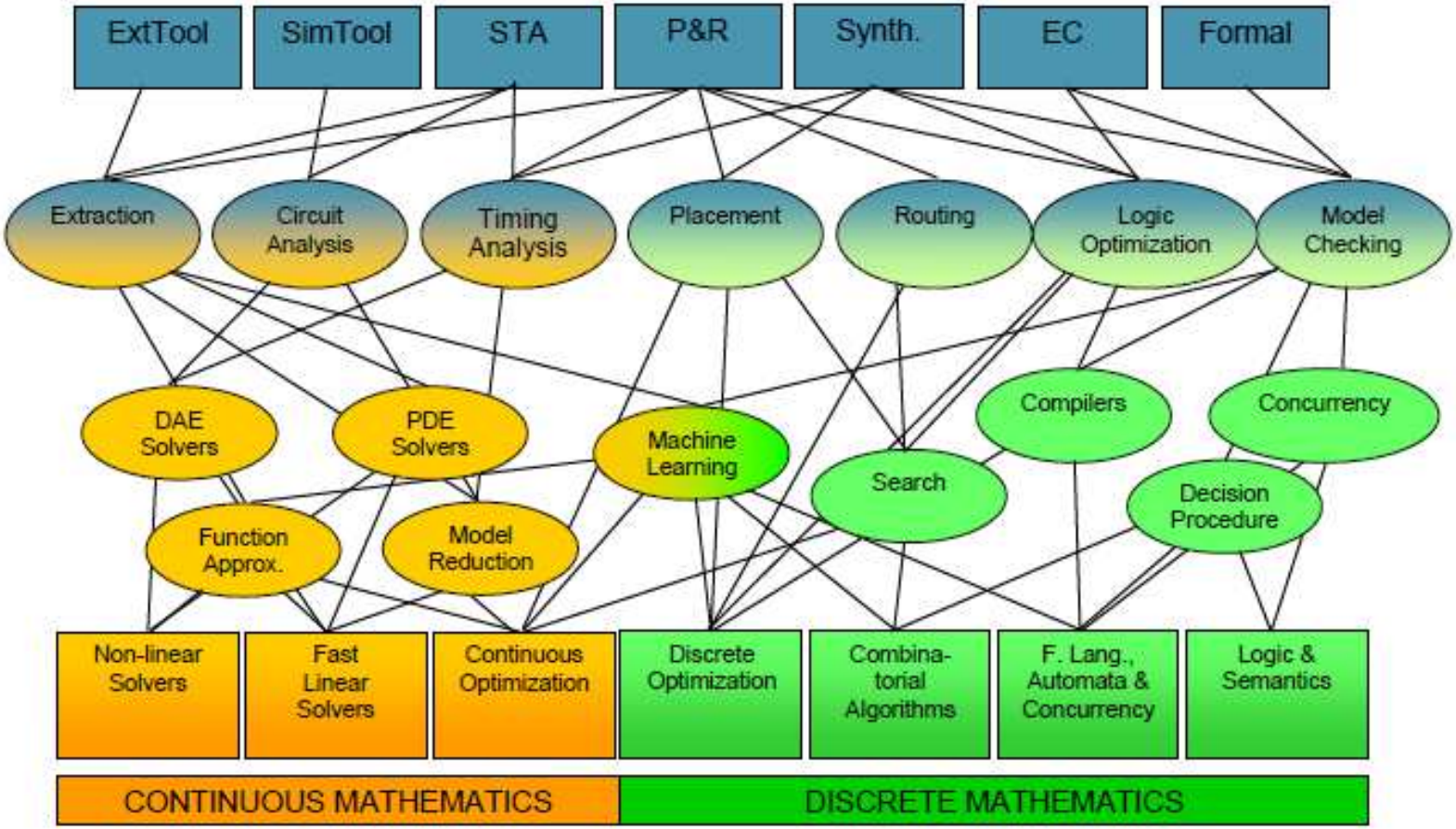
CAD Flow for FPGAs
“Introduction”

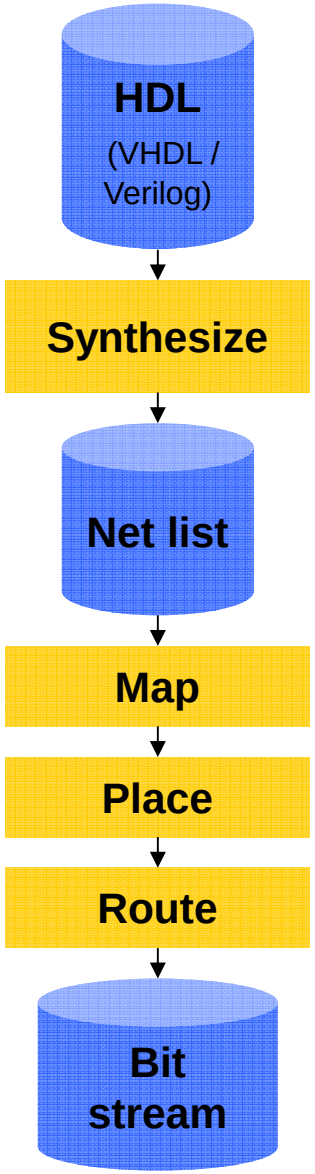
What is EDA?

- *EDA → Electronic Design Automation or (CAD)*
- *Methodologies, algorithms and tools*, which assist and automate the design, verification, and testing of electronic systems.
- A general *methodology for refining* a high-level description down to a detailed physical implementation for designs ranging from
  - Integrated circuits (including system-on-chips),
  - Field Programmable Gate Arrays,
  - Printed circuit boards (PCBs) and
  - Electronic systems.
- *Why?*
  - *Manual design is unrealistic*
  - *Fewer errors*
  - *Time to market.*

Areas & Domain Knowledge in EDA

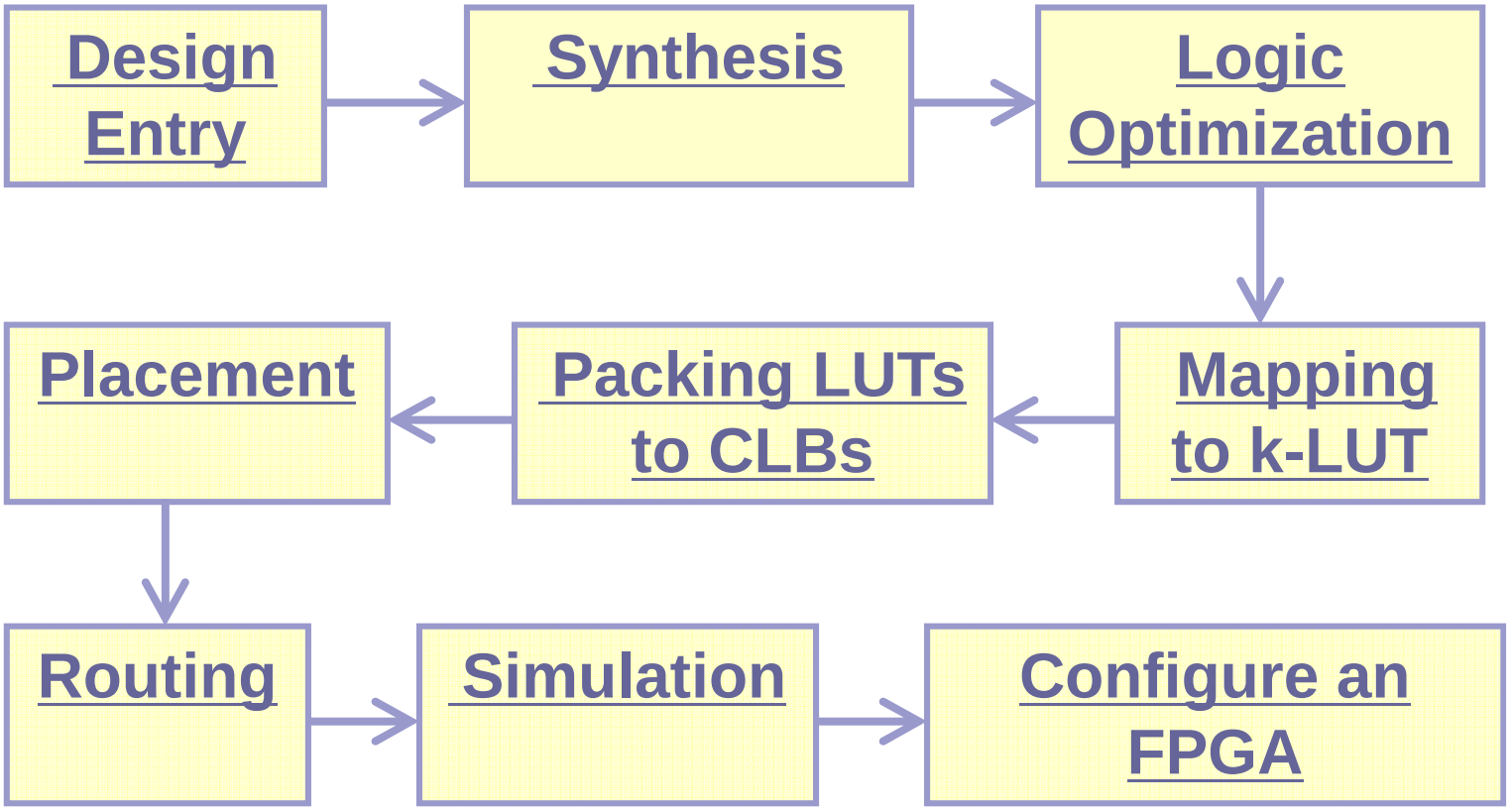


FPGA tool flow

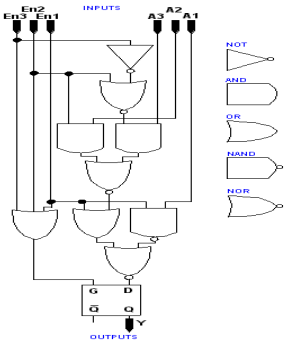


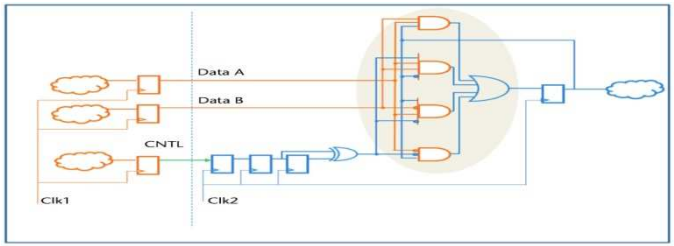
- Hardware design is traditionally done by modeling the system in a hardware description language
- An FPGA “compiler” (synthesis tool) generates a netlist
- which is then **mapped** to the FPGA technology
- the inferred components are **placed** on the chip
- and the connecting signals are **routed** through the interconnection network
- A bit stream is finally produced which can be used to program the FPGA.

EDA (CAD) Flow for FPGAs



“Synthesis”

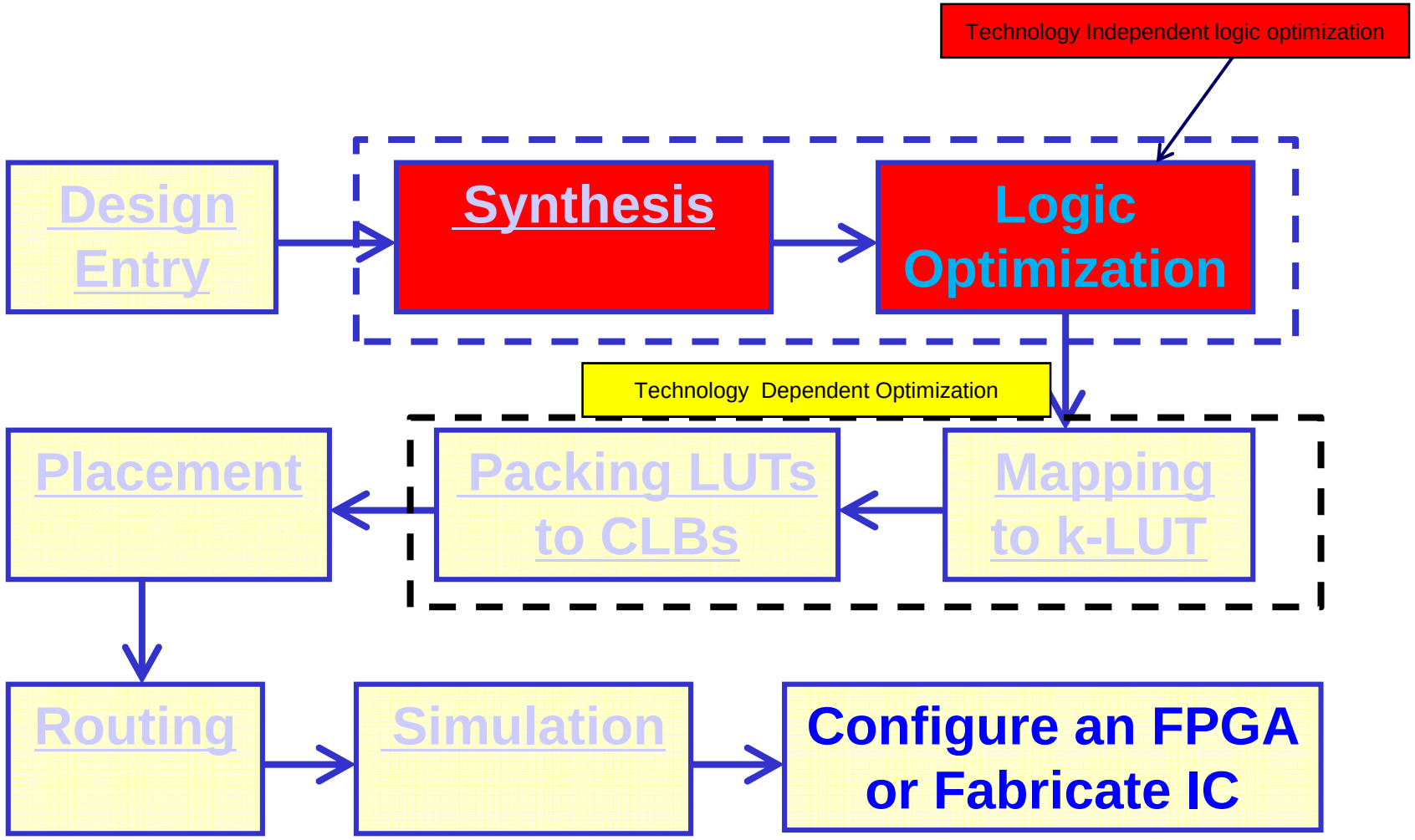




Synthesis is the process by which the system specifications and constraints ***are translated*** to an implementation (a net list of connected components).

- **Synthesis is considered to be a key stage in *automated design tools (CAD Tools).***
- **A significant area of research in EDA is in the development of tools that can synthesize hardware from a design written in the form of high-level programming language such as C.**
- **It is believed that these ``hardware compilers” will help to decrease development time, thus shortening the crucial time to market for designs.**

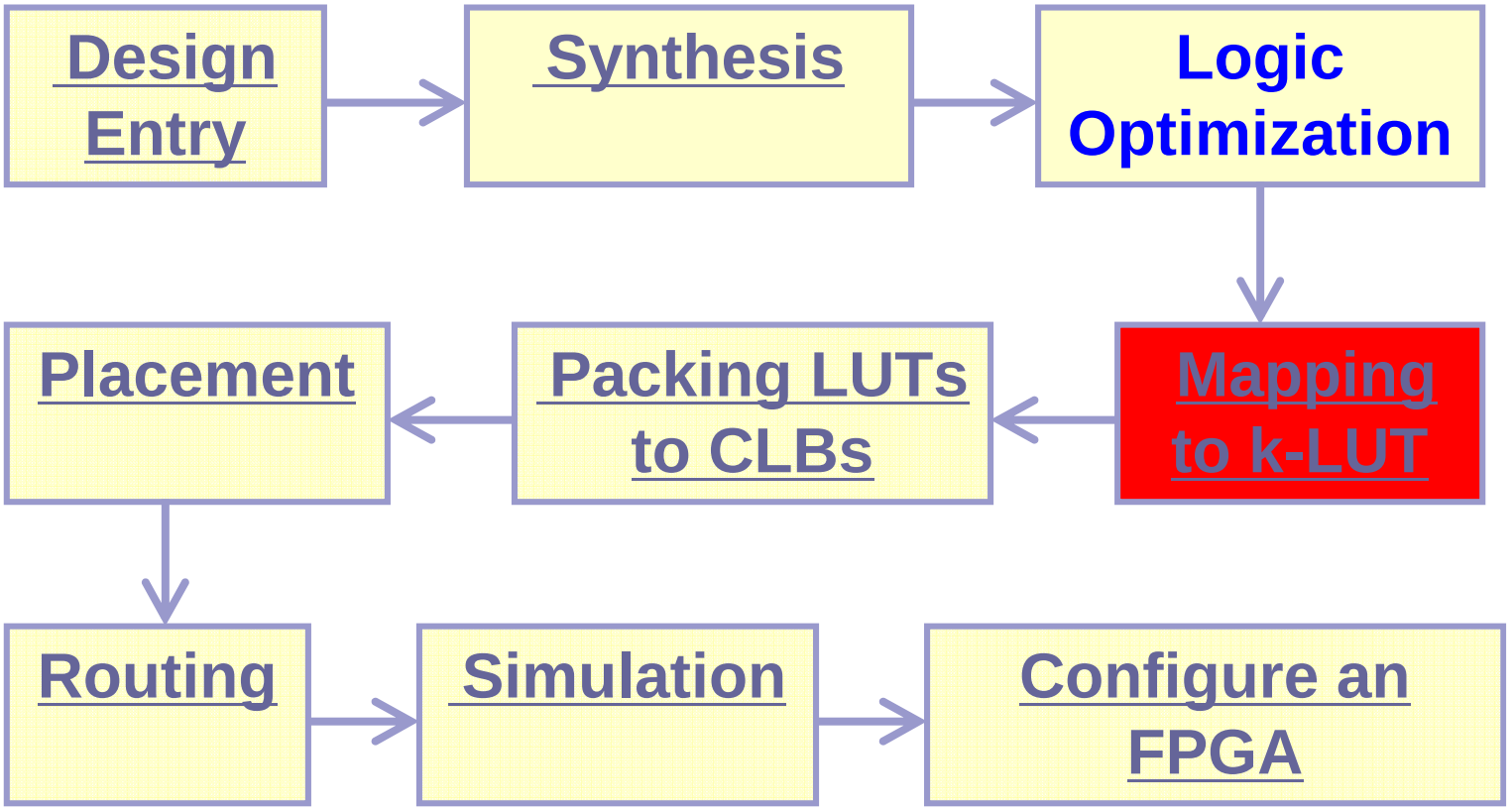
CAD for IC Design: Synthesis



Objective Function for Synthesis

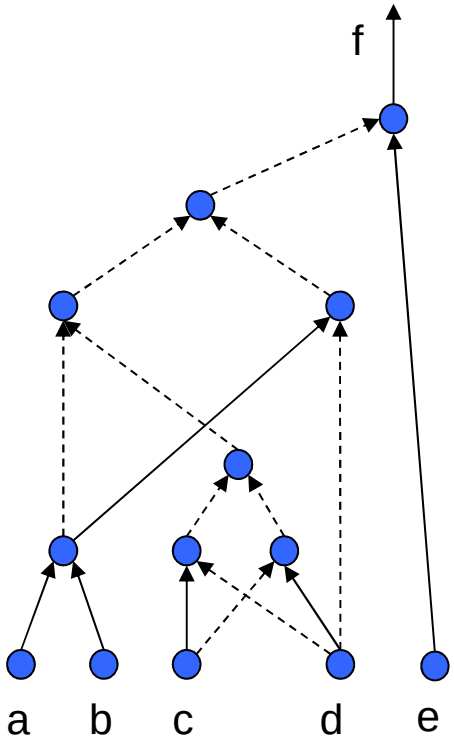
- Minimize area
  - in terms of literal count, cell count, register count, etc.
- Minimize power
  - in terms of switching activity in individual gates, deactivated circuit blocks, etc.
- Maximize performance
  - in terms of maximal clock frequency of synchronous systems, throughput for asynchronous systems
- Any combination of the above
  - combined with different weights
  - formulated as a constraint problem
    - "minimize area for a clock speed > 300MHz"
- More global objectives
  - ***feedback from layout***
    - actual physical sizes, delays, placement and routing

