Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

Optimal utilization of adjustable delay clock buffers for timing correction in designs with multiple power modes

Juyeon Kim, Deokjin Joo, Taewhan Kim*

Department of Electrical and Computer Engineering, Seoul National University, Seoul, Korea

ARTICLE INFO

Article history:
Received 18 March 2015
Received in revised form
15 June 2015
Accepted 30 August 2015
Available online 11 September 2015

Keywords:
Adjustable delay buffers
Timing
Clock skew
Power modes
Resource allocation
Design optimization

ABSTRACT

Meeting clock skew constraint is one of the most important tasks in the synthesis of clock trees. Moreover, the problem becomes much hard to tackle as the delay of clock signals varies dynamically during execution. Recently, it is shown that adjustable delay buffer (ADB) whose delay can be adjusted dynamically can solve the clock skew variation problem effectively. However, inserting ADBs requires non-negligible area and control overhead. Thus, all previous works have invariably aimed at minimizing the number of ADBs to be inserted, particularly under the environment of multiple power modes in which the operating voltage applied to some modules varies as the power mode changes. In this work, unlike the previous works which have solved the ADB minimization problem heuristically or locally optimally, we propose an elegant and easily adoptable solution to overcome the limitation of the previous works. Precisely, we propose an $O(n \log n)$ time (bottom-up traversal) algorithm that (1) optimally solves the problem of minimizing the number of ADBs to be allocated with continuous delay of ADBs and (2) enables solving the ADB allocation problem with discrete delay of ADBs to be greatly simple and predictable. In addition, we propose (3) a systematic solution to an important extension to the problem of buffer sizing combined with the ADB allocation to further reduce the ADBs to be used. The experimental results on benchmark circuits show that compared to the results produced by the best known ADB allocation algorithm, our proposed algorithm uses, on average under 30–50 ps clock skew bound, 13.5% and 15.8% fewer numbers of ADBs for continuous and discrete ADB delays, respectively. In addition, when buffer sizing is integrated, our algorithm uses 31.7% and 31.3% fewer numbers of ADBs, even reducing the area of ADBs and buffers by 15.0% and 16.3% for continuous and discrete ADB delays, respectively.

1. Introduction

Clock is one of the most important signals on a chip of synchronous based system, as all the synchronous components on the chip such as flip-flops (FFs) rely on it. Clock tree is a commonly used structure of circuits that distributes the clock signal from the clock source to all the *clock sinks* (e.g., FFs and latches), where the clock signal is required. It is imperative that the maximum of the arrival time difference between the clock sinks, which is known as *global clock skew*, should be maintained under a certain bounded value typically within 10% of the clock period, as a large clock skew may cause timing violation on the circuits. (If no confusion occurs, the *global clock skew* is simply referred to as *clock skew* in this presentation.)

Many research works on the clock tree optimization such as clock routing, clock buffer insertion/sizing, and wire sizing have been performed to control or minimize the clock skew [1–7]. While these approaches were effective, advanced low power design techniques introduced new challenges to the clock skew control problem. Specifically, for multiple power mode designs, where the supply voltage to the circuit components varies dynamically depending on modes, the clock arrival time also varies accordingly.

Even though the previous works can consider the clock skew constraint on every power mode, it would be highly likely that the resulting clock tree uses a substantially long wirelength or there exists no clock tree that satisfies the clock skew constraint on every power mode. On the other hand, post-silicon tuning (e.g., [8–11]) such as inserting adjustable delay buffers (ADBs) is a widely used method to deal with the timing problem caused by process and environment variations. Because the delay of an ADB can be controlled by its delay control inputs [12], the clock skew

* Corresponding author.
E-mail address: tkim@ssl.snu.ac.kr (T. Kim).

http://dx.doi.org/10.1016/j.vlsi.2015.08.005
0167-9260/© 2015 Elsevier B.V. All rights reserved.

variation caused by process variation can be tuned by properly inserting ADBs after the manufacturing stage has been completed. The idea of using ADBs in multiple power modes is to replace some of normal clock buffers with ADBs so that the clock skew constraint on each power mode can be met; when the power mode changes during execution, e.g., from power mode *mode-1* to power mode *mode-2*, the delays of ADBs in clock tree that have been adjusted under *mode-1* are readjusted to meet the clock skew constraint under *mode-2*. Since ADB logic component is much bigger than normal buffer and it requires control line as well as switching logic, the set of related problems to be solved for the ADB-based clock skew optimization in multiple power modes are allocating a minimum number of ADBs, finding the normal buffers (or locations) in the clock tree that are to be replaced by ADBs, and determining the delay value of ADBs to be assigned on each power mode. We call these problems collectively *ADB allocation problem*.

Su et al. [13,14] proposed a linear-time optimal algorithm for the delay assignment problem and exploits the algorithm to solve the rest of two subproblems of the ADB allocation problem heuristically in a greedy manner. Lin et al. [15] proposed an efficient algorithm of two-stage approach which performs a top-down ADB allocation followed by a bottom-up ADB elimination. Even though the approach reduces the run time over that in [13,14], it still does not guarantee an optimality of ADB allocation. Lim and Kim [16] proposed a linear-time algorithm for the ADB allocation problem where they solved the problem optimally for *each* power mode. However, merely collecting the optimal results on individual power modes does not mean globally optimal for all power modes. In this work, we revisit the ADB allocation problem and propose a set of solutions to overcome the limitation of the previous works. More precisely, we propose (1) an $O(n \log n)$ time algorithm that optimally solves the problem of minimizing the number of ADBs to be allocated for all power modes with continuous delay of ADBs and (2) enables solving the ADB allocation problem with discrete delay of ADBs to be greatly simple and predictable. In addition, we propose an effective solution to an important extended problem: (3) the ADB allocation problem combined with buffer sizing. (A preliminary version, which contains concise descriptions and no proofs, of our work can be found in [17].)

