Framework of Incremental Detailed VLSI Routing Analysis and Optimization

Chung-Kuan Cheng[1], Daeyeal Lee[2], and Dongwon Park[2]

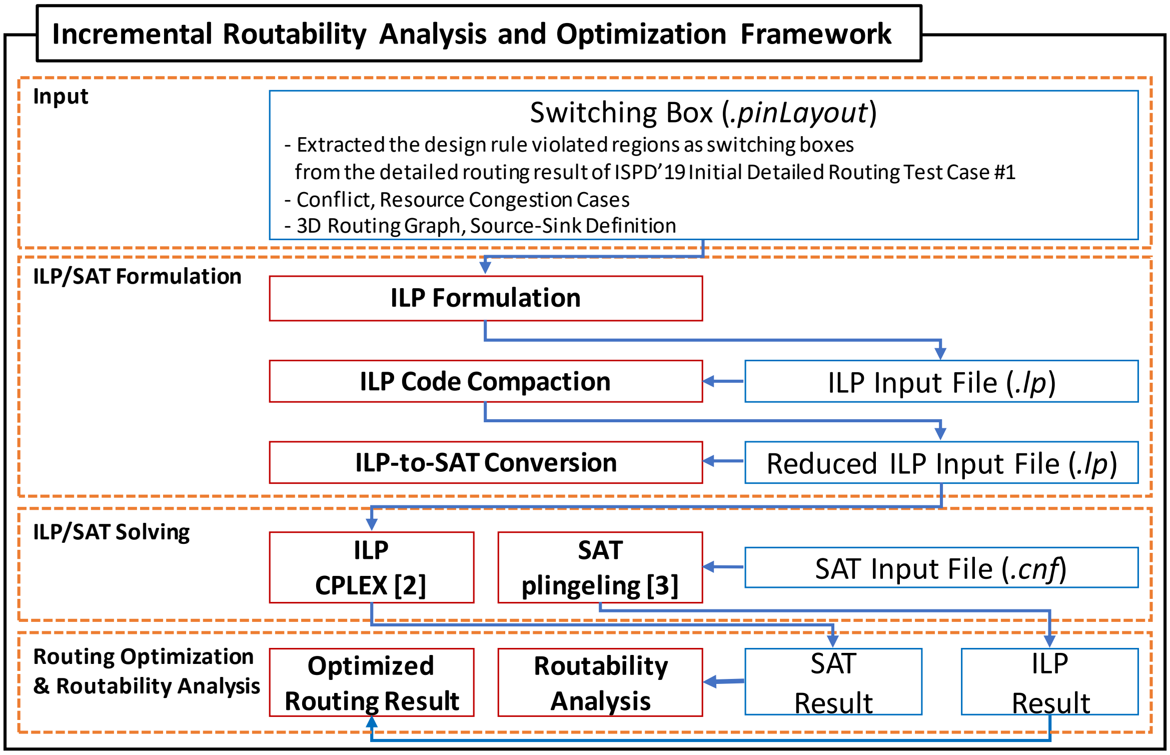
[1]Computer Science and Engineering, [2]Electrical and Computer Engineering, UC San Diego, La Jolla, California

{ckcheng, ldaeyeal, dwp003}@ucsd.edu

1. Overview

This manual briefly summarizes the following flows to generate (i) ILP formulation file (*.lp* file), (ii) SAT formulation file (*.cnf* file), and (iii) solution files to review the detailed layout result. With the given switching box inputs (*.pinLayout* file) which are extracted from the pre-routed 2019 ISPD initial detailed routing test case #1, our flow generates the ILP formulation and convert the ILP formulation to the SAT formulation. We provide a solution viewer to validate the detailed routing result of ILP formulation. We employ *IBM ILOG CPLEX 12.7.1* [1] and p*lingeling (Ver. bcj)* [2] as our ILP and SAT solvers, respectively. Please find more details from our papers [3][4].

(1) Flow Chart for Our Proposed Framework


Incremental Routability Analysis and Optimization Framework
Input
Switching Box (.pinLayout)
- Extracted the design rule violated regions as switching boxes from the detailed routing result of ISPD'19 Initial Detailed Routing Test Case #1
- Conflict, Resource Congestion Cases
- 3D Routing Graph, Source-Sink Definition
ILP/SAT Formulation
ILP Formulation
ILP Code Compaction
ILP Input File (.lp)
ILP-to-SAT Conversion
Reduced ILP Input File (.lp)
ILP/SAT Solving
ILP CPLEX [2]
SAT plingeling [3]
SAT Input File (.cnf)
Routing Optimization & Routability Analysis
Optimized Routing Result
Routability Analysis
SAT Result
ILP Result

