Placement
Introduction

• A very important step in physical design cycle.
  – A poor placement requires larger area.
  – Also results in performance degradation.

• It is the process of arranging a set of modules on the layout surface.
  – Each module has fixed shape and fixed terminal locations.
  – A subset of modules may have pre-assigned positions (e.g., I/O pads).
The Placement Problem

• **Inputs:**
  – A set of modules with
    • well-defined shapes
    • fixed locations of pins.
  – A netlist.

• **Requirements:**
  – Find locations for each module so that no two modules overlap.
  – The placement is routable.

• **Objectives:**
  – Minimize layout area.
  – Reduce the length of critical nets.
  – Completion of routing.
Placement Problems at Different Levels

1. **System-level placement**
   - Place all the PCBs together such that
     - Area occupied is minimum
     - Heat dissipation is within limits.

2. **Board-level placement**
   - All the chips have to be placed on a PCB.
     - Area is fixed
     - All modules of rectangular shape
   - Objective is to
     - Minimize the number of routing layers
     - Meet system performance requirements.
3. **Chip-level placement**

- Normally, floorplanning / placement carried out along with pin assignment.
- Limited number of routing layers (2 to 4).
  - Bad placements may be unroutable.
  - Can be detected only later (during routing).
  - Costly delays in design cycle.
- Minimization of area.
Problem Formulation

• **Notations:**

  - \( B_1, B_2, \ldots, B_n \): modules/blocks to be placed
  - \( w_i, h_i \): width and height of \( B_i \), \( 1 \leq i \leq n \)
  - \( N = \{ N_1, N_2, \ldots, N_m \} \): set of nets (i.e. the netlist)
  - \( Q = \{ Q_1, Q_2, \ldots, Q_k \} \): rectangular empty spaces for routing
  - \( L_i \): estimated length of net \( N_i \), \( 1 \leq i \leq m \)
• **The problem**

Find rectangular regions $R = \{R_1, R_2, \ldots, R_n\}$ for each of the blocks such that

- Block $B_i$ can be placed in region $R_i$.
- No two rectangles overlap, $R_i \cap R_j = \emptyset$.
- Placement is routable (Q is sufficient to route all nets).
- Total area of rectangle bounding $R$ and $Q$ is minimized.
- Total wire length $\sum L_i$ is minimized.
- For high performance circuits, $\max \{L_i \mid i=1,2,\ldots,m\}$ is minimized.

- General problem is NP-complete.
- Algorithms used are heuristic in nature.
CAD for VLSI
Interconnection Topologies

• The actual wiring paths are not known during placement.
  – For making an estimation, a placement algorithm needs to model the topology of the interconnection nets.
    • An interconnection graph structure is used.
    • Vertices are terminals, and edges are interconnections.
Estimation of Wirelength

• The speed and quality of estimation has a drastic effect on the performance of placement algorithms.
  – For 2-terminal nets, we can use Manhattan distance as an estimate.
  – If the end co-ordinates are \((x_1,y_1)\) and \((x_2,y_2)\), then the wire length
    \[
    L = |x_1 - x_2| + |y_1 - y_2|
    \]
• How to estimate length of multi-terminal nets?
Modeling of Multi-terminal Nets

1. **Complete Graph**
   - \( \binom{n}{2} = \frac{n(n-1)}{2} \) edges for a n-pin net.
   - A tree has \((n-1)\) edges which is \(\frac{2}{n}\) times the number of edges of the complete graph.
   - Length is estimated as \(\frac{2}{n}\) times the sum of the edge weights.

2. **Minimum Spanning Tree**
   - Commonly used structure.
   - Branching allowed only at pin locations.
   - Easy to compute.
3. **Rectangular Steiner Tree**
   - A Steiner tree is the shortest route for connecting a set of pins.
   - A wire can branch from any point along its length.
   - Problem of finding Steiner tree is NP-complete.

4. **Semi Perimeter**
   - Efficient and most widely used.
   - Finds the smallest bounding rectangle that encloses all the pins of the net to be connected.
   - Estimated wire length is half the perimeter of this rectangle.
   - Always underestimates the wire length for congested nets.
Example

Complete Graph

Minimum Spanning Tree

Steiner Tree

Semi Perimeter
Design Style Specific Issues

• **Full Custom**
  – Placing a number of blocks of various shapes and sizes within a rectangular region.
  – Irregularity of block shapes may lead to unused areas.

