

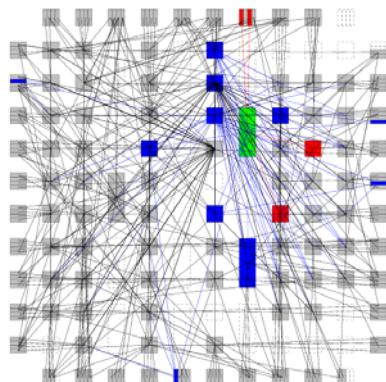
# *FPGA Placement*

1



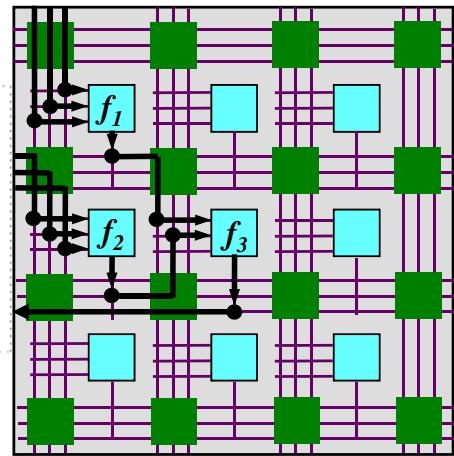
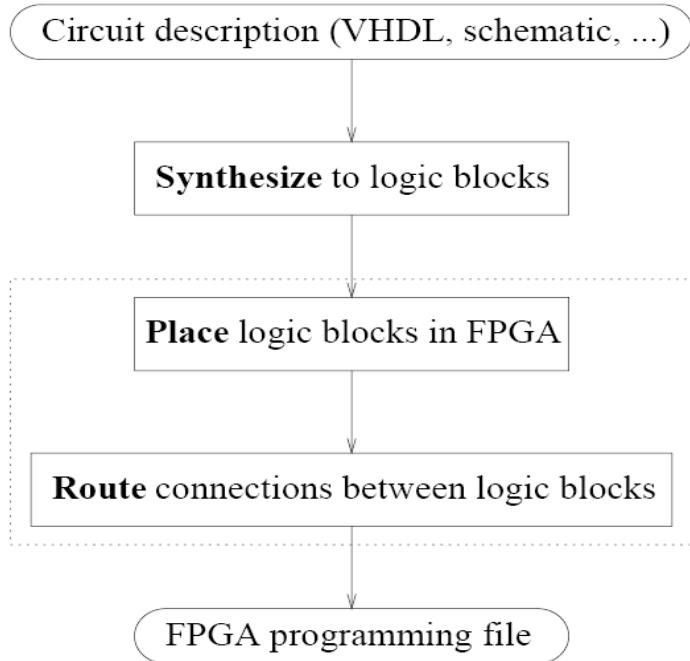
## Outline

- Constraints and objectives
- Placement approaches
  - Simulated annealing-based
  - Partitioning-based
  - Analytical method
- Representative placers
  - VPR placer (SA-based)
  - PPFF (partitioning-based)
  - RAAAP (analytical)



2

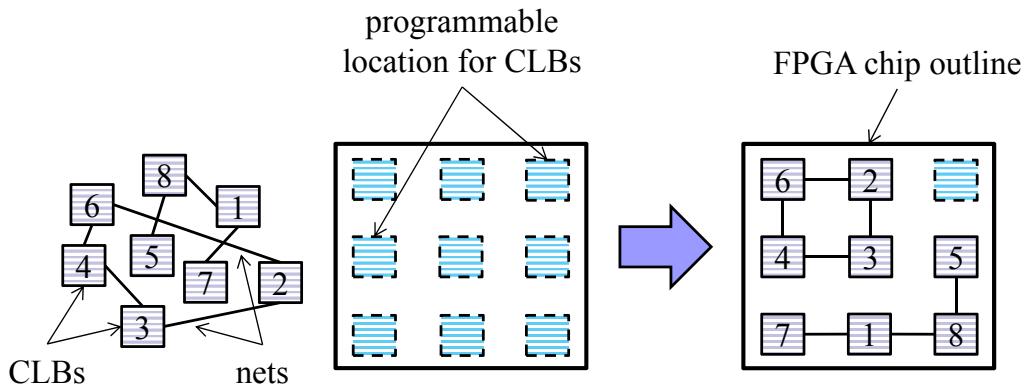
# FPGA CAD Flow



3

## FPGA Placement

- Goal: Place each logic block of a netlist to a unique and legal position on an FPGA



4

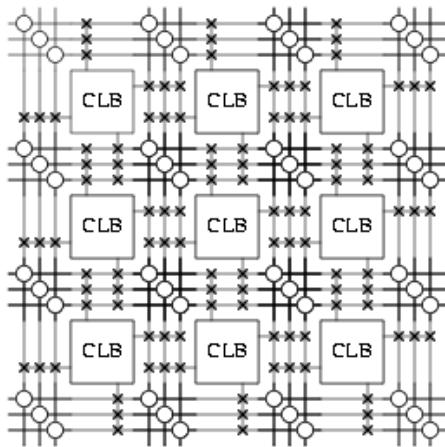
# Placement Objectives

- Minimize the required wiring
  - *wirelength-driven* placement.
- Balance the wiring density across the FPGA
  - *routability-driven* placement.
- Maximize circuit speed
  - *timing-driven* placement.

wire length driven  
Shorter WL 、 lower Power 、 faster Speed 、 少部分影響routability  
Routability driven  
wire density才是主要針對routability

5

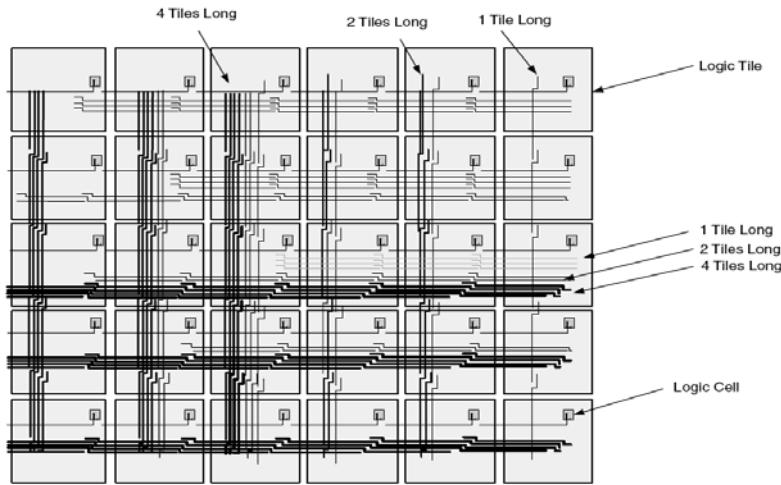
# FPGA-Specific Placement Issues



- Has to deal with *fixed carrier dimensions*.
- *Channel density* in every channel cannot exceed the number of routing tracks available.
  - How to estimate the channel density?

