Observability Don't Care Sets and Boolean Relations

Maurizio Damiani and Giovanni De Micheli
Center for Integrated Systems
Stanford University
Stanford, CA 94305

Abstract

A new algorithm is presented that computes exact or approximate *observability don't care* (ODC) sets for multiple-level combinational network. The proposed algorithms are efficient because they use only local information. A method for deriving the equivalence classes of a *Boolean relation* from the observability *don't care sets* is then proposed. Experimental results on computing ODC sets are reported.

1 Introduction.

Over the past few years, the problem of computing efficiently and correctly observability *don't care* (ODC) sets has emerged as a central one in the synthesis of combinational networks [1], [2], [3]. The knowledge of the ODC sets is important in several respects, namely: 1) local minimization of functions in a Boolean network, 2) synthesis of 100% testable networks, 3) test pattern generation [4].

Bartlett proposed in [1] a computation of the ODC sets requiring the representation of the primary output expression in terms of the network intermediate variables. Such a representation may be subject to the explosion in size of the representation. Muroga proposed in [3] exhaustive simulations of the circuit for determining the observability *don't care* sets of vertices with reconvergent fanout. Other authors pointed out that it would be desirable and computationally much more efficient to derive the ODC set of a vertex of a Boolean network from the ODC sets of its direct fanout vertices [5], [8]. They showed, however, that a straightforward application of this idea could lead to erroneous results, because of the effect of reconvergent fanout. Therefore only approximate solutions have been proposed to compute the ODC sets [8], [9]. A critical review of previous work and its relation to the method for ODC computation presented here is reported in [11].

In this paper we present a new expression for the observability *don't care set* of a gate with reconvergent fanout, and we show how an exact computation of ODC sets is indeed possible by using only the ODC sets of the immediate fanout variables. We propose an algorithm for such a computation that traverses the network backward from the primary outputs to the primary inputs processing each vertex only once.

Unfortunately this method, leading to an exact computation of the ODC sets, involves implicitly Boolean complementations and it is consequently prone to the well-known phenomenon of "combinational explosion". We therefore propose two other algorithms for the computation of subsets of the actual ODC sets, still based on local information. The first algorithm computes an ODC subset at each vertex of the Boolean network from the ODC subsets of its direct fanout vertices. The second one computes *both* subsets of the actual *care* and *don't care* sets from those of its fanout. We implemented both approaches in computer program SPY. Full ODC sets have been computed for some difficult benchmark examples.

Eventually the relationship between observability *don't care* sets and the equivalence classes of *Boolean Relations* [10]) is explored. In particular, it is shown how the output equivalence classes of a network can be computed directly from the ODC sets of the individual network outputs, thus avoiding the necessity of any flattening operation.

2 Definitions and Notations.

2.1 Combinational Boolean Networks.

In this paper we model multiple-output combinational circuits by Boolean networks [1]. A Boolean network N with n input vertices and m output vertices realizes a function $\underline{\mathcal{F}} : B^n \rightarrow B^m$ [1], where B is the Boolean set $\{0,1\}$. Underlining is used for denoting vector quantities in this paper.

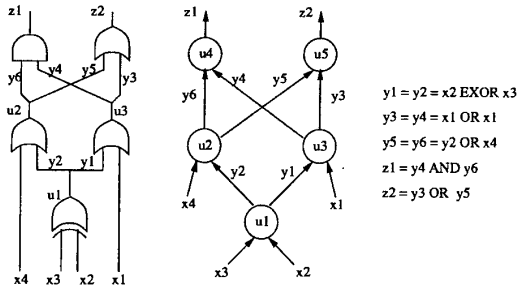

Figure 1: A multiple-level combinational circuit and its associated Boolean network representation.

The network is specified by an acyclic graph $G = (V, E)$. The elements of the vertex set $V = V^I \cup V^G \cup V^O = \{v\}$ are in one-to-one correspondence with primary inputs, logic gates, and primary outputs, respectively. There is a directed edge e from a vertex μ to a vertex ν if the output of the gate in μ is connected to an input of the gate in ν. In contrast to [1], and similarly to [3], we associate a network variable y_i to each edge $e_i \in E$. An example of a circuit and of its associated Boolean network is shown in Fig. 1.

A network variable y_i is said to be a fanout (fanin) variable of a vertex $\nu \in V$ if e_i is an edge whose tail (head) end-point is ν. We associate to each vertex ν an expression $f_\nu(y_1, y_2, \cdots, y_n)$ of its fanin edge variables. The expression f_ν, describing the functionality of the gate in ν, specifies all the fanout variables of ν in terms of the fanin variables.

