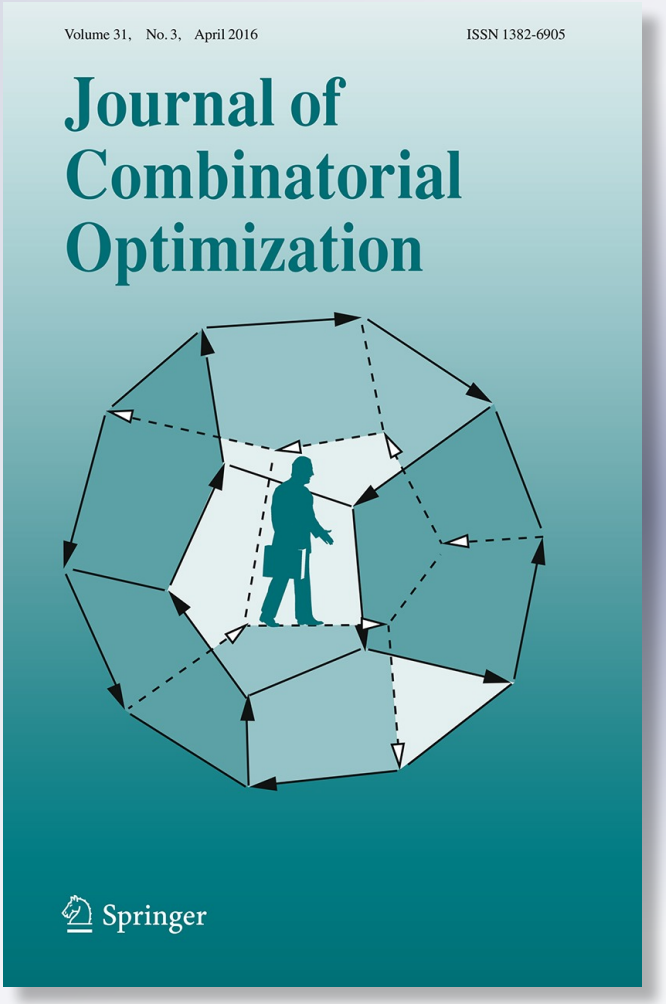
Optimal weight allocation in rooted trees

Shmuel Wimer

Journal of Combinatorial Optimization

ISSN 1382-6905
Volume 31
Number 3

J Comb Optim (2016) 31:1023-1033
DOI 10.1007/s10878-014-9807-0



Springer

J Comb Optim (2016) 31:1023–1033
DOI 10.1007/s10878-014-9807-0

Optimal weight allocation in rooted trees

Shmuel Wimer

Published online: 21 October 2014

Abstract Some apparently different VLSI circuit design optimization problems can be mapped to the problem of allocating weights (hardware circuits) to the nodes of a tree, such that their total sum (delay) along root-to-leaf paths, or their total product (amplification) along root-to-leaf paths, satisfy given demands (delays or amplifications, respectively) at the tree's leaves. Node's weight is shared by all the leaves of its emanating sub-tree. For both the sum and product constraints cases, $O(n)$ weights allocation algorithms are presented, supplying the demands at the leaves, while the total sum of nodes' weights (hardware cost) is minimized. When the assignment of the demands to leaves is not predetermined, it is shown that monotonic order of the demands at leaves is optimal for both cases.

Keywords Weight allocation · Binary tree optimization · Clock-tree · Amplifier-tree

1 Motivations

A tight control of the timing of the clock signal is crucial in today's very large scale integration (VLSI) systems. *Clock skew* control (Fishburn 1990), fixing delay problems by *time borrowing* (Tam et al. 2000), power-supply noise reduction (Jiang and Cheng 1999) by spreading the clock arrival time to various elements of the system (Benini et al. 1997), are a few examples. The adjustment of the clock signal timing so that the chip can operate in higher speed, a technique called *binning* (Churcher and Longstaff

S. Wimer (✉)
Engineering Faculty, Bar-Ilan University, Ramat Gan, Israel
e-mail: wimers@biu.ac.il

S. Wimer
EE Department, Technion-Israel Institute of Technology, Haifa, Israel

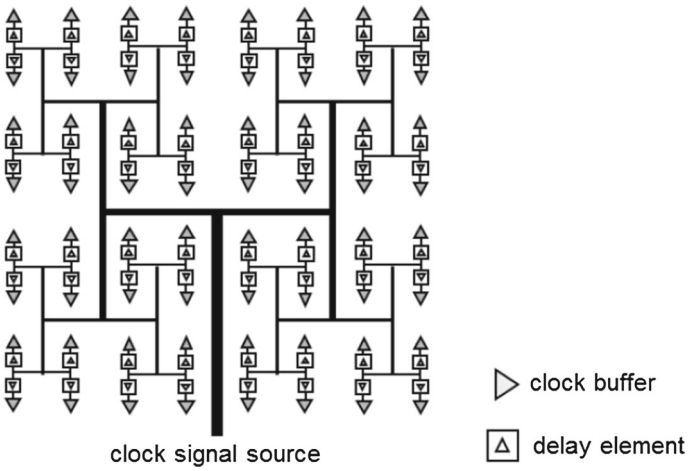


Fig. 1 H-tree clock distribution network

1998), is used to maximize the profit, and *securing VLSI systems against differential power analysis* (Kocher et al. 1999), are more examples.

