Derivation of *Don't care* Conditions by Perturbation Analysis of Combinational Multiple-Level Logic Circuits

Maurizio Damiani Giovanni De Micheli

Center for Integrated Systems
Stanford University

Abstract

In this paper we present a framework for the derivation of *don't care* conditions by perturbation analysis of combinational multiple-level digital circuits.

The contribution of the paper is two-fold. First, different approximations of observability *don't care* sets are compared quantitatively. Second, the perturbation analysis is used to derive new compatible observability *don't care* sets, that are larger than those previously derived.

1 Introduction

Multiple-level logic optimization strategies [9, 10, 12, 4] are based on circuit transformations that preserve the circuit behavior and improve its quality. Different flavors of circuit transformations have been proposed. Optimization algorithms based on Boolean transformations, such as those used in [9] [4] and [12], have shown to be very effective in reducing the circuit area and delay as well as improving its testability properties.

Don't care conditions play a central role in the specification and optimization of logic circuits. Indeed, they represent the degrees of freedom of transforming a network into an equivalent one. The computation of exact and approximate *don't care* sets has been object of extensive investigation. Several algorithms have been proposed [7, 10, 13, 16, 12, 15] for the calculation of observability *don't care* sets by backward network traversal. It has been observed that the optimization of a gate changes the *don't care* of other gates in the network. For this reason, the concept of *compatible don't care* sets has been introduced [12, 15].

The contribution of this paper is two-fold. First we review the analysis of *don't care* conditions in combinational multiple-level circuits, and present a quantitative comparative analysis of approximate *don't care* set. Second, we analyze the problem of computing compatible and maximally compatible *don't care* sets, and show in particular that it is indeed possible to compute compatible *don't care* sets that are larger than those previously believed maximal [15].

2 Basic concepts and definitions

We consider in this paper combinational multiple-level logic circuits. We assume that these circuits consist of an interconnection of multiple-input single-output combinational logic gates. Fanout points are modeled by single-input multiple-output *copy* gates, as shown in Fig. (1).

We model these circuits by **Boolean networks**. A Boolean network is described by a directed acyclic graph $G = (V, E)$. The elements of the vertex set V correspond to logic gates (including *copy* gates), while edges

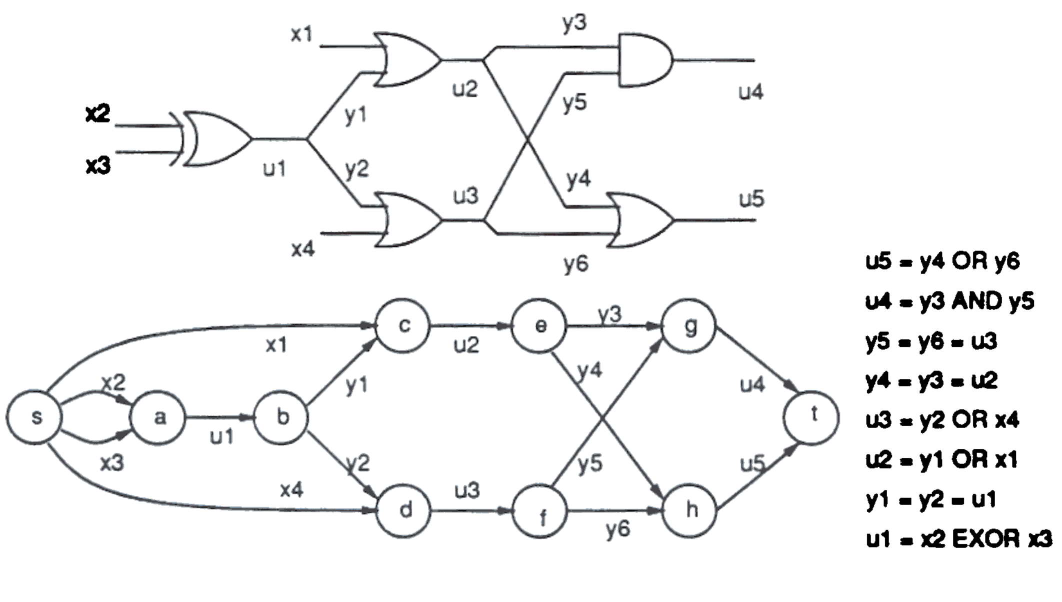


Figure 1: Combinational Boolean Network and corresponding graph

correspond to interconnections. In our network model, each variable is associated to an edge, and it is denoted by a string (e.g. x, $sample$). The edge is indicated by a subscript (e.g. e_x, e_{sample}). A variable x is said to be a fanout (fanin) variable of a vertex ν if e_x is an edge whose tail (head) end-point is ν. The head and tail vertices of an edge e are denoted by $head(e)$ and $tail(e)$, respectively. The set of fanout (fanin) edges of a vertex ν is indicated by $FO(\nu)$ $(FI(\nu))$. To represent primary input and output variables, a *source* and a *sink* vertex are added. An edge e_x joins the source to a vertex ν if x is a primary input variable feeding the gate corresponding to ν. Similarly, an edge e_z joins a vertex μ to the sink vertex if z is a primary output variable computed by the gate corresponding to ν. Note in particular that $FI(source) = FO(sink) = \phi$. The number of primary inputs and outputs are $n_i = |FO(source)|$ and $n_o = |FI(sink)|$, respectively. The number of variables is $n_e = |E|$.

Vertices represent the computational elements of the network. In particular, each fanout variable y of a vertex ν is associated to a function f_y of the fanin variables of ν. For *copy* gates, the function reduces to identity.

