Relaxed Simulated Tempering for VLSI Floorplan Designs*

Jason Cong, Tianming Kong, Dongmin Xu
Computer Science Department
University of California, Los Angeles, CA 90095
{cong, kongtm, dongmin}@cs.ucla.edu

Faming Liang, Jun S. Liu, Wing Hung Wong
Department of Statistics
University of California, Los Angeles, CA 90095
Department of Statistics, Stanford University, Stanford, CA 94305
{fmliang, junliu, whwong}@stat.ucla.edu

Abstract

In the past two decades, the *simulated annealing* technique has been considered as a powerful approach to handle many NP-hard optimization problems in VLSI designs. Recently, a new Monte Carlo and optimization technique, named *simulated tempering*, was invented and has been successfully applied to many scientific problems, from random field Ising modeling to the traveling salesman problem. It is designed to overcome the drawback in *simulated annealing* when the problem has a rough energy landscape with many local minima separated by high energy barriers. In this paper, we have successfully applied a version of relaxed *simulated tempering* to slicing floorplan design with consideration of both area and wirelength optimization. Good experimental results were obtained.

1 Introduction

Since its introduction in early 1980s, *simulated annealing* has been one of the most popular optimization algorithms used in the VLSI CAD field in the past two decades. It has been applied to almost every step of the IC design process, including (but not limited to) scheduling, resource allocation, logic synthesis, partitioning, floorplanning, and placement. However, the efficiency of *simulated annealing* depends much on the energy landscape. If the global minimum solution has a small basin of attraction and is well separated by many local minima with high energy barriers, *simulated annealing* tends to be trapped in a local minimum solution and is not able to explore the solution space efficiently. The reason is that the temperature cooling schedule used in practice in the simulated annealing process is usually faster than that required by the theory (exponential cooling is used in practice as opposed to logarithmic cooling required by the theory). The solution quality of the simulated annealing based optimization tends to get worse as the design complexity increases, as faster cooling schedules have to be used to assure a reasonable runtime.

In this paper, we use a new optimization technique, named *simulated tempering* [MP92, GT95], to solve VLSI floorplan problem. *Simulated tempering* was invented to overcome the drawback of *simulated annealing* by taking temperature as an additional random variable during the optimization. It provides a chance for the system to correct the mistakes possibly made by *simulated annealing* due to its fast-freezing annealing process. The system can then be warmed up from time to time, which makes it possible for the process to escape from a local minimum. As a general Monte Carlo and optimization method, *simulated tempering* has been applied successfully to many scientific areas, such as random field Ising model [MP92], spin glass model [KR94] and ancestral gene inference [GT95]. An extension of *simulated tempering*, called *dynamic weighting* [WL97], in which a dynamic weight is employed to help the system to jump between adjacent temperature levels, has been successfully applied to the traveling salsman problem and neural network training.

In this paper, we have successfully applied a version of relaxed *simulated tempering* to both area and wirelength optimization for slicing floorplan design based on Polish expression representation [WL89]. There are several reasons that we choose floorplan design as the first application to demonstrate the effectiveness of *simulated tempering* in VLSI CAD optimization. First, floorplan design is an important step in VLSI physical design. It plays an increasingly important role in linking design planning, RTL synthesis, and physical design. The floorplan design quality can affect the final design quality significantly. Second, much effort has been made by the VLSI design community to use the *simulated annealing* approach to solve the floorplan design problem. By applying the *simulated tempering* algorithm to floorplan design, we can make an objective measurement of its effectiveness in solving VLSI design problems as compared to *simulated annealing*.

In order to achieve smooth temperature transition, during our implementation of *simulated tempering* algorithm, we found that the temperature levels have to be closely spaced so as to make the energy distribution of two adjacent temperature levels sufficiently overlap. As a result, too many temperature levels have to be used to explore a large temperature range to sample a good solutions. In our implementation, we expand the energy distribution of two adjacent temperature levels by multiplying a constant to the energy values. This heuristic approach works reasonably well and can help the system jump between adjacent temperature levels even if they are well separated. We call our approach *relaxed simulated tempering*. Our experimental results indicate that the *relaxed simulated tempering* approach outperforms *simulated annealing* in general. Compared with simulated annealing based slicing floorplan [WL89] design, we have improved wirelength by up to 12.2% with very small area variation.

