Incremental Timing Budget Management in Programmable Systems

E. Bozorgzadeh[†] S. Ghiasi[‡] A. Takahashi[*] M. Sarrafzadeh[‡]

[†] Computer Science Department
University of California, Irvine
e-mail: eli@ics.uci.edu

[‡] Computer Science Department
University of California, Los Angeles (UCLA)
e-mail: soheil, majid@cs.ucla.edu

[*] Department of Communications and Integrated Systems
Tokyo Institute of Technology
email:atsushi@lab.ss.titech.ac.jp

ABSTRACT

Delay budget is an excess delay that each component of a design can tolerate under a given timing constraint. Delay budgeting has been widely exploited to improve the design quality. This paper presents the idea of incrementally re-assigning the delay budgets allotted to different components of a design, which leads to avoiding the re-execution of the intensive budget assignment procedure in each iteration of the tools. Given a budgeting solution and a local change, our approach can re-assign the budget values such that the timing constraints are met. More importantly, it only explores the components locally, which preserves its incremental *nature. General sufficient conditions have been presented under which, our approach is provably effective. Experimental results on converting non-integral budgeting assignments to integral ones, advocate our technique's efficiency. The budget assignments have been used to map a number of applications to an FPGA platform. The results have been compared to other existing methods. Our approach exhibits* 32% *improvement over other similar techniques in terms of total delay budget assigned to design components. This can be exploited to significantly improve other design metrics such as area and tool runtime. Our experimental results have verified that.*

Keywords

FPGA CAD algorithms, core-based implementation, library mapping, incremental optimization, timing budget or slack

1. INTRODUCTION

The process of computer aided design of digital circuits is composed of many computationally intensive tasks. As the size of designs increases, the computational cost of these tasks becomes even more significant. Furthermore, the process of realizing a design, is almost never a single iteration procedure. In practice, many steps of the CAD flow have to be repeated over and over in order to meet design constraints and/or improve its quality.

Most of the CAD flow algorithms, perform local changes on the design at each iteration. Starting from a given solution, they change a small portion of the design and re-execute the estimation engine to estimate the design quality after the change. This is done in order to to meet a violated design constraint and/or improve the design quality. Therefore, it is beneficial to exploit this property and develop *incremental* techniques that can handle local changes as opposed to re-executing the algorithm on the entire design. Such techniques can speedup the design process significantly and are definitely required for large designs [1, 2].

On the other hand, optimization techniques need to be applied in multiple stages starting from high levels of abstraction down to layout level. In order to abstract the complexity, each design is decomposed into a set of sub-designs. The essential constraint during the design optimization flow is the timing constraint. The sub-designs along the critical paths are the most constrained components during the optimization process. However, timing constraint is loose on the other sub-designs. Hence the allowable delay allocated on each sub-design can be greater than actual/intrinsic delay of the sub-design. This excess delay is referred to as *delay budget* (or *timing budget*). Delay budgeting has been exploited through the entire CAD flow to improve the design quality [8, 6, 7, 12, 9, 10, 11].

Each design can be represented by a directed acyclic graph (DAG $G = (V, E)$). There is a delay associated with each node. Delay budget at each node is the extra delay the component (node) can tolerate such that no timing constraint is violated. Similar definition can be applied to budget of an edge. Budget of each node/edge is related to timing slack of the node/edge. If there is any node or an edge with negative slack, timing constraint is violated. However, due to dependency between the nodes, the total timing slack of the node/edges is not the total budgets nodes/edges can tolerate. There are several existing techniques for assigning the delay budget to nodes of the graph under timing constraints [5, 4, 13, 14, 15].

In this paper, we study the problem of *incremental time-budgeting* in a design modeled by a DAG. We assume that a delay budgeting solution is available. Then, the design undergoes a local change by some other process performing incremental changes, which modifies the delay value of a node in the DAG. The objective is to locally re-assign the timing budget values to meet the constraints.

The rest of the paper is organized as follows: In section 2, the problem is formally defined. In Section 3, feasible and conservative budget re-assignments are presented and some mathematical properties are stated. Section 4 discusses some sufficient conditions for a conservative incremental budget re-assignment and applies these techniques to the incremental budgeting problem. In section 5, the experimental results of applying our method on an LP solution are reported. Incremental budget re-assignment is used to convert the non-integral LP solution to a corresponding integral solution. The solution is used in mapping applications onto an FPGA platform. The trade-off between latency and area by budgeting technique are presented. In Section 6, conclusions and some possible future directions are outlined.

2. PROBLEM STATEMENT

In this section, we formally present the problem of incremental delay budgeting in a directed acyclic graph (DAG). We assume that the computations or netlists can be modeled by a DAG. Thus, we present our results on the DAG corresponding to a computation or

netlist.

