments, summarized over all nets, of p_i (“crosstalk”).

It is important to note that the variable weight factors w_1, \dots, w_3 enable us to easily adjust routing quality objectives, 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 [19], in order to control the relative range of fitness in the subpopulation. Fitness scaling is performed *local* to the specific subpopulation with the scaled fitness $F_{\max} = 2 \times F'_{\text{avg}}$ (F'_{avg} = average raw fitness) [19].

3) *Selection*: Our selection strategy, which is responsible for choosing the parents for the crossover procedure, is stochastic sampling with replacement (“roulette-wheel selection”) [19]. That means any individual $p_i \in \mathcal{P}_c$ is selected with a probability given by

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

4) *Crossover*: During a crossover, two individuals are combined to create a descendant. Our crossover operator is a 1-point operator [19] that gives high-quality routing parts of the parents an increased probability of being transferred intact to their descendant. At the same time it guarantees enough randomness to explore new regions of the search space.

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

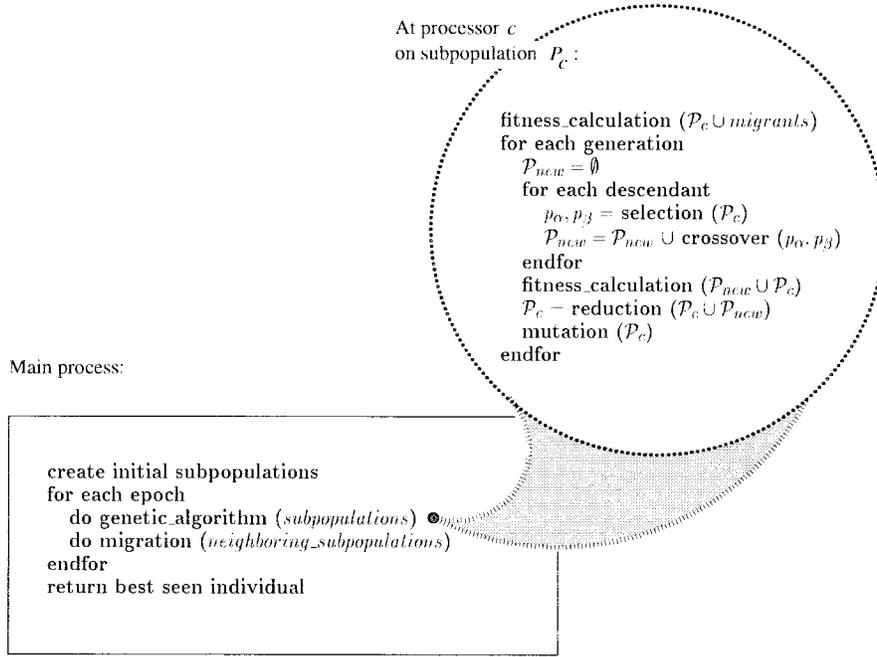


Fig. 4. Overview of our algorithm. See text for details.

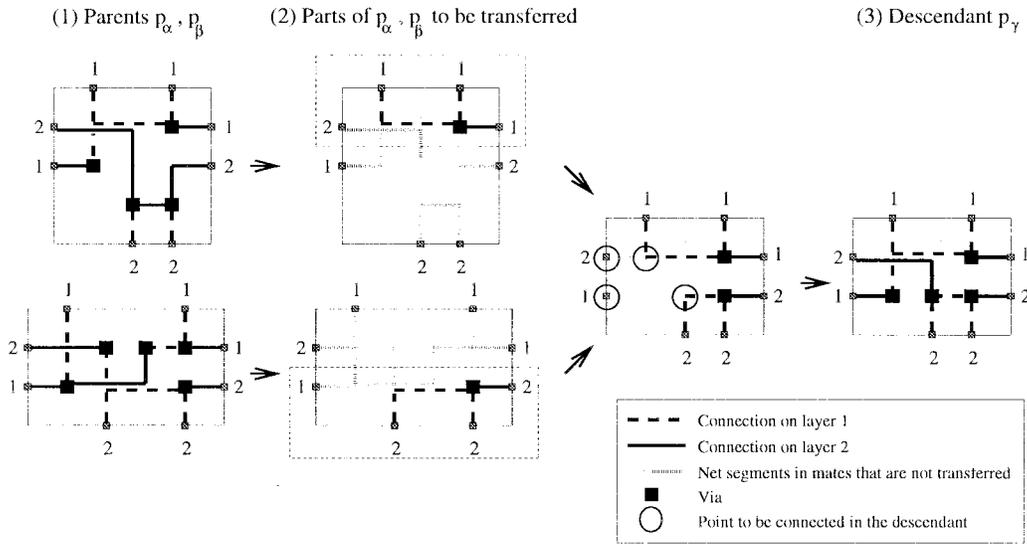


Fig. 5. Crossover of parents p_α and p_β to create a descendant p_γ .

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 [24].

A simple example of a crossover procedure is shown in Fig. 5.

