Integrating Machine Learning within FPGA Placement

Shawki Areibi, Gary Grewal
Guelph FPGA CAD Group
https://fpga.socs.uoguelph.ca
Why Machine Learning in EDA?

- **Limitations of current Algorithmic based EDA/CAD flows:**
  - Some problems are **too complex for handwritten rules**
    - Congestion Estimation, Routing Prediction, Selection of Optimal Flow, etc.
  - The **rules of a task are constantly changing**
    - FPGA architectures and constraints imposed, etc.
  - Nature of the **data itself keeps changing**
    - Increasing size and complexity of architectures and applications

- **Benefits of Machine Learning for CAD:**
  - Data-driven
  - No explicit programming
  - May assist in cutting CPU time & improve QOR
  - Provides guidance to the flow

**ML for CAD seems like a perfect match!**
Machine Learning in FPGA CAD Flow

Machine Learning + FPGA CAD Algorithms = Smart Flows

Design Entry

RTL Synthesis

Technology Mapping

Circuit Placement

Machine Learning Framework

Global and Detailed Routing

Timing Analysis

Bit Stream
Examples of ML in FPGA Placement

Several problems in FPGA placement can be targeted using ML including:

1. Congestion:
   - Estimation,
   - Forecasting,
   - Management

2. Routability prediction
3. Flow selection/recommendation
4. Timing estimation
5. .......
Outline

- FPGA Placement
- Congestion Estimation
- Routability Prediction
- Adaptive Smart Flow
- Conclusions & Future work
Given a circuit in the form of a netlist, and an FPGA architecture

- Map the components in the netlist onto locations (resources) on the FPGA such that:
  - **Objectives:** Minimize \( \rightarrow \) wirelength, delay, congestion, etc.
  - **Constraints:** based on FPGA Architecture (no overlap, legality control set constraints …)

![Netlist to FPGA mapping diagram]
FPGA Analytical Placement

• Prior approaches to placement used Simulated Annealing.
• Recently, more attention has been directed towards analytic placement, which scales better on large problem instances.

• Analytic placement approach
  1: Convert netlist to graph using Net model
  2: Perform pin propagation
  3: repeat
  4: solve non-linear equation system
  5: partition solution to enforce legality constraints
  6: until termination criteria satisfied
FPGA Analytical Placement

• Prior approaches to placement used Simulated Annealing.

• Recently, more attention has been directed towards analytic placement, which scales better on large problem instances.

• Analytic placement approach
  1: Convert netlist to graph using Net model
  2: Perform pin propagation
  3: repeat
  4:   solve non-linear equation system
  5:   partition solution to enforce legality constraints
  6: until termination criteria satisfied
GPlace3.0

Phase I

Preplacement

WL-driven
Global Placement

Star+ Solver

Bipartitioning Legalization

Find min. window

LUT Sharing

LUT bi-partitioning

FF bi-partitioning

DSP bi-partitioning

BRAM bi-partitioning

Phase II

Congestion Estimation

Cell (LUT) Inflation

Congestion-driven
Global Placement

Star+ Solver

Bipartitioning Legalization

Find min. window

LUT Sharing

(LUT Density) bi-partitioning

FF bi-partitioning

DSP bi-partitioning

BRAM bi-partitioning

Phase III

Detailed Placement (DOISM)

ISM
(minimize Ext. Pins)

ISM
(minimize Wirelength)

Outline

- FPGA Placement
- Congestion Estimation
- Routability Prediction
- Adaptive Smart Flow
- Conclusions & Future work

“A Machine Learning Based Congestion Estimation for Modern FPGAs”,
International Conference on Field Programmable Logic & Applications (FPL 2018), Ireland, pp. 427-434
Congestion occurs when the demand for routing resources exceeds the supply in some region of a design.

