Painting on Placement: Forecasting Routing Congestion using Conditional GANs

Cunxi Yu and Zhiru Zhang
ECE
Cornell University
Motivation

- Physical design
  - Logic netlist to layout
    - Input: Directed graph
    - Output: physical netlist

LUT netlist - $G(V,E)$

- Packing
- Placement
  - Place elements on the previous img
- Routing
  - Route based on $G$ and design rules

Availability:
- Available CLB spot
- Available I/O pads
- Multiplier
- Adder chain
- CLB or I/O placed

Hours ~ Days
Routing Congestion

- Why congestion is important?
  - Routability
  - Timing closure

- How is congestion evaluated?
  - Utilization of routing channels

- What makes differences?
  - Placement

Very time-consuming to get post-routing congestion.
Related Works (Todo)

- **ML for EDA**
  - Front-end applications
    - Synthesis [Dai et al., FCCM’18] [Haaswijk ISCAS’18]
    - Design flows [Yu et al., DAC’18] [Ustun et al., FCCM’19]
  - Back-end applications
    - Place and route [Xie ICCAD’18] [Pui ICCAD’17][Huang et al., DATE19]
    - Manufacturability [Xu et al., TCAD’18] [Ye et al., DAC’19]
  - Analog IC design
    - Design and layout of analog designs [Wang et al., arxiv][Xu et al., DAC’19]

- **Limitations**
  - Hotspots/partial congestion estimation only [Xie ICCAD’18] [Pui ICCAD’17] [Yang et al., TCAD’18]
  - Require early routing information as features [Xu et al., TCAD’18] [Huang et al., DATE19]
Approach – Painting on Placement

- Forecast congestion from placement
  - Learning a image-to-image mapping \cite{Isola et al, CVPR'17}
    - Congestion and placement can be represented as image

- Forecast as \textit{image colorization}
  - Colorizing the routing channels of the placement image
    - Exact the same underlying structure
Features

- **Embedding of placement and G(V,E)**
  - Placement image (RGB)
  - Visualization of netlist after placement (grayscale)
  - Final input: stack two images as a four-channel matrix
    - e.g., 256-256-4

![Features Image]

- **Tips for constructing feature images**
  - **Color code** to distinguish different types of cells
  - **Adjust resolution** to distinguish each cell as a objective
Generative Adversarial Nets

- **Generative Adversarial Nets (GAN)** [Goodfellow NeurIPS14]
  - Discriminator: learns to **classify** true or fake
  - Generator: learns to **fool** discriminator
  - No control over modes of the data to be generated

- **Conditional GAN (cGAN)** [Mirza arxiv14]
  - Adding the additional parameter to control the generator
  - Model used in this work
    - G: Fully Convolutional Networks (FCN) with 7 down/up-sampling layers
    - D: CNN based binary classifier
Training and Inference

- **Training**
  - Discriminator
    - Learns to **classify** true or fake
  - Generator
    - Learns to **fool** the discriminator
  - Input-output pair
    - Stacked matrix
      - 256-256-4
    - Congestion heatmap
      - 256-256-3

- **Inference**
  - Run generator
Dataset and Results

- Image generator implemented based on VPR
  - VPR configs: seed, ALPHA_T, INNER_NUM, place_algorithm
  - Training: 1x1080Ti, < 3 hours
  - Inference: < 0.2 s on 1x1080Ti (batch=1)

- Min-congestion exploration
  - e.g., target design = \textit{dcsg}

<table>
<thead>
<tr>
<th>Design</th>
<th># LUTs</th>
<th># FFs</th>
<th># Nets</th>
<th># samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>diffeq1</td>
<td>563</td>
<td>193</td>
<td>2059</td>
<td>200</td>
</tr>
<tr>
<td>diffeq2</td>
<td>419</td>
<td>96</td>
<td>1560</td>
<td>200</td>
</tr>
<tr>
<td>SHA</td>
<td>2501</td>
<td>1047</td>
<td>5023</td>
<td>200</td>
</tr>
<tr>
<td>OpenRISC</td>
<td>2823</td>
<td>911</td>
<td>10910</td>
<td>200</td>
</tr>
<tr>
<td>ODE</td>
<td>5488</td>
<td>670</td>
<td>12336</td>
<td>150</td>
</tr>
<tr>
<td>bfly</td>
<td>9503</td>
<td>1748</td>
<td>38582</td>
<td>150</td>
</tr>
<tr>
<td>dcsg</td>
<td>9088</td>
<td>1618</td>
<td>36912</td>
<td>150</td>
</tr>
</tbody>
</table>
Results – Placement Exploration

- Constrained placement exploration
  - Placement with biased routing congestion region

![Placement](image1)
![Predicted](image2)
![Ground truth](image3)
Results – Real-time Forecast

- Visualization of *simulated annealing* placement
  - Swap locations iteratively
  - Every 200 iterations
Results – Real-time Forecast

- Visualization of *simulated annealing* placement
  - Swap locations iteratively
  - Every 200 iterations
Conclusion

- **Forecasting Routing Congestion using Conditional GANs**
  - Estimate routing utilization of all routing channels
  - Generate high-quality full routing congestion heatmap
    - Image-to-image mapping as image colorization

- **Limitations**
  - Model is device specific
  - Still require 5-10 image pairs for new design
  - Black-box inference is inefficient while is integrated with PnR tools

- **Future work**
  - Accelerate timing closure by packing/placement exploration
  - Integrate with Yosys-NextPnR flow and VTR(ABC-VPR) flow
Thank you!
Conditional Generative Adversarial Nets

▸ Why conditional GAN (cGAN) ?
  – Condition on the input image
  – Input and output has similar underlying structure

▸ Generator
  – Fully Convolutional Networks (FCN)

▸ Discriminator
  – CNN binary classifier

Encoder (down sampling)  Decoder (up sampling)
Results - Analysis of Skips and L1 Loss
Results - Analysis of Skips and L1 Loss