A Theory of Non-Deterministic Networks

Alan Mishchenko and Robert Brayton
Portland State Univ. and Univ. of California, Berkeley
email: alanmi@ece.psu.edu **and** brayton@eecs.berkeley.edu

Abstract

Both non-determinism and multi-level networks compactly characterize the flexibility allowed in implementing a circuit. A theory for representing and manipulating non-deterministic (ND) networks is introduced. The theory supports the usual network manipulations, done on deterministic binary networks, such as node minimization, elimination, decomposition etc. Three ways to interpret the behavior of an ND network are given. Operations performed on ND networks are analyzed for how they can change behavior under each interpretation. Some common network operations can increase different behaviors and thus might cause the network to violate its external specification. Several methods to correct this problem are proposed.

1 Introduction

A non-deterministic (ND) network is similar to a Boolean network, except that nodes have multi-valued (MV) outputs and are represented by non-deterministic relations. Don't cares are a form of non-determinism, where for some input minterms, the output takes any allowed value. More generally, non-determinism occurs when, for an input minterm, the output takes values from a *subset* of allowed values.

Non-determinism and multi-level networks offer compact ways to represent behaviors. Non-determinism arises naturally in a synthesis setting. A system's specification is given by an ND network or ND automaton. A known or well-defined part of the system may be given also. To be synthesized is an unknown component. The set of all possible behaviors of the unknown component can be derived as an ND relation or automaton, using complementation and composition operations [10].

In logic synthesis, an initial network representation is given along with possibly compatible don't cares at the primary outputs. In some RTL specifications, incomplete specification can be specified at internal nodes. Incomplete specification is interpreted as don't cares, i.e. for some inputs, the output can be any value allowed for the variable. Don't cares are also derived from the network's functionality as observability or satisfiability don't cares. When these concepts are generalized to MV networks, they give rise to non-determinism.

Using the initial specification, a Boolean network is manipulated to obtain a smaller one, which finally is mapped into a set of logic gates for implementation in hardware. An analogous network and set of operations is desired for ND networks. It has been found that the use of such networks can lead to smaller deterministic binary implementations, since the generalization to MV ND networks allows a wider scope for optimization algorithms [5].

In developing a theory for ND networks, some of the classical operations need to be modified to account for the presence of non-determinism, and the different effects caused by non-deterministic nodes need to be clarified. We define three network simulation models (SS, NS, NSC) for ND networks which reduce to the same behavior if the network is deterministic. One of these is equivalent, in the binary case, to simulation with three values, {0,1,X} [1]. We prove results about how the corresponding ND network behaviors can be changed under some common network operations, such as decomposition, substitute, eliminate, collapse, node minimize, and merge [8]. We also study the limits, within which the functionality of a node in an ND network can be changed, without violating the external specification. We show that, for all behaviors, the complete flexibility (CF), as computed in [4], correctly specifies all *deterministic* behavior that can be implemented at the node, but (unlike for NS and NSC) for SS, it incorrectly specifies ND behavior.

The computational procedures for network optimization proposed in [4] for finding a small ND implementation of an ND relation are supplemented with one that finds an *exact minimum* ND representation. Minimum ND representations are never larger than minimum deterministic ones and often much smaller.

In Section 2, we define an ND network and introduce some notation. Section 3 compares three methods for interpreting the behavior of an ND network. Section 4 analyzes the changes in the

behaviors caused by node minimization. Section 5 examines the node elimination process, Section 6 extraction and decomposition, and Section 7 merging. Section 8 discusses the relative merits of the two computationally viable behaviors, NSC and SS. Section 9 provides three methods for minimizing an ND relation, including a new one for finding an exact minimum cover. Section 10 discusses how a circuit can become non-compliant when SS-behavior is used and gives one method to correct this. Section 11 concludes with some longer-term goals for the application of this theory.

Because of space limitations, we do not give any proofs of the theorems, which can be found in [6]. The main purpose of this paper is to present the theory and provide the set of results known so far.

2 ND Networks

An ND network is an acyclic directed graph. A node represents an ND relation between its inputs and its single output. An edge is directed from node *i* to *j* if the relation at node *j* depends syntactically on the variable y_i, associated with node *i*. The output of node *i* can take values from domain $D_i = \{0, \cdots, n_i - 1\}$.

Primary input nodes (PI) have no inputs. Primary output nodes (PO) deliver the functionality of the network to its environment. Single input and output storage element nodes have the next state (NS) variables as inputs, and the present state (PS) variables as outputs. Since this paper is concerned only with the combinational portion of the network, the set (PI, PS) is denoted CI and represented by the vector *X*, and the set (PO, NS) is denoted CO and represented by the vector *Z*. The CI and CO are the combinational inputs and outputs, respectively.

An external specification of a network, $R^{spec}(X,Z)$, is the set of all acceptable (CI, CO) minterm pairs, (m_X, m_Z), such that $R^{spec}(m_X, m_Z) = 1$. These sets of pairs can be in two forms, *compatible* (or output-symmetric) and *free*. A free external specification has no restrictions on which pairs are allowed, except that it is "well-defined".

Definition: *A relation is* well-defined *if for each input minterm, there exists at least one output minterm in the relation.*

A free specification in the binary case has been called a Boolean relation [9]. A compatible specification has an additional "symmetry" restriction.

Definition: *A relation* $R(X,Z)$ *is* output-symmetric *when the following is true. If input* m_X *can produce a value for output* z_i *in a set* S_i *and* m_X *can produce a value for output* z_j *in a set* S_j*, then for any choice* $a \in S_i$ *and* $b \in S_j$ *there exists* m_Z *such that* $m_{z_i} = a$, $m_{z_j} = b$, *and* $(m_X, m_Z) \in R(X,Z)$.

> *Example*. Consider a network with two binary outputs, z_1 and z_2. Suppose, for some minterm, the values the outputs can take are {00, 01}. The relations $R(X, Z)$ is output-symmetric in this minterm, because $z_1 \in \{0\}$, $z_2 \in \{0,1\}$, and every combination from the set $\{0\}\times\{0,1\}=\{00, 01\}$ belongs to the relation. If the same outputs were to take values {00, 01, 11} in this minterm, it would not be output-symmetric because $z_1 \in \{0,1\}$, $z_2 \in \{0,1\}$, and there exists a combination {11} in $\{0,1\}\times\{0,1\}=\{00, 01, 10, 11\}$, which does not belong to the relation.

