Optimal Integer Delay Budget Assignment on Directed Acyclic Graphs

E. Bozorgzadeh†  S. Ghiasi†  A. Takahashi‡  M. Sarrafzadeh†

† Computer Science Department
University of California, Los Angeles (UCLA)
Los Angeles, CA 90095, USA
e-mail: elib, soheil, majid@cs.ucla.edu

‡ Department of Communications and Integrated Systems
Tokyo Institute of Technology
Tokyo 152-8552, Japan
e-mail: atsushi@lab.ss.titech.ac.jp

Abstract

Excess delay that each component of a design can tolerate under a given timing constraint, is referred to as delay budget. Delay budgeting has been widely exploited to improve the design quality in VLSI CAD optimization flow. Problem of delay budgeting is to maximize the total delay budget assigned to each node in a graph under a given timing constraint. Due to numerical instability and/or discreteness of libraries of components during design optimization flow, integer solution for delay budgeting is essential. We present an optimal integer delay budgeting algorithm. We prove that the problem can be solved optimally in polynomial time. In addition, we look at different extensions of the delay budgeting problem, such as maximization of weighted summation of delay budgets assigned to the nodes with constraints on lower bound and upper bound on the delay budget allocated to each node. We prove that for both aforementioned extensions, our algorithm can produce an optimal integer solution in polynomial time. Our algorithm is generic and can be applied in different design tasks at different levels of abstraction. We applied optimal delay budgeting in mapping applications on an FPGA platform using pre-optimized cores of FPGA libraries. For each application, we go through synthesis and place and route stages in order to obtain accurate results. Our optimal algorithm outperforms ZSA algorithm [4] in terms of area by 10% on average for all applications. In some applications, optimal delay budgeting can speedup runtime of place and route up to 2 times.

1 Introduction

Due to the complexity of system design and high uncertainty of timing issues and quality metrics in higher levels of abstraction, it is not effective to optimize performance intensively in earlier stages. Instead, the optimization should aim at ensuring correctness and convergence of the design. Optimization techniques need to be applied in multiple stages starting from high level of abstraction down to gate level and physical design. Mapping sub-designs to already existing pre-optimized and synthesized components is an unavoidable scheme to abstract away the complexity in VLSI/CAD design automation flow.

The essential constraint during the design optimization flow is the timing constraint. A percentage of the delay along the paths in a complex design is dedicated to each sub-design. The sub-designs along the critical
paths are the most constrained components during the optimization process in CAD flow. However, timing constraint is loose on the other sub-designs. Hence the delay allocated to 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 whole CAD design flow to improve the design quality.

![Figure 1: Delay Budgeting Problem in DAG.](image)

Each design is represented by a directed acyclic graph (DAG $G=(V,E)$). There is a delay associated with each node. Timing constraint is the delay at the output. Delay or latency at the output is computed as the longest path delay in the graph from input to output. The delay along each path is the total delay associated with the nodes and/or edges along the path. Under a given timing constraint, delay budget at each node is the extra delay the component 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 nodes/edges is not the total budgets nodes/edges can tolerate. In Figure 1, two different methods of delay budgeting ($A$ and $B$) are applied on a DAG. Columns “Budget A” and “Budget B” of the table correspond to excess delay (delay budget) assigned to each node under timing constraint ($13\text{nsec}$) in approach $A$ and approach $B$. After applying any of budgeting $A$ or $B$ on the graph, no other node can tolerate any excess delay. Total delay budget after budgeting $A$ is 17 while the total delay budget after budgeting $B$ is 12.

In this paper, we study the problem of the assignment of maximum total budget in a graph. The budgeting problem in a graph is well studied in theory and practice and is widely used in today’s industry and research. Delay budgeting has several applications in design optimization as follows:

- **Design timing closure**: During design optimization flow, timing budget is allocated to each node under a given timing constraint and optimization is applied. If timing constraint is not met, the delay budget is re-allocated [9]. Exploiting the maximal budgeting can lead to earlier convergence.

- **Timing-driven placement and floorplanning**: Delay budgeting during placement and floorplanning has

<table>
<thead>
<tr>
<th>Node</th>
<th>Delay (nsec)</th>
<th>Budget A</th>
<th>Budget B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Total Budget</td>
<td>--</td>
<td>17</td>
<td>12</td>
</tr>
</tbody>
</table>

In this paper, we study the problem of the assignment of maximum total budget in a graph. The budgeting problem in a graph is well studied in theory and practice and is widely used in today’s industry and research. Delay budgeting has several applications in design optimization as follows:

- **Design timing closure**: During design optimization flow, timing budget is allocated to each node under a given timing constraint and optimization is applied. If timing constraint is not met, the delay budget is re-allocated [9]. Exploiting the maximal budgeting can lead to earlier convergence.

- **Timing-driven placement and floorplanning**: Delay budgeting during placement and floorplanning has
been extensively studied by several researchers [10, 7, 8]. In timing-driven placement, the goal is to optimize the path delays with fewer number of iterations. Delay budget is assigned to edges in the graph. Per-net delay bounds are considered in order to have a better distribution of delay budgets in the graph. In [6, 7], placement and re-budgeting are combined. The optimization problem of budgeting on the edges in a graph is formulated as a piece-wise linear objective function and solved using a modified graph-based Simplex algorithm ([1]).

- Gate/wire sizing and power optimization- Under timing constraint, gate sizing problem is to find a set of nodes/edges in the graph such that their physical size can be reduced by mapping to smaller cell instances with larger delays from a target library [17, 18]. In general, delay budgeting can be applied during library mapping stage. Delay budget at each node can be exploited to map the node to a smaller cell (or with a lower power consumption) with a larger delay [10].

- VLSI layout compaction- The main objective is to minimize the physical area of the layout. In addition, minimizing wirelength cannot be ignored during the optimization. Concept of budget in such problems is exploited to reduce wirelength [14]. An important constraint in analog IC design is the symmetry constraint in layout. With multiple symmetry constraints, layout compaction is solved using LP solver[15]. In [16], a graph-based simplex method is applied to improve the runtime of linear programming algorithm. LP formulation of compaction is similar to formulation of delay budgeting problem. The space budget is assigned along the x-axis or y-axis to leave a sufficient space for wiring.

- Exploiting slack in high-level synthesis- There are several related work in the area of high-level synthesis where timing slack of the nodes in the data flow graphs are considered for better optimization in area and power. Examples are the algorithms and techniques developed for area minimization in pipelined datapath [21], power minimization under timing constraint [19, 20], etc. In [21], the design entry is a pipelined datapath. In the problem formulation, there are a set of constraints regarding the number of registers and depth of pipeline stages, which are not considered in budgeting on directed acyclic graphs. All the proposed algorithms are heuristic sub-optimal algorithms.

There are heuristic algorithms in literature and industry to solve delay budgeting problem such as VISA [3] and ZSA [4] algorithms. In delay budgeting, the objective is to maximize the value of an expression, which is a function of budgets associated with nodes/edges in a graph. The most popular and efficient algorithm for delay budgeting is zero-slack algorithm [4, 5]. The solution is not optimal and can be far away from optimal result. MISA algorithm proposed in [3] finds the total budget in the graph with a more sophisticated and intuitive technique using maximum independent set in graph. MISA algorithm finds a potential slack which correlates strongly with the total budget in the graph. However, both ZSA and MISA algorithms cannot solve the budgeting problem optimally.

