Performance Driven Standard-cell Placement Using the Genetic Algorithm

Habib Youssef Sadiq M. Sait Khaled Nassar Muhammad S. T. Benten.

Department of Computer Engineering
King Fahd University of Petroleum and Minerals
Dhahran-31261, Saudi Arabia
e-mail: facy009@saupm00.bitnet

Abstract

Current placement systems attempt to optimize several objectives, namely area, connection length, and timing performance. In this paper we present a timing-driven placer for standard-cell IC design. The placement algorithm follows the genetic paradigm. Besides optimizing for area and wire length, the placer minimizes the propagation delays on a predicted set of critical paths. The paths are enumerated using a new approach based on the notion of α-criticality. Experiments with test circuits demonstrate delay performance improvement by up to 20%.

1 Introduction

Placement consists of assigning cells of a given circuit to physical locations on a 2-dimensional layout surface. In this work we consider standard-cell placement. Contemporary placement tools are multi-objective, where timing performance is one of the main quality measures used when looking for a solution.

The speed of a circuit is determined by the time it takes for a signal to travel on its *longest* path. A signal traveling on any path π is constrained to reach the path end point no later than its latest required arrival time ($LRAT_\pi$). A design is free from long path timing problems if, for every path π,

$$T_\pi \leq LRAT_\pi \tag{1}$$

where T_π is the overall delay on path π. The problem of performance driven placement consists of finding suitable locations of cells so as to minimize the total wirelength and area, while satisfying Equation 1 for each path π.

In this work a linear cell delay model is used. The delay TD of a cell is computed as follows,

$$TD = BD + LD + ID \tag{2}$$

where, LD is the load delay due to loading pins of the net driven by the cell, and ID is the interconnect delay on the net. Expressions for these quantities are given below.

$$LD = LF \times C_{in} \tag{3}$$

$$ID = LF \times C_{net} + R_{net} \times (C_{net} + C_{in}) \tag{4}$$

where, LF is the load factor, C_{in} is the total input capacitance of the loading cells, R_{net} is the total interconnect resistance of the net, and C_{net} is the total interconnect capacitance of the net (fringe plus surface).

Numerous attempts have been reported which tried to make the physical design sensitive to the timing requirements. In [10], path timing constraints are transformed into bounds on interconnect delays, which are used by the following placement procedure. A constructive placement method that uses a cost function that captures the timing behavior was presented in [9]. In [4, 8], performance driven placement is solved using mathematical programming.

Iterative improvement techniques have also been employed. In [2], the authors employ simulated annealing to improve both the wirelength and performance. Other iterative nondeterministic techniques that have been applied to the placement are genetic algorithm (GA) [1, 7] and simulated evolution [5]. But for both, the placement objective was the minimization of wirelength, timing performance was not an issue.

In this work we describe a timing driven genetic algorithm for placement. Initially, a number of placement configurations are constructively produced. Then, the genetic algorithm is used to iteratively search for a new solution that combines the good characteristics of the initial configurations. The overall objective is two-fold; (1) satisfy path timing constraints and (2) minimize overall wiring length (area).

1.1 Timing Prediction for Placement

The timing data passed from the timing analyzer to TDGAP consists of a predicted set of the most critical paths. This set is predicted as follows. From past layouts of circuits with similar complexity, the average and standard deviation of net lengths are estimated for each type of net (2 pin-, 3 pin-,..., k pin-nets). These are converted to capacitances for the particular technology of the design at hand.

Let T_π and S_π be the overall delay (including the net delay estimations) and standard deviation along path π. Let $T_{\max}$ be the estimated delay of the longest path in the design, that is,

$$T_{\max} = \max_{\pi}\{T_\pi\} \tag{5}$$

A path π is called *α-critical* if and only if,

$$T_\pi + \alpha \times S_\pi \geq T_{\max} \tag{6}$$

The parameter α acts as a confidence level. This prediction approach was effective in predicting all of the critical paths in the design we experimented with.[1]

2 TDGAP: Timing Driven Genetic Algorithm for Placement

Genetic algorithm is a search technique which emulates the natural process of evolution as a means of progressing toward the optimum.

Starting with an initial population of placement configurations, TDGAP applies its operators and functions to improve the overall fitness of its individuals from a generation to the next. The search continues until all the timing constraints are met or a large number of generations were bred. Timing constraints consist of delay bounds on the critical paths of the circuit.

2.1 Solution Representation

Each individual (solution) in the population is encoded as a set of rows. Each row contains modules (genes) that are represented as a set of three integers indicating the cell serial number, the row number, and the displacement from the left edge of the layout.