Output symmetry has been used to define "compatible" external don't cares in binary networks. The advantage is that the choice of value made at one output can be made independently of the choice made at other outputs. For a free specification, once a choice is made at one output, the choices possible at another output may be restricted.

Definition: *The* B-behavior *of an ND network is the set of all input-output pairs that can be simulated using the simulation of type B.*

An ND relation giving the functionality of a node in a network can be represented by the characteristic function relating inputs and outputs. For a single output relation, it is more efficient to store a set of deterministic binary functions, the i^{th} of which is 1 if some fanin minterm can produce value *i* at the output. Such functions are called the *i*-sets of the multi-valued (MV) relation at the node. A smaller representation can obtained by designating one of the *i*-sets as the default, which is not represented explicitly, but is assumed to be the complement of the union of the other *i*-sets. However, general ND relations cannot be represented this way. For example, for binary relations, the default is either the 0-set or the 1-set; thus only deterministic binary relations can be represented if a default value is used.

The notation $R^{spec}(X,Z)$ will be used to represent the specification of the network. An ND network *conforms* to or *complies* with its external specification if when the network is simulated with input m_X, any output minterm m_Z satisfies $(m_X, m_Z) \in R^{spec}(X,Z)$. We will define three types of simulations for an

ND network {NS, NSC, SS}, all of which agree with the usual one if the network is deterministic. Compatible specifications have the advantage that a set of individual single-output relations, one for each CO, can be used to represent them.

The behavior of a network with respect to the simulation of type B is represented as $R^B(X,Z)$. We often use the arguments of a relation to identify it, e.g. $R(X,Y_j)$ and $R(Y_j,y_j)$ denote different relations even though eac is named *R*. The relation at a node *j* in the network is denoted $R_j(Y_j,y_j)$ where Y_j is the set of fanin variables and y_j is the single output variable of the node.

Binary-output multi-valued-input functions can be minimized using a program like Espresso-MV. This results in a minimized MV sum-of-products (SOP) expression for the function. A product is the conjunction of multi-valued literals. A literal of MV variable *y*, for example, $y^{\{0,3,5\}}$, is the binary function, which is 1 if *y* takes any value from the set $\{0,3,5\}$ and 0 otherwise.

3 Behaviors of ND Networks

Different interpretations, associated with different ways to simulate an ND network, can be used for its behavior. All of these yield the same behavior if the network is deterministic. The interpretations discussed are listed in the order of increasing amount of behavior:

1. Behavior by normal simulation (NS-behavior).
2. Behavior by normal simulation made compatible (NSC-behavior).
3. Behavior by set simulation (SS-behavior).

We define each and discuss their relative merits.

It is important in manipulating a network to consistently use only one interpretation of a network's behavior, because, during a network's manipulation, its behavior is periodically compared with the specification. It is possible that an ND network satisfies its specification under one interpretation but not another. During an optimization sequence, it is common to refine and optimize the network. In many applications, even though the final network must be deterministic (for hardware implementation) switching between different interpretations of behavior might lead to a final network that does not satisfy the specifications.

3.1 Behavior by Normal Simulation (NS)

NS is the most intuitive simulation. It proceeds in topological order. Each node non-deterministically selects one of its allowed values and transmits this to all of its fanouts. Even though it is easy to obtain single pairs (m_X,m_Z), it is difficult to obtain all pairs, which makes it the most complex simulation method computationally.

The complete NS-behavior can be given by the MV Boolean relation,

$$R^{NS}(X,Z) = \underset{i\in \text{internal nodes}}{\exists_{y_i}} \prod_j R_j(Y_j,y_j), \tag{1}$$

This is not the same as eliminating nodes in some order since here, each time an existential quantification on an internal node variable is done, the effect is as if the fanout nodes are merged into a single multi-output node, and a Boolean relation is given for this node. After all quantifications are done, the MV Boolean relation is obtained for the entire circuit. A pair (m_X,m_Z) is in an MV Boolean relation $R^{NS}(X,Z)$ precisely if there is a choice at each node such that if m_X is given at the CI, and at each node, the choice is propagated, then the vector m_Z can be seen at the CO.

A possibly more efficient method for computing $R^{NS}(X,Z)$ is to use early quantification of a conjunctive relation as is done in formal verification applications. However, this computation is still problematic since the final resulting relation connects all CI with all CO. In contrast, the other two types of behaviors can be represented by *N* relations, each connecting the CI with one CO, $z_k, k\in\{1,\cdots,N\}$.

3.1.1 Input Determinization (ID)

An approach used for binary circuits with incompletely specified internal nodes introduces a new binary variable for each incompletely specified node. This variable is used to determinize the node; then collapsing the resulting deterministic network leads to a representation of the behavior. This gives

for each output a well-defined deterministic function depending on the CI, *X*, and the set of parameters (or pseudo-inputs), *P*, used to determinize the internal nodes.

We generalize this to ND MV networks. An MV pseudo-input variable p_i is introduced at each ND node y_i, where the range of p_i is the range of y_i. The relation at the node is made deterministic using p_i. Such a determinization can be computed using the compatible projection operator, m_y, defined by Lin et al [3]. This operator selects for each input minterm a single representative value, say the least value allowed. The computation is $\tilde{R}_i(Y_i, p_i, y_i) = R_i(Y_i, y_i)(p_i = y_i) + \exists_{\hat{y}}[\overline{R_i}(Y_i, \hat{y})(p_i = \hat{y})]m_{y_i}(R_i(Y_i, y_i))$. The second term on the right-hand-side finds all the values not allowed and associates these with the value selected by the compatible projection operator. The relation is deterministic and well defined.

Example: Let the range of y_i be {0,1,2,3} and let {0,2} be the output values for a fanin minterm m_{Y_i}; thus currently $(m_{Y_i}, y_i^{\{0,2\}}) \in R_i(Y_i, y_i)$. This relation is determinized by adding p_i to control the ND choice, replacing $(m_{Y_i}, y_i^{\{0,2\}})$ with $(m_{Y_i}, p_i^{\{0\}} y_i^{\{0\}} + p_i^{\{2\}} y_i^{\{2\}} + p_i^{\{1,3\}} y_i^{\{0\}})$. The last term is added to make the relation well-defined for all values of p_i.

