# Integrated Floorplanning with Buffer/Channel Insertion for Bus-Based Microprocessor Designs[1]

Faran Rafiq
Intel Microlectronics Services, 20325 NW Von Neumann Dr. AG3-318, Beaverton, OR 97006
faran.rafiq@intel.com

Malgorzata Chrzanowska-Jeske
ECE Portland State University, 1900 SW 4th Ave. Portland, OR 97207
jeske@ee.pdx.edu

Hannah Honghua Yang
Strategic CAD Labs, Intel Corporation, Hillsboro, Oregon, 97214
honghua.yang@intel.com

Naveed Sherwani
Intel Microlectronics Services, 20325 NW Von Neumann Dr. AG3-318 Beaverton, OR 97006
naveed.sherwani@intel.com

## ABSTRACT
A new approach to the interconnect-driven floorplanning problem that integrates bus planning with floorplanning is presented. The integrated floorplanner is intended for bus-based designs. Each bus consists of a large number of wires. The floorplanner ensures routability by generating the exact location and shape of interconnects (above and between the circuit blocks) and optimizes the timing. Experiments with MCNC benchmarks clearly show the superiority of integrated floorplanning over the classical floorplan-analyze-and-then-re-floorplan approach. Our floorplans are routable, meet all timing constraints, and are on average 12-13% smaller in area as compared to the traditional floorplanning algorithms.

## Categories and Subject Descriptors
B.7.2 [Hardware}: Integrated Circuits-Placement and Routing; J.6 [Computer Applications]: Computer-aided Engineering – CAD.

**General Terms**: Algorithms, Performance.

**Keywords:** Floorplanning, Routability, Interconnect Estimation.

## 1. INTRODUCTION
One of the critical problems in physical design, interconnect-driven floorplanning, is attracting more attention as technology moves deeper into deep-sub-micron (DSM) and the timing convergence problem is getting more difficult. Floorplanning issues start to be considered very early in a design cycle and it is not uncommon to evaluate possible floorplanning solutions on behavioral and RTL levels. A number of quality measures, like timing, power, congestion and routabilty need to be evaluated repeatedly, which could be very time consuming. Therefore, it seems logical to consider specific features of different design types when developing approaches to a floorplanning problem. At the full chip level of microprocessor designs, 30-40 large blocks are connected by a large number of nets that originate and terminate at the topologically same locations. Some of these nets are logical buses, such as data and control buses, but others are simplly individual nets that just originate and terminate at the same locations. This is in sharp contrast to ASIC chips, where a large number of small blocks are connected by tens of thousands of nets. Therefore, in microprocessor and also System-On-Chip (SOC) type designs, many of interconnects can be treated as busses composed of bundles of wires with related timing, topology and parasitic requirements. Such bus-based approach reduces a number of connections and, therefore, allows for easier evaluation of a floorplan quality in respect to delay and routability.

Hence, for microprocessor and SOC type designs we need a floorplanning algorithm that plans the bus topology and performs its optimization within the optimization of the floorplan itself, and also inserts channels and repeater blocks, as needed, to satisfy interconnect timing constraints.

The influence of floorplanning on timing performance of a design has usually been evaluated using a simple estimation of wire lengths. Although, most of the existing algorithms [2, 3, 4, 5, 6, 8, 9] have a wire cost in their objective function, it is in the form of very approximate total center-to-center or half-perimeter length evaluation. In DSM designs where interconnects play a dominant role in the overall performance of the chip, this simple cost estimation technique does not ensure routability nor does it address the issues of timing and congestion optimization. Moreover, none of the existing algorithms consider the area increment due to channel or buffer insertions that might be necessary during the later stages of the design process. Failure to consider this area penalty during floorplanning will definitely incur expensive area expansion later. Realizing the importance of interconnect planning at an early stage of design cycle, Nakatake et al. [5] have combined the routability issue with BSG (Bound Sliced Grid). However, their method deals with channel routing only and is not applicable for chip level designs where routing can be done above the blocks.. In [1] Chen et al. integrated interconnect planning with floorplanning for net-based designs. They used a probabilistic router using edge congestion for routing estimation. Their experimental results show that their method is superior to traditional algorithms. However, their approach is applicable for net-based (ASICs) designs only.