2.1 General Delay Budgeting Problem

In a given directed acyclic graph $G = (V, E)$, associated with each node v_i, there is a delay variable $d_i > 0$ and budget variable b_i. edge e_{ij} is incident to node v_j and incident from node v_i. Edge e_{ij} is called an $outgoing$ edge with respect to node v_i and an $incoming$ edge with respect to node v_j. $V_I(i)$ is the set of incoming edges to node v_i. $V_O(i)$ is the set of outgoing edges from node v_i. Primary inputs (PIs) are the nodes with no incoming edges. Primary outputs (POs) are the node with no outgoing edges.

arrival time of v_i: If input to primary input of graph is ready at time 0, the output of node v_i is ready at a_i which can be calculated as $a_i = \max_{v_j \in V_I(i)} a_j + (d_i + b_i)$, $a_i = 0$ for $v_i \in PI$.

required time of v_i: Required time r_i, is computed as $\min_{v_j \in V_O(i)}(r_j - (d_j + b_j))$. $r_i = T$ for $v_i \in PO$. T is required time at primary outputs in graph G.

Arrival time at a primary output is the maximum summation of budget and delay associated with each node along the path from primary input up to primary output. Arrival time at each primary output cannot exceed a fixed value, T. This is referred as *required time* at primary outputs. Although requited time at primary outputs and arrival time at primary inputs can be different, for simplicity, we assume that arrival time at each primary input is zero and required time at primary outputs is T.

Delay budgeting formulation: On a directed acyclic graph $G = (V, E)$ with delay d_i associated with each node v_i and required time T:

$$Max \quad \sum_{v_i \in V} b_i \tag{1}$$

$$a_i \leq T \quad \forall v_i \in PO \tag{2}$$

$$a_i = 0 \quad \forall v_i \in PI \tag{3}$$

$$a_j \geq a_i + b_j + d_j \quad \forall e_{ij} \in E \tag{4}$$

This problem has been studied by a number of researchers [5, 4, 13]. Particularly, the optimal solution to the integer version of the problem has been proposed recently in [13].

2.2 Incremental Budgeting Problem

Many design optimization techniques start from a given solution and perform local modifications iteratively to improve the design quality. These methods are in general referred to as *incremental* refinement techniques. Examples include the timing optimization procedures that deal with re-routing one net or modifying one standard cell at a time.

The incremental budgeting problem can be stated similarly. We assume that a feasible budgeting solution is given and the delay associated with one of the nodes in the graph, v_i, changes by the value δ_i. Note that this change in the delay of node v_i might make the previous solution infeasible. The objective is to find another feasible solution, with desirably good quality, by performing local changes. This can be formally stated as below:

- Given a DAG with n nodes, a budgeting solution, i.e. vectors $\mathbf{d}$ and $\mathbf{b}$ representing node delays and budgets, a maximum affordable delay at the primary outputs, T, and the amount of delay variation in one of the nodes, v_i, denoted by δ_i
- The objective is to update the vector $\mathbf{b}$
- such that the timing constraints (refer to equation 4) are met. Also, we are only allowed to exploit information about the nodes close to v_i. Obviously, this does not allow us to re-execute the delay budgeting algorithm on the entire graph.

3. BUDGET REASSIGNMENT

In this section, we first define the maximal budgeting on a given directed graph $G = (V, E)$ with required time T at primary outputs. Arrival time of any node cannot exceed T. Otherwise the dependency constraints in Equation 4 are not satisfied. A solution that meets the constraints in Equation 4 is called a *feasible solution* or a *feasible budget assignment*. Due to space constraints, lemmas and theorems are stated with no proof. Some basic definitions used in this section are as follow:

Definitions: slack at node v_i is $s_i = r_i - a_i$. *a-slack* of edge e_{ij}, $\epsilon_{a_{ij}}$, is: $(a_j - (d_j + b_j)) - a_i$, $e_{ij} \in E$. Similarly, *r-slack* of e_{ij}, $\epsilon_{r_{ij}}$ is: $(r_j - (d_j + b_j)) - r_i$, $e_{ij} \in E$. Edge e_{ij} is said to be $critical$ if the *a-slack* value and *r-slack* value associated with edge e_{ij} are zero. A path in a graph which includes only critical edges is called *critical path*.

The following lemma can be easily derived from the abovementioned definitions:

LEMMA 1. *In a directed graph G, if $e_{ij} \in E$ and $s_i = s_j$, then $\epsilon_{a_{ij}} = \epsilon_{r_{ij}} = \epsilon_{ij}$.*

Maximal Budgeting Graph (G, B_m): B_m is a feasible solution to budgeting problem on a directed acyclic graph G. Feasible solution B_m with associated objective value, $|B_m|$, is called maximal budgeting if no more budget can be given to any node while the budget of any other node does not decrease.

The maximum solution B^* is also a maximal solution. A maximal budgeting solution B_m can be obtained by applying different existing algorithms [4, 5, 13].

LEMMA 2. *In (G, B_m), the slack of each node is zero if and only if B_m is a maximal budgeting.*

Non-critical edges are referred to as ϵ-edges. According to Lemma 1 the a-slack and r-slack of a ϵ-edge in (G, B_m) are equal, that is $\epsilon_{ij} = \epsilon_{a_{ij}} = \epsilon_{r_{ij}}, \forall e_{ij} \in (G, B_m)$.

