Compatible Observability Don't Cares Revisited

R. K. Brayton
EECS Dept.
University of California, Berkeley
brayton@eecs.berkeley.edu

Abstract

CODCs stand for compatible observability don't cares. We first examine the definition of compatibility and when a set of CODCs is compatible. We then discuss Savoj's CODC computation for propagating CODCs from a node's output to its fanins, and show by example, that the results can depend on the current implementation of the node. Then we generalize the computation so that the result is independent of the implementation at the node. The CODCs propagated by this computation are proved to be maximal in some sense. Local don't cares (LDCs) are CODCs of a node, pre-imaged to the primary inputs and then imaged and projected to the local fanins of the node. LDCs combine CODCs with SDCs (satisfiability don't cares), but only the CODC part is propagated to the fanin network. Another form of local don't cares, propagates both the CODC and SDC parts to the fanin network. Both are shown to be compatible in some sense, but conservative. We give a method for updating both kinds of local don't cares incrementally when other nodes in the network are changed.

1 Introduction

Permissible functions were first defined by Muroga [1]. These are incompletely specified functions (ISFs) in terms of the primary inputs of a NOR network. For a given node in such a network, its permissible function describes a set of functions any one of which can be used at the node without changing the functionality of the network. Also, compatible sets of permissible functions (CSPFs) were defined, which can be used independently on a set of nodes without changing functionality. Later, this work was generalized from NOR networks to arbitrary Boolean networks and called compatible observability don't cares (CODCs) [2]. CODCs differed also in that they were expressed in terms of intermediate signals. CODCs are used in SIS [3], but compatibility is used only to provide an efficient computation method. It is not used to allow simultaneous changes in the circuit; in SIS, once a CODC is computed at a node, the node is immediately minimized and the CODC is only used for propagating CODCs to the fanin edges, after which it is thrown away.

In this paper we first review the notion of compatibility for ISFs and the results of [4, 5] which give some generalizations on the computation of CODCs. We then outline the CODC computation of Savoj [2] which is used to propagate CODCs from the output of a node to its fanin edges. We point out that the computation can depend on the current implementation of the node; different CODCs are obtained for the fanin edges for different node implementations. This leads to the question of whether there is a maximal such computation, i.e. a choice of implementation at the node which yields maximal CODCs on the fanin edges. We show, instead, a computation which is independent of the current implementation of the node and provides the maximum don't care set on the fanin with the highest priority.

Next we look at local don't cares which are used by ESPRESSO to minimize a node. Local don't cares (LDCs) are derived from CODCs of a node, pre-imaged to the primary inputs and then imaged and projected to the local fanins of the node. Thus they combine CODCs with SDCs (satisfiability don't cares). There are two forms of these. LDCs are computed by propagating only the CODC part to the fanin network. Another form of local don't cares, propagates both the CODC and SDC parts to the fanin network. Both are shown to be compatible in some sense, but conservative. We give a method for updating both kinds of local don't cares incrementally when other nodes in the network are changed.

2 Compatibility

CSPFs and CODCs are compatible; we re-examine what this means. Let $\mathcal{B}$ be a Boolean network and $\mathcal{S}(\mathcal{B})$ be its external specification. $\mathcal{B}(F_1)$ is the same Boolean network, except for the representation of node η_1 replaced by F_1.

Definition 1 *An ISF $\mathcal{F}_i$ for a node η_i in $\mathcal{B}$ is valid if*

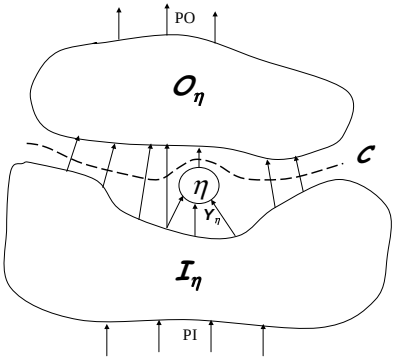

Figure 1: 3-node view of network.