We have developed a bus-based approach to floorplanning of microprocessor designs, including channel and buffer insertions. It is the first reported algorithm for bus-based designs that allocates the actual geometric locations for busses above and between the blocks. It is accomplished by (1) using a mixed integer linear programming (MILP) for bus routing, (2) inserting channels when routing is not possible above the blocks, and (3) inserting repeaters for buses that are missing their timing specifications. The objective


*ISPD'02,* April 7-10, 2002, San Diego, California, USA.


[1] This work was supported in part by the NSF grants MIP-9629419 and CCR-9988402

is to optimize the 'overall' area (block, channel and repeater areas) while meeting the timing specifications. Our approach is based on a well-known simulated annealing (SA) algorithm presented by Wong and Liu in [10]. We have tested our floorplanner on MCNC benchmark circuits and obtained, on an average, 42.5% and 89% improvements in terms of routability and timing, respectively. Moreover, since the area increment due to channel and buffer insertions is considered during the optimization of the floorplann our algorithm produces, on an average, 13% improvement in terms of the overall area.

This paper is organized as follows: Section 2 describes the traditional floorplanning flow and some of its limitations. Section 3 gives the description of our new approach. Section 4 and 5 describe bus planning and channel/repeater insertion approaches, respectively. Results are given in Section 6 and section 7 concludes the paper.

## 2. OVERVIEW

The traditional approach to floorplanning (Fig.1) starts with the optimization of blocks' locations, such that the area of the smallest enclosing rectangle containing all blocks is minimized. This is followed by interconnect planning and timing analysis to uncover any timing violations. For nets that are missing their timing specification repeaters are inserted. Then, if routing congestion does not allow all nets to be routed, channels are inserted. Unfortunately, both repeater and channel insertions might invalidate the previous timing, and may require re-floorplanning of the design that can lead to a convergence problem. Given the complexity of the chips, it is not uncommon for microprocessor designs to sometimes take months to converge. Hence, to create a performance-feasible floorplan, a floorplanner that simultaneously considers area, exact interconnect locations, and repeater and channel insertions is needed.

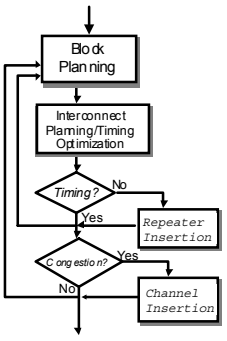


**Figure.1 Traditional Floorplanning Flow**

## 3. INTEGRATED FLOORPLANNING AND BUFFER INSERTION PROBLEM (IFBP)

### 3.1 Problem Formulation

The input to the floorplanning problem is a set of $n$ rectangle blocks $\{B_1, B_2, \ldots, B_n\}$ and a corresponding list of $n$ triplets of numbers $(A_1,r_1,s_1), (A_2,r_2,s_2), (A_3,r_3,s_3),\ldots, (A_n,r_n,s_n)$. We are given two sets $S_1$ and $S_2$ such that $S_1 \cup S_2 = \{B_1,B_2,\ldots,B_n\}$. The set $S_1$ specifies the set of modules with fixed orientations and the set $S_2$ represents the modules with free orientation. The triplets of numbers $(A_i,r_i,s_i)$, with $r_i \le s_i$ specifies the area and shape constraints for module $i$. Symbols $r_i$ and $s_i$ are the lower and upper bound constraints on the shape (aspect ratio) of block $i$ .( $r_i \neq s_i$ if the block is flexible, and $r_i = s_i$ if the block is rigid). Symbols $p$ and $q$ represent the upper and lower bound constraints on the shape of the enclosing rectangle. We are also given a set of nets $\{N_1, N_2, , N_m\}$. Each net is assigned a width $z_j$ and a timing spec $T_j$.