LEMMA 3. *In a maximal budgeting (G, B_m), each node (except PIs and POs) has at least one critical incoming edge and at least one critical outgoing edge.*

Associated with solution B_m, *critical graph* $G_T \subseteq G = (V, E)$ is the graph obtained from the graph G by deleting all non-critical edges in G. $G_T = (V, E_T)$, $E_T = E - \{e_{ij} | \epsilon_{ij} \neq 0\}$.

In any budgeting on graph G, slack of each node and edge must be non-negative or in other words $a_i \leq T$. This is referred to as *feasibility* in graph. A graph with budgeting B is not feasible if slack of a node or an edge is negative. We propose a budget re-assignment method on a given maximal budgeting.

Conservative Budget Re-assignment on (G, B_m): In graph G with maximal budgeting solution B_m, the budgets of the nodes are changed such that the new budgeting B'_m is still a maximal budgeting (G, B'_m). Budget re-assignment on graph G transforms the budgeting from solution B_m to B'_m.

Theorem 1 presents two sufficient conditions for conservative budget re-assignment.

THEOREM 1. *The re-assignment of budget at each node in graph (G, B_m) is a conservative budget re-assignment if*

- *The total amount of change in the budget of the nodes along each critical path from PI to PO is zero (Figure 1), and*

- *For each ϵ-edge e_{il}, $\epsilon_{il} \geq (k_i - k_j)$, where edge e_{jl} is critical. k_i and k_j are the amount of change in the budget along any critical path [1] from PI to node v_i and v_j, respectively [13] (Figure 1(b)).*

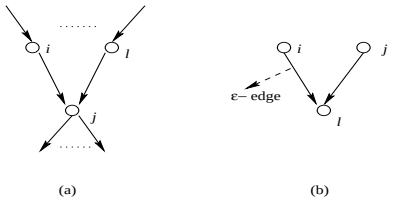

Figure 1: Two Sufficient Conditions for β-budget Re-assignment.

We define an equivalence relation on the given graph that partitions the graph into a number of equivalent classes. We show that the budget exchange between two sub-graphs formed by this relation satisfies the aforementioned conditions, hence it is a conservative budget re-assignment in graph (G, B_m).

Parent/Child Relation: In a directed graph G, edge $e_{ij} \in E$ and e_{ij} is critical. Node v_j is child of node v_i. $c(v_i)$ is used to refer to a child of node v_i. Node v_i is said to be the parent of node v_j. $p(v_j)$ is used to refer to a parent of node v_j. If v_i and v_k have common child, $v_i \sim_p v_k$. If $v_1 \sim_p v_2 ... \sim_p v_n$, then $v_1 \sim_p^* v_n$. $\sim_p^*$ is an equivalent relation, called parent relation. If v_i and v_l have common parent, $v_i \sim_c v_l$. If $v_1 \sim_c v_2 ... \sim_c v_n$, then $v_1 \sim_c^* v_n$. Similar to parent relation, $\sim_c^*$, called child relation, is an equivalent relation.

LEMMA 4. *$v_i \sim_c^* v_j$, if and only if $p(v_i) \sim_p^* p(v_j)$.*

LEMMA 5. *In (G, B_m), if $v_i \sim_p^* v_j$, arrival time at nodes v_i and v_j are equal; $a_i = a_j$.*

According to Lemma 3, each node is incident to/from a critical edge. Consider node v_i in graph $G = (V, E)$. Let $S_p(v_i) = \{v_j | v_i \sim_p^* v_j\}$ be a parent set. Let v_l be a child node of v_i. $S_c(v_l) = \{v_j | v_j \sim_c^* v_l\}$. According to Lemma 4, sets $S_p(v_j)$ and $S_c(v_l)$ are a pair of sets such that all the child nodes of the nodes in S_p are in S_c. Similarly, all the parent nodes of the nodes in set S_c are in S_p. The sets $S_p(v_i)$ and $S_c(v_l)$ are called *parent-child* set (S_p, S_c) associated with node v_i. Parent-child set (S_p, S_c) is shown in Figure 2. The followings are the propositions regarding the parent-child set in (G, B_m).

LEMMA 6. *If nodes $v_i \sim_p^* v_j$, there is no directed critical path between v_i and v_j if $\forall v_i \in V, d_i > 0$. Similarly, if nodes $v_i \sim_c^* v_j$, there is no directed critical path between v_i and v_j if $\forall v_i \in V, d_i > 0$.*

LEMMA 7. *In a parent-child set (S_p, S_c), S_p and S_c do not intersect if $\forall v_i \in V, d_i > 0$.*

Given a parent-child set (S_p, S_c), assume that the process of budget re-assignment decreases the budget of all of the nodes in S_p by δ and increases the budget of all of the nodes in S_c by δ, $\delta > 0$.

LEMMA 8. *In a given (S_p, S_c) in (G, B_m), if $\delta \leq min(\epsilon_{ij})$, where e_{ij} is an ϵ-edge with $v_j \in S_c$ and $v_i \notin (S_p, S_p)$ (incoming ϵ-edges to S_c), the δ-budget exchange is a conservative budget re-assignment in (G, B_m).*

[1] Note that because of the first sufficient condition, the change in budget along all of the critical paths, from PIs to a node are equal.

