Developing Synthesis Flows without Human Knowledge

Cunxi Yu¹, Houping Xiao², Giovanni De Micheli¹
EPFL¹, SUNY Buffalo²
June 26, 2018
Overview of Design Flows

Design flow

- A sequence of design transformations

always @(posedge clk) begin
if ( r ) then p <= 0
else p <= p+1;
end if;
end
Introduction

- Quality of Designs
  - Design flows play the main role
- Time-to-Market
- Saving human
  - "No human"

[1] Survey by Arteris
Introduction

- Challenges in new design styles
  - e.g., Neuromorphic chip [2], Quantum chip [3]
    - Limited knowledge compared to ICs

[3] IBM 20-QBit
Example 1 – Permutation of Flows

- 128-bit AES
  - Same transformations
  - 50,000 random flows
    - Random permutation

- QoR collected after tech-map
  - Delay and Area
  - 14nm
Example 1 – Permutation of Flows

\texttt{resyn} * 4 (24)
\texttt{resyn2} * 4 (40)
\texttt{resyn3} * 4 (36)
Example 2 – Design Specific

- 128-bit AES vs. 64-bit ALU
  - Same 50,000 flows
  - Statistically significant

- QoR collected after tech-map
  - Delay and Area
  - 14nm
Objective

- Performance estimation

Flow_1, Flow_2, Flow_3, Flow_4, ..., Flow_n

Model

Perfect!
Very Good
Good
Not Good.
...
OMG!!
Definition and Search Space

- Decision make in design flows
  - Case 1: none-repetition flow
    - Each transformation appears **only once** (+ map)
    - Example: n=4, balance (b), rewrite (rw), refactor (rf), resub (rs)

Search space

\[ f(n) = n! \]

4! = 24
Definition and Search Space

- Decision make in design flows
  - Case 2: $m$-repetition flow
    - Each transformation appears $m$ times (+ map)
    - Example: $m=2$, $n=3$, balance (b), rewrite (rw), refactor (rf)

Search space

$$f(n) = \frac{(n \cdot m)!}{(m!)^n}$$

$$(2 \cdot 3)!/(2!)^3 = 90$$
Definition and Search Space

- Decision making in design flows
  - Different flow likely produces different QoR
    - Find the Best (Angel) and Worst (Devil)

\[ n=6 \quad m=4 \quad \approx 3.25e+15 \]
Approach

- MultiClass Classification
  - Modeling
    - Features
    - Ground truth
  - CNNs-Classifier
    - Why Deep Learning? Why CNNs?
    - Architecture
    - Training
    - Inference: finding best/worst flows by
Features – Flow in One-hot Matrix

- Flow in One-hot Matrix
  - Example: balance (b), rewrite (rw), rw -z (rwz)
    - b = [1 0 0]
    - rw = [0 1 0]
    - rwz = [0 0 1]
  - Flow: rw -> b -> rwz = [rw;b;rwz]
Ground Truth – Performance Class of Flows

- Labels of flows
  - Labeling rule 1: single metric (e.g. delay)
  - Labeling rule 2: multi-metric (e.g. delay+area)
Convolutional Neural Network

- Convolutional Neural Networks (CNNs)
  - Deep, feed-forward artificial neural networks
  - Multilayer perceptron
    - Conv, Pool, Fully connected layers

- Why CNNs Classifier?
  - “0” of hand writing
Why CNNs–classifier?
- “0” of handwritten digit
- “0” of Logic Synthesis (ABC)
Convolutional Neural Network

○ Why CNNs–classifier?
  ○ “digits” of flows
    • shape of best flow – “0”
      • Global or local shape
    • shape of worst flow – “6”

○ CNN classifier
  ○ 2 Conv + 3 Dense
    • Softmax dim = 6
  ○ Training
    • RSMProp + SELU

Code: https://github.com/yunxi/FlowGen-CNNs-DAC18
Datasets

- Training/testing inputs
  - Three designs
  - ABC “map” mapper + 14nm
  - ~300,000 data points
    - Failure cases eliminated
  - 20% for training, 80% for testing

<table>
<thead>
<tr>
<th>Design</th>
<th>64-bit Mont</th>
<th>64-bit ALU</th>
<th>128-bit AES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data points</td>
<td>99,997</td>
<td>100,000</td>
<td>99,737</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Features</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow</td>
<td>(1, 144) → (24, 6)</td>
</tr>
</tbody>
</table>
Results – Prediction Accuracy

- Confusion matrix
  - Design: 128-bit AES
Results – Generated Flows

- System overview
  - Training set generation
    - Online collection
  - Training the classifier
  - Inference on a large number of untested flows

[Diagram showing the flow of data processing with stages labeled 1, 2, and 3.]
Results – Generated Flows
Conclusion

- **Generic Flow Classification**
  - Deep Learning, CNNs
  - Demonstration with Logic Synthesis Tool
    - Design specific
    - Overall >70%, best/worst class > 86%
      - Three realistic designs

- **Future Work**
  - Cross-layer design flows (end-to-end)
  - Transfer-learning cross technology/designs
  - Explainability
  - Open database
# Open Source

- Code & Datasets

https://github.com/ycunxi/FLowGen-CNNs-DAC18

<table>
<thead>
<tr>
<th>Folder</th>
<th>Description</th>
<th>Last Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>.ipynb_checkpoints</td>
<td>adding pre-trained model; adding inference only demo</td>
<td>a month ago</td>
</tr>
<tr>
<td>datasets</td>
<td>Update README.md</td>
<td>a month ago</td>
</tr>
<tr>
<td>LICENSE.md</td>
<td>license</td>
<td>2 months ago</td>
</tr>
<tr>
<td>README.md</td>
<td>Update README.md</td>
<td>a month ago</td>
</tr>
<tr>
<td>dac18_GF_delay_model_save_test.h5</td>
<td>adding pre-trained model; adding inference only demo</td>
<td>a month ago</td>
</tr>
<tr>
<td>dac18_keras_inference_only.ipynb</td>
<td>adding pre-trained model; adding inference only demo</td>
<td>a month ago</td>
</tr>
<tr>
<td>dac18_keras_train_n_inference.ipynb</td>
<td>adding pre-trained model; adding inference only demo</td>
<td>a month ago</td>
</tr>
</tbody>
</table>

**FLowGen-CNNs-DAC18:**

Demo datasets generated with open source Logic Synthesis framework ABC from UC Berkeley [see https://github.com/berkeley-abc/abc]


License: MIT | Implementation: tensorflow | python 3.4
Thank you!