ObfusX: Routing Obfuscation With Explanatory Analysis of a Machine Learning Attack

Wei Zeng, Azadeh Davoodi
Department of Electrical and Computer Engineering
University of Wisconsin–Madison, USA

Rasit Onur Topaloglu
IBM, USA

WISCAD
Electronic Design Automation Lab
http://wiscad.ece.wisc.edu/
Background

Split Manufacturing

• Motivation
  – Avoids disclosing complete wiring information of a design layout to an untrusted foundry

• Overview
  – **Untrusted foundry** fabricates the lower (public) part, i.e., transistors and lower metal layers
  – **Trusted foundry** adds the higher (private) part, i.e., remaining metal layers, to complete the chip

Background

Attack Model

• Given:
  – the split level (i.e., topmost public layer)
  – the public part of the design (including v-pins)

• Try to guess connections above split level
  – For a pair of v-pins, the attack model tells if they will be connected to each other on the private part
Related Works

Obfuscation

• Placement-Based
  – Pin swapping
  – Cell insertion
  – Cell location perturbation

• Routing-Based
  – Routing blockage insertion
  – Routing perturbation
  – Wire lifting

• Combination of Techniques

Attack

• Proximity Attack
  – Congestion
  – Network flow
  – Machine learning
Related Works

Obfuscation

• Placement-Based
  – Pin swapping
  – Cell insertion
  – Cell location perturbation

• Routing-Based
  – Routing blockage insertion
  – Routing perturbation
  – Wire lifting

• Combination of Techniques

Attack

• Proximity Attack
  – Congestion
  – Network flow
  – Machine learning

Our Contribution

• ObfusX: the first work incorporating explainability into an obfuscator
  • Identify vulnerable parts and best way to obfuscate
  • Better tradeoff between performance and cost of obfuscation
ML-Based Attack*

High-level workflow

- **Cross validation** is used for testing and training at a specific split level
  - One design is used for testing (leave-one-out)
  - The rest of the designs are used for training
  - Different models built per design

ML-Based Attack*

Extracted features for an individual v-pin

- \( v_x, v_y \)
- \( p_x, p_y \)
- \( \text{InArea, OutArea} \)
- \( W \)

Considering various layout features capturing information from placement, routing, and standard cell library.
ML-Based Attack*

Training sample generation

- Using extracted features, training samples are generated for a pair of v-pins to check if they are a match
  - Including equal number of positive and negative samples in the training set
- Total of 9 features are included in a sample

1. \( \text{diffVpinX} = |v_x1 - v_x2| \)
2. \( \text{diffVpinY} = |v_y1 - v_y2| \)
3. \( \text{diffPinX} = |p_x1 - p_x2| \)
4. \( \text{diffPinY} = |p_y1 - p_y2| \)
5. \( \text{manhattanPin} = \text{diffPinX} + \text{diffPinY} \)
6. \( \text{manhattanVpin} = \text{diffVpinX} + \text{diffVpinY} \)
7. \( \text{totalWireLength} = W_1 + W_2 \)
8. \( \text{totalCellArea} = \sum \text{InArea} + \sum \text{OutArea} \)
9. \( \text{diffCellArea} = \sum \text{OutArea} - \sum \text{InArea} \)

Class: positive (“match”) / negative (“not match”)

ML model: a variant of random forest (an aggregation of REPTrees in Weka#)

# https://www.cs.waikato.ac.nz/ml/weka/
SHAP-Guided Obfuscation

Overview

1. Develop ML attack model
   - Testing set only from the design to obfuscate, i.e., target design
   - Training set from the other designs

2. Perform SHAP-guided obfuscation
   - Use the ML attack model to find the probability that a pair of matching (i.e., actually connected) v-pins in target design may be revealed by the attack
   - Identify vulnerable v-pin pairs for obfuscation
   - For each vulnerable v-pin pair, apply SHAP analysis to its ML prediction and identify its top explaining features
   - Obfuscate each pair iteratively using SHAP-guided via perturbation or wire lifting
SHAP-Based Analysis for a Matching V-Pin Pair

- **Force plot** generated by SHAP analysis shown for a pair of matching v-pins in a target design
  - SHAP analysis decomposes each predicted output by ML into sum of a “base value” and individual contributions of each feature (“SHAP values”)
  - Pink/blue bars correspond to the features that positively/negatively affect the prediction output
  - Length of the bars indicate the magnitude of the SHAP values, i.e. the individual contributions to the prediction
- Quantifies **to what extent, each layout feature** used by the ML attack predicts that matching pair are connected in the private layers
Motivation for SHAP-Guided Obfuscation