Congestion leads to:
- Placements solutions with excessive wirelength
- Degraded performance of the Router
- Subsequent routing stage may fail
Features

- Four features are calculated for each G-Cell of the FPGA
- Each feature is designed to capture routability information at each switch

- $f_1$: Wire Length Per Area
  
  $$f_1 = \sum_{n \in N_i} \frac{w_n \cdot HPWL_n}{\#gc_{ell_n}}$$

- $f_2$: Pin Count (Density)
  
  $$f_2 = \sum_{n \in N_i} \#pins_{n, gc_{ell_i}}$$

- $f_3$: NCPR (5x5)
  
  $$f_3 = |W_{5x5}|$$

- $f_4$: NCPR (9x9)
  
  $$f_4 = |W_{9x9}|$$
We start with 12 ISPD benchmarks:

<table>
<thead>
<tr>
<th>Design</th>
<th>#LUTs (util)</th>
<th>#Flops (util)</th>
<th>#RAMB36</th>
<th>#DSPs</th>
<th>#Control Sets</th>
<th>Rent Exponent</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA-1</td>
<td>50K (9%)</td>
<td>55K (5%)</td>
<td>0 (0%)</td>
<td>0 (0%)</td>
<td>12</td>
<td>0.4</td>
</tr>
<tr>
<td>FPGA-2</td>
<td>100K (19%)</td>
<td>66K (6%)</td>
<td>100 (6%)</td>
<td>100 (13%)</td>
<td>121</td>
<td>0.4</td>
</tr>
<tr>
<td>FPGA-3</td>
<td>250K (47%)</td>
<td>170K (16%)</td>
<td>600 (35%)</td>
<td>500 (65%)</td>
<td>1281</td>
<td>0.6</td>
</tr>
<tr>
<td>FPGA-4</td>
<td>250K (47%)</td>
<td>172K (16%)</td>
<td>600 (35%)</td>
<td>500 (65%)</td>
<td>1281</td>
<td>0.7</td>
</tr>
<tr>
<td>FPGA-5</td>
<td>250K (47%)</td>
<td>174K (16%)</td>
<td>600 (35%)</td>
<td>500 (65%)</td>
<td>1281</td>
<td>0.8</td>
</tr>
<tr>
<td>FPGA-6</td>
<td>350K (65%)</td>
<td>352K (33%)</td>
<td>1000 (58%)</td>
<td>600 (78%)</td>
<td>2541</td>
<td>0.6</td>
</tr>
<tr>
<td>FPGA-7</td>
<td>350K (65%)</td>
<td>355K (33%)</td>
<td>1000 (58%)</td>
<td>600 (78%)</td>
<td>2541</td>
<td>0.7</td>
</tr>
<tr>
<td>FPGA-8</td>
<td>500K (93%)</td>
<td>216K (20%)</td>
<td>600 (35%)</td>
<td>500 (65%)</td>
<td>1281</td>
<td>0.7</td>
</tr>
<tr>
<td>FPGA-9</td>
<td>500K (93%)</td>
<td>366K (34%)</td>
<td>1000 (58%)</td>
<td>600 (78%)</td>
<td>2541</td>
<td>0.7</td>
</tr>
<tr>
<td>FPGA-10</td>
<td>350K (65%)</td>
<td>600K (56%)</td>
<td>1000 (58%)</td>
<td>600 (78%)</td>
<td>2541</td>
<td>0.6</td>
</tr>
<tr>
<td>FPGA-11</td>
<td>480K (89%)</td>
<td>363K (34%)</td>
<td>1000 (58%)</td>
<td>400 (52%)</td>
<td>2091</td>
<td>0.7</td>
</tr>
<tr>
<td>FPGA-12</td>
<td>500K (93%)</td>
<td>602K (56%)</td>
<td>600 (35%)</td>
<td>500 (65%)</td>
<td>1281</td>
<td>0.6</td>
</tr>
</tbody>
</table>

- We also used 372 benchmarks synthesized using an internal netlist-generation tool based on Generate Netlist (Gnl), and provided by our industrial partner – Xilinx Inc.
MLCong: An ML Framework for Congestion

Training

- 372 Ultrascale Benchmarks
- FPGA placer
- Placement file results
- Extract congestion map with TCL script
- Xilinx Detailed Router
- Congestion features
  - NCPR
  - WLPA
  - etc

Filtering

- Training
- Filtering

Testing

- New Circuit
- FPGA placer
- Placement file results
- Congestion features
  - NCPR
  - WLPA
  - etc

Prediction Model

Deployment

- Predicted Congestion of New Circuit

 Offline

Some records correspond to unused regions
# Congestion Estimation Techniques: A Comparison