After this is done for all ND nodes and the circuit is collapsed, at each output, z_k, its global MV-function is obtained in terms of the CI *X* and the set of pseudo-inputs *P*, $z_k = G_k(X, P)$. This is precisely what can be simulated with normal simulation mode, since at each internal node, the output value is controlled by p_i, and this single value is propagated to all fanouts. If the above equations are combined to give an MV Boolean relation at the outputs, the result is $R^{NS}(X, Z) = \exists_P \prod_{k=1}^{N} (z_k = G_k(X, P))$, the same as in Equation (1).

3.2 Behavior by NS Compatibility (NSC)

Each output can be converted into a separate relation to obtain a set of compatible relations,

$$R_k^{NSC}(X, z_k) = \exists_P (z_k = G_k(X, P)) \tag{2}$$

for all outputs, $k = 1, \cdots, N$. This increases the behavior since the existential quantification is done independently at each output. Equation (2) represents the second type of behavior, called *NSC-behavior*, since it represents the operation of making the NS-behavior compatible, or output-symmetric. Thus NSC is the same as NS, except that each output is treated independently. If there is only one output, then NS and NSC are the same.

Theorem 1: *The NSC-behavior is equivalent to not introducing pseudo-inputs, but by eliminating the nodes in the network in* reverse *topological order.*

Thus there is no need for introducing pseudo-input parameters if NSC-behavior is used. Collapsing in reverse topological order yields the smallest set of output symmetric relations that *contain* the NS behavior of the network.

A way to understand NSC behavior is to consider each CO output cone as being cut away from the network and then doing NS simulation on that cone. The set of values that an internal node can have during the NSC simulation of an input vector m_X is the union of all values obtained by normal simulation of each output separately. Hence, this set is exactly the set of values that node can assume by normal simulation of the network. However, note that NSC simulation of the whole network has the effect that different fanouts of an internal node *i* can have different values during the same simulation cycle *if* these values go along paths to different CO. The image that a set of fanins to a node can have under input m_X is the same as with NS. We will use this later in computing the flexibility of a node where we need to know the set of values that a set of nodes can have under different forms of simulation.

Theorem 2: *The NS and NSC behaviors of a network are not changed by eliminating any or all deterministic nodes in any order.*

3.3 Behavior by Set Simulation (SS)

Set simulation is performed as follows. The CI are assigned values, m_X. If a node has all its fanins assigned a set of values, the value of the output of the node is the *set* of all values possible for that node given its fanin sets. For example, each input has a set of values, S_{i_k}. We evaluate the output of a node *i* as the set, $S_i = \{v \mid R_i(V_i, v) = 1, V_i \in S_{i_1} \times S_{i_2} \times \cdots \times S_{i_{|y_i|}}\}$. Each edge $i \to j$ is assigned the set S_i. When all

nodes have been computed, the cross product of the CO sets forms the set of minterms { m_z } allowed for m_x . Such a pair (m_x, m_z) is in the SS-behavior of network.[1]

This is the method that has been implemented currently in MVSIS [7] and for which we have some experimental results. We will show that a key operation, elimination of an internal node, cannot increase the SS-behavior of an ND network. Note that the SS-behavior of a network is a compatible (output-symmetric) relation and hence can be represented by a set of *i*-set covers, one set for each output. Like NSC behavior, a key advantage is that the network can be manipulated as a network of single output MV nodes. In contrast, NS-behavior must deal with multi-output nodes and MV Boolean relations at these nodes or must introduce possibly many pseudo-inputs.

3.3.1 Binary Interpretation

A good way to understand SS-behavior is to consider the ND network as a set of binary deterministic nodes, one for each *i*-set of each MV node in the network. For example, consider node *j*, which has 3 values, and thus a 0-set, a 1-set and a 2-set; each is represented by an MV-input binary-output SOP. However, each internal MV signal and each CO is replaced by a set of binary signals and each corresponding literal in its fanouts is converted to a sum of binary literals, e.g. $y^{\{1,3,5\}} = b_1^y + b_3^y + b_5^y$ where b_j^y is the binary signal controlled by the j^{th} *i*-set of *y*. This network is deterministic and can be manipulated like any other deterministic network. The only MV signals are the CI.

Lemma 1: *The SS-behavior of an ND network can be obtained by treating each i-set as a separate binary function and eliminating all internal nodes in the network in any order.*

3.3.2 Eliminating in Topological Order

Collapsing (eliminating all internal nodes) can also be done in the usual way if the elimination process is done in topological order.

Lemma 2: *The SS-behavior of a network is exactly that obtained by eliminating the nodes of the network in topological order.*

The effect that an ND node can have on the behavior of this type of simulation is controlled by the set of all paths from the ND node to the CO outputs. Each time a partial path branches, additional independent "copies" of the ND node are made. This provides a good intuitive way to think about SS behavior.

3.4 Comparison

Each of the three types of behaviors, NS, NSC and SS will be examined, relative to a) the ease with which the network is kept compliant when a network operation is performed, b) the ease of computation during network manipulations, c) the ease of representation of the network, and d) its relative optimization potential.

A network's specification gives an upper bound on the network's allowed behavior. The specification can be output-symmetric (compatible relations - similar to don't cares) or free (a Boolean relation relating all outputs at once). Operations on an ND network can change its behavior, no matter how behavior is defined. An increase in behavior is allowed as long as it is still contained in the specification. The node minimization operation (see Section 4) directly uses the specification to test how much the network behavior can be increased without violating the specification. Then, an ND relation at a node is computed to describe a flexibility allowed in implementing the node. Different interpretations of the network behavior will affect the flexibilities allowed. Minimization of these flexibilities is done in order to choose a sub-relation with a small representation for the node. An ND relation is always a smallest representation.

The second aspect concerns how the network specification is represented and compared. Output symmetric specifications can be stored individually, at each output, as a set of binary-output *i*-set functions. Other specifications may require a single Boolean relation. Although Boolean relations can

[1] This is similar to what is done in X valued simulation, or 3-valued simulation where values 0,1,X are propagated. X stands for set {0,1}. The truth table for each gate is made conservative; a logic function produces an X only if the set of inputs with known values (non X) is not a controlling set of values.

be determinized by using pseudo-inputs and stored individually at each output, many pseudo-inputs might have to be introduced making this representation cumbersome.

