Constraint-Driven Floorplan Repair

Michael D. Moffitt  Artificial Intelligence Laboratory (speaker)
Aaron N. Ng       Advanced Computer Architecture Laboratory
Igor L. Markov    Advanced Computer Architecture Laboratory
Martha E. Pollack Artificial Intelligence Laboratory

University of Michigan
Department of Electrical Engineering and Computer Science
Outline of Talk

Legalization: Motivation and Previous Work
- Correct-by-construction approaches
- Post-placement legalization

The FLOORIST (“Floorplan Assistant”) Algorithm
- Step 1: Creation of constraint graphs
- Step 2: Conflict-Directed Iterative Repair
- Step 3: Translation to Fixed-placement via Emulation

Experimental Setup and Results

Conclusion and Future Work
Why floorplanning?

“Hard macros will revolutionize SoC design”
Enno Wein & Jacques Benkoski, EEDesign, Aug 20, 2004

- Hundreds of predesigned macros
  - Embedded memories, analog circuitry, IP blocks
- Macro placement is usually separate from standard cell placement (done once & never repeated)

Growth in the number of hard macros in SoC designs

Hard macros vs. standard cell area
Why floorplan *legalization*?

Reason #1: Existing packages fail to produce legal floorplans

Reason #2: Legal floorplans susceptible to change
- Resizing of blocks
- Dynamically acquired design constraints
- Minor adjustments from chip architect

Solutions to IBM-HB 09:
- Feng Shui 5.1 [Khatkhate et al., 2004]
- APlace 2.0 [Kahng et al., 2005]
- Capo 9.4 [Roy et al., 2005]
Correct-by-Construction Approaches

- **Guarantee (or at least attempt) legalization at each step in search**

- **mPG** [Chang et al., 2003]
  - Enforces legalization at every level of a cluster hierarchy

- **Capo 9** [Adya et al. 2004, Roy et al., 2006]
  - Performs legalization of subproblems resulting from min-cut placement
  - Legalization failures propagate upwards

- **PolarBear** [Cong et al., 2005]
  - “Pre-legalization”: all subproblems are ensured to be legal
  - Uses a simple row-based block packing scheme
Legalization by Post-processing

- **Feng Shui** [Khatkhate et al., 2004] and **APlace** [Kahng et al., 2004]
  - Postpone legalization until global solution obtained
  - Use greedy Tetris-like algorithm [Hill, 2002] to pack cells

- **Other works in cell-placement**
  - Network flows formulation [Brenner et al., 2004]
  - Diffusion-based approach [Ren et al., 2005]
  - Do not generally extend to macros

- **Limited work in floorplanning using sequence pairs** [Nag et al, 1999] and traditional constraint-graphs [Cong et al., 2006]
  - Remove all overlaps initially
  - Iteratively “squeeze” floorplan into enclosing space
  - Do not encode violations or extend to other constraints
FLOORIST ("Floorplan Assistant")

- Begins with a coarse, global floorplan
  - May have been produced by a chip architect
  - May have been produced by a global floorplanner

- Constructs a pair of constraint graphs, except...
  - Violated non-overlap constraints are explicitly encoded
  - Does not correspond to a feasible layout

- Performs a greedy, conflict-directed iterative repair
  - Uses constraint graphs to determine possible pairwise relationships between overlapping modules
  - Extracts explanations for overlaps, removing culprits

- Translates layout back to fixed-placement floorplan
  - Attempts to emulate initial layout as closely as possible
FLOORIST ("Floorplan Assistant")

Step 1: Translation to Constraint Graphs

Step 2: Conflict-Directed Iterative Repair

while (illegal)

Step 3: Translation to Fixed-Placement
Step 1: Translate to Constraint Graphs

[Liao and Wong, 1983] Horizontal and vertical constraint graphs ($G_H$ and $G_V$) containing:
- A node $i$ for each module $M_i$
- An directed, weighted edge $E_{i,j}$ between pairs of nodes (direction depending on the pairwise relationship of modules $M_i$ and $M_j$)

Over past decade, phased out in favor of:
- Sequence pairs [Murata, 1995]
- O-Trees [Pang et al., 2000]
- Many others; see [Yao et al., 2003]

Some advantages of the graph representation:
- Recently shown to be extremely efficient for (optimal) area-minimization [Moffitt & Pollack, ICAPS 2006]
- Can express a wide variety of constraint types [Young et al., 2002]
- (With some work) it can encode an infeasible layout
Step 1: Translate to Constraint Graphs

Relational view:
- $L(1, 2)$ … “1 to the left of 2”
- $A(3, 6)$ … “3 is above 6”

Blocks that overlap?
- $E(4, 5)$ … empty relationship
- $E(2, 3)$ … empty relationship
Step 2: Conflict-Directed Iterative Repair

First Phase: Remove “trivial” overlaps

Can be resolved by sliding Block 3 to the right!

