# 2022 ICCAD ProblemB

0713126 王婕恩
0811562 何祁恩

# Contents


Abstract

Problem Introduction

Procedure

- Readfile and Build Data structure
- Partition
- Standard Cell placement
- Simulate Annealing
- Hybrid Bonding Placement
- Placement visualization

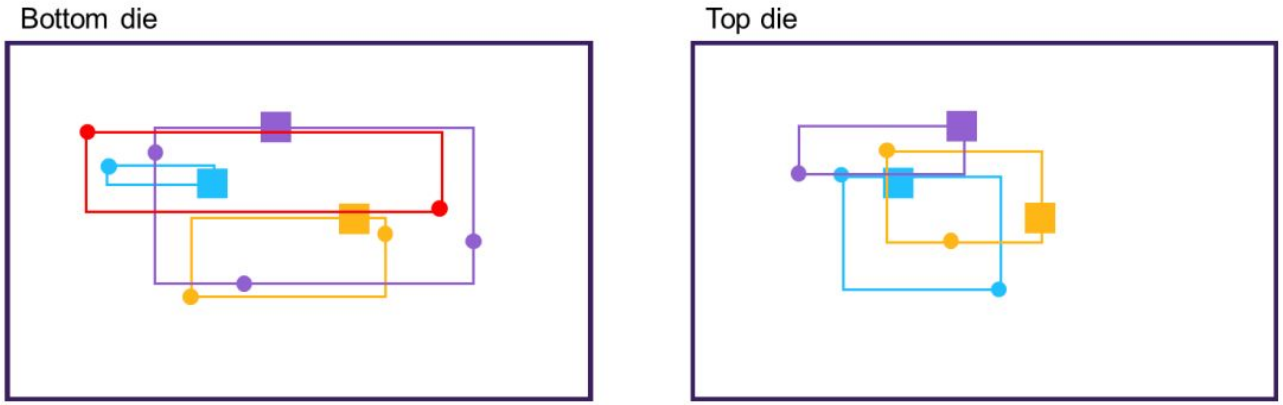
Results And Verifier & Alpha Test Submission

Improvement

Work Assignment

FeedBack

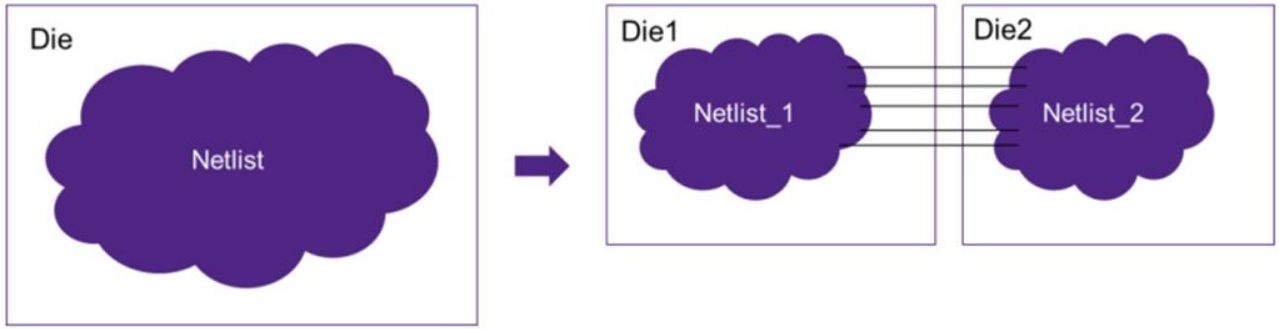
# Abstract




In the future chip design, the trend is to spilt a large die into two or more dies with die to die vertical connections. By doing so, there are some advantages: (1) better yield, timing, and cost (2)each die can be fabricated by different technologies.

In this final project, we combine some algorithms and optimize them so that we can heuristically minimize the HPWL of nets. After that, we can perform global/detailed routing on the result.

# Problem Introduction




Given:

1)Standard cell libraries contain standard cell size and pin location in different technologies.

2) Top die and bottom die technology.

3) The netlist.

Goal:

Place the standard cells on two dies so that minimize the half perimeter wirelength (HPWL) with following brief procedure: Partition, Placement, and Simulate Annealing Algorithm

# Procedure - ReadFile

```
NumTechnologies <technologyCount>: 2

Tech <techName> <libCellCount>: TA 3:
        LibCell <libCellName> <libCellSizeX> <libCellSizeY> <pinCount>: MC1 7 10 1
                Pin <pinName> <pinLocationX> <pinLocationY>: P1 5 7

        LibCell <libCellName> <libCellSizeX> <libCellSizeY> <pinCount>: MC2 14 10 2
                Pin <pinName> <pinLocationX> <pinLocationY>: P1 10 3
                Pin <pinName> <pinLocationX> <pinLocationY>: P2 3 6

        LibCell <libCellName> <libCellSizeX> <libCellSizeY> <pinCount>: MC3 16 10 3
                Pin <pinName> <pinLocationX> <pinLocationY>: P1 5 3
                Pin <pinName> <pinLocationX> <pinLocationY>: P2 3 6
                Pin <pinName> <pinLocationX> <pinLocationY>: P3 10 8



Tech <techName> <libCellCount>: TB 3:
        LibCell <libCellName> <libCellSizeX> <libCellSizeY> <pinCount>: MC1 7 15 1
                Pin <pinName> <pinLocationX> <pinLocationY>: P1 2 11

        LibCell <libCellName> <libCellSizeX> <libCellSizeY> <pinCount>: MC2 12 15 2
                Pin <pinName> <pinLocationX> <pinLocationY>: P1 5 12
                Pin <pinName> <pinLocationX> <pinLocationY>: P2 8 3

        LibCell <libCellName> <libCellSizeX> <libCellSizeY> <pinCount>: MC3 16 15 3
                Pin <pinName> <pinLocationX> <pinLocationY>: P1 2 12
                Pin <pinName> <pinLocationX> <pinLocationY>: P2 3 3
                Pin <pinName> <pinLocationX> <pinLocationY>: P3 15 7
```

We create two types of data structure to store the information about dies and technologies. After that, we can use O(1) time to access these data structure by index.

After partitioning instances into bottom die and top die, we can find cell size of these instances by using the above data structure efficiently.

# Procedure - Partition

Partition the netlist with heuristic algorithms, and try to get the min cut between the two partitions. Since our goal is to minimum the HPWL of the two dies, however we still need to consider the inter connection between two dies so that we can find a great position for hybrid bonding terminal. We use the average size of the standard cell size in different technologies as the weighted vertex, and therefore we might get a better partition result which is area-balanced and might be min-cut. However, there exist a max utility constraints for both bottom die and top die. Therefore, we just keep partitioning with different unbalanced factor to get a legal partition result.

$$\forall StdCell_i,$$

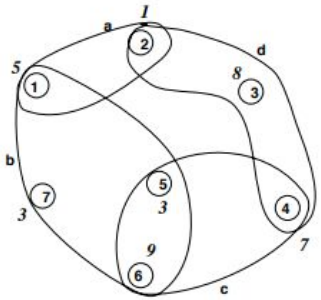
$$w_i = \sum_{NumTechnologies} \frac{Technologies \rightarrow libCellRowLength}{rowCount}$$

# Procedure - Partition - Implementation Detail

Since the size of Hybrid Bonding terminal is much larger than the standard cell size, and the number of hybrid bonding terminals is limited by the die size, therefore, we try to partition instances with min cut so that it will be more convenient when deciding where to place those hybrid terminals.

We use hMETIS to help us partition instances because it is a powerful hypergraph partitioning package and is available using by research. Therefore, we choose this package to help us to perform 2-way partition.

# Procedure - Partition - Implementation Detail




GraphFile

```
4 7 10
1 2
1 7 5 6
5 6 4
2 3 4
5
1
8
7
3
9
3
```

(c)

**shmetis** *HGraphFile* *Nparts* *UBfactor*

After calculating weights of instances, we generate a HGraphFile to feed shmetis a Graphfile like picture(c) shows. Then we can get a partition result.

