TWO-DIMENSIONAL PLACEMENT USING TABU SEARCH

John M. Emmert[1] and Dinesh K. Bhatia[2]

[1] Design Automation and Test Laboratory, Dept of EE
University of Kentucky
Lexington, KY 40506-0046, USA
emmert@engr.uky.edu
[2] Design Automation Laboratory, Dept of ECECS
University of Cincinnati
Cincinnati, OH 45221-0030, USA
dinesh@ececs.uc.edu

Abstract

Search based placement of modules is an important problem in VLSI design. It is always desired that the search should converge quickly to a high quality solution. This paper presents a tabu search based optimization technique to place modules on a regular two-dimensional array. The goal of the technique is to speed up the placement process. The technique is based on a two-step placement strategy. The first step is targeted toward improving circuit routability and the second step addresses circuit performance. The technique is demonstrated through placement of several benchmark circuits on academic as well as commercial FPGAs. Results are compared to placements generated by commercial CAE tools and published simulated annealing based techniques. The tabu search technique compares favorably to published simulated annealing based techniques, and it demonstrates an average execution time speedup of 20 with no impact on quality of results when compared to commercial tools.

Key Words: Placement, Floorplanning, Tabu Search, FPGA, VLSI, Circuit Mapping

1 INTRODUCTION

Two-dimensional placement is a well studied topic. However, the importance of placement cannot ever be ignored due to changing design complexities and requirements. One technology that is evolving very rapidly is field programmable gate array (FPGA). Currently, commercially available devices can map up to one million gate equivalent designs[12] and some of the newly announced products like Altera's APEX series[11] will map two million gate equivalent designs. Typical CAD flow for mapping circuits to FPGAs takes place in four inter-dependent steps: design entry, technology mapping, physical placement, and interconnect routing. Improvements in CAD tool technology for mapping circuits to FPGAs has not kept pace with hardware improvements. Currently it takes minutes to hours to map circuits of 10K gate equivalent designs to FPGAs. We need faster algorithms that provide high quality (relative to mapped circuit performance) results. Our goal is to speed up the mapping process by speeding up the placement step. Motivated with this goal, we have designed a two-step fast placement tool for array based designs. Our algorithms make use of tabu search[9] based optimization for finding good placement solutions. Circuit routability is enhanced using an *edge based* model for minimizing the total wire length of the circuit. The circuit timing optimizations are carried out using an edge based model for critical path length minimization. As a practical demonstration, we map several benchmark designs on FPGAs and compare the results against a simulated annealing search based technique for placement [21]. We also compare the results to the simulated annealing based, ultra fast placement work done in [22], and we compare results with commercial CAE tools from Xilinx, both the `XACT` 5.2 `PPR` tools and the `M1` tools. We show significant speedup relative to convergence on good quality solutions when compared to simulated annealing. Similarly we show favorable results when compared to [22], and we show an average execution time speedup of 20 with no impact on quality of results when compared to commercial tools.

This paper is organized as follows. In section 2 we describe fundamentals of the tabu search optimization technique. In sections 3 and 4 we formally describe the placement problem and related research. In section 5 we describe our two-dimensional placement solution. We formally describe the model we used for placing circuits on two-dimensional arrays. Then we describe our placement based on total wire length minimization and critical edge length minimization respectively. In sections 6 and 7 we describe our test methodology and analyze the data, and in section 8 we conclude the paper by providing a summary and ideas for future work.

```
Procedure GENERIC_TS()
   begin
   Initialize solution;
      repeat
         Perform IntenseSearch();
         Perform DiverseSearch();
      until stop criterion met;
      return BestSolution;
   end;
```

2 TABU SEARCH

In this section we present an overview to the tabu search optimization technique. Tabu search is a meta-heuristic approach for solving constrained optimization problems. When used properly, tabu search approaches near optimal solutions in a relatively short amount of time compared to other non-deterministic *random move* based methods [9]. Unlike approaches like simulated annealing that rely on good random choices, tabu search exploits both good and bad strategic choices to guide the search process. Tabu search uses the idea of a *move* to define the neighborhood of any given solution, and imposes restrictions in the form of a *tabu* on certain moves to avoid local optima.

As a meta-heuristic, tabu search guides local heuristic search procedures beyond local optima. In tabu search, a list of possible moves is created. In the short term, as moves in the list are executed, *tabu*, or restrictions, are placed on the executed moves in order to avoid local optima. This *tabu* is typically in the form of a time limit, and unless certain conditions are met (e.g. *aspiration criteria*), the move will not be performed again until the time limit has expired. This short term phase is associated with intensification of the search strategy. During the intensification stage, short term memory is used to explore closely related solutions in the local neighborhood. Good tabu search strategies also include long term memory that is used to diversify the search. Diversification moves the current solution out of the local neighborhood. For example, frequency of moves is a good candidate for long term memory storage. Penalties can be placed on moves that are frequently executed. Then, less frequently executed moves can lead the search to unexplored areas. Procedure GENERIC_TS shows a typical tabu search algorithm for solving general optimization problems.

3 PROBLEM

In this section, we formally describe the placement problem. We use M to represent a set of modules that will be placed onto a two-dimensional array L. The set S is used to represent the signals connecting the modules in M together.

Given a set of modules $M = \{m_1, m_2, ..., m_n\}$ and a set of signals $S = \{s_1, s_2, ..., s_q\}$, we associate with each module $m_i \in M$ a set of signals S_{m_i}, where $S_{m_i} \subseteq S$. Similarly, with each signal $s_i \in S$ we associate a set of modules M_{s_i}, where $M_{s_i} = \{m_j \mid s_i \in S_{m_j}\}$. M_{s_i} is said to be a *signal net*. We are also given a set of locations $L = \{l_1, l_2, ..., l_p\}$, where $p \geq \mid M \mid$. The placement problem then becomes how to assign each module $m_i \in M$ to a unique location $l_j \in L$ such that an objective function is optimized[4]. In our case, the objective function is to maximize circuit performance by minimizing path length. For the case of mapping $m_i \in M$ to a regular two-dimensional array, each $l_j \in L$ is represented by a unique (x_j, y_j) location on the surface of the two-dimensional array where x_j and y_j are integers. Figure 1 shows the 16 element set L for an example 4×4 two-dimensional array.

