## Logic Synthesis and Verification

# Jie－Hong Roland Jiang <br> 江介宏 <br> Department of Electrical Engineering National Taiwan University 

Fall 2014

# Don＇t Cares and Node Minimization 

## Reading：

## Logic Synthesis in a Nutshell Section 3 （§3．4）

## Node Minimization

## Problem:

■ Given a Boolean network, optimize it by minimizing each node as much as possible

Note:

- Assume initial network structure is given
aTypically obtained after the global optimization, e.g. division and resubstitution
$\square$ We minimize the function associated with each node


## Permissible Functions of a Node

-In a Boolean network, we may represent a node using the primary inputs $\left\{x_{1}, \ldots, x_{n}\right\}$ plus the intermediate variables $\left\{y_{1}, \ldots, y_{m}\right\}$, as long as the network is acyclic

Definition:
A function $g_{j}$, whose input variables are a subset of $\left\{x_{1}, \ldots, x_{n}, y_{1}, \ldots, y_{m}\right\}$, is implementable at a node $j$ if
$\square$ the variables of $g_{j}$ do not intersect with TFO $_{j}$ $\square \mathrm{TFO}_{j}=\{$ node $\mathrm{i}: \mathrm{i}=\mathrm{j}$ or $\exists$ path from j to i$\}$
$\square$ the replacement of the function associated with $j$, say $f_{j}$, by $g_{j}$ does not change the functionality of the network

## Permissible Functions of a Node

$\square$ The set of implementable (or permissible) functions at $j$ provides the solution space of the local optimization at node j


TFOj $=$ \{node $\mathrm{i}: \mathrm{i}=\mathrm{j}$ or $\exists$ path from j to i$\}$

## Prime and Irredundant Boolean Network

Consider a sum- of -products expression $F_{j}$ associated with a node j$\square$ Definition: $F_{j}$ is prime (in a multi-level sense) if for all cubes $c \in F_{j}$, no literal of c can be removed without changing the functionality of the network
$\square$ Definition: $F_{j}$ is irredundant if for any cube $c \in F_{i}$, the removal of $c$ from $F_{j}$ changes the functionality of the network
$\square$ Definition: A Boolean network is prime and irredundant if $F_{j}$ is prime and irredundant for all j

## Node Minimization

Goals:
$\square$ Given a Boolean network:

1. make the network prime and irredundant
2. for a given node of the network, find a least-cost SOP expression among the implementable functions at the node

Note:
■ Goal 2 implies Goal 1

- There are many expressions that are prime and irredundant, just like in two-level minimization. We seek the best.


## Taxonomy of Don't Cares

External don't cares - XDCThe set of don't care minterms (in terms of primary input variables) given for each primary output is denoted $\mathrm{XDC}_{k}, \mathrm{k}=1, \ldots, \mathrm{p}$

- Internal don't cares - derived from the network structure
- Satisfiability don't cares - SDC

■ Observability don't cares - ODC
$\square$ Complete Flexibility -

- CF is a superset of SDC, ODC, and localized XDC


## Satisfiability Don't Cares

$\square$ We may represent a node using the n primary inputs plus the $m$ intermediate variables

- Boolean space is $\mathrm{B}^{\mathrm{n}+\mathrm{m}}$
$\square$ However, intermediate variables depend on the primary inputs
$\square$ Thus not all the minterms of $\mathrm{B}^{\mathrm{n}+\mathrm{m}}$ can occur:
$\square$ use the non-occuring minterms as don't cares to optimize the node function
$\square$ we get internal don't cares even when no external don't cares exist


## Satisfiability Don't Cares

Example

$$
\begin{aligned}
& y_{1}=F_{1}=\neg x_{1} \\
& y_{j}=F_{j}=y_{1} x_{2} \\
& \square \text { Since } y_{1}=\neg x_{1}, y_{1} \oplus \rightarrow x_{1} \text { never } \\
& \text { occurs. So we may include these } \\
& \text { points to represent } F_{j} \\
& \Rightarrow \text { Don't Cares } \\
& \square S D C=\left(y_{1} \oplus \neg x_{1}\right)+\left(y_{j} \oplus y_{1} x_{2}\right)
\end{aligned}
$$