[hMETIS - Hypergraph & Circuit Partitioning | Karypis Lab]

# Procedure - Partition - Implementation Details

```
Numterminal = 1
Bottom Die: Violate the MaxUtil -> Partition again!
<test for MaxUtil> MaxUtil of current die: 90 Current Partition Util: 103.333333

Numterminal = 1
MaxUtil of current die: 90 Current Partition Util: 66

MaxUtil of current die: 80 Current Partition Util: 68

repartition 25 times
```

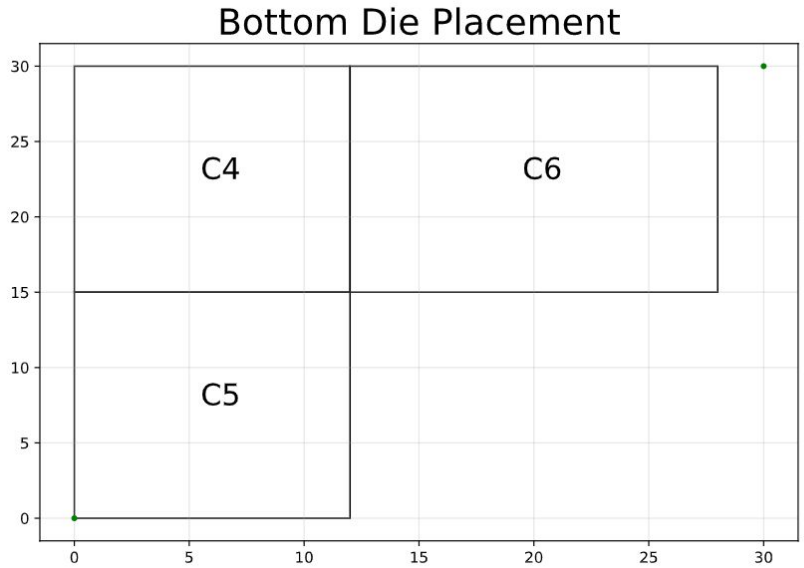
After getting the partition result from shmetis and each size of cell from the data structure we build, we can decide whether top die or bottom die is legal by calculating their sum of rowlengths. If uitility of one of two dies is bigger than its maxima uitlity, we partition again by using different Unblanced factor until we get a legal partition result.

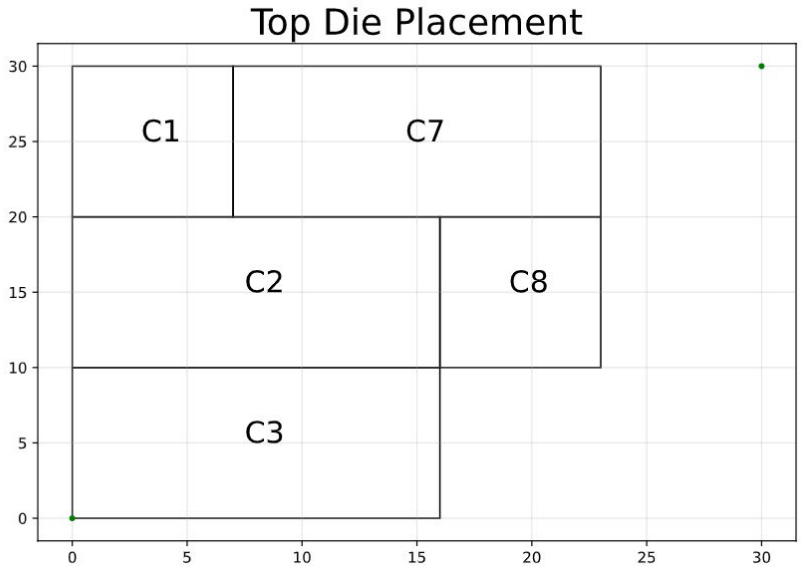
# Procedure - Instance Placement

How to generate rough placement (initial placement)? Simply combine the concept of the hash table and optimized left-edge algorithm. Record the rightmost location of each row and perform linear probing on the whole row until find a non-overlap placement.

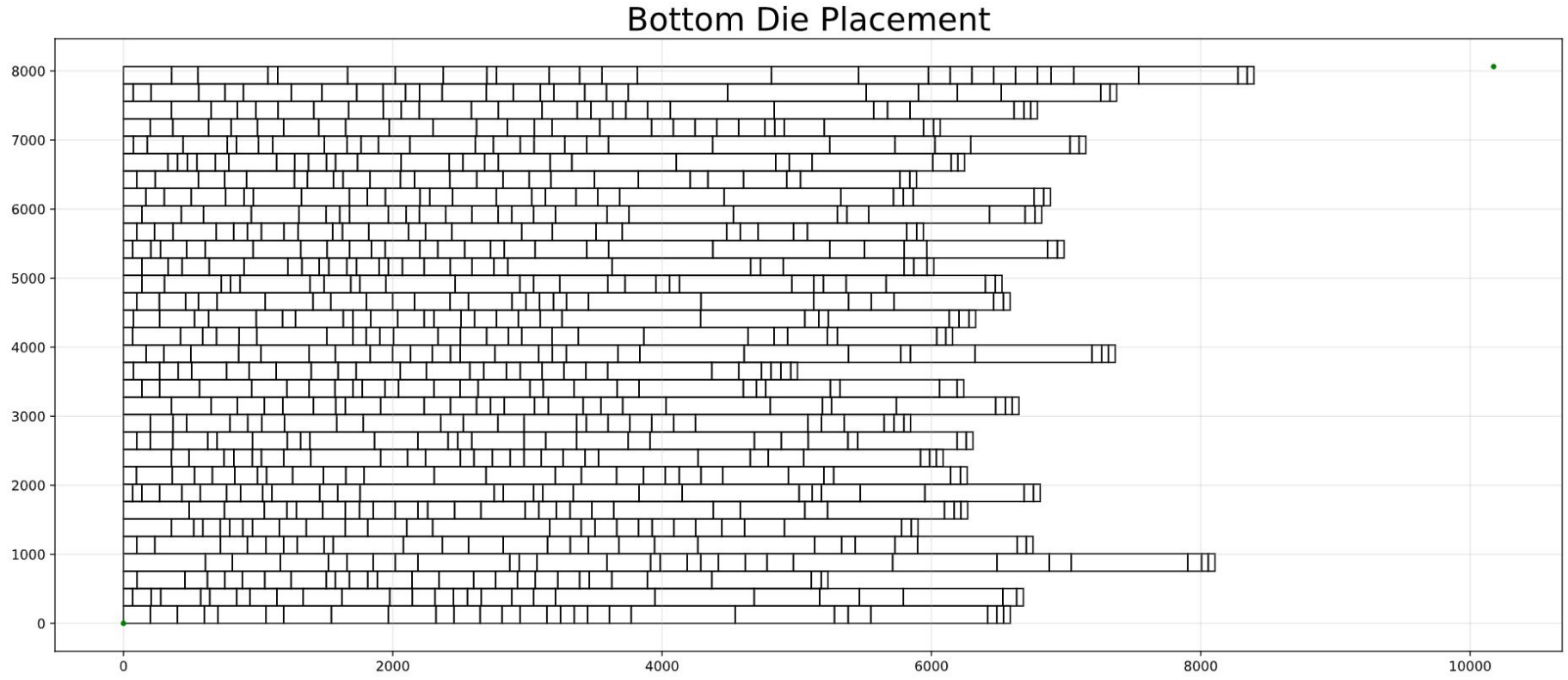
```
watermark[#cell] = {0};
for cell in this die{
        for j = 1 : die->repeatCount{
                int rowSpacing = 2;
                int placeRow = j + rowSpacing % die-> repeatCount
                if(NotOverlap){
                        watermark[j] += cell.length
                }
        }
}
```