$F_i \in \mathcal{F}_i \Rightarrow \mathcal{B}(F_i) \in \mathcal{S}(\mathcal{B})$. *A don't care set is valid if the implied ISF is valid.*

Definition 2 *A set of ISFs $\{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ at a set of nodes in $\mathcal{B}$ is compatible if $\forall F_i \in \mathcal{F}_i, \mathcal{B}(F_1, \ldots, F_n) \in \mathcal{S}(\mathcal{B})$.*

When we compute CODCs, the ISFs at a node η, F_η, is specified in terms of variables that form a minimal cutset, $\mathcal{C}$, (which must include η) between the PIs and the POs. The cutset could be the variables which feed into the transitive fanout set of η, or could include η plus PIs which are not dominated by η (if η is a dominator of a PI, then that PI would not be part of a minimal cutset). If $\mathcal{C}$ is the latter, then the ISF obtained with this set of don't cares is a function of only primary inputs and is the same as the CSPF for η (assuming that the same ordering was used in deriving the two). The CODC derived for a particular node, η, will depend on the ordering of the nodes used during the derivation as well as the particular implementation of the fanout network (topology and node functions).

For each node η and cutset $\mathcal{C}$, the maximum ODC can be computed [4, 5] by viewing the network as consisting of 3 nodes as shown in Figure 1;

1. the node η itself which has its immediate fanins, Y_η, as inputs, and a single output,
2. a fanin super-node $\mathcal{I}_\eta$ with multiple outputs which are the cutset variables $\mathcal{C} - \{\eta\}$ plus Y_η, i.e. the outputs of $\mathcal{I}_\eta$ are $\mathcal{C} - \{\eta\} \cup Y_\eta$, and
3. a fanout super-node $\mathcal{O}_\eta$, with $\mathcal{C}$ as inputs and the primary outputs as outputs.

The maximum ISF at η, $\mathcal{F}_\eta$, is a function of $Y_\eta \cup \mathcal{C} - \{\eta\}$. It is a function of $\mathcal{C}$ because it is computed this way, and it is a function of Y_η because its current cover is expressed in Y_η. This ISF does not change if the implementations of $\mathcal{I}_\eta$ and $\mathcal{O}_\eta$ change; clearly, the ISF does not even depend on $\mathcal{I}_\eta$; for $\mathcal{O}_\eta$, the non-observability from any input pin of $\mathcal{O}_\eta$ through $\mathcal{O}_\eta$ is the same as the non-observability from that pin through any other decomposition of $\mathcal{O}_\eta$. However, when we compute compatible ODCs we are computing subsets, which can depend on the decomposition and the ordering of the nodes.

Observation 1 *Any CODC at η computed using any valid decomposition of $\mathcal{O}_\eta$, remains valid for any other decomposition of $\mathcal{O}_\eta$.*

Now consider two nodes in $\mathcal{B}$. Since $\mathcal{B}$ is acyclic, we can choose the one nearest the PIs , η_1, and the other η_2 is either in its fanout network or not. In the first case, $ISF(\eta_1)$ remains valid independent of the implementation at η_2 because it is in $\mathcal{O}(\eta_1)$, and $ISF(\eta_2)$ remains valid independent of the implementation at η_1 since it is in $\mathcal{I}(\eta_1)$. In the second case, each is in the $\mathcal{I}_\eta$ or $\mathcal{O}_\eta$ of the other so their ISFs are compatible. This means that we can change the implementation of one, and the ISF of the other remains valid. Thus we have the following theorem.

Theorem 1 *The set of ISFs for all nodes in the network, where the ISFs are computed by any valid CODC algorithm (no matter which cutset is chosen), is compatible.*

The CODC algorithms in [1, 2] are valid.