4 RELATED WORK

Much research is associated with placement and floorplanning for FPGAs [1, 2, 3, 4, 5, 6, 7, 8, 10, 13, 14, 15, 16, 17, 18, 19, 20, 23, 24, 25, 26]. However, use of the tabu search technique for placement is very limited. In this section we briefly describe research related to tabu search for placement or floorplanning.

Song and Vannelli developed a tabu search based placement algorithm for minimizing total wire length[23]. Their cost function was based only on total wire length, and therefore, designed to enhance routability and not performance, whereas our algorithm also improves circuit performance.

Lim, Chee, and Wu developed a tabu search based placement with global routing strategy for standard cells[14][15]. Their algorithm for standard cells is a divide and conquer strategy based on successive partitioning while ours uses force directed placement.

5 PLACEMENT

In this section we describe our two-dimensional placement solution. First we describe our model for abstracting the information from M and S into a Graph G. Then we describe our total wire length minimization and critical edge length minimization tabu searches respectively.

5.1 Model

We convert each multi-terminal net to a set of edges where each edge consists of the driving terminal and one driven terminal. We use this model to keep net sources and sinks in close proximity thereby enhancing circuit performance. We create the set of edges by converting

the hyper-graph input circuit model described earlier to a graph $G = (V, E)$ where $V = \{v_1, v_2, ...v_n\}$, $\mid V \mid = n$, $E = \{e_1, e_2, ...e_m\}$, and $\mid E \mid = m$. Each vertex $v_i \in V$ corresponds to a circuit module $m_i \in M$. Each edge $e_i \in E$ connects a pair of vertices $(v_j, v_k) \mid v_j, v_k \in V$. The elements of E are created by considering each signal, $s_i \in S$. If we let $m_j \in M_{s_i}$ be the source module for signal s_i then an edge (v_j, v_k) is added to E for each $m_k \in M_{s_i} \mid j \neq k$. At any given time, each element of V is mapped to a unique element of L, and the minimum requirement for mapping is $\mid V \mid \leq \mid L \mid$.

For the first step of our tabu search based placement strategy, TS_TWL, we seek to enhance routability by minimizing total wire length (TWL). We estimate TWL using the Manhattan length of each edge $e_i \in E$, and we seek to minimize the following function:

$$TWL = \sum_{\forall e_i \in E} MLength(e_i) \quad .$$

The second step of our tabu search based placement strategy, TS_EDGE, seeks to enhance circuit performance by minimizing the length of critical circuit edges. To accomplish this, we traverse G and determine a path weight pw_i for each path $p_i \in P$ where P is the set of all paths for G. For simplicity we let pw_i be the maximum level for each $p_i \in P$. Edges in critical paths receive a higher weight. Figure 2 shows an example circuit with six paths. In figure 2 path p_1 is at level 1; paths p_2 and p_3 are at level 2; and paths p_4, p_5, and p_6 are at level 3. We associate with each edge $e_j \in E$ a set of paths P_{e_j}, where $P_{e_j} \subseteq P$. For example, in figure 2 we have $P_{e_1} = \{p_1\}$ for $e_1 \in E$, $P_{e_2} = \{p_2, p_3, p_4, p_5, p_6\}$ for $e_2 \in E$, $P_{e_3} = \{p_4, p_5, p_6\}$ for $e_3 \in E$, $P_{e_4} = \{p_2\}$ for $e_4 \in E$, $P_{e_5} = \{p_3\}$ for $e_5 \in E$, $P_{e_6} = \{p_4\}$ for $e_6 \in E$, $P_{e_7} = \{p_5\}$ for $e_7 \in E$, and $P_{e_8} = \{p_6\}$ for $e_8 \in E$. Then we determine a weight w_j for each edge $e_j \in E$.

$$\forall\ e_j \in E, w_j = max(pw_i)\ \forall\ p_i \in P_{e_j}$$

For example, in figure 2 the weight for edge e_2 is the maximum path weight for each path in the set $\{p_2, p_3, p_4, p_5, p_6\}$ or $w_2 = 3$. Similarly for e_3 in figure 2, w_3 is the maximum path weight for the all paths in the set $\{p_4, p_5, p_6\}$ or $w_3 = 3$. The determination of the edge weights is accomplished with a breadth first search. Then we weight the Manhattan length of each edge $e_j \in E$ by multiplying the Manhattan length of edge e_j by its corresponding weight w_j. For our timing driven tabu search based step, TS_EDGE, we use a two part optimization function. First we minimize the weighted length of the longest edge. Second, since many configurations may have the same weighted longest edge length, we add together n of the longest edges (NLE) and minimize NLE in the event of a tie.

$$NLE = \sum_{i=1}^{N} MLength(e_i) \times w_i$$

5.2 Wire Length Minimization

Key to the development of a tabu search is a search list. For TS_TWL our search list U consists of all possible swaps of vertices occupying adjacent locations in L. This implies two basic swap moves: horizontal (swap of adjacent vertices with the same y coordinate) and vertical (swap of adjacent vertices with the same x coordinate). Valid swaps also include the exchange of a vertex from a position in L into an adjacent empty location in L. There are two reasons this move type was chosen: 1) to keep the move list short, and 2) to minimize the overhead of updating the move list after a move is executed. Given a two-dimensional array L of width W units and height H units, there are $\mid U \mid = ((H \times (W-1)) + ((H-1) \times W)) \approx 2(H \times W)$ possible swaps or moves in U. Therefore $U = \{u_1, u_2, ..., u_n\}$ where $n = \mid U \mid$. Figure 3 shows an example horizontal swap move u_i and vertical swap move u_j. For TS_TWL, given a random initial placement in L (by selecting an appropriate sequence of moves from U) we seek to optimize our objective function, minimization of TWL.

