IJETIE International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue 3, March 2015
ISSN: 2394 - 6598
www.ijetie.org

Clustering Based Multi Flip Flop Merging Using Agglomerative Clustering Algorithm

V.Gopi1, R.Sankar2

1Professor, Department of ECE, PSN College of Engineering and Technology, Tirunelveli.
1Email:Kanyavicky@gmail.com
2M.E (Applied Electronics), PSN College of Engineering and Technology, Tirunelveli.
2Email:rmksankar@gmail.com

Abstract—**Based on the elimination feature of redundant inverters in merging 1-bit flip-flops into multi-bit flip-flops, gives reduction of wired length and this result in reduction of power consumption. With the growing popularity of portable devices, power reduction has become a popular design goal for advanced design application. 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. In this paper, we propose agglomerative clustering algorithm to find the nearest clustering of flip flops for merging the flip flops. T Based on the elimination feature of redundant inverters in merging 1-bit flip-flops into multi-bit flip-flops, gives reduction of wired length and this result in reduction of power consumption.. With the growing popularity of portable devices, power reduction has become a popular design goal for advanced design application. 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. In this paper, we propose agglomerative clustering algorithm to find the nearest clustering of flip flops for merging the flip flops. This algorithm finds the clusters of flip flop and finally combine FFs to reduce the wire length.**

Keywords— Flip flops, Power Efficient, Clock power, merging.

I. 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.

In this section, we will introduce multi-bit flip-flop conception. Before that, we will review single-bit flip-flop. Figure 1 shows an example of single-bit flip-flop. A single-bit flip-flop has two latches (Master latch and slave latch). The latches need "Clk" and "Clk' " signal to perform operations. In order to have better delay from Clk-> Q, we will regenerate "Clk" from "Clk' ". Hence we will have two inverters in the clock path.

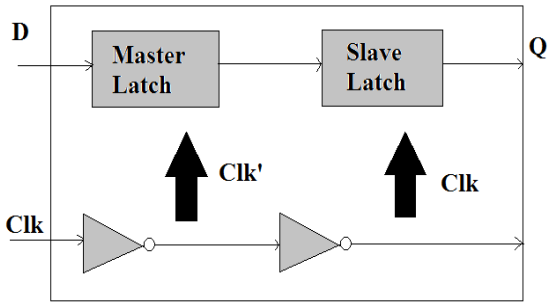

Fig: Single-Bit Flip-Flop

Each 1-bit flip-flop contains two inverters, master-latch and slave-latch. Due to the manufacturing 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 single-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.

II. MERGING TWO 1-BIT FLIP-FLOPS INTO ONE 2-BIT FILP FLOP

It has two data input pins, two data output pins, one clock pin and reset pin. Use dual-bit flip-flop can get the benefits of lower power consumption then single-bit, and almost no other additional costs to

IJETIE International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue 3, March 2015
ISSN: 2394 - 6598
www.ijetie.org

pay. Figure shows the true table of dual-bit flip-flop cell. We could find that when CK is positive edge, the value of Q1 will pass to D1, and the value of Q2 will pass to D2. Or Q1 and Q2 will keep original value.

In our proposed work, we implement the multi-bit Flip Flop concept in merging of two or more flip flops according to application to reduce the block size and power consumption. Here the merging of flip flops are done by forming clusters by agglomerative clustering algorithm. This clustering algorithm forms the separation of clusters according to the minimum distance between Flip-Flops. From the neighboring information, we merge the flip flops.

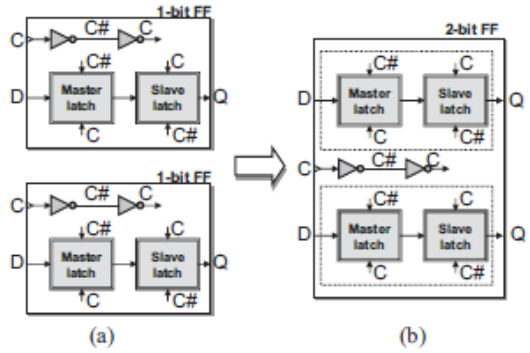

Fig: Example of merging two 1-bit flip-flops into one 2-bit flip-flop.(a) Two 1-bit flip-flops (before merging). (b) 2-bit flip-flop (after merging).