(2) Contents in the TAR ball

```
(Current Path)┬/pinLayouts/ispd19_test1_x83200_y104000_w28_h2_t9_d100.pinLayout
              │           /ispd19_test1_x120300_y75000_w28_h2_t9_d100.pinLayout
              │           /ispd19_test1_x80800_y107000_w28_h2_t9_d100.pinLayout
              │           /ispd19_test1_x91500_y92000_w28_h2_t9_d100.pinLayout
              │           /ispd19_test1_x114300_y86000_w28_h2_t9_d100.pinLayout
              ├/inputsILP/ispd19_test1_x91500_y92000_w28_h2_t9_d100.lp
              ├/inputsReducedILP/ispd19_test1_x91500_y92000_w28_h2_t9_d100.lp
              ├/inputsSAT/ispd19_test1_x91500_y92000_w28_h2_t9_d100.cnf
              │          /ispd19_test1_x91500_y92000_w28_h2_t9_d100.variables
              ├/scripts/genILPInput_Ver1.0.pl
              │         /genReducedILPInput_Ver1.0.pl
              │         /genSATInput_Ver1.0.pl
              │         /convILPResult_Ver1.0.pl
              │         /convSATResult_Ver1.0.pl
              ├/solutionsILP/ispd19_test1_x91500_y92000_w28_h2_t9_d100_0_C_812_609_203.conv
              ├/solutionsSAT/ispd19_test1_x91500_y92000_w28_h2_t9_d100_0_C_838_248.conv
              │             /ispd19_test1_x91500_y92000_w28_h2_t9_d100_0_C_838_248.sol
              └/LayoutViewer_Ver1.0.xlsm
```

2. Our Tool-Chain Scripts and Commands with User-Specified Options

Our tool-chain scripts are written in *Perl*. ILP and SAT solvers are *IBM ILOG CPLEX* and *Plingeling* (C language based open-source SAT/SMT solver), respectively.

(1) Testcases (`.pinlayout`)

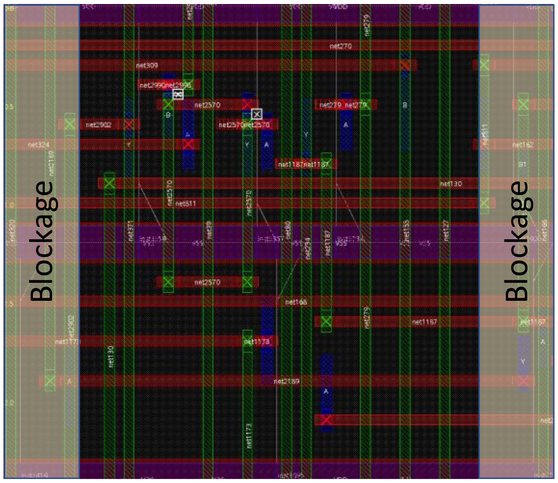
We provide 5 testcases which are extracted from the routing result of ISPD'19 initial detailed routing test #1. Design Rules which are used to route the ISPD'19 testcase are matched with our framework's design rule parameters (MAR=2/EOL=2/VR=2.24/PRL=1/SHR=1).

The description of each testcase is as follows.

Intrinsic Conflict Case

Inputname : ispd19_test1_x114300_y86000_w28_h2_t9_d100.pinLayout
Benchmark : ISPD 2019, test1
Switching Box : (114.3, 86.2) (116.3, 83.8) (um)
# of Vertical/Horizontal Tracks : 20/21
# of Horizontal Blockage : 8

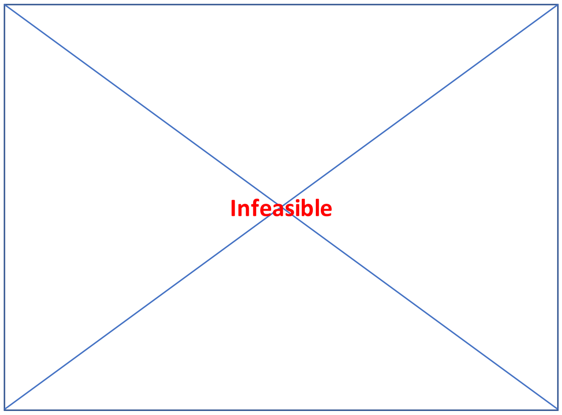
# of Design Rule Violation : 1 [Via(1)]