CAD for FPGAs: Mapping & Covering



Mapping

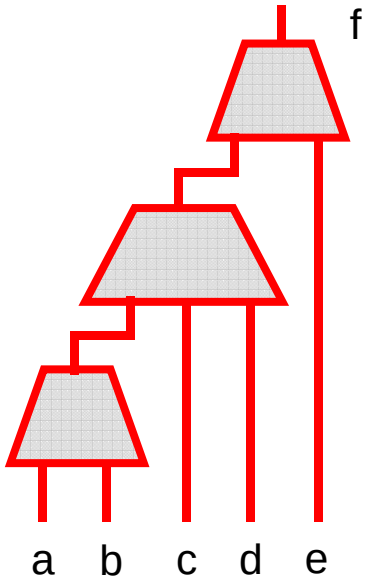
Input: A Boolean network



The subject graph

Output: A netlist of *k*-LUTs implementing the Boolean network optimizing some cost function

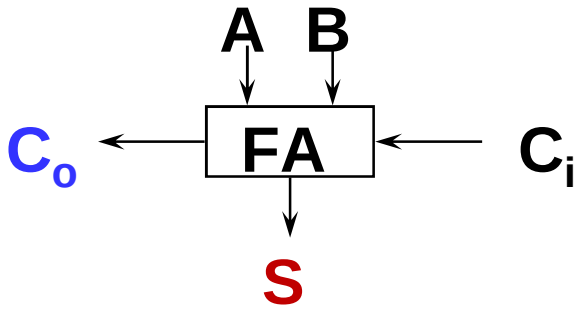
Technology Mapping



The mapped netlist

Technology Mapping: A Simple Example

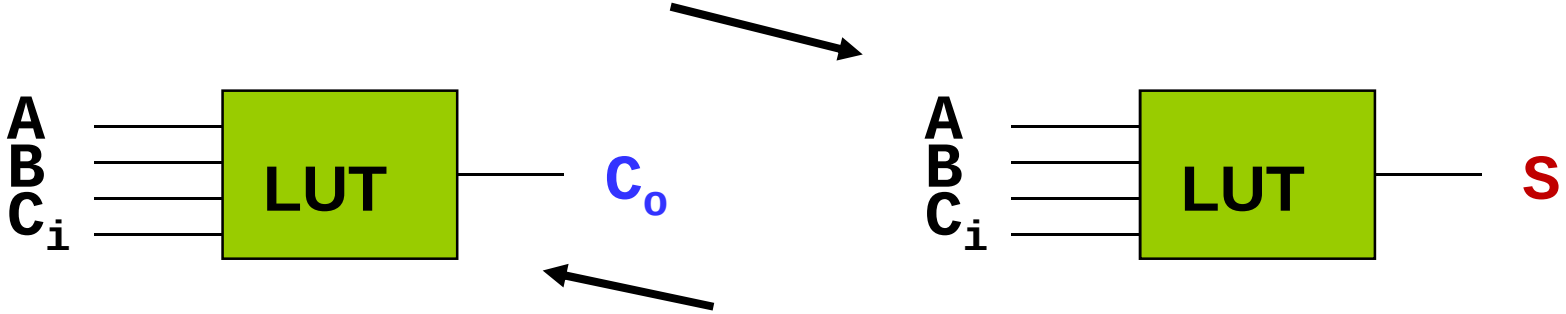
A Full Adder Implementation:



$S = A + B$

Logic synthesis tool reduces circuit to SOP form

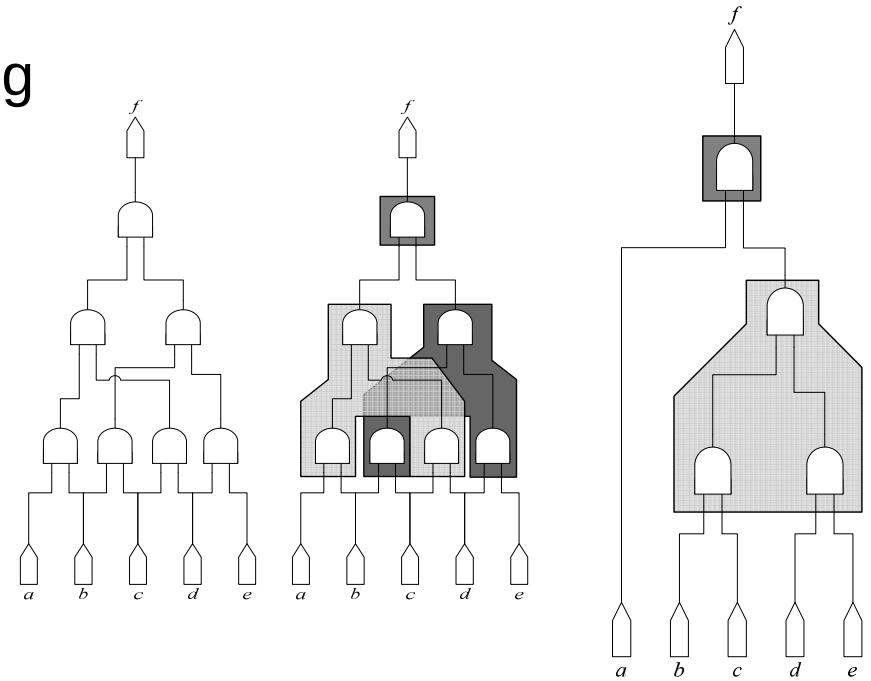
$$S = \overline{A}B\overline{C_i} + A\overline{B}\overline{C_i} + \overline{A}\overline{B}C_i + ABC_i$$



$$C_o = \overline{A}BC_i + A\overline{B}C_i + AB\overline{C_i} + ABC_i$$

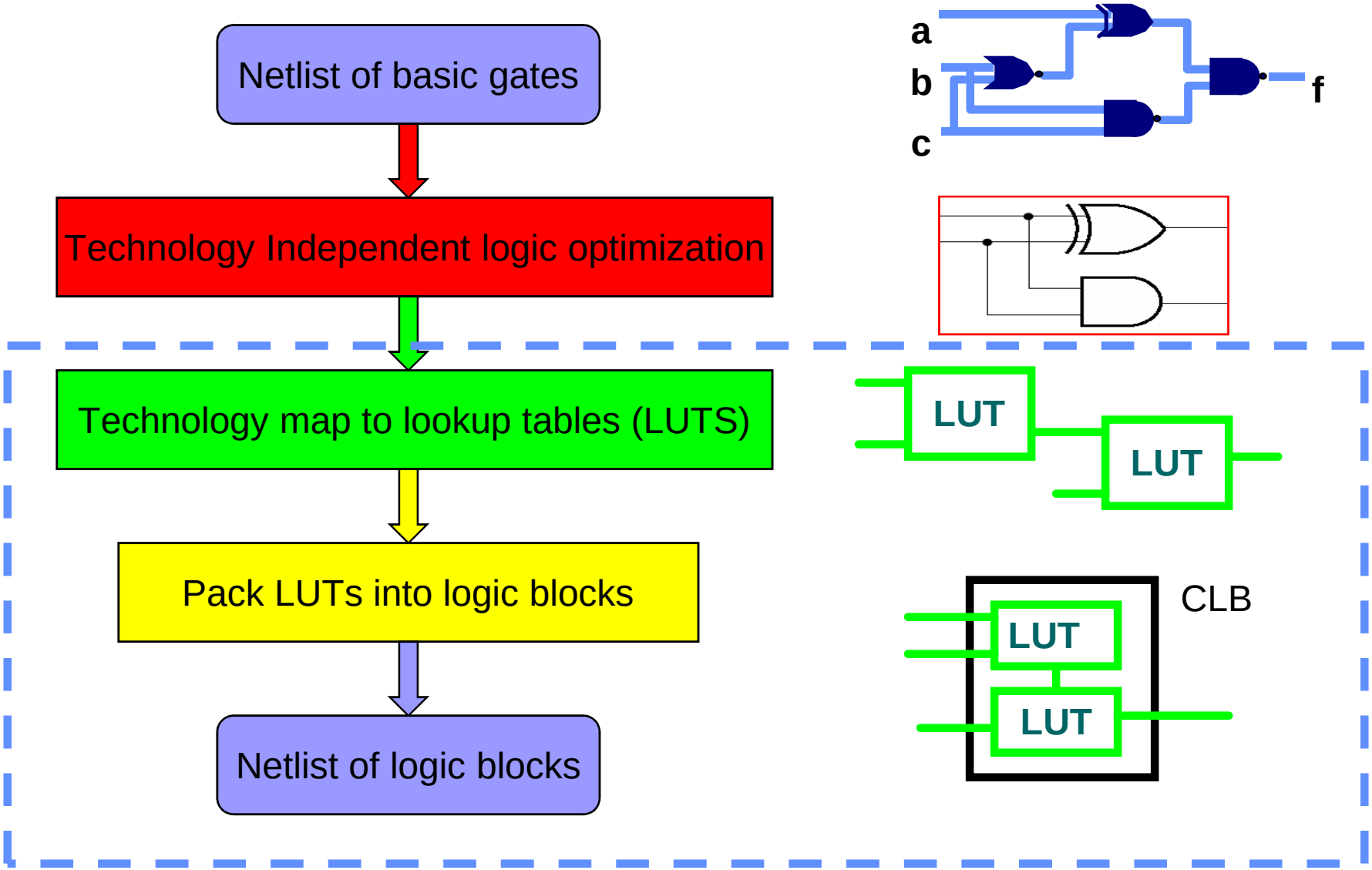
Logic Synthesis in FPGAs

- Logic Optimization
- Technology Mapping



- Definitions
  - **Technology Mapping**
  - **Logic Optimization**
  - **Logic Synthesis = Logic Optimization + Tech Mapping**

Mapping and Packing



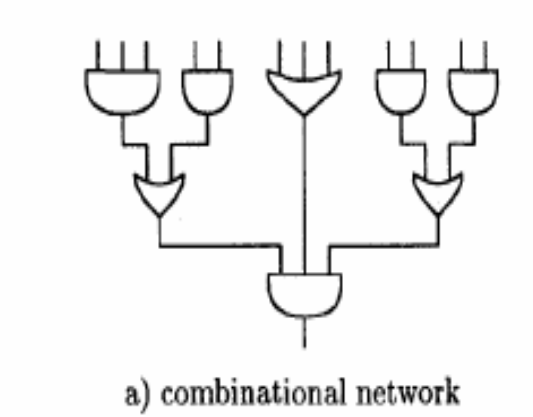
Objective Function for Mapping

- Minimize area
  - in terms of number of LUTs
- Minimize power
  - in terms of switching activity in individual LUTs.
- Maximize performance
  - in terms of connectivity (depth of LUT implementation)
- Any combination of the above (multi-objective)
  - combined with different weights

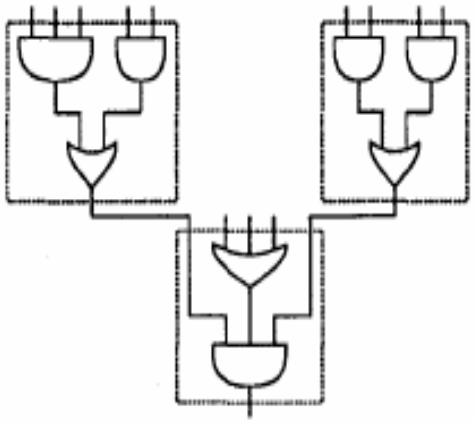
Mapping: Example

In this example, the circuit in Figure (a) can be implemented by:

- **the circuit of three *5-input lookup tables* shown in Figure (b)**
- **Notice that one of the lookup tables uses only 4 of the available 5 inputs.**

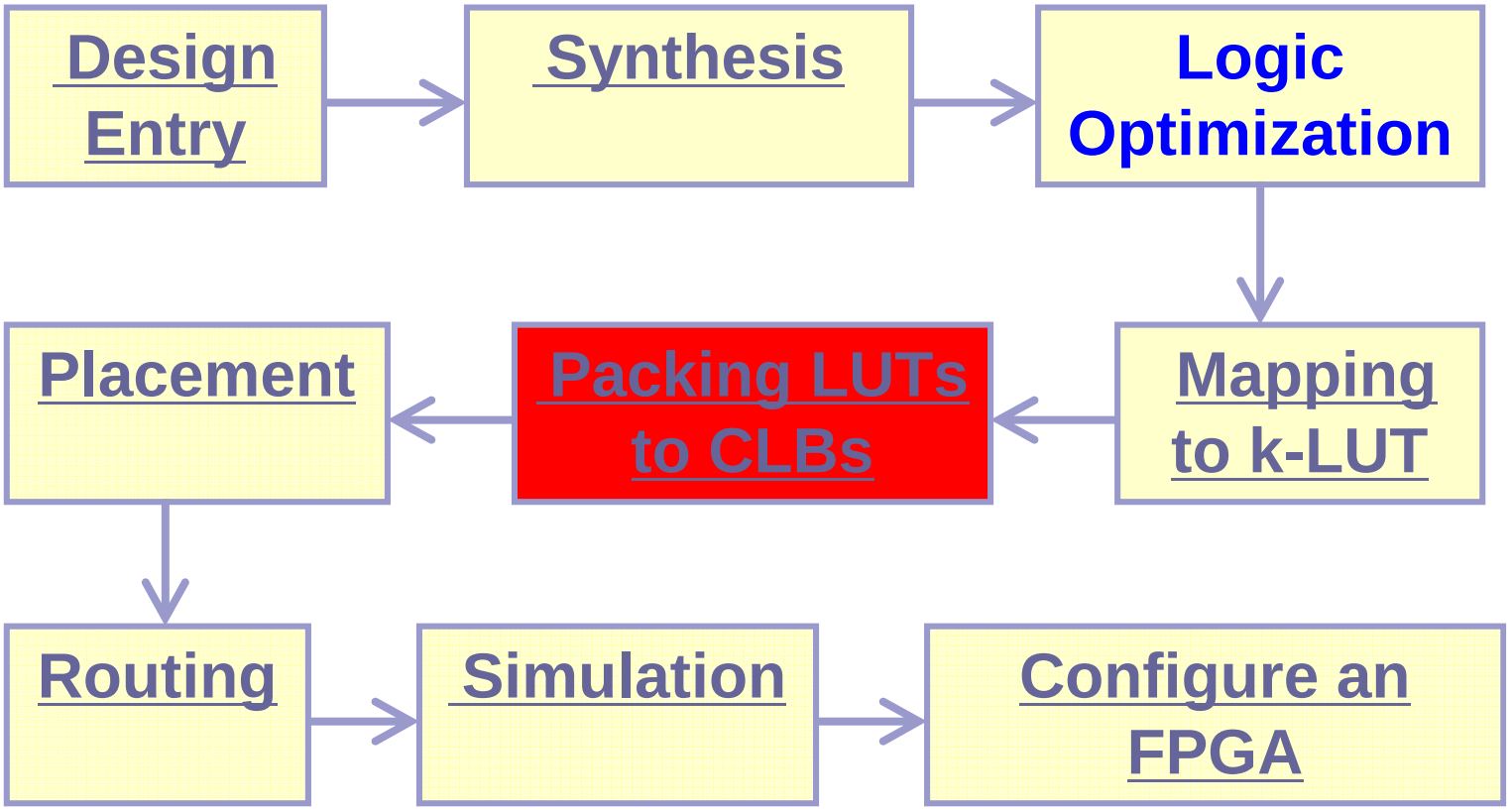


a) combinational network



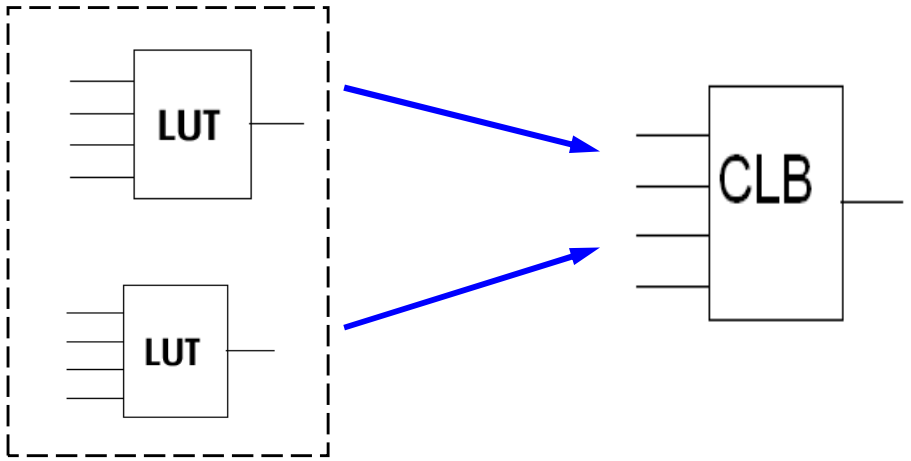
b) circuit of 5-input lookup tables

FPGA CAD Flow: Packing



Packing LUTs into CLBs

- Logic block packing groups several LUTs and registers into one logic block.
- This step is necessary whenever the FPGA logic block contains more than a single LUT.