- SHAP-based analysis reveals that the top identifying features may vary across the matching v-pin pairs
- Examples shown for two matching pairs from the same design
  - Feature $\text{diffVpinY}$ is the most contributing feature in predicting pair (a) as “match,” while it actually negatively contributes to predicting pair (b)
  - $(0.96 \rightarrow 0.37)$ vs $(0.82 \rightarrow 0.64)$
Motivation for SHAP-Guided Obfuscation

• The figure shows contributions (SHAP values) of two routing features (diffVpinY and manhattanVpin) to a “successful attack” before and after obfuscation
  - Shown as a distribution for all the matching v-pin pairs in design superblue1 with split level M6
• Our SHAP-guided routing obfuscator flattens the distribution and decreases the top contributions
SHAP-Guided Obfuscation

Two Routing Techniques

• Via perturbation  
  – Move the v-pins on the split level
• Wire lifting  
  – Lift wires from a public layer to a private layer

Iterative Process

• Determine promising v-pins (details later)
• Repeat  
  – An obfuscation move (details later)
• Loop until reaching WL budget or no more vulnerable v-pin pairs
SHAP-Guided Obfuscation by Via Perturbation

Identify essential v-pin pairs in target design

Only essential v-pin pairs are considered for perturbation

1. For each net, identify a driving v-pin group which are the v-pins connected to a driving pin using public layers

2. Identify the remaining v-pin groups of the net as non-driving v-pin groups for that net

3. Identify essential v-pin pairs:
   - An essential v-pin pair \((v, v')\) consists of two v-pins
     - One from a non-driving group \(G\)
     - One from a driving v-pin group \(G'\)
   - If there are more than one v-pin in \(G\) or \(G'\), choose one v-pin from each group so that the essential v-pin pair has the closest possible Manhattan distance
SHAP-Guided Obfuscation by Via Perturbation

Only obfuscate eligible v-pins

• For a split layer where vertical wires are preferred, we prioritize perturbations in Y direction.

• Eligible v-pins should satisfy four requirements including:
  1. They are from an essential v-pin pair $p$ in the same net
     • This condition ensures avoiding duplicated or invalid perturbations
  2. The largest SHAP values for $p$ is either from $\text{manhattanVpin}$ or $\text{diffVpinY}$
     • These two features are identified as most contributing to a successful attack in general
     • This condition ensures obfuscating v-pin pairs that are likely identifiable by these two features
  3. The SHAP value from $\text{diffVpinY}$ should not be less than that from $\text{diffVpinX}$
     • This condition ensures effectiveness of perturbing v-pins in Y direction
  4. A v-pin is not eligible if it is involved in more than one essential v-pin pair in its net
     • This is to avoid affecting multiple essential v-pin pairs at the same time when perturbing a v-pin

• For a split layer where horizontal wires are preferred, switch X & Y in the conditions
SHAP-Guided Obfuscation by Via Perturbation

Procedure for SHAP-guided obfuscation (one iteration)

1. Maintain a list of eligible v-pins, sorted in decreasing order of max SHAP values for the pair it belongs; select the top v-pin in the list.
2. Apply trial perturbing moves within a predefined small radius.
3. Calculate the trial gain of each move defined as
   \[
   \text{gain} = \begin{cases} 
   -\Delta f(x)/\Delta WL, & \text{if } \Delta f(x) < 0 \text{ and } \Delta WL \geq 1 \\
   1 - \Delta f(x), & \text{if } \Delta f(x) < 0 \text{ and } \Delta WL \leq 0 \\
   0, & \text{if } \Delta f(x) \geq 0 \text{ or not feasible}
   \end{cases}
   \]

   \(\Delta f(x)\): change in ML model output
   \(\Delta WL\): change in wirelength

4. Commit the perturbing move with the highest gain.
5. Update the feature vector and the SHAP values, re-check the v-pin eligibility, and go to the next iteration.
SHAP-Guided Obfuscation by Via Perturbation

Example showing how perturbing moves (“rip-up and reroute”/RRR) are made

The original wires and vias of the net containing \(v\) and \(v'\). The gray segments are to be removed.

The new location of \(v\) after perturbation is identified. The unconnected parts (including both endpoints of \(v\) and rerouting goals) are identified in the public layers and private layers.