Original Routing Result

SAT Unsatisfiable

ILP

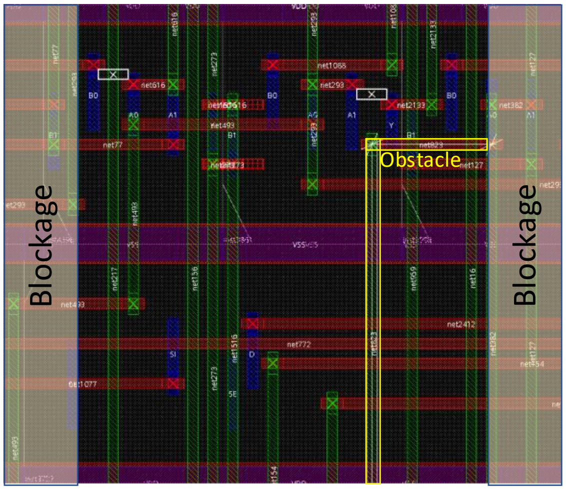


Optimized Routing Result

Obstacle Conflict Case

Inputname : ispd19_test1_x83200_y104000_w28_h2_t9_d100.pinLayout
Benchmark : ISPD 2019, test1
Switching Box : (83.2, 104.2) (85.2, 101.8) (um)
# of Vertical/Horizontal Tracks : 20/21
# of Horizontal Blockage : 8

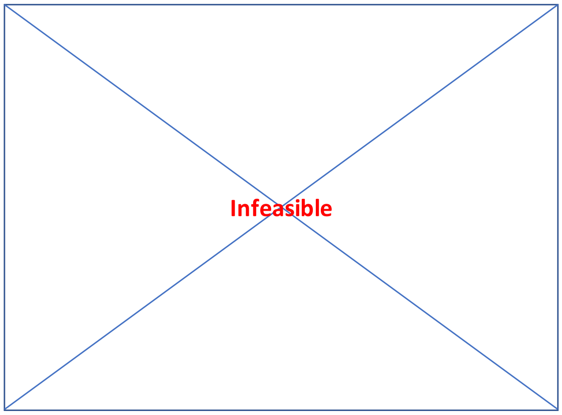
# of Design Rule Violation : 2 [Via(2)]



Original Routing Result

SAT Unsatisfiable

ILP

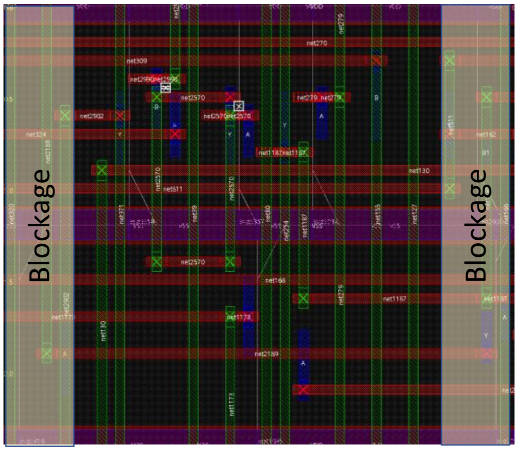


Optimized Routing Result

Resource Congestion Case #1

Inputname : ispd19_test1_x91500_y92000_w28_h2_t9_d100.pinLayout
Benchmark : ISPD 2019, test1
Switching Box : (91.5, 92.2) (93.5, 89.8) (um)
# of Vertical/Horizontal Tracks : 20/21
# of Horizontal Blockage : 8

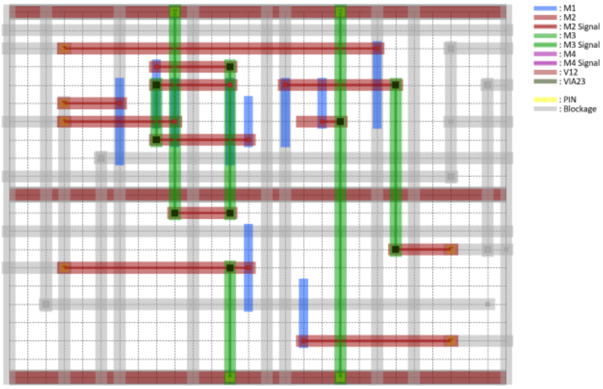
# of Design Rule Violation : 2 [Parallel Run Length(1), Via(1)]