Given a cell library L and a placement which contains a lot of flip-flops, our target is to merge as many flip-flops as possible in order to reduce the total power consumption. If we want to replace some flip-flops $f1,\ldots, f\,j-1$ by a new flip flop $f\,j$, the bit width of $f\,j$ must be equal to the summation of bit widths in the original ones (i.e., $\_bi = b\ j$, $i = 1$ to $j-1$). Besides, since the replacement would change the routing length of the nets that connect to a flip-flop, it inevitably changes timing of some paths.

Finally, to ensure that a legalized placement can be obtained after the replacement, there should exist enough space in each bin. To consider these issues, we define two constraints as follows.

1) Timing Constraint for a Net Connecting to a Flip-Flop f j from a Pin pi :

To avoid that timing is affected after the replacement, the Manhattan distance between *pi* and *f j* cannot be longer than the given constraint *S(pi)* defined on the pin *pi* [i.e., $M(pi , f\,j) \leq S(pi)$]. Based on each timing constraint defined on a pin, we can find a feasible placement region for a flip-flop ***fj*** .

Assume pins $p1$ and $p2$ connect to a 1-bit flip-flop $f1$. Because the length is measured by Manhattan distance, the feasible placement region of $f1$ constrained by the pin *pi* [i.e., $M(pi , f1) \leq S(pi)$] would form a diamond region, which is denoted by $Rp(pi)$, $i = 1$ or 2. See the region enclosed by dotted lines in the figure. Thus, the legal placement region of $f1$ would be the overlapping region enclosed by solid lines, which is denoted by $R(f1)$. $R(f1)$ is the overlap region of $Rp(p1)$ and $Rp(p2)$.

2) Capacity Constraint for Each Bin Bk :

The total area of flip-flops intended to be placed into the bin *Bk* cannot be larger than the remaining area of the bin *Bk* (i.e., $\_A(fi) \leq RA(Bk)$).

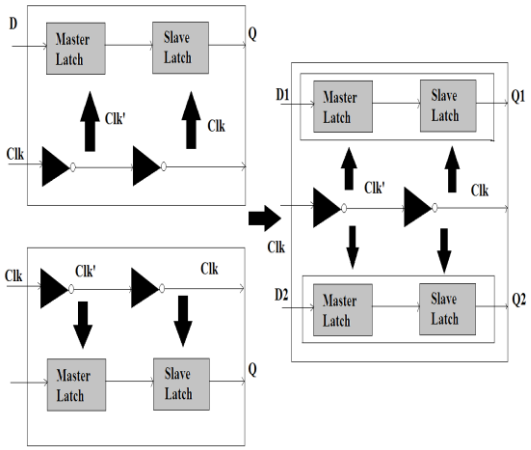

Fig: An example of merging two 1-bit flip-flops into one 2-bit flip-flop.

To reduce the complexity, we first divide the whole placement region into several sub regions, and use the combination table to replace flip-flops in each sub region. Then, several sub regions are combined into a larger sub region and the flip-flops are replaced again. So, that those flip-flops in the neighboring sub regions can be replaced further.

1) Region Partition (Optional):

To speed up our problem, we divide the whole chip into several sub regions. By suitable partition, the computation complexity of merging flip-flops can be reduced significantly.

IJETIE
International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue 3, March 2015
ISSN: 2394 - 6598
www.ijetie.org

2) Replacement of Flip-flops in Each Sub region:

Before illustrating our procedure to merge flip-flops, we first give an equation to measure the quality if two flip-flops are going to be replaced by a new flip flop as follows:
Where routing length denotes the total routing length between the new flip-flop and the pins connected to it, and available area represents the available area in the feasible region for placing the new flip-flop. α is a weighting factor. Thus, we will give it a smaller cost. Once the flip-flops cannot be merged to a higher-bit type.