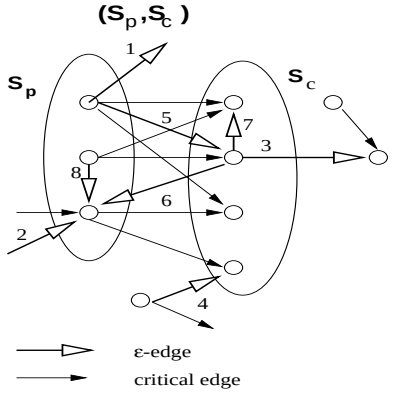

Figure 2: ϵ-edges with respect to Parent-Child Set (S_p, S_c).

4. INCREMENTAL DELAY BUDGETING

When the delay of node v_i is changed by δ_i, it can affect the feasibility of the current budgeting in the graph or criticality of the edges. In case of the effect on feasibility, we need to re-assign delay budget in some of the nodes in order to avoid violation in timing. Re-assignment of delay budget of the nodes can be as costly as applying delay budgeting algorithm again. This is not efficient. The goal is to apply budget re-assignment incrementally. Incremental budget re-assignment means to re-assign the delay budget of the nodes locally within a close distance from node v_i where delay has increased. We provide a set of sufficient conditions under which the delay budget re-assignment can be applied incrementally on graph G once the delay of node $v_i \in G$ is changed by δ_i. We divide the conditions into two groups. In the first group, we present a set of sufficient conditions to apply feasible conservative budget re-assignment incrementally, and in the next group we look at the sufficient conditions to obtain a feasible maximal budgeting after incremental budget re-assignment. We apply the concepts of budget re-assignment and the proposed lemmas in the previous section.

Feasible Conservative Incremental Budgeting- Let d_i and b_i be the original delay and delay budget of node v_i, respectively. In the previous section, we showed that between a parent-child set in a graph, the same value of budget can be re-assigned. The delay budget of the nodes in parent set can increase with the same value as the delay budget of the nodes in the corresponding child set decreases by the same value and vice versa. Starting from primary inputs of graph G with maximal budgeting B_m, we construct the parent-child sets while traversing the graph until primary outputs. Each node v_i belongs to one parent set $S_p(v_i)$ and one child set $S_c(v_i)$. $S_p(v_i)$ and $S_c(v_i)$ are not necessarily equal. Figure 3 shows an example of unequal parent set and child set associated with a node in a graph. The construction of parent sets and child sets can be done and pre-computed before any budget re-assignment.

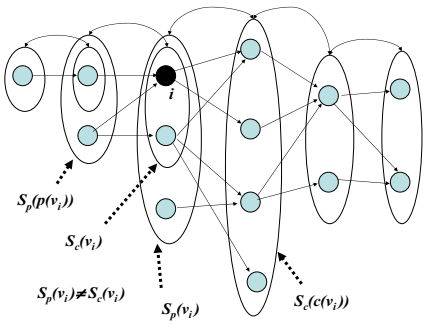

Figure 3: Parent-child Set Construction in A DAG.

We want to re-assign a budget value of Δ on node v_i in the graph. Assume that $\Delta < \epsilon ij$ for any non-critical edge e_{ij} and the amount of budget on each node is at least Δ. Simply, we can re-assign the budget between $S_p(v_i)$ and the corresponding child set of this parent set. Similarly, we can re-assign the budget between $S_c(v_i)$ and the parent set associated with this child set. After this re-assignment, the critical edges are still critical and a new maximal budgeting B'_m is obtained.

In incremental budgeting problem, intrinsic delay of node v_i is changed by δ_i. Therefore delay of node v_i becomes $d_i + \delta_i$. If $\delta_i < 0$, the delay of node has decreased and budget at node v_i increases by $|\delta_i|$. Therefore the budget assignment in graph remains feasible and criticality of the edges remains unchanged. When $\delta_i > 0$, delay of node v_i increases by δ_i. If $b_i > \delta_i$, there is sufficient delay budget at the node to keep the delay budgeting on graph G feasible. Only the delay budget of node v_i decreases by δ_i. In the case of $b_i < \delta_i$, the delay budgeting is not feasible and delay re-assignment needs to be applied. In order to obtain a feasible and conservative budgeting (keeping critical edges critical), we need to increase the delay budget at node v_i by $\Delta = \delta_i - b_i$. We can increase the delay budget of node v_i by Δ, applying budget re-assignment on parent-child set at node v_i.

One way is to apply budget re-assignment between $S_p(v_i)$ and $S_c(c(v_i)$ by increasing the budget in parent set by Δ and decreasing the delay budget of the nodes in child set with the same value. Another way is to apply budget re-assignment between $S_c(v_i)$ and $S_p(p(v_i)$, decreasing delay budget of the nodes in parent set and increasing the delay budget of the nodes in child set with the same value of Δ. Note that in order to be able to apply budget re-assignment, there has to exist sufficient delay budget at each node in the set where delay budget is decreased.

