ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Clock Power Reduction Using Merged Flip Flops Technique

S.Murugan
ME VLSI Design, SCAD College of Engineering and Technology,
Cheranmahadevi,Tirunelveli,Tamilnadu, India
Murugan11111988@gmail.com

Abstract

The main constraint in any VLSI chip design are reducing power consumption and area and increasing speed. In this project my aim is to obtain reduced clock power by replacing single bit flip flops into multi bit flip flops. To perform a co-ordinate transformation, identify those flip flops that can be merged and their legal regions. The legal placement region of the flip flop can be obtained by the overlapped area of these regions and these regions are in the diamond shape, it is not easy to identify the overlapped area. The overlapped area can be identified more easily to get rectangular regions. To avoid wasting time in finding impossible combination of flip flops, first build a combination table before actually merging two flip flops. All possible combinations of flip flops in order to get a new multi-bit flip flops provided by the library. The flip flops can be merged with the help of the combination table. Then partition a chip in to several sub regions and perform replacement in each sub region to reduce the complexity. Then combine several bins into a larger bin and repeat this step until no flip flop can be merged any more. It is applicable for other low power design circuits such as counter and shift register which are used in data processing applications.

Keywords: Clock power reduction, merging, multi-bit flip flop, replacement, wire length.

Introduction

Due to the popularity of portable electronic products, low power system has attracted more attention in recent years. As technology advances, a system on a chip design can contain more and more components that lead to a higher power density. Reducing the power consumption not only can enhance battery life but also can avoid the overheating problem, which would increase the difficulty of packaging or cooling. Given a design that the locations of the cells have been determined, the power consumed by clocking can be reduced further by replacing several flip flops with multi-bit flip flops. During clock tree synthesis, less number of flip flops means less number of clock sinks. Thus, the resulting clock network would have smaller power consumption and uses less routing resource. As CMOS technology progresses the driving capability of an inverter based clock buffer increases significantly.

Merging two 1-bit flip flops into single 2-bit flip flops, we also need to check whether the cell library provides the type of the new flip flop. For example, we have to check the availability of a 3-bit flip flops in the cell library when we desire to replace 1- and 2-bit flip flops by a 3-bit flip flops. Figure 1. shows the block diagrams of 1 and 2-bit flip flops.

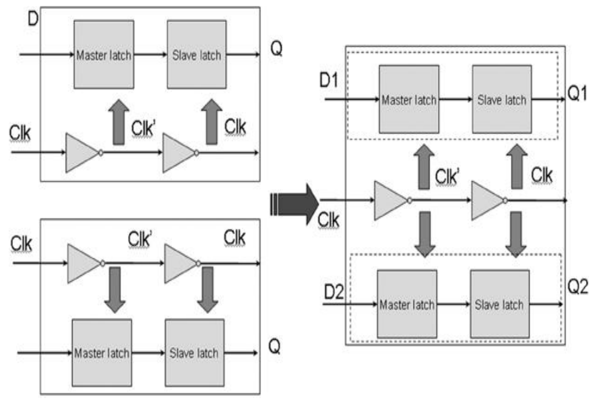

Figure 1. Merging two 1-bit flip flops into single 2-bit flip flops

Related Work

Chang Y-T et al [6] used the multi-bit flip flop technique in the post-placement stage to reduce power consumption. In this work a graph based approach is used to reduce the clock power. Each node in the graph represents a flip flop. Two flip flops which satisfy the timing and capacity constraint can be replaced by an edge which is built between the corresponding nodes. The replacement of the flip-flop is done using m-clique in the graph. The flip flops

ISSN: 2277-9655
Impact Factor: 1.852

corresponding to the nodes in an m-clique can be replaced by a m-bit flip flop. The branch-and-bound and backtracking algorithm is made use to find the m-cliques in a graph. In this work there is a possibility that one flip flop may belong to more number of m-bit flip-flop. A greedy heuristic algorithm is used to find the maximum independent set of cliques, in that each node belongs to only one clique. In this method there is a chance of getting a multi-bit flip-flop which has a bit width outside the required library. So finding out this undesired bit width may lead to the wastage of time.