**Congestion Estimation Methods**

<table>
<thead>
<tr>
<th>Cong. Estimation Methods</th>
<th>Accuracy Metrics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SAD</td>
<td>AANE</td>
</tr>
<tr>
<td>MLCong</td>
<td>2891.28</td>
<td>6.73</td>
</tr>
<tr>
<td>GR</td>
<td>3126.27</td>
<td>7.34</td>
</tr>
<tr>
<td>fGREP</td>
<td>4351.59</td>
<td>9.66</td>
</tr>
<tr>
<td>NCPR</td>
<td>5254.11</td>
<td>11.47</td>
</tr>
<tr>
<td>WLPA</td>
<td>6185.44</td>
<td>14.20</td>
</tr>
</tbody>
</table>

- Sum of Absolute Error (SAD)
- Average Absolute Normalized Error (AANE)
- Root Mean Square Error (RMSE)

MLCong produces congestion heatmaps that are close to those produced by Vivado detailed router.
Comparison of Machine Learning Models

<table>
<thead>
<tr>
<th>Cong. Estimation Methods</th>
<th>Congestion Metrics</th>
<th>Prediction Accuracy Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SAD</td>
<td>AANE</td>
</tr>
<tr>
<td>MLCong</td>
<td>2891.28</td>
<td>6.73</td>
</tr>
<tr>
<td>$HK_{M7}$ [1]</td>
<td>7983.41</td>
<td>19.23</td>
</tr>
</tbody>
</table>

Case Study

<table>
<thead>
<tr>
<th>Congestion Estimation Method</th>
<th># Failures</th>
<th>Routed-WL (Norm.)</th>
<th>Router Runtime (Norm.)</th>
<th>Placer Runtime (Norm.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{MLCong}</td>
<td>2</td>
<td>1.00x</td>
<td>1.00x</td>
<td>1.00x</td>
</tr>
<tr>
<td>mPFGR</td>
<td>2</td>
<td>1.00x</td>
<td>1.19x</td>
<td>1.17x</td>
</tr>
<tr>
<td>No Estimation</td>
<td>225</td>
<td>1.03x</td>
<td>3.15x</td>
<td>0.47x</td>
</tr>
</tbody>
</table>

On Average MLCong is \textbf{300x faster} than the \textit{global router} as a standalone congestion estimation technique.
Congestion Estimation: DLCong

D. Maarouf, A. Shamli, T. Martin, G. Grewal, S. Areibi
“A Deep Learning Framework for Predicting Congestion during FPGA Placement”,
International Conference on Field Programmable Logic & Applications (FPL 2020), Sweden, pp. 138-144
In this work, we claim that:

- The performance of our previous work (MLCong [1]) can be improved by using a deep learning Convolutional Encoder Decoder (CED), that is coined DLCong.
  - DLCong is capable of capturing global behavior of the detailed router by training on feature maps (i.e., images) rather than local individual switches.
  - It is also capable of modeling the non-linear relationships between placement features and routing resources on the FPGA.

- DLCong achieves a prediction accuracy of 94%.
  - A 9% improvement in accuracy over MLCong.

- Scales well with increasing congestion.
- It’s inference time is a few milliseconds.
• CED consists of **five layers** in the encoder and the decoder portion.
• The **input** to the CED are the **feature maps**.
• The **output** of the CED is the estimated **congestion**.
• Convolutional layers **capture** the spatial relation of a switch with its surrounding switches.
### Unstructured metrics

<table>
<thead>
<tr>
<th>Method</th>
<th>RMSE</th>
<th>MAE</th>
<th>$R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>WLPA</td>
<td>9.925</td>
<td>7.195</td>
<td>65.06%</td>
</tr>
<tr>
<td>NCPR5</td>
<td>9.428</td>
<td>7.527</td>
<td>53.76%</td>
</tr>
<tr>
<td>NCPR9</td>
<td>10.205</td>
<td>8.103</td>
<td>52.61%</td>
</tr>
<tr>
<td>fGREP</td>
<td>8.98</td>
<td>6.391</td>
<td>68.39%</td>
</tr>
<tr>
<td>MLCong</td>
<td>5.678</td>
<td>4.109</td>
<td>85.55%</td>
</tr>
<tr>
<td><strong>DLCong</strong></td>
<td><strong>3.74</strong></td>
<td><strong>2.87</strong></td>
<td><strong>94.33%</strong></td>
</tr>
</tbody>
</table>