2.2 Observability *don't care* sets.

By cutting an edge e_i and by considering y_i as a primary input variable, the new network N^c realizes a function $\underline{\mathcal{F}^c}(\underline{x}, y_i) : B^{n+1} \rightarrow B^m$. Given a primary input assignment $\underline{x}_0$, the variable y_i of N is not observable if the *vector* equality [6]

$$\underline{\mathcal{F}^c}(\underline{x}_0, 0) = \underline{\mathcal{F}^c}(\underline{x}_0, 1). \tag{1}$$

is satisfied. Recall that the cofactor $f|_{y_i}$ of a function f with respect to y_i is the function obtained by assuming $y_i = 1$. Similarly, $f|_{\overline{y}_i}$ is the function obtained by setting $y_i = 0$. The *vector* function:

$$\underline{ODC}_{y_i} = \underline{\mathcal{F}^c}|_{\overline{y}_i} \overline{\oplus} \underline{\mathcal{F}^c}|_{y_i} \stackrel{\text{def}}{=} \overline{\left(\frac{\partial \underline{\mathcal{F}}}{\partial y_i}\right)}. \tag{2}$$

therefore describes the observability of y_i. In particular, y_i will be observable at the k^{th} network output ($k = 1, \cdots, m$) if the k^{th} component of $\underline{ODC}_{y_i}$ is 0. The quantity $\partial \underline{\mathcal{F}} / \partial y_i$ is usually termed *Boolean difference* [6] of $\underline{\mathcal{F}}$ with respect to the (possibly internal) variable y_i. Its k^{th} component describes the network configurations that make the variable y_i observable at the k^{th} output. Therefore it is called the observability *care* set of y_i ($\underline{OC}_{y_i}$) and it corresponds to $\overline{\underline{ODC}_{y_i}}$.

Given a network, it is possible in principle to compute $\underline{ODC}_y$ for any internal variable y by flattening the network N^c and applying Eq. (2). We show here that it is possible to avoid the flattening operation on the network and compute exact and approximate versions of the functions $\underline{ODC}_y$ with a single traversal of the network.

If the *don't care* set of a vertex ν is known, then it is easy to obtain an

expression of $\underline{ODC}_y$ for any fanin variable y of ν from [1]:

$$\underline{ODC}_y = \underline{ODC}_\nu + (1,1,\cdots,1)^T\overline{\left(\frac{\partial f_\nu}{\partial y}\right)}. \quad (3)$$

The vector $(1,1,\cdots,1)^T$ is used to add $\partial f_\nu/\partial y$ to all the components of the vector $\underline{ODC}_y$.

If the network has a tree structure, then it is possible to obtain all the ODC sets by traversing backwards the network and applying iteratively Eq. (3). If a vertex has reconvergent fanout, however, the observability conditions of the vertex do not coincide with those of its fanout variables. We present here methods to compute the observability *don't care* of a vertex from those of its fanout variables.

2.3 ODC sets in presence of reconvergent fanout.

Consider a vertex ν with reconvergent fanout, and suppose that all the edges $e_1, e_2, \cdots, e_n$, whose tail-end point is ν, are cut. Let $\underline{\mathcal{F}}^c(\underline{x}, y_1, y_2, \cdots, y_n)$ be the function realized by the new network N^c, obtained by adding the variables $y_1, \cdots, y_n$, corresponding to the cut edges, to the primary inputs. Then, the observability *don't care* set of ν is described by the function

$$\underline{ODC}_\nu = \underline{\mathcal{F}}^c(\underline{x}, 0, \cdots, 0)\overline{\oplus}\underline{\mathcal{F}}^c(\underline{x}, 1, \cdots, 1) \quad (4)$$

or, equivalently, by

$$\underline{ODC}_\nu = \underline{\mathcal{F}}^c|_{\overline{y}_1,\overline{y}_2,\cdots,\overline{y}_n}\overline{\oplus}\underline{\mathcal{F}}^c|_{y_1,y_2,\cdots,y_n}. \quad (5)$$

For the sake of simplicity, we describe first the case in which $n = 2$, so that there are only two fanout variables, y_1 and y_2. We will generalize the result down below.

The observability *don't care* set is described by the function

$$\underline{ODC}_\nu = \underline{\mathcal{F}}^c|_{\overline{y}_1,\overline{y}_2}\overline{\oplus}\underline{\mathcal{F}}^c|_{y_1,y_2}. \quad (6)$$

By manipulating Eq. (6), $\underline{ODC}_\nu$ can be rewritten as

$$\underline{ODC}_\nu = \left(\underline{\mathcal{F}}^c|_{\overline{y}_1,\overline{y}_2}\overline{\oplus}\underline{\mathcal{F}}^c|_{y_1,\overline{y}_2}\right)\overline{\oplus}\left(\underline{\mathcal{F}}^c|_{y_1,\overline{y}_2}\overline{\oplus}\underline{\mathcal{F}}^c|_{y_1,y_2}\right) \quad (7)$$

where the term $\underline{\mathcal{F}}^c|_{y_1,\overline{y}_2}$ has been "added and subtracted" in Eq. (6).