The unconnected parts are reconnected (in blue) using public and private layers respectively, with A* search algorithm in 3D space.
SHAP-Guided Obfuscation by Wire Lifting

- Wire lifting moves wires from public layers to private layers
  - Creates more v-pins which can make the attack more difficult

- A similar flow as in via perturbation is used:
  1. Only consider vias connecting the topmost public layer to the layer immediately below it
  2. Identify vias that, when lifted above the split layer, create an essential v-pin pair
  3. Identify the via (from the ones in step 2) whose maximal SHAP value (from its top contributing feature) is the lowest compared to the other via candidates
  4. Perform wire lifting by applying a similar RRR procedure as in via perturbation
  5. Apply a higher weight for wires in public part to encourage rerouting using private part
Experiment 1

SHAP-guided via perturbation
- **Used designs**: five designs from ISPD’11 benchmark suite
- **Attack model (for evaluation)**: ML attack*
- **Compared cases**:
  1. Original, non-obfuscated design
  2. Obfuscator which perturbed vias by a Gaussian distributed random amount with standard deviation of 1% of the layout height (similar to the one used in work*)
  3. Our SHAP-guided via obfuscator

Experiment 1

Metrics to evaluate the effectiveness of obfuscation (ISPD’11)

- **Hit Rate (HR) at X%:** \((lower = better)\)
  - For a v-pin \(v\), we first build the list of candidates including the top X% of other v-pins \(u\) with the highest ML model output for essential v-pin pair \((v,u)\). These v-pins are predicted by ML to most likely be the match for \(v\).
  - We call it a “hit” of \(v\) if its real matching v-pin is among the v-pins identified above.
  - HR is the average percentage of hits of all v-pins \(v\) in the design, reported for X=0.01 and X=0.1

- **WL Overhead (\(\Delta WL\)%):** \((lower = better)\)
  - Percentage of increase in WL after the obfuscation

- **Perturbed Nets (PN%):** \((lower = better)\)
  - Number of perturbed nets divided by total number of nets that contain any v-pin

- **Perturbed Vias (PV%):** \((lower = better)\)
  - Number of perturbed vias divided by total number of v-pins

- **Total CPU time of obfuscation:** \((lower = better)\)
### Results: SHAP-Guided Via Perturbation

<table>
<thead>
<tr>
<th>Split layer</th>
<th>Design (#v-pins)</th>
<th>No obfuscation HR@0.01% / 0.1%</th>
<th>Random via perturbation (*)</th>
<th>SHAP-guided via perturbation (this work)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>HR@0.01% / 0.1%</td>
<td>ΔWL%</td>
<td>PN% / PV%</td>
</tr>
<tr>
<td>M6</td>
<td>sb1 (44486)</td>
<td>23.79 / 63.33</td>
<td>2.19 / 11.58</td>
<td>3.03</td>
</tr>
<tr>
<td></td>
<td>sb5 (60034)</td>
<td>29.47 / 63.96</td>
<td>5.75 / 20.38</td>
<td>4.09</td>
</tr>
<tr>
<td></td>
<td>sb10 (89846)</td>
<td>31.84 / 64.34</td>
<td>10.24 / 28.31</td>
<td>4.52</td>
</tr>
<tr>
<td></td>
<td>sb12 (80816)</td>
<td>33.01 / 75.58</td>
<td>8.23 / 24.78</td>
<td>3.31</td>
</tr>
<tr>
<td></td>
<td>sb18 (36026)</td>
<td>20.06 / 66.11</td>
<td>4.27 / 16.55</td>
<td>2.64</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>27.63 / 66.66</td>
<td>6.14 / 20.32</td>
<td>3.52</td>
</tr>
<tr>
<td>M4</td>
<td>sb1 (150510)</td>
<td>49.82 / 68.33</td>
<td>6.46 / 25.37</td>
<td>9.50</td>
</tr>
<tr>
<td></td>
<td>sb5 (179844)</td>
<td>38.78 / 60.40</td>
<td>7.54 / 23.84</td>
<td>9.86</td>
</tr>
<tr>
<td></td>
<td>sb10 (200896)</td>
<td>33.50 / 60.21</td>
<td>13.16 / 37.36</td>
<td>8.53</td>
</tr>
<tr>
<td></td>
<td>sb12 (173294)</td>
<td>47.07 / 71.52</td>
<td>9.01 / 22.40</td>
<td>7.61</td>
</tr>
<tr>
<td></td>
<td>sb18 (86658)</td>
<td>29.83 / 59.89</td>
<td>5.15 / 17.89</td>
<td>6.43</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>39.80 / 64.07</td>
<td>8.26 / 25.37</td>
<td>8.39</td>
</tr>
</tbody>
</table>

- HR of ML attack at 0.01% and 0.1% v-pin lists drops drastically after obfuscation
- WL overhead of SHAP-guided obfuscation is less than 1/5 of the random case
- With SHAP-guided obfuscation, only around 30% of v-pins and 60% of nets (that contain v-pins) are perturbed, compared to nearly-all nets and v-pins when perturbed at random
Results: SHAP-Guided Via Perturbation