LEMMA 9. *Suppose the delay of node v_i is increased by δ_i and $b_i < \delta_i$. We can obtain feasible budgeting by applying budget re-assignment between parent-child set constructed at node v_i on either of the two sets, if there is at least budget of $\Delta = \delta_i - b_i$ at each node in $S_p(p(v_i))$ or $S_c(c(v_i))$ and the slack of non-critical edges incident to $S_c(v_i)$ or incident from $S_p(v_i)$ are at least Δ.*

Lemma 9 provides a set of sufficient conditions for incremental conservative budget re-assignment. As discussed above, there are two different parent-child set defined at each node v_i in graph G, depending on either node v_i belongs to a parent set or a child set. If both sets hold the conditions in Lemma 9 for budget re-assignment of $\frac{\Delta}{2}$, we can apply budget re-assignment on both sets and increase the delay budget at node v_i by Δ. This means that as long as there is a delay budget of $\frac{\Delta}{2}$ on each node on the both parent set and child set of $S_p(p(v_i))$ and $S_c(c(v_i))$ and slack of non-critical edges incident to these sets is at most this value, the feasible budgeting is obtained. Incremental budgeting is applied by re-assignment of budget between the two immediate parent-child sets defined at a node in the graph. We can even extend this technique to re-assign the budget from one level further than the immediate parent-child set. In this way, the incremental radius has expanded to more levels (say k levels). In this general case we have the following lemma.

LEMMA 10. *Assume that the delay of node v_i is increased by δ_i where $b_i < \delta_i$. Furthermore, assume that there is at least budget of $\Delta = (\delta_i - b_i)/k$ available at each node in $S_p(p(p(...v_i))$ up to k levels or $S_c(c(c(...(v_i))$ up to k levels, and $(\delta_i - b_i)$ is at most the slack of non-critical edges incident to $S_p(p(p...(v_i))$ or incident from $S_c(c(c...(v_i))$. Then, feasible conservative budget re-assignment can be applied to accommodate the extra delay on node v_i.*

Feasible Incremental Budgeting- The conditions in Theorem 1 are sufficient to obtain a conservative feasible maximal budgeting. If the only concern is feasibility of the solution after budget re-assignment, we can apply the following technique.

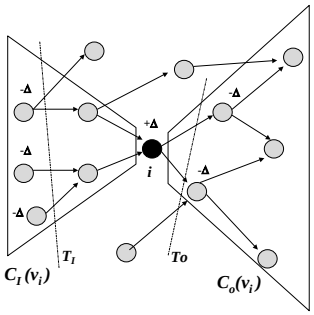

Figure 4: Example of output cone and input cone associated with node v_i.

Let $C_o(v_i)$ be the output cone of v_i, an induced subgraph including all the nodes along the critical paths from node v_i until POs. If delay of node v_i is increased by δ_i and $\delta_i < b_i$, we need to increase the budget at node v_i by $\delta_i - b_i$. Let T be a cut set in $C_o(v_i)$ in which each node has the delay budget as large as $\delta_i - bi$. If we decrease the delay budget of the nodes in the cut set by $\Delta = \delta_i - b_i$ and increase the budget of node v_i by the same value, the resulting budgeting is feasible. Figure 4 shows an example. Similarly we can define $C_I(v_i)$ as the input cone of node v_i, including all the nodes along the critical paths from PIs to node v_i. If we decrease the delay budget of the node in cutset T of this cone by $\Delta = \delta_i - b_i$, we can achieve feasible budgeting. Therefore we can have the following sufficient condition:

LEMMA 11. *If delay of node v_i is increased by $\delta_i > b_i$, if there is a cutset T in $C_o(v_i)$ or $C_I(v_i)$ in which the delay budget of the nodes is at least $\delta_i - b_i$ and $\delta_i - b_i \leq \epsilon_{kj}$ where e_{kj} is a non-critical edge incident to or from any node in the cutset, feasible budgeting can be obtained by incremental delay budget re-assignment.*

Similar to our proposed technique for incremental conservative delay re-assignment, we can extend the level of incremental assignment, decompose Δ to k value (say Δ/k) and find k cutset and apply the delay re-assignment. In this extension, the sufficient condition for minimum available delay budget on each node in the cut and minimum slack on non-critical edges can be relaxed to Δ.

The condition proposed in Lemma 11 is sufficient to have a feasible budgeting after decreasing the delay budget of the nodes in the cutset T. In this solution, the critical edges in the graph does not necessarily remain critical but we still obtain a maximal feasible budgeting by incremental budget re-assignment.

5. APPLICATION

In this section, we apply our technique for feasible conservative budget re-assignment to find an integer solution for delay budgeting from an optimal solution in which delay budget can be non-integer. We show that how the incremental and conservative manner in our algorithm leads to obtain an integer solution which is still optimal.

