Simultaneous Placement and Clock Tree Construction for Modern FPGAs

Wuxi Li $^1$  Mehrdad E. Dehkordi $^2$  Stephen Yang $^2$  David Z. Pan $^1$

$^1$Electrical & Computer Engineering, University of Texas at Austin

$^2$Vivado Implementation Team, Xilinx Inc.
Outline

Introduction

Proposed Algorithms

Experimental Results

Conclusion
Outline

Introduction

Proposed Algorithms

Experimental Results

Conclusion
Placement for Modern FPGAs

**Input**  A netlist of cells (LUT, FF, DSP, RAM, . . .)

**Output**  Cell physical locations in the FPGA layout

**Objectives**  Wirelength, timing, power, routability, . . .

**Constraints**  Clock network feasibility, . . .
Xilinx UltraScale Clocking Architecture

- Layout is divided into a grid of clock regions (CRs)
- Clock network consists of routing Layer (HR/VR) and distribution Layer (HD/VD)
- 24 HR/VR/HD/VD tracks in each CR
- Clock tree consists of D-layer vertical trunk tree + R-layer 2-pin route
### Problem Statement

Simultaneous Placement and Clock Tree Construction Problem

<table>
<thead>
<tr>
<th><strong>Input</strong></th>
<th>A netlist of cells</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output</strong></td>
<td>A global placement solution</td>
</tr>
<tr>
<td></td>
<td>A clock routing solution</td>
</tr>
<tr>
<td><strong>Objectives</strong></td>
<td>Min. wirelength</td>
</tr>
<tr>
<td><strong>Constraints</strong></td>
<td>No logic resource overflow</td>
</tr>
<tr>
<td></td>
<td>No clock routing overflow</td>
</tr>
</tbody>
</table>
Previous Works

Simulated annealing-based approach
► [Lamoureux+, TRETS’08]
► Incorporating clock cost in objective
► Generic to any clocking architecture
► Slow convergence

Bounding box-based approach
► [Kuo+, ICCAD’17], [Pui+, ICCAD’17], [Li+, TODAES’18]
► Greedily shrinking clock net bounding boxes to reduce overflow
► Cheap computation and fast convergence
► Often overestimates clock routing demand
Our Contribution

- Explicit clock tree construction
- Solution space of clock routing $\rightarrow$ tree space
  Clock routing $\rightarrow$ tree space exploration process
- Inspired by branch-and-bound idea, an iterative algorithm is proposed to efficiently explore the tree space
- A Lagrangian relaxation-based clock tree construction technique is also proposed to achieve feasible clock routing solutions
- Experiments demonstrate the effectiveness/efficiency of our approach over previous works.
Overall Flow

Quadratic programming (min. wirelength) → Clock network planning (honor clock constr.) → CR-wise rough legalization (spread cells)
Overall Flow
Clock Network Planning Problem

Problem Statement

Input  A placement produced by quadratic programming
Output  A cell-to-CR assignment
         A clock routing solution
Objectives  Min. cell displacement
Constraints  No logic resource overflow
            No clock routing overflow

Mathematical Formulation

\[
\min_x \sum_{v \in V} \sum_{r \in R} D_{v,r} \cdot x_{v,r},
\]

s.t.  
\[
\begin{align*}
x_{v,r} & \in \{0, 1\}, \forall v \in V, \forall r \in R, \\
\sum_{r \in R} x_{v,r} & = 1, \forall v \in V, \\
\sum_{v \in V} A_v \cdot x_{v,r} & \leq C_r, \forall r \in R,
\end{align*}
\]

Exist a legal clock routing w.r.t. \( x \).
Branch-and-Bound Idea

Problem Properties
- Integer minimization problem with complex constraints
- Hard to solve directly
- Can be efficiently solved by relaxing some constraints

