Graceful Register Clustering by Effective Mean Shift Algorithm for Power and Timing Balancing

Iris Hui-Ru Jiang

Ya-Chu Chang  Tung-Wei Lin  Gi-Joon Nam
Outline

- Introduction
- Preliminaries and problem formulation
- Effective mean shift
- Experimental results
- Conclusion
Why Register Clustering?

- Dynamic power!! Clock power dominates!!
- Reduce the switching capacitance in a clock network.

<table>
<thead>
<tr>
<th>Switching capacitance</th>
<th>Clock power saving</th>
<th>Other benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock sinks (Register capacitance)</td>
<td>Shared clocking circuitry; #leafs ↓</td>
<td>Smaller area</td>
</tr>
<tr>
<td>Clock network (Wirelength, clock buffers)</td>
<td>#leaf ↓ ⇒ depth ↓</td>
<td>Simpler topology and easier skew control</td>
</tr>
</tbody>
</table>

![Diagram of clock network with 8C_{FF} and 3C_{FF}]

Clock sinks (Register capacitance) and Clock network (Wirelength, clock buffers) show significant benefits in terms of reduced capacitance, clock power saving, and improved topology and skew control.
Two Register Cluster Designs

- **Rigid cell**
  - Discrete bits: 1, 2, 4, 8, 16, 32, 64

- **Flexible template**
  - Structured latch template:
    - 1, 2, 3~4, 5~8, 9~16, 17~32, 33~64

Dual-bit flip-flop

Single-bit flip-flop

Master latch

Slave latch

$D_1 \rightarrow$ Master latch \rightarrow Slave latch \rightarrow $Q_1$

$D_2 \rightarrow$ Master latch \rightarrow Slave latch \rightarrow $Q_2$

clk
Prior Work (1/3)

- In-placement or post-placement

Source: IBM
Prior Work (2/3)

- **Clique partitioning**
  - Constructs a clustering compatibility graph based on timing feasible regions
  - Extracts maximal cliques to form multi-bit registers without timing degradation

- **Up-to-date: [Seitanidis+, DAC-17]**
  - Clique enumeration + ILP
  - High complexities!
  - Scalability issue for large-scale design
Prior Work (3/3)

● **K-means**
  – Relaxes timing constraints to maximum displacement constraints
  – Starts with a prespecified # of clusters and initial cluster centers
  – Assigns registers to nearest clusters iteratively until convergence

● **State-of-the-art: Weighted K-means [Wu+, DAC-16]**
  – Is sensitive to initializations and outliers (distant from others)
  – Intends to form large clusters (nearly max. allowable bits)
  – Possibly moves outliers far away
  – Needs extra processes to fix over-displacement & size overflow
Investigations

- Creating large clusters or dragging outliers far away causes large disruption to placement thus incurring significant timing degradation
  - The more timing degradations, the more ECO efforts.
- We can save power even few registers are clustered
What’s a Good Register Clustering Algorithm?

- 1) Requires no prespecified number of clusters
- 2) Is insensitive to initializations
- 3) Is robust to outliers
- 4) Is tolerant of various register distributions
- 5) Is efficient and scalable
- 6) Balances power and timing
Our Contributions

- Propose **effective mean shift** to perform **graceful register clustering** for reducing clock power while minimizing timing degradation.

- Augment classic mean shift with special treatments for register clustering to attain these goals.

- Key idea: Conceptually, clusters are expected to reside in dense regions of registers. Our idea is to direct registers towards their nearest densest spots to form clusters naturally.
Classic Mean Shift

- Generate a density surface
- Iteratively shift each point uphill
- Time complexity is of $O(Tn^2)$: $T$ iterations, $n$ points

$$f(x) = \frac{1}{nh^d} \sum_{i=1}^{n} k\left(\frac{x - x_i}{h}\right)$$

$$m(x) = \frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)} - x$$

- peak
- outlier
- cluster
- data point
Problem Formulation

Register clustering

Min. #clusters
Min. displacement (Manhattan)
s.t. the cluster size constraint, Max. displacement constraints

Logic synthesis
Timing-driven placement
Register clustering
Legalization
Clock tree synthesis
Routing
Tape Out

Initial placement
Tech file
Register library

Clock tree report
Timing report
Outline

- Introduction
- Preliminaries and problem formulation
- Effective mean shift
- Experimental results
- Conclusion
## Classic vs. Adaptive vs. Effective

The register distribution is mapped to a density surface. Dense regions form hills.

