Design Closure Driven Delay Relaxation Based on Convex Cost Network Flow*

Chuan Lin
Magma Design Automation
Santa Clara, CA 95054
clin@magma-da.com

Aiguo Xie
Calypto Design Systems
Santa Clara, CA 95054
axie@calypto.com

Hai Zhou
EECS Department
Northwestern University
Evanston, IL 60208
haizhou@eecs.northwestern.edu

Abstract

Design closure becomes hard to achieve at physical layout stage due to the emergence of long global interconnects. Consequently, interconnect planning needs to be integrated in high level synthesis. Delay relaxation that assigns extra clock latencies to functional resources at RTL (Register Transfer Level) can be leveraged. In this paper we propose a general formulation for design closure driven delay relaxation problem. We show that the general formulation can be transformed into a convex cost integer dual network flow problem and solved in polynomial time using the convex cost-scaling algorithm in [1]. Experimental results validate the efficiency of the approach.

1 Introduction

Design closure is said to occur when constraints from high levels are satisfied at the low level. With the advent of ultra deep sub micron era, achieving design closure is becoming harder and harder. Inaccuracy of system level predictions, unpredictability in circuit behavior, critical design objectives, high degree of sensitivity among various design objectives are among a few factors to name.

Delay relaxation is a technique at RTL (Register Transfer Level) that relaxes the timing constraints of functional resources by assigning extra clock latencies (or clock cycles), called budgets, to them without violating any of the data flow or scheduling constraints. The problem of delay relaxation with maximum sum of budgets is refereed to as *maximum budget delay relaxation* problem. Polynomial algorithms can be found in [12]. In [4], a polynomial optimal algorithm was proposed using network flow technique. The same technique was used in [16] for sequential budgeting. It was shown in [15] that an RTL design with maximum budget resulted in fewer design iterations and faster design closure.

However, a design with maximum budget usually has many critical edges (edges with zero latency slack) since the budget of none of the resources can be further increased. These critical edges will have tighter timing constraints in later stages of placement and routing. On the other hand, due to aggressive technology scaling and increasing operating frequencies, the delay of a global interconnect can be longer than one clock period even with buffer insertion, which makes the timing constraints on critical edges even harder to satisfy.

In [17, 3, 13, 9, 10], wire pipelining was applied while considering placement and availability of pipeline registers. In [17, 9, 10] retiming was explored to distribute pipeline registers to fulfill communication buffering requirements on global interconnects. Although retiming helps to relieve the criticality of global interconnects, it cannot change the total number of registers along a topological cycle. Therefore, we need to integrate interconnect planning in high level synthesis to reduce the complexity of physical layout stage and leverage that burden on early stages of design flow.

Recently, researchers in [14] explored delay relaxation for interconnect budgeting at RTL. They proposed to find an RTL design with a maximal budget such that the number of non-critical edges is also maximized. Note that a maximal budgeting only requires that the budget of none of the resources can be further increased. In other words, a maximum budgeting is necessarily a maximal budgeting but the reverse is not true. They formulated the problem as a mixed integer linear programming and proposed a heuristic algorithm to solve it. Compared with an arbitrary maximal budget solution, design closure was achieved 2.8 times faster on average using the solution given by their algorithm. In addition, they found that the trade-off between resource budgeting and interconnect budgeting can be further leveraged to improve circuit performance.

Our contribution in this paper is twofold. Firstly, we propose a general formulation for design closure driven delay relaxation problem. Unlike [14], our objective is a maximum budgeting, which is desirable in many applications [4]. We also relate the problem to retiming and propose a convex retiming formulation. Secondly, we show that the general formulation can be transformed into a convex cost integer dual network flow problem and solved in polynomial time by the approach in [1]. The transformed formulation is also amicable to incorporating the trade-off between resource budgeting and interconnect budgeting. Experimental results confirm the efficiency of the approach.

2 Problem formulation

We model a data flow graph (DFG) as a directed acyclic graph (DAG) $G = (V, E)$, where V is the set of inputs and outputs of functional resources and $E = E_1 \cup E_2$ is the union of two subsets: input-to-output relations E_1 and interconnects E_2 among resources. Figure 1 illustrates an example DFG and its corresponding DAG, where dashed edges are in E_1 and solid edges are in E_2.

For all $e \in E$, we use $d(e) \in \mathcal{Z}^+$ (positive integer set) to denote the original clock latencies (or clock cycles) on that edge. For example, if $e \in E_1$, $d(e)$ could be the minimum amount of clock cycles determined by the functionality of that resource. We use $b(e) \in \mathcal{Z}^+$ to denote the budget we assign on edge $e \in E$ during delay relaxation. Each resource edge $e \in E_1$ is associated with a concave function δ_e such that $\delta_e\big(b(e)\big) \in \mathcal{Z}^+$ represents the benefit we gain

*This work was partially done at Northwestern University and supported by the NSF under CCR-0238484.

