Slack Budgeting and Slack to Length Converting for Multi-bit Flip-Flop Merging

Chia-Chieh Lu
Computer Science and Engineering
Yuan Ze University
Chung-Li, Taiwan

Rung-Bin Lin
Computer Science and Engineering
Yuan Ze University
Chung-Li, Taiwan

Abstract—**In this paper we propose a flexible slack budgeting approach for post-placement multi-bit flip-flop (MBFF) merging. Our approach considers existing wiring topology and flip-flop delay changes for achieving more accurate slack budgeting. Besides, we propose a slack-to-length converting approach to translating timing slack into equivalent wire length for simplifying a merging process. We also develop a merging method to evaluate our slack budgeting approach. Our slack budgeting and MBFF merging programs are fully integrated into an industrial design flow. Experimental results show that our approach on average achieves 3.4% area saving, 50% clock tree power saving, and 5.3% total power saving.**

Keywords—Multi-bit flip-flop; slack budgeting; low power

I. Introduction

Slack or timing budgeting [1-8] distributes timing slack of a path in a circuit to the circuit elements on the path. Timing slack of a path is a difference between clock period and its path delay. During early 90's, as interconnect delay became relatively larger and less predictable than logic gate delay, slack (delay) budgets were set up for interconnects to constrain a placement or routing process for timing closure [4,9]. Later, slacks assigned to circuit elements had an extended use for timing, power, and area optimization. Slack budgeting approaches thus vary depending on the kinds of optimization performed. Accordingly, timing slacks must be translated into various forms to facilitate different kinds of optimization. For instance, timing slack of a logic gate can be translated into a change of transistor sizes for gate sizing, a change of supply voltage for multi-supply voltage designs [5], a change of transistor threshold voltage for multi-threshold voltage designs [10,11], a change of flip-flop position in the circuit (retiming) [12]. Yet another example, timing slack of a net can be translated into a change of net length for placement and routing to achieve timing closure [9]. Converting timing slack into a change of something is not intuitively simple because of nonlinear dependence of logic gate delay on capacitance loading, input slew, threshold voltage or some other things such as requiring insertion of level converters for multi-supply voltage designs.

Zero-slack algorithm (ZSA) [1] was an early slack budgeting algorithm. It finds repeatedly a path that has the minimum positive slack and averagely distributes the slack to all logic gates on the path. *Minimax* [2] argued that timing slack should be distributed based on physical and electrical characteristics of logic gates. Subsequently, many slack budgeting algorithms were proposed [4-8]. Most of them try to maximize the total slack distributed to the circuit elements. Even clock skew is employed to increase the total distributed slack [8]. The question is that total distributed slack should not be blindly maximized regardless of targeted applications. Otherwise, slacks assigned to some circuit elements may not be effectively exploited for optimization.

Recently, combining flip-flops (FFs) in close proximity into a multi-bit flip-flop (MBFF) after placement has been pursued to reduce clock tree power [12-22]. This task is commonly called MBFF merging. All single-bit FFs in a MBFF are designed to share the same clock driver. A MBFF should be relocated to a new place without causing any timing violation. Previous work for this usually assumed that timing slacks of flip-flops in a MBFF are given beforehand and can be easily converted into equivalent wire lengths to confine MBFF relocation. However, distributing path slack and converting timing slack into wire length for MBFF merging is not trivial. In this work, we proposed a slack budgeting approach to generating timing slacks for the nets connected to flip-flops. We also develop a method to convert slack into wire length. Our major contributions are as follows.

- Our slack budgeting approach considers the effect of assigning a slack to one flip-flop on the amount of delay added to the path leading to another flip-flops.
- Our slack-to-length converting approach analytically includes load dependent delay of driving cells into delay calculation.
- We present a MBFF merging algorithm to evaluate our slack budgeting and slack-to-length converting approaches.
- We integrate our slack budgeting and MBFF merging tools into an industrial standard cell design flow. Our approach on average achieves 3.4% area saving, 50% clock tree power saving, and 5.3% total power saving for tight timing requirement designs. If timing requirement is relaxed by 10%, we attain 3.7% area saving, 60% clock tree power saving, and 7.2% total power saving. All the results are obtained with even better timing.

This article is organized as follows. Section II gives a problem definition. Section III presents a case study of MBFF. Section IV depicts our slack budgeting approach. Section V presents our slack-to-length converting method. Section VI illustrates an application of slack budgeting for post-placement MBFF merging. Section VII presents some experimental results. Section VIII draws a conclusion.

II. Problem Definition

In this section we briefly introduce the concept of MBFF merging and how it is related to slack budgeting. We then give a problem definition for our work.

A MBFF is a type of flip-flop that can store multiple bits. A 1-bit flip-flop (FF) usually has two latches and a clock driver formed by two inverters. A MBFF can be built by combining two or more 1-bit FFs that share the clock driver as shown Fig. 1. It can be used to reduce clock tree power and chip area. The problem is that which FFs should be combined and where to place the MBFF without causing any timing violation. As such, timing slack should be allocated to the nets connected to FFs for restricting the placement of MBFFs. Hence, our slack budgeting problem is defined as follows.