In this paper, we focus on theoretical study of integer delay budgeting problem on the nodes in a directed acyclic graph. Objective function in our 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 and need to be normalized. 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 grid-based global routing. In datapath level, 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 [12]. 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 the budgets are all integer. ZSA and MISA algorithms can be modified to generate integer budgets, but with no guarantee on the optimality of the solutions.

The complexity of integer delay budgeting problem on DAGs has been an open problem since budgeting problem was first formulated by Wong, et. al. in [11]. Applying rounding techniques to LP optimal solution of budgeting problem cannot preserve the optimality of the integer solution. In this paper, we propose our novel efficient graph-based transformation technique to produce optimal integer solution from optimal LP solution. We prove that integer budgeting problem can be solved optimally by transformation from LP relaxation solution to an integer solution in polynomial time \(O(V^2)\) while objective value is still optimal. The preliminary version of this work is published in DAC'03. In this paper, we describe the detailed analysis of our delay budgeting algorithm. In addition, we look at different extensions of the delay budgeting problem, such as maximization of weighted summation of delay budgets assigned to the nodes and additional constraints on lower bound and upper bound on the delay budget allocated to each node. We prove that in both aforementioned extensions, our algorithm can produce an optimal integer solution in polynomial time.

We apply delay budgeting technique in mapping a given datapath on a FPGA platform. For faster compilation and exploiting the architectural features of FPGAs, FPGA vendors provide a relatively rich IP (Intellectual Property) library of arithmetic functions and application-specific operations such as MAC, FFT, and DCT in DSP domain. An example is CoreGen module library of Xilinx™. In behavioral level of description of a design, datapath of an application is mapped to components of a library customized for the target programmable architecture (e.g. FPGA libraries). We applied timing budgeting algorithm in selecting the components of library and mapping to different components of the application such that the design complexity is reduced without violation of timing constraints. Using IP library of FPGAs, we show that the delay budgeting resolves the trade-off between latency of a datapath and area of hardware resources. Our empirical results show that delay budgeting yields a solution with smaller area and faster design time compared to the case in which no delay budgeting is applied. We compare our proposed optimal delay budgeting algorithm with ZSA, a sub-optimal heuristic algorithm. The decrease in complexity of datapath improves the runtime of place and route stage, which is the most time-consuming stage in mapping an application on FPGA platforms. Our experimental results show the effectiveness of budgeting on IP-based application mapping.

The rest of the paper is organized as follows: In Section 2, the problem is formally defined. In Section 3, the budget re-assignment is proposed. Applying budget re-assignment on LP solution of budgeting problem is described in Section 4 and it is proven that the final solution is integer and optimal. In Section 5, two different extensions to the formulation of integer delay budgeting problem are presented on which our algorithm can be applied to produce optimal integer solution. In Section 6, the experimental results on trade-off between latency and area by budgeting technique in FPGA platform are presented. In Section 7, conclusions and some possible future directions are outlined.

## 2 LP Formulation of Delay Budgeting Problem

In a given directed acyclic graph \(G = (V, E)\), associated with each node \(v\), there is a delay variable \(d > 0\). If inputs to node \(v\) are ready at time \(t\), the output of node \(v\) is ready at time \(d + t\). \(b_i\) is the extra potential delay the node can accept. edge \(e_{ij}\) is incident to node \(v_j\) and incident from node \(v_i\). In a directed graph \(G\), edge \(e_{ij}\) is incident to node \(v_j\) and incident from node \(v_i\). \(V_i(i)\) 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. The following are some basic definitions corresponding to a directed acyclic graph \(G\).
**Arrival time of** \( v_i \): Arrival time at node \( v_i \) is defined as the maximum of total delay and budget assigned to the nodes along a path from PI to node \( v_i \) among all the paths from PIs to node \( v_i \). 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 computed as

\[
a_i = \max_{v_j \in V(i)} (a_j + d_i).
\]

Arrival time at a primary output is maximum summation of budget and delay associated with each node along the path from primary input 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 required 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 \( d \) associated with each node \( v_i \) and required time \( T \):

\[
\max \sum_{v_i \in V} b_i
\]

\[
a_j \geq a_i + b_j + d_j \quad \forall e_{ij} \in E
\]

\[
a_i \leq T \quad \forall v_i \in PO
\]

\[
a_i = 0 \quad \forall v_i \in PI
\]

\[
d_i, b_i, T \in \mathbb{Z}_+ \quad \forall v_i \in V.
\]

General LP formulation of budgeting problem is \( \max \{ \sum_{i=1}^{\vert V \vert} x_i \mid Ax \leq b \} \). Constraint matrix \( A \) corresponding to abovementioned LP formulation of budgeting problem is as follows. Variable \( x \) corresponds to \( a_i \) if \( 1 \leq j \leq n \) and corresponds to \( b_j \) if \( n < j \leq 2n \). Each row index corresponds to an edge in graph \( G \). Assume \( A = (a_{ij})_{m \times n} \). For \( 1 \leq j \leq n, a_{ij} = 1 \) if edge \( e_i \) is incident from node \( j \) and \( a_{ij} = -1 \) if edge \( i \) is incident to node \( j \) (\( e_{ij} \) is incoming edge to node \( v_i \)). Otherwise it is set to zero. For \( n < j \leq 2n, a_{ij} = 1 \) is edge \( i \) is incident to node \( j - n \).

In area of linear programming theory, there has been a deep study on the linear programs that have optimal integer solutions. In particular, it is the case for network flow problems. If matrix \( A \) is **totally unimodular**, the linear programming relaxation can solve the ILP, proposed by Heller and Tompkins [2].

**Totally Unimodular Matrix (TU):** A matrix \( A \) is totally unimodular (TU) if every square sub-matrix of \( A \) has determinant \(+1, -1, \) or \( 0 \) [2].

**Lemma 1** If matrix \( A \) is TU, the linear programming relaxation can solve the ILP [2].

A set of sufficient conditions for matrix \( A = (a_{ij})_{m \times n} \) to be totally unimodular is proposed by Heller and Tompkins [2] as follows:

**Lemma 2** A matrix \( A_{m \times n} = (a_{ij})_{m \times n} \) is TU if

- \( a_{ij} \in \{-1, +1, 0\}, \forall a_{ij}, 1 \leq i \leq m, 1 \leq j \leq n. \)
- Each column contains at most two non-zero coefficients, i.e. \( \sum_{i=1}^{m} \mid a_{ij} \mid \leq 2 \).
primary outputs. There has to exist at least one outgoing edge from \( v_i \). Maximal Budgeting Graph (\( G \))

In a maximal budgeting, Lemma 5 be added to the budget of node \( v_i \) from node \( v_i \). By the way of contradiction, we assume that there is no critical outgoing edge from \( v_i \). Proof:

\[ \sum_{i \in M_1} a_{ij} - \sum_{i \in M_2} a_{ij} = 0 \] [2].

By total unimodularity (TU) of coefficient matrix every extreme point of LP relaxation is integral regardless of objective function.

**Theorem 1** The linear programming relaxation of integer budgeting problem gives optimal integer solution if the input graph is a directed path.

The aforementioned sufficient condition does not necessarily hold for general directed acyclic graph other than a directed path. In the following sections, we prove that the integral budgeting problem can be solved optimally in polynomial time, using the solution of the linear programming relaxation problem.