• Tradeoff between performance and cost of obfuscation

• Compared to random via perturbation:
  – For the same WL overhead of 0.5%, the SHAP-guided version achieves 87% and 97% lower HR in 0.1% and 0.01% v-pin lists, respectively
  – For the same reduction of HR, SHAP-guided version is 3–5X more efficient in WL overhead
Experiment 2

SHAP-guided wire lifting

- **Used designs**: six designs from the ISCAS’85 benchmarks
  - Significantly smaller designs than ISPD’11 but allows comparing with prior work
- **Attack model (for evaluation)**: network flow attack model
- **Compared cases**:
  1. Original, non-obfuscated design
  2. Obfuscator from prior work
  3. Our SHAP-guided obfuscation with wire lifting
     - Used same original layouts and same WL overhead budget as reported in prior work
     - The functional equivalency before and after obfuscation is verified with Synopsys Formality

Experiment 2

Metrics to evaluate the effectiveness of obfuscation (ISCAS’85)

- **Percentage of netlist recovery (PNR):** *(lower = better)*
  - The percentage of correctly reconstructed nets.
  - This quantifies how well the attack can recover the whole design.

- **Output error rate (OER):** *(higher = better)*
  - The probability that there is any error bit in outputs of the reconstructed circuit.

- **Hamming Distance (HD):** *(closer to 50% = better)*
  - The percentage of different bits between outputs of the original circuit and of the reconstructed.

- **WL Overhead (ΔWL%):** *(lower = better)*
  - The percentage of increase in WL after the obfuscation.

- **Total CPU time of obfuscation** *(lower = better)*
Results: SHAP-Guided Wire Lifting

<table>
<thead>
<tr>
<th>Design</th>
<th>#Nets</th>
<th>No obfuscation</th>
<th>Routing-based obfuscation ($)</th>
<th>SHAP-guided wire lifting (this work)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PNR% OER% HD%</td>
<td>PNR% OER% HD% ΔWL%</td>
<td>PNR% OER% HD% ΔWL% t_cpu (min)</td>
</tr>
<tr>
<td>c880</td>
<td>252</td>
<td>100.0 0.0 0.0</td>
<td>91.7 99.9 18.0 4.3</td>
<td>85.3 100.0 23.3 3.4 2.4</td>
</tr>
<tr>
<td>c2670</td>
<td>607</td>
<td>95.8 99.9 7.0</td>
<td>87.1 100.0 14.0 4.4</td>
<td>77.8 100.0 23.5 3.2 7.0</td>
</tr>
<tr>
<td>c3540</td>
<td>638</td>
<td>97.2 95.4 18.2</td>
<td>93.5 100.0 33.4 2.5</td>
<td>84.5 100.0 38.2 2.5 18.4</td>
</tr>
<tr>
<td>c5315</td>
<td>997</td>
<td>98.7 98.7 4.3</td>
<td>95.0 100.0 18.1 1.7</td>
<td>88.9 100.0 23.2 1.7 13.6</td>
</tr>
<tr>
<td>c6288</td>
<td>1921</td>
<td>99.8 36.8 3.0</td>
<td>98.6 100.0 42.1 1.8</td>
<td>95.3 100.0 45.3 1.8 14.1</td>
</tr>
<tr>
<td>c7552</td>
<td>1041</td>
<td>99.6 69.5 1.6</td>
<td>95.3 100.0 20.3 2.2</td>
<td>87.5 100.0 27.2 2.2 12.7</td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td>98.5 66.7 5.7</td>
<td>93.5 (15.0) 100.0 24.3 (18.6)</td>
<td>86.5 (12.0) 100.0 30.1 (24.4) 2.5 11.4</td>
</tr>
</tbody>
</table>

- The proposed SHAP-guided wire lifting performs better than prior work:
  - Reaches 100% for OER
  - Achieves better obfuscation in the reduction of PNR (12% vs 5% on average, or 2.4X better)
  - More increase in HD (24.4% vs 18.6% on average, or 31% better)
  - With same or less WL overhead
- Results in prior work are from a combination of three obfuscation techniques including wire lifting and via perturbation, whereas in our results, wire lifting is applied alone

Conclusion

• Presented a novel approach to incorporate explainability for routing obfuscation
  – Developed two routing obfuscation techniques based on via perturbation and wire lifting
  – Both techniques were guided by explanatory analysis of a ML-based attack, but were verified using an independent attack besides the ML attack
  – Our two obfuscation techniques may be combined with each other (future work)

• Our explanatory analysis and obfuscation approach were not restricted to routing obfuscation; may be generalized in the future
Thank you!

Questions?