5) *Reduction*: We use a deterministic reduction strategy which guarantees that high-quality individuals survive in as many generations as they are superior. 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. This strategy, which is the same as the $(\mu + \lambda)$ strategy often applied in evolution

strategies and evolutionary programming [3], is derived from the characteristic of our crossover operator that a high-quality parent does not necessarily produce a high-quality descendant, and in such a case, the parent should survive rather than the descendant.

6) *Mutation*: Mutation operators perform random modifications on an individual. The purpose is to overcome local optima and to exploit new regions of the search space.

Our mutation operator works as follows. A surrounding rectangle with random sizes (x_r, y_r) around a random center position (x, y, z) is defined. All interconnections inside this rectangle are deleted. The remaining net points on the edges of this rectangle are now connected again in a random order with our random routing strategy [24].

V. IMPLEMENTATION AND RESULTS

The algorithm has been implemented on a network of (up to) eight SPARC workstations (SunOS and Solaris systems). The parallel computation environment is provided by the Mentat system, an object-oriented parallel processing system [21], [27]. The program, written in C++ and Fortran, comprises approximately 10 000 lines of source code. The experimental results have been achieved with the machines running their normal daily loads in addition to our algorithm.

A. 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 I). 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 (1) are set to $w_1 = 1.0$ and $w_2 = 2.0$.

GAP was executed 120 times per benchmark with varying parameters (presented later). Table I presents the best-ever-seen results for all algorithms. The results from GAP 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 Fig. 6.

All executions of GAP were based on arbitrary initializations of the random number generator. Due to the stochastic nature of a genetic algorithm, the best-ever-seen results of GAP were not achieved in all executions. However, we should note that solutions equal to the best-ever-seen results were obtained in at least 50% of the individual GAP executions. The specific "success rates" for some benchmarks are: Burstein's Difficult Channel: 82%, Joo6_13 Channel 76%, Joo6_16 Channel 57%, Joo6_17 Switchbox 68%, and Pedagogical Switchbox 54%.

Please note that these "success rates" were achieved with different (including unfavorable) parameter settings (see Section VI) and thus, can be considered as lower bound in the individual variability of the results.

B. Crosstalk Reduction

By adjusting the value of the weight w_3 , our algorithm can optimize the interconnections regarding crosstalk. Hence, our router can construct solutions that 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

TABLE I

COMPARISON OF GAP WITH SOME WELL-KNOWN ALGORITHMS FOR BENCHMARK CHANNELS (UPPER HALF) AND SWITCHBOXES (LOWER HALF). THE RUNTIME OF GAP IS AVERAGED OVER THE RUNS THAT LED TO THE PRESENTED RESULTS. BEST RESULTS (ACCORDING TO NUMBER OF VIAS AND NETLENGTH) ARE IN BOLDFACE

Benchmark	Algorithm	Columns	Rows	Net-length	Vias	Time (sec)
Yoshimura-Kuh Channel	Yosh.-Kuh[33]	12	5	75	21	?
	WEAVER[22]	12	4	69	12	126
	Monreale[16]	12	4	74	11	?
	GAP	12	4	70	11	8
Burstein's Difficult Channel	PACKER[18]	12	4	82	10	87
	Monreale[16]	12	4	82	10	?
	GAP	12	4	82	8	16
Joo6_12	WEAVER[22]	12	4	79	14	134
	PACKER[18]	12	4	82	18	6
	Monreale[16]	12	4	84	13	?
	GAP	12	4	79	14	23
Joo6_13	WEAVER[22]	18	7	167	29	312
	Silk[26]	18	6	168	28	?
	PACKER[18]	18	6	167	25	710
	SAR[1]	18	6	166	25	70
	GAP	18	6	164	23	172
Joo6_16	WEAVER[22]	11	8	131	23	220
	WEAVER ^a [22]	11	7	121	21	216
	Monreale[16]	11	7	120	19	?
	GAP	11	6	115	15	207
Joo6_17	WEAVER[22]	11	9	166	19	325
	Silk[26]	11	9	166	18	?
	GAP	11	9	165	16	217
Pedagogical Switchbox	BEAVER ^b [8]	15	16	396	38	1
	PACKER[18]	15	16	406	45	91
	SAR[1]	15	16	393	31	146
	GAP	15	16	394	29	682
Burstein's Difficult Switchbox	WEAVER[22]	23	15	531	41	1508
	BEAVER ^b [8]	23	15	547	44	1
	PACKER[18]	23	15	546	45	56
	PARALLEL.EX[6]	23	15	539	59	25
	GAP	23	15	538	36	1831
Dense Switchbox	WEAVER ^a [22]	16	17	517	31	1087
	Silk[26]	16	17	516	29	?
	SAR[1]	16	17	519	31	150
	GAP	16	17	516	29	2380
Augmented Dense Switchbox	BEAVER ^b [8]	16	18	529	31	1
	PACKER[18]	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.

already during the routing process. The parameter $cross$ denotes the sum of $netcross(i)$ over all nets.