978-3-9810801-2-4/DATE07 © 2007 EDAA

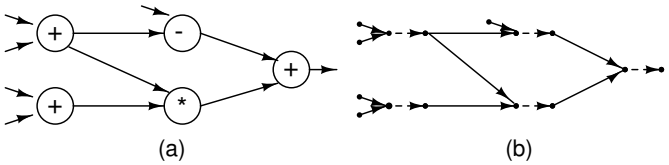

Figure 1: (a) A DFG and (b) its corresponding DAG.

by assigning $b(e)$ budget on edge e. More latencies for a resource means more possible slow-down in its logic components, which can be transformed to improvements in area, power dissipation, or other design quality metrics. Similarly, each interconnect edge $e \in E_2$ is associated with a concave function λ_e such that $\lambda_e\big(b(e)\big) \in \mathcal{Z}^+$ represents the budgeting gain on it. More latencies for an interconnect helps to further relieve its criticality. Intuitively, concave functions δ_e and λ_e help to distribute budgets evenly over all edges. Without loss of generality, we assume that all concave functions are linear between successive integers. We use $T \in \mathcal{Z}^+$ to denote the given latency upper bound at primary outputs (POs).

We introduce $t(v)$ to represent the arrival time at $v \in V$. More specifically, $t(v)$ is the maximum latency along any path from primary inputs (PIs) to v. The validity of a delay relaxation is defined by the following conditions.

$$P0(t) \triangleq \big(\forall v \in V : 0 \le t(v) \le T\big)$$

$$P1(b) \triangleq \big(\forall e \in E : 0 \le b(e) \le T - d(e)\big)$$

$$P2(t,b) \triangleq \big(\forall (u,v) \in E : t(v) \ge t(u) + d(u,v) + b(u,v)\big)$$

The condition $P0$ follows from the requirement that the arrival times at each PO after delay relaxation should be no larger than the given latency constraint T. For the same reason, the condition $P1$ assigns a budget between 0 and $T - d(e)$ to each edge $e \in E$. The condition $P2$ computes the arrival time at each vertex. Based on these conditions, we can characterize a valid delay relaxation b by $P(b)$, defined as

$$P(b) \triangleq \big(\exists t : P0(t) \wedge P1(b) \wedge P2(t,b)\big)$$

We use $\Delta(b)$ and $\Lambda(b)$ to denote the total gain on E_1 and E_2 respectively, i.e.,

$$\Delta(b) \triangleq \sum_{e \in E_1} \delta_e\big(b(e)\big), \qquad \Lambda(b) \triangleq \sum_{e \in E_2} \lambda_e\big(b(e)\big).$$

The problem we want to solve can be formulated as follows.

Problem 1
Given a directed acyclic graph $G = (V, E_1, E_2, d, \delta, \lambda, T)$, find a valid delay relaxation b with maximum $\Delta(b)$ such that $\Lambda(b)$ is also maximized.

This formulation is very general. δ_e and λ_e can be chosen independently according to the application's needs. For example, maximum budget delay relaxation is given by choosing $\delta_e\big(b(e)\big) = b(e)$, $\forall e \in E_1$. If we set

$$\lambda_e\big(b(e)\big) = \begin{cases} 0, & \text{if } b(e) = 0 \\ 1, & \text{if } b(e) > 0 \end{cases}$$

for all $e \in E_2$, we actually maximize the number of non-critical edges. Both δ_e and λ_e defined above are concave. Therefore, the problem of finding a maximum budget delay relaxation with minimal number of critical edges is a special case of Problem 1.

3 Transformation to a convex cost integer dual network flow problem

The formulation of Problem 1 seems to imply a two-step process, i.e., finding the delay relaxations b that maximize $\Delta(b)$ and then choosing among them the one that has the maximum $\Lambda(b)$. Intuitively, if these two objectives can be appropriately combined into one equivalent alternative, then we may be able to solve it efficiently in one step.

First of all, we define λ_s as follows.

$$\lambda_s \triangleq \sum_{e \in E_2} \Big(\max_{0 \le b(e) \le T - d(e)} \lambda_e\big(b(e)\big) - \min_{0 \le b(e) \le T - d(e)} \lambda_e\big(b(e)\big) \Big)$$

Note that λ_s is a constant under given d, λ and T.

Consider the following problem.

Problem 2

$$\begin{array}{ll} \text{Maximize} & (\lambda_s + 1)\Delta(b) + \Lambda(b) \\ \text{subject to} & P(b) \end{array}$$

It turns out that a solution to the above problem is the solution we want.

Theorem 1 *A solution to Problem 2 is also a solution to Problem 1.*

Proof: Let b' be a solution to Problem 1 and b^* be a solution to Problem 2. We first show that $\Delta(b^*) = \Delta(b')$. Suppose otherwise $\Delta(b^*) \neq \Delta(b')$, we have $\Delta(b^*) < \Delta(b')$ since b' maximizes $\Delta(b)$ for all valid b. On the other hand, since b^* maximizes $(\lambda_s + 1)\Delta(b) + \Lambda(b)$ and both b^* and b' are valid, we have

$$(\lambda_s + 1)\Delta(b^*) + \Lambda(b^*) \ge (\lambda_s + 1)\Delta(b') + \Lambda(b'),$$

that is, $(\lambda_s + 1)\big(\Delta(b') - \Delta(b^*)\big) \le \Lambda(b^*) - \Lambda(b')$. Given that $\Delta(b')$ and $\Delta(b^*)$ are in $\mathcal{Z}^+$, it follows that $\Delta(b') - \Delta(b^*) \ge 1$. Thus, $\Lambda(b^*) - \Lambda(b) \ge \lambda_s + 1$. However, the definition of λ_s implies that $\Lambda(b^*) - \Lambda(b) \le \lambda_s$, which is a contradiction. Therefore, $\Delta(b^*) = \Delta(b')$.