Example 1. A combinational Boolean network and its graph are shown in Fig. (1).

3 *Don't care* conditions in combinational networks

It is customary to view a Boolean network as realizing a function $\underline{\mathcal{F}} : B^{n_e} \rightarrow B^{n_e}$ of the network variables, where B denotes the Boolean set $\{0, 1\}$. $\underline{\mathcal{F}}$ therefore associates a n_o-dimensional Boolean vector to each point of B^{n_e}. Similarly, each expression f_y is thought of representing a scalar Boolean function over the same domain.

The embedding of the network in a larger digital system is modeled by introducing *external don't care* conditions. Two types of external don't cares are typically considered: *controllability* and *observability don't cares*, respectively. In [10] they are defined as primary input assignments that are impossible or such that a primary output is not observed, respectively.

It is worth noting that in practice observability *don't care* conditions are available in terms of primary output variables. During logic optimization, the mapping of these conditions in terms of primary inputs is subject to change. The definition provided in [10] is therefore not suited for these cases.

We define as observability *don't care* condition of an output any assignment of the network variables, including in particular in particular primary inputs and outputs, such that the output is not observed.

Controllability *don't cares* are here indicated by an expression CDC_{ext} of the primary input variables, while observability *don't cares* are indicated by a n_o-dimensional vector $\underline{ODC}_{ext}$. The i^{th} component ($i = 1, \cdots, n_o$) of $\underline{ODC}_{ext}$ is an expression in terms of network variables, representing the set of conditions for which the i^{th} output is not observed.

The set of internally impossible variable assignments is usually termed *satisfiability don't care set* and is denoted by $SDC_{\mathcal{F}}$:

$$SDC_{\mathcal{F}} = \sum_{e_y \in E} y \oplus f_y \tag{1}$$

We define an auxiliary n_o-dimensional vector $\underline{1} = (1, 1, \cdots, 1)$. For ease of treatment, satisfiability and controllability *don't care* sets are represented in the sequel by vectors $\underline{SDC} = SDC\underline{1}$ and $\underline{CDC}_{ext} = CDC_{ext}\underline{1}$.

4 *Don't care* conditions for single-vertex optimization

Logic optimization algorithms refine an original network iteratively by replacing local functions f_y by others that improve some property of the network, typically area occupation or timing performance. This is accomplished by first determining the set of functions g_y that can replace a given function f_y.

The change of a function f_y into g_y affects the network functionality by *perturbing* the value of the variable y, corresponding to those assignments for which $f_y \neq g_y$. This effect can be modeled by means of the following definitions.

Definition 4.1 *Let y denote the output variable of a logic gate in a Boolean Network N. We call* **perturbed network** *N_y the network obtained by replacing the function f_y with the function $f_y \oplus \delta$, where δ is an added primary input.*

The functionality of the perturbed network N_y is described by a function $\underline{\mathcal{F}}_y(\delta) : B^{n_i+1} \rightarrow B^{n_o}$. In particular $\underline{\mathcal{F}}_y(0) = \underline{\mathcal{F}}$ and the functionality of any network N' obtained by replacing f_y with an arbitrary function g_y is described by $\underline{\mathcal{F}}_y(f_y \oplus g_y)$.

The perturbed gate contributes to the satisfiability *don't care set* of the perturbed network with a term $SDC_\delta = y \oplus (f_y \oplus \delta)$, while the remainder of the network contributes with

$$SDC_y = \sum_{e_z \in E - \{e_y\}} z \oplus f_z \ , \tag{2}$$

so that the satisfiability *don't care* of the network is expressed by $SDC_{\mathcal{F}_y} = SDC_y + SDC_\delta$.

Definition 4.2 *Given a cut network N_y, the quantity*

$$\underline{ODC}_y = \underline{\mathcal{F}}_y(\delta = 1)\overline{\oplus}\underline{\mathcal{F}}_y(\delta = 0) \tag{3}$$

is termed **observability don't care vector** *for the variable y. Its i^{th} component ($i = 1, \quad , n_o$) describes the set of network assignments for which y does not affect the i^{th} network output.*

The complement of the observability *don't care* vector is the ordinary *Boolean difference* $\partial\underline{\mathcal{F}}/\partial y$ of $\underline{\mathcal{F}}$ with respect to y.

Definition 4.3 *Consider a network N', obtained from N by replacing a function f_y with g_y. N' is functionally equivalent to N if the vector equality*

$$\underline{\mathcal{F}}_y(f_y \oplus g_y) = \underline{\mathcal{F}}_y(0) \tag{4}$$

is satisfied for all the observable components of $\underline{\mathcal{F}}_y$, for all the feasible variable assignments of N_y.