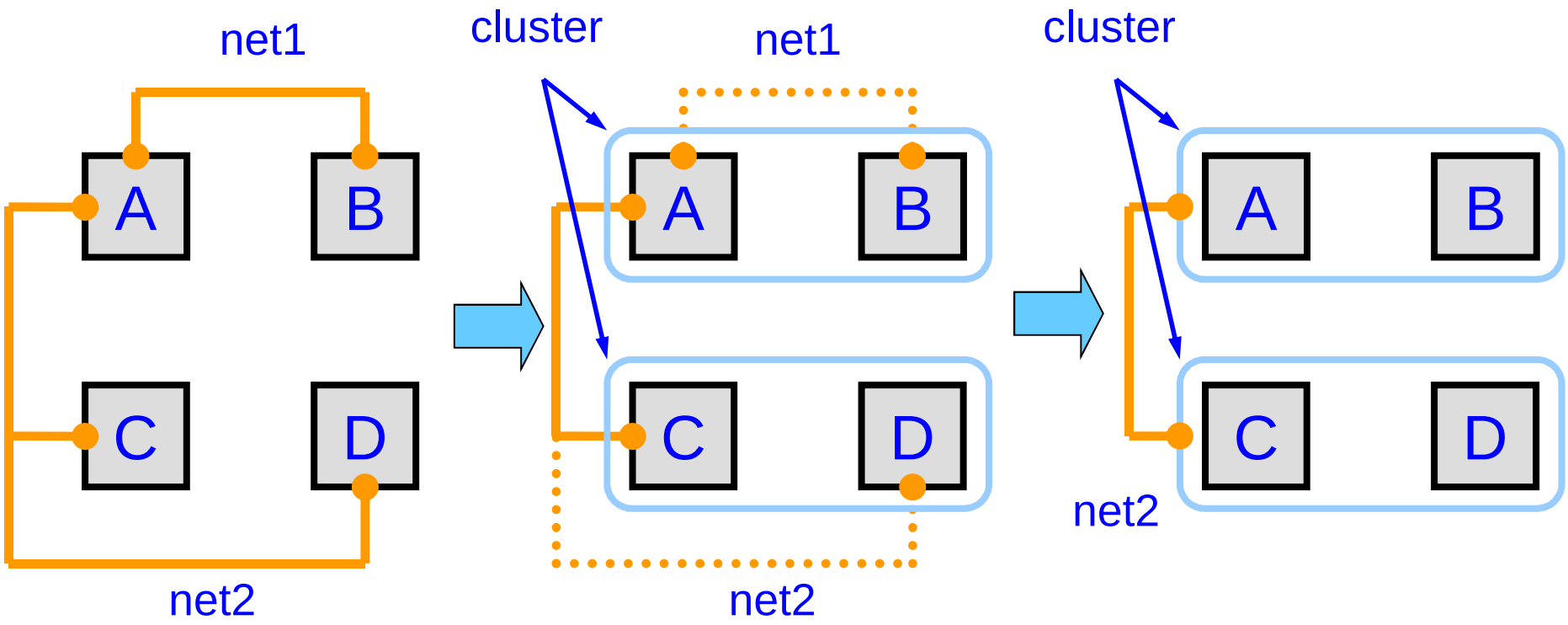


Highly connected LUTs

Packing

- Logic Block packing ***groups*** several LUTs and registers into one Configurable Logic Block (CLB), respecting limitations such as:
  i. Number of LUTs a CLB may contain
  ii. The number of distinct input signals/clocks a CLB may contain
- The optimization goals in this phase are to pack connected LUTs together to Minimize:
  i. The number of signals to be routed between logic blocks
  ii. The number of logic blocks used (by filling each logic block to its capacity)
- This problem is a form of ***clustering***.

Clustering "Packing"

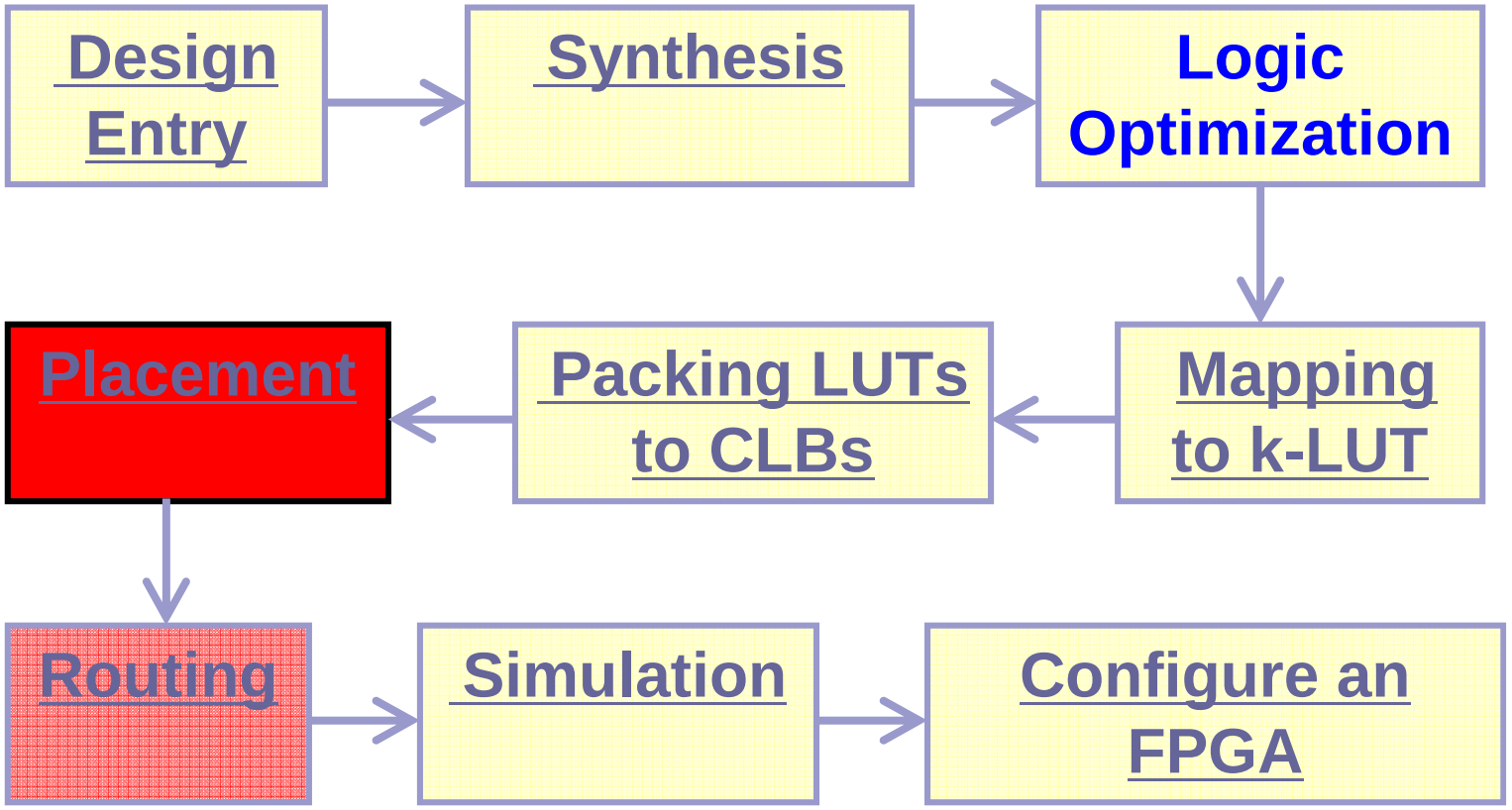


before clustering

clustering

after clustering

CAD for FPGAs: Place & Route



Problem Formulation

Given :

- Set of modules M = { m_1, m_2, ….m_n}
- Set of signals S = { s_1, s_2, ….s_q}
- Set of location L = { l_1, l_2, ….l_p}, p ≥ |M|

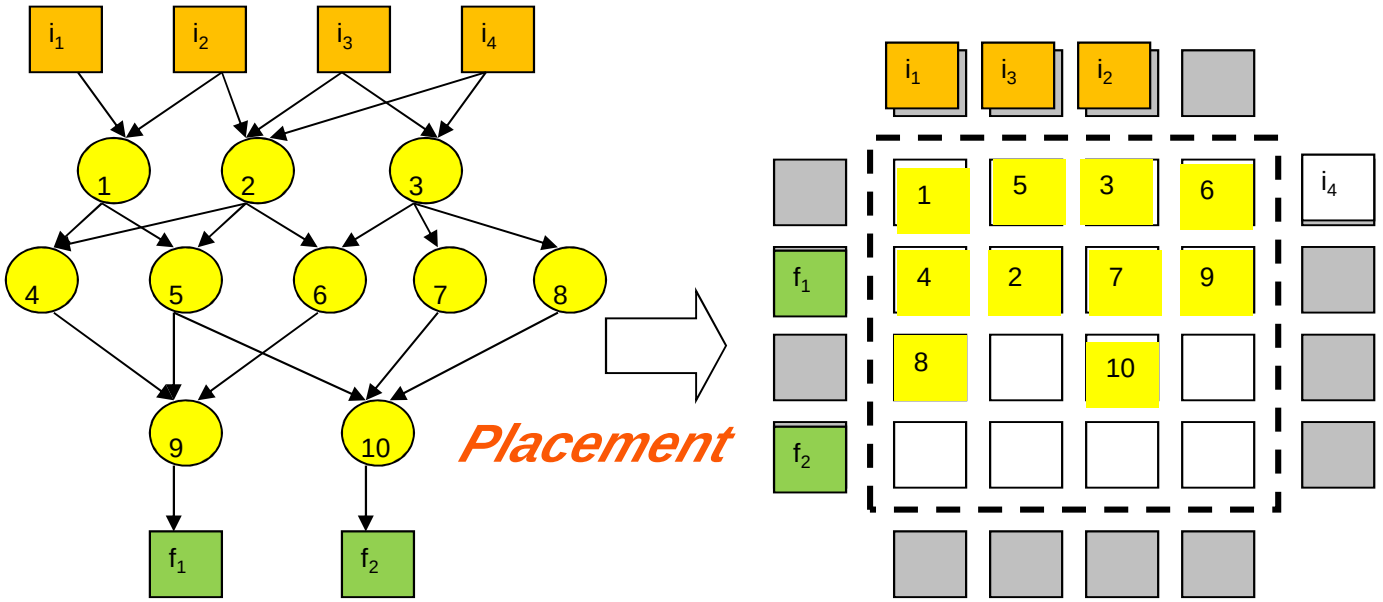
- ∀ $m_i \in$ M, there is a set of signals $S_{m_i} \subseteq S$
- ∀ $s_i \in$ S, there is a set of modules M_{s_i}, $M_{s_i} = \{ m_j | s_i \in S_{m_j} \}$
- M_{s_i} is said to be a ***signal net***

Goal : To assign each module $m_i \in$ M to a location $l_j \in$ L such that the chosen objective function is optimized.

FPGA Placement Problem

- **Input** – A ***technology mapped*** netlist of ***Configurable Logic Blocks (CLB)*** realizing a given circuit.

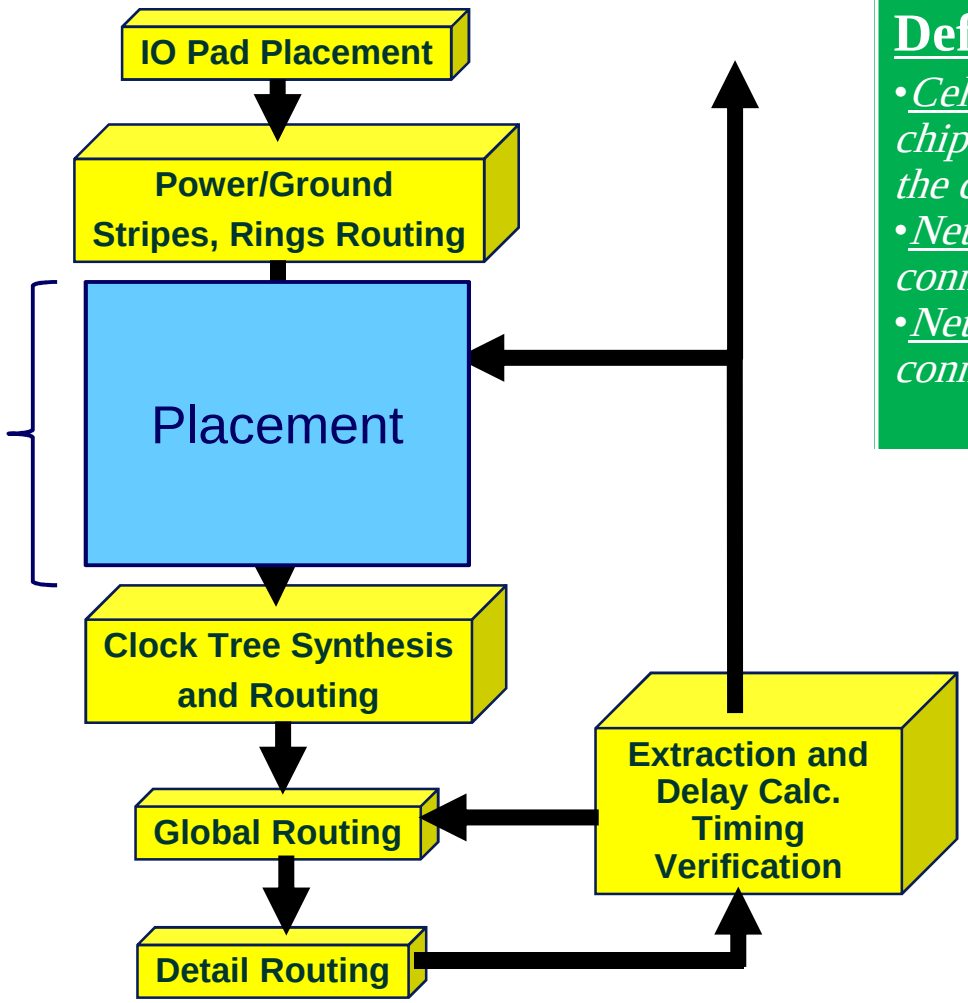
- **Output** – ***CLB netlist*** placed in a two dimensional array of slots such that total wirelength is minimized.



CLB Netlist

FPGA

VLSI Design Flow and Physical Design



Definitions:

- *Cell: a circuit component to be placed on the chip area. In placement, the functionality of the component is ignored.*
- *Net: specifying a subset of terminals, to connect several cells.*
- *Netlist: a set of nets which contains the connectivity information of the circuit.*

Placement is an **NP-Complete problem, therefore** we seek to simplify the solution to the problem by breaking it down further into smaller problems:

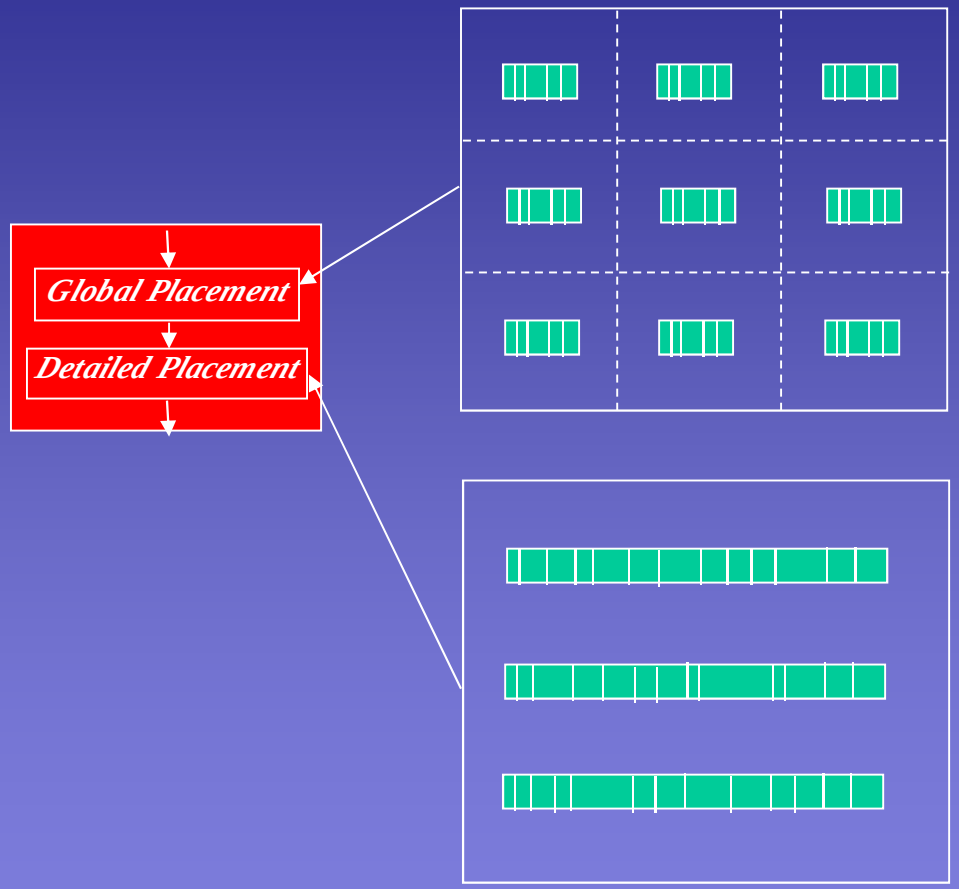
- Global Placement
- Detailed Placement

Global and Detailed Placement

In global placement, we decide the approximate locations for cells by placing cells in global bins.

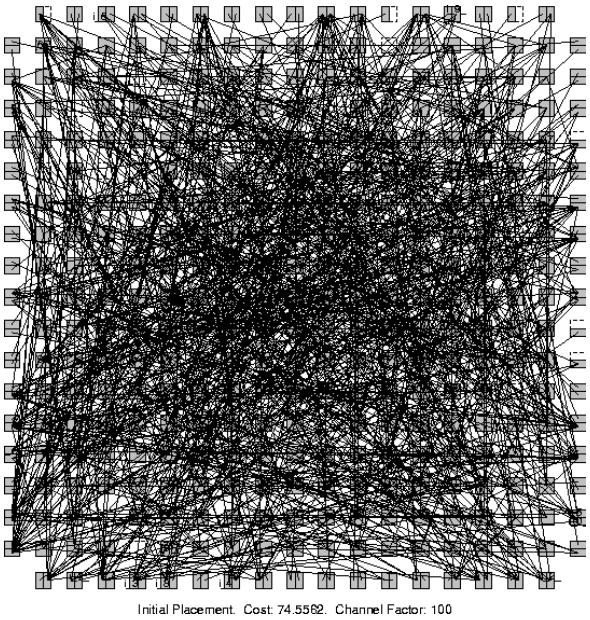
In detailed placement, we make some local adjustment to obtain the final non-overlapping placement.