2.2 Initial Population Constructors

Initial solution construction is very critical to GA. Five initial population constructors were investigated. They are: (1) constructor IPC_1 selects modules at random and places them in rows; (2) constructor IPC_2 attempts to clusters cells affecting the same path; (3) constructor IPC_3 is similar to IPC_2 except that IPC_2 places cells left to right starting from row 0, while IPC_3 places cells starting from the middle row and proceeding outward; (4) constructor IPC_4 combines individuals from IPC_1 and IPC_3; and (5) constructor IPC_5 is similar to constructor IPC_4 with the difference that it includes in its initial population a placement configuration obtained using the mincut partitioning algorithm (a solution with good wiring characteristics).

2.3 Choice Function

The choice function adopted is based on the *stochastic remainder without replacement* scheme [3]. This scheme works as follows. Let $exp\_count(\mathcal{P}(i))$ be the value of the expected count of an individual $\mathcal{P}(i)$.

$$exp\_count(\mathcal{P}(i)) = \frac{cost(\mathcal{P}(i))}{\overline{cost}} \tag{7}$$

where,

$$cost\mathcal{P}(i) = \text{ cost value of individual } \mathcal{P}(i) \tag{8}$$

[1]Typical values of α are: $\alpha \times S_\pi \leq 5ns$.

and,

$$\overline{cost} = \frac{1}{N_p} \times \sum_{i=1}^{N_p} cost(\mathcal{P}(i)) \tag{9}$$

For each individual $\mathcal{P}(i)$, $\lfloor exp\_count(\mathcal{P}(i)\rfloor$ instances of $\mathcal{P}(i)$ are included in a list L. The fractional part $f_i = exp\_count(\mathcal{P}(i) - \lfloor exp\_count(\mathcal{P}(i)\rfloor$ is interpreted as a probability, that is, with probability f_i one more instance of $\mathcal{P}(i)$ is included in the list L. This operation is repeated until all individuals are processed. Following this step, parents are randomly selected from the list L for crossover (two at a time).

2.4 Crossover $\mathcal{X}$

Crossover is the most significant genetic operator and has the most effect on the convergence rate and the quality of solution. Two types of crossover operators $\mathcal{X}_1$ and $\mathcal{X}_2$ are considered in TDGAP. Both operators are aimed at improving the timing aspects of the reported *α-critical* paths. They try to pass the information about some of the satisfied paths from one generation to another. Operator $\mathcal{X}_1$ does this by maintaining the same locations of the cells affecting these satisfied paths. Operator $\mathcal{X}_2$, however, does it by keeping the cells affecting these satisfied paths within a certain window.

Let $\mathcal{P}(s)$ and $\mathcal{P}(t)$ be the passing and target parents respectively, and CP be the set of the *α-critical* paths of the circuit. Operator $\mathcal{X}_1$ starts by making an identical copy ($\mathcal{P}(o)$) of $\mathcal{P}(t)$. Then a critical path cp is selected from CP such that cp has better timing in the source parent then in the target. $\mathcal{X}_1$ reconfigures offspring $\mathcal{P}(o)$ such that the cells of cp occupy the same locations as in the passing parent. Collisions are resolved by interchanging the locations of the colliding modules.

On the other hand, crossover operator $\mathcal{X}_2$ identifies the size and location of the smallest bounding window ω_s that encloses the cells of cp in $\mathcal{P}(s)$. A window ω_t is also determined in parent $\mathcal{P}(t)$ with the same dimensions and location as ω_s in $\mathcal{P}(s)$. Comparing the contents of these two windows, three sets are defined:

σ = cells in cp but not in ω_t;
ρ = cells $\in \omega_s$ but are neither in ω_t nor in cp; and
η = cells $\in (\omega_s \cap \omega_t)$.

Then operator $\mathcal{X}_2$ reconfigures $\mathcal{P}(o)$ as follows. It first defines a window ω_o in $\mathcal{P}(o)$ of the same size and location as ω_s. After that, it scans the contents of ω_o cell by cell and one row at a time. Cells that are in η are not affected by $\mathcal{X}_2$. For each scanned cell e_i, operator $\mathcal{X}_2$ reconfigures the offspring according to the algorithm given in Figure 1.