The third aspect is the ease with which the computations involved in the network manipulations can be done. The three behaviors differ in this respect. The most efficient seems to be SS-behavior since this behavior is related to collapsing in topological order. This allows the building of global BDDs where only CI variables are needed. NSC is also relatively easy because reverse topological order can be used, but building global BDDs is more difficult. NS-behavior requires use of Boolean relations everywhere or the introduction of pseudo-inputs, *P*.

We saw that NSC-behavior is defined in terms of NS behavior: $R_k^{NSC}(X, z_k) = \exists_P R_k^{NS}(X, P, z_k)$. Thus NS behavior is contained in NSC-behavior. Also NSC is a subset of the SS-behavior, since in NSC, various copies of ND relations, which lead to the same CO, are kept correlated (by the parameters *P*) during the collapsing process. On the other hand, with SS, all correlations between different fanouts of an ND node are lost when the node is eliminated. Defining $R^{NSC}(X, Z) \equiv \prod_k R_k^{NSC}(X, z_k)$, we have

$$R^{NS}(X, Z) \subseteq R^{NSC}(X, Z) \subseteq R^{SS}(X, Z). \quad (3)$$

In Section 4, it is shown that this ordering has the reverse effect on optimization potential.

All behaviors can be seen from a common point of view of quantifying internal variables.

1. NS: Multiply all relations together and then existentially quantify all internals variables: $\exists_{y_i \text{ internal}} \prod_j R_j(Y_j, y_j)$.
2. NSC: Intermix the products and existential quantifications so that the quantifications are done in reverse topological order. The same variable may be quantified several times.
3. SS: Intermix the products and existential quantifications so that the quantifications are done in topological order. The same variable may be quantified several times.

4 Node Minimization

The node minimization process consists of deriving a flexibility for the node being minimized, and then replacing the current representation at the node with a smaller one contained in the flexibility. We first examine how the flexibility is computed, and then the possible changes, in the various types of behaviors of the network, that can happen when the current representation is replaced. The minimization of non-deterministic relations is the subject of Section 9.

4.1 Deriving Complete Flexibilities

The computation of the complete flexibility, CF, at a node y_i can be described generically for all the behaviors $B \in \{NS, NSC, SS\}$.

Cut the network at y_i and consider a new network (the *cut* network) with y_i as a new primary input. Require that the B-behavior of the cut network, $R^B(X, y_i, Z)$, comply with the network specification $R^{spec}(X, Z)$,

$$R^B(X, y_i) = \forall_Z (R^B(X, y_i, Z) \Rightarrow R^{spec}(X, Z)). \quad (4)$$

This is called the Observability Partial Care (**OPC**) for the node and is analogous to the observability don't care set for a node in a binary network.

Theorem 3: *$R^B(X, y_i)$ is maximal, i.e. if any deterministic function $y_i = f_i(X)$ is used to replace node i, such that $(y_i = f_i(X)) \not\subseteq R^B(X, y_i)$, then $\widetilde{R}^B(X, Z) \not\subseteq R^{spec}(X, Z)$, i.e. the B-behavior of the new network $\widetilde{N}$ does not satisfy the specification.*

Note that by Equation (3),

$$R^{SS}(X, y_i) \subseteq R^{NSC}(X, y_i) \subseteq R^{NS}(X, y_i). \quad (5)$$

Next we bring in the "satisfiability don't cares" (**SDC**) to derive a local "complete" flexibility (CF). Define the relation between CI minterms and vectors of values that the fanin variables Y_i of y_i can take during the B-simulation of the network as $M^B(X, Y_i)$. Then the CF is computed by the formula

$$R^B(Y_i, y_i) = \forall_X (M^B(X, Y_i) \Rightarrow R^B(X, y_i)). \quad (6)$$

Because of Equations (3) and (5) ,

$$R^{SS}(Y_i, y_i) \subseteq R^{NSC}(Y_i, y_i) \subseteq R^{NS}(Y_i, y_i) . \quad (7)$$

In general, the CFs, $R^B(Y_i, y_i)$ for $B \in \{NS, NSC, SS\}$, are ND relations and since the current relation, $R_i(Y_i, y_i)$, is well defined and $R_i(Y_i, y_i) \subseteq R^B(Y_i, y_i)$, then so is $R^B(Y_i, y_i)$.

We will show that all CFs have the property that any well-defined *deterministic* relation contained in them can be used to replace the current relation at node *i* without causing the B-behavior of the network to violate the specification. In Section 4.2 we will show that NS and NSC allow the current relation to be replaced by any well-defined *ND relation* contained in it; unfortunately, this does not hold for SS-behavior.

The SDC are added in those cases where a fanin Y_j minterm, m_{Y_j}, is not associated (through B-behavior) with any m_X . In this case, the node at y_i on input m_{Y_j} can be allowed to produce any value in the range of y_j, and thus m_{Y_j} is a don't care. Since in general, $M^B(X, Y_j)$ is ND, $M^B(m_X, Y_j)$ can produce a set $A = \{m_{Y_j}\}$ and similarly $R^B(m_X, y_j)$ can produce a set $V = \{m_{y_j}\}$ of values for y_j. Consider the double complemented RHS, $R^B(Y_j, y_j) = \overline{\exists_X M^B(X, Y_j) \cap \overline{R^B}(X, y_j)}$. The expressions under the complement symbol relate, for m_X, those $m_{Y_j} \in A$ with those values in $\overline{V}$. Then all those pairs (m_{Y_j}, v_{y_j}) not so related are put in $R^B(Y_j, y_j)$. Thus increasing $M^B(X, Y_j)$ (e.g. by making some nodes ND, or moving from NS to NSC to SS behaviors) causes two effects. First, it decreases the SDC, and second it puts more values in the sets *A* thereby increasing the pairing of values with $\overline{V}$ and thus reducing the pairs in the complement.

4.2 Computing with Different Flexibilities

4.2.1 NS Behavior

Theorem 4: *If a well-defined ND relation contained in* $R^{NS}(X, y_j)$ *is inserted at node j, the new network* $\widetilde{N}$ *remains compliant, i.e.* $\widetilde{R}^{NS}(X, Z) \subseteq R^{spec}(X, Z)$.

Theorem 5: *If a well-defined ND sub-relation contained in* $R^{NS}(Y_j, y_j)$ *is inserted at node j, the new network,* $\widetilde{N}$ *, remains compliant, i.e.* $\widetilde{R}^{NS}(X, Z) \subseteq R^{spec}(X, Z)$.