Table II 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 was considered appropriate. The results show that an increase of w_3 leads to significantly fewer parallel routed net segments ("cross") and fewer violations (denoted with "V") of the net-specific crosstalk requirement ($netcross(i) \leq maxcross(i)$). However, as can be seen in Table II, the minimization of crosstalk leads in general to an increase in both the netlength and the number of vias.

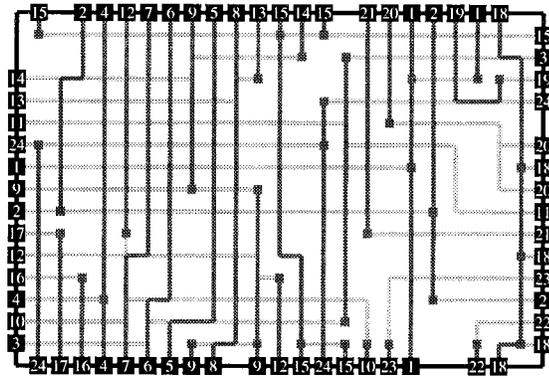


Fig. 6. Our routing solution of Burstein’s Difficult Switchbox. Black lines represent interconnections on one layer (metal 1), and grey lines denote interconnections on the other layer (metal 2).

TABLE II

REDUCTION OF CROSSTALK ACHIEVED BY INCREASING THE WEIGHT FACTOR FOR CROSSTALK, w_3 , FOR THREE CHANNELS AND ONE SWITCHBOX ($w_1 = 1.0$, $w_2 = 2.0$). *cross* REPRESENTS THE OVERALL LENGTH OF ALL ADJACENT, PARALLEL ROUTED NET SEGMENTS PER BENCHMARK. *V* DENOTES THE NUMBER OF NETS FOR WHICH THEIR UPPER BOUND OF PARALLEL ROUTED SEGMENTS IS EXCEEDED, I.E., THAT REPORT A VIOLATION OF THEIR INDIVIDUAL CROSSTALK CONSTRAINT. THE RESULTS PER BENCHMARK ARE AVERAGED OVER FIVE RUNS

Bench- mark	$w_3=0.01$			$w_3=1.0$			$w_3=4.0$		
	Nl./Vias ^a	<i>cross</i>	<i>V</i>	Nl./Vias ^a	<i>cross</i>	<i>V</i>	Nl./Vias ^a	<i>cross</i>	<i>V</i>
Burstein ^b	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 Burstein’s Difficult Channel.

Practical applications of this multi-objective optimization problem require the designer to specify the weights according to his/her optimization priorities. Practical approaches might include the generation of alternative solutions with emphasis on different optimization goals. From the output solution set, the designer then chooses a specific solution representing the preferred tradeoff.

VI. INVESTIGATION OF GAP PARAMETERS

As mentioned earlier, a PGA with punctuated equilibria alternates the maintenance of subpopulations isolated in different environments (to allow the development of individuals) with the introduction of individuals to new environments (to motivate further development of the individuals). We create different environments by defining the fitness of an individual relative to the quality of the other individuals in its current subpopulation (fitness scaling [19]). Exchanging individuals between subpopulations, i.e., migration, will alter the fitness values of the individuals within the subpopulation and introduce new competitors. Migration, of course, is based on various parameters, such as how often, how much, who, size and number of subpopulations, among others beyond the scope of this paper. We have performed several experiments to understand the specific effects of these parameters in order to guide further applications of PGA’s with punctuated equilibria.

TABLE III

THE FIVE BENCHMARKS CHOSEN FOR SUBSEQUENT EXPERIMENTS AND THEIR SPECIFIC PARAMETERS. “BEST-KNOWN SCORE” REPRESENTS THE BEST-EVER-SEEN RESULT OF EACH BENCHMARK (SEE TABLE I)

Bench- mark	Sub- population Size	Descendants per Generation per Subpopulation	Number of Generations	Best Known Score
	P_c	P_{new}		
Burstein ^a	50	20	100	74
Joo6_13	50	20	500	177
Joo6_16	50	20	500	123
Joo6_17	50	20	500	168
Ped. SB ^b	50	20	500	395

^a Burstein’s Difficult Channel.

^b Pedagogical Switchbox.

A. Measurement Conditions

In Table III, we show the five problem instances (three channel benchmarks and two switchbox benchmarks) chosen for our experiments. These benchmarks were selected because of their diversity and the availability of numerous published routing results. We compare the results of GAP in the following experiments with the best-known scores for these benchmarks (the rightmost column of Table III). These scores reflect the netlength and the number of vias as presented in Table I. All results presented in Tables IV–VIII are normalized as the percentage exceeding this best-known score, with the percentage averaged over five runs.

For comparison purposes, we also applied a sequential genetic algorithm (SGA) on the *total* population size (combined set of all subpopulations). This sequential genetic algorithm has been shown to produce the best results of any genetic algorithm for the considered benchmarks to date [24], [25].

