Hierarchical Extraction and Verification of Symmetry Constraints for Analog Layout Automation

Sambuddha Bhattacharya, Nuttorn Jangkrajarng, Roy Hartono and C-J. Richard Shi

Department of Electrical Engineering, University of Washington
Seattle, WA 98195, USA
{sbb, njangkra, rhartono, shi}@ee.washington.edu

Abstract - Device matching and layout symmetry are of utmost importance to high performance analog and RF circuits. In this paper, we present HiLSD, the first CAD tool for the automatic detection of layout symmetry between two or more devices in a hierarchical manner. HiLSD first extracts the circuit structure from the layout, then applies an efficient pattern-matching algorithm to find all the subcircuits automatically, and finally detects layout symmetry on the portion of the layout that corresponds to extracted subcircuit instances. On a set of practical analog layouts, HiLSD is demonstrated to be much more efficient than direct symmetry detection on a flattened layout. Results from applying HiLSD to automatic analog layout retargeting for technology migration and new specifications are also described.

I Introduction

Variations in the process poly-silicon etch rate, dopant concentration and gradients in temperature, stress and oxide thickness affect the threshold voltage, mobility and current-factors in MOS transistors [1]. These effects on the device characteristics introduce mismatches in transistors that are designed to behave identically. Such mismatches drastically affect analog circuit performance leading to DC offsets, finite even-order distortion and lower common-mode rejection [2]. Symmetric layout of matched transistors alleviates the effects of mismatch in analog/RF circuits.

Device matching and symmetry along with floorplanning, placement and parasitic-driven wiring considerations pose considerable challenge to the automation of analog/RF layouts [2][3]. Over the years, macro-cell based automated placement and routing methodologies have been proposed for analog circuits [4][5]. These layout automation schemes, despite their effectiveness and generality, often fail to incorporate the expertise of the layout designer and are seldom accepted in the industry.

For technology migration and changes in performance specification of analog/RF circuits, a layout reuse methodology promises to be a viable alternative. Such methodologies for analog layout retargeting through layout-template creation by a procedural-language or graphical-user-interface have been proposed in [6][7]. Unfortunately, creation of such templates demands substantial effort from the user. In contrast, [8] recently proposed an automatic layout retargeting methodology for analog circuits, in which an already fine-tuned layout is used to automatically create a symbolic structural template incorporating floorplan, symmetry and device/wiring alignment information. The new device sizes under retargeting are imposed on the template and the output layout is generated by layout compaction with symmetry constraints [9].

In [8], the axes of symmetry obtained from the existing layout are used as constraints in the structural template. As will be elaborated later, the complexity of such layout retargeting methods is strongly dependent on the number of symmetry axes and corresponding constraints. Therefore, the efficient detection of layout symmetry represents an essential step for the analog layout retargeting process.

An algorithm was proposed in [10] for the detection of layout symmetry. Under this scheme, symmetry detection is accomplished by scanning the entire layout for all horizontally or vertically aligned equi-sized transistors. Unfortunately, this leads to the detection of all unintended axes of symmetry that reside in the layout. Such redundant axes over-constrain the structural template thereby rendering the layout retargeting process computationally expensive.

In this paper, we present a CAD tool, HiLSD (Hierarchical Layout Symmetry Detector), which automatically detects layout symmetry based on circuit hierarchy. First, the layout is extracted for the circuit netlist. Then, the circuit hierarchy is established from this flat netlist based on a library of subcircuits that contain device matching information. The detection of the axes of symmetry in the layout is then initiated from the hierarchical netlist. By triggering symmetry detection from the circuit-specific information, HiLSD significantly curtails the search-space and ignores all unintended axes of symmetry that reside in the layout. HiLSD generates a very concise set of symmetry constraints for the automatic layout retargeting process.

Furthermore, in a typical design company, layout and circuit designs are seldom accomplished by the same personnel. For the conscientious circuit designer, HiLSD provides an interactive mode of layout symmetry verification from its graphical user interface.

