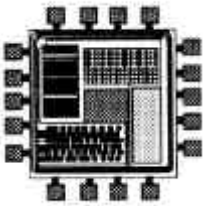
Floorplanning

Outline

- Problem formulation
- ILP approach
- SA approach, general
- Slicing floorplans
- Non-slicing floorplans
- Shaping

Hierarchical Design

- Several blocks after partitioning:

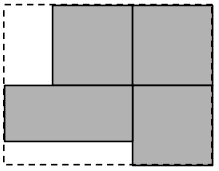


- Need to:
  - Put the blocks together.
  - Design each block.

  Which step to go first?

Hierarchical Design

- How to put the blocks together without knowing their shapes and the positions of the I/O pins?
- If we design the blocks first, those blocks may not be able to form a tight packing.



Floorplanning

The floorplanning problem is to plan the *positions* and *shapes* of the modules at the beginning of the design cycle to optimize the circuit performance:

- chip area
- total wirelength
- delay of critical path
- routability
- others, e.g., noise, heat dissipation, etc.

Floorplanning v.s. Placement

- Both determine block positions to optimize the circuit performance.
- Floorplanning:
  - Details like shapes of blocks, I/O pin positions, etc. are not yet fixed (blocks with flexible shape are called soft blocks).
- Placement:
  - Details like module shapes and I/O pin positions are fixed (blocks with no flexibility in shape are called hard blocks).

Floorplanning Problem

- Input:
  - n Blocks with areas $A_1, \ldots, A_n$
  - Bounds r_i and s_i on the aspect ratio of block B_i
- Output:
  - Coordinates (x_i, y_i), width w_i and height h_i for each block such that $h_i w_i = A_i$ and $r_i \leq h_i/w_i \leq s_i$
- Objective:
  - To optimize the circuit performance.

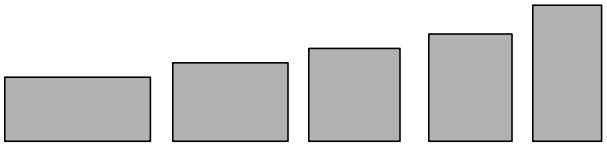
Bounds on Aspect Ratios

If there is no bound on the aspect ratios, we can surely pack very tightly:

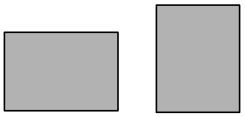
But we don't want to layout blocks as long strips, so we require $r_i \leq h_i/w_i \leq s_i$ for each i.

Bounds on Aspect Ratios

- We can allow several shapes for each block:



- For hard blocks, only the orientations can be changed:



Objective Function

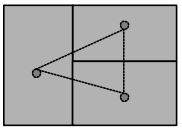
A commonly used objective function is a weighted sum of area and wirelength:

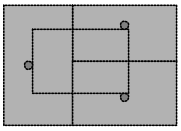
$$\text{cost} = \alpha A + \beta L$$

where A is the total area of the packing, L is the total wirelength, and α and β are constants.

Wirelength Estimation

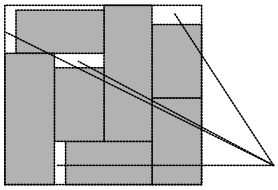
- Exact wirelength of each net is not known until routing is done.
- In floorplanning, even pin positions are not known yet.
- Some possible wirelength estimations:
  - Center-to-center estimation
  - Half-perimeter estimation





Deadspace

- Deadspace is the space that is wasted:



Deadspace

- Minimizing area is the same as minimizing deadspace.
- Deadspace percentage is computed as

$$(A - \Sigma_i A_i) / A$$

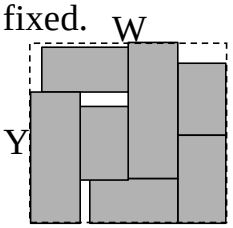
Linear Programming Approach

Integer Linear Program

- A mathematical program such that:
  - The objective is a linear function.
  - All constraints are linear functions.
  - Some variables are real numbers and some are integers, i.e., “mixed integer”.
- It is almost like a linear program, except that some variables are integers.