To ensure a fair comparison, the following characteristics were considered. 1) Experimental results showed that the combined set of all subpopulations is not an “unfavorable setting” for the sequential genetic algorithm; on the contrary, the results are consistently better than the ones achieved with any smaller population size. 2) The sequential algorithm is operationally equivalent to the genetic algorithm that evolves each subpopulation in GAP. 3) In the experiments, the sequential genetic algorithm was set to perform the same number of recombinations per generation as GAP does over all subpopulations, namely, *number of subpopulations* \times *descendants per subpopulation*. 4) The sequential genetic algorithm and GAP were run the same total number of generations (see Table III). The number of generations is in accordance with the “optimal” values of the sequential genetic algorithm achieved in previous experiments [24], [25].

To demonstrate the importance of migration, we also report the results achieved with GAP when the subpopulations evolve in isolation (“0 Migrants”).

B. Number of Migrants and Epoch Lengths

We investigated the influence of different epoch lengths (number of generations between migration) for different num-

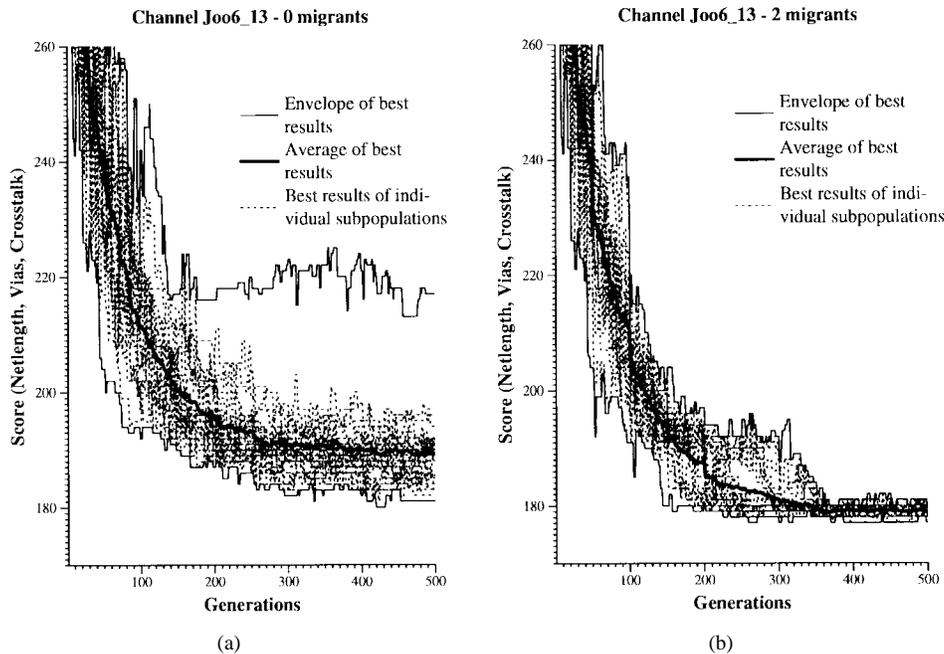


Fig. 7. Comparison of the convergence of the best individuals in the individual, parallel evolving subpopulations. Plotted are five runs with nine subpopulations each, i.e., 45 curves, in (a) isolation and (b) with two migrants. The “lower” curves and the smaller envelope of the right-hand plot indicate better results and less variation throughout the subpopulations.

bers of migrants (number of individuals sent to each of the four neighbors). The migrants were chosen randomly, with each migrant allowed to be sent only once. (As described earlier, the migrants are removed from one subpopulation and added to another.) Table IV shows that the sequential approach is outperformed by all parallel variations, including the version without any migration, when averaged over all considered benchmarks. Thus, the splitting of the total population size into parallel evolving subpopulations already increases the probability that at least one of these subpopulations will evolve toward a better result.²

Table IV also shows that a limited migration between the subpopulations further enhances the advantage of the PGA. Two migrants to each neighbor with an epoch length of 50 generations resulted in the best parameters when averaged over all problem instances. On the one hand, more migrants or too short epoch lengths are counterproductive to the idea of disjointly and parallel evolving subpopulations. They diminish the genetic diversity between the subpopulations by “pulling” them all into the same part of the search space, thereby approaching the behavior of a single-population genetic algorithm. On the other hand, insufficient migration (epoch length 75 generations) simulates the isolated parallel approach (zero migrants)—the genetic richness of the neighboring subpopulations does not have enough chance to spread out. Increasing the number of migrants can help in this case, although doing so does not achieve the good results of a “moderate” epoch length combined with a low number of migrants.

²Later, we will see that this conclusion requires that the number of individuals per subpopulation, depending on the problem size, is sufficiently large. Obviously, 50 individuals per subpopulation as in our case fulfills this requirement.