A common clock network implementation is the well-known *H-tree* shown in Fig. 1 (Friedman 2001). H-tree is a balanced binary tree, symmetrically embedded in the 2D plane. The clock buffer at a leaf is driving some underlying module (circuit or block). The delays required at its leaves are adjusted by appropriate delay elements. A recent work in Kim et al. (2013) presented an $O(n \log n)$ algorithm to distribute the delay elements in the internal nodes of the clock-tree such that the total cost of the delay elements is minimized.

A different VLSI design situation where circuits are allocated to tree's nodes occurs in *Analog to Digital Converter* (ADC) circuits (Baker 2011, Chap. 29). There, the input, time-varying, analog signal is compared to various reference voltages to produce a binary code of the input voltage. In Jiren and Piper (1999) an ADC implementation comprising a delay-balanced input amplifier network was presented. The network was implemented as a binary tree, producing amplifications by $2^{(i-1)k}$ of the input analog signal at the tree's leaves, $1 \leq i \leq n$, $k \in \{1, 2\}$. Such implementation ensured the required amplifications, while maintaining delay equality at all leaves. The cost of an amplifier at a node is proportional to its amplification factor, where unit amplification is the minimum possible.

The above VLSI design problems can be mapped to the problem of allocating weights (hardware circuits) to tree's nodes. Total weight sum (delay) along root-to-leaf paths, or total weight product (amplification) along root-to-leaf paths, have to satisfy given demands (delays or amplifications, respectively) at the tree's leaves. Node's weight is shared by all the leaves of its emanating sub-tree. For both the sum and product constraints cases, $O(n)$ weights allocation algorithms are presented (Sects. 2, 4, respectively), supplying the demands at the leaves, while the total sum of nodes' weights (hardware cost) is minimized. When the assignment of the demands to leaves

is not predetermined, it is shown that monotonic order of the demands at leaves is optimal for both cases (Sects. 3, 5, respectively).

2 Optimal weight allocation to nodes subject to sum constraints

Balanced binary tree is discussed first. As shown later, all the results hold for any rooted tree. Let T be a balanced binary tree with $n = 2^N$ leaves μ_i, associated with demands $\delta_i \geq 0,\ 1 \leq i \leq n$. Denote by $P^+(v, \rho)$ the path from the root $\rho \in T$ to a node $v \in T$, where the + superscript stands for a weight addition operation defined on the path. To satisfy the demands, weights $w(v) \geq 0,\ v \in T$, are allocated such that

$$\sum\nolimits_{v \in P^+(\mu_i, \rho)} w(v) = \delta_i,\ 1 \leq i \leq n. \tag{1}$$

Let $\mathbf{W}^+$ be the set of weight allocations satisfying (1). The allocation w^0, defined by $w^0(\mu_i) = \delta_i,\ 1 \leq i \leq n$ and $w^0(v) = 0, v \in T \setminus \{\mu_i\}_{i=1}^n$ trivially satisfies (1), hence $w^0 \in \mathbf{W}^+$. Consider the total sum of weights $\Delta(w)$ allocated to T's nodes,

$$\Delta(w) = \sum\nolimits_{v \in T} w(v). \tag{2}$$

Denote by Δ^* the minimum total weights across all $w \in \mathbf{W}^+$,

$$\Delta^* = \min_{w \in \mathbf{W}^+} \{\Delta(w)\}. \tag{3}$$

An allocation algorithm $w^+ \in \mathbf{W}^+$, satisfying $\Delta(w^+) = \Delta^*$ is subsequently presented. It determines the weights of T's nodes in a bottom-up traversal. Let μ' and μ'' be two sibling leaves of T, sharing a common parent v. Let their demands be δ' and δ'', respectively, and $\delta'' \geq \delta'$. The algorithm fixes $w^+(\mu') = 0$ and $w^+(\mu'') = \delta'' - \delta'$, and *temporarily* allocates the weight δ' to v. This way the two sons share the weight allocated to their parent, while the excess is permanently allocated to the more demanding leaf. w^+ thus yields δ' *weight saving*.

w^+proceeds bottom-up level-by-level up to T's root ρ, which is necessarily allocated with the smallest demand among all the leaves. For two sibling nodes v' and v'' with a common parent, w^+ fixes at least one of the weights $w^+(v')$ and $w^+(v'')$ to 0. It turns out that among the $2n - 1$ T's nodes, the weight of at least $n - 1$ nodes is fixed by w^+ to 0.

