Power and Timing Optimization Using Multibit Flip-Flop

Sheng-Wei Yang, Tzu-Hsuan Chen, Jhih-Wei Hsu, Ting Wei Li, Cindy Chin-Fang Shen

Synopsys, Inc.

0. Revision History

2024-06-28 - Added binary naming instructions.

2024-06-20 - 7th Revision. Revised TNS definition. Corrected NumNets in example.

2024-05-10 - Sixth Revision. Revised timing calculation example.

2024-05-05 - Fifth Revision. Added banking and debanking criteria. Revised example.

2024-04-28 - Fourth Revision.

2024-04-14 - Third Revision.

2024-02-14 - Second Revision. Added the definition of 'displacement' and QpinDelay with more pictures. Also clarify the goal.

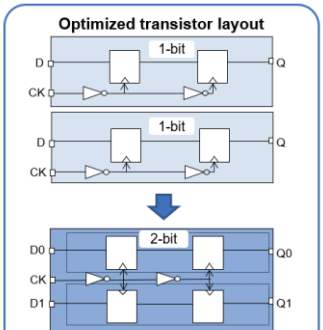
2024-02-01 - First Major Revision

1. Introduction

In modern technology nodes, the power and area minimization has become one of the most studied topics in semiconductor industry. One commonly applied concept is to replace single-bit flip-flops with multibit flip-flops. By using one multibit flip-flop to replace multiple single-bit flip-flops, more area can be freed up in the design as one multibit flip-flop takes less area to place than the single-bit flip-flops it replaces. Furthermore, before changing single-bit flip-flops to a multibit-flip-flop, each single-bit flip-flop has its own power, ground, and clock pin connections. Replacing them with one multibit-flip-flop can also efficiently reduce power, ground, and clock net routing complexity. Therefore, in modern technology nodes, this technique has been widely applied and referred to as "multibit flip-flop banking".

However, with the advance in technology nodes, timing, power, and area optimization has become a much more convoluted problem. For some timing critical nets, the original idea of multibit-flip-flop banking attempts would possibly result in worsening the timing, which hampers the overall optimization objective. Therefore, sometimes we have to divide a multibit flip-flop into several single-bit flip-flops to further optimize timing critical nets. This technique is oftentimes referred to as "multibit flip-flop debanking".

In this contest problem we will simulate the multibit flip-flop banking and debanking decisions in some virtual designs as testcases so that contestants would need to take timing, power, and area objectives together to find the best possible optimization solutions for each testcase.



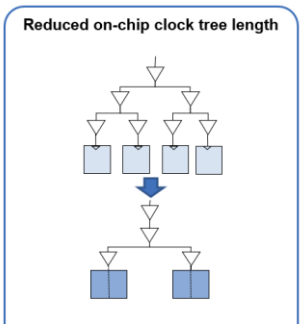


Figure 1(a): A simple transistor layout of two single-bit flip-flops and one multibit flip-flop. (left)

Figure 1(b): A diagram illustrating how dynamic power is reduced due to the reduction of on-chip clock tree length. (right)

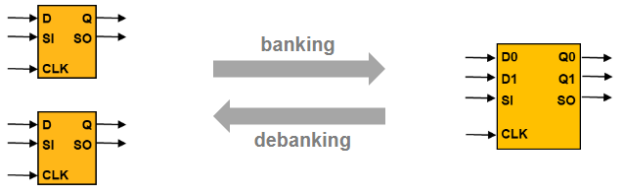


Figure 2: A simplified diagram of how two single-bit flip-flops can be banked into one two-bit flip-flop, and debanked vice versa. In this diagram, the upper left D pin of the single-bit flip-flop is mapped to the D0 pin of the two-bit flip-flop; the lower left D pin of the single-bit flip-flop is mapped to the D1 pin of the two-bit flip-flop.

2. Contest Objective

In this contest there are two kinds of cells given as data input: combinational gates and sequential flip-flops. For combinational gates, contestants are required to honor combinational gates' placement and connectivity information. Contestants can move, bank, or debank flip-flops in the testcase, but contestants should not displace or change any combinational gates.

The contestants need to develop a banking & debanking algorithm that can take any given testcase and produce a placement result that satisfies cell density constraints and with the resultant timing, power, and area optimized. No cell overlapping, combinational or sequential, is allowed in the placement result. A cost metric will be given for each testcase to identify the weight for each timing, power, and area calculation. The cost metrics of calculating timing, power, and area for this contest is as follows:

$$\sum_{\forall i \in FF} \left(\alpha \cdot TNS(i) + \beta \cdot Power(i) + \gamma \cdot Area(i)\right) + \lambda \cdot D$$

Where *i* stands for every flip-flop instance in the design. *TNS(i)* stands for the total negative slack of the flip-flop. *Power(i)* stands for the power consumption of the flip-flop. *Area(i)* stands for the area cost of the flip-flop. The design is divided into several bins, and we will define a utilization rate threshold for each bin in the design. *D* stands for the number of bins that violates the utilization rate threshold. α, β, γ, and λ are weight for each cost. The weighted summation of the four cost metrics above represents quality of result used in this contest.