*The work is partially supported by DARPA/ETO under Contract DAAL01-96-K-3600, NSF Young Investigator Award MIP9357582.

*The work by W.H. Wong and J.S. Liu is supported by NSF grants DMS-9703918 and DMS-9596096, respectively.

Since *simulated tempering* is a general Monte Carlo simulation technique, it has the potential to be applied to other VLSI design problems. In view of the impact of *simulated annealing* on the VLSI CAD community, it is reasonable to expect that *simulated tempering* can play an important role in solving many optimization problems in the VLSI CAD field.

The rest of this paper is organized as follows: Section 2 introduces the *simulated tempering* algorithm. Section 3 presents *simulated tempering* based slicing floorplan design. Section 4 shows our experimental results. Section 5 concludes the paper.

2 Simulated Tempering Algorithm

Simulated annealing [KGJ83] is one of the most popular stochastic optimization methods, and has been applied successfully to many optimization problems in VLSI layout area. This algorithm simulates the annealing process, in which the material is first heated to a high temperature near melting point, and then slowly cooled down so that it will crystallize into a highly ordered state. The time spent at each temperature must be sufficiently long to allow a thermal equilibrium to be approached. Theoretically, the temperature descent should be logarithmic [GG84]. In practice, much faster temperature cooling schedules have been used, such as geometrical, straight and reciprocal [SS94], in order to achieve reasonable runtime for practical solution. Thus, the high possibility to converge to the global minimum solution is no longer guaranteed.

In order to overcome this difficulty, *simulated tempering* treats the temperature as a random variable, so that the temperature of system can go up and down as the time progresses. The system can go back to high temperatures from time to time, thus provides a chance for the system to escape from some local minima. It is obvious that this process is different from that of *simulated annealing*, in which the temperature is strictly decreasing.

Generally, the *simulated tempering* approach selects a finite (often small) sequence of temperature levels, denoted by $t_1, t_2, \cdots, t_m$ ($t_1 > t_2 > \cdots > t_m$). For each temperature level, the corresponding Boltzmann distribution can be defined as follows:

$$h_i(x) = \alpha_j \exp\{-cost(x)/t_i\}, \qquad (1)$$

where $cost(x)$ is the cost function of configuration x; and α_j is the normalizing constant. These constants can be preliminarily estimated by an iteration procedure [KR94]. The configuration is updated following a Metropolis transition rule so that the configurations at each level t_i will be distributed according to Equation (1) at equilibrium. Usually, we are only interested in the configurations sampled at the lowest temperature, because better configurations are most likely to appear at lower temperatures. In *simulated tempering* , the Markov chain state space is augmented to be (x, i), where x takes values from the configuration space defined for $h_i(x)$, i is the temperature index and is now treated as a random variable. One iteration of *simulated tempering* involves an update of the current configuration x under temperature level i, followed by an update of temperature level i guided by Metropolis-Hastings algorithm [Ha70].

The general *simulated tempering* algorithm works as follows: we start from the highest temperature t_1. At each temperature t_i, we update the system configuration by s steps of Metropolis or Gibbs [GG84] iterations. Then we propose to move temperature t_i to an adjacent temperature level t_j according to the so-called Metropolis-Hastings ratio $r = \frac{\alpha_j}{\alpha_i} \exp\{-cost(x)(1/t_j - 1/t_i)\} \frac{q_{j,i}}{q_{i,j}}$ where $q_{1,2} = q_{m,m-1} = 1$ and $q_{i,i+1} = q_{i,i-1} = 0.5$ $(1 < i < m)$.

The algorithm is shown in Figure 1:

STA()
i = 1;
while (stop criterion is not satisfied)
  Update configuration x by s steps of Metropolis or Gibbs iterations;
  Propose a transition to an adjacent temperature according to $q_{i,j}$;
  Calculate The Metropolis-Hastings ratio r
  Accept the proposal(set i to j) with probability min(r,1) ;
endwhile

Figure 1: Simulated Tempering Algorithm STA()

By construction of the Markov process, *detailed balance* [KW86] will be automatically satisfied at every step. Thus after reaching equilibrium, the system will be distributed according to equation (1), and the significant local minima of the cost function will be sampled with high probabilities.

