Taylor & Francis
Taylor & Francis Group

Optimization

A Journal of Mathematical Programming and Operations Research

ISSN: 0233-1934 (Print) 1029-4945 (Online) Journal homepage: https://www.tandfonline.com/loi/gopt20

Resource allocation in rooted trees for VLSI applications

Uri Wimer & Shmuel Wimer

To cite this article: Uri Wimer & Shmuel Wimer (2019): Resource allocation in rooted trees for VLSI applications, Optimization, DOI: 10.1080/02331934.2019.1576669

To link to this article: https://doi.org/10.1080/02331934.2019.1576669

Published online: 14 Feb 2019.

OPTIMIZATION
https://doi.org/10.1080/02331934.2019.1576669

Taylor & Francis
Taylor & Francis Group

Resource allocation in rooted trees for VLSI applications

Uri Wimer[a] and Shmuel Wimer [b]

[a]Department of Mathematics, Hebrew University, Jerusalem, Israel; [b]Engineering Faculty, Bar-Ilan University, Ramat Gan, Israel

ABSTRACT

Several optimization problems of modifying the weight of vertices in rooted trees, some of which are special cases of the inverse 1-median problem, are solved. Such problems arise in Very Large Scale Integration (VLSI) design of hardware security circuits, circuit synchronization, and analog-to-digital converters. These problems require assigning costly hardware (demands) to the leaves of rooted trees. One common property of these problems is that a resource allocated to an internal node can be shared by the entire sub-tree emanated at the node. Two types of problems are studied here. (1) A tree node employs an addition operation and the demands at the leaves are obtained by summing the resources allocated to nodes along the root-to-leaf paths. A linear-time bottom-up algorithm is shown to minimize the total resources allocated to tree nodes. (2) A tree's node employs a multiplication operation and the demands at the leaves are obtained by multiplying the resources allocated to nodes along the root-to-leaf paths. A bottom-up dynamic programming algorithm is shown to minimize the total resources allocated to the tree's nodes. While the above problems are usually solved by design engineers heuristically, this paper offers optimal solutions that can be easily programmed in CAD tools.

ARTICLE HISTORY
Received 23 June 2018
Accepted 26 January 2019

KEYWORDS
Resource allocation; dynamic programming; inverse 1-median; circuit security; delay distribution; analog-to-digital converter

MATHEMATICS SUBJECT CLASSIFICATIONS
90C27; 90C39

1. Motivation

Works on weight modification of the vertices in weighted tree graphs are studies by both the OR and the VLSI design automation communities. While the former are mostly interested in the combinatorial optimization aspects of such problems, the latter are focus on various emerging applications, some of which described below, but with less awareness of their combinatorial optimization properties.

Seemingly different problems occurring in VLSI circuits and system designs share a common tree optimization problem paradigm. Good examples are electronic devices containing private secret information such as smart cards, radio frequency identification (RFID) tags and other devices that require data protection by a secret key. This is done through cryptographic algorithms running

CONTACT Shmuel Wimer wimers@biu.ac.il

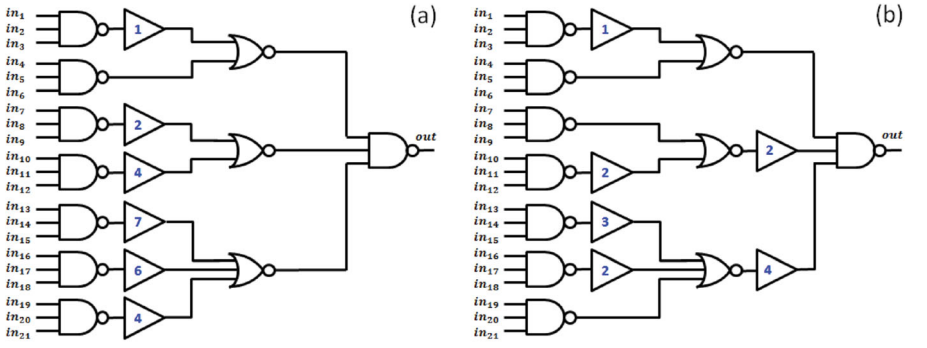

Figure 1. Securing ciphering logic by the insertion of delay buffers.

on dedicated hardware called a *substitution box* (SBOX) [1]. Unfortunately this hardware fails to protect the secret key from *side channel attacks* gleaned from hardware information leaks [2].

Countermeasures against side channel attacks are embedded in the cryptographic hardware that flatten their current waveform [3] , thus reducing the correlation between the current drawn from the power supply (observable by the attacker) and the secret key processed by the SBOX. This flattening is implemented by distributing delay buffers in the circuit as illustrated in Figure 1. There, a logic tree implements the ciphering of one bit of the output word. Every output bit has its own tree that implements a different ciphering logic. As shown in Figure 1(a), the signals from the inputs to the output need to be delayed by some prescribed amount written within a delay buffer. These buffers are costly and thus should be shared by the input-to-output paths. Figure 1(b) shows a possible distribution, such that the cumulative leaf-to-root delays satisfy the specifications in Figure 1(a) with a delay buffer cost reduced from 24 to 14 by sharing some of the delays among several input-to-output paths. An optimal delay distribution should minimize the total cost.

