Table of Contents

CircuitNet

CircuitNet: An Open-Source Dataset for Machine Learning Applications in Electronic Design Automation (EDA)

CircuitNet is an open-source dataset dedicated to machine learning (ML) applications in electronic design automation (EDA). We have collected more than 10K samples from versatile runs of commercial design tools based on open-source RISC-V designs with various features for multiple ML for EDA applications.

This documentation is organized as followed:

- Introduction: introduction and quick start.
- Feature Description: name conventions, calculation method, characteristics and visualization.

This project is under active development. We are expanding the dataset to include diverse and large-scale designs for versatile ML applications in EDA. If you have any feedback or questions, please feel free to contact us.

Citation

```
@article{chai2022circuitnet,
  title = {CircuitNet: An Open-Source Dataset for Machine Learning Applications in Electronic Design Automation (EDA)},
  author = {Chai, Zhuomin and Zhao, Yuxiang and Lin, Yibo and Liu, Wei and Wang, Runsheng and Huang, Ru},
  journal={arXiv preprint arXiv:2208.00287},
  year = {2022}
}
```

Intro

CircuitNet

CircuitNet is an open-source dataset dedicated to machine learning (ML) applications in electronic design automation (EDA). We have collected more than 10K samples from versatile runs of commercial design tools based on open-source RISC-V designs with various features for multiple ML for EDA applications. The features are saved seperately as below:

```
.
├── routability_features
| ├── cell_density
| └── congestion
|| ├── congestion_early_global_routing
||| ├── overflow_based
|||| ├── congestion_eGR_horizontal_overflow
|||| └── congestion_eGR_vertical_overflow
||| └── utilization_based
||| ├── congestion_eGR_horizontal_util
||| └── congestion_eGR_vertical_util
|| └── congestion_global_routing
|| ├── overflow_based
||| ├── congestion_GR_horizontal_overflow
||| └── congestion_GR_vertical_overflow
|| └── utilization_based
|| ├── congestion_GR_horizontal_util
|| └── congestion_GR_vertical_util
| ├── DRC
|| ├── DRC_all
|| └── DRC_seperated
| ├── macro_region
| └── RUDY
| ├── RUDY
| ├── RUDY_long
```

```
|  ├── RUDY_short
|  ├── RUDY_pin
|  └── RUDY_pin_long
├── IR_drop_features
|  ├── power_i
|  ├── power_s
|  ├── power_sca
|  ├── power_all
|  ├── power_t
|  └── IR_drop
├── graph_features
|  ├── instance_placement
|  └── netlist
├── doc
|  └── user_guide.pdf
└── script
|  ├── decompress_routability.py
|  ├── decompress_IR_drop.py
   └── generate_training_set.py
```

We separate the features and store them in different directories to enable custom applications. Thus they need to be preprocessed and combined in certain arrangement for training. Our scripts can preprocess and combine different features for training and testing. But we also encourage to implement different preprocessing methods and use different combinations of features.

Quick Start

(1)Based on your target tasks, download Routability Features(for congestion and DRC) or IR Drop Features(for IR drop).

Google Drive

Baidu Netdisk

Decompress with scripts in the script dir

```
python decompress_routability.py
```

or

```
python decompress_IR_drop.py
```

This may take sometime, please be patient.