Figure 2 illustrates w^+ for a tree with demands of 1 to 8 units, specified below their corresponding leaves. Temporary weights are shown in blue and fixed weights in red. The total sum of fixed weights along a root-to-leaf path satisfies the demand at the leaf.

While the total sum of demands specified at leaves is 36, the total sum of weights distributed at nodes has been reduced to 22 units, thus yielding savings of 14 units. The following Lemma shows that $\Delta(w^+) = \Delta^*$.

Lemma 1 *Let T be a balanced binary tree with $n = 2^N$ leaves μ_i, associated with non-negative demands $\delta_i \geq 0,\ 1 \leq i \leq n$. There is $\Delta(w^+) = \Delta^*$.*

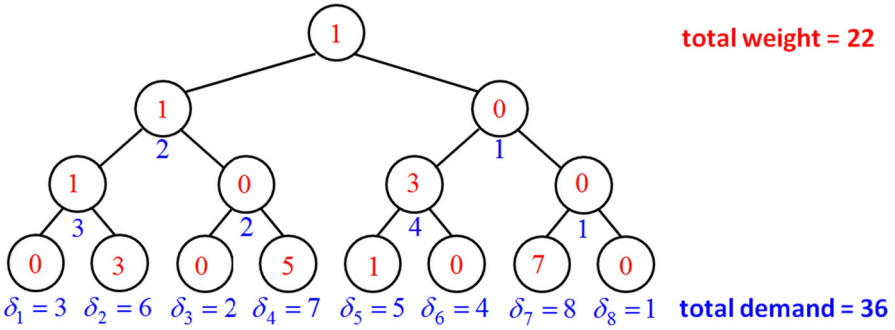


Fig. 2 Weight allocation by w^+

Proof Assume in contrary that $\Delta(w^+) = \sum_{v\in T} w^+(v) > \Delta^*$, so there is another $w^{**} \in \mathbf{W}$ such that $\sum_{v\in T} w^{**}(v) = \Delta^*$. We first claim that w^{**} must also have the property that for any sibling sons μ' and μ'' sharing a common parent, at least one of $w^{**}(\mu') = 0$ and $w^{**}(\mu'') = 0$ must hold. Otherwise, $\sum_{v\in T} w^{**}(v)$ could further be reduced by adding $\varepsilon = \min\{w^{**}(\mu'), w^{**}(\mu'')\}$ to their parent and subtracting ε from both, yielding $\sum_{v\in T} w^{**}(v) < \Delta^*$ which contradicts (3).

If $\sum_{v\in T} w^+(v) > \sum_{v\in T} w^{**}(v) = \Delta^*$ was true, there would exist at least one node $\mu' \in T$ such that $w^+(\mu') > w^{**}(\mu') \geq 0$. Let μ'' be the sibling of μ', sharing a common parent μ. By the above claim there is $w^{**}(\mu'') = w^+(\mu'') = 0$. If μ' was a leaf, let δ' and δ'' be the demands associated with μ' and μ'', respectively. By (1) there is

$$\sum\nolimits_{v\in P^+(\mu'',\rho)} w^+(v) = \sum\nolimits_{v\in P^+(\mu'',\rho)} w^{**}(v) = \delta''.$$

It follows from $w^{**}(\mu'') = w^+(\mu'') = 0$ that

$$\sum\nolimits_{v\in P^+(\mu,\rho)} w^+(v) = \sum\nolimits_{v\in P^+(\mu,\rho)} w^{**}(v) = \delta''.$$

But then

$$\delta' = \sum\nolimits_{v\in P^+(\mu,\rho)} w^+(v) + w^+(\mu') > \sum\nolimits_{v\in P^+(\mu,\rho)} w^{**}(v) + w^{**}(\mu') = \delta',$$

which is impossible. Hence $w^+(\mu') > w^{**}(\mu')$could not occur at a leaf.

If μ' is an internal node, let l, $N > l > 0$, be T's lowest level where $w^+(\mu') > w^{**}(\mu')$ occurs (T's leaves are at level $N = \log_2 n$ and T's root is at level 0). In that case we could ignore all the levels from $l + 1$ down to N, were w^+ and w^{**} are identical, and be left with a reduced balanced binary tree having 2^l leaves. Since the inequality in the reduced tree occurs at a leaf, same contradiction follows as before. □

It is straightforward to generalize w^+ for any rooted tree. If T is not balanced, it can be modified to make all its leaves residing at the same level as follows. A higher