Another example has to do with the clock signals driving synchronous digital circuits. The clock signal is fed at the root of a clock-tree, shown in Figure 2, and distributed to the circuits connected at the leaves, where it should reach all of them simultaneously. The simultaneous switching of the clock signals at the leaves results in a high current peak, which is a source of *power supply noise* (instantaneous voltage fluctuations) that causes a failure in the circuit operation. Figure 2(b) illustrates how the insertions of delays into the clock-tree remedy this issue. The slightly misaligned clock signals result in a smaller current pulse, spread over a longer period of time. The displacement of the current peaks is obtained by inserting delay buffers in the nodes of the clock-tree [4]. While the required delays at the leaves are predefined, the total amount of delay inserted at the tree nodes need to be minimized. The work in [5] presented an $O(n \log n)$ algorithm to distribute the delay elements in the internal nodes of the clock-tree

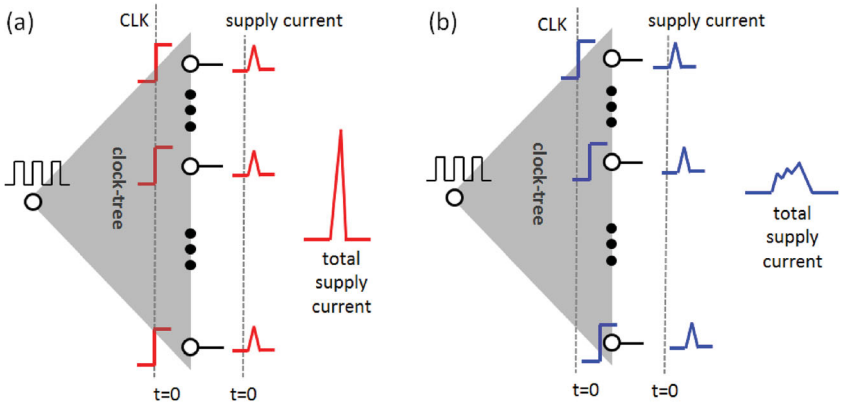

Figure 2. Reducing power supply noise by delay insertion into a clock-tree.

such that the total cost of the delay elements is minimized. This algorithm, however, is not optimal. A linear time algorithm for the special case of balanced binary trees was shown in [6]. This paper shows an $O(n)$ algorithm for any rooted trees.

Delay buffers at the nodes of clock-trees also support the tight control of the timing of the clock signal, which is crucial in VLSI systems. These include the *Clock skew* control [7] that fixes delay problems by *time borrowing* [8] and the adjustment of the clock signal timing so that the chip can operate at higher speeds, a technique called *binning* [9], which is used to maximize the selling price of chips.

The third example is the *analog-to-digital converter* (ADC) design [10] (Ch. 29), where amplifiers are connected at the tree's leaves as shown in Figure 3. There, the time-varying analog signal input is compared to various reference voltages to produce a binary code of the input voltage [11,12]. The cost of the circuits is proportional to its amplification factors, where unit amplification is the minimum possible. The amplifiers at the leaves can be replaced by amplifiers allocated at the internal nodes. The root-to-leaf amplification is then obtained by multiplying the factors along the corresponding path. These different VLSI problems are mapped to allocations of resources (delay buffers, amplifiers) to the tree nodes. The sum or the product of the resources along the root-to-leaf paths must satisfy the demands specified at the tree leaves, whereas their total sum needs to be minimized.

The remainder of this paper is organized as follows. Section 2 overviews related combinatorial optimization and VLSI work. Section 3 presents a linear-time resource allocation algorithm for the resource sum operation that supplies the demands at the leaves at minimal cost. Section 4 discusses the case where the assignment of the prescribed demands to the leaves is not predetermined. It shows that an assignment in increasing order of the demands is optimal. Section 5 presents a dynamic programming resource allocation algorithm for resource product operations.

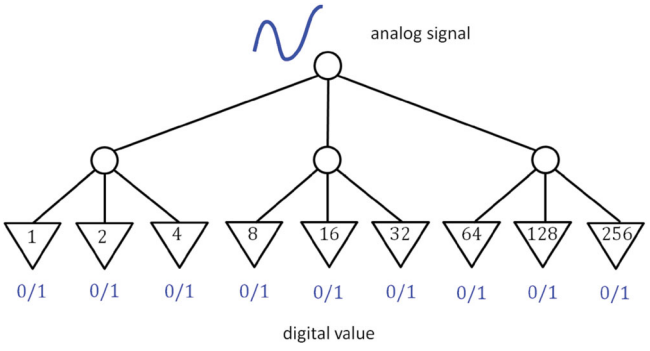

Figure 3. Analog-to-digital conversion tree.

2. Related work

Consider a tree $T(V, E)$ whose n vertices have nonnegative weights $w(v) \geq 0$, $v \in V$ and edges having nonnegative length $l(e) \geq 0$, $e \in E$. Let $P(u, v)$ be the path connecting u and v, $u, v \in V$, and let $L(P) = \sum_{e \in P} l(e)$ be the length of the path. A well-known combinatorial optimization problem is the 1-median tree (see e.g. [13]), aiming at finding a vertex $u^* \in V$ minimizing

$$\sum_{v \in V} w(v)L(P(u^*, v)). \tag{1}$$

The interest in 1-median tree problem is dates back to solving agricultural economics [14], and transportation [15] problems, both proved an interesting property that the solution depends solely on the vertex weights, regardless of the edge weights (nonnegative). The interest in the problem continues with extensions and variants, such as for cactus graphs [16] among many others. Most of the 1-median tree problem instances are solvable in linear time.

The study of the 1-median was followed by an inverse problem where the weights of the vertices are allowed to change within some prescribed bounds, such that a given vertex $u^{**} \in V$, $u^{**} \neq u^*$, is turning into $1 -$ median. The authors of [13] devised a greedy algorithm of $O(n \log n)$ time complexity which modifies the vertices' weights to turn u^{**} into 1-median, or else report that no feasible solution exists. Incorporation of some cost due to edge or vertex weight modifications have also been studied. In [17], a certain vertex of a tree was given, where it was desired to minimize the total sum of all other vertices distances to it, where the weights can be changed within some prescribed bounds. Assuming that each edge or vertex is associated with a unit-length or unit-weight change cost, it was desired that the total sum of cost changes is minimized. The authors solved it in $O(n^2)$ time complexity by splitting the tree into two sub-trees and balancing their heights (maximum distance to leaves). Some of the problems, we study in this paper are a special case which can be solved in $O(n)$ time. Another work in [18] presented a generalization of trees to block graphs, where the edge

lengths within each block had equal length. By using the convexity of the cost function the author showed solution of $O(n \log n)$ time.