We then show that $\Lambda(b^*) = \Lambda(b')$. Otherwise $\Lambda(b^*) < \Lambda(b')$, thus $(\lambda_s + 1)\Delta(b^*) + \Lambda(b^*) < (\lambda_s + 1)\Delta(b') + \Lambda(b')$, which contradicts that b^* maximizes $(\lambda_s + 1)\Delta(b) + \Lambda(b)$. Therefore, the theorem is true. □

If we denote $k = \lambda_s + 1$, $h = 1$, and define

$$\bar{F}_e\big(b(e)\big) \triangleq \begin{cases} -k\delta\big(b(e)\big), & \text{if } e \in E_1 \\ -h\lambda\big(b(e)\big), & \text{if } e \in E_2 \end{cases}$$

then we can rewrite Problem 2 as

$$\begin{array}{ll} \text{Minimize} & \sum_{e \in E} \bar{F}_e\big(b(e)\big) \\ \text{subject to} & P(b) \end{array}$$

Note that $\bar{F}_e$ is convex on each $e \in E$. This problem is also known as *convex cost integer dual network flow problem* [1]. It is worthy to point out that the trade-off between resource budgeting $\Delta(b)$ and interconnect budgeting $\Lambda(b)$ can be handled smoothly by changing Δ and Λ.

4 Convex retiming formulation

In this section, we relate Problem 2 to retiming [8]. We propose a convex retiming formulation for Problem 2 and

show that it can be solved by relocating the latencies in the graph, which is similar to traditional retiming where registers are relocated for performance/area optimization.

First of all, note that if $t(v) > t(u) + d(u,v) + b(u,v)$ for some edge $(u,v) \in E_1$ in the solution, then $\delta_{(u,v)}\big(b(u,v)\big)$ has to be the maximum over the range $[0, T - d(u,v)]$ since $\delta_{(u,v)}$ is convex. We can modify $\delta_{(u,v)}$ by assigning $\delta_{(u,v)}(i) = \delta_{(u,v)}\big(b(u,v)\big)$ for all $b(u,v) \le i \le T - d(u,v)$. Similar modification applies to λ_e. By doing so, the modified problem has a solution with all edges being critical, from which a solution to the original problem can be easily generated. We henceforth assume that there exists a solution to Problem 2 with all edges being critical after the budget assignment.

Such a budget assignment b_0 can be obtained by the following process. First of all, we compute the arrival time at each vertex with respect to $b(e) = 0$, $\forall e \in E$, i.e., no budgeting at all. After that, we assign $b_0(u,v) = T - t(u)$ for each (u,v) with v being a PO, and assign $b_0(u,v) = t(v) - t(u)$ for other edges. We use $r : V \to \mathcal{Z}$ to represent the number of latencies that are moved from the outgoing edges to the incoming edges of each vertex. Denote

$$b_r(u,v) \triangleq b_0(u,v) + r(v) - r(u).$$

Consider the following problem.

$$\begin{array}{ll} \text{Minimize} & \sum_{e\in E} \bar{F}_e\big(b_r(u,v)\big) \\ \text{subject to} & b_r(u,v) \ge 0, \quad \forall (u,v) \in E \\ & r(v) = 0, \quad \forall v \in \{PI, PO\} \end{array}$$

We refer to it as *convex retiming* formulation. The next theorem shows that a solution to the convex retiming formulation can be used to generate a solution to Problem 2.

Theorem 2 *Let r^* be a solution to the convex retiming formulation. Then $b_{r^*}(u,v) = b_0(u,v) + r^*(v) - r^*(u)$, $\forall (u,v) \in E$, is a solution to Problem 2.*

Proof: We first show that b_{r^*} is valid, i.e., $P(b_{r^*})$. Let t^* be the arrival times with respect to b_{r^*}, thus $P2(t^*, b_{r^*})$ is true. With $r^*(v) = 0$, $\forall v \in \{PI, PO\}$, the number of latencies along any path from PI to PO cannot be changed by retiming, and thus are kept as T. Therefore, $P0(t^*)$ is true, which, together with $P2(t^*, b_{r^*})$, implies $b_{r^*}(e) \le T - d(e)$, hence $P1(b_{r^*})$ is also satisfied. What remains is to show that b_{r^*} minimizes $\sum_{e\in E} \bar{F}_e\big(b(u,v)\big)$.