Figure 3 shows an example of the density constraint of a placement. A whole placement region is divided into several uniform bins. For each bin, we set up a density threshold representing the upper bound of the total area of the cells allowed to be placed within the bin. The cells include both combinational gates and sequential flip-flops. Should the total cell area exceeds the threshold, we consider this bin violates the density constraints, hence accrue the penalty score by density weight λ. In the contest, the same threshold is applied to every uniform bin, which is defined in the BinMaxUtil of the input file.

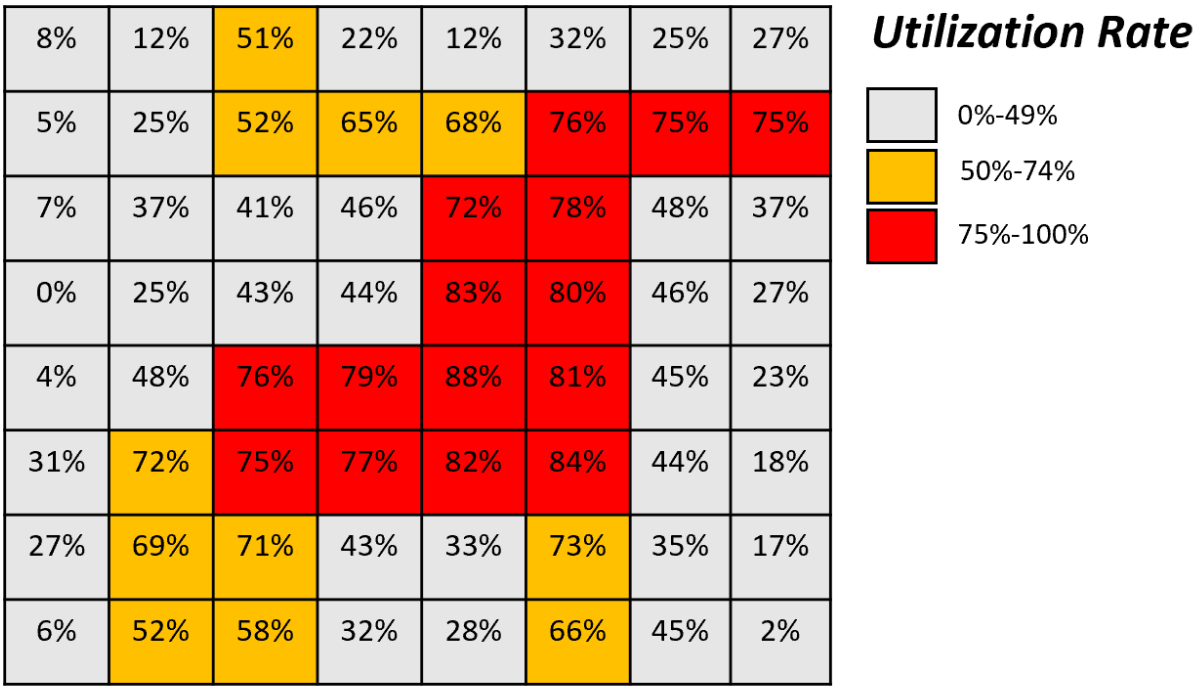


Figure 3: A representation of cell density constraint. The whole placement region is divided into several uniform bins, and a threshold is applied to every bin. Should the total cell area within the bin exceeds the threshold, density constraint penalty score is accrued.

3. Problem Formulation and Input/Output Format

Given:

1. Weight factors for cost metrics
2. Die size and input output ports
3. Cell library and flip-flops library information
4. A cell placement result with flip-flops cells
5. Max placement utilization ratio per uniform bin
6. Netlist
7. Timing slack and delay information
8. Power information

Output:

1. A flip-flop placement solution
2. A list of pin mapping between each input flip-flop pins and the output flip-flop pins

The output must have no cell overlaps and every flip-flop placed on the defined sites of the placement rows while satisfying max placement utilization ratio. The max placement utilization ratio is a cell density constraint implemented by dividing the design into several placement bins, and each bin would have a maximum ratio defining the percentage of area coverage allowed in each bin.

3.1 Format of Input Data

In this section we will define each syntax for data input. Most of the syntaxes are floating point numbers and we expect contestants to take care of the data range within DBL_MAX.

Weight factors of the cost metrics: α, β, γ, and λ values are given out as Alpha, Beta, Gamma, and Lambda, respectively.

Syntax

Alpha <alphaValue>

Beta <betaValue>

Gamma <gammaValue>

Lambda <lambdaValue>

Example

Alpha 1

Beta 5

Gamma 5

Lambda 10

Die size and input output ports: DieSize describes the dimension of the die, namely the placement area of the design. NumInput describes the number of input pins of the die. Each "Input" syntax describes the input pin name and its location. NumOutput describes the number of output pins of the die. Each "Output" syntax describes the output pin name and its location. Contestants cannot extend the die size (or region) to solve the density violation or cell overlap issues. The goal is to optimize the multiple objectives: timing and power; without increasing the die's region or area (die size).

