New Results for Floorplanning with Interconnect Planning

Yuchun Ma, Song Chen, Shou Zhou, Xianlong Hong, Sheqin Dong, Yici Cai, C. K. Cheng

Dept. Of Computer Science and Technology
Tsinghua University, Beijing, China
Outline

- Introduction
- Research work on floorplanning
- New Results on floorplanning
  - Sub CBL in rectilinear blocks packing
  - Interconnect-driven floorplanning with buffer insertion
  - Buffer insertion based on dead space redistribution
- Further Work
Introduction

- Floorplanning and BBL placement has received much more attention recently:
  - Floorplan results affect chip performance
  - Performance driven and Timing Driven
  - Hierarchical design methodology
  - IP blocks are widely used in SOC

- For random optimization approaches the floorplanning representation is the key technique.
Existing Representations

- Binary tree & Polish expression for slicing structure (D.F. Wong, 1986)
- Sequential Pair (SP) (Murata, 1995)
- Bounded-Slicing line Grid (BSG) (Nakatake, 1996)
- O-Tree (P. N. Guo & C. K. Cheng, 1999)
- B* Tree (Su-Wei Wu, 2000)
- Corner Block List (X.L. Hong, 2000)
CBL Representation

♦ For each block deletion
  ♦ block name $S_i$
  ♦ block orientation $L_i$
  ♦ number of attached T-junction $T_i$

♦ At the end of deletion
  ♦ $\{S_n, S_{n-1}, \ldots, S_1\}$
  ♦ $\{L_n, L_{n-1}, \ldots, L_2\}$
  ♦ $\{T_n, T_{n-1}, \ldots, T_2\}$
Interconnect-driven Floorplanning

- In deep submicron design, interconnect delay and routability have become the dominant factor:
  - The VLSI circuits are scaled into nanometer dimensions and operate in gigahertz frequencies
  - To ensure the timing closure of design, interconnects must be considered as early as possible in the design flow
Buffer Insertion

Buffer insertion has shown to be an effective approach to achieve timing closure.

- As transistor count and chip dimension get larger and larger, more and more buffers are expected to be needed for high performance; 700K buffers will be inserted on a single chip in the 70nm technology.
- They cannot be placed over the existing circuit blocks.
- Placing a large number of buffers between circuit blocks could significantly impact the chip floorplan.
- Therefore, it is necessary to start buffer planning as early as possible.
Outline

- Introduction

- *Research work on floorplanning*
  - New Results on floorplanning
    - Sub CBL in rectilinear blocks packing
    - Interconnect-driven floorplanning with buffer insertion
    - Buffer insertion based on dead space redistribution

- Further Work
the Extension of CBL

- The floorplanning with fix topology
  - CBL representation of the floorplan is independent of the block widths and heights, so we can use corner block list to optimize the blocks with multiple configurations of widths and heights.

- ECBL---- An extended CBL (ISPD 2000)
  - Adding enough number of dummy blocks in CBL will make its solution space including the optimal solution

- The compact approach (ASPDAC 2002)
  - Adding dummy blocks dynamically & intelligently without increase of complexity.
## The Best Results Comparison

<table>
<thead>
<tr>
<th>circuit</th>
<th>CBL with compaction</th>
<th>ECBL</th>
<th>CBL</th>
<th>O-tree</th>
<th>Cluster (size = 4 blocks)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>area/time</td>
<td>area/time</td>
<td>area/time</td>
<td>area/time</td>
<td>area/time</td>
</tr>
<tr>
<td>Ami33</td>
<td>1.191/62</td>
<td>1.192/73</td>
<td>1.201/36</td>
<td>1.242/119</td>
<td>1.207/603.4</td>
</tr>
<tr>
<td>Ami49</td>
<td>36.62/101</td>
<td>36.70/117</td>
<td>38.58/65</td>
<td>37.73/526</td>
<td>37.69/1861.7</td>
</tr>
</tbody>
</table>
## Floorplan Results