We are interested in determining the set of functions g_y that are equivalent to f_y, and then choose the one of minimum cost. To this end, we introduce the function

$$\underline{EQV}(\delta) \overset{\text{def}}{=} \underline{\mathcal{F}}_y(\delta)\overline{\oplus}\underline{\mathcal{F}}_y(0)\ . \tag{5}$$

A function g_y is equivalent to f_y as long as the introduced perturbation is such that

$$\underline{EQV} + \underline{ODC}_{ext} + \underline{CDC}_{ext} + \underline{SDC}_{\mathcal{F}_y} = 1\ . \tag{6}$$

When $\delta = 0$, from Eq. 5, $\underline{EQV}(0)$ is identically $\underline{1}$, and therefore Eq. 6 is a tautology. For $\delta = 1$, Eq. 6 holds as long as

$$\underline{EQV}(1) + \underline{ODC}_{ext} + \underline{CDC}_{ext} + SDC_y + (y \oplus \overline{f}_y)\underline{1} = \underline{1} \tag{7}$$

By observing that $\underline{EQV}(1) = \underline{ODC}_y$, it follows that the perturbation δ must satisfy the inequality

$$\delta\underline{1} \subseteq \underline{ODC}_y + \underline{ODC}_{ext} + \underline{SDC}_y + \underline{CDC}_{ext} + (y \oplus \overline{f}_y)\underline{1} \overset{\text{def}}{=} \underline{DC}_y \tag{8}$$

The expression 8 for the *don't care* set of y differs from the ones usually considered and presented, for example, in [10], because of the last term $y \oplus \overline{f}_y$.

5 *Don't care* computation

Computing the satisfiability *don't care* set is straight-forward. In contrast, the computation of observability *don't care* sets has been recognized as a problem of its own. Several methods for approximating the observability *don't care* sets have been proposed in the past [9, 3, 8, 16], by using backward network traversal. Except for the *chain rule* [7], all these methods either abandon the network traversal approach in presence of reconvergent fanout or resort to some approximations.

We present first an exact method for computing the ODCs. We then compare the method to other previous approximations to the ODC computation. We comment then on another new ODC approximation.

5.1 Exact ODC computation

In [13] the authors presented a method for computing exact observability *don't cares*. We summarize it briefly here.

The computation is based on a backward network traversal and on the following steps:

1) In presence of a logic gate, with output variable y and input variables $y_1, \cdots, y_n$, we compute the observability of the variables y_i by

$$\underline{ODC}_{y_i} = \underline{ODC}_y + \overline{\left(\frac{\partial f_y}{\partial y_i}\right)}\underline{1}\ ; \tag{9}$$

2) In presence of a fanout point, with output variables y_1 and y_2, and input variable y, the observability *don't care* of y is computed according to the rule

$$\underline{ODC}_y = \underline{ODC}_{y_1}|_{y_2=0}\overline{\oplus}\underline{ODC}_{y_2}|_{y_1=1}\ . \tag{10}$$

or

$$\underline{ODC}_y = \underline{ODC}_{y_1}|_{y_2=1}\overline{\oplus}\underline{ODC}_{y_2}|_{y_1=0}\ .$$

These equations can be rewritten in a more compact form as:

$$\underline{ODC}_y = \underline{ODC}_{y_1}\overline{\oplus}\underline{ODC}_{y_2}|_{y_1=\overline{y}}\ .$$

Table(1) shows the CPU time and average number of literals spent for the exact and approximate computation of unminimized ODC sets.

Circuit	Exact formula (10)		Simplified formula (20)	
	CPU time	# literals	CPU time	# literals
C17	0	2	0	2
C432	105	1990	68	2017
C499	12	75	6	88
C880	11	70	6	60
C1908	*	*	175	273
C6288	*	*	4295	689
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. (* denotes out of memory.)

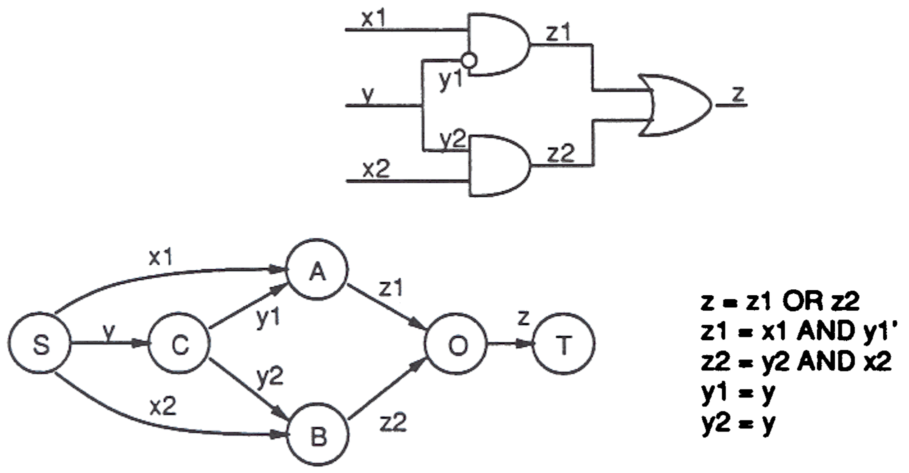


Figure 2: Multiplexer model of a circuit with reconvergent fanout.

5.2 Comparative analysis of different ODC approximations

We use here Eq. 10 first for examining the quality of previous approximations, and then for proposing a novel solution.

For the sake of simplicity, let us restrict our attention to single-output networks (so that the *ODC* vectors have only one component) and to a *copy* vertex with only two output variables, as shown in Fig. (2).

