Machine Learning for Integrated Circuit Design

CS3308 Machine Learning

April 27, 2024

1 Introduction

- Three students at most form a group.
- Deadline for this project is June 9, 2024, 23:59. Each group needs to submit a report pdf on Canvas and the source code is also required by providing the link to your github repo.
- This project will be evaluated from workload (20%), model performance (20%), results analysis (40%) and report writing (20%). It is more crucial to conduct a reasonable analysis of the experimental results.
- Data required for this project is available in `https://jbox.sjtu.edu.cn/l/01O2vH`.
- Integrated circuit design, also known as IC design, is the process of creating electronic circuits that are integrated onto a single semiconductor chip. The process of IC design involves several stages, starting from conceptualization and specification of the circuit's functionality to the final layout and fabrication of the chip. This project is about the high level synthesis and logic synthesis step. High level synthesis translate the system specification, which can be natural language, or C/C++ code, into hardware description language (HDL) including Verilog and VHDL. Logic synthesis is to translate the HDL into logic representation.

2 Task1: Logic synthesis evaluation prediction

In logic synthesis, the HDL is first transformed into an And-Inverter Graph (AIG), which is a logic graph containing only and gate and inverter gate. Several transformations can be conducted to optimize the representation of AIG. Post-technology mapping is performed using a 7nm technology library to obtain the final netlist. The area and the time delay of the designed chip is estimated as the evaluation of the final AIG. In this task, you need to build a model to predict the evaluation of the AIG. Yosys package is needed to process the AIG graph (`https://github.com/YosysHQ/yosys`). This is a framework for RTL synthesis tools. It currently has extensive Verilog-2005 support and provides a basic set of synthesis algorithms for various application domains. Yosys can be adapted to perform any synthesis job by combining the existing passes (algorithms) using synthesis scripts and adding additional passes as needed by extending the yosys C++ code base.

Hint: Each file in the given dataset is a generation path of the logic synthesis, containing the input state and the learning target. The input state consists of the circuit name and the actions that have been executed.

If you want to obtain the current AIG,

```
state = 'alu2_0130622'
circuitName, actions = state.split('_')
circuitPath = './InitialAIG/train/' + circuitName + '.aig'
libFile = './lib/7nm/7nm.lib'
logFile = 'alu2.log'
nextState = state + '.aig' # current AIG file
synthesisOpToPosDic = {
      0: "refactor",
      1: "refactor -z",
      2: "rewrite",
      3: "rewrite -z",
      4: "resub",
      5: "resub -z",
      6: "balance"
}
action_cmd = ''
for action in actions:
     actionCmd += (synthesisOpToPosDic[int(action)] + ';')
abcRunCmd = "./yosys-abc -c \"read " + circuitPath + ";" + actionCmd + ";
    read_lib " + libFile + ";  write " + nextState + "; print_stats\" > " +
    logFile
os.system(abcRunCmd)
```

If you want to evaluate the AIG with Yosys,

```
abcRunCmd = "./yosys-abc -c \"read " + AIG + "; read_lib " + libFile + ";
    map ; topo;stime \" > " + logFile
os.system(abcRunCmd)
with open(logFile) as f:
     areaInformation = re.findall('[a-zA-Z0-9.]+', f.readlines()[-1])
eval = float(areaInformation[-9]) * float(areaInformation[-4])
```

The evaluation is regularized by the *resyn2*

```
RESYN2_CMD = "balance; rewrite; refactor; balance; rewrite; rewrite -z;
    balance; refactor -z; rewrite -z; balance;"
abcRunCmd = "./yosys-abc -c \"read " + circuitPath + ";" + RESYN2_CMD + "
    read_lib " + libFile + ";  write " + nextState + "; write_bench -l " +
    nextBench + "; map; topo; stime\" > " + logFile
os.system(abcRunCmd)
with open(self.logFile) as f:
     areaInformation = re.findall('[a-zA-Z0-9.]+', f.readlines()[-1])
     baseline = float(areaInformation[-9]) * float(areaInformation[-4])
eval = 1 - eval / baseline
```

AIG can be represented through the node connectivity and the features for each node. For example, *data['edge_index']* stores the edge in the AIG from *edge_src_index* to *edge_target_index*. Each node is represented by its nodeType and the number of inverted predecessors. abc-py is required to process the following example code (`https://github.com/krzhu/abc_py`).

```
_abc = abcPy.AbcInterface()
_abc.start()
_abc.read(state)
data = {}
```

```python
numNodes = _abc.numNodes()
data['node_type'] = np.zeros(numNodes, dtype=int)
data['num_inverted_predecessors'] = np.zeros(numNodes, dtype=int)
edge_src_index = []
edge_target_index = []
for nodeIdx in range(numNodes):
    aigNode = _abc.aigNode(nodeIdx)
    nodeType = aigNode.nodeType()
    data['num_inverted_predecessors'][nodeIdx] = 0
    if nodeType == 0 or nodeType == 2:
        data['node_type'][nodeIdx] = 0
    elif nodeType == 1:
        data['node_type'][nodeIdx] = 1
    else:
        data['node_type'][nodeIdx] = 2
        if nodeType == 4:
            data['num_inverted_predecessors'][nodeIdx] = 1
        if nodeType == 5:
            data['num_inverted_predecessors'][nodeIdx] = 2
    if (aigNode.hasFanin0()):
        fanin = aigNode.fanin0()
        edge_src_index.append(nodeIdx)
        edge_target_index.append(fanin)
    if (aigNode.hasFanin1()):
        fanin = aigNode.fanin1()
        edge_src_index.append(nodeIdx)
        edge_target_index.append(fanin)
data['edge_index'] = torch.tensor([edge_src_index, edge_target_index],
    dtype=torch.long)
data['node_type'] = torch.tensor(data['node_type'])
data['num_inverted_predecessors'] = torch.tensor(data['
    num_inverted_predecessors'])
data['nodes'] = numNodes
```