Proposed System

The D Flip flop is the edge-triggered variant of the transparent latch. On the rising (usually, although negative edge triggering is just as possible) edge of the clock, the output is given the value of the D input at that moment. The output can be only change at the clock edge, and if the input changes at other times, the output will be unaffected. D flip flops are by far the most common type of flip flops and some devices are made entirely from D flip flops. They are commonly used for counter, shift-registers and input synchronization.

Objectives

- Reduce the power consumption.
- To reduce the area.
- To reduce the delay and power of a clock network.
- To control clock skew because of common clock signal.

The above objectives can be achieved by merging several flip flops and synchronizing with clock signals.

Problem Statement

The following problem statement has been identified:

- Several flip flops needs a separate clock signal, hence Power consumption is high.
- Since several flip flops needs a separate clock signal area consumed is also high.

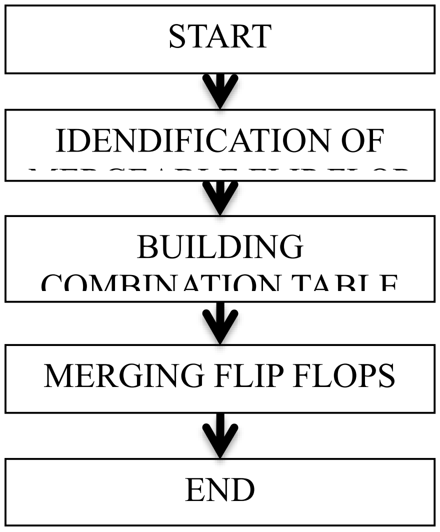

Figure 2. Flow chart of the proposed algorithm

A. Building the Combination Table

```
Step 1. T = InitializationCombinationTable(L);
Step 2 .InsertPseudoType(L);
Step 3 .SortByBitNumber (L);
Step 4. for each $n_i$ in T do
Step 5 .InsertChildrens (ni, NULL, NULL);
Step 6. index = 0;
Step 7 .while index != size(T) do
Step 8 .range_first = index;
Step 9 .range_second = size(T);
Step 10. index = size(T);
Step 11. for each $n_i$ in T
Step 12. for j = 1 to range_first do TypeVerify($n_i$, $n_j$, T);
Step 13. for j = i to range_second do TypeVerify($n_i$, $n_j$, T);
Step 14.T = DuplicateCombinationDelete(T);
Step15.T = UnusedCombinationDelete(T);
```

```
InsertPseudoType(L):

Step 1. for i = ($b_{min}$+1) to ($b_{max}$-1)

Step 2. if (L does not contain a type whose bit width is equal to i )

Step 3. insert a pseudo type $type_j$ with bit width i to L;
```

```
InsertChildrens(n, $n_1$, $n_2$):

Step1. n.left_child $\leftarrow$ $n_1$;

Step 2. n.right_child $\leftarrow$ $n_2$;
```

TypeVerify(n_1, n_2, T):

Step 1. bsum = $b(n_1) + b(n_2)$;

Step 2. if (L contains a type whose bit width is b_{sum})

Step 3. insert a new combination n whose bit width b_{sum} to T;

Step 4. InsertChildrens(n , n_1, n_2);

Figure 2. Shows the flow chart of the proposed algorithm. To transform the coordinate system equations (1) and(2) are used. The location of the point in original system is denoted by (x, y), and the new coordinate is denoted by (x', y'). Let x'' and y'' denote the transformed locations.