6

# FPGA-Specific Placement Issues



- FPGA contains routing tracks of various lengths.
  - How to estimate interconnection delay?
- Simple interconnection *delay estimation* model based on net length or fanout are not accurate for FPGA.

7

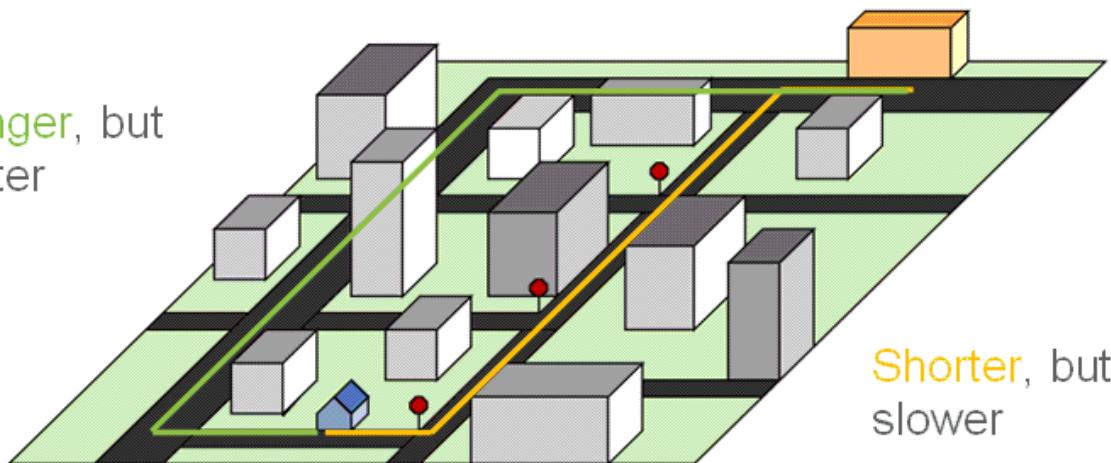
## On Delay Estimation



Which is the best route from home to the office?

delay不只看距離

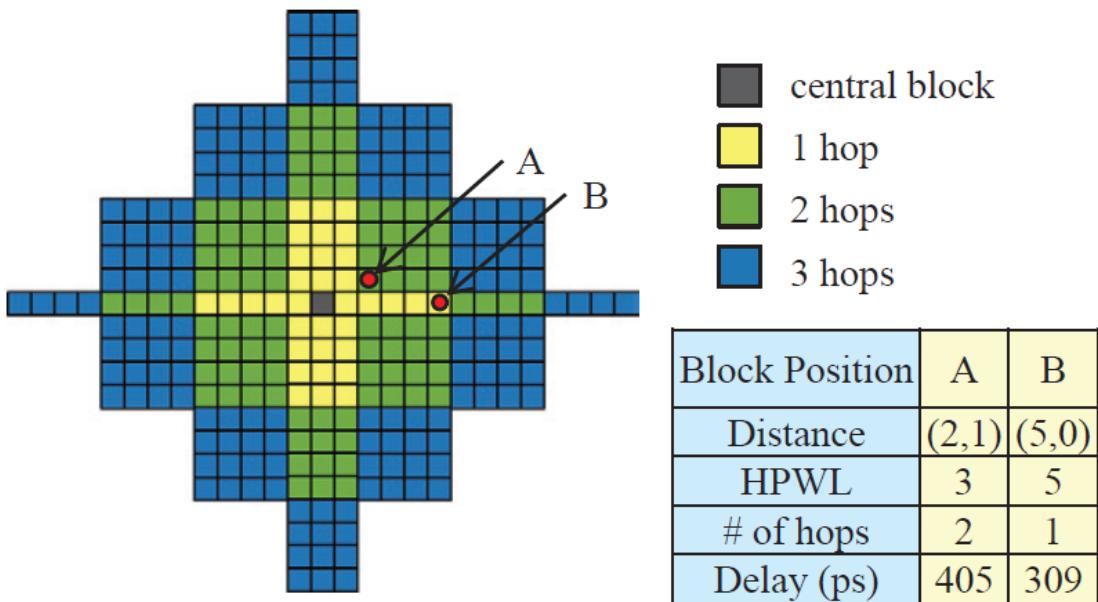
Longer, but faster



Source: Synplicity

8

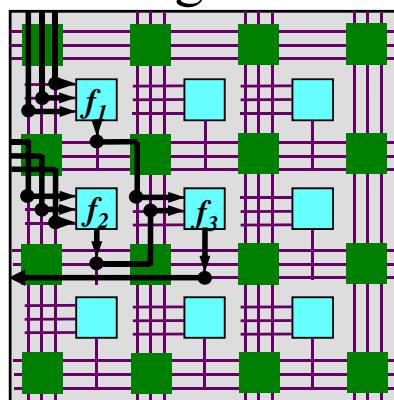
# On Delay Estimation



9

# Close Relation with Routing

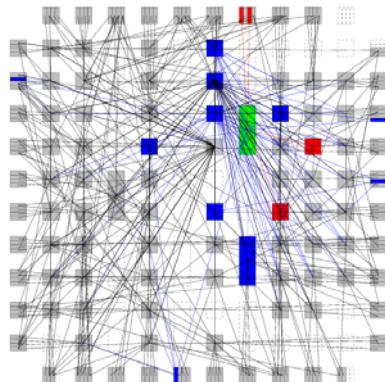
- Ideally, placement and routing (P & R) should be performed simultaneously as they depend on each other's results.
- In practice, placement is done prior to routing as simultaneous placement and routing is too complicated.