### Structured metrics

<table>
<thead>
<tr>
<th>Method</th>
<th>GAME(3)</th>
<th>GAME(4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WLPA</td>
<td>7900.8</td>
<td>9242.9</td>
</tr>
<tr>
<td>NCPR5</td>
<td>7845.3</td>
<td>8547.9</td>
</tr>
<tr>
<td>NCPR9</td>
<td>6062.7</td>
<td>7112</td>
</tr>
<tr>
<td>fGREP</td>
<td>6179.7</td>
<td>6855.2</td>
</tr>
<tr>
<td>MLCong</td>
<td>4551.13</td>
<td>5117.9</td>
</tr>
<tr>
<td><strong>DLCong</strong></td>
<td><strong>252.6</strong></td>
<td><strong>322.7</strong></td>
</tr>
</tbody>
</table>

**DLCong** improves upon **MLCong** by **1.42x**, **1.51x** and **8.78%** for RMSE, MAE and $R^2$
A visual comparison of various congestion estimation methods was used.

The intensity of congestion in the upper part of the congestion map is underestimated by linear techniques (fGREP), while DLCong is predicting it accurately.
DLCong: Case Study

- DLCong was tested on placement solutions produced by other state-of-the-art academic tools and commercial placement tools other than GPlace.
- The results indicate that DLCong can generalize to other placers congestion with an accuracy up to 91.28%.

<table>
<thead>
<tr>
<th>Placer</th>
<th>Routable placement</th>
<th>RMSE</th>
<th>MAE</th>
<th>R²</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ripple</td>
<td>336</td>
<td>7.3</td>
<td>4.86</td>
<td>82.45%</td>
</tr>
<tr>
<td>UTplace</td>
<td>317</td>
<td>5.37</td>
<td>4.03</td>
<td>89.17%</td>
</tr>
<tr>
<td>Vivado</td>
<td>262</td>
<td>5.21</td>
<td>3.88</td>
<td>91.28%</td>
</tr>
</tbody>
</table>
Outline

- FPGA Placement
- Congestion Estimation
- Routability Prediction
- Adaptive Smart Flow
- Conclusions & Future work
Routability Prediction: DLRoute

A. Alhyari, A. Shamli, Z. Abuwaimer, S. Areibi and G. Grewal,
"A Deep Learning Framework to Predict Routability for FPGA Circuit Placement."
International Conference on Field Programmable Logic & Applications (FPL 2019), Barcelona, Spain, pp. 334-341
In this work we present a novel, deep-learning framework based on Convolutional Neural Networks to accurately predict the routability of a placement. Integrating such a predictor in an FPGA placement flow can assist in improving QoR/CPU.

Today’s largest FPGA designs can easily take hours to place with no guarantee of routing success. It is crucial for the placement tool to know as early as possible whether a design is routable.

- Unroutable Placement due to Congestion
  ✓ Congestion free, routable placement
An early estimate of routability can help the placer:

I. Avoid pursuing dead-end paths that do not lead to a feasible routing solution,
II. Enables the placer to improve its optimization strategy!
The proposed DL model for predicting routability is integrated within Gplace:

- **Provides feedback** to further improve optimization
- **Avoids** wasting time running the router on a placement which will fail to route
The network takes a congestion heat map of size 480x168 as input.
Four convolutional layers with a depth of 32 filters are used to extract features.
Two fully connected layers are used to classify the flattened vector of features.
A sigmoid output neuron generates a binary label of \{0, 1\} as routability label.
(I) Training and Testing

1. Placer
2. Placement Files
3. Feature Extraction {F1, F2, F3, F4}
4. Vivado Router
5. Labels
6. Training set
7. Testing set
8. CNN Model

(II) Deployment

New placement
Feature Extraction {F1, F2, F3, F4}
Heatmap Generator
Best CNN Model
Routability Label
### Overall Performance

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>Precision</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>M</th>
<th>Train Time</th>
<th>Test Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>97.4%</td>
<td>0.961</td>
<td>0.980</td>
<td>0.970</td>
<td>0.876</td>
<td>115.8 (min)</td>
<td>7.8 (ms)</td>
</tr>
</tbody>
</table>

