National Yang Ming Chiao Tung University
前瞻半導體研究所 / 電子研究所
Advanced EDA Topics, Spring 2024

April 16, 2024
Instructor: Mark P.-H. Lin
TAs: Jing-Yao Weng

Programming Assignment #2

Clock Power Optimization with Multi-Bit Flip-Flops

Due: May 5

1 Problem Description

See appendix while ignoring the placement density constraint.

2 Input

See appendix while ignoring the placement density constraint.

3 Output

See appendix while ignoring the placement density constraint.

4 Language/Platform

1. Language: C or C++ is preferred.
2. Platform: Linux.

5 Submission

You need to submit the following in a "tar" file to E3 (`https://e3.nycu.edu.tw/`) by the deadline. Please put all required files in a folder: (1) source codes, (2) Makefile, (3) a text readme file (readme.txt) stating how to build and use your program. The folder name

must be your student ID. Be sure to compress the folder in the linux environment with the following command.

tar cvf Student_ID.tar Student_ID

6 Grading Policy

This programming assignment will be graded based on (1) the **correctness**, (2) **solution quality**, and (3) **running time**. For each case, the runtime limit is **1 hours**. It will be regarded as "failed" for the case if it takes more than 1 hours.

There will be 20% penalty per day for late submission.

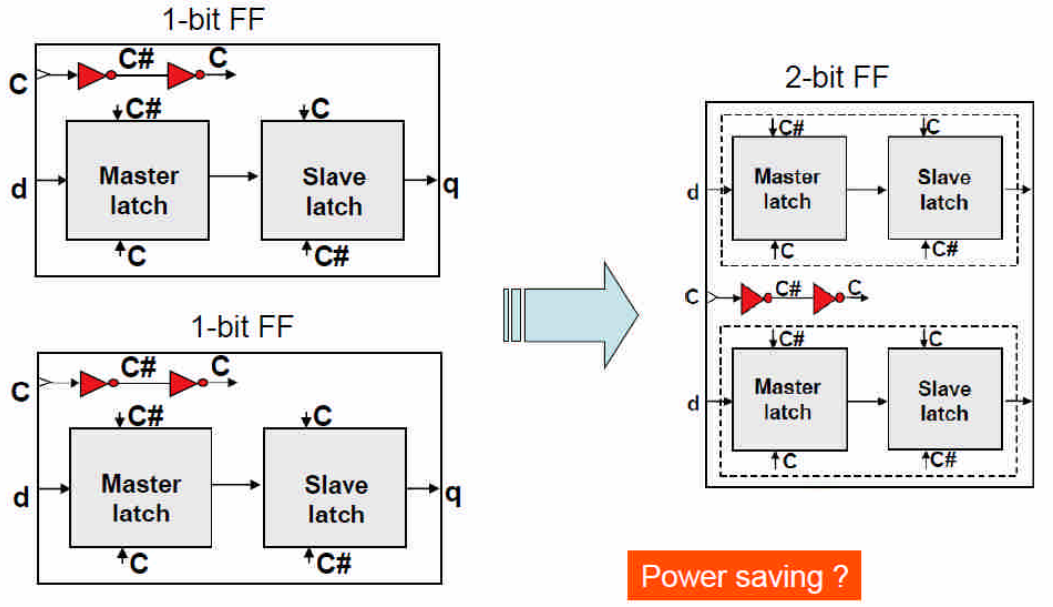
Appendix

Using Multi-Bit Flip-Flops for Clock Power Saving

Introduction

Multi-bit flip-flop is an effective power-saving implementation methodology by merging single-bit flip-flops in the design. Using multi-bit flip-flops can reduce clock dynamic power and the total flip-flop area effectively.

Figure 1 shows an example of merging two 1-bit flip-flops into one 2-bit flip-flop. Each 1-bit flip-flop contains two inverters, master-latch and slave-latch. Due to the manufacturing ground rules, inverters in flip-flops tend to be oversized. As the process technology advances into smaller geometry nodes like 65nm and beyond, the minimum size of clock drivers can drive more than one flip-flop. Merging 1-bit flip-flops into one multi-bit flip-flop can avoid duplicate inverters, and lower the total clock dynamic power consumption. The total area contributing to flip-flops can be reduced as well.