From Eq.(2), the first term within parentheses is $\underline{ODC}_{y_1}|_{\overline{y}_2}$, while the second parentheses describe $\underline{ODC}_{y_2}|_{y_1}$. It then follows that

$$\underline{ODC}_\nu = \underline{ODC}_{y_1}|_{\overline{y}_2}\overline{\oplus}\underline{ODC}_{y_2}|_{y_1}. \quad (8)$$

Notice that in Eq. (5), $\underline{ODC}_\nu$ can be also rewritten as

$$\underline{ODC}_\nu = \left(\underline{\mathcal{F}}^c|_{\overline{y}_1,\overline{y}_2}\overline{\oplus}\underline{\mathcal{F}}^c|_{\overline{y}_1,y_2}\right)\overline{\oplus}\left(\underline{\mathcal{F}}^c|_{\overline{y}_1,y_2}\overline{\oplus}\underline{\mathcal{F}}^c|_{y_1,y_2}\right) = \underline{ODC}_{y_2}|_{\overline{y}_1}\overline{\oplus}\underline{ODC}_{y_1}|_{y_2} \quad (9)$$

from which we obtain the identity

$$\underline{ODC}_\nu = \underline{ODC}_{y_1}|_{\overline{y}_2}\overline{\oplus}\underline{ODC}_{y_2}|_{y_1} = \underline{ODC}_{y_1}|_{y_2}\overline{\oplus}\underline{ODC}_{y_2}|_{\overline{y}_1}. \quad (10a)$$

Similarly, for the *care* set we obtain

$$\underline{OC}_\nu = \underline{OC}_{y_1}|_{\overline{y}_2} \oplus \underline{OC}_{y_2}|_{y_1} = \underline{OC}_{y_1}|_{y_2} \oplus \underline{OC}_{y_2}|_{\overline{y}_1}. \quad (10b)$$

These algebraic manipulations can be extended to the general case of $f \geq 2$ fanout variables as follows.

Theorem 2.1 *The observability don't care set at a vertex ν is given by:*

$$\underline{ODC}_\nu = \overline{\bigoplus}_{i=1}^{f}\underline{ODC}_{y_i}|_{y_1,\cdots,y_{i-1},\overline{y}_{i+1},\cdots,\overline{y}_f}. \quad (11)$$

where $y_1, \cdots, y_f$ are the fanout variables of ν.

Proof.

It can be easily verified that for $f \geq 2$ the following identity can be derived from Eq. (5):

$$\underline{ODC}_y = \left(\underline{\mathcal{F}}^c|_{\overline{y}_1,\overline{y}_2,\cdots,\overline{y}_f}\overline{\oplus}\underline{\mathcal{F}}^c|_{y_1,\overline{y}_2,\cdots,\overline{y}_f}\right)\overline{\oplus}$$

$$\left(\underline{\mathcal{F}}^c|_{y_1,\overline{y}_2,\cdots,\overline{y}_f}\overline{\oplus}\underline{\mathcal{F}}^c|_{y_1,y_2,\cdots,\overline{y}_f}\right)\overline{\oplus}\cdots\overline{\oplus}\left(\underline{\mathcal{F}}^c|_{y_1,y_2,\cdots,y_{f-1},\overline{y}_f}\overline{\oplus}\underline{\mathcal{F}}^c|_{y_1,y_2,\cdots,y_{f-1},y_f}\right) \quad (12)$$

This can be rewritten as:

$$\underline{ODC}_\nu = \overline{\bigoplus}_{i=1}^{f}\underline{\mathcal{F}}^c|_{y_1,\cdots,y_{i-1},\overline{y}_i,\cdots,\overline{y}_f}\overline{\oplus}\underline{\mathcal{F}}^c|_{y_1,\cdots,y_i,\cdots,\overline{y}_{i+1},\cdots,\overline{y}_f} \quad (13)$$

Eq. (11) then follows by observing that each term of the sum in Eq. (13) is precisely $\underline{ODC}_{y_i}|_{y_1,\cdots,y_{i-1},\overline{y}_{i+1},\cdots,\overline{y}_f}$. □.

For the *care* set we have:

$$\underline{OC}_\nu = \bigoplus_{i=1}^{f}\underline{OC}_{y_i}|_{y_1,\cdots,y_{i-1},\overline{y}_{i+1},\cdots,\overline{y}_f}. \quad (14)$$

Similarly to the case of 2 fanout variables, changing the order in which the fanout variables are complemented results in different expressions of $\underline{ODC}_\nu$. For f fanout variables, there are $f!$ possible orderings, hence $f!$ different possible expressions for $\underline{ODC}_\nu$.

Corollary 2.1 *Let $(i_1, i_2, \cdots, i_f)$ denote a permutation of $(1, 2, \cdots, f)$. Then, the following equalities hold:*

$$\underline{ODC}_\nu = \overline{\bigoplus}_{j=1}^{f}\underline{ODC}_{y_{i_j}}|_{y_{i_1},\cdots,y_{i_{j-1}},\overline{y}_{i_{j+1}},\cdots,\overline{y}_{i_f}}. \quad (15)$$

$$\underline{OC}_\nu = \bigoplus_{j=1}^{f}\underline{OC}_{y_{i_j}}|_{y_{i_1},\cdots,y_{i_{j-1}},\overline{y}_{i_{j+1}},\cdots,\overline{y}_{i_f}}. \quad (16)$$

3 ODC sets computation.

3.1 An exact algorithm for the ODC sets computation.

Given the ODC set of a vertex ν, it is possible to compute the ODC set of all its fanin edges by means of Eq. (3). In turn, Eq. (11) allows us to compute a vertex ODC set, given those of its fanout variables. It is thus now possible to visit the Boolean network backwards from the primary outputs to its inputs and to determine the ODC sets of each vertex also in presence of reconvergent fanout.

