CS 516000 FPGA Architecture & CAD Programming Assignment – Congestion Driven FPGA Placement

(Due: 2025/12/15 23:59)

---

1. Objective

In this assignment, you will implement a **congestion-driven placer** for an island-style FPGA.

Given:

- An FPGA consisting of a $R \times C$ grid of logic block sites.
- A circuit consisting a set of movable logic blocks.
- A set of pre-placed I/O pins of the circuit with predefined coordinates on the FPGA boundary.

Your program must:

1. Produce a **legal placement**.
2. Minimize **congestion**.
3. Minimize **HPWL**.

You are free to choose any algorithmic approach (partition-based, simulated annealing, analytical, hybrids, etc.) as long as you follow the required input/output format and rules.

---

2. Legal Placement

A placement is **legal** if and only if:

1. Every movable logic block appears **exactly once** in the output.
2. Each movable logic block of the circuit is assigned to a logic block site with integer coordinates (x, y) s.t. $0 \le x < C$, $0 \le y < R$.
3. No two movable logic blocks are assigned to the same logic block site.
4. The coordinates of all fixed I/O pins given in the input are preserved and must not be changed.

Any testcase with an illegal placement will receive **0** for that testcase.

---

3. Congestion Metric: Bounding-Box Overlap Congestion Coefficient (CC)

We use the bounding-box overlap congestion coefficient from:

"A Congestion Driven Placement Algorithm for FPGA Synthesis," FPL 2006.

3.1 Bounding Box & HPWL Definition

Given an R × C grid of sites, each movable logic block occupies exactly one site. Placing a logic block at (x, y) means it occupies the site whose lower-left corner is at (x, y).

We assume the I/O pins of the circuit have been pre-placed, which are modeled as points with given coordinates (x, y).

For each net *e*, we collect the coordinates of all its terminals:

- **Movable blocks**: each block b is located at integer site coordinates (x_b, y_b), which represent the lower-left corner of the site it occupies.
- **Fixed I/O pins**: each I/O pin p is a point located at (x_p, y_p).

Let

- x_min = $min_{b,p \in e}$ (x_b, x_p)
- x_max = $max_{b,p \in e}$ (x_b + 1, x_p)
- y_min = $min_{b,p \in e}$ (y_b, y_p)
- y_max = $max_{b,p \in e}$ (y_b + 1, y_p)

Then, (x_min, y_min) is the lower-left corner of the bounding box, and (x_max, y_max) is the upper-right corner.

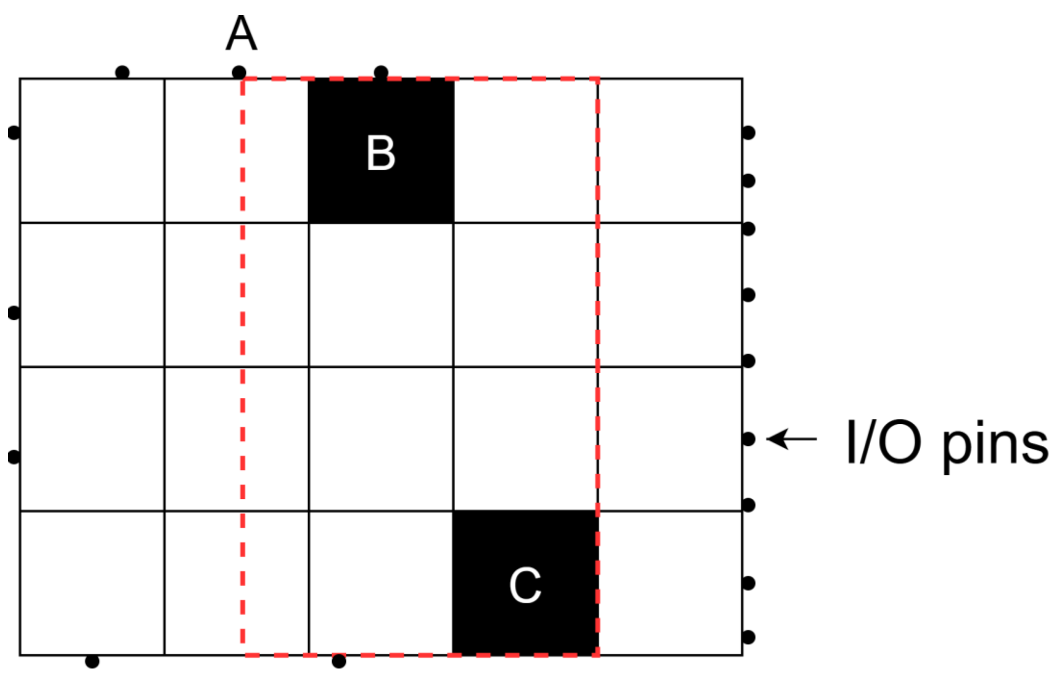
HPWL(n) = (x_max – x_min) + (y_max – y_min)