The required output is an enveloping rectangle $R$ subdivided by horizontal and vertical line segments into $n$ non-overlapping rectangles labeled $B_1,B_2,,B_i,B_n$ such that the following constraints are satisfied:

*i.* $w_i\ h_i = A_i$ *for all* $B_i \in S_1$
ii. $r_i \le h_i / w_i \le s_i$ *for all* $B_i \in S_2$
*iii.* $r_i \le h_i / w_i \le s_i$ *or* $1/s_i \le h_i/w_i \le 1/r_i$ *if* $B_i \in S_2$ *for all i*
*iv.* $x_i \ge w_i$ *and* $y_i \ge h_i$ , $1 \le i \le n$, *for all i, where* $x_i$ *and* $y_i$ *are the dimensions of the basic rectangle i, i.e. every rectangle i is large enough to accommodate module i.*
*v. The overall area A=HW is minimized*
*vi.* $p \le H/W \le q$, *where H and W are the height and width of the enveloping rectangle R.*
*vii. The number of routed buses is maximized, where a bus is formed by bundling together the nets that start and end at the same locations*
*viii. The number of buses missing their timing specification is minimized*

*The positions and sizes of all channel blocks for the un-routed buses and all repeater blocks for buses that are missing their timing specification are also required.*

### 3.2 Algorithm

The general flow of our algorithm is given in Fig 2. We use Polish-expression-based slicing-tree method and soft-module area optimization as given in [10]. Our approach is based on two key observations. Firstly, we observe that slicability does not impose an undue constraint on the floorplanning, secondly, it makes sense to restrict a number of routing options to only few and we restrict them to I, L, Z, C type of routes.

Our algorithm consists of four key steps: block planning, bus planning, solution perturbation and evaluation, and channel/repeater insertion. First, we bundle the wires together to create buses. Some buses are logical (such as data and control buses), while others are just simple bundles of wires. For each block topology (floorplan) generated by simulated annealing (SA), possible route options are generated for all busses. Buses that are critical or lie in congested area are assigned more routing options. A MILP algorithm is then used at every SA iteration step to determine the exact location of each bus. A number of unrouted busses is used to evaluate a cost of a solution. If at the end of SA process, a solution with all busses being routed has not been generated, we either go back and re-floorplan (another run of SA algorithm – not shown on Fig.2), or go ahead to channel insertion. The decision is entirely based on routability. Timing analysis uncovers how many buses need repeater blocks. Channels and repeaters are inserted to make the floorplan feasible.

The bussing concept offers significantly reduced complexity of the route-planning algorithm because we consider a much smaller number of net groups. The reduced complexity enabled us to use routing topology enumeration and MILP based route-planning algorithm to plan the bus route globally.

During bus planning, we consider different routing options for each bus/net and find a globally optimized solution to maximize routability. The cost function for evaluating a solution is given as:

$$Cost=A + a*R + b*T,$$

where $A$ = area, $R$ = percentage of unrouted buses, $T$ = percentage of buses missing timing specification, and constants $a$ and $b$ are used to balance/unbalance the cost function. The channel/repeater insertion stage ensures that if we have un-routed nets or nets failing timing constraints, channel blocks and repeater blocks are inserted in the floorplan to achieve a feasible solution with minimal area increase.