After obtaining the representation of the AIG, models can be trained for prediction of the AIG evaluation. The dataset needs to be splitted into training set and test set.It is important to note that the number of nodes in the AIG is different. Graph neural networks can be considered in this case.

3 Task2: Logic synthesis decision

In decision-making, the future expected rewards of the current state are more informative than the evaluation of the current state alone. The dataset contains the current state along with its corresponding expected future rewards, you need to train a model to predict future rewards. (The maximum number of actions taken to optimize the AIG is 10. The expected future reward is defined as the evaluation of the final AIG minus the evaluation of the current AIG.)

Now, you have two model to predict the evaluation of current AIG and the future expected reward of the AIG, respectively. Design a search algorithm based on these two models to identify an action sequence to optimize the AIG on the test dataset.

Hint: The greedy method based on the evaluation of AIG is given as an example.

```python
synthesisOpToPosDic = {
     0: "refactor",
    1: "refactor -z",
    2: "rewrite",
    3: "rewrite -z",
    4: "resub",
    5: "resub -z",
    6: "balance"
}
AIG = 'alu4.aig'
libFile = './lib/7nm/7nm.lib'
logFile = 'alu4.log'
for step in range(10):
    childs = []
    for child in range(7):
        childFile = 'alu4_' + str(child) + '.aig'
        abcRunCmd = "./yosys-abc -c \"read " + AIG + ";" +
            synthesisOpToPosDic[child] + "; read_lib " + libFile + "; write
             " + childFile + "; print_stats\" > " + logFile
        os.system(abcRunCmd)
        childs.append(childFile)
    childScores = Evaluation(childs)
    action = argmax(childScores)
    AIG = childs[action]
abcRunCmd = "./yosys-abc -c \"read " + AIG + "; read_lib " + libFile + ";
    map ; topo;stime \" > " + logFile
os.system(abcRunCmd)
with open(logFile) as f:
    areaInformation = re.findall('[a-zA-Z0-9.]+', f.readlines()[-1])
    adpVal = float(areaInformation[-9]) * float(areaInformation[-4])
return (baseline - adpVal) / baseline
```

The real evaluation obtained by yosys-abc can also be used in the decision process along with the learned model.

Search algorithms (such as depth first search, width first search, beam search, A* search, Monte Carlo Tree Search) are recommended. Multiple decision sequence can be generated by the search algorithm and the best one can be as the final result.

4 Task3: High level synthesis with large language model

Large language models (LLM) can be utilized for code generation. Find an example task and using existing LLMs to generate the required Verilog code based on your specific requirements.

Hint: Various large-scale models can be considered, not limited to the following ones: Commercial large-scale models including ChatGPT, GPT4.0, Claude3, WenXinYiYan; open-source LLMs like LLaMa, Gemini, Mixtral; Code generation LLMs.

The task for testing can be chosen according to your own preference. For example, write a code for Row-wise Product in Verilog.

In-context learning is a widely used in LLM, which can used the training samples learning without the need to adjust the parameters of the LLM. The training samples are used as prompts for LLM to assist in generating answers. If feasible, it is worth attempting In-Context learning with the LLM and iteratively generating better solutions. Initial solutions can be generated using the trained model, which can further used as new prompts along with the evaluation to generate more accurate and refined answers using the LLM.

5 Project Report

Each group is required to turn in a project report with your main ideas, utilized methods and algorithms, experimental settings, finally experimental results, and your discussion about the results. The project report (.pdf) can be written either in English (encouraged) or in Chinese.

At the end of the report, please attach the contribution of each member as a percentage. And work done by each student is needed to be clarified. For example,

Name	Student ID	Score	Work
A	00000000000	30%	-
B	00000000001	30%	-
C	00000000002	40%	-

You are also required to submit the source code of your classification model by providing the link to your github repo in the report. If you do not know how to use github, please visit its tutorial (`https://guides.github.com/activities/hello-world/`) for some advice

6 Reference material

Chowdhury, Animesh Basak, et al. "Retrieval-Guided Reinforcement Learning for Boolean Circuit Minimization." The Twelfth International Conference on Learning Representations. 2023.

Fu, Yonggan, et al. "Gpt4aigchip: Towards next-generation ai accelerator design automation via large language models." 2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD). IEEE, 2023.

Chang, Kaiyan, et al. "Chipgpt: How far are we from natural language hardware design." arXiv preprint arXiv:2305.14019 (2023).