This paper is organized as follows. Section II discusses the background and the motivation for this work. Section III illustrates the methodology employed for symmetry detection in HiLSD. Section IV explains the process of netlist and hierarchy extraction. Section V describes the actual detection of symmetry from the layout. Section VI presents the experimental results of HiLSD and its application in analog layout automation. Section VII concludes the paper.

II. Background and Motivation

A. Background

A MOS transistor in a layout is defined as an overlap between two rectangles in the poly-silicon and diffusion mask layers and has three terminals, viz., the gate terminal in the poly-silicon layer and the source and drain terminals in the diffusion layer. Good matching between any pair of transistors is established by laying out the transistors symmetrically. Two transistors are deemed to be symmetric if their layouts are geometric mirror images of each other. As illustrated in Fig. 1, this implies equi-sized channel, drain and source regions, identical orientation and close proximity of the two transistors. For large or multi-fingered transistors, simple geometric mirroring may not establish acceptable matching due to the thermal and process gradients. Such transistor-pairs are often laid-out cross-coupled in one dimension, Fig. 2, or in the two-dimensional cross-coupled form of Fig. 3 also known as the common-centroid layout.
The layout symmetry detection algorithm presented in [10], henceforth called Direct Layout Symmetry Detection (DLSD), relies on scanning the entire layout for symmetric transistors. The nets and transistors in the layout are identified and all transistors are stored in a queue sorted by their bottom-edges. Devices connected by a net and with same ordinate of bottom-edges are then pairwise compared for the existence of geometric mirror images. After detection of all symmetric transistor-pairs, all axes of symmetry with same abscessa or ordinate are merged into a single axis. Under this scheme, the layout of Fig. 2 has eleven axes of symmetry marked by the axes \( s_1 \) to \( s_{11} \) and sixty-six (selecting 2 from 12) matched transistor pairs. The layout in Fig. 3 has six axes of symmetry as indicated by the axes \( s_1 \) to \( s_6 \) and thirty matched transistor pairs.

### B. Motivation: Analog Layout Retargeting

The automatic layout retargeting methodology [8] provides an efficient way of reusing existing fine-tuned analog layouts over changes in technology and design specifications. The re-targeting tool reads in a hand-crafted analog layout, the source and target technology-dependent design rules and automatically creates a symbolic structural template. By imposing the new device sizes pertaining to new specifications on the template, the tool generates a target layout that maintains all the designer expertise embedded in the source layout. The internal flow diagram of the retargeting tool is shown in Fig. 4.

The re-targeting tool-suite consists of a template extractor and a layout generator. The symbolic template, extracted from the source layout by the template extractor, comprises the design-rules, connectivity and symmetry constraints. The following equations represent the symmetry constraints generated for the layout of Fig. 1.

\[
\begin{align*}
(c_{\text{top}} - f_{\text{top}}) &= (c_{\text{bottom}} - f_{\text{bottom}}) = 0 \quad (1) \\
(s_g - s_{\text{right}}) &= (h_g - s_g) = 0 \quad (2) \\
(s_{\text{right}} - s_{\text{bottom}}) &= (h_{\text{right}} - h_g) = 0 \quad (3)
\end{align*}
\]

Here, \( s_g \) represents the symmetry axis and all other variables represent the edges of the rectangles. Eq. (1) enforces the alignment at the same ordinate and the equality of the widths of the transistors. The equidistance of the transistors from the symmetry axis is imposed by Eq. (2). The equality of the gate-lengths is enforced by Eq. (3).
constraint-graph, the symbolic compaction problem is solved using the shortest-path algorithm.

Thus, each symmetry axis introduces numerous variables and necessitates multiple transformations of the constraint-graph into the core-graph [9]. A large number of symmetry axes render the problem computationally intensive. Also, as we found during our retargeting experiments, too many redundant symmetry constraints may even render the problem unsolvable. Clearly, reducing the number of symmetry axes and avoiding all redundant constraints is essential for efficient layout retargeting.

