DiffPattern: Layout Pattern Generation via Discrete Diffusion

Zixiao Wang\textsuperscript{1}, Yunheng Shen\textsuperscript{2}, Wenqian Zhao\textsuperscript{1}, Yang Bai\textsuperscript{1}, Guojin Chen\textsuperscript{1}, Farzan Farnia\textsuperscript{1}, Bei Yu\textsuperscript{1}

\textsuperscript{1}Chinese University of Hong Kong
\textsuperscript{2}Tsinghua University
Background Knowledge
VLSI layout patterns provide critical resources in various designs for manufacturability research, from early technology node development to back-end design and sign-off flows [DAC'19].

---

An End-to-End Learning Solution?

- Maybe No
- Gap between Discrete Rules and Continuous DNN Model

The three basic DRC checks:

- Width
- Spacing
- Enclosure
Squish Pattern Representation

Lossless and efficient representation method

Encodes layout into pattern topology matrix and geometric information

Problem #1: information density of each pixel is still not satisfactory

\[
\begin{align*}
\text{Topology:} & \quad \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \\
\text{Geometry:} & \quad \Delta_x = [\delta_{x1}, \delta_{x2}, \delta_{x3}, \delta_{x4}] \\
& \quad \Delta_y = [\delta_{y1}, \delta_{y2}, \delta_{y3}, \delta_{y4}] 
\end{align*}
\]
Novel Pattern Generation

- Generate gray image (topology) and transfer it into a binary image
- May lead to a deduction of information
- **Problem #2**: How to generate a binary mask directly?

---

Pattern Legalization

Examples of DRC Rule

Finding legal distance vector for each topology

- Solving a Linear System (1D pattern) [DAC’19].
- Using Exist Distance Vector (2D pattern) [ICCAD’20]
- Problem #3: 2D pattern introduces non-linear constraint, hard to solve!
Pattern Diversity. Shannon entropy of the pattern complexity.

\[ H = - \sum_i \sum_j P(c_{xi}, c_{yj}) \log P(c_{xi}, c_{yj}), \]  

(1)

Pattern Legality.

\[ L = \frac{\# \text{ Legal Patterns}}{\# \text{ All Patterns}}. \]  

(2)
Illustration of denoising diffusion process.

Forward Process: \( q(T_k | T_{k-1}) := \mathcal{N}(T_k; \sqrt{1 - \beta_k} T_{k-1}, \beta_k I) \).
Reverse Process: \( p_\theta(T_{k-1} | T_k) := \mathcal{N}(T_{k-1}; \mu_\theta(T_k, k), \Sigma_\theta(T_k, k)) \).
Proposed Method: DiffPattern
An illustration of the Diffpattern framework for reliable layout pattern generation.
The Topology Tensor is a lossless and compact representation of the topology matrix.

The Naive Concatenating brings unbalanced power to each bit and an exponentially increasing state space.
Problem #2: Topology Tensor Generation

Forward Process \( q (x_k \mid x_{k-1}) := \text{Cat}(x_k; p = x_{k-1}Q_k) \),
Multiple step forward at once. \( q(x_k \mid x_0) = \text{Cat}(x_k; p = x_0\bar{Q}_k), \quad \bar{Q}_k = Q_1 Q_2 \cdots Q_k \)
Reverse Process \( p_\theta (x_{k-1} \mid x_k) = \sum_{\tilde{x}_0} q(x_{k-1} \mid x_k, \tilde{x}_0) p_\theta (\tilde{x}_0 \mid x_k) \).
Training Loss Function: \( L = D_{KL}(q(x_{k-1} \mid x_k, x_0) \parallel p_\theta (x_{k-1} \mid x_k)) - \lambda \log p_\theta (x_0 \mid x_k) \),
A uniform stationary distribution is a natural choice in topology tensor generation. Given any $x_0$, the distribution of every entry $x_k$ should follow,