### Performance on Each Placement Phase

<table>
<thead>
<tr>
<th>Phase</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global (Wirelength) Placement</td>
<td>0.988</td>
<td>0.955</td>
<td>0.993</td>
<td>0.986</td>
<td>0.967</td>
</tr>
<tr>
<td>Global (Congestion) Placement</td>
<td>0.958</td>
<td>0.944</td>
<td>0.962</td>
<td>0.954</td>
<td>0.915</td>
</tr>
<tr>
<td>Detailed Placement</td>
<td>0.983</td>
<td>0.987</td>
<td>0.995</td>
<td>0.826</td>
<td>0.864</td>
</tr>
</tbody>
</table>
The proposed routability predictor was tested on Xilinx Vivado placer and several state-of-the-art academic placers. Each placer was used to generate placements for the 372 benchmarks and routed. Each placer had a success/failure rate to route these benchmarks. These placers could have avoided routing if DLRout were used. The savings in time ranges from 42.7% to 82.1%.

<table>
<thead>
<tr>
<th>Placer</th>
<th>Routability</th>
<th>CPU Time</th>
<th>Saving</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Routable</td>
<td>Non-Routable</td>
<td>Routed</td>
</tr>
<tr>
<td>UTPlace[8]</td>
<td>317 (85%)</td>
<td>55 (15%)</td>
<td>315473</td>
</tr>
<tr>
<td>Ripple[9]</td>
<td>336 (90%)</td>
<td>36 (10%)</td>
<td>338626</td>
</tr>
<tr>
<td>Vivado2015.4</td>
<td>262 (70%)</td>
<td>110 (30%)</td>
<td>209402</td>
</tr>
<tr>
<td>Vivado2018.1</td>
<td>327 (88%)</td>
<td>45 (12%)</td>
<td>527227</td>
</tr>
</tbody>
</table>
Six highly-congested benchmarks were unroutable by GPlace3.0

Integrating the CNN into a placement tool was used to adaptively improve its optimization strategy.

To be able to route highly-congested placement, a feedback from the CNN is used to control the cell inflation parameters.

The six highly-congested benchmarks are now routable.

### Benchmark Routing Results

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Routing Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Wirelength</td>
</tr>
<tr>
<td>FPGA5-6</td>
<td>9900742</td>
</tr>
<tr>
<td>FPGA5-11</td>
<td>11814937</td>
</tr>
<tr>
<td>FPGA5-16</td>
<td>11858397</td>
</tr>
<tr>
<td>FPGA5-19</td>
<td>12069961</td>
</tr>
<tr>
<td>FPGA5-26</td>
<td>12035954</td>
</tr>
<tr>
<td>FPGA7-7</td>
<td>9540692</td>
</tr>
</tbody>
</table>
A. Al-hyari, A. Shamli, T. Martin, S. Areibi and G. Grewal

“An Adaptive Analytic FPGA Placement Framework Based on Deep-Learning”,
In this work we propose a deep-learning framework to accurately forecast congestion that will be present at subsequent placement iterations based on congestion features obtained during the early phases of placement.

We then show how this forecast can be used early in the placement flow to look ahead and make smart optimization decisions with the goal of reducing placement runtimes while maintaining quality of results.
DLForecast Approach: Objective

- DLForecast is used to accurately predict the congestion that could be present at later placement iterations.
- Results are then fed back to the placer and forwarded to DLRoute to obtain the probability of achieving a feasible routing solution.
- A controller within GPlace can then use the routing probability and congestion forecast to decide between several alternative courses of action.
The input to the model is:

I. A set of concatenated placement feature maps (images) each describing a particular feature present in the current placement.

II. Ground-truth features used as a label.

The Model is used to forecast the congestion feature maps of subsequent iterations including the final iteration of Phase I.

Once trained it is deployed and integrated within a placement tool.
DLForecast Architecture

- The **input** of the network is the current placement feature maps (size 480x85).
- The **output** of the network is the predicted congestion at later iterations.
We employ two most commonly used metrics for quantifying prediction accuracy:

- Mean Absolute Error (MAE)
- Coefficient of Determination ($R^2$)

<table>
<thead>
<tr>
<th>Metric</th>
<th>MAE</th>
<th>$R^2$</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Result</td>
<td>0.026</td>
<td>93.5%</td>
<td>0.898</td>
</tr>
</tbody>
</table>
We also employ the Structural Similarity Index Metric (SSIM).

- SSIM is used to measure the similarity between images.
- Unlike MAE and $R^2$, SSIM accounts for luminance distortion, contrast distortion, and loss of correlation.