<table>
<thead>
<tr>
<th>Examples</th>
<th>Area (mm²)</th>
<th>usage</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ami33</td>
<td>1.176</td>
<td>0.983</td>
<td>87</td>
</tr>
<tr>
<td>Xerox</td>
<td>19.75</td>
<td>0.979</td>
<td>76</td>
</tr>
<tr>
<td>Hp</td>
<td>62.93</td>
<td>0.980</td>
<td>75</td>
</tr>
<tr>
<td>Apte</td>
<td>46.63</td>
<td>0.966</td>
<td>78</td>
</tr>
<tr>
<td>Ami49</td>
<td>36.09</td>
<td>0.982</td>
<td>179</td>
</tr>
</tbody>
</table>

The floorplan of ami49. The area is 35.99 mm² and the area usage is 98.4%; running time is 57sec.
Boundary Constraints

- Floorplanning with Boundary Constraint (ASPDAC2001)
  - Limit some specified blocks to be adjacent to I/O pads for external communication.
  - Proved a Necessary-sufficient condition for boundary blocks.
  - Check the boundary condition and fix the CBL sequence to satisfy the constraints as much as possible.
  - Use penalty function to punish CBL, which violates the boundary constraints.
Packing Results

The floorplan of P_65

Area Ratio: 0.98238; wirelength: 74.88mm running time: 238 seconds

2002-9-6 Tsinghua University
Abutment and L/T-shaped blocks

- Floorplanning with Abutment Constraint (DAC2001)
  - Want to have the logic blocks in a pipeline of a circuit to abut one after another to favor the transmission of data.
  - The Abutment between Blocks
    - Horizontal abutment
    - Vertical abutment

- Floorplanning with L/T-shaped blocks (DAC2001)
  - Some IP blocks have L/T-shape in SOC.
  - The partition of L/T-shaped blocks
    - Abutment
    - Alignment
Experimental Results

A result packing of ami49 with a horizontal chain and a vertical chain was packed in 62 seconds with the dead space of 2.57%.

A packing result with 7 L-shaped blocks and 2 T-shaped blocks was packed in 36 seconds with the dead space of 2.05%.
Less-Flexibility-First heuristic

- Less-Flexibility-First Heuristic based deterministic floorplanning (ASP-DAC2001)
- Use mason’s principle of the packing.
- Define flexibility of empty space, rectangular block and between two blocks.
- Use LFF heuristic to place blocks.
- Complexity is $O(n^5 \log n)$.
- Very fast and better results in chip area usage.
Solution Space Smooth based floorplanning

- Floorplanning is an optimization problem with multiple local minimal solutions.
- Simulated annealing needs to spend time to select and adjust annealing parameters.
- SSS: build a series optimization problems, which are simpler than original one, have the approximate outline and the same valley with the original one.
- Much better results and robust.
The minimum solution in original space

The smoothed solution space 1
The smoothed solution space 2
The smoothed solution space n
The original solution space

The initial search point in original space

The minimum solution in original space

An example of solution space smoothing: the minimum solution of solution space $i$ will be the initial starting point in solution space $i+1$
Outline

- Introduction
- Research work on floorplanning
- New Results on floorplanning
  - Sub CBL in rectilinear blocks packing
  - Interconnect-driven floorplanning with buffer insertion
  - Buffer insertion based on dead space redistribution
- Further Work
T-cuts in the packing

- The rooms of two blocks are separated by T-cuts.

- 4 kinds of T-cuts

- The necessary and sufficient conditions for T-cuts.

(a) 0 (b) 90 (c) 180 (d) 270

Four different types of T-cuts
Sub CBL

- The rectilinear blocks in placement can be regarded as the sub-placement.