- Legalization
- Local Improvement

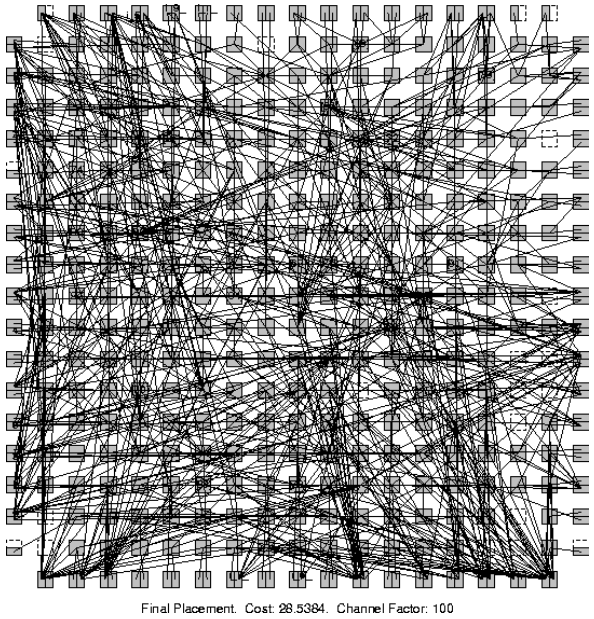


Placement Makes a Difference

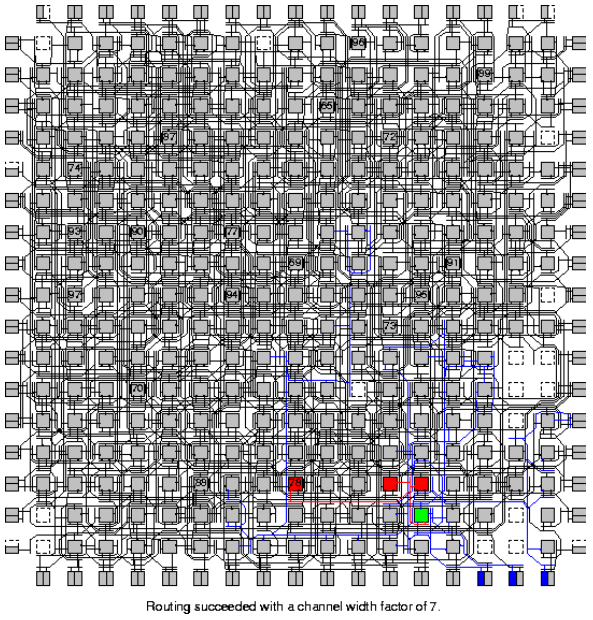
MCNC Benchmark circuit e64 (contains 230 4-LUT).
Placed to a FPGA.



Random Initial Placement



Global Placement



After Detailed Routing

Placement and Routing

- Two critical phases of layout design:
  - ❖ Placement of components on the chip;
  - ❖ Routing of wires between components.
- Placement and routing ***interact***, but separating layout design into phases:
  - ✓ ***Reduces the complexity*** of the problem.
  - ✗ Due to interaction solutions obtained are ***suboptimal*** (lose critical information).

FPGA Compile Time

- ***Compiling time ≈***

  (placement time) + (routing time)

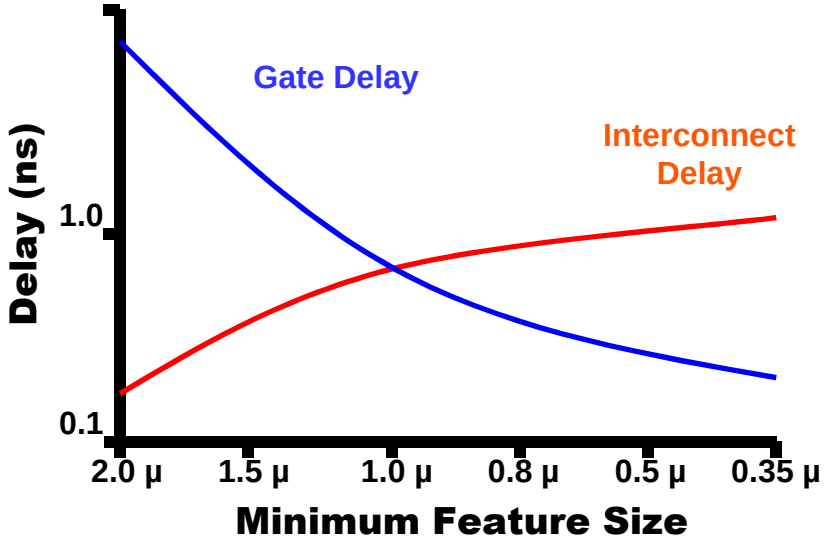
  - Currently, placement and routing time for large FPGAs can easily take hours or even days to complete.
  - These prohibitively long compiling times definitely nullified the time-to-market advantages of FPGAs!!.

- ***Heuristic methods are used to find sub-optimal solutions in reasonable amount of time.***

Placement: Why Important?

Reasons:

- Serious ***interconnect issues*** (delay, routability, noise) in deep-submicron design
- Need placement information even in the early designing stages, e.g., ***logic synthesis***
- Cong et al. [ASPDAC-03, ISPD-03, ICCAD-03] point out that ***existing placers are far from optimal***, not scalable, and not stable



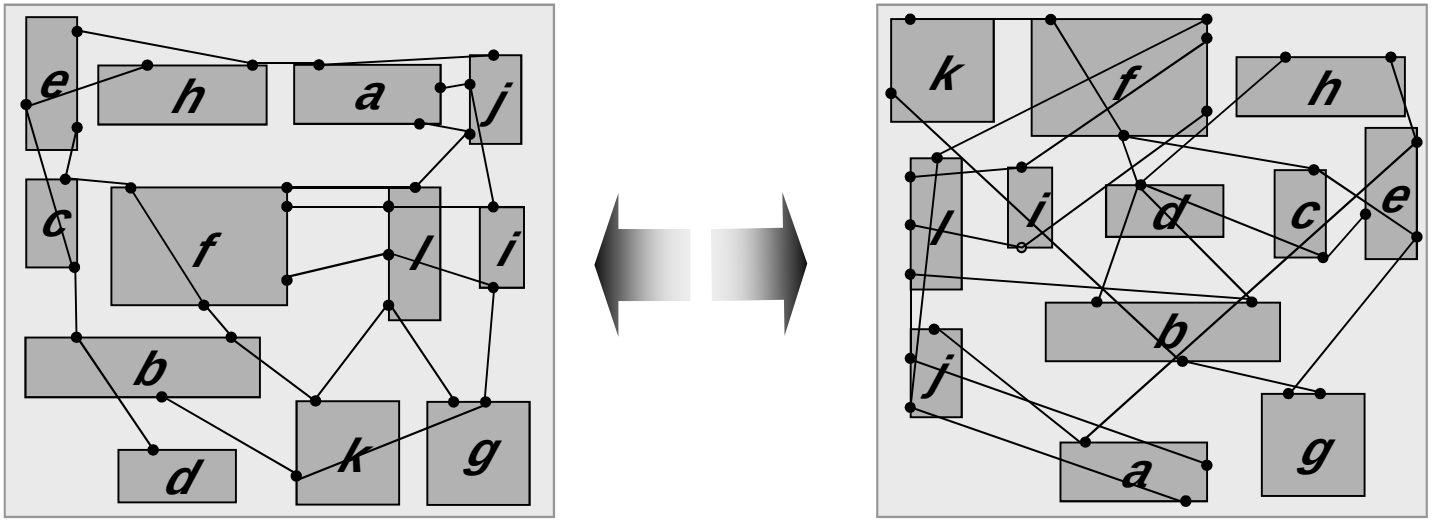
Placement: Objectives

- ***Placement*** is the procedure to determine the physical location of each CLBs and I/O pads on the target FPGA based on some specific objective:
  1. Wire length,
  2. Delay,
  3. Power,
  4. Congestion,
  5. Routability, ….
- Placement has been proven to be an ***NP-hard problem*** (Non-Deterministic Polynomial)
  - It cannot be solved exactly in linear time (i.e., If there are N blocks need to be placed, the complexity of the problem will be N!)

Placement metrics

- Area, delay, congestion and power are determined partly by wiring.
- ***How do we judge a placement Quality?***
  1. ***Perform exact wire length measurement.***
  2. ***Estimate wire length*** without actually performing routing.
- Design time may be important for FPGAs
  - Designers might want to experiment with different architectures and therefore ***compile time becomes crucial.***

Total Wire-length



- Wire-length can be measured or estimated using several techniques:
  - Half Perimeter Wire Length (HPWL) → Good estimate
  - Steiner Trees (most accurate!)
  - Spanning Trees (close to Steiner Trees)
  - Star Model
  - …..

Total Wire-Length Cost

Total wirelength with net weights (weighted wirelength)

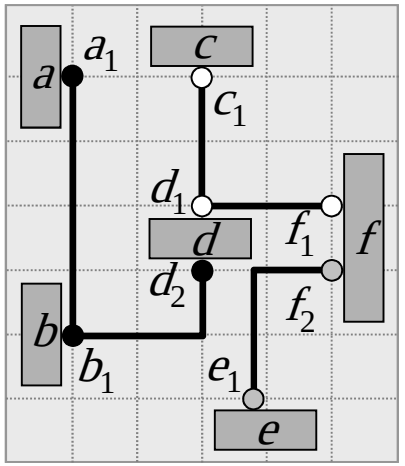
- For a placement ***P***, an estimate of total weighted wirelength is

$$L(P) = \sum_{net \in P} w(net) \cdot L(net)$$

where ***w(net)*** is the weight of ***net***, and ***L(net)*** is the estimated wirelength of ***net***.

- Example:

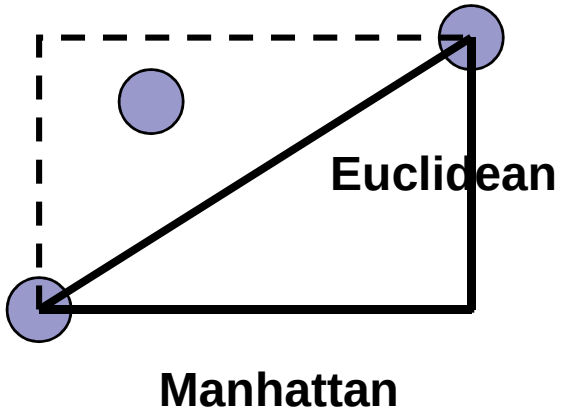
Nets	Weights
$N_1 = (a_1, b_1, d_2)$	$w(N_1) = 2$
$N_2 = (c_1, d_1, f_1)$	$w(N_2) = 4$
$N_3 = (e_1, f_2)$	$w(N_3) = 1$



$$L(P) = \sum_{net \in P} w(net) \cdot L(net) = 2 \cdot 7 + 4 \cdot 4 + 1 \cdot 3 = 33$$

Wire length measures

- Estimate wire length by distance between components.
- Possible distance measures:
  1. Euclidean distance (sqrt($x^2 + y^2$));
  2. Manhattan distance (x + y).
  3. ***HPWL is also a measure***.
- Multi-point nets must be broken up into trees for good estimates.



Wire-length Estimation

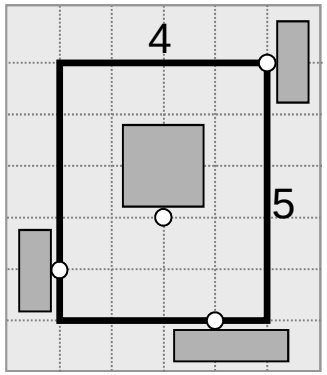
Wirelength estimation for a given placement

Half-perimeter wirelength (HPWL)

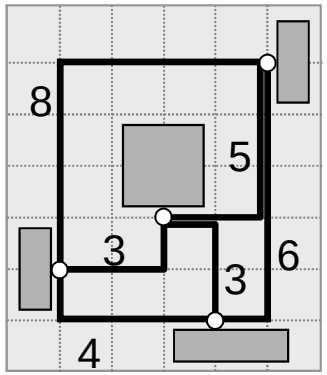
Complete graph (clique)

Monotone chain

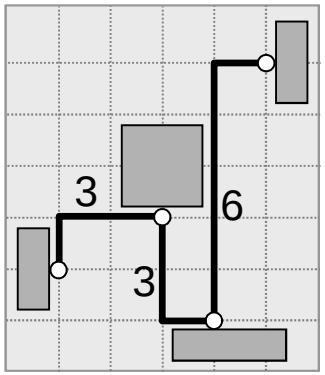
Star model



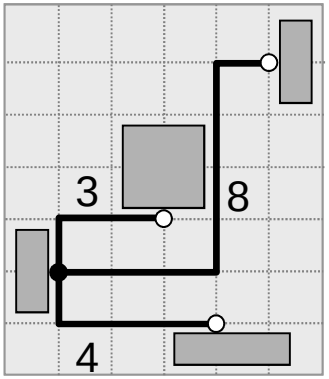
HPWL = 9



Clique Length = $(2/p)\Sigma_{e \in \text{clique}} d_M(e) = 14.5$



Chain Length = 12



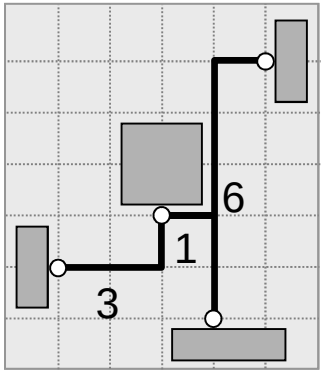
Star Length = 15

HPWL

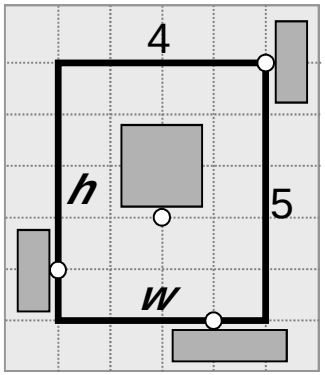
Wirelength estimation for a given placement (cont‘d.)

Preferred method: Half-perimeter wirelength (HPWL)

- Fast (order of magnitude faster than RSMT)
- Equal to length of RSMT for 2- and 3-pin nets
- Margin of error for real circuits approx. 8% [Chu, ICCAD 04]