$$x' = \frac{x+y}{\sqrt{2}} => x'' = \sqrt{2} \cdot x' = x + y \qquad (1)$$

$$y' = \frac{-x+y}{\sqrt{2}} => y'' = \sqrt{2} \cdot y' = -x + y. \qquad (2)$$

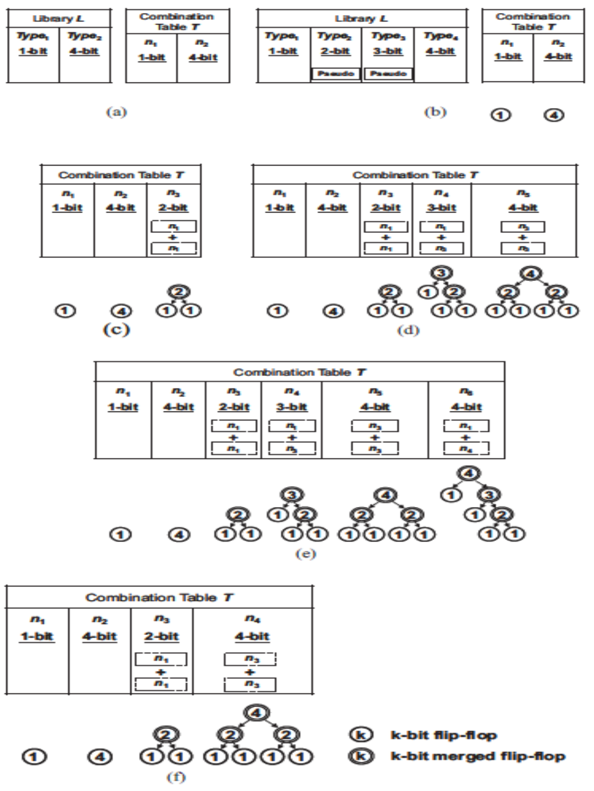

Figure 3. Example of building the combination table. (a)The library and combination table (b) Insertion of pseudo type (c) combine two n1s and form n2. (d)From n1 and n3 n5 is obtained. (e)From n1 and n4 n6 is obtained (f) Final combination table.

The flip flops can be replaced by new flip flops if and only if the new flip flops are present in the library. The combination table keeps the record of all the flip flops that can be replaced by new flip flops. The flip flops are replaced gradually according to the order of the combinations of flip flops on the table. While doing the replacement only one combination of flip flops must be considered each time. To prepare the combination table the concept of pseudo type is used. It is shown in Algorithm. A binary tree is used to represent one combination of simplicity. Each node in the tree denotes one type of flip flop in the given library. The types of flip flops denoted by the leaves constitute the type of flip flop in the root.

In the case of each node the bit width of the flip flop equals to the sum of bit width of left and right child. The combination is denoted by n_i and b (n_i) denotes its bit width. In the beginning of the algorithm the combination n_i is initialized for all flip flops in the given library. To represent each code the concept of pseudo type is included. The pseudo type includes those flip flop combinations that are not provided by the library.

The function **InsertPseudoType** is used to create pseudo type as shown in Algorithm. Let b_{max} and b_{min} the maximum and minimum bit width of flip-flops in library. The function **InsertPseudoType** is used to insert the pseudo types that can have bit widths more than b_{min} and lesser than b_{max}. The pseudo types are the one which is not provided by the library. After that all the combinations are sorted out in the ascending order. Finally try to combine each combination to get a new one. To check the feasibility of the combination the function **TypeVerify** is used. If the combination is feasible it is added to the table.

