Analytical Placers - NTUplace3

Nikos Xiromeritis

Outline

Introduction to NTUplace3

- About NTUplace
- NTUplace3 features overview

Analytical Placement Model

- Circuit Formulation
- GP Problem Formulation (part 1)
- Wirelength Model
- Density/Potential Model
- GP Problem Formulation (part 2)
- Overlap Weight Intuition

Global Placement

- GP Algorithm
- Multilevel Framework
- CG Search with Dynamic Step Sizes
- Base Potential Smoothing
- Macro Shifting
- WSA for Density Control

Legalisation

- Mixed-Size LG
- Look-Ahead LG

Detailed Placement

- Wirelength Minimization
- Density Optimization

About NTUplace

- NTUplace3 motivation

- Placers solely driven by wirelengh tend to pack blocks (cells/components) together to minimize the wirelength.

- Additionally the possible existence of preplaced (or fixed) blocks impose even more constrains to the placement problem.

- What is NTUplace?

- NTUPlace3 is a high-quality mixed-size analytical placement algorithm considering preplaced blocks and density constraints.

- The placer is based on a three-stage technique: 1) Global Placement (GP); 2) Legalization (LG); and 3) Detailed Placement (DP)

NTUplace3 Features

- NTUPlace3 has the following distinguished features.

1) NTUplace3 is based on the log-sum-exp wirelength net model.

2) The placement minimization problem is solved by Conjugate Gradient (CG) method with dynamic step sizes.

3) Preplaced blocks are handled by a two-stage smoothing technique (Gaussian + Level smoothing)

4) Whitespace reallocation using partitioning and cut-line shifting.

5) A macro shifting technique is used between levels of GP to find better macro position that are easier for the Legalization.

6) Look-ahead LG scheme is used during GP. The legalizer is called several times near the end of GP.

Outline

Introduction to NTUplace3

- About NTUplace
- NTUplace3 features overview

Analytical Placement Model

- Circuit Formulation
- GP Problem Formulation (part 1)
- Wirelength Model
- Density/Potential Model
- GP Problem Formulation (part 2)
- Overlap Weight Intuition

Global Placement

- GP Algorithm
- Multilevel Framework
- CG Search with Dynamic Step Sizes
- Base Potential Smoothing
- Macro Shifting
- WSA for Density Control

Legalisation

- Mixed-Size LG
- Look-Ahead LG

Detailed Placement

- Wirelength Minimization
- Density Optimization

Circuit Formulation

- Circuit formulated as a Hypergraph (H = (V, E))
  - Vertices $V = \{u_1, u_2, \ldots u_n\}$ represent blocks.
  - Hyperedges $E = \{e_1, e_2, \ldots e_m\}$ represent nets.
  - Let $\boldsymbol{x_i}$ and $\boldsymbol{y_i}$, be the $\boldsymbol{x}$ and $\boldsymbol{y}$ coordinates of the center of block $\boldsymbol{u_i}$
  - Let $\boldsymbol{a_i}$ be the area of the block $\boldsymbol{u_i}$

x_i, y_i	center coordinate of block v_i
w_i, h_i	width and height of block v_i
w_b, h_b	width and height of bin b
P_b	base potential (area of preplaced blocks) in bin b
D_b	potential (area of movable blocks) in bin b
M_b	the maximum potential in bin b
$t_{density}$	target placement density

GP Problem Formulation (part 1)

- Formulation with **Bin Grids**

To evenly distribute the blocks the placement region is divided into uniform nonoverlaping **bin grids**. Then, the GP problem can be formulated as a constrained minimization problem:

$$\begin{aligned} \min \quad & W(\mathbf{x}, \mathbf{y}) \\ \text{s.t.} \quad & D_b(\mathbf{x}, \mathbf{y}) \leq M_b, \quad \text{for each bin } b \end{aligned} \qquad (1)$$

Where,

$\mathbf{W(x, y)}$ is the wirelength function

$\mathbf{D_b(x, y)}$ is the potential function (total area of movable blocks in bin b)

$\mathbf{M_b}$ is the maximum allowable are of movable blocks in bin b.

and $M_b = t_{density}(w_b h_b - P_b)$

where $t_{density}$ is a user specified target density value for each bin.

Wirelength Model

- HPWL - Approximation

The most widely used model that describes the wirelength (*W(x, y)*) is the *total-half-perimeter wirelength (HPWL).*

$$W(\mathbf{x},\mathbf{y}) = \sum_{\text{net } e} \left(\max_{v_i,v_j \in e} |x_i - x_j| + \max_{v_i,v_j \in e} |y_i - y_j| \right). \quad (3)$$

Although, the HPWL function is not smooth and nonconvex and thus it is hard to directly minimize it. NTUplace 3 uses a **log-sum-exp** function which gives a HPWL approximation (for a reasonably small γ).

$$W(\mathbf{x},\mathbf{y}) = \gamma \sum_{e \in E} \left((\log \sum_{v_k \in e} \exp(x_k/\gamma) + \log \sum_{v_k \in e} \exp(-x_k/\gamma) \right.$$
$$\left. + \log \sum_{v_k \in e} \exp(y_k/\gamma) + \log \sum_{v_k \in e} \exp(-y_k/\gamma) \right) \quad (4)$$

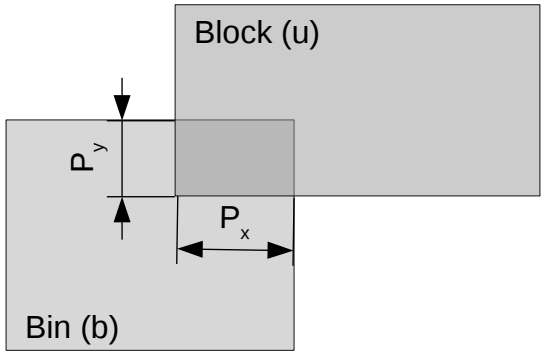
Density/Potential Model

- Initial Density Model

Function $D_b(x, y)$ describes the potential (area of movable blocks) in bin *b.*

$$D_b(\mathbf{x}, \mathbf{y}) = \sum_{v \in V} P_x(b, v) P_y(b, v) \qquad (5)$$

P_x and P_y are the overlap functions of bin *b* and block u.