10

# Placement Approaches

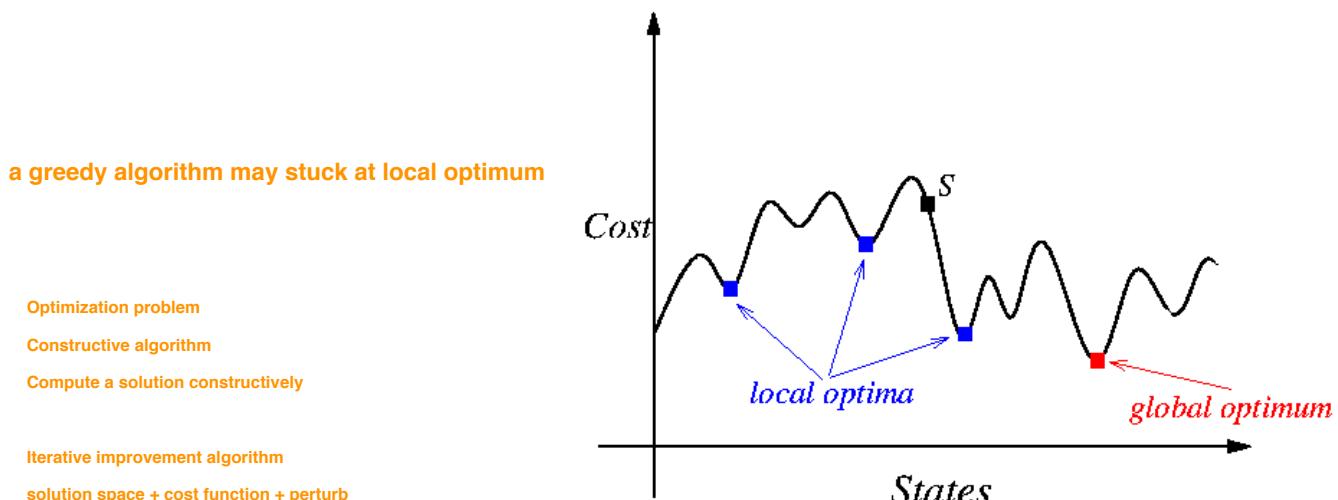
- 3 major classes of placers:
  - *Simulated annealing* based placers
  - *Min-cut (partitioning-based)* placers
  - *Analytic* placers



11

# Simulated Annealing

- Kirkpatrick, Gelatt, and Vecchi,  
“Optimization by simulated annealing”,  
*Science*, May 1983.



12

## Simulated Annealing Basics

- Simulated annealing (SA) is a *stochastic search technique* to search for a near optimal solution for an optimization problem.
- SA mimics the annealing process used to gradually cool molten metal to produce a high-quality crystal structure.
- SA takes an existing solution and then makes successive changes in a series of random moves.
- Each move is accepted or rejected based on an *energy function*.

13

## Simulated Annealing Basics

- Non-zero probability for “*up-hill*” moves.
- Probability depends on
  1. magnitude of the “up-hill” movement
  2. temperature

$$Prob(S \rightarrow S') = \begin{cases} 1 & \text{if } \Delta C \leq 0 \quad /* "down-hill" moves */ \\ e^{-\frac{\Delta C}{T}} & \text{if } \Delta C > 0 \quad /* "up-hill" moves */ \end{cases}$$

- $\Delta C = cost(S') - Cost(S)$
- $T$ : Control parameter (*temperature*)
- Annealing schedule:
  - $T = T_0, T_1, T_2, \dots$ , where  $T_i = r^i T_0$ ,  $r < 1$ .

14

# Generic Simulated Annealing Algorithm

```
1 begin
2 Get an initial solution  $S$ ;
3 Get an initial temperature  $T > 0$ ;
4 while not yet “frozen” do
5   for  $1 \leq i \leq P$  do
6     Pick a random neighbor  $S'$  of  $S$ ;
7      $\Delta \leftarrow \text{cost}(S') - \text{cost}(S)$ ;
     /* downhill move */
8     if  $\Delta \leq 0$  then  $S \leftarrow S'$ 
     /* uphill move */
9     if  $\Delta > 0$  then  $S \leftarrow S'$  with probability  $e^{-\frac{\Delta}{T}}$  ;
10     $T \leftarrow rT$ ; /* reduce temperature */
11  return  $S$ 
12 end
```

15

## Basic Ingredients for Simulated Annealing

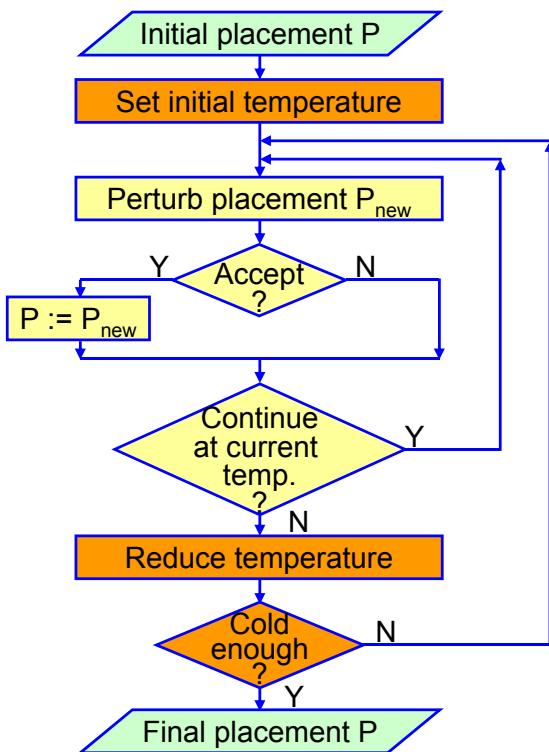
### ■ Analogy:

Physical system	Optimization problem
state	configuration
energy	cost function
ground state	optimal solution
quenching	iterative improvement
careful annealing	simulated annealing

- Basic Ingredients for Simulated Annealing:
  - Solution space
  - Neighborhood structure
  - Cost function
  - Annealing schedule

16

# Simulated Annealing Based Placement

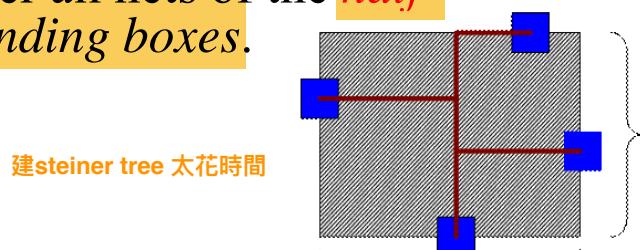


17

# Simulated Annealing Based Placement

- Get initial placement by assigning logic blocks of the circuit randomly to the available locations in the FPGA.
- Cost function

- For wirelength-driven placement, a common cost function is the sum over all nets of the **half-perimeter** of their *bounding boxes*.