Given an already placed design and a targeted clock period, determine the timing slacks assigned to the input nets and output nets of each 1-bit FF so that any path delay including the given timing slacks and flip-flop setup time will not exceed the clock period.

Without loss of generality, a 1-bit FF is assumed to have a single output. Fig. 2 shows placement of a MBFF.

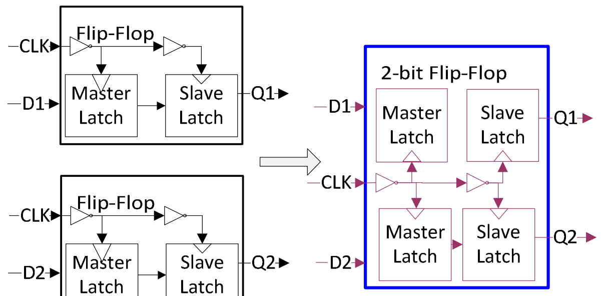

Fig. 1. Forming a two-bit FF.

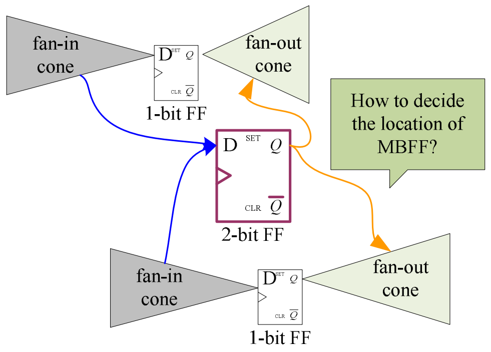

Fig. 2. Placement of a MBFF.

After we obtain timing slack for a net, we have to convert it into equivalent wire length to facilitate placement of a MBFF. A simple conversion approach will not work if it does not consider driving capability and loading changes of the logic element that drives the nets involving MBFF merging.

The two wire lengths derived from the timing slacks given to the input and output nets of a FF will define a ***roaming region***. A FF can move around within its roaming region without causing any timing violation whether it be merged with other FFs. Basically, two FFs can be merged if their roaming regions are intersected. This non-empty region is called ***destined region*** as shown in Fig. 3.

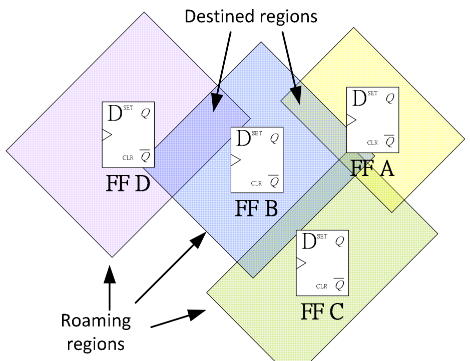

Fig. 3. Roaming regions and destined regions.

III. Multi-Bit Flip-Flops

We have designed and implemented k-bit MBFFs for $k = 1, \ldots, 8$ based on UMC 90nm technology. All the flip-flops in a MBFF share the same clock driver. In this work, a k-bit MBFF is denoted by MBFFk. Our MBFFs have the same cell height and power/ground width as that of the cells in an industrial standard cell library provided by Faraday Technology. Figure 4 shows the layout of our MBFF2. Clock driver is located at the center. Table I shows that area reduction ratio per bit increases with the number of bits, being a consequence of sharing a clock driver. We observe that the fall and rise delays of MBFFk's are increased with the number of bits, but their fall transition times almost remain the same. We also observe that setup time and hold time have more complicate variations with increasing number of bits. Hence, a slack-to-length converting approach should consider all these timing complications.

Clock driver
DIFF
Contact
Poly
Metal 1
Via 12
Metal 2

Fig. 4. Layout of MBFF2.

TABLE I. AREA OF OUR MBFFK'S

Bit #	Width (um)	Width/Bit #	Reduction ratio
1	8.830	8.8300	0.00%
2	16.390	8.1950	7.19%
3	23.585	7.8617	10.97%
4	30.995	7.7488	12.25%
5	38.430	7.6860	12.96%
6	45.790	7.6317	13.57%
7	53.225	7.6036	13.89%
8	60.585	7.5731	14.23%

Internal power of a MBFFk has two important parts: clock pin power and output pin power. Clock pin power increases with the number of bits. However, clock pin power per bit does not follow this trait as shown in Fig. 5. Note that clock pin power per bit of MBFF2, MBFF3, and MBFF4 is smaller than that of MBFF1 whereas clock pin power per bit of MBFF5, MBFF6, MBFF7 and MBFF8 is generally larger than that of MBFF1. Clearly, from power minimization point of view, merging more than five FFs into a MBFF will not have a maximum power reduction. Nevertheless, this situation could change if FFs and MBFFs are designed otherwise.

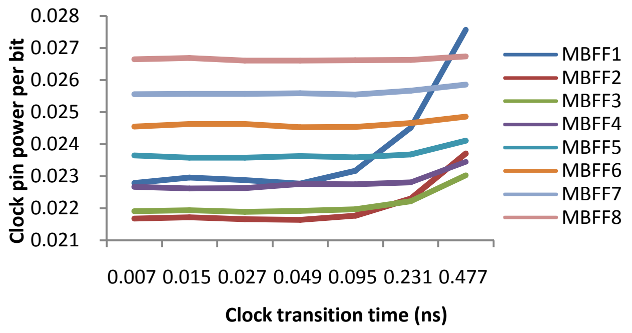

Fig. 5. Clock pin internal power (pJ) per bit.