Turning to VLSI design optimization, there are quite a few problems reminiscent of the inverse 1-median. There, the root of a tree distributes electrical signals to circuits connected at the leaves. The signals are modified by circuitry located at the nodes, which can be modelled by weights representing the complexity of the underlying circuits. In [3] delay elements (weights) were introduced into the logic tree of a cryptographic SBOX (see Figure 1), as a countermeasure against side channel attacks. The same authors presented in [19] a heuristic delay assignment algorithm for the case where different logic trees share common nodes. Though not discussed, its algorithm's pseudocode implies $O(n^3)$ time complexity. Considering their objectives, the underlying problem can be easily casted to inverse 1-median by using the cost in (1) as follows. First, by defining the unit-weight change cost at a node as the reciprocal of the number of leaves in its emanating subtree, and secondly, by setting the length of all the edges to zero except those connected to leaves, which are set to one.

Another type of VLSI design optimization problems regards the distribution of the clock signal over the chip as in the seminal work [7] and in [8]. Such problems are solved as a part of the clock-tree synthesis (CTS), an essential part of chip design flow. CTS is attracting lately high attention and extensive study due to the very aggressive design constraints imposed on the clock signal distribution in the era of nanometer-scale VLSI silicon technologies. A well-known technique to meet such constraints is to skew the clock signal by inserting delay buffers (weights) into the clock tree's nodes. The authors of [20,21] considered such problem for chips operated in multiple voltages. They claimed for optimal solution (by experiments and simulations) with the objective of minimizing the number of adjustable delay buffers, with delays bounded in some range. Their algorithm, however, is inefficient since it is based on solving *set-covering* problem, which is intractable and not scalable to industrial-size problems. It can be however be transformed similarly to inverse 1-median problem.

Constraints imposed on the clock signal by different operational modes and temperatures were solved in [22]. The CTS optimization incorporated iterations of solving a linear programming problem followed by an ad-hoc heuristic to adjust the delays (weights) inserted to the tree's nodes. Though LP has polynomial time complexity, it is still inefficient, since similar casting to inverse 1-median problem applies. The next section shows that due to its special setting, the corresponding optimization problem of weight allocations to tree's nodes can be solved directly in $O(n)$ time.

3. Optimal resource allocation subject to sum constraints

We use $\delta(\mu)$ to denote the demand (required delay) at a leaf μ. Let T be a rooted tree having leaves μ_i, associated with corresponding demands $\delta(\mu_i) = \delta_i \geq 0$, $1 \leq i \leq n$. Denote by $P(\mu, \rho)$ the path from a node $\mu \in T$ to the root $\rho \in T$. To satisfy the demands, resources $w(\mu)$, $\mu \in T$, are allocated such that

$$\sum_{\mu \in P(\mu_i, \rho)} w(\mu) = \delta(\mu_i),\ w(\mu) \geq 0,\ 1 \leq i \leq n. \tag{2}$$

Let $\mathbf{W}^+$ be the set of resource allocations satisfying (2). The allocation w^0, defined by $w^0(\mu_i) = \delta_i$, $1 \leq i \leq n$, and $w^0(\mu) = 0$, $\mu \in T \backslash \{\mu_i\}_{i=1}^n$ satisfies (2) trivially; hence $w^0 \in \mathbf{W}^+ \neq \emptyset$. Let $w \in \mathbf{W}^+$ and consider the total sum of resources $\Delta(w)$ allocated to T's nodes,

$$\Delta(w) = \sum_{\mu \in T} w(\mu). \tag{3}$$

Denote by Δ^* the minimum total sum of resources over $\mathbf{W}^+$,

$$\Delta^* = \min_{w \in \mathbf{W}^+} \{\Delta(w)\}. \tag{4}$$

A simple, linear-time resource allocation algorithm, $w^+ \in \mathbf{W}^+$, satisfying $\Delta(w^+) = \Delta^*$ is described below. It determines the resources of T's nodes in a bottom-up traversal. Let M be a set of sibling leaves sharing a common parent ν whose son-degree is $d(\nu) \stackrel{\Delta}{=} |\mathrm{M}|$. Let $m = \min_{\mu \in \mathrm{M}}\{\delta(\mu)\}$ and $\mathrm{M}_{\min} \subseteq \mathrm{M}$ be defined by $\mathrm{M}_{\min} = \{\mu \in \mathrm{M} | \delta(\mu) = m\}$. The algorithm *fixes* $w^+(\mu) = 0\, \forall \mu \in \mathrm{M}_{\min}$ and $w^+(\mu) = \delta(\mu) - m\, \forall \mu \in \mathrm{M} \backslash \mathrm{M}_{\min}$. It then *temporarily* allocates the resource m to ν. This way the sibling leaves share the resource m allocated to their parent ν, while the excess is *permanently* allocated to the more demanding leaves. The allocation w^+ thus yields $(d(\nu) - 1)m$ *resource savings*. The allocation w^+ proceeds bottom-up node-by-node and level-by-level up to T's root ρ. It turns out that the smallest demand out of all the n leaves floats and is allocated to the root $\rho \in T$. The resource allocation algorithm w^+ runs in linear time.

Figure 4 illustrates w^+ for a tree with prescribed demands of 1 to 11 units specified under their corresponding leaves. Temporary resources are shown in blue and permanent ones in black. The total sum of permanent resources along every leaf-to-root path satisfies the demand at the leaf. The total sum of demands is 66, but the total sum of resources allocated to the nodes is reduced to 42, thus yielding savings of 24. We show below that w^+ is optimal; namely, $\Delta(w^+) = \Delta^*$.

