Andy, Yu-Guang Chen

EE6094 CAD for VLSI Design
Programming Assignment 2: Partitioning
(Due: 23:59:59, 2024/04/23)

Introduction

During the front-end design stage, designers synthesize the RTL design into a gate-level netlist before proceeding to the physical design stage. The initial phase of the physical design stage involves partitioning the entire gate-level netlist into multiple sub-blocks, allowing for placement according to floorplanning results. This process, known as Partitioning, aims to minimize the number of connections between sub-blocks by dividing the gate-level netlist appropriately.

Background

Physical design is a crucial step in the standard design cycle of integrated circuits, as depicted in Figure 1. Following circuit design, this stage involves converting circuit representations of components (devices and interconnects) into geometric shapes. These shapes, when manufactured in corresponding material layers, ensure the proper functioning of the components and collectively form the integrated circuit layout. The physical design process comprises various sub-steps encompassing both layout design and verification and validation.

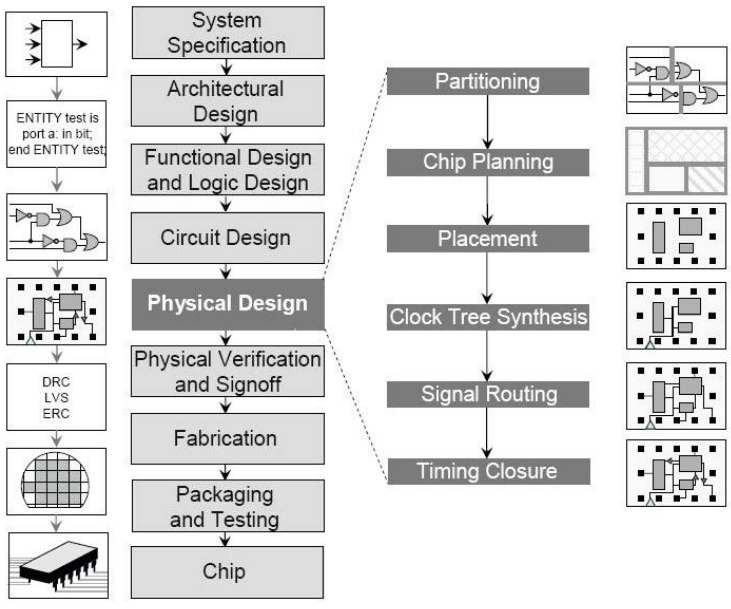


Figure1. VLSI Design Procedure

Circuit partitioning holds significant importance during the physical design phase. Its primary objective is to divide the circuit into segments where the component sizes fall within specified ranges, while minimizing the complexity of connections between them. A well-executed circuit partitioning can streamline subsequent procedures in physical design, leading to improved outcomes.

Problem Definition

Let $C = \{C_1, C_2, \ldots C_n\}$ be a set of cells and $N = \{N_1, N_2, \ldots, N_m\}$ be a set of nets. Each net N_i, $i = 1, 2, \ldots, m$, connects a subset of cells. The two-way min-cut partitioning problem is to partition the cell set into two groups *A* and *B*. The size of group *A*, *size*(*A*), is the number of cells in group *A*. *Size*(*B*) is the number of cells in group B. The cost of a two-way partitioning is measured by the cut size, which is the number of nets having cells in both groups. The problem is defined as follows:

Input: A net-list for a circuit.

Output: Partition the circuit to two sub-circuits *A* and *B* so that the cut size of sub-circuits *A* and *B* is minimized.

Constraints:

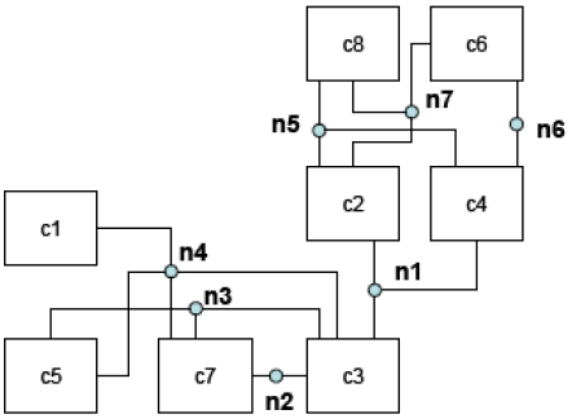
1. $|size(A) - size(B)| <= n/5$, where n is the number of cells in the circuit and n always > 5.
2. The runtime of your program is limited to at most 10 minutes per testcase.

Input file format

Input is a list of nets. Each net statement starts with the keyword "NET" and the name of the net. The cells that are connected by the net are listed between a pair of braces following the net name. The net may be provided in random sequence (ex: n1, n7, n2, n5 …).