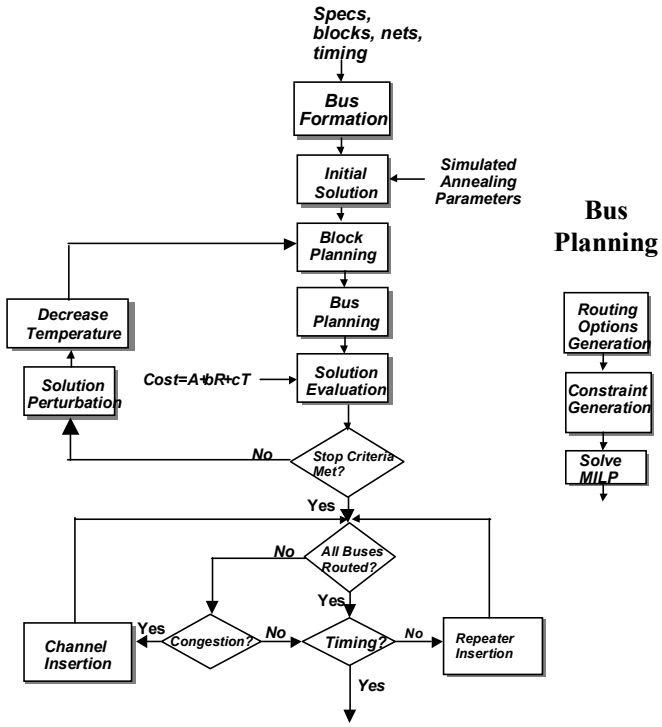


**Figure 2. Overall Algorithm**

# 4. BUS PLANNING

## 4.1 Generating Routing Options for Busses

A route generator has the knowledge of all available metal layers such as direction requirements, the minimum spacing requirement and the minimum width requirement. A routing option can occupy any valid combination of metal layers provided that it satisfies design rules. The generator uses a grid-less approach to allow a large number of options for each bus.

The generator is also sensitive to the SA Temperature. To balance the routabilty and the time complexity, the generator generates only a few routing options for each bus at a high temperature when the solution space for the floorplanning is too broad, and generates a large amount of options at a low temperature when the solution space is narrow.

For a two-terminal bus, a routing option can be any rectilinear shape. For a multi-terminal bus, a routing option can be any Steiner-tree shape. In the following, we shall focus on two-terminal busses to illustrate how routing options are generated for a bus. For a two-terminal bus, an *I-shaped (L-shaped, Z-shaped)* route is the shortest Manhattan distance route with at most *0 (1, 2)* bends (Figure 3). Besides having the smallest length, these shapes of routes introduce the minimum number of vias when different metal layers are used for different directions. Therefore, the generator produces only these shapes of routing options at higher temperature levels. However, at lower temperatures, routes are allowed to go out of the bounding box, as shown in Figure 4, to ensure routabilty.

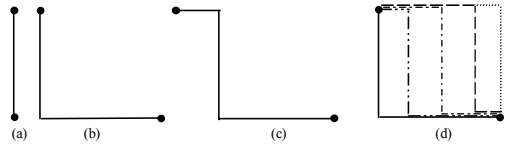


**Figure 3.a) I-shaped, L-shaped , and Z-shaped Interconnections. b) Some of the possible interconnections between two points**

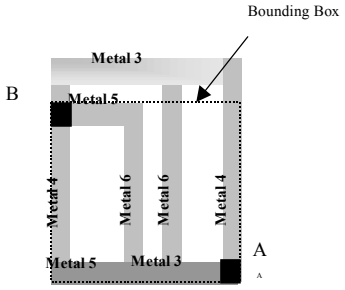


**Figure 4. Four possible routing options for a bus**

## 4.2 Mixed Integer-Linear Programming Formulation for Bus Planning

Given a set of routing options for each bus $\{B_1, B_2, B_n\}$ on a fixed floorplan, we solve the bus planning problem using a mixed integer linear programming approach. We say two routes for two different busses are *incompatible* if they have segments overlapping or too close to each other on the same layer. We construct an *incompatibility graph G=(V, E)* where $V$ consists of a set of vertices representing routing options and $E$ consists of a set of edges, where each edge connects a pair of incompatible routes. Furthermore, $V$ is partitioned into $n$ partitions, each of which corresponds to the set of routing options for a single bus, as shown in Figure 5. The bus-planning problem is then reduced to a version of the independent set problem, i.e., to find a set of vertices S in V of size $n$, such that (1) no two vertices are from the same partition, and (2) no two vertices are connected by an edge.