Theorem 3.1: *Let T be a rooted tree having n leaves μ_i, associated with demands $\delta_i \geq 0$, $1 \leq i \leq n$. There is $\Delta(w^+) = \Delta^*$.*

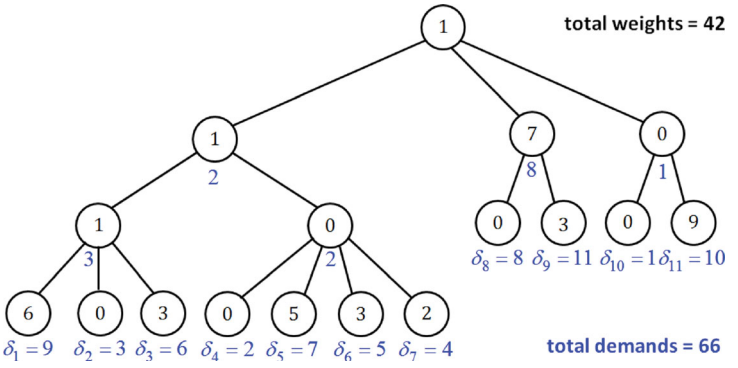

Figure 4. Resource allocation by w^+.

Proof: Let $w^* \in \mathbf{W}$ be such that $\sum_{\mu \in T} w^*(\mu) = \Delta^*$. We show by induction on the tree levels that there must exist $w^*(\mu) = w^+(\mu), \forall \mu \in T$. We first claim that w^* must also possess the property that for any set Mof sibling sons sharing a common parent ν, there must be a subset $\emptyset \neq \mathrm{M}^*_{\min} \subseteq \mathrm{M}$, with resources $w^*(\mu) = 0\, \forall \mu \in \mathrm{M}^*_{\min}$. Otherwise, if $\mathrm{M}^*_{\min} = \emptyset$, w^* can be modified to yield total resources less than Δ^* as follows. Since $\mathrm{M}^*_{\min} = \emptyset$, there is $\varepsilon = \min_{\mu \in \mathrm{M}}\{w^*(\mu)\} > 0$. Derive a new resource allocation w^{**} from w^* by subtracting ε from the resources allocated to every $\mu \in \mathrm{M}$ and adding ε to the parent ν of M's nodes, thus reducing Δ^* by $(|\mathrm{M}| - 1)\varepsilon$. It turns out that $\sum_{\mu \in T} w^{**}(\mu) < \Delta^*$, in contradiction to (4).

Assume that there exists a node $\mu' \in T$ such that $w^+(\mu') \neq w^*(\mu')$, and let it be w.l.o.g $w^+(\mu') > w^*(\mu') \geq 0$. We show that μ' cannot be a leaf of T. If it was, let M be the sibling leaves sharing a common parent ν such that $\mu' \in \mathrm{M}$ as shown in Figure 5. By w^+ definition and $w^+(\mu') > 0$, there is $\emptyset \neq \mathrm{M}^+_{\min} \subseteq \mathrm{M}$, $w^+(\mu) = 0\, \forall \mu \in \mathrm{M}^+_{\min}$ but $\mu' \notin \mathrm{M}^+_{\min}$. It was shown above that there is a subset $\emptyset \neq \mathrm{M}^*_{\min} \subseteq \mathrm{M}$, such that $w^*(\mu) = 0\ \forall \mu \in \mathrm{M}^*_{\min}$. There is $\mathrm{M}^+_{\min} \cap \mathrm{M}^*_{\min} \neq \emptyset$. To see this, let the leaf $\mu'' \in \mathrm{M}$ satisfy $\delta(\mu'') = \min_{\mu \in \mathrm{M}}\{\delta(\mu)\}$, so $\mu'' \in \mathrm{M}^+_{\min}$. If $\mu'' \notin \mathrm{M}^*_{\min}$ there is $w^*(\mu'') > 0$, and since $\mathrm{M}^*_{\min} \neq \emptyset$ there would exist another $\mu''' \in \mathrm{M}^*_{\min}$, satisfying$w^*(\mu''') = 0$. The parent ν is common to both μ'' and μ''', and hence

$$\delta(\mu'') = \sum_{\mu \in P(\mu'', \rho)} w^*(\mu) = w^*(\mu'') + \sum_{\mu \in P(\nu, \rho)} w^*(\mu) > \sum_{\mu \in P(\nu, \rho)} w^*(\mu) = w^*(\mu''') + \sum_{\mu \in P(\nu, \rho)} w^*(\mu) = \delta(\mu'''),$$

in contradiction to $\delta(\mu'') = \min_{\mu \in \mathrm{M}}\{\delta(\mu)\}$. Hence $\mu'' \in \mathrm{M}^+_{\min} \cap \mathrm{M}^*_{\min} \neq \emptyset$. This is depicted in Figure 5.

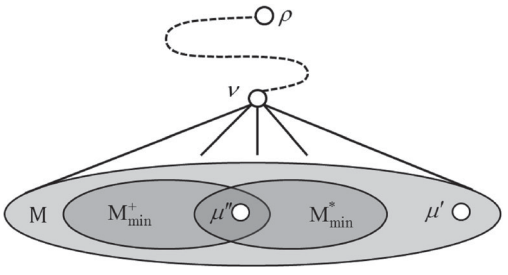

Figure 5. Illustration of Theorem 3.1.

Since μ'' is a leaf, there is by (2)

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

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

$$\sum_{\mu\in P(\nu,\rho)} w^+(\mu) = \sum_{\mu\in P(\nu,\rho)} w^*(\mu) = \delta(\mu'').$$

But since $w^+(\mu') > w^*(\mu') \geq 0$, there is

$$\delta(\mu') = \sum_{\mu\in P(\nu,\rho)} w^+(\mu) + w^+(\mu') > \sum_{\mu\in P(\nu,\rho)} w^*(\mu) + w^*(\mu') = \delta(\mu'),$$

which is impossible. Hence $w^+(\mu') > w^*(\mu') \geq 0$ cannot occur at a leaf and we conclude that w^* and w^+ are identical on T's leaves.