- Smoothed overlap functions

Due to the non-smooth characteristic of the previous overlap functions, the bell-shaped potential functions p_x and p_y are adopted to replace P_x and P_y. Variables d_x (d_y) is the center-to-center distance of the block *u* and the bin *b* in the *x (y)* direction

$$p_x(b,v) = \begin{cases} 1 - a d_x^2, & 0 \le d_x \le \frac{w_v}{2} + w_b \\ b\left(d_x - \frac{w_v}{2} - 2w_b\right)^2, & \frac{w_v}{2} + w_b \le d_x \le \frac{w_v}{2} + 2w_b \\ 0, & \frac{w_v}{2} + 2w_b \le d_x \end{cases} \qquad (6)$$

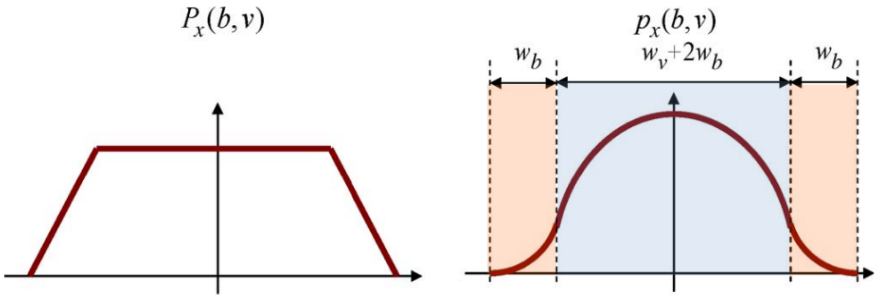
where

$$a = \frac{4}{(w_v + 2w_b)(w_v + 4w_b)}$$

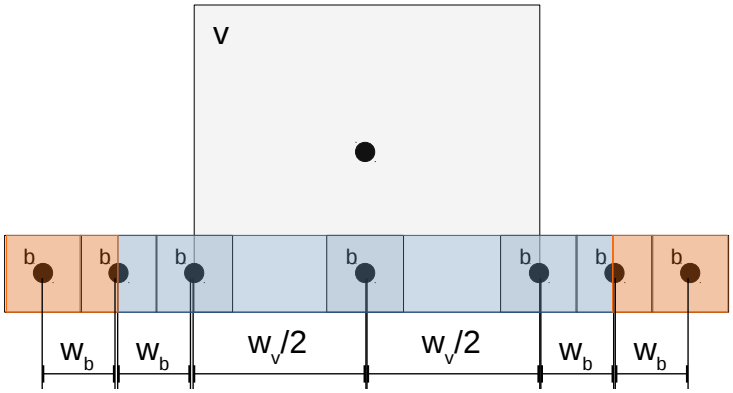
$$b = \frac{2}{w_b(w_v + 4w_b)} \qquad (7)$$

Density/Potential Model (cont.)

- Visualizing the Overlap functions



$$p_x(b,v) = \begin{cases} 1 - ad_x^2, & 0 \le d_x \le \frac{w_v}{2} + w_b \\ b\left(d_x - \frac{w_v}{2} - 2w_b\right)^2, & \frac{w_v}{2} + w_b \le d_x \le \frac{w_v}{2} + 2w_b \\ 0, & \frac{w_v}{2} + 2w_b \le d_x \end{cases} \quad (6)$$



Judging from the non-smooth function, someone might think that the overlap function's value:
1) Starts increasing from the very moment the block touches the bin while moving towards it.
2) Reaches the maximum value when the bin is completely covered by the cell.
3) Is decreasing while the cell exits the bin

Although, the above statements are **incorrect** as the smoothed function gives us non-zero values even when the block is away from the bin. (why?)

Density/Potential Model (cont.)

- Redefining the Density Function

Function $D_b(x, y)$ now can be replaced by a smooth one.

$$\hat{D}_b(\mathbf{x}, \mathbf{y}) = \sum_{v \in V}^{n} c_v p_x(b, v) p_y(b, v) \qquad (8)$$

Where, c_u is a normalization factor so that the total potential of a block equals its area.

- More about the Normalization Factor

The max value of the bell function p_x (p_y) is 1 when d_x (d_y) is 0. The c_u factor must be properly assigned for each block to normalize the max potential value to the block area.

GP Problem Formulation (part 2)

- Quadratic Penalty Method

To solve the GP problem the quadratic penalty method is used, implying that we solve a sequence of unconstrained minimization problems of the form:

$$\min \quad W(\mathbf{x},\mathbf{y}) + \lambda \sum_{b} \left(\hat{D}_b(\mathbf{x},\mathbf{y}) - M_b \right)^2 \qquad (9)$$

with **increasing λ's** (overlap weight). The solution of the previous problem is used as the initial solution for the next one. The minimization problem is solved by the CG (Conjugate Gradient) method with a dynamic step size (since it is more runtime friendly)

- The Sum-of-Squares Term

If the minimization problem didn't contain the sum-of-squares term, the blocks would tend to be packed together in order to minimize the wirelength. The role of the second term is to 'push' the movable blocks towards the maximum allowable density as the value of the term increases when the cells come too close or too for away from each other.

Overlap Weight Intuition

- Weight λ affects GP's Density Significance

With the **overlap weight λ** increasing, the significance of the second term of the minimization problem increases in later solutions. This prohibits cells from comming too close to or too far away from each other.

Example: Let λ=1 initially and λ=2 after the first solution.