3) Bottom-Up Flow of Sub region Combinations (Optional):

To reduce power consumption furthermore, we can combine several sub regions to obtain a larger sub region and perform the replacement again in the new sub region again. The procedure repeats until we cannot achieve any replacement in the new sub region.

As the procedure repeats in a higher level, the number of merge able flip-flops gets fewer. However, it would spend much time to get little improvement for power saving. To consider this issue, there exists a trade-off between power saving and time consuming in our program.

III. TRANSFORMATION OF PLACEMENT SPACE

We have shown that the shape of a feasible placement region associated with one pin *pi* connecting to a flip-flop *fi* would be diamond in Section II. Since there may exist several pins connecting to ***f i*** **,** the legal placement region of ***f i*** are the overlapping area of several regions. There are two pins *p*1 and *p*2 connecting to a flip-flop *f*1, and the feasible placement regions for the two pins are enclosed by dotted lines, which are denoted by *Rp(p*1*)* and *Rp(p*2*)*, respectively. Thus, the legal placement region *R(f*1*)* for *f*1 is the overlapping part of these regions. In Fig.4 *R(f*1*)* and *R(f*2*)* represent the legal placement regions of *f*1 and *f*2. Because *R(f*1*)* and *R(f*2*)* overlap, we can replace *f*1 and *f*2 by a new flip-flop *f*3 without violating the timing constraint.

However, it is not easy to identify and record feasible placement regions if their shapes are diamond. Moreover, four coordinates are required to record an overlapping region. Thus, if we can rotate each segment 45°, shapes of all regions would become rectangular, which makes identification of overlapping regions become very simple.

For example, the legal placement region, enclosed by dotted lines can be identified more easily if we change its original coordinate system. In such condition, we only need two coordinates, which are the left-bottom corner and right-top corner of a rectangle, as shown in to record the overlapped area instead of using four coordinates. Suppose the location of a point in the original coordinate system is denoted by (x, y). After coordinate transformation, the new coordinate is denoted by $(x\_, y\_)$. In the original transformed equations, each value needs to be divided by the square root of 2, which would induce a longer computation time. Since we only need to know the relative locations of flip-flops, such computation are ignored in our method. Thus, we use $x\_\_$ and $y\_\_$, to denote the coordinates of transformed locations.

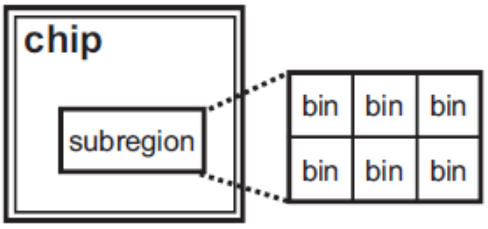

Fig: Example of region partition with six bins in one sub region.

IV. SYSTEM FORMULATION

In the beginning, they identify a legal placement region for each flip-flop fi. First, the feasible placement region of a flip-flop associated with different pins is found based on the timing constraints defined on the pins. Then, the legal placement region of the flip-flop fi can be obtained by the overlapped area of these regions. However, because these regions are in the diamond shape, it is not easy to identify the overlapped area.

Therefore, the overlapped area can be identified more easily if we can transform the coordinate system of cells to get rectangular regions. In the second stage, we would like to build a combination table, which defines all possible combinations of flip-flops in order to get a new multi-bit flip-flop provided by the library.