TABLE IV

OBTAINED CHANNEL AND SWITCHBOX RESULTS WITH DIFFERENT NUMBERS OF MIGRANTS AND EPOCH LENGTHS. FOR COMPARISON REASONS, THE RESULTS OF A SEQUENTIAL GENETIC ALGORITHM (SGA) ARE ALSO GIVEN. ALL RESULTS ARE AVERAGED OVER FIVE RUNS AND NORMALIZED AS PERCENT EXCEEDING THE BEST-KNOWN SCORE IN TABLE III. THUS, THE SMALLER THE VALUE, THE BETTER THE AVERAGE RESULT OF THE PARTICULAR CONFIGURATION

Bench- mark	SGA	Mig.	Epoch Length								
			25 Gen.			50 Gen.			75 Gen.		
			0	2	4	6	2	4	6	2	4
Burstein	4.32	1.08	2.16	2.16	3.24	1.08	1.08	1.08	n/a	n/a	n/a
Joo6.13	1.32	2.26	1.13	2.45	2.45	0.19	2.45	1.69	2.26	1.88	2.26
Joo6.16	5.96	3.52	4.34	5.69	4.12	4.06	4.88	4.88	4.06	4.18	3.71
Joo6.17	2.78	3.17	2.23	1.93	3.02	2.08	2.23	2.98	1.64	1.79	2.23
Ped. SB	8.19	6.60	5.51	7.08	8.05	7.58	8.46	8.52	7.01	6.68	6.68
Average	4.51	3.33	3.07	3.86	4.18	3.00	3.82	3.83	3.74	3.63	3.72
%SGA	100	74	68	86	93	66	85	85	83	81	82

Number of subpopulations : 9

Migrant selection strategy : random

Fig. 7 shows this behavior in the context of all subpopulations, that is, it presents the convergence behavior of the best individuals in each of the parallel evolving subpopulations. It indicates the importance of migration to avoid premature stagnation by implementing new genetic material into a stagnating subpopulation. Furthermore, the plot points out the “stabilizing” effect of migration as expressed in the limited variation among the best subpopulations gained in five independent runs [see Fig. 7(b)].

C. Variable Epoch Lengths

The ideas surrounding punctuated equilibria might be used to suggest: 1) a population in a constant environment will sta-

TABLE V

COMPARISON OF CHANNEL AND SWITCHBOX RESULTS BETWEEN A FIXED (50 GENERATIONS) AND A VARIABLE EPOCH LENGTH. THE VARIABLE EPOCH LENGTH WAS TERMINATED INDIVIDUALLY IN EACH SUBPOPULATION AFTER 25 GENERATIONS WITH NO IMPROVEMENT OF THE BEST INDIVIDUAL. ALL RESULTS ARE AVERAGED OVER FIVE RUNS AND NORMALIZED AS PERCENT EXCEEDING THE BEST KNOWN SCORE IN TABLE III

Bench- mark	SGA	Epoch Length					
				50 Gen.		Variable	
		Migr.	Migrants		Migrants		
			0	2	4	2	4
Burstein	4.32	1.08	1.08	1.08	1.08	1.08	
Joo6_13	1.32	2.26	0.19	2.45	2.07	1.88	
Joo6_16	5.96	3.52	4.06	4.88	2.71	2.44	
Joo6_17	2.78	3.17	2.08	2.23	2.18	2.58	
Pcd. SB	8.19	6.60	7.58	8.46	6.68	6.77	
Average	4.51	3.33	3.00	3.82	2.94	2.95	
%SGA	100	74	66	85	65	65	

Number of subpopulations : 9

Migrant selection strategy : random

TABLE VI

COMPARISON OF CHANNEL AND SWITCHBOX RESULTS WITH DIFFERENT MIGRANT SELECTION STRATEGIES (NO RESTRICTION ON MIGRANTS, MIGRANTS WITH FITNESS ABOVE MEDIAN FITNESS, BEST INDIVIDUALS AS MIGRANTS). ALL RESULTS ARE AVERAGED OVER FIVE RUNS AND NORMALIZED AS PERCENT EXCEEDING THE BEST-KNOWN SCORE IN TABLE III

Bench- mark	SGA	Migrant Selection Strategy							
				Random		Top 50%		Best	
		Migr.	Migrants		Migrants		Migrants		
			0	2	4	2	4	2	4
Burstein	4.32	1.08	1.08	1.08	1.08	1.08	1.08	1.08	
Joo6_13	1.32	2.26	0.19	2.45	1.88	3.58	1.32	2.82	
Joo6_16	5.96	3.52	4.06	4.88	2.44	3.79	4.88	3.79	
Joo6_17	2.78	3.17	2.08	2.23	1.98	2.18	3.17	1.59	
Pcd. SB	8.19	6.60	7.58	8.46	7.94	7.10	5.51	7.77	
Average	4.51	3.33	3.00	3.82	3.06	3.55	3.19	3.41	
%SGA	100	74	66	85	68	79	71	76	

Number of subpopulations : 9

Epoch length : 50 generations

bilize over time with little motivation for further development (“stasis” [12]), and 2) bursts of rapid evolution are often caused by small sets of individuals migrating to a new environment (“allopatric speciation” [12]). We, however, are interested in evolutionary systems for optimization, so we have used these ideas to design a model in which the evolutionary system has several subpopulations. These subpopulations are considered to be usefully evolving until they reach a stasis condition, at which point the model calls for migration in order to instigate further useful evolution. Most published computation models that are based on punctuated equilibria use a fixed number of generations between migration. Thus, they do not exactly duplicate the model that migration occurs only after a stage of equilibrium has been reached within a subpopulation.

