Introduction to Machine Learning for Computer-Aided Design (CAD)

EECS 151/251A
Josh Kang (advised by John Wawrzynek)
Mar 21 2023
Agenda

- Overview of Recent Trends in ML for CAD/EDA
  - Deeper Dive: ML-Driven Logic Synthesis Optimization
- Challenges in ML for CAD
- Research @ Berkeley on ML-CAD
1

Overview of Recent ML-CAD Research
ML for Various Stages of Digital IC Design

- Active research on applying ML (notably Deep Learning) to each stage of EDA
- Each stage can have multiple tasks to target:
  - e.g. in P&R:
    - Predict routing congestion
    - Predict routing result
  - Full or partial automation
  - Improve optimization of quality of results (QoR)
- But also, ML for AMS Design

Recent Trends: ML-CAD @ Industry

Google Research
The Potential of Machine Learning for Hardware Design

Can we learn to design chips in days or weeks?

Presenting the work of many people at Google, especially
David Bieber, Roger Carpenter, Anna Goldie, William Hang, Richard Ho, Safeen Huda, Joe Jiang, Wenjie Jiang, Eric Johnson, James Laudon, Quoc Le,
Young-Joon Lee, Azalia Mirhoseini, Azade Nazi, Jiwoo Pak, Omkar Pathak, Kartik Prabhu, C. Richard Ho, Hamid Shojaei, Rishabh Singh, Ebrahim Songhori, Kavya Srinivasa, Charles Sutton, Andy Tong, Emre Tuncer, Quoc V Le, Shobha Vasudevan, Shen Wang, Mustafa Yazgan, and Dan Zhang,

Jeff Dean, Google Research, @JeffDean and ai.google/research/people/jeff

Google @DAC 2022

Machine Learning and Algorithms: Let Us Team Up for EDA

Haoming Ren, Brucee Khallany
NVIDIA Corporation, Austin, TX 78717 USA

Matthew Fojtik
NVIDIA, Durham, NC 27713 USA

Yanqing Zhang
NVIDIA, Inc., Santa Clara, CA 95051 USA

Cadence
@DesignCon
2023
Recent Trends: ML-CAD @ Academia

Major CAD conferences (DAC, ICCAD, DATE, etc.) are featuring increasing numbers of research on ML-CAD

Specific workshop conference: ACM/IEEE Workshop on ML-CAD
Deeper Dive: ML for Logic Synthesis

- Early Stage in IC Design: unoptimized layout from synthesis will hurt end PPA

- “Re-inventing” Logic Synthesis tools with ML methods is impractical / too complex

- Instead, replace heuristic-based decisions made during logic synthesis with ML
  - Order sequence of Boolean optimizations:
    - Flowtune (Multi-Arm Bandit), BOiLS (GP Bayes-Opt), DeepRL-based works, etc.
  - Predict post-synthesis PPA: SNS, etc.
  - Choice of optimizer / logic representation (AIG/MIG): LSOracle, etc.
Deeper Dive: ML for Logic Synthesis

- Instead, **replace heuristic-based decisions** made during logic synthesis with ML
  - Order sequence of Boolean optimizations:
    - Flowtune (Multi-Arm Bandit), **BOiLS** (GP Bayes-Opt), DeepRL-based works, etc.
Deeper Dive: ML for Logic Synthesis

BOiLS: Bayesian Optimisation for Logic Synthesis

Antoine Grosnit*  
Huawei Noah's Ark Lab  
antsine.grosnit@huawei.com

Cedric Malherbe*  
Huawei Noah's Ark Lab  
cedric.malherbe@huawei.com

Jun Wang  
Huawei Noah's Ark Lab  
w.j@huawei.com

Rasul Tutunov  
Huawei Noah's Ark Lab  
rasul.tutunov@huawei.com

Haiham Bou Ammar  
Huawei Noah's Ark Lab  
haitham.bouammar@huawei.com

Xingchen Wan  
University of Oxford  
xingchen.wan@huawei.com

seq* ∈ arg min_{seq∈Alg^K} QoR_C(seq) ≡ arg max_{seq} -QoR_C(seq)  
This paper’s focus

QoR_C(seq) = \frac{\text{Area}_C(seq)}{\text{Area}_C(\text{ref})} + \frac{\text{Delay}_C(seq)}{\text{Delay}_C(\text{ref})}.