In general,

$$
S D C=\sum_{j=1}^{m}\left(y_{j} \overline{F_{j}}+\overline{y_{j}} F_{j}\right)
$$

Note: $S D C \subseteq B^{n+m}$

## Observability Don't Cares

$\mathrm{y}_{\mathrm{j}}=\neg \mathrm{x}_{1} \mathrm{x}_{2}+\mathrm{x}_{1} \neg \mathrm{x}_{3}$
$z_{k}=x_{1} x_{2}+y_{j} \neg x_{2}+\neg y_{j} \neg x_{3}$Any minterm of $x_{1} x_{2}+\neg x_{2} \neg x_{3}+x_{2} x_{3}$ determines $z_{k}$ independent of $y_{j}$
$\square$ The ODC of $y_{j}$ w.r.t. $z_{k}$ is the set of minterms of the primary inputs for which the value of $y_{j}$ is not observable at $z_{k}$
$O D C_{j k}=\left\{\left.x \in B^{n}\left|z_{k}(x)\right|_{y_{j}=0} \equiv z_{k}(x)\right|_{y_{j}=1}\right\}$


This means that the two Boolean networks,
■ one with $y_{j}$ forced to 0 and

- one with $y_{j}$ forced to 1
compute the same value for $z_{k}$ when $x \in O D C_{j k}$The ODC of $y_{j}$ w.r.t. all primary outputs is $O D C_{j}=\cap_{k} O D C_{j k}$


## Observability Don't Cares

$O D C_{j k}=\left\{x \in B^{n}\left|z_{k}(x)\right|_{y_{j}=0}=\left.z_{k}(x)\right|_{y_{j}=1}\right\}$
denote $O D C_{j k}=\frac{\overline{\partial z_{k}}}{\partial y_{j}}$
where $\left.\frac{\partial z_{k}}{\partial y_{j}}=\left.\left.z_{k}(x)\right|_{y_{j}=0} \oplus z_{k}(x)\right|_{y_{j}=1}\right\}$


## Observability Don't Cares

-The ODCs of node $i$ and node $j$ in a Boolean network may not be compatible - Modifying the function of node i using ODC may invalidate ODC $_{j}$

- It brings up the issue of compatibility ODC (CODC)
- Computing CODC is too expensive to be practical
$\square$ Practical approaches to node minimization often consider one node at a time rather than multiple nodes simultaneously


## External Don't Cares

$\square$ The XDC global for an entire Boolean network is often given
$\square$ The XDC local for a specified window in a Boolean network can be computed
$\square$ Question:
$\square$ How do we represent XDC?

- How do we translate XDC into local don't care?
-XDC is originally in PI variables
-Translate XDC in terms of input variables of a node


## External Don't Cares

## $\square$ Representing XDC


multi-level Boolean network for z

## Don't Cares of a Node

-The don't cares of a node j can be computed by

$$
D C_{j}=\sum_{i \notin F O_{j}}\left(y_{i} \bar{F}_{i}+\bar{y}_{i} F_{i}\right)+\prod_{k=1}^{p}\left(O D C_{j k}+X D C_{k}\right)
$$



## Don't Cares of a Node

$\square$ Theorem: The function $\mathscr{f}_{\mathrm{j}}=\left(\mathrm{F}_{\mathrm{j}}-\mathrm{DC}_{\mathrm{j}}, \mathrm{DC}_{\mathrm{j}}, \neg\left(\mathrm{F}_{\mathrm{j}}+\mathrm{DC}_{\mathrm{j}}\right)\right)$ is the complete set of implementable functions at node $j$
$\square$ Corollary: $\mathrm{F}_{\mathrm{j}}$ is prime and irredundant (in the multi-level sense) iff it is prime and irredundant cover of $\mathscr{F}_{j}$
$\square$ A least-cost expression at node j can be obtained by minimizing $\mathscr{F}_{\mathrm{j}}$
$\square$ A prime and irredundant Boolean network can be obtained by using only 2 -level logic minimization for each node $j$ with the don't care $\mathrm{DC}_{\mathrm{j}}$