We already mentioned that for a given node η, different cutsets can be chosen. This just means that the corresponding CODCs are functions of different sets of variables. Each gives a set of minterms under which the output of η is not observable at any of the primary outputs. Some minterms may not be satisfiable, so they are not relevant. These depend on $\mathcal{I}_\eta$ which gives the satisfiability conditions. It is interesting to note that as the cutset $\mathcal{C}$ is moved more towards the primary inputs, the number of minterms in the variables of $\mathcal{C}$ that are not satisfiable decreases, because $\mathcal{I}$ decreases. In the limit, $\mathcal{C}$ becomes only PIs (and η) which has no unsatisfiable minterms (unless external don't cares have been specified).

3 Savoj's CODC Computation

A method for computing CODCs at the fanins of a node in a Boolean network is given by Savoj's formula (1). This is the method used in SIS. Here η is a node in the network and $CODC_\eta$ is the CODC already computed for η. Don't cares are propagated to the fanins of η, $y_1, y_2, \ldots$, according to some ordering $y_1 < y_2 < \ldots$, where y_1 gets the most don't cares possible, y_2 next, etc. The don't cares are assigned to the fanins so that they are compatible. F is the

current implementation at the node and $\frac{\partial F}{\partial x} = F_x \oplus F_{\overline{x}}$.

$$CODC_{y_k \to \eta} = (\frac{\partial F}{\partial y_1} + \forall_{y_1}) \ldots (\frac{\partial F}{\partial y_{k-1}} + \forall_{y_{k-1}}) \overline{\frac{\partial F}{\partial y_k}} + CODC_\eta \quad (1)$$

For y_1 we get

$$CODC_{y_1 \to \eta} = \overline{\frac{\partial F}{\partial y_1}} + CODC_\eta$$

The operator $(\frac{\partial F}{\partial y_i} + \forall_{y_i})$ operates on the function resulting from the computation on its right and ensures that $CODC_{y_k \to \eta}$ is compatible with the $CODC_{y_i \to \eta}$ already assigned, i.e. $i < k$. The argument for the operator providing compatibility is that don't care terms coming from $\overline{\frac{\partial F}{\partial y_k}}$ should be such that the current implementation F is sensitive to y_i $(\frac{\partial F}{\partial y_i})$ (and thus could not have been assigned as a don't care for y_i) or independent of y_i $(\forall_{y_i})$ (and thus it is independent of how the don't care might be used in simplifying y_i). This method for making the CODCs of the fanins compatible breaks a circularity which can happen if one input relies on another for giving a proper input value, and vice versa. In the case of such a circularity, both may become don't care and a proper input value is not maintained.

Once all fanins of a node are assigned CODC's, they can be propagated to the sources of the fanin signals. At such a source node, the CODCs of all its fanout wires are intersected to obtain the CODC of the node:

$$CODC_k = \bigcap_{\eta \in FO(k)} CODC_{y_k \to \eta}$$

The legality of this intersection is based on the fanout CODCs being compatible [2].

Another important point in this computation is that the node CODCs are computed from POs in reverse topological order. For each node η, its CODC is expressed in terms of the signals in its transitive fanout plus immediate side inputs to the transitive fanout set. Signals internal to $\mathcal{O}_\eta$ can be eliminated by substitution in terms of inputs to $\mathcal{O}_\eta$. As discussed in Section 2, the CODC is independent on the network $\mathcal{I}_\eta$ driving $\mathcal{O}_\eta$. The CODC simply states the unobservability of η's output in terms of the inputs of $\mathcal{O}_\eta$, which are treated as independent signals.

Note that in (1), $CODC_{y_k \to \eta}$ may depend on the variable y_k both through $\frac{\partial F}{\partial y_i}$ as well as $CODC_\eta$ since y_k may be a side input to the transitive fanout of η. However, there is no dependency of the fanin part of the network. As discussed in Section 1, if the fanins functions change, the CODCs computed for the nodes are still valid and compatible; i.e. if $\mathcal{O}_\eta$ or $\mathcal{I}_\eta$ is changed, $CODC_\eta$ remains valid.