In TS_TWL, each move $u_i \in U$ has an associated attractiveness A_i or sum of the adjacent forces pulling on the vertices v_j and v_k that make up move u_i. For the vertical move we have

$$A_i = M(v_j) \times PE(v_j) + M(v_k) \times PW(v_k)$$

and for the horizontal move

$$A_i = M(v_j) \times PN(v_j) + M(v_k) \times PS(v_k).$$

Each vertex $v_i \in V$ has one multiplication factor $M(v_i)$ and four associated pulls or forces: $PN(v_i)$, $PE(v_i)$, $PS(v_i)$, and $PW(v_i)$. If the functions $X(v_i)$ and $Y(v_i)$ respectively return the current x and y coordinates of vertex v_i then,

$$PN(v_i) = \sum_{\forall e_k \in E_{v_i} | e_k = (v_i, v_j)} Y(v_j) - Y(v_i)$$

$$PE(v_i) = \sum_{\forall e_k \in E_{v_i} | e_k = (v_i, v_j)} X(v_j) - X(v_i)$$

$$PS(v_i) = \sum_{\forall e_k \in E_{v_i} | e_k = (v_i, v_j)} Y(v_i) - Y(v_j)$$

$$PW(v_i) = \sum_{\forall e_k \in E_{v_i} | e_k = (v_i, v_j)} X(v_i) - X(v_j).$$

For example figure 4 shows edges $e_1 = (v_1, v_2)$ and $e_3 = (v_1, v_3) \in E_{v_1}$. Therefore we can calculate $PN(v_1) = (Y(v_2) - Y(v_1)) + (Y(v_3) - Y(v_1)) = (2) + (-1) = 1$, $PS(v_1) = (Y(v_1) - Y(v_2)) + (Y(v_1) - Y(v_3)) = (-2) + (1) = -1$, $PE(v_1) = (X(v_2) - X(v_1)) + (X(v_3) - X(v_1)) = (-1) + (-2) = -3$, and $PW(v_1) = (X(v_1) - X(v_2)) + (X(v_1) - X(v_3)) = (1) + (2) = 3$. Overall

we see vertex v_1 has a slight pull to the *north* and a strong pull to the *west*. Similarly in figure 4 we see vertex v_4 has $PN(v_4) = PS(v_4) = 0$, $PE(v_4) = 2$, and $PW(v_4) = -2$ due to edge $e_2 = (v_4, v_5) \in E_{v_4}$. Figure 4 shows horizontal move u_i consists of swapping the positions of vertices v_1 and v_4. If initially $M(v_j) = 1\ \forall v_j \in V$ we can calculate the attractiveness A_i for horizontal move u_i in figure 4, $A_i = M(v_4) \times PE(v_4) + M(v_1) \times PW(v_1) = 1 \times 2 + 1 \times 3 = 5$. In a similar manner A_i is calculated for each $u_i \in U$. (For a given move $u_i \in U$ if one(both) of the adjacent slots is(are) empty of vertices then the pull(s) corresponding to the empty slot(s) is(are) set to 0.)

If we used the move list U in a typical greedy search strategy (i.e. given an initial placement find a move that would improve the minimum TWL) we would quickly reach a local optima. This local optima would probably not be close to the global optima or minimum TWL. However, by applying the concepts of tabu search i.e. accepting strategic moves that may not improve the current minimum TWL we climb out of local optima to rapidly converge on near optimal solutions. After executing move $u_i \in U$ we set a tabu tenure for u_i. Move u_i will not be executed again until the tabu tenure has expired or our aspiration criteria is satisfied. In this way we climb out of local optima and accept the current *best* move even if it does not improve the current minimum TWL.

5.3 Timing Driven Placement

For our timing driven tabu search based algorithm, TS_EDGE, we use the edge list E as our search list. We order our edge list E in descending order according to each edge's weighted Manhattan length. Then search the edge list looking at each of the two vertices attached to each edge as possible candidates for a move. Therefore in algorithm TS_EDGE, $E = \{e_1, e_2, ..., e_n\}$ where $n = \mid E \mid$ is our search or move list. The vertices attached to the edges with the longest weighted Manhattan lengths are the most attractive candidates for moving closer together. By moving these vertices closer together, the longest edges are shortened thereby enhancing circuit performance and reducing the longest paths. Once an edge is selected from the search list, we look at only one of the edge's two vertices as a possible move candidate. For simplicity we pick one of two possible moves for the vertex selected: vertical swap or horizontal swap. For the horizontal swap adjacent vertices with the same y coordinate are swapped. For the vertical swap adjacent vertices with the same x coordinate are swapped. Figure 5 shows an example horizontal swap move for vertex v_2 attached to vertex v_1 by edge $e_i \in E$. In this case the Manhattan length of edge e_i is reduced by one. Figure 6 shows an example vertical swap move for the vertex v_5 attached to vertex v_4 by edge $e_j \in E$. Similarly edge e_j is reduced by one. In our tabu search based algorithm, TS_EDGE, given a random or otherwise initial placement in L (by selecting an appropriate sequence of moves from E)

we seek to optimize our objective function, minimization of the longest weighted edge length (or in the case of a tie, NLE).

If we used the search list E in a typical greedy search strategy (i.e. given an initial placement find a move that would improve the current minimum longest edge length) we would quickly reach a local optima. This local optima would probably not be close to the global optima or minimum longest edge length/NLE combination. However, by applying the concepts of tabu search, i.e. accepting strategic moves that may not improve the current minimum NLE, our TS_EDGE algorithm climbs out of local optima to rapidly converge on near optimal solutions. After executing a move for a vertex on edge $e_i \in E$ we set a tabu tenure (number of iterations a vertex' position is locked) for the moved vertex on edge e_i. This vertex on edge e_i will not be moved again until the tabu tenure has expired or our *aspiration criteria* is satisfied. In this way we climb out of local optima and accept the current *best* move even if it does not improve the current best solution.