## Mapping Don't Cares to Local Space

$\square$ How can ODC + XDC be used for optimizing a node $j$ ?
$■$ ODC and XDC are in terms of the primary input variables
$\square$ Need to convert to the input variables of node $j$


## Mapping Don't Cares to Local Space

$\square$ Definition: The local space $\mathrm{Br}^{r}$ of node j is the Boolean space spanned by the fanin variables of node ( plus maybe some other variables chosen selectively)

- A don't care set $\mathrm{D}\left(\mathrm{y}^{\mathrm{r}+}\right)$ computed in local space spanned by $\mathrm{y}^{\mathrm{r}+}$ is called a local don't care set. (The " + " stands for additional variables.)
■ Solution: $\operatorname{Map} \operatorname{DC}(x)=O D C(x)+X D C(x)$ to local space of the node to find local don't cares, i.e.,

$$
D\left(y^{r+}\right)=I M G_{g_{F_{j}^{\prime}}}(\overline{D C}(x))
$$

## Mapping Don't Cares to Local Space

- Computation in two steps:

1. Find $D C(x)$ in terms of primary inputs
2. Find $D$, the local don't care set, by image computation and complementation

$$
D\left(y^{r+}\right)=\overline{I M G_{g_{\mathrm{FIF}_{j}}}(\overline{\overline{D C}}(x))}
$$



Mapping Don't Cares to Local Space Global Function of a Node

$$
y_{j}=\left\{\begin{array}{l}
f_{j}\left(y_{k}, \cdots, y_{l}\right) \\
g_{j}\left(x_{1}, \cdots, x_{n}\right) \text { global function }
\end{array}\right.
$$

$$
B^{m+n} \rightarrow B^{n}
$$



Mapping Don't Cares to Local Space Don't Cares in Primary Inputs
-BDD based computation

- Build BDD's representing global functions at each node
$\square$ in both the primary network and the don't care network, $g_{j}\left(x_{1}, \ldots, x_{n}\right)$


## ■use BDD_compose

$\square$ Replace all the intermediate variables in (ODC+XDC) with their global BDDs

$$
\begin{aligned}
& \tilde{h}(x, y)=D C(x, y) \rightarrow h(x)=D C(x) \\
& \tilde{h}(x, y)=\tilde{h}(x, g(x))=h(x)
\end{aligned}
$$

## Mapping Don't Cares to Local Space

$\square$ Example

$y_{10}=x_{1} x_{3} \quad y_{11}=x_{2} x_{4}$
$X D C_{2}=y_{12}$
$g_{12}=X_{1} X_{2} X_{3} X_{4}$
z (output)

$O D C_{2}=y_{1}$
$g_{1}=X_{1} X_{2} X_{3} X_{4}$
$D C_{2}=O D C_{2}{ }^{+} X D C_{2}$
$D C_{2}=X_{1} X_{2} X_{3} X_{4}{ }^{+} X_{1} X_{2} X_{3} X_{4}$

## Mapping Don't Cares to Local Space Image Computation

$\square$ Local don't cares are the set of minterms in the local space of $y_{i}$ that cannot be reached under any input combination in the care set of $y_{i}$ (in terms of the input variables).
$\square$ Local don't care set: $D_{i}=\overline{\operatorname{IMAGE}}{ }_{\left(g_{1}, g_{2}, \cdots, g_{r}\right)}$ [care set]
i.e. those patterns of $\left(y_{1}, \ldots, y_{r}\right)$ that never appear as images of input cares.


$O D C_{2}=y_{1}$
$O D C_{2}=y_{12}$
$D C_{2}=X_{1} X_{2} X_{3} X_{4}{ }^{+} X_{1} \bar{X}_{2} X_{3} \bar{X}_{4}$ $\overline{D C_{2}}=\bar{X}_{1}+\bar{X}_{3}+X_{2} \bar{X}_{4}+\bar{X}_{2} X_{4}$ $D_{2}=y_{7} y_{8}$