The following algorithm performs the computation of the ODC sets. It uses the subset S of the vertices whose ODC set is known. Initially S is the set of primary output vertices with empty fanout set.

```
OBSERVABILITY(G);
S := {primary output vertices with empty fanout set};
while ( S ≠ V ) {
    select ν ∈ {V − S} such that FO(ν) ⊆ S;
    foreach fanout variable y_i of ν {
        μ = head vertex of edge e_i
        /* compute ODC_y by Eq. (3) */
        ODC_{y_i} = ODC_ν + (∂f_μ/∂y_i)
    }
    /* compute ODC_ν by Eq. (11) */
    ODC_ν = ⊕̄_{i=1}^{f} ODC_{y_i}|_{y_1,...,y_{i-1},ȳ_{i+1}...ȳ_f}
    S := S ∪ {ν};
}
```

We illustrate here its operation on the circuit shown in Fig. 1.

At the beginning, $S = \{u_4, u_5\}$, and

$$\underline{ODC}_{u_4} = \begin{pmatrix}0\\1\end{pmatrix}\ ;\underline{ODC}_{u_5} = \begin{pmatrix}1\\0\end{pmatrix}$$

The first and second component of the vectors $\underline{ODC}$ describe the observability with respect to z_1 and z_2, respectively.

First the vertices u_2, u_3 are considered. By applying Eqs. (3) and (13)

$$\underline{ODC}_{u_3} = \underline{ODC}_{y_3}|_{\overline{y}_4}\overline{\oplus}\underline{ODC}_{y_4}|_{y_3} = \begin{pmatrix}1\\y_5\end{pmatrix}\overline{\oplus}\begin{pmatrix}\overline{y}_6\\1\end{pmatrix} = \begin{pmatrix}\overline{y}_6\\y_5\end{pmatrix}$$

$$\underline{ODC}_{u_2} = \underline{ODC}_{y_6}|_{\overline{y}_5}\overline{\oplus}\underline{ODC}_{y_5}|_{y_6} = \begin{pmatrix}\overline{y}_4\\1\end{pmatrix}\overline{\oplus}\begin{pmatrix}1\\y_3\end{pmatrix} = \begin{pmatrix}\overline{y}_4\\y_3\end{pmatrix}$$

Then $S = \{u_2, u_3, u_4, u_5\}$ and u_1 is selected. Its fanout variables are y_1, y_2, and according to Eq. (3):

$$\underline{ODC}_{y_1} = \begin{pmatrix}\overline{y}_6 + x_1\\y_5 + x_1\end{pmatrix} = \begin{pmatrix}\overline{y}_2\overline{x}_4 + x_1\\y_2 + x_4 + x_1\end{pmatrix}$$

[1] Notice that Eq. (3) may contain internal network variables. These, however, may be resolved by back-substitution to obtain an expression of primary input variables only, so that there is no real contrast with the definition (2).

$$\underline{ODC}_{y_2} = \begin{pmatrix} \overline{y}_4 + x_4 \\ y_3 + x_4 \end{pmatrix} = \begin{pmatrix} \overline{y}_1\overline{x}_1 + x_4 \\ y_1 + x_1 + x_4 \end{pmatrix}$$

so that, using Eq. (13),

$$\underline{ODC}_{u_1} = \underline{ODC}_{y_1}|_{\overline{y}_2} \overline{\oplus} \underline{ODC}_{y_2}|_{y_1} = \begin{pmatrix} \overline{x}_4 + x_1 \\ x_4 + x_1 \end{pmatrix} \overline{\oplus} \begin{pmatrix} x_4 \\ 1 \end{pmatrix} = \begin{pmatrix} x_1 x_4 \\ x_1 + x_4 \end{pmatrix}$$

which is the correct result.

Finally the algorithm computes the ODC sets of the primary inputs, that may be used as external ODC sets for the minimization of the logic feeding the circuit.

The product of all the components of $\underline{ODC}_\nu$ gives the conditions for which the gate output at vertex ν is not observable at *any* output. In the case of the gate in u_1, the product of the components of $\underline{ODC}_{u_1}$ yields $x_1 x_4$.

3.2 Computation of approximate ODC sets.

It is of practical interest to consider the case in which the ODC sets are approximated by subsets, because of their size. Note that excess approximations of the ODC sets are of no practical value for logic minimization. Therefore it is useful to derive ODC subsets at a vertex from the ODC sets of its fanout vertices. Unfortunately, Eq. (7) and Eq. (11) may not yield an ODC subset from subsets of the actual ODC sets of each edge variable.