RSMT Length = 10



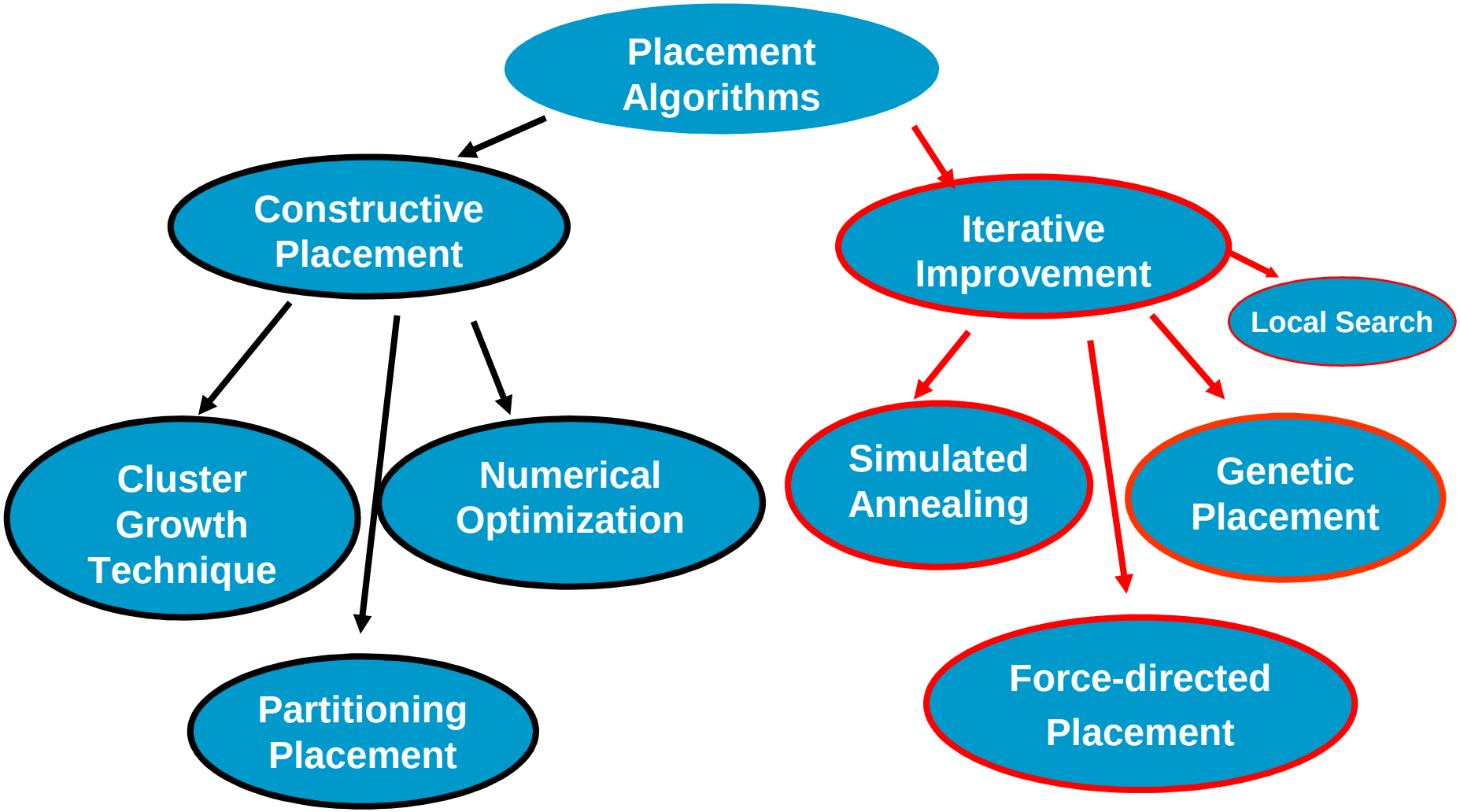
HPWL = 9

$$L_{\text{HPWL}} = w + h$$

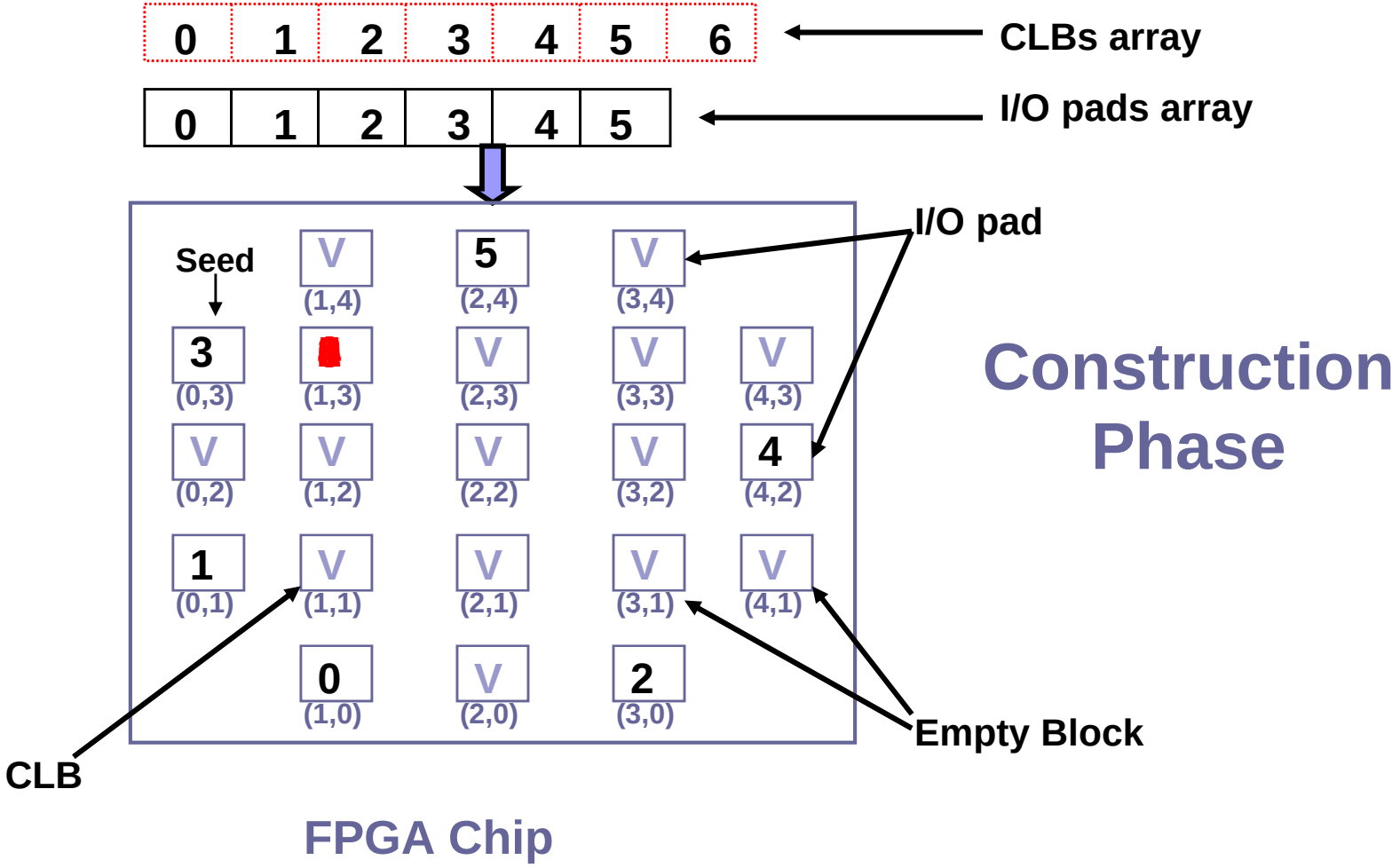
Placement Phases

- Construction Phase
  - Generate a good/feasible ***initial solution***
- Iterative Improvement Phase
  - Local search techniques are applied to ***improve current*** initial solution

Placement Techniques



Cluster Growth Constructive



Construction Phase

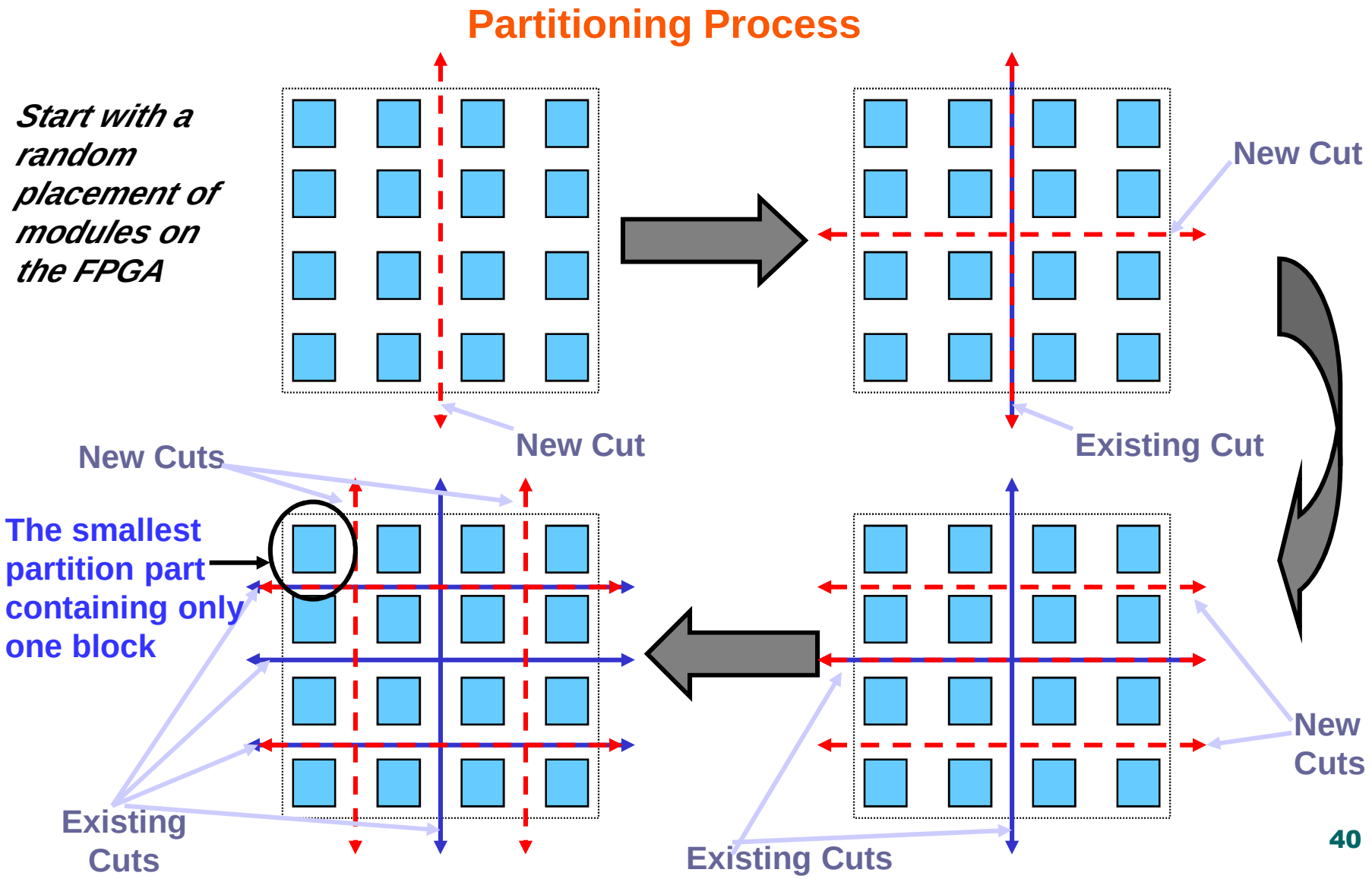
Heuristic methods for FPGA Placement

- Partitioning-based placement.
- Iterative placement.
  - Simulated-annealing algorithm.
  - Simple local search.

Currently, simulated-annealing is one of the dominant methods used by both academia and industry.

VPlace (VPR) is the leading tool for FPGA placement that developed in UNIV. of Toronto.

Partitioning-based Algorithm

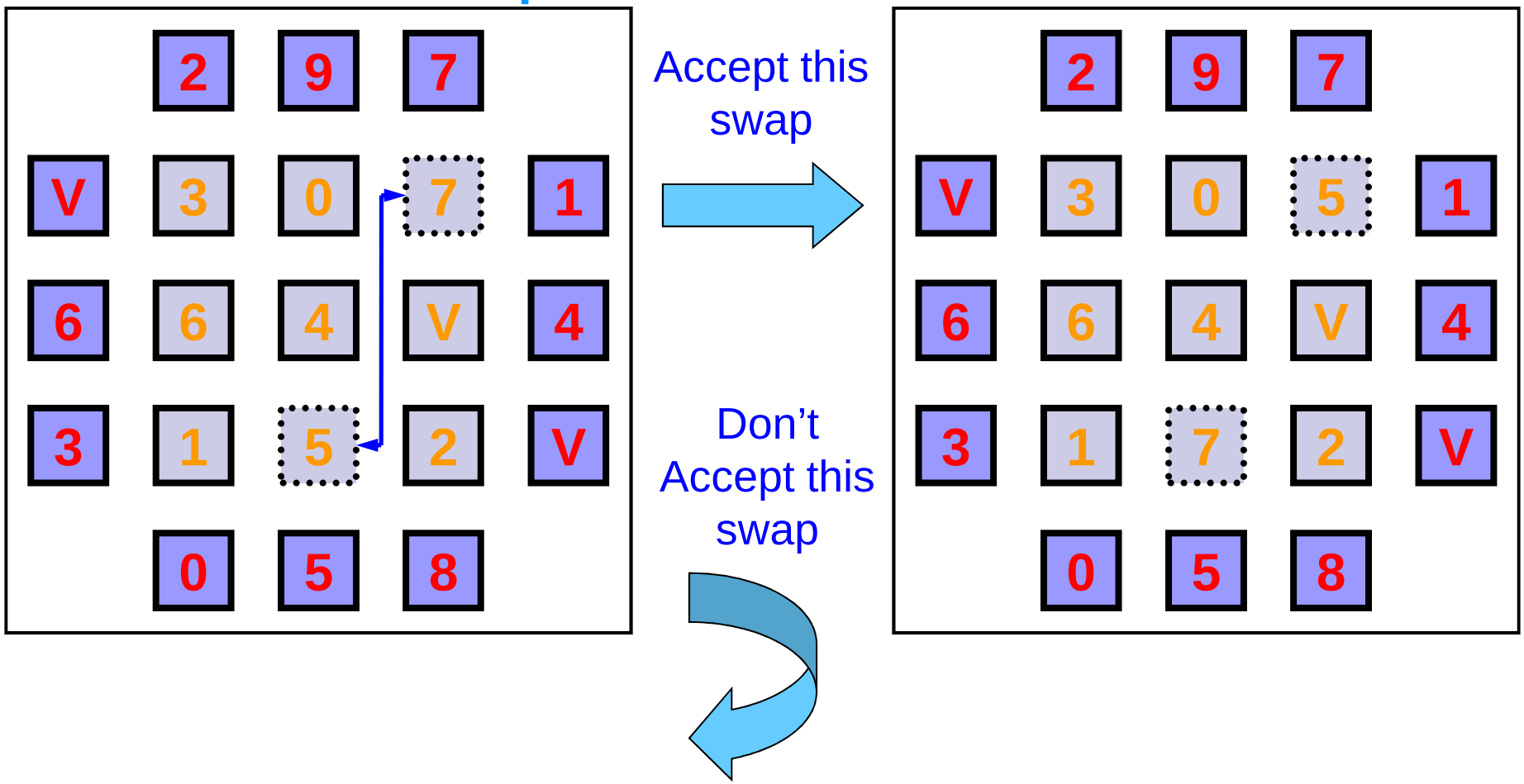


Iterative improvement

- Start from a ***feasible*** (legal) solution.
  - Random start
  - Constructive based
- Seek improvements by making ***small perturbations.***
  - Create a neighboring solution (candidate solution).
  - Select (randomly) two blocks, then swap their locations.

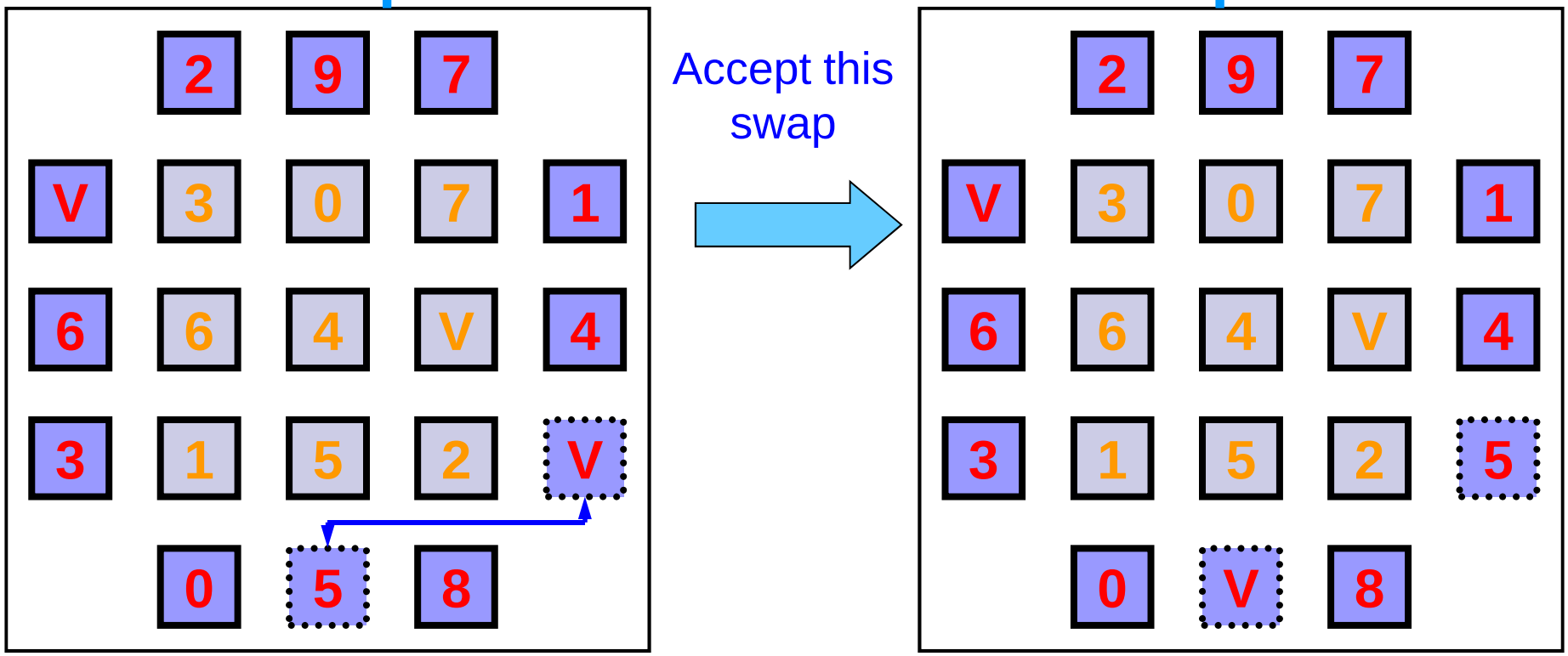
Create a neighboring solution:

Swap between two CLBs



Create a neighboring solution:

Swap between two I/O pads



Iterative Improvement

General method to solve combinatorial optimization problems

Principle:

- Start with initial configuration
- Repeatedly search neighborhood and select a neighbor as candidate
- Evaluate some cost function (or fitness function) and accept candidate if "better"; if not, select another neighbor
- Stop if:
  - quality is sufficiently high, or
  - no further improvement can be found, or
  - after some fixed time or iterations

Iterative Improvement

Simple Iterative Improvement or Hill Climbing:

- Candidate is always and only accepted if cost is lower (or fitness is higher) than current configuration
- Stop when no neighbor with lower cost (higher fitness) can be found

Disadvantages:

- Local optimum as best result
- Local optimum depends on initial configuration
- Generally no upper bound on iteration length

Criteria of accepting swaps

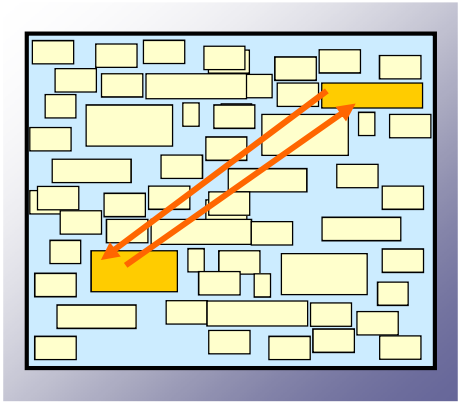
- Improving swaps are always accepted.
- Non-improving swaps:
  - Simple local search:
    - Do Not Accept!
  - Meta-Heuristic (Simulated-annealing: Accept)
    - Based on a controlled probability. ($e^{-\Delta C/T}$)
    - In the initial start phase accept with high probability and as we proceed accept bad moves with low probability

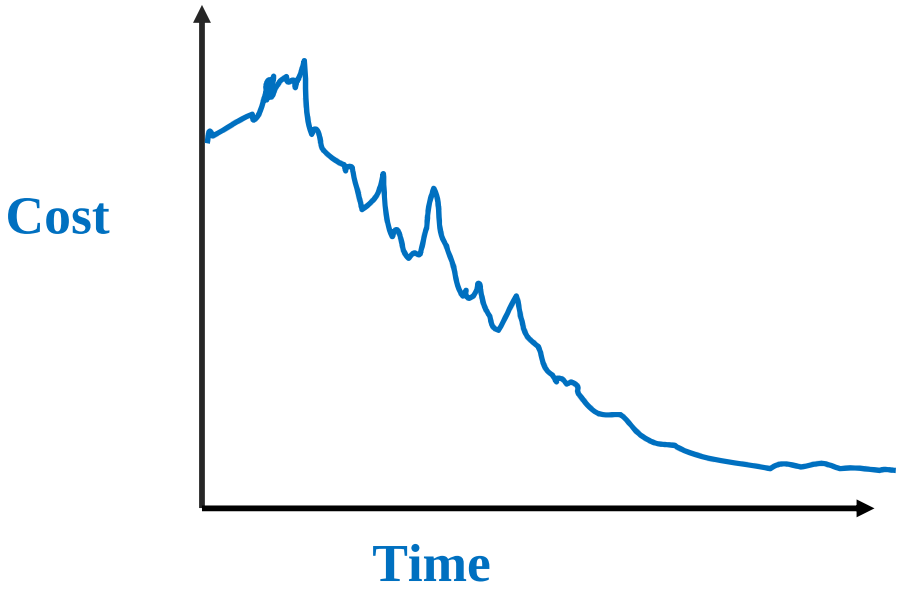
Simulated Annealing Placement

☞ An Initial Placement is Improved through Swaps and Moves

☞ **Always** accept a Swap/Move if it improves (reduces) cost

☞ **Conditionally** accept a Swap/Move that degrades cost under some probability conditions





Simulated Annealing

- **Motivated by the Physical Annealing Process**
  - **Material is heated and slowly cooled down, so that all particles arrange in the ground energy state into a uniform structure (i.e., create perfect crystals).**
  - **At each temperature wait until the solid reaches its thermal equilibrium.**
- **Simulated annealing mimics this process**
- **The first SA algorithm was developed in 1953.**

Simulated Annealing

- **Compared to hill climbing the main difference is that SA allows upwards moves**
- **In Simulated Annealing:**
  - **Good moves are always accepted.**
  - **Bad moves accepted with probability**
- **Simulated annealing also differs from hill climbing in that a move is selected at random and then decides whether to accept it**

To accept or not to accept?

$$P = \exp(-\Delta \textbf{cost} / T) > r$$

Where:

- **P: Probability of accepting a move**
- **Δ cost: change in the evaluation function**
- ***T*: the current temperature**
- ***r* is a random number between 0 and 1**

When to Accept?