IV. Slack Budgeting

Slack budgeting determines how much slack should be assigned to a FF. Since the input of a FF serves as the sink of a path and its output serves as the source of a path, slacks are respectively assigned to the nets connected to the input and output of a FF. Hence, to perform slack budgeting we must find out the path whose delay determines the path slack of a source-sink pair and then distribute the path slack to the source side (called *Q-side* for simplicity) and the sink side (*D-side*). Formally, we create a graph $G = (V, E)$ where a vertex represents either a Q-side or a D-side. All Q-side vertices form a subset V_{source} and all D-side vertices form a subset V_{sink} and $V = V_{source} \cup V_{sink}$. We create a directed edge $(v_i, v_j), v_i \in V_{source}$ and $v_j \in V_{sink}$ if there is a signal path from the Q-side associated with v_i to the D-side associated with v_j. We also create an edge (v_i, v_j) for $v_i,\ v_j \in V_{sink}$ if there exists a signal path via the D-side associated with v_i and then immediately reaching the D-side associated with v_j. Such a graph is called ***SS-graph*** in this article. Note that a *SS-graph* has no cycle because no path can start from a D-side and terminate at a Q-side. Figure 7 shows the *SS-graph* for the circuit in Fig. 6. There is an edge from FF1/D and FF2/D. An edge of this sort is employed to model how a slack allocated to FF1/D affects the slack being allocated to FF2/D. We call such a situation ***branch-before-flip-flop*** (BBFF).

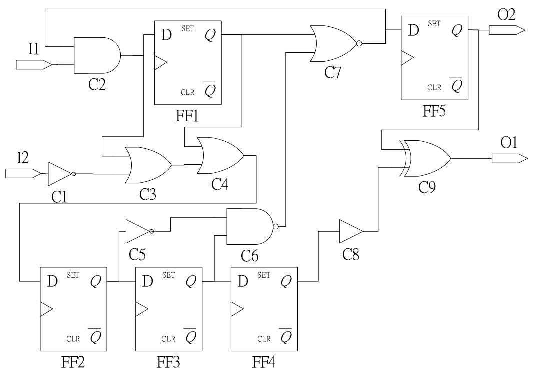

Fig. 6. A circuit for slack budgeting.

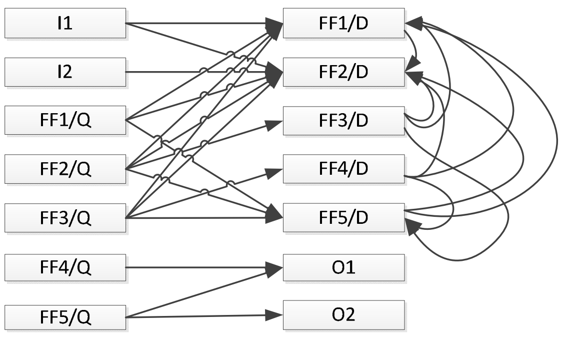

Fig. 7. A SS-graph representing relations between FFs.

Based on a SS-graph, our slack budgeting problem can be formulated as a linear programming model as follows.

Objective function:

$$\text{Maximize } \sum_{v_i \in V_{sink}} w_i \Delta d_i + \sum_{v_j \in V_{source}} w_j \Delta d_j \qquad (1)$$

where w_i is a weight for vertex v_i. Δd_i is the timing slack allocated to vertex v_i. The vertices correspond to primary I/Os can be assigned a timing slack of zero. This objective function maximizes the total weighted slack. The constraints for the model are as follows:

Source-sink pair constraints:

For each path $p = (v_1, v_2)$, $v_1 \in V_{source}$ and $v_2 \in V_{sink}$ in a SS-graph, we have

$$\Delta d_1 + \Delta d_2 \le T - PD(p) - Setup_{v_2} \qquad (2)$$

where T is clock period. $PD(p)$ is the delay of the longest path in the circuit from the Q-side associated with v_1 to the D-side associated with v_2. $Setup_{v_2}$ is the setup time of the FF associated with v_2. $T - PD(p) - Setup_{v_2}$ is the path slack. We consider both rise and fall delays.

BBFF constraints:

For each path $p = (v_1, v_2, \dots, v_n)$ in a SS-graph for $n > 2, v_1 \in V_{source}$, $v_i \in V_{sink}, i = 2, \dots, n$, we have

$$\Delta d_1 + \Delta d_n \le T - PD(p) - \sum_{i=2}^{n-1} f(\Delta d_i) - Setup_{v_n} \qquad (3)$$