For example, in the circuit of Fig. 1, if we assume

$$\underline{ODC}_{y_1} = \begin{pmatrix} x_1 \\ x_1 \end{pmatrix}$$

(actually a subset of the true $\underline{ODC}_{y_1}$), we find for $\underline{ODC}_{u_1}$ the estimate:

$$\underline{ODC}_{u_1} = \begin{pmatrix} x_1 x_4 + \overline{x}_1 \overline{x}_4 \\ x_1 \end{pmatrix}$$

which is no longer a subset of the true $\underline{ODC}_{u_1}$.

Two different approaches for the computation of ODC subsets are fully reported in [11]. Their results are briefly summarized in this section. In particular two formulae can replace Eq. (11) in the previous algorithm. The first formula computes *both* subsets of the actual *care* and *don't care* sets of a vertex from those of its fanout variables. The second one, simpler but less accurate, computes ODC subsets only.

We give here the formulae for computing ODC_ν in the case of two fanout variables. A straightforward generalization to larger fanout can be achieved by viewing a vertex with outdegree (fanout) $f > 2$ as the root of a binary tree [11]. We refer the reader to [11] for the details.

Let y_1 and y_2 denote the fanout variables of a vertex ν. Assume that only subsets $\underline{\widetilde{ODC}}_{y_i}$, $\underline{\widetilde{OC}}_{y_i}$ of $\underline{ODC}_{y_i}$, $\underline{OC}_{y_i}$ are available. Then, $\underline{\widetilde{ODC}}_\nu$ and $\underline{\widetilde{OC}}_\nu$ can be computed by:

$$\underline{\widetilde{ODC}}_\nu = \underline{\widetilde{ODC}}_{y_1}|_{\overline{y}_2} \underline{\widetilde{ODC}}_{y_2}|_{y_1} + \underline{\widetilde{ODC}}_{y_1}|_{y_2} \underline{\widetilde{ODC}}_{y_2}|_{\overline{y}_1} + \\ + \underline{\widetilde{OC}}_{y_1}|_{\overline{y}_2} \underline{\widetilde{OC}}_{y_2}|_{y_1} + \underline{\widetilde{OC}}_{y_1}|_{y_2} \underline{\widetilde{OC}}_{y_2}|_{\overline{y}_1} \qquad (17a)$$

and by

$$\underline{\widetilde{OC}}_\nu = \underline{\widetilde{OC}}_{y_1}|_{\overline{y}_2} \underline{\widetilde{ODC}}_{y_2}|_{y_1} + \underline{\widetilde{OC}}_{y_1}|_{y_2} \underline{\widetilde{ODC}}_{y_2}|_{\overline{y}_1} + \\ + \underline{\widetilde{ODC}}_{y_1}|_{\overline{y}_2} \underline{\widetilde{OC}}_{y_2}|_{y_1} + \underline{\widetilde{ODC}}_{y_1}|_{y_2} \underline{\widetilde{OC}}_{y_1}|_{\overline{y}_1} \qquad (17b)$$

They are subsets of the exact $\underline{ODC}_\nu$ and $\underline{OC}_\nu$, respectively.

If only subset of *don't care* conditions $\underline{\widetilde{ODC}}_{y_i}$ are available, then $\underline{\widetilde{ODC}}_\nu$ can be computed from:

$$\underline{\widetilde{ODC}}_\nu = \underline{\widetilde{ODC}}_{y_1}|_{\overline{y}_2} \underline{\widetilde{ODC}}_{y_2}|_{y_1} + \underline{\widetilde{ODC}}_{y_1}|_{y_2} \underline{\widetilde{ODC}}_{y_2}|_{\overline{y}_1} \qquad (18)$$

Note that this formula allows us to drop elements from the ODC sets at our convenience. In particular, if we choose a subset $\underline{\widetilde{ODC}}_{y_1}$ of $\underline{ODC}_{y_1}$ that does not depend on y_2, then

$$\underline{\widetilde{ODC}}_\nu = \underline{\widetilde{ODC}}_{y_1} \underline{\widetilde{ODC}}_{y_2} \subseteq \underline{\widetilde{ODC}}_{y_1} \left(\underline{\widetilde{ODC}}_{y_2}|_{\overline{y}_1} + \underline{\widetilde{ODC}}_{y_2}|_{y_1} \right) \qquad (19)$$

is a subset of $\underline{ODC}_\nu$.

In this case no cofactoring operations are needed, differently from Eq. (11) or (18). A much faster computation is therefore expected if the independence of $\underline{\widetilde{ODC}}_{y_1}$ from y_2 can be guaranteed at each point of reconvergent fanout. This can be accomplished by eliminating mutual dependencies at the point of reconvergence, as shown with the following example referring to Fig. 2.