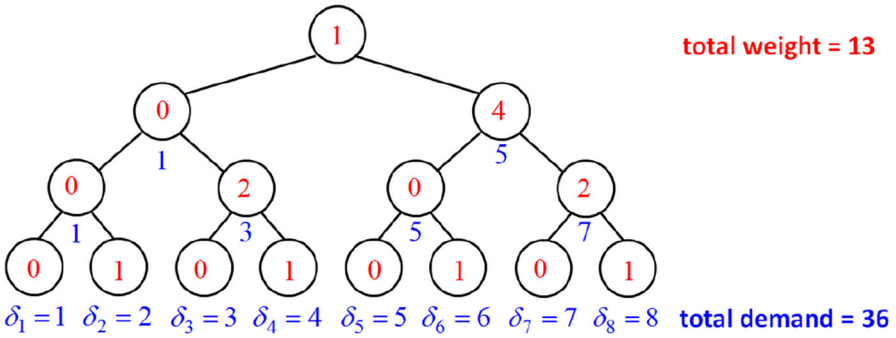


Fig. 3 Better assignment of demands to leaves, yielding smaller w^+

level leaf is augmented with a chain having appropriate number of edges, and then put the demand at the new leaf located at the opposite end of the augmenting path. Once T is balanced, let a node μ of a tree have d sons $\mu_1, \ldots, \mu_d$, whose temporary weights are $\delta_1, \ldots, \delta_d$, respectively. w^+ then floats $\delta = \min_{1 \le i \le d} \{\delta_i\}$ to μ and fixes μ_i's weight to $\delta_i - \delta$. The weight thus being saved at $\mu_1, \ldots, \mu_d$ is $\delta(d-1)$. Lemma 1 which states the optimality of w^+ for binary trees works similarly for any balanced tree.

3 Optimizing the assignment of demands subject to weight sum constraints

Lemma 1 showed that the weight allocation w^+ yields Δ^* defined in (3). If the mapping of demands to leaves is not predetermined, Δ^* could further be reduced by applying $w^+$for each of the $n!$ possible mappings. An example is shown in Fig. 3, where the demands have been differently mapped to leaves than in Fig. 2. w^+ allocated weights to nodes, yielding 13 total sum of weights, compared to 22 in Fig. 2.

Let $0 \le \delta_1 \le \delta_2 \le \cdots \le \delta_n$ be n demands, and μ_i, $1 \le i \le n$, be the leaves of a balanced binary tree T. We assume w.l.o.g that $\delta_1 < \delta_2 < \cdots < \delta_n$ since equalities can arbitrarily be resolved. Let $\mathbf{\Pi}$ be the set of $n!$ permutations. An assignment of demands to T's leaves is a permutation $\pi \in \mathbf{\Pi}$. The notation $\Delta(w, \pi)$ is used to denote the dependence of the total sum of weights on both the assignment of the demands to leaves, and the weight allocation $w \in \mathbf{W}$ to supply those. We therefore look for $\pi^* \in \mathbf{\Pi}$ satisfying

$$\Delta(w^+, \pi^*) = \min_{\pi \in \mathbf{\Pi}} \left\{ \Delta(w^+, \pi) \right\}. \tag{4}$$

Notice that the total sum of weights obtained by w^+ is invariant of a swap of the left and the right sub-trees rooted at a node. Since T has $n-1$ internal nodes, it turns out that such swaps induce 2^{n-1} isomorphic assignments, reducing the solution space to size $n!/2^{n-1}$. It is subsequently shown that the identity permutation $\pi^{\mathrm{id}}(i) = i$ satisfies (4).

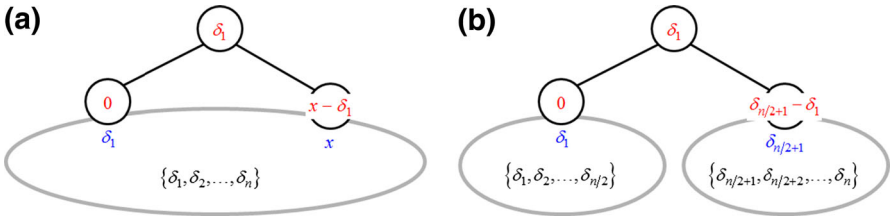


Fig. 4 First induction step

Lemma 2 *Let $n = 2^N$, and $0 \le \delta_1 < \delta_2 < \cdots < \delta_n$ be demands assigned to the leaves of a balanced binary tree T. The identity permutation $\pi^{id}(i) = i,\ 1 \le i \le n$, satisfies*

$$\Delta(w^+, \pi^{id}) = \min_{\pi \in \mathbf{\Pi}} \left\{ \Delta(w^+, \pi) \right\}. \tag{5}$$

Proof Let μ_i, $1 \le i \le n$, be the left-to-right ordered leaves of T and $\delta_{\pi^{-1}(i)}$ be their corresponding weights assigned by $\pi \in \mathbf{\Pi}$. To prove (5), it will be shown that $\Delta(w^+, \pi^{\mathrm{id}})$ maximizes the weight savings given by $\sum_{i=1}^{n} \delta_i - \Delta(w^+, \pi^{\mathrm{id}})$. Recall that by w^+ definition, the floatation of the smallest weight among two sibling sons to their parent results in weight saving by that amount.