Original Routing Result

SAT Satisfiable

ILP

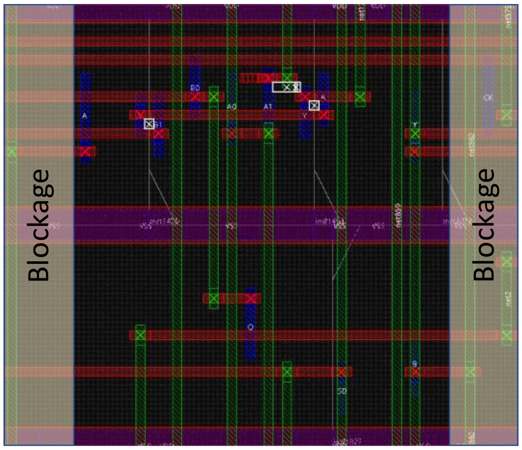


Optimized Routing Result

Resource Congestion Case #2

Inputname : ispd19_test1_x120300_y75000_w28_h2_t9_d100.pinLayout
Benchmark : ISPD 2019, test1
Switching Box : (120.3, 75.4) (122.3, 73) (um)
# of Vertical/Horizontal Tracks : 20/21
# of Horizontal Blockage : 8

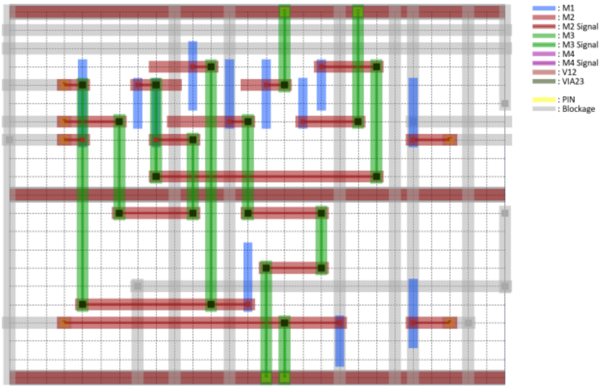
# of Design Rule Violation : 4 [Parallel Run Length(1), Via(3)]



Original Routing Result

SAT Satisfiable

ILP

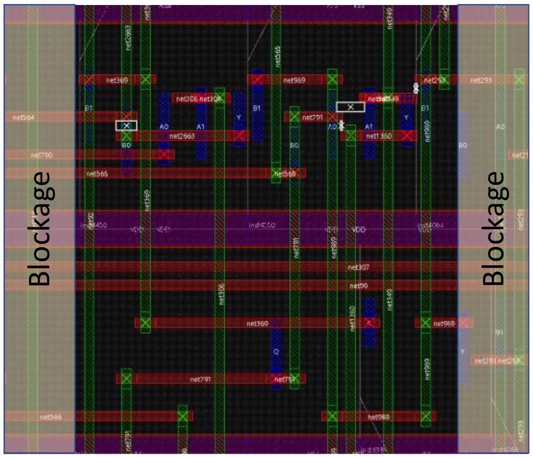


Optimized Routing Result

Resource Congestion Case #3

Inputname : ispd19_test1_x80800_y107000_w28_h2_t9_d100.pinLayout
Benchmark : ISPD 2019, test1
Switching Box : (80.8, 107.8) (82.8, 105.4) (um)
# of Vertical/Horizontal Tracks : 20/21
# of Horizontal Blockage : 8

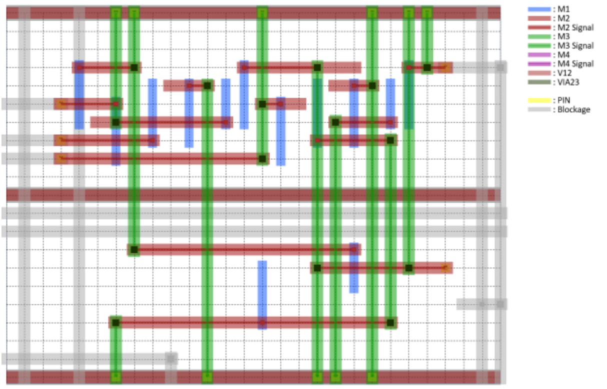
# of Design Rule Violation : 4 [Parallel Run Length(3), Via(1)]



Original Routing Result

SAT Satisfiable

ILP



Optimized Routing Result

(2) ILP Formulation Generation (`genILPinput_Ver1.0.pl`)

[Usage]

```
$ ./scripts/genILPInput_Ver1.0.pl [pinLayout file(.pinLayout)] [MAR Parameter] [EOL Parameter]
[VR Parameter] [PRL Parameter] [SHR Parameter]
```

* Please refer our papers [3][4] for the detailed information of each design rule parameter. The design rules of all testcase are matched with the design rule parameter **MAR=2, EOL=2, VR=2.24, PRL=1, and SHR=1**.