Problem Formulation

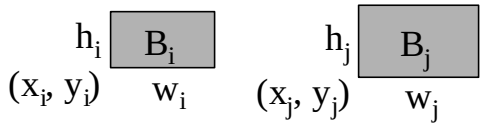
- Minimize the packing area:
  - Assume that one dimension W is fixed.
  - Minimize the other dimension Y.
- Need to have constraints so that blocks do not overlap.
- Associate each block B_i with 4 variables:
  - x_i and y_i: coordinates of its lower left corner.
  - w_i and h_i: width and height.



Non-overlapping Constraints

For two non-overlapping blocks B_i and B_j, at least one of the following four linear constraints must be satisfied:

(1)	$x_i + w_i \le x_j$	if B_i is to the left of B_j
or (2)	$x_i - w_j \ge x_j$	if B_i is to the right of B_j
or (3)	$y_i + h_i \le y_j$	if B_i is below B_j
or (4)	$y_i - h_j \ge y_j$	if B_i is above B_j



Integer Variables

- Use integer (0 or 1) variables x_{ij} and y_{ij}:
  $x_{ij}=0$ and $y_{ij}=0$ if (1) is true.
  $x_{ij}=0$ and $y_{ij}=1$ if (2) is true.
  $x_{ij}=1$ and $y_{ij}=0$ if (3) is true.
  $x_{ij}=1$ and $y_{ij}=1$ if (4) is true.
- Let W and H be upper bounds on the total width and height. Non-overlapping constraints:

$$
\begin{array}{ll}
(1') & x_i + w_i \le x_j + W(x_{ij} + y_{ij}) \\
(2') & x_i - w_j \ge x_j - W(1 + x_{ij} - y_{ij}) \\
(3') & y_i + h_i \le y_j + H(1 - x_{ij} + y_{ij}) \\
(4') & y_i - h_j \ge y_j - H(2 - x_{ij} - y_{ij})
\end{array}
$$

Formulation

$$
\begin{array}{lll}
\text{Min.} & Y & \\
\text{s.t.} & 0 \le x_i, x_i + w_i \le W & 1 \le i \le n \\
 & 0 \le y_i, y_i + h_i \le Y & 1 \le i \le n \\
 & x_i + w_i \le x_j + W(x_{ij} + y_{ij}) & 1 \le i < j \le n \\
 & x_i - w_j \ge x_j - W(1 + x_{ij} - y_{ij}) & 1 \le i < j \le n \\
 & y_i + h_i \le y_j + H(1 - x_{ij} + y_{ij}) & 1 \le i < j \le n \\
 & y_i - h_j \ge y_j - H(2 - x_{ij} - y_{ij}) & 1 \le i < j \le n \\
 & x_{ij} = 0 \text{ or } 1 & 1 \le i < j \le n \\
 & y_{ij} = 0 \text{ or } 1 & 1 \le i < j \le n
\end{array}
$$

Formulation with Hard Blocks

If the blocks can be rotated, use a 0-1 integer variable z_i for each block B_i s.t. $z_i = 0$ if B_i is in the original orientation and $z_i = 1$ if B_i is rotated 90^o.

Min. Y

s.t.	$0 \le x_i, x_i + z_i h_i + (1 - z_i) w_i \le W$	$1 \le i \le n$
	$0 \le y_i, y_i + z_i w_i + (1 - z_i) h_i \le Y$	$1 \le i \le n$
	$x_i + z_i h_i + (1 - z_i) w_i \le x_j + W(x_{ij} + y_{ij})$	$1 \le i < j \le n$
	$x_i - z_j h_j - (1 - z_j) w_j \ge x_j - W(1 + x_{ij} - y_{ij})$	$1 \le i < j \le n$
	$y_i + z_i w_i + (1 - z_i) h_i \le y_j + H(1 - x_{ij} + y_{ij})$	$1 \le i < j \le n$
	$y_i - z_j w_j - (1 - z_j) h_j \ge y_j - H(2 - x_{ij} - y_{ij})$	$1 \le i < j \le n$
	$x_{ij} = 0 \text{ or } 1$	$1 \le i < j \le n$
	$y_{ij} = 0 \text{ or } 1$	$1 \le i < j \le n$