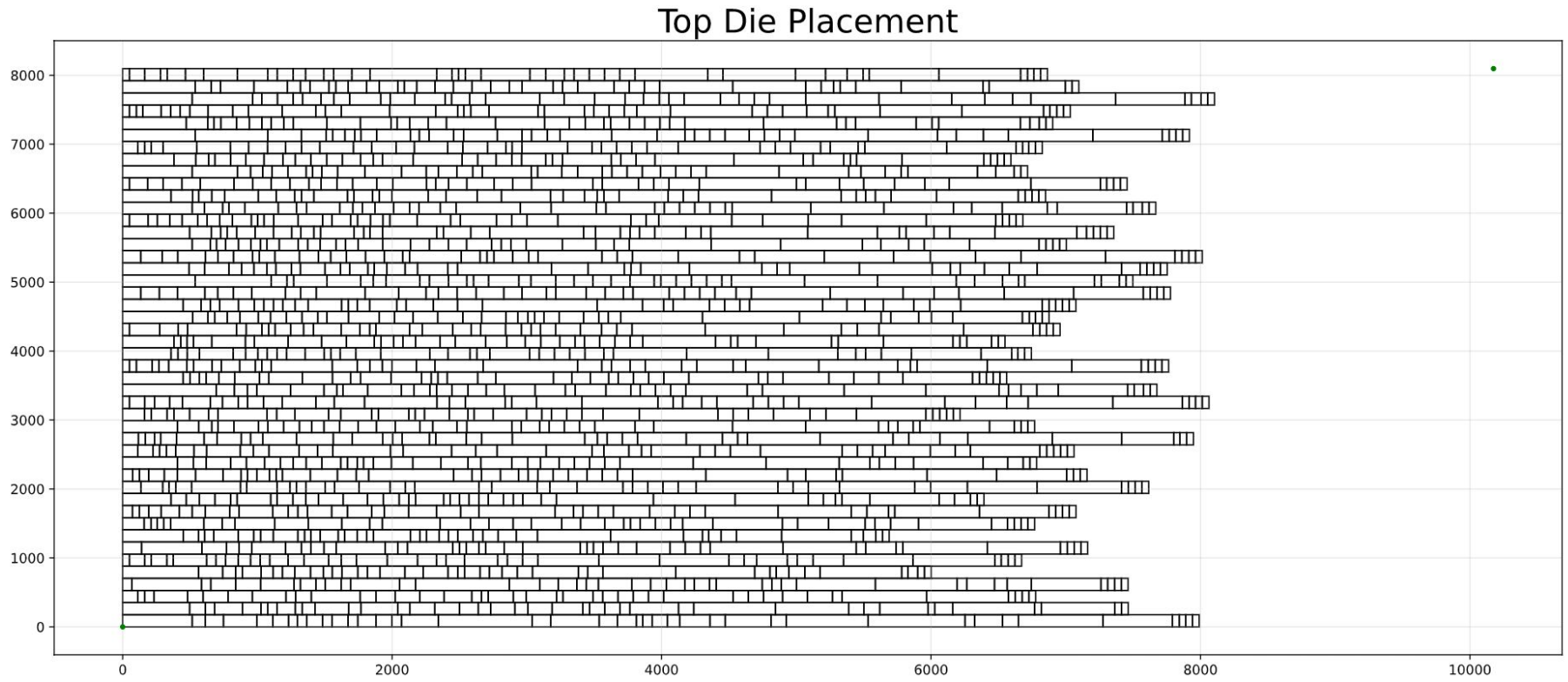
# Procedure - Instance Placement (Case 1)



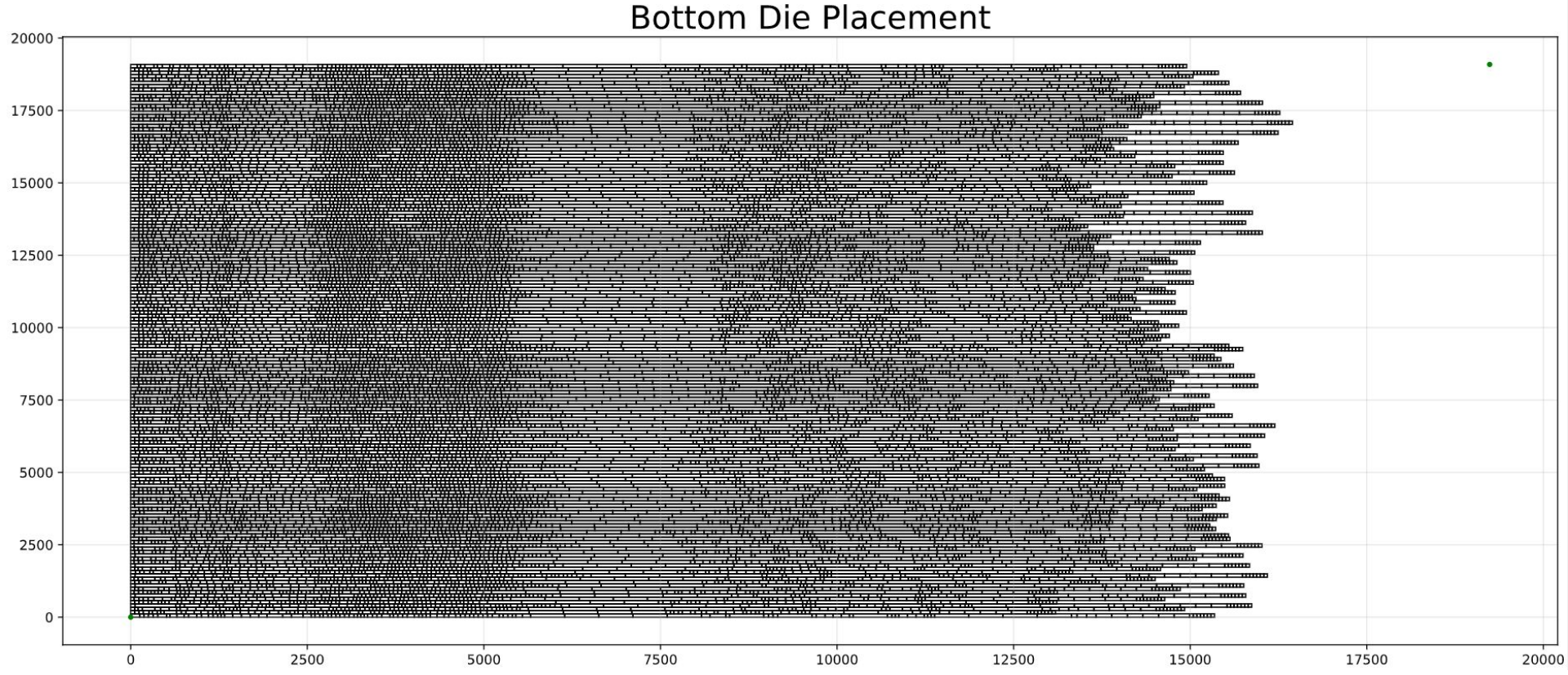
# Procedure - Instance Placement (Case 2)