### III. Hierarchical Symmetry Detection Flow

As discussed in Section II, reduction of symmetry constraints and avoidance of unintentional symmetry is a prime requirement for successful and efficient layout retargeting. The method proposed in this work is based on layout proximity based clustering of netlist and extraction of hierarchy information from the circuit. This is illustrated in Fig. 5.

![Hierarchical Symmetry Detection Methodology](Image)

**Fig. 5: Hierarchical Symmetry Detection Methodology.** The oval blocks are modules of HiLSD.

First, the Netlist Extractor generates the circuit netlist from the layout information. The netlist is then clustered into groups based on physical proximity in the layout. A designer-provided library consists of the netlists of the building blocks, and matching and symmetry information of individual devices. The subcircuits in the library can be any commonly used analog circuit like differential pair, current mirror or larger hierarchical blocks like comparators, operational amplifiers, etc. For simple building blocks such as multi-fingered transistor has two contiguous sets of three unit transistors. In the layout, each multi-fingered transistor $M$ consists of the gate, source, and drain terminals. The netlist database stores the poly-silicon and the diffusion layers. The netlist database stores the location, size, orientation and terminal information for each unit transistor. Once the transistors are extracted, a simple recursive algorithm detects the nets from the layout using the terminals of the transistors as the starting points.

#### B. Proximity Based Netlist Clustering

The netlist clustering process is especially important as it reduces the number of symmetry axes for multi-fingered transistors. In the layout, each multi-fingered transistor $M$ contains multiple contiguous elements $C$, where each contiguous element consists of physically contiguous unit transistors $T$. The clustering scheme partitions the netlist based on the manner in which the transistors are laid out.

The netlist, which at the end of extraction comprised of the set of unit transistors $T^a$ and the set of nets $N^a$, now consists of the same set of nets $N^a$ and the set of multi-fingered transistors $M^a$ defined as $\{M \mid |M| \geq 3 \forall \exists \{G_s, S_d, D_m\} \in N^a \}$ where $\{G_s, S_d, D_m\}$ is the set of the gate, source and drain nets of the multi-fingered transistor $M$. Each multi-fingered transistor $M$ is a set of physically contiguous elements $C^S$ i.e., $C \in M$ or in other words, $M = C^S = \{C\}$. And each contiguous element is defined as $C = \{T \mid T \in T^a, \forall T \{G_s, S_d, D_m\} = \{G_s, S_d, D_m\}, \land \forall T \in C$ are physically contiguous $\}$. For the one-dimensional cross-coupled symmetric pair of Fig. 2, each multi-fingered transistor has three contiguous sets of two unit transistors each. In Fig. 3, each multi-fingered transistor has two contiguous sets of three unit transistors each.

### IV. Netlist, Cluster and Hierarchy Extraction

#### A. Netlist Extraction

A transistor with a single rectangle each for its gate, source, and drain terminals is henceforth called a unit transistor. A net is defined as an electrical connection between the terminals of transistors or external ports.

The layout representation and netlist extraction schemes are adopted from the Magic VLSI layout system [11]. Unit transistors are detected by an efficient search for overlaps between the poly-silicon and the diffusion layers. The netlist database stores the location, size, orientation and terminal information for each unit transistor. Once the transistors are extracted, a simple recursive algorithm detects the nets from the layout using the terminals of the transistors as the starting points.

#### B. Proximity Based Netlist Clustering

The netlist clustering process is especially important as it reduces the number of symmetry axes for multi-fingered transistors. In the layout, each multi-fingered transistor $M$ contains multiple contiguous elements $C$, where each contiguous element consists of physically contiguous unit transistors $T$. The clustering scheme partitions the netlist based on the manner in which the transistors are laid out.