Fig. 1. Average QoR results over 10 EPFL circuits to recover BOiLS’ QoR values after only 200 trials.
Deeper Dive: ML for Logic Synthesis

- Early Stage in IC Design: unoptimized layout from synthesis will hurt end PPA

- “Re-inventing” Logic Synthesis tools with ML methods is impractical / too complex

- Instead, replace heuristic-based decisions made during logic synthesis with ML
  - Order sequence of Boolean optimizations:
    - Flowtune (Multi-Arm Bandit), BOiLS (GP Bayes-Opt), DeepRL-based works, etc.
  - Predict post-synthesis PPA: SNS, etc.
  - Choice of optimizer / logic representation (AIG/MIG): LSOracle, etc.
Deeper Dive: ML for Logic Synthesis

SNS’s not a Synthesizer: A Deep-Learning-Based Synthesis Predictor

Ceyu Xu
ceyu.xu@duke.edu
Duke University
Durham, North Carolina, USA

Chris Kjellqvist
christopher.kjellqvist@duke.edu
Duke University
Durham, North Carolina, USA

Lisa Wu Wills
lisa@cs.duke.edu
Duke University
Durham, North Carolina, USA

Figure 8: Boom DSE Result: The HighPerf design is plotted using a triangle shape, the PowerEff design is plotted using a diamond shape, and the AreaEff design is plotted using a square shape. The coremark scores are linearly normalized such that the fastest core in our design has a score of one.

Deeper Dive: ML for Logic Synthesis

- Early Stage in IC Design: unoptimized layout from synthesis will hurt end PPA

- “Re-inventing” Logic Synthesis tools with ML methods is impractical / too complex

- Instead, replace heuristic-based decisions made during logic synthesis with ML
  - Order sequence of Boolean optimizations:
    - Flowtune (Multi-Arm Bandit), BOiLS (GP Bayes-Opt), DeepRL-based works, etc.
  - Predict post-synthesis PPA: SNS, etc.
  - Choice of optimizer / logic representation (AIG/MIG): LSOracle, etc.
## Deeper Dive: ML for Logic Synthesis

**LSOracle: a Logic Synthesis Framework Driven by Artificial Intelligence**

*Invited Paper*

Walter Lau Neto  
LNIS, University of Utah, Salt Lake City, UT, USA

Max Austin  
LNIS, University of Utah, Salt Lake City, UT, USA

Scott Temple  
LNIS, University of Utah, Salt Lake City, UT, USA

Luca Amara  
Synopsys Inc., Sunnyvale, CA, USA

Xifan Tang  
LNIS, University of Utah, Salt Lake City, UT, USA

Pierre-Emmanuel Gaillardon  
LNIS, University of Utah, Salt Lake City, UT, USA

### Table: Technology-independent Results

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Original</th>
<th>ABC</th>
<th>CirKit</th>
<th>LSOracle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#nodes</td>
<td>depth</td>
<td>#nodes</td>
<td>depth</td>
</tr>
<tr>
<td>Pico-RV</td>
<td>17,019</td>
<td>36</td>
<td>16,483</td>
<td>36</td>
</tr>
<tr>
<td>oc_aquarius</td>
<td>16,653</td>
<td>206</td>
<td>16,321</td>
<td>38</td>
</tr>
<tr>
<td>s38417</td>
<td>12,394</td>
<td>36</td>
<td>8,352</td>
<td>27</td>
</tr>
<tr>
<td>chip_bridge</td>
<td>72,190</td>
<td>26</td>
<td>8,135</td>
<td>21</td>
</tr>
<tr>
<td>FPU</td>
<td>124,565</td>
<td>29</td>
<td>8,559</td>
<td>28</td>
</tr>
<tr>
<td>sum</td>
<td>243,844</td>
<td>522</td>
<td>64,876</td>
<td>22</td>
</tr>
<tr>
<td>ratio</td>
<td>1.00</td>
<td>1.00</td>
<td>0.73</td>
<td>0.73</td>
</tr>
</tbody>
</table>