(2)Run preprocessing script to generate training set for corresponding tasks. Specify your task with option: congestion/DRC/IR_drop.

```
python generate_training_set.py --task [congestion/DRC/IR_drop] --data_path [path_to_decompressed_dataset] --save_path [path_to_save_output]
```
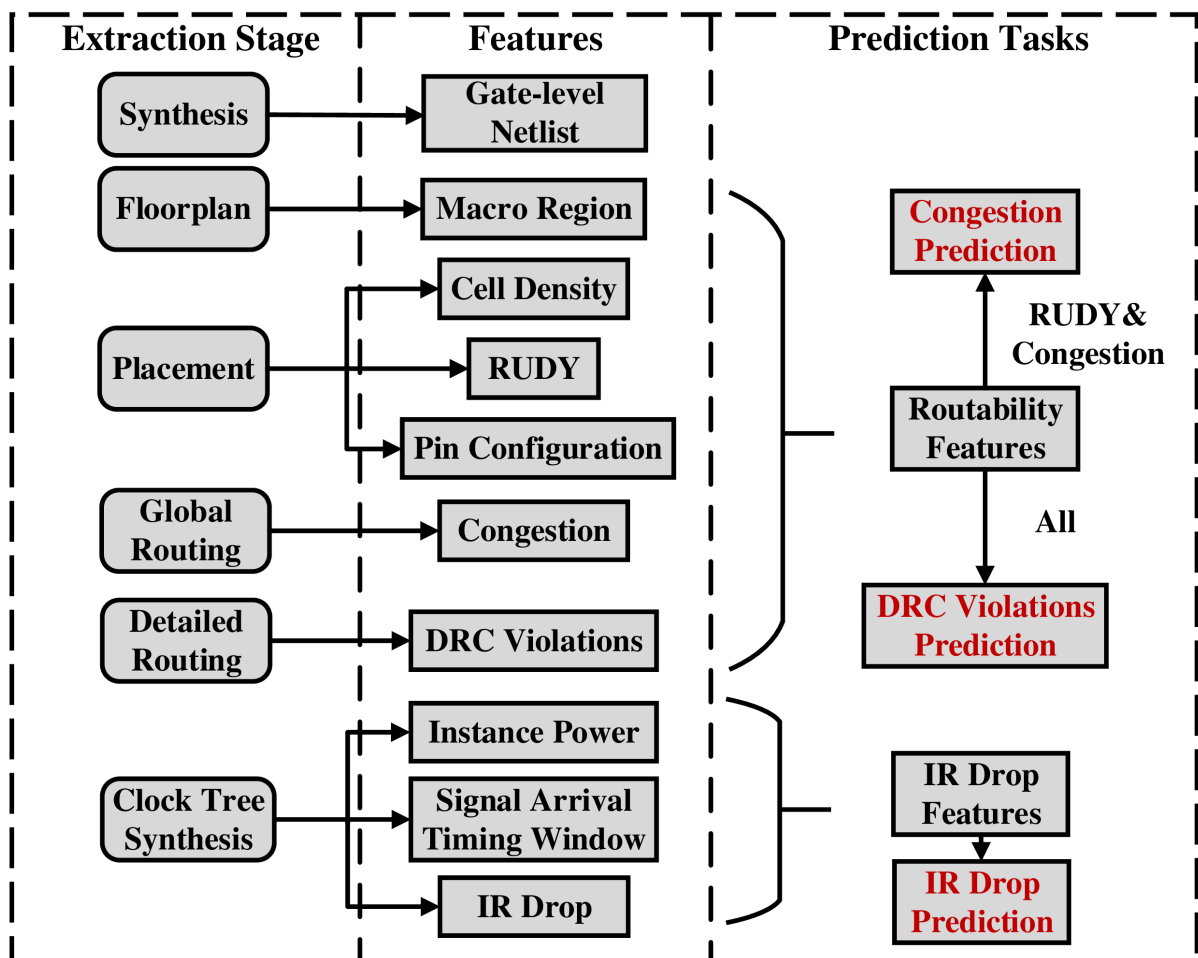
Dataset Overview

The dataset now mainly provide support for three cross-stage prediction tasks in back-end design: congestion prediction, DRC violations prediction and IR drop prediction. The common practice in these tasks is to leverage computer vision methods(e.g. CNN or FCN), thus the main part of CircuitNet is 2D image-like data.

Image-like Feature Maps

The information on layout is converted into image-like feature maps based on tiles of size

$1.5\mu m \times 1.5\mu m$, and they make up the main part of CircuitNet.

Extraction Stage	Features	Prediction Tasks
Synthesis	Gate-level Netlist	
Floorplan	Macro Region	Congestion Prediction
Placement	Cell Density	RUDY& Congestion
	RUDY	Routability Features
	Pin Configuration	All
Global Routing	Congestion	
Detailed Routing	DRC Violations	DRC Violations Prediction
Clock Tree Synthesis	Instance Power	IR Drop Features
	Signal Arrival Timing Window	IR Drop Prediction
	IR Drop	

- **Macro Region**:

  the regions covered by macros, used for estimation routing resources available in each tile.

- **Routability Features**:

  (1) Cell density: the cell number counted in each tile.

  (2) RUDY: a routing demand estimation for each net over spatial dimension. It is widely used for its high efficiency and accuracy. A variation named pin RUDY is also included as the pin density estimation.