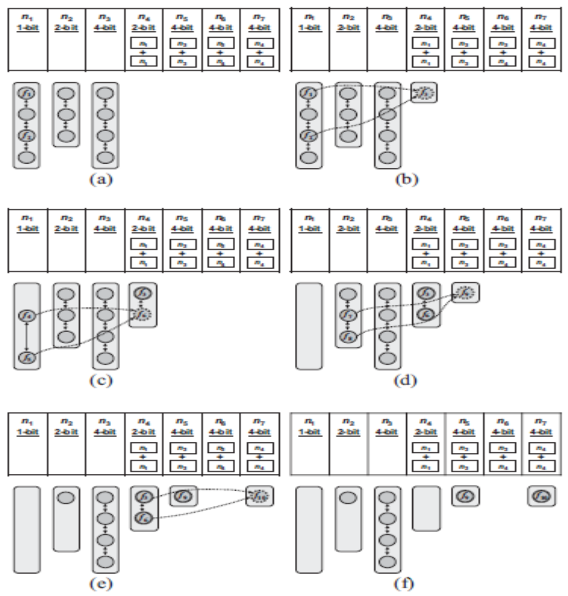

Figure 4. Example for replacement of flip flops. (a) flip flops before merging(b)f1 and f2 replaced by f3 (c) f4 and f5 replaced by f6 (d) f7 and f8 replaced by f9(e)f3 and f6 replaced f10(f) flip-flops after merging.

In **TypeVerify** the bits of the combinations are added and stored in the b_{sum}. This b_{sum} is further added to the remaining combinations. After completing all these procedures there may be some unused or duplicate combinations. This unused and duplicate combination is deleted. In order to delete those two functions **DuplicatecombinationDelete** and **Unusedcombination Delete** are used.

Suppose the library only provides the two types of flip flops with bit widths 1 and 4 as in Figure 4 (a) and then initialize the two combinations n1 and n2 to represent the two flip flops. The function **InsertPseudoType** is used next to check the flip flop type with bit width between 1 and 4 exist or not. After that the flip flop with bit widths 2 and 3 are added to the table. For each combination in the table a binary tree with 0-level is built. The root of the binary tree denotes the combination. From the Figure 4 by combining the two 1-bit flip flops a new combination n3 is obtained. The n4 is obtained by combining n1 and n3, and n5 by combining two n3s. Finally n6 is obtained by combining n1 and n4. All the possible combinations are shown in Figure 4(e). In these combinations n5 and n6 are duplicate, because both representing the same condition. So n6 can be deleted to speed up the program. In this n4 is an unused combination so it can be eliminated.

B. Merge Flip Flops

```
InsertPseudoType(L):
Step 1 for eachtype$_j$ in L do
Step 2 PseudoTypeVerifyInsertion( type$_j$, L) ;
PseudoTypeVerifyInsertion( type$_j$, L):
Step 1 if (mod (b(type$_j$) /2) == 0)
Step 2 b1 = [b(type$_j$)/2], b2 = [b(type$_j$)/2];
Step 3 else
Step 4 b1 = [b(type$_j$)/2], b2 = b(type$_j$) - [b(type$_j$)/2];
Step 5 for i = 1 to 2
Step 6 if ((bi > $b_{min}$) && (L does not contain a type whose bit width is
        equal to bi))
Step 7 insert a pseudo type type$_j$ with bit width bi to L;
Step 8 PseudoTypeVerifyInsertion(type$_j$, L);
```

In the case of combination table all the combination of the flip flops must be entered. i.e. both the flip flops which are present in the table as well as the combination which are not present. But inserting the not present flip flops consumes more time so only some types of flip flops are inserted. To insert a flip flop combination ni whose type is type j, only those flip flops whose bit widths are (b(type j)/2) and (b(type j)– b(type j)/2) should exist. Algorithm shows the enhanced procedure to insert flip flops of pseudo types. In the case of type j the function **PseudoTypeVerifyInsertion** checks the flip flops whose bit widths are [b (type j)/2] and add to the library if it is not exist in it.

