An Efficient Algorithm for Finding Double-Vertex Dominators in Circuit Graphs

Maxim Teslenko Elena Dubrova
Royal Institute of Technology, IMIT/KTH, 164 46 Kista, Sweden

Abstract

Graph dominators provide a general mechanism for identifying re-converging paths in circuits. This is useful in a number of CAD applications including computation of signal probabilities for test generation, switching activities for power and noise analysis, statistical timing analysis, cut point selection in equivalence checking, etc. Single-vertex dominators are too rare in circuit graphs to handle re-converging paths in a practical way. This paper addresses the problem of finding double-vertex dominators, which occur more frequently. First, we introduce a data structure, called dominator chain, which allows representing all possible $O(n^2)$ double-vertex dominators of a given vertex in $O(n)$ space, where n is the number of vertices of the circuit graph. Dominator chains can be efficiently manipulated, e.g. it takes constant time to look-up whether a given pair of vertices is a double-vertex dominator. Second, we present an efficient algorithm for finding double-vertex dominators. The experimental results show that the presented algorithm is an order of magnitude faster than existing algorithms for finding double-vertex dominators. Thus, it is suitable for running in an incremental manner during logic synthesis.

1 Introduction

This paper considers the problem of finding dominators in circuit graphs. A vertex *v* is said to *dominate* another vertex *u* if every path from *u* to the output of the circuit contains *v* [1]. For example, in the circuit graph in Figure 1, vertex *n* dominates vertex *e*; vertex *p* dominates vertex *h*.

Dominators provide a general mechanism for identifying re-converging paths in circuits. If a vertex *v* is the origin of a re-converging path, then the immediate dominator of *v* is the earliest point at which such a path converges. For example, in Figure 1, the re-converging path originated at *e* ends at *n*; the re-converging path originated at *g* ends at f.

Knowing the precise starting and ending points of a re-converging path is useful in a number of applications including computation of signal probabilities for test generation, switching activities for power and noise analysis, statistical timing analysis, cut point selection in equivalence checking, etc.

The *signal probability* of a net in a combinational circuit is the probability that a randomly generated input vector will produce the value one on this net [2]. Efficient signal probability analysis allows to improve the coverage of test generation for biased random simulation. The average *switching activity* in a combinational circuit is the probability of its net values to change from 0 to 1 or vice versa. It correlates directly with the average power dissipation of the circuit [3], thus its analysis is useful for guiding logic optimization methods targeting low power.

Computation of signal probabilities and switching activities based on topologically processing the circuit from inputs to outputs and evaluating the gate functions generally produces incorrect results due to higher-order exponents introduced by correlated signals. For example, if the functions f and g have variables in common, then $P[f \wedge g] \neq P[f] \cdot P[g]$, where P is the signal probability. Dominators provide the earliest points during topological processing at which the re-converging paths meet and thus the signals cease to be correlated. Therefore, the computation of signal probabilities and switching activities can be efficiently partitioned along the dominator points [4, 5, 6]. At the origin of a reconverging path, *v*, an auxiliary variable is introduced. At the end of the path, the immediate dominator of *v*, this variable is eliminated. As a result, the computation is carried out using a minimum set of variables.

Single-vertex dominators can be found in time linear in the number of vertices of the graph [7, 8, 9]. However, they are quite rare in circuits. It is more common that a vertex is dominated by a set of vertices. For example, in Figure 1, primary input *b* is dominated by the set $\{e,h\}$. To be able to deal with re-converging paths in a practical way, an efficient algorithm for computing multiple-dominators of a small size is needed. Small size is important because usually 2^k combinations of values of a *k*-vertex dominator have to be manipulated [4].

General algorithms for finding multiple-vertex dominators have exponential worst case complexity [10]. Multiple-vertex dominators of a fixed size *k* can be computed in $O(n^k)$ time, where *n* is the number of vertices of the circuit graph [11]. This paper presents an algorithm for finding double-vertex dominators ($k = 2$), which is significantly faster than the algorithm [11]. The efficiency of our algorithm is due to a number of interesting properties of double-vertex dominators. The most important one is that the set of all possible double-vertex dominators of a given vertex can be represented by a unique *dominator chain* of linear size which can be looked-up in constant time.

Being able to efficiently represent and manipulate *all* double-vertex dominators for a given vertex is important, because it makes the computation of common dominators easy. As we show in the paper, double-vertex dominators for a set of vertices can be derived from the dominator chains of individual vertices.

The paper is organized as follows. Section 2 presents the notation. Section 3 summarizes the previous work. Sections 4 and 5 introduce the new data structure and the dominator algorithm, respectively. The experimental results are shown in Section 6.

1530-1591/05 $20.00 © 2005 IEEE

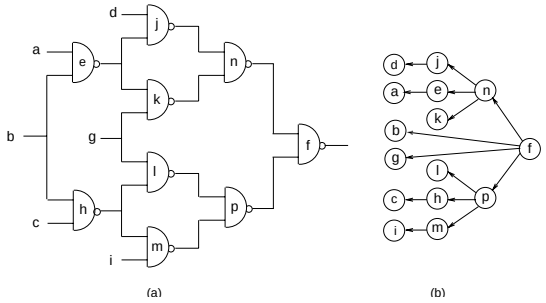

Figure 1: (a) Example circuit; (b) Its dominator tree.