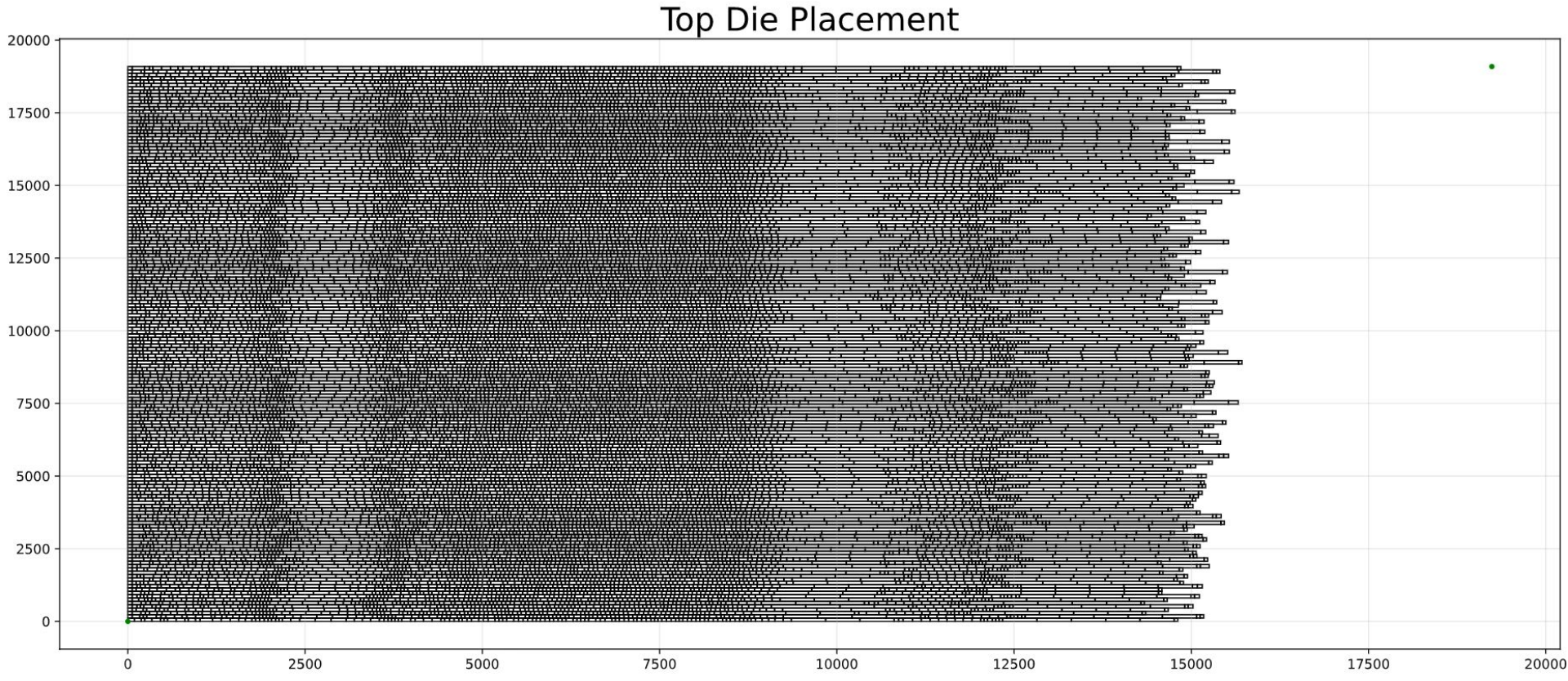



Observe the above pictures, we can find that every instances aggregate on the left hand side of each die for the reason that we make some optimization based on left edge algorithm.

# Procedure - Instance Placement (Case 3)

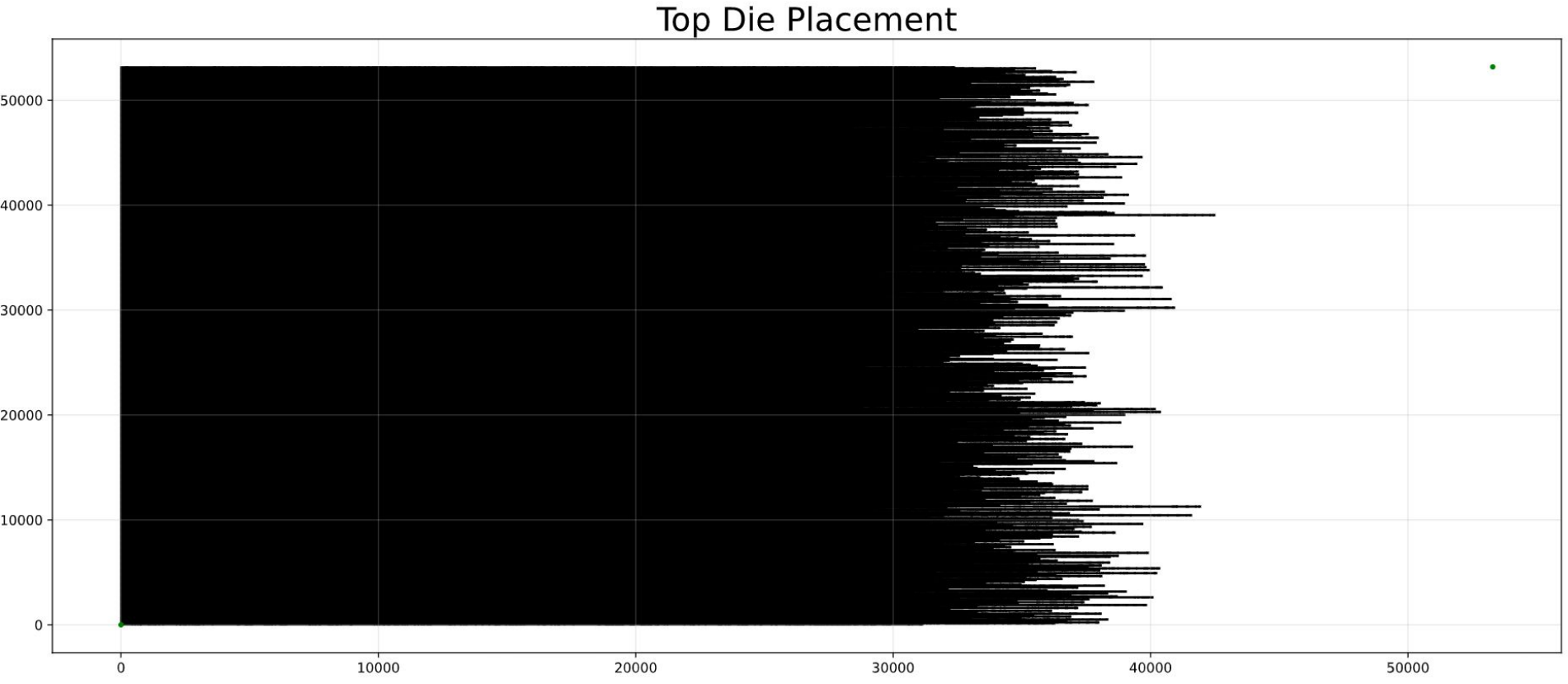


# Procedure - Instance Placement (Case 3)



# Procedure - Instance Placement (Case 4)

# Procedure - Instance Placement (Case 4)



# Procedure - Simulate Annealing

First, we define the cost function when performing the simulate annealing algorithm:

Each Net in Netlist can be represent as a hypergraph H(V,E), where V is the set of the pins that connect with, and E is a set of Nets that connect to those pins. Pin location can be represented as a set of coordinate:

$$PinLoc: \{(x_1, y_1), (x_2, y_2), \dots, (x_i, y_i)\}$$

Then, HPWL for $Net_i$ can be calculated by:

$$HPWL_i = \max_{i \in PinLoc}\{x_i\} - \min_{i \in PinLoc}\{x_i\} + \max_{i \in PinLoc}\{y_i\} - \min_{i \in PinLoc}\{y_i\}$$

Finally, our cost for current placement is:

$$cost = \sum_{i=1}^{NumNets} HPWL_i$$

One should note that our cost function didn't care about the pin exact partition result, since we need to consider both bottom die and top die relationship, therefore, the HPWL might be a 3D bounding box if the pins in some nets are in different layers.

# Procedure - Simulate Annealing Parameter Setting

- Initial Temperature: 200 · INITIAL_COST
- Terminate Temperature: 5E-6 · INITIAL_COST · NumNets
- Loop times per Temperature: 20 · NumInstances
- Temperature decay rate: 0.95

Reference:


AmeerAbdelhadi/Simulated-Annealing-Cell-Based-Placer: This is an implemention of a simulated-annealing standard-cell placement tool; The tool assigns physical locations to each cell in a circuit. (github.com)

Walid Coap 2004 | PDF | Mathematical Optimization | Applied Mathematics (scribd.com)

# Procedure - Simulate Annealing Main Process

```
while (INITIAL_TEMPERATURE > TERMINATE_TEMPERATURE){
        for (int i=0; i < Loop times per Temperature; i++){
                int random_instance_id;
                new_result = find_new_place(random_instance_id);
                if(accept(cost(new_result), cost(old_result), Temperature)){
                        old_result = new_result;
                }
        }
        INITIAL_TEMPETATURE = INITIAL_TEMPETATURE * Temperature decay rate;
        if(ConvergeCondition){
                return placement result;
        }
}
```