Example (Bounding Box & HPWL)

In the example below, pin A(on the top boundary), and blocks B and C all belong to the same net. Their placement coordinates are:

- A = (1.5, 4)
- B = (2, 3)
- C = (3, 0)

The red dashed rectangle is the net’s bounding box with **HPWL = (4−1.5) + (4−0) = 6.5**



3.2 Compute coverage U[x,y]

Initialize U[x,y] = 0 for all sites.

For each net n:

```
1. Find its bounding box (x_min, x_max, y_min, y_max) as defined in Section 3.1.
2. For all integers x and y satisfying
     o  x_min ≤ x < x_max
     o  y_min ≤ y < y_max
          i.  U[x, y] += 1.
```

Thus, U[x,y] is the number of nets whose bounding boxes cover the site (x, y).

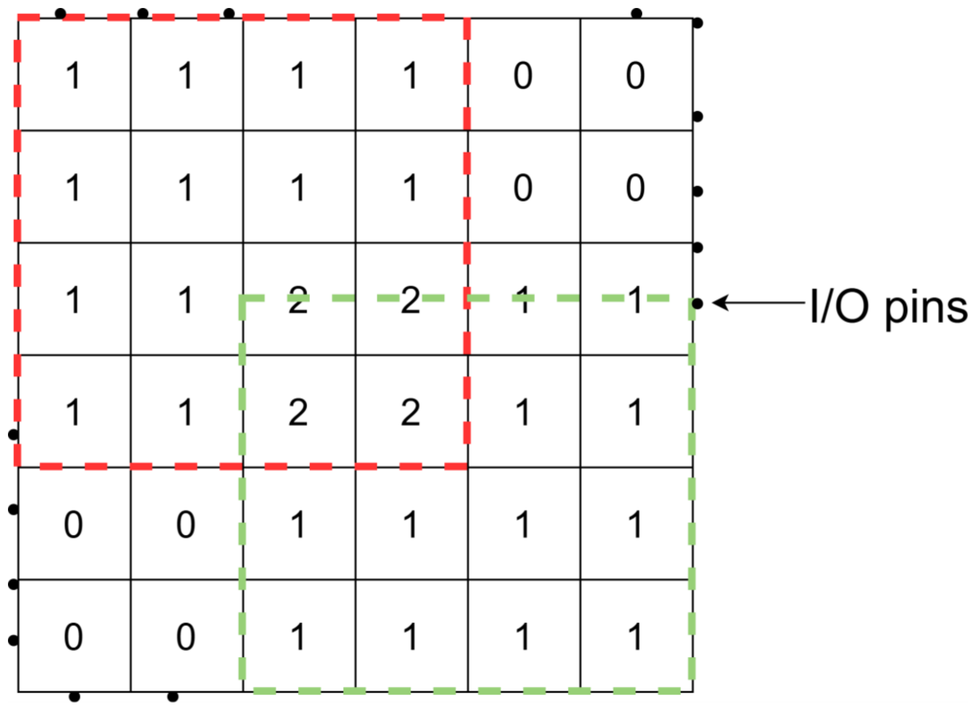
Example (Coverage U[x, y])

In the figure below, two nets are shown:

- The red dashed box is the bounding box of Net 1.
- The green dashed box is the bounding box of Net 2.

The grid spans from **(0, 0)** at the lower-left corner to **(5, 5)** at the upper-right corner.

Each grid cell contains a number indicating how many nets' bounding boxes cover that site. Thus, U[x, y] is the number of nets whose bounding boxes include site (x, y).