6 TEST METHOD

We empirically tested TS_TWL, TS_EDGE, and the 2-step (TS_TWL followed by TS_EDGE) tabu search based placement methodologies described above using `Xilinx Netlist Format` (XNF) benchmark circuits available from MCNC (`email benchmarks@mcnc.org`) and netlist benchmark circuits from the University of Toronto (http://www.eecg.toronto.edu/~vaughn). For comparison to the simulated annealing technique, we placed the benchmark circuits using our algorithms as well as a simulated annealing placement algorithm with the same TWL cost function [21]. For the XNF benchmark circuits (see table I), we used Xilinx routers to route all placed circuits. Additionally we placed and routed the XNF benchmark circuits using Xilinx `PPR` and the more recent `M1` tools for comparison of execution times and result quality. We used statistics available from the Xilinx tools to compare XNF circuit placement quality. For the benchmark circuits from Toronto (see table II), we placed the circuits using our TS_TWL algorithm and compared the execution time to the ultra fast placement [22] work done at Toronto. We assume each of the vertices in the circuits can be mapped to one and only one location on the smallest square array L such that $\mid L \mid = \lceil \sqrt{\mid V \mid} \rceil^2$. The `unix` *time* function was used to determine system placement times for our tabu search algorithms and the simulated annealing based algorithm.

7 RESULTS

Figures 7, 8, and 9 show respectively execution time on the x-axis and current minimum TWL on the y-axis for example runs of TS_TWL and simulated annealing on XNF circuits c2670, c3540, and c6288. In each of the figures, the solid line shows TWL versus execution time for TS_TWL and the dashed lines show TWL versus execution time for the simulated annealing algorithm set for three different rates of convergence. These figures demonstrate the fast convergence of the generic tabu search on good solutions relative to the generic simulated annealing algorithm. The simulated annealing algorithm approaches the near optimal solution of minimum TWL, but it suffers from either slow start up time or slow overall rate of convergence depending on how the simulated annealing parameters are chosen. Overall, these graphs demonstrate the applicability of using tabu search as a stand alone placement tool or as an ideal first pass placement method for initializing the input for further refinement by simulated annealing or other random move based approaches to placement. It should be noted that many methods exist to enhance simulated annealing, but many of these can also be applied to tabu search to speed it up.

Table III shows average execution times for the placement step of circuit mapping by `PPR`, `M1`, TS_TWL, TS_EDGE, and combined tabu search approach (TS_TWL then TS_EDGE). The same random placements were used as inputs to all algorithms (except for `M1` where we had no control over technology mapping). All algorithms and tools were executed on an `ULTRASPARC1` workstation. Times for `PPR` and `M1` were taken for the default tool settings and just used for comparison. We found that performing a 2-step approach (TS_TWL then TS_EDGE) could greatly reduce the execution time of our tabu search based algorithms. Obviously a true comparison cannot be made between tabu search and the commercial tools since the objective functions of the commercial tools are unknown; however, we provide data from the commercial tools for informational purposes. The 2-step tabu search is approximately 25 times faster than `PPR` and 20 times faster than `M1`.

Table IV shows the static delay calculations done on the postrouted XNF circuits. The worst case static pad-to-pad delay for the 2-step tabu search is very similar to that of the XNF circuits placed by `PPR` and `M1`. Therefore for the benchmark XNF circuits used to test the tabu search method, result quality was similar to that of the commercial tools.

Table V shows the execution time comparison of TS_TWL to the Ultra Fast placement tool and the modified VPR tool (modified to improve execution time at the cost of placement quality[22]) from the University of Toronto[22]. A direct comparison of the algorithms cannot be made since they have different goals and cost functions; however, relative to execution time, the tabu search method is on the same order as that of the tools from Toronto.

8 CONCLUSIONS

We have described a two-step routability and performance driven tabu search based search algorithm for placement of circuits on two-dimensional arrays. Our tabu search based method performs extensive local and diversified searches that result in very well placed circuits resulting in high performance. We demonstrated the approach with benchmark circuits from MCNC and the University of Toronto. For the benchmark circuits from MCNC, we compared the results to both simulated annealing[21] and commercial tools. Our results demonstrate that good placement was determined much quicker and of similar quality to that of the commercial tools. For the benchmark circuits from Toronto, our placement times compared favorably to the Ultra Fast placement work at the University of Toronto. We feel the tabu search based methods presented here can be used in a stand alone fast placement tool for large designs or for fast initial placement to use as input to other placement algorithms.

Our future work includes adding routability and performance estimation to the tabu search based placement approach. This will reduce the necessity of successively performing the placement step and allow feedback for any necessary iterations of the circuit mapping process.

9 REFERENCES

[1] M. J. Alexander, J. P. Cohoon, J. L. Colflesh, J. Karro, E. L. Peters, and G. Robins, "Placement and Routing for Three-Dimensional FPGAs," *Proceedings of the 4th Canadian Workshop on Field-Programmable Devices*, pages 11–18, May 1996.

[2] V. Betz and J. Rose, "VPR: A New Packing, Placement, and Routing Tool for FPGA Research," *Lecture Notes in Computer Science*, volume 1304, pages 213–222, Springer-Verlag, 1997.

[3] J. P. Blanks, "Near-Optimal Placement Using a Quadratic Objective Function," *Proceedings of the 22nd ACM/IEEE Design Automation Conference*, pages 609–615, 1985.

[4] M. A. Breuer, "Min-Cut Placement," *Journal of Design Automation and Fault Tolerant Computing*, volume 1, pages 343–362, October 1977.

[5] T. J. Callahan, P. Chong, A. DeHon, and J. Wawrzynek, "Fast Module Mapping and Placement for Datapaths in FPGAs," *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 123–132, February 1998.