Problem Descriptions

Multi-bit flip-flops are capable of decreasing the power consumption because they can lower total inverter number by sharing the inverters in the flip-flops. Meanwhile, the total area is reduced. To acquire these benefits, the users must ensure that the design meets certain constraints after changes. One is ***timing slack constraint***. Figure 2 shows an example for the timing slack constraint problem. When a multi-bit flip-flop is used to replace multiple 1-bit flip-flops, the new multi-bit flip-flop may incur additional routing length due to the change of the location, and therefore worsens the timing slack. Timing slack constraint enforces that this replacement should not make the new slack value less than zero.

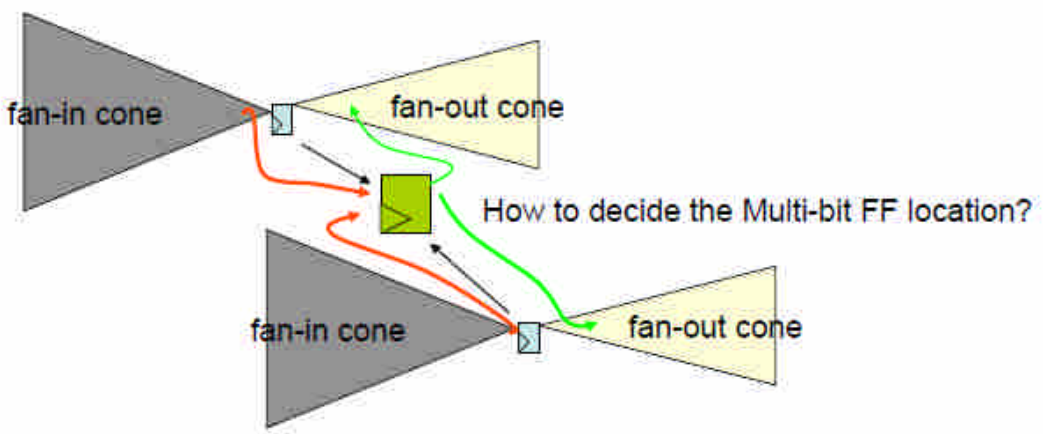


Figure 2: New flip-flop location incurs additional routing length

The other constraint is the ***placement density constraint***. Figure 3 shows an example for the placement density constraint problem. Because the multi-bit flip-flop is inserted in the new location with the larger area, we must consider the congestion change and make sure our design does not violate the placement density constraint.
In order to meet both the timing slack constraint and placement density constraint, we can only place multi-bit flip-flops in certain locations.