Suppose otherwise that b' is the solution to Problem 2 such that $\sum_{e\in E} \bar{F}_e\big(b'(e)\big) < \sum_{e\in E} \bar{F}_e\big(b_{r^*}(e)\big)$. Then, we can obtain a valid retiming r' from b' such that $r'(v) - r'(u) \ge b'(u,v) - b_0(u,v)$, $\forall (u,v) \in E$. This is possible by longest path computation on the acyclic graph G as follows. First of all, let $r'(i) = 0$, $\forall i \in \{PI\}$. Consider any $(u,v) \in E$. Let p_1 be the longest path to v, p_2 be the longest path to u, and $p_3 = p_2 \cup \{(u,v)\}$. We use $b(p)$ to denote the latency of path p. Then $r'(v) = b'(p_1) - b_0(p_1)$ and $r'(u) = b'(p_2) - b_0(p_2)$. Since b_0 has no non-critical edges, any path to v should have the same latency in b_0. This is also true for b'. Thus, $b'(p_1) - b_0(p_1) = b'(p_3) - b_0(p_3)$. In other words, $r'(v) = r'(u) + b'(u,v) - b_0(u,v)$, $\forall (u,v) \in E$. Consider any PO j and the longest path p to it from a PI i, we have $r'(j) = r'(i) + b'(p) - b_0(p)$. Since $b'(p) = b_0(p) = T$, it follows that $r'(j) = r'(i) = 0$. Therefore, r' satisfies both constraints of convex retiming. However,

$$\sum_{e\in E} \bar{F}_e\big(b_{r'}(u,v)\big) = \sum_{e\in E} \bar{F}_e\big(b'(e)\big) < \sum_{e\in E} \bar{F}_e\big(b_{r^*}(u,v)\big),$$

which contradicts that r^* minimizes $\sum_{e\in E} \bar{F}_e\big(b_r(u,v)\big)$. It concludes the proof. □

When both δ_e and λ_e are linear, the convex retiming formulation is reduced to minimum area retiming [8], which can be solved by a minimum cost flow algorithm. For general convex functions, Ahuja *et al.* [1] showed that Problem 2 can be transformed into a convex primal network flow problem and solved in polynomial time by cost-scaling approach. We review his approach in the following.

5 Transformation to a primal network flow problem

5.1 Constraint simplification

First of all, $\bar{F}_e$ can be modified to eliminate the bounds on $b(e)$ as follows.

$$F'_e\big(b(e)\big) \triangleq \begin{cases} \bar{F}_e(0) - Mb(e), & \text{if } b(e) < 0 \\ \bar{F}_e\big(b(e)\big), & \text{if } 0 \le b(e) \le T - d(e) \\ \bar{F}_e\big(T - d(e)\big) + M\big(b(e) + d(e) - T\big), & \text{otherwise} \end{cases}$$

where M is a sufficiently large number such that F'_e is still a convex function and minimizing $\sum_{e\in E} F'_e\big(b(e)\big)$ subject to only $P0(t)$ and $P2(t,b)$ will not have a solution that violates $P1(b)$. For example, $M = \sum_{e\in E} \max_{0\le b(e)\le T-d(e)} \bar{F}_e\big(b(e)\big)$ will suffice.

Similarly, the bounds on $t(v)$ can also be eliminated by adding into the objective a convex cost function $B_v\big(t(v)\big)$ defined as follows.

$$B_v\big(t(v)\big) \triangleq \begin{cases} -Mt(v), & \text{if } t(v) < 0 \\ 0, & \text{if } 0 \le t(v) \le T \\ M\big(t(v) - T\big), & \text{if } t(v) > T \end{cases}$$

Together with the discussion in Section 4 that there exists a solution for which $P2(t,b)$ is an equality constraint, Problem 2 can be transformed to

$$\begin{array}{ll} \text{Minimize} & \sum_{e\in E} F'_e\big(b(e)\big) + \sum_{v\in V} B_v\big(t(v)\big) \\ \text{subject to} & t(u) - t(v) = -d(u,v) - b(u,v), \quad \forall (u,v) \in E \end{array}$$

To further simplify it, let

$$w(e) \triangleq -d(e) - b(e), \quad \forall e \in E$$

Define function $F_e\big(w(e)\big)$ such that $F_e\big(w(e)\big) = F'_e\big(b(e)\big)$, which is illustrated in Figure 2.

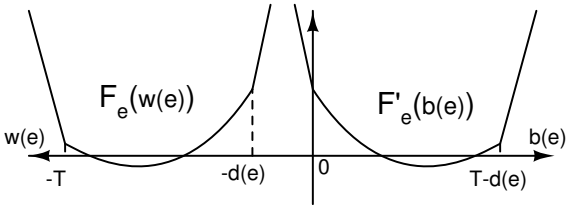

Figure 2: Illustration of $F'_e\big(b(e)\big)$ and $F_e\big(w(e)\big)$.

Note that $F_e\big(w(e)\big)$ is also convex. Substituting $F'_e\big(b(e)\big)$ by $F_e\big(w(e)\big)$, Problem 2 becomes

$$\begin{array}{ll} \text{Minimize} & \sum_{e\in E} F_e\big(w(e)\big) + \sum_{v\in V} B_v\big(t(v)\big) \\ \text{subject to} & t(u) - t(v) = w(u,v), \quad \forall (u,v) \in E \end{array}$$

5.2 Problem transformation by Lagrangian relaxation

In this section we will apply Lagrangian relaxation [11] to transform Problem 2 to a primal network flow problem. Lagrangian relaxation is a general technique for solving constrained optimization problems. In Lagrangian relaxation, "troublesome" constraints are "relaxed" and incorporated into the objective after multiplying them by constants called Lagrangian multipliers, one multiplier for each constraint. For given multipliers, the relaxed problem is called *Lagrangian subproblem*. Finding the optimal multipliers under which the Lagrangian subproblem attains the best objective value is called *Lagrangian multiplier problem*.