Δ cost	Temp	exp(-C/T)		Δ cost	Temp	exp(-C/T)
0.2	0.95	0.810157735		0.2	0.1	0.135335283
0.4	0.95	0.656355555		0.4	0.1	0.018315639
0.6	0.95	0.53175153		0.6	0.1	0.002478752
0.8	0.95	0.430802615		0.8	0.1	0.000335463

$$P = \exp(-\Delta \text{ cost}/T) > r$$

- When the temperature is high (at the start of the search) the value of $P = \exp(-\Delta \text{ cost}/T)$ is high and in most cases will be higher than the random number generated "r".
- However, as we decrease the temperature (at the end of the search T is getting smaller) the value of $P = \exp(-\Delta \text{ cost}/T)$ is low and P is not going to be greater than the random number generated "r".

Simulated Annealing

Step 1: Initialize – Start with a random initial placement. Initialize T to a very high value or “temperature”.

Step 2: Move – Perturb the placement through a defined move.

Step 3: Calculate score – calculate the change in the score due to the move made.

Step 4: Choose – Depending on the change in score, accept or reject the move. The prob of acceptance depending on the current “temperature”.

Step 5: Update and repeat – Update the temperature value by lowering the temperature. Go back to Step 2.

The process is done until “Freezing Point” is reached.

Simulated Annealing

- Initialize temperature $T_{initial}$ =(High Value), $T_{final,}$ = (Low Value)
- Create an initial random solution S = Random Placement
- While ($T_{initial} > T_{final}$)

{

Repeat step 1, 2 and 3 a fixed number of times:

1. Generate a new solution S'
2. If cost(S') < cost (S),

accept S', i.e., S $\Leftarrow$ S'

3. Else

if random() < $e^{-k\Delta cost/T}$

accept S'

else

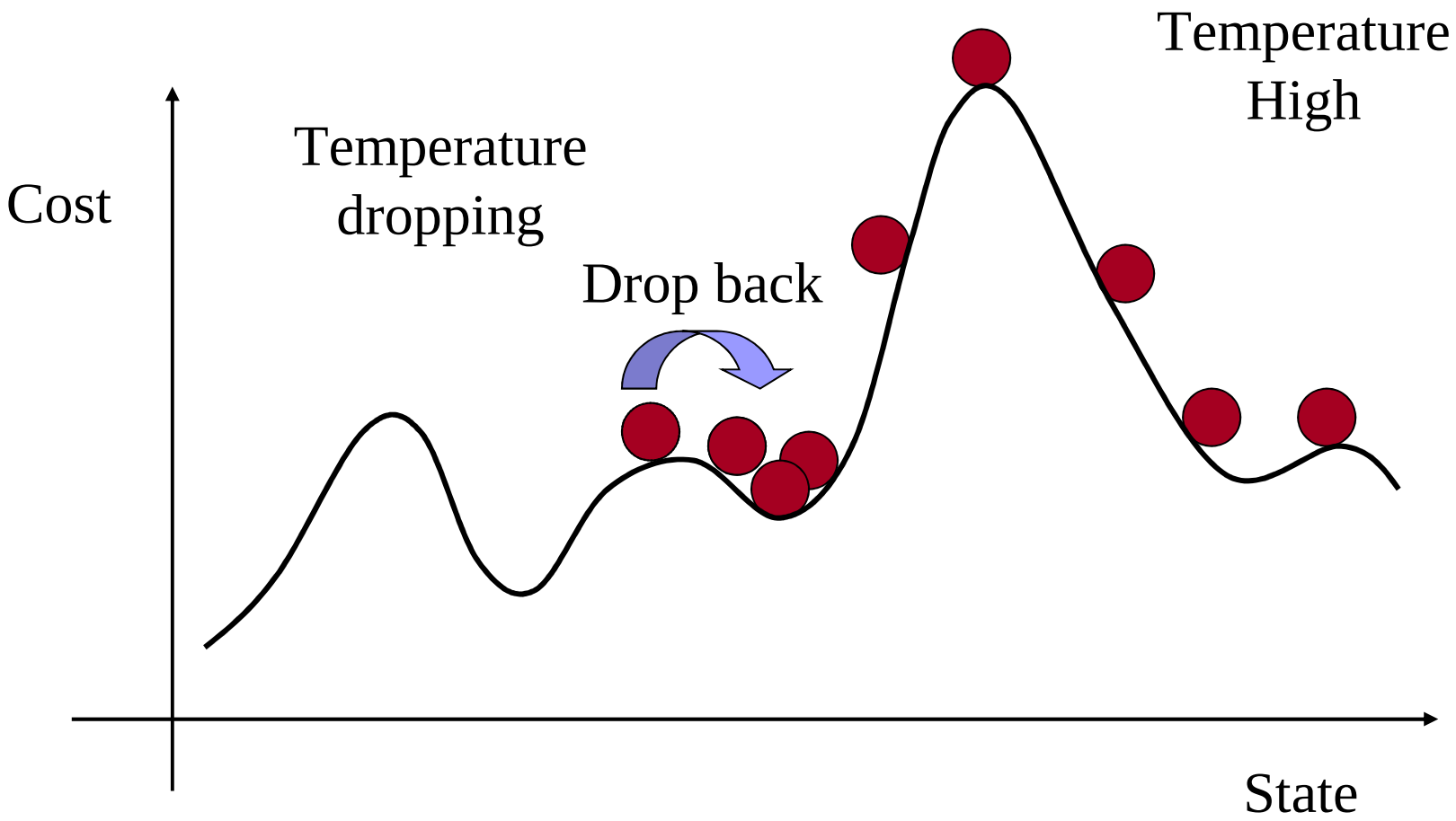
reject S'

Decrease T

}

report (best solution)

Simulated Annealing



Pros and Cons of SA

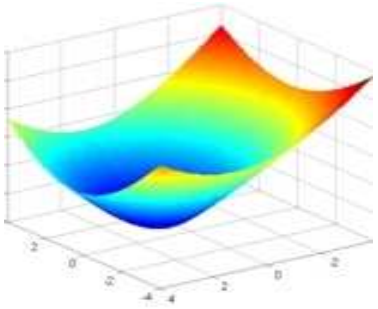
- Pros:
  - Can Reach Globally Optimal Solution (given “enough” time)
  - Open Cost Function.
  - Can Optimize Simultaneously all Aspects of Physical Design
  - Can be Used for End Case Placement
- Cons:
  - Extremely Slow Process of Reaching a Good Solution
  - We may revisit solutions visited before (unproductive)

Analytic Placement

- Also referred to as force-directed or quadratic placement
- Analytic Placement minimizes a given objective, such as wire-length or circuit delay, using mathematical techniques such as numerical analysis or linear programming.
- Such methods often require certain assumption such as the differentiability of the obj Function.
- Algorithm:
  - Solve a set of linear equations to find an intermediate solution
  - Repeat the process until equilibrium

Quadratic Placement

- Write down the placement problem as an analytical mathematical problem
- Suppose that all nets in the circuit are 2-pin nets
  - Consider a net {i,j}, the wire-length is given by Manhattan distance:
  - $L_{\{i,j\}} = |x_i - x_j| + |y_i - y_j|$
- This is usually referred to as a linear wire-length (non-differentiable)!!!.
- Solution?
- A common idea is to consider the squared Euclidean distance between modules instead:
  - $\tilde{L}_{\{i,j\}} = (x_i - x_j)^2 + (y_i - y_j)^2$
  - So the wire-length minimization problem can be formulated as a quadratic program.
  - It can be proved that the quadratic program is convex, hence polynomial time solvable



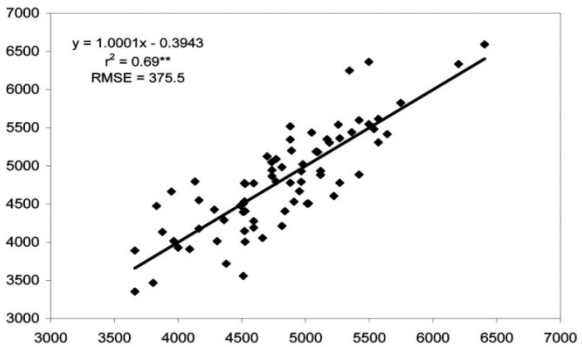
Quadratic Placement

- Objective function is quadratic; sum of (weighted) squared Euclidean distance represents placement objective function

$$L(P) = \frac{1}{2} \sum_{i,j=1}^{n} c_{ij} \left((x_i - x_j)^2 + (y_i - y_j)^2 \right)$$

where n is the total number of cells, and $c(i,j)$ is the connection cost between cells i and j.

- Only two-point-connections
- Minimize objective function by equating its derivative to zero which reduces to solving a system of linear equations
- Similar to Least-Mean-Square Method (root mean Square)



NOTE: In quadratic placement techniques, it is more convenient to set the cost function L(P) to half of the total weighted quadratic wirelength so that the derivatives will have simpler forms

Quadratic Placement

$$L(P) = \frac{1}{2} \sum_{i,j=1}^{n} c_{ij} \left(\left(x_i - x_j \right)^2 + \left(y_i - y_j \right)^2 \right)$$

where n is the total number of cells, and $c(i,j)$ is the connection cost between cells i and j.

- Each dimension can be considered independently:

$$L_x(P) = \sum_{i=1,j=1}^{n} c(i,j)(x_i - x_j)^2 \qquad L_y(P) = \sum_{i=1,j=1}^{n} c(i,j)(y_i - y_j)^2$$

- Convex quadratic optimization problem: any local minimum solution is also a global minimum
- Optimal x- and y-coordinates can be found by setting the partial derivatives of $L_x(P)$ and $L_y(P)$ to zero

Quadratic Placement

$$L(P) = \frac{1}{2}\sum_{i,j=1}^{n} c_{ij}\left(\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2\right)$$

where n is the total number of cells, and $c(i,j)$ is the connection cost between cells i and j.

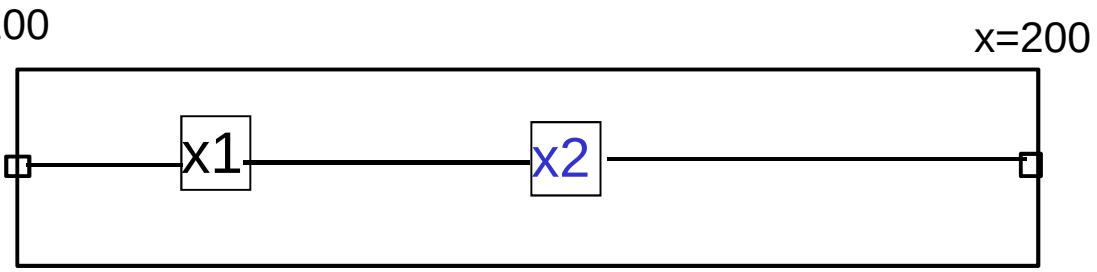
- Each dimension can be considered independently:

$$L_x(P) = \sum_{i=1,\, j=1}^{n} c(i,j)(x_i - x_j)^2 \qquad L_y(P) = \sum_{i=1,\, j=1}^{n} c(i,j)(y_i - y_j)^2$$

$$\frac{\partial L_x(P)}{\partial X} = AX - b_x = 0 \qquad \frac{\partial L_y(P)}{\partial Y} = AY - b_y = 0$$

- where A is a matrix with $A[i][j] = -c(i,j)$ when $i \neq j$,
  and $A[i][i]$ = the sum of incident connection weights of cell i.
- X is a vector of all the x-coordinates of the non-fixed cells, and b_x is a vector with $b_x[i]$ = the sum of x-coordinates of all fixed cells attached to i.
- Y is a vector of all the y-coordinates of the non-fixed cells, and b_y is a vector with $b_y[i]$ = the sum of y-coordinates of all fixed cells attached to i.

Toy Example:



$$Cost = (x_1 - 100)^2 + (x_1 - x_2)^2 + (x_2 - 200)^2$$

$$\frac{\partial}{\partial x_1} Cost = 2(x_1 - 100) + 2(x_1 - x_2)$$

$$\frac{\partial}{\partial x_2} Cost = -2(x_1 - x_2) + 2(x_2 - 200)$$

where ***A*** is a matrix with ***A***[*i*][*j*] = -***c***(*i*,*j*) when ***i*** ≠ ***j***, and ***A***[*i*][*i*] = the sum of incident connection weights of cell ***i***.

X is a vector of all the ***x***-coordinates of the non-fixed cells, and b_x is a vector with b_x[*i*] = the sum of ***x***-coordinates of all fixed cells attached to ***i***.

Y is a vector of all the ***y***-coordinates of the non-fixed cells, and b_y is a vector with b_y[*i*] = the sum of ***y***-coordinates of all fixed cells attached to ***i***.

setting the partial derivatives = 0 we solve for the minimum Cost:

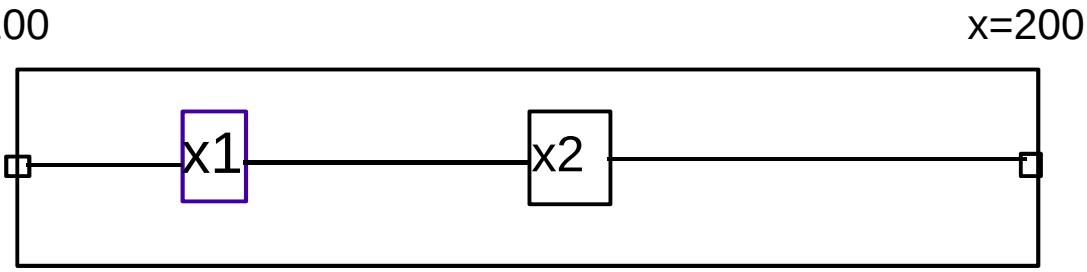
Ax + B = 0

$$\begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -200 \\ -400 \end{bmatrix} = 0$$

$$\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -100 \\ -200 \end{bmatrix} = 0$$

x1=400/3 x2=500/3

Example:



setting the partial derivatives = 0 we solve for the minimum Cost:

$$Ax + B = 0$$

$$\begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}+\begin{bmatrix} -200 \\ -400 \end{bmatrix} = 0$$

$$\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}+\begin{bmatrix} -100 \\ -200 \end{bmatrix} = 0$$

x1=400/3 x2=500/3

Interpretation of matrices A and B:

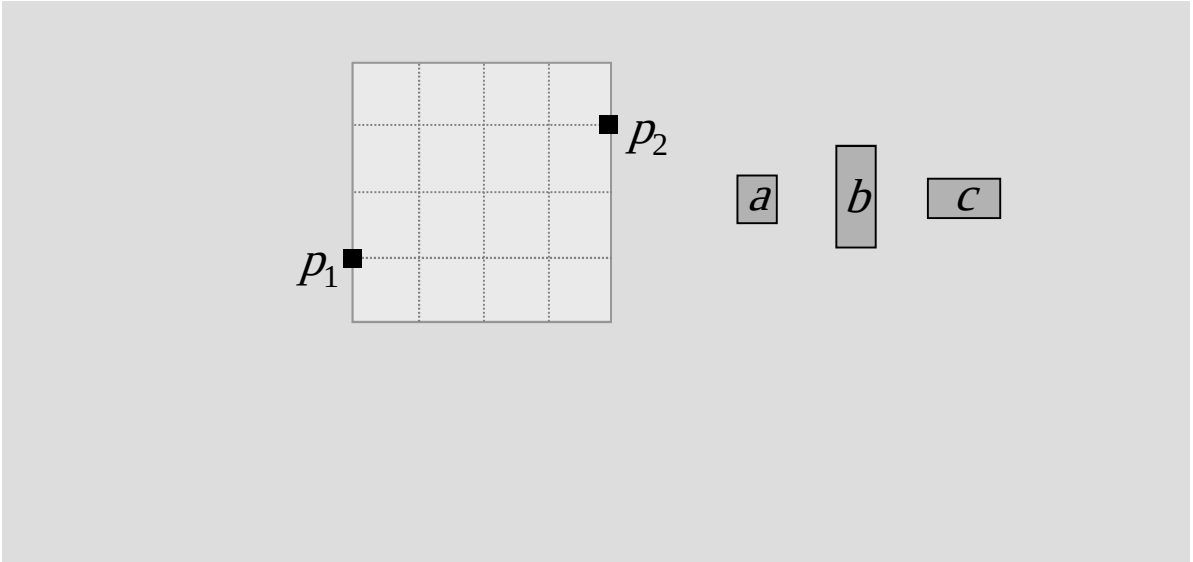
The diagonal values A[i,i] correspond to the number of connections to xi
The off diagonal values A[i,j] are -1 if object i is connected to object j, 0 otherwise
The values B[i] correspond to the sum of the locations of fixed objects connected to object i

Example II:

Given:

1. Placement P with two fixed points P_1 (100,175) and P_2 (200,225)
2. Three free blocks a,b,c
3. Four nets N_1 – N_4

   N_1 (P_1, a), N_2 (a,b), N_3 (b,c) and N_4 (c,P_2)



Task: find the coordinates of blocks (x_a,y_a), (x_b,y_b) and (x_c,y_c)

Solution of Example II:

First

1. Solve for x-coordinates
2. $L_x(P) = (100 - x_a)^2 + (x_a - x_b)^2 + (x_b - x_c)^2 + (x_c - 200)^2$

$$\frac{\partial L_x(P)}{\partial Xa} = 0 \qquad \frac{\partial L_x(P)}{\partial Xb} = 0 \qquad \frac{\partial L_x(P)}{\partial Xc} = 0 \qquad AX = b_x$$

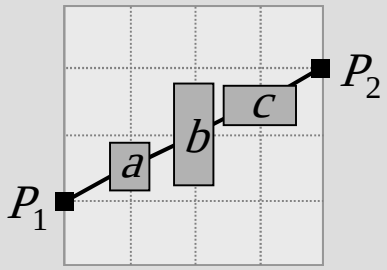
Then:

1. Solve for y-coordinates
2. $L_y(P) = (175 - y_a)^2 + (y_a - y_b)^2 + (y_b - y_c)^2 + (y_c - 225)^2$

$$\frac{\partial L_y(P)}{\partial Ya} = 0$$

$$\frac{\partial L_y(P)}{\partial Yb} = 0 \qquad AY = b_y$$

$$\frac{\partial L_y(P)}{\partial Yc} = 0$$

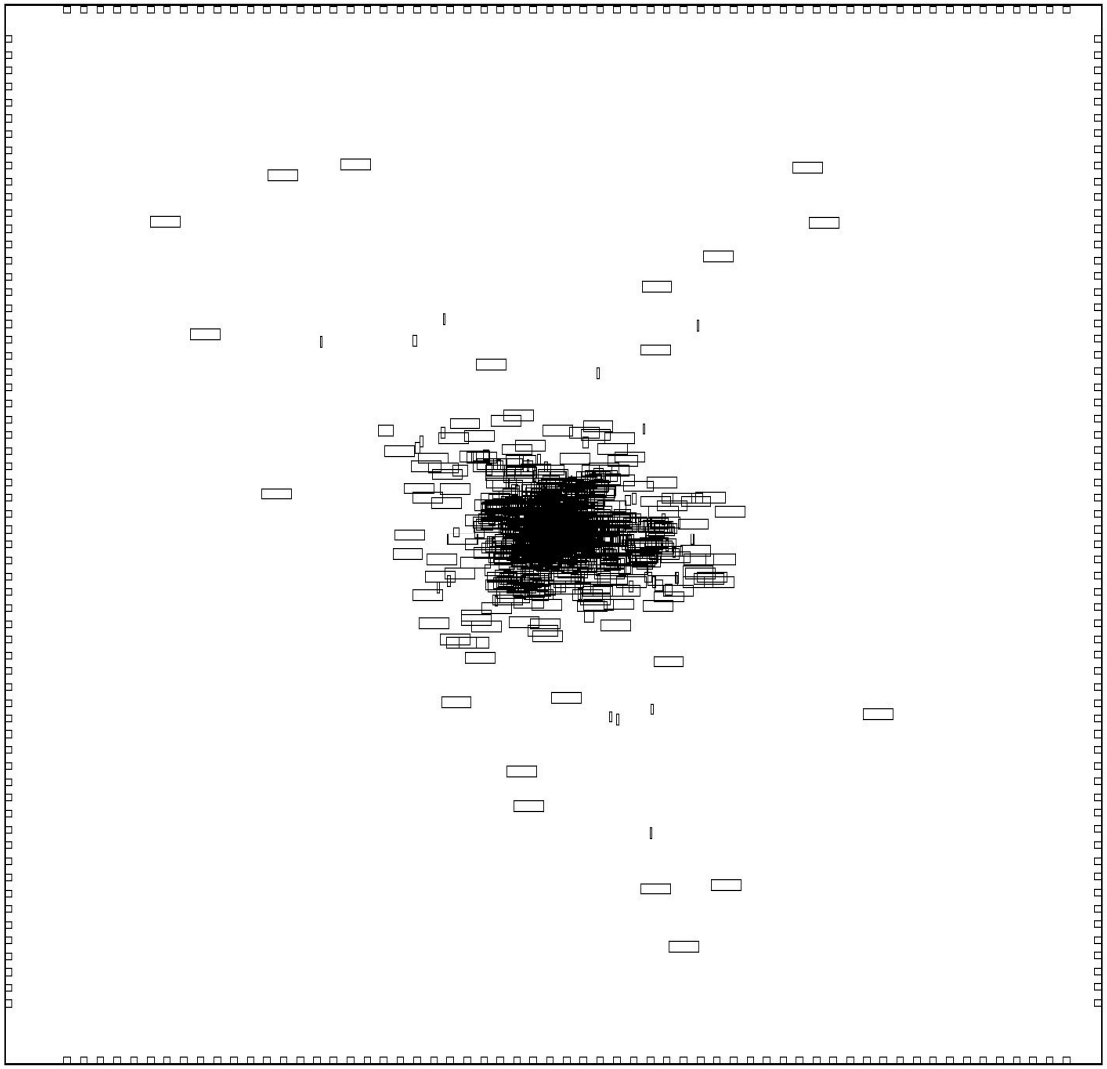


Solve for X: $x_a = 125$, $x_b = 150$, $x_c = 175$

Solve for Y: $y_a = 187.5$, $y_b = 200$, $y_c = 212.5$

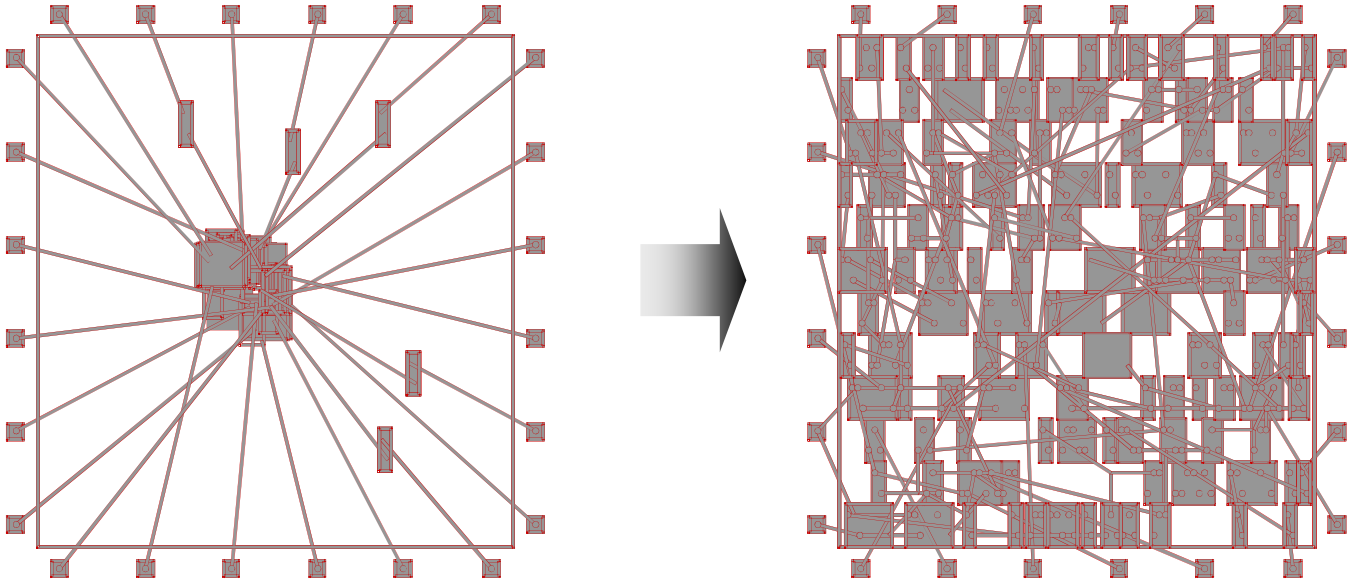
Final Solution: a(125,187.5), b(150,200) and c(175,212.5)

Solution of the Original QP



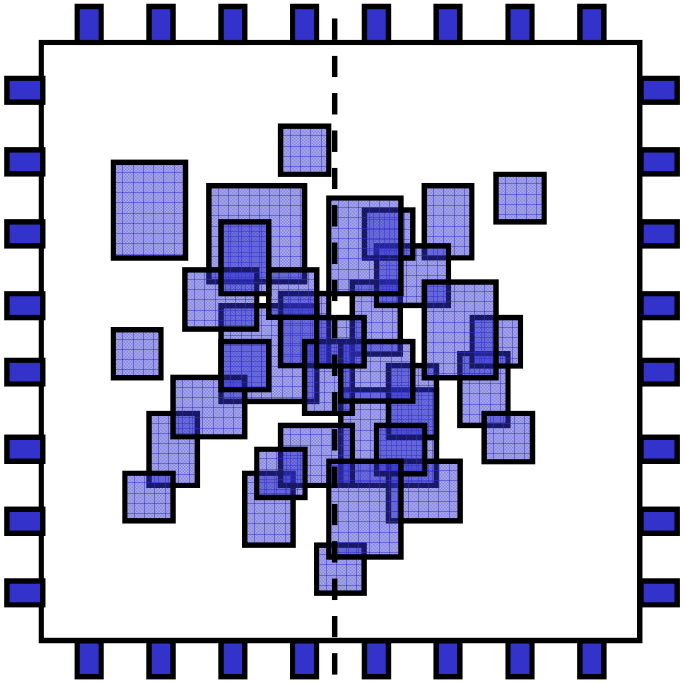
Quadratic Placement

- Second stage of quadratic placers: cells are spread out to **remove overlaps**
- Methods:
  - Adding fake nets that pull cells away from dense regions toward anchors
  - Geometric sorting and scaling
  - Repulsion forces, etc.



Partitioning

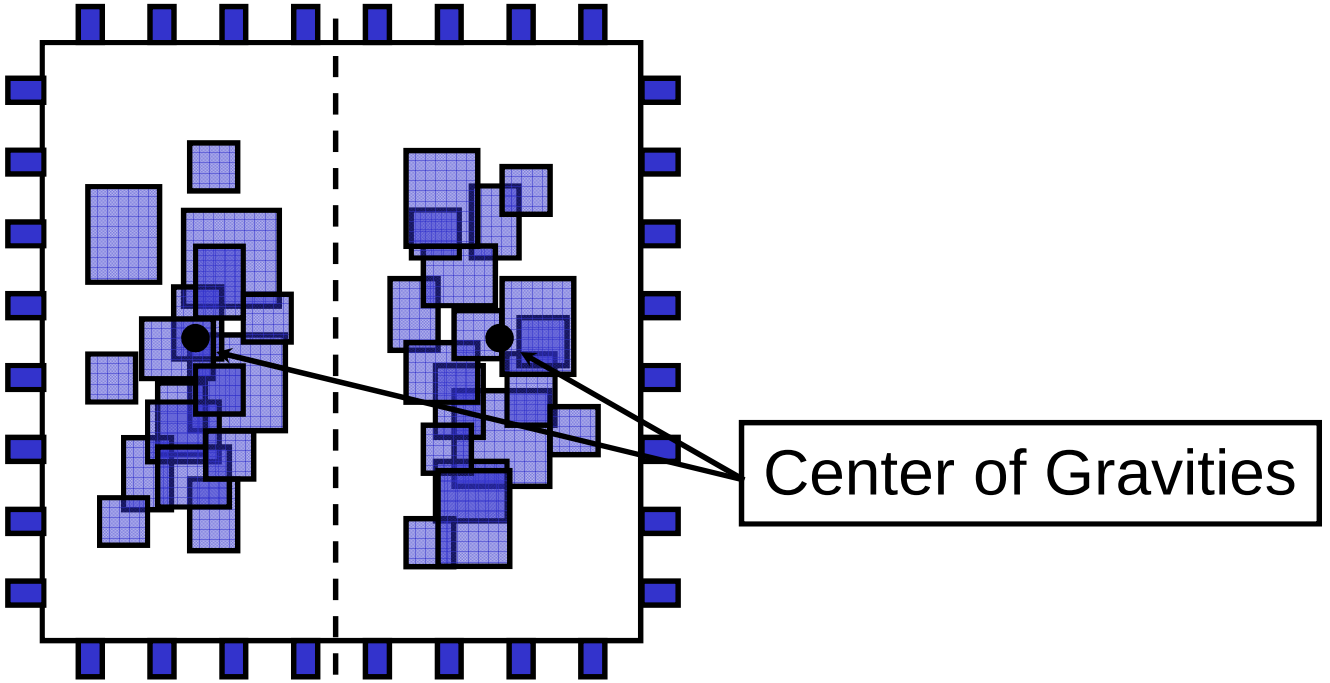
- Find a good cut direction and position.



- Improve the cut value using FM.

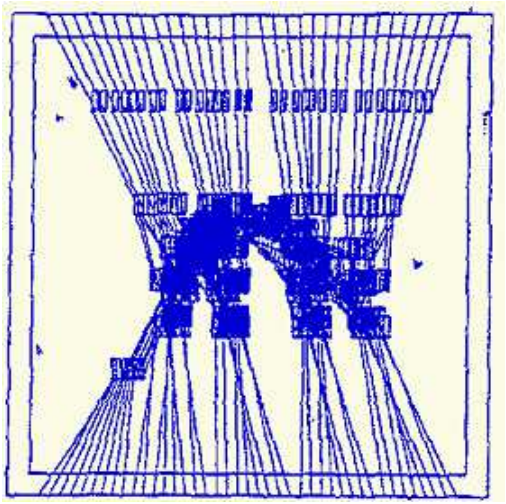
Applying the Idea Recursively

- Before every level of partitioning, do the Global Optimization again with additional constraints that the center of gravities should be in the center of regions.

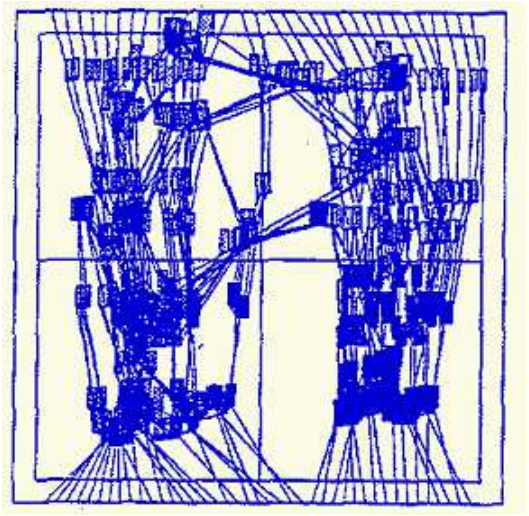


- Always solve a single QP (i.e., global).

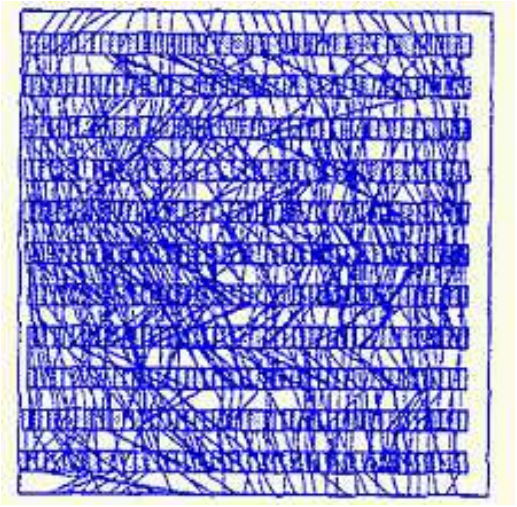
Analytic Placement



(a) Global placement with 1 region



(b) Global placement with 4 region



(c) Final placements

Pros and Cons of AP (QP/FD)

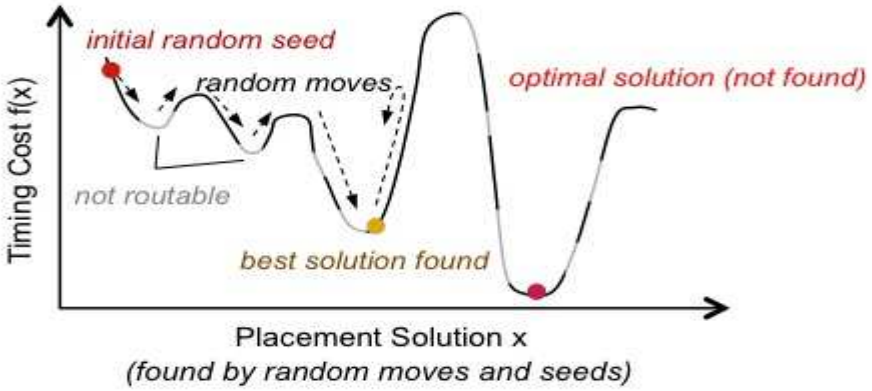
✓ Pros:

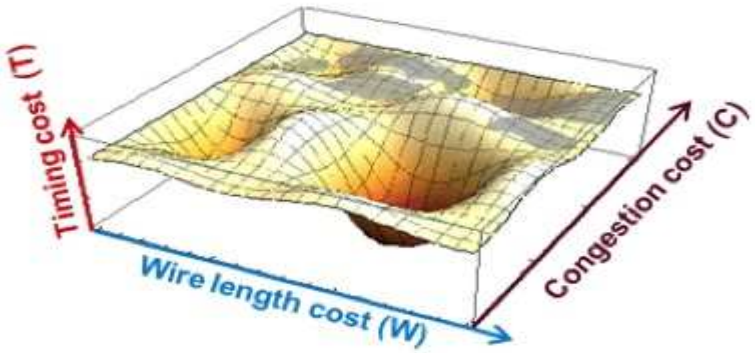
- **Very Fast** Analytical Solution
- Can Handle Large Design Sizes
- Can be Used as an Initial Seed Placement Engine

✘ Cons:

- Can Generate Overlapped Solutions: Postprocessing Needed
- Might not be suitable for Timing Driven Placement
- Not Suitable for Simultaneous Optimization of Other Aspects of Physical Design (clocks, crosstalk…)
- Gives Trivial Solutions without Pads ..