Formulation with Soft Blocks

- If B_i is a soft block, $w_i h_i = A_i$. But this constraint is quadratic!
- Linearized by taking the first two terms of the Taylor expression of $h_i = A_i / w_i$ at w_{imax} (max. width of block B_i).

$$h_i = h_{imin} + \lambda_i (w_{imax} - w_i)$$

where $h_{imin} = A_i / w_{imax}$ and $\lambda_i = A_i / w_{imax}^2$

Formulation with Soft Blocks

- If B_i is soft and B_j is hard:

$$
\begin{aligned}
&(1) \quad x_i + w_i \le x_j + W(x_{ij} + y_{ij}) \\
&(2) \quad x_i - w_j \ge x_j - W(1 + x_{ij} - y_{ij}) \\
&(3) \quad y_i + h_{imin} + l_i(w_{imax} - w_i) \le y_j + H(1 - x_{ij} + y_{ij}) \\
&(4) \quad y_i - h_j \ge y_j - H(2 - x_{ij} - y_{ij})
\end{aligned}
$$

- If both B_i and B_j are soft:

$$
\begin{aligned}
&(1) \quad x_i + w_i \le x_j + W(x_{ij} + y_{ij}) \\
&(2) \quad x_i - w_j \ge x_j - W(1 + x_{ij} - y_{ij}) \\
&(3) \quad y_i + h_{imin} + l_i(w_{imax} - w_i) \le y_j + H(1 - x_{ij} + y_{ij}) \\
&(4) \quad y_i - h_{jmin} - l_j(w_{jmax} - w_j) \ge y_j - H(2 - x_{ij} - y_{ij})
\end{aligned}
$$

21

Solving Linear Program

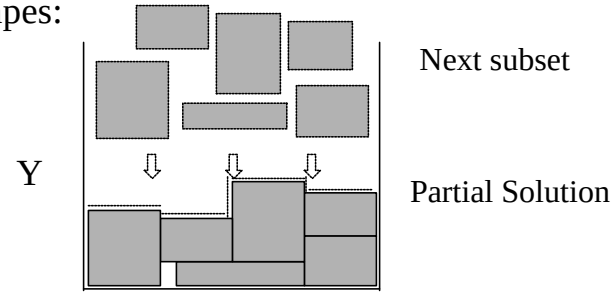
- Linear Programming (LP) can be solved by classical optimization techniques in polynomial time.
- Integer LP (ILP) is NP-Complete.
  - The run time of the best known algorithm is exponential to the no. of variables and equations.

Complexity

- For a problem with n blocks, and for the simplest case, i.e., all blocks are hard:
  - 4n continuous variables (x_i, y_i, w_i, h_i)
  - n(n-1) integer variables (x_{ij}, y_{ij})
  - $2n^2$ linear constraints
- Practically, this method can only solve small size problems.

Successive Augmentation

A classical greedy approach to keep the problem size small: repeatedly pick a small subset of blocks to formulate an ILP, solve it together with the previously picked blocks with fixed locations and shapes:

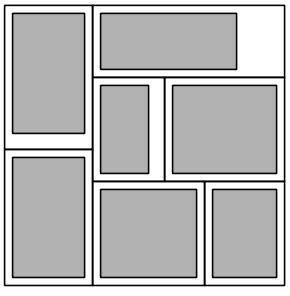


Simulated Annealing Approach

- Many floorplanning tools are based on some simulated annealing approach.
- In simulated annealing, we need to have a good representation for each candidate floorplan solution.
- There are three kinds of floorplans: slicing, mosaic and non-slicing.