Example:

```
NET n1 { c2 c3 c4 }
NET n2 { c3 c7 }
NET n3 { c3 c5 c7 }
NET n4 { c1 c3 c5 c7 }
NET n5 { c2 c4 c8 }
NET n6 { c4 c6 }
NET n7 { c2 c6 c8 }
```



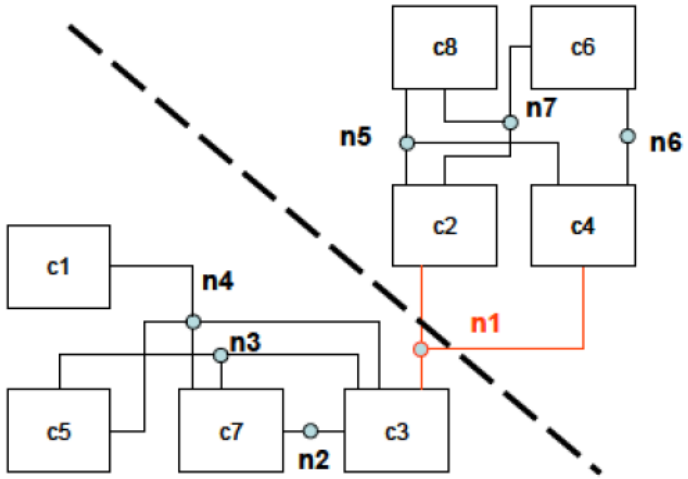
Output file format

Report the cells in each group and the cut-size. **Please follow the output format.**

```
cut_size #
A
cell_ID
…
B
cell_ID
…
…
```

Example:

cut_size 1
A
c1
c3
c5
c7
B
c2
c4
c6
c8



Algorithm

You are required to implement the Simulated Annealing (SA) Algorithm to solve the two-way min-cut partitioning problem. You have the flexibility to employ any method for structuring the initial solution. However, if you do not implement the SA algorithm, you will receive a penalty that will affect your correctness score.

Requirement

1. You have to write this program in C or C++. We will verify your program on a workstation (the same one for your PA1). No open source codes are allowed to use. (i.e., you MUST implement the tool by yourself).

2. All files should be submitted through ee-class. The files you need to submit are as follows:

   (1) A source code file named *StudID*_PA2.cpp

   (ex: 9862534_PA2.cpp)

   (2) A report named *StudID_Name*_PA2_report.pdf

   (ex: 9862534_陳聿廣_PA2_report.pdf)

   (3) A makefile

   Note that the only acceptable report file format is .pdf, no .doc/.docx or other files are acceptable. **BE SURE to follow the naming rule mentioned above. Otherwise, 10% penalty will be applied for the file(s) with incompatible name(s).** Note that **duplicating filename extension is considered as an incompatible name.**

3. The output file of your program should be named as *name***.out** where *name* is the input file name. You also need to write a makefile to compile and run your program. We will verify your program with the makefile you provided. Please see the "Makefile" section for more details.

4. We don't restrict the report format and length. In your report, you have to at least include:
   (1) How to compile and execute your program; (You can use screenshot to explain)
   (2) The completion of the assignment; (If you complete all requirements, just specify all)
   (3) The design of your initial solution, neighborhood structure and the cooling schedule;
   (4) The hardness of this assignment and how you overcome it;
   (5) Any suggestions about this programming assignment?

Makefile

Your makefile should at least contain these 3 commands, which is (1) **make all**, (2) **make run**, and (3) **make clean**. The descriptions of each command is shown below.

(1) **make all**: This command will automatically compile your source codes and generate the corresponding objects and executable file.
(2) **make run**: This command will execute your executable file and run your program.
(3) **make clean**: This command will automatically remove all the objects and executable file generated by make all.

Grading

The grading is as follows:

(1) Correctness of your solutions: 30%
(2) Quality of your solutions: 20%
(3) Readability of your code: 10%
(4) The report: 10%
(5) Demo session: 30%

Please submit your assignment on time. Otherwise, the penalty rule will apply:

- Within 24hrs delay: 20% off
- Within 48hrs delay: 40% off
- More than 48hrs: 0 point

Contact

For all questions about PA2, please send E-mail to TA 施奕瑄 (ruby68680@gmail.com)

Reference

[1] C. M. Fiduccia and R. M. Mattheyses, "A Linear-Time Heuristic for Improving Network Partitions," 19th Design Automation Conference, 1982, pp. 175-181, doi: 10.1109/DAC.1982.1585498.