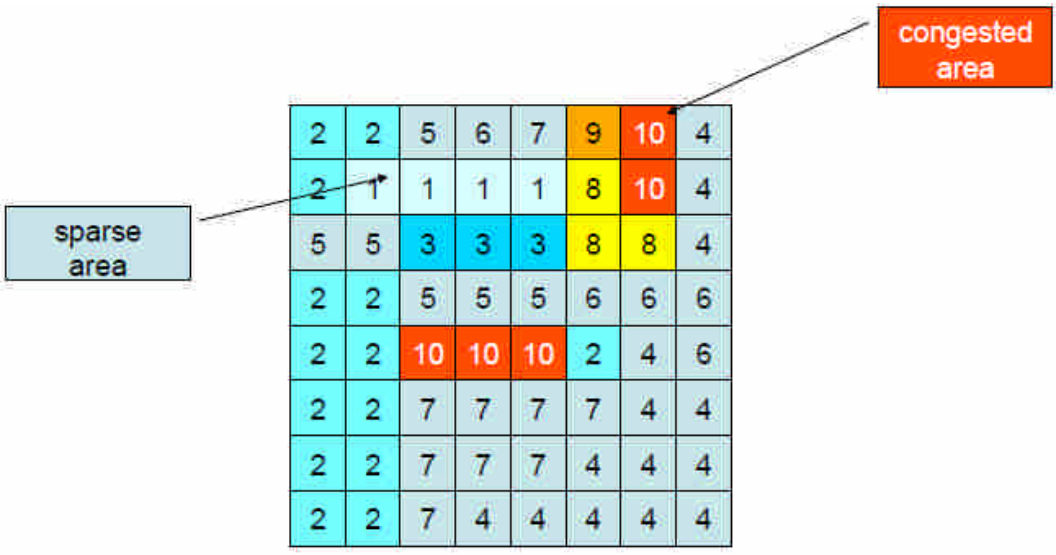


Figure 3: Congested area (>9) is not allowed for new flip-flop insertion.

Given a set of multi-bit flip-flops, our problem is to determine 1-bit flip-flops to be merged into corresponding multi-bit flip-flops in the new locations and meanwhile to meet the timing slack constraint and placement density constraint. The total power consumption of flip-flops is calculated according to the following equation:

$$FFPS(\text{Design}) = \sum_{i \in Flip-Flops\_in\_Design} (power(FF(i)))$$

The objective of this problem is to minimize the above function for power consumption where $FF(i)$ can be one 1-bit flip-flop or multi-bit flip-flop. $power(FF(i))$ denotes the power consumption of $FF(i)$. Different types of flip-flops can have different area and power consumption. Figure 4 shows an example. Please note that the real power consumption and area can be different for designs and depend on the flip-flop library in use.

	1-bit FF	2-bit FF	4-bit FF	8-bit FF
Normalized Area	1	1.92	2.85	7.52
Normalized AC Power (per bit)	1	0.86	0.78	0.75

Figure 4: Area and power consumption of multi-bit flip-flops

The power-consumption ratio of “after merging 1-bit FFs” to “before merging 1-bit

FFs" can be shown as:

$$\text{Objective function} = \frac{\text{FFPS(after merging 1-bit FFs)}}{\text{FFPS(before merging 1-bit FFs)}}$$

The objective of this problem is to minimize the above function and meet the timing slack constraint and placement density constraint at the same time.

Problem Assumptions

- The chip is a cell-based design, and hard macro blocks are included.
- The origin of the coordinates on the chip is set to the bottom left corner of the chip, i.e. (*0,0*) as shown in Figure 5.
- Cell location (*x,y*) is relative to the origin of the coordinates on the chip. See Figure 5.
- In order to simplify this problem, the sizes of all cell pins are assumed to zero, and all pins are connected to the cell's bottom left corner.
- One flip-flop (or block) belongs to the placement bin which contains the flip-flop's (or block's) bottom left corner.
- If the flip-flop's (or block's) bottom left corner locates on the placement boundary, such flip-flop belongs to the most right-top placement bin. See Figure 6.
- The placement density is defined as the total flip-flop area + total block area in one placement bin.
- In order to simplify this problem, the flip-flops and blocks can overlap with each other.
- Each grid can only allocate one cell (either one flip-flop or one pin). The bottom left corner of one flip-flop can not overlap with that of one cell pin.