For every $E(i,j) \in S$

If (exists $P(i,j)$ such that consistent($S \cup \{P(i,j)\}$))

$S = S \cup \{P(i,j)\} - \{E(i,j)\}$
Second Phase: Identify culprits and perform “safe” swaps

Identity blocks on critical paths
Check if slack available in alternate graph
If so, swap edge
Repeat first phase
Step 3: Translation to Fixed-Placement

**Goal:** Emulate the initial placement as closely as possible

**Solution:** For each module (in descending order of size),

- Can it be given its original *horizontal* coordinate?
  - If so, add additional edges to enforce this
  - If not, slide it as far left / right as slack allows

- Can it be given its original *vertical* coordinate?
  - If so, add additional edges to enforce this
  - If not, slide it as far up / down as slack allows

- Propagate these adjustments through graphs
Repairing Other Constraint Types

Non-overlap constraints are just one type of violation.

The violation of any “edge-based constraint” can be repaired in the same manner!

Initial Placement  Module Movement  Final Placement
Experimental Setup

IBM-HB Benchmarks [Cong et al., 2004]
- 18 instances, between 550 and 1650 macros
- No cells (pure floorplanning instances)

Three global floorplanners
- Capo 9.4 [Roy et al., 2005] (Note: recent Capo 10 is better)
- Feng Shui 5.1 [Khatkhate et al., 2004] (only global placer)
- APlace 2.01 [Kahng et al., 2005]

Three legalization tools
- Feng Shui 5.1’s legalizer [Khatkhate et al., 2004]
- Parquet 4.5 [Adya and Markov, 2003]
- FLOORIST (our work)

Measure % overlap, HPWL, legalization time
### Experimental Results (Capo 9.4 layouts)

- **Very minor violations**
- **All layouts legal**
- **Negligible wirelength increase**
- **Extremely fast**

<table>
<thead>
<tr>
<th>whitespace</th>
<th>Capo 9.4 Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ovlp (%)</td>
</tr>
<tr>
<td>HB01</td>
<td>0.16</td>
</tr>
<tr>
<td>HB02</td>
<td>0.21</td>
</tr>
<tr>
<td>HB03</td>
<td>0.07</td>
</tr>
<tr>
<td>HB04</td>
<td>0.17</td>
</tr>
<tr>
<td>HB05</td>
<td>0.24</td>
</tr>
<tr>
<td>HB06</td>
<td>0.13</td>
</tr>
<tr>
<td>HB07</td>
<td>0.16</td>
</tr>
<tr>
<td>HB08</td>
<td>0.12</td>
</tr>
<tr>
<td>HB09</td>
<td>0.20</td>
</tr>
<tr>
<td>HB10</td>
<td>0.26</td>
</tr>
<tr>
<td>HB11</td>
<td>0.06</td>
</tr>
<tr>
<td>HB12</td>
<td>0.03</td>
</tr>
<tr>
<td>HB13</td>
<td>0.08</td>
</tr>
<tr>
<td>HB14</td>
<td>0.13</td>
</tr>
<tr>
<td>HB15</td>
<td>0.09</td>
</tr>
<tr>
<td>HB16</td>
<td>0.12</td>
</tr>
<tr>
<td>HB17</td>
<td>0.07</td>
</tr>
<tr>
<td>HB18</td>
<td>0.10</td>
</tr>
<tr>
<td>Avg.</td>
<td>0.12</td>
</tr>
</tbody>
</table>
Experimental Results: Legalization Success

- **Capo 9.4**: 83% Legalizes, 17% Fails to Legalize, 100% Seg. Faults
- **FS 5.1 (glob)**: 43% Legalizes, 57% Fails to Legalize, 100% Seg. Faults
- **APlace 2.0**: 71% Legalizes, 29% Fails to Legalize, 100% Seg. Faults

- **FS 5.1 (legalizer)**: 56% Legalizes, 44% Fails to Legalize, 100% Seg. Faults
- **Parquet 4.5**: 79% Legalizes, 21% Fails to Legalize, 100% Seg. Faults
- **FLOORIST**: 100% Legalizes, 0% Fails to Legalize, 100% Seg. Faults

Legend:
- **Yellow**: Seg. Faults
- **Green**: Legalizes
- **Red**: Fails to Legalize
Experimental Results: Wirelength & Runtime

- % Increase in HPWL
  - Capo's
  - FS's
  - Aplace's

- Runtime (seconds)
  - Capo's
  - FS's
  - Aplace's

... initial solutions

- Parquet
- Floorist
Experimental Results (Pictures)
Ongoing Work

- Heuristics for choosing swaps made during repair phase
  - Currently guided by amount of available slack
  - Could potentially identity most “common” culprits

- Replace emulation phase with explicit optimization
  - Employ an LP-formulation to minimize wirelength

- Create a tighter coupling between FLOORIST and global floorplanning system
  - Use as subroutine within placement algorithm
  - Communicate hierarchical cuts to improve speed of graph operations
Conclusion

FLOORIST: a tool for legalizing layouts when
- Global floorplanner fails to legalize
- Layout undergoes dynamic changes
- Chip architect sketches rough floorplans manually

Performs iterative repair by:
- Identifying conflicts responsible for violated constraints
- Invoking gradual changes that preserve initial quality

By postprocessing APlace layouts, generates floorplans 7% better in wirelength than best known solutions