We solve this independent set problem by mixed integer linear programming. We assign a (0,1) variable to each vertex $v_i$ in $V$, and a real variable $e_k$ to each edge in $E$. Then the following constraints must be met.

Overlapping Constraints:

$e_k \geq v_i + v_j - 1$

$e_k \geq 0,$

*for every pair of vertices* $v_i$, $v_j$ *connected by an edge*

$$e_k = \left\{ \begin{array}{c} = 0 \text{ iff at most one of the options } v_i \text{ or } v_j \text{ is selected} \\ > 0 \text{ iff both } v_i \text{ and } v_j \text{ are selected} \end{array} \right\}$$

Partitioning Constraints:

$$\sum_{v_i \in P} v_i = 1$$

for each partition P.

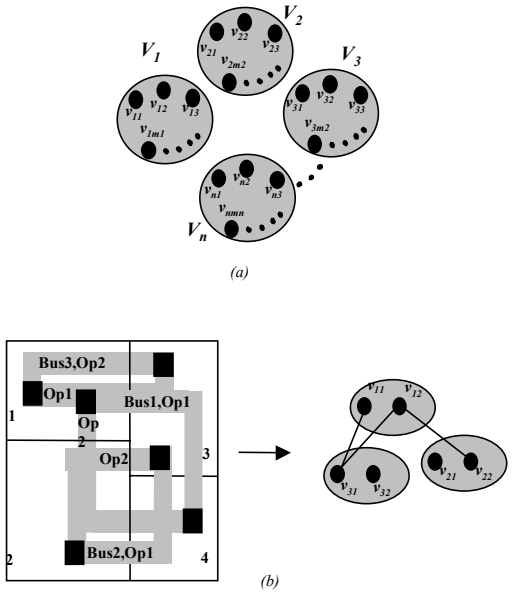


**Figure 5.** ***a)*** **Vertices in the incompatibility graph,** ***b)*** **Construction of incompatibility graph**

The objective is to minimize the number of overlapping routes and the objective function is given by

$$obj = \sum_{e_k \in E} e_k$$

The minimization forces an $e_k$, in the above expression, to *0* if at most one of the vertices $v_i$ or $v_j$ connected by $e_k$ is selected. Otherwise, $e_k$ must be 1 because $v_i$ and $v_j$ are integer variables. Therefore, the number of $e_k$'s remaining to be 1 is the number of overlapping routes. By declaring $e_k$'s to be real, we reduce the number of integer variables (the same effect as forcing $e_k$ to be either *1* or *0*), and improve the efficiency of MILP. We solve MILP using the CPLEX package. The percentage of unrouted busses is then used as one of the components of the cost function of a floorplanning solution.

# 5. CHANNEL AND BUFFER INSERTION

For buses missing their timing specification, repeaters have to be inserted. Moreover, a channel $c_{ij}$ is also inserted when it is impossible to route a bus over the blocks. Channel insertion is similar to the buffer insertion process. *Allowable* locations for the channel/repeater insertion can only be at the *cut* in order to maintain slicability. For every bus $b_{ij}$, that requires a channel/repeater, a set $S=\{i_1, i_2, i_3, \ldots i_n\}$ of possible repeater insertion points (the cuts in the path of $b_{ij}$) is determined. Our algorithm selects a *subset* $S'=\{j_1, j_2, \ldots, j_m\} \in S$, such that $|j_{k+1} - j_k| < L$, where $L$ is the longest span allowed without a repeater.