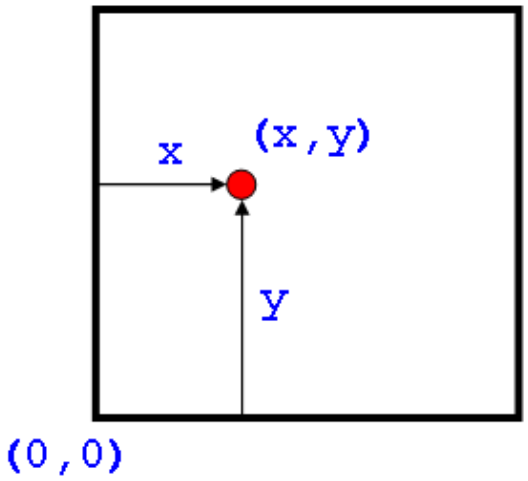


Figure 5: The coordinates on the chip

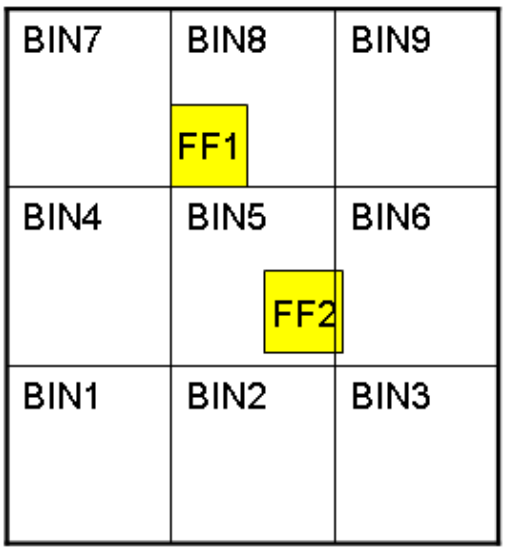


Figure 6: The placement bins: FF1 belongs to BIN8 and FF2 belongs to BIN5

Input/Output Files and Formats

Input parameters:

1. Chip size (WxH).
2. Bin sizes (BinWxBinH) for the placement density constraint calculation
3. Placement density constraint

Input file format:

There are three sections in each input file. The first part specifies the program parameters, including the chip size, grid size (basic resolution unit) and placement density constraint. The second part is the library section that specifies the basic properties of multi-bit flip flops. The third part, called the design section, defines the location and name of a flip-flop and the input/output ports which are connected to the flip-flop in the design.

Syntax:

#

'#' is the leading character of a comment which is ended by a newline.

CHIP_SIZE: W x H

This line specifies the size of the chip, where ***W*** is the width of the chip and ***H*** is the height of the chip.

GRID_SIZE: x y

This line specifies the grid size, where ***x*** is the distance between two consecutive x-grid lines and ***y*** is the distance between two consecutive y-grid lines. Grid is the basic resolution unit. Please note that cells must be placed on grid.

BIN_SIZE: BinW x BinH

This line specifies the bin size where ***BinW*** and ***BinH*** are the width and height of one bin, respectively.

PLACEMENT_DENSITY_CONSTRAINT: ConstraintValue

This line specifies the placement density constraint where the sum of all flip-flops area in a placement bin cannot be larger than ***ConstraintValue***.

```
[LIBRARY]
    [FLIP_FLOP_PROPERTY]
        [FLIP_FLOP FF_name1]
            BIT_NUMBER n1
            POWER_CONSUMPTION p1
            AREA a1
        [END FLIP_FLOP]
        [FLIP_FLOP FF_name2]
            BIT_NUMBER n2
            POWER_CONSUMPTION p2
            AREA a2
        [END FLIP_FLOP]
                …
    [END FLIP_FLOP_PROPERTY]
[END LIBRARY]
[DESIGN]
    [FLIP_FLOP_LIST]
        FF_name FLIP_FLOP_NAME1 (x1, y1)
        FF_name FLIP_FLOP_NAME2 (x2, y2)
        …
    [END FLIP_FLOP_LIST]
    [PIN_LIST]
        INPUT PIN_NAME1 (x1, y1)
        OUTPUT PIN_NAME2 (x2, y2)
```

…

[END PIN_LIST]

[NET_LIST]

PIN_NAME1 FLIP_FLOP_NAME1 SlackValue1

PIN_NAME2 FLIP_FLOP_NAME1 SlackValue2

…

[END NET_LIST]

[BLOCK_LIST]

BLOCK_NAME1 (x1, y1) AreaValue1

BLOCK_NAME2 (x2, y2) AreaValue2

…

[END BLOCK_LIST]

[END DESIGN]

where

[LIBRARY] :

Start of library section.

[FLIP_FLOP_PROPERTY] :

Start of flip-flop properties section.

[FLIP_FLOP FF_name1]:

Specifies the properties of a flip-flop.

FF_name1 is the flip-flop name.

BIT_NUMBER n1:

Specifies the bit number of this flip-flop as ***n1***.

POWER_CONSUMPTION p1:

Specifies the power consumption of this flip-flop as ***p1***.

AREA a1:

Specifies the area of this flip-flop as ***a1***.