Consider the two following combinations:

a) $W(\mathbf{x},\mathbf{y}) = 20$ while $\sum_b \left(\hat{D}_b(\mathbf{x},\mathbf{y}) - M_b\right)^2 = 0.1$

b) $W(\mathbf{x},\mathbf{y}) = 10$ while $\sum_b \left(\hat{D}_b(\mathbf{x},\mathbf{y}) - M_b\right)^2 = 6$

We can see that combination b) is better for the first solution and a) for the second solution. The significance of the second term is increased as λ increases.
(meaning that the value of the second term gets more critical to the minimization problem)

Outline

Introduction to NTUplace3

- About NTUplace
- NTUplace3 features overview

Analytical Placement Model

- Circuit Formulation
- GP Problem Formulation (part 1)
- Wirelength Model
- Density/Potential Model
- GP Problem Formulation (part 2)
- Overlap Weight Intuition

Global Placement

- GP Algorithm
- Multilevel Framework
- CG Search with Dynamic Step Sizes
- Base Potential Smoothing
- Macro Shifting
- WSA for Density Control

Legalisation

- Mixed-Size LG
- Look-Ahead LG

Detailed Placement

- Wirelength Minimization
- Density Optimization

GP Algorithm

- Algorithm Characteristics

- Two-stage flow of clustering followed by declustering.
- GP is performed at each level of declustering.
- Preplaced blocks handled by a two-stage smoothing (Gaussian + Level smoothing)
- White space distribution, to allocate white space to areas with density overflows
- Macro shifting and look-ahead LG applied at that GP stage.

- Divided Into Sections

- Multilevel Framework
- CG Search With Dynamic Step Sizes
- Base Potential Smoothing
- Macro Shifting
- WSA for Density Control

Multilevel Framework

Algorithm: **Multilevel Global Placement**
Input:
hypergraph H_0: a mixed-size circuit
n_{max}: the maximum block number in the coarsest level
Output:
(x^*, y^*): optimal block positions without considering block overlaps

01. $level = 0$;
02. **while** $(BlockNumber(H_{level}) > n_{max})$
03. $level$++;
04. $H_{level} = FirstChoiceClustering(H_{level-1})$;
05. initialize block positions by $SolveQP(H_{level})$;
06. **for** $currentLevel = level$ **to** 0
07. initialize bin grid size;
08. initialize base potential for each bin;
09. initialize $\lambda_0 = \frac{\sum |\partial W(\mathbf{x},\mathbf{y})|}{\sum |\partial \hat{D}_b(\mathbf{x},\mathbf{y})|}$; $m = 0$;
10. **do**
11. solve min $W(\mathbf{x},\mathbf{y}) + \lambda_m \sum (\hat{D}_b - M_b)^2$;
12. $m++$;
13. $\lambda_m = 2\lambda_{m-1}$;
14. **if** $(currentLevel == 0$ && $overflow\_ratio < 10\%)$
15. call $LookAheadLegalization()$ and save the best result;
16. compute $overflow\_ratio$;
17. **until** (spreading enough or no further reduction in $overflow\_ratio$)
18. **if** $(currentLevel == 0)$
19. restore the best look-ahead result;
20. **else**
21. call $MacroShifting()$;
22. call $WhiteSpaceAllocation()$;
23. decluster and update block positions.

Coarsening Stage (lines 1-4)

The number of movable blocks is reduced by **first-choice (FC) clustering**. The blocks are examined one-by-one, and the two blocks with the highest connectivity are clustered. The clustered block area should not be 1.5 times larger than the average area of clustered blocks.

The process continues until the number of blocks is reduced by 5 times and then a new circuit level is defined. The FC clustering algorithm is applied until the block number in the resulting clustered circuit is less than n_{max} (6000 by default)

Initial Placement (line 5)

The initial placement is generated by minimizing the quadratic wirelength using the CG method.

Multilevel Framework (cont.)

Algorithm: **Multilevel Global Placement**
Input:
hypergraph H_0: a mixed-size circuit
n_{max}: the maximum block number in the coarsest level
Output:
(x^*, y^*): optimal block positions without considering block overlaps

01. $level = 0$;
02. **while** $(BlockNumber(H_{level}) > n_{max})$
03. $level$++;
04. $H_{level} = FirstChoiceClustering(H_{level-1})$;
05. initialize block positions by $SolveQP(H_{level})$;
06. **for** $currentLevel = level$ **to** 0
07. initialize bin grid size;
08. initialize base potential for each bin;
09. initialize $\lambda_0 = \frac{\sum |\partial W(\mathbf{x},\mathbf{y})|}{\sum |\partial \hat{D}_b(\mathbf{x},\mathbf{y})|}$; $m = 0$;
10. **do**
11. solve min $W(\mathbf{x},\mathbf{y}) + \lambda_m \sum (\hat{D}_b - M_b)^2$;
12. $m++$;
13. $\lambda_m = 2\lambda_{m-1}$;
14. **if** $(currentLevel == 0$ && $overflow\_ratio < 10\%)$
15. call $LookAheadLegalization()$ and save the best result;
16. compute $overflow\_ratio$;
17. **until** (spreading enough or no further reduction in $overflow\_ratio$)
18. **if** $(currentLevel == 0)$
19. restore the best look-ahead result;
20. **else**
21. call $MacroShifting()$;
22. call $WhiteSpaceAllocation()$;
23. decluster and update block positions.

<u>Uncoarsening Stages</u> (lines 6-23)

- The placement problem is solved from the coarsest to the finest level.

- The placement for the current level provides the initial placement for the next level.

- The bin dimensions are set to the square root of the number of clusters in the current level.
bin_grid_dim = $grid\_num\_h = \sqrt{BlockNumber(H_{level})}$

- After the bin dimension computation, P_b is computed for each bin and M_b is updated accordingly.

Multilevel Framework (cont.)