Although *simulated tempering* algorithm can help the system to jump out of local minima in the appearance of high energy barriers, it sometimes cannot mix the energy distribution of two adjacent temperatures well, if they are well separated. As a result, too many temperature levels have to be used to explore a large temperature range to sample a good solutions. This means too much computation time has to be spent. In order to further expand temperature range without using too many temperature levels, we applied a heuristic approach by modifying Metropolis-Hastings ratio calculation as following: $r = \frac{\alpha_j}{\alpha_i} \exp\{-\beta \times cost(x)(1/t_j - 1/t_i)\} \frac{q_{j,i}}{q_{i,j}}$, where β is the relaxation factor, which is a predefined constant taking value between 1 and 20. In our implementation, we set β to be 20.

After the *simulated tempering* algorithm **STA**() is finished, one usually samples a number of solutions from low temperature levels for doing fast *simulated annealing* and select the best result as the final output.

3 Simulated Tempering Based Slicing Floorplan Designs

The floorplan problem can be divided into slicing floorplan and non-slicing floorplan. Both slicing floorplan [St83, Ot83, WL89] and non-slicing floorplan approaches [WW90, PL93, MFNK95, NFMK96] have been investigated extensively. In this paper, we take slicing floorplan approach as an example and apply *simulated tempering* algorithm **STA**() for optimization. In [WL89], slicing floorplan design is obtained using simulated annealing algorithm based on normalized polish expression representation: Let horizontal and vertical cut be denoted by the operator + and *, respectively; and the modules be denoted by operands. There exists an one-to-one mapping from the set of slicing trees to the set of normalized Polish expressions. To explore different floorplan configurations using Polish expression approach, three type of moves, $M1$, $M2$, $M3$, were defined in [WL89]. Operation $M1$ swaps two adjacent operands; Operation $M2$ interchanges the operators * and + for a chain of

Table 1: Average Results for Slicing Floorplan

circuit	Simulated Annealing			Relaxed Simulated Tempering			Improvement		TimberWolf		
	area (mm^2)	w.l. (μm)	time (sec)	area (mm^2)	w.l. (μm)	time (sec)	area (%)	w.l. (%)	area (mm^2)	w.l. (μm)	time (sec)
apte	48.55	248189	46.85	48.55	240373	65.89	0	3.2	48.50	487710	66.0
xerox	21.44	478396	65.89	21.17	443097	42.99	1.3	7.4	22.64	526920	101.2
hp	9.78	140072	82.05	9.74	131100	59.33	0.4	6.4	9.58	186760	91.4
ami33	1.28	57795	228.00	1.32	50758	261.66	-3.1	12.2	1.27	71800	221.0
ami49	40.08	752546	439.72	40.94	750769	469.50	-2.2	0.2	40.81	814200	472.8
playout.xlii	94.85	5662390	1917.03	96.26	5362223	2249.60	-1.5	5.3	115.52	6220900	1599.2

nonzero length of adjacent operators; Operation $M3$ swaps two adjacent operand and operator. In our implementation, we added two more operations, named $M4$ and $M5$. Operation $M4$ swaps any two operands; Operation $M5$ swaps any two operand and operator if the resulting Polish expression is still normalized. By adding these two operations, we can make some global configuration change to speed up the algorithm. In our *simulated tempering* based floorplan algorithm, the operation is randomly selected from $M1$-$M5$ with equal probability to generate a new configuration. Whether this new configuration will be rejected or accepted is decided by the *simulated tempering* algorithm: **STA()**.

Suppose that a floorplan configuration is represented by $\mathcal{F}$, and that total area and total wirelength of $\mathcal{F}$ are denoted by $A(\mathcal{F})$ and $W(\mathcal{F})$ respectively. The commonly used cost function is as follows:

$$cost(\mathcal{F}) = A(\mathcal{F}) + \lambda W(\mathcal{F}) \quad (2)$$

However, the above cost function is not scalable in general. One often has to adjust coefficient λ for each individual floorplan instance. We choose to use the following normalized cost function in this paper.

$$cost(\mathcal{F}) = \gamma \frac{A(\mathcal{F})}{A_{ref}} + (1-\gamma)\frac{W(\mathcal{F})}{W_{ref}}; \quad (0 \leq \gamma \leq 1) \quad (3)$$

where γ is a constant between 0 and 1; A_{ref} and W_{ref} are pre-calculated by a very fast simulated annealing run. In our current implementation, routing area is not considered.