For each vertex v_i for $1 < i < n$, if v_i is assigned a slack of Δd_i, $f(\Delta d_i)$ will be the delay added to the path in a circuit from the Q-side associated with v_1 to the D-side associated with v_n. As shown in Fig. 8, a slack assigned to FF1/D (denoted by v_2) will incur a delay that should be added to the circuit path from FF1/Q (v_1) to FF2/D (v_3). The delay equals $f(\Delta d_2)$. We assume that a slack of Δd_2 assigned to FF1/D will correspond to a wire length increase of ΔL. We also assume that on-resistance of the driving cell is R_d and resistance of the path from the driving cell to FF1 for $\Delta L = 0$ is R_o. Clearly, we have a wire load increase of $c\Delta L$ and a wire resistance increase of $r\Delta L$ where r and c are resistance and capacitance per unit wire length, respectively. The increase of wire load is maximal when $\Delta d_2 = (R_d + R_o)c\Delta L + 0.5rc\Delta L^2$. Solving for ΔL we obtain

$$\Delta L = \frac{-(R_d+R_o)c+\sqrt{((R_d+R_o)c)^2+2rc\Delta d_2}}{rc} \qquad (4)$$

Then,

$$f(\Delta d_2) = (R_d + R_o)c\Delta L = \frac{-(R_d+R_o)^2c+(R_d+R_o)\sqrt{((R_d+R_o)c)^2+2rc\Delta d_2}}{r} \qquad (5)$$

Since $f(\Delta d_2)$ is a nonlinear function of Δd_2, we compute a linear approximation of it using Taylor series of $f(\Delta d_2)$ up to the first order of Δd_2. The linear approximation is shown in (6).

$$f(\Delta d_2) \cong \Delta d_2 \qquad (6)$$

Basically, we have $f(\Delta d_i) \le \Delta d_i$. Hence, $f(\Delta d_i) = \Delta d_i$ is a conservative approximation of $f(\Delta d_i)$.

Slack upper bound constraints:

The above constraints are sufficient to guarantee a solution. However, in order to avoid allocating too much slack to some vertices and too less to others, we have a constraint for each v_i,

$$\Delta d_i \le R_i B_i \qquad (7)$$

where B_i is the maximal timing slack that can be assigned to v_i, i.e., path slack associated with v_i. R_i is a given value between 0 and 1. Note that we can also set up a lower bound constraint, but this could result in no solution.

To create such a model, we have to implement a timing analysis tool. Our timing analysis tool assumes a tree topology obtained by FLUTE [23] for a net. The number of variables in our model is linear in terms of the number of FFs. The number of constraints is quadratic in terms of the number of FFs.

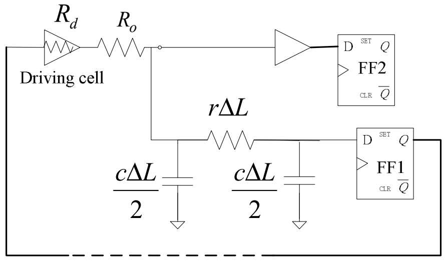

Fig. 8. Circuit model used to explain how slack assigned to a sink would become a delay added to another path.

V. Slack to Length Conversion

Our slack-to-length converting approach considers many issues related to the loading and delay changes at a Q-side and D-side after

MBFF merging. To be consistent with slack budgeting, we will use the same tree topologies for slack-to-length conversion. We define an anchor point on a tree as a place to which the wire segment associated with a D-side or Q-side can be attached, as shown in Fig. 9. An anchor point could be a Steiner point. If it is not, it will be located at the output of a driving cell or the input of a driven cell.

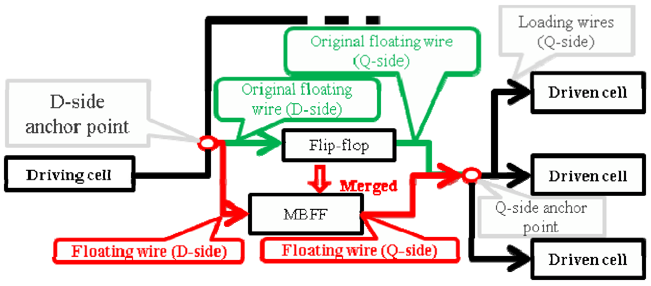

Fig. 9. Tree topology before and after merging.

No matter whether a driving cell is a FF or a combinational gate, its delay will change if the wire loading it drives is different. We perform a least square surface fitting of cell delays in terms of input slew and output loading for each cell and flip-flop. The fitted cell delay is as follows:

$$C\_Dly(w,l) = c_0 + c_1w + c_2w^2 + c_3l + c_4wl + c_5l^2 \qquad (8)$$

where w is the input transition time of the latest arriving signal at a driving cell. l is the total output capacitance of a driving cell, and $c_0 \sim c_5$ are coefficients. Since we can obtain input transition time of a driving cell after timing analysis, the delay of a driving cell is assumed only a function of its loading. Hence, (8) becomes a quadratic polynomial of load capacitance l as shown in (9).

$$C\_Dly(l) = (c_0 + c_1w + c_2w^2) + (c_3 + c_4w)l + c_5l^2 \qquad (9)$$

The error of surface fitting is on average quite small for most of the cells. Surface fitting is done only once per cell library.

A. Conversion at Q-Side

There are three delay components at Q-side.