- Move types
  - Exchange locations of two randomly selected logic blocks
  - Move a logic block to an empty location

18

# VPR Placer (SA-based)

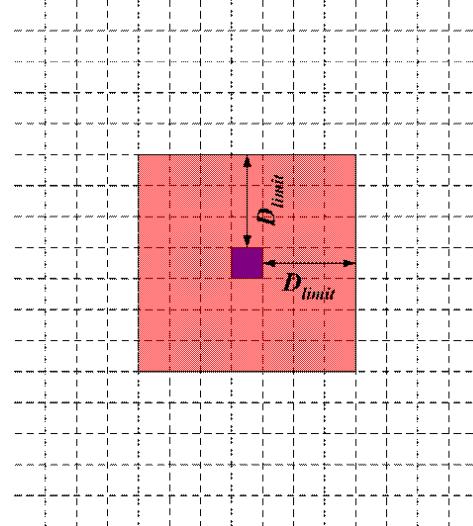
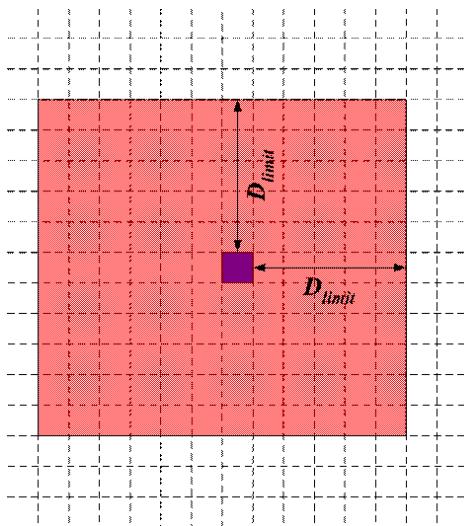
- VPR: versatile place and route
- A *simulated annealing-based* placer.
- *Modified temperature updating scheme*
  - Accelerated temperature decrease when move acceptance rate  $\alpha$  is very high or low
  - $T_{new} = r T_{old}$        $r = \begin{cases} 0.5 & \text{if } \alpha > 0.95 \\ 0.9 & \text{if } 0.8 < \alpha \leq 0.95 \\ 0.95 & \text{if } 0.15 < \alpha \leq 0.8 \\ 0.8 & \text{if } \alpha \leq 0.15 \end{cases}$
- Fast *incremental net bounding box updating*.

算 wirelength 快一點 不用全部重算 找有變得重算就好

19

# VPR Placer

- Gradually reduce the scope of cell moves or exchanges (long range to short range) to *keep the move acceptance rate* close to 0.44.



20

# VPR Placer

- *Congestion model* for non-uniform channel capacity (penalize solutions requiring more routing in the narrower channels).
- Cost function of basic mode:

$$Cost = \sum_{n=1}^{N_{nets}} q(n) \left[ \frac{bb_x(n)}{C_{av,x}(n)} + \frac{bb_y(n)}{C_{av,y}(n)} \right]$$

- $bb_x(n)$ : horizontal span of net n
- $bb_y(n)$ : vertical span of net n
- $q(n)$ : *adjustment factor* depending on #terminals of net n

$q(n) = 1$  if n has  $\leq 4$  terminal

.....

$q(n) = 2.79$  if n has  $\geq 50$  terminal

21

# Timing-Driven VPR Placer

- Add a timing cost to the objective function
- Timing cost is the *weighted delays* of all connections (the weight reflects a connection's *criticality*)
  - $w(e) = (1 - slack(e)/T)^\beta$  where  $T$  is the current longest path delay,  $\beta$  is a constant
- *Self-normalization*: changes of timing cost and wirelength of a move are normalized by their previous values
 
$$\text{normalize} \Rightarrow (a * \Delta\text{timing cost} / \text{previous timing cost}) + (b * \Delta\text{WL} / \text{previous WL})$$

$slack(e)$  : how much e can be slowed down  
without degrading the desired timing performance of the circuit

if  $slack(e) == 0$  e ∈ critical path

timing cost =  $\sum [w(e) * delay(e)]$

Approaches for timing-driven placement:

1. try to reduce critical path delay<sup>22</sup>

2. try to reduce a weighted sum of net delay

## Timing-Driven VPR Placer

- Use pre-computed *delay lookup matrix* – delay for  $(\Delta x, \Delta y)$
- Run *static timing analysis* periodically to update slacks and hence criticalities

23

## Partitioning-based Placement

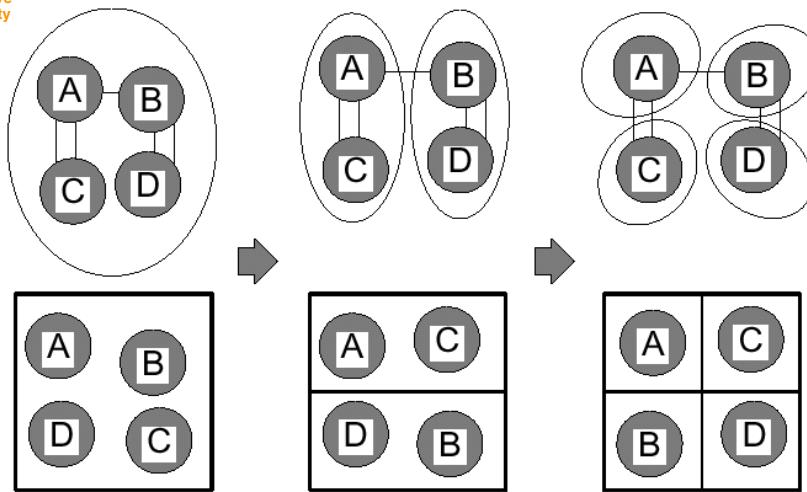
- The netlist is repeatedly partitioned into two sub-circuits.
- Meanwhile, the chip area is *partitioned* into two sub-regions *recursively*.
- Each sub-circuit is assigned to a sub-region.
- The process is repeated until each sub-circuit consists of a single logic block and has a unique location on the chip area.

# Partitioning-based Placement

- During partitioning, *minimize # cut nets* based on the intuition that densely connected sub-circuits should be placed closely.

SA starts with some initial placement try to improve the placement gradually with hill-climbing capability

Partitioning-based(min-cut-based) placement construct a good placement solution directly using a divide and conquer strategy

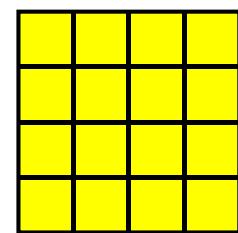


25

## PPFF Placer (partitioning-based)

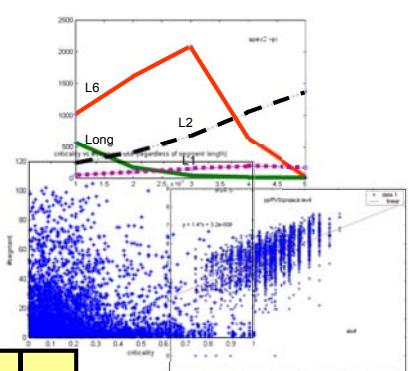
- Top-down partitioning-based* placement

- Fast (divide&conquer)