2 Properties of Dominators

Let $C = (V, E, root)$ denote a single-output directed acyclic circuit graph, where V represents the set of gates and primary inputs and $E \subseteq V \times V$ describes the nets connecting the gates. A particular vertex $root \in V$ is marked as the circuit output.

A vertex v *dominates* another vertex u if every path from u to the *root* contains v [1]. Vertex v is the *immediate dominator* of u, if v dominates u and every other dominator of u dominates v. For example, in Figure 1, vertex n is the immediate dominator of j, e and k; vertex f is the immediate dominator of n and p. Every vertex $v \in V$ except *root* has a unique immediate dominator, $idom(v)$ [12]. The edges $\{(idom(v), v) \mid v \in V - \{root\}\}$ form a directed tree $T(C)$ rooted at *root*, called the *dominator tree* of C. Figure 1(b) shows the dominator tree for the circuit in Figure 1(a).

Dominators provide a general mechanism for identifying re-converging paths in circuit graphs. Every edge of the dominator tree $(idom(v), v) \in T(C)$ represents the starting and the ending points of a path. If the fanout degree of v, $Fanout(v) = \{u \mid (v,u) \in E\}$, is one, then the re-converging path is trivial (i.e. an edge). Otherwise, vertex v is the origin of a re-converging path and vertex $idom(v)$ is the earliest point at which such a path converges. For example, in Figure 1(a), the re-converging path originated at e ends at $idom(e) = n$; the re-converging path originated at b ends at $idom(b) = f$.

Many graphs do not have any single-vertex dominators except *root*. It is more common that a vertex is dominated by a set of vertices. Below we give a general definition for a set of vertices being dominated by another set. When the sizes of both sets are one, this definition reduces to the above mentioned definition from [1].

Definition 1 *A set of vertices $\{v_1, \ldots, v_k\}$ is a common multiple-vertex dominator of size k for a set of vertices $\{u_1, \ldots, u_l\} \subset V - \{v_1, \ldots, v_k\}$, if*

1. *every path from any u_j, $p \in \{1, \ldots, l\}$, to root contains some v_i, $i \in \{1, \ldots, k\}$*

2. *for every v_i, there exists at least one path from some u_p, $p \in \{1, \ldots, l\}$, to root which contains v_i and does not contain any other v_j, $i, j \in \{1, \ldots, k\}$, $i \neq j$.*

In the paper, we omit the word "common" when $l = 1$, i.e. when a set dominates a single vertex.

The second condition of the Definition 1 is needed to remove redundancies. For example, in Figure 1(a), all paths from e to f pass through the set of vertices $\{j, n\}$. However, vertex j is redundant, because n is a single-vertex dominator of e.

Note, that the notion of dominator is more general than the notion of *min-cut* in circuit partitioning [13]. A min-cut is required to dominate all vertices in its transitive fanin.

A number of properties are specific for multiple-vertex dominators. First, in single-vertex case, any vertex $v \in V - \{root\}$ is dominated by at least one vertex, *root*. Multiple-vertex dominators may not exist for some vertices. For example, a tree-like circuit does not have any multiple-vertex dominators. In a tree structure the condition (2) of the Definition 1 is not satisfied for any subset $\{v_1, \ldots, v_k\}$, $k > 1$, since the individual vertices v_i, $i \in \{1, \ldots, k\}$, dominate all vertices in their transitive fanins.

Second, the immediate k-vertex dominators are not unique for $k > 2$. We define immediate k-vertex dominators as follows.

Definition 2 *The set $W = \{v_1, \ldots, v_k\}$ in an immediate common k-vertex dominator of $\{u_1, \ldots, u_l\}$, if W is a common k-vertex dominator of $\{u_1, \ldots, u_l\}$ and there is no other common k-vertex dominator of $\{u_1, \ldots, u_l\}$, W', such that each vertex of W' is either dominated by W or belongs to W.*

Figure 1 gives an example. Vertex b has two immediate 3-vertex dominators: $\{e, l, m\}$ and $\{h, j, k\}$. In Section 4 we show that immediate k-vertex dominators are unique for $k = 2$.

3 Previous Work

The problem of finding single-vertex dominators was first considered in global flow analysis and program optimization. Lorry and Medlock [12] presented an $O(n^4)$ algorithm for finding all immediate single-vertex dominators in a flowgraph with n vertices. Successive improvements of this algorithm were done by Aho and Ullman [14], Purdom and Moore [15], and Tarjan [16], culminating in Lengauer and Tarjan's [1] $O(e\alpha(e,n))$ algorithm, where e is the number of edges and α is the standard functional inverse of the Ackermann function which grows slowly with e and n.

The asymptotic time complexity of finding single-vertex dominators was reduced to linear by Harel [7], Alstrup et al. [8] and Buchsbaum et al. [9]. However, these improvements in asymptotic complexity did not contribute much to reducing the actual runtime. For example, the algorithm [9] runs 10% to 20% slower than Lengauer and Tarjan's [1]. Lengauer and Tarjan algorithm appears to be the fastest of algorithms for single-vertex dominators on graphs of large size.