3.3 Compute Congestion Coefficient(CC)

$$CC = \frac{\frac{1}{N}\sum_{x=1}^{R}\sum_{y=1}^{C} U[x,y]^2}{\left(\frac{1}{N}\sum_{x=1}^{R}\sum_{y=1}^{C} U[x,y]\right)^2}$$

Properties:

- CC ≥ 1.
- CC = 1 when all U[x, y] are equal (perfectly uniform demand).
- Larger CC indicates more uneven and congested demand.

---

4. Input Format

Each testcase is a plain text file.

1. First line:

   R C NUM_LOGIC_BLOCKS NUM_IO_PINS NUM_NETS where R and C are integers denoting the numbers of rows and columns of logic block sites, NUM_LOGIC_BLOCKS, NUM_IO_PINS, and NUM_NETS are integers representing the number of movable logic blocks, number of I/O pins, and number of nets in the circuit.

2. Next NUM_LOGIC_BLOCKS lines:

   <block_name>

   where <block_name> is the name of a movable logic block.
3. Next NUM_IO_PINS lines:

   <pin_name> <x> <y>

   where pin_name is the name of a fixed I/O pin, and x, y are coordinates (can be non-integers) of this I/O pin on the FPGA boundary.
4. Next NUM_NETS lines (one per net):

   <net_name> <degree> <terminal_1> <terminal_2> ...

   where net_name is the name of the net, degree is an integer representing the number of terminals connected to this net, and each terminal_i is either a movable logic block name or an I/O pin name.

Constraints:

- NUM_LOGIC_BLOCKS $\leq$ R * C.
- degree $\geq$ 2.
- NUM_LOGIC_BLOCKS $\leq$ 20000

---

5. Output Format

Output a legal placement as plain text, one **movable logic block** per line:

<block_name> <x> <y>

where block_name is the name of a movable logic block,
x and y are integers representing the coordinates of the block.

Requirements:

- Each movable logic block appears exactly once.
- $0 \leq x < C$, $0 \leq y < R$.
- No two movable logic blocks share the same (x, y) coordinates.
- Fixed I/O pins must **not** be listed in the output; their coordinates are given in the input and must remain unchanged.
- Order of lines is arbitrary.

---

6. Submission

Submit a single compressed archive named:

<student_id>.zip

The zip file must contain:

- **src/** directory with all source codes
- **Makefile** (must compile by typing make)
- **report.pdf** describing your approach

Structure example:

```
<student_id>.zip
├── src/
│   ├── *.c
│   ├── *.cpp
│   └── *.h
├── Makefile
└── report.pdf
```

Requirements:

- The archive must extract cleanly on Linux.
- The source code must be written in C/C++ and compiled on a Linux platform.
- The program must compile using make without modifying any file.
- Any missing component (Makefile, report, source files) may result in score deduction. Submissions that do not follow the required format (wrong folder structure, wrong zip name, missing directories, etc.) will receive a **5-point** deduction.
- The executable name must be **placer**.
- The program must run as:
  ./placer <input_file> <output_file>

Runtime & Memory Limit

- Each testcase will be executed on a Linux environment.
- The **runtime limit** per testcase is **240 seconds**.
- The **memory limit** per testcase is **8 GB**.
- Programs that crash, exceed runtime or memory limits, or output invalid files will receive **0 points** for that testcase.

---

7. Grading Policy

- The completeness of your program and report (20%)
- The solution quality, including hidden testcases (80%)

Per-Testcase Quality Scoring

$$F_1 = \frac{CC - CC_{\min}}{CC_{\max} - CC_{\min}}$$

$$F_2 = \frac{HPWL - HPWL_{\min}}{HPWL_{\max} - HPWL_{\min}}$$

$$\text{Score} = 100 - 15 \times F_1 - 15 \times F_2$$

- F1 : normalized congestion
- F2 : normalized wirelength
- Best (both minimal) → 100 points; worst in both → 70 points
- Intermediate solutions are linearly scaled between 70–100.

---

8. Rules

- You may implement any algorithmic strategy (partition-based, SA, hybrid, etc.).
- You may consult and implement ideas from published work, but you **must**:
  - Write your own code.
  - Cite all references in your report.
- You may not:
  - Call external FPGA placement tools (e.g., VPR) as part of your solution.
  - Hard-code placements for specific testcases.
  - Share code between groups or copy code from others.
- **Any plagiarism will result in a 0 grade for the project.**

Reference
"A Congestion Driven Placement Algorithm for FPGA Synthesis", In Proceedings of the International Conference on Field Programmable Logic and Applications (FPL 2006)