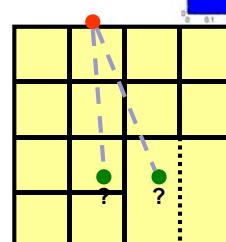
- Delay estimation

- Empirical routing delay analysis
- Correlate net criticality and routing resource usage



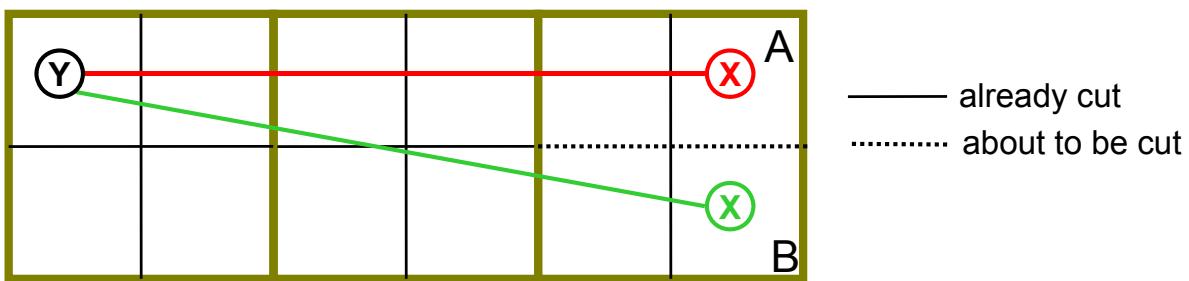
- Delay optimization

- Align critical connections
- Dynamically update net delays
- Net weight to represent timing criticality



26

## Net Terminal Alignment in PPFF

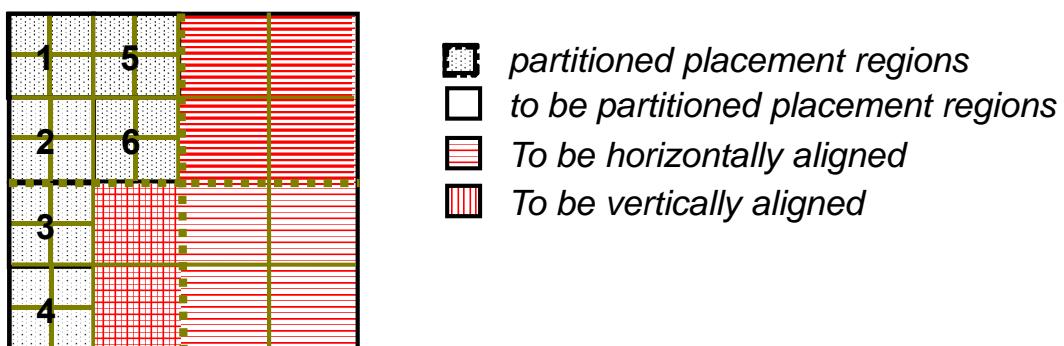


- Put node X into subregion A or B?
- Alignment helps reduce # routing segments
- Y treated as anchor point
- **X → A preferred** (one segment possible)

© Kia Bazargan

## Partitioning Order in PPFF

- Top-to-bottom and left-to-right.
- Alignment dependence “propagates”

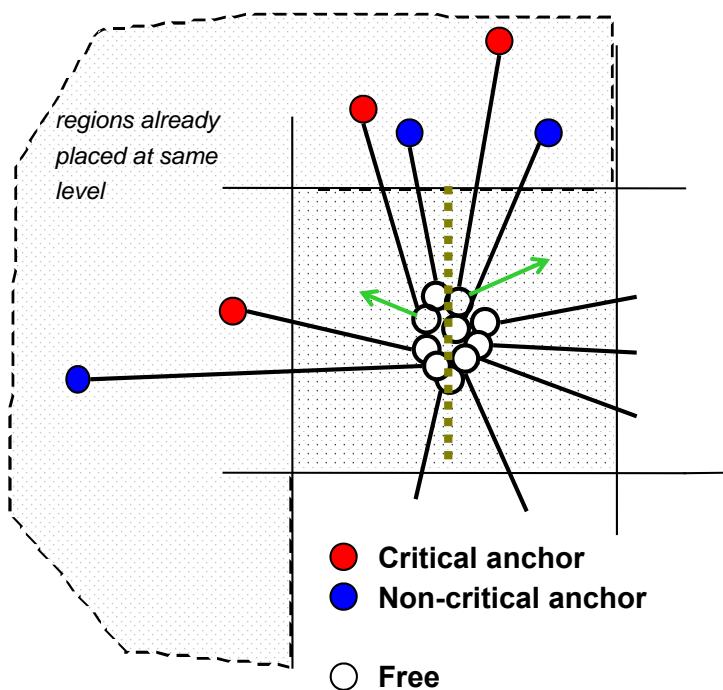


- Also proposed a more sophisticated method to determine a good partitioning order.

© Kia Bazargan

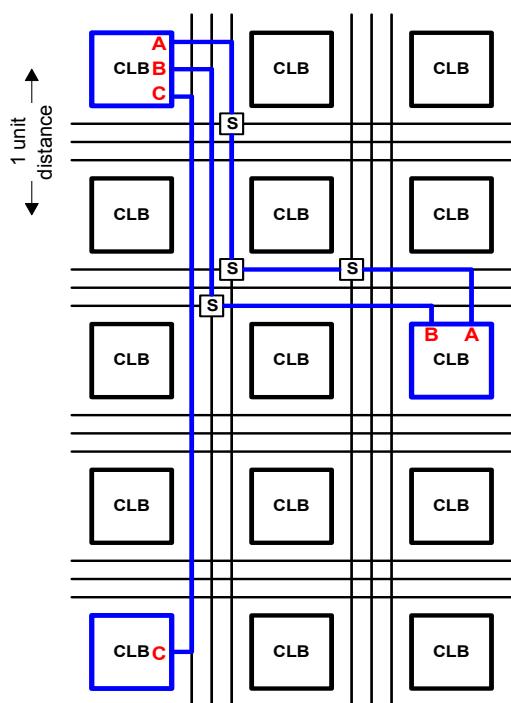
# Alignment Details

- Higher priority for timing critical node to align with its anchor in the direction of a cut