• **Standard Cell**
  – Minimization of the layout area means:
    • Minimize sum of channel heights.
    • Minimize width of the widest row.
    • All rows should have equal width.
  – Over-the-cell routing leads to almost “channel-less” standard cell designs.
• **Gate Arrays**
  - The problem of partitioning and placement are the same in this design style.
  - For FPGA’s, the partitioned sub-circuit may be a complex netlist.
    • Map the netlist to one or more basic blocks (placement).
Classification of Placement Algorithms

Placement Algorithms

Simulation Based
- Simulated Annealing
- Simulated Evolution
- Force Directed

Partitioning Based
- Breuer’s Algorithm
- Terminal Propagation

Other
- Cluster Growth
- Force Directed
Simulated Annealing

- Simulation of the annealing process in metals or glass.
  - Avoids getting trapped in local minima.
  - Starts with an initial placement.
  - Incremental improvements by exchanging blocks, displacing a block, etc.
  - Moves which decrease cost are always accepted.
  - Moves which increase cost are accepted with a probability that decreases with the number of iterations.

- Timberwolf is one of the most successful placement algorithms based on simulated annealing.
Algorithm SA_Placement
begin
    T = initial_temperature;
P = initial_placement;
    while ( T > final_temperature) do
        while (no_of_trials_at_each_temp not yet completed) do
            new_P = PERTURB (P);
            ΔC = COST (new_P) – COST (P);
            if (ΔC < 0) then
                P = new_P;
            else if (random(0,1) > exp(ΔC/T)) then
                P = new_P;
            end
        T = SCHEDULE (T); /** Decrease temperature **/ 
    end
end
TimberWolf

• One of the most successful placement algorithms.
  – Developed by Sechen and Sangiovanni-Vincentelli.

• Parameters used:
  – Initial_temperature = 4,000,000
  – Final_temperature = 0.1
  – \text{SCHEDULE}(T) = \alpha(T) \times T
    • \alpha(T) specifies the cooling rate which depends on the current temperature.
    • \alpha(T) is 0.8 when the cooling process just starts.
    • \alpha(T) is 0.95 in the medium range of temperature.
    • \alpha(T) is 0.8 again when temperature is low.
The SCHEDULE Function

1. Simulated Annealing
   - Simulated annealing is a method of optimization that uses a probabilistic approach to escape local minima in the search for a global minimum. It is inspired by the annealing process in metallurgy, where a material is heated and then slowly cooled to reduce defects and improve its structure.

2. SCHEDULE Function
   - The SCHEDULE function is used to define the order of operations in a process. It is often used in scheduling problems where tasks need to be executed in a specific sequence to optimize resource usage or meet deadlines.

Graphical representation:
A graph showing the relationship between iteration number and log of time. The graph indicates a decreasing trend, suggesting that as the iteration number increases, the log of time decreases.

Mathematical expression:
\[ \text{Log } T = \text{INIT-TEMP} \cdot \text{LNs(NEW-TEMP)} \cdot \text{NUM-ITER} \]
The PERTURB Function

- New configuration is generated by making a weighted random selection from one of the following:
  a) The displacement of a block to a new location.
  b) The interchange of locations between two blocks.
  c) An orientation change for a block.
     - Mirror image of the block’s x-coordinate.
     - Used only when a new configuration generated using alternative (a) is rejected.
The COST Function

- The cost of a solution is computed as:
  \[ \text{COST} = \text{cost}_1 + \text{cost}_2 + \text{cost}_3 \]
  where 
  - cost1: weighted sum of estimated length of all nets
  - cost2: penalty cost for overlapping
  - cost3: penalty cost for uneven length among standard cell rows.
  - Overlap is not allowed in placement.
  - Computationally complex to remove all overlaps.
  - More efficient to allow overlaps during intermediate placements.
- Cost function (cost2) penalizes the overlapping.
The algorithm starts with an initial set of placement configurations. Called the population.

The process is iterative, where each iteration is called a generation.

The individuals of a population are evaluated to measure their goodness.

To move from one generation to the next, three genetic operators are used:

- Crossover
- Mutation
- Selection
CROSSOVER Operator

• Choose a random cut point.
• Generate offsprings by combining the left segment of one parent with the right segment of the other.
  – Some blocks may get repeated, while some others may get deleted.
  – Various ways to deal with this problem.
• Number of times the “crossover” operator is applied is controlled by crossover rate.
MUTATION Operator

- Causes incremental random changes to an offspring produced by crossover.
- Most common is pairwise exchange.
- Number of times this is done is controlled by mutation rate.
SELECT Operator

• Select members for crossover based on their fitness value.
  – Obtained by evaluating a cost function.