$$q(x_k|x_0) \rightarrow [0.5, 0.5], \text{ when } k \rightarrow K. \quad (3)$$

$$Q_k = \begin{bmatrix}
1 - \beta_k & \beta_k \\
\beta_k & 1 - \beta_k
\end{bmatrix}, \quad (4)$$

$$\beta_k = \frac{(k - 1)(\beta_K - \beta_1)}{K - 1} + \beta_1, \ k = 1, ..., K, \quad (5)$$

where $\beta_1$ and $\beta_K$ are hyperparameters.
Examples of DRC Rule

\[
\begin{align*}
\delta_{xi}, \delta_{yj} &> 0, & \forall \delta_{xi}, \delta_{yj}; \\
\sum \delta_{xi} &= \sqrt{CM}, \quad \sum \delta_{yj} = \sqrt{CM}; \\
\sum_{i=a}^{b} \delta_i &\geq \text{Space}_{\text{min}}, & \forall (a, b) \in \text{Set}_S; \\
\sum_{i=a}^{b} \delta_i &\geq \text{Width}_{\text{min}}, & \forall (a, b) \in \text{Set}_W; \\
\sum \delta_{xi}\delta_{yj} &\in [\text{Area}_{\text{min}}, \text{Area}_{\text{max}}], & \forall \text{Polygon}.
\end{align*}
\]
Experiment Results
## Diversity and Legality

<table>
<thead>
<tr>
<th>Set/Method</th>
<th>Generated Topology</th>
<th>Generated Patterns</th>
<th>Legal Patterns</th>
<th>Diversity (↑)</th>
<th>Legality (↑)</th>
<th>Diversity (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Patterns</td>
<td>-</td>
<td>-</td>
<td>13869</td>
<td>10.777</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAE [DAC’19]</td>
<td>100000</td>
<td>100000</td>
<td>19</td>
<td>3.7871</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VCAE [ICCAD’20]</td>
<td>100000</td>
<td>100000</td>
<td>2126</td>
<td>9.9775</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAE+LegalGAN [ICCAD’20]</td>
<td>100000</td>
<td>100000</td>
<td>3740</td>
<td>5.8142</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VCAE+LegalGAN [ICCAD’20]</td>
<td>100000</td>
<td>100000</td>
<td>84510</td>
<td>9.8669</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LayouTransformer [ICCAD’22]</td>
<td>-</td>
<td>100000</td>
<td>89726</td>
<td>10.527</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DiffPattern-S</td>
<td>100000</td>
<td>100000</td>
<td><strong>100000</strong></td>
<td><strong>10.815</strong></td>
<td></td>
<td><strong>10.815</strong></td>
</tr>
<tr>
<td>DiffPattern-L</td>
<td>100000</td>
<td>100000</td>
<td>10000000</td>
<td>10.815</td>
<td><strong>10000000</strong></td>
<td><strong>10.815</strong></td>
</tr>
</tbody>
</table>

- DiffPattern achieves a perfect performance (i.e. 100%) under the metric of legality.
- DiffPattern also gets reasonable improvement (10.527 → 10.815) on the diversity.
- We generate 100 different layout patterns from each topology in DiffPattern-L.
Flexibility: Generate Different Patterns from Single Topology.

Different layout patterns that are generated from a single topology with the same design rule.
Flexibility: Generate Legal Patterns with Different Design Rules.

Layout patterns that are generated from the same topology with different design rules: (a) Normal rule; (b) Larger $space_{\text{min}}$; (c) Smaller $Area_{\text{max}}$. 
Distribution of Complexity

An illustration of complexity distribution.
Model Efficiency

<table>
<thead>
<tr>
<th>Phase/Method</th>
<th>Cost Time (s)</th>
<th>Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling</td>
<td>0.544</td>
<td>N/A</td>
</tr>
<tr>
<td>Solving-R</td>
<td>0.269</td>
<td>1.00×</td>
</tr>
<tr>
<td>Solving-E</td>
<td>0.117</td>
<td>2.30×</td>
</tr>
</tbody>
</table>

• Initializing with existing results achieves 2.30× acceleration on CPU.
THANK YOU!