<table>
<thead>
<tr>
<th>Density estimator</th>
<th>Adaptive Mean Shift (Variable Bandwidth)</th>
<th>Effective Mean Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{1}{nh^d} \sum_{i=1}^{n} k\left(\frac{x - x_i}{h_i}\right)$</td>
<td>$\frac{1}{n} \sum_{i=1}^{n} \frac{1}{h_i^d} k\left(\frac{x - x_i}{h_i}\right)$</td>
<td>$\frac{1}{n} \sum_{i \in \text{KNN}'(x)} \frac{1}{h_i^d} k\left(\frac{x - x_i}{h_i}\right)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Shift point</th>
<th>Adaptive Mean Shift</th>
<th>Effective Mean Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{\sum_{i=1}^{n} x_i g\left(\frac{|x - x_i|^2}{h}\right)}{\sum_{i=1}^{n} g\left(\frac{|x - x_i|^2}{h}\right)}$</td>
<td>$\frac{\sum_{i=1}^{n} x_i}{h_i^{d+2}} g\left(\frac{|x - x_i|^2}{h_i}\right)$</td>
<td>$\frac{\sum_{i \in \text{KNN}'(x)} x_i}{h_i^{d+2}} g\left(\frac{|x - x_i|^2}{h_i}\right)$</td>
</tr>
</tbody>
</table>

1. $k(x) = \kappa(\|x\|^2)$, Gaussian kernel
2. $g(x) = -\kappa'(x)$
3. $d = 2$
Overview

Logic synthesis → Timing-driven placement → Register clustering → Legalization → Clock tree synthesis → Routing → Tape Out

Effective Mean Shift

- For each register:
  - Identifying effective neighbors
  - Setting timing-aware bandwidth
  - Constructing density surface
  - Shifting to local maximum
  - Clustering by local maxima
  - Relocating clusters and registers

Initial placement → Tech file → Register library

Clock tree report → Timing report
Variable Bandwidth Selection

Set K-NN

Set Bandwidth

\[
\frac{1}{n} \sum_{i \in KNN(x)} \frac{1}{h_i} k\left(\frac{x - x_i}{h_i}\right)
\]

\[
h_i = \min\left(h_{\text{max}}, \alpha \|x_i - x_{i,M}\|\right)
\]

Cluster

Shift

Local max

register

M=1
Identifying Effective Neighbors

- Points that correspond to the tails of the underlying density function receive small weights, and thus they are almost automatically discarded.
- Consider only effective neighbors.
- Iteratively updating effective neighbors may still be computation intensive.
- Computing KNN only once
  - Neighbors barely change, effective neighbors can be identified only once (at the beginning).
  - Analysis of distinct neighbors (K=140)

<table>
<thead>
<tr>
<th>Circuit</th>
<th># of Iterations</th>
<th># of Total Distinct Neighbors</th>
<th># of Distinct Neighbors per Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Superblue16</td>
<td>213</td>
<td>158.25</td>
<td>0.74</td>
</tr>
<tr>
<td>Superblue18</td>
<td>315</td>
<td>158.09</td>
<td>0.50</td>
</tr>
<tr>
<td>Superblue10</td>
<td>533</td>
<td>156.13</td>
<td>0.29</td>
</tr>
</tbody>
</table>
Setting K-Nearest Neighbors

- Constraint: maximum displacement

\[
\Sigma_{i \in \text{KNN}(x)} \frac{x_i}{h_i} \frac{1}{d+2} g\left(\frac{\|x - x_i\|}{h_i}\right)
\]

\[
\Sigma_{i \in \text{KNN}(x)} \frac{1}{h_i^d} \frac{1}{d+2} g\left(\frac{\|x - x_i\|}{h_i}\right)
\]
Shifting to Local Density Maxima