IJETIE
International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue 3, March 2015
ISSN: 2394 - 6598
www.ijetie.org

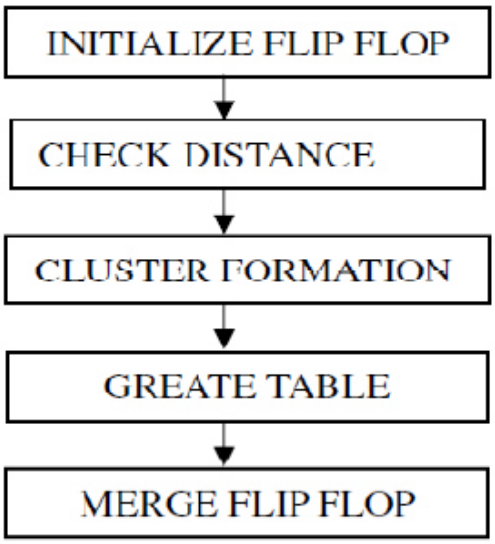

Fig: Formulation of flip flop

The flip-flops can be merged with the help of the table. After the legal placement regions of flip-flops are found and the combination table is built, we can use them to merge flip-flops. To speed up our program, we will divide a chip into several bins and merge flip-flops in a local bin.

However, the flip-flops in different bins may be mergeable. Thus, we have to combine several bins into a larger bin and repeat this step until no flip-flop can be merged anymore.

V. PROPOSED WORK

We propose agglomerative clustering algorithm to find the nearest clustering of flip flops for merging the flip flops. The algorithm forms clusters in a bottom-up manner, as follows:

- ✓ Initially, put each article in its own cluster.
- ✓ Among all current clusters, pick the two clusters with the smallest distance.
- ✓ Replace these two clusters with a new cluster, formed by merging the two original ones.
- ✓ Repeat the above two steps until there is only one remaining cluster in the pool.

This algorithm finds the clusters of flip flop and finally combine FFs to reduce the wire length. The agglomerative clustering algorithm will result in a binary cluster tree with single article clusters as its leaf nodes and a root node containing all the articles. In the clustering algorithm, we use a distance measure based on log likelihood.

VI. CONSTRUCT D-FLIP FLOPS:

A master–slave D flip-flop is created by connecting two gated D latches in series, and inverting the enable input to one of them. This flip-flops gets single bit as input and generate output as Q and Q`. At a time any one of D flip-flop was activated according to clock pulse generation. Each FF handle single bit input for separated clock generator.

1. CHECK DISTANCE:

Normally, flip flops in a logic circuit was handling single bit input with separated clock generation. This may consume more power. This can reduce by merging of flip flops. In this module, they find the distance between the neighboring flip flops which gets the distance between each flip flops.

The Manhattan distance function computes the distance that would be traveled to get from one data point to the other if a grid-like path is followed. The Manhattan distance between two items is the sum of the differences of their corresponding components.

The formula for this distance between a point $\boldsymbol{X}$=($\boldsymbol{X1}$, $\boldsymbol{X2}$, etc.) and a point $\boldsymbol{Y}$=($\boldsymbol{Y1}$, $\boldsymbol{Y2}$, etc.) is:

$$\mathbf{d} = \sum_{\mathbf{i}=\mathbf{1}}^{\mathbf{n}} \left| \mathbf{X_i} - \mathbf{Y_i} \right|$$

Where n is the number of variables, and $\boldsymbol{Xi}$ and $\boldsymbol{Yi}$ are the values of the ith variable, at points $\boldsymbol{X}$ and $\boldsymbol{Y}$ respectively.

The following figure illustrates the difference between Manhattan distance and Euclidean distance:

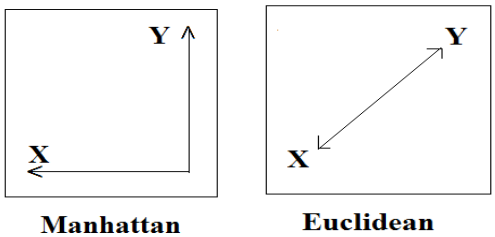

Fig: Distance between Manhattan and Euclidean

2. FORM CLUSTER FLIP-FLOP:

In this module we form flip flops in clusters according to the distance information. This clustering represents what are the flip flops can be made to merge and form as a multi bit flip flops.

IJETIE
International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue 3, March 2015
ISSN: 2394 - 6598
www.ijetie.org

This clustering was found by agglomerative clustering algorithm to form clustered flip flop.