We subsequently show by induction on the levels of T, that the maximization of the total weight saving must separate the demands assigned to left and right sub-trees rooted at a node, the smaller weights to one sub-tree and the larger ones to the other.

By w^+ definition, the total saving is the sum of the temporary weights floating to nodes across all tree's levels, from level $N-1$ (parents of leaves), up to the root at level 0. For the first step of the induction the saving obtained at level 1 is considered, as shown in Fig. 4, where temporary weights are colored in blue and fixed weights are colored in red. It is shown that to maximize the weight saving incurred at level 1, the separation of the demands must be such that $\{\delta_i\}_{i=1}^{n/2}$ and $\{\delta_i\}_{i=n/2+1}^{n}$ are assigned to the leaves of different sub-trees.

By w^+ definition, the smallest demand δ_1 must float all the way up from a leaf to the root. It must therefore be temporarily allocated to one of the two sons at level 1. Let it be w.l.o.g the left son, as shown in Fig. 4a.

It is subsequently shown that $\delta_{n/2+1}$ is the largest demand that can float from the leaves and temporarily be allocated to the right son as shown in Fig. 4b. Assume in contrary that w^+ floated other demand $\delta_p > \delta_{n/2+1}$ to the root of the right sub-tree. If the demand $\delta_{n/2+1}$ was also assigned to a leaf of the right sub-tree, that would contradict the property of w^+, floating the smallest value up to the root. If however $\delta_{n/2+1}$ was assigned to a leaf of the left sub-tree, there must be some other demand $\delta_m < \delta_{n/2+1}$ that was assigned to a leaf of the right sub-tree. But then $\delta_m < \delta_{n/2+1} < \delta_p$, and δ_p could not float to the right son of the root. We conclude that to maximize the weight saving by w^+ at level 1, $\{\delta_i\}_{i=1}^{n/2}$ must be assigned (in any order) to the leaves $\{\mu_i\}_{i=1}^{n/2}$, whereas $\{\delta_i\}_{i=n/2+1}^{n}$ must be assigned (in any order) to the leaves $\{\mu_i\}_{i=n/2+1}^{n}$ as shown in Fig. 4b.

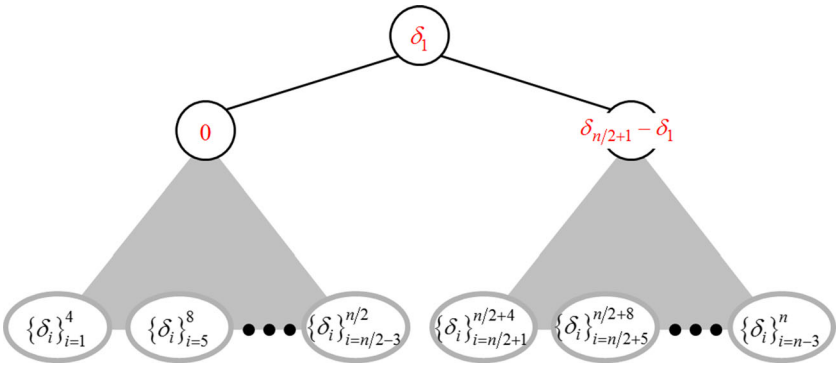


Fig. 5 $N-2$ induction step

Assume by induction that the total weight saving maximization from the root down to level $N-2$ by $\pi \in \mathbf{\Pi}$ must satisfy the assignment defined in (6), shown in Fig. 5.

$$\pi\{4j+i\}_{i=1}^{4} = \{4j+i\}_{i=1}^{4}, \quad 0 \le j \le 2^{N-2}-1. \tag{6}$$

It is required to show that the assignment of demands to leaves maximizing the total saving from root down to level $N-1$ complies with the induction hypothesis stated in (6). Recall that there is $w^{+}(\mu_{2k+1}) = 0$ and $w^{+}(\mu_{2(k+1)}) = \delta_{\pi^{-1}(2(k+1))} - \delta_{\pi^{-1}(2k+1)}$ (up to a swap), where $\delta_{\pi^{-1}(2k+1)}$ floats to the parent of $\left\{\mu_{2k+1}, \mu_{2(k+1)}\right\}$. The total sum of weights saving incurred at level $N-1$ is therefore

$$\sum_{k=0}^{n/2-1} \delta_{\pi^{-1}(2k+1)}. \tag{7}$$

To maximize (7), $\pi^{-1}(2(k+1)) = \pi^{-1}(2k+1)+1$ must hold. That follows by noticing that if $a < b < c < d$ then $a+c = \min(a,b)+\min(c,d) > \min(a,d) + \min(b,c) = a+b$ and $a+c = \min(a,b)+\min(c,d) > \min(a,c)+\min(b,d) = a+b$. Such relation must hold for any quadruple of $\delta_1 < \delta_2 < \cdots < \delta_n$, yielding the ordering $\left\{\left\{\delta_{4j+1}, \delta_{4j+2}\right\}, \left\{\delta_{4j+3}, \delta_{4j+4}\right\}\right\}_{j=0}^{n/4-1}$, which complies with (7). We conclude that up to sub-trees swap, the identity permutation $\pi^{\mathrm{id}}(i) = i$, $1 \le i \le n$ complies with (6), and together with the induction hypothesis yields optimal assignment. □

The optimality of π^{id} for general rooted tree follows by similar arguments as in the comments following Lemma 1.