### 3 Delay Budget Re-assignment

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 6 are not satisfied. Some basic definitions used in this section are as follow:

**Definitions:**
- required time at \( v_i \), \( r_i \), is computed as
  \[ \min_{j \in V_{ij}} (r_j - (d_j + b_j)) \]. \( r_i \) is required time at primary outputs in graph \( G \). slack at node \( v_i \) is \( s_i = r_i - a_i \). a-slack of edge \( e_{ij} \), \( \varepsilon_{a_{ij}} \), is: \( (a_{ij} - (d_j + b_j)) - a_i, e_{ij} \in E \). Similarly, r-slack of \( e_{ij} \), \( \varepsilon_{r_{ij}} \), 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 3** In a directed graph \( G \), if \( e_{ij} \in E \) and \( s_i = s_j \), then \( \varepsilon_{a_{ij}} = \varepsilon_{r_{ij}} = \varepsilon_{ij} \).

**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. Maximal budgeting solution \( B_m \) can be obtained by applying different algorithms such as MISA algorithm [3] and ZSA algorithm [4].

**Lemma 4** In \( (G, B_m) \), if the slack of each node is zero, \( B_m \) is a maximal budgeting.

Non-critical edges are referred to as \( \varepsilon \)-edges. According to Lemma 3 the a-slack and r-slack of a \( \varepsilon \)-edge in \((G, B_m)\) are equal, that is \( \varepsilon_{ij} = \varepsilon_{a_{ij}} = \varepsilon_{r_{ij}}, \forall e_{ij} \in (G, B_m) \).

**Lemma 5** 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.

**Proof:** By the way of contradiction, we assume that there is no critical outgoing edge from \( v_i \). \( v_i \) is not a primary output. There has to exist at least one outgoing edge from \( v_i \). If there are \( k \) non-critical outgoing edges from node \( v_i \) then \( \varepsilon_{ij}, j = 1, ..., k \) is not zero. The slack of each node is zero. Therefore \( \min(\varepsilon_{ij}), j = 1, ..., k \) can be added to the budget of node \( v_i \) while all the arrival time and required time constraints for the whole graph are met. Hence we get more budget and this contradicts the definition of maximal budgeting. Similar argument can be applied to prove that at least one incoming edge to each node must be critical.
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 $G$. $G_T = (V,E_T)$, $E_T = E - \{e_{ij}|e_{ij} \neq 0\}$.

In any budgeting on graph $G$, slack of each node and edge must be non-negative or in other words $\alpha \leq T$. 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.

**Feasible Budget Re-assignment on $(G,B_m)$:** In a 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 $B'_m$. Feasible $\beta$-budget re-assignment on $(G,B_m)$ is a feasible budget re-assignment in which the change of budget in each node is either $\pm \beta$ or 0.

After feasible budget re-assignment, the budgeting is maximal and feasible. Assume that in a re-assignment of budget of $\{\pm \beta, 0\}$ at each node in graph $G$, the total amount of change in the budget of the nodes along each critical path is zero. In this case, arrival time at each node $v$ is changed by $k_i \beta$. Since budget of each node changes either $\beta$ or $-\beta$, the change of budget along each critical path from PI to node $v$ is multiple of $\beta$, called $k_i, k_i \in Z$. Theorem 2 presents two sufficient conditions for feasible $\beta$-budget re-assignment.

**Theorem 2** The re-assignment of budget of $\{0, \pm \beta\}$ at each node in graph $(G,B_m)$ is a feasible $\beta$-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, and
- for each $\varepsilon$-edge $e_{ij}, \varepsilon_{ij} \geq (k_i - k_j) \cdot \beta$, where edge $e_{ij}$ is critical. $k_i \beta$ and $k_j \beta$ are the amount of change in total budget along any critical path from PI to node $v$ and $v_j$, respectively.

**Proof:** We prove that after the budget re-assignment of $\{\pm \beta, 0\}, (G,B'_m)$ is a feasible maximal budgeting. Assume $(G_T,B_m)$ is the critical graph before budget re-assignment. $a_i$ is arrival time at node $v_i$ before budget re-assignment. By induction, it can be shown that arrival time at $v_j$ is $a_j + k_j \beta$ after budget re-assignment. $k_j \beta$ is total budget re-assignment along the critical paths from PI to node $v_j$. In Figure 2(a). Edges $e_{ij}$ and $e_{ij}$ are both critical in graph $(G,B_m)$. After budget re-assignment, arrival time at node $v_j$ is $a_j + k_j \beta$. Arrival time at node $v_j$ is $a_j + k_j \beta$. Since along the critical path from PI to PO through edge $e_{ij}$ total change in budget is zero, the amount of change in budget from node $v_j$ until PO is $-k_i \beta = \beta$ if budget of $\pm \beta$ changes at node $v_j$. Since the critical path from PI until node $v_j$ through node $v_i$ to PO is critical, total budget of $k_j \beta + k_i \beta = 0$. Hence $k_i = k_i$. However, non-critical edges in the graph need to be considered. There is a slack of $\varepsilon$ associated with each non-critical edge. In Figure 2(b), edge $e_{ij}$ is a non-critical edge and $\varepsilon_{ij} = a_j - a_i$. After budget re-assignment edge $e_{ij}$ has to remain critical. Based on the second condition in the Theorem $\varepsilon_{ij} \geq (k_i - k_j) \cdot \beta$, arrival time $a_i$ cannot become greater than arrival time $a$. Hence, edge $e_{ij}$ remains critical.

Now assume node $v_i$ is a primary output. Arrival time at node $v_i$ is $a_i + k_i \beta$. $k_i \beta$ is the total change of budget along the critical paths from PIs to node PO. Due to first condition $k_i$ is zero. Hence, arrival time at node $v_i$ does not change after budget re-assignment, i.e. feasibility is satisfied ($\alpha \leq T$).

The arrival time at node $v_j$ is $a_j + k_j \beta$. Similar argument can be applied to show that the amount of change in the required time at node $v_j$ is $r_j - k'_j \beta$, where $k'_j \beta$ is the amount of change in the total budget along the critical paths from primary outputs to node $j$. Slack of node $v_j$ after budget change is $\eta = a_i - k'_j - k_i$. According to Lemma 4, $s_j = r_j - a_i = 0$ since $B_m$ is a maximal budgeting. $k_j \beta + k'_j \beta$ is equivalent to total change of budget along the critical path through node $v_j$, which is zero. Hence slack of node $v_j$ after budget change is still zero. Therefore we have a feasible maximal budgeting.
Lemma 6 \( v_i \sim^c v_j \) iff \( p(v_i) \sim^p p(v_j) \).

Proof: If \( v_i \sim^c v_j \), there exists node \( v_l \) such that \( v_i \sim v_l \) and \( v_j \sim v_l \). Hence, nodes \( v_i \) and \( v_j \) share a parent, i.e., \( p(v_i) \sim p(v_l) \). According to transitive property in child and parent relation, it can be shown that \( p(v_l) \sim p(v_j) \).

Lemma 7 In \((G, B_m)\), if \( v_i \sim^p v_j \), arrival time at nodes \( v_i \) and \( v_j \) are equal; \( a_i = a_j \).

