Algorithm for Synthesis and Exploration of Clock spines

Youngchan Kim, Taewhan Kim
Department of Electrical and Computer Engineering
Seoul National University, Korea
yckim@snucad.snu.ac.kr, tkim@ssl.snu.ac.kr
Clock Distribution Network

**Clock Tree**
- Low resource consumption
- Suffers timing variation

**Clock Mesh**
- High resource consumption
- Tolerance to timing variation

**Cross Links**
- Compromises clock resource with timing variation

**Clock spine**
- Another alternative to the clock tree with links
Clock Spine Network Structure

- **Clock spine**
  - Horizontal or vertical wire

- **Sinks with stubs**
  - Every sink is attached to its nearby spine

- **Spine buffers**
  - Drive clock spines and sinks

- **Top-level tree**
  - Deliver the clock signal from Source to sink

- **Multiple clock paths**
  - Tolerance to timing variation
Overall Synthesis Flow

1. Synthesize the clock spine
   - Spine allocation and placement
   - Stub allocation
   - Buffer allocation

2. Refine the spine network
   - Spine merging
   - Spine extension

3. Synthesize top-level tree
   - Drives spine buffers
Algorithm for Synthesizing Clock Spine Networks

1. Clock spine allocation and placement problem

2. Transform into SFO problem

3. Solve SFO problem

4. Refinement

Application of SFO algorithm, Simulated annealing based algorithm

Slicing floorplan optimization problem, Single clock spine allocation and placement problem

Spine merging, Spine extension
Clock Spine Allocation and Placement Problem

**Given**
- Placed clock sinks
- Buffer library $\mathcal{B}$

**Constraints**
- $L_{\text{bound}}$: minimum length of clock spine
- $l_{\text{bound}}$: maximum length of stub wire
- The number of clock spines $N$

**Objective**
- Generate a clock spine network $S$
- Minimizes the quantity
  \[
  C(S) = \alpha_1 W_{\text{sp}}(S) + \alpha_2 W_{\text{st}}(S) + \beta B\mathcal{A}(S)
  \]
**Slicing Floorplan Optimization Problem**

**Given**
- Plane of $m \times n$ grids
- Cost function $f(b)$ for a rectangle block
- Parameters
  - The number of sliced blocks $N$

**Objective**
- Generate a slicing floorplan $\mathcal{F}$
  - by recursively slicing the plane $N - 1$ times
  - Minimizes the quantity
    \[
    C'(\mathcal{F}) = f(b_1) + f(b_2) + \cdots + f(b_N)
    \]
Transformation of CSAP Problem into SFO Problem

Transforming an instance of CSAP problem into an equivalent instance of SFO problem:

- Assign only one clock spine to one block $b$
- Set cost function $f(b_i)$ for a rectangle block $b_i$ as $c(s_i)$
- $f(b_i) = c(s_i) = \alpha_1 W_{sp}(s_i) + \alpha_2 W_{st}(s_i) + \beta BA(s_i)$

Need “good single spine allocation and placement”
Single Clock Spine Allocation and Placement

Given
A set of clock sinks in a block \( b \)
Buffer library \( B \)
Parameters
- \( L_{\text{bound}} \)
- \( l_{\text{bound}} \)

Objective
Allocate a clock spine \( s \) to one block
Minimizes the quantity
\[
f(b_i) = c(s_i) = \alpha_1 WL_{sp}(s_i) + \alpha_2 WL_{st}(s_i) + \beta BA(s_i)
\]
How to Allocate One Spine to One Block

**Spine directions**
- Horizontal / Vertical
- Direction with lower cost is selected
- $L_{\text{bound}}$

**Stub allocation**
- Determined by position of spine
- Sweep the spine back and forth
- Minimize $W_{\text{st}}(.)$ cost
- $L_{\text{bound}}$

**Buffer allocation**
- With buffer library $\mathcal{B}$
- Exhaustive trial
- Set maximum driving capacitance for each buffer in $\mathcal{B}$
- Meet slew constraint
- Minimize $BA(.)$ cost
Application of SFO Algorithm

- Any SFO algorithm can be applied
  Simulated annealing (SA) method based on the postfix expression
  - proposed by Wong and Liu [16]

- Postfix expression of slicing floorplan

Application of SFO Algorithm

Move operations
To traverse solution space of SFO problem
- Change the index of operator
- Complement an operator
- Swap operators
- Swap an operator with a bullet

Cost function for simulated annealing
\[ C'(F) = f(b_1) + f(b_2) + \cdots + f(b_N) \]
- \[ f(b_i) = c(s_i) = \alpha_1 WL_{sp}(s_i) + \alpha_2 WL_{st}(s_i) + \beta BA(s_i) \]
Refinement of Clock Spine Network

**Isolated clock sinks**
Variability vulnerable sinks
To convert all isolated clock sinks to unisolated ones

**Spine Merging**
Merge two nearby spines
- Distance threshold $d_{\text{max}}$
- Resize nearby buffers to drive merged spine
Refinement of Clock Spine Network

Isolated clock sinks
Variability vulnerable sinks
To convert all isolated clock sinks to unisolated ones

Spine extension
Extend the spines
- Distance threshold $d_{\text{max}}$
- Insert new buffer to drive extended portion
- Resize nearby buffers to drive extended spine
Experimental Environments

Libraries and benchmarks
- PTM 45nm library [18]
- ISPD 2010 benchmarks [17]

Clock environment
- Supply voltage: 1.0V
- Period: 0.5ns
- Maximum slew rate: 50ps

Simulation
- 100 Monte Carlo runs for each CDN
- Synopsys HSPICE simulation

Process variation
- $x_i \sim x_i^0 + N(0, \sigma_i^2)$
  - $x_i^0$: nominal value of random variable $x_i$
  - $\sigma_i$: standard deviation of random variable $x_i$, set to 10% of nominal value

Variables:
- Channel length of buffer
- Wire width
- Sink input capacitance
Experimental Environments

Clock Tree
Synthesized by [3]
Zero skew tree

Clock Mesh with top-level tree
Synthesized by [7]
Top-level tree is synthesized by [3]

Clock Spine with top-level tree
Top-level tree is synthesized by [3]
Global clock skew
Unit: ps
Mean value of 100 Monte Carlo runs
38% decreased skew than clock tree
8.7% higher than clock mesh
Experimental Results

Total wire length
Unit: μm
5% decreased wire usage than clock tree
Experimental Results

Total buffer area

Unit: $\mu m^2$
15% higher buffer area than clock tree
6% usage compared to the clock mesh
Experimental Results

Total power consumption

Unit: mW

11% less than clock tree
36% less than clock mesh
Experimental Results

Clock spine synthesis considering clock gating

Comparison of results of clock-gating aware clock spine synthesis in [14] and ours

Cost function is updated

\[ c(s_i) = \alpha_1 W L_{sp}(s_i) + \alpha_2 W L_{st}(s_i) + \beta BA(s_i) + \gamma PWR(s_i) \]

- 22% less clock skew
- 40% less wire length
- 60% less buffer area
- 45% less power consumption

Conclusions

- Addressed the problem of automating the synthesis of clock spine networks
  Never been automated as yet

- Clock spine is tolerant to the clock skew variation while using less resource and power
  Comparable to the clock mesh
Q & A