The netlist, which at the end of extraction comprised of the set of unit transistors $T^a$ and the set of nets $N^a$, now consists of the same set of nets $N^a$ and the set of multi-fingered transistors $M^a$ defined as $\{M \mid |M| \geq 3 \forall \exists \{G_s, S_d, D_m\} \in N^a \}$ where $\{G_s, S_d, D_m\}$ is the set of the gate, source and drain nets of the multi-fingered transistor $M$. Each multi-fingered transistor $M$ is a set of physically contiguous elements $C^S$ i.e., $C \in M$ or in other words, $M = C^S = \{C\}$. And each contiguous element is defined as $C = \{T \mid T \in T^a, \forall T \{G_s, S_d, D_m\} = \{G_s, S_d, D_m\}, \land \forall T \in C$ are physically contiguous $\}$. For the one-dimensional cross-coupled symmetric pair of Fig. 2, each multi-fingered transistor has three contiguous sets of two unit transistors each. In Fig. 3, each multi-fingered transistor has two contiguous sets of three unit transistors each.

### Table 1: Outline of the HiLSD algorithm.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>detectNetsTransistors</td>
<td>Extracts the netlist from the layout</td>
</tr>
<tr>
<td>clusterTransistors</td>
<td>Groups transistors that are physically contiguous</td>
</tr>
</tbody>
</table>

**Table 2: Algorithm for netlist partitioning.**

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkCreateMFT</td>
<td>Creates a multi-fingered transistor</td>
</tr>
<tr>
<td>checkCreateContiguous</td>
<td>Checks if two multi-fingered transistors are contiguous</td>
</tr>
<tr>
<td>insertSorted</td>
<td>Inserts a netlist into a sorted list</td>
</tr>
</tbody>
</table>

The `clusterTransistors` procedure in Table 2 presents the algorithm for partitioning the netlist. Each multi-fingered transistor is stored in a hashtable with hash key formed by the drain, gate and source terminals.
source nodes. For each unit transistor $T$ connected to a net $N$, a new multi-fingered transistor $M$ is created if it does not already exist in the hashtable. This is accomplished by a call to the routine `checkCreateMFT`. The routine `checkCreateContiguous` then checks if the unit transistor $T$ is aligned with one of the contiguous elements in $M$. If $T$ is not physically contiguous with any $C \in M$, a new contiguous element is created. In either case, the routine `insertSorted` inserts $T$ into a list of unit transistors of the corresponding contiguous element. This list of transistors in a contiguous element is doubly sorted with respect to the $x$ and $y$ coordinates.

C. Hierarchy Extraction

The designer-intended transistor-matching information is embedded in the subcircuits in the library. Identifying instances of these commonly used subcircuits in the main netlist maps the non-redundant matching information to the devices in the layout. This is accomplished by an efficient subgraph isomorphism algorithm [12] in the `mapSubcircuits` routine of Table 1.

First, both the subcircuit and the main circuit are implicitly partitioned by an iterative labeling algorithm to reduce the search space. This identifies a set of nodes in the main circuit and a single node, called a key node, in the subcircuit. The set of nodes in the main circuit obtained by this iterative labeling algorithm are potential start-points for checking a pattern match with the subcircuit. From each potential node in the main circuit and the key node in the subcircuit, another labeling algorithm accomplishes detection of an isomorphism with the subcircuits graph.

V. Layout Symmetry Detection

The hierarchy extraction process generates a subcircuit-based netlist. From the subcircuit-based netlist, a list of designer-intended non-redundant matched multi-fingered transistor pairs is created. The layout symmetry detection scheme identifies if each pair of these multi-fingered transistors is actually laid out symmetrically. The process also generates the corresponding constraints for the ensuing compaction step in layout automation [8].