Proof: Nodes \( v_i \sim^p v_j \). Let \( v_k \) be the child node of nodes \( v_i \) and \( v_j \). Since \( v_i \) is a parent of node \( v_k \), \( a_k = a_i + b_k + d_k \). Similarly \( a_k \) is equal to \( a_j + b_k + d_k \). Hence, \( a_i = a_j \). If \( v_i \) and \( v_j \) do not share a common child, due to transitive property in equivalent parent relation, \( v \sim^p v_k \sim^p ... v_i \sim^p v_j \), arrival time at \( v_i \) is equal to arrival time at \( v_j \).

According to Lemma 5, each node is incident to/from a critical edge. Consider node \( v \) in graph \( G = (V, E) \). Let \( S_p(v_i) = \{ v_j | v_i \sim^p v_j \} \) be a parent set. Let \( v_i \) be a child node of \( v_i \). \( S_c(v_i) = \{ v_j | v_j \sim^c v_i \} \). According to Lemma 6, sets \( S_p(v_j) \) and \( S_c(v_i) \) are a pair of sets such that all the child nodes of the nodes in \( S \) 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 \( S_c(v_i) \) 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 3. The followings are the propositions regarding the parent-child set in \((G, B_m)\).

Lemma 8 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 \( \forall v_i \in V, d_i > 0 \).
Proof: Since \( v_i \sim_p v_j \), by Lemma 7, \( a_i = a_j \). If there is a critical path between \( v_i \) and \( v_j \), then \( a_i \) cannot be equal to \( a_j \) according to definition of arrival time and critical edges with assumption of \( q > 0 \). Hence there cannot exist any critical path between any two nodes in the parent set.

Lemma 9 If nodes \( v_i \sim_p v_j \), there is no directed critical path between \( v_i \) and \( v_j \) if \( \forall v_l \in V, d_l > 0 \).

Proof: Assume that there is a critical path \( P_{ij} = \{v_i, \ldots, v_k, v_j\} \) connecting nodes \( v_i \) and \( v_j \). Since \( v_k \) is \( p(v_j) \), \( v_k \) belongs to the parent set according to Lemma 8. Hence \( p(v_l) \sim_p v_k \). That is \( a_k = a_{p(v_l)} \). However, since there is path between \( v_i \) and \( v_k \) and delay of each node is non-zero, \( a_{p(v_l)} < a_{v_l} < a_{v_k} \). Therefore, by the way of contradiction, the path \( P_{ij} \) cannot be critical.

Lemma 10 In a parent-child set \((S_p, S_c)\), \( S_p \) and \( S_c \) do not intersect if \( \forall v_l \in V, d_l > 0 \).

Proof: Assume that the two sets intersect at node \( v_k \). Since node \( v_k \) is in set \( S_p \), node \( v_k \) has at least a child in set \( S_c \), say node \( v_l \). Therefore, there is an edge from node \( v_k \) to node \( v_l \) both belonging to \( S_c \). This contradicts Lemma 9.

![Figure 3: \( \varepsilon \)-edges with respect to Parent-Child Set \((S_p, S_c)\).](image)

Let \( \beta \)-budget exchange in parent-child set \((S_p, S_c)\) be decreasing the budget of the nodes in \( S_p \) by \( \beta \) and increasing budget of nodes in \( S_c \) by \( \beta \), \( \beta > 0 \).

Lemma 11 In a given \((S_p, S_c)\) in \((G, B_m)\), if \( \beta \leq \min(e_{ij}) \), where \( e_{ij} \) is an \( \varepsilon \)-edge with \( v_j \in S_c \) and \( v_i \notin (S_p, S_p) \) (incoming \( \varepsilon \)-edges to \( S_c \)), the \( \beta \)-budget exchange is a feasible \( \beta \)-budget re-assignment in \((G, B_0)\).

Proof: We show that the sufficient conditions in Theorem 2 are satisfied during budget exchange between a parent-child set. Since there is no critical path between any two nodes in \( S \) or \( S_p \), the critical paths in \( S_c \cup S_p \)
consist of two nodes, one in parent set and the other in child set. At each parent node \( v_i \), the amount of change in arrival time is \(-\beta\). At each child node \( v_j \), the amount of change in arrival time is zero. Hence, along each critical path in this subgraph, the total amount of change in budget is zero. Also, there is no change in budget or arrival time at any other nodes outside the parent-child set in graph \( G \). Therefore the first sufficient condition in Theorem 2 is satisfied. The \( \epsilon \)-edges can be categorized based on where the two ends of the edges are located. Figure 3 shows all different possible such edges with respect to a given parent-child set. At each child node \( v_j \), the amount of change in arrival time is zero. Therefore, arrival time at a child node and hence the criticality of the edges connecting the child nodes to the rest of the graph remain unchanged. Similarly, the criticality of incoming edges to parent nodes are unchanged after budget exchange, i.e., \( \epsilon \)-edges 3 and 2 remains non-critical. The inequality \( \epsilon_1 \geq \beta \) is satisfied as well. For \( \epsilon \)-edge 4, the inequality \( \epsilon_1 \geq \beta \) is held since \( \beta \geq \epsilon \) for all incoming \( \epsilon \)-edges to child set. There cannot exist any \( \epsilon \)-edges between two parent nodes, two child nodes, or between a child and a parent node. Hence the second sufficient condition in Theorem 2 is satisfied.

Similarly, the budget can be increased by \( \beta \) in parent set and reduced by \( \beta \) in child set. This is called \((-\beta)\)-budget exchange in \((S_p, S_c)\). Lemma 11 can be adjusted to be applied for \((-\beta)\)-budget exchange on parent-child set as well. In this paper, we apply \( \beta \)-budget exchange on a given parent-child set. In the next section, we apply \( \beta \)-budget re-assignment on LP solution which is a maximal budgeting on \( G \) in order to obtain integer solution.

### 4 Integer Solution to Delay Budgeting Problem

\((G, B^*)\) 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 define \( \beta \) in \( \beta \)-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 \(|B|\).

**Integral sequence:** A sequence of nodes \( IS_n = <v_1, v_2, ..., v_n> \) along a critical path in \((G, B^*)\) is called Integral Sequence if \( a_1, a_n \in Z_+ \) and \( a_2, ..., a_{n-1} \notin Z \).

#### Lemma 12
The total budget of the nodes along any integral sequence in \((G, B_n)\) is integer if \( d_i, T \in Z \).

**Proof:** Since the arrival time of the nodes at the two ends of an integral sequence \( IS_n \) is integer, \( \sum_{j \in IS_n} (d_j + b_j) \in Z_+ \). Since each \( d_j \) is integer, \( \sum_{j \in IS_n} b_j \in Z_+ \).

**Corollary 1** The total budgeting on any critical path from PI (Primary Input) to PO (Primary Output) is integral.

Based on Lemma 12, each node with fractional budget belongs to an integral sequence. Hence, within an integral sequence, it is sufficient to re-assign the fractional budget only on the nodes in an integral sequence. On the other hand, in graph \( G \), there are several integral sequences connected to each other. Therefore in re-assigning the budget between the nodes, the required conditions in Theorem 2 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 in \((G, B^*)\) to obtain integer solution. Since the budget re-assignment needs to be applied between the nodes with fractional budget, we reduce the graph \((G, B)\) to graph \( G_f \), the fractional adjacency graph defined as follows:

**Fractional Adjacency Graph** : Graph \( G_f \) is the fractional adjacency graph corresponding to given graph \((G, B^*)\). The nodes in graph \( G_f \) are a subset of nodes in graph \( G \) that have non-integer (fractional)
budget. A critical edge between two nodes in graph $G_f$ represents the existence of a directed critical path between two nodes in graph $G$ such that there is no fractional budget along the path and arrival time of each node along the path is not integer. There is an $\epsilon$-edge between two nodes $v_i$ and $v_j$, if there is no critical path between the two nodes but at least a path with $\epsilon$-edges along the path. Among all different paths between the two nodes, the minimum of total $\epsilon$ value of the $\epsilon$-edges along each path is the $\epsilon$ value of the $\epsilon$-edge in graph $G_f$.