4 Dependency on Implementation F at the Node

The results computed by (1) depend on the particular implementation of η, i.e. the particular cover F at η. This is shown by the following example in which we implement a node η in different ways using a given $CODC_\eta$ as don't care and get different $CODC_{y_k \to \eta}$s.

Example 1 *Let $f = ab + ac + a'b'c'$ with $CODC_f = a'b'c + a'bc'$. Using a as y_1, i.e. a is the fanin with the highest priority, if f were implemented with different covers, we get the following different CODCs on the signal $a \to \eta$, using (1):*

$$\begin{aligned}
&F_0 = ab + ac + a'b'c' \Longrightarrow \\
&CODC_{a \to \eta} = CODC_f = a'b'c + a'bc'; \\
&F_1 = ac + ab + a'b' \Longrightarrow \\
&CODC_{a \to \eta} = b'c + CODC_f = b'c + a'bc'; \\
&F_2 = ac + ab + a'c' \Longrightarrow \\
&CODC_{a \to \eta} = bc' + CODC_f = bc' + a'b'c; \\
&F_3 = ab + b'c + a'c' \Longrightarrow \\
&CODC_{a \to \eta} = b'c + bc' + CODC_f = b'c + bc'.
\end{aligned}$$

Here we have assumed that a is the first fanin in the ordering so that it gets the most don't cares. Note that a is also an input to $CODC_f$.

5 An Implementation Independent Computation

An interesting problem is how to choose the implementation at η which yields the most don't cares to its fanins. As the next theorem shows, we can generalize Savoj's formula to get fanin don't cares which are independent of the implementation F and which yield maximal don't cares, at least for the first fanin.

Theorem 2 *Consider the following generalization of the Savoj formula.*

$$\begin{aligned}
&CODC_{y_k \to \eta} = \\
&(\overline{CODC}_{y_1 \to \eta} + \forall_{y_1}) \ldots (\overline{CODC}_{y_{k-1} \to \eta} + \forall_{y_{k-1}}) \\
&((F_{y_k} \overline{\oplus} F_{\overline{y}_k}) + \exists_{y_k} CODC_\eta) + CODC_\eta
\end{aligned}$$

where F is any implementation at the node. $CODC_{y_k \to \eta}$ is valid and is independent of F for all k. Further, $CODC_{y_1 \to \eta}$ is maximal.

Proof. The interpretation is that each operator of the type

$$(\overline{CODC}_{y_i \to \eta} + \forall_{y_i})$$

says that of the terms that come from the calculation to the right of the operator, we only keep those which are cares for y_i or the terms which do not depend on y_i. The terms given by

$$((F_{y_k} \overline{\oplus} F_{\overline{y}_k}) + \exists_{y_k} CODC_\eta)$$

are a generalization of $\overline{\frac{\partial F}{\partial y_k}}$.

Let F be any implementation of the node η with the don't care for η given by $CODC_\eta$. For simplicity of notation let $d = CODC_\eta$, $g = F\overline{d}$, and $h = \overline{F}\ \overline{d}$. Thus $\mathcal{F} = (g, d, h)$ is independent of F and is simply the ISF at η, according to the don't care set $CODC_\eta$. We first compute where $\mathcal{F}$ is **sensitive** to the value of y_k. These are terms where $g = 1$ for one value of y_k and $h = 1$ for the other. The terms are obtained by considering a minterm in the space of inputs to η, Y_η, and toggling the value for y_k. Thus, we get the terms,

$$g_{y_k} h_{\overline{y}_k} + g_{\overline{y}_k} h_{y_k}.$$

Substituting $g = F\overline{d}$, and $h = \overline{F}\ \overline{d}$, we get

$$g_{y_k} h_{\overline{y}_k} + g_{\overline{y}_k} h_{y_k} = (F_{y_k} \overline{F}_{\overline{y}_k} + F_{\overline{y}_k} \overline{F}_{y_k}) \overline{d}_{y_k} \overline{d}_{\overline{y}_k}$$

Hence

$$\overline{g_{y_k} h_{\overline{y}_k} + g_{\overline{y}_k} h_{y_k}} = \overline{\frac{\partial F}{\partial y_k}} + \exists_{y_k} d \qquad (2)$$

Clearly, (2) is independent of the representation F used at the node, since g and h are independent of F. We can add to the expression, $CODC_\eta$ since these are terms where the output of η is don't care. This is also independent of F. Finally, for the general term $CODC_{y_k \to \eta}$, each $CODC_{y_i \to \eta}$ is independent of F, by induction, and hence the result.