2.5 Selection (§) of the Next Generation

The selector function is used to maintain the population size fixed. It determines which individuals to use as part of the next generation. We experimented with four selector functions. Let $\mathcal{P}$ be the current population and $\mathcal{J} = \mathcal{P} \cup Offsprings$. Selector $\S_1$ selects from $\mathcal{J}$ the best scoring individual and $N_p - 1$

other individuals at random, where $N_p = \mid \mathcal{P} \mid$. Selector $\S_2$ selects the best 10% of N_p and the rest are selected at random. Selector $\S_3$ selects all N_p individuals from $\mathcal{J}$ at random. Selector $\S_4$ selects individuals on a competitive basis with each individual $\mathcal{P}(j)$ having a probability $Prob(j)$ to be selected, where

$$Prob(j) = \frac{score(\mathcal{P}(j))}{\sum_{i=1}^{q} score(\mathcal{P}(i))} \qquad (10)$$

where, $q = \mid \mathcal{J} \mid$, and $score(\mathcal{P}(i))$ is the fitness of individual $\mathcal{P}(i)$. With this selector, the algorithm has a higher probability than with other selectors to be trapped into local minima. This might happen because individuals with low fitness values are quickly discarded at the early generations.

2.6 Mutation μ

Two mutation operators μ_1 and μ_2 were investigated. Operator μ_1 is targeted toward improving the timing of the placement, while operator μ_2 is targeted at improving the wirelength of the placement. Except for the best individual, any individual in the newly selected generation may be a candidate for mutation.

Mutation operator μ_1 works as follows. For each individual $\mathcal{P}(i)$ that is selected for mutation, first a critical path *cp* with long path problem in $\mathcal{P}(i)$ is selected. Next, a module e_s which is affecting the performance of *cp* is randomly selected. The module e_s is pairwise interchanged with the module at the center-of-mass of Net_s.

Mutation operator μ_2 operates in a similar manner. It starts by selecting at random a two-pin net, where neither of the two pins is an I/O pad. Then, one of the two modules is chosen to be swapped with a module at the location of the center of the selected net. The requirement that the selected net be a two-pin net is motivated by experimental observations. Analysis of several layouts revealed that most of the two-pin nets that are on critical paths have their modules separated by large distances. This wide separation has two undesirable effects: (1) it increases the number of feedthroughs and (2) it increases the total wirelength.

2.7 Score Function

The score function is a weighted sum of three terms that are directed toward the improvement of the circuit performance and total wirelength. Fitness is increasing with increasing score values. The score of a given individual $\mathcal{P}(i)$ is computed as follows:

$$Score(\mathcal{P}(i)) = w_1 \times S_i \; + \; w_2 \times W_i \; + \; w_3 \times R_i \qquad (11)$$

where w_1, w_2, and w_3 are different weights assigned for each term. S_i and W_i are measures of the timing and wirelength aspects of individual $\mathcal{P}(i)$, while R_i is a relaxation factor. It is a measure of the amount by which satisfied paths can be made longer and remain problem free. This is to give the wirelength metric a chance to improve.

2.8 Experiments and Results

Extensive experimentation was conducted with all suggested operators. The operators that performed best with respect to wirelength and timing are indicated[2] in Table 1. These operators showed a superior performance as well as faster convergence.

```
ALGORITHM(Reconfigure)
Stop=0;
Repeat
  If $e_i \in \eta$ Then
    skip this cell and go to the next one;
  ElseIf $\sigma \notin \emptyset$ Then
    Begin
      pick a module $e_j$ from $\sigma$ and swap $e_i$ and $e_j$;
      remove module $e_j$ from $\sigma$
    End
  ElseIf $\rho \notin \emptyset$ Then
    Begin
      pick a module $e_j$ from $\rho$ and swap $e_i$ and $e_j$;
      remove module $e_j$ from $\rho$;
    End
  Else
    Begin
      skip this cell (all other cells will stay in their locations);
      Stop=1
    End
Until(all cells $\in \omega_o$ are scanned or Stop=1)
```

Figure 1: Algorithm used by $\mathcal{X}_2$ to reconfigure an offspring ξ_o.

We run TDGAP on four circuits (Table 2). Ck1 is a sample AHPL model that performs part of the stop and wait protocol; CRC16 is a 16-bit Cyclic Redundancy Checker; Highway is a traffic light controller; and Fract is a fractional multiplier.

A summary of the initial and final values of the best solution for all test cases with respect to timing and area metrics is given in Table 3. The slack values given are obtained after the placement phase, but before routing. To obtain these values we have developed and used a timing evaluator that checks for timing violations based on the reported α-critical paths. The placements obtained by TDGAP were evaluated with respect to timing as well as overall wirelength and compared with placements produced by OASIS[6]. Timing performance improvements of up to 20% were obtained. The area values given are obtained after completing the routing phase and generating the layout. The improvement achieved by TDGAP with respect to timing aspects has resulted in a slight increase in the overall area (between 1% and 9%). All results were obtained after 100,000 generations (run time between 5 and 10 hours).