B&B Algorithm
- Keeping solving the relaxed problem in iteratively branching spaces
- Tracking the lowest cost of feasible solutions found as the upper bound of optimum
- Pruning branches with lower bound costs worse than this upper bound
Clock Network Planning Algorithm

\[
\begin{align*}
(cost^*, x^*, \gamma^*) & \leftarrow (+\infty, -, -) \\
\kappa^{(0)} & \leftarrow 1 \\
\kappa_{e,r} & \leftarrow 1 \\
\text{stack.push}(\kappa^{(0)})
\end{align*}
\]

\[
\begin{align*}
\kappa & \leftarrow \text{stack.fetch.top()}
\end{align*}
\]

\[
\begin{align*}
cost(\kappa), x(\kappa) & \leftarrow \text{cell-to-CR assign. with } \kappa
\end{align*}
\]

\[
\begin{align*}
\gamma(\kappa) & \leftarrow \text{clock routing of } x(\kappa)
\end{align*}
\]

\[
\begin{align*}
\text{No: } \gamma(\kappa) & \text{ is feasible?}
\end{align*}
\]

\[
\begin{align*}
\text{No: } & \text{ derive new constr. } K' \text{ from } \kappa \\
& \text{ remove } \kappa' \in K' \text{ with LB cost } \geq cost^*
\end{align*}
\]

\[
\begin{align*}
\text{Yes: } & \text{ if } cost(\kappa) < cost^* \\
& \text{ update } (cost^*, x^*, \gamma^*) \\
& \text{ push } \kappa' \in K' \text{ into stack}
\end{align*}
\]
Clock Network Planning Algorithm

\[(cost^*, x^*, \gamma^*) \leftarrow (+\infty, -, -)\]
\[\kappa_{e, r}^{(0)} \leftarrow 1\]
\[\text{stack.push}(\kappa_{e, r}^{(0)})\]

\[\kappa \leftarrow \text{stack.fetch \_top()}
\]

\[\text{cost}^*(\kappa), x^*(\kappa) \leftarrow \text{cell-to-CR assign. with } \kappa\]

\[\gamma^*(\kappa) \leftarrow \text{clock routing of } x^*(\kappa)\]

\[\gamma^*(\kappa) \text{ is feasible?}\]

- Yes
  - if \(\text{cost}^*(\kappa) < \text{cost}^*\)
    - update \((\text{cost}^*, x^*, \gamma^*)\)
    - derieve new constr. \(K'\) from \(\kappa\)
    - push \(\kappa' \in K'\) into stack

- No
  - remove \(\kappa' \in K'\) with LB cost \(\geq \text{cost}^*\)

Initialization

- Set the best solution found as NONE
- Allow any cell-to-CR assignment \(\kappa^{(0)}\)
- Initialize the stack with only \(\kappa^{(0)}\)
Clock Network Planning Algorithm

\[(\text{cost}^*, x^*, \gamma^*) \leftarrow (+\infty, -, -)\]

\[\kappa_e, r \leftarrow 1\]

stack.push(\(\kappa(0)\))

\[\kappa \leftarrow \text{stack.fetch.top()}\]

\[\text{cost}(\kappa), x(\kappa) \leftarrow \text{cell-to-CR assign. with } \kappa\]

\[\gamma(\kappa) \leftarrow \text{clock routing of } x(\kappa)\]

\(\gamma(\kappa)\) is feasible?

Yes

No

derive new constr. \(K'\) from \(\kappa\)

if \(\text{cost}(\kappa) < \text{cost}^*\)

update \((\text{cost}^*, x^*, \gamma^*)\)

remove \(\kappa' \in K'\) with LB cost \(\geq \text{cost}^*\)

push \(\kappa' \in K'\) into stack

Cell-to-CR assignment problem

- Relax the clock constraint
- Solve the clock-unconstrained version of the original problem in subspace \(\kappa\)

\[
\begin{align*}
\min_x & \quad \sum_{v \in V} \sum_{r \in R} D_{v,r} \cdot x_{v,r}, \\
\text{s.t.} & \quad x_{v,r} \in \{0, 1\}, \quad \forall v \in V, \forall r \in R, \\
& \quad \sum_{r \in R} x_{v,r} = 1, \quad \forall v \in V, \\
& \quad \sum_{v \in V} A_v \cdot x_{v,r} \leq C_r, \quad \forall r \in R,
\end{align*}
\]

Exist a legal clock routing w.r.t. \(x\).

