FPGA Final Project Report

姓名: 葉財豪
學號: 112062638

作業為單人完成

(1) The HPWL of each testcase

	testcase1	testcase2	testcase3	testcase4
HPWL	12592.5	8658512.5	58651.5	56616247

大致上會在以上數字左右，跟當下執行環境的使用情況相關。

(2) How to Compile and Execute Code

編譯:

```
fpga-112062638@MakLab:~/Final$ make clean
rm -f ./legalizer Legalizer.o main.o FPGA.o DetailedPlacer.o ../bin/example
fpga-112062638@MakLab:~/Final$ pwd
/home/fpga23/fpga-112062638/Final
fpga-112062638@MakLab:~/Final$ make
g++ -std=c++11 -O3 -Wall -Wextra -I../include -c Legalizer.cpp -o Legalizer.o
g++ -std=c++11 -O3 -Wall -Wextra -I../include -c main.cpp -o main.o
g++ -std=c++11 -O3 -Wall -Wextra -I../include -c FPGA.cpp -o FPGA.o
g++ -std=c++11 -O3 -Wall -Wextra -I../include -c DetailedPlacer.cpp -o DetailedPlacer.o
g++ -o legalizer Legalizer.o main.o FPGA.o DetailedPlacer.o
fpga-112062638@MakLab:~/Final$
```

執行:

```
fpga-112062638@MakLab:~/Final$ make
g++ -std=c++11 -O3 -Wall -Wextra -I../include -c Legalizer.cpp -o Legalizer.o
g++ -std=c++11 -O3 -Wall -Wextra -I../include -c main.cpp -o main.o
g++ -std=c++11 -O3 -Wall -Wextra -I../include -c FPGA.cpp -o FPGA.o
g++ -std=c++11 -O3 -Wall -Wextra -I../include -c DetailedPlacer.cpp -o DetailedPlacer.o
g++ -o legalizer Legalizer.o main.o FPGA.o DetailedPlacer.o
fpga-112062638@MakLab:~/Final$ ./legalizer ./input/testcase1/architecture.txt ./input/testcase1/instance.txt
 ./input/testcase1/netlist.txt ./output/output1.txt
=======================================
Global placement HPWL: 15701
=======================================
```

參考 **README**，

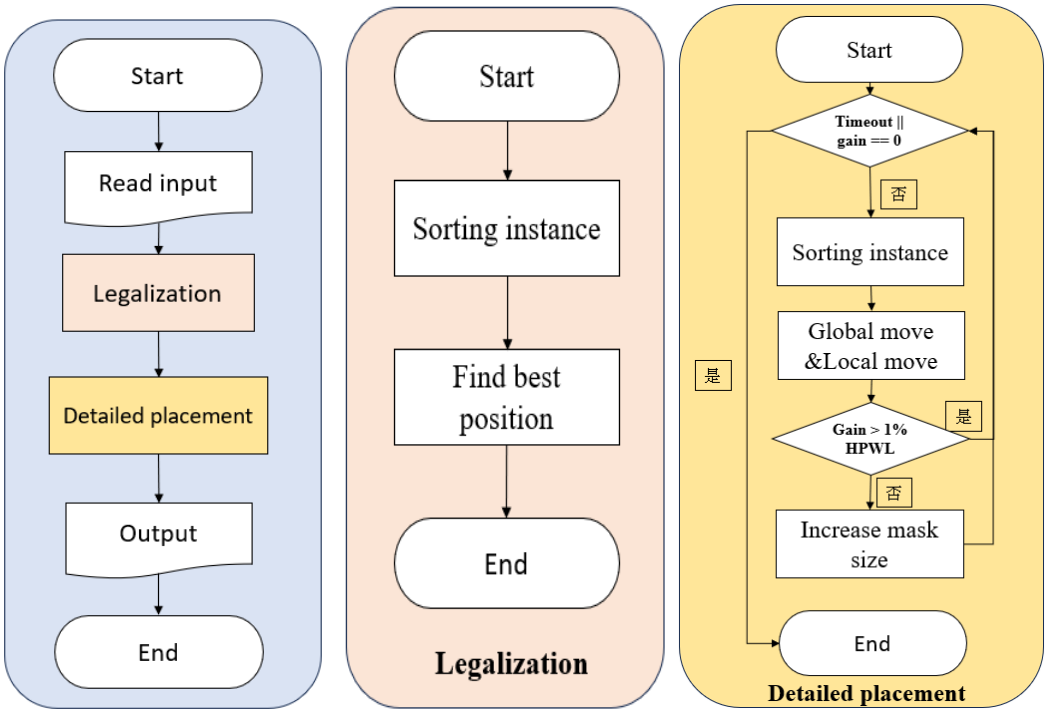
(3) Method:

1. **Legalization: (Tetris)**
   - 根據每個 **instance** 的 **global placement (GP)**的位置由左下到右上進行排序
   - 使用排序的順序對每個 **instance** 找到一個最靠近 **GP** 位置且 **available** 的 **resource**。會透過找目前最靠近 **instance** 的 **column**，找到這個 **column** 最靠近 **instance** 的 **available resource**，接下來再對鄰近的 **column** 找最靠近 **instance** 的 **available resource**，當某一個 **column** 與 **instance** 的水平距離大於 **instance** 與先前找到的最近距離時，這些 **column** 將不再需要計算。

2. **Detailed placement:**
   - **C(vi, pj) 代表 vi 這個 instance 在 vj 的 position 上時，所有與 vi 有相連的 net 的 HPWL 總和。**
     當兩個 instance 交換位置時:

$$C(v_i, p_i) + C(v_j, p_j) > C(v_i, p_j) + C(v_j, p_i)$$

     上述式子成立代表交換兩個 instance 能夠降低整體的 HPWL。
   - **透過對每個 instance 使用 C(vi, pi) 做排序由大到小，依序對 instance 進行 global mov 再進行 local move，當所有 instance 做完一遍 global move 與 local move，計算這輪 move 的總 WL 下降量，當下降量不足 1%時會增加 local move 時的 window size。程式會不斷進行兩種 move 直到 Timeout 或者整輪沒有任何 WL 下降。**
   - **Global move: 對於每個 instance，計算其 optimal region ([2])，嘗試與 optimal region 中的位置互換，如果可以降低 HPWL，則進行交換。**
   - **Local move [1]: 以 instance 為中心，在長寬都為 maskSize*2+1 的 window 當中，嘗試讓中心的 instance 與所有 window 當中的位置進行交換計算出交換能夠帶來的 WL 下降量，最後與 window 中有最大下降量的位置交換。**



Reference:

[1] Sheng-Yen Chen and Yao-Wen Chang, “Routing Architecture-Aware Analytical Placement for Heterogeneous FPGAs”, Proc. of DAC’2015.

[2] T.-H. Lin, P. Banerjee, and Y.-W. Chang, “An Efficient and Effective Analytical Placer for FPGAs,” Proc. of DAC, 2013.