[6] C. S. Chen, Y. W. Tsay, T. T. Hwang, A. C. H. Wu, and Y. L. Lin, "Combining Technology Mapping and Placement for Delay-Minimization in FPGA Designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 14, pages 1076–1084, September 1995.

[7] J. P. Cohoon and W. D. Parris, "Genetic Placement," In *Proceedings of the IEEE International Conference on Computer-Aided Design*, pages 422–425, 1986.

[8] C. Ebeling, L. McMurchieand S. A. Hauck, and S. Burns, "Placement and Routing Tools for the Triptych FPGA," *IEEE Transactions on VLSI Systems*, volume 3, pages 473–482, December 1995.

[9] F. Glover and M. Laguna, *Tabu Search*, Kluwer Academic Publishers, 1997.

[10] K. M. Hall, "An r-Dimensional Quadratic Placement Algorithm," *Management Science*, pages 219–229, November 1970.

[11] Altera Inc., *http://www.altera.com*.

[12] Xilinx Inc., *http://www.xilinx.com*.

[13] R. M. King and P. Banerjee, "ESP: Placement by Simulated Evolution," *IEEE Transactions on Computer-Aided Design*, volume 8, pages 245–256, March 1989.

[14] A. Lim, "Performance Driven Placement Using Tabu Search," *Informatica*, volume 7, number 1, 1996.

[15] A. Lim, Y. M. Chee, and C. T. Wu, "Performance Driven Placement with Global Routing for Macro Cells," *Proceedings of Second Great Lakes Symposium on VLSI*, pages 35–41, 1991.

[16] A. Mathur and C. L. Liu, "Compression-Relaxation: A New-Approach to Performance Driven Placement for Regular Architectures," *Proceedings of the International Conference on Computer Aided Design*, pages 130–136, 1994.

[17] M. Mogaki, C. Miura, and H. Terai, "Algorithm for Block Placement with Size Optimization Technique by the Linear Programming Approach," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pages 80–83, 1987.

[18] S. K. Nag and R. A. Rutenbar, "Performance-Driven Simultaneous Place and Route for Island-Style FPGAs," *SRC Pub C95174*, June 1995.

[19] S. Raman, C. L. Liu, and L. G. Jones, "Timing-Constrained FPGA Placement: A Force-Directed Formulation & Its Performance Evaluation," *VLSI Design: An International Journal of Custom-Chip Design, Simulation, and Testing*, 1994.

[20] K. Roy, B. Guan, and C. Sechen, "A Sea-of-Gates Style FPGA Placement Algorithm," *Journal of VLSI Design: Special Issue on FPGAs*, 1993.

[21] S. M. Sait and H. Youssef, *VLSI Physical Design Automation*, IEEE Press, 1995.

[22] Y. Sankar and J. Rose, "Trading Quality for Compile Time: Ultra-Fast Placement for FPGAs," *ACM Seventh International Symposium on Field-Programmable Gate Arrays*, pages 157–166, February 1999.

[23] L. Song and A. Vannelli, "A VLSI Placement Method Using Tabu Search," *Microelectronics Journal*, volume 23, pages 167–172, May 1992.

[24] P. R. Suaris and G. Kedem, "An Algorithm for Quadrisection and Its Application to Standard Cell Placement," *IEEE Transactions on Circuits and Systems*, volume 35, pages 294–303, March 1988.

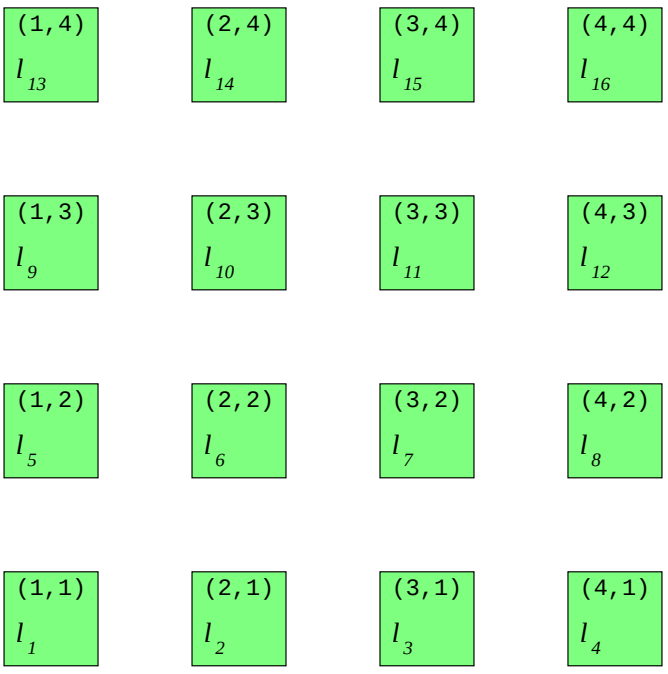
[25] A. Subramaniam and D. Bhatia, *Timing Driven Placement for Logic Cell Arrays*, PhD thesis, University of Cincinnati, 1994.