While it is possible to compute all single-vertex dominators in linear time, algorithms for finding all multiple-vertex dominators for a directed graph have exponential worst case complexity [10]. In [11], it was shown that it is possible to compute multiple-vertex dominators of a fixed size k in polynomial time. The algorithm presented in [11] finds the set of all possible k-vertex dominators for a circuit graph $C = (V, E, root)$ by iteratively restricting C with respect to one of its vertices, $v \in V$. The restriction is done by removing from V all vertices dominated by v, $S(v)$. Dominators of size $k-1$ are computed for the resulting restricted graph $C' = (V', E', root)$, with $V' = V - S(v)$ and

$E' = E - \{(u,w) | u \in S(v) \vee w \in S(v)\}$, by applying the same technique recursively. Once k is reduced to 1, a single-vertex dominator algorithm is used. Since single-vertex dominators can be computed in linear time, the overall complexity of the algorithm [11] is bounded by $O(|V|^k)$.

4 Dominator Chain

In this section, we introduce a data structure called *dominator chain* which allows representing all possible $O(|V|^2)$ double-vertex dominators of a given vertex in $O(|V|)$ space.

Definition 3 *For any $u \in V$, the dominator chain $D(u)$ is a vector of type*

$$\langle \{V_{11}, V_{21}\}, \{V_{12}, V_{22}\}, \ldots, \{V_{1m}, V_{2m}\} \rangle$$

whose elements V_{ij}, $i \in \{1,2\}$, $j \in \{1,\ldots,m\}$, $0 \leq m < |V|/2$, are vectors of vertices of V. Every pair $\{V_{1j}, V_{2j}\}$ satisfies the following properties:

1. *For every $v \in V_{ij}$, there exists a matching vector $W = \langle w_1, \ldots, w_s \rangle$, which is a sub-vector (i.e. linear interval) of V_{kj}, $k \in \{1,2\}, k \neq i$, such that*
   - *for all $r \in \{1,\ldots,s\}$, $\{v, w_r\}$ is a double-vertex dominator of u;*
   - *no other pair of vertices $\{v, v'\}$, $v' \in V - W$, is a double-vertex dominator of u;*
   - *the order of the elements of the matching vector W is given by:*

     if $\{v, w_r\}$ is a double-vertex dominator of w_t, then $t < r$, for all $r,t \in \{1,\ldots,s\}$, $r \neq t$.
2. *The immediate double-vertex dominator of u is the first elements of V_{11} and V_{21}. For all $j \in \{2,\ldots,m\}$, the immediate common double-vertex dominator of the last elements of V_{1j-1} and V_{2j-1} is the first elements of V_{1j} and V_{2j}. There is no common double-vertex dominator of the last elements of V_{1m} and V_{2m}.*
3. *No pair $\{V_{1j}, V_{2j}\}$, $\forall j \in \{1,\ldots,m\}$, can be partitioned into two pairs $\{V_{1j_1}, V_{2j_1}\}$ and $\{V_{1j_2}, V_{2j_2}\}$, where $V_{1j_1} \cup V_{1j_2} = V_{1j}$, $V_{1j_1} \cap V_{1j_2} = \emptyset$, $V_{2j_1} \cup V_{2j_2} = V_{2j}$, $V_{2j_1} \cap V_{2j_2} = \emptyset$, which satisfy properties 1 and 2.*

As an example, consider the circuit shown in Figure 2. The set of all double-vertex dominators for u is: $\{a,b\}$, $\{a,c\}$, $\{a,d\}$, $\{e,c\}$, $\{e,d\}$, $\{h,c\}$, $\{h,d\}$, $\{h,g\}$, $\{k,l\}$, $\{m,l\}$, $\{k,n\}$, $\{m,n\}$. The dominator chain for u is

$$\langle \{V_{11}, V_{21}\}, \{V_{12}, V_{22}\} \rangle = \langle \{\langle a,e,h \rangle, \langle b,c,d,g \rangle\}, \{\langle k,m \rangle, \langle l,n \rangle\} \rangle$$

The matching vector W of any vertex v in the dominator chain contains all vertices w such that $\{v,w\}$ is a double-vertex dominator of u. In Figure 2, the matching vector for vertex a is $\langle b,c,d \rangle$; the matching vector for d is $\langle a,e,h \rangle$. The first elements of V_{11} and V_{21}, $\{a,b\}$, are the immediate double-vertex dominator of u. The first elements of V_{12} and V_{22}, $\{k,l\}$, are the immediate common double-vertex dominator of the last elements of V_{11} and V_{21}, $\{h,g\}$. The pair $\{m,n\}$ does not have any common double-vertex dominator.

To prove that the dominator chain contains all possible double-vertex dominators of a vertex, we first show several fundamental properties of double-vertex dominators. Let $Dom(u)$ denote the set of all possible double-vertex dominators of u. The first Lemma says that, if two dominators have a vertex in common, then one of them dominates the non-common vertex of the other one.