Figure 4: In graph $G_f$, nodes $v_i$ and $v_j$ share a child (Node $v_k$) while there is a directed path from $v_i$ to $v_j$.

Two adjacent nodes $v_i$ and $v_j$ in graph $G_f$ represent the two immediate nodes on a directed critical path in graph $G$ with fractional budget, both belonging to same integral sequence. 

$\beta$-budget re-assignment is applied on graph $G_f$ 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 $\beta$ 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 $G$. Similar argument can be applied to graph $G_f$ as follows:

**Lemma 13** In graph $G_f$, if node $v_i \sim_p v_j$, the fractional values of arrival time at both nodes are equal, i.e., $a_i - [a_i] = a_j - [a_j]$.

**Proof:** Assume $v_i \sim_p v_j$. Let $v_k$ be the child node of both nodes $v_i$ and $v_j$. Arrival time at node $v_k$ is equal to fractional value of summation of fractional value of arrival time at node $v_i$ and fractional value of budget at node $v_k$. Budget of the nodes along the critical path from $v_i$ to $v_k$ in graph $G$ are all integer. Similarly, arrival time at node $v_k$ is equal to fractional value of summation of fractional value of arrival time at node $v_j$ and fractional value of budget at node $v_k$. Hence, $a_i - [a_i] = a_j - [a_j]$. If $v_i$ and $v_j$ do not share a child node, due to transitivity in parent relation, we still have $a_i - [a_i] = a_j - [a_j]$. \hfill \blacksquare

**Lemma 14** If nodes $v_i \sim_p v_j$ in graph $G_f$ and there is a directed critical path between nodes $v_i$ and $v_j$ in graph $G$, there has to exist at least one node on the path between the nodes $v_i$ and $v_j$ in graph $G$.

Figure 5: In graph $G_f$, $v_i \sim_p v_j$ while there is a directed path from $v_i$ to $v_j$. 

11
Proof: Assume there is a path between node $v_i$ and $v_j$ in graph $G$. Let node $v_k$ be the child node of nodes $v_i$ and $v_j$. There are two paths from node $v_i$ to $v_k$, one is the direct edge $e_{ik}$ and the other is the path $(v_i, v_j)(v_j, v_k)$. See Figure 4. The fractional value at the node $v_k$ from the first path is $a_i - [a_i]$ and from the other path is $a_i - [a_i] - a_j + [a_j]$. According to Lemma 13, these two values need to be equal. This is possible iff $a_i - [a_i] = 0$ which contradicts that $e_{jk} \in E(G_f)$. Therefore there has to exit at least one node say $v_l$ on the path from $v_i$ to $v_j$ such that $a_i - [a_i] + a_l - [a_l] = 0$. Similarly if the two nodes $v_i$ and $v_j$ do not have a same child, we can prove that the total fractional value on the path from $v_i$ to $v_j$ including $v_j$ needs to be integral, i.e. there has to exist at least one node on the path between $v_i$ and $v_j$. Figure 5 shows such a case.

![Parent Set $S_p$ and Child Set $S_c$](image)

Figure 6: Parent-Child Set $(S_p, S_c)$ in graph $G_f$ of graph $G$.

The set $S_p(v_i) = \{ j | v_j \sim_p v_i \}$ is the set of nodes in graph $G_f$ such that each node shares at least a common child with another node in $S(v_i)$. The set $S_c(v_i) = \{ j | v_j \sim_c v_i \}$ is the set of nodes in which each node in the set shares a parent at least with one another node in the set. In Figure 6, a parent-child set in $G$ is shown.

According to Lemma 14, Lemma lemma:int is derived.

**Lemma 15** Set $S_p(v_i)$ and $S_c(v_j)$ do not intersect ($e_{ij} \in E(G_f)$).

**Proof:** Assume that the two sets intersect, i.e., there is a node $v_l$ belonging to both sets. Since node $v_l$ is in set $S_c(v_j)$, it has at least one parent, say $v_l \in S_p(v_i)$. Therefore there is an edge from node $v_l$ to node $v_k$. On the other hand, $v_k \in S_p(v_i)$. That is there is a direct edge between two nodes $v_l, v_l \in S_p(v_i)$. This contradicts Lemma 14.

On a given parent-child set in graph $G_f$, we apply $\beta$-budget exchange. If fractional budget in graph $G_f$ 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 $\epsilon$-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 2 are satisfied in a given graph $G_f$ as well.

**Lemma 16** $\beta$-budget exchange on a parent-child set in graph $G_f$ is a feasible $\beta$-budget re-assignment if $\beta \leq \min(\epsilon_{ij}, \alpha)$, where $\epsilon_{ij}$ is $\epsilon$-edge. $e_{ij}$ is an incoming edge to child set. $\alpha_i$ is the fractional value at parent nodes.
Proof: In Figure 6, a set of parent-child set is shown in a given graph $G_f$. In a budget exchange on the set, there is an alternative $\pm \beta$ budget exchange along each critical edge in graph $G$. $\beta$ corresponds to total change of budget along the critical paths from $PI$ to node $v_i$. At each child node $v_j$, the corresponding $k_i = -1$ when budget in parent set is decreased by $\beta$. Hence the first sufficient condition in Theorem 2 is satisfied. We prove that as long as $\beta \leq \varepsilon$, the budget exchange is a feasible $\beta$-budget re-assignment on a given graph $G$. There are 8 possible type of $\varepsilon$-edges with respect to $(S_p, S_c)$. At each edge, we check the inequality defined in Theorem 2 after budget exchange. $\varepsilon$-edges 2 and 3 will not change since the arrival time at the child set and incoming edges to parent set are not affected by budget exchange. At $\varepsilon$-edge 1, the inequality $\varepsilon \geq (-1 - 0)\beta$ is True for any $\beta > 0$. At $\varepsilon$-edge 5, the inequality $\varepsilon \geq (-1 - (-1))\beta = 0$ is True for any $\beta > 0$. At $\varepsilon$-edge 6, the slack will not change since the arrival time at child node does not change. At $\varepsilon$-edge 8, the inequality $\varepsilon \geq (1 - (-1))\beta$ is True for any $\beta > 0$. At $\varepsilon$-edges 7 and 4, the arrival time at the nodes incident from $\varepsilon$-edges do not change. $\varepsilon \geq (0 - (1))\beta$ is satisfied as well. Therefore both sufficient conditions in Theorem 2 are satisfied.

If $\beta$ is less than the fractional value of budget in parent nodes, after budget re-assignment, arrival time at parent node is reduced by $\beta$. Hence, if $\beta$ is equal to fractional value of the arrival time, arrival time at all parent nodes become integer. On the other hand, $\beta$ need to be at most as large as the minimum available budget in parent nodes.