Let $w^+(\mu') \neq w^*(\mu')$ occur at an internal node. Assume by induction that all the nodes belonging to the rooted sub-tree $T(\mu')$, satisfy $w^+(\mu) = w^*(\mu)$ $\mu \in T(\mu')\backslash\{\mu'\}$. Let us obtain a tree T' by deleting $T(\mu')\backslash\{\mu'\}$. Since w^+ and $w^*$are identical on $T(\mu')\backslash\{\mu'\}$, the resource allocations w^+ and w^* have same total sum along leaf-to-root path for all of T''s leaves. On the other hand, μ' is a leaf of T' for which $w^+(\mu') \neq w^*(\mu')$, which has already been shown to be impossible. ■

4. Optimizing the assignment of demands at leaves

Theorem 3.1 showed that the resource allocation w^+ yields Δ^* as defined in (4). If the assignment to leaves of the prescribed demands is not predetermined, Δ^*can be further reduced by considering w^+ for each of the $n!$ possible assignments. An example is shown in Figure 6, where the demands are assigned to leaves differently than in Figure 4. The w^* resource allocation yields a total of 21 compared to 42 in Figure 4. This degree of optimization is very useful when the specific assignment of demands (delays) to leaves is immaterial as in the case of the ciphering logic tree in Figure 1 and the current pulse spreading in Figure 2. It enables significant hardware savings.

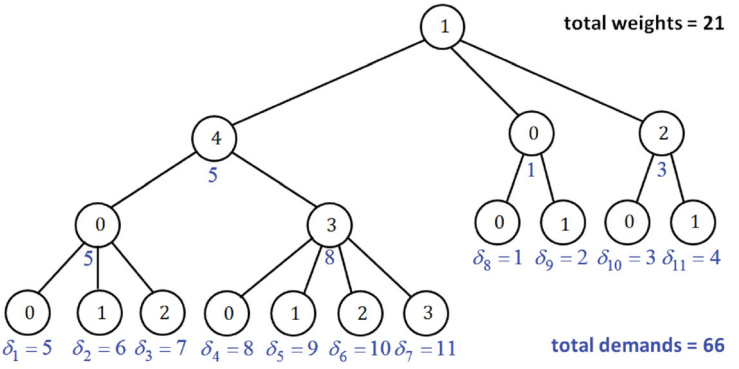

Figure 6. Better assignment of demands to leaves, yielding smaller w^*.

Let $0 \leq \delta_1 \leq \delta_2 \leq \cdots \leq \delta_n$ be n demands, and μ_i, $1 \leq i \leq n$, be the leaves of a tree T. We may assume w.l.o.g. that $\delta_1 < \delta_2 < \cdots < \delta_n$ since equalities can be arbitrarily 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 $\Delta(w, \pi)$ denotes the dependence of the total sum of the resources on both the assignment of the demands to leaves and the resource allocation $w \in \mathbf{W}$ to supply these demands. We therefore look for $\pi^+ \in \mathbf{\Pi}$ satisfying

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

It is shown below that π^+ must assign the demands to sibling leaves sharing a common parent contiguously, such that the ranges of demands allocated to sibling leaves having different parents are disjoint.

Lemma 4.1: *Let* M' *and* M'' *be any two sets of sibling leaves having different parents. If the permutation* π^+ *satisfies (5) then either* $\max_{\mu \in \mathrm{M}'} \delta(\mu) < \min_{\mu \in \mathrm{M}''} \delta(\mu)$ *or* $\max_{\mu \in \mathrm{M}''} \delta(\mu) < \min_{\mu \in \mathrm{M}'} \delta(\mu)$.

Proof: If neither of the inequalities is satisfied, then one of the two situations must occur

$$\begin{aligned} &\text{(a)} \ \min_{\mu \in \mathrm{M}'} \delta(\mu) < \min_{\mu \in \mathrm{M}''} \delta(\mu) < \max_{\mu \in \mathrm{M}'} \delta(\mu), \ \text{(b)} \ \min_{\mu \in \mathrm{M}''} \delta(\mu) \\ &< \min_{\mu \in \mathrm{M}'} \delta(\mu) < \max_{\mu \in \mathrm{M}''} \delta(\mu). \end{aligned} \tag{6}$$

Consider case (a) in (6). Let α and β be the leaves on which M' and M'' achieve their minimum, namely, $m' = \delta(\alpha) = \min_{\mu \in \mathrm{M}'} \delta(\mu)$ and $m'' = \delta(\beta) = \min_{\mu \in \mathrm{M}''} \delta(\mu)$. Let γ be the leaf on which M' achieves its maximum, namely, $\delta(\gamma) = \max_{\mu \in \mathrm{M}'} \delta(\mu)$. Let ν' and ν'' be the parents of M' and M'', respectively. The savings obtained by w^+ at ν' and ν'' are $m'(d(\nu') - 1)$ and $m''(d(\nu'') - 1)$, respectively. Let us swap the demands assigned to β and γ.

Since $\delta(\gamma) > \delta(\beta) = m''$, it turns out that the new minimum demand $\tilde{m}''$ assigned to M'' is larger than m'', whereas the minimum demand assigned to M' is unchanged. Consequently, the savings obtained by w^+ at v'' are increased by $(\tilde{m}'' - m'')(d(v'') - 1) > 0$, whereas the savings $m'(d(v') - 1)$ at v' are unchanged. Case (b) in (6) is handled similarly. ■

The condition in Lemma 3.1 is unfortunately not sufficient in the general case. However, for the special case of a regular balanced tree it is, as shown in the following. Note that the total sum of leaf-to-root allocated resources is invariant to permuting the sibling sub-trees rooted at a node. Given a regular balanced tree T of son-degree d and depth N (its root is at level 0), there are d^N leaves and $(d^N - 1)/(d - 1)$ internal nodes. A certain allocation of resources to T's nodes implies therefore $(d!)^{\frac{d^N-1}{d-1}}$ *equivalent* allocations with an identical sum of leaf-to-root resources.