Lemma 1 *If $\{v_1, v_2\} \in Dom(u)$ and $\{v_2, v_3\} \in Dom(u)$, then either $\{v_1, v_2\} \in Dom(v_3)$ or $\{v_2, v_3\} \in Dom(v_1)$.*

Proof: Suppose v_3 is not dominated by $\{v_1, v_2\}$. Since $\{v_2, v_3\} \in Dom(u)$, there exists a path from u to *root*, $p_{u \to root}$, which contains v_3 and does not contain v_2. Since $\{v_1, v_2\} \in Dom(u)$, $p_{u \to root}$ contains v_1. Furthermore, v_1 precedes v_3 in $p_{u \to root}$, because, by assumption, $\{v_1, v_2\} \notin Dom(v_3)$ and thus there exists a path $p_{v_3 \to root}$ which does not contain neither v_1 nor v_2.

The part $p_{u \to v_1}$ of the path $p_{u \to root}$ does not contain v_2 and v_3. Each path $p_{v_1 \to root}$ contains either v_2 nor v_3, because otherwise the existence of the path $p_{u \to v_1}$ $p_{v_1 \to root}$ would contradict $\{v_2, v_3\} \in Dom(v_1)$. Thus, by Definition 1, $\{v_2, v_3\} \in Dom(v_1)$. Similarly we can show that $\{v_2, v_3\} \notin Dom(v_1)$ implies $\{v_1, v_2\} \in Dom(v_3)$.

□

The second Lemma covers the case of two dominators with no vertices in common.

Lemma 2 *If $\{v_1, v_2\} \in Dom(u)$ and $\{v_3, v_4\} \in Dom(u)$, such that at least one of the vertices of $\{v_1, v_2\}$ is not dominated by $\{v_3, v_4\}$ and vice versa, then either*

$$\{v_1, v_4\} \in Dom(u) \text{ and } \{v_2, v_3\} \in Dom(u)$$

or

$$\{v_1, v_3\} \in Dom(u) \text{ and } \{v_2, v_4\} \in Dom(u)$$

Proof: Suppose that $\{v_3, v_4\} \notin Dom(v_1)$ and $\{v_1, v_2\} \notin Dom(v_4)$. Then, there exists a path $p_{v_1 \to root}$ which does not contain neither v_3 nor v_4. Also, there exists a path $p_{v_4 \to root}$ which does not contain neither v_1 nor v_2. Two cases are possible: (1) there is a path $p_{v_1 \to v_4}$; (2) there is no such path.

case 1: (a) Suppose that v_1 precedes v_4 in $p_{v_1 \to v_4}$. Then, all paths from u to v_1 contain v_3, since $\{v_3, v_4\} \in Dom(u)$ and $p_{v_1 \to root}$ exists. Thus, every path $p_{u \to root}$ containing v_1 contains v_3 as well. Since $\{v_1, v_2\} \in Dom(u)$, this implies that $\{v_2, v_3\} \in Dom(u)$.

(b) Suppose that v_4 precedes v_1 in $p_{v_1 \to v_4}$. Then, all paths $p_{u \to v_4}$ contain v_2, since $\{v_1, v_2\} \in Dom(u)$ and $p_{v_4 \to root}$ exists. Thus, every path $p_{u \to root}$ containing v_4 contains v_2 as well. Since $\{v_3, v_4\} \in Dom(u)$, this implies that $\{v_2, v_3\} \in Dom(u)$.

case 2: If there is no path $p_{v_1 \to v_4}$, then, similarly to (a), every path $p_{u \to root}$ containing v_1 should contain v_3 as well. Since $\{v_1, v_2\} \in Dom(u)$, this implies that $\{v_2, v_3\} \in Dom(u)$.

Consider vertices v_2 and v_3. Two cases are possible: (1) there exists a path $p_{v_2 \to v_3}$, (2) there is no such path.

case 1: (a) Suppose that v_2 precedes v_3 in $p_{v_2 \to v_3}$. Then $\{v_1, v_2\} \in Dom(v_3)$ implies that v_1 is a single-vertex dominator of v_3. Thus, $\{v_3, v_4\} \in Dom(u)$ implies that $\{v_1, v_4\} \in Dom(u)$.

(b) Suppose that v_3 precedes v_2 in $p_{v_2 \to v_3}$. Then, $\{v_3, v_4\} \in Dom(v_2)$ implies that v_4 is a single-vertex dominator of v_2. Thus, $\{v_1, v_2\} \in Dom(u)$ implies that $\{v_1, v_4\} \in Dom(u)$.

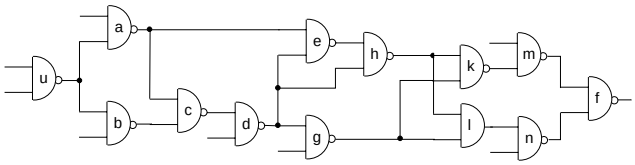

Figure 2: Example circuit.

case 2: If there is no path $p_{v_2 \to v_3}$, then, similarly to (a), $\{v_1, v_2\} \in Dom(v_3)$ implies that v_1 is a single-vertex dominator of v_3. Thus, $\{v_3, v_4\} \in Dom(u)$ implies that $\{v_1, v_4\} \in Dom(u)$. □

Next, we prove that immediate double-vertex dominators are unique. As we showed in Section 2, this property does not extend to dominators of larger size.

