Report of Final Project CS 516000 FPGA Architecture & CAD

11th Jan. 2022

I. How to compile

```
$ cd src
$ make clean
$ make
$ cd ../bin
```

II. How to run

```
$ ./project ../benchmarks/case2.arch ../benchmarks/case2.module ./benchmarks/
case2.net ../outputs/case2.floorplan
```

III. Introduction to my working progress

TRIAL1 (FAILED)

In the beginning I was trying out to implement the program using Simulated Annealing with B*-tree structure. I constructed the IRL of all modules WITHOUT dimensional constraints such as height of multiples of 3 and I also eliminated redundant realizations.

Three kinds of perturbation are used,

Operation 1: Randomly change chosen realization
Operation 2: Randomly swap 2 nodes (=modules) in the B*-tree structure
Operation 3: Randomly move a node to some random destination

The objective function is a sum of all normalized terms:

1. HPWL
2. Area (= floorplan_width * floorplan_height)
3. Aspect Ratio
4. Penalties of exceeding the fixed-outline on both x and y directions

The program was completed on Jan 10th after a huge struggle. And the outcome was awful as the fixed-outline constraint can never be satisfied no matter how hard I modified the program.
Therefore, under the kind suggestion from the TA, I decided to use a simpler way to

place the modules as described in **TRIAL2**.

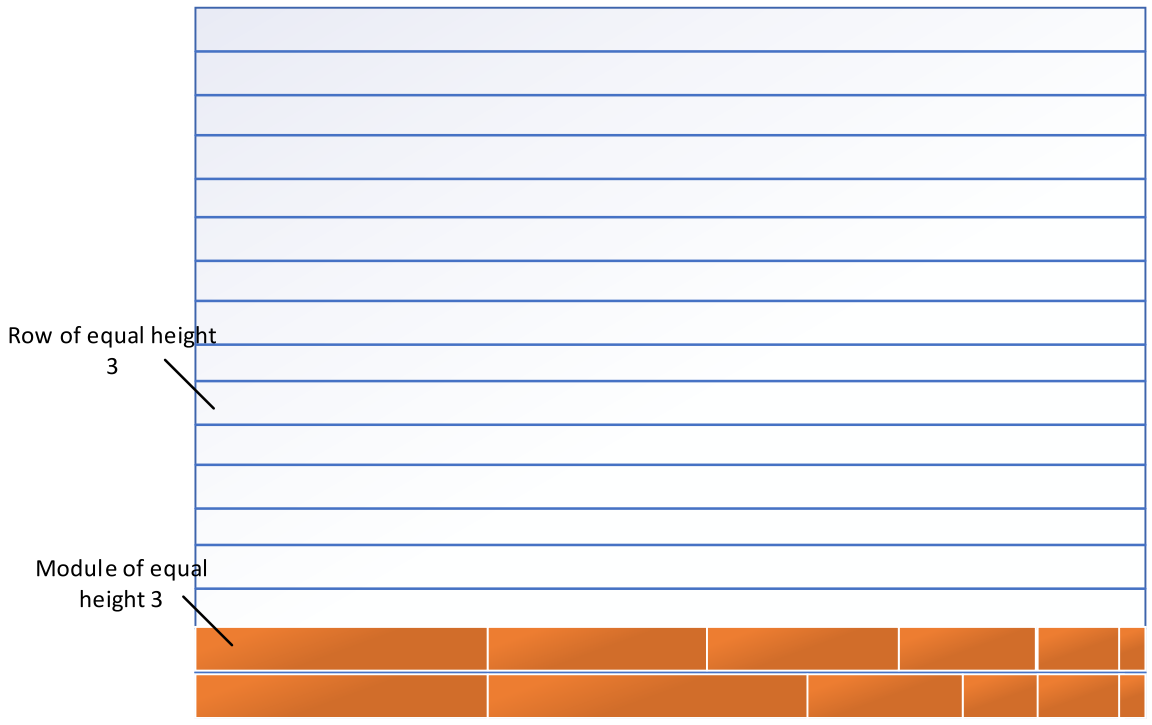
TRIAL2 (SUCEEDED)

This algorithm is quite heuristic and simple. The main difference in TRIAL2 is that the each realization has only height of 3, indicating only 1 element in each IRL at each position on x axis. The idea of HPWL minimization is discarded in TRIAL2. So, each module has only one representation at each position. I calculated the "average_width" of each module by considering all its possible widths at different position on x axis.

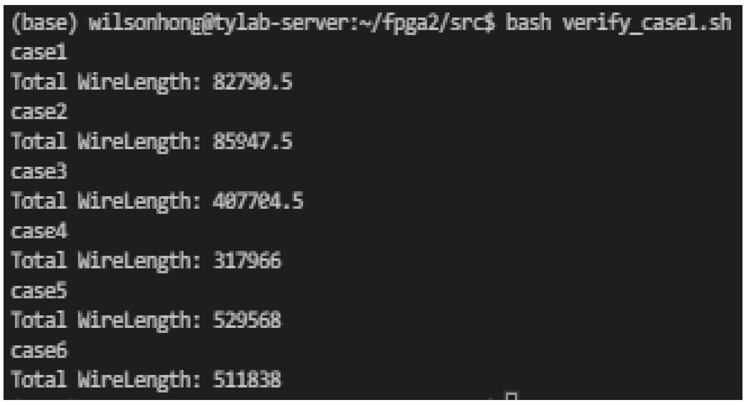
The whole floorplan is split into rows of height 3. Modules are then sorted by average_width and modules with larger average_width are placed first from left to right and from bottom to top. If a module goes over the right outline of floorplan, we go on to try the next module with smaller average_width. If a row is full or cannot accommodate any modules, we go to the next empty row.

In this way, I was able to generate valid solutions for all cases as verified by the provided verifier (on the right).

My placement method is as illustrated below.



IV. Results



```
(base) wilsonhong@tylab-server:~/fpga2/src$ bash verify_case1.sh
case1
Total WireLength: 82790.5
case2
Total WireLength: 85947.5
case3
Total WireLength: 407704.5
case4
Total WireLength: 317966
case5
Total WireLength: 529568
case6
Total WireLength: 511838
```

	Case1	Case2	Case3	Case4	Case5	Case6
HPWL	82790.5	85947.5	407704.5	317966	529568	511838