It should be mentioned that the work in [16] is completely different from our proposed optimal algorithm by a simple reasoning: For example, [16] requires optimally two ADBs, each in clock nodes 1 and 2, for power mode 1 while requiring optimally two ADBs, each in nodes 3 and 4, for power mode 2. Thus, the combined ADB allocation is four ADBs, each in nodes 1, 2, 3, and 4 to meet timing for all power modes. On the other hand, ours produces an optimal ADB allocation result considering power modes *all together*. The globally optimal allocation may be three ADBs (i.e., not four ADBs), say, each in nodes 1–3. This reasoning clearly foresees that as the number of power modes increases, the gap (i.e., ADB difference) between [16] and ours will increase.

The rest of the paper is organized as follows. Section 2 illustrates the structure of ADB implementation and shows an example of using ADBs for timing correction. Section 3 defines the ADB allocation problem and shows an example to motivate the work. Then, Section 4 proposes an optimal algorithm of ADB allocation with continuous delay values and a modification of the algorithm to support ADBs with discrete delay values. Section 5 proposes a solution to the extended problem of integrating buffer sizing into ADB allocation. Experimental results are provided in Section 6 to show the effectiveness of our proposed ADB allocation algorithms. Finally, a conclusion of the work given in Section 7.

2. ADB structure and example of ADB utilization

Fig. 1 shows the structure of a capacitor bank based implementation of ADB [18]. This implementation of a well-known capacitor bank based ADB consists of two inverters at the input and output ports, and in the middle there is an array of capacitors with switch transistors attached. The switches are controlled by the capacitor bank controller, which controls the number of active capacitors according to the control bits. Activating more capacitors increases the total capacitance between the two inverters, which in turn increases the signal propagation delay between the input and output ports. Inverter based ADB [19] is another implementation structure of ADB, but the adjustable delay values are less fine-grained than that of the capacitor bank based one.

Fig. 2(a) shows an example of clock tree that has four sinks *s*1, *s*2, *s*3, and *s*4, two ADBs replacing two clock buffers, and ADB control logic. Suppose there are two power modes *mode-1* and *mode-2* in this design. Then, the two numbers separated by a slash next to each sink indicate the clock signal arrival times in *mode-1* and *mode-2*. When the clock skew bound is given to 10, the clock tree causes clock skew violations in both modes if ADBs were not used. With the replacement of two clock buffers by ADBs, the two numbers next to each ADB indicate the delay increments (simply called *delay values*) in *mode-1* and *mode-2*: The ADB on the left adds delay of 2 in *mode-1*, thus the clock signal arrival time at *s1* in *mode-1* becomes 6. Likewise, the ADB on the right adds delay of 3 in *mode-2*, increasing the arrival time at *s*3 in *mode-2–6*. To control the ADBs' delay, a mode signal is required. In addition, depending on the implementation of the ADBs, control logic that

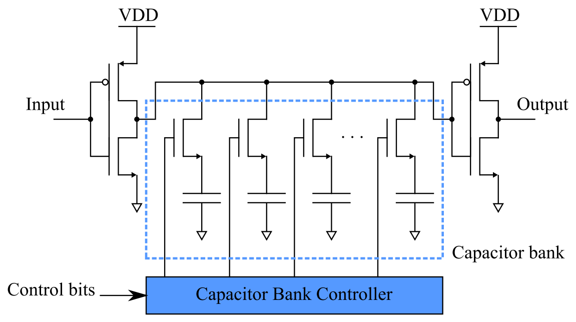

Fig. 1. The structure of a capacitor bank based ADB. The capacitor bank adjusts signal propagation delay from the input to output ports.

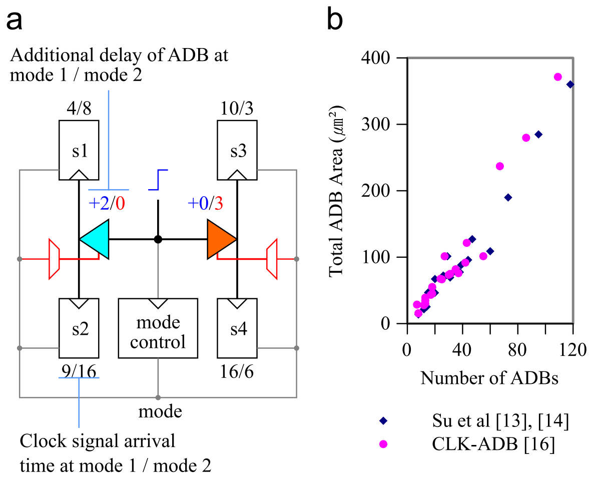

Fig. 2. (a) An example of clock tree with the replacement of two clock buffers with ADBs. (b) The relationship between the number of ADBs and the total ADB area (including logic overhead) used by [13,14,16].