Theorem 1 *For every vertex $u \in V$, the immediate double-vertex dominator, if it exists, is unique.*

Proof: By contradiction. Suppose u has two immediate double-vertex dominators: $\{v_1, v_2\}$ and $\{v_3, v_4\}$. Two cases are possible: (1) $\{v_1, v_2\}$ and $\{v_3, v_4\}$ have one common vertex, and (2) $\{v_1, v_2\}$ and $\{v_3, v_4\}$ do not have common vertices.
case 1: Suppose v_2 is the common vertex, i.e. the second immediate dominator is $\{v_2, v_3\}$. By Definition 2, every vertex of an immediate dominator cannot be dominated by any other dominator. Since v_2 is common, it should hold that $\{v_1, v_2\} \notin Dom(v_3)$ and $\{v_2, v_3\} \notin Dom(v_1)$. This contradicts Lemma 1, which says that either $\{v_1, v_2\} \in Dom(v_3)$ or $\{v_2, v_3\} \in Dom(v_1)$.
case 2: By Lemma 2, either (1) $\{v_1, v_2\}$ dominates both vertices in $\{v_3, v_4\}$ (or vice versa), or (2) there exists two other dominators of u, each having a vertex in common with both $\{v_1, v_2\}$ and $\{v_3, v_4\}$. In the first case, one of the dominators does not satisfy Definition 2. In the second case, **case 1** applies to show that only one of the overlapping dominators can be immediate.

□

Now we give the main result of the paper. The proof is based on Lemmata 1 and 2 and Theorem 1.

Theorem 2 *The dominator chain exists for any $u \in V$. It contains all possible double-vertex dominators of u. All pairs $\{V_{1j}, V_{2j}\}$, $j \in \{1, \ldots, m\}$, are uniquely defined. For each $i \in \{1,2\}$, the overall number of vertices $\sum_{j=1}^{m} |V_{ij}|$ is smaller than the longest path from u to root.*

For vertices with no double-vertex dominators, e.g. *root*, the dominator chain is an empty vector.

The following Lemma shows that different vectors V_{ij}'s of a dominator chain do not intersect.

Lemma 3 *Any two distinct vectors V_{ij} and V_{kl} in the dominator chain do not have vertices in common:*

$$V_{ij} \cap V_{kl} = \begin{cases} V_{ij} = V_{kj}, & \text{if } i = k \text{ and } j = l, \\ \emptyset, & \text{otherwise} \end{cases}$$

for all $i, k \in \{1,2\}$, $j, l \in \{1, 2, \ldots, m\}$.

It follows from Lemma 3 that the dominator chain can be represented in an $O(|V|)$ space. To make possible constant-time look-up in the dominator chain $D(u)$, three parameters are assigned to vertices:

- For all $v \in V$: $flag(v) \in \{1,2\}$, distinguishing whether v belongs to V_{1j} or to V_{2j}. In the example in Figure 2, vertices b, c, d and g belong to V_{21}, hence their $flag$ equals 2.
- For all $v \in D(u)$: $index(v) \in \{1, 2, \ldots, \sum_{j=1}^{m} |V_{ij}|\}$ indicating the position of v in the vector $\langle V_{i1}, V_{i2}, \ldots, V_{im} \rangle$, $i = flag(v)$. By Lemma 3, $index(v)$ is uniquely defined (up to permutation of vectors V_{1j} and V_{2j} in the pairs $\{V_{1j}, V_{2j}\}$). In the example in Figure 2, $index(b) = 1$, $index(c) = 2$, $index(l) = 5$ and $index(n) = 6$.
- For all $v \in D(u)$: a pair $(min(v), max(v) = (index(w_1), index(w_{|W|}))$, where w_1 and $w_{|W|}$ are the first and the last vertices of the matching vector W of v. In the example in Figure 2, $(min(b), max(b)) = (1,1)$, $(min(c), max(c)) = (1,3)$, $(min(d), max(d)) = (1,3)$ and $(min(g), max(g)) = (3,3)$.

Then, checking whether $\{v_1, v_2\}$ dominates u can be done as follows:

1. Check whether $flag(v_1)$ is not equal to $flag(v_2)$. If yes, go to step 2. Otherwise, $\{v_1, v_2\} \notin D(u)$.
2. Check whether $min(v_1) \leq index(v_2) \leq max(v_1)$. If yes, the $\{v_1, v_2\} \in D(u)$. Otherwise, $\{v_1, v_2\} \notin D(u)$.

For example, suppose we check whether $\{d, h\}$ dominates u in the example in Figure 2. Vertex d is in V_{21}, therefore $flag(d) = 2$. Vertex h is in V_{11}, i.e. $flag(h) = 1$. Since $flag(d) \neq flag(h)$, we continue to the step 2. We have $min(d) = 1$, $max(d) = 3$ and $index(h) = 2$. Since $1 \leq 2 \leq 3$ holds, we conclude that $\{d, h\}$ dominates u.