Note that $CODC_{y_1 \to \eta}$ always contains that given by (1) and can be strictly larger only if $CODC_\eta$ depends explicitly on y_1. To prove that

$$\overline{\frac{\partial F}{\partial y_k}} + \exists_{y_k} d$$

is maximal, suppose there is another term (in the space $\mathcal{C} \cup Y_\eta - \eta$ that can be added to it. Then it would be a term (independent of y_k) where $\eta = 1$ for one value of y_k and $\eta = 0$ for the other value (no matter what the implementation F is). If $CODC_\eta$ is maximal, then this would mean that as η toggles, it would cause toggling at one of the primary outputs, which would mean that output network is sensitive to η under the conditions of inputting the new term to $\mathcal{O}_\eta$. **QED**

Although $CODC_{y_1 \to \eta}$ in Theorem 2 is larger than that of (1), this cannot always be claimed for the next fanin y_2; even though $(F_{y_2} \overline{\oplus} F_{\overline{y}_2}) + \exists_{y_2} CODC_\eta$ starts out larger (if $CODC_\eta$ depends on y_2), the operator $(\overline{CODC}_{y_1 \to \eta} + \forall y_1)$ may trim it down more than in (1) because $\overline{CODC}_{y_1 \to \eta}$ might be smaller than before.

Also note that we can increase the possibility that $CODC_\eta$ will depend explicitly on $y_k \in FI(\eta)$ by moving the cut to include those $y_k \in FI(\eta)$ that have reconvergent fanout. The largest set is obtained if the cutset is in terms of only PI's (and the node η), and we express η in terms of the PIs in its transitive fanin, since then the possibility of reconvergent fanout is maximized. Since a PI in $TFI(\eta)$, not dominated by η, is an input to $CODC_\eta(PI)$, the new computation should give better results. This is the situation in Example 1.

In general, we can collapse η a number of levels into its fanins and make the cut include the new inputs to η. Then the new computation would give larger don't cares on the fanin edges to the new inputs of the collapsed node η. Note that the new computation is independent of what particular internal representation one has in η. This contrasts with CSPF computations, which require that a node is first decomposed into a subnetwork of NOR gates. Then the computation is dependent on how the decomposition was done and does not represent all decompositions.

Example 2 *In Example 1, $f = ab + ac + a'b'c'$ and $d = a'b'c + a'bc'$. Since $fd = \emptyset$, $g = f$. Since $(g_a \overline{\oplus} g_{\overline{a}}) = \emptyset$ we get*

$$CODC_{a \to \eta} = d_a + d_{\overline{a}} = b'c + bc'$$

which is the largest set obtained of the 4 implementations discussed in Example 1.

6 Local Don't Cares

In SIS, after a CODC is computed at a node, it is projected onto the primary inputs and then its complement is projected back to the space of the immediate fanin signals of η, after which the result is complemented. The resulting don't care set is a local don't care set and is called (LDC_η). The associated ISF is now only dependent on the fanins of η and is suitable for input to ESPRESSO (which is given the current cover of η and the don't care set LDC_η). Then the node is minimized using LDC_η and if the cover is improved, the representation for node η is changed. Thus each node η in the network has an LDC_η which now reflects both the observability don't cares (ODCs) and satisfiability don't cares (SDCs). SDCs reflect the interdependence of the variables in $\mathcal{C} - \{\eta\} \cup Y_\eta$. It is important to recognize that the LDCs are not necessarily compatible because of the SDCs.