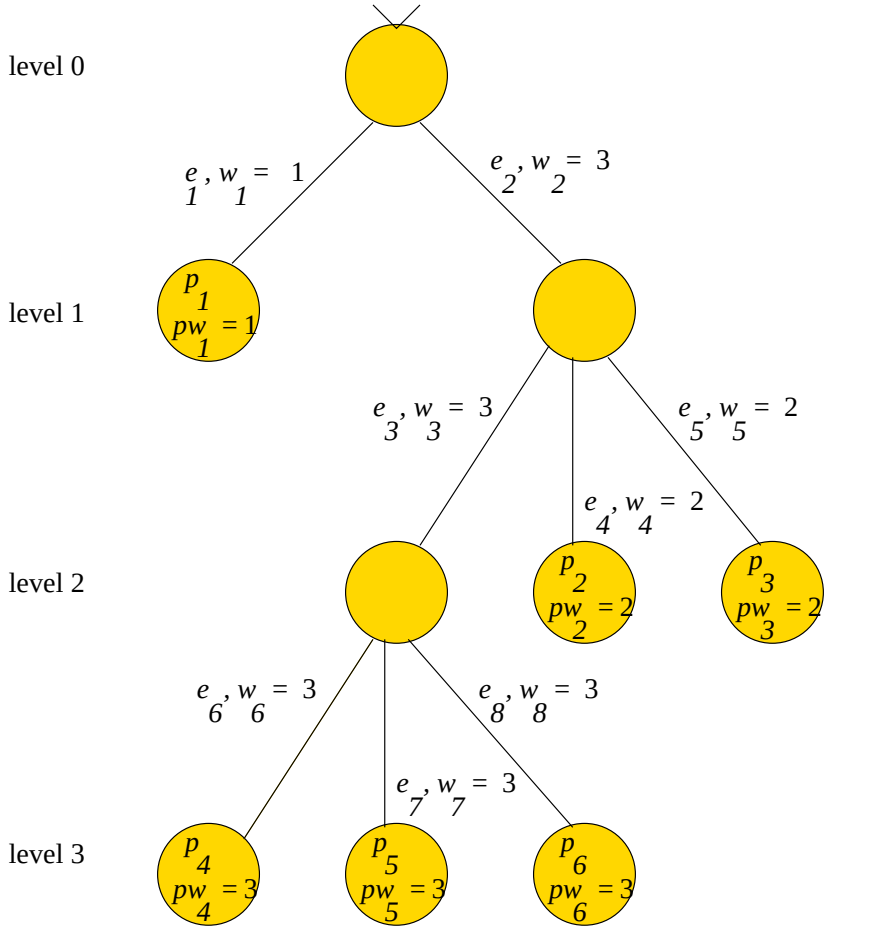
[26] N. Togawa, K. Hagi, M. Yanagisawa, and T. Ohtsuki, "An Incremental Placement and Global Routing Algorithm for Field-Programmable Gate Arrays," *Asia-South Pacific Design Automation Conference Proceedings*, page 8C.2, 1998.

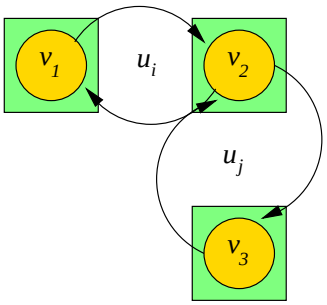
10 BIOGRAPHIES

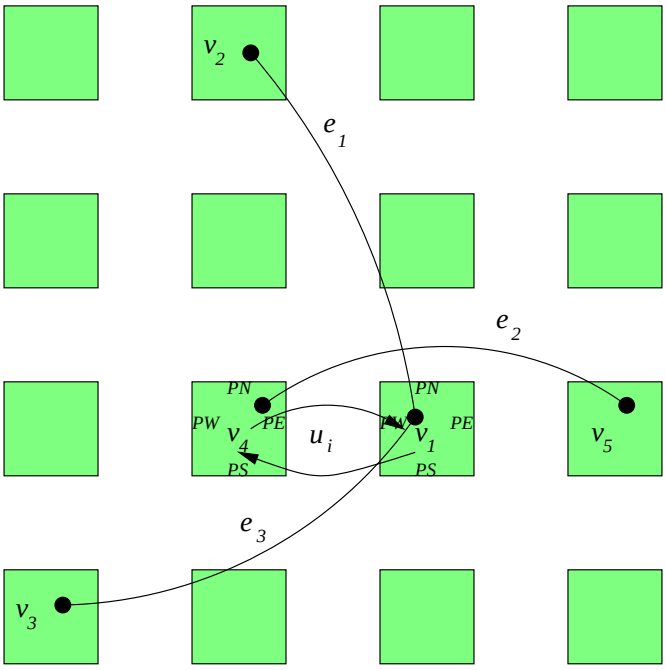
John Emmert received a Bachelor's of Science degree in Electrical Engineering from the University of Kentucky, Lexington, Kentucky in 1987. He was commissioned and spent seven years on active duty with the United States Air Force. In 1993 he received a MS in Electrical Engineering from the Air Force Institute of Technology, Dayton, Ohio, and in 1999 he received a Ph.D. in Computer Science and Engineering from the University of Cincinnati, Cincinnati, OH. His doctoral work was sponsored by grants from the state of Ohio and Defense Advanced Research Projects Agency (DARPA). Currently he is an Assistant Professor in the Department of Electrical Engineering at the University of Kentucky. He is also co-director of the Design Automation and Test Laboratory. He is a member of IEEE, ACM, and Eta Kappa Nu.

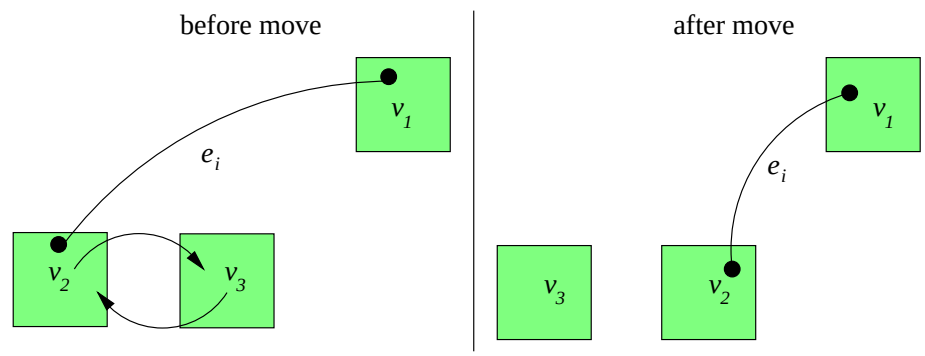
Dinesh Bhatia received a Bachelor's in Electrical Engineering from Regional Engineering College, Suratkal, India in 1985 followed by a MS and Ph.D. in Computer Science from University of Texas at Dallas in 1987 and 1990 respectively. His doctoral work was supported by ACM SIGDA scholarships. He joined the Computer Science and Engineering Department at the Southern Methodist University in Dallas in 1990 and moved to University of Cincinnati in 1991. Currently he is Associate Professor in the Department of Electrical and Computer Engineering and Computer Science at the University of Cincinnati. He also directs the Design Automation Laboratory and his research interests include all aspects of architecture and CAD for field programmable gate arrays, reconfigurable and adaptive computing, physical design automation of VLSI systems, applied graph theory, and algorithms. He has authored over fifty papers and has been invited to present tutorials at various conferences. He also served as a guest editor for a special issue on field programmable gate arrays (FPGAs) for the VLSI Design journal. He is an associate editor of IEEE Transactions on Computers. His research is actively supported by Defense Advanced Research Projects Agency (DARPA), the United States Air Force, and various industries. He is a member of IEEE, ACM, and Eta Kappa Nu.