We modified the algorithm to investigate the importance of this characteristic. Rather than having a fixed number of generations between migrations, we introduced a stop criterion that takes effect when stagnation in the convergence behavior within a subpopulation has been reached. We defined a suitable stop criterion to be 25 generations with no improvement in the best individual within a subpopulation.

Again, to ensure a fair comparison, we kept the overall number of generations the same as in all other experiments. This led to varying numbers of epochs between the parallel evolving subpopulations (due to different epoch lengths) and resulted in longer overall completion time.

Our results, presented in Table V, suggest that a slight improvement compared with a fixed epoch length can be achieved by this method. However, it is important to note that this comparison is made with a fixed epoch length that has been shown to be (nearly) optimal after numerous experiments (see Table IV). Thus, a variable epoch length based on the convergence behavior within the subpopulations can be useful when 1) a time effective usage of computational resources does not have highest priority, and/or 2) no prior experiences with an appropriate epoch length exist.

D. Different Migrant Selection Strategies

We investigated the influence of the quality of the migrants on the routing results. Three migrant selection strategies were compared: “Random” (migrants were chosen randomly among the entire subpopulation), “Top 50%” (migrants were chosen randomly among the individuals with a fitness above the median fitness of the subpopulation), and “Best” (only the best individuals of the subpopulation migrated). The migrants were sent in a random order to the four neighbors.

As Table VI indicates, we cannot find any improvement in the obtained results by using migrants with better quality. On the contrary, selecting better (or the best) individuals to migrate led to a faster convergence—the final results were not as good as those achieved with a less elitist selection strategy. According to our observations, this is due to the dominance of the migrants having their (locally good) genetic material reach all the subpopulations, thus leading the subpopulation searches into the same part of the search space concurrently.

E. Different Number of Subpopulations

To compare the influence of the number of subpopulations, we first kept the size of the subpopulations constant and increased the number of subpopulations to 16 and 25. Accordingly, we increased the population size and the number of recombinations of the sequential genetic algorithm to maintain a fair comparison. As expected, the sequential genetic algorithm improves its performance due to the larger number of solutions evaluated (see Table VII). The same is true for the parallel approach. The larger total population size and thus higher overall number of recombinations led to better results.

An interesting variation on this experiment is the division into different numbers of subpopulations while keeping the total population size constant. Thus, with 16 subpopulations, the subpopulation size is reduced to 28 individuals; with 25 subpopulations, the size decreases to only 18 individuals.

TABLE VII

COMPARISON OF CHANNEL AND SWITCHBOX RESULTS WITH DIFFERENT NUMBERS OF SUBPOPULATIONS. THE SIZE OF THE SUBPOPULATIONS IS KEPT CONSTANT AT 50 INDIVIDUALS. THE RESULTS OF THE SEQUENTIAL GENETIC ALGORITHM FOR THE RESULTING DIFFERENT OVERALL POPULATION SIZES ARE ALSO GIVEN. ALL RESULTS ARE AVERAGED OVER FIVE RUNS AND NORMALIZED AS PERCENT EXCEEDING THE BEST-KNOWN SCORE IN TABLE III

Bench- mark	Number of Subpopulations								
	9			16			25		
	SGA	Migrants		SGA	Migrants		SGA	Migrants	
		0	2		0	2		0	2
Burstein	4.32	1.08	1.08	2.16	1.08	0.00	0.00	0.00	0.00
Joo6_13	1.32	2.26	0.19	2.07	2.18	0.18	0.94	0.00	0.00
Joo6_16	5.96	3.52	4.06	3.79	1.87	1.72	2.44	1.87	1.01
Joo6_17	2.78	3.17	2.08	3.57	3.10	1.08	1.98	1.51	0.10
Ped. SB	8.19	6.60	7.58	8.27	6.00	5.21	7.69	5.98	5.18
Average	4.51	3.33	3.00	3.97	2.85	1.64	2.61	1.87	1.26
%SGA	100	74	66	100	71	41	100	85	48

Migrant selection strategy : random
Epoch length : 50 generations

The results presented in Table VIII show the relationship between the problem size and the minimal number of individuals per subpopulation necessary to prevent premature convergence. Relatively small problem sizes (Burstein, Joo6_13) benefit from a further splitting into more (but smaller) subpopulations. The advantage of more varied evolving subpopulations outperforms the drawback of a smaller subpopulation size up to a certain level. This level seems to be reached shortly below 50 individuals per subpopulation for the switchbox Joo6_17 and the Pedagogical Switchbox—their results suffer significantly from the loss of genetic diversity due to smaller subpopulations.

VII. CONCLUSIONS

We presented a PGA for two detailed routing problems in VLSI circuits. The approach includes a new measure of the netlength to better reflect the electrical delay in submicron regimes. Importantly, the 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 submicron VLSI design.