# Procedure - Simulate Annealing Inner Function

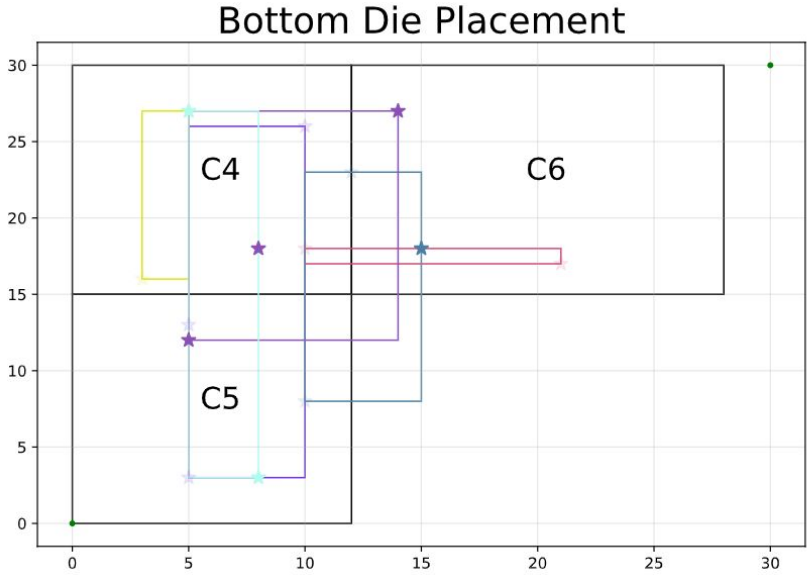
```
bool accept(int new_cost, int old_cost, Temperature){
        int delta_cost = new_cost - old_cost;
        double y = f (delta_cost, temperature);
        double r = random (0, 1);
        if (r < y) return true;
        return false;
}
```

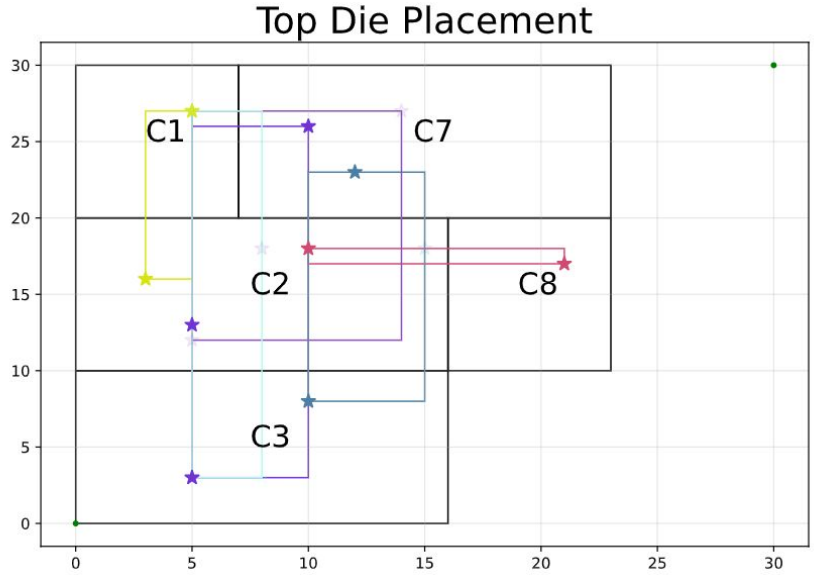
```
double f (delta_cost, temperature){
        return min[1, exp (-1·delta_cost / temperature)];
}
```

Reference:
The TimberWolf placement and routing package | IEEE Journals & Magazine | IEEE Xplore

# Procedure - Simulate Annealing Visualization (Case 1)

<Before Simulate Annealing>  Cost = 124

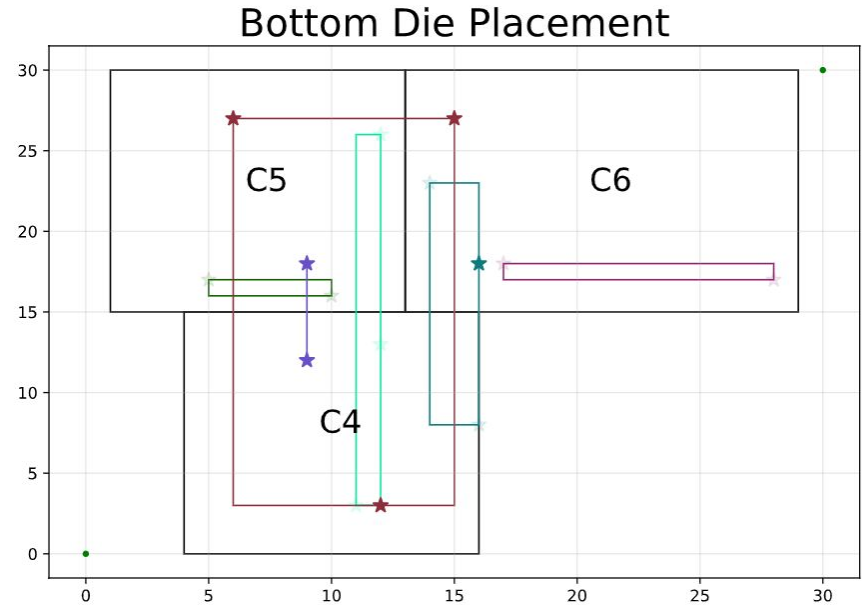


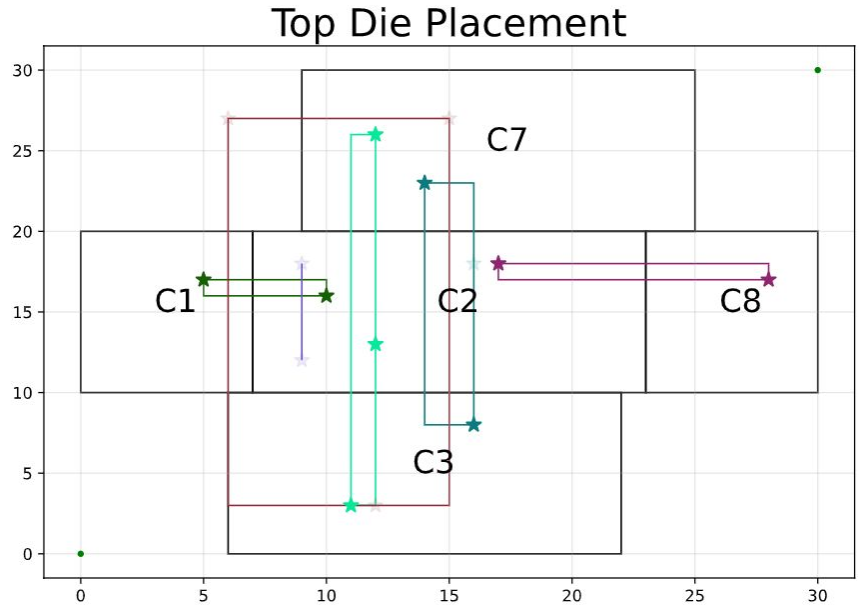




We combine two dies to draw a bounding box (also for calculating costs) for each net so that two dies will be more related. Each star represents a pin and we connect every pins on the same net to draw a bounding box. A transparent pin indicates that it is located on another die.