5.1 Integer Delay Budgeting

Objective function in delay budgeting problem is to maximize the total delay budget of the nodes under a given timing constraint. The general problem can be formulated as a linear programming problem. However, the solution can have fractional value. According to the following reasons optimal *integer* solution is preferred: First, the budget at each node is mostly used to map the sub-design to another component in a target library which inherently is discrete rather than continuous. For example, delay on interconnect is discrete in a grid-based routing methodology. In a data path, delay of each component can be given in terms of number of clock cycles under a given frequency. Delay of gates can be scaled to integer values. In VLSI compaction, grid constraints require integer solution [10]. Secondly, due to numerical instability in representation of real numbers, linear

programming solvers suffer from instability and difficulty in convergence. Therefore we assume the variables associated with budgets are all integer. Existing heuristic algorithms, ZSA and MISA, can be modified to generate integer budgets, but with no guarantee on the optimality of the solutions.

(G, B^*) is the optimal solution to linear programming relaxation of integer budgeting problem. B^* is also a maximal budgeting. Hence, budget re-assignment is applicable to (G, B^*). In addition, since B^* is the optimal solution, $B_m \leq B^*$ for any maximal budgeting B_m. We apply budget re-assignment on graph (G, B^*) such that the budget of all the nodes become integer. We show that during this transformation from optimal solution to integer solution $(B^*)'$, the objective value of new solution is equal to $|B^*|$. In our previous work [13], we have shown that integer delay budgeting is optimally solvable in polynomial time. In this section, we describe how based on our incremental conservative delay budget re-assignment proposed in previous section optimal integer solution can be obtained.

THEOREM 2. *The total budgeting on any critical path from PI (Primary Input) to PO (Primary Output) is integral.*

Each node with fractional budget belongs to an integral critical path. Hence, within an integral path, it is sufficient to re-assign the fractional budget only on the nodes along the path. On the other hand, in graph G, there are several integral paths connected to each other. Therefore in re-assigning the budget between the nodes, the required conditions in Theorem 1 have to be satisfied in all those sequences. Hence, the goal is to apply budget re-assignment of the fractional budgets on the nodes in graph (G, B^*) to obtain integer solution. Budget re-assignment is applied on graph G such that the budget of all the nodes become integer. Only fractional value of budgets need to be re-assigned in order to obtain integer solution. Hence the re-assigned delay budget is a fractional value less than unit. As described in previous section, feasible budget-reassignment can be applied on a parent-child set on graph G.

On a given parent-child set in graph G, we apply budget exchange. If fractional budget in graph G are re-assigned by budget re-assignment on parent-child set, the fractional budget is removed from each parent node and re-assigned to one of its successor in the graph. Hence, the fractional budgets are re-assigned from PIs to POs, in one direction within an integral sequences. There are ϵ-edges in a given graph G. In order to have a feasible budget re-assignment on parent-child set, we show that the sufficient conditions outlined in Theorem 1 are satisfied in a given graph G as well.

THEOREM 3. *In any feasible budget re-assignment on parent-child set (S_p, S_c) on the nodes with fractional delay budget in graph (G, B^*), the total budget does not change.*

Hence after applying the budget re-assignment on (G, B^*), the solution is still optimum . The reason is that the cardinality of any parent-child set in this graph is same and hence exchange of delay budget between the two sets does not affect the total delay budget.

Each parent-child set construction takes $O(|E|)$ and budget re-assignment takes $O(|E|)$. This repeats $O(|V|)$ times. However, by amortized analysis we see that the complexity of sequential construction of parent-child sets and delay re-assignment is $O(|E|)$. The result is transformation from solution B^* to a new solution $(G, (B^*)')$ in which integer budget is assigned to each node while objective value does not change, i.e., $|B^*|$.

5.2 Experimental Setup

In Figure 5, CAD flow of IP-based (or *core-based*) mapping an application on a FPGA is illustrated. Xilinx CoreGen tool generates and delivers parameterizable cores optimized for target architecture. The parameters include data width, registered output, number of pipeline stages, etc. Core layout is specified up front. Cores are delivered with optimally floorplanned layouts. Also, performance of cores are independent of FPGA device size. Hence, more predictable results can be obtained during front-end optimization. Since CoreGen cores are pre-optimized, they are considered as black boxes during the synthesis. Hence, synthesis is ignored in core-based design. In a rich core library, there can exist several cores realizing same functionality with different implementation and latency (in terms of clock cycle). Figure 6 demonstrates a trade-off between latency and area of a CoreGen 16-bit multiplier with target FPGA VirtexE, Xilinx. *Slices* are the logic blocks in VirtexE FPGA series.

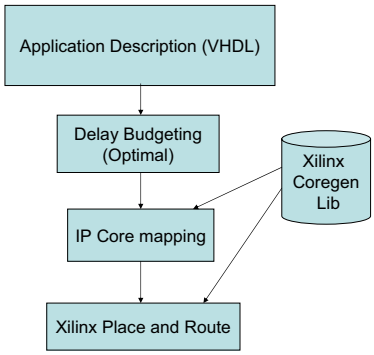

Figure 5: Mapping an Application on FPGA Using IP Library.

We start from a DAG representation of an application. Benchmark in our experiments is a set of some standard DSP benchmarks. The type of computations are multiplier, adder, subtracter and shifter. We assume all the data paths are 16-bit wide.