In Figure 7, an $\varepsilon$-edge incident to a child node is shown. Let $\alpha_i$ and $\alpha_j$ be the fractional value of arrival time at nodes $v_i$ and $v_j$, respectively. In $\beta$-budget re-assignment, if $\beta = \alpha_i$, for $\varepsilon \geq 1$, $\varepsilon > \beta$ is True. Assume $\varepsilon < 1$. The value of $\varepsilon$ is computed as follows:

$$\varepsilon_{jp} = \begin{cases} 
\alpha_i - \alpha_j & \text{if } \alpha_i > \alpha_j \\
1 + \alpha_i - \alpha_j & \text{if } \alpha_i < \alpha_j
\end{cases} \quad (7)$$

When $\alpha_i < \alpha_j$, $\varepsilon > \alpha_i$. Since $\beta = \alpha_i$, $\varepsilon > \beta$. Hence the inequality of Theorem 2 is held. Hence $\beta$ value in $\beta$-budget re-assignment can be computed independent of $\varepsilon$-edges incident to child set as follows:

Lemma 17 Let $(S_p, S_c)$ be a parent-child set with $\alpha_p$, the fractional value at the arrival time at the parent nodes. Assume that $\alpha_p$ is the smallest fractional value of arrival time at all the nodes in graph $G_f$. $\beta$-budget exchange of $\beta = \alpha_p$ from parent nodes to child nodes is a feasible budget re-assignment.
Proof: In order to be able to re-assign budget of $\beta$ from parent nodes, each parent node must have at least budget of $\beta$, i.e. $\forall v_j \in S_p, b_j \geq \beta$. Assume that there is a node $v_j \in S_p$ such that $b_j \leq \beta$, hence $b_j \leq \alpha_p$. In this case, arrival time at parent of node $v_j$ is $\alpha_p - b_j < \alpha_p$ and this contradicts the fact that fractional value of arrival time at no other nodes other than parent nodes can be as small as $\alpha_p$. Hence each $b_j \geq \beta$. Next, consider $\epsilon$-edges connected to $(S_p, S_c)$. According to Lemma 16, only two types of $\epsilon$-edges, $\epsilon$-edges 4 and 7 as shown in Figure 3 are under the condition that $\epsilon$ value of such edges have to be larger than $\beta$ for edges with $\epsilon < 1$. According to Equation 7, since parent set has the smallest fractional value ($\alpha < \alpha_j$), $\beta < \epsilon$. This ends the proof that the budget re-assignment is feasible.

After budget re-assignment on parent-child set $(S_p, S_c)$, arrival time at each parent node becomes integer with $\beta = \alpha_p$. If budget of any node in parent set or child set becomes integer, the node is removed from $G$. In this budget re-assignment, an integer budget of any node in graph $G$ never becomes fractional. Hence no node is added to graph $G_f$ after budget re-assignment. Since arrival time at a parent node becomes integer, all the edges connecting the parent nodes to the child nodes are removed from graph $G$. Similarly no edge is added to graph $G_f$ after budget re-assignment.

An important fact is that after budget re-assignment, the parent nodes do not have any outgoing edges in updated graph $G_f$. Hence, the corresponding nodes cannot become parent nodes anymore. Therefore we have the following lemma:

Lemma 18 Each node in graph $G_f$ can only be once in a parent set during sequential parent-child budget re-assignment.

Note that after each $\beta$-budget exchange, the outgoing edges of parent nodes are removed. No more outgoing edges are added to parent nodes in $G_f$ since arrival time at parent nodes are integer. On the other hand, integer budget of a node never becomes fractional after any $\beta$-budget exchange. Since each node can only once appear in a parent set, the number of parent-child which can be generated followed by budget re-assignment on each set is $O(|V|)$, where $V$ is set of nodes in graph $G$.

Theorem 3 Sequentially generating parent-child set followed by $\beta$-budget re-assignment in the order of increasing fractional value of arrival time at parent nodes of the parent-sets with $\beta = \alpha_p$, $G_f = \emptyset$ in $O(|V|)$.

If graph $G_f = \emptyset$, the budget of all the nodes in graph $G$ are integer. Hence, Theorem 3 shows that a maximal integer solution can be obtained from LP solution using $\beta$-budget exchange on graph $G$. The following lemma proves that during budget re-assignment optimality is preserved.

Lemma 19 In graph $G_f$ corresponding to $(G, B')$, $|S_p(v_i)| = |S_c(v_j)|$ if $e_{ij} \in G_f$.

Proof: Assume that there are more number of nodes in one of the sets , say $S(v_j)$. After budget re-assignment of minimum budget, say $f_{min}$, the total budget changes by $|S_p(v_j)| \cdot f_{min} - |S_p(v_i)| \cdot f_{min}$. This contradicts the optimality of budget in $(G, B')$.

Theorem 4 In any feasible $\beta$-budget re-assignment on parent-child set $(S_p, S_c)$ 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 . Each parent-child set construction takes $O(|E|)$, budget re-assignment takes $O(|E|)$. Updating graph $G$ takes $O(|E|)$. This repeats $O(|V|)$ times. However, by amortized analysis we see that the complexity of $O(|E|)$
during the process applies to a set of edges during the current iteration and then those edges are removed from graph $G_f$ before the next budget re-assignment. Hence the total complexity is $O(|E|) = O(|V|^2)$. 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., $|B'|$.

**Theorem 5** The solution to linear programming relaxation problem of integer delay budgeting problem on graph $G = (V,E)$ can be transformed to equivalent integer solution in polynomial time ($O(|V|^4)$) with same objective value.

5 Extension of Integer Delay Budgeting Problem

We proved that the maximum integer delay budgeting problem as formulated in Section 2, is polynomially solvable. In this formulation, there are two major simplifications. First, the objective function is simply a summation of the delay budgets on the nodes. This means that the objective is independent on the type of the operation on each node. Depending on the type and complexity of the operation at each node, the extra budget can have a different impact. We extend the problem to maximization of weighted summation of delay budgets assigned to the nodes. We assume that based on the complexity and type of operation, a non-negative weight is given for each node. This value determines the rate of relaxation on the structure of the component and/or synthesis effort on the component for each extra delay budget assigned to the node. The second important simplification in the original problem is that the delay budget assigned to a node can be unlimited. However, in reality, the delay budget can be exploited within a certain range and beyond that range, it is more beneficial to assign the remaining budget to other nodes in the graph. Both extensions are still integer linear programming problems. The formulation of the extended integer delay budgeting problem is:

\[
\text{Max } \sum_{v_i \in V} w_i b_i
\]

(8)

\[
a_j \geq a_i + b_j + d_j \quad \forall e_{ij} \in E
\]

(9)

\[
a_i \leq T \quad \forall v_i \in PO
\]

(10)

\[
a_i = 0 \quad \forall v_i \in PI
\]

(11)

\[
b_i \leq u_i \quad \forall v_i \in V
\]

(12)

\[
b_i \in Z
\]

(13)

The following propositions prove that in both aforementioned extensions of integer delay budgeting problem, the optimal integer solution can be obtained using our algorithm.

**Lemma 20** For each parent-child set $(S_p, S_c)$ in fractional Adjacency graph $(G_f)$ corresponding to $(G, B')$, the condition $\sum_{v_i \in S_p} w_i = \sum_{v_j \in S_c} w_j$ is satisfied.