The computation of $M^{NS}(X, Y_j)$ can be done using a method similar to the image computation procedure of [2], where output cofactoring is done.

After $R^{NS}(Y_i, y_i)$ is minimized, a new relation is inserted at node *i*. If the minimized relation is ND, then a new pseudo-input needs to be introduced. As the manipulation continues, the set of pseudo-inputs may be different compared to the original network. Checking that the new network conforms to its specification requires verifying that $\widetilde{R}^{NS}(X, Z) \subseteq R^{spec}(X, Z)$, where $\widetilde{R}^{NS}(X, Z) \equiv \exists_Q \prod_k (z_k = R_k(X, Q))$. Thus the verification problem is a difficult one in which two Boolean relations, each relating all CI to all CO, are compared.

4.2.2 NSC Behavior

NSC-behavior is computationally more viable than NS; since it is equivalent to collapsing in reverse topological order, there is no need for introducing pseudo-inputs. Alternatively, it is possible to perform collapsing in direct topological order while performing some additional manipulations. For each ND node with reconvergent fanout, the *i*-sets in the fanout cone are computed in terms of CI and a temporary MV variable representing each ND node. After the computation reaches the meeting point of the reconvergent paths (called the *assembly point*) of a temporary variable, it is possible to get rid of it by substituting the *i*-sets of the node expressed in terms of CI. This way, the temporary variable synchronizes the behavior of the node along the reconvergent paths and makes it compatible with the NSC model.

However, in computing and using the flexibility, a comparison with the specification $R^{spec}(X, Z)$ is required. If this is given as a Boolean relation, and NSC behavior is used, it is best to project the

specification to an output symmetric form a priori, by computing $R^{spec}(X, z_k)$ such that $\prod_k R^{spec}(X, z_k) \subseteq R^{spec}(X, Z)$. The computation for the OPC at a node then becomes,

$$R^{NSC}(X, y_j) = \prod_{k=1}^{N} (\forall_{z_k} (R_k^{NSC}(X, y_j, z_k) \Rightarrow R^{spec}(X, z_k))) .$$

Theorem 6: *If a well-defined ND relation contained in* $R^{NSC}(X, y_j)$ *is inserted at node j, the new network* $\widetilde{N}$ *remains compliant, i.e.* $\widetilde{R}^{NSC}(X, z_k) \subseteq R^{spec}(X, z_k)$ *for all* $1 \le k \le N$.

However, it still remains to be proved that using an ND sub-relation of the local CF will not violate the specification.

Theorem 7: *If a well-defined ND relation contained in* $R^{NSC}(Y_j, y_j)$ *is inserted at node j, the new network* $\widetilde{N}$ *remains compliant, i.e.* $\widetilde{R}^{NSC}(X, z_k) \subseteq R^{spec}(X, z_k)$ *for all* $1 \le k \le N$.

Note that in computing $R^{NSC}(Y_j, y_j)$, we can use that $M^{NS}(X, Y_j) = M^{NSC}(X, Y_j)$.

4.2.3 SS Behavior

Theorem 8: *If a well-defined* deterministic *relation contained in* $R^{SS}(X, y_j)$ *is inserted at the node j, the new network* $\widetilde{N}$ *remains compliant, i.e.* $\widetilde{R}^{SS}(X, Z) \subseteq R^{spec}(X, Z)$.

Unlike the other behaviors, with SS, we cannot guarantee conformance if any ND sub-relation of $R^{SS}(X, y_j)$ is allowed at node *j*. The SS-behavior can increase if an ND subset is used and the node has multiple paths to one of the CO (reconvergent fanout). The following network, structurally similar to that of Figure 1 below, gives an example.

*Example***:** Consider the following deterministic network:

$$y^{\{1\}} = a^{\{1\}} b^{\{0\}} + a^{\{0\}} b^{\{1\}}$$
$$v^{\{1\}} = a^{\{1\}} + y^{\{1\}}$$
$$w^{\{1\}} = b^{\{1\}} y^{\{0\}}$$
$$z^{\{1\}} = v^{\{1\}} w^{\{1\}} + v^{\{0\}} w^{\{0\}}$$

Let this network also serve as the specification. Since the network is deterministic, the specification can be obtained by collapsing the network. It is easy to show that the $R^{SS}(a, b, y) = 1$ (since *y* is redundant). If we insert this ND relation in for *y* and eliminate *y, v, w* in that order we get

$$z^{\{0\}} = 1 = a^{\{0,1\}} b^{\{0,1\}}$$
$$z^{\{1\}} = 1 = a^{\{0,1\}} b^{\{0,1\}} ,$$

i.e. $R^{SS}(a, b, z) = 1$. This has a behavior, for example $a = 1$, $b = 1$, $z = 0$, which was not in the original network specification,

$$z^{\{1\}} = a^{\{1\}} b^{\{1\}} + a^{\{0\}} b^{\{0\}}$$
$$z^{\{0\}} = a^{\{1\}} b^{\{0\}} + a^{\{1\}} b^{\{0\}}$$

Theorem 9: *If any well-defined* deterministic *sub-relation of* $R^{SS}(Y_j, y_j)$ *is put at j, the new network remains compliant.*

The same example after Theorem 8 demonstrates that an ND sub-relation of the local flexibility cannot be used in general.

5 Elimination

Elimination is the process of substituting the relation of a node into all the relations of its fanouts. Substitution of a relation into another is defined as follows. Let the relation at node *i* be $R_i(Y_i, y_i)$ and suppose *k* is a fanout of *i* with a similar relation R_k. Then, substitution of *i* into *k* yields the new relation at *k*, $\exists_{y_i} R_i(Y_i, y_i) R_k(Y_k, y_k)$, which replaces R_k, where y_i is one of the Y_k. After R_i has been substituted into all its fanouts, it can be removed from the network. We remark that this kind of elimination is to be distinguished from first determinizing all nodes and then existentially quantifying out all internal

variables. The latter generates correlations between the outputs of the fanout nodes, while the former keeps the fanout nodes independent.