Agglomerative hierarchical clustering is a bottom-up clustering method where clusters have sub-clusters, which in turn have sub-clusters, etc. The classic example of this is species taxonomy. Gene expression data might also exhibit this hierarchical quality (e.g. neurotransmitter gene families). Agglomerative hierarchical clustering starts with every single object (gene or sample) in a single cluster.

Then, in each successive iteration, it agglomerates (merges) the closest pair of clusters by satisfying some similarity criteria, until all of the data is in one cluster.

The hierarchy within the final cluster has the following properties:

- ✓ Clusters generated in early stages are nested in those generated in later stages.
- ✓ Clusters with different sizes in the tree can be valuable for discovery.

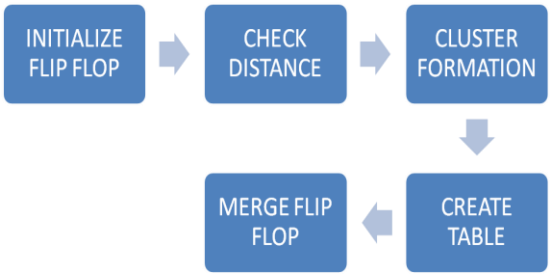

Fig: Block Diagram

3. CREATE TABLE:

In this module, we construct a table called as combinational table which contains cluster information of each flip flop and distances between them. From this, the decision was taken to merge FFs. This table was update according to the application usage.

4. MERGE FLIP-FLOP:

In this stage, we finally merge flip flops according to the acceptable range from combinational table. This formation made the individual FFs in to multi-bit flip-flops which contains common clock for FFs and form multi bit FF.

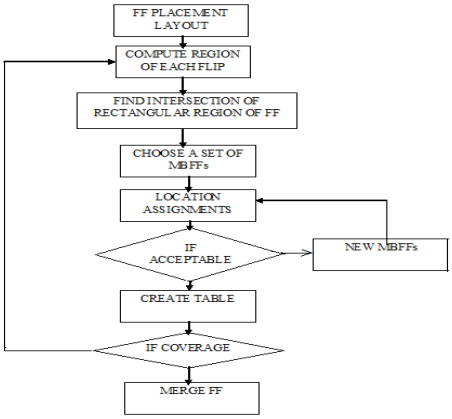

Fig: System Flow Diagram.

VII. EXPERIMENTAL RESULTS

We implemented our algorithm in C++ language, and all experiments were run on workstation with a 3.33- GHz Intel Core i7-980X processor with 16-GB memory. Our experiment can be divided into two parts.

In our algorithm, there exist two values which would affect our results: the first one is the dimension of a sub region since we would partition a chip into several sub regions. The second one is the parameter used in the cost function and these are showed in the following tables.

The procedure of flip-flop replacements is depending on the combination table, which records the relationships among the flip-flop types. The concept of pseudo type is introduced to help to enumerate all possible combinations in the combination table. By the guidelines of replacements from the combination table, the impossible combinations of flip-flops will not be considered that decreases execution time.

Besides power reduction, the objective of minimizing the total wire length also is considered to the cost function. The experimental results show that our algorithm can achieve a balance between power reduction and wire length reduction.

IJETIE
International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue 3, March 2015
ISSN: 2394 - 6598
www.ijetie.org

Circuit	Approach in [6]			Our Approach		
	PR_Ratio (%)	WR_Ratio (%)	Times (sec)	PR_Ratio (%)	WR_Ratio (%)	Times (sec)
C1	14.8	0.917	0.01	15.9	0.928	0.07
C2	16.9	0.947	0.04	18.0	0.934	0.12
C3	17.1	0.948	0.10	17.8	0.928	0.24
C4	16.8	0.945	0.28	17.6	0.932	0.84
C5	17.1	0.949	0.60	17.8	0.936	1.51
C6	17.2	0.949	78.92	17.9	0.938	30.43
Comp	0.95	1.01	2.41	1.00	1.00	1.00