The Lagrangian subproblem of Problem 2 is as follows.

$$\begin{aligned} L(x) &= \min_{w,t} \Big(\sum_{e\in E} F_e\big(w(e)\big) + \sum_{v\in V} B_v\big(t(v)\big) \\ &\quad - \sum_{(u,v)\in E} x(u,v)\big(w(u,v) + t(v) - t(u)\big)\Big), \end{aligned}$$

where $x(u,v)$ is the Lagrangian multiplier associated with the constraint $t(u) - t(v) = w(u,v)$. Note that

$$\begin{aligned} &\sum_{(u,v)\in E} x(u,v)\big(t(v) - t(u)\big) \\ &= \sum_{v\in V} t(v)\big(\sum_{(i,v)\in E} x(i,v) - \sum_{(v,j)\in E} x(v,j)\big) \end{aligned}$$

Define

$$x(v,0) \stackrel{\triangle}{=} \sum_{(i,v)\in E} x(i,v) - \sum_{(v,j)\in E} x(v,j), \quad \forall v \in V$$

Substituting it into $L(x)$ yields

$$\begin{aligned} L(x) &= \min_{w,t} \Big(\sum_{e\in E} \big(F_e\big(w(e)\big) - x(e)w(e)\big) \\ &\quad + \sum_{v\in V} \big(B_v\big(t(v)\big) - x(v,0)t(v)\big)\Big) \end{aligned}$$

To simplify $L(x)$, a vertex 0 is introduced into G as well as edges $(v,0)$, $\forall v \in V$. Let $G^0 = (V^0, E^0)$ be the resultant DAG. We define

$$w(v,0) \stackrel{\triangle}{=} t(v), \qquad F_{(v,0)}\big(w(v,0)\big) \stackrel{\triangle}{=} B_v\big(t(v)\big)$$

Then the Lagrangian subproblem becomes

$$\begin{aligned} L(x) &= \min_{w} \sum_{e\in E^0} \big(F_e\big(w(e)\big) - x(e)w(e)\big) \\ \text{subject to} &\quad \sum_{(i,v)\in E^0} x(i,v) = \sum_{(v,j)\in E^0} x(v,j), \quad \forall v \in V^0 \end{aligned}$$

It is important to notice that, for a given x, each term of $\sum_{e\in E^0} F_e\big(w(e)\big) - x(e)w(e)$ is a function of $w(e)$. Since $w(e)$ are independent among all edges $e \in E^0$, the objective is minimized only when each term of it is minimized. In other words, we can minimize each term separately. Define

$$H_e\big(x(e)\big) \stackrel{\triangle}{=} \min_{w} \Big(F_e\big(w(e)\big) - x(e)w(e)\Big), \quad \forall e \in E^0$$

Then it becomes $L(x) = \sum_{e\in E^0} H_e\big(x(e)\big)$. The *Lagrangian multiplier problem* of Problem 2 can be formulated as follows.

$$\begin{aligned} L(x^*) &= \max_x L(x) = \max_x \sum_{e\in E^0} H_e\big(x(e)\big) \\ \text{subject to} &\quad \sum_{(i,v)\in E^0} x(i,v) = \sum_{(v,j)\in E^0} x(v,j), \quad \forall v \in V^0 \end{aligned}$$

This is a primal network flow problem. The constraint is also known as *flow conservation*, i.e., incoming flow equals outgoing flow. x is called a flow on G^0 if it satisfies flow conservation. For example, $x(e) = 0$ for all $e \in E^0$ is a flow. $H_e\big(x(e)\big)$ is the gain function on $e \in E^0$ with respect to the flow $x(e)$. The Lagrangian multiplier problem asks for an optimal flow such that the total gain over G^0 is maximized. The following theorem [2] establishes a connection between Problem 2 and its corresponding Lagrangian multiplier problem.

Theorem 3 *Let x^* be a solution to the Lagrangian multiplier problem of Problem 2. Then $L(x^*)$ equals the optimal objective value of Problem 2.*

6 Convex cost-scaling approach

In the following, we will characterize function H_e and elaborate the algorithm developed in [1] for computing x^*. In Section 6.3 we describe how to use x^* to construct a solution to Problem 2.

6.1 Function $H_e\big(x(e)\big)$

Consider the case when $x(e) > M$ for some $e \in E^0$, where M is the large number we introduced in Section 5.1 to eliminate the bounds on $b(e)$. By the definition of F_e, we have

$$\begin{aligned} &\lim_{w(e)\to\infty} \Big(F_e\big(w(e)\big) - x(e)w(e)\Big) \\ = &\lim_{w(e)\to\infty} \big(Mw(e) - x(e)w(e)\big) = -\infty \end{aligned}$$

Thus $H_e\big(x(e)\big) = -\infty$ when $x(e) > M$. Recall that the Lagrangian multiplier problem asks for an x^* that maximizes $\sum_{e\in E^0} H_e\big(x(e)\big)$. It implies that we can assume $x^*(e) \le M$. Similarly, it can be shown that $H_e\big(x(e)\big) = -\infty$ when $x(e) < -M$, which implies that $x^*(e) \ge -M$. Therefore we consider $-M \le x(e) \le M$, $\forall e \in E^0$, in the Lagrangian multiplier problem.