(3) Pin configuration: a high resolution representation of pin and routing blockage shapes that conveys pin accessibility in routing.

(4) Congestion: the overflow of routing demand in each tile.

(5) DRC violations: the number of DRC violations in each tile.

- **IR Drop Features**:

(1) Instance power: the instance level internal, switching and leakage power along with the toggles rate from a vectorless power analysis.

(2) Signal arrival timing window: the possible switching time domain of the instance in a clock period from a static timing analysis for each pin.

(3) IR drop: the IR drop value on each node from a vectorless power rail analysis.

Supported Prediction Tasks

Congestion Prediction

Predict congestion at post-placement stages.

Input features:

- Macro region
- RUDY
- Pin RUDY

Label:

Congestion

DRC Violations Prediction

Predict DRC violations at post-global-routing stages.

Input features:

- Macro region
- RUDY
- Pin RUDY
- Cell density
- Congestion

Label:

DRC violations

IR Drop Prediction

Predict IR drop at post-CTS stages.

Input features:

Spatial and temporal power maps

Label:

IR drop

Basic Properties

All features are tile-based. Most information in layout is mapped into tiles with a size of 1.5μm$\times1.5\mu$m. Moreover, layouts are around 450μm$\times450\mu$m, resulting in feature maps of around 300×300 tiles. **In summary, most of the feature maps are 2-dimension numpy array [w, h] unless otherwise indicated.** Their detailed calculations are described in the following sections.

Note that the features need to be preprocessed for training, including resizing and normalization. We provide script of our customized preprocessing method used in our experiment, but there is more than one way to complete preprocessing.

Naming Conventions

10242 samples are generated for feature extraction from 6 original RTL designs with variations in synthesis and physical design as shown in table below.

Design	Synthesis Variations		Physical Design Variations			
	#Macros	Frequency (MHz)	Utilizations (%)	#Macro Placement	#Power Mesh Setting	Filler Insertion
RISCY-a	3/4/5	50/200/500	70/75/80/85/90	3	8	After Placement /After Routing
RISCY-FPU-a						
zero-riscy-a						
RISCY-b	13/14/15					
RISCY-FPU-b						
zero-riscy-b						

The naming convention for extracted feature maps is defined as: {Design name}-{#Macros}-c{Clock}-u{Utilizations}-m{Macro placement}-p{Power mesh setting}-f{filler insertion}

Here is an example: RISCY-a-1-c2-u0.7-m1-p1-f0

Comparison table		
Design name	6 RTL designs	
#Macros	3/4/5 or 13/14/15	1/2/3
Clock	Frequency 500/200/50 MHz	Clock period 2/5/20 ns
Utilizations	70/75/80/85/90%	0.7/0.75/0.8/0.85/0.9
Macro placement	3	1/2/3
Power mesh setting	8	1/2/3/4/5/6/7/8
filler insertion	After placement/After routing	1/0

Routability Features

Macro Region ①

The region on the layout covered by macro which shows the relative routing resource distribution. Region covered and uncovered by macro denoted as different grey scale, 1 and 0, respectively.
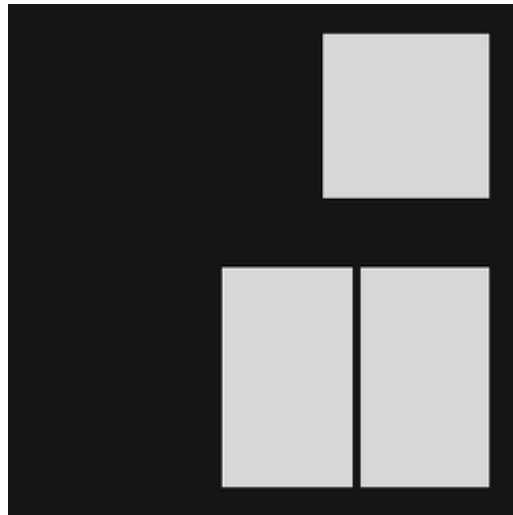
Cell Density ②