We define the auxiliary quantities

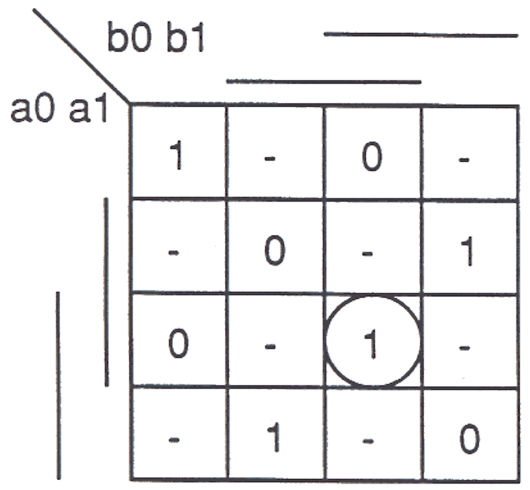
$$a_0 = ODC_{y_1}|_{y=0}, \quad a_1 = ODC_{y_1}|_{y=1}, \quad b_0 = ODC_{y_2}|_{y=0}, \quad b_1 = ODC_{y_2}|_{y=1}$$

so that

$$ODC_{y_1} = a_0\overline{y} + a_1 y; \quad ODC_{y_2} = b_0\overline{y} + b_1 y.$$

By substituting these expressions in Eq. 10 and 11 the following identity must hold:

$$ODC_y = (a_0\overline{y} + a_1 y)\overline{\oplus}(b_0 y + b_1\overline{y}) = (a_0 y + a_1\overline{y})\overline{\oplus}(b_0\overline{y} + b_1 y)$$



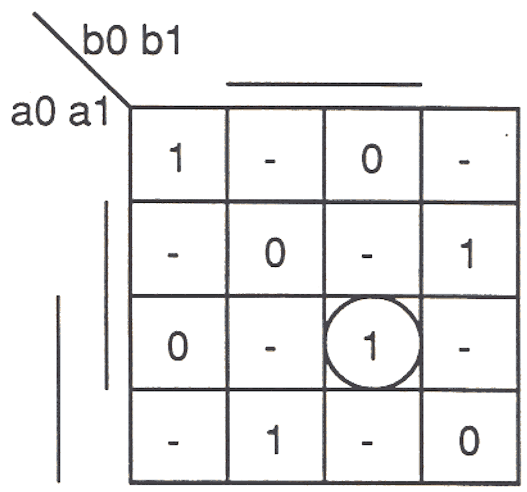


Figure 3: Map of ODC_y in terms of the variables a_0, a_1, b_0, b_1, for $y = 0$ and $y = 1$ respectively. Circles represent the approximation given by Eq. (16)

Any assignment to a_0, a_1, b_0, b_1 violating identity 14 must therefore be contained in the *satisfiability don't care* of the network. Namely:

$$a_0 \oplus a_1 \oplus b_0 \oplus b_1 \subseteq SDC_y \ . \tag{15}$$

The Karnaugh map of ODC_y in terms of a_0, a_1, b_0, b_1 and y is shown in Fig. (3), where 1's denote the observability *don't care* , 0's the observability *care*, and – denotes the impossible assignments given by Eq. 15, *i.e.* the satisfiability *don't cares.*

All "local" approaches attempt approximate ODC_y by some other function of a_0, a_1, b_0, b_1 and, possibly, y. We denote an approximation to ODC_y by $\widetilde{ODC}_y$. The quality of any such approximation can thus be measured by the number of covered 1's in the Karnaugh map.

The most "conservative" simplification is taken in the program MIS-II, as reported in [9]. There, ODC_y is computed by assuming that the variables z_1, z_2 are fully observable. The observability $ODC^{MIS}_{y_i}$ of each variable y_1, y_2 is thus obtained from Eq. 9 by neglecting the first term of the sum. In our case, each $\widetilde{ODC}_{y_i}$ is independent from any other variable y_j, *i.e.* $\widetilde{ODC}_{y_1} \subseteq a_0a_1$ and $\widetilde{ODC}_{y_2} \subseteq b_0b_1$. $\widetilde{ODC}_y$ is computed as the product of $\widetilde{ODC}_{y_1}$ and $\widetilde{ODC}_{y_2}$; therefore, $\widetilde{ODC}_y^{MIS} \subseteq a_0a_1b_0b_1$.

In [16], the following approximation is proposed:

$$\widetilde{ODC}_{y_1} = C_{y_2}(ODC_{y_1}) = a_0a_1 \ ;$$
$$\widetilde{ODC}_{y_2} = C_{y_1}(ODC_{y_2}) = b_0b_1 \ ;$$
$$\widetilde{ODC}_y = \widetilde{ODC}_{y_1}\widetilde{ODC}_{y_2} = a_0a_1b_0b_1 \tag{16}$$

According to this approximation, first the portions of ODC_{y_i} independent from y_j, $j \neq i$ are computed, *i.e.* the observability *don't care* vectors of the fanout variables are *decoupled.* It is then possible to form their product to obtain a correct approximation. Note that, with this approach, it is necessary to have explicitly ODC_{y_i} in terms of all variables y_j. These approximations capture at most the circled minterms in the map of Fig. (3), *i.e.* only two 1's out of the 8 possible.

Muroga proposes in [3] an apparently better approximation. It consists in computing only $\widetilde{ODC}_{y_2} = C_{y_1}(ODC_{y_2}) = b_0b_1$, and to compute $\widetilde{ODC}_y$ according to

$$\widetilde{ODC}_y = ODC_{y_1}\widetilde{ODC}_{y_2} = (a_0\overline{y} + a_1 y)b_0b_1 \ . \tag{17}$$

[t]