# Procedure - Simulate Annealing Visualization (Case 1)

<After Simulate Annealing>  Cost = 98

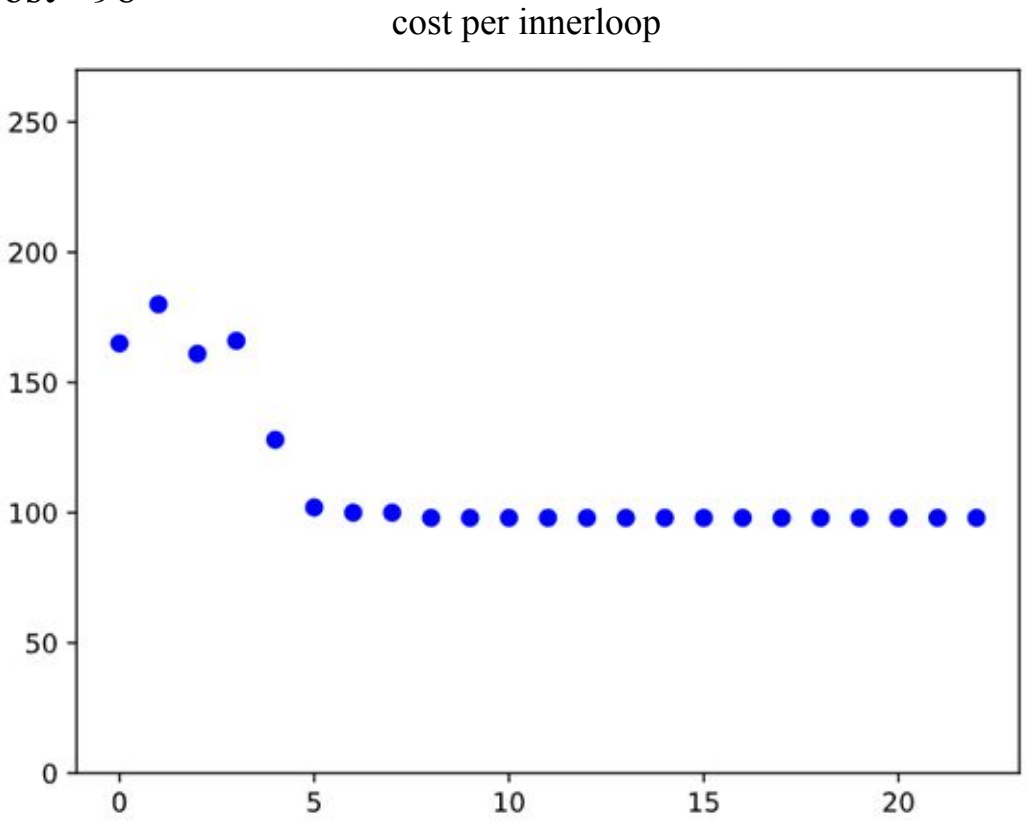



As you can see, after simulate annealing, every instances will find a better place to realize a lower HPWL.

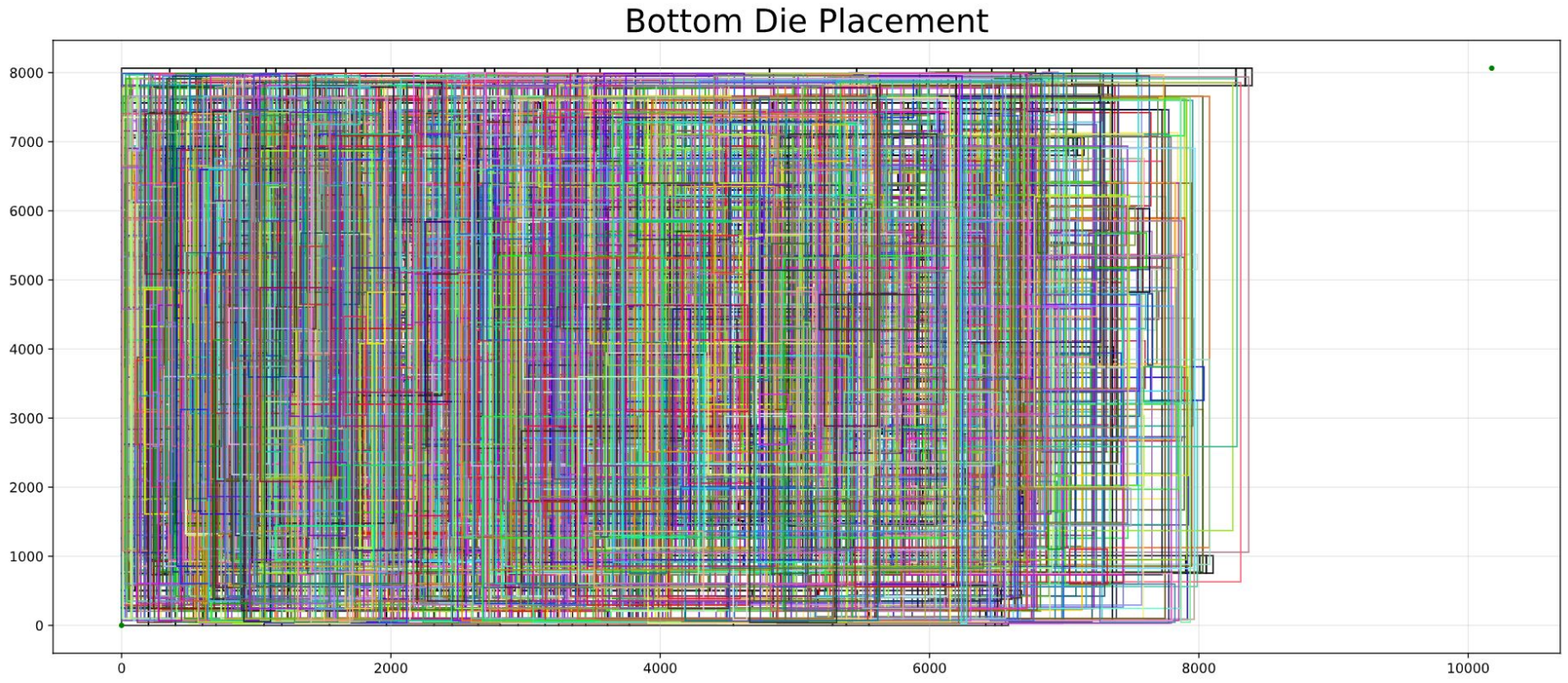
# Procedure - Simulate Annealing Visualization (Case 1)

< Cost converge to cost =98 >



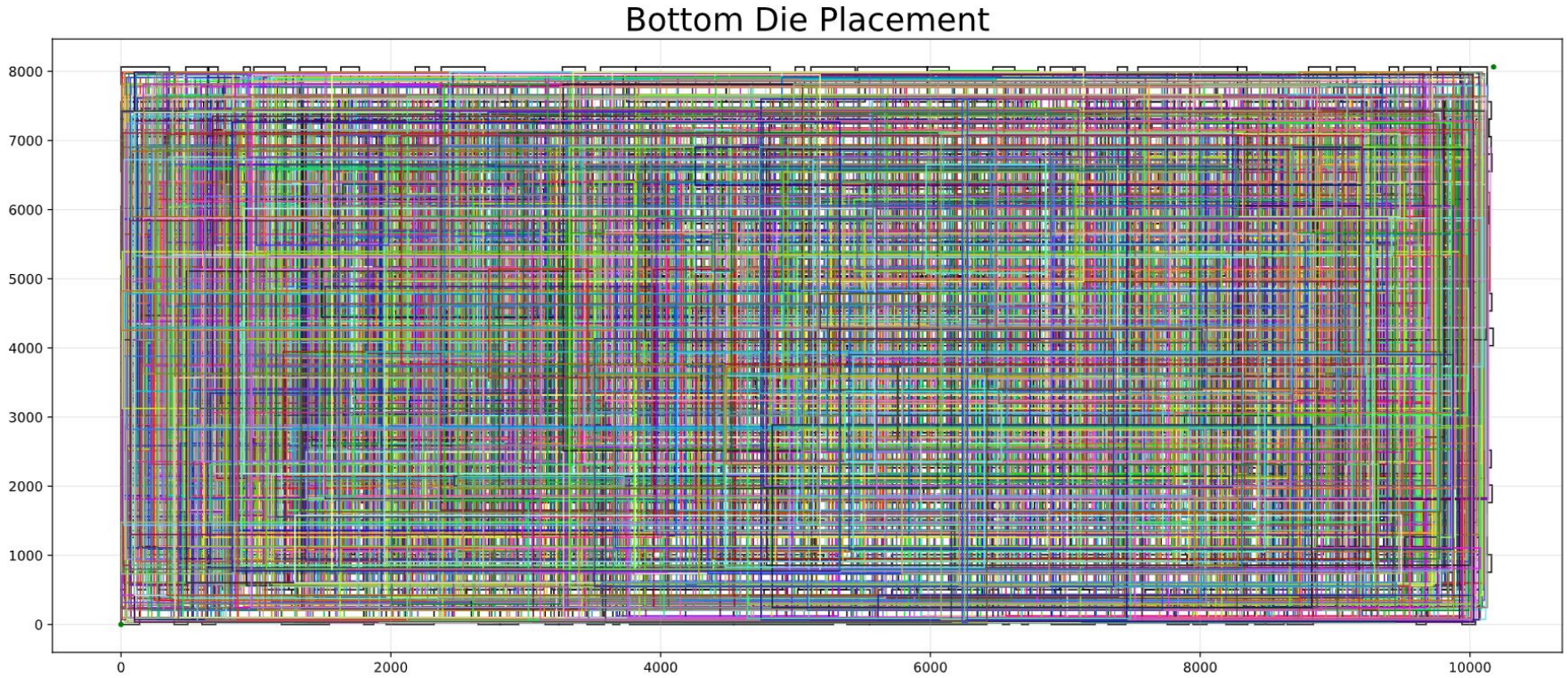
# Procedure - Simulate Annealing Visualization (Case 2)