29

# Delay Estimation



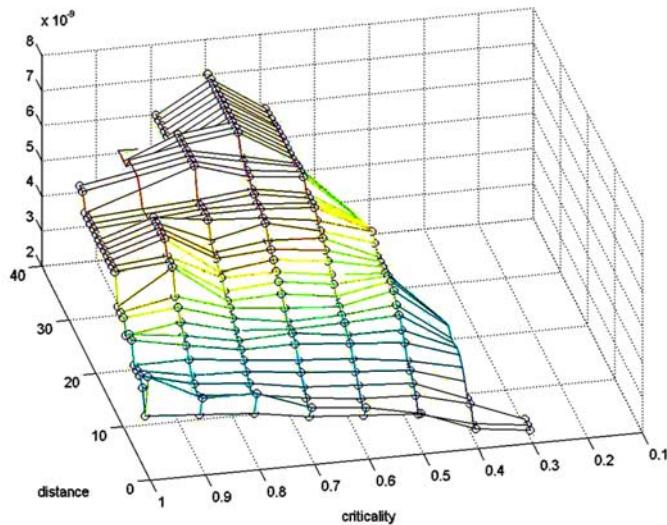
- Wire length / bounding box **NOT** accurate
  - Same BBox, fewer segs → smaller delay (e.g. A vs B)

30

# PPFF Delay Estimation

## ■ Delay estimation

- *Correlate delay with distance and criticality*
- Pre-computed table based on empirical data by a router on a large no. of circuits



- Data shows delay as a function of net criticality and distance
- Delay is net delay after VPR routing
- Each point is the average over all nets falling in the (dist, crit) range.

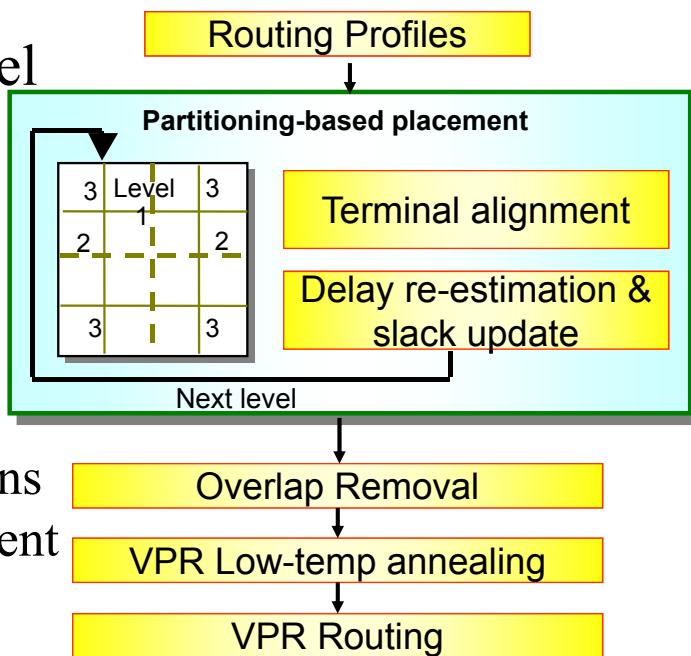
© Kia Bazargan

# Edge Weight

- Each multi-terminal net is decomposed into multiple 2-terminal nets (edges)
- Edge weight indicates its timing criticality
  - large edge weight will discourage partitioner to cut it
- At the end of each partitioning level
  - run static timing analysis to update slack of each edge
  - update criticality or weight of each edge based on updated slacks

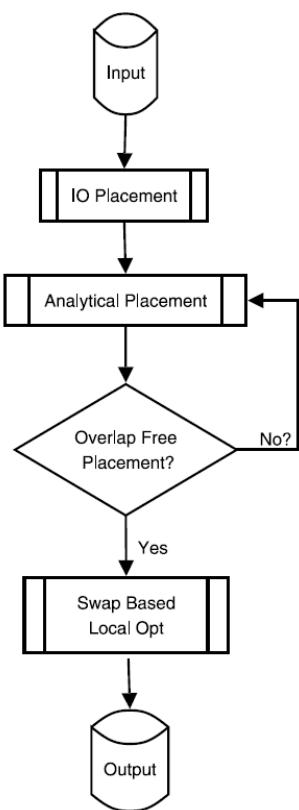
# PPFF Summary

- Initial analysis
- At each partitioning level
  - terminal alignment
  - delay re-estimation and slack update
- Refinement
  - Move non-critical CLBs from overcrowded regions while preserving alignment
  - *Low-temperature annealing* to further optimize the placement



© Kia Bazargan

# Analytical Placement

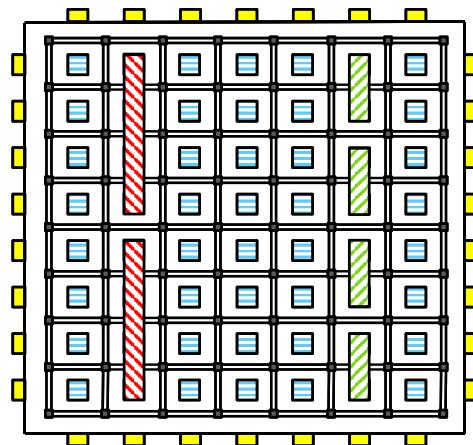


- Used in Xilinx's placer
- Represent placement objective and constraints as analytical functions of  $x_i$  and  $y_i$
- Constraints are modified iteratively to move logic blocks away from highly congested regions
- After a legal placement is found, result is further optimized by local swap

# Heterogeneous FPGA

- Traditionally, FPGA consists of CLB, IO, and programmable routing arch
- Nowadays, main FPGA components:
  - Configurable logic block (CLB 
  - Input/output block (IOB 
  - Programmable routing architecture
  - Random access memory (RAM 
  - Digital signal processing (DSP 

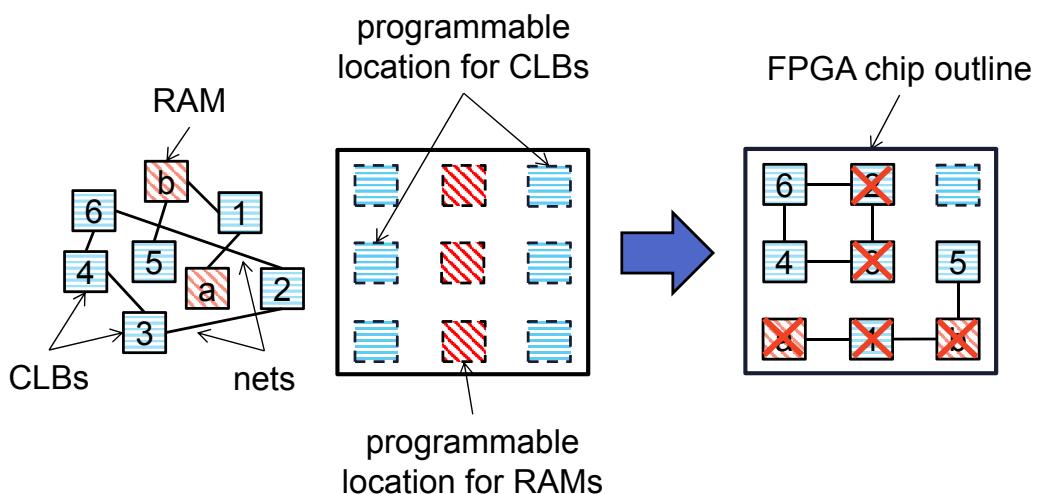
SA slow  
Partition based may not have good performance