Gate G_6 is the point of reconvergence for the output signals of G_1 (through the variables z_1 and z_2) and of G_2 (through z_1 and z_3). Thus, $ODC_{z_1} = \overline{z_2 z_3}$. To eliminate the dependencies of ODC_{y_1}, ODC_{y_4} from ODC_{y_2}, ODC_{y_3} the observability of z_2 is approximated by the portion independent from z_1 ,i.e. $\widetilde{ODC}_{z_2} = ODC_{z_2}|_{z_1} ODC_{z_2}|_{\overline{z}_1} = \overline{z}_3$. Similarly, $\widetilde{ODC}_{z_3} = \overline{z}_2$. The observability of y_1 and y_4 is now made independent from y_2 and y_3, respectively, so that Eq. (19) can be used to compute $\widetilde{ODC}_{G_1} = \widetilde{ODC}_{y_1} \widetilde{ODC}_{y_2} = (\overline{a} + \overline{z}_3)(\overline{y}_3 + \overline{z_1 z_3})$ and $\widetilde{ODC}_{G_2} = \widetilde{ODC}_{y_3} \widetilde{ODC}_{y_4} = (\overline{f} + \overline{z}_1)(\overline{y}_2 + \overline{z_1 z_3})$.

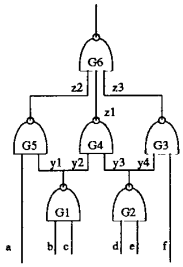

Figure 2: Example NAND circuit for the application of Eq. 19

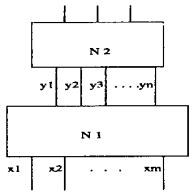

Figure 3: Cascaded subnetworks

4 ODC sets and Boolean Relations.

Most techniques based on *don't care* sets aim at the simplification of Boolean functions at single vertices (*i.e.* single-output functions) of a Boolean network. However, more powerful techniques have been recently proposed, that attempt a simultaneous minimization [5] of two or more vertices. The problem can be approached by observing that when a Boolean network N_1 feeds another network N_2 (see Fig. 2), simultaneous changes of several outputs of N_1 can occur without affecting the outputs of N_2. Note that observability *don't care* sets capture only the possible changes at *single* outputs of N_1.

Two output patterns of N_1 are said to be *equivalent* if they yield the same output at N_2. Given a "reference" output pattern $\underline{y}_r = (y_{1r}, \cdots, y_{mr})$, its *equivalence class* is the set of equivalent patterns that are possible outputs of N_1. The example shown in Fig. 4 is borrowed from [5]. An algorithm that minimizes a 2-level representation of N_1 using its outputs equivalence classes is presented in [10].

A problem associated to this minimization procedure is determining the output equivalence classes, *i.e.* finding the *Boolean Relation* that describes the degrees of freedom for optimizing N_1. Indeed, flattening N_2 so as to re-express its outputs in terms of $y_1, \cdots, y_n$ is inefficient if we wish to change dynamically the boundary between N_1 and N_2, and possibly unfeasible.

We show here a method for determining the equivalence classes of N_1 that requires only the knowledge of the individual ODC sets of the variables y_i. We present this result referring first to the example circuit of Fig. 4. The observability of the variables y_0, y_1, y_2 (as found, for example, by the OBSERVABILITY algorithm) are given by

$$\underline{ODC}_{y_0} = \begin{pmatrix} \overline{y}_2 + y_1 \\ \overline{y}_1 + y_2 \end{pmatrix}; \quad \underline{ODC}_{y_1} = \begin{pmatrix} \overline{y}_2 + y_0 \\ \overline{y}_0 + y_2 \end{pmatrix}; \quad \underline{ODC}_{y_2} = \begin{pmatrix} \overline{y}_1 \overline{y}_0 \\ y_1 y_0 \end{pmatrix}$$

The equivalence class of the input pattern $y_1, y_2, y_3 = 0, 0, 0$ is the set of patterns y_0, y_1, y_2 that satisfies

$$\underline{\mathcal{F}}(y_2, y_1, y_0) = \underline{\mathcal{F}}(0, 0, 0)$$

and is therefore described by the function

$$\underline{EQV}_{0,0,0} = \underline{\mathcal{F}}(y_2, y_1, y_0) \overline{\oplus} \underline{\mathcal{F}}(0, 0, 0)$$

By "adding and subtracting" the terms $\underline{\mathcal{F}}(y_2, y_1, 0)$, $\underline{\mathcal{F}}(y_2, 0, 0)$ the function $\underline{EQV}_{0,0,0}$ can be expressed by:

$$\underline{EQV}_{0,0,0} = \left(\underline{\mathcal{F}}(y_2, y_1, y_0) \overline{\oplus} \underline{\mathcal{F}}(y_2, y_1, 0) \right) \overline{\oplus} \left(\underline{\mathcal{F}}(y_2, y_1, 0) \overline{\oplus} \underline{\mathcal{F}}(y_2, 0, 0) \right)$$

$$\overline{\oplus} \left(\underline{\mathcal{F}}(y_2, 0, 0) \overline{\oplus} \underline{\mathcal{F}}(0, 0, 0) \right) =$$