\[x_{v,r} = 0, \forall (v, r) \in \{\exists e \in E(v) \text{ s.t. } \kappa_{e, r} = 0\}.\]
Clock Network Planning Algorithm

\[(cost^*, x^*, \gamma^*) \leftarrow (+\infty, -, -)\]
\[\kappa_0 \leftarrow 1\]
\[\text{stack.push}(\kappa_0)\]

\[\kappa \leftarrow \text{stack.fetch.top}()\]

\[\text{cost}(\kappa), x(\kappa) \leftarrow \text{cell-to-CR assign. with } \kappa\]

\[\gamma(\kappa) \leftarrow \text{clock routing of } x(\kappa)\]

\[\gamma(\kappa) \text{ is feasible?}\]

- Yes
  - derive new constr. \(K'\) from \(\kappa\)
  - if \(\text{cost}(\kappa) < \text{cost}^*\)
    - update \((\text{cost}^*, x^*, \gamma^*)\)
  - push \(\kappa' \in K'\) into stack

- No
  - remove \(\kappa' \in K'\) with LB cost \(\geq \text{cost}^*\)

Cell-to-CR assignment problem

- Can be nearly optimally solved by a minimum-cost flow approximation
Clock Network Planning Algorithm

\[(\text{cost}^*, x^*, \gamma^*) \leftarrow (+\infty, -, -)\]
\[\kappa_{e,r}^{(0)} \leftarrow 1\]
\[\text{stack.push}(\kappa_{e,r}^{(0)})\]

\[\kappa \leftarrow \text{stack.fetch.top()}\]

\[\text{cost}(\kappa), x(\kappa) \leftarrow \text{cell-to-CR assign. with } \kappa\]

\[\gamma(\kappa) \leftarrow \text{clock routing of } x(\kappa)\]

\[\gamma(\kappa) \text{ is feasible?}\]

- No
  - derive new constr. \(K'\) from \(\kappa\)
  - remove \(\kappa' \in K'\) with LB cost \(\geq\) cost*

- Yes
  - if cost(\kappa) < cost*
    - update (cost*, x*, γ*)
  - push \(\kappa' \in K'\) into stack

D-layer clock tree candidates generation

- Each D-layer clock is a vertical trunk tree
- There are \(m\) candidates for each clock on a CR grid with \(m\) columns
- Total of \(m|E|\) clock tree candidates
Clock Network Planning Algorithm

Let $(cost^*, x^*, \gamma^*) \leftarrow (+\infty, -, -)$

$$\kappa_e, r \leftarrow 1$$

`stack.push(\kappa(0))`

$$\kappa \leftarrow \text{stack.fetch.top()}$$

$$cost(\kappa), x(\kappa) \leftarrow \text{cell-to-CR assign. with } \kappa$$

$$\gamma(\kappa) \leftarrow \text{clock routing of } x(\kappa)$$

If $\gamma(\kappa)$ is feasible?

- Yes
  - derive new constr. $K'$ from $\kappa$
  - update $(cost^*, x^*, \gamma^*)$
  - push $\kappa' \in K'$ into stack

- No
  - remove $\kappa' \in K'$ with LB cost $\geq cost^*$
  - if $cost(\kappa) < cost^*$

Clock tree candidate selection problem

- Minimize a topology-dependent cost
- Capacity constraints make the problem intractable
- Feasible solution may not exist

$$\min_x \sum_{t \in T} \phi_t \cdot z_t, \quad \text{s.t.} \quad z_t \in \{0, 1\}, \forall t \in T,$$

$$\sum_{t \in T(e)} z_t = 1, \forall e \in E,$$

$$\sum_{t \in T} H_{t, r} \cdot z_t \leq 24, \forall r \in R,$$

$$\sum_{t \in T} V_{t, r} \cdot z_t \leq 24, \forall r \in R.$$
Clock Network Planning Algorithm

