### Behavioral Verilog
- \( C = A + B \)
- \( D = X \times Y \)

### Structural RTL
- OR(…)
- AND(…)
- XOR(…)

### Tree
- AND
- Inputs: …
- Outputs: …

### Edge List
- A \( \text{AND} \) B
- C \( \text{OR} \) D

### Matrix
- \[
\begin{bmatrix}
1 & 2 & 3 & 4 & 5 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\end{bmatrix}
\]

### CNN
- 1 Conv Layer (1 Conv Layer)
- 1 Max Pool Layer (1 Max Pool Layer)
- 1 FC Layer (1 FC Layer)
- L2 Loss (L2 Loss)

---

### Behavioral Verilog
- \( C = A + B \)
- \( D = X \times Y \)

### Structural RTL
- OR(…)
- AND(…)
- XOR(…)

### Tree
- AND
- Inputs: …
- Outputs: …

### Graph
- AND1 \( \text{AND} \) AND2
- AND2 \( \text{OR} \) 7

### Edge List
- AND1 \( \text{AND} \) AND2
- AND2 \( \text{OR} \) 7

### Matrix
- \[
\begin{bmatrix}
\vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\end{bmatrix}
\]

### CNN
- 1 Conv Layer (1 Conv Layer)
- 1 Max Pool Layer (1 Max Pool Layer)
- 1 FC Layer (1 FC Layer)
- L2 Loss (L2 Loss)
Node 0: AND
Node 1: OR
Node 2: OR

Adjacency Matrix

Node 0
Node 1
Node 2

\[
\begin{pmatrix}
0 & 0 & 1 \\
0 & 0 & 1 \\
1 & 1 & 0 \\
\end{pmatrix}
\]

Node Feature Matrix

\[
\begin{pmatrix}
AND \\
OR \\
OR \\
\end{pmatrix}
\]
A, X → gc 128 → gc 64 → gc 32 → pool → dense
$A, X$ → ac 128 → ac 64 → ac 32 → pool → dense
PREDICTING POWER AND AREA FOR CHIP DESIGN

Kalhan Koul
kkoul@stanford.edu

Kartik Prabhu
kprabhu7@stanford.edu

Introduction

Motivation
- Chip designers write code in a hardware description language (ex. Verilog), which is consumed by computer-aided design tools to realize the actual transistors on a chip, as well as their layout
- These CAD tools have a long runtime, owing to their computationally expensive algorithms
- As a result, it can take several hours for a designer to get feedback on the power consumption and area of their design, making it difficult to effectively iterate on designs
- With a deep learning approach, it is possible to predict the area and power of a design in a fraction of the time that CAD tools take

Approach
- Representing code as an input to a neural network is nontrivial and will greatly influence the accuracy of the neural network
- Some work exists that demonstrate approaches to representing code as a vector, but these are only effective for programming languages, not hardware description languages
- Instead, our approach converts Verilog code into a graph of logic gates, which can be represented as an adjacency matrix and node feature matrix to the neural network

Model
- Circuits have a variable number of gates, so each graph has a variable number of nodes
- Utilize a disjoint union of each graph per batch

Baselaline Model
- Consists of GraphConv (gc) layers
- X is node features matrix and \( \hat{A} \) is the normalized Laplacian matrix
- W, b are trainable parameters
- Activation Function: ReLU

\[
\begin{align*}
Z &= \sigma(\hat{A}XW + b) \\
Laplacian Matrix &= Degree Matrix - Adjacency Matrix
\end{align*}
\] (1): Traditional GCN Layer

ARMA Model
- Consists of ARMAConv (ac) layers
- X is node features matrix and \( \hat{A} \) is the normalized Laplacian matrix
- W, V are trainable parameters
- Activation Function: ReLU
- Utilizes skip connections

\[
\begin{align*}
\hat{X}_{t+1}^{(T)} &= \sigma(\hat{A}\hat{X}_{t}^{(T)}W^{(t)} + XV^{(t)}) \\
Z &= 1/K \sum_{k=1}^{K} \hat{X}_{t}^{(T)}
\end{align*}
\] (2): Skip Layer

(3): ARMA Layer

Results and Discussion

Metric
- Percentage of power and area estimates within 20%

<table>
<thead>
<tr>
<th>Model</th>
<th>Training data (4113)</th>
<th>Test Data (456)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphConv</td>
<td>Power: 68%</td>
<td>Power: 58%</td>
</tr>
<tr>
<td></td>
<td>Area: 65%</td>
<td>Area: 61%</td>
</tr>
<tr>
<td>ARMA</td>
<td>Power: 75%</td>
<td>Power: 63%</td>
</tr>
<tr>
<td></td>
<td>Area: 69%</td>
<td>Area: 61%</td>
</tr>
</tbody>
</table>

Runtime
- Synthesis runtime is design size dependent, but inference is constant
- Runtime on 64-bit carry look ahead adder

Discussion
- The model does relatively well in the context of the problem
- Provides an almost instantaneous method of getting a rough estimate of power and area

Future Work
- Add constraints (clock speed, etc.) as inputs to the neural network
- Add process technology as inputs
- Train larger, varied networks, with data from real designs

References