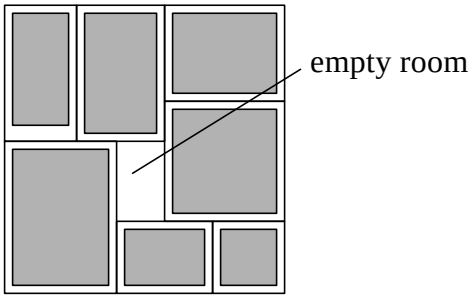
Slicing Floorplan

A floorplan that can be obtained by recursively cutting a rectangle into two by either a vertical line or a horizontal line:



Non-Slicing Floorplan

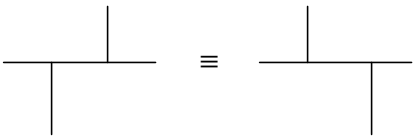
Any general floorplan which is not necessarily obtained by recursively subdividing rectangles.



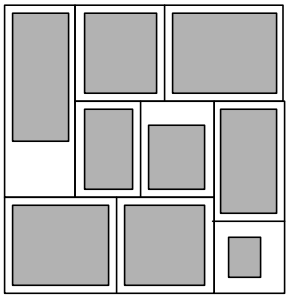
Mosaic Floorplan

A floorplan that:

- Has no empty room
- Every module corner (except those at the four corners of the chip) is formed by a T-junction.
- The non-crossing segment of a T-junction can slide along the crossing segment and gives the same packing:



Mosaic Floorplan



Note that there is no empty rooms.

P-admissible Representation

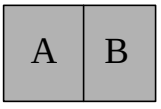
A packing representation is P-admissible if:

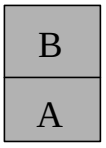
- The solution space is finite.
- Every solution corresponds to a feasible packing.
- Evaluation for each solution, i.e., computing the cost value, is possible in polynomial time, and so is the realization of the corresponding packing.
- The optimal packing is included in the solution space and corresponds to the one with the best evaluated cost value.

Slicing Floorplan Representation

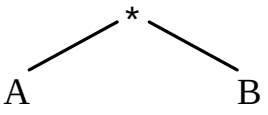
Slicing Trees

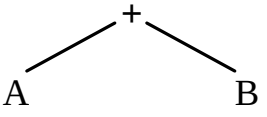
Slicing Floorplan





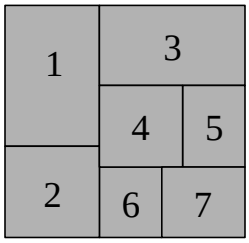
Slicing Tree



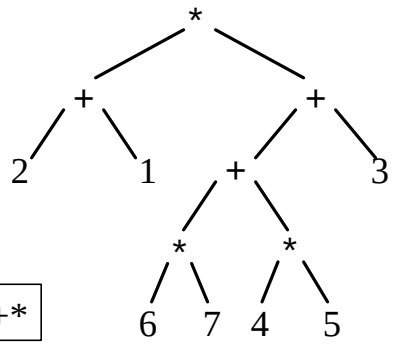


Slicing Trees

Slicing Floorplan



Slicing Tree



21+67*45*+3+*

Polish Expression
(postorder traversal of slicing tree)

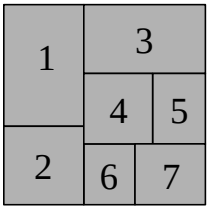
Polish Expression (PE)

- A postorder traversal of the slicing tree:
  1. Left sub-tree
  2. Right sub-tree
  3. Current root
- For *n* blocks, a PE contains *n* operands (blocks) and *n*-1 operators (*, +).
- One slicing floorplan can have more than one slicing tree (and hence more than one PE). Redundancy exists, which is not good.

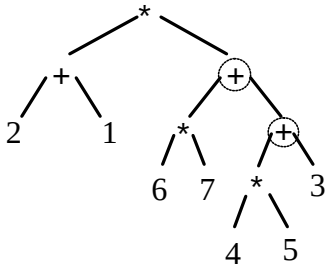
Normalized PE

A normalized Polish Expression has no consecutive + or *.