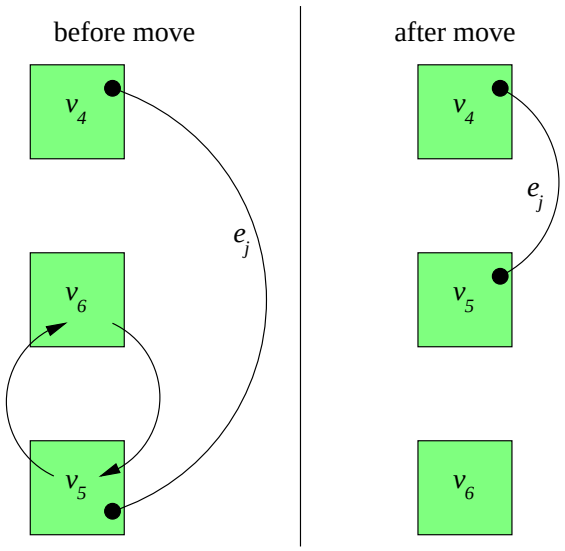

(1,4)
l_{13}
(2,4)
l_{14}
(3,4)
l_{15}
(4,4)
l_{16}
(1,3)
l_{9}
(2,3)
l_{10}
(3,3)
l_{11}
(4,3)
l_{12}
(1,2)
l_{5}
(2,2)
l_{6}
(3,2)
l_{7}
(4,2)
l_{8}
(1,1)
l_{1}
(2,1)
l_{2}
(3,1)
l_{3}
(4,1)
l_{4}

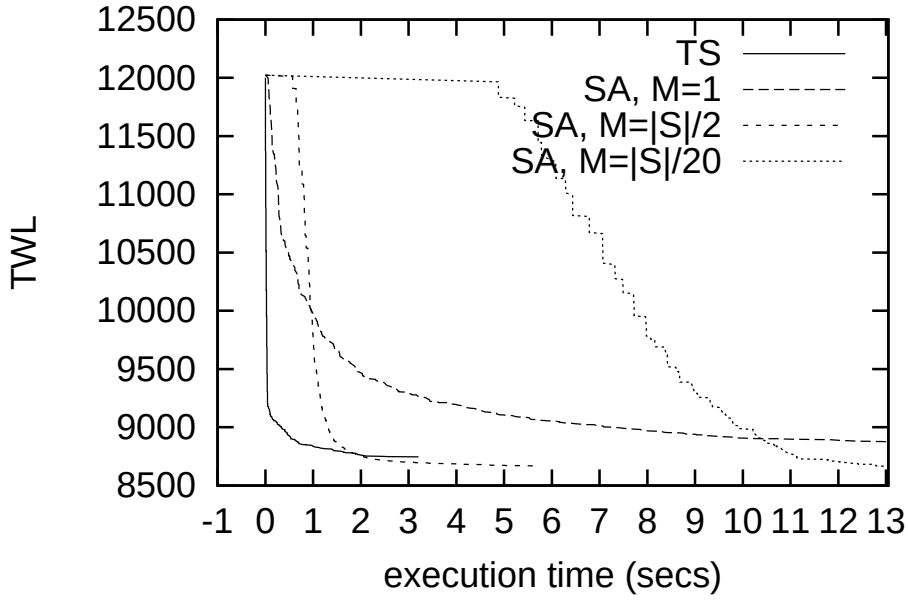

level 0
level 1
level 2
level 3
e1, w1 = 1
e2, w2 = 3
e3, w3 = 3
e4, w4 = 2
e5, w5 = 2
e6, w6 = 3
e7, w7 = 3
e8, w8 = 3
p1 pw1 = 1
p2 pw2 = 2
p3 pw3 = 2
p4 pw4 = 3
p5 pw5 = 3
p6 pw6 = 3