- Each register undergoes the following steps to seek the local density maximum

1. Set the initial coordinates, \( y_j^0 = x_j, j = 1..n \)

2. Identify effective neighbors, \( KNN'(y_j^0) \); set bandwidth \( h_j \)
   - Then, the density surface is formed

3. Compute the mean shift vector \( m(y_j^t) \)

4. Shift each register, \( y_j^{t+1} = y_j^t + m(y_j^t) = \frac{\sum_{i \in KNN'(y_j^0)} x_i \frac{y_j^t - x_i}{h_i}}{\sum_{i \in KNN'(y_j^0)} \frac{1}{h_i}} \)

5. Iterate steps 3 and 4 until convergence, \( |y_j^{t+1} - y_j^t| < \delta \)
Clustering by Local Density Maxima

- Compensate the approximation error of KNN

Set K-NN  ➔  Set Bandwidth  ➔  Shift  ➔  Cluster

- (a) Small threshold
- (b) Medium threshold
- (c) Large threshold
Relocation for Timing and Displacement

- The previous steps in effective mean shift can be viewed as seeking the locations of clusters.

- Reassign registers and relocate clusters for improving timing and displacement.
  - Manhattan distance

- Relocate each cluster to the median coordinate of its register members for minimizing displacement and reducing timing degradation.
Complexity Analysis

- Classic mean shift: $O(Tn^2)$, $T$ iterations, $n$ registers

- Effective mean shift: $O(TKn + Cn)$, $K$ effective neighbors, $C$ clusters.
  - Shifting to local density maxima: $O(TKn)$ time, $K \ll n$
  - Register reassignment and cluster relocation: $O(Cn)$ time, $C \ll n$
Parallelization

Set KNN

Set Bandwidth

Shift to Local Maximum

For each register
- Identifying effective neighbors
- Setting timing-aware bandwidth
- Constructing density surface
- Shifting to local maximum

Thread0: Reg. 8m
Thread1: Reg. 8m+1
Thread2: Reg. 8m+2
Thread3: Reg. 8m+3
Thread4: Reg. 8m+4
Thread5: Reg. 8m+5
Thread6: Reg. 8m+6
Thread7: Reg. 8m+7

Start
Outline

- Introduction
- Preliminaries and problem formulation
- Effective mean shift
- Experimental results
- Conclusion
Experimental Setting

- C++; Intel Xeon 2.6 GHz CPU and 256 GB memory
- 2015 CAD contest in incremental timing-driven placement

<table>
<thead>
<tr>
<th>Circuit</th>
<th># of Cells</th>
<th># of Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>superblue16</td>
<td>981,559</td>
<td>142,543</td>
</tr>
<tr>
<td>superblue18</td>
<td>768,068</td>
<td>101,758</td>
</tr>
<tr>
<td>superblue4</td>
<td>796,645</td>
<td>167,731</td>
</tr>
<tr>
<td>superblue5</td>
<td>1,086,888</td>
<td>110,941</td>
</tr>
<tr>
<td>superblue3</td>
<td>1,213,253</td>
<td>163,107</td>
</tr>
<tr>
<td>superblue1</td>
<td>1,209,716</td>
<td>137,560</td>
</tr>
<tr>
<td>superblue7</td>
<td>1,931,639</td>
<td>262,176</td>
</tr>
<tr>
<td>superblue10</td>
<td>1,876,130</td>
<td>231,747</td>
</tr>
</tbody>
</table>

- Pseudo power of multi-bit register library

<table>
<thead>
<tr>
<th># of Bits</th>
<th>Normalized Pseudo Power per Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.000</td>
</tr>
<tr>
<td>2~3</td>
<td>0.860</td>
</tr>
<tr>
<td>4~7</td>
<td>0.790</td>
</tr>
<tr>
<td>8~15</td>
<td>0.755</td>
</tr>
<tr>
<td>16~31</td>
<td>0.738</td>
</tr>
<tr>
<td>32~63</td>
<td>0.729</td>
</tr>
<tr>
<td>64~80</td>
<td>0.724</td>
</tr>
</tbody>
</table>