*Insert (B1, B2, A)* inserts a Buffer/Channel of area *A* between blocks B1and B2 where

- *B1* and *B2* share a cut completely or partially
- *B1* is to the *left / below B2*
- The inserted channel will act as a *flexible* block *N* with a fixed area.
- Post-order traversal of the new tree will give the minimum area realization of the floorplan including the channel

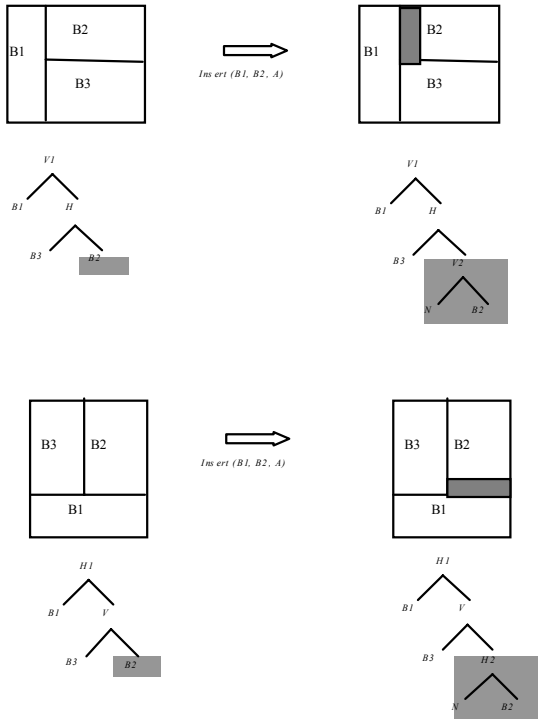


**Figure 6. Operation on the slicing tree for *Case 1***

There are only three cases possible, as listed below.

- The common edge *completely* contains one side of *both B1* and *B2*.
- The common edge *completely* contains one side of *either B1* or *B2*
- The common edge *partially* contains *one* side of *B1* and *B2*

We discuss only one of the above cases, *case 1*.

**Case 1**. The common edge completely contains one side of ***B2*** (Figure 6.)

*Operation on the Slicing Tree:*

- Search for*B2* in Slicing Tree

- Replace *B2* with *B1's* parent cut operator
- Put *B2* as the right child and the new channel block as the left child of the new node.

## 6. EXPERIMENTAL RESULTS

We have tested our approach on MCNC benchmarks. All the experiments were carried out on a HP PA7200 workstation with a 120Mhz processor. We consider only two terminal nets. Note that the timing specification for each net in a bus might be different. For our experiments, however, the timing specification of a bus is equal to that of the most critical net among all the nets in that bus.

In table 1 we compare the net-based approach and bus-based approach. For *both* the net-based and bus-based approaches we used our Integrated Floorplanner with three routing options and two metal layers. However, for the net-based approach, we considered individual nets and for the bus-based approach nets were bundled together to form buses. For both approaches, the reported results are the best of ten runs of SA. As listed in table1, when we use the bus-based approach we have an average of 52.5%, 38.8% and 30% improvements, over the Net-based Approach, in CPU time, Buffer Area and Channel Area, respectively.

Table 2 compares the traditional approach of floorplan-then-route with our Integrated Floorplanner. For all the experiments in this set the number of routing options is 3 and the number of available metal layers is 2. For both approaches and for all examples, a feasible solution was generated by the insertion of channels and repeaters. Since no currently available floorplan-then-route floorplanner allocates the routing space, we use the Wong and Liu algorithm [10] with the objective function equal to area and total wire-length for the floorplan-then-route approach. Then we use our routing algorithm for bus routing on the final floorplan. We have achieved an average of 31% improvement in routabilty when we use Integrated Floorplanning Approach. This result shows the significance of optimizing the routability "while" floorplanning. Moreover, with the integrated approach, an average of 56.4% improvement in Buffer Area is also obtained. Furthermore, we have an average of 12.48% improvement in Overall Area (includes the channel and buffer areas) with the integrated approach. As listed in table 2 the integrated approach takes much more CPU time as compared to the floorplan-then-route approach. This was expected because of many runs of routing algorithm in Integrated Floorplanner during the iterations of simulated annealing vs. only one run in the floorplan-then-route approach. Overall, within tolerable time limits our Integrated Floorplanner performs significantly better than the conventional approach, emphasizing the advantage of incorporating the bus optimization (routability and timing) within the iterative optimization of the floorplan itself.