[END FLIP_FLOP]:

End of a flip-flop properties section.

[END FLIP_FLOP_PROPERTY]:

End of flip-flop properties section.

[END LIBRARY]:

End of library section.

[DESIGN]:

Start of design section.

[FLIP_FLOP_LIST]:

Start of flip-flop list which is in the design section.

FLIP_FLOP_NAME1 (x1, y1):

Specifies this flip-flop. ***(x1, y1)*** is the coordinates of this flip-flop's bottom left corner.

[END FLIP_FLOP_LIST]

End of flip-flop list section.

[PIN_LIST]

Start of pin list which is in the design section.

PIN_NAME1 (x1, y1)

Specifies this pin. ***(x1, y1)*** is the coordinates of this pin.

[NET_LIST]

Start of net list which is in the design section.

PIN_NAME1 FLIP_FLOP_NAME1 SlackValue1

Specifies this net which is connected between ***PIN_NAME1*** and ***FLIP_FLOP_NAME1***. ***SlackValue1*** is this path slack.

[END NET_LIST]

End of net list section.

[BLOCK_LIST]

Start of block list which is in the design section.

BLOCK_NAME1 (x1, y1) AreaValue1

Specifies this block name. ***(x1, y1)*** is the coordinates of this block's bottom left corner. ***AreaValue1*** is the area of this block

[END BLOCK_LIST]

End of net list section.

[END PIN_LIST]

End of pin list which is in the design section.

[END DESIGN]

End of design section.

An example illustrating the input file format is shown as follows:

```
# comment is started by '#' chars.
# Specify the size of the chip
CHIP_SIZE    3000 x 3000
#specify the Bin size
GRID_SIZE    5 x 5
#specify the Bin size
BIN_SIZE    600 x 600
PLACEMENT_DENSITY_CONSTRAINT 9000
```

```
[LIBRARY]
    [FLIP_FLOP_PROPERTY]
        [FLIP_FLOP FF1]
            BIT_NUMBER 1
            POWER_CONSUMPTION 100
            AREA 100
        [END FLIP_FLOP]
        [FLIP_FLOP FF2]
            BIT_NUMBER 2
            POWER_CONSUMPTION 172
            AREA 192
        [END FLIP_FLOP]
        [FLIP_FLOP FF4]
            BIT_NUMBER 4
            POWER_CONSUMPTION 312
            AREA 285
        [END FLIP_FLOP]
    [END FLIP_FLOP_PROPERTY]
[END LIBRARY]
[DESIGN]
    [FLIP_FLOP_LIST]
        FF1 FLIP_FLOP1 (680, 1230)
        FF1 FLIP_FLOP2 (580, 650)
        FF1 FLIP_FLOP3 (2300, 2450)
     [END FLIP_FLOP_LIST]
    [PIN_LIST]
        INPUT PIN1 (570, 1250)
        OUTPUT PIN2 (1210, 1200)
        INPUT PIN3 (650, 800)
        OUTPUT PIN4 (1000, 630)
        INPUT PIN5 (2000, 2000)
        OUTPUT PIN6 (2700,2820)
    [END PIN_LIST]
    [NET_LIST]
      PIN1 FLIP_FLOP1 800
      PIN2 FLIP_FLOP1 700
      PIN3 FLIP_FLOP2 700
      PIN4 FLIP_FLOP2 550
```

PIN5 FLIP_FLOP3 600

PIN6 FLIP_FLOP3 500

[END NET_LIST]

[BLOCK_LIST]

BLOCK1 (1000, 900) 100

[END BLOCK_LIST]

[END DESIGN]

In this example, the size of this chip is 3000 x 3000, X-Y grid size is 5 x 5. The placement bin size is 600 x 600. The placement density constraint is set 9000. Three types of flip-flops, FF1, FF2 and FF4 are used according to the library. There are total three input buffers, three output buffers, one block and three flip-flops in this design. [NET_LIST] section lists the connections from pins to flip-flops followed by the slack values. For example, PIN1 is connected to FLIP_FLOP1 and its slack value is 800. To simplify the problem, the slack value can be viewed as the X-Y distance for trade-off. For example, the distance between PIN1 and FLIP_FLOP1 is (680-570) + (1250-1230) = 130, and the slack is 800. It means that the ***maximum*** distance (or total slack) from PIN1 to FLIP_FLOP1 can be 130+800=930. The final maximum placement density is 192+100=292 in the placement bin contained *NEW_FLIP_FLOP* in Figure 8. Figure 7 shows an example of the original design.