### Table: ASIC Technology Mapped Results

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Original</th>
<th>ABC</th>
<th>CirKit</th>
<th>LSOracle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#nodes</td>
<td>depth</td>
<td>#nodes</td>
<td>depth</td>
</tr>
<tr>
<td>Pico-RV</td>
<td>17,019</td>
<td>36</td>
<td>16,483</td>
<td>36</td>
</tr>
<tr>
<td>oc_aquarius</td>
<td>16,653</td>
<td>206</td>
<td>16,321</td>
<td>38</td>
</tr>
<tr>
<td>s38417</td>
<td>12,394</td>
<td>36</td>
<td>8,352</td>
<td>27</td>
</tr>
<tr>
<td>chip_bridge</td>
<td>72,190</td>
<td>26</td>
<td>8,135</td>
<td>21</td>
</tr>
<tr>
<td>FPU</td>
<td>124,565</td>
<td>29</td>
<td>8,559</td>
<td>28</td>
</tr>
<tr>
<td>sum</td>
<td>243,844</td>
<td>522</td>
<td>64,876</td>
<td>22</td>
</tr>
<tr>
<td>ratio</td>
<td>1.00</td>
<td>1.00</td>
<td>0.73</td>
<td>0.73</td>
</tr>
</tbody>
</table>

---

2

Challenges in ML-CAD
When is ML/DL Successful?

- **Abundance of Data** and **Availability of Compute**
- Suppose compute is not a problem (enough GPUs for the model/dataset size)
- Example Datasets in Computer Vision: Image Classification
  - CIFAR-10: 6000 images per class, total 60K images
  - ImageNet: ~1000 images per synonym set, 100K+ synonym sets
- Example Datasets in NLP: Language Modeling
  - WikiText-103: 100M tokens from > 28K selected Wikipedia articles
Challenges in ML-CAD

● Lack of Publicly Available Digital Circuit Datasets
  ● Relative shortage of Open-Source RTL
    ○ < 50 substantial & full-developed open-source RTL projects/suites
    ○ “Standardization” of data format a huge burden
      ■ e.g. if use Graph Neural Networks (GNNs), which netlist representation (AST, Gate-level, etc.) to use and how to extract graphs from different HDLs?
  ● Most work employ **Supervised Learning**:
    labeling cost often incurs 1+ run of commercial/gold-standard EDA flow
    ○ For realistic workloads, could take order of minutes/hours/days per datapoint
  ● Similar for Deep RL: true reward evaluation incurs significant overhead
  ● **Data and label efficiency is critical to success**
Challenges in ML-CAD

- Implications of the Lack of Established, Large (Supervised) Datasets
  - (1) Model cannot scale and difficult to generalize across multiple circuit designs
    - Greatly limits practical usage of ML in EDA workflows
  - (2) Many works are not evaluated against realistic designs and/or sufficient numbers of such designs
  - (3) Substantial data collection (per-task) + ML training cost: could have used run traditional toolflows longer for the same amount of time

- Ideas to Overcome these Limitations
  - Data Augmentation and Transfer Learning
    - Train with easier-to-obtain labels; transfer to actual task
    - Domain adaptation to different circuit designs
      (Mirhoseini et al. “Chip Placement with Deep Reinforcement Learning” 2020, etc.)
Challenges in ML-CAD

- Ideas to Overcome this Limitations
  - Data Augmentation Example:
    - SNS’s not a Synthesizer (Xu et al. ISCA 2022)
    - Sample FF-to-FF paths from RTL graph
    - Model trained to predict PPA of individual paths;
      Generalize to predicting full circuit PPA

Figure 2: Transformation from Source Circuit to GraphIR Representation with Path Information and Graph Statistics
Research on ML-CAD @ Berkeley
Berkeley Research on ML-CAD

- **ML for Analog-Mixed Signal (AMS) Circuit Design:**
  Previous students at Stojanovic and Nikolic Groups

BagNet: Berkeley Analog Generator with Layout Optimizer Boosted with Deep Neural Networks

AutoCkt: Deep Reinforcement Learning of Analog Circuit Designs

Kourosh Hakhamaneshi, Nick Werblun, Pieter Abbeel, Vladimir Stojanović
Email: kourosh_hakhamaneshi, nwerblun, vlada@berkeley.edu, pabbeel@cs.berkeley.edu