In the following, we will consider a more local computation of local don't cares. We use the notation $N(X, Y)$ for the relation between the primary inputs X and a subset of variables Y in the current network.

Algorithm 1

1. *Order the nodes in reverse topological order.*
2. *For each primary output j, define*

$$L_{j\to out_j}(Y_j) = \overline{\exists_X N(X,Y_j)\overline{EXDC_j(X)}}$$

3. *In general, at node j, map each fanout don't care set into the local input space,*

$$L_{j\to k}(Y_j) = \overline{\exists_X N(X,Y_j)\overline{\exists_{Y_k} N(X,Y_k)L_{j\to k}(Y_k)}}$$

4. *At each node j, define*

$$L_k(Y_j) = \bigcap_{k\in FO(j)} L_{j\to k}(Y_j)$$

For the first node, $L_j(Y_j) = L_{j\to out_j}(Y_j)$ since there is only one fanout.

5. *Use Theorem 2 with $L_j(Y_j)$ replacing $CODC_j$ in the formula to compute $L_{y_i\to j}(Y_j)$ for each fanin edge:*

$$\begin{aligned} L_{y_i\to j} &= (\overline{L}_{y_1\to j} + \forall_{y_1})\ldots(\overline{L}_{y_{i-1}\to j} + \forall_{y_{i-1}}) \\ &\quad ((F_{y_i}\overline{\oplus}F_{\overline{y}_i}) + \exists_{y_i} L_j) + L_j \end{aligned}$$

This computation differs from the computation of $LDC(Y_j)$ in the use of $L_j(Y_j)$ instead of $CODC_j(\mathcal{C}_j)$ in the propagation of don't cares through a node to its fanin edges. For $LDC(Y_j)$ we have,

Algorithm 2

1. *For each primary output define*

$$LDC_{j\to out_j}(Y_j) = \overline{\exists_X N(X,Y_j)\overline{EXDC_j(X)}}$$

2. *Map each fanout don't care set into the local input space,*

$$LDC_{j\to k}(Y_j) = \overline{\exists_X N(X,Y_j)\overline{\exists_{\mathcal{C}_k} N(X,\mathcal{C}_k)CODC_{j\to k}(\mathcal{C}_k)}}$$

3. *At each node j, define*

$$CODC_j(\mathcal{C}_j) = \bigcap_{k\in FO(j)} CODC_{j\to k}(\mathcal{C}_j)$$

4. *Use Theorem 2 with $CODC_j$ in the formula to compute $CODC_{y_i\to j}(\mathcal{C}_j)$ for each fanin edge.*

We remark that it is unlikely that either set, $\{LDC_j\}$ or $\{L_j\}$, is compatible, since they both depend on an image computation, which depends on the current implementation of the fanin network. However, we will show that they are compatible in some sense and give a method for incrementally updating the L_j after nodes in the network have been changed.

Suppose the representation of node k has changed. If k is in $\mathcal{O}_j$ then nothing needs to be done. However if k is in $\mathcal{I}_j$ we need to change L_j. Consider the following method for updating L_j.

Algorithm 3

1. *Compute $C(Y_j) = \overline{L_j(Y_j)}$, the care set for j. Let $\tilde{N}$ represent the changed network and*

$$E(Y_j,\tilde{Y}_j) = \exists_X N(X,Y_j)\tilde{N}(X,\tilde{Y}_j).$$

2. *Compute*

$$\tilde{C}_j(\tilde{Y}_j) = \exists_{Y_j} C(Y_j)E(Y_j,\tilde{Y}_j)$$

3. $L_j(\tilde{Y}_j) = \overline{\tilde{C}_j(\tilde{Y}_j)}$.

A similar algorithm can be applied to LDC_j and is valid for the same reasons as proved in Theorem 3 below.