The algorithm for layout symmetry detection is shown in Table 3. For each transistor pair intended to be matched, the `detectTopology` routine identifies the pair’s layout topology by traversing through the list of contiguous elements. Based on the topology, the unit transistors are inserted into two or four sorted lists. Thus, for the common-centroid topology of Fig. 3, the six unit transistors in the top and bottom halves of the transistors $M_1$ and $M_2$ respectively are collected into a list $L_x$. The bottom and top halves of $M_1$ and $M_2$ are collected into another list $L_y$. The unit transistors in $L_x$ and $L_y$ are then pairwise compared in the `checkSymmetry` routine to detect the vertical axis of symmetry, $s_x$, and generate the corresponding constraints. For the horizontal symmetry axis $s_y$, the bottom halves of both $M_1$ and $M_2$ are collected into a list $L_y$, and the top halves are collected into a list $L_x$ and pairwise compared. For the layout of Fig. 2, six unit transistors are inserted into each list $L_x$ and $L_y$ and a single axis of symmetry $s_y$ is detected. Prior co-ordinate based double sorting of the unit transistors in each multi-fingered transistor ensures that pairwise comparison can detect axes of symmetry.

VI. Results

A. Symmetry Detection Experiments

The HiLSD program was employed to detect symmetry in various analog/RF layouts and generate constraints for the layout retargeting methodology [8] illustrated in Fig. 4. Table 4 compares the symmetry detection data for HiLSD with the DLSD method presented in [10]. Various symmetry topologies were employed on the different layouts. The differential amplifier, the latched comparator and the 4:1 comparator used symmetric transistors with minimal multi-fingered structures. The voltage-controlled oscillator was laid out with extensive multi-fingered symmetric transistors. The two-stage and folded-cascade operational amplifiers utilized multi-fingered interleaved and common-centroid symmetry topologies. And the 5-bit flash analog-to-digital converter consisted of 31 instances of a latched-comparator laid out in an array of 8x4.

For each method, the number of symmetry axes detected, the number of symmetric transistor pairs, and the number of constraints due-to-symmetry are reported. The DLSD method extracted a large number of redundant symmetry axes. As it detected symmetry between every pair of unit transistors in each multi-fingered transistor, a large number of axes were observed for the two-stage operational amplifier and the voltage-controlled oscillator circuits. For the array structure of the comparator blocks in the 5-bit analog-to-digital converter, the DLSD method detected symmetry for every transistor in one comparator cell to every transistor in another comparator cell in the same row and column. These redundant constraints not only slowed down the compaction steps in layout retargeting, but also rendered the problem unsolvable in some cases.

Table 3: Algorithm for symmetry detection.

```
detectLayoutSymmetry
begin
  // ListSym = \{ (M_i, M_j) | M_i and M_j are intended matched pair \}
  for each (M_i, M_j) \in ListSym
  topology = detectTopology (M_i, M_j)
  if (topology == common_centroid) then
    L_x = insertToList (M_i, M_j, left)
    L_y = insertToList (M_i, M_j, right)
    L_y = insertToList (M_i, M_j, bottom)
    L_x = insertToList (M_i, M_j, top)
    checkSymmetry (L_x, L_y)
    checkSymmetry (L_y, L_x)
  else if (topology == horizontal_interleaving) then
    L_x = insertToList (M_i, M_j, left)
    L_y = insertToList (M_i, M_j, right)
    checkSymmetry (L_x, L_y)
  else if (topology == vertical_interleaving) then
    L_y = insertToList (M_i, M_j, bottom)
    L_x = insertToList (M_i, M_j, top)
    checkSymmetry (L_x, L_y)
  else // simple transistor layout
    checkSymmetry (M_i, M_j)
end if
end
```

We compare the scaling of the symmetry detection by the two methods with arrays of comparators. Fig. 6 shows the number of symmetric transistor pairs detected by DLSD and HiLSD as the number of comparators is scaled. The y-axis is in logarithmic scale. The graph shows that DLSD detects a huge number of redundant symmetry axes.
Table 4: Comparison between Hierarchical Symmetry Detection (HiLSD) and Direct Symmetry Detection (DLSD)