Let $\perp(e) = -T$ and $\top(e) = -d(e)$ for $e \in E$, and $\perp(e) = 0$ and $\top(e) = T$ for $e \in E^0 - E$. For each $e \in E_0$, we define

$$f_e(\theta) \stackrel{\triangle}{=} F_e(\theta+1) - F_e(\theta), \quad \perp(e) \le \theta \le \top(e) - 1 \qquad (1)$$

Then $H_e\big(x(e)\big)$ for $-M \le x(e) \le M$ can be analytically expressed by a set of linear segments specified in the next lemma [1].

Lemma 1 *$H_e\big(x(e)\big)$ for $-M \le x(e) \le M$ is a piecewise linear concave function of $x(e)$ as follows, where $\perp(e) < i < \top(e)$,*

$$H_e\big(x(e)\big) = \begin{cases} F_e\big(\perp(e)\big) - \perp(e)x(e), & \text{if } -M \le x(e) \le f_e\big(\perp(e)\big) \\ \cdots\cdots & \\ F_e(i) - ix(e), & \text{if } f_e(i-1) \le x(e) \le f_e(i) \\ \cdots\cdots & \\ F_e\big(\top(e)\big) - \top(e)x(e), & \text{if } f_e\big(\top(e)-1\big) \le x(e) \le M \end{cases}$$

Figure 3 gives an illustration of $H_e\big(x(e)\big)$ for $F_e\big(w(e)\big) = -1/w(e)$, $\perp(e) = -5$ and $\top(e) = -1$.

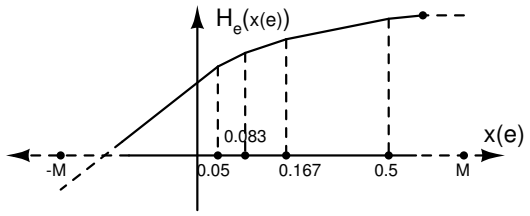

Figure 3: Illustration of $H_e\big(x(e)\big)$.

6.2 Algorithm

Define cost function $C_e\big(x(e)\big) = -H_e\big(x(e)\big)$. The Lagrangian multiplier problem can be alternatively restated as

$$\text{Minimize} \quad \sum_{e\in E^0} C_e\big(x(e)\big)$$

$$\text{subject to} \quad \sum_{(i,v)\in E^0} x(i,v) = \sum_{(v,j)\in E^0} x(v,j), \quad \forall v \in V^0$$

$$-M \le x(e) \le M, \quad \forall e \in E^0$$

Notice that if C_e is linear, i.e., $C_e\big(x(e)\big) = \rho x(e)$ where ρ is a constant, then the above problem is a minimum cost flow problem, which can be solved in polynomial time [2]. For general convex C_e, since it is piecewise linear, a straightforward approach is to represent C_e as a set of segments, each of which is linear within a small range, and solve it using any minimum cost flow algorithm. However, since the number of segments for each e is $\top(e) - \perp(e) + 1$, minimum cost flow algorithms would not in general run in polynomial time. Intuitively, if the minimum cost flow algorithm can be modified to be aware of the change of cost when the flow is changed, then we can apply it directly to C_e, which may help to solve the problem more efficiently.

In [1], a polynomial time algorithm was developed for general convex functions based on cost-scaling algorithm [6].

The cost-scaling algorithm defines *pseudoflow* x such that x satisfies $-M \le x(e) \le M$ for all $e \in E$ but may violate the flow conservation. For any pseudoflow x, the *imbalance* of vertex v is defined as $o(v) = \sum_{(i,v)\in E^0} x(i,v) - \sum_{(v,j)\in E^0} x(v,j)$. If $o(v) > 0$, v is called an *excess* vertex. The algorithm proceeds by constructing and manipulating the *residual graph* $\mathcal{G}(x)$ defined as follows with respect to a pseudoflow x. For each $(u,v) \in E^0$, $\mathcal{G}(x)$ may contain two edges: *forward* edge (u,v) and *backward* edge (v,u). Forward edge (u,v) has a cost $c(u,v)$ equal to the right slope of $C_{(u,v)}\big(x(u,v)\big)$ at $x(u,v)$, and a capacity of $cap(u,v) = M - x(u,v)$. Backward edge (v,u) has a cost $c(v,u)$ equal to the negative of the left slope of $C_{(u,v)}\big(x(u,v)\big)$ at $x(u,v)$, and a capacity of $cap(v,u) = x(u,v) - (-M)$. Note that if $C_{(u,v)}\big(x(u,v)\big) = \rho x(u,v)$, then $c(u,v) = -c(v,u) = \rho$. The residual graph consists only of edges with positive capacities. The algorithm also maintains a value $\pi(v)$ for each $v \in V^0$, which is refereed to as the *vertex potential*.

