Machine Learning to Predict Successful FPGA Compilation Strategy

PI: Sung Kyu Lim (GT)
Students: Jee Hyun Lee and Anthony Agnesina
Machine Learning to Predict Successful FPGA Compilation Strategy  
**PI:** Sung Kyu Lim, **Students:** Jee Hyun Lee and Anthony Agnesina  
**Funding:** $50K (January 2018 - )  

---  

**Project Definition**  
- Predict compilability of FPGA designs  
  - Compile time and winning strategies  
- Relevance to Industry  
  - Improve compilation time & success rate  
  - Improve FPGA compilation tools  
  - Wider acceptance of FPGA  

---  

**Results and Significance**  
- Built a model for binary classification of  
  - “easy” vs. “hard” designs  
  - Obtained F1 score of 0.8  
- Extracted meaningful features for prediction  
- Analyzed the weaknesses of our model  

---  

**Progress**  
- **Milestone**  
  - Built binary classification model  
- **Deliverables**  
  - Trained models  
  - Python codes  
- **Publications**  
  - ICCAD 2018 submission  

---  

**Future Outlook**  
- Predict winning strategy  
- Improve prediction accuracy  
  - Better tackle data incompleteness  
  - Study predictor dependencies  
- Exploit database update  
  - More data and more predictors  
  - Netlist graphs to become available  

---  

---
Typical FPGA Compilation Process

- We want to predict:
  - Easy vs. hard netlist
  - Winning strategy
  - Improve compilation time & success rate
• Synopsys gave us access to 7 million Zebu logs
  o **Utilization** related: #LUTs, #Regs, #Wires, etc
  o **Connectivity** related: extracted GTL (Group of Tangled Logic), etc
  o **Device** related: FPGA family, generation, etc
  o **Machine** related: processors, CPU freq, total memory, etc

<table>
<thead>
<tr>
<th># wire</th>
<th># LUT</th>
<th>mem free</th>
<th># reg</th>
<th># CLK</th>
<th>CPU MHz</th>
<th>max GTL</th>
<th>Port cut</th>
<th>FPGA</th>
</tr>
</thead>
</table>

Company secret: we can’t show raw data.

• **Non-ideal**
  ▪ Entries missing, majority is hard-netlist
Metrics and Models Used

- **Confusion matrix and F1 score**
  - Positive means hard-netlist
  - Recall = TP / (TP + FN)
  - Precision = TP / (TP + FP)
  - F1 = harmonic mean of above

- **Random forest algorithm**
  - Better than others (SVM, GMM, KNN) for our problem

```
<table>
<thead>
<tr>
<th>Actual</th>
<th>prediction</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neg</td>
<td>TN</td>
<td>N</td>
</tr>
<tr>
<td>Pos</td>
<td>FN</td>
<td>P</td>
</tr>
</tbody>
</table>

# Inst
```
Improving Random Forest Model

- We tried
  - Baseline: unbalanced, non-weighted
  - Balanced (down) sampling
  - Weighted Random Forest
    - Weighted voting 1 (easy) : 9 (hard)
    - Unbalanced training data
    - Best so far!

<table>
<thead>
<tr>
<th>Test</th>
<th>baseline</th>
<th>balanced</th>
<th>weighted</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N</td>
<td>P</td>
<td>N</td>
<td>P</td>
</tr>
<tr>
<td>Actual</td>
<td>12471</td>
<td>3237</td>
<td>12141</td>
<td>3567</td>
</tr>
<tr>
<td></td>
<td>1019</td>
<td>7892</td>
<td>967</td>
<td>7944</td>
</tr>
<tr>
<td></td>
<td>F1 = 0.78</td>
<td>F1 = 0.78</td>
<td>F1 = 0.80</td>
<td>24916</td>
</tr>
</tbody>
</table>

easy:hard = 90:10 database

training

easy netlist

test

hard netlist

sampling

balanced easy:hard = 50:50
• Observations
  
  o Utilization- and machine-related predictors are important
  o Our ROC curve is approaching optimal point

<table>
<thead>
<tr>
<th>rank</th>
<th>feature</th>
<th>rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>n_wire_data</td>
<td>0.082</td>
</tr>
<tr>
<td>2</td>
<td>n_LUT</td>
<td>0.065</td>
</tr>
<tr>
<td>3</td>
<td>mem_free</td>
<td>0.064</td>
</tr>
<tr>
<td>4</td>
<td>n_LUT6</td>
<td>0.057</td>
</tr>
<tr>
<td>5</td>
<td>N_REG</td>
<td>0.050</td>
</tr>
<tr>
<td>6</td>
<td>USD_REG</td>
<td>0.044</td>
</tr>
<tr>
<td>7</td>
<td>CPU_speed</td>
<td>0.042</td>
</tr>
</tbody>
</table>

feature importance

ROC curve (best F1 @ threshold 0.3)
• **Optimize the time cutoff**
  
  o **Improve both the compile time and number of tasks launched**

---

**CT vs #jobs**

- **RF**
- **Perfect Classifier**
- **Normal Operation**

**Axes:**
- **X-axis:** Worst case average Compile time
- **Y-axis:** Average #jobs

---

**Legend:**
- RF (Blue)
- Perfect Classifier (Orange)
- Normal Operation (Green)
Ongoing Directions

- More prediction
  - Predict winning strategy
  - Predict compilation time

- Keep improving accuracy
  - Fine better ways to tackle data incompleteness
  - Better tackle data incompleteness
  - Study predictor dependencies
  - Try other models such as ANN

- Exploit new data
  - More data and more predictors
  - Netlist graphs to become available