4 Experimental Results

We have compared *simulated annealing* and *simulated tempering* on 6 MCNC benchmark circuits [MCNC].

For each circuit, we ran each algorithm five times and reported average and minimum values in our experiment. In our implementation, the coefficient γ in the cost function (3) is set to be 0.5, which means that both area and total wirelength are equally important. For the *simulated tempering* algorithm, the number of sampled solutions p is set to be 5. The comparison of *simulated annealing* and *simulated tempering* for slicing floorplan is shown in Table 1 and Table 2. CPU times are measured on a Sun UltraSparc-1 workstation. We use the minimum bounding box metric to estimate total wirelength. Figure 2 shows a slicing floorplan solution obtained by using *simulated tempering* for circuit *playout.xlii*.

From Table 1 and 2, we can see that the wirelength is improved by up to 12.2% by the *simulated tempering* algorithm, when compared with the *simulated annealing* algorithm. Although the areas are slightly increased in some case, *simulated tempering* outperforms *simulated annealing* for the cost function defined in Equation 3.

To furthur validate our results, we did placement for these circuits using a commercial tool TimberWolf1.3.3. In our test, although we turned off the routing space estimation ability to let TimberWolf generate solutions without routing space, as our floorplan tool currently do not reserve routing space. Since TimberWolf allows the overlap of modules, we did compaction/decompaction to remove any overlaps to make a fair comparison. The results are also shown in Table 1. From these data, we can conclude that our results are better than those of TimberWolf's in general.

By analyzing the cost value of final solutions generated by *simulated annealing* and *simulated tempering* algorithms from different runs, we found that the cost values generated by *simulated annealing* are usually distributed in a very wide range; while the cost values generated by *simulated tempering* usually lie in a small range. This phenomenon indicates that *simulated annealing* solutions are more dependent on the initial solutions, which means they could be trapped at some local minima easily; while *simulated tempering* is more stable since it can escape from some local minima by warming up the system from time to time.

To make the *simulated tempering* algorithm to work well, the system should be able to move among the lowest and highest temperatures freely. Figure 3(a) shows a typical temperature versus time curve obtained by the original *simulated tempering* optimization process without relaxation for circuit *ami33*. Figure 3(b) shows the curve obtained from the *relaxed simulated tempering* optimization process for circuit *ami33*. It can be seen that the *relaxed simulated tempering* is very effective to help the system move among the lowest and highest temperature levels.

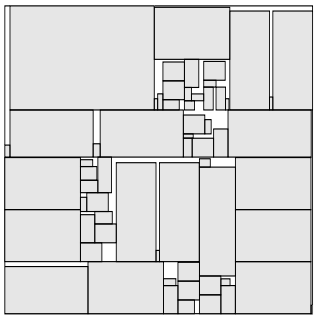

Figure 2: A slicing floorplan obtained by using *simulated tempering* for circuit *playout.xlii*

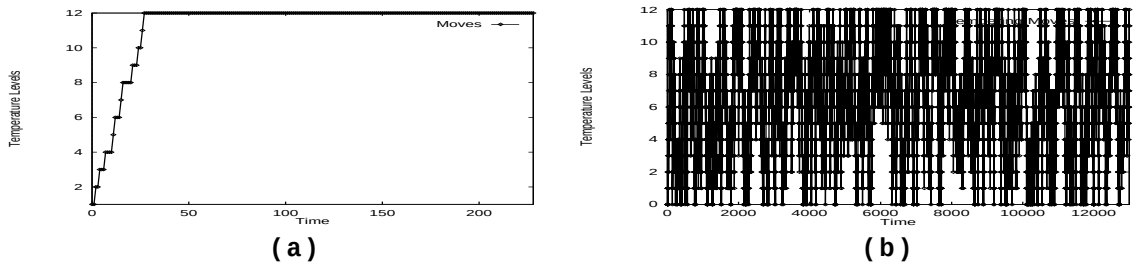

Figure 3: Temperature vs. time. (a) without relaxation; (b) with relaxation factor 20