Slicing Floorplan



Slicing Tree



Polish Expression 21+67*45*3++*

Normalized PE

- There is a 1-1 correspondence between slicing floorplan and normalized PE.
- Normalized Polish Expression is commonly used to represent slicing floorplans.

Moves

- Chain: +*+*+*.... or *+*+*+….

16+35*2+*74+*

— — —— —— Chains

- Three kinds of moves:

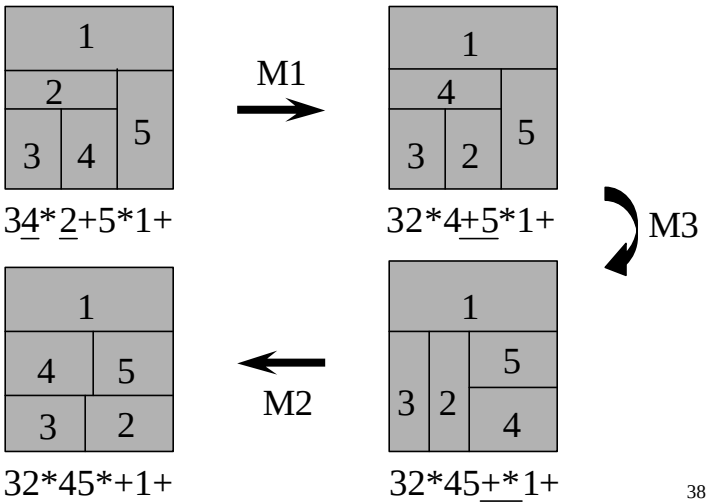
  M1: Swap 2 adjacent operands (ignoring chains)

  M2: Complement a chain

  M3: Swap 2 adjacent operand and operator

  (Note: M3 can give an invalid normalized PE, so we need to check for validity after M3.)

Examples of Moves



Notes

- The normalized PE representation for slicing floorplan is P-admissible
- The size of the solution space is $O(n!2^{3n-3})$
- Given an PE we can in linear time obtain a slicing floorplan and its size.
- All slicing floorplans with *n* modules are reachable starting from any one by using the given set of move operations.

Non-Slicing Floorplan Representation

Sequence Pair (SP)

A floorplan is represented by a pair of permutations of the module names:

e.g. 1 3 2 4 5

3 5 4 1 2

A sequence pair (s_1, s_2) of n modules can represent all possible floorplans formed by the n modules by specifying the pair-wise relationship between the modules.

Sequence Pair

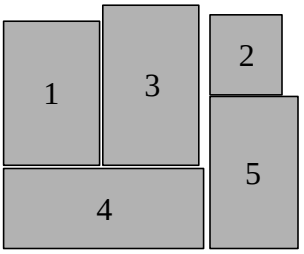
Consider a pair of modules A and B. If the arrangement of A and B in s_1 and s_2 are:

- (…A…B…, …A…B…), then the right boundary of A is on the left hand side of the left boundary of B.
- (…A…B…, …B…A…), then the upper boundary of B is below the lower boundary of A.

Example

Consider the sequence pair:

(13245,41352)



Equivalent: (13425,41352), etc.

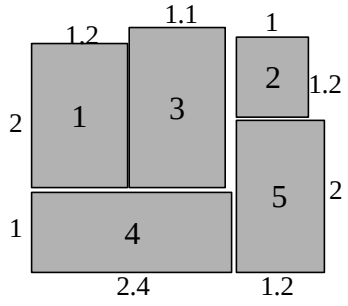
Floorplan Realization

- Floorplan realization is the step to construct a floorplan from its representation.
- How to construct a floorplan from a sequence pair?
- We can make use of the horizontal and vertical constraint graphs (G_h and G_v).