Theorem 3 *Algorithm 3 is correct, i.e.*

$$L_j(\tilde{Y}_j) \equiv \overline{\tilde{C}(\tilde{Y}_j)} \subseteq \overline{\exists_X \tilde{N}(X,\tilde{Y})\overline{CODC}_j(X)}$$

but conservative, i.e.

$$\tilde{C}(\tilde{Y}_j) \neq \exists_X \tilde{N}(X,\tilde{Y})\overline{CODC}_j(X)$$

in general.

Proof. Denote $CS(X) = \overline{CODC}_j(X)$, $Y = Y_j$, $\tilde{L}(\tilde{Y}) = \tilde{L}_j(\tilde{Y}_j)$, $\tilde{C}(\tilde{Y}) = \tilde{C}_j(\tilde{Y}_j)$ and $C(Y) = \overline{L_j(Y_j)}$. We will prove that

$$\exists_X \tilde{N}(X,\tilde{Y})CS(X) = \exists_Y E(Y,\tilde{Y})C(Y)$$

Denote the left hand side as $\hat{C}(\tilde{Y})$. This is the correct care set in the $\tilde{Y}$ space, since $CS(X)$ is the correct care set in the X space. Now, the right-hand side is

$$\begin{aligned} &= \exists_Y E(Y,\tilde{Y})C(Y) \\ &= \exists_Y \exists_{X'} N(X',Y)\tilde{N}(X',\tilde{Y})C(Y) \\ &= \exists_Y \exists_{X'} N(X',Y)\tilde{N}(X',\tilde{Y})\exists_X N(X,Y)CS(X) \\ &= \exists_X \exists_{X'} \exists_Y N(X',Y)\tilde{N}(X',\tilde{Y})N(X,Y)CS(X) \\ &= \exists_X \exists_{X'} [\exists_Y N(X',Y)N(X,Y)]\tilde{N}(X',\tilde{Y})CS(X) \end{aligned}$$

This expression is $\tilde{C}(\tilde{Y})$. Clearly $\tilde{C}(\tilde{Y}) \supseteq \hat{C}(\tilde{Y})$ since we can get the latter by restricting $X = X'$.

To demonstrate that the algorithm is conservative, assume that there is a minterm, $(x',y,x,\tilde{y}) \in X'\times Y\times X\times \tilde{Y}$ such that

$$N(x',y)N(x,y)\tilde{N}(x',\tilde{y})CS(x) = 1, \qquad (3)$$

but $\tilde{y}$ is such that

$$\forall_{X''}[\tilde{N}(X'', \tilde{y})C(X'') = 0]. \quad (4)$$

(3) implies that $CS(x) = 1$ which implies by (4) that $\tilde{N}(x, \tilde{y}) = 0$. Also, (3) implies that $\tilde{N}(x', \tilde{y}) = 1$, which implies by (4) that $CS(x') = 0$. Thus we may get an extra don't care minterm only by the mechanism that a don't care term, x', ($CS(x') = 0$) maps (under N) into the same y as a care minterm x ($CS(x) = 1$), i.e. ($N(x', y)N(x, y) = 1$) and then x maps (under $\tilde{N}$) to a different $\tilde{y}$ than x' ($\tilde{N}(x', \tilde{y}) = 1$, $\tilde{N}(x, \tilde{y}) = 0$). However, we still must treat x like a don't care point since it is not differentiated by N. Since the only information we keep is $C(Y)$, (we do not keep $CS(X)$) in the algorithm, we cannot expect to recover $CS(X)$, i.e. that x is a care point. If there is no other don't care point that maps into $\tilde{y}'$ where $\tilde{N}(x, \tilde{y}') = 1$, then $\tilde{y}'$ is a point in $\tilde{C}(\tilde{Y})$ that is not in $\hat{C}(\tilde{Y})$. **QED**

L_j is a combination of a compatible observability don't care part and a satisfiability don't care part. Theorem 3 is interesting because it says that the observability part remains valid and compatible even if other nodes in the network have been changed. Only the satisfiability part needs to be updated and this can be done by mapping minterms in the Y_j space into different minterms in the $\tilde{Y}_j$ space.