4 Allocation of weight to nodes subject to product constraints

Referring to the amplification tree mentioned at Sect. 1, let us consider a balanced binary tree T with $n = 2^N$ leaves μ_i, associated with demands $\delta_i \ge 1$, $1 \le i \le n$. Denote by $P^{\bullet}(\nu, \rho)$ the path from the root $\rho \in T$ to a node $\nu \in T$, where the $\bullet$

superscript stands for a weight product operation defined on the path. To satisfy the demands, weights $w(v) \geq 1,\ v \in T$, are allocated such that

$$\prod_{v \in P^{\bullet}(\mu_i, \rho)} w(v) = \delta_i,\ 1 \leq i \leq n. \tag{8}$$

Let $\mathbf{W}^{\bullet}$ be the set of weight allocations satisfying (8). The allocation w^1, defined by $w^1(\mu_i) = \delta_i,\ 1 \leq i \leq n$ and $w^1(v) = 1, v \in T \setminus \{\mu_i\}_{i=1}^n$ trivially satisfies (8), hence $w^1 \in \mathbf{W}^{\bullet}$. Consider the total sum of weights $\Delta(w)$ allocated to T's nodes,

$$\Delta(w) = \sum_{v \in T} w(v). \tag{2}$$

Denote by Δ^* the minimum total weights across all $w \in \mathbf{W}^{\bullet}$,

$$\Delta^* = \min_{w \in \mathbf{W}^{\bullet}} \{\Delta(w)\}. \tag{9}$$

An allocation algorithm $w^{\bullet} \in \mathbf{W}^{\bullet}$, satisfying $\Delta(w^{\bullet}) = \Delta^*$, is subsequently presented. It determines the weights of T's nodes in a bottom-up traversal. Let μ' and μ'' be two sibling leaves of T, sharing a common parent v. Let their demands be δ' and δ'', respectively, and $\delta'' \geq \delta'$. The algorithm fixes $w^{\bullet}(\mu') = 1$ and $w^{\bullet}(\mu'') = \delta''/\delta'$, and *temporarily* allocates the weight δ' to v. This way the two sons share the weight allocated to their parent, while the ratio $\delta''/\delta' \geq 1$ is permanently allocated to the more demanding leaf. $w^{\bullet}$ thus yields $(\delta' + \delta'' + 1) - (1 + \delta''/\delta' + \delta') = \delta'' - \delta''/\delta' \geq 0$ *weight saving*.

$w^{\bullet}$ proceeds bottom-up level-by-level up to T's root ρ, which is necessarily allocated with the smallest demand among all the leaves. For two sibling nodes v' and v'' with a common parent, $w^{\bullet}$ fixes at least one of the weights $w^{\bullet}(v')$ and $w^{\bullet}(v'')$ to 1. It turns out that among the $2n - 1$ T's nodes, the weight of at least $n - 1$ nodes is fixed by $w^{\bullet}$ to 1.

Figure 6 illustrates $w^{\bullet}$ for an example of a tree with demands of 1 to 8 units, specified below their corresponding leaves. Temporary weights are shown in blue and fixed weights are shown in red. The total product of fixed weights along a root-to-leaf path satisfies the demand at the leaf.

While the total sum of demands specified at leaves is 36, the total sum of weights distributed at nodes has been reduced to 30.25 units, thus yielding savings of 5.75 units. The following Lemma shows that $\Delta(w^{\bullet}) = \Delta^*$.

Lemma 3 *Let T be a balanced binary tree with $n = 2^N$ leaves μ_i, associated with demands $\delta_i \geq 1,\ 1 \leq i \leq n$. There is $\Delta(w^{\bullet}) = \Delta^*$.*

Sketch of roof A straight forward modification of the proof of Lemma 1. The replacement of 0s by 1s, and sum of weights along root-to-node paths by their product, leads to same conclusions as in Lemma 1. □

By the same replacements as in Lemma 3, the optimality of $w^{\bullet} \in \mathbf{W}^{\bullet}$ for any rooted tree holds as in $w^{+} \in \mathbf{W}^{+}$.