**Proof:** Assume that the condition does not hold in $(G, B)$. Assume $\sum_{v_i \in S_p} w_i > \sum_{v_j \in S_c} w_j$. After $\beta$-budget re-assignment of minimum budget, say $f_{\text{min}}$, the total budget changes by $f_{\text{min}} \cdot \sum_{v_i \in S_c} w_i - f_{\text{min}} \cdot \sum_{v_i \in S_p} w_i$. Hence, total value of objective increases and this contradicts the optimality of budget in $(G, B)$.

Based on this lemma, applying budget re-assignment iteratively on $G_f$ does not change the value of objective. Hence the integer solution is optimal.

**Lemma 21** In budget re-assignment algorithm on $G_f$, the budget of each node never exceeds the corresponding upper bound on the budget at each node.
Proof: The order based on which the parent-child set is constructed and budget re-assignment is applied, depends on the fractional value of arrival time at the nodes. Due to this ordering, when budget of a node is increased by $\beta$ in the child set, it is guaranteed that the total budget at the node cannot exceed the upper bound on this node. When budget re-assignment is applied on a parent-child set, the fractional value of arrival time at each child node is either zero or greater than the fractional value of arrival time at parent set ($q_i$). After budget re-assignment of $\beta$, the summation of fractional value of the budget at each node and the increase in budget ($\beta$) is at most 1. Hence, the total budget at each child node never exceeds its upper bound.

The lower bound on the budget can be added to the original delay of each node and if the initial solution remains feasible under a given timing constraint, we apply the integer delay budgeting algorithm to assign the extra delay budget to the nodes.

In this paper, we formulated the delay budgeting problem as an integer linear programming problem. The main assumption in this problem and its aforementioned extensions is that the objective value increases linearly with any unit of delay budget assigned to each node. However, in several applications, the gain is obtained only if the budget is a certain value. We call this problem discrete budgeting which can be formulated as a mixed 0–1 integer linear programming problem as follows:

\[
\begin{align*}
\text{Max} & \quad \sum_{j \in V} w_j x_j \\
a_j & \geq a_i + b_j x_j + d_j \quad \forall e_{ij} \in E \\
a_i & \leq T \quad \forall v_i \in PO \\
a_i & = 0 \quad \forall v_i \in PI \\
x_i & \in \{0, 1\}
\end{align*}
\]

In [22], it is proved that this problem is an NP-hard problem and an approximation algorithm on a rooted tree has been proposed. This is out of the scope of this paper and our experiments in this paper are only based on regular integer delay budgeting. In the next section, we show that the application on which our technique is applied, the regular unit change in the budget has almost linear correlation with the objective function. Hence, delay budgeting as formulated and solved in this paper can be applied to solve the delay budget assignment.

6 Application

Delay budgeting algorithm is very generic and can be applied in different design tasks at different stages of CAD flow such as gate sizing in logic synthesis, timing optimization in placement, and library mapping in datapath level. In this section, we apply integer delay budgeting in mapping datapath of an application on FPGA platform. Delay budgeting is exploited in library mapping. First we describe the experimental setup and then we present some experimental results applied to some DSP benchmarks. The results show that early management of timing budget on IPs can lead to a faster compilation in physical implementation level.

6.1 Experimental Setup

Mapping sub-designs to already existing pre-optimized and synthesized components is an unavoidable scheme in design automation flow of embedded systems. This is called IP-based (or core-based) implementation of an application on a target platform. In general, in system level, blocks and macros are basic system design elements. IP-based design brings scalability, robustness, and predictability in design flow. IPs or customized cores are modeled as black boxes through synthesis of the system. Hence characterizing the timing and other
quality factors of such optimized blocks (mostly belonging to a specific technology-dependent library of components) is an important issue. Especially in programmable systems such as FPGAs, design and market of soft IPs are growing rapidly, hence providing a rich library of various functional components. Along with this growth, design automation and synthesis flow need to be able to exploit the existing libraries with better design planning.

In Figure 8, CAD flow of IP-based (or core-based) mapping of 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 9 demonstrates a trade-off between the latency and the area of a CoreGen 16-bit multiplier mapped on FPGA VirtexE, Xilinx. Slices are the logic blocks in VirtexE FPGA series which consist of registers, LUTs (lookup tables), and other specific features.

![Diagram of mapping an application on FPGA using IP library](image-url)

**Figure 8: Mapping an Application on FPGA Using IP Library.**

We start from a DAG representation of an application. Each node corresponds to a computation in data path. Benchmark in our experiments is a set of some standard DSP benchmarks. The type of computations are multiplier, adder, subtractor, division, and shifter. We assume all the datapaths 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. After library mapping and synthesis, the whole circuit is placed and routed on a FPGA device. We used *ISE 4.1* place and route tool provided by *Xilinx*™.

Among different computations in the applications, CoreGen has a relatively complete library (See Figure 9). We conducted two sets of experiments. Once we applied our optimal delay budgeting and once we applied a heuristic budgeting (ZSA like) to distribute the latency in graph.
6.2 Experimental Results

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

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Nodes</th>
<th>Latency</th>
<th>Slices</th>
<th>LUTs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diffeq</td>
<td>10</td>
<td>18</td>
<td>780</td>
<td>1030</td>
</tr>
<tr>
<td>ARF</td>
<td>28</td>
<td>20</td>
<td>1982</td>
<td>2476</td>
</tr>
<tr>
<td>FDCT</td>
<td>42</td>
<td>14</td>
<td>2044</td>
<td>1734</td>
</tr>
<tr>
<td>EWF</td>
<td>34</td>
<td>25</td>
<td>1138</td>
<td>1472</td>
</tr>
<tr>
<td>DCT</td>
<td>33</td>
<td>14</td>
<td>1618</td>
<td>1338</td>
</tr>
</tbody>
</table>

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 at the output is the original latency reported in Table 1. The excess latency is distributed in graph using delay budgeting algorithm. We use both exact (our optimal method) and heuristic (ZSA like) method. However, when no budgeting is applied, each operation is mapped to the library component with minimum delay. Area (number of used slices of FPGA device) and place-and-route runtime and total budget are reported. The results show that place and route compilation time of application DCT can speedup up to two times when optimal budgeting is applied. In all applications, applying budgeting reduces the compilation time and the number of resources. On average, the area is reduced by 5.5% while the total budget in the graph is increased by 27%.