Note that $D_{2}$ is given in this space $y_{5}, y_{6}, y_{7}, y_{8}$. Thus in the space $(--10)$ never occurs.
Can check that $\overline{D C_{2}} D_{2}=\varnothing=\overline{D C_{2}}\left(x_{1} x_{3}\right)\left(x_{2} \overline{x_{4}}+\overline{x_{2}} x_{4}\right)$
Using $D_{2}=y_{7} y_{8}, \mathrm{f}_{2}$ can be simplified to
$f_{2}=y_{7} y_{8}+y_{5} y_{6}$

## Image Computation

- Two methods:

1. Transition relation method
$\square \mathrm{f}: \mathrm{B}^{\mathrm{n}} \rightarrow \mathrm{B}^{r} \Rightarrow \mathrm{~F}: \mathrm{B}^{\mathrm{n}} \times \mathrm{B}^{r} \rightarrow \mathrm{~B}$
( $F$ is the characteristic function of $f!$ )

$$
\begin{aligned}
F(x, y) & =\{(x, y) \mid y=f(x)\} \\
& =\prod_{i \leq r}\left(y_{i} \equiv f_{i}(x)\right) \\
& =\prod_{i \leq r}\left(y_{i} f_{i}(x)+\bar{y}_{i} \bar{f}_{i}(x)\right)
\end{aligned}
$$

2. Recursive image computation (omitted)

## Image Computation Transition Relation Method

$\square$ Image of set $A$ under $f: f(A)=\exists_{x}(F(x, y) \wedge A(x))$

where $\exists_{x}=\exists_{x_{1}} \cdots \exists_{x_{n}}$ and $\exists_{x_{i}} g=g_{x_{i}}+g_{\bar{x}_{i}}$
$\square$ The existential quantification $\exists_{x}$ is also called "smoothing" Note: The result is a BDD representing the image,
i.e. $f(A)$ is a BDD with the property that
$\operatorname{BDD}(y)=1 \Leftrightarrow \exists x$ such that $f(x)=y$ and $x \in A$.

## Node Simplification



Express ODC in terms of variables in $\mathrm{B}^{\mathrm{n}+\mathrm{m}}$

## Node Simplification



## Complete Flexibility

-Complete flexibility (CF) of a node in a combinational network
■ SDC + ODC + localized XDC
$\square$ Used to minimize one node at a time
$\square$ Not considering compatible flexibilities among multiple nodes
$\square$ Different from CODC, where don't cares at different nodes are compatible and can minimize multiple nodes simultaneously

## Complete Flexibility

$\square$ Definition: A flexibility at a node is a relation (between the node's inputs and output) such that any well-defined subrelation used at the node leads to a network that conforms to the external specification
$\square$ Definition: The complete flexibility (CF) is the maximum flexibility possible at a node


Combinational Logic Network

## Complete Flexibility

$\square$ Computing complete flexibility


Note: Specification relation $S(X, Z)$ may contain nondeterminism and subsumes XDC. Influence relation $I\left(X, y_{i}, Z\right)$ subsumes ODC.

## Complete Flexibility

$\square$ Computing complete flexibility


## Complete Flexibility

$\square$ Computing complete flexibility

$C F\left(Y_{i}, y_{i}\right)=\forall X .\left[E\left(X, Y_{i}\right) \Rightarrow \forall Z .\left[I\left(X, y_{i}, Z\right) \Rightarrow S(X, Z)\right]\right]$

$$
=\forall X, Z \cdot\left[\overline{E\left(X, Y_{i}\right) \cdot I\left(X, y_{i}, Z\right) \cdot \overline{S(X, Z)}}\right]
$$

Note: The same computation works for multiple $y_{i}{ }^{\prime} \mathrm{s}$

## Window and Don't Care Compuation

- Definition: A window for a node in the network is the context in which the don'tcares are computedA window includes
- n levels of the TFI
- m levels of the TFO
- all re-convergent paths captured in this scope
- Window with its PIs and POs can be considered as a separate network
$\square$ Optimizing a window is more computationally affordable than optimizing an entire network