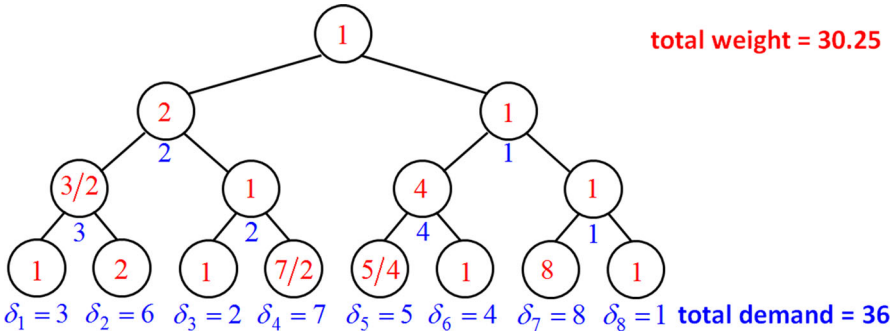


Fig. 6 Weight allocation by $w^\bullet$

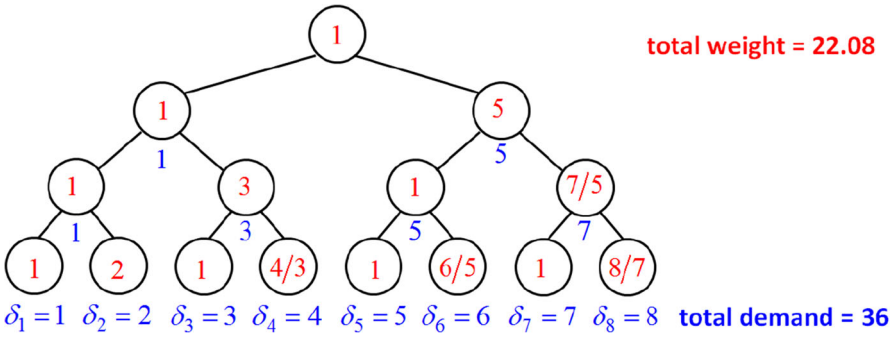


Fig. 7 Better assignment of demands to leaves, yielding smaller $w^\bullet$

5 Optimizing the assignment of demands subject to weight product constraints

Lemma 3 showed that the weight allocation $w^\bullet$ yields Δ^* defined in (9). Similarly to $w^+ \in \mathbf{W}^+$ discussed in Section 3, if the mapping of demands to leaves can arbitrarily be predetermined, Δ^* could further be reduced by considering $w^\bullet$ for each of the $n!$ possible mappings. An example is shown in Fig. 7, where the demands have been permuted and $w^\bullet$ yields 22.08 total sum of weights, compared to 30.25 in Fig. 6.

Let $1 \le \delta_1 \le \delta_2 \le \cdots \le \delta_n$ be n demands, and μ_i, $1 \le i \le n$, be the leaves of a balanced binary tree T. Similarly to (4), we look for $\pi^* \in \mathbf{\Pi}$ satisfying

$$\Delta(w^\bullet, \pi^*) = \min_{\pi \in \mathbf{\Pi}} \left\{ \Delta(w^\bullet, \pi) \right\}. \tag{10}$$

Consider the weight savings obtained by $w^\bullet$. Let $\{x, y, u, v\}$ be the weights assigned to the leaves of the tree in Fig. 8a. Since a flip of a sub-tree at a node does not matter for the total cost, it is assumed w.l.o.g that $x < y$ and $u < v$. Applying $w^\bullet$ results in the tree shown in Fig. 8b, with the following total weight saving at leaves

$$[(x + y) + (u + v)] - (y/x + v/u). \tag{11}$$

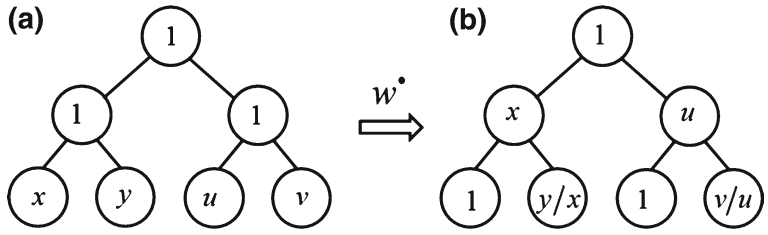


Fig. 8 Weight floatation under product constraints

Consider the demands $\{a, b, c, d\}$, $a < b < c < d$. How should those be assigned to the leaves in Fig. 8 so that the total saving of $w^\bullet$ is maximized? Up to sub-tree swap, there are three assignments to consider: $\{(a, b), (c, d)\}$, $\{(a, c), (b, d)\}$, and $\{(a, d), (b, c)\}$. We subsequently show that $\{(a, b), (c, d)\}$is the best. The term $[(x + y) + (u + v)]$ in (11) is independent of the order hence $(y/x + v/u)$ should be minimized. Comparing $\{(a, b), (c, d)\}$with $\{(a, c), (b, d)\}$, the weight saving difference is

$$\left(\frac{c}{a} + \frac{d}{b}\right) - \left(\frac{b}{a} + \frac{d}{c}\right) = (c - b)\left(\frac{1}{a} + \frac{d}{bc}\right) > 0, \tag{12}$$

hence $\{(a, b), (c, d)\}$ is favored. To compare $\{(a, b), (c, d)\}$ with $\{(a, d), (b, c)\}$, the weight saving difference is

$$\left(\frac{d}{a} + \frac{c}{b}\right) - \left(\frac{b}{a} + \frac{d}{c}\right) = \frac{c^2 - bd}{bc} + \frac{d - b}{a}. \tag{13}$$