Table 3 presents the effect of the number of available routing options on the routability and CPU time. It can be observed that with the additional number of routing options 100% routabilty was achieved. Table 4 shows the effect of increasing the number of available metal layers, using the same number of available routing options (here we use 3 options). The results show a significant improvement in terms of routability. Please observe that adding more routing options has more influence on routability of a floorplan than increasing a number of available metal layers. A floorplan of ami33 is shown in Figure 7.

## 7. CONCLUSIONS

An integrated approach to interconnect planning and floorplanning problems for bus-based designs is presneted. We take advantage of the fact that a smaller number of buses with restricted topologies is amenable to a mixed integer-linear solver that produces an exact solution. Our algorithm not only finds an exact location of each bus but also automatically inserts channels and repeater blocks to meet congestion and timing constraints. The experimental results are encouraging and indicate that an integrated approach not only yields better results but is also practical. Our algorithm allows design teams to converge their floorplan very quickly by simultaneously taking into account area, timing and congestion constraints. Since the most CPU intensive operation is the solution of the MILP, we are considering new heuristic approaches to solve the bus routing problem. Similar integrated approach can be developed for non-slicing floorplans using any of recently introduced representations like sequence pair [4] or O-tree [11]. We are integrating this approach with ELF-SP floorplanner [12] that uses the Evolutionary Algorithm and Lagrangian Relaxation for sequence-pair represented floorplans.

## REFERENCES


[1] H. Chen, H. Zhou, F.Y. Young, D.F. Wong, H. Yang, and N. Sherwani, " Integrated Floorplanning and Interconnect Planning", IEEE International Conference on CAD, 1999.

[2] T. Chen and M. K. H. Fan, "On Convex Formulation of the Floorplan Area Minimization Problem", International Symposium on Physical Design, pp. 124-128, April 1998.

[3] T. S. Moh, T.S. Chang, and S. L. Hakimi, "Globally Optimal Floorplanning for a Layout Problem", IEEE Transactions on Circuits and Systems: Fundamental Theory and Applications, Vol. 43, pp. 713-720, Sep 1996.

[4] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-packing-Based Module Placement", Proc. of International Conference on CAD, pp. 472-479, 1995.

[5] S. Nakatake, K. Sakanushi, Y. Kajitani and M. Kawatika, "The Channeled BSG: A universal floorplanner for Simultaneous Place/Route with IC applications", *Proc. IEEE International Conference on CAD,* pp.418-425, 1998.

[6] R. H. J. M. Otten, "Efficient Floorplan Optimization," *IEEE International Conference on CAD,* pp. 499-502, 1983.

[7] Sherwani, "Algorithms for VLSI Physical Design Automation", *Kluwer Academic Publishers*, 1998.

[8] Peichan Pan and C. L. Liu, "Area Minimization of General Floorplans", *Proc. of International Conference on Computer Aided Design*, pp 606-609, 1992.

[9] P. Pan and C. L. Liu, "Area Minimization for general floorplans," Proc. *IEEE International Conference on Computer Aided Design,* pp. 606-609, 1993.

[10] D.F. Wong and C. L. Liu, "A new Algorithm for Floorplan Design," *Proc. of the Design Automation Conference,* 1986.

[11] N. P. Guo, C-K Chen and T. Yoshimura, "An O-tree representation of non-slicing floorplan and its application," *Proc.of DAC*, pp. 268-273, June 1999.

[12] M. Chrzanowska-Jeske, G. Greenwood, B. Wang, "Combing Evolution Strategies with Lagrangian Relaxation for Constructing Non-slicing VLSI Floorplans with Soft Modules," Congress on Evolutionary Computing, 2002.