Syntax

DieSize <lowerLeftX> <lowerLeftY> <upperRightX> <upperRightY>

NumInput <inputCount>

Input <inputName> <x-coordinate> <y-coordinate>

NumOutput <outputCount>

Output <outputName> <x-coordinate> <y-coordinate>

Example

DieSize 0 0 500 450

NumInput 2

Input INPUT0 0 25

Input INPUT1 0 5

NumOutput 2

Output OUTPUT0 500 25

Output OUTPUT1 500 5

Cell library: Cell library stands for the library of cells that would appear in the design. For flip-flops, the syntax would be FlipFlop, followed by its bits, name, width, height, and pin count. For combinational gates, the syntax would be Gate, followed by its name, width, height, and pin count.

Syntax

FlipFlop <bits> <flipFlopName> <flipFlopWidth> <flipFlopHeight> <pinCount>

Pin <pinName> <pinLocationX> <pinLocationY>

Gate <gateName> <gateWidth> <gateHeight> <pinCount>

Pin <pinName> <pinLocationX> <pinLocationY>

Example

FlipFlop 1 FF1 5 10 2

Pin D 0 8

Pin Q 5 8

FlipFlop 2 FF2 8 10 4

Pin D0 0 9

Pin Q0 8 9

Pin D1 0 6

Pin Q1 8 6

FlipFlop 2 FF2A 10 10 4

Pin D0 0 9

Pin D1 0 6

Pin Q0 8 9

Pin Q1 8 6

Placement result: The x and y coordinates describe the bottom-left corner of the cell:

Syntax

NumInstances <instanceCount>

Inst <instName> <libCellName> <x-coordinate> <y-coordinate>

Example:

NumInstances 2

Inst C1 FF1 50 20

Inst C2 FF1 50 0

Netlist:

Syntax

NumNets <netCount>

Net <netName> <numPins>

Pin <instName>/<libPinName>

Example

NumNets 4

Net N1 2

Pin INPUT0

Pin C1/D

Net N2 2

Pin INPUT1

Pin C2/D

Net N3 2

Pin C1/Q

Pin OUTPUT0

Net N4 2

Pin C2/Q

Pin OUTPUT1

Max placement utilization ratio: The die area of the design is divided into small bins whose dimensions are (BinWidth, BinHeight). The first bin starts from (<lowerLeftX>, <lowerLeftY>) of DieSize, and repeats with (BinWidth, BinHeight) throughout the DieSize. BinMaxUtil is a percentage value and it is calculated by dividing the total area of cells placed in the bin by the total area of the bin itself:

Syntax

BinWidth <width>

BinHeight <height>

BinMaxUtil <util>

Example

BinWidth 100

BinHeight 100

BinMaxUtil 75

The formulation of utilization ratio of a Bin = $\frac{\sum(area\ of\ each\ cell\ on\ the\ bin)}{area\ of\ the\ bin}$

Placement rows: The given row would start from (<$startX$>, <$startY$>) and has a <$totalNumOfSites$> of *cell sites* repetitively placed back-to-back in x direction. A cell site is a small rectangle defined in PlacementRows with a dimension of (<$siteWidth$>, <$siteHeight$>). The leftmost coordinate of the PlacementRows is <$startX$> and the rightmost coordinate of the PlacementRows is <$startX + siteWidth \times totalNumOfSites$>. Cell sites are placed back-to-back in x from <$startX$> to <$startX + siteWidth \times totalNumOfSites$>. This syntax defines the placement sites in the design. The height of PlacementRow is standard cell site height, and we will also give the site count. which means that the height of PlacementRow is the height of a cell site and the width of. Every cell should be placed on the PlacementRows, with its lower left corner aligning to the site grid of the PlacementRows.