Let $f(b, c, d) = (c^2 - bd)/bc$. There is $\partial f/\partial c = 2/b + d/c^2 > 0$, namely, $f(b, c, d)$ is monotonic increasing in c. Since $b < c < d$, the right hand side of (13) is minimized for $c = b$, yielding $(d - b)(1/a - 1/b) > 0$, concluding that the weight assignment $\{(a, b), (c, d)\}$ yields higher saving than $\{(a, d), (b, c)\}$.

Lemma 2 which stated that $\pi^{\mathrm{id}} \in \mathbf{\Pi}$ minimizes $w^+ \in \mathbf{W}^+$, has a similar counterpart for $w^\bullet \in \mathbf{W}^\bullet$as follows.

Lemma 4 *Let $n = 2^N$, and $1 \le \delta_1 < \delta_2 < \cdots < \delta_n$ be demands assigned to the leaves of a balanced binary tree T. The identity permutation $\pi^{id}(i) = i,\ 1 \le i \le n$, satisfies*

$$\Delta(w^\bullet, \pi^{id}) = \min_{\pi \in \mathbf{\Pi}} \left\{\Delta(w^\bullet, \pi)\right\}. \tag{14}$$

Sketch of roof Analogous to the proof of Lemma 2, where one have to replace the 0s by 1s, and sum of weights along root-to-node paths by their product. The proof follows by induction on maximizing the weight savings obtained from the root down to a certain level of the tree. Similarly to Lemma 2, δ_1 must by $w^\bullet$ definition float to the root. The maximal possible savings from the root down to the first level is then $\delta_{n/2+1} - \delta_{n/2+1}/\delta_1$, implying the floatation of $\delta_{n/2+1}$ to the other root's son. With appropriate replacement of 0 by 1 and $\delta_{n/2+1} - \delta_1$ by $\delta_{n/2+1}/\delta_1$ in Fig. 4b, the temporary assignment of δ_1 and $\delta_{n/2+1}$ to the root's sons, necessarily requires that $\{\delta_i\}_{i=1}^{n/2}$ and $\{\delta_i\}_{i=n/2+1}^{n}$ be assigned to the leaves of different sub-trees.

The induction hypothesis about the assignment of the demands to different sub-trees at level $N-2$ is shown in Fig. 5 with appropriate replacement of 0 by 1 and $\delta_{n/2+1}-\delta_1$ by $\delta_{n/2+1}/\delta_1$. The completion of the induction follows from Fig. 8 and (12) and (13), which proves that the assignment of demands to leaves must be in monotonic order. □

Conclusions

This paper showed that apparently two different VLSI design optimization problems share a common tree structure, on which appropriate sum or product operations are defined. An $O(n)$ time complexity bottom-up weight allocation algorithm has been shown to optimally solve those problems. It is interesting to further explore other operations and constraints defined on trees that can optimally be solved by such algorithm.

References

Fishburn JP (1990) Clock skew optimization. IEEE Trans Comput 39(7):945–951

Tam S, Rusu S, Desai UN, Kim R, Zhang J, Young I (2000) Clock generation and distribution for the first IA-64 microprocessor. IEEE J Solid-State Circ 35(11):1545–1552

Jiang YM, Cheng KT (1999) Analysis of performance impact caused by power supply noise in deep submicron devices. In: Proceedings of the 36th annual ACM/IEEE Design Automation Conference

Benini JL, Vuillod P, Bogliolo A, De Micheli G (1997) Clock skew optimization for peak current reduction. J VLSI Sig Process 16:117–130

Churcher S, Longstaff SA (1998) Programmable Delay Element, US Patent 5,841,296

Kocher P, Jaffe J, Jun B (1999) Differential power analysis, Advances in Cryptology–CRYPTO'99, Lecture Notes in Computer Science. Springer, Berlin

Friedman EG (2001) Clock distribution networks in synchronous digital integrated circuits. In: Proceedings of the IEEE 89(5):665–692

Juyeon Kim, Deokjin Joo, Taewhan Kim (2013) An Optimal Algorithm of Adjustable Delay Buffer Insertion for Solving Clock Skew Variation Problem. In: Proceedings of the 50th Annual Design Automation Conference, Article No. 90.

Baker RJ (2011) CMOS: circuit design, layout, and simulation. Wiley, New York

Jiren Y, Piper J (1999) Floating-point analog-to-digital converter. In: Proceedings of ICECS'99. The 6th IEEE International Conference on Electronics, Circuits and Systems, vol. 3, pp. 1385–1388. IEEE