An Introduction to FPGA Placement

Yonghong Xu
Supervisor: Dr. Khalid
Island FPGA Architecture

Abstract logic block

X Channel Segment

Y Channel Segment

Input Pin

Output Pin

I/O block
An Abstract Logic Block

- **Inputs**
  - $I_1$
  - $I_2$
  - $I_3$
  - $I_4$

- **4-input LUT**

- **D-FF**

- **Out**

- **Clock**
IC Design Flow

- Specification
- Architectural design
- Logic design
- Circuit design
- Physical design
- Test/Fabrication
- Physical design
  - Partitioning
  - Placement
  - Routing
Placement Definition

- **Input:**
  - Netlist of logic blocks
  - I/O pads
  - Inter connections

- **Output:**
  - Coordinates \((x_i, y_i)\) for each block.
  - The total wire length is minimized.
  - Also need to consider delay and routability.
An Example

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

- NP (non-deterministic polynomial-time)
  - the class NP consists of all those decision problems whose positive solutions can be verified in polynomial time given the right information, or equivalently, whose solution can be found in polynomial time on a non-deterministic machine
- NP-Complete basically means that there is no efficient way to compute a solution using a computer program
- Approximate solutions for NP-complete problems can be found using heuristic method.
Major Placement Techniques

• Partitioning-Based Placement
• Simulated Annealing Placement
• Quadratic Placement
• Hybrid and Hierarchical Placement
Partitioning-Based Placement

- Partitioning: Decomposition of a complex system into smaller subsystems
- Partitioning-based placement recursively apply min-cut partitioning to map the netlist into the layout region
- Cutsize: the number of nets not contained in just one side of the partition
- e.g. FM partitioning algorithm
Partitioning-Based Placement

• Pros:
  • Open Cost Function(partitioning cost)
    • Minimize net cut, edge cut etc.
  • It is Move Based, Suitable to Timing Driven Placement

• Cons:
  • Lots of “indifferent” moves
  • May not work well with some cost functions
  • Multi partitioning
Simulated Annealing Placement

- Simulated annealing algorithm mimics the annealing process used to gradually cool molten metal to produce high quality metal structures
- Initial placement improved by iterative swaps and moves
- Accept swaps if they improve the cost
- Accept swaps that degrade the cost under some probability conditions to prevent the algorithm from being trapped in a local minimum
Simulated Annealing Placement

• Pros:
  • Open Cost Function
    • Wire length cost
    • Timing cost
  • Can Reach Globally Optimal Solution given enough time

• Cons:
  • Slow
Quadratic Placement

• Use squared wire length as objective function
• Minimize the objective function by solving linear equations
• Expand the design to entire chip area
Quadratic Placement

\[ \text{Min} \ [(x_1-x_3)^2 + (x_1-x_2)^2 + (x_2-x_4)^2] : F \]

\[ \frac{\delta F}{\delta x_1} = 0; \]
\[ \frac{\delta F}{\delta x_2} = 0; \]

\[ A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \quad B = \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \]
Quadratic Placement

• Pros:
  • Very fast
  • Can handle large design

• Cons:
  • Minimize squared wire length while not the wire length
  • Not Suitable for Simultaneous Optimization of Other Aspects of Physical Design
  • Gives Trivial Solutions without Pads
Hybrid and Hierarchical Placement

- The scale of logic circuits grows fast
- Compile time grows exponential with the number of logic blocks
- Hierarchical methods are introduced to reduce the compile time
- in different hierarchies use different algorithms
- E.g.
  - GORDIAN quadratic + partitioning
  - Dragon: partitioning + Simulated Annealing
Timing Optimization

- To minimize the longest path delay or maximize the minimum slack
- Existing Algorithms
  - Path based algorithms
  - Net weighting algorithms
Path Based Algorithms

• Directly minimize the longest path delay
• Accurate timing view during optimization
• High computational cost
• Popular approaches:
  • Explicitly reduce the maximum length of a set of paths; the set could be pre-computed or dynamically adjusted
  • Mathematical programming by introducing auxiliary variables (arrival time, required time)
Net Weighting Algorithms

- Timing criticalities are translated into net weights; then compute a placement which minimized total weighted delay or wire length
- Edges have smaller slack are assigned higher weight
- Edges shared by more path are assigned higher weight
- Less accurate, less computational cost
Dominating Placement algorithm: VPR

- Simulated Annealing based
- Adaptive annealing schedule
- Use delay profile of target FPGA to evaluate the delay of each connection
- Trades off between circuit delay and wiring cost optimization
S = RandomPlacement();
T = InitialTemperature();
R_limit = InitialR_limit();
Criticality_Exponent = ComputeNewExponent();

ComputeDelayMatrix();

while (ExitCriterion() == False) {         /* “Outer loop” */
    TimingAnalyze();                     /* Perform a timing-analysis and update each connections criticality */
    Previous_Wiring_Cost=Wiring_Cost(S);  /* wire-length minimization normalization term */
    Previous_Timing_Cost = Timing_Cost(S); /* delay minimization normalization term */

    while (InnerLoopCriterion() == False) { /* “Inner loop” */
        S_new = GenerateViaMove(S, R_limit);
        ΔTiming_Cost = Timing_Cost(S_new) - Timing_Cost(S);
        ΔWiring_Cost = Wiring_Cost(S_new) - Wiring_Cost(S);
        ΔC = λ·(ΔTiming_Cost/Prev_Timing_Cost) +
            (1-λ)·(ΔWiring_Cost/Previous_Wiring_Cost); /* new cost fcn */
        if (ΔC < 0) {
            S = S_new; /* Move is good, accept */
        } else {
            r = random (0,1);
            if (r < e^{-ΔC/T}) {
                S = S_new; /* Move is bad, accept anyway */
            }
        }
    } /* End “inner loop” */

    T = UpdateTemp();
    R_limit = UpdateR_limit();
    Criticality_Exponent = ComputeNewExponent();
} /* End “outer loop” */
VPR Cost Function

\[ \Delta C = \lambda \cdot \frac{\Delta \text{Time\_Cost}}{\text{Previous\_Time\_Cost}} + (1 - \lambda) \frac{\Delta \text{Wiring\_Cost}}{\text{Previous\_Wiring\_Cost}} \]

Wiring\_Cost = \sum_{i=1}^{N_{nets}} q(i) \cdot [bb_i(x) + bb_i(y)]

Time\_cost = \sum_{\forall i, j \in \text{circuit}} \text{Time\_Cost}(i, j)

Time\_cost(i, j) = Delay(i, j) \cdot \text{Criticality}(i, j)^{\text{criticality\_exponent}}
VPR Stopping Criteria

- Inner loop
  
  \[ \text{InnerNum} \cdot N_{\text{blocks}}^{4/3} \]

- Outer loop
  
  \[ T < 0.005 \times \text{Cost} / N_{\text{nets}} \]
VPR Temperature Update

\[ T_{\text{new}} = \alpha \cdot T_{\text{old}} \]

<table>
<thead>
<tr>
<th>Fraction of Moves Accepted (R\text{accept})</th>
<th>Temperature Update Factor (\alpha)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R\text{accept} &gt;0.96</td>
<td>0.5</td>
</tr>
<tr>
<td>0.8&lt;R\text{accept} &lt;=0.96</td>
<td>0.9</td>
</tr>
<tr>
<td>0.15&lt;R\text{accept} &lt;=0.8</td>
<td>0.95</td>
</tr>
<tr>
<td>R\text{accept} &lt;=0.15</td>
<td>0.8</td>
</tr>
</tbody>
</table>
Thank you