Theorem 10: *Eliminating a node can increase the NS or NSC behaviors of a network only if the node is ND and has more than one fanout.*

If there is more than one fanout, then the effect is as if a new copy of the node is assigned to each fanout. Each new fanout node has the behavior of the combined old node and the copied ND node. Since each copy acts independently, then each can select a different value in the same simulation cycle. This can lead to an increase of the network's NS or NSC behavior, i.e. these behaviors of the new network with the node eliminated can be a strict superset of the corresponding behaviors of the original network. Further elimination may cause additional increases in these behaviors. In fact, if the elimination is done in topological order and the network is collapsed to single nodes for each output, then the NS and NSC behaviors of the collapsed network will coincide with SS-behavior of the original network.

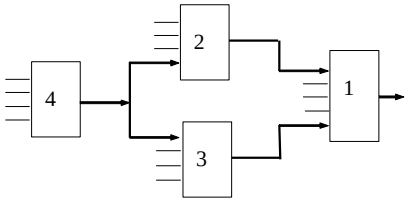
As already discussed, we can manipulate NS-behavior by determinizing the network first. Suppose a fanout *j* is given by the function $R_j(Y_j, p_j, y_j)$ and similarly for node *i*. Then the new function at *j* after node *i* is eliminated is $\widetilde{R_j}(\widetilde{Y_j}, p_i, p_j, y_j) = \exists_{y_i} R_i(Y_i, p_i, y_i) R_j(Y_j, p_j, y_j)$ and thus it has two pseudo-inputs. Existentially quantifying out p_i and p_j leads to the same result as elimination without determinization. Note that p_i and p_j cannot be replaced with a single pseudo-input because this would cause any correlation between the different fanouts of node *i* (through the parameter p_i) to be lost. Thus, in general, as the network is manipulated, the nodes in the network will depend on an increasing number of pseudo-inputs. Although the determinized elimination process done this way will not increase the NS-behavior of the network, it is more expensive computationally.

Theorem 11: *Eliminating a node can increase a network's NSC behavior if and only if the node is ND and has reconvergent fanout.*

The reason for initially considering the SS-behavior of a network is because elimination effectively substitutes a copy of the eliminated node into each fanout. Each copy acts independently of the other copies and is thus like broadcasting independent *sets* of values to the fanouts. Thus elimination does not increase the SS-behavior of a network. However, elimination may decrease SS-behavior if several nodes are eliminated and a proper order is not followed.

*Example***:** Consider the network in Figure 1. There is reconvergent fanout from node 4. Suppose 2 and 3 are eliminated first and then node 4. After eliminating 2 and 3, 4 has only one fanout, so intuitively only one copy is made to be substituted. In contrast, if 4 is eliminated first, then two copies of 4 are made in 2 and 3.

Figure 1. An ND network.



Eliminating 2 and 3 first has the same effect as if the network is simulated with a single value on the fanouts of 4. Thus we could lose the behavior where the two fanouts of 4 can have different values (assuming that 4 is non-deterministic). This can be seen by observing how the relation at 1 changes under different orders of elimination. For the elimination order 4 3 2, we get $(\exists y_2(\exists y_3(\exists y_4 R_4 R_3) R_1)(\exists y_4 R_4 R_2))$, and for order 3 2 4, we get $(\exists y_4(\exists y_2(\exists y_3 R_3 R_1) R_2) R_4)$. The difference is that in the first expression, there are two existential quantifications on y_4 which are uncorrelated.

The following theorem states sufficient conditions when a node's elimination preserves the SS-behavior. The global SS-behavior of a node *A* is defined to be that obtained by eliminating all nodes in the TFI of *A* in topological order.

Theorem 12: *Eliminating a node A does not change the SS-behavior of the network if*

1. *there are no ND nodes in the TFI of A, or*
2. *for every fanout of A, its fanins are disjoint from the fanins of A.*

Note that collapsing in topological order is not necessary to preserve SS-behavior, since in the above example we could eliminate in the order of 3 4 2 1 because this would lead to $(\exists y_2(\exists y_4(\exists y_3 R_3 R_1)R_4)(\exists y_4 R_4 R_2))$ which can be obtained from the topological elimination result.

Corollary 1: *Eliminating a node can never increase the SS-behavior of a network.*

Corollary 2: *Elimination a node A can decrease the SS-behavior of a network only if*

1. *A has an ND node B in its TFI and*
2. *a fanin of A in the TFO of B is also a fanin of a fanout of A (see Figure 2).*

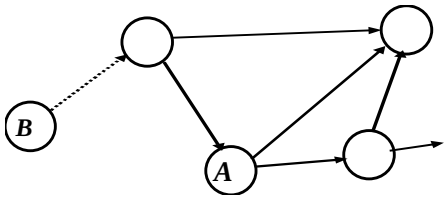


Figure 2. Topology where number of paths from *B* can decrease.

6 Extraction and Decomposition

Extraction and decomposition are similar; the latter operates on a single node at a time, while the former operates on a set of nodes. With decomposition, a new node (divisor) is created, which has only a single fanout; with extraction there are at least two fanouts. The objective is the same, to find a good divisor. There are two forms of extraction, disjoint and non-disjoint. Extraction is disjoint if the fanins of the new node are not also fanins of the fanouts of the new node.

Theorem 13: *Extraction and decomposition cannot increase the NS and NSC behaviors of an ND network.*

Theorem 14. *The SS-behavior of a network is not changed if a node extraction is disjoint, or there are no ND nodes in the TFI of the extracted node.*

A non-disjoint extraction will increase the number of paths. As an example, consider the network in Figure 3.

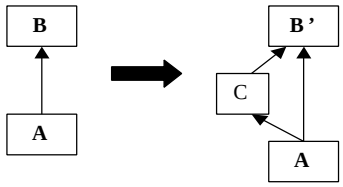


Figure 3. Non-disjoint extraction from B

B has been decomposed into a non-disjoint extraction because the inputs of C are not disjoint from the inputs of B'. Thus the number of paths from A to B has increased. If A is non-deterministic or there is an ND node in TFI(A), according to Theorem 12, the SS-behavior could increase.

7 Merging

Merging is the process of combining two or more nodes (the merging set) into a single node with more values [7]. A constraint on the merging set is that after merging, the network should remain acyclic. The *i*-sets of the new node are composed of intersections of the *i*-sets of the merging set.

Example: Suppose two nodes are to be merged with ranges 3 and 5 respectively. Then the 0-set of the new node is the intersection of the 0-sets of the two relations, the 1-set is the intersection of the 0-set and the 1-set, the 2-set the intersection of the 0-set and 2-set, etc, and the 14-set the intersection of the 2-set and the 4-set.