- We use the sub_CBL \((S_{\text{sub}}, B_{\text{align}}, T_{\text{cut}})\):
  - \(S_{\text{sub}}\) is defined similarly to the name list \(S\) in CBL.
  - \(B_{\text{align}}\) records the abutted block which is aligned with the corresponding block in list \(S_{\text{sub}}\).
  - \(T_{\text{cut}}\) records the type of the \(T_{\text{cut}}\) between two neighbor blocks.

\[\begin{align*}
S_{\text{sub}} &= 1 \ 2 \ 3 \ E \ 4 \\
T_{\text{cut}} &= 270, \ 90, \ X, \ 90 \\
B_{\text{align}} &= 1,2,X,3
\end{align*}\]

(a) the original shape

(b) The original shape and the cut type.
Embedding the sub CBL

- **Block sequence**
  - Other circuit blocks are between the sub blocks
  - The sequence in $S_{\text{sub}}$ keeps unchanged

- **Block alignment**
  - Following the T-cuts described in sub CBL

- **Concave blocks**
  - Concave constraints
  - Fix the violations
    - By exchanging the violated blocks with the dummy blocks afterward
Packing Results

The result of ami33 with 6 rectilinear blocks. The area usage of the final packing is 93.4% and the running time is 60 seconds.

The result of ami49 with 5 rectilinear blocks. The dead space is only 9.8% while the running time is 120 seconds.
Outline

- Introduction
- Research work on floorplanning

New Results on floorplanning

- Sub CBL in rectilinear blocks packing
- Interconnect-driven floorplanning with buffer insertion
- Buffer insertion based on dead space redistribution

Further Work
Previous Works on Buffer insertion

- Feasible Region ---- J. Cong, T. Kong et al.
  - The feasible region for a buffer is the maximum region where the buffer can be located such that the target delay of the net can be satisfied. They make use of dead space and channel region between circuit blocks to insert buffers.

- Independent Feasible Region and Congestion-driven buffer insertion --- P. Sarkar, C. K. Koh

- Net Flow algorithm----- Tang and Wong
- multi-commodity flow-based approach ----- F. F. Dragan et al
  - They assume that buffers be allowed to be inserted inside macro blocks and their approach will distribute buffer sites all over the layout.

- Sham et al [13] proposed a routability driven floorplanner, which can estimate buffer usage and buffer resource for the congestion constraint.
Problem Definition

- Given the timing constraints on each net, we should give a floorplanning with buffers inserted to meet the timing constraints. Also we should find the number and locations of buffers at the same time.

  - produce the optimal floorplan such that the floorplan area and wire length are minimized and the buffers can be inserted in the dead spaces as much as possible.
  - the insertion of buffers should be in the dead spaces between circuit blocks.
  - Buffer insertion increases the congestion of the floorplanning
Sarkar and Koh gives the notion of independent feasible regions (IFR)

- Each driver/buffer is modeled as a switch-level RC circuit and the Elmore delay formula is used for delay computations.
- Since the feasible region will be reduced by the circuit blocks, the feasible region for buffer insertion in the packing is a very complex polygon, normally concave polygon.
The Dead space in CBL packing

Based on CBL, we propose the algorithm to obtain the dead space blocks in the floorplanning while doing the packing.

The number of the dead space blocks (NDS) in CBL packing should be less than \( n - 1 \), where \( n \) is the number of the blocks.

The example packing process
The computation of possible buffer insertion sites

The computation of buffer insertion sites is the most difficult and time-consuming part when doing the buffer planning.

Instead of computing the size of the dead space in each grid in the packing, we compute the intersection between the dead space blocks and the FRs in a 2-step method.

- the first step is to compute the intersected blocks between dead space blocks and the bounding box of the source and sink;
- the second step is to compute the overlapping between the result blocks in the first step and the region between two parallel lines which are the two edges of the FR.
Possible buffer insertion sites are between grid 4 and grid 17
Buffer planning

The budget of buffer insertion

- Suppose that the probability of buffer insertion at each grid is equal.
- The capacity of a grid is $R (R = \text{Area of the grid}/\text{Area of the buffer})$
- If the probability of the grid is larger than the capacity, we think the buffers inserted will be too crowded thus we should take some measure to control it.

We divide the annealing process into two phases: timing optimization phase and buffer insertion phase.