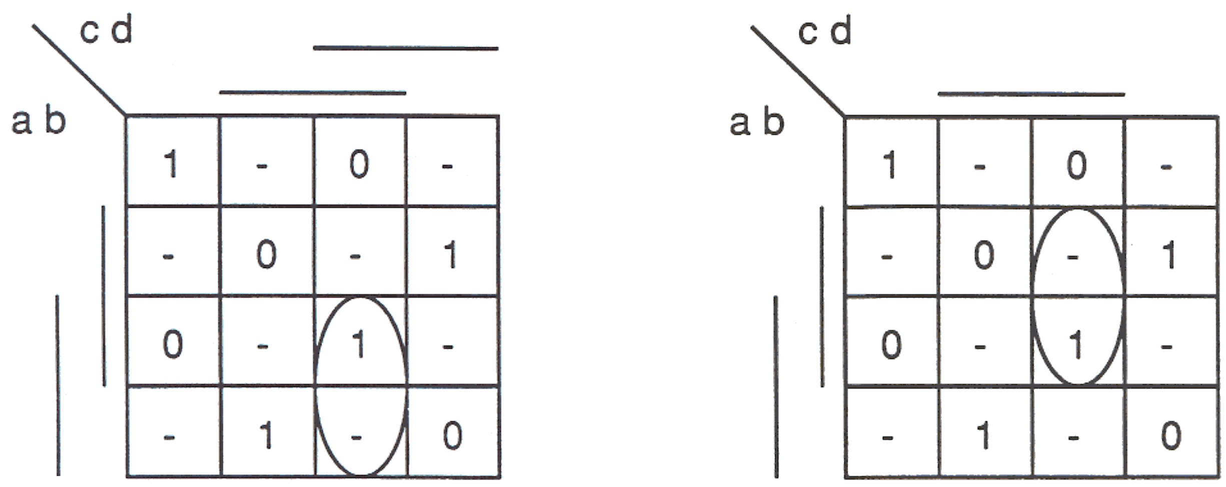


Figure 4: Approximation of ODC_y for $y = 0$ and $y = 1$ respectively provided by Eq. (17)

The portion of ODC_y covered with this second approach is shown in Fig.(4). Interestingly, the accuracy is not greater than that of Eq. 16. Note, however, that Eq. 17 requires fewer computations: only ODC_{y_2} needs to be decoupled from y_1. To realize this decoupling, a further simplification is proposed. It occurs at the vertex where the actual reconvergence occurs. For example, in Fig. (2), it occurs at vertex C and consists in computing the portion of ODC_{z_2} that is independent from z_1 (*i.e.* a *compatible don't care* subset for z_2). The approximation of ODC_{y_2} that can be obtained in this way is thus automatically independent from y_1, and can be used directly in Eq. 17.

Finally, Savoj *et al.* propose in [15] a yet conceptually better approximation. In the case of two fanout variables it reduces to computing first a subset of

$$\widetilde{ODC}_{y_2} = ODC_{y_2}|_{y=0} ODC_{y_2}|_{y=1} + ODC_{y_2}\overline{ODC}_{y_1}$$

Although this represents a larger subset of ODC_{y_1} than provided by b_0b_1, still when the product $ODC_{y_1}\widetilde{ODC}_{y_2}$ is formed, we obtain Eq. 17 again.

From the above analysis it turns out that all the methods proposed so far capture essentially the same portion of the ODC sets, although with different degrees of computational efficiency.

5.3 A new approximation

We present here new approximation methods that result in a substantially larger coverage of the ODC_y map. This approximation can readily be obtained by expanding the EXORs appearing in Eq. 10:

$$ODC_y = ODC_{y_1} ODC_{y_2}|_{y_1=\overline{y}} + \overline{ODC}_{y_1}\overline{ODC}_{y_2}|_{y_1=\overline{y}} +$$

$$+ ODC_{y_2} ODC_{y_1}|_{y_2=\overline{y}} + \overline{ODC}_{y_2}\overline{ODC}_{y_1}|_{y_2=\overline{y}} \qquad (19)$$

If subsets of ODC_{y_i} and $\overline{ODC}_{y_i}$ are available, then Eq. 19 automatically provides a subset of the true ODC_y. This approximation was first described in [14] and requires the explicit knowledge of subsets of ODC_{y_i} and OC_{y_i}.

Eq. 19 also shows that

$$\widetilde{ODC}_y = ODC_{y_1} ODC_{y_2}|_{y_1=\overline{y}} + ODC_{y_2} ODC_{y_1}|_{y_2=\overline{y}}$$