The results also showed that, when applied to our routing problem, the PGA—based on concepts of punctuated equilibria—consistently performs better than a sequential genetic algorithm.

In investigating the parameters of the algorithm, the following conclusions have been reached.

- A small number of migrants (1–3 per neighbor) combined with a “moderate” epoch length (approximately 5–10% of the total number of generations) leads heuristically to the best results.
- Variable epoch lengths determined via equilibrium measures within subpopulations achieve overall results that

TABLE VIII

COMPARISON OF CHANNEL AND SWITCHBOX RESULTS WITH DIFFERENT NUMBERS OF SUBPOPULATIONS. THE SIZE OF THE TOTAL POPULATION IS KEPT CONSTANT AT 450 INDIVIDUALS. ALL RESULTS ARE AVERAGED OVER FIVE RUNS AND NORMALIZED AS PERCENT EXCEEDING THE BEST-KNOWN SCORE IN TABLE III

Bench- mark	SGA	Number of Subpopulations					
		9		16		25	
		Migrants		Migrants		Migrants	
		0	2	0	2	0	2
Burstein	4.32	1.08	1.08	1.08	0.00	0.00	0.00
Joo6_13	1.32	2.26	0.19	2.12	0.00	1.80	0.00
Joo6_16	5.96	3.52	4.06	3.52	3.81	7.10	6.00
Joo6_17	2.78	3.17	2.08	4.18	3.12	14.0	9.81
Ped. SB	8.19	6.60	7.58	13.1	11.8	42.0	37.2
Average	4.51	3.33	3.00	4.80	3.75	13.0	10.6
%SGA	100	74	66	106	83	288	235

Migrant selection strategy : random
Epoch length : 50 generations

are slightly better than those obtained with (near-) optimized fixed epoch lengths. Practical applications of this “strict punctuated equilibria method” require the user to weigh the advantage of this “self-adjustment” against its main drawback, decreased time efficiency.

- Quality constraints on the migrants (e.g., to be above median fitness) do not improve the overall behavior of the algorithm, on the contrary, quality requirements on the selection of migrants led to premature stagnation.
- Given a sufficient number of individuals per subpopulation, a larger number of parallel evolving subpopulations will produce better routing results (for a fixed number of total evaluations). The size of the problem and the minimal subpopulation size have a direct correlation that must be taken into account when dividing a population into subpopulations.

Our algorithm can be easily implemented on any distributed network of conventional workstations. We believe this approach promises to be a useful tool in VLSI design.

ACKNOWLEDGMENT

Special thanks to W. N. Martin and J. P. Cohoon (University of Virginia) for fruitful discussions and their support. The comments and suggestions of the reviewers were greatly appreciated. Finally, the author wishes to thank the Computer Science Department, University of Virginia, for providing the facilities crucial to this work.

REFERENCES

- [1] A. Acan and Z. Ünver, “Switchbox routing by simulated annealing: SAR,” in *Proc. IEEE Int. Symp. Circuits and Syst.*, 1992, vol. 4, pp. 1985–1988.
- [2] P. Adamidis, “Review of parallel genetic algorithms,” Dept. Elec. Comp. Eng., Aristotle Univ. Thessaloniki, Greece, Tech. Rep., 1994.
- [3] T. Bäck and H.-P. Schwefel, “An overview of evolutionary algorithms for parameter optimization,” *Evolutionary Computation*, vol. 1, no. 1, pp. 1–23, 1993.
- [4] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*. Reading, MA: Addison-Wesley, 1990.