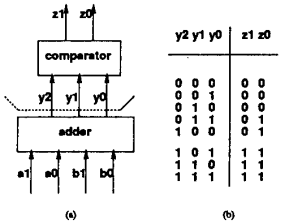

Figure 4: Cascaded adder - comparator: (a) schematic; (b) comparator truth table; (c) adder output equivalence classes

$$\left(\overline{y}_0 + \underline{ODC}_{y_0}\right)\overline{\oplus}\left(\overline{y}_1 + \underline{ODC}_{y_1}|_{\overline{y}_0}\right)\overline{\oplus}\left(\overline{y}_2 + \underline{ODC}_{y_2}|_{\overline{y}_1\overline{y}_0}\right) =$$
$$\left(\begin{matrix}\overline{y}_0 + y_1 + \overline{y}_2 \\ \overline{y}_0 + \overline{y}_1 + y_2\end{matrix}\right)\overline{\oplus}\left(\begin{matrix}\overline{y}_1 + \overline{y}_2 \\ 1\end{matrix}\right)\overline{\oplus}\left(\begin{matrix}1 \\ \overline{y}_2\end{matrix}\right) = \left(\begin{matrix}\overline{y}_2 + \overline{y}_1\overline{y}_0 \\ \overline{y}_2(\overline{y}_1 + \overline{y}_0)\end{matrix}\right)$$

where we used the identity [6]

$$\underline{\mathcal{F}}(\underline{x}, y)\overline{\oplus}\underline{\mathcal{F}}(\underline{x}, 0) = \overline{\left(y\frac{\partial \underline{\mathcal{F}}}{\partial y}\right)} = \overline{y} + \underline{ODC}_y. \tag{20}$$

Each component of $\underline{EQV}_{0,0,0}$ describes the equivalence class of the refence pattern 0, 0, 0 with respect to an output. We are interested in the equivalence class with respect to *all outputs*, described by the product of all the components of $\underline{EQV}_{0,0,0}$, that in this example is: $\overline{y}_2\overline{y}_0 + \overline{y}_2\overline{y}_1$.

For this example, we have thus been able to re-express the equivalence class of the adder outputs in terms of the individual ODC sets of its output variables.

This result can be generalized as follows. Consider a cutset of f edges of a Boolean network, as shown in Fig. 4. Let $\underline{y} = (y_1, y_2, \cdots, y_f)$ denote the variables associated to the cutset and $\underline{\mathcal{F}}^c$ the function realized by N_2.

Theorem 4.1 *The equivalence class of any given configuration* $\underline{y}_r = (y_{1r}, y_{2r}, \cdots, y_{fr})$ *is:*

$$\underline{EQV}_{y_{1r}, y_{2r}, \cdots, y_{fr}} = \overline{\bigoplus}_{i=1}^{f} y_i\overline{\oplus}y_{ir} + ODC_{y_i}|_{y_{1r}, \cdots, y_{i-1r}}. \tag{21}$$

Proof. The equivalence class of any given configuration $\underline{y}_r = (y_{1r}, y_{2r}, \cdots, y_{fr})$ is the set of configurations that satisfies

$$\underline{EQV}_{\underline{y}_r} = \underline{\mathcal{F}}^c(\underline{y})\overline{\oplus}\underline{\mathcal{F}}^c(\underline{y}_r). \tag{22}$$

The following identity holds:

$$\underline{EQV}_{y_{1r}, y_{2r}, \cdots, y_{fr}} = \left(\underline{\mathcal{F}}(y_1, y_2, \cdots, y_f)\overline{\oplus}\underline{\mathcal{F}}(y_{1r}, y_2, \cdots, y_f)\right)\overline{\oplus}$$
$$\left(\underline{\mathcal{F}}(y_{1r}, y_2, \cdots, y_f)\overline{\oplus}\underline{\mathcal{F}}(y_{1r}, y_{2r}, \cdots, y_f)\right) \tag{23}$$
$$\overline{\oplus}\cdots\overline{\oplus}\left(\underline{\mathcal{F}}(y_{1r}, y_{2r}, \cdots, y_{f-1r}, y_f)\overline{\oplus}\underline{\mathcal{F}}(y_{1r}, y_{2r}, \cdots, y_{fr})\right) =$$
$$= \overline{\bigoplus}_{i=1}^{f}\left(\underline{\mathcal{F}}(y_{1r}, y_{2r}, \cdots, y_{i-1r}, y_i, \cdots, y_f)\overline{\oplus}\underline{\mathcal{F}}(y_{1r}, y_{2r}, \cdots, y_{ir}, y_{i+1}, \cdots, y_f)\right) \tag{24}$$

By using the identity (20), it can be verified that the i^{th} term of the sum (24) is

$$y_i\overline{\oplus}y_{ir} + ODC_{y_i}|_{y_{1r}, \cdots, y_{i-1r}} \tag{25}$$

so that

$$\underline{EQV}_{y_{1r}, y_{2r}, \cdots, y_{fr}} = \overline{\bigoplus}_{i=1}^{f} y_i\overline{\oplus}y_{ir} + ODC_{y_i}|_{y_{1r}, \cdots, y_{i-1r}}\square \tag{26}$$

With this theorem, we can express each equivalence class for the outputs of N_1 in terms of its individual output ODC sets.