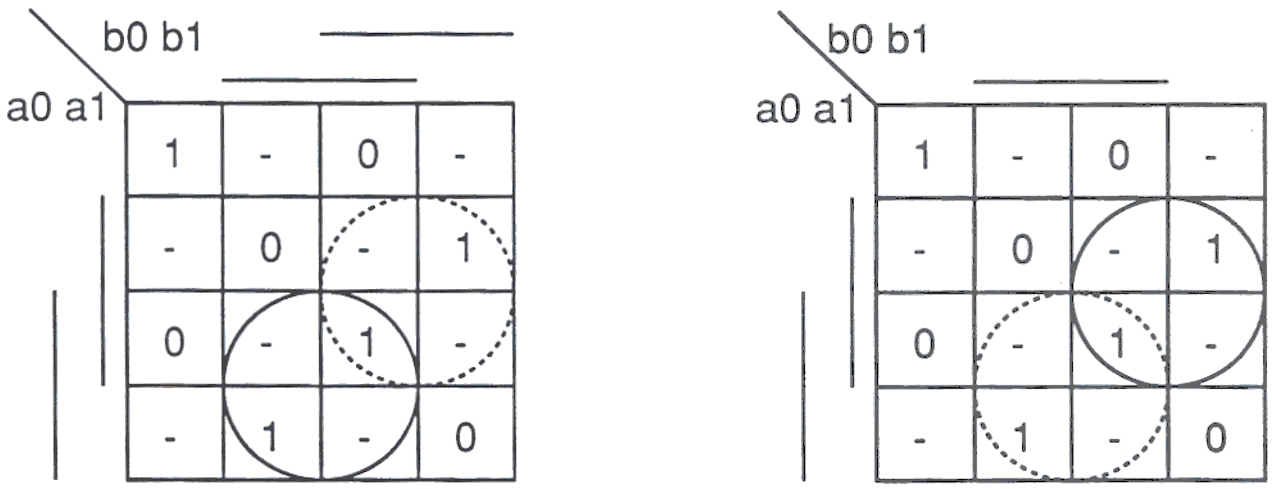


Figure 5: Approximation for the ODC sets for $y = 0$ and $y =$ respectively. Solid circles: coverage provided by Eq. (20); dotted circles: coverage provided by Eq. 21.

is also a subset of ODC_y. In order to compare this latter approximation with the previous ones, we rewrite it in terms of a_0, a_1, b_0, b_1 :

$$\widetilde{ODC}_y = (a_0\overline{y} + a_1 y)(b_0 y + b_1\overline{y}) + (a_0 y + a_1\overline{y})(b_0\overline{y} + b_1 y)$$

The maximum coverage provided by Eq. 21 is shown in Fig. (5). Solid circles correspond to the maximum coverage provided by the first product term in Eq. 20, while dashed circles correspond to the second one. Notice that each of them can cover half of the actual ODC_y, and that their sum covers up to 75% of the actual ODC_y, a substantial improvement with respect to all previous local approximations.

6 Multiple-vertex optimization and compatible *don't care* sets

It is known that the optimization of a function f_y alters, in general, the *don't care* set associated to the functions in its transitive fanin. Consequently, either the vertices are optimized in reverse topological order, or the full observability *don't care* sets have to be iteratively updated. Furthermore, the result of the optimization is generally dependent on the order with which the vertices are optimized.

To overcome these difficulties, the use of *compatible* observability *don't cares* has been proposed in [12, 15].

6.1 Multiple-vertex optimization

The optimization of n functions $f_{y_1}, \cdots, f_{y_n}$ in the network can be modeled by means of multiple perturbations $\delta_1, \cdots, \delta_n$. Let $\underline{y}$ denote the vector $(y_1, \cdots, y_n)$. We denote by $\mathcal{F}_{\underline{y}}$ the function realized by the perturbed network. By introducing the function

$$EQV(\delta_1, \quad ,\delta_n) = \mathcal{F}_{\underline{y}}(\delta_1, \quad ,\delta_n)\overline{\oplus}\mathcal{F}_{\underline{y}}(0, \quad ,0)$$

by arguments similar to those presented in Sect. 2, the set of feasible perturbations is represented by the equation

$$EQV + SDC_{\mathcal{F}_{\underline{y}}} + SDC_{ext} + ODC_{ext} \equiv 1$$

Finding the functions $g_{y_1}, \cdots, g_{y_n}$ of minimum total cost such that the corresponding perturbations satisfy Eq. 22 is the well known Boolean Relation minimization problem [5]. Its solution is usually difficult even for relatively small values of n because of the underlying binate covering problem.

Eq. 22 can be manipulated to obtain a representation of the possible perturbations in the form $\delta_i \subseteq DC_{y_i}$; $i = 1, \cdots, n$ as follows.

The function EQV can be rewritten as

$$EQV = \left(\mathcal{F}_{\underline{y}}(\delta_1, \cdots, \delta_n) \overline{\oplus} \mathcal{F}_{\underline{y}}(0, \delta_2, \cdots, \delta_n)\right) \overline{\oplus} \left(\mathcal{F}_{\underline{y}}(0, \delta_2, \cdots, \delta_n) \overline{\oplus} \mathcal{F}_{\underline{y}}(0, \cdots, 0)\right) =$$

$$\left(\overline{\delta}_1 + ODC_{y_1}(\delta_2, \cdots, \delta_n)\right) \overline{\oplus} EQV(0, \delta_2, \cdots, \delta_n)$$

Given expressions a, b, c, we recall that the Boolean equation $a \overline{\oplus} b + c = 1$ holds if and only if

$$a + \overline{b} + c =$$

$$b + \overline{a} + c = 1.$$

By setting $a = \overline{\delta}_1 + ODC_{y_1}(\delta_2, \cdots, \delta_n)$, $b = EQV(0, \delta_2, \cdots, \delta_n)$, $c = SDC_{\mathcal{F}_{\underline{y}}} + SDC_{ext} + ODC_{ext}$, Eq. 22 reduces to

$$\overline{\delta}_1 + ODC_{y_1}(\delta_2, \cdots, \delta_n) + \overline{EQV}(0, \delta_2, \cdots, \delta_n) + SDC_{\mathcal{F}_{\underline{y}}} + SDC_{ext} + ODC_{ext} \equiv 1 \tag{23}$$

$$EQV(0, \delta_2, \cdots, \delta_n) + \delta_1 \overline{ODC}_{y_1}(\delta_2, \cdots, \delta_n) + SDC_{\mathcal{F}_{\underline{y}}} + SDC_{ext} + ODC_{ext} \equiv 1 \tag{24}$$