<table>
<thead>
<tr>
<th>Circuits</th>
<th># Multi-Fingered Transistors</th>
<th># Unit Transistors</th>
<th>Design Rule Constraints</th>
<th>Symmetry Axes</th>
<th>Symmetry Transistor Pairs</th>
<th>Symmetry Axes</th>
<th>Symmetry Transistor Pairs</th>
<th>Symmetry Axes</th>
<th>Symmetry Transistor Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Differential Amplifier</td>
<td>5</td>
<td>5</td>
<td>1,602</td>
<td>1</td>
<td>2</td>
<td>18</td>
<td>1</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>Latched Comparator</td>
<td>15</td>
<td>20</td>
<td>8,639</td>
<td>10</td>
<td>19</td>
<td>132</td>
<td>6</td>
<td>20</td>
<td>168</td>
</tr>
<tr>
<td>2-stage Opamp</td>
<td>9</td>
<td>48</td>
<td>5,902</td>
<td>69</td>
<td>262</td>
<td>1,578</td>
<td>2</td>
<td>12</td>
<td>78</td>
</tr>
<tr>
<td>Folded Cascode Opamp</td>
<td>14</td>
<td>43</td>
<td>8,352</td>
<td>29</td>
<td>173</td>
<td>1,206</td>
<td>6</td>
<td>20</td>
<td>168</td>
</tr>
<tr>
<td>4:1 Comparator</td>
<td>20</td>
<td>32</td>
<td>26,182</td>
<td>26</td>
<td>166</td>
<td>1,026</td>
<td>3</td>
<td>13</td>
<td>78</td>
</tr>
<tr>
<td>VCO</td>
<td>16</td>
<td>198</td>
<td>645,986</td>
<td>680</td>
<td>5,525</td>
<td>33,156</td>
<td>4</td>
<td>362</td>
<td>2,178</td>
</tr>
<tr>
<td>5-bit Flash ADC</td>
<td>435</td>
<td>590</td>
<td>320,937</td>
<td>261</td>
<td>6,218</td>
<td>4,193</td>
<td>12</td>
<td>186</td>
<td>1,302</td>
</tr>
</tbody>
</table>

Table 4: Comparison between Hierarchical Symmetry Detection (HiLSD) and Direct Symmetry Detection (DLSD)

B. Automatic Analog Layout Retargeting with HiLSD

We performed experiments on analog layout retargeting [8] to a new technology and specifications based on the hierarchical and direct symmetry detection methods. Fig. 7 shows a comparator layout in TSMC 0.25um CMOS process. This layout was retargeted under new specifications to the TSMC 0.18um CMOS technology using both DLSD and HiLSD based symmetry detection.

The symmetry constraints generated by the two methods were passed onto the resizing tool. Table 5 shows the number of symmetry axes, transistor pairs, symmetry constraints, and user runtime for the resizing schemes under the two methods. The retargeted layout obtained by using HiLSD for symmetry detection is shown in Fig. 8. The retargeted layout under this preserved all the required matching considerations, while incorporating a lesser number of symmetry constraints. The circuit performance of the latched-comparator in the two technologies achieved by these methods is reported in Table 6.

Fig. 7: Comparator Layout in TSMC 0.25um technology.

Fig. 8: Retargeted Layout of comparator in TSMC 0.18um technology.

Fig. 6: Comparison of HiLSD and DLSD symmetry detection for array of comparators. X-axis represents number of comparators in the array. Y-axis denotes number of symmetric transistor pairs (log scale).

The analog comparator section of the 5-bit flash analog-to-digital converter was constructed by placing 31 units of the latched-comparator into an 8x4 array. Each unit comparator was aligned and matched with other units in the same row and column. For any unit comparator in the section, another comparator corresponding to its preceding or following bit was positioned next to each other to minimize the mismatch. The layout of the comparator section of the ADC in TSMC 0.25um CMOS technology is shown in Fig. 9.

Fig. 9: Comparator block of a 5-bit flash ADC in TSMC 0.25um.