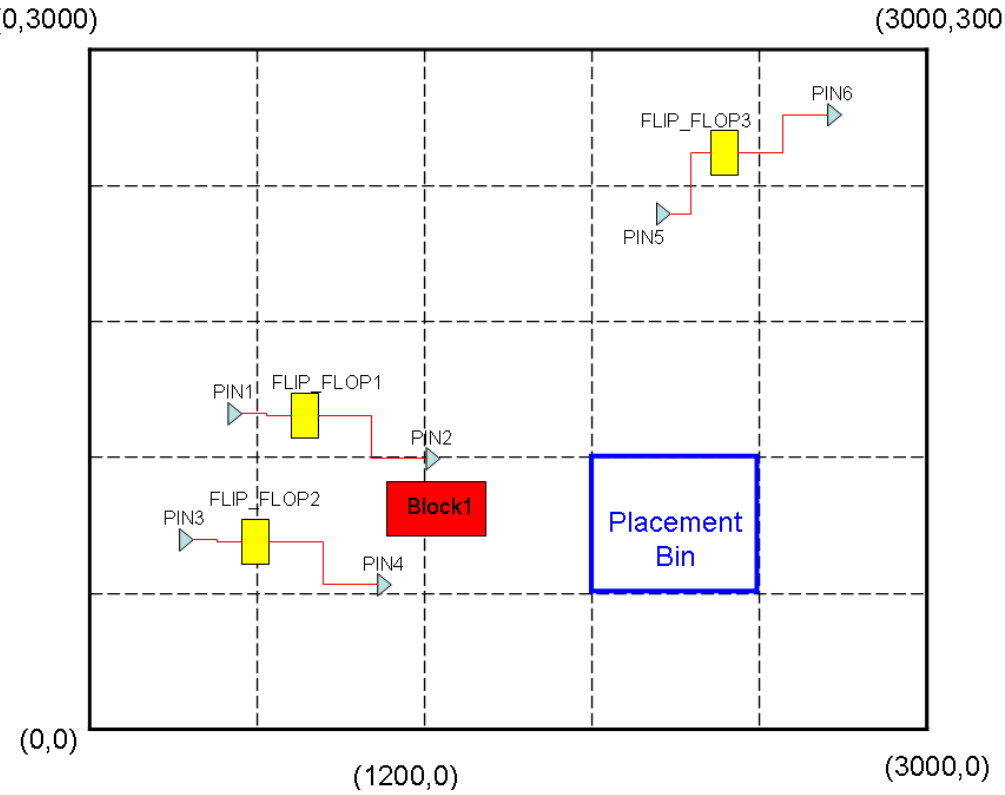


Figure 7: The given original design

Output file format:

The output file must contain:

1. The coordinates of all flip-flops.
2. All connections between pins and flip-flops and their new slack values.
3. Values of the maximum placement density.
4. Value of the objective function.
5. Execution time.

Syntax of the output file:

[FLIP_FLOP_LIST]

FF_name FLIP_FLOP_NAME1 (x1, y1)

FF_name FLIP_FLOP_NAME2 (x2, y2)

…

[END FLIP_FLOP_LIST]

[NET_LIST]

PIN_NAME1 FLIP_FLOP_NAME1 NewSlackValue1

PIN_NAME2 FLIP_FLOP_NAME1 NewSlackValue2

…

[END NET_LIST]

MAX_PLACEMENT_DENSITY = md

The maximum placement density of this design.

EXECUTION_TIME=m sec.

The execution time of the program.