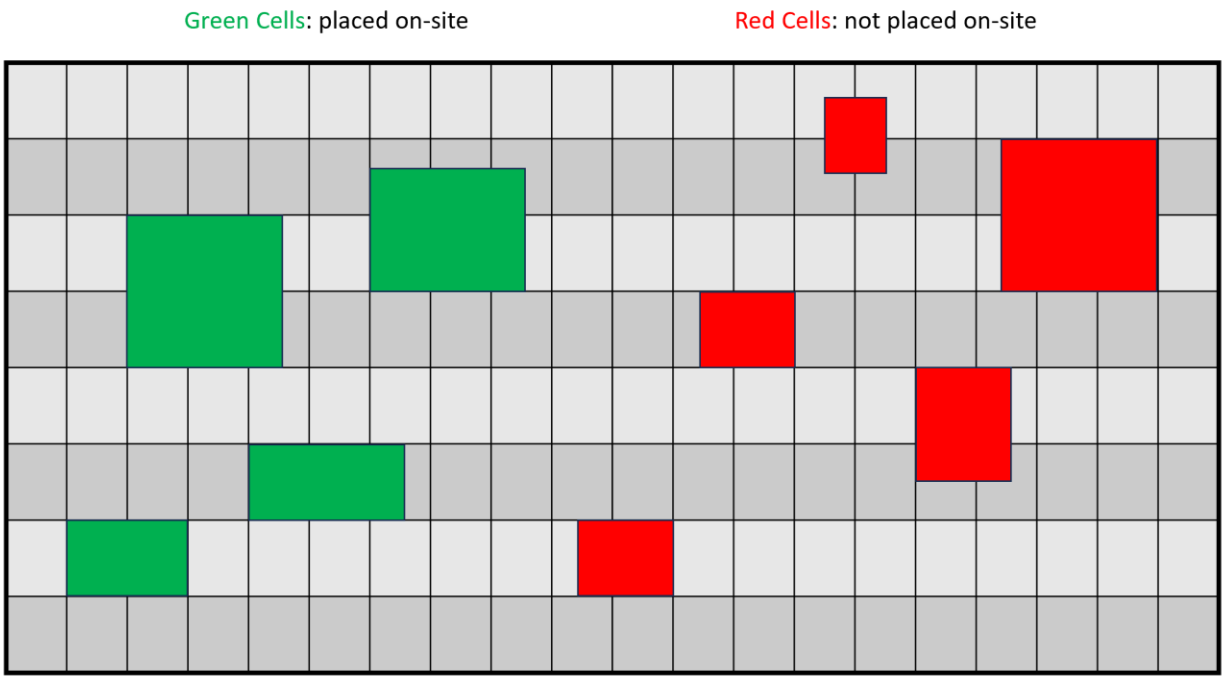


Figure 4: An example to illustrate what placing on-site means. We define on-site as cells with their bottom-left corner aligning to the site grid given from the PlacementRows from the design data input. The 4 green cells on the left part of the figure are placed on-site as their bottom-left corners are aligning with the site grids of the PlacementRows; the 5 red cells on the right side of the figure are not placed on-site as none of the cells have their bottom-left corner placed on the site grid.

Syntax

PlacementRows <startX> <startY> <siteWidth> <siteHeight> <totalNumOfSites>

Example

PlacementRows 0 0 2 10 400

Timing slack and delay information. For each flip-flop instance's D pin in the design, we will give out a timing slack information. The delay model is formulated by displacement delay and Q-pin delay. The displacement delay is the difference of half-perimeter wirelength between the previous output to the current gate input. For any cell displacement, we timess the coefficient with the difference of half-perimeter wirelength to get the displacement delay. For every flip-flop gate defined in the library we define a Q-pin delay for it.

Syntax

DisplacementDelay <coefficient>

QpinDelay <libCellName> <delay>

TimingSlack <instanceCellName> <PinName> <slack>

Example

DisplacementDelay 0.01

QpinDelay FF1 1

QpinDelay FF2 3

QpinDelay FF2A 2

TimingSlack C1 D 1

TimingSlack C2 D 1

Power consumption information: for every cell there is a power consumption rate.

Syntax

GatePower <libCellName> <powerConsumption>

Example

GatePower FF1 10

GatePower FF2 17

GatePower FF2A 18

3.2 Format of Output Data

The expected output is a list of bottom-left coordinates of each cell instance in the design. Contestants are expected to utilize multibit flip-flop banking and debanking techniques. Contestants are expected to provide a list of information describing the changed cells and their corresponding pin mappings. For the output cells, all cell names should not have existed in the input cell list. The contestants are expected to output the resultant cells and mapping only. No intermittent cells or any combinational gates should be printed out. The cell instance count could change according to difference banking and debanking methods contestants used to produce their output. Cell instances need to be placed on-site of the PlacementRows. When a cell is placed on-site, the lower left corner of the cell is at the same coordinate of the lower left corner of a placement site.

Syntax

CellInst <InstCount>

Inst <instName> <locationX> <locationY>

<originalCellPinFullName> map <resultCellPinFullNameName>

Example

CellInst 1

Inst C3 FF2 48 1

C1/D map C3/D0

C1/Q map C3/Q0

C2/D map C3/D1

C2/Q map C3/Q1

4. Example

We take the following circuit as a sample input:

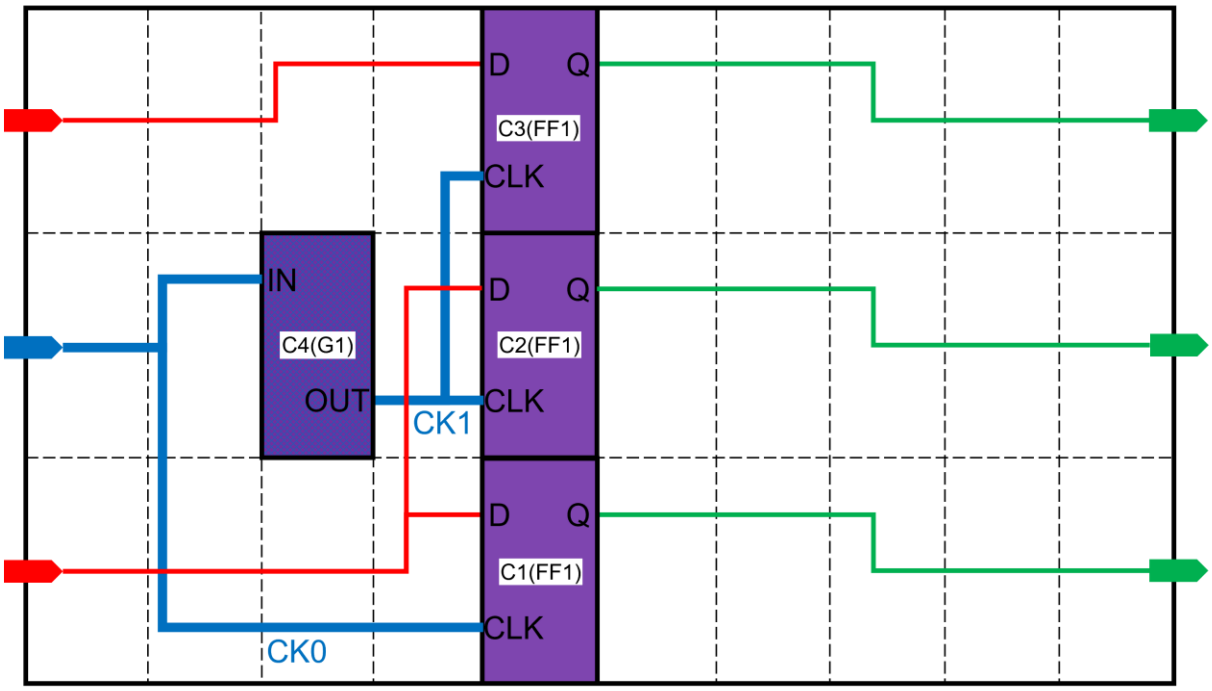


Figure 5: A visualization of the example data input circuit. CK0 and CK1 are two different clock nets.

	1-bit FF	2-bit FF
Normalized Area	1	1.6
Normalized Power	1	1.7
Normalized Q-Pin Delay	1	2

Alpha 1

Beta 5

Gamma 5

Lambda 1

DieSize 0.0 0.0 50.0 30.0

NumInput 3

Input INPUT0 0 5

Input INPUT1 0 25

Input CK0 0 15

NumOutput 2

Output OUTPUT0 50 5

Output OUTPUT1 50 15

Output OUTPUT2 50 25

FlipFlop 1 FF1 5.0 10.0 3

Pin D 0.0 8.0

Pin Q 5.0 8.0

Pin CLK 0.0 2.0

FlipFlop 2 FF2 8.0 10.0 5

Pin D0 0.0 9.0

Pin D1 0.0 6.0

Pin Q0 8.0 9.0

Pin Q1 8.0 6.0

Pin CLK 0.0 2.0

Gate G1 5.0 10.0 2

Pin IN 0.0 8.0

Pin OUT 5.0 2.0

NumInstances 4

Inst C1 FF1 20.0 0.0

Inst C2 FF1 20.0 10.0

Inst C3 FF1 20.0 20.0

Inst C4 G1 10.0 10.0

NumNets 7

Net N1 3

Pin INPUT0

Pin C1/D

Pin C2/D

Net N2 2

Pin INPUT1

Pin C3/D

Net N3 2

Pin C1/Q

Pin OUTPUT0

Net N4 2

Pin C2/Q

Pin OUTPUT1

Net N5 2

Pin C3/Q

Pin OUTPUT2

Net CK0 3

Pin CLK0

Pin C1/CLK

Pin C4/IN

Net CK1 3

Pin C4/OUT

Pin C2/CLK

Pin C3/CLK

BinWidth 10.0

BinHeight 10.0

BinMaxUtil 79.0

PlacementRows 0.0 0.0 2.0 10.0 25

PlacementRows 0.0 10.0 2.0 10.0 25

PlacementRows 0.0 20.0 2.0 10.0 25

DisplacementDelay 0.01

QpinDelay FF1 1.0

QpinDelay FF2 2.0

TimingSlack C1 D 1.0

TimingSlack C2 D 1.0

TimingSlack C3 D 1.0

GatePower FF1 10.0

GatePower FF2 17.0

The simple output:

Based on the given input, here's an example of output:

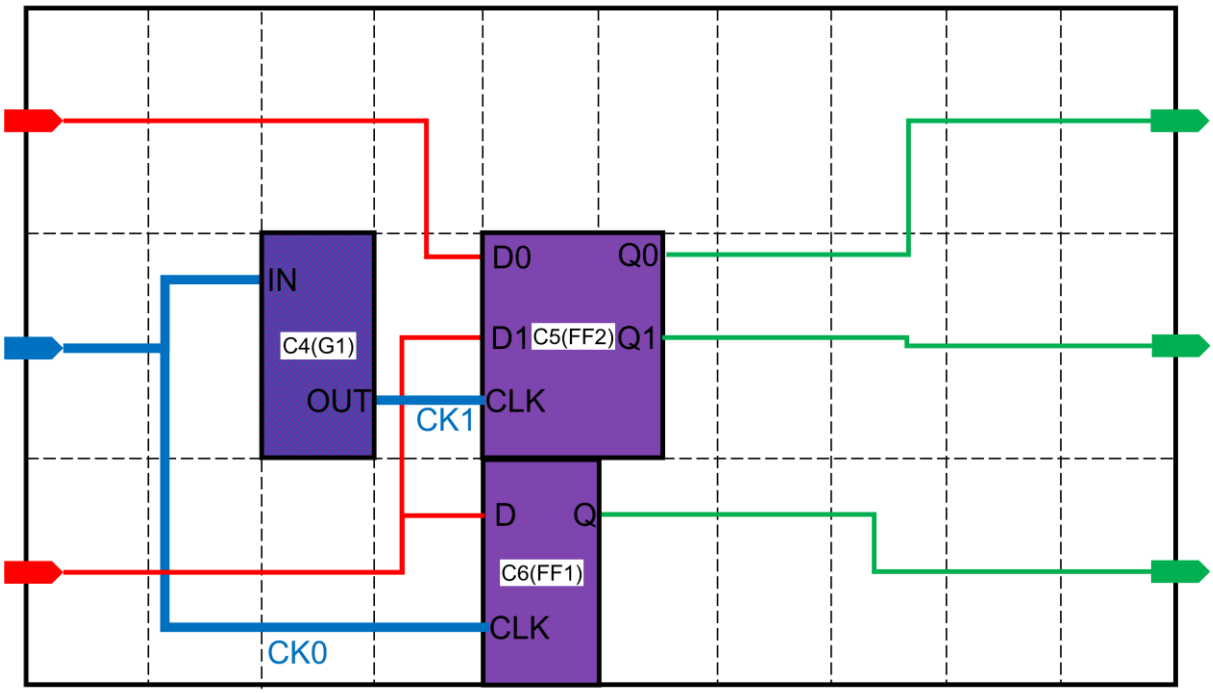


Figure 6: A visualization of the example data output circuit. CK0 and CK1 remain two different clock nets.

CellInst 2

Inst C5 FF2 20 10

Inst C6 FF1 20 0

C1/D map C6/D

C1/Q map C6/Q

C1/CLK map C6/CLK

C2/D map C5/D1

C2/Q map C5/Q1

C2/CLK map C5/CLK

C3/D map C5/D0

C3/Q map C5/Q0

C3/CLK map C5/CLK

5. Evaluation

The expected output should satisfy the following constraints:

A. All instances must be placed within the die region.
B. All instances must be without overlap and placed on-site of PlacementRows.
C. A list of banking and debanking mapping needs to be provided
D. Nets connected to the flip-flops must remain functionally equivalent to the data input. The result should not leave any open or short net.

Failure to satisfy any of the above constraints would result in no score in the testcase.

Banking Criteria:

Contestants are allowed to perform banking by clustering several lower-bit flip-flops to one higher-bit flip-flop. However, all D pin and Q pin connections need to remain functionally equivalent after banking, and all CLK pins need to be connected to the same clock net.

Debanking Criteria:

Contestants are allowed to perform debanking by split a higher-bit flip-flop to several lower-bit flip-flops. However, all D pin and Q pin connections need to remain functionally equivalent after debanking, and all CLK pins needs to remain connected to the same clock net.

Initial score is calculated by taking weighted timing, power, area, and utilization rate constraints into consideration. The cost metrics is as follows:

$$Initial\ score = \sum_{\forall i \in FF} \left(\alpha \cdot TNS(i) + \beta \cdot Power(i) + \gamma \cdot Area(i)\right) + \lambda \cdot D$$

TNS stands for total negative slack. We'll calculate the slack values of each input of flip-flop and accumulate all negative slack values to formulate TNS cost.

In each testcase we will give out pin delay information. This would be a reference value for contestants to optimize. The new timing delay would be related to how much displacement the contestant made, and also the increase or decrease of QpinDelay.

Taking one single-bit flip-flop FF_0 for example, let WL_0^Q be the original testcase' half-perimeter wirelength of the Q pin of FF_0, and δ_0 be QpinDelay of the FF_0. Let the next flip flop that connects to the Q pin of FF_0 be FF_N, where N stands for next level. Given that the Q pin of FF_0 connects to combinational gate G_1 and the D pin of FFN connects to combinational gate G_2, let WL_C stands for the half-perimeter wirelength of the combinational circuit between G_1 and G_2, WL_N^D be the half-perimeter wirelength of the D pin of FF_N. Let S_{FF0} denote the D pin slack of FF_0 and S_{FFN} be the D pin slack of FF_N, S_{FFN} can be calculated as:

$S_{FFN} = S_{FF0} + \delta_0 + \text{DisplacementDelay} * (WL_0^Q + WL_C + WL_N^D)$

Should the contestants' submitted result displaced FF_0 to a new location, the new wirelength delay slow down by DisplacementDelay * $(WL_0^{Q\prime} - WL_0^Q)$, where $WL_0^{Q\prime}$ stands for the new wirelength of the Q pin of the displaced FF_0; If FF_0 is changed to another flip-flop FF_0' in the submitted result, with δ_0' being the new QpinDelay, the increased gate delay would be $\delta_0' - \delta_0$.

In the end, the delay would be propagated to the D pin of the next stage of flip flop FF_N, where N denotes the next level. The new D-pin slack for FF_N, which we denote it as S_{FFN}', is calculated as follows:

$S_{FFN}' = S_{FFN} + (\delta_0 - \delta_0') + \text{DisplacementDelay} * (WL_0^Q - WL_0^{Q\prime}) + \text{DisplacementDelay} * (WL_N^D - WL_N^{D\prime})$

Where $WL_N^{D\prime}$ stands for the new half-perimeter wirelength of the D pin of FF_N', if FF_N is displaced to a new place or swapped to FF_N'.