For a given residual graph $\mathcal{G}(x)$ and a set of vertex potentials, it defines the *reduced cost* of edge (u,v) as $c^\pi(u,v) = c(u,v) + \pi(v) - \pi(u)$. For a given value of ϵ, an edge (u,v) is called *admissible* if $-\epsilon \le c^\pi(u,v) < 0$. A flow or pseudoflow x is called *ϵ-optimal* for some $\epsilon \ge 0$ if for some vertex potential π, the following *ϵ-optimality condition* is satisfied: $c^\pi(u,v) \ge -\epsilon$ for all $(u,v) \in \mathcal{G}(x)$. The cost-scaling algorithm treats ϵ as a parameter and iteratively obtains *ϵ-optimal* flows for successively smaller values of ϵ. Initially, ϵ is set to be the maximum edge cost (which is T in our problem), thus any flow is *ϵ-optimal*. The algorithm then performs cost-scaling phases by repeatedly applying an *improve-approximation* procedure that transforms an *ϵ-optimal* flow into an *$\epsilon/2$-optimal* flow. When $\epsilon < 1/|V^0|$, the algorithm terminates with an optimal flow x^*. The pseudocode of the algorithm is presented in Figure 4.

```
Algorithm convex cost-scaling
Input: A circuit G = (V^0, E^0) and C_e, ∀e ∈ E^0
Output: An optimal flow x*.

   π ← 0 and ε ← max_{e∈E^0} ⊤(e);
   Let x* be any flow;
   While (ε ≥ 1/|V^0|) do
     improve-approximation(ε, x*, π);
     ε ← ε/2;
   Return x*;

Procedure improve-approximation(ε, x, π)
   For each admissible edge (u,v) ∈ E^0 do
     Send q(u,v) flow on (u,v);
   Compute flow imbalance on each vertex;
   While there is an excess vertex u do
     If there is an admissible edge (u,v) then
       push min(o(u), q(u,v)) flow on (u,v);
     Else
       π(u) ← π(u) + ε/2;
```

Figure 4: Pseudocode of convex cost-scaling algorithm.

The basic operation in the *improve-approximation* procedure is to select an excess vertex u with $o(u) > 0$. When u has no admissible edges, its potential is increased by $\epsilon/2$; otherwise, it performs *pushes* on admissible edges emanating from it. If $C_e\big(x(e)\big)$ is linear, the amount of flow pushed on an admissible edge (u,v) will be the minimum of the excess and the edge capacity, i.e., $\min\big(o(u), cap(u,v)\big)$. However, when $C_e\big(x(e)\big)$ is piecewise linear, the cost (slope) will change with the change of the flow. For this case, [1] showed that the amount of flow should be $\min\big(o(u), q(u,v)\big)$, where $q(u,v)$ is defined as follows. If (u,v) is a forward edge, then

$$q(u,v) = \begin{cases} M - x(u,v), & \text{if } \pi(u) - \pi(v) \ge \top(u,v) \\ f_{(u,v)}(\lfloor \pi(u) - \pi(v) \rfloor) - x(u,v), & \text{otherwise} \end{cases}$$

where $f_{(u,v)}$ is defined in (1) in Section 6.1. If (u,v) is a backward edge, then

$$q(u,v) = \begin{cases} M + x(v,u), & \text{if } \pi(v) - \pi(u) \le \perp(v,u) \\ x(v,u) - f_{(v,u)}(\lfloor \pi(v) - \pi(u) \rfloor), & \text{otherwise} \end{cases}$$

Intuitively, we push the flow along the current linear segment of the piecewise linear function until we hit a joint point where the cost is about to change. The change of flow is followed by an update in edge cost (slope) based on Lemma 1. The procedure terminates when there is no more excess vertex. Note that the replacement of $cap(u,v)$ by $q(u,v)$ and the following cost update is the only difference between a convex cost-scaling algorithm and a linear cost-scaling algorithm.

The correctness and complexity of the algorithm is given in the next theorem [1].

Theorem 4 *The convex cost-scaling algorithm in Figure 4 solves the Lagrangian multiplier problem of Problem 2 in* $O(|V||E|\log(|V|^2/|E|)\log(|V|T))$ *time, where the complexity of the improve-approximation procedure is bounded by* $O(|V||E|\log(|V|^2/|E|))$ *and* $O(\log(|V|T))$ *bounds the number of iterations.*

The practical efficiency of the algorithm is confirmed by our experimental results in Section 7.

6.3 Solution transformation

The convex cost-scaling algorithm upon termination gives an optimal flow x^* and the corresponding potentials π. Both x^* and π may not be integer. However, since each cost function is piecewise linear, it follows that there always exists an integer optimal potential π^*. To determine it, we construct the residual graph $\mathcal{G}(x^*)$ with respect to x^* and solve a shortest path problem to determine the shortest path distance $d(v)$ from vertex 0 to every other vertex $v \in V$ in terms of costs. Since all edge costs in $\mathcal{G}(x^*)$ are integer, d is also integer. Then $\pi^*(v) = -d(v)$, $\forall v \in V$, gives an integer potential for the Lagrangian multiplier problem. The solution (t^*, w^*, b^*) to Problem 2 is obtained by assigning $t^* = \pi^*$, $w^*(u,v) = t^*(u) - t^*(v)$, and $b^*(u,v) = -d(u,v) - w^*(u,v)$, $\forall (u,v) \in E$.

7 Experimental results

We used the linear cost-scaling algorithm by Goldberg in [5] and adapted it to convex cost case. We set δ_e and λ_e as in Section 2 so that we focus on the problem of finding a maximum budget delay relaxation with minimal number of critical edges. We used the test files extracted from MediaBench [7]. In addition, since the test files are relatively small, we included ISCAS-85 benchmark suite. Note, however, that we treated each gate as a resource. This is reflected by adding $_{DFG}$ as a postfix to the test names in Table 1. Without loss of generality, the original clock latency was set to 1 for each resource and 0 for each interconnect. The latency constraint T was set to the maximum number of resources in a PI-to-PO data flow path. All tests were conducted in a PC with a 2.4 GHz Xeon CPU, 512 KB 2nd level cache memory and 1GB RAM. The results are reported in Table 1, where column "$|E_2^*|$" lists the number of non-critical interconnects in the obtained optimal budgeting solution, column "$\sum b^*$" lists the sum of budgets, which is the maximum for the test, and column "t(sec)" lists the running time in seconds.