Example of the output file format:

[FLIP_FLOP_LIST]

FF2 NEW_FLIP_FLOP (700, 800)

FF1 FLIP_FLOP3 (2300, 2450)

[END FLIP_FLOP_LIST]

[NET_LIST]

PIN1 NEW_FLIP_FLOP 350

PIN2 NEW_FLIP_FLOP 350

PIN3 NEW_FLIP_FLOP 870

PIN4 NEW_FLIP_FLOP 520

PIN5 FLIP_FLOP3 600

PIN6 FLIP_FLOP3 500

[END NET_LIST]

MAX_PLACEMENT_DENSITY = 292
EXECUTION_TIME = 1000 sec.

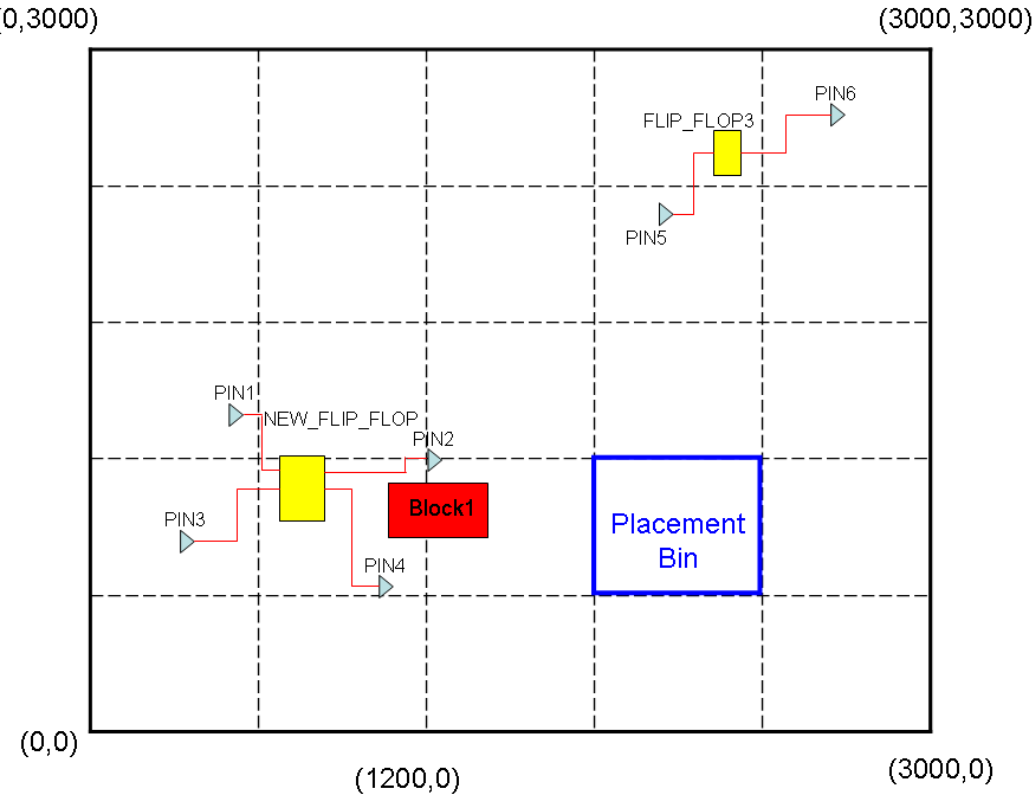


Figure 8: The new design with multi-bit flip-flop replacement

Language and Platform

Programming language should be either C or C++, and the program must be developed under Sun/Solaris, HP/UNIX or x86/LINUX based platform.

Grading Strategy

The functional correctness of the final result should be guaranteed; otherwise, the score for one testcase will be zero. If core dump occurs or the CPU time is more than 8 hours while executing a testcase, the score for this testcase will be zero as well.

When you design your graphical interface, you can play your creativity. Because we have only defined the area and bottom left corner of the flip-flops and blocks, the shape of flip-flops and blocks can be either a square or a rectangle.

Total score of a test case is calculated as follows:

- The total power consumption (50%)
- The final routing cost (10%)
- CPU time and memory usage (30%)
- Graphical interface for displaying the selection results (10%)