Lagrangian relaxation of the candidate selection problem

- Relax the capacity constraints and introduce Lagrangian multipliers $\lambda$
- Iteratively solve the relaxed problem and update $\lambda$ until it converges

\[
\min_x \sum_{t \in T} (\phi_t + \lambda_t) \cdot z_t, \\
\text{s.t. } z_t \in \{0, 1\}, \forall t \in T, \\
\sum_{t \in T(e)} z_t = 1, \forall e \in E, \\
\sum_{t \in T} H_{t,r} \cdot z_t \leq 24, \forall r \in R, \\
\sum_{t \in T} V_{t,r} \cdot z_t \leq 24, \forall r \in R.
\]
Clock Network Planning Algorithm

\[
(c_{\ast}, x_{\ast}, \gamma_{\ast}) \leftarrow (+\infty, -, -)
\]
\[
\kappa_{c,r} \leftarrow 1
\]
\[
\text{stack.push}(\kappa_{0})
\]

\[
\kappa \leftarrow \text{stack.fetch.top()}
\]
\[
\text{cost}(\kappa), x(\kappa) \leftarrow \text{cell-to-CR assign. with } \kappa
\]
\[
\gamma(\kappa) \leftarrow \text{clock routing of } x(\kappa)
\]

If \( \gamma(\kappa) \) is feasible

- Update the best solution if \( \text{cost}(\kappa) \) is better than the previous best \( \text{cost}^{\ast} \)
- Fetch the next \( \kappa \) in the stack to explore other subspaces (branches)

- If \( \text{cost}(\kappa) < \text{cost}^{\ast} \), update \( (\text{cost}^{\ast}, x^{\ast}, \gamma^{\ast}) \)
- Derive new constr. \( K' \) from \( \kappa \)
- Remove \( \kappa' \in K' \) with LB cost \( \geq \text{cost}^{\ast} \)
- Push \( \kappa' \in K' \) into stack

\[
\kappa_{0} \in (\kappa_{0})
\]
\[
\text{cost}^{\ast} \rightarrow (-\infty, -, -)
\]
Clock Network Planning Algorithm

\[(cost^*, x^*, \gamma^*) \leftarrow (\infty, -, -)\]
\[\kappa(0) \leftarrow 1\]
\[
\text{stack.push}(\kappa(0))
\]
\[
\kappa \leftarrow \text{stack.fetch.top()}
\]
\[
\text{cost}(\kappa), x(\kappa) \leftarrow \text{cell-to-CR assign. with } \kappa
\]
\[
\gamma(\kappa) \leftarrow \text{clock routing of } x(\kappa)
\]
\[
\gamma(\kappa) \text{ is feasible?}
\]

**Derive new constraints } \kappa' \in K' \text{ from } \kappa**

- Each } \kappa' \text{ is a subspace of } \kappa
- Want } \kappa' \in K' \text{ can encourage more clock-friendly cell-to-CR assignment}
- Forbid some cell-to-CR assign. that can potentially reduce clock overflow on top of } \kappa

0. Derive new constr. } K' \text{ from } \kappa
1. if } \text{cost}(\kappa) < \text{cost}^* \text{ update } (\text{cost}^*, x^*, \gamma^*)
2. remove } \kappa' \in K' \text{ with LB cost } \geq \text{cost}^*
3. push } \kappa' \in K' \text{ into stack
Clock Network Planning Algorithm

\[(\text{cost}^*, x^*, \gamma^*) \leftarrow (+\infty, -, -)\]
\[\kappa(0) \leftarrow 1\]
\[\text{stack.push}(\kappa(0))\]
\[\kappa \leftarrow \text{stack.fetch.top()}\]
\[\text{cost}(\kappa), x(\kappa) \leftarrow \text{cell-to-CR assign. with } \kappa\]
\[\gamma(\kappa) \leftarrow \text{clock routing of } x(\kappa)\]
\[\gamma(\kappa) \text{ is feasible?} \]
\[\text{if } \text{cost}(\kappa) < \text{cost}^* \text{ update } (\text{cost}^*, x^*, \gamma^*)\]
\[\text{derive new constr. } K' \text{ from } \kappa\]
\[\text{remove } \kappa' \in K' \text{ with LB cost } \geq \text{cost}^*\]
\[\text{push } \kappa' \in K' \text{ into stack}\]