As another example, let us check whether $\{g, a\}$ dominates u. On step 1, $flag(g) = 2$ and $flag(a) = 1$. Since $flag(g) \neq flag(a)$, we continue to the step 2. We have $min(g) = max(g) = 3$, and $index(a) = 1$. Since $3 \leq 1 \leq 3$ does not hold, we conclude that $\{g, a\}$ does not dominate u.

The problem of computing common double-vertex dominators for a set of vertices $u_1, u_2, \ldots, u_k$ can be transformed to the problem of computing double-vertex dominators for a single vertex using the following technique. We add a "fake" vertex u as a predecessor of $u_1, u_2, \ldots, u_k$. Clearly, each $\{v_1, v_2\} \in D(u)$ is a common dominator for the set $u_1, u_2, \ldots, u_k$ as well. Thus, Definition 3 can be extended to the set of vertices $D(u_1, u_2, \ldots, u_k)$ with a small modification that the first elements of V_{11} and V_{21} represent the immediate common double-vertex dominator of the set $u_1, u_2, \ldots, u_k$. Similarly, all presented theorems and lemmata can be extended to common double-vertex dominators.

Dominator chain $D(u_1, u_2, \ldots, u_k)$ can be computed directly from the dominator chains of individual vertices $D(u_i)$ in $O(k \cdot min\{|D_{u_1}|, |D_{u_2}|, \ldots, |D_{u_k}|)$ time, $i \in \{1, \ldots, k\}$.

5 Dominator Algorithm

The presented algorithm takes as its input a Boolean circuit $C = (V, E, root)$ and a vertex $u \in V$. It returns the dominator chain $D(u)$. The pseudo-code is shown in Figure 3.

```
algorithm DOMINATORCHAIN(V,E,root,u)
  v = u;
  k = 1;
  last_index_1 = 0; last_index_2 = 0;
  while 1 do
      idom(v) = SINGLEIDOM(v,V,E,root);
      /* defines the end of search region */
      S = {v};
      while 1 do
          i = 1; j = 1;
          V_1k = Ø; V_2k = Ø;
          {w_1,w_2} = DOUBLEIDOM(S,V,E,idom(v));
          if {w_1,w_2} = Ø then
              break ;
          /* w_1 and w_2 become the 1st elements of V_1k and V_2k */
          index(w_1) = last_index_1 + 1; index(w_2) = last_index_2 + 1;
          ADDVECTOR(V_1k,1,⟨w_1⟩,w_2);
          ADDVECTOR(V_2k,2,⟨w_2⟩,w_1);
          while i ≤ |V_1k| or j ≤ |V_2k| do
              if i < |V_1k| then
                  i = UPDATECHAIN(1,2,i,k,w);
              else
                  j = UPDATECHAIN(2,1,j,k,w);
          /* end of while i ≤ |V_1k| or j ≤ |V_2k| do loop*/
          v_1 is the last element of V_1k;
          v_2 is the last element of V_2k;
          last_index_1 = index(v_1);
          last_index_2 = index(v_2);
          S = {v_1,v_2};
          /* v_1 and v_2 are the last elements of V_1k and V_2k */
          D(u) = APPEND(D(u),{V_1k,V_2k});
          k++;
      /* end of inner while 1 do loop */
      if idom(v) = root then
          return D(u);
      else
          v = idom(v);
  /* end of outer while 1 do loop */
end
```

Figure 3: Pseudo-code of the presented algorithm.

The outer **while**-loop partitions the circuit graph into regions using single-vertex dominators of u as cut points. Double-vertex dominators of u are searched within these regions.

The immediate double-vertex dominator for a given set S is obtained by $DoubleIDom(S,V,E,idom(v))$ by computing the *maximum flow* between the multiple sources defined by S and the sink $idom(v)$. Each vertex in the V except the source and sink vertices is assigned a unit capacity. The source and sink vertices are assigned infinite capacity. The capacity limits the amount of flow through a vertex. According to min-cut maxflow theorem [17], the maximum possible flow is equal to the capacity of the min-cut disconnecting the source and the sink. In our case, the maximal-volume min-cut of size two corresponds to the immediate double-vector dominator for S. If the size of the cut is larger than two, DOUBLEIDOM returns an empty set.

Maximum flow is computed by constructing *augmenting paths* from the sources to the sink [17]. The algorithm repeatedly seeks an augmenting path and uses it to increase the maximum flow. At the same time, the capacitance of all edges involved in the path gets reduced by a unit. The algorithm stops when no such augmenting path exists. Our version of the augmenting path algorithm uses vertex capacitances instead of edge capacitances. Edges are used to direct possible path construction only.

If the double-vertex dominator $\{w_1,w_2\}$ computed by DOUBLEIDOM is not an empty set, then w_1 is added to V_{1k} and w_2 is added to V_{2k} as first elements.

```
algorithm UPDATECHAIN(a,b,i,k,root)
  Find ith element of V_ak, v;
  W = FINDMATCHINGVECTOR(v,V_bk,root);
  ADDVECTOR(V_bk,b,W,v);
  i++;
  return i;
end

algorithm FINDMATCHINGVECTOR(v,V_bk,root)
  Find w ∈ V_bk with index(w) = min(v);
  W = ⟨w⟩;
  V' = V − {v};
  E' = E − {(y,v) | y ∈ V − {v}};
  while 1 do
      idom(w) = SINGLEIDOM(w,V',E',root);
      if idom(w) = root do
          return W;
      else
          W = APPEND(W,idom(w));
          w = idom(w);
end

algorithm UPDATEINDEX(i,W)
  Find ith element of W, v_i;
  if index(v_i) is not defined
      index(v_i) = UPDATEINDEX(i − 1,W) + 1;
  return index(v_1);
end
```

Figure 4: Pseudo-codes of UPDATECHAIN, FINDMATCHING VECTOR and UPDATEINDEX.