Theorem 4.1: *Let $0 \le \delta_1 < \delta_2 < \cdots < \delta_{d^N}$ be demands assigned to the leaves of a regular balanced tree T of depth N and node degree d. Up to equivalence, the identity permutation $\pi^{\mathrm{id}}(i) = i$, $1 \le i \le d^N$, satisfies $\pi^+ = \pi^{\mathrm{id}}$.*

Proof: Let μ_i, $1 \le i \le d^N$, be the left-to-right ordered leaves of T and $\delta_{\pi^{-1}(i)}$ be their corresponding demands assigned by $\pi \in \mathbf{\Pi}$. To prove (5) we will show that $\Delta(w^+, \pi^{\mathrm{id}})$ maximizes the resource savings given by $\sum_{i=1}^{d^N} \delta_i - \Delta(w^+, \pi^{\mathrm{id}})$. The proof of (5) proceeds by induction on the levels of T. We will show that the adherence of the separation of demands assigned to subtrees emanated at a node of a given level, as stated in Lemma1, maximizes the total resources saved from the root down to that level. At each level the demands assigned to all the leaves of the subtrees emanating at a node must be separated, which complies with any permutation equivalent to π^{id}.

Note that the floatation of the minimum resource m among the son nodes of a parent v, by w^+, yields resource savings of $m(d - 1)$, since m is trimmed from every son and added to the parent. Note also that due to the resource allocation equivalence under subtree permutations we can index the root's sons arbitrarily. For the first step of the induction the savings obtained at the d root's sons are considered. The largest demand that can float to any root's son, say v_d (permuting sons is equivalent), is $\delta_{d^N - d^{N-1}+1}$. This can only happen if it is the smallest demand among the d^{N-1} leaves $\delta_{d^N - d^{N-1}+1}, \delta_{d^N - d^{N-1}+2}, \ldots, \delta_{d^N}$, sharing the common ancestor v_d. Similarly, the second largest demand that can float to any son of the root, say v_{d-1}, is $\delta_{d^N - 2d^{N-1}+1}$, and as previously, can only happen if it is the smallest demand among the d^{N-1} leaves sharing the common ancestor v_{d-1}. Repeating this argument d times, we end up with the root's son v_1, to which the demand δ_1 is floated, where the demands $\delta_1, \delta_2, \ldots, \delta_{d^{N-1}}$ are assigned to the leaves of the subtree rooted at v_1. This assignment complies with any permutation equivalent to π^{id}.

Assume by induction that an assignment by a permutation equivalent to π^{id}, satisfying

$$\pi\{d^2k+i\}_{i=1}^{d^2} = \{d^2k+i\}_{i=1}^{d^2},\ 0 \le k \le d^{N-2}-1, \tag{7}$$

maximizes the total resource savings from the root down to level $N-2$. Specifically, a set of d^2 demands having contiguous values $\{\delta_{d^2k+i}\}_{i=1}^{d^2}$, $0 \le k \le d^{N-2}-1$, are assigned to d^2 leaves sharing a common ancestor at level $N-2$. The savings obtained by w^+ for the entire T are the sum of the total resource savings from the root down to level $N-2$ plus the resource savings at level $N-1$ (the leaves' parents).

Using the induction hypothesis, it remains to show that maximization of the resource savings incurred at level $N-1$ complies with any permutation equivalent to π^{id}. It follows from Lemma1 that the separation of the demands assigned to leaves into groups of d leaves satisfying (6) maximizes the resource savings at their parent. Consider the following grouping of demands

$$\left\{\left\{\left\{\delta_{(d^k-1)d^2+jd+i}\right\}_{i=1}^{d}\right\}_{j=0}^{d-1}\right\}_{k=0}^{N-2}. \tag{8}$$

The grouping in (8) satisfies the separation conditions of Lemma1 and hence maximizes the resource savings obtained by w^+ at level $N-1$. It also complies with (7) and with a permutation equivalent to $\pi^{\mathrm{id}}(i) = i$, $1 \le i \le d^N$, which completes the proof. ■

5. Optimal resource allocation subject to product constraints

Based on the amplification tree example in Section 1, and similar to Section 2, we are interested in rooted trees to which resources (amplifications) are allocated at their nodes, such that their product along the leaf-to-root paths yields the prescribed demands required at the leaves. Below we replace the sum operation $+$ by the product operation $\times$, and 0, which is the unit element of addition, by 1, the unit element of multiplication, which is the smallest possible resource. We use the notation $\delta(\mu)$ to denote the demand (required amplification) at a leaf μ.

Let T be a rooted tree having n leaves μ_i, associated with corresponding demands $\delta(\mu_i) = \delta_i \ge 1$, $1 \le i \le n$. To satisfy the demands, resources $w(\mu)$, $\mu \in T$, are allocated such that

$$\prod_{\mu \in P(\mu_i,\rho)} w(\mu) = \delta(\mu_i),\ w(\mu) \ge 1,\ 1 \le i \le n. \tag{9}$$

Let $\mathbf{W}^{\times}$ be the set of resource allocations satisfying (9). The allocation w^1, defined by $w^1(\mu_i) = \delta_i$, $1 \le i \le n$, and $w^1(\mu) = 1$, $\mu \in T \backslash \{\mu_i\}_{i=1}^{n}$ satisfies (9)

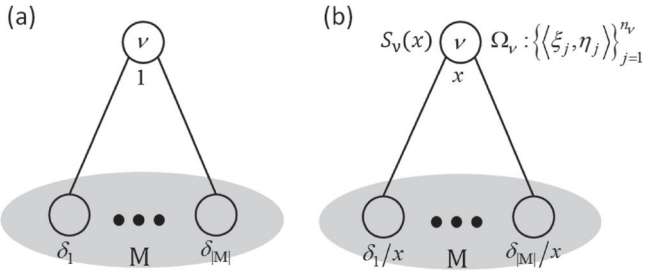

Figure 7. Scaling of the resources allocated at leaves.