The second step of merging involves substituting the new node into the union of the fanouts of the merging set by replacing some literals of each cube in the *i*-set covers of a fanout by a single literal of the new variable.

Example: In the above example, if a fanout cube in some *i*-set involves the product $x^{\{0,2\}}y^{\{1,3,4\}}$ (*x* and *y* are three-valued and five-valued MV variables, respectively), this cube is replaced by the literal of the new 15-valued variable, say *z*, $z^{\{1,3,4,11,13,14\}}$. A cube with the literal $x^{\{1\}}$ but no *y* is replaced by $z^{\{5,6,7,8,9\}}$ since the absence of *y* implies all values of *y*.

Thus the number of *i*-set cubes in the fanouts cannot increase and might be decreased by making the resulting *i*-sets prime and irredundant.

Example: An example of the reduction is given by the binary EXOR gate with inputs *x* and *y*: $x^{\{0\}}y^{\{1\}} + x^{\{1\}}y^{\{0\}}$. It has two cubes and four literals. If *x* and *y* are merged into a single node *z*, the MV-SOP of the gate becomes one cube with one literal, $z^{\{1,2\}}$.

Theorem 15. *The merging of two or more nodes cannot change the NS or NSC behaviors and can't increase the SS behavior of the network.*

8 Two Computationally Viable Theories

There seem to be two computationally viable choices for a theory of non-deterministic networks, NSC and SS behavior. Comparison leads to the following statements.

1. Both methods give output-symmetric relations at the outputs.
2. The computational process for SS seems simpler since collapsing in topological order leads to an efficient method for building global BDDs.
3. Both methods lead to network operations that are similar to those used for binary networks.
4. No pseudo-inputs need to be introduced for either method to control the choice at a node.
5. As the network is manipulated, both allow changes in behavior to be analyzed and easily understood.
6. NSC leads to the smallest increase in behavior over the more intuitive NS behavior. SS gives the largest increase.
7. NSC can provide more flexibility at a node.
8. Both methods may cause the network to become non-compliant. However, the non-compliance is more easily controlled in NSC, since only one operation, elimination, can cause non-compliance, and only under the case that a ND node with reconvergent fanout is eliminated.

The theory based on SS-behavior is currently implemented in MVSIS. Although it suffers from a higher probability that the network will become non-compliant during network manipulation, no difficulties have been encountered because of this. We are currently exploring methods which would allow efficient computations when NSC behavior is used.

Table 1 compares NSC behavior with SS-behavior in terms of possible non-compliance of the network after the corresponding operation.

Operation	**SS-behavior**	**NSC-behavior**
elimination	compliance	non-compliance
extraction	non-compliance	compliance
node minimization	non-compliance	compliance
node flexibility	less	more
merging	can't increase	can't change

Table 1. Comparing two computationally viable theories

9 Minimizing an ND Relation

Given a flexibility (ND relation) at a node, the goal is to construct the smallest (in terms of the total number of *i*-set cubes) well-defined relation contained in it.

Definition: *The* i-set *of an ND relation,* $R(Y,y)$*, is the set of minterms that can produce i, i.e.* $R_{y^{\{i\}}}(Y,y)$*, the cofactor of* $R(Y,y)$ *with respect to the literal* $y^{\{i\}}$*. The* essential i-set *is the minterms that can only produce i.*

9.1 Deterministic MV-SOP Minimization

The computation starts by ordering the *i*-sets heuristically. Then in increasing order, for each *i*, we extract the minterms of its *i*-set not yet covered by *i*-set covers already computed. The minimized SOP for the *i*-set is computed by a call to an SOP minimizer using the part of the *i*-set not yet covered by preceding *i*-set covers and which aren't in the remaining *i*-sets, as the on-set and the rest of the remaining uncovered terms in the *i*-set as the don't-care set. Since the remaining *i*-sets computed in each step do not overlap with the covers selected for the previous *i*-sets, the resulting MV-SOP is disjoint and, therefore, deterministic.

9.2 Heuristic ND MV-SOP Minimization

The computation proceeds in two steps. First, the essential part of each *i*-set is minimized using the rest of that *i*-set as don't-care. Computed this way, the *i*-sets are allowed to overlap resulting in a non-deterministic cover. This cover cannot be larger than the deterministic cover if we use the same ordering of the *i*-sets. If at this point, all minterms are covered, the algorithm has computed the exact minimum cover (provided that the MV-input binary-output covers for each *i*-set have been minimized exactly). Surprisingly, in our experience, this has been the case for about 90% of MV-SOP minimization problems that arise in the simplification of non-deterministic networks implemented in MVSIS.

If there are remaining uncovered minterms, then each must be associated with at least two *i*-sets. If there is at least one output value common to all remaining minterms, then all these are added to the common value. This situation has occurred in about 9% of the cases, leaving only about 1% to be processed further. Finally, a simple greedy approach is taken. Considering values one by one in some heuristic order, as many minterms as possible are added to each of the successive *i*-sets.

9.3 Exact ND MV-SOP Minimization

An exact minimum cover of a relation can be found as follows.

Procedure: For each i-set, generate its set of primes. Form a combined unate covering problem where the minterms to be covered is the entire input space and the union of primes of all *i*-sets is the set of covering cubes. Solve for a minimum cover. Each prime chosen in the minimum cover is put into its appropriate *i*-set to form the minimum *i*-set covers.

Theorem 16: *The above procedure gives a set of i-set covers which has the minimum number of cubes. Each i-set cover is prime and irredundant.*

A common situation is that a default *i*-set is used which is never represented since it can be obtained by complementing all other *i*-sets.[2] The problem is to choose the default so that the remaining *i*-sets can be covered with the minimum number of cubes. This can be solved as follows.

Procedure: For each *i*, form the covering problem as in Theorem 16, *except* force the primes of the *i*-set to be in the cover. The measure of the solution obtained is the number of cubes in the cover, not counting those in the i^{th} set. Do this for each *i* and choose the one, *k*, that leads to the smallest measure, to be the default and the cubes of the k^{th} covering problem, with the primes of the k^{th} *i*-set deleted, as the final covers.

Theorem 17: *The above procedure leads to the minimum set of covers when the default cover is not counted.*

In some cases, it is desired to reduce the number of values produced by the node to a minimum (value reducing minimization). This can be done by solving the following covering problem.