Algorithm: Multilevel Global Placement
Input:
hypergraph H_0: a mixed-size circuit
n_{max}: the maximum block number in the coarsest level
Output:
(x^*, y^*): optimal block positions without considering block overlaps

01. $level = 0$;
02. **while** $(BlockNumber(H_{level}) > n_{max})$
03. $level$++;
04. $H_{level} = FirstChoiceClustering(H_{level-1})$;
05. initialize block positions by $SolveQP(H_{level})$;
06. **for** $currentLevel = level$ **to** 0
07. initialize bin grid size;
08. initialize base potential for each bin;
09. initialize $\lambda_0 = \frac{\sum |\partial W(\mathbf{x},\mathbf{y})|}{\sum |\partial \hat{D}_b(\mathbf{x},\mathbf{y})|}$; $m = 0$;
10. **do**
11. solve min $W(\mathbf{x},\mathbf{y}) + \lambda_m \sum (\hat{D}_b - M_b)^2$;
12. $m++$;
13. $\lambda_m = 2\lambda_{m-1}$;
14. **if** $(currentLevel == 0$ && $overflow\_ratio < 10\%)$
15. call $LookAheadLegalization()$ and save the best result;
16. compute $overflow\_ratio$;
17. **until** (spreading enough or no further reduction in $overflow\_ratio$)
18. **if** $(currentLevel == 0)$
19. restore the best look-ahead result;
20. **else**
21. call $MacroShifting()$;
22. call $WhiteSpaceAllocation()$;
23. decluster and update block positions.

<u>Uncoarsening Stages</u> (lines 6-23)

- For each level, the factor λ is initialized according to the strength of wirelength and density gradients as:

$$\lambda = \frac{\sum |\partial W(\mathbf{x},\mathbf{y})|}{\sum \left|\partial \hat{D}_b(\mathbf{x},\mathbf{y})\right|} \quad (10)$$

(Intuition?)
If the wirelength's total rate of change is bigger than density's rate of change, (i.e. λ>1, the wirelength's absolute value changes more 'violently') the λ factor will hold a big value and the density term of the problem will have a more critical role during the minimization.

- The value of λ is multiplied by two at each iteration.

Multilevel Framework (cont.)

Algorithm: **Multilevel Global Placement**
Input:
hypergraph H_0: a mixed-size circuit
n_{max}: the maximum block number in the coarsest level
Output:
(x^*, y^*): optimal block positions without considering block overlaps

01. $level = 0$;
02. **while** $(BlockNumber(H_{level}) > n_{max})$
03. $level$++;
04. $H_{level} = FirstChoiceClustering(H_{level-1})$;
05. initialize block positions by $SolveQP(H_{level})$;
06. **for** $currentLevel = level$ **to** 0
07. initialize bin grid size;
08. initialize base potential for each bin;
09. initialize $\lambda_0 = \frac{\sum |\partial W(\mathbf{x},\mathbf{y})|}{\sum |\partial \hat{D}_b(\mathbf{x},\mathbf{y})|}$; $m = 0$;
10. **do**
11. solve min $W(\mathbf{x},\mathbf{y}) + \lambda_m \sum (\hat{D}_b - M_b)^2$;
12. $m++$;
13. $\lambda_m = 2\lambda_{m-1}$;
14. **if** $(currentLevel == 0$ && $overflow\_ratio < 10\%)$
15. call $LookAheadLegalization()$ and save the best result;
16. compute $overflow\_ratio$;
17. **until** (spreading enough or no further reduction in $overflow\_ratio$)
18. **if** $(currentLevel == 0)$
19. restore the best look-ahead result;
20. **else**
21. call $MacroShifting()$;
22. call $WhiteSpaceAllocation()$;
23. decluster and update block positions.

<u>Uncoarsening Stages</u> (lines 6-23)

- During uncoarsening, all blocks inside a cluster inherit the center position of the original cluster.

- Macro shifting for LG and WSA for density control are applied between uncoarsening levels.

- Overflow ratio, describes the total overflow area in all bins over the area of total movable blocks:

$$overflow\_ratio = \frac{\sum_b \max\left(D_b(\mathbf{x},\mathbf{y}) - M_b, 0\right)}{\sum \text{total movable area}} \qquad (11)$$

We have overflow when $D_b(x, y) \geq M_b$
(also see <u>discrepancy</u>: maximum ratio of the actual total block area to the maximum allowable block area over all windows within the chip (KW2?))

CG Search With Dynamic Step Sizes

- Conjugate Gradient Step Size

After computing CG direction d_k, the step size a_k is computed by: $$\alpha_k = \frac{s w_b}{\|d_k\|_2} \quad (12)$$

where s is a user-specified scaling factor, and w_b is the bin width.

By doing so, the step size of block spreading can be limited since the total quadratic Euclidean movement is fixed as:

$$\sum_{v_i \in V} \left(\Delta x_i^2 + \Delta y_i^2\right) = \|\alpha_k d_k\|_2^2 = s^2 w_b^2 \quad (13)$$

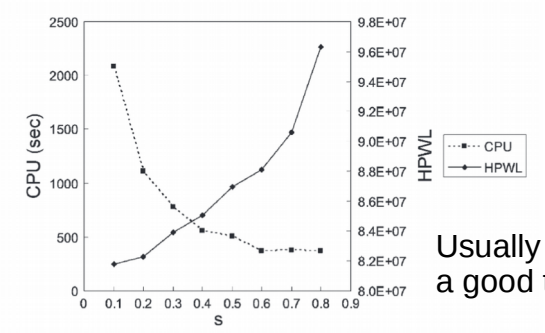
where Δx_i and Δy_i are the amount of movement along the x and y-directions for the block u_i in each iteration

Algorithm: Conjugate Gradient Algorithm with Dynamic Step-Size Control

Input:
$f(x)$: objective function
x_0: initial solution
s: step size

Output:
optimal x^*

01. initialize $g_0 = 0$ and $d_0 = 0$;
02. **do**
03. compute gradient directions $g_k = \nabla f(x_k)$;
04. compute the Polak-Ribiere parameter $\beta_k = \frac{g_k^T(g_k - g_{k-1})}{||g_{k-1}||^2}$;
05. compute the conjugate directions $d_k = -g_k + \beta_k d_{k-1}$;
06. compute the step size $\alpha_k = s/||d_k||_2$;
07. update the solution $x_k = x_{k-1} + \alpha_k d_k$;
08. **until** $(f(x_k) > f(x_{k-1}))$