Algorithms 1 and 2 propagate don't cares differently. L_j loses information, because whenever a local image is produced, some of the primary input minterms cannot be distinguished, e.g. when $N(x, y) = N(x', y)$. Since CODC computation does not do an imaging operation, no information is lost in Algorithm 2.

7 Conclusions

We discussed the details of CODC and local don't care computations. A formula was given which allows CODC propagation from a node's output to its fanin edges to be independent of the node's representation and is maximal for the first fanin. We investigated the compatibility of a set of nodes. The set of all CODCs and the set of all CSPFs in the entire network are both compatible sets. However, when an image computation is used to project don't cares onto a local space, generally compatibility is not maintained. Therefore, we gave a procedure for updating local don't cares when a set of changes has been made to nodes in the network. The procedure was shown to be correct but conservative. This shows that the information contained in a local don't care is still valid (in some sense) and just has to be re-mapped according to the changes made in its transitive fanin network.

CODCs can be useful for SPFD computations [6, 7, 8]. One of the problems with SPFDs is that if a node is changed according to the flexibility allowed by an SPFD, it may be necessary to change all nodes in the transitive fanout of that node. However, if the SPFD computation is started with CODCs at nodes that are, say 2 fanouts away from the node, then the CODCs block node change propagation beyond this point. In [8, 9, 10], we experimented with this use of CODCs with SPFDs to block long-range propagation of node changes.

The computation of SPFDs at a node is similar to that given in Algorithm 1. Also, SPFDs are propagated through a node to its fanin edges, independent of the current implementation of the node. Thus Theorems 2 and 3 have an analogy in SPFD computations.

Acknowledgements: We thank the SRC for supporting this work under contract 683-002 and the California Micro program and our sponsors under this program, Fujitsu, Cadence, and Synopsys.

References

[1] S. Muroga, Y. Kambayashi, H. C. Lai, and J. N. Culliney, "The Transduction Method - Design of Logic Networks Based on Permissible Functions," in *IEEE Trans. Computers*, Oct. 1989.

[2] H. Savoj and R. K. Brayton, "The Use of Observability and External Don't Cares for the Simplification of Multi-Level Networks," in *Proc. of the Design Automation Conf.*, pp. 297–301, June 1990.

[3] E. M. Sentovich, K. J. Singh, C. Moon, H. Savoj, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Sequential Circuit Design Using Synthesis and Optimization," in *Proc. of the Intl. Conf. on Computer Design*, pp. 328–333, Oct. 1992.

[4] Y. Watanabe and R. K. Brayton, "The Maximum Set of Permissible Behaviors for FSM Networks," in *Proc. of the Intl. Conf. on Computer-Aided Design*, 1993.

[5] Y. Watanabe, L. Guerra, and R. K. Brayton, "Logic optimization with multi-output gates," in *Proceedings of the International Conference Computers and Devices*, Sept. 1993.

[6] S. Yamashita, H. Sawada, and A. Nagoya, "A new method to express functional permissibilities for LUT based FPGAs and its applications," in *Proceedings of the International Conference on Computer-Aided Design*, pp. 254–61, Nov 1996.

[7] R. Brayton, "Understanding SPFDs: A new method for specifying flexibility," in *Workshop Notes, International Workshop on Logic Synthesis*, (Tahoe City, CA), May 1997.

[8] S. Sinha and R. Brayton, "Implementation and use of SPFDs in optimizing boolean networks," in *Proceedings of the International Conference on Computer-Aided Design*, pp. 103–10, Nov 1998.

[9] S. Khatri, S. Sinha, A. Kuehlmann, R. Brayton, and A. Sangiovanni-Vincentelli, "SPFD based wire removal in a network of PLAs," in *Workshop Notes, International Workshop on Logic Synthesis*, (Tahoe City, CA), May 1999.

[10] S. Sinha and R. K. Brayton, "Robust and efficient spfd computations," in *IWLS*, June 2001.