35

# Heterogeneous FPGA Placement

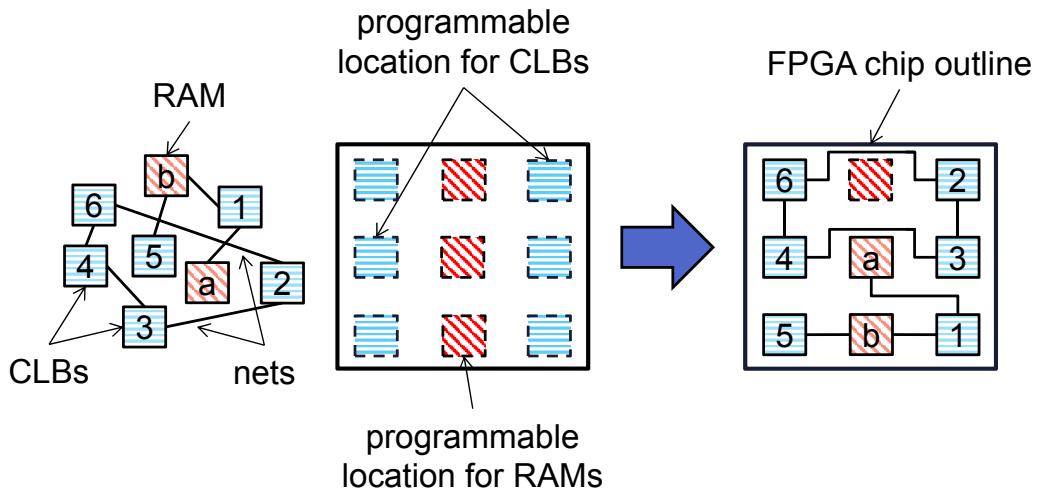
- Place each block of a netlist to a unique and legal position on an FPGA



36

# Heterogeneous FPGA Placement

- Place each block of a netlist to a unique and legal position on an FPGA



37

we want to avoid introducing a huge number of non-overlapping constraints between the blocks and a large number of binary variables.

## Analytical Placement Formulation

- Given the chip region and block dimensions, determine  $(x, y)$  for all movable blocks

A diagram of an FPGA chip showing a grid of bins. A green block is highlighted within one of the bins. The text 'bin' is labeled next to a dashed box indicating a bin boundary.

$$\begin{aligned}
 & \text{min } W(x, y) // \text{wirelength function} \\
 & \text{s.t. } D_b(x, y) \leq M_b
 \end{aligned}$$

$D_b$ : density for bin  $b$   
 $M_b$ : max density for bin  $b$

Density =  $\frac{A_{\text{block}}}{A_{\text{bin}}}$

- Relax the constraints into the objective function (penalty)

$$\min W(x, y) + \lambda \sum (\max(D_b(x, y) - M_b, 0))^2$$

- Apply **differentiable** wirelength and density models
- Use the gradient method to solve the optimization problem
- Increase  $\lambda$  gradually to meet density constraints

38

## Differentiable Wirelength Model

$$r^* \log(e^{(y_1/r)} + e^{(y_2/r)} + e^{(y_3/r)} + e^{(y_4/r)}) \text{ 大約 } \max(y_1, y_2, y_3, y_4)$$

$$r^* \log(e^{-(y_1/r)} + e^{-(y_2/r)} + e^{-(y_3/r)} + e^{-(y_4/r)}) \text{ 大約 } -\min(y_1, y_2, y_3, y_4)$$

- Log-sum-exp wirelength model [Naylor et al., 2001]

$$\gamma \sum_{e \in E} \left( \log \sum_{v_k \in e} \exp(x_k/\gamma) + \log \sum_{v_k \in e} \exp(-x_k/\gamma) + \right. \\ \left. \log \sum_{v_k \in e} \exp(y_k/\gamma) + \log \sum_{v_k \in e} \exp(-y_k/\gamma) \right).$$

- Is an effective **differentiable** function for HPWL approximation and approaches exact HPWL when  $\gamma \rightarrow 0$

39

## Differentiable Density Model

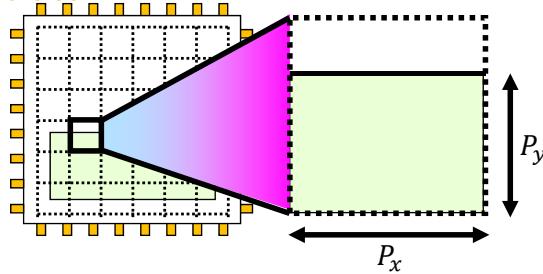
- Bell-shaped density model [Kahng et al., ICCAD'04]