## Boolean network

Window POs


## SAT-based Don't Care Computation

"Miter" constructed for the window POs

window

## SAT-based Don't Care Computation

Compute the care set

- Simulation
$\square$ Simulate the miter using random patterns
$\square$ Collect x minterms, for which the output of miter is 1
$\square$ This is a subset of a care set
- Satisfiability
- Derive set of network clauses
$\square$ Add the negation of the current care set
$\square$ Assert the output of miter to be 1
- Enumerate through the SAT assignments
$\square$ Add these assignments to the care set



## Resubstitution for Circuit Minimization

Resubstitution considers a node in a Boolean network and expresses it using a different set of fanins

X


X

## Resubstitution with Don't Cares

## $\square$ Consider all or some nodes in Boolean network

■ Create window

- Select possible fanin nodes (divisors)

■ For each candidate subset of divisors
$\square$ Rule out some subsets using simulation
$\square$ Check resubstitution feasibility using SAT
$\square$ Compute resubstitution function using interpolation

- A low-cost by-product of completed SAT proofs

■ Update the network if there is an improvement

## Resubstitution with Don't Cares

$\square$ Given:

- node function $F(x)$ to be replaced
- care set $C(x)$ for the node
- candidate set of divisors $\left\{g_{j}(x)\right\}$ for re-expressing $\mathrm{F}(\mathrm{x})$

- Find:
- A resubstitution function $h(y)$ such that $F(x)=h(g(x))$ on the care set
- Necessary and sufficient condition: For any minterms a and $b, F(a) \neq$ $F(b)$ implies $g_{i}(a) \neq g_{i}(b)$ for some $g_{i}$



## Resubstitution

$\square$ Example

## Given:

$F(x)=\left(x_{1} \oplus x_{2}\right)\left(x_{2} \vee x_{3}\right)$
Two candidate sets:
$\left\{g_{1}=x_{1}{ }^{\prime} x_{2}, g_{2}=x_{1} x_{2}{ }^{\prime} x_{3}\right\}$,
$\left\{g_{3}=x_{1} \vee x_{2}, g_{4}=x_{2} x_{3}\right\}$
Set $\left\{g_{2}, g_{4}\right\}$ cannot be used for resubstitution while set $\left\{g_{1}, g_{2}\right\}$ can.

| $x$ | $F(x)$ | $g_{1}(x)$ | $g_{2}(x)$ | $g_{3}(x)$ | $g_{4}(x)$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 000 | 0 | 0 | 0 | 0 | 0 |
| 001 | 0 | 0 | 0 | 0 | 0 |
| 010 | 1 | 1 | 0 | 1 | 0 |
| 011 | 1 | 1 | 0 | 1 | 1 |
| 100 | 0 | 0 | 0 | 1 | 0 |
| 101 | 1 | 0 | 1 | 1 | 0 |
| 110 | 0 | 0 | 0 | 1 | 0 |
| 111 | 0 | 0 | 0 | 1 | 1 |

## SAT-based Resubstitution

Miter for resubstitution check


Resubstitution function exists if and only if SAT problem is unsatisfiable Note: Care set is used to enhance resubstitution check

## SAT-based Resubstitution

$\square$ Computing dependency function $h$ by interpolation

- Consider two sets of clauses, $A(x, y)$ and $B(y, z)$, such that $A(x, y) \wedge B(y, z)=0$
■ y are the only variables common to $A$ and $B$
- An interpolant of the pair ( $A(x, y), B(y, z)$ ) is a function $\mathrm{h}(\mathrm{y})$ depending only on the common variables y such that $A(x, y) \Rightarrow h(y) \Rightarrow \neg B(y, z)$

Boolean space ( $x, y, z$ )


## SAT-based Resubstitution

Problem: Find function $h(y)$, such that $C(x) \Rightarrow[h(g(x)) \equiv F(x)]$, i.e. $F(x)$ is expressed in terms of $\left\{g_{i}\right\}$

- Solution:
- Prove the corresponding SAT problem "unsatisfiable"
- Derive unsatisfiability resolution proof [Goldberg/Novikov, DATE'03]
- Divide clauses into A clauses and B clauses
- Derive interpolant from the unsatisfiability proof [McMillan, CAV'03]
- Use interpolant as the dependency function, $\mathrm{h}(\mathrm{g})$
- Replace $F(x)$ by $h(g)$ if cost function improved