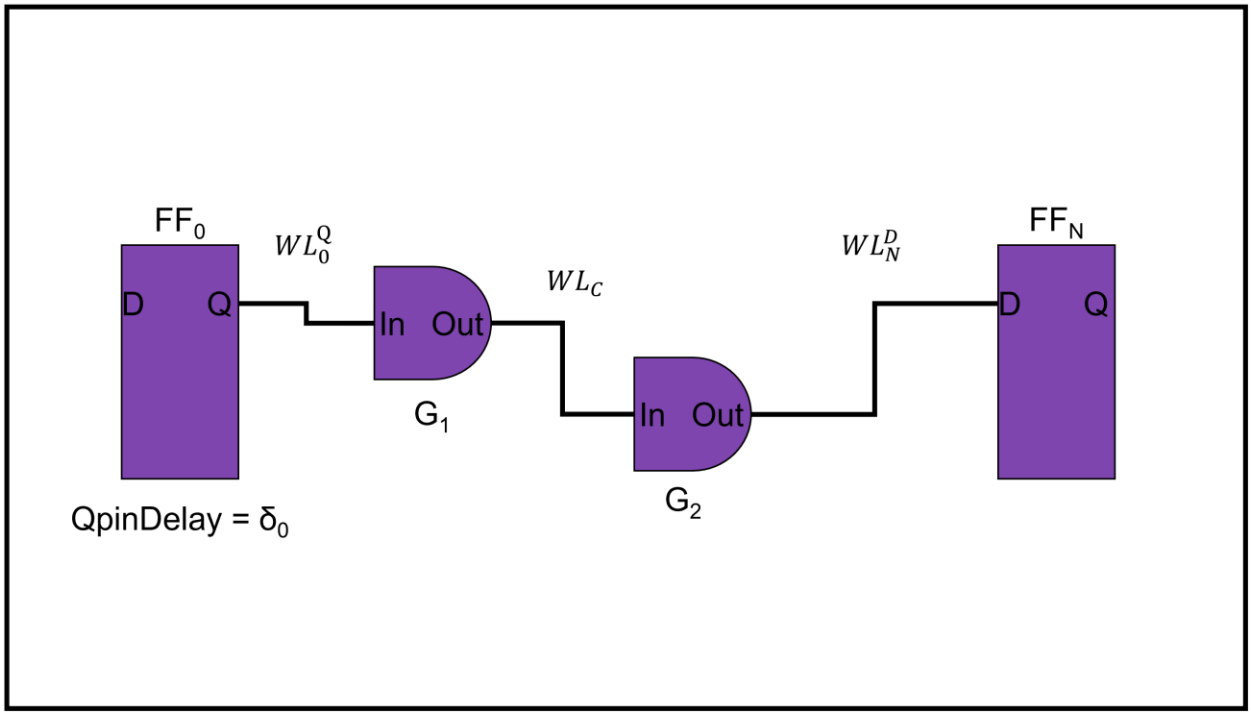


Figure 7(a): An illustration of data input showing how delay is calculated. Let the left flip-flop be FF_0 and the right flip-flop be FF_N, where N stands for the next stage. Let FF_0's QpinDelay be δ_0, the half-perimeter wirelength of the net of the Q pin of FF_0 be WL_0^Q, and the half-perimeter wirelength of the net of the D pin of FF_N be WL_N^D.