- *Delay change from clock input to the output of a FF due to merging*: As mentioned in Section III, cell delay of a 1-bit FF would increase when it is merged to form a MBFF. Such a delay change is called $\Delta DFF(FF\_type)$.
- *Delay change on a driving cell or a FF due to floating wire length change*: Delay change due to a wire length change of ΔL is given in (10).
$$\Delta C\_Dly(\Delta L) = C\_Dly(l_{org} + r\Delta L) - C\_Dly(l_{org}) = k_c c^2 \Delta L^2 + (2l_{org}c\, k_c + j_c c)\,\Delta L = a_c\,\Delta L^2 + b_c \Delta L \qquad (10)$$
where $j_c = (c_3 + c_4w)$ and $k_c = c_5$ from (9). l_{org} is the original loading capacitance.
- *Delay change on the floating wire*: This delay change has two parts: delay change on the floating wire itself and delay change due to charging the downstream capacitance. Before we calculate delay change, we give a formula to calculate the delay caused by the floating wire. The delay $FWD\_Q(L)$ on the floating wire of length L is calculated based on the equivalent circuit in Fig. 10.
$$FWD\_Q(L) = 0.5\,(rcL^2 + (rC_v + R_vc)L + R_vC_v) + rC_xL + R_vC_x = a_wL^2 + b_wL + c_w \qquad (11)$$
where $a_w = 0.5rc$, $b_w = rC_x + 0.5(rC_v + R_vc)$, and $c_w = R_vC_x + 0.5R_vC_v$. C_x is downstream loading capacitance, R_v and C_v are via resistance and capacitance (assuming an L-shape floating wire). Hence, delay change on a floating wire due to a length change of ΔL is as follows.

$$\Delta FWD\_Q(\Delta L) = FWD\_Q(L_{org} + \Delta L) - FWD\_Q(L_{org}) = a_w\Delta L^2 + (2a_wL_{org} + b_w)\Delta L \qquad (12)$$

where L_{org} is the original length of a floating wire.

The delay change at a Q-side is as follows.

$$\Delta Dly\_Q(\Delta L) = \Delta C\_Dly(\Delta L) + \Delta FWD\_Q(\Delta L) + \Delta DFF(FF\_type) = (a_c + a_w)\,\Delta L^2 + (b_c + 2a_wL_{org} + b_w)\Delta L + \Delta DFF(FF\_type) = a_{delay}\Delta L^2 + b_{delay}\Delta L + c_{delay} \qquad (13)$$

where $a_{delay} = a_c + a_w$, $b_{delay} = b_c + 2a_wL_{org} + b_w$, and $c_{delay} = \Delta DFF(FF\_type)$. Assuming the slack assigned to a Q-side is S, we should have

$$a_{delay}\,\Delta L^2 + b_{delay}\Delta L + c_{delay} \le S \qquad (14)$$

Solving (14) with equality for ΔL, we obtain

$$\Delta L = \frac{-b_{delay} + \sqrt{b_{delay}^2 + 4a_{delay}(S - c_{delay})}}{2a_{delay}} \qquad (15)$$

ΔL is the maximum length into which a slack assigned to a Q-side can be converted.

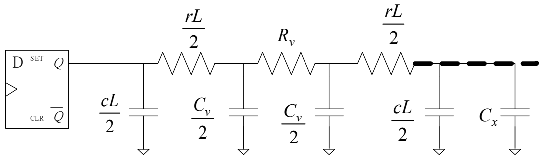

Fig. 10. Equivalent circuit for computing Q-side delay change.

B. Conversion at D-side

Similarly, we can convert a slack S assigned to a D-side based on the equivalent circuit in Fig. 11. The delay change at a D-side consists of only two components. The first component is the exactly same as that presented in (10). The second component is also given in (12) but with $b_w = R_c c + rC_p + 0.5(rC_v + R_vc)$ and $c_w = R_cC_v + R_vC_p + 0.5R_vC_v$. C_p is input pin capacitance of a FF. Using a similar approach, we obtain

$$\Delta L = \frac{-b_{delay} + \sqrt{b_{delay}^2 + 4a_{delay}S}}{2a_{delay}} \qquad (16)$$

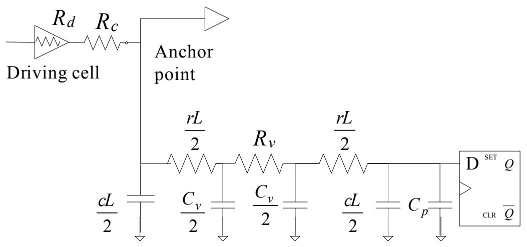

Fig. 11. Equivalent circuit for computing D-side delay change.

VI. Multi-Bit Flip-Flop Merging

We implement a MBFF merging algorithm as shown in Fig. 12 to evaluate our slack budgeting and slack-to-length converting approaches. We first convert slacks into wire length and calculate the roaming regions of FFs. We then build a relation graph where a vertex denotes a FF and an edge denotes a non-empty intersection of roaming regions of any two FFs. We then find a hardest merged node. In our implementation, a node with the smallest degree is the hardest merged node. We put the hardest merged node in a stack. The nodes which are closest to all the nodes in the stack will be subsequently inserted into the stack if their destined region is not empty. When the number of nodes in a stack reaches a maximally allowed number or no nodes can be inserted, we pop all nodes out of the stack and combine them into a MBFF. This process is repeated until all nodes are investigated.

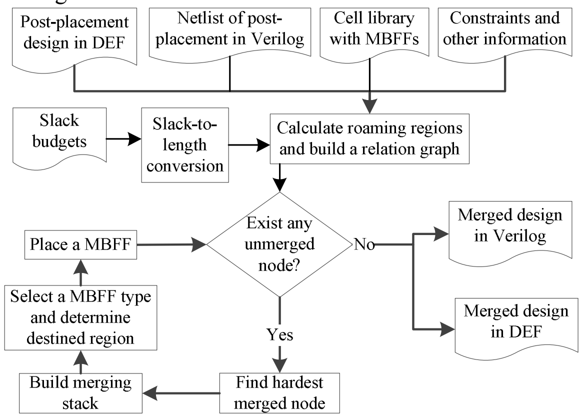
Post-placement design in DEF
Netlist of post-placement in Verilog
Cell library with MBFFs
Constraints and other information
Slack budgets
Slack-to-length conversion
Calculate roaming regions and build a relation graph
Exist any unmerged node?
No
Yes
Merged design in Verilog
Merged design in DEF
Place a MBFF
Select a MBFF type and determine destined region
Build merging stack
Find hardest merged node

Fig. 12. MBFF merging method.

Discussion:

We can apply our slack budgeting and slack-to-length converting approaches dynamically during MBFF merging. If slacks assigned to some FFs are not fully consumed, slack re-budgeting can be invoked. In some situations, slack-to-length converting has to be redone to consider the effects of input slew change at some driving cells in the middle of MBFF merging.

VII. Experimental Results

We use C++ to implement timing analysis, slack budgeting and MBFF merging methods, IBM ILOG CPLEX Optimizer to solve linear programming models, and Python to implement slack-to-length conversion. We use some ITC99 benchmarks in Table II for our experiments. We use Synopsys Design Compiler to synthesize benchmarks and Cadence SoC Encounter to do physical design and timing/power analysis. The entire flow runs on Intel Xeon 2.33GHz with 32 GB memory. In Table II, *BBFF sink* denotes the number of sinks, each of which has a BBFF situation. Clock period of each circuit is taken from the longest path delay of the circuit which has been optimized for tight timing requirement during logic synthesis. UMC 90nm technology is used for designing MBFFs.

In our implementation, the weighing factor in (1) is set to one. The upper bound ratio in (7) is set to 0.1, 0.2, 0.3, …, 1.0, respectively. Fig. 13 shows how total slack of b17 and b19 increases with upper bound ratio. Interestingly, the curves for b14 and b15 are similar to those for b17, i.e., sources obtaining much more slack than sinks. On the other hand, the curves for b18, b20, b21, and b22 are similar to those for b19, i.e., up to an upper bound ratio of 0.7 or 0.8, sources and sinks obtaining a similar amount of slack. It is also interesting to see that sources get more slacks than sinks. **Runtime for slack budgeting** including the time for timing analysis, building models, and solving models is about 11 minutes for the largest circuit b19. Runtime for slack-to-length conversion takes about 13 minutes for b19. Not shown here is that similar results are obtained for the cases of relaxing clock period by 10%. In the following, results associated with not relaxing clock period will be tagged with 1.0*clock period whereas those associated with relaxation will be tagged with 1.1*clock period.

Figure 14 shows the number of FFs after MBFF merging using slacks generated with different upper bound ratios. Runtime taken by MBFF merging for b19 on average is about 6 minutes. Clearly, the

Table II. Benchmark circuits.

Test cases	Clock period	# FFs	Total cells	PI	PO	BBFF sink
b14	2.2	215	37370	76	54	3
b15	1.5	417	41091	185	70	0
b17	2.1	1,317	136183	61	97	2
b18	3.4	3,020	384438	130	23	10
b19	4.3	6,042	747711	139	28	20
b20	2.1	430	87625	152	22	6
b21	1.9	430	78826	150	22	6
b22	2.0	613	127504	181	22	9

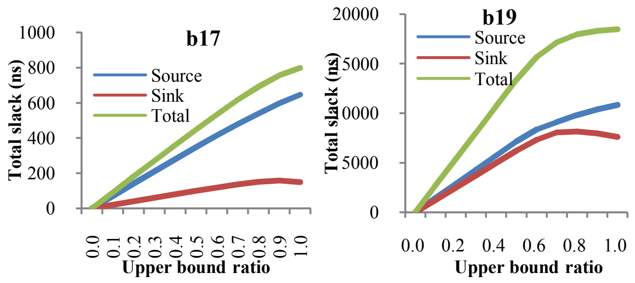

Fig. 13. Total slack obtained with different upper bound ratios.

number of FFs, including 1-bit FFs, decreases with increasing upper bound ratio up to 0.8. It then increases with upper bound ratios of 0.9 and 1.0 although the total slacks with these two ratios are the largest. The reason for this is that, with an upper bound ratio of 0.9 or 1.0, some of the sinks tend to give up their slacks to the extent that their small roaming regions prevent them from being merged with other FFs. Fig. 14 also shows that relaxing clock period by 10% results in 14.7% reduction in the number of FFs for upper bound ratio of 0.8. It also shows that upper bound ratio of 0.8 attains the best results and ratios of 0.7 and 0.9 are equally good. Hence, in the following we will present results only for ratios of 0.8 and 0.9.

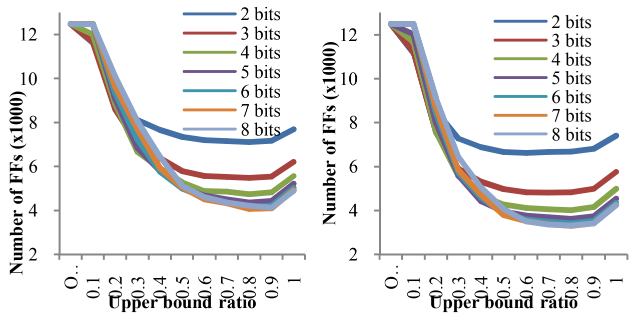

Fig. 14. Number of FFs, 1.0*clock period(left) and 1.1*clock period(right).

Table III shows the average worst negative slack (WNS, positive indicating no timing violations) of all benchmark circuits obtained by budgeting slack with upper bound ratios of 0.8 and 0.9 under 1.0*clock period. The column "Org" denotes without MBFF merging. Table IV shows the results for the cases under 1.1*clock period. A row labeled by *k bit* denotes that we allow at most *k* FFs to be merged into a MBFF. Clearly, our approach even results in better timing, by 2.8% for the case of 1.1*clock period and upper bound ratio of 0.8. Area savings are also shown in Table III and IV. Area savings are on average 3.4% for 1.0*clock period and 3.7% for 1.1*clock period.

Table V and VI shows the power saving data. In general, clock tree power is reduced with increasing number of bits in a MBFF. Clock tree power savings are on average 50% for 1.0*clock period and 60% for 1.1*clock period. With regard to total power saving, the cases for 2, 3, and 4 bits are better than the other cases. The data shown in Fig. 5 can explain this phenomenon. The cases for 2, 3, and 4 bits can save not only clock tree power but also the internal power of FFs. Basically, total power savings on average are 5.3% for 1.0*clock period and 7.2% for 1.1*clock period. Note that the extent of power and area savings highly depends how tight the timing is employed for logic synthesis, how MBFFs are designed, how much timing can be used for tradeoff, how slacks are distributed, how a clock tree is synthesized, etc.

TABLE III. TIMING AND AREA (1.0*CLOCK PERIOD).

	Average WNS (ns)			Total area (mm^2)		
	Org	0.8	0.9	Org	0.8	0.9
2-bit	0.149	0.139	0.129	1.208	1.183	1.184
3-bit	0.149	0.148	0.146	1.208	1.170	1.171
4-bit	0.149	0.132	0.139	1.208	1.166	1.167
5-bit	0.149	0.114	0.130	1.208	1.164	1.166
6-bit	0.149	0.133	0.133	1.208	1.163	1.164
7-bit	0.149	0.143	0.130	1.208	1.162	1.162
8-bit	0.149	0.120	0.105	1.208	1.165	1.163

TABLE IV. TIMING AND AREA (1.1*CLOCK PERIOD).

	Worst negative slack (ns)			Total area (mm^2)		
	Org	0.8	0.9	Org	0.8	0.9
2 bits	0.392	0.383	0.366	1.208	1.180	1.181
3 bits	0.392	0.378	0.353	1.208	1.167	1.168
4 bits	0.392	0.368	0.330	1.208	1.162	1.163
5 bits	0.392	0.368	0.301	1.208	1.161	1.161
6 bits	0.392	0.353	0.292	1.208	1.159	1.159
7 bits	0.392	0.336	0.285	1.208	1.158	1.158
8 bits	0.392	0.356	0.277	1.208	1.157	1.161

TABLE V. CLOCK TREE AND TOTAL POWER (MW) (1.0*CLOCK PERIOD).

	Clock network power			Total power		
	Org	0.8	0.9	Org	0.8	0.9
2 bits	28.5	19.6	19.8	334.2	309.9	310.3
3 bits	28.5	14.3	14.7	334.2	301.6	304.7
4 bits	28.5	13.2	13.7	334.2	312.1	307.0
5 bits	28.5	12.5	13.8	334.2	312.7	314.7
6 bits	28.5	12.1	12.7	334.2	320.7	320.0
7 bits	28.5	11.1	11.3	334.2	326.0	325.2
8 bits	28.5	14.8	13.0	334.2	331.5	332.3
Average	28.5	14.0	14.1	334.2	316.4	316.3

TABLE VI. CLOCK TREE AND TOTAL POWER (MW) (1.1*CLOCK PERIOD).

	Clock network power			Total power		
	Org	0.8	0.9	Org	0.8	0.9
2 bits	25.9	15.7	16.2	304.0	273.0	275.2
3 bits	25.9	12.1	12.6	304.0	269.8	271.1
4 bits	25.9	10.1	10.6	304.0	273.8	275.1
5 bits	25.9	10.3	10.1	304.0	281.2	285.8
6 bits	25.9	9.4	8.8	304.0	290.5	293.1
7 bits	25.9	8.4	8.2	304.0	292.1	291.5
8 bits	25.9	8.4	11.6	304.0	293.5	299.9
Average	25.9	10.6	11.1	304.0	282.0	284.5

VIII. CONCLUSIONS

In this work we presented a slack budgeting and slack-to-length converting approaches for MBFF merging. We also developed a merging method to evaluate the budgeting and converting approaches. The experimental results showed that our approach on average leads to 3.4% area saving, 50% clock tree power saving, and 5.3% total power saving for tight timing requirement designs. In the future we will re-budget timing slack during a MBFF merging process.

REFERENCES

[1] R. Nair, C. L. Berman, P. S. Hauge, and E. J. Yoffa, “Generation of performance constraints for layout,” IEEE Trans. Computer-Aided Design, vol. CAD-8, pp . 860-874, Aug. 1989.

[2] H. Youssef, R. B. Lin and E. Shragowitz, “Bounds on net delays for VLSI circuits,” IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing, VOL. 39, NO. 11, pp. 815-824, Nov. 1992.

[3] W. K. Luk, “A fast constraint generator for timing driven layout,” in ACM /IEEE Proc. 28th Design Automation Conf, pp. 626-631, 1991.

[4] M. Sarrafzadeh, D. A. Knol, and G. E. T´ellez, “A delay budgeting algorithm ensuring maximum flexibility in placement,” IEEE Transactions on TCAD, Vol. 16, No. 11, pp. 1332-1341, Nov. 1997.

[5] C. Chen, A. Srivastava, and M. Sarrafzadeh, "On gate level power optimization using dual-supply voltages," IEEE Transactions on VLSI Systems, vol.9, no.5, pp.616-629, Oct 2001.

[6] E. Bozorgzadeh, S. Ghiasi, A. Takahashi, and M. Sarrafzadeh, “Optimal integer delay budgeting on directed acyclic graphs,” DAC, pp. 920-925, 2003.

[7] S. Ghiasi, E. Bozorgzadeh, P. K. Huang, R. Jafari, and M. Sarrafzadeh, “A unified theory of timing budget management,” IEEE Transactions on TCAD, Vol. 25, No. 11, pp. 2364-2375, Nov. 2006.

[8] K. Wang and M. Marek-Sadowska, "Potential slack budgeting with clock skew optimization," ICCD, pp. 265- 271, Oct. 2004.

[9] J. Frankle, “Iterative and adaptive slack allocation for performance-driven layout and FPGA routing,” in ACM / IEEE Proc. 29th Design Automation Conf, pp. 536-542, 1992.

[10] J. Seomun, S. Paik, and Y. Shin, “Bounded potential slack: enabling time budgeting for dual-Vt allocation of hierarchical design,” ASPDAC, pp. 581-586, 2010.

[11] E. Pakbaznia, M. Pedram, “Coarse-grain MTCMOS sleep transistor sizing using delay budgeting,” DATE, pp. 385-390, 2008.

[12] S. Liu, Y. Ma, X. Hong, and Y. Wang, “Simultaneous slack budgeting and retiming for synchronous circuits optimization,” ASP-DAC, pp.49-54, Jan. 2010.

[13] Y. Kretchmer, “Using multi-bit register inference to save area and power: the good, the bad, and the ugly,” EE Times Asia, May 2001.

[14] M. P. H. Lin, C. C. Hsu, and Y. T. Chang, “Recent research in clock power saving with multi-bit flip-flops,” MWSCAS, 2011.

[15] M. P. H. Lin, C. C. Hsu, and Y. T. Chang, “Post-placement power optimization with multi-bit flip-flops,” IEEE Transactions on TCAD, vol. 30, no. 12, pp. 1870-1882, Dec. 2011

[16] Y. T. Chang, C. C. Hsu, M.P.-H. Lin, Y., W. Tsai, and S. F. Chen, “Post-placement power optimization with multi-bit flip-flops,” ICCAD, pp.218-223, Nov. 2010.

[17] J. T. Yan and Z. W. Chen, “Construction of constrained multi-bit flip-flops for clock power reduction,” International Conference on Green Circuits and Systems, pp.675-678, June 2010.

[18] J. T. Yan and Z. W. Chen, “Routability-driven flip-flop merging process for clock power reduction,” Computer Design (ICCD), 2010 IEEE International Conference on, pp. 203-208, Oct. 2010.

[19] I. H. R. Jiang, C. L. Chang, Y. M. Yang, E. Y.-W. Tsai, L. S. F. Chen, “INTEGRA-fast multi-bit flip-flop clustering for clock power saving based on interval graphs,” ISPD, pp. 115-121, 2011.

[20] S. H. Wang, Yu-Yi Liang, T. Y. Kuo and W. K. Mak. “Power-driven flip-flop merging and relocation,” ISPD, pp. 107-114, 2011.

[21] Y. Y. Liang, T. Y. Kuo and W. K. Mak. “Power-driven flip-flop merging and relocation,” Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions.

[22] S. S. Y. Liu, C. J. Lee and H. M. Chen, “Agglomerative-based flip-flop merging with signal wirelength optimization,” Design, Automation & Test in Europe Conf, pp. 1391-1396, 2012.

[23] C. Chu and Y. C. Wong, “FLUTE: Fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design,” IEEE Trans. on TCAD, Vol. 27, No. 1, pp. 70-83, Jan. 2008..