Remove suboptimal subspaces $\kappa' \in K'$

- We can safely prune subspaces with LB costs no better than $\text{cost}^*$

\[
\text{cost}_{\text{LB}}(\kappa) = \sum_{v \in V} \min_{r \in R} \{r \in R | \kappa_{e,r} = 1, \forall e \in E(v)\} D_{v,r}.
\]

\[
\min_x \sum_{v \in V} \sum_{r \in R} D_{v,r} \cdot x_{v,r},
\]
\[\text{s.t. } x_{v,r} \in \{0, 1\}, \forall v \in V, \forall r \in R,\]
\[\sum_{r \in R} x_{v,r} = 1, \forall v \in V,\]
\[\sum_{v \in V} A_v \cdot x_{v,r} \leq C_r, \forall r \in R,\]
\[x_{v,r} = 0, \forall (v, r) \in \{\exists e \in E(v) \text{ s.t. } \kappa_{e,r} = 0\}.\]
Outline

Introduction

Proposed Algorithms

Experimental Results

Conclusion
Experimental Setup

Implemented in C++ on top of placement framework [Li+, TCAD’18]

Linux, Intel Core i9-7900X CPUs (3.30 GHz and 10 cores) and 128 GB RAM

Routed by Vivado v2016.4

ISPD 2017 contest benchmark

- Xilinx UltraScale architecture
- 0.5M - 1.0M cells
- 32 - 58 clock nets
Comparison with Other State-of-the-Art Placers

Achieved the best routed WL with feasible clock routing

- On average, outperforms [Li+, TODAES’18] / [Kuo+, ICCAD’17] / [Pui+, ICCAD’17] / [Li+, TCAD’18] by 4.3% / 0.5% / 2.0% / 1.4% in routed WL
Comparison with Other State-of-the-Art Placers

Achieved the best runtime

- On average, runs $2.70\times / 1.64\times / 2.66\times$ faster than [Li+, TODAES’18] / [Pui+, ICCAD’17] / [Li+, TCAD’18]
- [Kuo+, ICCAD’17] did not report placement runtime

<table>
<thead>
<tr>
<th></th>
<th>CLK-FPGA01</th>
<th>CLK-FPGA02</th>
<th>CLK-FPGA03</th>
<th>CLK-FPGA04</th>
<th>CLK-FPGA05</th>
<th>CLK-FPGA06</th>
<th>CLK-FPGA07</th>
<th>CLK-FPGA08</th>
<th>CLK-FPGA09</th>
<th>CLK-FPGA10</th>
<th>CLK-FPGA11</th>
<th>CLK-FPGA12</th>
<th>CLK-FPGA13</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Norm. Routed WL</strong></td>
<td>Li+ TODAES’18</td>
<td>Pui+ ICCAD’17</td>
<td>Li+ TCAD’18</td>
<td>Proposed</td>
<td>Li+ TODAES’18</td>
<td>Pui+ ICCAD’17</td>
<td>Li+ TCAD’18</td>
<td>Proposed</td>
<td>Li+ TODAES’18</td>
<td>Pui+ ICCAD’17</td>
<td>Li+ TCAD’18</td>
<td>Proposed</td>
<td></td>
</tr>
</tbody>
</table>
Comparison Under Different Clock Capacities (CC)

Apple-to-apple comparison with [Li+, TODAES’18] under different CC

![Graph 1: Normalized Routed WL vs Clock Capacity]

![Graph 2: Number of Successes vs Clock Capacity]

![Graph 3: Normalized Runtime vs Clock Capacity]
Branch-and-Bound Tree Exploration

- First 30 feasible solutions found in clock network planning algorithm
- The best solution among them is achieved at #27, which is 4% better than #1
Outline

Introduction

Proposed Algorithms

Experimental Results

Conclusion
Conclusion

- A generic FPGA placement framework that simultaneously optimizes placement quality and ensures clock feasibility by explicit clock tree construction.

- The proposed framework significantly reduces the placement quality degradation while honoring the clock feasibility for designs with high clock utilization.

- A branch-and-bound-inspired clock network planning algorithm and a Lagrangian relaxation-based clock tree construction technique are proposed.

- The proposed approach outperforms other state-of-the-art approaches in routed wirelength with competitive runtime.
Thank You!