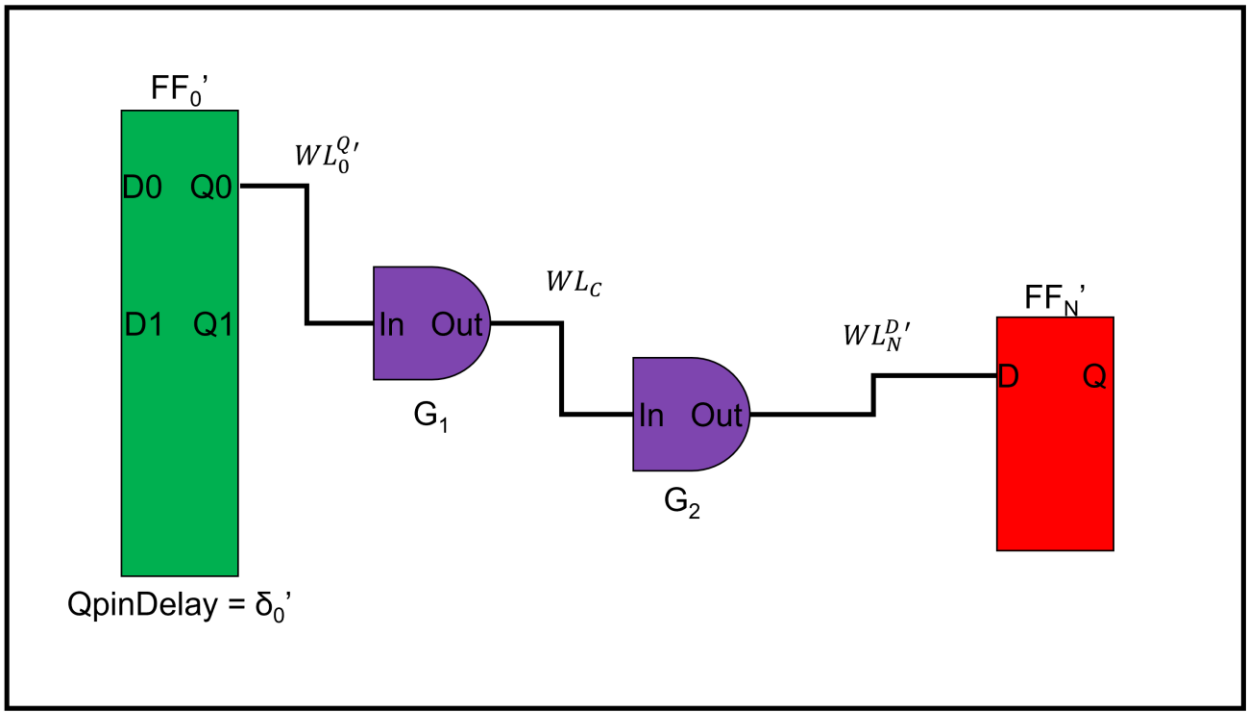


Figure 7(b): An illustration of the output showing how delay is calculated. Let the left flip-flop be changed to FF_0' and the right flip-flop be FF_N'. Let the QpinDelay of FF_0' be δ_0', the half-perimeter wirelength of the net of the Q pin of FF_0' be $WL_0^{Q\prime}$, and the half-perimeter wirelength of the net of the D pin of FF_N' be $WL_N^{D\prime}$. Let the slack of the D pin of FF_N' be S_{FFN}' and that of FF_N be S_{FFN}, the new slack of the D pin of FF_N' is calculated as: $S_{FFN}' = S_{FFN} + (\delta_0 - \delta_0') + \text{DisplacementDelay} * (WL_0^Q - WL_0^{Q\prime}) + \text{DisplacementDelay} * (WL_N^D - WL_N^{D\prime})$

For each FF_0, the evaluator would map the contestant's output FF_0' from its original FF_0 based on contestants' mapping list, and calculate the new slack S_{FF0}'.

$Power$ stands for total power consumption. We'll sum up the power consumption value from each flip-flop to formulate $Power$ cost.

$Area$ stands for total area. We'll sum up the area cost of each flip-flop to formulate $Area$ cost.

D stands for the number of bins that violates the utilization rate threshold. We'll calculate the total area cost of the flip-flops taking up in each bin and sum up all violating bins to formulate D cost.

If the program and the output data violate any of these above bullets, you will get 0 score for the corresponding test case.

4.1 Runtime factor

Runtime limit is 60 minutes for each case in the evaluation machine. The hidden cases will be in the same scale as public cases. We would like to introduce the runtime factor in this contest as formulated below to encourage the ideas with faster turnaround-time.

The number of CPU cores available for your program is 8 cores in the evaluation:

$$runtime\ factor = 0.02 \times \log_2\left(\frac{elapse\ time\ of\ test\ binary}{median\ elapse\ time}\right)$$

$$runtime\ factor\ bounded = max\left(-0.1, min(0.1, runtime\ factor)\right)$$

$$\boldsymbol{Final\ score} = Initial\ score \times (1.0 + runtime\ factor\ bounded)$$

With this proposed runtime factor, that means if the binary is 2x faster/slower, it would get 2% of initial score advantage/disadvantage. If the binary is 4x faster/slower, it would get 4% of initial score advantage/disadvantage. The runtime factor is bounded to 10%.

4.2 Program Requirements

Your program should be able execute like following:

./cadb_0000_alpha <input.txt> <output.txt>

Please follow the naming convention by prefixing your binary as cadb_0000*, where 0000 is the placeholder of contestants serial number. Please replace 0000 with your registered team number.

The *_alpha stands for the alpha testing stage. For each stage please specify with different postfixes (*_beta, *_final).

Please note that contestants should follow the naming description otherwise the score will be annulled.

Contestants will have access to the machine provided by ICCAD. Contestants are required to upload their binary along with every modules required to run their binary. Contestants should note that if the public account could not run the contestant's binary due to any reason, the score would result in failure.

5. References

[1] Ya-Chu Chang, Tung-Wei Lin, Iris Hui-Ru Jiang, and Gi-Joon Nam. 2019. Graceful Register Clustering by Effective Mean Shift Algorithm for Power and Timing Balancing. In Proceedings of the 2019 International Symposium on Physical Design (ISPD '19).

[2] Meng-Yun Liu, Yu-Cheng Lai, Wai-Kei Mak, and Ting-Chi Wang. 2022. Generation of Mixed-Driving Multi-Bit Flip-Flops for Power Optimization. In Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design (ICCAD '22).

[3] Gang Wu, Yue Xu, Dean Wu, Manoj Ragupathy, Yu-yen Mo, and Chris Chu. 2016. Flip-flop clustering by weighted K-means algorithm. In Proceedings of the 53rd Annual Design Automation Conference (DAC '16)