[Example]

Generating the ILP formulation file (*.lp* file) for the switching box layout "`ispd19_test1_x91500_y92000_w28_h2_t9_d100.pinLayout`" with the Design Rule Parameter (MAR=2, EOL=2, VR=2.24, PRL=1, SHR=1)

```
$ ./scripts/genILPInput_Ver1.0.pl ./pinLayouts/ispd19_test1_x91500_y92000_w28_h2_t9_d100.pinLa
yout 2 2 2.24 1 1
```

This will create "`ispd19_test1_x91500_y92000_w28_h2_t9_d100.lp`" file in the `inputsILP` directory. For the *.lp* file format, please visit the following links:

https://www.ibm.com/support/knowledgecenter/en/SS9UKU_12.4.0/com.ibm.cplex.zos.help/FileFormats/topics/LP.html

http://lpsolve.sourceforge.net/5.0/CPLEX-format.htm

(3) ILP Code Compaction (`genReducedILPinput_Ver1.0.pl`)

[Usage]

```
$ ./scripts/genReducedILPInput_Ver1.0.pl [ILP Inputfile(.lp)]
```

[Example]

Generating the Reduced ILP formulation file (*.lp* file) for the Original ILP formulation file “`inputsILP/ispd19_test1_x91500_y92000_w28_h2_t9_d100.lp`”

```
$ ./scripts/genReducedILPInput_Ver1.0.pl ./inputsILP/ispd19_test1_x91500_y92000_w28_h2_t9_d100.lp
```

This will create “`ispd19_test1_x91500_y92000_w28_h2_t9_d100.lp`” file in the `inputsReducedILP` directory.

(4) SAT Formulation Generation (`genSATInput_Ver1.0.pl`)

[Usage]

```
$ ./scripts/genSATInput_Ver1.0.pl [ILP Inputfile(.lp)]
```

[Example]

Converting the ILP formulation file (*.lp* file) “`ispd19_test1_x91500_y92000_w28_h2_t9_d100.lp`” to the SAT formulation file(*.cnf* file)

```
$ ./scripts/genSATInput_Ver1.0.pl ./inputsILP/ispd19_test1_x91500_y92000_w28_h2_t9_d100.lp
```

or

```
$ ./scripts/genSATInput_Ver1.0.pl ./inputsReducedILP/ispd19_test1_x91500_y92000_w28_h2_t9_d100.lp
```

This will create “`ispd19_test1_x91500_y92000_w28_h2_t9_d100.cnf`” file and “`ispd19_test1_x91500_y92000_w28_h2_t9_d100.variables`” in the `inputsSAT` directory. The file with *.variables* extension is the mapping table of variables in *.cnf* file. For the *.cnf* file format, please visit the following links:

https://www.dwheeler.com/essays/minisat-user-guide.html

http://people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html

(5) RUN ILP/SAT Solver (`CPLEX/plingeling`)

[Usage]

```
ILP Solving & Storing solution
$ cplex -c "read [inputFile(.lp)]" "set threads [# of Threads]" "opt" "write
[solPath/solutionName]"
```

```
SAT Solving & Storing solution
$ plingeling -t [# of Threads] [inputFile(.cnf)] > [solPath/solutionName]
```

(6) Solution Converter (`convILPResult_Ver1.0.pl, convSATResult_Ver1.0.pl`)

[Usage]

```
ILP Solution Converter
$ ./scripts/convILPResult_Ver1.0.pl [solPath/solutionName]"
```

This will create "`[solutionName]_0_C_[TotalCost]_[MetalCost]_[WireCost].conv`" file in the `solutionsILP` directory.

```
SAT Solution Converter
$ ./scripts/convSATResult_Ver1.0.pl [solPath/solutionName]"
```

This will create "`[solutionName]_0_C_[TotalCost]_[MetalCost]_[WireCost].conv`" file and "`[solutionName]_0_C_[TotalCost]_[MetalCost]_[WireCost].sol`" in the `solutionsSAT` directory. The file with *.sol* extension is the mapped solution with *.variables* file.

The converted solution files(*.conv* file) can be reviewed using Solution Viewer (`LayoutViewer_Ver1.0.xlsm`)

3. References

[1] *IBM ILOG CPLEX*, http://www.ilog.com/products/cplex/.

[2] *Plingeling*, Multi-Threading SAT Solver, http://fmv.jku.at/lingeling/.

[3] I. Kang, D. Park, C. Han, and C.-K. Cheng, “*Fast and precise routability analysis with conditional design rules*,” in Proceedings of the 20th System Level Interconnect Prediction Workshop, p. 4, ACM, 2018.

[4] D. Park, I. Kang, Y. Kim, S. Gao, B. Lin, and C.-K. Cheng, “*ROAD: Routability analysis and diagnosis framework based on sat techniques*.,” in ISPD, pp. 65–72, 2019.