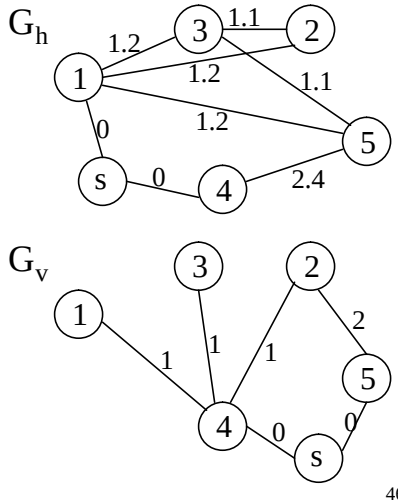
Floorplan Realization

- Whenever we see (…A…B…, …A…B…), add an edge from A to B in G_h with weight w_A.
- Whenever we see (…A…B…, …B…A…), add an edge from B to A in G_v with weight h_B.
- Add a source vertex s to G_h and G_v pointing, with weight 0, to all vertices without incoming edges.
- Finally, find the longest paths from s to every vertex in G_h and G_v, giving the coordinates of the lower left corner of the module in the packing.

Example



(13245,41352)



Constraint Graphs

- O(n^2) edges in G_h and G_v in total
- Transitive edges in G_h and G_v can be removed
- Still O(n^2) pairs A,B to consider – no guarantees for fast solutions

Moves

- Three kinds of moves in the annealing process:
  - M1: Rotate a module, or change the shape of a module
  - M2: Interchange 2 modules in both sequences
  - M3: Interchange 2 modules in the first sequence
- This set of move operations ensure reachability

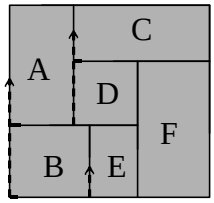
Pros and Cons of SP

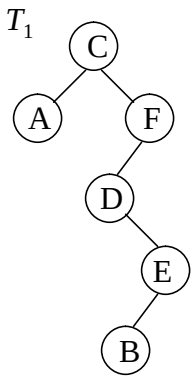
- Advantages:
  - Simple representation
  - All floorplans can be represented.
  - The solution space is finite ($O(n!^2)$)
- Disadvantages:
  - Redundant representation. The representation is not 1-to-1.
  - The size of the constraint graphs, and thus the runtime to construct the floorplan, is quadratic.

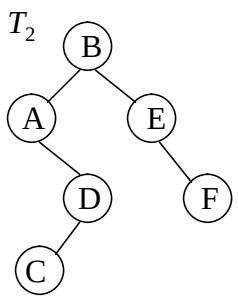
Mosaic Floorplan Representation

Mosaic Floorplan

A mosaic packing can be represented by a pair of twin binary trees:





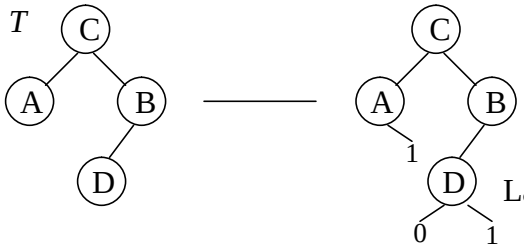


Twin Binary Trees

An arbitrary pair of trees may not correspond to a valid packing. They must be twin binary to each other, i.e., their labelings are complement of each other.

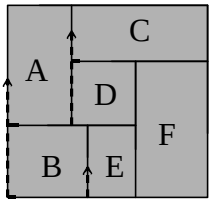
Labeling

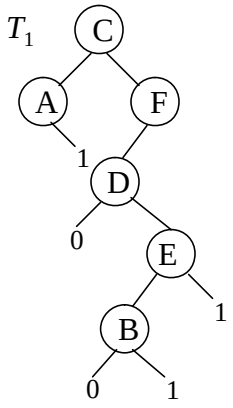
The labeling of a tree T can be obtained by traversing the tree in in-order and append a bit "0" ("1") to the labeling if a node, except the leftmost (rightmost) node, with no left (right) child is visited:



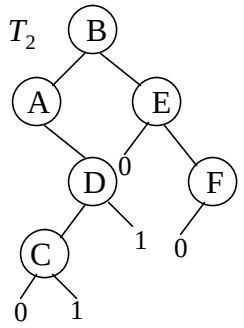
Labeling of T = 101

Twin Binary Trees





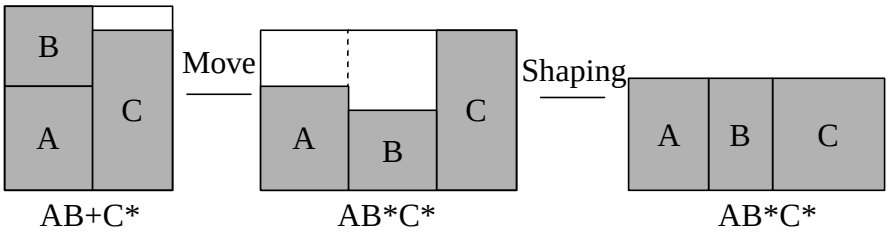
Labeling of T_1 = 10011



Labeling of T_2 = 01100

Shaping

In floorplan design, we need to determine the positions and **shapes** of the modules:



Shaping

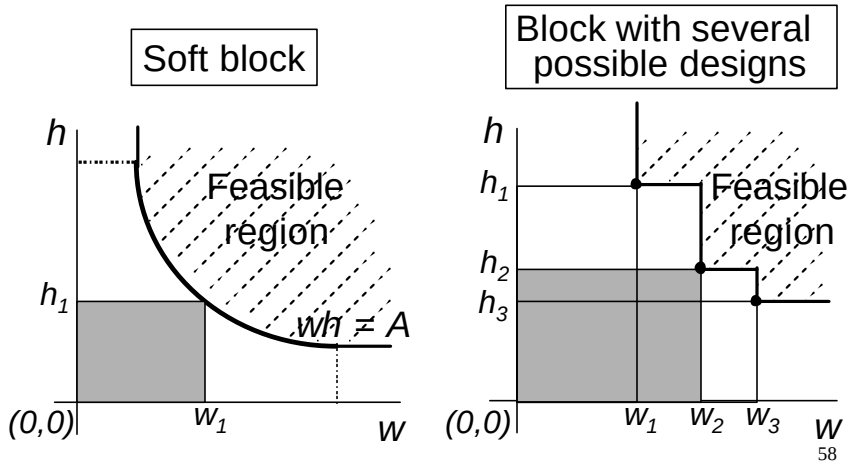
- There is a perfect method to do shaping in slicing floorplans optimally and efficiently.
- Shaping in non-slicing floorplans can be done optimally but is very time consuming.
- Shaping in mosaic floorplans has not been studied well yet.

Shaping in Slicing Floorplans

- Shaping in slicing floorplans can be done by "shape curve computation".
- Given a Polish expression of *n* modules and the areas of the modules, how do we determine their dimensions to minimize the total area of the floorplan?

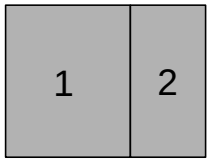
Shape Curve

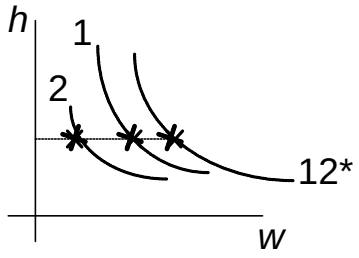
To represent the possible shapes of a block.



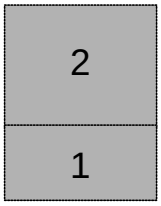
Combining Shape Curves

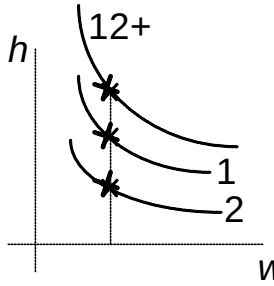
12*:





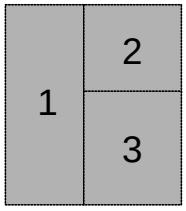
12+:

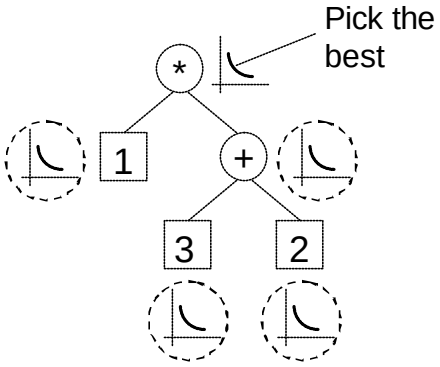




Finding the Smallest Area

Recursively combining shape curves.