Usually s ε [0.2, 0.3] which gives a good tradeoff between runtime and quality

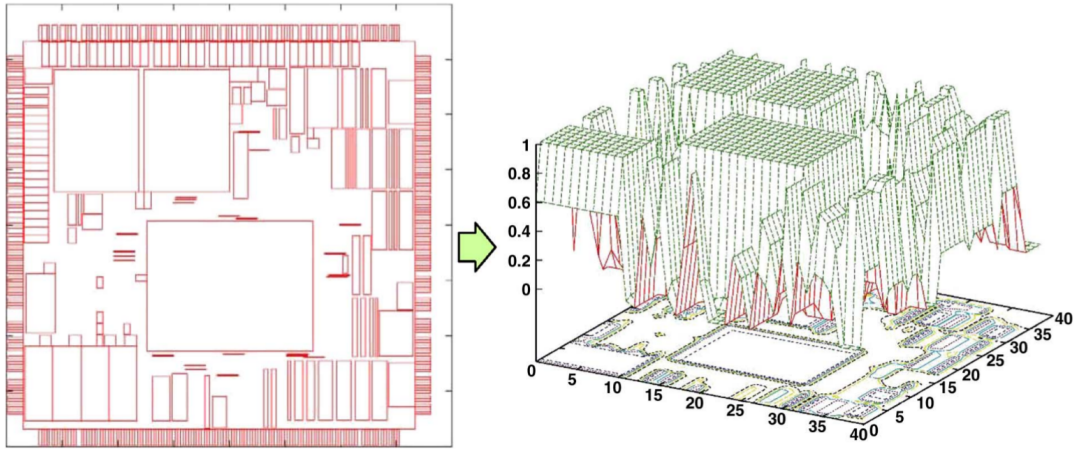
Base Potential Smoothing

- The non-smooth Base Potential Problem

- Since the base potential is not smooth, it forms mountains that prevent movable blocks from passing though the preplaced block regions. But how does the non-smooth base potential create such problem?

Recall that P_b is used in the M_b computation (max potential)**:** $M_b = t_{density}(w_b h_b - P_b)$

Also M_b plays as significant role to the second term of the minimization problem's function.

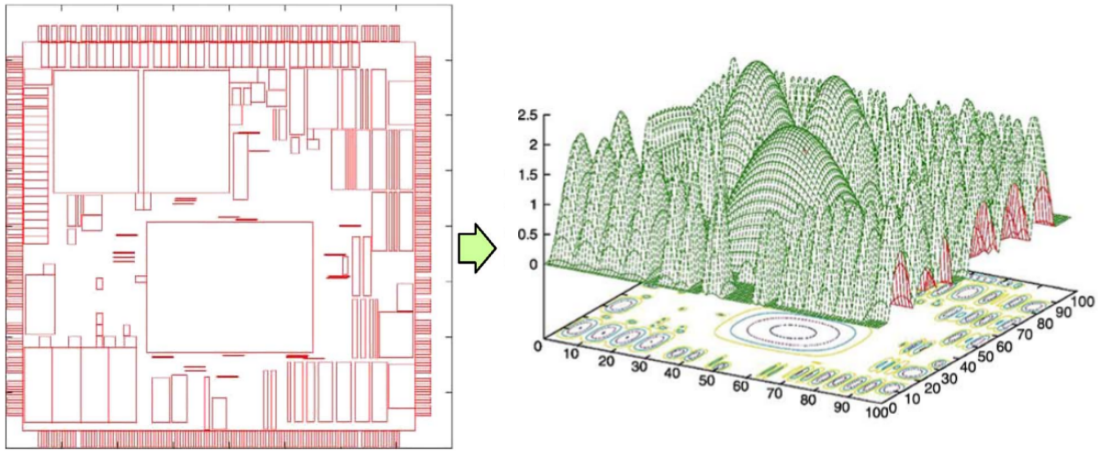


Base Potential Smoothing (cont.)

- Bell-Shaped Smoothing

- A way to smooth the base potential if by using the bell-shaped function. However, the resulting smooth base potential has “valleys” between the adjacent regions of blocks. The z-coord is the value of $P_b/(w_b h_b)$. If $z > 1$, the potential in the bin is larger than the bin area.

- Although, with this smoothing technique, a large number of blocks may spread into the “valley” regions which sometimes is incorrect, as the regions do not necessarily represent free space.



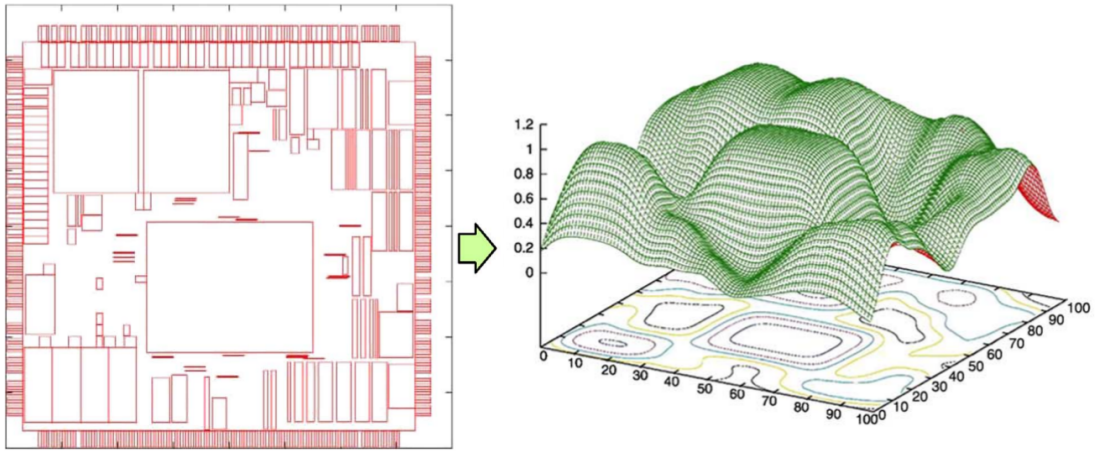
Base Potential Smoothing (cont.)