C. Merge all Flip Flops in Subregion

```
Step 1 For each combination n in a combination table
Step 2 lright ←the lists for the combination of the right-child of n;
Step 3 lleft← the list for the combination of the left-child of n;
Step 4 for each flip flop $f_j$ in the list $l_{left}$
Step 5 $c_{best} \leftarrow \infty$
Step 6 for each flip flop $f_j$ in the list $l_{right}$
Step 7 if ($f_i$ and $f_j$ can be merged) then
Step 8 Compute cost c;
Step 9 If c<$c_{best}$ then $c_{best}$ =c,$f_{best} = f_j$
Step 9 end
Step 10 end
Step 11 Add flip flop f' merged from $f_i$ $f_{best}$ to combination n;
Step 12 Remove $f_i$ from $l_{left}$;
Step 13 Remove the flip flop recorded by $f_{best}$ from $l_{right}$
Step 14 end
Step 15 end
```

Algorithm shows the procedure to get a new flip flop corresponding to the combination n. The binary tree helps to find out the combinations associated with the left and right child of the root. Hence, the flip flops in the lists, named l_{left} and l_{right}, linked below the combinations of its left child and its right child are checked. Then, for each flip flop in l_{left}, the best flip flop f_{best} in l_{right}, which is the flip-flop that can be merged with the smallest cost recorded in C_{best}, is picked. Finally, add a new flip flop in the list of the combination and remove the picked flip flops.

Results and Discussions

In this section shows the results and discussions. This we have merged the 1 bit and 2 bit flip flop to create the 4 bit and using this default library is to create the other possible combinations. After that, introduce the application module as 10-bit down counter using multi bit flip flops, after looking the combination table. The experimental results obtained are shown in table I. Usually the conventional counter in power 67mW but by using the proposed multi-bit method it is found that 52mW. The maximum combinational path delay is 3.967ns but in proposed system, it is reduced to 1.949ns. The proposed system has memory usage of 152MB whereas in existing system it is 160MB. The experiment is carried out using Xilinx ISE Design suite 9.1 keeping Spartan 3E-XC3S500 as the target device.

TABLE I. COMPARISON

PARAMETERS	**CONVENTIONAL COUNTER**	**COUNTER USING MULTI-BIT FLIP FLOPS**
Power	67mW	52mW
Maximum combinational path delay	3.967ns	1.949ns
Average combinational path delay	2.03ns	1.040ns
Peak memory usage	160MB	152MB
Maximum pin delay usage	3.54ns	2.72ns

The 2^{10} Counter contains inputs with clock period and outputs and the counter are sequenced in the following order 2^{10}-1, 1023, 1022 ….. 0. The same clock pulses are applied to the clock inputs to all flip flops simultaneously. Finally the 10-bit counter circuit is achieved by using verilog design and the same is simulated by Xilinx ISE and the simulation result shown by Figure 5.

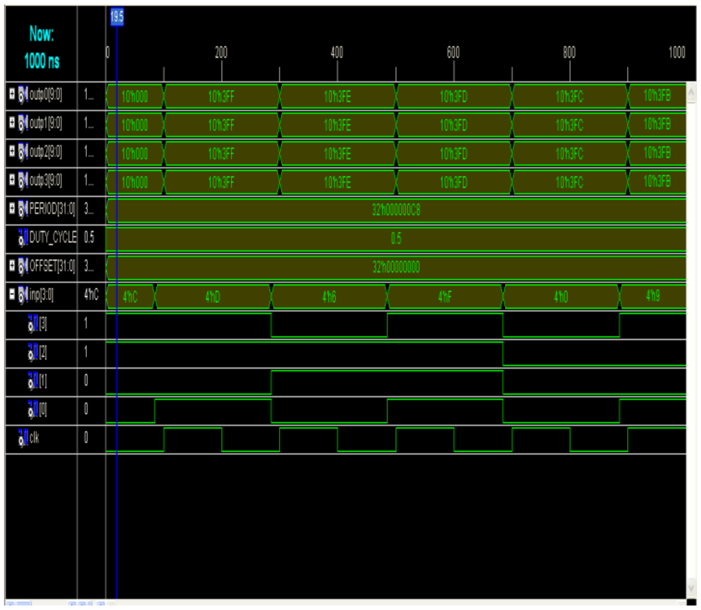

Figure 5. Simulation Result

Conclusion

The proposed algorithm for multi-bit flip replacement reduces the power, complexity, total wire length and area in digital integrated circuit design. The flip flop replacements are depending on the combination table. The pseudo type is introduced to help all possible combinational flip flops in the combination table. The combination table is used to replace the flip flops in sub regions. By using this technique, the power consumption and the wire length are reduced. It will be used in calculators and data processing systems.