$p_x(b, v)$  is the overlap function of bin  $b$  and block  $v$

along the  $x$  direction as a function of

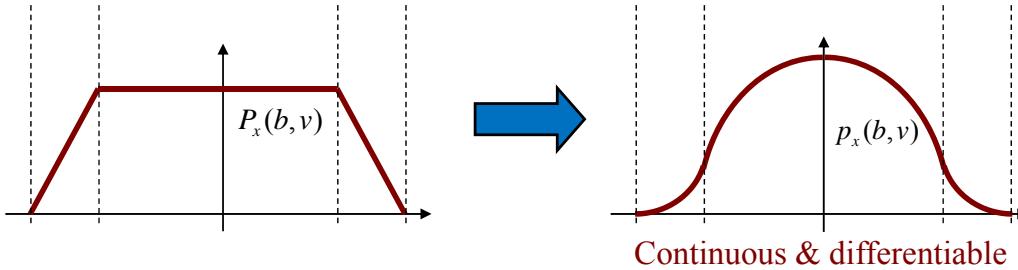
center-to-center distance

between  $b$  and  $v$



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

$$D_b'(\mathbf{x}, \mathbf{y}) = \sum_{v \in V} c_v p_x(b, v) p_y(b, v)$$

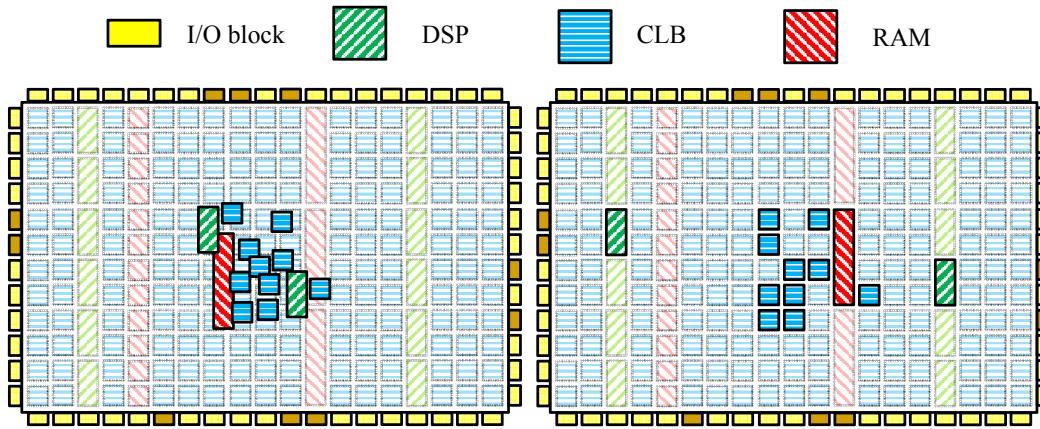


40



# Mismatch between GP and Legalization Results

- Global placement → Continuous solutions
- Legalization → Discrete and scattered legal locations
- Issue: DSP/RAM blocks get large displacement from their global placement positions



41



# Separate Density Model

- Setup a density constraint for each type of resource  
e.g.  $D^R_b(x, y) \leq M^R_b$  where  
 $D^R_b$ : RAM density in bin  $b$   
 $M^R_b$ : max RAM density in bin  $b$
- Unconstrained formulation after “smoothing” and constraint relaxation with quadratic penalty method

$$\begin{aligned} \min \quad & \hat{W}(\mathbf{x}, \mathbf{y}) + \lambda \sum_b \max(\hat{D}_b(\mathbf{x}, \mathbf{y}) - M_b, 0)^2 \\ & + \lambda_R \sum_b \max(\hat{D}_b^R(\mathbf{x}, \mathbf{y}) - M_b^R, 0)^2 \\ & + \lambda_D \sum_b \max(\hat{D}_b^D(\mathbf{x}, \mathbf{y}) - M_b^D, 0)^2, \end{aligned}$$

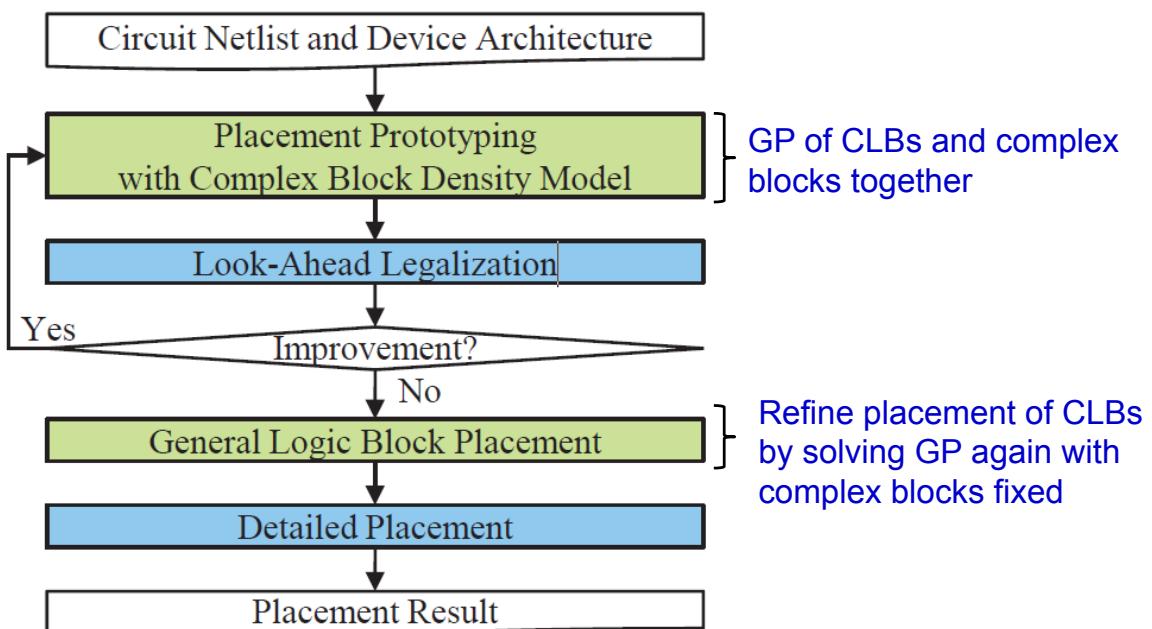
42

# Look-Ahead Legalization

- Give a quick forecast of legalized placement
  - Achieve more accurate wirelength estimation
  - Speedup the convergence of placement solutions
- RAMs and DSPs
  - Move blocks to the closest legal column
  - Apply bipartite matching for blocks and empty slots on each column
- General logic blocks
  - Apply the Tetris algorithm [Hill, 2002] which greedily packs blocks from the left to the right

43

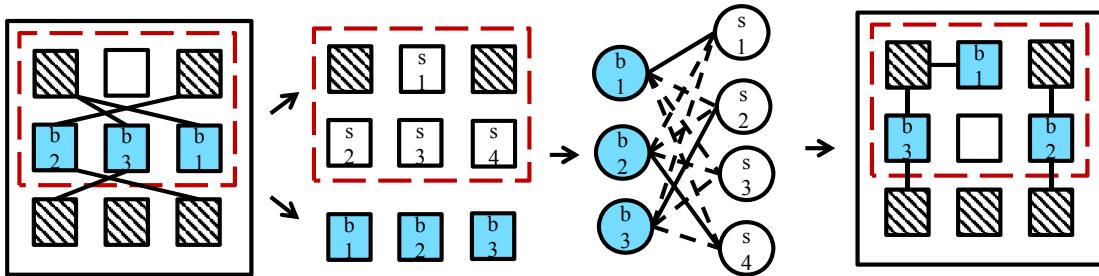
# Overall Flow



44

# Detailed placement

- Apply window-based cell matching and cell swapping [Chen et al., TCAD'08]



45

# References

- “VPR: A New Packing, Placement and Routing Tool for FPGA Research,” in *FPL’97*
- “Timing-Driven Placement for FPGAs”, in *FPGA’00*
- “Fast Timing-Driven Partitioning-based Placement for Island Style FPGAs”, in *DAC’03*
- “Architecture-Specific Packing for Virtex-5 FPGAs”, in *FPGA’08*.
- “Routing Architecture-Aware Analytical Placement for heterogeneous FPGAs”, in *DAC’15*.

46