<Before Simulate Annealing> Cost = 11602499



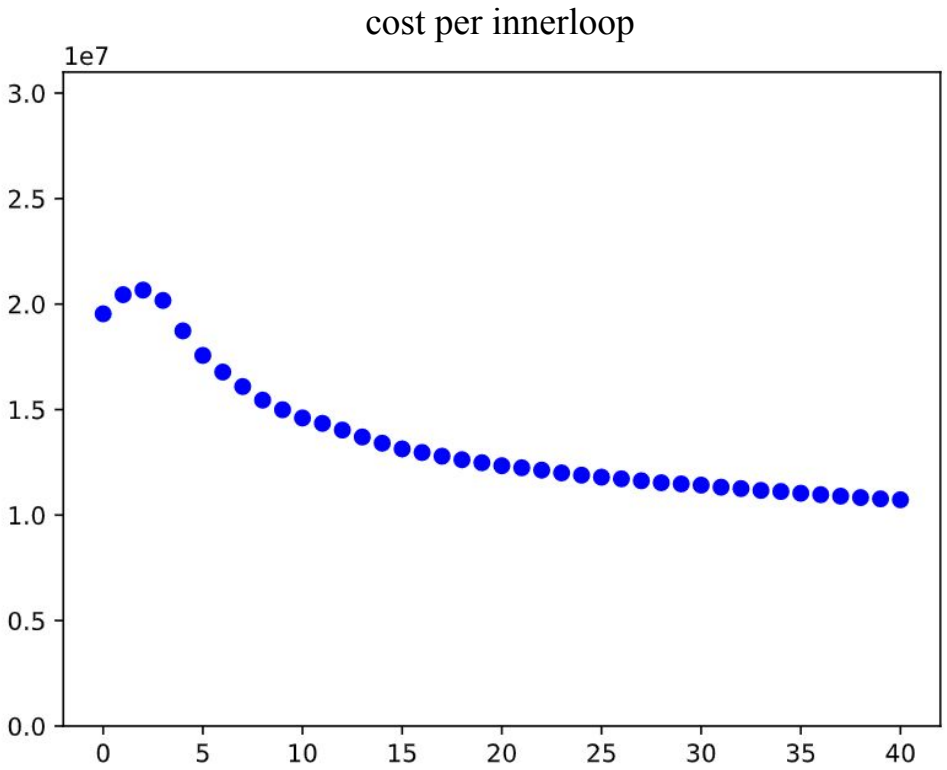
# Procedure - Simulate Annealing Visualization (Case 2)

<After Simulate Annealing>  Cost = 10745328



# Procedure - Simulate Annealing Visualization (Case 2)

< Cost converge to cost = 10745328 >



# Procedure - Hybrid Bonding Placement

**Constraints:**

Spacing between terminals and between boundry and terminal.

**Abstract:**

We get almost min cut at partition stage, therefore, place the hybrid bonding terminal is very easy for us. We store each maximum X/Y location and minimum X/Y location of each nets so that we put hybrid bonding terminals inside each bonding box without HPWL increasing.

# Procedure - Hybrid Bonding Placement Detail

```
for each net->bbox{
        if(hasHybridTerminal){
                //consider the spacing between the spacing
                update (x_min, y_min, x_max, y_max);

                //find a place for the terminal
                if(isValidPlacement(x_min, y_min, x_max, y_max, terminal->spacing)){
                        placeTerminal();
                }
        }
}
```

# Procedure - Placement Visualization

**Purpose:**

We decided to draw the placement instead of printing so much infomation on terminal so that it will be easier for us to debug. Furthermore, we also output the cost simulate annealing per termperature, so that we can monitor whether the result is converged or not.

# Procedure - Placement Visualization

**Methods:**

Using matplotlib.pyplot and matplotlib.patches to visualize the placement and bounding boxes result. Simply create a new file with all infos we need to draw the above picture in last few slides!

# Results and Verifier (Case 1)

Usage: ./Problem\ B_evaluator_0525 Benchmark/case1.txt case1.out

```
==========================================================================================
Start to check if the placment result is legal.
==========================================================================================
Check utilization for top die
Done
Check if inst is placed inside top die
Done
Check if inst overlaps with each other
Done
Check utilization for bottom die
Done
Check if inst is placed inside bottom die
Done
Check if inst overlaps with each other
Done
==========================================================================================
Summery
==========================================================================================
[TopDie]  Area(used/max): 620/900 Util: 68.89 Max Util: 80
[BottomDie] Area(used/max): 600/900 Util: 66.67 Max Util: 90
Total HPWL for this design is 104
```

# Results and Verifier (Case 2)

Usage: ./Problem\ B_evaluator_0525 Benchmark/case2.txt case2.out

```
==========================================================================================
Start to check if the placment result is legal.
==========================================================================================
Check utilization for top die
Done
Check if inst is placed inside top die
Done
Check if inst overlaps with each other
Done
Check utilization for bottom die
Done
Check if inst is placed inside bottom die
Done
Check if inst overlaps with each other
Done
==========================================================================================
Summery
==========================================================================================
[TopDie]  Area(used/max): 57619232/82936425 Util: 69.47 Max Util: 70
[BottomDie] Area(used/max): 52477488/82936425 Util: 63.27 Max Util: 75
Total HPWL for this design is 12079315
```

# Results and Verifier (Case 3)

Usage: ./Problem\ B_evaluator_0525 Benchmark/case3.txt case3.out

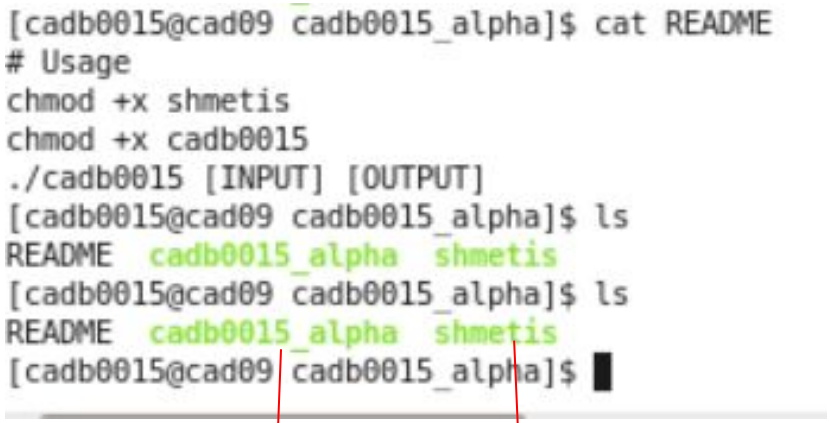
```
================================================================================
Start to check if the placment result is legal.
================================================================================
Check utilization for top die
Done
Check if inst is placed inside top die
Done
Check if inst overlaps with each other
Done
Check utilization for bottom die
Done
Check if inst is placed inside bottom die
Done
Check if inst overlaps with each other
Done
================================================================================
Summery
================================================================================
[TopDie]  Area(used/max): 284697105/369254080 Util: 77.10 Max Util: 78
[BottomDie] Area(used/max): 281537595/369254080 Util: 76.24 Max Util: 78
Total HPWL for this design is 403625580
```