Table 2: Minimum Results for Slicing Floorplan

circuit	Simulated Annealing			Relaxed Simulated Tempering			Improvement	
	area (mm^2)	w.l. (μm)	time (sec)	area (mm^2)	w.l. (μm)	time (sec)	area (%)	w.l. (%)
apte	48.50	229659	68.62	48.50	223761	25.00	0.0	2.6
xerox	21.05	422250	65.69	20.50	417368	35.76	2.6	1.2
hp	9.58	126670	76.90	9.58	114864	53.03	0.0	9.3
ami33	1.26	52746	217.24	1.29	48649	249.20	-2.4	7.8
ami49	39.42	714803	416.83	39.24	715818	429.50	0.5	-0.1
playout.xlii	92.51	5280413	1914.53	94.35	5100660	2242.94	-2.0	3.4

5 Conclusion

In this paper, we studied the effectiveness of *simulated tempering* technique, and have applied a relaxed version of *simulated tempering* algorithm to slicing floorplan design. Our experimental results indicate that *simulated tempering* with proper relaxation is a better Monte Carlo and optimization technique compared with *simulated annealing*. This is the first work adopting *simulated tempering* technique for solving optimization problems in the VLSI CAD field. We plan to extend this work to constrained floorplan designs and non-slicing floorplan designs in future. We expect to see more applications of *simulated tempering* to other VLSI design problems.

References

[GG84] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. In *IEEE Transaction on Pattern Analysis and Machine Intelligence*, pages 721-741, 1984.

[GT95] C.J. Geyer and E.A. Thompson Annealing Markov chain Monte Carlo with applications to ancestral inference. In *Journal of the American Statistical Association*, pages 909-920, 1995.

[Ha70] W.K. Hastings Monte Carlo sampling methods using Markov chains and their applications. In *Biometrika*, pages 97-109, 1970.

[KGJ83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi Jr. Optimization by simulated annealing. In *Science*, pages 671–680, May 1983.

[KR94] W. Kerler and P. Rehberg. Simulated tempering procedure for spin-glass simulations. In *Physical Review E*, pages 4220-4225, 1994.

[KW86] M.H. Kalos, P.A. Whitlock. Monte Carlo methods In *New York : Wiley*, 1986.

[MFNK95] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. Rectangle-packing-based module placement. In *Proc. of Int. Conf. on Computer-Aided Design*, pages 472–479, 1995.

[MP92] E. Marinari, and G. Parisi. Simulated tempering: a new Monte Carlo scheme. In *Europhysics Letters*, vol.19, (no.6), 15 July, 1992, pages 451–455.

[NFMK96] S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitani. Module placement on BSG-structure and IC layout applications. In *Proc. of Int. Conf. on Computer-Aided Design*, pages 484-491, 1996

[Ot83] R. Otten. Efficient floorplan optimization. In *Proc. of Int. Conf. on Computer-Aided Design*, pages 499–502, 1983.

[SS94] J. Stander and B.W. Silverman. Temperature schedules for simulated annealing. In *Statistics and Computing*, pages 21-32, 1994.

[PL93] P. Pan and C. L. Liu. Area minimization for general floorplans. In *Proc. of Int. Conf. on Computer-Aided Design*, pages 606–609, 1993.

[St83] L. Stockmeyer. Optimal orientation of cells in slicing floorplan designs. In *Information and Control*, pages 91–101, 1983.

[WL89] D.F. Wong and C. L. Liu. Floorplan design of VLSI circuits. In *Algorithmica*, pages 263–291, 1989.

[WL97] W. H. Wong and F. Liang. Dynamic weighting in Monte Carlo and optimization. In *Proc. National Academic Science, USA*, pages 14220–14224, Dec 1997.

[WW90] Ting-Chi Wang, and D.F. Wong. An optimal algorithm for floorplan area optimization. In *Proc. of Design Automation Conf.*, pages 180–186, 1990

[MCNC] "http://www.cbl.ncsu.edu/benchmarks/"