Density distribution of cells, which is equivalent to the cell counts in each tile.
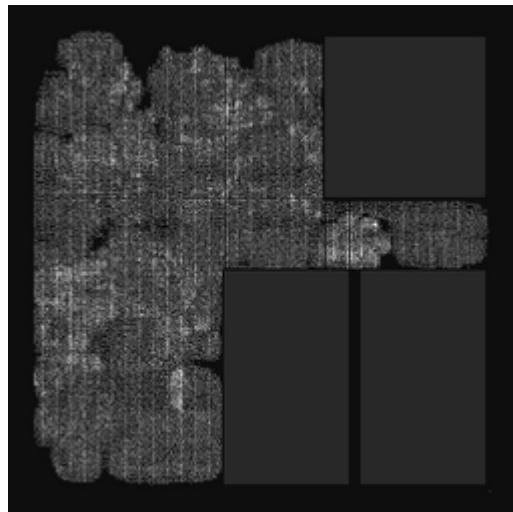
Congestion ③ ~ ⑩

name	computation apporach	stage	direction	used task
congestion_eGR_horizontal_overflow ③	overflow	early global routing	horizontal	Congestion/DRC
congestion_eGR_vertical_overflow ④			vertical	
congestion_GR_horizontal_overflow ⑤		global routing	horizontal	
congestion_GR_vertical_overflow ⑥			vertical	
congestion_eGR_horizontal_util ⑦	utilization	early global routing	horizontal	none
congestion_eGR_vertical_util ⑧			vertical	
congestion_GR_horizontal_util ⑨		global routing	horizontal	
congestion_GR_vertical_util ⑩			vertical	

- Computation method:

  Congestion is computed based on the routing resources reported by Innovus, and there are 2 computation method, overflow based and utilization based. The report basically contains 3 information: total tracks, remain tracks and overflow, based on each GCell, aka tile. Wires have to be routed on tracks, thus tracks are equivalent to routing resources.

  Overflow based congestion is computed as $\frac{overflow}{totaltracks}$. Overflow is the extra demand over total tracks and reflects where congestion occurs.

  Utilization based congestion is computed as $\frac{remaintracks}{totaltracks}$. Utilization reflects the distribution of routing resources.

- Stage: Congestion is reported by Innovus in 2 different stage, eGR and GR. eGR is early global routing, aka trial routing. It is done after placement as a quick and early estimation for congestion. GR is global routing, and the congestion is more accurate than eGR in this stage.

- Direction: The tech lef we use is of type HVH, which meaning that the wires on M1 is horizontal, the ones on M2 is vertical and so on. In this way, the congestion is divided into 2 directions, horizontal and vertical.

RUDY ⑪ ~ ⑮

RUDY refers to Rectangular Uniform wire DensitY which works as a early routing demand estimation after placement. There are several derivatives:

- RUDY ⑪
- RUDY long ⑫
- RUDY short ⑬
- RUDY pin ⑭
- RUDY pin long ⑮

(1) For the kth net with bounding box $(x_{k,min}, x_{k,max}, y_{k,min}, y_{k,max})$, its *RUDY* at tile (i, j) with bounding box $(x_{i,min}, x_{i,max}, y_{j,min}, {j,max})$ is defined as

$$w_k = x_{k,max} - x_{k,min}$$

$$h_k = y_{k,max} - y_{k,min}$$

$$s_k = (min(x_{k,max}, x_{i,max}) - max(x_{k,min}, x_{i,min})) \times (min(y_{k,max}, y_{j,max}) - max(y_{k,min}, y_{j,min}))$$

$$s_{ij} = (x_{i,max} - x_{i,min}) \times (y_{j,max} - y_{j,min})$$

$$RUDY_k(i,j) = \frac{w_k + h_k}{w_k \times h_k} \frac{s_{ij}}{s_k}$$

where min()/max() return the smaller/larger value among 2 inputs, s_{ij} is the area of tile (i, j) and s_k denotes the area of tile (i, j) covered by net k.

(2) *RUDY long* and *RUDY short* are the decomposition of *RUDY*, concerning the length of net k. If net k covers more than 1 tile, it contributes to *RUDY long*. Otherwise, net k covers only 1 tile, then it contributes to *RUDY short*.

(3) *RUDY pin* is calculated on the basis of each pin and the net connected the pin, and it is in analog for pin density. For tile (i, j), *RUDY pin* of a pin belonging to net k is calculated as

$$RUDY pin(i,j) = \frac{w_k + h_k}{w_k \times h_k}$$

RUDY pin long is defined in symmetry with *RUDY long* as the decomposition of *RUDY pin* , i.e., if net k covers more than 1 tile, its pins contributes to *RUDY pin long*.