- Gaussian Smoothing

- To avoid the bell-shaped smoothing problem, the Guassian function is used to smooth the initial non-smooth base potential P_b.

- The 2-D Gaussian has the-form: $$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (14)$$

- Calculating the convolution of G with P we get: $$P'(x,y) = G(x,y) * P(x,y) \quad (15)$$



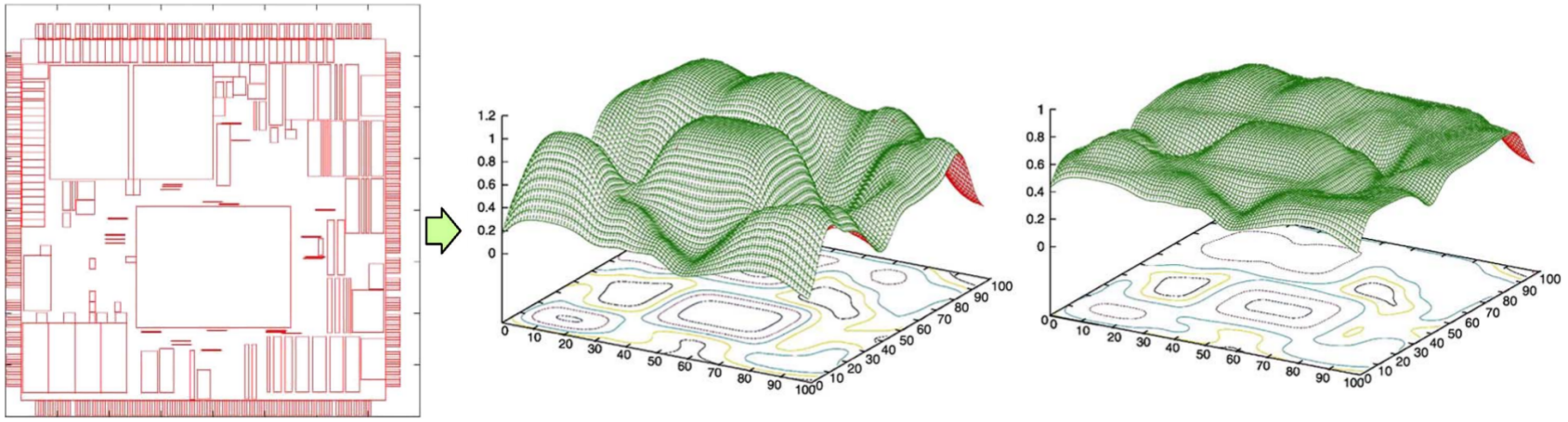
Base Potential Smoothing (cont.)

- Gaussian Smoothing with Level Smoothing

After Gaussian smoothing, we apply another landscape smoothing function to reduce potential levels. The smoothing function os defined as:

$$P''(x,y) = \begin{cases} \overline{P'} + \left(P'(x,y) - \overline{P'}\right)^{\delta}, & \text{if } P'(x,y) \geq \overline{P'} \\ \overline{P'} - \left(\overline{P'} - P'(x,y)\right)^{\delta}, & \text{if } P'(x,y) \leq \overline{P'} \end{cases} \quad (16)$$

Where $\overline{P'}$ is the average value of $P'(x, y)$ and $\delta \geq 1$.

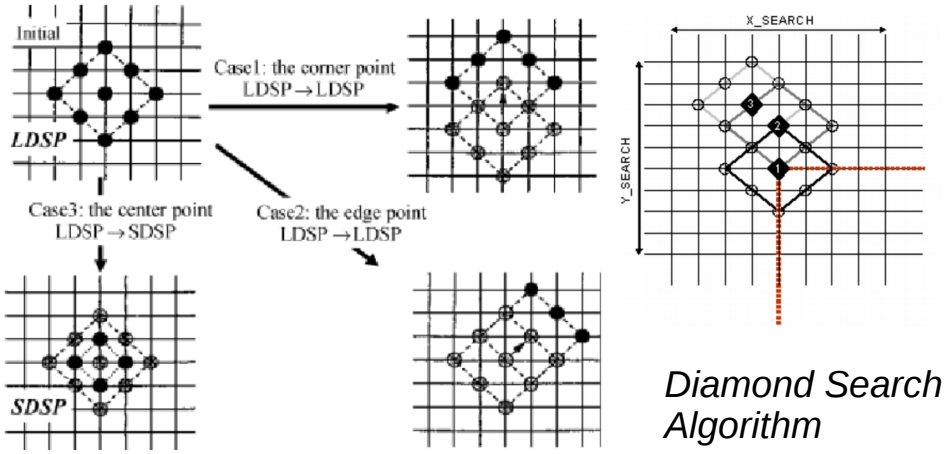


Macro Shifting

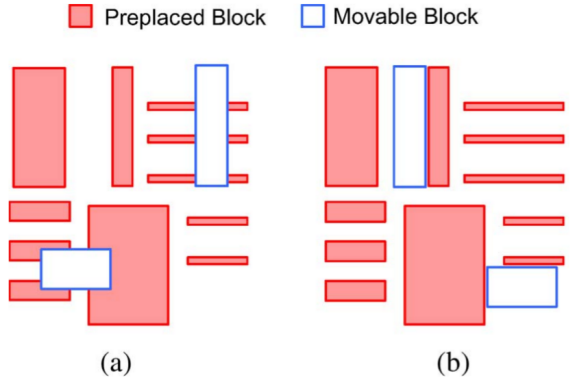
- Macro Shifting Motivation

- In the GP stage, it is important to preserve legal macro positions since illegal position may make the task of LG much more difficult.

- Macro shifting is applied at each declustering level of the GP stage. It first determines an order for all unclustered macros by both their coordinates and sizes (similar to the mixed-size LG described later) Finally, the macros are moved to their closest legal positions by diamond search.



Diamond Search Algorithm



a) A given GP with two macros being placed at illegal positions. b) Macro shifting result with legal macro positions

WSA for Density Control

- White Space Allocation

- After block spreading, some regions may still have overflows. Although, these overflows can be reduced by assigning an appropriate amount of white space.