**Table 1: Comparison Of Net-Based And Bus-Based Approaches**

| Data | n | Net Based Approach | | | | Bus Based Approach | | | | % Improvement | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | nets | CPU Time *(t)* *(sec)* | Buffer Area *(B)* *($x10^3 um^2$)* | Channel Area *(C)* *($x10^3 um^2$)* | Buses | CPU Time *(t)* *(sec)* | Buffer Area (B) ($x10^3um^2$) | Channel Area (C) ($x10^3um^2$) | t | B | C |
| apte | 9 | 98 | 8273 | 5452 | 7765 | 36 | 3476 | 2496 | 4576 | 58 | 54.1 | 41.1 |
| xerox | 10 | 203 | 12345 | 1675 | 2134 | 98 | 6546 | 1123 | 1435 | 46.9 | 32.9 | 32.8 |
| hp | 11 | 83 | 4124 | 912 | 1324 | 37 | 1456 | 634 | 956 | 64.3 | 30.4 | 27.7 |
| ami33 | 33 | 123 | 10234 | 52 | 68 | 53 | 5986 | 32 | 46 | 41.7 | 38.6 | 32.8 |
| ami49 | 49 | 408 | 22425 | 176 | 196 | 156 | 11234 | 110 | 162 | 49.9 | 37.4 | 17.2 |

**Table 2 (Experiment Set 2): Performance Of Integrated Floorplanner compared With Traditional Approach [10]**

| Data | *n* | Buses | Floorplan-then-route | | | | Integrated Floorplanning | | | | % Improvement | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | CPU *(t)* *(sec)* | %Routability *(R)* | Timing Vios. (V) | [1]Overall Area (A) ($x10^3um^2$) | CPU *(t)* *(sec)* | %Routability *(R)* | Timing Vios. *(V)* | [1]Overall Area *(A)* *($x10^3 um^2$)* | *R* | *V* | *A* |
| apte | 9 | 36 | 7 | 55.7 | 12 | 61692.38 | 3476 | 80.9 | 2 | 55468.96 | 45.2 | 83.3 | 13.1 |
| xerox | 10 | 98 | 12 | 64.5 | 9 | 24626.64 | 1472 | 79.8 | 6 | 22210.5 | 23.7 | 33.3 | 10.1 |
| hp | 11 | 37 | 16 | 65 | 21 | 14210.76 | 1425 | 76 | 9 | 12576.5 | 16.6 | 57.1 | 11.4 |
| ami33 | 33 | 53 | 435 | 58.7 | 17 | 1626.78 | 6425 | 78.6 | 6 | 1425.7 | 33.4 | 64.7 | 12.4 |
| ami49 | 49 | 156 | 913 | 61.3 | 27 | 4692.68 | 8265 | 81.6 | 15 | 3976.8 | 34.4 | 44.4 | 15.4 |

[1]includes channel and buffer areas

**Table 3 : Effect of no. of available routing options on routability and CPU time**

| Data | No. of routing options | Routability | CPUtime |
|---|---|---|---|
| apte | 3 | 80.9 | 3476 |
| | 5 | 88.4 | 6328 |
| | 7 | 100 | 8452 |
| ami33 | 3 | 78.6 | 6425 |
| | 5 | 91.2 | 7896 |
| | 7 | 100 | 10120 |

**Table 4: Effect of no. of available routing layers**

| Data | No. of Metal Layers | Routability |
|---|---|---|
| apte | 2 | 89.1 |
| | 4 | 94.3 |
| | 6 | 97.8 |
| ami33 | 2 | 78.6 |
| | 4 | 91.2 |
| | 6 | 96.3 |

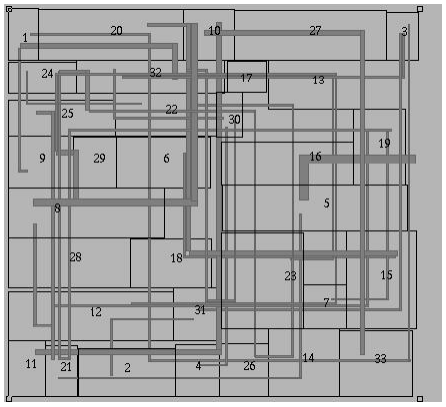

**Figure 7: Floorplan for ami33**