The general solution of Eq. 23 is

$$\delta_1 \subseteq ODC_{y_1}(\delta_2, \cdots, \delta_n) + \overline{EQV}(0, \delta_2, \cdots, \delta_n) + SDC_{\mathcal{F}_{\underline{y}}} + SDC_{ext} + ODC_{ext} \overset{\text{def}}{=} DC_{y_1} \tag{25}$$

while Eq. 24 is certainly satisfied by any perturbation $\delta_2, \cdots, \delta_n$ satisfying

$$EQV(0, \delta_2, \cdots, \delta_n) + SDC_{\mathcal{F}_{\underline{y}}} + SDC_{ext} + ODC_{ext} \equiv 1 \tag{26}$$

Eq. 26 can iteratively be solved by a similar technique, thus leading to a set of sufficient conditions of type

$$\delta_i \subseteq DC_{y_i}(0, \cdots, 0, \delta_{i+1}, \cdots, \delta_n); \quad i = 1, \cdots, n$$

It is worth noting in particular the dependencies of the r.h.s. of Eq. 25 from the perturbations $\delta_2, \cdots, \delta_n$. They state precisely the intuition that the optimal choice of the function g_i replacing f_i depends on that of the other functions $g_2, \cdots, g_n$, *i.e.* that the optimization problems for the individual gates are coupled.

Compatible *don't care* sets.

Compatible *don't care* sets have been shown to be useful in "decoupling" the optimization problem for the individual gates. We present here a definition of compatible *don't care* sets in terms of perturbation analysis of combinational networks.

Definition 6.1 *A don't care set* DC_{y_1}, **independent from** $\delta_2, \cdots, \delta_n$, *is termed* **compatible** *with respect to* $\delta_2, \cdots, \delta_n$ *if it represents a don't care set for the variable* y_1, *regardless of the perturbations* $\delta_2, \cdots, \delta_n$, *satisfying*

$$EQV(0, \delta_2, \cdots, \delta_n) + SDC_{\mathcal{F}_{\underline{y}}} + SDC_{ext} + ODC_{ext} \equiv 1$$

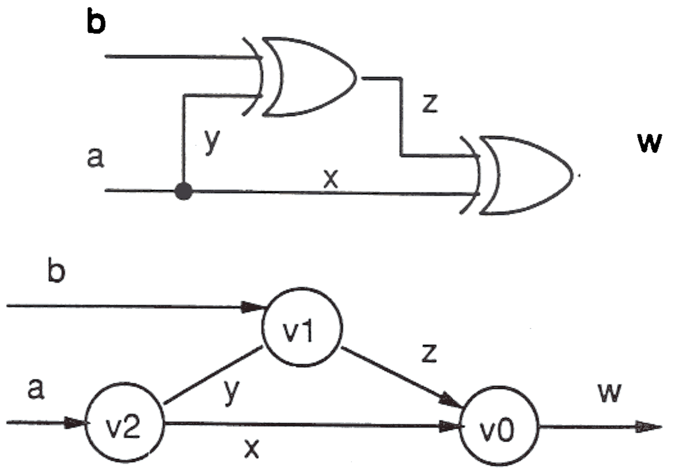


Figure 6: Circuit topology for the analysis of Compatible observability *don't cares.*

Thus, the compatible *don't care* set DC_{y_1} describes the degrees of freedom for optimizing f_{y_1} independently of the functions $g_2, \cdots, g_n$ replacing $f_2, \cdots, f_n$. The dependencies on $\delta_2, \ldots, \delta_n$ can be removed by performing an iterated consensus on Eq. 25.

In [15] an algorithm is presented that is claimed to compute *maximally compatible* observability *don't care* sets. We analyze here the algorithm with respect to the circuit shown in Fig. (6).

The algorithm begins by selecting a topological ordering of the vertices from the primary outputs. A possible order is $v_0 > v_1 > v_2$, and we assume that no external *don't care* conditions are given. The observability *don't care* set for z is computed according to

$$\widetilde{ODC}_z = \overline{\frac{\partial F}{\partial z}} = 0. \tag{27}$$

Next, in order to compute the *don't care* set for a, first the compatible *don't care* sets of x and y are determined:

$$\widetilde{ODC}_x = 0; \quad \widetilde{ODC}_y = 0.$$

and finally their product is formed:

$$ODC_a = ODC_y ODC_x = 0.$$

In the above derivation it is implicitly assumed that, in order to derive a compatible *don't care* set $\widetilde{ODC}_a$, first *compatible don't care* sets for the variables x, y must be obtained. On the other hand, these latter sets need not be compatible, as they are never used for logic optimization, and only the *don't care* set of the variable a is required to be compatible with those of z and w. In particular, for the circuit of Fig. 6.2, the set $DC_a = 1$ is actually compatible with the other *don't care* sets.

6.3 Local rules for maximally compatible *don't care* sets

The general rule for computing the maximal compatible *don't care* set for the fanout stem x from those of two fanout variables y_1 and y_2 can be derived as follows. Let $\delta_1, \cdots, \delta_n$ denote n arbitrary perturbations of the network, and let $\delta_{n+1}, \delta_{n+2}$ denote the perturbations for the two fanout variables y_1 and y_2. The equivalence of the perturbed network to the original one is described by the equivalence function

$$EQV(\delta_1, \quad , \delta_n, \delta_{n+1}, \delta_{n+2}) = \mathcal{F}_{\underline{y}}(\delta_1, \quad , \delta_n, \delta_{n+1}, \delta_{n+2}) \overline{\oplus} \mathcal{F}_{\underline{y}}(0, \quad , 0) \tag{30}$$

Since for any *don't care* set $\widetilde{ODC}_x$ the variables y_1 and y_2 are subject to the same perturbation, $\delta_{n+1} = \delta_{n+2}$.