ADDVECTOR(V_{ak},a,W,v) modifies V_{ak} to accommodate the matching vector W of v as a part of it. Then, indexes of vertices of V_{ak} are updated by the function UPDATEINDEX$(|V_{ak}|,V_{ak})$ (Figure 4). Besides, ADDVECTOR assigns $(min(v),max(v)) = (index(w_1),index(w_{|W|}))$ in V_{ak}, where w_1 and $w_{|W|}$ are the first and the last elements of W, respectively. Further, for all w_i, $i \in \{1,\ldots,|W|\}$, the value a assigned to $flag(w_i)$, and the values of $min(w_i)$ and $max(w_i)$ are updated as follows:

1. $min(w_i)$ is assigned the minimal of values $\{min(w_i), index(v)\}$,
2. $max(w_i)$ is assigned the maximal of values $\{max(w_i), index(v)\}$,
3. if $min(w_i)$ and $max(w_i)$ are not defined yet, $(min(w_i),max(w_i))$ are set to $(index(v),index(v))$.

Next, the dominator chain is updated. UPDATECHAIN applies FINDMATCHINGVECTOR$(v,V_{bk},root)$ to find the matching vector for v which is ith element of V_{ak}. The resulting vector is added to V_{bk} using ADDVECTOR.

FINDMATCHINGVECTOR$(v,V_{bk},root)$ works as follows. First, we look-up V_{bk} to find a vertex w with $index(w) = min(v)$. Such a vertex w always exists. This vertex becomes the first element of W. Then, we compute the immediate single-vertex dominator of w for the restricted circuit $C' = (V',E',root)$, with $V' = V - v$ and $E' = E - \{(y,v)|y \in V - \{v\}\}$. By restricting the circuit we exclude from consideration all paths from u to *root* (*root* is local to FINDMATCHINGVECTOR$(v,root)$) which contain v. The computed $idom(w)$ together with v is a double-vertex dominator for u, thus $idom(w)$ is appended to the end of W. **While**-loop continues until the local *root* is reached.

When the current pair $\{V_{1k},V_{2k}\}$ is updated for all vertices, it is added to $D(u)$ as next element. When *root* is reached, DOMINATORCHAIN terminates by returning the resulting dominator chain $D(u)$ for u.

name	in	out	N single doms	N double doms	runtime, sec [11] t_1	runtime, sec new t_2	runtime, sec impr. t_1/t_2
C1355	41	32	6	10512	3.5	0.45	7.78
C1908	33	25	636	5696	1.5	0.36	4.17
C2670	233	140	2091	410	1.55	0.23	6.74
C3540	50	22	727	5657	6.85	0.42	16.31
C432	36	7	195	2127	0.3	0.17	1.76
C499	41	32	960	9968	2.3	0.45	5.11
C5315	178	123	4093	11068	5.5	0.71	7.75
C6288	32	32	480	3366	58.89	0.88	66.92
C7552	207	108	4604	14728	7.27	1.16	6.27
C880	60	26	432	1309	0.26	0.18	1.44
alu2	10	6	48	55	0.81	0.16	5.06
alu4	14	8	77	214	3.36	0.16	21.00
apex5	114	88	800	8107	3.21	0.61	5.26
apex6	135	99	525	1169	0.42	0.24	1.75
apex7	49	37	140	476	0.17	0.15	1.13
cmb	16	4	38	60	0.16	0.09	1.78
comp	32	3	8	439	0.16	0.12	1.33
cordic	23	2	38	65	0.12	0.1	1.20
des	256	245	3361	2349	8.19	0.77	10.63
frg2	143	139	1502	3609	1.76	0.44	4.00
i8	133	81	2068	3296	2.87	0.5	5.74
i9	88	63	876	1827	0.95	0.3	3.17
i10	257	224	6446	30608	16.32	1.57	10.39
pair	173	137	2459	9196	1.82	0.63	2.89
rot	135	107	1657	4617	1.49	0.38	3.92
term1	34	10	46	453	0.31	0.16	1.94
too_large	38	3	971	1467	423.73	0.69	614.1
x1	51	35	366	1297	0.99	0.22	4.50
x3	135	99	495	1801	0.68	0.22	3.09
x4	94	71	305	2250	0.41	0.18	2.28
average	95	67	1215	4607	18.5	0.42	27.65

Table 1: Benchmark results.