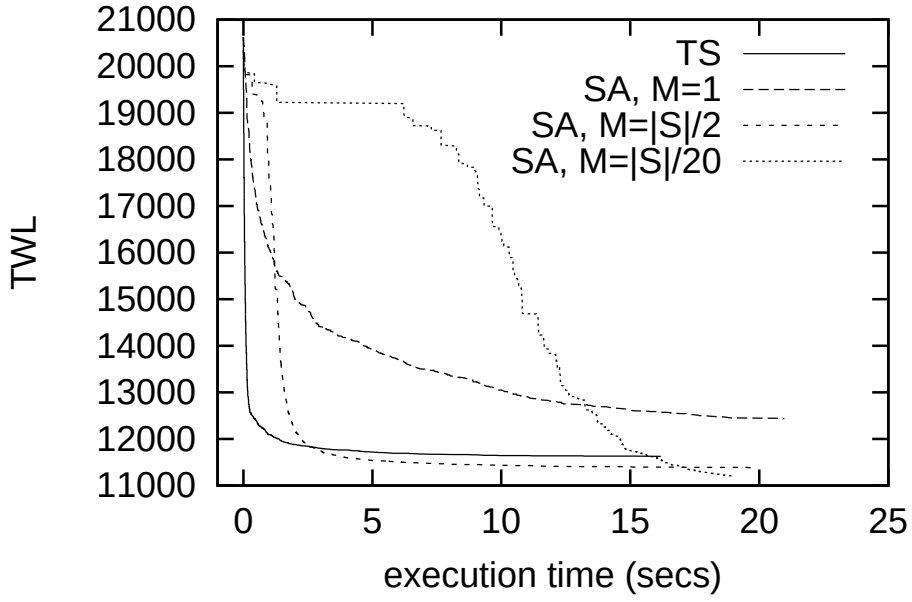

v_1
u_i
v_2
u_j
v_3

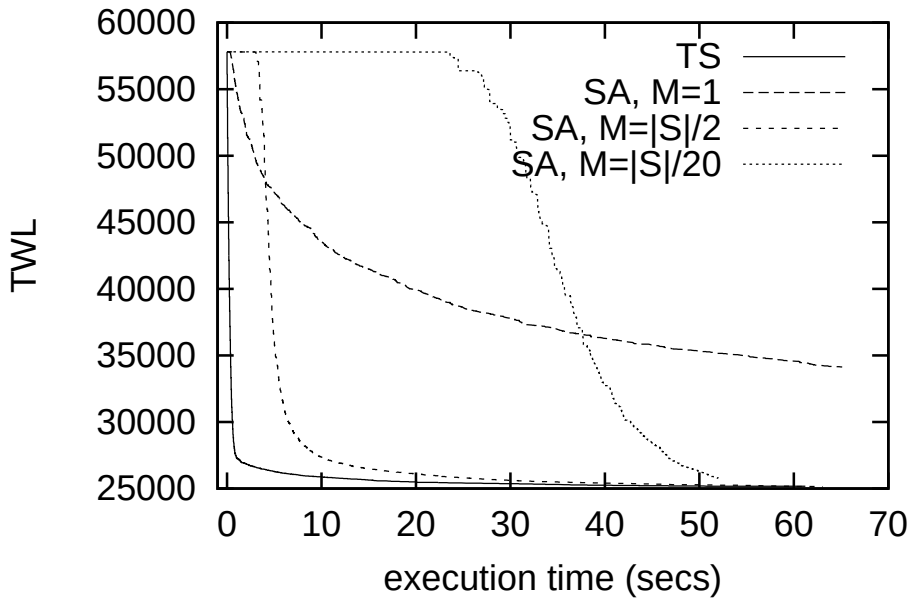

v_2
e_1
e_2
PN
PW
PE
v_4
PS
u_i
PN
PW
PE
v_1
PS
v_5
e_3
v_3


before move
after move
v_1
e_i
v_2
v_3
v_3
v_1
e_i
v_2


before move
after move
v_4
v_6
v_5
e_j
v_4
e_j
v_5
v_6


12500
12000
11500
11000
10500
10000
9500
9000
8500
TWL
-1 0 1 2 3 4 5 6 7 8 9 10 11 12 13
execution time (secs)
TS
SA, M=1
SA, M=|S|/2
SA, M=|S|/20


21000
20000
19000
18000
17000
16000
15000
14000
13000
12000
11000
TWL
0
5
10
15
20
25
execution time (secs)
TS
SA, M=1
SA, M=|S|/2
SA, M=|S|/20


60000
55000
50000
45000
40000
35000
30000
25000
TWL
0
10
20
30
40
50
60
70
execution time (secs)
TS
SA, M=1
SA, M=|S|/2
SA, M=|S|/20

XNF Circuit Data		
Ckt	$\mid V \mid$	$\mid E \mid$
c432	88	237
c499	129	270
c880	181	417
c1355	288	581
c1908	185	563
c2670	398	758
c3540	361	1427
c6288	848	2992

University of Toronto Circuit Data	
Ckt	$\mid V \mid$
clma.net	8383
elliptic.net	3604
ex1010.net	4598
frisc.net	3556
pdc.net	4575
s38417.net	6406
s38584.net	6447
spla.net	3690

Execution Times (secs)					
Ckt	**PPR**	**M1**	**TS_TWL**	**TS_EDGE**	**TS 2-Step**
c432	16	27	0.6	1.9	0.9
c499	19	34	0.7	2.0	0.9
c880	34	35	0.8	3.8	1.3
c1355	37	40	1.4	2.7	1.8
c1908	51	75	1.7	3.0	2.3
c2670	90	75	2.1	4.8	3.8
c3540	166	159	4.5	7.2	6.2
c6288	355	169	8.7	15.9	13.1

Static Timing Analysis (ns)						
Ckt	**SA**	**PPR**	**M1**	**TS_TWL**	**TS_EDGE**	**TS 2-Step**
c432	65.5	63.4	67.6	65.9	62.8	63.8
c499	40.5	40.0	45.3	40.1	37.4	40.0
c880	49.1	55.7	49.0	54.5	56.4	54.8
c1355	60.0	58.3	61.7	63.9	65.6	64.3
c1908	62.1	56.8	67.1	60.2	62.5	59.7
c2670	93.4	100.6	118.5	112.0	110.0	93.6
c3540	84.9	82.9	79.6	115.5	98.4	91.5
c6288	451.0	378.0	380.8	429.0	414.9	398.0

Execution Times (secs)			
Ckt	**Toronto Ultra Fast**	**Modified VPR**	**TS_TWL**
clma.net	21.71	29.79	55.5
elliptic.net	6.05	7.16	2.1
ex1010.net	7.96	10.53	14.4
frisc.net	6.15	7.04	4.4
pdc.net	8.43	10.35	18.4
s38417.net	13.33	16.88	4.2
s38584.1.net	14.55	18.35	32.1
spla.net	6.37	7.26	13.4