Circuit	Exact formula (13) CPU time	Exact formula (13) # literals	Simplified formula (19) CPU time	Simplified formula (19) # literals
C17	0	2	0	2
C432	105	1990	39	890
C499	12	75	6	78
C880	11	70	6	60
C1908	*	*	123	202
C6288	*	*	3956	465
apex6	40	38	28	43
apex7	26	9	18	11
CM138	0	2	0	2

Table 1: CPU time and average number of literals for the ODC sets of some benchmark circuits. Symbol * means that SPY ran out of memory.

5 Implementation and Experimental Results.

The algorithm OBSERVABILITY, that uses both exact and approximate formulae, has been implemented in program SPY. SPY is written in C and consists of about 3000 lines of code. We have successfully tested SPY on several benchmark circuits, including some critical ones proposed by Brglez and Fujiwara, as reported in Table 1. The number of literals refers to the average size of unminimized expressions of the ODC sets, as generated by the algorithm. Final literal counts are typically smaller, but we report here on the size of the uncompacted ODC sets because they are the direct implementation of the formulae shown in the paper. The run times are in seconds on a DEC 3100 workstation.

6 Summary.

New formulae and algorithms for the computation of exact and approximate observability *don't care* sets have been proposed. The algorithms are efficient because they use local information, *i.e.* the computation of the ODC sets for a vertex requires only the knowledge of the *don't care* sets at the adjacent vertices. The algorithms have been implemented in program SPY an tested on a set of benchmark examples.

Expressions have also been then derived that link the observability *don't care* sets of the individual vertices of a network to the equivalence classes of a Boolean relation, thus filling a gap in the theory of uncompletely specified functions. Eventually it has been shown how these equivalence classes can be efficiently derived from the computed ODC sets.

7 Acknowledgements.

This research was supported in part by a fellowship of the Rotary Foundation and by AT & T and DEC, jointly with NSF, under a PYI award program. We also acknowledge support from NSF-ARPA under contract # MIP 8719546.

References

[1] K. A. Bartlett, R. K. Brayton, G. D. Hachtel, R. M. Jacoby, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang, " Multilevel Logic Minimization Using Implicit Don't Cares", *IEEE Transactions on CAD/ICAS*, vol. CAD-7, No. 6, pp. 723-739, June 1988.

[2] G. D. Hachtel and M. R. Lightner, " Top-Down Synthesis of Multilevel Logic Networks" *Proc. ICCAD 1987*, pp. 316-319, S. Clara, Nov. 1987.

[3] S. Muroga, Y.Kambayashi, H.Lai and J.Culliney, "The Transduction Method - Design of Logic Networks Based on Permissible Functions ", *IEEE Trans. Comp.*, vol. 38, No. 10, pp. 1404-1424, 1989.

[4] D. Bostick, G. D. Hachtel, R. M. Jacoby, M. R. Lightner, P. Moceyunas, C. R. Morrison, and D. Ravenscroft, " The Boulder Optimal Logic Design System", *Proc. ICCAD 1987*, pp. 62-65, S. Clara, Nov. 1987.

[5] R. K. Brayton and F. Somenzi, "Boolean Relations and the Incomplete Specification of Logic Networks", *Proc. VLSI '89*, pp. 231-240, Munich, August 1989.

[6] H. Fujiwara, *Logic Design and Design for Testability*, MIT Press, Cambridge, 1985.

[7] A. C. L. Chang, I. S. Reed, A. V. Banes, "Path Sensitization, Partial Boolean Difference and Automated Fault Diagnosis", *IEEE Transactions on Computers*, C-21, pp. 189 - 194, Feb. 1972.

[8] G. D. Hachtel, R. M. Jacoby, P. H. Moceyunas, " On Computing and Approximating the Observability Don't Care Set ", *Proceedings on the International Workshop on Logic Synthesis*, Research Triangle Park, May 1989.

[9] R. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, A. Wang , "MIS: A Multiple-Level Logic Optimization System", *IEEE Transactions on CAD/ICAS*, Vol. CAD-6, No. 6, pp. 1062-1081, November 1987.

[10] R. K. Brayton, F. Somenzi, " An Exact Minimizer for Boolean Relations", *Proc. ICCAD 1989*, pp. 316 - 319, S. Clara, Nov. 1989.

[11] M.Damiani and G. De Micheli, "Efficient Computation of the Exact and Approximate Observability Don't Care Sets in Multiple-Level Logic Synthesis" *Proceedings of the IFIP working conference on Logic and Architecture Synthesis*, Paris, May 1990, and Stanford *CSL Report* CSL- TR-90-424.