6 Experimental Results

This section compares the performance of the presented algorithm to the algorithm [11]. Table 1 summarizes the results for 30 largest benchmarks from IWLS'02 benchmark set. Columns 1, 2 and 3 show the name of the function, the number of primary inputs and primary outputs, respectively. Column 5 shows the total sum of double-vertex dominators which dominate at least one primary input. This number is the same for the presented algorithm and the algorithm [11], because they both compute all possible double-vertex dominators for a given vertex. For a comparison, we also show in Column 4 the total sum of single-vertex dominators which dominate at least one primary input, computed using Lengauer and Tarjan's algorithm [1]. In both cases, common dominators are counted only once. Every output is treated as a separate function. The numbers shown in Columns 4 and 5 are the total sum for all outputs of the circuit.

Columns 7 and 8 show runtime, in seconds, measured using the Unix command *time* (user time). The experiments were performed on a PC with a 650 MHz Pentium3 CPU and 256 MByte main memory. One can see that, on average, the presented algorithm is 27 times faster than the algorithm [11].

Some circuits may have less double-vertex dominators than single-vertex ones (*C2670, des*). Usually these are circuits with many single-vertex dominators. Recall that the definition of multiple-vertex dominator excludes redundancies. Therefore, in the extreme case of a tree-like circuit with n vertices "N single doms" would be n and "N double doms" would 0. No pair of vertices in a tree-like circuit satisfies the Definition 1.

7 Conclusion

This paper has two main contributions. First, we introduce a data structure for representing double-vertex dominators, which has a linear size and can be efficiently manipulated. Second, we design an algorithm for finding double-vertex dominators, which is, on average, an order of magnitude faster than the algorithm [11]. The speed of the presented algorithm makes it suitable for running in an incremental manner during logic synthesis.

Future work includes exploring new applications of the presented algorithm, e.g. statistical timing analysis.

Acknowledgments

We are grateful to the anonymous reviewer who gave us many valuable comments over the manuscript.

This work was supported in part by the Research Grant 2002-4300 from the Swedish Research Council Vetenskpsrådet.

References

[1] T. Lengauer and R. E. Tarjan, "A fast algorithm for finding dominators in a flowgraph," *Transactions of Programming Languages and Systems*, vol. 1, pp. 121–141, July 1979.

[2] K. P. Parker and E. J. McCluskey, "Probabilistic treatment of general combinational networks," *Transactions on Computers*, pp. 668–670, June 1975.

[3] F. Najm, "Transition density: A new measure of activity in digital circuits," *Transactions on Computer-Aided Design*, vol. 12, pp. 310–323, February 1993.

[4] R. Krenz, E. Dubrova, and A. Kuehlmann, "Fast algorithm for computing spectral transforms of Boolean and multiple-valued functions on circuit representation," in *Proceedings of the International Symposium on Multiple-Valued Logic*, (Tokyo, Japan), pp. 334–339, May 2003.

[5] S. C. Seth, L. Pan, and V. D. Agrawal, "PREDICT-probabilistic estimation of digital circuit testability," in *Proceeding of International Symposium on Fault-Tolerant Computing*, pp. 220–225, June 1985.

[6] J. Costa, J. Monteiro, and S. Devadas, "Switching activity estimation using limited depth reconvergent path analysis," in *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 184 –189, 1997.

[7] D. Harrel, "A linear time algorithm for finding dominators in flow graphs and related problems," *Annual Symposium on Theory of Computing*, vol. 17, no. 1, pp. 185–194, 1985.

[8] S. Alstrup, D. Harel, P. W. Lauridsen, and M. Thorup, "Dominators in linear time," *SIAM Journal on Computing*, vol. 28, no. 6, pp. 2117–2132, 1999.

[9] A. L. Buchsbaum, H. Kaplan, A. Rogers, and J. R. Westbrook, "A new, simpler linear-time dominators algorithm," *ACM Transactions on Programming Languages and Systems*, vol. 20, no. 6, pp. 1265–1296, 1998.

[10] R. Gupta, "Generalized dominators and post-dominators," in *Proceedings of 19th Annual ACM Symposium on Principles of Programming Languages*, pp. 246–257, 1992.

[11] E. Dubrova, M. Teslenko, and A. Martinelli, "On relation between non-disjoint decomposition and multiple-vertex dominators," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, IEEE, 2004.

[12] E. S. Lowry and C. W. Medlock, "Object code optimization," *Communications of the ACM*, vol. 12, pp. 13–22, January 1969.

[13] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning of electrical circuits," *Bell Systems Tech. Journal*, vol. 9, pp. 291–307, 1970.

[14] A. V. Aho and J. D. Ullman, *The Theory of Parsing, Translating, and Compiling, Vol. II*. Englewood Cliffs, NJ: Prentice-Hall, 1972.

[15] P. W. Purdom and E. F. Moore, "Immediate predominators in a directed graph," *Communications of the ACM*, vol. 15, pp. 777–778, August 1972.

[16] R. E. Tarjan, "Finding dominators in a directed graphs," *Journal of Computing*, vol. 3, pp. 62–89, March 1974.

[17] L. R. Ford and D. R. Fulkerson, "Maximum flow through a network," *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, 1956.

[18] J. Moondanos, C. H. Seger, Z. Hanna, and D. Kaiss, "CLEVER: Divide and conquer combinational logic equivalence verification with false negative elimination," in *Computer Aided Verification (CAV'01)*, (Paris, France), pp. 131–143, July 2001.