From experiments we identified two parameters that, if allowed to adapt, would lead to superior results: (1) the crossover probability; and (2) the mutation probability. From experimentation, starting with a high crossover probability of 90% then gradually re-

[2]discussions and graphs are omitted due to lack of space.

Constructor function	IPC_5
Crossover operator	$\mathcal{X}_1$
Mutation operator	$\mu_1 \cup \mu_2$
Selector function	$\S_1$
Population size	24
Crossover probability	0.5 - 0.7
Mutation probability	0.1

Table 1: A summary of the genetic parameters that were found to perform better than others with TDGAP.

Circuit Name	# of cells	Clock period	# of critical paths	# of rows in final layout
Ck1	209	21 ns	200	8
CRC16	209	18 ns	330	9
Highway	56	20 ns	14	4
Fract	149	38 ns	368	6

Table 2: Characteristics of the circuits used.

ducing it to 70% seems to be a better choice then keeping it constant. A similar strategy can be also applied for the mutation probability, that is, starting with relatively large value of 20% and then gradually reducing it until it reaches 10%. These observations require further investigations.

TDGAP is implemented in the C language. Experiments were performed on a 64-bit DEC Alpha workstation that is running OSF/1 operating system at the speed of 100 MIPS.

3 Conclusions

In this paper we presented a timing-driven placement program. The placement procedure follows the genetic algorithm. The program uses path timing data from a timing analyzer. The timing analyzer uses a new criterion (*α-criticality*) to predict the critical paths prior to placement. Extensive experiments were conducted to tune the parameters of the program and evaluate the extent of timing improvement. Results from benchmark circuits show that timing improvement of upto 20% can be achieved by our placement system without any change to the logic of the circuit. The sizable timing improvement is accompanied with a very slight increase in the area of the circuit.

Acknowledgments

Authors acknowledge King Fahd University of Petroleum and Minerals for all support. This research is supported by KFUPM Project # COE/VLSIDESIGN/162

Circuit Name	Area			Slack		
	TDGAP	Increase	OASIS	TDGAP	Impr	OASIS
Ck1	928 ×1016	9.1%	923 × 936	+0.52	8.1%	-1.14
CRC16	1131 × 968	5.5%	1072 × 968	+0.61	17.7%	-2.47
Highway	478 × 496	6.2%	465 × 480	+0.51	11.4%	-1.72
Fract	798 × 824	5.9%	768 × 808	+0.31	6.5%	-2.14

Table 3: Area and Slack performance comparison between TDGAP and OASIS.

References

[1] J. P. Cohoon and W. D. Paris. Genetic placement. *IEEE Transactions on Computer Aided Design*, CAD-6:956–964, November 1987.

[2] W. Donath et al. Timing-driven placement using complete path delays. *Proceedings of 27th Design Automation Conference*, pages 84–89, June 1990.

[3] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Company, INC., 1989.

[4] M. A. B. Jackson and E. S. Kuh. Performance-driven placement of cell-based IC's. *Proceedings of 26th Design Automation Conference*, pages 370–375, June 1989.

[5] R. M. Kling and P. Banerjee. Empirical and theoretical studies of the simulated evolution method applied to standard cell placement. *IEEE Transactions on Computer Aided Design*, CAD-10:1303 – 1315, October 1991.

[6] MCNC Group. *OASIS 2.0 Reference Manual*, 1990.

[7] K. Shahookar and P. Mazumder. A genetic approach to standard cell placement using meta-genetic parameter optimization. *IEEE Transactions on Computer Aided Design*, 9(5):500–511, May 1990.

[8] Arvind Srinivasan, Kamal Chaudhary, and Ernest S. Kuh. Ritual: a performance-driven placement algorithm. *IEEE Transactions on Circuits and Systems*, pages 825–840, November 1992.

[9] S. Suthanthavibul, E. Shragowitz, and Rung-Bin Lin. An adaptive timing-driven placement for high performance VLSI's. *IEEE Transactions on Computer Aided Design*, 12(10):1488–1498, October 1993.

[10] H. Youssef and E. Shragowitz. Timing constraints on signal propagation in VLSI. *Proceedings of International Conference on Computer-Aided Design*, pages 24–27, 1990.