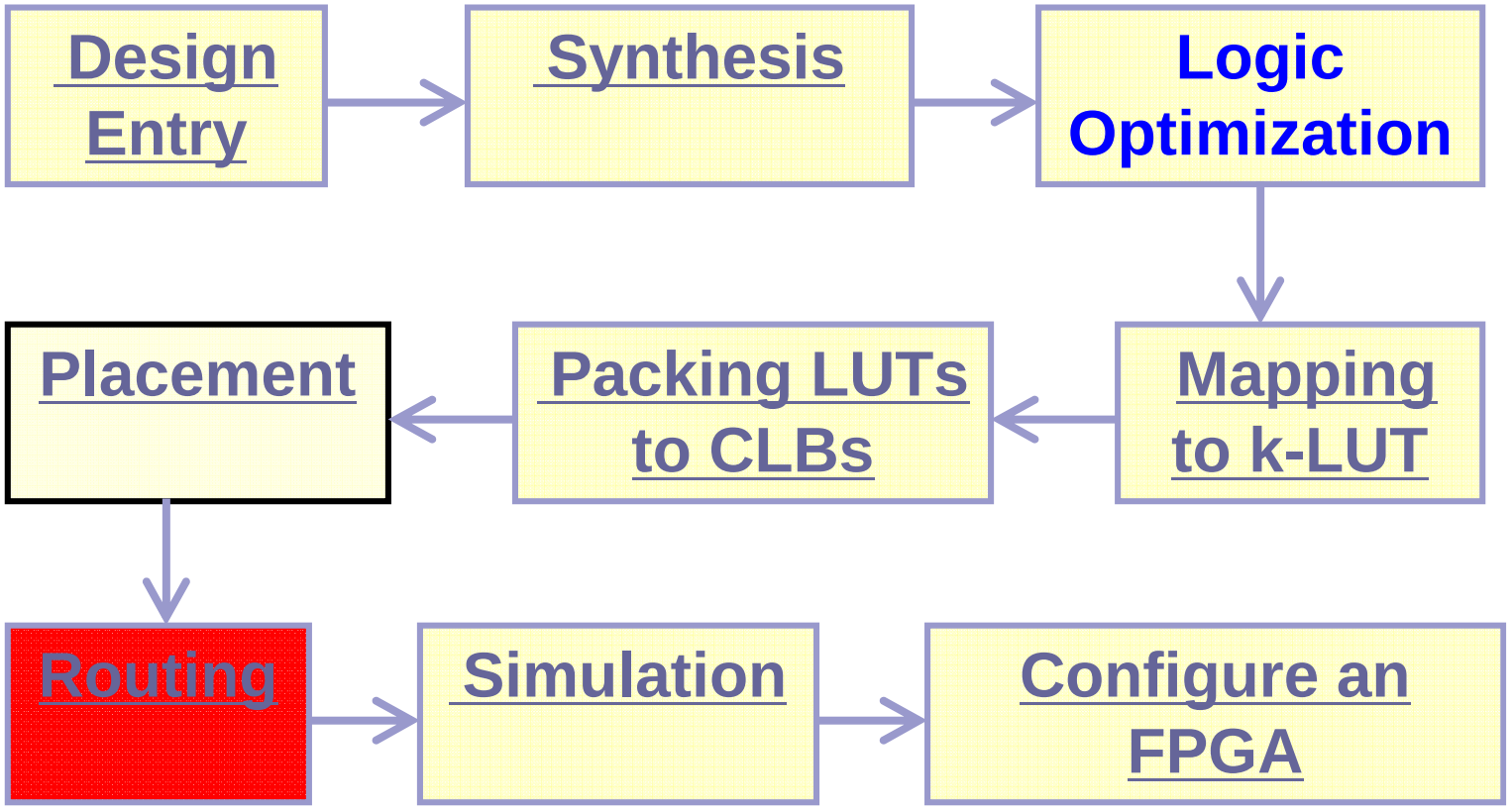
Xilinx ISE vs. Xilinx Vivado





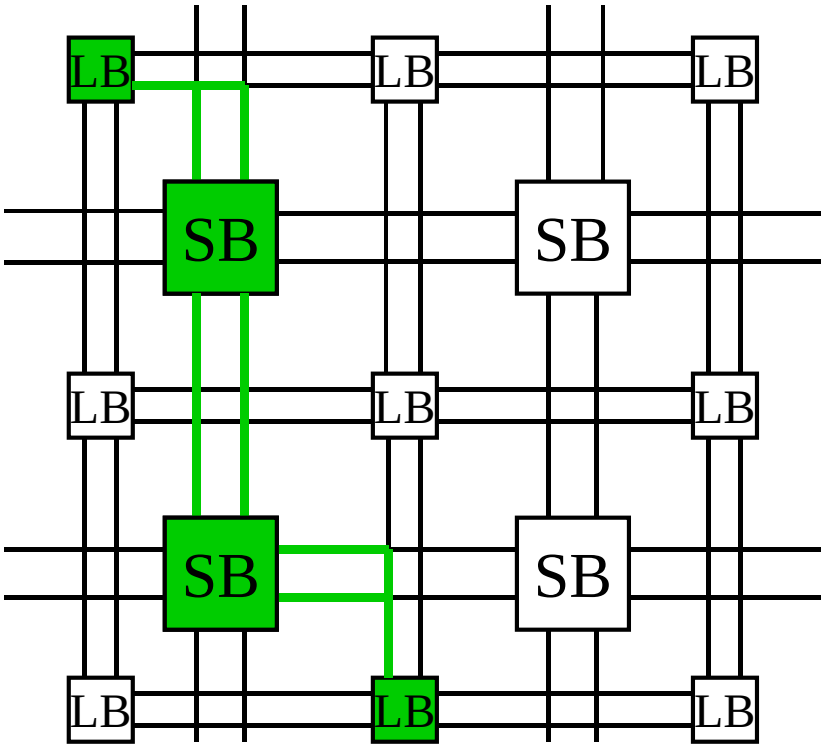
	Traditional P&R	Vivado P&R
"Cost" Criteria	1 dimension: timing minimization	3 dimensions: timing, congestion, wire length minimization
Primary Algorithm	"Simulated Annealing": Random, iterative search based on initial seed	Analytical: solves simultaneous equations to minimize all dimensions
Runtime	Unpredictable Due to random nature of algorithm. Blows up with congestion	Very predictable – Exponential with congestion
Scalability	Poor results as design approaches 1 M logic cells	Will handle 10M+ logic cells with predictable results

FPGA Physical Design

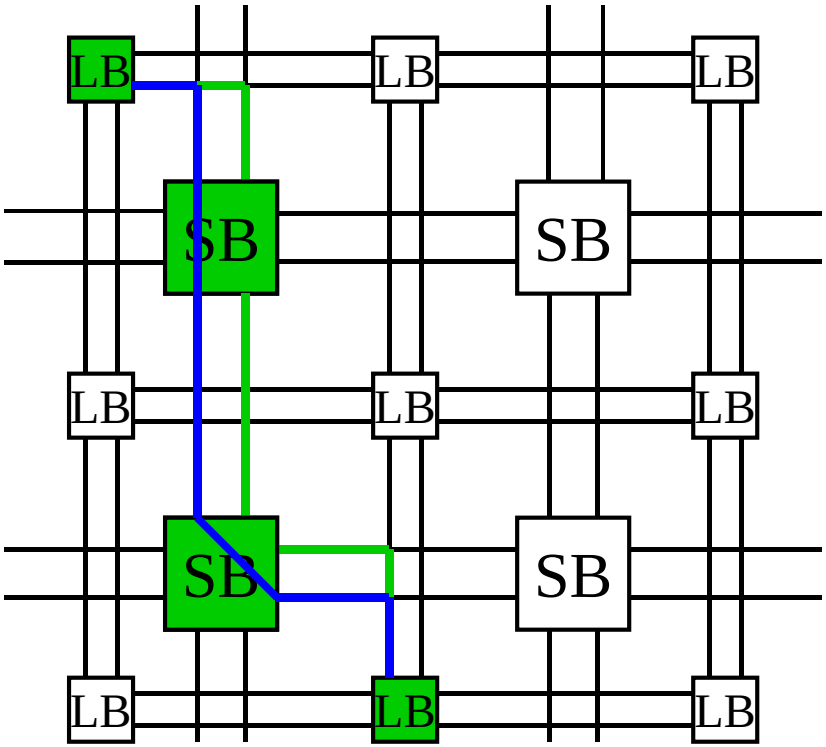


Global vs. Detailed Routing

- Global routing

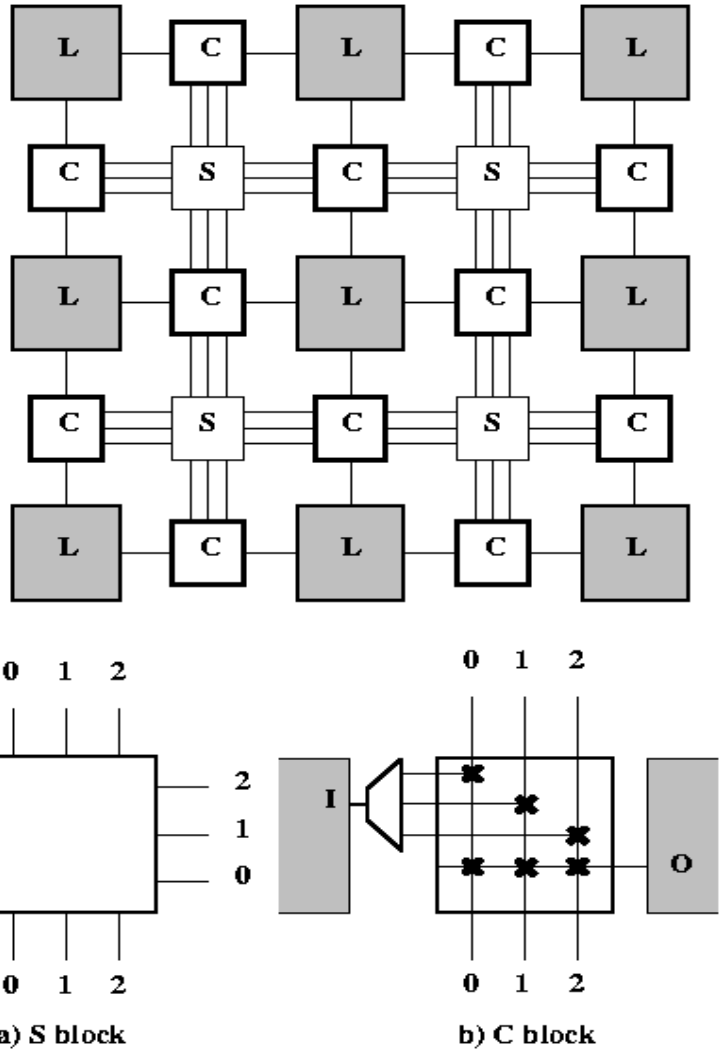


- Detailed routing



Routing is Architecture Dependent

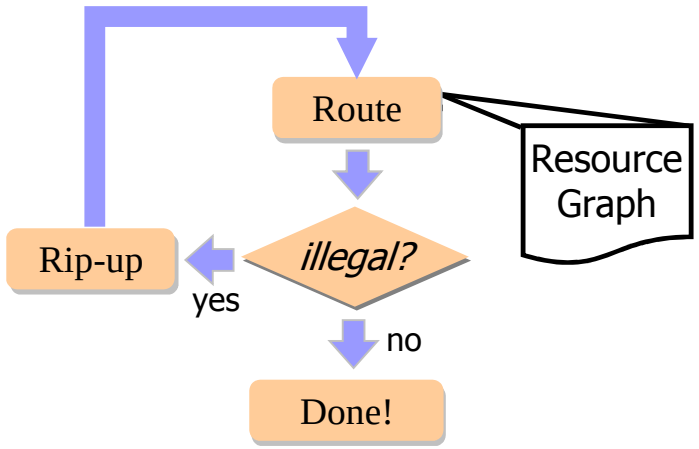
- Connection Boxes “C”
  - Flexibility, FC (# of wires each logic pin can connect to)
  - Topology (pattern of switches)
- Switch Boxes “S”
  - Flexibility, FS
  - Topology
- Length of wires:
  - Single Length Lines
  - Double Length Lines



FPGA Routing: VPR

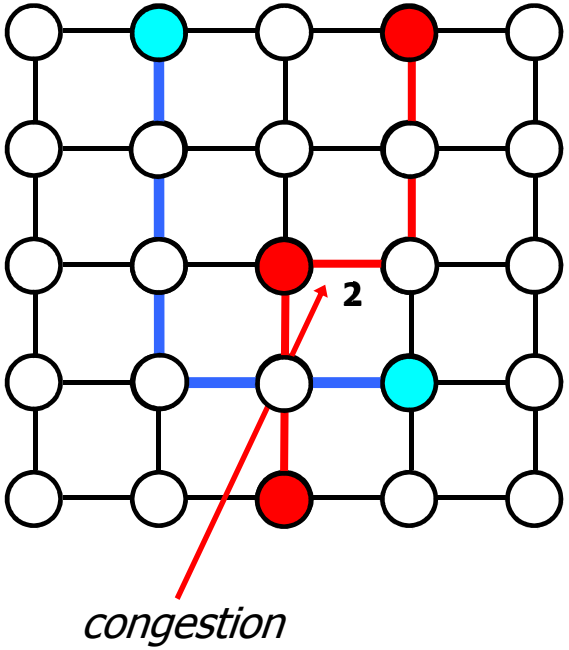
VPR – Versatile Place and Route *[Betz, et al., 1997]*

- Uses a ***Pathfinder algorithm***
- Increase performance over original Pathfinder algorithm
- Routability-driven routing
  - Goal: Use fewest tracks possible
- Timing-driven routing
  - Goal: Optimize circuit speed

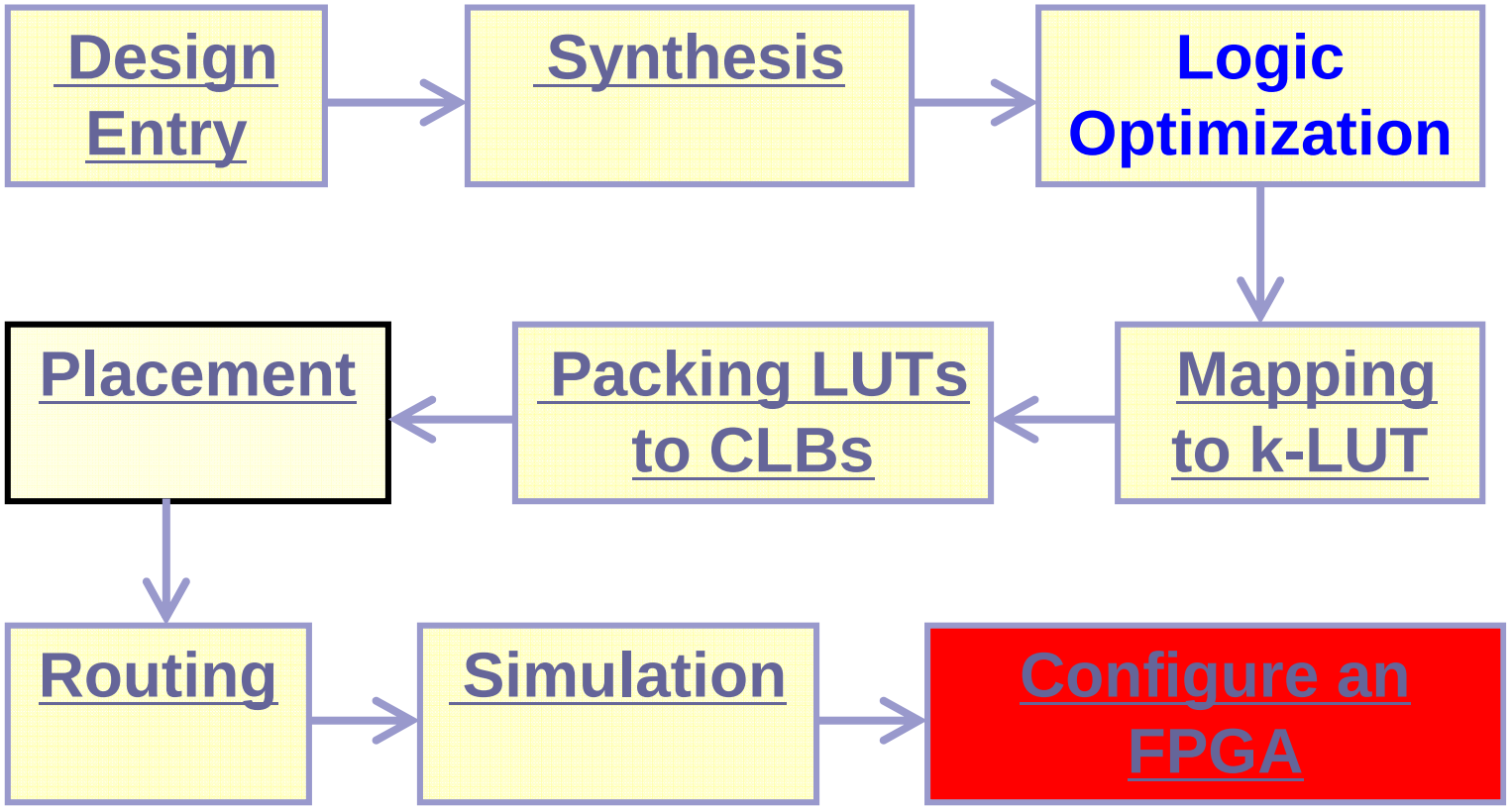


FPGA Routing: PathFinder

- Pathfinder *[Ebeling, et al., 1995]*
  - Introduced negotiated congestion
  - During each routing iteration, route nets using shortest path
    - Allows overuse *(congestion)* of routing resources
  - If congestion exists *(illegal routing)*
    - **Update cost** of congested resources based on the amount of overuse
    - Rip-up all routes and reroute all nets

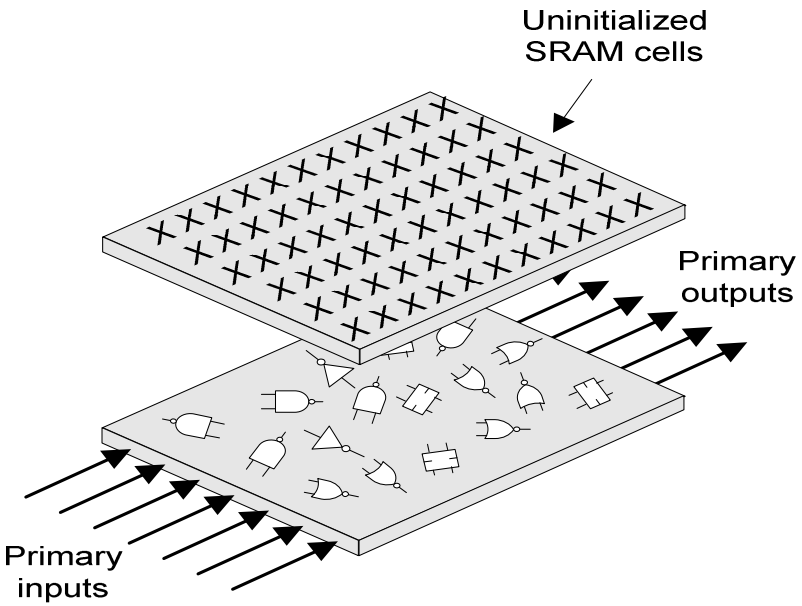


FPGA Physical Design

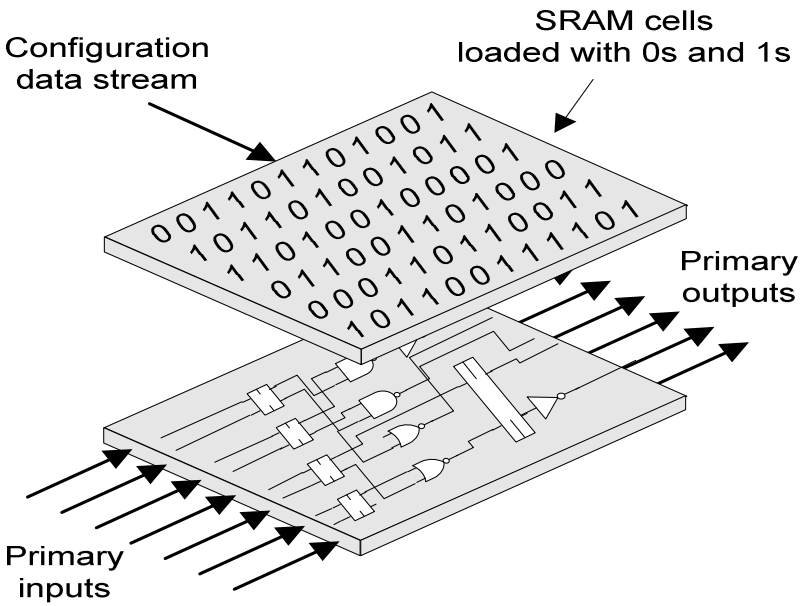


FPGA Configuration

- Following all CAD steps the end result is a *configuration file* which contains the information that will be uploaded into the FPGA in order to program it to perform a specific function.
- Interfaces available to configure the FPGA?.



(a) Unconfigured



(b) Configured

End Slides