The conditions for the perturbation of the fanout variable x are thus stated by

$$EQV(\delta_1, \cdots, \delta_{n+1}, \delta_{n+1}) + SDC_{\mathcal{F}_{\underline{y}}} + SDC_{ext} + ODC_{ext} = 1$$

By manipulations similar to those leading to Eq. 6.1, we have the conditions:

$$\mathcal{F}_{\underline{y}}(\delta_1, \quad, \delta_n, \delta_{n+1}, \delta_{n+1})\overline{\oplus}\mathcal{F}_{\underline{y}}((\delta_1, \cdots, \delta_n, 0, 0) + \overline{EQV}(\delta_1, \cdots, \delta_n) + SDC_{\mathcal{F}_{\underline{y}}} + SDC_{ext} + ODC_{ext} = 1 \tag{31}$$

and

$$EQV(\delta_1, \cdots, \delta_n) + SDC_{\mathcal{F}_{\underline{y}}} + SDC_{ext} + ODC_{ext} = 1$$

Eq. 31 is solved by

$$\delta_{n+1} \subseteq \mathcal{F}_{\underline{y}}(\delta_1, \cdots, \delta_n, 1, 1)\overline{\oplus}\mathcal{F}_{\underline{y}}((\delta_1, \cdots, \delta_n, 0, 0) + \overline{EQV}(\delta_1, \cdots, \delta_n) + SDC_{\mathcal{F}_{\underline{y}}} + SDC_{ext} + ODC_{ext} \tag{32}$$

The maximally compatible *don't care* set is thus the iterated consensus of the r.h.s. of Eq. 32 w.r.t. $\delta_1, \cdots, \delta_n$.

We now relate this set to the ones of the fanout variables y_1, y_2. By transforming

$$\mathcal{F}_{\underline{y}}(\delta_1, \cdots, \delta_n, 1, 1)\overline{\oplus}\mathcal{F}_{\underline{y}}(\delta_1, \quad, \delta_n, 0, 0) =$$

$$\left(\mathcal{F}_{\underline{y}}(\delta_1, \cdots, \delta_n, 1, 1)\overline{\oplus}\mathcal{F}_{\underline{y}}(\delta_1, \cdots, \delta_n, 1, 0)\right)\overline{\oplus}\left(\mathcal{F}_{\underline{y}}(\delta_1, \cdots, \delta_n, 1, 0)\overline{\oplus}\mathcal{F}_{\underline{y}}(\delta_1, \cdots \ , \delta_n, 0, 0)\right)$$

$$ODC_{y_2}(\delta_1, \cdots, \delta_n)|_{y_1=\overline{x}}\overline{\oplus}ODC_{y_1}(\delta_1, \ \cdots, \delta_n)|_{y_2=x}$$

we finally obtain the compatible *don't care* set of x:

$$C_{\delta_1, \cdots, \delta_n}\Big((ODC_{y_2}|_{y_1=\overline{x}} + \overline{EQV}(\delta_1, \cdots, \delta_n) + SDC_{\mathcal{F}_{\underline{y}}} + SDC_{ext} + ODC_{ext})\overline{\oplus}$$

$$(ODC_{y_1}|_{y_2=x} + \overline{EQV}(\delta_1, \quad, \delta_n) + SDC_{\mathcal{F}_{\underline{y}}} + SDC_{ext} + ODC_{ext})\Big)$$

where EXNOR is applied to two terms representing the *don't care* of the individual stems. It is possible to verify that this formula yields the maximal compatible ODC for the example of Figure 6.

It is easy to verify that, unfortunately, the *consensus* operation on the EXNOR of two expression is not equal to the EXNOR of the consensus of the two individual expressions, *i.e.* C is not distributive w.r.t. the EXNOR. Therefore, the maximally compatible *don't care* set for x is not in general derivable from those of the fanout branches.

We can approximate the EXNOR operation by a product, as discussed in Sect. 5.3. Since the *consensus* operation is distributive over the product operation, it is then possible to approximate the maximally compatible *don't care* set of x as a product of compatible ODCs of the individual stems.

We conclude by observing that the approach just outlined for the extraction of maximally compatible *don't care* sets can be extended to the case of arbitrary combinational networks with multiple-way forks of arbitrary size.

7 Conclusions

In this paper, we presented a comparative analysis of approximations to the observability *don't care* sets. The comparison has shown that all "local" current approximations capture (at most) a fraction of the actual *don't care* sets. A new approximation has been proposed, that has been shown to capture a larger portion of the observability *don't care* sets than the previous ones.

The computation of compatible observability *don't care* has been advocated for logic optimization. In this paper we have shown that compatible *don't care* sets can be derived in an alternative way, that yields larger sets than those previously computed [15].

8 Acknowledgements.

This research was supported by 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 Multiple level 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. Mocey 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, Aug ust 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] 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.

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

[12] 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.

[13] M. Damiani and G. De Micheli, " Observability *don't care* sets and Boolean Relations ", *Proc. ICCAD 1990*, S. Clara, Nov. 1990.

[14] 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 *CSL Report* CSL- TR-90-424.

[15] H.Savoj and R.Brayton, 'The use of Observability and External don't care sets for the simplification of multi-level networks', *Proc. DAC 90.*, pp. 297-301.

[16] P.Mc Geer and R.Brayton ' The Observability Don't care set and its Approximations' *Proc ICCD 90*, pp. 45-48, Oct 90.