trivially, hence $w^1 \in \mathbf{W}^\times \neq \emptyset$. Consider the total sum of resources $\Delta(w)$ allocated to T's nodes by $w \in \mathbf{W}^\times$,

$$\Delta(w) = \sum_{\mu \in T} w(\mu). \tag{10}$$

Denote by Δ^* the minimum total sum of resources over $\mathbf{W}^\times$,

$$\Delta^* = \min_{w \in \mathbf{W}^\times} \{\Delta(w)\}. \tag{11}$$

A dynamic programming resource allocation algorithm $w^\times$ which achieves (11) is described below. It determines the resources of T's nodes in a bottom-up traversal. Figure 7(a) shows a set M of sibling leaves and their parent nodeν. A leaf $\mu \in \mathrm{M}$ is assigned with some demand $1 \leq \delta(\mu)$, whereas the parent ν is allocated with one unit resource. Let $m = \min_{\mu \in \mathrm{M}}\{\delta(\mu)\}$. In order to reduce the cost (10) of the multiplication tree T, the transformation depicted in Figure 7(b) takes place. Some resource $1 \leq x \leq m$ is temporarily allocated at ν, whereas the sons are fixed with resources $w^\times(\mu) = \delta(\mu)/x \geq 1$, thus satisfying (9). The cost saving $S_\nu(x)$ obtained by such transformation is given by

$$S_\nu(x) = \underbrace{\left(1 - \frac{1}{x}\right) \sum_{\mu \in \mathrm{M}} \delta(\mu)}_{(a)} + \underbrace{(1 - x)}_{(b)}, \; 1 \leq x \leq m, \; \frac{\delta(\mu)}{x} \geq 1, \forall \mu \in \mathrm{M}. \tag{12}$$

Term (a) in (12) is the saving obtained by scaling down the resources allocated to leaves, whereas term (b) is the resource addition incurred at their parent ν.

The resources are realized in practice by VLSI amplifiers, whose amplification assumes discrete values with some resolution $0 < \varepsilon$. It follows from $w(\mu) \geq 1$ that the largest allocation at any node $\mu \in T$ cannot exceed $\max_{1 \leq i \leq n}\delta_i$. The amount R of distinct resource values is therefore bounded by

$$R = \left\lceil \frac{\max_{1 \leq i \leq n}\delta_i}{\varepsilon} \right\rceil. \tag{13}$$

We assume that the demands assigned at the leaves and the resources allocated to nodes are always an integral multiplication of ε. Written next to ν in Figure

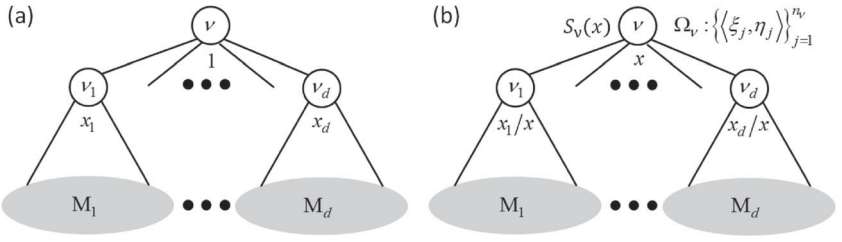

Figure 8. Upper level scaling of the resources.

7(b), the saving $S_\nu(x)$ is recorded for further minimization of the total resources (10) allocated to T. Recording at ν is made by a list of pairs $\Omega_\nu : \{\langle \xi_j, \eta_j \rangle\}_{j=1}^{n_\nu}$, $1 \leq \xi_j \leq m$, $n_\nu = (m-1)/\varepsilon$, sorted in increasing order of ξ_j, where η_j is obtained by substitution of ξ_j in (12).

Let us illustrate how savings are obtained by scaling a node ν at an upper level, as illustrated in Figure 8(a). Let M_i, $1 \leq i \leq d$, be the sets of sibling grandsons, and let ν_i be their parents. The nodes ν_i are the sons of ν, which is the grandparent of M_is. The node ν is initially allocated with one unit resource. Let the resources allocated at ν's grandsons be permanently fixed, after being scaled by some x_i, where

$$1 \leq x_i \leq m_i, m_i = \min_{\mu \in M_i} \{w^\times(\mu)\},\ 1 \leq i \leq d. \tag{14}$$

It follows that (9) is satisfied for all M_is. Assume further that for the temporary resource scaling at ν_i, the maximal resource savings function $S_{\nu_i}(x_i)$ is derived and recorded in a list $\Omega_{\nu_i} : \{\langle \xi_j, \eta_j \rangle\}_{j=1}^{n_{\nu_i}}$, as illustrated in Figure 7(b), where ξ_j is an admissible value of x_i.

Let $m = \min_{1 \leq i \leq d}\{m_i\}$, where m_is are defined in (14). As shown in Figure 8(b) a resource $1 \leq x \leq m$ is temporarily allocated at ν, whereas the sons ν_i are fixed with $w(\nu_i) = x_i/x \geq 1$, thus satisfying (9). The resource scaling yields savings of

$$S_\nu(x) = \max_{x_1,\ldots,x_d} \sum_{i=1}^{d} \left[S_{\nu_i}(x_i) + \left(1 - \frac{1}{x}\right) x_i \right] + (1-x),\ \frac{x_i}{x} \geq 1,\ 1 \leq x_i \leq m_i,\ 1 \leq i \leq d. \tag{15}$$