Each computation is assigned to a resource generated from CoreGen tool based on delay budget allocated to the node. We apply a delay budgeting algorithm to allocate the delay budget at each node. Then the whole circuit is placed and routed on a FPGA device. We used *ISE 4.1* place_and_route tool provided by $Xilinx^{TM}$. The target device is VirtexE 300.

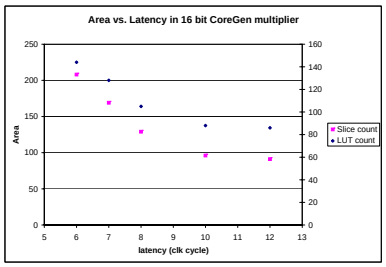

Figure 6: Area vs. Latency for a 16-bit CoreGen Multiplier.

Among different computations in the applications, CoreGen has a relatively complete library (See Figure 6). Hence we applied delay budgeting only among the nodes that correspond to computation type *multiply*. We conducted two sets of experiments. Once we applied our optimal delay budgeting and once we applied a heuristic sub-optimal budgeting (ZSA like) to distribute the latency in graph. Since the latency of the components is based on clock cycles, we need to assign integer delay budget. Hence, in the first set of experiments we apply our incremental conservative budgeting on optimal solution by LP solver and produce optimal delay assignment to the nodes. ZSA algorithm can be simply justified to assign integer delay to the components.

5.3 Experimental Results

The original latency and other characteristics of the benchmarks are given in Table 1.

Benchmark	Runtime Area	ΔT=0	ΔT=1 clk cycle			ΔT=2 clk cycle		
			Heuristic	Optimal	Imp	Heuristic	Optimal	Imp
Diffeq	area(slices)	780	740	700	5.7%	708	652	8%
	PAR(sec)	15	10	10	1	11	9	1.2
	Budget(clk cyc)	-	2	3	50%	4	6	50%
ARF	area(slices)	1982	1806	1803	0.2%	1670	1665	0.3%
	PAR(sec)	45	42	29	1.5	43	26	1.65
	Budget(clk cyc)	-	32	38	19%	36	48	33%
Fdct	area(slices)	2044	1867	1734	7.1%	1728	1491	14%
	PAR(sec)	48	39	39	1	38	36	1.05
	Budget(clk cyc)	-	14	20	43%	22	34	54%
Ewf	area(slices)	1138	1094	1016	7.2%	1058	982	7.2%
	PAR(sec)	24	21	18	1.67	19	17	1.11
	Budget(clk cyc)	-	2	6	200%	4	10	150%
Dct	area(slices)	1338	1091	1032	5.4%	1038	996	4%
	PAR(sec)	38	19	18	1	20	15	1.33
	Budget(clk cyc)	-	24	27	12.5%	30	34	13.3%
Average	area(slices)	1456	1327.6	1257	5.4%	1240	1157.2	7%
	PAR(sec)	34	26.2	22.8	1.15	26.2	20.6	1.27
	Budget(clk cyc)	-	14.8	18.8	27%	19.2	26.4	37.5%

Table 2: Area (#slices-logic blocks), total Budget, and Runtime of Place-and-Route (sec) vs. delay budget (clk cyc).Imp column compares Optimal over heuristic. It indicates the percentage of improvement for area and budget and the ratio of runtime for PAR runtime.

Benchmark	Nodes	Latency	Slices	LUTs
Diffeq	10	18	780	1030
ARF	28	20	1982	2476
FDCT	42	14	2044	1734
EWF	34	25	1138	1472
DCT	33	14	1618	1338

Table 1: Benchmark Information and Core-based Implementation Results.

Table 2 summarizes the implementation results of applying delay budgets to the applications. Latency of each application is the original latency reported in Table 1 plus the excess latency (ΔT) applied to the circuit. The excess latency in distributed in graph using delay budgeting algorithm. We use both exact (our optimal method) and heuristic (ZSA like) methods. Area (number of used slices of FPGA device) and place-and-route runtime and total budget are reported. The first column shows the place_and_route (PAR) runtime, area of slices when no delay budget is applied. The next columns show the area and PAR runtime for different excess delay to required time (ΔT) of 1 and 2 clk cycles. The *Imp* column shows the percentage of improvement in area in different delay budgeting computed as $\frac{(Area(Heu)-Area(opt))}{Area(Heu)} \times 100$. Similarly Imp is computed for total budget. The *Imp* column computes the improvement in runtime as ratio of $\frac{PAR\_runtime(Heu)}{PAR\_runtime(opt)}$.

The results show the average improvement in area for 5.2%, 7%, in terms of number of slices when optimal algorithm is used for budgeting compared to area resulted by heuristic delay budgeting for different ΔT. Although the area of applications by optimal delay budgeting is always smaller than the area resulted by heuristic method by 6% on average, runtime of place_and_route in some application does not speed up. One reason is that some of applications such as $Fdct$ are I/O bounded. A main portion of place and route is dedicated to I/O placement and routing. In other benchmark such as *ARF* the runtime of place_and_route gets almost two times faster. As a result, delay budgeting gives the opportunity of mapping the applications to components in the target library with simpler structure and smaller area. Comparing the result of the first column when no budget is applied with the results of the next columns demonstrates this fact. However, the current libraries are not rich enough and do not contain different components with different latencies for same functionality. Developing complete libraries facilitates the design CAD tool to exploit the existing delay budget to improve design quality.