- Cadence Innovus
### Effective Mean Shift vs. Weighted K-Means

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Method</th>
<th>Cluster Size</th>
<th>Displacement</th>
<th>Runtime (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Average</td>
</tr>
<tr>
<td>superblue16</td>
<td>WK</td>
<td>34</td>
<td>80</td>
<td>56000.54</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>1</td>
<td>55</td>
<td>22353.75</td>
</tr>
<tr>
<td>superblue18</td>
<td>WK</td>
<td>35</td>
<td>80</td>
<td>60843.50</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>1</td>
<td>70</td>
<td>25792.54</td>
</tr>
<tr>
<td>superblue4</td>
<td>WK</td>
<td>34</td>
<td>80</td>
<td>48129.71</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>1</td>
<td>56</td>
<td>19446.86</td>
</tr>
<tr>
<td>superblue5</td>
<td>WK</td>
<td>32</td>
<td>80</td>
<td>69453.46</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>1</td>
<td>78</td>
<td>29747.90</td>
</tr>
<tr>
<td>superblue3</td>
<td>WK</td>
<td>28</td>
<td>80</td>
<td>54968.00</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>1</td>
<td>79</td>
<td>25696.45</td>
</tr>
<tr>
<td>superblue1</td>
<td>WK</td>
<td>42</td>
<td>80</td>
<td>64158.15</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>1</td>
<td>62</td>
<td>24456.02</td>
</tr>
<tr>
<td>superblue7</td>
<td>WK</td>
<td>39</td>
<td>80</td>
<td>54761.63</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>1</td>
<td>79</td>
<td>26048.28</td>
</tr>
<tr>
<td>superblue10</td>
<td>WK</td>
<td>26</td>
<td>80</td>
<td>57643.75</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>1</td>
<td>79</td>
<td>27914.53</td>
</tr>
<tr>
<td><strong>Ratio</strong></td>
<td><strong>WK/Ours</strong></td>
<td><strong>2.33</strong></td>
<td><strong>215.03</strong></td>
<td><strong>39.42</strong></td>
</tr>
</tbody>
</table>

The maximum allowable cluster size is 80 (same setting as weighted K-means (WK))
The maximum allowable displacement is 400 nm
For effective mean shift, $K = 140$ for KNN
Convergence threshold $\delta = 0.0001$ units
Cluster merging threshold $\varepsilon = 5000$ units (2000 unit length = 1 nm)
## Timing and Power Comparison

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Method</th>
<th>Timing</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>WNS</td>
<td>TNS (ns)</td>
</tr>
<tr>
<td>superblue16</td>
<td>NC</td>
<td>-6.2</td>
<td>-1532.0</td>
</tr>
<tr>
<td></td>
<td>WK</td>
<td>-6.6</td>
<td>-2120.9</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>-6.2</td>
<td>-1629.8</td>
</tr>
<tr>
<td>superblue18</td>
<td>NC</td>
<td>-9.1</td>
<td>-5148.3</td>
</tr>
<tr>
<td></td>
<td>WK</td>
<td>-9.4</td>
<td>-5834.8</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>-9.7</td>
<td>-5250.0</td>
</tr>
<tr>
<td>superblue4</td>
<td>NC</td>
<td>-30.2</td>
<td>-19866.8</td>
</tr>
<tr>
<td></td>
<td>WK</td>
<td>-32.3</td>
<td>-20607.3</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>-30.3</td>
<td>-19898.6</td>
</tr>
<tr>
<td>superblue5</td>
<td>NC</td>
<td>-18.9</td>
<td>-7892.9</td>
</tr>
<tr>
<td></td>
<td>WK</td>
<td>-19.7</td>
<td>-8584.5</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>-18.9</td>
<td>-8106.1</td>
</tr>
<tr>
<td>superblue3</td>
<td>NC</td>
<td>-10.2</td>
<td>-6778.5</td>
</tr>
<tr>
<td></td>
<td>WK</td>
<td>-10.5</td>
<td>-7825.5</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>-10.2</td>
<td>-7334.7</td>
</tr>
<tr>
<td>superblue1</td>
<td>NC</td>
<td>-19.4</td>
<td>-12531.2</td>
</tr>
<tr>
<td></td>
<td>WK</td>
<td>-20.9</td>
<td>-13591.3</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>-19.2</td>
<td>-12757.0</td>
</tr>
<tr>
<td>superblue7</td>
<td>NC</td>
<td>-48.7</td>
<td>-151000.0</td>
</tr>
<tr>
<td></td>
<td>WK</td>
<td>-42.7</td>
<td>-139000.0</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>-42.3</td>
<td>-141000.0</td>
</tr>
<tr>
<td>superblue10</td>
<td>NC</td>
<td>-0.00%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WK</td>
<td>-10.88%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>-1.95%</td>
<td></td>
</tr>
</tbody>
</table>