Procedure: Create a column for each *i*-set, which has a 1 in it if the minterm is in the *i*-set. A minimum cover gives a minimum set of values. Now restrict to these *i*-sets and find a minimum set of *i*-set covers for these using either Theorem 16 or Theorem 17. It might happen that there are several sets of minimum values. It that case, the set that leads to the minimum *i*-set covers is required. Thus all minimum value subsets can be found, and for each, the minimum *i*-set covering problem is solved. This is done by simply deleting those primes not in the value set being solved and using the procedure of either Theorem 16 of Theorem 17. The set of all minimum value sets can be obtained by finding the complement of the unate function associated with the covering matrix and choosing those primes with the least number of literals.

[2] In binary logic synthesis, we usually implement only the onset of a node; if the offset is required, it is produced by an invertor.

10 Managing Non-Compliance When Using SS

A goal in manipulating an ND network is often to derive an efficient network representation *contained* in the original ND specification.[3] The current network (called the *cover network*) can be incrementally modified, so that its behavior is contained in the specification given for the network. As the network is manipulated, a subset of behaviors is allowed, usually restricted to those that always satisfy the external specification.

Several network operations can cause an ND network to increase its behavior and possibly cause the network to become non-compliant. We discuss only the use of the CF for SS (SS-CF) and leaving an ND subset of this at the node. In our experiments so far with using the SS-CF, where an ND relation at a node replaces the old one, we have never experienced a problem of not being able to verify the final network. Since this seems experimentally to work, we examine some reasons for this.

Consider the following scenario. Suppose an ND relation is put at a node during node minimization, where the SS-CF is used, and this makes the network non-compliant but this is not detected. When this happens, there is a subset of COs that have values not allowed by the external specification for a particular CI minterm. Moving to the next node, its SS-CF is derived. There are two cases.

1. $R^{SS}(Y,y)$ for the new node is well-defined. Then a sub-function can be chosen and part of the non-compliance will be corrected without ever knowing that the network was non-compliant.
2. $R^{SS}(Y,y)$ for the next node is *not* well-defined; this can be easily detected. It means that all non-conformance in the COs in the TFI of the node can't be corrected by changing only this node.[4]

Our current strategy is to correct when possible and to leave the current relation alone when the SS-CF is not well-defined. This has the advantage that the SS-CF can be used and no extra checking is done.

11 Conclusions

Both multi-level networks and non-determinism are efficient ways to compactly represent behavior. To merge these two concepts, a theory of non-deterministic networks was presented and the legality of various network manipulations was analyzed assuming three different definitions of behavior. Such networks can have, at any node, a non-deterministic relation, which specifies how the node behaves. Behaviors of such networks were defined in terms of normal (NS), normal compatible (NSC), and set simulation (SS). SS was shown to be the same as that obtained by collapsing the network in topological order and NSC was shown to be equivalent to collapsing in reverse topological order. It was shown that some operations might cause the network to become non-compliant, and there are several methods to bring the network back to compliance.

We observed that only *deterministic* node implementations contained in SS-CF flexibility can be guaranteed to create a compliant network. However, subsequent extractions done on the network might cause non-compliance. Conditions were given, under which this increase in SS-behavior could happen.

A new method was given for computing a *minimum* well-defined relation contained in an ND relation. The size of a relation is measured in terms of the number of cubes in all its minimized *i*-sets. The minimized relation is never larger than any contained deterministic realization.

The manipulation of ND networks using SS-behavior has been implemented in MVSIS [7]. Our initial experience with using operations that could cause the network to become non-compliant has been that it happens rarely, and when it does, simple greedy procedures bring it back to compliance. We are currently developing a number of Boolean operations using ND networks and multi-valued variables to enhance various optimization algorithms, even when a final binary deterministic implementation is sought.

Our longer-range goal is to use ND network manipulations to operate on ND regular automata with the hope that the operations of complementation and composition, applied directly to multi-level network representations of the automata, can be done efficiently in many practical cases.

---

[3] Some applications require a final deterministic representation in order to implement each node as a circuit. However, a final step can accomplish this, while intermediate steps can take advantage of the compactness and generality of ND representations. Determinization can be done using the heuristic in Section 9.1 applied to the CF for each node.

[4] However, it may be that some non-compliance can be corrected, but we do not know exactly how this should be done. If it was known that the network was compliant before the last node was changed, we could backtrack and change that node back.

Acknowledgements

The first author was partially supported by a research grant from Intel Corporation. The second author acknowledges the generous support of the SRC under contract 683.004, the GSRC and the California Micro program with our industrial sponsors, Cadence and Synplicity.

References

[1] R. E. Bryant. Boolean Analysis of MOS Circuits. *IEEE Trans. on CAD*, July 1987.

[2] Y. Jiang and R. Brayton. Don't-Cares and Multi-Valued Logic Network Optimization. *Proc. ICCAD'00*. Nov. 2000.

[3] B. Lin and A. R. Newton. Efficient Symbolic Manipulation of Equivalence Relations and Classes. *Proc. Int. Workshop on Formal Methods in VLSI Design*, Jan. 1991.

[4] A. Mishchenko and R. Brayton. Simplification of Non-Deterministic Multi-Valued Networks, *IWLS'02*, June 2002.

[5] A. Mishchenko and R. Brayton. A Boolean Paradigm for Multi-Valued Logic Synthesis, *IWLS'02*, June 2002.

[6] A. Mishchenko and R. Brayton, A Theory of Non-Deterministic Networks. UCB ERL Technical Report. Dept. of EECS, Univ. of California, Berkeley. To appear.

[7] MVSIS Group. *MVSIS.* UC Berkeley: http://www-cad.eecs.berkeley.edu/mvsis/

[8] E. Sentovich, et al. "SIS: A System for Sequential Circuit Synthesis", *Tech. Rep. UCB/ERI, M92/41*, ERL, Dept. of EECS, Univ. of California, Berkeley, 1992.

[9] Y. Watanabe, L. Guerra, and R. K. Brayton. Logic Optimization with Multi-Output Gates. *Proc. ICCD '93*, Sept. 1993.

[10] N. Yevtushenko, T. Villa, R. K. Brayton, A. Petrenko, and A. L. Sangiovanni-Vincentelli. Solution of Parallel Language Equations for Logic Synthesis. *Proc. ICCAD '01*, Nov. 2001.