The DLForecast predicted WLPA feature map for FPGA5. The evaluation metrics for this feature are:

- MAE = 0.025,
- $R^2 = 94.01\%$ and
- SSIM = 0.902
Columns 2 and 3 compare the runtimes of **GPLace3.0** with that of **DLForecast** respectively.

Column 4 shows that GPLace3.0 with DLForecast achieves runtime improvements in the range of 27% to 40%.

Column 5 shows the percentage increase (+) and decrease (-) in wirelength of GPLace3.0 with DLForecast compared to GPLace3.0.

It is clear that the increase/decrease in wirelength are small, with no overall difference in wirelength across the 12 benchmarks.
We next show that the final routed wirelength obtained by integrating GPlace3.0 with DLForecast is comparable to that obtained by other state-of-the-art analytic placement tools.

Column 3 shows that GPlace3.0 with DLForecast obtains a small 1.40\% overall improvement in wirelength compared to RippleFPGA.

Column 5 shows that GPlace3.0 with DLForecast obtains an even larger 2.98\% overall improvement in wirelength compared to UTPlace.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>RippleF [3]</th>
<th>% Imp.</th>
<th>UTPlace [10]</th>
<th>%Imp.</th>
<th>DLFor WL</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA-1</td>
<td>352628</td>
<td>-5.07%</td>
<td>356769</td>
<td>-3.85%</td>
<td>370503</td>
</tr>
<tr>
<td>FPGA-2</td>
<td>645400</td>
<td>-0.59%</td>
<td>642108</td>
<td>-1.11%</td>
<td>649238</td>
</tr>
<tr>
<td>FPGA-3</td>
<td>3262106</td>
<td>+3.34%</td>
<td>3215087</td>
<td>+1.93%</td>
<td>3153013</td>
</tr>
<tr>
<td>FPGA-4</td>
<td>5509661</td>
<td>+1.88%</td>
<td>5409765</td>
<td>+0.07%</td>
<td>5406070</td>
</tr>
<tr>
<td>FPGA-5</td>
<td>9968955</td>
<td>-3.85%</td>
<td>9659958</td>
<td>-7.17%</td>
<td>10352730</td>
</tr>
<tr>
<td>FPGA-6</td>
<td>6180104</td>
<td>+3.61%</td>
<td>6487628</td>
<td>+8.18%</td>
<td>5956791</td>
</tr>
<tr>
<td>FPGA-7</td>
<td>9639639</td>
<td>-1.39%</td>
<td>10104837</td>
<td>+3.28%</td>
<td>9773651</td>
</tr>
<tr>
<td>FPGA-8</td>
<td>8156951</td>
<td>+0.10%</td>
<td>7879022</td>
<td>-3.43%</td>
<td>8149034</td>
</tr>
<tr>
<td>FPGA-9</td>
<td>12305192</td>
<td>+2.62%</td>
<td>12369055</td>
<td>+3.12%</td>
<td>11982531</td>
</tr>
<tr>
<td>FPGA-10</td>
<td>7139694</td>
<td>+3.84%</td>
<td>8794515</td>
<td>+21.94%</td>
<td>6865298</td>
</tr>
<tr>
<td>FPGA-11</td>
<td>11022815</td>
<td>+6.81%</td>
<td>10196038</td>
<td>-0.75%</td>
<td>10272573</td>
</tr>
<tr>
<td>FPGA-12</td>
<td>7363451</td>
<td>-1.47%</td>
<td>7755443</td>
<td>+3.66%</td>
<td>7471911</td>
</tr>
<tr>
<td>Total</td>
<td>81546596</td>
<td>+1.40%</td>
<td>82870225</td>
<td>+2.98%</td>
<td>80403343</td>
</tr>
</tbody>
</table>
Algorithmic CAD based on analytic solutions need human guidance.

Data-Driven ML/DL CAD can aid designers with fast QoR evaluation and guide algorithmic CAD with optimal inputs.

Machine Learning can further assist EDA in several directions:
- Adaptive Hyperparameter tuning
- Guidance to designer to choose best options
- Enhance productivity of optimization techniques.
Thank You

Guelph FPGA CAD Group
Website: https://fpga.socs.uoguelph.ca
Email: sareibi@uoguelph.ca
We are currently adding qualified graduate students to our team. Contact us if interested.