- [5] H. H. Chen and C. K. Wong, "Wiring and crosstalk avoidance in multi-chip module design," in *Proc. Custom Integrated Circuits Conf.*, 1992, pp. 28.6.1–28.6.4.
- [6] T. W. Cho, S. S. Pyo, and J. R. Heath, "PARALLEX: A parallel approach to switchbox routing," *IEEE Trans. Computer-Aided Design*, vol. 13, no. 6, pp. 684–693, 1994.
- [7] J. P. Cohoon, S. U. Hedge, W. N. Martin, and D. S. Richards, "Punctuated equilibria: A parallel genetic algorithm," in *Proc. 2nd Int. Conf. Genetic Algorithms*, 1987, pp. 148–154.
- [8] 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, pp. 684–697, 1988.
- [9] J. P. Cohoon, W. N. Martin, and D. S. Richards, "Genetic algorithms and punctuated equilibria in VLSI," in *Parallel Problem Solving from Nature*, H. P. Schwefel and R. Männer, Eds. Berlin, Germany: Springer-Verlag, Lecture Notes in Computer Science, 1991, vol. 496, pp. 134–144.
- [10] J. Cong, K. S. Leung, and D. Zhou, "Performance-driven interconnect design based on distributed RC delay model," in *Proc. IEEE Design Automation Conf.*, 1993, pp. 606–611.
- [11] W. M. Dai, R. Kong, J. Jue, and M. Sato, "Rubber band routing and dynamic data representation," in *Proc. Int. Conf. Computer-Aided Design*, 1990, pp. 52–55.
- [12] N. Eldredge and S. J. Gould, "Punctuated equilibria: An alternative to phyletic gradualism," in *Models of Paleobiology*, T. J. M. Schopf, Ed. San Francisco, CA: Freeman, Cooper, 1972, pp. 82–115.
- [13] M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, no. 1, pp. 1–16, 1995.
- [14] T. Gao and C. L. Liu, "Minimum crosstalk channel routing," in *Proc. Int. Conf. Computer-Aided Design*, 1993, pp. 692–696.
- [15] ———, "Minimum crosstalk switchbox routing," in *Proc. Int. Conf. Computer-Aided Design*, 1994, pp. 610–615.
- [16] M. Geraci, P. Orlando, F. Sorbello, and G. Vasallo, "A genetic algorithm for the routing of VLSI circuits," in *Proc. Euro Asic '91*, 1991, pp. 218–223.
- [17] N. Göckel, G. Pudelko, R. Drechsler, and B. Becker, "A hybrid genetic algorithm for the channel routing problem," in *Proc. IEEE Int. Symp. Circuits and Systems*, 1996, pp. 675–678.
- [18] S. H. Gerez and O. E. Herrmann, "Switchbox routing by stepwise reshaping," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 1350–1361, Dec. 1989.
- [19] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
- [20] M. Gorges-Schleuter, "Explicit parallelism of genetic algorithms through population structures," in *Parallel Problem Solving from Nature*, H. P. Schwefel and R. Männer, Eds. Berlin, Germany: Springer-Verlag, Lecture Notes in Computer Science, 1991, vol. 496, pp. 150–159.
- [21] A. S. Grimshaw, "Easy-to-use object-oriented parallel processing with mentat," *IEEE Computer*, vol. 26, no. 5, pp. 39–51, 1993.
- [22] R. Joobbani, *An Artificial Intelligence Approach to VLSI Routing*. Boston, MA: Kluwer, 1986.
- [23] B. Kröger, "Parallel genetic algorithms for solving the two-dimensional bin packing problem," Ph.D. thesis, Dept. Math. Comp. Sci., Univ. Osnabrück, Germany, 1993 (in German).
- [24] J. Lienig and K. Thulasiraman, "A genetic algorithm for channel routing in VLSI circuits," *Evolutionary Computation*, vol. 1, no. 4, pp. 293–311, 1994.
- [25] ———, "GASBOR: A genetic algorithm for switchbox routing in integrated circuits," *J. Circ., Syst., Comput.*, vol. 6, no. 4, 1996, pp. 359–373.
- [26] 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.
- [27] Mentat Homepage: "<http://www.cs.virginia.edu/~mentat/>".
- [28] A. Onozawa, K. Chaudhary, and E. S. Kuh, "Performance driven spacing algorithms using attractive and repulsive constraints for sub-micron LSI's," *IEEE Trans. Computer-Aided Design*, vol. 14, no. 6, pp. 707–719, 1995.
- [29] B. B. Prahlada Rao, L. M. Patnaik, and R. C. Hansdah, "An extended evolutionary programming algorithm for VLSI channel routing," in *Evolutionary Programming IV: Proc. 4th Annu. Conf. Evolutionary Programming*, J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, Eds. Cambridge, MA: MIT Press, 1995.
- [30] ———, "A genetic algorithm for channel routing using inter-cluster mutation," in *Proc. 1st IEEE Int. Conf. Evolutionary Computation*, 1994, pp. 97–103.
- [31] B. B. Prahlada Rao and R. C. Hansdah, "Extended distributed genetic algorithm for channel routing," in *Proc. IEEE Symp. Parallel and Distributed Proc.*, 1993, pp. 726–733.
- [32] A. T. Rahmani and N. Ono, "A genetic algorithm for channel routing problem," in *Proc. 5th Int. Conf. Genetic Algorithms*, 1993, pp. 494–498.
- [33] T. Yoshimura and E. S. Kuh, "Efficient algorithms for channel routing," *IEEE Trans. Computer-Aided Design*, vol. CAD-1, no. 1, pp. 25–35, 1982.



Jens Lienig (M'97) received the M.S. and Ph.D. degrees in electrical engineering from the University of Technology, Dresden, Germany, in 1988 and 1991, respectively.

From 1988 to 1991, he was a Research Assistant, Department of Electronics Engineering, University of Technology, Dresden, where he was responsible for the development of routing algorithms for an MCM design system. From 1991 to 1994, he was a Postdoctoral Fellow, Department of Electrical and Computer Engineering, Concordia University, Montréal, P.Q., Canada. From 1994 to 1996, he was a Visiting Assistant Professor with the Department of Computer Science, University of Virginia, Charlottesville. While in Montréal and Charlottesville, he worked on the application of genetic algorithms to VLSI physical design. Since 1996, he has been with Tanner Research, Inc., Pasadena, CA. His research interests are in the physical design of VLSI circuits, MCM's, and PCB's, with a special emphasis on the development of routing tools.