The topology and connectivity in applications affect the distribution of delay budget in the graph. If most of the paths in the graph are critical paths, there is not much timing slack in the graph in order to be able to compare different delay budget distribution and its effect on component selection and library mapping. In the second set of experiments, we assume that the timing constraint at the output of each application is the original latency reported in Table 1 plus the excess latency ($\Delta T$) applied to the circuit. Therefore, depending on $\Delta T$, more timing slack is injected to the graph before applying different delay budgeting algorithms. Table 3 shows the area and PAR runtime for different excess delay ($\Delta T$) of 2, 4, and 6 clk cycles. The Imp column shows
<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Runtime Area</th>
<th>No-Budget</th>
<th>ZSA</th>
<th>Opt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diffeq</td>
<td>area(slices)</td>
<td>780</td>
<td>740</td>
<td>700</td>
</tr>
<tr>
<td>PAR(sec)</td>
<td>15</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Budget</td>
<td>-</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>ARF</td>
<td>area(slices)</td>
<td>1982</td>
<td>1806</td>
<td>1803</td>
</tr>
<tr>
<td>PAR(sec)</td>
<td>45</td>
<td>42</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>Budget</td>
<td>-</td>
<td>32</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>FDCT</td>
<td>area(slices)</td>
<td>2044</td>
<td>1867</td>
<td>1734</td>
</tr>
<tr>
<td>PAR(sec)</td>
<td>48</td>
<td>39</td>
<td>39</td>
<td></td>
</tr>
<tr>
<td>Budget</td>
<td>-</td>
<td>14</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>EWF</td>
<td>area(slices)</td>
<td>1138</td>
<td>1094</td>
<td>1016</td>
</tr>
<tr>
<td>PAR(sec)</td>
<td>24</td>
<td>21</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>Budget</td>
<td>-</td>
<td>2</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>DCT</td>
<td>area(slices)</td>
<td>1338</td>
<td>1091</td>
<td>1032</td>
</tr>
<tr>
<td>PAR(sec)</td>
<td>38</td>
<td>19</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>Budget</td>
<td>-</td>
<td>24</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>area(slices)</td>
<td>1456</td>
<td>1327.6</td>
<td>1257</td>
</tr>
<tr>
<td>PAR(sec)</td>
<td>34</td>
<td>26.2</td>
<td>22.8</td>
<td></td>
</tr>
<tr>
<td>Budget</td>
<td>-</td>
<td>14.8</td>
<td>18.8</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Area (#slices-logic blocks), total Budget, and Runtime of Place-and-Route (sec) vs. delay budget (clk cyc).

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Runtime Area</th>
<th>∆T=2 clk cycle</th>
<th>∆T=4 clk cycle</th>
<th>∆T=6 clk cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ZSA</td>
<td>Opt</td>
<td>Imp</td>
</tr>
<tr>
<td>Diffeq</td>
<td>area(slices)</td>
<td>708</td>
<td>652</td>
<td>8%</td>
</tr>
<tr>
<td></td>
<td>PAR(sec)</td>
<td>14</td>
<td>9</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td>Budget</td>
<td>8</td>
<td>12</td>
<td>50%</td>
</tr>
<tr>
<td>ARF</td>
<td>area(slices)</td>
<td>1670</td>
<td>1665</td>
<td>0.3%</td>
</tr>
<tr>
<td></td>
<td>PAR(sec)</td>
<td>42</td>
<td>25</td>
<td>1.68</td>
</tr>
<tr>
<td></td>
<td>Budget</td>
<td>36</td>
<td>48</td>
<td>33%</td>
</tr>
<tr>
<td>FDCT</td>
<td>area(slices)</td>
<td>1728</td>
<td>1491</td>
<td>14%</td>
</tr>
<tr>
<td></td>
<td>PAR(sec)</td>
<td>36</td>
<td>36</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Budget</td>
<td>22</td>
<td>34</td>
<td>54%</td>
</tr>
<tr>
<td>EWF</td>
<td>area(slices)</td>
<td>1058</td>
<td>982</td>
<td>7.2%</td>
</tr>
<tr>
<td></td>
<td>PAR(sec)</td>
<td>20</td>
<td>17</td>
<td>1.17</td>
</tr>
<tr>
<td></td>
<td>Budget</td>
<td>4</td>
<td>10</td>
<td>150%</td>
</tr>
<tr>
<td>DCT</td>
<td>area(slices)</td>
<td>1038</td>
<td>996</td>
<td>4%</td>
</tr>
<tr>
<td></td>
<td>PAR(sec)</td>
<td>20</td>
<td>15</td>
<td>1.33</td>
</tr>
<tr>
<td></td>
<td>Budget</td>
<td>42</td>
<td>48</td>
<td>14%</td>
</tr>
<tr>
<td>Average</td>
<td>area(slices)</td>
<td>1240</td>
<td>1157.2</td>
<td>7%</td>
</tr>
<tr>
<td></td>
<td>PAR(sec)</td>
<td>26.2</td>
<td>20.6</td>
<td>1.27</td>
</tr>
<tr>
<td></td>
<td>Budget</td>
<td>19.2</td>
<td>26.4</td>
<td>37.5%</td>
</tr>
</tbody>
</table>

Table 3: Area (#slices-logic blocks), total Budget, and Runtime of Place-and-Route (sec) vs. delay budget (clk cyc). Column Imp compares optimal over heuristic (ZSA). It indicates the percentage of improvement for area and budget and the ratio of runtime for PAR runtime.
the percentage of improvement in area in different delay budgeting computed as \(\frac{\text{Area(Heu)} - \text{Area(opt)}}{\text{Area(Heu)}} \times 100\). Similarly Imp is computed for total budget. The Imp column computes the improvement in runtime as ratio of \(\frac{\text{PAR runtime(Heu)}}{\text{PAR runtime(opt)}}\).

The results show the average improvement in area for 7%, 10% and 14.6% in terms of number of slices when optimal algorithm is used for budgeting compared to area resulted by heuristic delay budgeting for different \(\Delta T\). The larger \(\Delta T\), the more delay budget is distributed. Although budget increases significantly by \(\Delta T\), the improvement in area is not as significant as budget. One reason is that there does not necessarily exist another component in the target library for large delay budget. For example in FDCT, there are some multipliers on non-critical path with large delay budget which is not exploited in library mapping. Although the area of applications by optimal delay budgeting is always smaller than the area resulted by heuristic method by 10% 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. On average for excess delay budgeting of 6 cycles, the runtime of place and route gets faster by factor of 1.7. Although speedup in PAR runtime were not significant in smaller applications, due to lesser complexity and smaller structure, the effect on runtime for place and route can be more visible when these applications are integrated into larger systems and mapped on large FPGAs.

As a result, delay budgeting gives the flexibility of mapping the applications to components in the target library with simpler structure and smaller area. 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.

7 Conclusion and Future Work

General delay budgeting can be solved using linear programming solver. However, due to numerical instability and discrete behavior of libraries of components, integer solution is required. In this paper, using optimal solution to LP relaxation of budgeting problem, we transform the solution to optimal integer solution. For this purpose, we introduce budget re-assignment in a directed acyclic graph. We re-assign the fractional value of budget associated with the nodes in the graph such that budget of each node becomes integer. We prove that during this transformation \(O(V^2)\), objective value from optimal LP solution does not change. Hence an optimal integer solution is obtained in polynomial time. In this paper, we describe the detailed analysis of our delay budgeting algorithm. In addition, we look at different extensions of the delay budgeting problem, such as maximization of weighted summation of delay budgets assigned to the nodes and additional constraints on lower bound and upper bound on the delay budget allocated to each node. We prove that in both aforementioned extensions, our algorithm can produce an optimal integer solution in polynomial time.

We applied our budgeting technique in mapping of applications on FPGA device. We applied timing budgeting algorithm in selecting the components of library and mapping to different components of the application such that the design complexity is reduced without violation of timing constraints. Using IP library of different computations, delay budget is exploited to improve the area and hence, to speedup the runtime of place-and-route. Our experimental results show the effectiveness of budgeting on IP-based application mapping. Our optimal algorithm outperforms ZSA algorithm [4] in terms of area and compilation runtime significantly.

Our polynomial algorithm in based on optimal LP solution. Developing a polynomial time graph-based algorithm for integer delay budgeting is the current problem we are working on. Discrete budgeting is another challenging problem which needs to be studied and intuitive heuristic algorithms need to be developed for variations of this problem. Other future directions are delay budgeting problem in pipelined datapaths and...
resource-shared datapaths in IP-based design implementation.

References