- WSA Algorithm Steps

- Partitioning & Slicing Tree Generation
- Whitespace Computation for each partition (bottom-up)
- Whitespace Distribution (top-down)

WSA for Density Control (cont.)

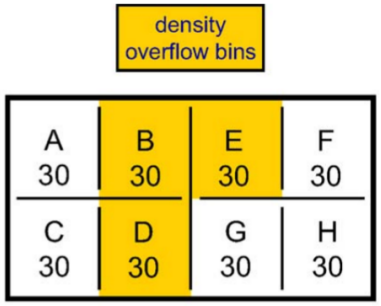
- Partitioning & Slicing Tree Generation

- The placement region is recursively partitioned and a slicing tree is constructed to record the cut directions and the blocks inside the partition.

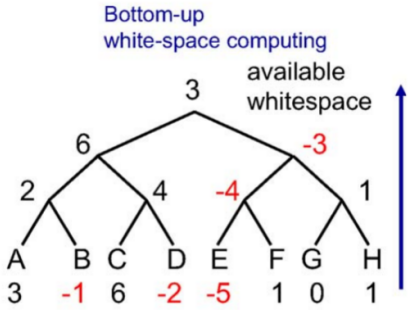
- The partition continues until the partitioned area is similar to that of a GP bin.

- In order to not generate subpartitions with large aspect ratios, the larger side of the partitions is chosen in order to evenly divide them.

- The partitioning is based on geometric locations of blocks instead of the cut size minimization.



The initial partitions and the cut lines. Each partition has an area of 30.



White space of every node after the bottom up process.

WSA for Density Control (cont.)

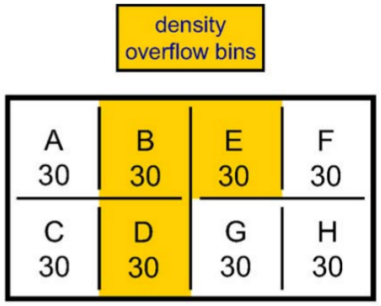
- Whitespace Computation for each Partition (bottom-up)

- After the construction of the partitions and the slicing tree, the whitespace of each partition is computed.

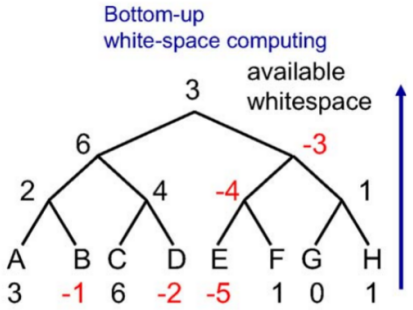
- The algorithm starts by updating the leaf nodes and proceeds to the parents until the root is reached.

- A negative white space value $w < 0$ means that the partition has an overflow area of $|w|$.

- For example the partitions B, D and E have overflow areas of 1, 2 and 5 respectively.



The initial partitions and the cut lines. Each partition has an area of 30.



White space of every node after the bottom up process.

WSA for Density Control (cont.)

- Whitespace Distribution (top-down)

- The white spaces are distributed to the two children (starting from the root) in a top- down process according to the following rules:

  1. If a child node has $w < 0$, we allocate 0 white space. The remaining white space are distributed to the other child.
  2. If the two children have $w \geq 0$, we allocate the white space proportional to their original whitespace amount.

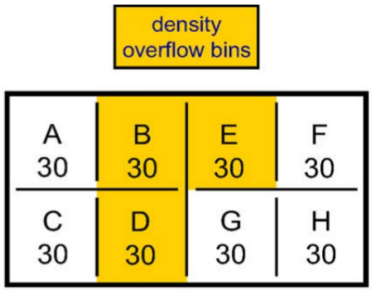
- The new partition area can be computed by: $a' = a + w' - w$

- The cut-line adjustment is also performed in a top-down fashion. The desired areas of the two subpartitions are known from the data structure of the two children.

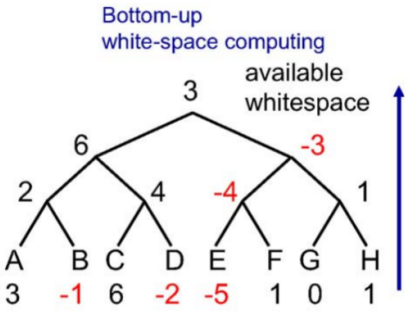
- The new block positions can be computed by linear interpolation of the coordinates of the old partition and the new one.

WSA for Density Control (cont.)

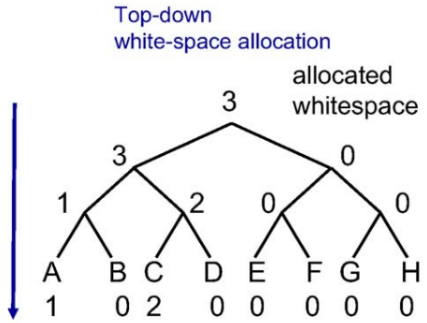
- Whitespace Distribution (top-down)



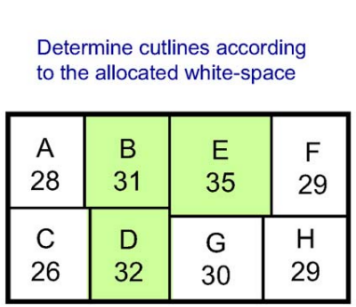
The initial partitions and the cut lines. Each partition has an area of 30.



White space of every node after the bottom up process.



Top down whitespace distribution.



a' = a + w' - w
for c: a_c' = 30 + 2 – 6 = 26

Outline

Introduction to NTUplace3

- About NTUplace
- NTUplace3 features overview

Analytical Placement Model

- Circuit Formulation
- GP Problem Formulation (part 1)
- Wirelength Model
- Density/Potential Model
- GP Problem Formulation (part 2)
- Overlap Weight Intuition

Global Placement

- GP Algorithm
- Multilevel Framework
- CG Search with Dynamic Step Sizes
- Base Potential Smoothing
- Macro Shifting
- WSA for Density Control

Legalisation

- Mixed-Size LG
- Look-Ahead LG

Detailed Placement

- Wirelength Minimization
- Density Optimization

Mixed-Size LG

- Block Priorities:

- LG stage removes all overlaps with minimal total displacement.