	Case 1	Case 2	Case 3	Case 4	Case 5
Library	1,2,4	1,2,4,4,8	1,2,4,6,13	1,2,4,8	1,2,4,8
Flip flop number	120	953	5524	60000	1728000
$Power_{ori}$ (unit 10^3)	12	95	552	6000	172800
$Power_{merged}$ (unit 10^3)	09	67	430	4208	136509
PR-Ratio(%)	20.97	28.80	22.11	29.87	21.00
WL_{ori} (unit 10^3)	83	577	3563	53625	1199304
WL_{merged} (unit 10^3)	71	506	2189	31008	1068961
WR_Ratio (%)	85.62	87.77	61.44	57.82	89.13
Times (s)	0.08	0.24	1.07	36.7	23.77
Times of parser	0.07	0.15	0.29	3.8	21.53

Fig: Experimental Results under different conditions

VIII. CONCLUSION

In this paper, we proposed merging of flip flops to reduce power reduction of single bit FF to form multi bit FF. This merging was found by acceptable distance range each neighboring flip flops. This was done by forming clusters of flip flops by using agglomerative clustering algorithm.

In this stage, we finally merge flip flops according to the acceptable range from combinational table. This formation made the individual FFs in to multi-bit flip-flops which contains common clock for FFs and form multi bit FF.

IX. REFERENCES

[1] P. Gronowski, W. J. Bowhill, R. P. Preston, M. K. Gowan, and R. L. Allmon, "High-performance microprocessor design," IEEE J. Solid-State Circuits, vol. 33, no. 5, pp. 676–686, May 1998.

[2] W. Hou, D. Liu, and P.-H. Ho, "Automatic register banking for lowpower clock trees," in Proc. Quality Electron. Design, San Jose, CA, Mar. 2009, pp. 647–652.

[3] D. Duarte, V. Narayanan, and M. J. Irwin, "Impact of technology scaling in the clock power," in Proc. IEEE VLSI Comput. Soc. Annu. Symp., Pittsburgh, PA, Apr. 2002, pp. 52–57.

[4] H. Kawagachi and T. Sakurai, "A reduced clock-swing flip-flop (RCSFF) for 63% clock power reduction," in VLSI Circuits Dig. Tech. Papers Symp., Jun. 1997, pp. 97–98.

[5] Y. Cheon, P.-H. Ho, A. B. Kahng, S. Reda, and Q. Wang, "Power-aware placement," in Proc. Design Autom. Conf., Jun. 2005, pp. 795–800.

[6] Y.-T. Chang, C.-C. Hsu, P.-H. Lin, Y.-W. Tsai, and S.-F. Chen, "Post-placement power optimization with multi-bit flip-flops," in Proc. IEEE/ACM Comput.-Aided Design Int. Conf., San Jose, CA, Nov. 2010, pp. 218–223.

[7] Faraday Technology Corporation [Online]. Available: http://www. faraday-tech.com/index.html

[8] C. Bron and J. Kerbosch, "Algorithm 457: Finding all cliques of an undirected graph," ACM Commun., vol. 16, no. 9, pp. 575–577, 1973.

[9] CAD Contest of Taiwan [Online]. Available: http://cad_contest.cs. nctu.edu.tw/cad11

[10] L.-T. Wang, Y.-W. Chang, and K.-T. Cheng, Eds., Electronic Design Automation: Synthesis, Verification, and Test. Burlington, MA: Elsevier/ Morgan Kaufmann, 2009.

[11] D. Duarte, V. Narayanan, and M. J. Irwin, "Impact of technology scaling in the clock power," in Proc. IEEE VLSI Comput. Soc. Annu. Symp., Pittsburgh, PA, Apr. 2002, pp. 52–57.

[12] P. Gronowski, W. J. Bowhill, R. P. Preston, M. K. Gowan, and R. L. Allmon, "High-performance microprocessor design," IEEE J. Solid-State Circuits, vol. 33, no. 5, pp. 676–686, May 1998.