**Average**
TNS Comparison

-40.00% -35.00% -30.00% -25.00% -20.00% -15.00% -10.00% -5.00% 0.00% 5.00% 10.00% 15.00% 20.00% 25.00% 30.00% 35.00% 40.00%

Weighted K-means

Ours

-38.44% -13.33% -6.82% -3.73% -2.70% -8.76% -8.21% -15.45% -8.46% 7.95% 6.62%

superblue16 superblue18 superblue4 superblue5 superblue3 superblue1 superblue7 superblue10
Clock Routed WL Comparison

Weighted K-means vs Ours

Percentage of improvements for different benchmarks:
- superblue16: 22.96% vs 21.03%
- superblue18: 22.88% vs 22.79%
- superblue4: 23.09% vs 21.08%
- superblue5: 31.37% vs 29.45%
- superblue3: 25.46% vs 25.07%
- superblue1: 25.04% vs 24.41%
- superblue7: 22.29% vs 21.28%
- superblue10: 24.60% vs 22.84%
Zooming In

superblue16

(a) Non-clustered  
(b) Weighted K-means  
(c) Effective mean shift
Non-Clustered

superblue4
Weighted K-means

superblue4
Ours: Effective Mean Shift

superblue4
Wirelength Optimum Sites

superblue4

- Based on wirelength optimum site of each register
Clock Sink Power vs. TNS Degradation

superblue16

<table>
<thead>
<tr>
<th>$M$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max cluster size</td>
<td>1</td>
<td>30</td>
<td>35</td>
<td>55</td>
<td>78</td>
<td>98</td>
</tr>
</tbody>
</table>

- $M=0$: WK
- $M=1$: Non-clustered
- $M=2$, $M=3$, $M=4$, $M=5$
Speedups by Multithreading

superblue18

- Runtime(s)

# of threads

1  2  3  4  5  6  7  8

138  74  53  41  34  30  27  25
Conclusion

1) Requires no prespecified number of clusters
   - Exploits the density of registers to generate clusters naturally

2) Is insensitive to initializations
   - Actually, no initial seeds are needed

3) Is robust to outliers
   - Our effective neighbor consideration and bandwidth setting prevent outliers in sparse regions from over-displacement

4) Is tolerant of various register distributions
   - According to local density and sparsity, our clustering can tolerate uneven register distribution

5) Is efficient and scalable
   - Our KNN and bandwidth setting expedites shift vector computation for each register, and our algorithm is highly parallelizable

6) Balances power and timing
   - Graceful register clustering!
Thank you!


References


Weighted K-Means

- Vanilla K-Means
  - Incur unbalanced cluster sizes

\[ \text{Cost} = |x_i - \mu_x(C_k)| + |y_i - \mu_y(C_k)| \]

- Weighted K-Means
  - Try to generate even-sized clusters

\[ \text{Cost} = (|x_i - \mu_x(C_k)| + |y_i - \mu_y(C_k)|) \times \max(\frac{|C_k|}{\text{size\_limit}}, 1) \]

- Introduce a cluster size balancing weight into displacement cost
- Intend to form large cluster (nearly max. allowable bits)
- Need additional over-displacement and over-size fixing
Variable Bandwidth Selection