The flash analog-to-digital converter was retargeted to the TSMC 0.18um CMOS process; first with the symmetry information obtained from DLSD method and then with the HiLSD algorithm. DLSD detected 6,218 symmetric transistor pairs in the layout, while the HiLSD method identified only 186 symmetric pairs. This huge difference in detected symmetric pairs is due to the redundant symmetric pairs from the transistors on different unit comparators.
listed in same row or column. During resizing, these unnecessary symmetric-pairs resulted in the increase of symmetry constraints from 1,302 to 41,943, which subsequently increased the runtime of solving the symmetry constraints from 48 minutes to 156 minutes. The overall runtime escalated from about 2 hours to 4 hours.

Nevertheless, both target layouts showed similar symmetries and matching. The original layout had an area of 12,780 \textmu m\(^2\). The target layout had an area of 12,780 \textmu m\(^2\). The reduction in area is attributed to the avoidance of unwanted axes of symmetry that constrain the layout. The retargeted layout obtained through HiLSD method is shown in Fig. 10.

The authors wish to thank Mr. Sankaran Aniruddhan, System-On-Chip Lab, University of Washington, for help with the design examples.

Acknowledgements

The authors wish to thank Mr. Sankaran Aniruddhan, System-On-Chip Lab, University of Washington, for help with the design examples.

### References


#### VII. Conclusions

A new symmetry detection tool, Hi-LSD, based on hierarchical extraction and subcircuit specific symmetric transistor pairs is presented. The tool significantly reduces search-space and ignores all unintended symmetry axes exhibited on the layout. Employing Hi-LSD on a 5-bit flash analog-to-digital converter ignores all unintended axes of symmetry and reduces the number of symmetric pairs from 6,218 to 186. When applied with the automatic analog layout-retargeting tool, the runtime for regenerating the new ADC layout is reduced from 4 hours to 2 hours.

With the symmetry constraints described in a hierarchical circuit netlist by circuit designers, Hi-LSD also provides the first automatic tool for verifying if a layout meets all the symmetry constraints required by circuit designers.

### Table 5: Comparison between layout retargeting with DLSD and HiLSD symmetry detection schemes.

<table>
<thead>
<tr>
<th>Design</th>
<th>Latched Comparator</th>
<th>5-bit Flash Analog-to-Digital Converter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symmetry Detection Method</td>
<td>DLSD</td>
<td>HiLSD</td>
</tr>
<tr>
<td>Multi-fingered Transistors</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Unit Transistors</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Design Rule Constraints</td>
<td>8,639</td>
<td>8,639</td>
</tr>
<tr>
<td>Symmetry Axes</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>Transistor Pairs</td>
<td>19</td>
<td>6</td>
</tr>
<tr>
<td>Additional Symmetry Constraints</td>
<td>132</td>
<td>42</td>
</tr>
<tr>
<td>Runtime on Solving Symmetry Constraints</td>
<td>0.65 s</td>
<td>0.26 s</td>
</tr>
<tr>
<td>Total Runtime for Retargeting Tool</td>
<td>10.06 s</td>
<td>9.31 s</td>
</tr>
<tr>
<td></td>
<td>4 hr 20 min</td>
<td>4 hr 20 min</td>
</tr>
</tbody>
</table>

#### Table 6: Comparison of a latched-comparator performance

<table>
<thead>
<tr>
<th>Design Specs.</th>
<th>Layout in 0.25um</th>
<th>Retargeted Layout in 0.18um</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Supply</td>
<td>2.5 V</td>
<td>1.8 V</td>
</tr>
<tr>
<td>Ref. Voltage</td>
<td>1.28 V</td>
<td>1.28 V</td>
</tr>
<tr>
<td>Frequency</td>
<td>500 MHz</td>
<td>750 MHz</td>
</tr>
<tr>
<td>Resolution</td>
<td>20 mV</td>
<td>20 mV</td>
</tr>
<tr>
<td>Area</td>
<td>369 um²</td>
<td>225 um²</td>
</tr>
<tr>
<td>Power</td>
<td>0.84 mW</td>
<td>0.45 mW</td>
</tr>
</tbody>
</table>