# Results and Verifier (Case 4)

Usage: ./Problem\ B_evaluator_0525 Benchmark/case4.txt case4.out

```
================================================================================
Start to check if the placment result is legal.
================================================================================
Check utilization for top die
Done
Check if inst is placed inside top die
Done
Check if inst overlaps with each other
Done
Check utilization for bottom die
Done
Check if inst is placed inside bottom die
Done
Check if inst overlaps with each other
Done
================================================================================
Summery
================================================================================
[TopDie]  Area(used/max): 1835525856/2838171970 Util: 64.67 Max Util: 66
[BottomDie] Area(used/max): 1962024890/2838171970 Util: 69.13 Max Util: 70
Total HPWL for this design is 5104794660
```

# Alpha Test Submission




binary excute file

partition software: system call by binary excute file

# Alpha Test Submission - Issues

g++ version of vda04:

```
17:48 2022EDA048@vda04 [~] >$ g++ -v
Using built-in specs.
COLLECT_GCC=g++
COLLECT_LTO_WRAPPER=/uhome/SWDevelop/Compiler/gcc/7.3.1/usr/bin/../libexec/gcc/x86_64-redhat-linux/7/lto-wrapper
Target: x86_64-redhat-linux
Configured with: ../configure --enable-bootstrap --enable-languages=c,c++,fortran,lto --prefix=/opt/rh/devtoolset-7/root/usr --mandir=/opt/rh/devtoolset-7/root/usr/share/man --infodir=/opt/rh/devtoolset-7/
root/usr/share/info --with-bugurl=http://bugzilla.redhat.com/bugzilla --enable-shared --enable-threads=posix --enable-checking=release --enable-multilib --with-system-zlib --enable-__cxa_atexit --disable-l
ibunwind-exceptions --enable-gnu-unique-object --enable-linker-build-id --with-gcc-major-version-only --enable-plugin --with-linker-hash-style=gnu --enable-initfini-array --with-default-libstdcxx-abi=gcc4-
compatible --with-isl=/builddir/build/BUILD/gcc-7.3.1-20180303/obj-x86_64-redhat-linux/isl-install --enable-libmpx --enable-gnu-indirect-function --with-tune=generic --with-arch_32=i686 --build=x86_64-redh
at-linux
Thread model: posix
gcc version 7.3.1 20180303 (Red Hat 7.3.1-5) (GCC)
17:48 2022EDA048@vda04 [~] >$
```

Compiler version of ICCAD hosting server: g++ (Version:4.4.7)

There is a little bit difference in two version:

1. 2D vector announcement: vector <vector< myVector >> is illegal, need to change by vector <vector< myVector > > with additional space between two >.
2. Vector Operation: myVector.emplace_back() is more efficient than myVector.push_back(), however, emplace_back() can't be used in ICCAD hosting server.

# case1

```
[cadb0015@cad09 cadb0015_alpha]$ ./Problem\ B_evaluator_0525 case1.txt case1.out

================================================================================
Start to check if the placment result is legal.
================================================================================
Check utilization for top die
Done
Check if inst is placed inside top die
Done
Check if inst overlaps with each other
Done
Check utilization for bottom die
Done
Check if inst is placed inside bottom die
Done
Check if inst overlaps with each other
Done
================================================================================
Summery
================================================================================
[TopDie]  Area(used/max): 620/900 Util: 68.89 Max Util: 80
[BottomDie] Area(used/max): 600/900 Util: 66.67 Max Util: 90
Total HPWL for this design is 104
[cadb0015@cad09 cadb0015_alpha]$
```

case 2

```
[cadb0015@cad09 cadb0015_alpha]$ make do2
time ./cadb0015_alpha Benchmark/case2.txt case2.out
1942.10user 1589.34system 58:49.99elapsed 100%CPU (0avgtext+0avgdata 665156maxre
sident)k
0inputs+8952outputs (0major+1078615569minor)pagefaults 0swaps
chmod +x Problem\ B_evaluator_0525
./Problem\ B_evaluator_0525 Benchmark/case2.txt case2.out

================================================================================
====
Start to check if the placment result is legal.
================================================================================
====
Check utilization for top die
Done
Check if inst is placed inside top die
Done
Check if inst overlaps with each other
Done
Check utilization for bottom die
Done
Check if inst is placed inside bottom die
Done
Check if inst overlaps with each other
Done
================================================================================
====
Summery
================================================================================
====
[TopDie]  Area(used/max): 56746976/82936425 Util: 68.42 Max Util: 70
[BottomDie] Area(used/max): 54243252/82936425 Util: 65.40 Max Util: 75
Total HPWL for this design is 12213608
```

# case 3

```
[cadb0015@cad09 cadb0015_alpha]$ make do3
time ./cadb0015_alpha Benchmark/case3.txt case3.out
2559.85user 949.38system 58:27.59elapsed 100%CPU (0avgtext+0avgdata 3031480maxresident)k
0inputs+71256outputs (0major+448298579minor)pagefaults 0swaps
chmod +x Problem\ B_evaluator_0525
./Problem\ B_evaluator_0525 Benchmark/case3.txt case3.out

================================================================================
Start to check if the placment result is legal.
================================================================================
Check utilization for top die
Done
Check if inst is placed inside top die
Done
Check if inst overlaps with each other
Done
Check utilization for bottom die
Done
Check if inst is placed inside bottom die
Done
Check if inst overlaps with each other
Done
================================================================================
Summery
================================================================================
[TopDie]  Area(used/max): 285736245/369254080 Util: 77.38 Max Util: 78
[BottomDie] Area(used/max): 280498455/369254080 Util: 75.96 Max Util: 78
Total HPWL for this design is 403211338
```

# case 4

```
[cadb0015@cad09 cadb0015_alpha]$ make do4
time ./cadb0015_alpha Benchmark/case4.txt case4.out
2938.26user 587.35system 58:44.93elapsed 100%CPU (0avgtext+0avgdata 23229000maxr
esident)k
92664inputs+468520outputs (0major+258986342minor)pagefaults 0swaps
chmod +x Problem\ B_evaluator_0525
./Problem\ B_evaluator_0525 Benchmark/case4.txt case4.out

==========================================================================================
====
Start to check if the placment result is legal.
==========================================================================================
====
Check utilization for top die
Done
Check if inst is placed inside top die
Done
Check if inst overlaps with each other
Done
Check utilization for bottom die
Done
Check if inst is placed inside bottom die
Done
Check if inst overlaps with each other
Done
==========================================================================================
====
Summery
==========================================================================================
====
[TopDie]  Area(used/max): 1839733108/2838171970 Util: 64.82 Max Util: 66
[BottomDie] Area(used/max): 1955449765/2838171970 Util: 68.90 Max Util: 70
Total HPWL for this design is 5105957423
```

# Improvement

In our design, the critical time spends on finding a legal partition and a new place for instance in Simulate Annealing Process. However, these procedure have no ordering issues, which mean that can be accellerated by multi-core parallel programming without race condition issues.

We had tried using OpenMP to parallel these process, however, the result still not converge. Therefore, we are looking forward to using pthread or CUDA to perform more specific parallel programming skills on our process.



# Work Assignment

- Readfiles – 祁恩 婕恩
- Partition – 祁恩 婕恩
- Placement – 祁恩 婕恩
- Simulating Annealing – 祁恩 婕恩
- Visualization – 祁恩 婕恩
- Paper work– 祁恩 婕恩



We work all together and using git for version controll!

# FeedBack

非常感謝祁恩在短時間內教會我這麼多東西, 沒有他的話我根本不可能自己完成這堂課, 特別感謝他在最後這段時間剛好遇上我的畢業典禮, 謝謝他的諒解讓我可以好好跟交大說再見, 你是我在交大遇到最凱瑞的夥伴, 祝福你未來的日子一帆風順唷XD 很可惜最後沒能合照, 下次回去拿畢業證書再合照一張吧:))

–婕恩

這算是我第一次寫這麼大型的code, 對我來說算是非常吃力, 從一開始只會C code到最後能夠使用C++的STL以加速開發速度, 也算是學了不少。很難想像一年前還在材料系當程式小白的我, 現在已經能夠與他人合作寫出了一個考慮完善的code且架構分明。非常感謝交大校花學姊, 帶我進入電機系然後找我一起上這門課!祝你未來在台大的研究順利!

–祁恩