References

[1] *Anand Rajaram, Jiang Hu, and Rabi Mahapatra (2006) "Reducing Clock Skew Variability via Crosslinks," IEEE Trans. CAD Integr. Syst., vol. 25, no. 6, pp. 1176-1182.*

[2] *Aurangzeb Khan, Philip Watson, George Kuo (2006) "A 90-nm Power Optimization Methodology With Application to the ARM 1136JF-S Microprocessor," IEEE J. Solid-State Circuits, vol. 41, no .8, pp. 1707–1717.*

[3] *Brenner.U and Vygen.J, (2004) "Legalizing a placement with minimum total movement," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 23, no. 12, pp. 1597–1613.*

ISSN: 2277-9655
Impact Factor: 1.852

[4] *Chang .S.J, Shyu.Y.T, Lin.J.M, Lin.C.W, and Lin.Y.Z, (2013)"Effective and efficient approach for power reduction by using multi-bit flip-flops," IEEE Trans. VLSI Syst., vol. 21, pp. 624–635.*

[5] *Chang.C.L, Jiang.I.H-R, Yang.Y.M, (2012) "INTEGRA: Fast multi-bit flip-flop clustering for clock power saving based on interval graphs," in Proc. ISPD, pp. 115–121.*

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

[7] *Chien-Cheng Yu, (2008) "Low-Power Double Edge-Triggered Flip-Flop Circuit Design," in Proc. Int. Conf.ICICIC6.*

[8] *Luo.T, Newmark.D, and Pan.D.Z, (2008) "Total power optimization combining placement, sizing and multi-Vt through slack distribution management," in Proc. IEEE/ACM Asia South Pacific Des. Autom. Conf., pp. 352-357.*

[9] *Mahmoodi.H Tirumalashetty.V Cooke.M and Roy.K, (2009) "Ultra low –power clocking scheme using energy recovery and clock gating," IEEE Trans. Very Large Scale Integr. Syst., vol. 17, no. 1, pp. 33–44.*

[10] *Naik.S and Chandel.R, (2010) "Design of a low power flip-flop using CMOS deep sub micron technology," in Proc. Int. Conf. Recent Trends Inform., Telecommun. Comput., pp. 253–256.*

[11] *Seyedi.A.S, Rasouli.S.H, Amirabadi.A, and Afzali-Kusha.A, (2006) "Low power low leakage clock gated static pulsed flip-flop," in Proc. IEEE Int. Symp. Circuits Syst., pp. 3658–3661.*

[12] *Shao-Huan Wang, Yu-Yi Liang, Tien-Yu Kuo, and Wai-Kei Mak, (2012)"Power-Driven Flip-Flop Merging and Relocation," IEEE Trans. CAD Integr. Syst., vol. 31, no. 2, pp.180-191.*

[13] *Wilke.G and Reis.R, (2008)"A new clock mesh buffer sizing methodology for skew and power reduction," in Proc. IEEE Comput. Soc. Annu. Symp. VLSI, pp. 227–232.*

[14] *Xu.H, Vemuri.R and Jone.W, (2011) "Dynamic characteristics of power gating during mode transition," IEEE Trans. Very Large Scale Integr. Syst., vol. 19, no. 2, pp. 237–249.*

[15] *Yan. J.-T. and Chen. Z.-W, (2010) "Construction of constrained multi-bit flip-flops for clock power reduction," in Proc. IEEE Int. Conf. Green Circuits Syst., pp. 675–678.*

[16] *Yu.C.C, (2007) "Design of low-power double edge-triggered flip-flop circuit," in Proc. IEEE Conf. Indust. Electron. Applicat, pp. 2054– 2057.*