University of California Berkeley, USA

Keertana Settaluri, Ameer Haj-Ali, Qiijng Huang, Kourosh Hakhamaneshi, Borivoje Nikolic
University of California, Berkeley
{ksettaluri6,ameerh,qiijng.huang,kourosh_hakhamaneshi,bora}@berkeley.edu

- **Machine Learning for Power:** Nikolic Group
- **Novel Self-Supervised Learning for Circuit Representation Learning:** Wawrzynek Group
ML-assisted Power Modeling

- Nayiri K. (advised by Prof. Bora Nikolic)

- Lack of standard benchmarks for both RTL + workloads
  - SOTA reports good errors, but on very different (& usually simple) test data
  - lack of *diversity* in benchmarks
  - need benchmarks that are minimal to avoid wasting compute resources

- Need a power model…
  - that is *auto-generated* (no human input) and *transferable* to…
    - new workloads
    - new RTL? significantly more challenging, not very useful beyond Arch-level
  - with good time & space granularity

- Power models must have good fidelity across design stage abstractions
  (Arch → RTL → Syn → PAR)
ML-assisted Power Modeling

- **Goal:** input stimuli-dependent dynamic power prediction
  - Input features: arch counter *deltas* every N cycles
  - Output: predicted power for a window of N cycles
- **Golden power**
  - .fsdb’s through Joules with Intel16
- **Quadratic regression model + ElasticNet(CV)**

<table>
<thead>
<tr>
<th>workload</th>
<th>train</th>
<th>test</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>dhrystone</td>
<td>0.93</td>
<td>0.15</td>
<td>38</td>
</tr>
<tr>
<td>median</td>
<td>0.93</td>
<td>0.70</td>
<td>31</td>
</tr>
<tr>
<td>mm</td>
<td>0.93</td>
<td>0.42</td>
<td>49</td>
</tr>
<tr>
<td>mt-matmul</td>
<td>0.93</td>
<td>-0.64</td>
<td>71</td>
</tr>
<tr>
<td>mt-vwadd</td>
<td>0.92</td>
<td>-12.63</td>
<td>41</td>
</tr>
<tr>
<td>multiply</td>
<td>0.93</td>
<td>0.44</td>
<td>24</td>
</tr>
<tr>
<td>pmp</td>
<td>0.92</td>
<td>0.81</td>
<td>18</td>
</tr>
<tr>
<td>qsort</td>
<td>0.92</td>
<td>0.72</td>
<td>26</td>
</tr>
<tr>
<td>rsort</td>
<td>0.91</td>
<td>0.21</td>
<td>43</td>
</tr>
<tr>
<td>spmv</td>
<td>0.92</td>
<td>0.31</td>
<td>40</td>
</tr>
<tr>
<td>towers</td>
<td>0.92</td>
<td>0.21</td>
<td>59</td>
</tr>
<tr>
<td>vvadd</td>
<td>0.93</td>
<td>0.72</td>
<td>36</td>
</tr>
</tbody>
</table>
Hierarchical Circuit Representation Learning

- Josh K. (advised by Prof. John Wawrzynek)

1) Pretrain a generalizable model on a larger set of representative circuits towards building a single model that can be fine-tuned to perform various downstream tasks
   - One large pretrained model ➤ finetune to any EDA task
     - Instead of designing and training small models for individual EDA tasks
     - Better performance even with smaller labeled dataset

2) Leverage intrinsic graph characteristics that are unique to netlists
   - **Hierarchical nature** of circuit netlists:
     - module-level, word-level, gate-level netlists
   - **Well-defined semantics for graph similarity**
     ➤ logically equivalent graphs should have similar representations
     Demonstrated by Wang et al. Functionality Matters in Netlist Representation Learning (DAC ‘22)
Hierarchical Circuit Representation Learning

- Gate-Level Connectivity
- Gate-Level Nettlist Graph
- IR-Level Nettlist Graph
- Mem16
- Add4
- Test Parameters
- GCN
- FC
- Coverage Prediction
- Branch / Functional Coverage
- IR-Level HCRL Model
- Gate-Level HCRL Model
- Synthesis Finetune Model
- GCN
- LSTM
- FC
- QoR Prediction
- Critical Path Delay
- Area
Questions