- In the timing optimization phase, we estimate the buffer insertion by probability budget;
- In the buffer insertion phase, we do the buffer allocation by the heuristic methods
Conclusion

- The buffer allocation is handled as an integral part in the floorplanning process.
- Not necessarily to scan the whole packing to find the dead spaces, we can partition the dead space into blocks while doing the packing.
- Instead of computing the size of the dead space in each grid, we compute the intersection between the dead space blocks and the FRs in a 2-step method.
Conclusion

- Since our method can give the range of the possible buffer insertion sites independent of the sizes of the grids, we give a probabilistic method to budget the buffer insertion.
- Besides, we divide the annealing process into two phases: timing optimization phase and buffer insertion phase.
- The experiments prove the effectiveness of our approach.
<table>
<thead>
<tr>
<th></th>
<th>Area(mm²)</th>
<th>Wire(mm)</th>
<th>#Inserted B/#B</th>
<th>#meet</th>
<th>#violation</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>F1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>F2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Xerox_1</strong></td>
<td>84.57</td>
<td>85.41</td>
<td>1343</td>
<td>1327</td>
<td>189/395</td>
<td>214/303</td>
</tr>
<tr>
<td><strong>Xerox_c</strong></td>
<td>91.16</td>
<td>86.32</td>
<td>1424</td>
<td>1439</td>
<td>70/345</td>
<td>84/319</td>
</tr>
<tr>
<td><strong>Ami33_1</strong></td>
<td>30.86</td>
<td>31.15</td>
<td>431.5</td>
<td>461.2</td>
<td>117/524</td>
<td>192/465</td>
</tr>
<tr>
<td><strong>Ami33_c</strong></td>
<td>34.04</td>
<td>36.07</td>
<td>515.6</td>
<td>503.9</td>
<td>216/501</td>
<td>172/307</td>
</tr>
<tr>
<td><strong>Ami49_1</strong></td>
<td>156.67</td>
<td>146.6</td>
<td>2922</td>
<td>2920</td>
<td>315/582</td>
<td>244/568</td>
</tr>
<tr>
<td><strong>Ami49_c</strong></td>
<td>175.27</td>
<td>183.15</td>
<td>3471</td>
<td>2940</td>
<td>198/511</td>
<td>363/546</td>
</tr>
<tr>
<td><strong>Apte_1</strong></td>
<td>48.15</td>
<td>48.14</td>
<td>484.3</td>
<td>459.5</td>
<td>11/111</td>
<td>44/107</td>
</tr>
<tr>
<td><strong>Apte_c</strong></td>
<td>49.55</td>
<td>50.08</td>
<td>520.2</td>
<td>478.8</td>
<td>21/154</td>
<td>53/89</td>
</tr>
<tr>
<td><strong>Hp_1</strong></td>
<td>38.61</td>
<td>38.86</td>
<td>424.4</td>
<td>392.2</td>
<td>23/350</td>
<td>40/106</td>
</tr>
<tr>
<td><strong>Hp_c</strong></td>
<td>40.64</td>
<td>40.61</td>
<td>485.9</td>
<td>486.9</td>
<td>53/416</td>
<td>69/106</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>+0.2%</td>
<td>-3.1%</td>
<td>--</td>
<td>--</td>
<td>+76%</td>
<td>-49%</td>
</tr>
</tbody>
</table>

2002-9-6  
Tsinghua University
Outline

- Introduction
- Research work on floorplanning

New Results on floorplanning
- Sub CBL in rectilinear blocks packing
- Interconnect-driven floorplanning with buffer insertion
- Buffer insertion based on dead space redistribution

- Further Work
Dead Space Redistribution

- Associated with circuit blocks under topological representations, the dead space can be redistributed by freely moving some circuit blocks within their rooms, while the total area and the topology of the placement keep unchanged.
- All buffers can be moved anywhere within their respective IFRs without violating the timing constraints.
Dead Space in rooms

- In the rooms of blocks, the dead space can be attached to the blocks. And the dead spaces are redistributed while the blocks are moved within their rooms.