- The LG orders of macros and cells are determined by the x coordinates, widths and heights.

- A block u_i has priority: $priority(v_i) = k_1 x_i + k_2 w_i + k_3 h_i$ (17)
  where k1, k2 and k3 are user-specified weights. (NTUplace3: k1 = 1000, k2 = k3 = 1)

- Macro – Standard Cell differences and LG:

- Since macros have larger widths/heights, they are legalized earlier than standard cells when they have the same x coordinate.

- Another difference between macros and cells is that cells are packed into rows while macros are placed to their nearest available positions (macro shifting). (during LG?)

Look-Ahead LG

- Why is Look-Ahead LG needed?

- If the blocks are not spread enough after GP, the wirelength may significantly be increase after LG.

- If the blocks are spread too much after GP, the wirelength before LG may not be good and it also may worsen during the LG process.

- With density playing a role in the minimization problem, the placement objective is becoming more complex. Thus, the LG process is incorporated into the GP process.

- Look-Ahead Legalisation

- Look-Ahead legalisation is applied at the finest level (where there are no clusters) after minimizing the nonlinear objective in each iteration.

- The best result with the minimum cost is recorded.

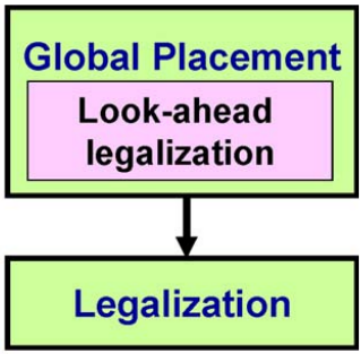
Look-Ahead LG (cont.)

- NTUPlace3 Look-Ahead LG

- Look-Ahead LG is used to reserve the desired GP results in the finest level.

- Applies a Tetris-like LG to fined the legalized results.

- Look-Ahead LG is performed only when the overflow ratio is less than 10% (why?)



Outline

Introduction to NTUplace3

- About NTUplace
- NTUplace3 features overview

Analytical Placement Model

- Circuit Formulation
- GP Problem Formulation (part 1)
- Wirelength Model
- Density/Potential Model
- GP Problem Formulation (part 2)
- Overlap Weight Intuition

Global Placement

- GP Algorithm
- Multilevel Framework
- CG Search with Dynamic Step Sizes
- Base Potential Smoothing
- Macro Shifting
- WSA for Density Control

Legalisation

- Mixed-Size LG
- Look-Ahead LG

Detailed Placement

- Wirelength Minimization
- Density Optimization

Detailed Placement

- NTUPlace3 Detailed Placement

- During the DP phase, the placement result is optimised without moving macro blocks

- NTUplace3 DP techniques:

  1) Cell Matching and branch-and-bound cell swapping for wirelength optimization

  2) Cell sliding for density optimization

- Prerequisites:

- Bipartite matching algorithm:

  A graph G = (V, E) is bipartite if the vertex set V can be partitioned into two sets A and B (the bipartition) such that no edge in E has both endpoints in the same set of the bipartition.

- Shortest Augmenting Path:

  Augmented path is a directed path from a source node to a sink node in a residual network.

Wirelength Minimization

- Wirelength Minimization Algorithm (NTUplace3)

- An extension of Window-Based DP (WDP). The algorithm finds a group of exchangeable cells inside a given window and formulates a bipartite matching problem by matching the cells to all empty slots in the window.

- Legality is kept, as the cells participating must have widths less or equal to the slot width.

- The shortest augmenting path algorithms used to solve the bipartite matching problem.

- The bipartite matching problem can very quickly be solved when the problem size is smaller than 100 cells. Thus, in addition to the cells selected from a local window, NTUplace3 might additionally **randomly** select cells from the full placement region in each run of the cell mathcing.

Density Optimization

- Cell Sliding is performed to reduce the density overflow.

- All of the macro blocks are fixed.
- The algorithm iteratively reduces the densities of overflowed bins by sliding the cells from denser bins to sparser ones.
- Only horizontal sliding is implemented (left sliding and right sliding), since vertical sliding often generates misalignment between standard cells and site rows.

- Area flow

- Calculate the density of each bin.
- Compute the area flow $f_{bb'}$ between bin b and its left or right bin b'. Value $f_{bb'}$ denotes the desired amount of movable cell area to move **from** bin b **to** bin b'.

- If $D_b \leq M_b$ or $D_b/M_b \leq D_{b'}/M_{b'}$, we set $f_{bb'} = 0$.

  Otherwise we calculate the $f_{bb'}$ according to the capacity of b'. If b' has enough free space, we move the overflow area of bin b to bin b'. Otherwise, the overflow area is evenly distributed between b and b'.

Density Optimization (cont.)

- Area flow expressed mathematically

$f_{bb'} = 0.$ *if* $D_b \le M_b$ *or* $D_b/M_b \le D_{b'}/M_{b'}$

Otherwise,

$$f_{bb'} = \begin{cases} D_b - M_b, & \text{if } (M_{b'} - D_{b'}) \ge (D_b - M_b) \\ \frac{D_b M_{b'} - D_{b'} M_b}{M_b + M_{b'}}, & \text{otherwise} \end{cases} \quad (18)$$

where the second condition is derived from

$$(Db - Mb) - ((Db - Mb) + (Db' - Mb')) \cdot \left(\frac{Mb}{Mb + Mb'}\right) =$$

$$D_b - \left(M_b + \frac{(D_b - M_b + D_{b'} - M_{b'})M_b}{M_b + M_{b'}}\right) = \frac{D_b M_{b'} - D_{b'} M_b}{M_b + M_{b'}}. \quad (19)$$