Updating Shape Curves

- If each shape curve has k points, the shape curve computational time for each normalized PE is $O(kn)$.
- After each move, there are only small changes in the floorplan, so there is no need to compute all the shape curves from scratch again.
- Shape curves can be updated incrementally after each move. Expected run time is $O(k \log n)$.

Shaping in Non-slicing Floorplan

"Floorplan Area Minimization using Lagrangian Relaxation", F.Y. Young, Chris C.N. Chu, W.S. Luk and Y.C. Wong, IEEE Transaction on CAD, 20(5):687-692, 2001.

Geometric Program

We formulate the problem as a geometric program:

Minimize: $x_{n+1}\, y_{n+1}$

Subject to: $x_i + w_i \leq x_j \quad \forall$ edge $(i, j) \in G_h$

$y_i + A_i/w_i \leq y_j \quad \forall$ edge $(i, j) \in G_v$

$L_i \leq w_i \leq U_i \quad \forall$ module i

Lagrangian Relaxation

By the Lagrangian relaxation procedure, we can move the constraints into the objective function:

Minimize:

$$\sum_{(i,j)\in G_h} \lambda_{ij}^h (x_i + w_i - x_j) + \sum_{(i,j)\in G_v} \lambda_{ij}^v (y_i + A_i/w_i - y_j)$$

Subject to: $L_i \leq w_i \leq U_i \quad \forall$ modules i

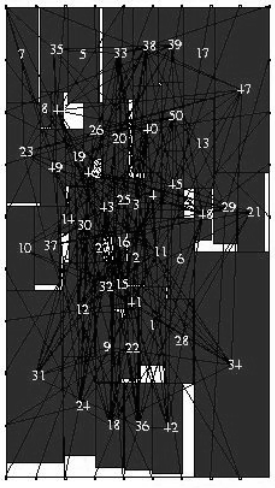
This is called the Lagrangian relaxation sub-problem and how to solve it is described in the paper.

Comparisons with Previous Results

Time for the whole annealing process

Data	*n*	This Method	Previous Method
xerox	10	8.3s	13.2 min
apte	9	11.6s	20 min
hp	11	18.9s	22.4 min
ami33	33	1 hrs	21 hrs
ami49	49	4.5 hrs	7 days
		600 MHz Pentium III	250 MHz DEC Alpha

Before Sizing - 50 Modules



Width: 1440.42

Height: 2646.5

Area: 3.81208e+06 (9.99%)

Wire Est.: 423932

Iterations: 573244

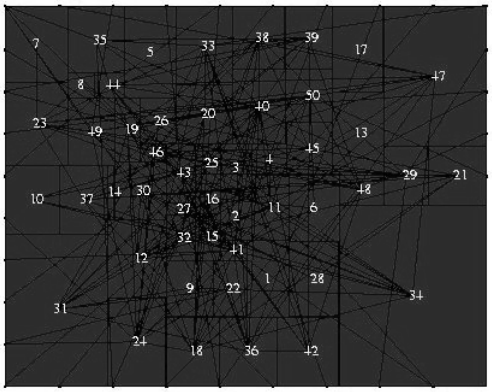
Initial T: 8.5467e+09

Lambda: 9.99325

Rotation: No

Time : 65.07(0.07)

After Sizing - 50 Modules



Width: 2103.14

Height: 1654.97

Area: 3.48064e+06 (0.42%)

Wire Est.: 393488

Iterations: 573765

Initial T: 8.5467e+09

Lambda: 9.99325

Rotation: No

Time : 193.35(0.13)

Conclusions

- Specific heuristics seem to be fast but producing not very good results
- Simulating Annealing dominating approach
- Much research about how to improve SA in this area