Maximization is possible by evaluating (15) for every admissible combination $(x_1, \ldots, x_d)$ among the $\prod_{i=1}^{d} n_{\nu_i}$ different possibilities. Knowing the maximal possible savings $S_\nu(x)$, the pair $\langle x, S_\nu(x) \rangle$ is recorded in the list $\Omega_\nu : \{\langle \xi_j, \eta_j \rangle\}_{j=1}^{n_\nu}$ sorted in increasing order of ξ_j, where every pair $\langle \xi_j, \eta_j \rangle$ memorizes $(x_1, \ldots, x_d)$ that obtained $\eta_j = S_\nu(\xi_j)$ in (15). The list Ω_ν thus obtained contains sufficient information representing the sub-tree rooted at ν for further minimization of the total resources allocated to T. Once the bottom-up traversal reaches the root ρ, the list $\Omega_\rho : \{\langle \xi_j, \eta_j \rangle\}_{j=1}^{n_\rho}$ yields the maximum resource saving achievable for every admissible resource ξ_j allocated to ρ. The maximum resource saving is obtained

by $\max_{1\le j\le n_\rho}\eta_j$. Since a pair $\langle\xi_j,\eta_j\rangle$ memorizes the sons' allocations $(x_1,\ldots,x_d)$ that yielded η_j, a top-down traversal can commit the resource allocated at every node which obtained the maximum resource savings achieved at the root.

To assess the computational complexity of the dynamic programming resource allocation $w^{\times}$, let the degree $d(\mu)$, $\mu\in T$ be bounded by $d(\mu)\le D$. Since T is a rooted tree having n leaves, it has a total of $O(n)$ nodes, where every node computes (15). The calculation of (15) examines all $(x_1,\ldots,x_{d(\mu)})$combinations, which from (13) is bounded by R^D. Each such combination is evaluated for every admissible value of x, yielding a total of R^{D+1} evaluations. We conclude that the overall computational complexity is bounded by $O(nR^{D+1})$. If T is a binary tree, which usually occurs in VLSI design, the complexity is bounded by $O(nR^3)$.

6. Conclusions

This paper showed that some different VLSI design problems all have the property that a resource allocated to an internal node of a rooted tree can be shared by the entire sub-tree emanated at the node. The goal of this allocation is to minimize the total sum of allocated resources. Two types of problems were studied that addressed trees whose nodes employ an addition operation, and trees whose nodes employ a multiplication operation. The former was solved by a linear-time bottom-up algorithm, whereas the latter was solved by a bottom-up dynamic programming algorithm.

Acknowledgements

We acknowledge the useful comments by the anonymous referee who enabled us to improve the manuscript.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This research was supported by the Israel Innovation Authority under the HiPer consortium of the MAGNET programme.

ORCID


Shmuel Wimer http://orcid.org/0000-0002-5728-0061

References

[1] Satoh A, Morioka S, Takano K, et al. A compact Rijndael hardware architecture with S-box optimization. International Conference on the Theory and Application of Cryptology and Information Security. Berlin: Springer; 2001.

[2] Kocher P, Jaffe J, Jun B, et al. Introduction to differential power analysis. J Cryptogr Eng. 2011;1(1):5–27.
[3] Levi I, Keren O, Fish A. Data-dependent delays as a barrier against power attacks. IEEE Trans Circ Syst I Reg Pap. 2015;62(8):2069–2078.
[4] Kaplan Y, Wimer S. Mixing drivers in clock-tree for power supply noise reduction. IEEE Trans Circ Syst I Reg Pap. 2015;62(5):1382–1391.
[5] Kim J, Joo D, Kim T. An optimal algorithm of adjustable delay buffer insertion for solving clock skew variation problem. Proceedings of the 50th Annual Design Automation Conference; 2013.
[6] Wimer S. Optimal weight allocation in rooted trees. J Combinat Optim. 2016;31(3): 1023–1033.
[7] Fishburn PJ. Clock skew optimization. IEEE Trans Comp. 1990;39(7):945–951.
[8] Tam S, Rusu S, Desai NU, et al. Clock generation and distribution for the first IA-64 microprocessor. IEEE J Solid-State Circ. Nov 2000;35(11):1545–1552.
[9] Churcher S, Longstaff SA. Programmable delay element. U.S. Patent 5,841,296. 1998 Nov 24.
[10] Baker JR. CMOS: circuit design, layout, and simulation, Vol. 1. John Wiley & Sons; 2008.
[11] Yuan J, Piper J. Floating-point analog-to-digital converter. The 6th IEEE International Conference on Electronics, Circuits and Systems, 1999. Proceedings of ICECS'99; 1999.
[12] Raphaeli D. Method and system for flash type analog to digital converter. U.S. Patent 8,957,801. 2015 Feb 17.
[13] Burkard RE, Pleschiutschnig C, Zhang J. Inverse median problems. Discrete Optim. 2004;1(1):23–39.
[14] Hua LK. Application of mathematical models to wheat harvesting. Chin Math. 1962;2:539–560.
[15] Goldman AJ. Optimal center location in simple networks. Trans Sci. 1971;5(2):212–221.
[16] Nguyen KT, Pham VC, Hai LH, et al. A simple linear time algorithm for computing a 1-median on cactus graphs. Appl Applied Math Int J. 2017;12(1):70–77.
[17] Alizadeh B, Burkard RE. Combinatorial algorithms for inverse absolute and vertex 1-center location problems on trees. Networks. 2011;58(3):190–200.
[18] Nguyen KT. Inverse 1-median problem on block graphs with variable vertex weights. J Optim Theory Appl. 2016;168(3):944–957.
[19] Levi I, Fish A, Keren O. CPA secured data-dependent delay-assignment methodology. IEEE Trans Very Large Scale Integr Syst. 2017;25(2):608–620.
[20] Kim J, Kim T. Useful clock skew scheduling using adjustable delay buffers in multi-power mode designs. Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific; 2015.
[21] Kim J, Kim T. Adjustable delay buffer allocation under useful clock skew scheduling. IEEE Trans Comp Aided Design Integr Circ Syst. 2017;36(4):641–654.
[22] Ewetz R, Koh C-K. MCMM clock tree optimization based on slack redistribution using a reduced slack graph. Design Automation Conference (ASP-DAC), 2016 21st Asia and South Pacific; 2016.