Table 1: Experimental Results

name	$\|V\|$	$\|E_1\|$	$\|E_2\|$	T	$\|E_2^*\|$	$\sum b^*$	t(sec)
inv1	400	200	197	9	2	2	0.01
inv2	430	215	211	16	17	32	0.01
inv3	702	351	355	15	14	14	0.02
jpg1	234	117	127	16	9	15	0.01
jpg2	530	265	262	13	5	5	0.01
mat1	218	109	112	10	0	0	0.00
mat2	192	96	104	14	0	0	0.00
rot1	160	80	73	6	3	3	0.00
$c1908_{DFG}$	1760	880	1420	39	236	1414	0.12
$c2670_{DFG}$	2386	1193	1850	31	334	1381	0.19
$c3540_{DFG}$	3338	1669	2633	46	404	1298	0.53
$c5315_{DFG}$	4614	2307	3878	48	712	8109	0.91
$c6288_{DFG}$	4832	2416	4288	123	1182	1702	1.57
$c7552_{DFG}$	7024	3512	5836	42	1187	6674	1.57

It can be seen that the algorithm is very efficient. All MediaBench test can be finished in 0.02 second. The running times appear to scale well with problem size in ISCAS-85.

Once an optimal budgeting solution is found, the corresponding data flow graph can be implemented, for example, using the synthesis flow proposed in [14], which includes the usage of Synplicity Synplify Pro 7.7.1 followed by Xilinx 6.3 Place And Route tool. They are omitted here to avoid repetition.

8 Conclusion

A general problem formulation is proposed for design closure driven delay relaxation. We show that it can be transformed to a convex cost integer dual network flow problem. It can also be viewed as a convex retiming problem. When cost functions are linear, it is reduced to minimum area retiming. The proposed formulation is amicable to incorporating the trade-off between resource budgeting and interconnect budgeting. Using Lagrangian relaxation technique, the dual network flow problem is further transformed to a primal network flow problem, which can be solved in polynomial time by the convex cost-scaling algorithm in [1].

References

[1] R. K. Ahuja, D. S. Hochbaum, and J. B. Orlin. Solving the convex cost integer dual network flow problem. *Management Science*, 49(7):950–964, July 2003.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Application*. Prentice Hall, 1993.

[3] J. Cong, Y. Fan, G. Han, X. Yang, and Z. Zhang. Architectural synthesis integrated with global placement for multicycle communications. In *ICCAD*, pages 536–543, 2003.

[4] S. Ghiasi, E. Bozorgzadeh, S. Choudhary, and M. Sarrafzadeh. Unified theory of timing budget management. In *ICCAD*, pages 653–659, 2004.

[5] A. V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, 22:1–29, 1997.

[6] A. V. Goldberg and R. E. Tarjan. Solving minimum cost flow problem by successive approximation. In *ACM Sympos. Theory Comput.*, pages 7–18, 1987.

[7] C. Lee, M. Potkonjak, and W. Mangione-Smith. Mediabench: A tool for evaluating and synthesizing multimedia and communications systems. In *International Symposium on Microarchitecture*, pages 330–335, 1997.

[8] C. E. Leiserson, F. M. Rose, and J. B. Saxe. Optimizing synchronous circuitry by retiming. In *Advanced Research in VLSI: Proc. of the Third Caltech Conf.*, pages 86–116, Rockville, MD, 1983. Computer Science Press.

[9] C. Lin and H. Zhou. Wire retiming as fixpoint computation. *IEEE TVLSI*, 13(12):1340–1348, December 2005.

[10] C. Lin and H. Zhou. Optimal wire retiming without binary search. *IEEE TCAD*, 25(9):1577–1588, September 2006.

[11] D. G. Luenberger. *Linear and nonlinear programming*. Addison-Wesley, Reading, Massachusetts, 1984.

[12] R. Nair, C. L. Berman, P. S. Hauge, and E. J. Yoffa. Generation of Performance Constraints for Layout. *IEEE TCAD*, 8(8):860–874, August 1989.

[13] V. Nookala and S. S. Sapatnekar. A method for correcting the functionality of a wire-pipelined circuit. In *DAC*, pages 570–575, 2004.

[14] L. Singhal and E. Bozorgzadeh. Fast timing closure by interconnect criticality driven delay relaxation. In *ICCAD*, pages 792–797, 2005.

[15] A. Srivastava, S. O. Memik, B.-K. Choi, and M. Sarrafzadeh. Achieving design closure through delay relaxation parameter. In *ICCAD*, pages 54–57, 2003.

[16] C.-Y. Yeh and M. Marek-Sadowska. Minimum-area sequential budgeting for FPGA. In *ICCAD*, pages 813–817, 2003.

[17] H. Zhou and C. Lin. Retiming for wire pipelining in system-on-chip. *IEEE TCAD*, 23(9):1338–1345, September 2004.