- Reflect timing criticality and local distribution

\[
\begin{align*}
    f(x) &= \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h_i^d} k \left( \frac{x-x_i}{h_i} \right) \\
    m(x) &= \frac{\sum_{i=1}^{n} \frac{x_i}{h_i^{d+2}} g \left( \frac{\|x-x_i\|}{h_i} \right)}{\sum_{i=1}^{n} \frac{1}{h_i^{d+2}} g \left( \frac{\|x-x_i\|}{h_i} \right)} - x \\
    h_i &= \min(h_{\text{max}}, \alpha \left\| x_i - x_{i,M} \right\|) \\
    h_{\text{max}} &: \text{the maximum allowable displacement} \\
    \left\| x_i - x_{i,M} \right\| &: \text{the Euclidean distance between register } i \text{ and its } M\text{-th nearest neighbor (} x_{i,0} = x_i \text{)} \\
    \alpha &: \text{a timing criticality coefficient; } \alpha \to 0 \text{ for the most critical register (i.e., a very tall and skinny kernel)}
\end{align*}
\]
### Identifying Effective Neighbors

- Classic mean shift considers all original data points during shift vector computation.
- However, the points that correspond to the tails of the underlying density function receive small weights, and thus they are almost automatically discarded.
- Moreover, we do not expect registers to travel far away (for minimizing disturbance to timing and placement), and try to avoid oversized clusters.
- Thus, we can ignore distant registers.

\[
\begin{align*}
  f(x) &= \frac{1}{n} \sum_{i \in KNN_r(x)} \frac{1}{h_i^d} k \left( \frac{x - x_i}{h_i} \right) \\
  m(x) &= \frac{\sum_{i \in KNN_r(x)} \frac{x_i}{h_i^d+2} g \left( \frac{\|x - x_i\|^2}{h_i^2} \right)}{\sum_{i \in KNN_r(x)} \frac{1}{h_i^d+2} g \left( \frac{\|x - x_i\|^2}{h_i^2} \right)} - x
\end{align*}
\]
Relocation for Timing and Displacement

- The previous steps in effective mean shift can be viewed as seeking the locations of clusters.
- Reassign registers and relocate clusters for improving timing and displacement.
  - Reduce to stable matching
  - The capacity of a cluster location equals the maximum allowable cluster size.
  - The preference is ranked in non-decreasing order of displacement (Manhattan distance)
- Relocate each cluster to the median coordinate of its register members for minimizing displacement and reducing timing degradation.
Experimental Setting

- C++ programming language and compiled by G++ 4.8.5
- Intel Xeon 2.6 GHz CPU and 256 GB memory
- ICCAD-2015 CAD contest in incremental timing-driven placement benchmark suite
- Cadence Innovus

Table 3. Benchmark Statistics.

<table>
<thead>
<tr>
<th>Circuit</th>
<th># of Cells</th>
<th># of Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>superblue16</td>
<td>981,559</td>
<td>142,543</td>
</tr>
<tr>
<td>superblue18</td>
<td>768,068</td>
<td>101,758</td>
</tr>
<tr>
<td>superblue4</td>
<td>796,645</td>
<td>167,731</td>
</tr>
<tr>
<td>superblue5</td>
<td>1,086,888</td>
<td>110,941</td>
</tr>
<tr>
<td>superblue3</td>
<td>1,213,253</td>
<td>163,107</td>
</tr>
<tr>
<td>superblue1</td>
<td>1,209,716</td>
<td>137,560</td>
</tr>
<tr>
<td>superblue7</td>
<td>1,931,639</td>
<td>262,176</td>
</tr>
<tr>
<td>superblue10</td>
<td>1,876,130</td>
<td>231,747</td>
</tr>
</tbody>
</table>

Table 4. Pseudo Power of Multi-bit Register Library.

<table>
<thead>
<tr>
<th># of Bits</th>
<th>Normalized Pseudo Power per Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.000</td>
</tr>
<tr>
<td>2-3</td>
<td>0.860</td>
</tr>
<tr>
<td>4-7</td>
<td>0.790</td>
</tr>
<tr>
<td>8-15</td>
<td>0.755</td>
</tr>
<tr>
<td>16-31</td>
<td>0.738</td>
</tr>
<tr>
<td>32-63</td>
<td>0.729</td>
</tr>
<tr>
<td>64-80</td>
<td>0.724</td>
</tr>
</tbody>
</table>