DRC ⑯

Design rule check violations counted in each tile. Different types of DRC are both saved together in one map and seperately saved.
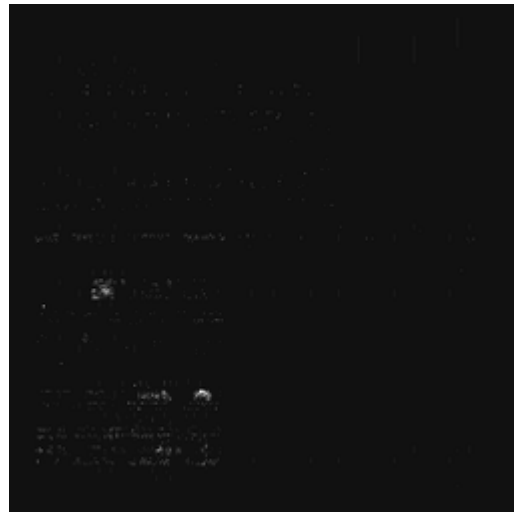
IR Drop Features

Power Maps

Including 5 component：1. internal power: $power_i$, 2. switching power: $power_s$, 3. toggle rate scaled power: $power_{sca}$, 4. all: $power_{all}$, 5. time-decomposed power: $power_t$. They are generated with power report and timing window report from Innovus.

(1) Power report contains instance level power and toggles rate from a vectorless power analysis.

- Internal power (p_i)

- Switching power (p_s)

- Leakage power (p_l)

- Toggles rate (r_{tog})

Then these instance level power is merged into corresponding tile to form power maps.

$$power_i \propto p_i$$

$$power_s \propto p_s$$

$$power_{sca} \propto (p_i + p_s) \times r_{tog} + p_l$$

$$power_{all} \propto p_i + p_s + p_l$$

(2) Timing window report contains possible switching time domain of the instance in a clock period from a static timing analysis for each pin. The clock period is decomposed evenly into 20 parts, and the cell contributes to power map $power_t$ only in the parts that it is switching.

$$power_t[0, 19] \propto p_{sca}$$

IR Drop Map

IR drop value on each node from a vectorless power rail analysis is merged into corresponding tile to form IR drop maps.
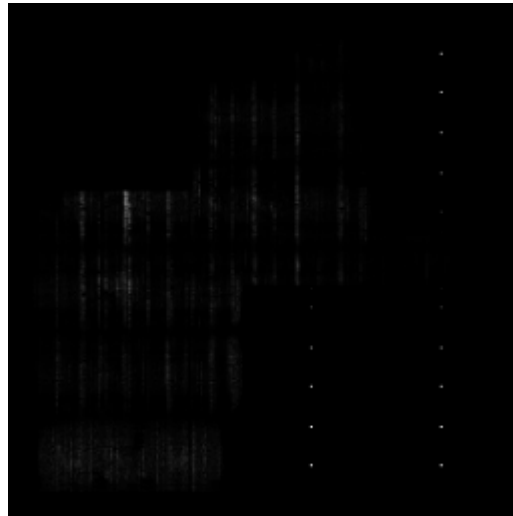
Graph Features

Gate-level Netlist & Instance Placement

To enable application for graph based methods, we further provide 54 gate-level netlists and 10242 instance placement information.

(1) The gate-level netlists are the ones used in data generation. They are synthesised from 6 RISC-V designs with commercial 28 nm library and Synopsys Design Compiler with multiple variations. (see page Feature for detailed information about variations)

The name of standard cell and IP is encrypted because of copyright issue.

(2) The instance, i.e., standard cell and IP, is placed at certain location on layout after placement stage in back-end design.

The placement information for each layout, i.e., the location of instances, is saved as a dictionary, containing the name of instance (consistent with the ones in netlist) and the coordination for the bounding box of instance on layout.

e.g., InstanceN : [left, bottom, right, top]

The dictionary can be loaded with

```
numpy.load(FILE_NAME, allow_pickle=True)
```

(3) Graph can be obtained with the connectivity information from netlist as edges, and the instance placement information as vertices.