- In the empty rooms, the dead space is moveless.
Algorithm 1 Buffer Planning

1. Build the tile data structure for all the dead-spaces.
2. Compute IFR for each buffer.
3. Compute the set of candidate tiles for each buffer.
4. Construct a bipartite graph $G (V, E)$, $V = V_1 \cup V_2$, where $V_1$ represents buffers and $V_2$ represents tiles, $E = \{(v_1, v_2), v_1 \in V_1, v_2 \in V_2, v_1$ can be inserted into $v_2\}$.
5. Construct an s-t graph from $G$.
6. Find the max flow from s to t and determine the location for each buffer.
Conclusion

☐ The dead space redistribution can be achieved by redistributing the Attached Dead-Spaces in the placement, while the topology and total area of the placement keep unchanged.

☐ The nets which satisfy the delay constraints increase 12.4%. The increment of the number of the nets that satisfy delay constraints is 9% on an average.
## Results

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Buffer Planning</th>
<th>Optimization</th>
<th>$N_{imp}$</th>
<th>$R_{imp}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>met</td>
<td>buffer</td>
<td>Time(s)</td>
<td>met</td>
</tr>
<tr>
<td>Apte</td>
<td>89</td>
<td>83</td>
<td>0.16</td>
<td>100</td>
</tr>
<tr>
<td>Xerox</td>
<td>275</td>
<td>152</td>
<td>0.1</td>
<td>315</td>
</tr>
<tr>
<td>Hp</td>
<td>129</td>
<td>179</td>
<td>0.25</td>
<td>139</td>
</tr>
<tr>
<td>Ami33</td>
<td>235</td>
<td>162</td>
<td>0.08</td>
<td>249</td>
</tr>
<tr>
<td>Ami49</td>
<td>437</td>
<td>236</td>
<td>0.51</td>
<td>457</td>
</tr>
</tbody>
</table>
Further Work

- **Multi_constraints**
  - Solve the different constraints simultaneously
  - Extend the sub CBL to handle other constraints such as distance constraints and reshape of the rectilinear blocks

- **Routability-driven Floorplanning:**
  - Congestion estimation.
  - Noise aware floorplanning
Thank You!

clara99@mails.tsinghua.edu.cn
An Example of CBL

Given CBL

- $S=(1234567)$
- $L=(010011)$
- $T=(1\ 0\ 1\ 0\ 10)$

$L_2=0, T_2=nil; CB=2$
$L_3=1, T_3=1; CB=3$
$L_4=0, T_4=0; CB=4$
$L_5=0, T_5=1; CB=5$
$L_6=1, T_6=0; CB=6$
$L_7=1, T_7=10; CB=7$
Advantages of Corner Block List

- Time complexity: $O(n)$;
- Number of combinations: $O(n!2^{3n-3}/n^{1.5})$.
- CBL takes only $n(3+[\lg n])$ bits to describe.
Concave blocks

Pair block

Suppose that the sub blocks $R_i$ and $B_{align}$ are separated by $T$-cut. The sub block $R_p$ is before block $R_i$ in list $S_{sub}$ and there are empty spaces between $R_p$ and $R_i$ in the packing.

Concave constraints

- If $T_{cut_i} = 0^\circ$, $X_{align} \leq X_{Rp}$ then $R_p$ is a pair block of $R_i$ and $R_p$ is below $R_i$;
- If $T_{cut_i} = 0^\circ$, $H_{Ri} > H_{align}$, and $Y_{Rp} - H_{Rp} \leq Y_{align} - H_{align}$ then $R_p$ is a pair block of $R_i$ and $R_p$ is at the left of $R_i$;
The alignment of concave block

\[ S_{\text{sub}} = (R_1, E, R_2, R_3) \]
\[ T_{\text{cut}} = (X, 0, 0) \]
\[ B_{\text{align}} = (X, R_1, R_2) \]

(a) the packed blocks
(b) fix the violations
(c) modify the X-positions
(d) align the Y-positions

the alignment of concave block