• Higher the fitness value of a solution, higher will be the probability for selection for crossover.
Force Directed Placement

- Explores the similarity between placement problem and classical mechanics problem of a system of bodies attached to springs.
- The blocks connected to each other by nets are supposed to exert attractive forces on each other.
  - Magnitude of this force is directly proportional to the distance between the blocks.
    - Analogous to Hooke’s law in mechanics.
  - Final configuration is one in which the system achieves equilibrium.
A cell $i$ connected to several cells $j$ experiences a total force

$$F_i = \sum_j (w_{ij} \cdot d_{ij})$$

where $w_{ij}$ is the weight of connection between $i$ and $j$, $d_{ij}$ is the distance between $i$ and $j$.

- If the cell $i$ is free to move, it would do so in the direction of force $F_i$ until the resultant force on it is zero.
- When all cells move to their zero-force target locations, the total wire length is minimized.
• For cell i, if \((x_i^0, y_i^0)\) represents the zero-force target location, by equating the x- and y-components of the force to zero, we get

\[
\begin{align*}
\Sigma_j \left( (w_{ij} \times (x_j - x_i^0)) \right) &= 0 \\
\Sigma_j \left( (w_{ij} \times (y_j - y_i^0)) \right) &= 0
\end{align*}
\]

• Solving for \(x_i^0\) and \(y_i^0\), we get

\[
\begin{align*}
x_i^0 &= \left( \Sigma_j \left( w_{ij} \times x_j \right) \right) / \left( \Sigma_j w_{ij} \right) \\
y_i^0 &= \left( \Sigma_j \left( w_{ij} \times y_j \right) \right) / \left( \Sigma_j w_{ij} \right)
\end{align*}
\]

• Care should be taken to avoid assigning more than one cell to the same location.
Example
Force Directed Approach for Constructive Placement

- The basic approach can be generalized for constructive placement.
  - Starting with some initial placement, one module is selected at a time, and its zero-force location computed.
  - The process can be iterated to improve upon the solution obtained.
  - The order of the cells can be random or driven by some heuristic.
    - Select the cell for which $F_i$ is maximum.
Breuer’s Algorithm

- Partitioning technique used to generate placement.
- The given circuit is repeatedly partitioned into two sub-circuits.
  - At each level of partitioning, the available layout area is partitioned into horizontal and vertical subsections alternately.
  - Each of the sub-circuits is assigned to a subsection.
  - Process continues till each sub-circuit consists of a single gate, and has a unique place on the layout area.
Contd.

- **Different sequences of cut lines used:**
  1. Cut Oriented Min-Cut Placement
  2. Quadrature Placement
  3. Bisection Placement
  4. Slice Bisection Placement
- **These are illustrated diagrammatically.**
Illustration

Cut-Oriented

Quadrature

Bisection

Slice Bisection

CAD for VLSI
Terminal Propagation Algorithm

- Partitioning algorithms merely reduce net cut.
- Direct use of partitioning algorithms would increase net length.
  - Also increases congestion in the channels.
- To prevent this, terminal propagation is used.
  - When a net connecting two terminals is cut, a dummy terminal is propagated to the nearest pin on the boundary.
  - When this dummy terminal is generated, the partitioning algorithm will not assign the two terminals in each partition into different partitions, as this would not result in a minimum cut.
Illustration :: Terminal Propagation

- :: Dummy terminal
- :: Terminal
Cluster Growth

- In this constructive placement algorithm, bottom-up approach is used.
- Blocks are placed sequentially in a partially completed layout.
  - The first block (seed) is usually placed by the user.
  - Other blocks are selected and placed one by one.
- Selection of blocks is usually based on connectivity with placed blocks.
Contd.

- **Layouts produced are not usually good.**
  - Does not take into account the interconnections and other circuit features.

- **Useful for generating initial placements.**
  - For iterative placement algorithms.
Algorithm Cluster_Growth
begin
  B = set of blocks to be placed;
  Select a seed block S from B;
  Place S in the layout;
  B = B – S;
  while (B ≠ φ) do
    begin
      Select a block X from B;
      Place X in the layout;
      B = B – X;
    end;
end
Performance Driven Placement

• The delay at chip level plays an important role in determining the performance of the chip.
  – Depends on interconnecting wires.
• As the blocks in a circuit becomes smaller and smaller:
  – The size of the chip decreases.
  – Interconnection delay becomes a major issue in high-performance circuits.
• Placement algorithms for high-performance chips:
  – Allow routing of nets within timing constraints.
• Two major categories of algorithms:
  1. **Net-based approach**
     • Try to route the nets to meet the timing constraints on the individual nets instead of considering paths.
     • The timing requirement for each net has to be decided by the algorithm.
     • Usually a pre-timing analysis generates the bounds on the net-lengths which must be satisfied during placement.
  2. **Path-based approach**
     • Critical paths in the circuit are considered.
     • Try to place the blocks in a manner that the path length is within the timing constraint.