OpenROAD Runtime Improvisation by updating C code

Sumanto Kar,
Mumbai, India
Jeetsumanto123@gmail.com

***Abstract*— OpenROAD is a popular Open Source Tool for the RTL to GDS flow. However, it becomes a necessity to improve the tool chain. One such approach would be to optimize the C code written.**

Keywords*— *OpenROAD Flow Scripts, RTL-to-GDSII flow, open-source tools, automated design, no-human-in-the-loop.

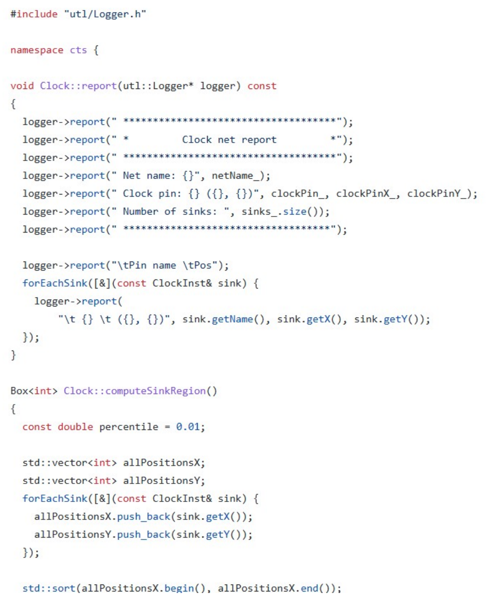
I. INTRODUCTION (*HEADING 1*)

The digital integrated circuit industry has widely embraced OpenROAD, a streamlined RTL-to-GDSII flow offering a comprehensive design implementation solution. This popularity can be attributed to the efficiency of the design flow and the fact that it is open-source. However, it is necessary to reduce the runtime of OpenROAD by improvising/optimizing the C code scripts. Here we look into the clock.cpp.

II. IMPROVEMENTS

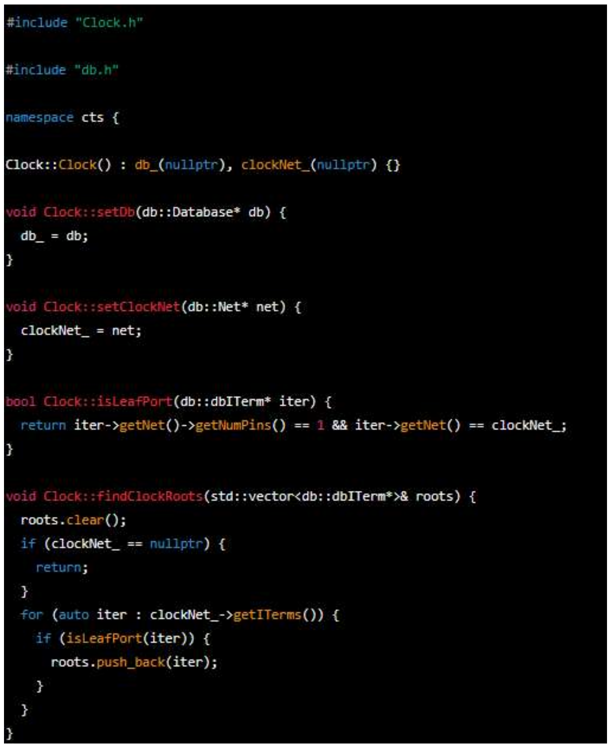
The before and after implementation of the clock.cpp is illustrated below:

Before:



```
#include "utl/Logger.h"

namespace cts {

void Clock::report(utl::Logger* logger) const
{
  logger->report(" ***********************************");
  logger->report(" *         Clock net report        *");
  logger->report(" ***********************************");
  logger->report(" Net name: {}", netName_);
  logger->report(" Clock pin: {} ({}, {})", clockPin_, clockPinX_, clockPinY_);
  logger->report(" Number of sinks: ", sinks_.size());
  logger->report(" **********************************");

  logger->report("\tPin name \tPos");
  forEachSink([&](const ClockInst& sink) {
    logger->report(
        "\t {} \t ({}, {})", sink.getName(), sink.getX(), sink.getY());
  });
}

Box<int> Clock::computeSinkRegion()
{
  const double percentile = 0.01;

  std::vector<int> allPositionsX;
  std::vector<int> allPositionsY;
  forEachSink([&](const ClockInst& sink) {
    allPositionsX.push_back(sink.getX());
    allPositionsY.push_back(sink.getY());
  });

  std::sort(allPositionsX.begin(), allPositionsX.end());
```

After:



```
#include "Clock.h"

#include "db.h"

namespace cts {

Clock::Clock() : db_(nullptr), clockNet_(nullptr) {}

void Clock::setDb(db::Database* db) {
  db_ = db;
}

void Clock::setClockNet(db::Net* net) {
  clockNet_ = net;
}

bool Clock::isLeafPort(db::dbITerm* iter) {
  return iter->getNet()->getNumPins() == 1 && iter->getNet() == clockNet_;
}

void Clock::findClockRoots(std::vector<db::dbITerm*>& roots) {
  roots.clear();
  if (clockNet_ == nullptr) {
    return;
  }
  for (auto iter : clockNet_->getITerms()) {
    if (isLeafPort(iter)) {
      roots.push_back(iter);
    }
  }
}
```

III. CONCLUSIONS AND RECOMMENDATIONS

In conclusion, the removal of many redundant segments of the code. This would decrease the runtime of the code and hence make the OpenROAD execution faster.

REFERENCES

Bibliography

[1] O. Project. [Online]. Available: https://github.com/The-OpenROAD-Project/OpenROAD.

[2] OpenROAD. [Online]. Available: https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts.