13.2

Delay Budgeting in Sequential Circuit with Application on FPGA Placement

Chao-Yang Yeh and Malgorzata Marek-Sadowska

Department of Electrical and Computer Engineering,
University of California, Santa Barbara, CA 93106, USA

ABSTRACT

Delay budgeting is a process of determining upper bounds for net delays to guide timing-driven placement. The existing approaches deal de facto only with combinational circuits. However, incorporating retiming into delay budgeting introduces more freedom to optimize sequential circuits. In this paper, we propose an approach for budgeting sequential circuits. We propose a new algorithm, T-SBGT, which uses an LP formulation to solve the budgeting problem in sequential circuits and guarantees that the clock period constraints are met. We then utilize the skew-retiming equivalence relation [9] and retime the circuit. We demonstrate usefulness of our approach in the context of FPGA placement flow. An effective algorithm to minimize Flip-Flops (FFs) number after placement using the net slack is also proposed. The results show the placement flow improves timing by 9%, and reduces budget violations by 16% compared to the traditional flow. The post-placement FF reduction algorithm decreases the FF count by 19% on average.

Categories and Subject Descriptors

B.7.2 **[Design Aids]**: Delay Budgeting in Sequential Circuits with Application on FPGA Placement

General Terms

Algorithms, Design, Theory and Performance.

Keywords

Delay budgeting, Sequential circuits, Placement, FPGA.

1. INTRODUCTION

Placement has always been a critical step in IC design. It affects greatly the circuit's area and performance. In order to achieve higher speed, several approaches have been proposed for timing-driven placement. One of them involves net budgeting [7][5]. With a user defined expected clock period, through budgeting, path timing constraints are translated into length, or timing upper bounds for nets. Those upper bounds are then used to guide placement and routing. The net-lengths or delay upper bounds constitute a *delay budget*. The first net budgeting approach for placement application was the *zero-slack algorithm* (ZSA) [7]. It is a greedy algorithm of assigning budgets to nets on long paths. ZSA ensures that the net budget is maximal, i.e. no more budget can be assigned to any of the nets without violating the path constraints. In [5], the authors propose to assign budget ensuring the "maximum flexibility" in placement. Their approach is able to adjust timing budget based on the initial information, for example, based on the results from a failed placement.

DAC 2003, June 2-6, 2003, Anaheim, California, USA.

Retiming, proposed by Leiserson and Saxe in [6], is a procedure of relocating FFs across combinational blocks to speed up the circuit. Clock skew-equivalence retiming [9] is a different way of looking at the retiming problem. The idea is to compute first the clock skew for each FF to minimize the clock period, and then to move the FFs using the skew-retiming equivalence relation.

The existing delay budgeting approaches work only for combinational circuits. In case of sequential circuits, their combinational blocks are budgeted individually. FFs are treated as primary inputs and outputs. In this paper, we introduce budgeting problem for sequential circuits and solve it by combining combinational budgeting technique with retiming. By doing so we have a larger solution space and we will have more chances to obtain a better result. We refer to the new formulation as T-SBGT. We solve the sequential budgeting in two steps. Step (1): In budgeting, we allow a free introduction of clock skew to every FF. To this end, we have modified the budgeting constraints so that they include FFs. In our formulation we include the clock period constraints to guarantee correct timing. After performing the clock period optimization, we obtain clock skew assignment for each FF. Step (2): We move FFs according to the skew-retiming equivalence relation [11]. We consider interconnect delay in this retiming procedure. We assume a linear interconnect delay model. The final retimed circuit satisfies the clock period constraints and optimizes budgeting.

We demonstrate the effectiveness of our algorithm in an FPGA placement flow. In our experiments, we assume that the FPGA has an island architecture, and each table look-up (TLB) block is associated with one FF. For placement, we first decouple FFs and TLBs and let the simulated annealing-based placer find the best positions for FFs on interconnect. After placement, an effective algorithm reduces the FF count using net slack without sacrificing timing. At the same time we pair FFs and TLBs.

This paper is organized as follows: In section 2, we define the terminology. In section 3, we introduce previous work and provide the background. We explain the traditional combinational budgeting (CDB) in section 3.1. In section 3.2, we discuss the skew-based retiming (SCO). In section 4, we show how CDB and SCO are combined and form the timing-aware sequential budgeting formulation (T-SBGT). CDB

provides the budgeting constraints in T-SBGT, and SCO provides the clock period constraints. In section 4.1, we extend CDB by including FFs into the constraints and obtain the sequential budgeting formulation (S-CDB). In section 4.2, we transform the skew variables in SCO into arrival time variables and combine SCO with S-CDB constraints. In section 4.3, we show the complete combined T-SBGT formulation. In section 5, we show the flow of our algorithm with FPGA placement. In section 6, we present experimental results. In section 7, we conclude this work.

2. DEFINITIONS

For a given circuit, we construct a directed graph $G(V,E)$, with a set of vertices V and a set of edges E. The vertices correspond to combinational modules, FFs, PIs and POs. The edges represent source-sink relations of nets. PIs are primary inputs and POs are primary outputs. An edge e_{ij} is created between the vertices i and j if in the circuit they are connected by a net and i drives j. We assume that gates have constant delays and we formulate budgeting problem in terms of interconnect delays. For each vertex i, we introduce two symbols: x_i and D_i which are the latest input arrival time and module's delay, respectively. PIs input arrival times are 0. For each node type PI, FF and PO, we introduce an additional variable s_i which represents the clock skew assigned to that node. For PIs and POs, their s-values are set to 0. The edge delay Υ_{ij} represents the delay from the fanout of i to the fanin of j. L_{ij} is the lower bound on delay of edge e_{ij}. It is used in budgeting formulation. We also create a set PS. If there is a combinational path from a PI or FF i to a PO or FF j, then we include p_{ij} into PS. The delay budget of the edge e_{ij} is denoted as B_{ij}. $max(P_{ij})$ and $min(P_{ij})$ denote the longest and shortest path delays from i to j, respectively. $Lmax(P_{ij})$ denotes the longest path delay from i to j using the L_{ij} as the net delays rather than Υ_{ij}. P is the clock period.

3. BACKGROUND

In this section, we first briefly introduce the traditional delay budgeting problem formulation (CDB). Then, we introduce the skew-based clock optimization formulation (SCO). They will be later extended and combined into the T-SBGT formulation.

3.1 Budgeting Formulation

Here, we summarize the budgeting formulation given in [5]. First, for a given circuit, we construct an edge-weighted, directed graph $G(V,E)$ as described in section 2. l_{ij} denotes a delay slack of the edge e_{ij} and is equal to $(x_j - x_i - D_i)$. $C_{ij}(l_{ij})$ is the cost for each edge in the objective function. In [5],the authors have tried several different objective functions and compared the results. We can set the objective function according to the needs of a heuristic. For example, we can: (a) allocate slacks to all nets evenly. (b) assign slacks based on the current or estimated net lengths. The general delay budgeting problem is formulated as follows:

Convex Delay Budgeting Problem (CDB): Given a convex function $C_{ij}(l_{ij})$, and a timing constraint graph $G(V,E)$:

minimize: $\sum_{e_{ij} \in E} C_{ij}(l_{ij})$

subject to:

$$l_{ij} = x_j - x_i - D_i,\ l_{ij} \geq 0,\ \forall e_{ij} \in E \qquad \text{(EQ 1)}$$

$$x_I = 0,\ \forall i \in PI,\ x_i \geq 0,\ \forall i \in V \qquad \text{(EQ 2)}$$

$$l \in R^V,\ x \in R^E.$$

This formulation constitutes a convex programming (CP) problem. The arrival time of PIs are 0. In [5], the authors converted this CDB problem into a linear programming problem and used graph-based simplex algorithm to solve it.

Later on we will transform CDB into its sequential version, S-CDB, and include FFs in the formulation.

3.2 Clock skew-equivalence retiming formulation

Clock skew-equivalence retiming has been proposed in [4]. There, the clock skew problem for minimizing the clock period is found by solving the following LP.

Skew-based Clock Optimization Formulation (SCO): Given a circuit with node and edge delays, the set PS and clock period P.

minimize: P

subject to:

$$s_j - (D_i + min(P_{ij})) + T_{hold} \leq s_i,\ \forall p_{ij} \in PS \qquad \text{(EQ 3)}$$

$$s_i + T_{setup} + (D_i + max(P_{ij})) < s_j + P,\ \forall p_{ij} \in PS \qquad \text{(EQ 4)}$$

T_{hold}, T_{setup} are the hold and setup times of FFs. The formula (EQ 4) states the *long path constraints* and (EQ 3) states the *short path constraints*. In [9], the authors use a formulation as stated above to do retiming. The procedure contains two steps:

Step 1: Solve the optimization and find skews assigned to FFs.

Step 2: Using the skew-retiming equivalence, move FFs to bring skews to *0* as much as possible. Moving FF across gates has a similar effect as introducing clock skew. If an FF has a positive (negative) skew, it is retimed backward (forward).

There are two significant points about the SCO formulation which makes it particularly useful for our purpose:

(a) We are going to assign clock skew to FFs when solving the sequential budgeting problem. The formulation of SCO also is based on skews. So, the constraints of both formulations can be combined. Additionally, we can include clock period constraints in sequential budgeting. But, modules other than FFs cannot be assigned skews, so we have to transform skew variables into the latest arrival time variables.

(b) Based on the skew-retiming equivalence relation, the assigned skews can be resolved by moving FFs.

4. TIMING-AWARE SEQUENTIAL DELAY BUDGETING

To allow the CDB formulation truly optimize sequential circuits, we introduce clock skew on FFs. We call this new formulation S-CDB. Using just S-CDB to optimize budgeting may affect the clock period requirement of a circuit, so we combine SCO with it. The combined formulation is called T-SBGT.

First, we describe, how we transform the CDB into the S-CDB. Next, we transform the skew variables in SCO into the latest arrival times. After having the new SCO constraints and the S-CDB, we combine them together into the T-SBGT. After optimization, FFs are assigned skews. The assigned clock skews can then be reduced by moving FFs using the skew-retiming equivalence relation.

4.1 Transforming budgeting formulation from combinational into sequential

Since the original CDB formulation applies only to combinational circuits, here we transform it such that it can handle sequential circuits.

First, based on the constraints $l_{ij} = x_j - x_i - D_i$ and $l_{ij} \geq 0$, the constraint (EQ 1) is transformed into:

$$x_i + D_i \leq x_j, \ \forall e_{ij} \in E \qquad \text{(EQ 5)}$$

(EQ 5) states that the latest arrival time at j must be bigger than the arrival time at a fanin i plus the delay of i. Suppose that originally if a clock arrives at time 0 to an FF i, i will have a correct value at its output at time D_i. If the latest fanin arrival time to i is x_i which is less than P, we can adjust i's clock skew to $x_i - P$. Now $x_i - P$ is a negative value. FF i will have a correct output at the time $x_i - P + D_i$. The allowed delay of combinational paths originating from i can be extended by $|x_i - P|$. If x_i is bigger than P, we can delay the clock signal by $x_i - P$. So the combinational paths originating at i have to be shortened by $x_i - P$. Based on this analysis, for an edge e_{ij}, if i is an FF, the timing of this edge has to satisfy $(x_i - P) + D_i \leq x_j$. We can transform (EQ 5) into the following constraints:

$$(x_i - P) - x_j \leq -D_i, \ \forall e_{ij} \in E \text{ if } i \in FF \qquad \text{(EQ 6)}$$

$$x_i - x_j \leq -D_i, \ \forall e_{ij} \in E \text{ if } i \notin FF \qquad \text{(EQ 7)}$$

In our formulation we also set budget lower bounds L_{ij}s on all edges. Those bounds can be obtained from the initial placement or can be predicted. We use them to guide the next placement run, hopefully, towards better results. Now the latest arrival time at j must be bigger than the arrival time at the fanin of i plus the delay of i and L_{ij}. Finally, we obtain the S-CDB formulation. In S-CDB, we use edge budgets as parameters in the cost function. We also include the edge timing constraints in the formulation. Note that unlike CDB, S-CDB considers FFs and budget lower bounds on the edges.

Sequential Circuit Convex Delay Budgeting Problem (S-CDB): Given a convex function C_{ij}, and a timing constraint graph *G(V,E)*:

minimize:

$$\sum_{e_{ij} \in E} \begin{bmatrix} C_{ij}(x_j - x_i - D_i) & if (i \notin FF) \\ C_{ij}(x_j - x_i + P - D_i) & if (i \in FF) \end{bmatrix} \qquad \text{(EQ 8)}$$

subject to:

$$(x_i - P) - x_j \leq -(D_i + L_{ij}), \ \forall e_{ij} \in E \text{ if } i \in FF \qquad \text{(EQ 9)}$$

$$x_i - x_j \leq -(D_i + L_{ij}), \ \forall e_{ij} \in E \text{ if } i \notin FF \qquad \text{(EQ 10)}$$

$$x_k \leq P, \ k \in PO; \ x_k = 0, \ k \in PI$$

P denotes the expected clock period of the circuit. For PO the arrival time must to be smaller than P. For PI the arrival time is set to 0. $(x_j - x_i - D_i)$ is the slack of e_{ij} if i is not an FF. $(x_j - x_i - D_i + P)$ is the slack if i is an FF. C_{ij} controls the weighting for each edge. (EQ 9) and (EQ 10) are delay constraints. This formulation allows us to optimize budgeting in sequential circuit.

4.2 Transforming the skew-based clock optimization (SCO) formulation

To apply the SCO constraints in the S-CDB, we have to change the clock skew variables in the SCO to the latest arrival times of some signals. Following the discussion from the previous subsection, we set the clock skew s_i of an FF i as the latest fanin arrival time $x_i - P$. Since PIs have skew 0, for PI j, the skew s_j is replaced by x_j. x_j of every FF will be also assigned 0. Based on this and assuming that the short path constraints are always satisfied, and setting T_{setup} to 0 to simplify the formulation, we obtain (EQ 11) and (EQ 12) from (EQ 4).

$$x_i - x_j \leq P - (D_i + max(P_{ij})), \ \forall p_{ij} \in PS \text{ if } i \in FF \qquad \text{(EQ 11)}$$

$$x_i - x_j \leq -(D_i + max(P_{ij})), \ \forall p_{ij} \in PS \text{ if } i \notin FF \qquad \text{(EQ 12)}$$

4.3 Adding the clock period constraints to the sequential budgeting formulation

After introducing the FFs in the CDB and transforming the skews into arrival times in SCO, we are ready to combine both constraints. The S-CDB constraints optimize budgeting and the new SCO formulation guarantees that the circuit meets clock period constraints. The new formulation is as follows:

Timing-aware Sequential Budgeting Formulation (T-SBGT): Given the clock period P, a convex function C_{ij} and a timing constraint graph *G(V,E)*:

minimize:

$$\sum_{e_{ij} \in E} \begin{bmatrix} C_{ij}(x_j - x_i - D_i) & if (i \notin FF) \\ C_{ij}(x_j - x_i + P - D_i) & if (i \in FF) \end{bmatrix}$$

subject to:

$$x_i - x_j \leq P - (D_i + max(P_{ij})), \ \forall p_{ij} \in PS \text{ if } i \in FF \qquad \text{(EQ 13)}$$

$$x_i - x_j \leq -(D_i + max(P_{ij})), \ \forall p_{ij} \in PS \text{ if } i \notin FF \qquad \text{(EQ 14)}$$

$$x_i - x_j \leq P - (D_i + L_{ij}), \ \forall e_{ij} \in E \text{ if } i \in FF \qquad \text{(EQ 15)}$$

$$x_i - x_j \leq -(D_i + L_{ij}), \ \forall e_{ij} \in E \text{ if } i \notin FF \qquad \text{(EQ 16)}$$

$$Lmax(P_{ij}) \leq max(P_{ij}), \ \forall p_{ij} \in PS \qquad \text{(EQ 17)}$$

$$x_k \leq P, \ k \in PO; \ x_k = 0, \ k \in PI$$

(EQ 13) and (EQ 14) are the clock period constraints from (EQ 11), (EQ 12); (EQ 15), (EQ 16) are the budgeting constraints from S-CDB. This LP structure is very simple and we can use graph-based simplex algorithm described in [5] to solve it. The retiming constraints guarantee that the circuit meets clock period constraints for a given P. It is necessary to add (EQ 17). This is so because we do not constraint the range of L_{ij} and allow it to be bigger than the original edge delay. We need to make sure that the longest path composed of edge budget lower bounds is smaller than the real longest path delay. Otherwise, the timing constraints (EQ 13) and (EQ 14) will be violated.

4.4 Designing the cost function

We would like our budgeting cost function to assign smaller budgets to long nets. The idea is to constraint longer nets and give shorter nets more flexibility. The net length can be predicted, can be known from an initial placement run, or can be assigned by a user. We use the product of Υ_{ij} and the budget of e_{ij} as the cost function for S-CDB (EQ 8). The terms D_i and P are constants in (EQ 8) and do not affect the optimization. Since the parameter Υ_{ij} is constant, the cost function assigned to each edge can be transformed to weights for each node. Instead of enumerating all the edges, the weight of a node can be obtained by a summation of fanin edge

weights minus the summation of fanout edge delay, as stated below:

$$\min \sum_{e_{ij} \in E} \begin{bmatrix} \Upsilon_{ij} \cdot (x_j - x_i - D_i) & if (i \notin FF) \\ \Upsilon_{ij} \cdot (x_j - x_i + P - D_i) & if (i \in FF) \end{bmatrix} \quad \text{(EQ 18)}$$

$$= \min \left[\sum_{e_{ij} \in E} \Upsilon_{ij} \cdot (x_j - x_i) \right] \quad \text{(EQ 19)}$$

$$= \min \left[\sum_{i \in V} \left(\sum_{k \in fanin(i)} \Upsilon_{ki} - \sum_{j \in fanout(i)} \Upsilon_{ij} \right) \cdot x_i \right] \quad \text{(EQ 20)}$$

4.5 Retiming implementation

After solving the T-SBGT formulation, in step 2 of the flow, we do retiming to move FFs and realize the clock skew assigned to each FF. Unlike previous algorithms which perform retiming only in the logic level, or use lumped wire delays [8], the retiming algorithm that we implemented considers the interconnect delay. So, even when an FF with non-zero skew cannot be retimed, because of other constraints, for example the skew is not big enough to retime across blocks, it can be still retimed across the interconnect. This is important because interconnect delays are becoming dominating factors of circuit performance [3].

4.6 An example

We illustrate the T-SBGT procedure using an example and show that it can optimize the budgeting in sequential circuit without violating the clock period constraint. In this example, for simplicity, we set L_{ij} equal to Υ_{ij}. We also use optimization goal (EQ 19) in CDB, so that we can explain the cost easier.

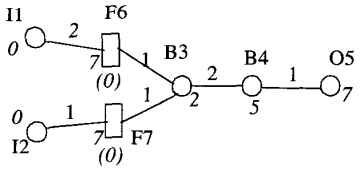

Figure 1. Combinational budgeting skew

In Figure 1, numbers on the edges are delay values. Node *O5* has delay of 0 units and all other nodes have delay of 1 unit. We set the clock period to 7 units. The original budgeting formulation cannot move FFs. If we run CDB, we obtain the arrival time assignments as those marked below each node. The braced number for FF is the time clock comes and FF has a correct value. According to the optimization function (EQ 19), the cost of $e_{I1,F6}$ is 2×7. Delay of this edge is 2 and 7 is the difference between the fanin arrival times of *I1* and *F6*. Similarly, the cost of $e_{F6,B3}$ is 1×2. The total cost of this netlist is 33.

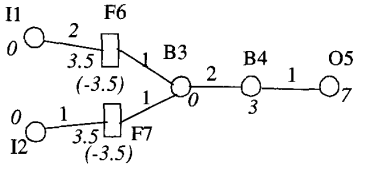

Figure 2. S-BGT skew

Figure 2 shows the arrival time assigned after we apply T-SBGT algorithm. The arrival time for both FFs are 3.5, so their skews are -3.5. The edge delays do not change, but their budgets have been changed. Now the cost for $e_{I1,F6}$ becomes 2×3.5 and the cost for $e_{F6,B3}$ becomes 1×3.5. The total cost of this netlist is 27.5, which is the minimum. The budgets for shorter nets have increased and the budgets for long nets have decreased.

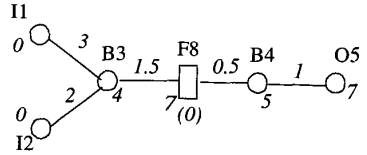

Figure 3. The budget assignment after moving FF

Figure 3 shows the result after moving F6 and F7 according to their skews, and merging them into a new flip-flop F8. The new edge delays are shown above each edge. The budgeting cost is 29 for this circuit. The cost after moving FFs is close to that before moving. It is still smaller than that obtained by CDB.

5. APPLICATION TO FPGA PLACEMENT

Figure 4 shows the new placement flow that we use in the experiment. We modified VPR and refer to the modified versions as VPR-FF and VPR-BGT. We have developed also a method of reducing the number of FFs after placement and maintaining the correct timing. These new algorithms will be explained in this section. In sub-section 5.3, we will explain the whole flow in more detail.

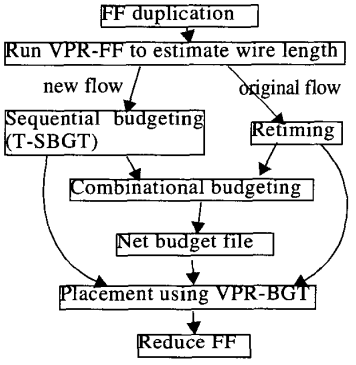

Figure 4. New and original placement flows

5.1 Modified placer

In many commercial FPGA architectures developed by Xilinx or Altera, the FFs and TLBs are paired and form the configurable logic blocks (CLBs), but can be accessed independently. However, in VPR, CLBs and TLBs can not be used independently. We modify VPR so that the FFs and TLBs do not have to be combined together. This provides us an advantage that the placer will decide the best locations for FFs on interconnect. This is important because interconnect delay accounts for more and more percentage in critical path delay [3]. The new placement algorithm is called VPR-FF.

To allow the placement algorithm to consider budgeting, we further modify the VPR-FF into the VPR-BGT. VPR-BGT has a new cost function. Originally the timing cost for each edge e_{ij} is Tt_{ij}. Tt_{ij} is computed by VPR as a product of net

delay from i to j and the net criticality. Now the timing cost is $Tt_{ij} + Bt_{ij}$. Bt_{ij} is the budgeting cost and is defined by (EQ 21):

$$Bt_{ij} = \begin{cases} 1000 \times (\Upsilon_{ij} - B_{ij})^{1.5} & if(\Upsilon_{ij} > B_{ij}) \\ 0.003 \times (\Upsilon_{ij} - B_{ij}) & else \end{cases} \quad \text{(EQ 21)}$$

There are two cases in (EQ 21). In the first case, we assign high costs for nets with delays larger than their budgets. In the second case, those with smaller delays get negative budgeting cost. The weights, 1000, in the first case is very big compared to the weight, 0.003, in the second case. If the weight difference is not big enough, the cost is dominated by the first case.

5.2 FF reduction after placement

Here we propose an algorithm for post-layout FF reduction. It uses net slacks to determine groups of FFs to be combined and reduced.

After placement, we can compute timing slack for each net. Knowing the slacks of nets connected to each FF, we can find timing feasible region for their placement. Timing feasible region is defined as the region where this FF can be placed without violating timing. In our implementation, this area is approximated by a circle with a radius equal to the minimum slack of all nets connecting to it. The problem is to find FF groups which can be combined together without timing violation. First, we create a graph *G(V,E)*. Its nodes *V* correspond to FFs. For every pair of FFs driven by the same module in the circuit, we create an edge between them if their feasible placement regions intersect. Those edges form the edge set *E*. Then we apply the minimum-clique-cover algorithm on *G*. Each clique represents an FF-group that can be reduced into one FF without timing violation. Since minimum-clique-cover algorithm is an NP-complete problem, we use a simple heuristic to find the cliques. We first find the maximum clique of the circuit and then replace all FFs in the clique by one FF. Then we find another maximum clique and continue the iterations.

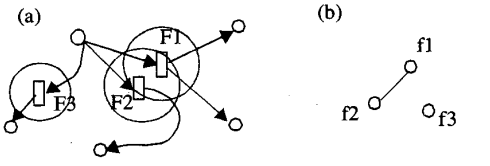

Figure 5. (a) movable area and (b) creation of clique

For example in Figure 5(a), the circles around FFs F1, F2 and F3 represent their movable areas. Figure 5(b) shows the clique graph, with three nodes f1, f2 and f3 corresponding to F1, F2 and F3 respectively. Since for F1 and F2, their feasible regions intersect and they are driven by the same node, an edge is created between them in the clique graph. The maximum clique found in G contains f1 and f2. F1 and F2 in the circuit will be replaced with one FF. After the reduction the total FF number in the circuit will be reduced from 3 to 2.

5.3 New placement flow

In this section we explain the placement flow of Figure 4 in greater detail. We apply FF duplication to the benchmark circuits before running the VPR-FF placement. We duplicate FFs with large number of fanout so we can utilize better the empty FF slots. Moderately duplicating FFs helps placement and routing.

In our case, the initial wire length estimation is obtained by running a fast mode VPR-FF placement. Since T-SBGT considers clock period constraints, it can also be used as a retiming algorithm. We compare it with retiming using the SCO formulation. In this step, for both T-SBGT and SCO algorithms, we set the clock period, *P*, to the clock period achieved by the fast placement run. We set L_{ij} equal to Υ_{ij} in T-SBGT. After running the T-SBGT and the original retiming algorithm, we obtain the retimed netlists. We set the skew of each FF to zero and run the combinational budgeting algorithm again using the cost function (EQ 20) and generate a budget file. With the retimed netlist and the budget file, we run the VPR-BGT placement algorithm. After VPR-BGT placement, we also compute the budget violation ratio by dividing the number of net budget violations by the total number of nets. Fewer budgeting violations mean that it is easier for the placement to meet timing goal and the budgeting is better [5]. We also apply the FF reduction algorithm at the end of both flows for post-placement optimization.

6. EXPERIMENTAL RESULTS

We use MCNC benchmark for our experiments. We route the circuits with larger channel widths than required, so the results are controlled by placement. We use 0.13um technology parameters to calculate the timing result. Table 1 shows the result. In the table, T-SBGT denotes the placement flow using T-SBGT for retiming and orig refers to the flow with the original retiming algorithm. The column labeled *#TLB* lists the number of TLB in the circuit. *#FF* denotes the initial number of FFs in the circuit. To take advantage of the empty FF slots in the circuit, we first duplicate FFs with high degree fanouts. In the fourth column, *max_fo* is the upper bound on the number of fanouts an FF can have after duplication. The fifth column, *#FF_d*, gives the number of FFs after duplication. In some circuits, like s298 and clma, the FF number increases a lot, because many FFs in these circuits have huge fanouts. We adjust the number of maximum-fanout allowed (*max_fo*), so that *#FF* will not be too big compared to *#TLB*. During the skew-equivalence retiming, FF number may increase considerably. We add constraints in the LP to limit the skew of all FF: $P \cdot k \geq x_i \geq P \cdot (2-k)$, $\forall (i \in FF)$ and $k < 2$. We adjust k by running the program several times so that the number of FFs generated after retiming will not exceed 80% of the total number of TLBs.

As shown in the sixth and seventh columns, after retiming, T-SBGT needs 5.05 times fewer FFs than the original flow. The difference is especially big for s298 and clma, which as mentioned earlier, have large average FF fanouts. We think the reason for the reduction is because the original retiming formulation only finds a feasible solution for the clock period and only the FFs on the critical path or critical loop are balanced. Those FFs not on the critical paths could have many unnecessary retiming moves, so the FF number could increase a lot. On the other hand, T-SBGT tries to balance FFs on all the paths and loops to optimize budgeting. It also allows FFs move to large delay paths to optimize budgeting. For the results, it seems that spreading FFs on all paths evenly is better for reducing FF number than placing them arbitrarily, even though timing is satisfied.

The new flow improves timing by about 9% compared to the original flow. The violation is also reduced by 16% compared to the original flow.

The last two columns show the FF reduction after running the clique-covering algorithm. We note that the number of FF reduced is about proportional to the number of FFs increase during the retiming step. The results show that the FF reduction algorithm is quite effective and can be used as a post-placement refinement procedure. On the average, the FF reduction is 19% compared to the original number of FFs.

7. CONCLUSIONS

In this work, we present a new budgeting algorithm which targets sequential circuits. This algorithm solves sequential circuits better because it allows retiming to further optimize budgeting. Besides optimizing budgeting, the formulation of the algorithm also includes clock period constraints, so that we guarantee timing satisfaction. Another post-layout optimization algorithm is also proposed to reduce FF numbers using slacks and preserving timing requirements. We apply our new algorithms in an FPGA placement flow.

The results show the placement flow improves timing by 9%, and reduces budget violations by 16% compared to the traditional flow. The post-placement FF reduction algorithm decreases the FF count by 19% on average.

Acknowledgement. This work was supported by the California MICRO program through Xilinx and Mentor Graphics.

8. REFERENCES

[1] V. Betz and J. Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Research", *International Workshop on Field Programmable Logic and Applications*, 1997.

[2] J. Cong and S. K. Lim, "Physical Planning with Retiming," *Proc. IEEE International Conference on Computer Aided Design*, San Jose, California, pp. 2-7, November 2000.

[3] J. Cong, http://ballade.cs.ucla.edu/~cong/slides/sasimi01_invited_final.pdf.

[4] J. P. Fishburn, "Clock skew optimization", *IEEE Trans. Comput.*, vol 39, pp 945-951, July, 1990.

[5] D. Knol, G. Tellez and M. Sarrafzadeh, "A Delay Budgeting Algorithm Ensuring Maximum Flexibility in Placement", *IEEE Transactions on Computer Aided Design*, vol 16, no 11, pp 1332-1341, 1997.

[6] C. E. Leiserson and J. B. Saxe, "Optimizing Synchronous Systems", *In Journal of VLSI and Computer Systems*, pp. 41-67, 1983.

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

[8] A. Ranjan, A. Srivastava, V. Karnam, M. Sarrafzadeh, "Layout aware retiming", *Proceedings of the 2001 conference on Great lakes symposium on VLSI*, p.25-30, March 2001.

[9] S. S. Sapatnekar, R. B. Deokar, "Utilizing the Retiming-Skew Equivalence in a Practical Algorithm for Retiming Large Circuits", *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol 15, no 10, pp 1237-1248, Oct, 1996.

Table 1. T-SBGT and original retiming in the new and traditional placement flow

circuit	# TLB	# FF	max_fo	# FF_d	# FF after retiming		clock period (ns)		budget violation%		FF reduction%	
					T-SBGT	orig	T-SBGT	orig	T-SBGT	orig	T-SBGT	orig
bigkey	1707	224	10	224	896	1468	5.90	7.1	6.01	5.20	0	13
dsip	1370	224	5	228	893	972	5.28	6.31	4.74	4.62	0	4.93
elliptic	3602	1122	100	1134	1154	1758	17.05	18.83	5.61	10.24	0.8	17.7
frisc	3539	886	10	995	1099	2400	23.36	23.13	3.33	4.55	6	39.8
s298	1930	8	50	50	50	1281	22.61	23.03	5.80	6.40	0	28.9
s38417	6096	1463	30	1589	1697	4664	14.36	15.68	5.28	7.02	3.7	44.9
tseng	1046	385	50	391	392	706	13.30	13.34	3.88	4.10	1.5	33.1
diffeq	1494	377	30	409	487	1153	13.97	14.12	4.74	6.62	6.6	33.1
s38584	6281	1260	10	1578	1623	4765	11.39	11.81	4.44	3.82	14.2	49.8
clma	8381	33	10	409	751	6458	25.7	31.8	7.49	6.21	25.4	56.6
					1	5.05	1	1.09	1.0	1.16	19.0	