6. CONCLUSIONS

This paper presents the idea of incrementally re-assigning the delay budgets allotted to different components of a design, which leads to avoiding the re-execution of the intensive budget assignment procedure in each iteration of the tools. In incremental delay budgeting problem, delay of a node is increased by δ. In order to achieve feasible solution, delay budget of the nodes are re-assigned. In incremental approach, the concern is to preserve the solution and minimize the modification in the solution. In this paper, we proposed a set of sufficient conditions under which the delay budget of the nodes are re-assigned incrementally and feasibility is guaranteed. Our technique is very general and can be applied and integrated in different delay budgeting algorithms and incremental optimization techniques in VLSI CAD. In this paper, using optimal solution to LP relaxation of budgeting problem, we transform the solution to integer solution using our technique for incremental conservative delay re-assignment. We prove that during this transformation ($O(|V|^2)$), objective value from optimal LP solution does not change. We applied our budgeting technique in mapping applications on FPGA device. Using IP library of different computations, delay budget is exploited to improve the area, hence to speedup the runtime of place-and-route. Our optimal algorithm outperforms ZSA algorithm [5] in terms of area and design time significantly.

7. REFERENCES

[1] O. Coudert, J. Cong, S. Malik, and M. Sarrafzadeh. " Incremental CAD. In Proc. of *IEEE/ACM International Conference on Computer-Aided Design*, pp. 236 244, 2000.

[2] J. Cong, and M. Sarrafzadeh. " Incremental Physical Design. In Proc. of *International Symposium on Physical Design*, pp. 84 92, 2000.

[3] L. A. Wolsey. *Integer Programming*. New York, NY, Wiley-Interscience Publisher, John Willey & Sons Inc., pg. 37-52, 1998.

[4] C. Chen, E. Bozorgzadeh, A. Srivastava, and Majid Sarrafzadeh. "Budget Management with Applications". In *Algorithmica*, vol 34, No. 3, pp. 261-275, July 2002.

[5] R. Nair, C. L. Berman, P. S. Hauge, E. J. Yoffa. "Generation of Performance Constraints for Layout In *IEEE Transactions on Computer-Aided Design*, Vol. 8, No. 8, pp. 860-874, August 1989.

[6] M.Sarrafzadeh, D. A. Knol, G.E. Tellez. "A Delay Budgeting Algorithm Ensuring Maximum Flexibility in Placement In IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, , Vol. 16, No. 11 , pp. 1332 -1341, Nov. 1997.

[7] C. Kuo and A. C.-H Wu. “Delay Budgeting for a Timing-Closure-Design Method”, In *International Conference on Computer-Aided Design*, pp. 202 207, 2000.

[8] C. Chen, X. Yang, M. Sarrafzadeh. “Potential Slack: An Effective Metric of Combinational Circuit Performance. In Proc. of ACM/IEEE International Conference on Computer-Aided Design, pp. 198-201, 2000.

[9] Y. Liao and C. K. Wong. “An Algorithm to Compact a VLSI Symbolic Layout with Mixed Constraints”. In Proceedings of *IEEE Transactions on CAD*, Vol. 2, No. 2, April. 1983.

[10] J. F. Lee and D. T. Tang. “VLSI layout compaction with grid and mixed constraints”. In Proceedings of *IEEE Transactions on CAD*, Vol. 6, No. 5, Sep. 1987.

[11] E. Felt , E. Charbon , E. Malavasi ,and A. Sangiovanni-Vincentelli. “An efficient methodology for symbolic compaction of analog IC’s with multiple symmetry constraints”. In Proceedings of *Conference on European Design Automation*, November 1992.

[12] P. Girard, C. Landrault, S. Pravossoudovitch, and D. Severac. “ A Gate Resizing Technique for High Reduction in Power Consumption. In Proc. of *International Symposium on Low Power Electronics and Design*, pp. 281 286, 1997.

[13] E. Bozorgzadeh, S. Ghiasi, A. Takahashi, and M. Sarrafzadeh. “Optimal Integer Delay Budgeting on Directed Acyclic Graphs. In Proc. of *Design Automation Conference*, pp. 920-925, 2003.

[14] S. Ghiasi, K. Nguyen, E. Bozorgzadeh, and M. Sarrafzadeh. “On Computation and Resource Management in Networked Embedded Systems. In Proc. of *International Conference on Parallel and Distributed Computing and Systems*, 2003.

[15] S. Ghiasi, K. Nguyen, E. Bozorgzadeh, and M. Sarrafzadeh. “On Computation and Resource Management in an FPGA-based Computing Environment. In Proc. of *International Symposium on Field-Programmable Gate Arrays(poster)*, 2003.