

Logic Synthesis and Verification

Jie-Hong Roland Jiang
江介宏

Department of Electrical Engineering
National Taiwan University



Fall 2011

1

Two-Level Logic Minimization (2/2)

Reading:
Logic Synthesis in a Nutshell
Section 3 (§3.1-§3.2)

most of the following slides are by
courtesy of Andreas Kuehlmann

2

Heuristic Two-Level Logic Minimization ESPRESSO

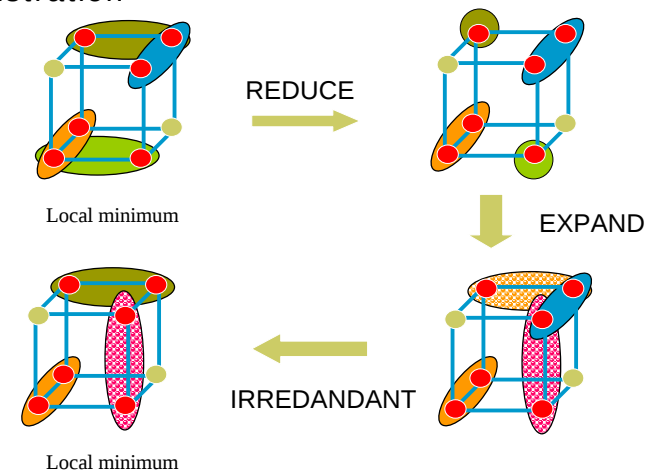
```

ESPRESSO( $\mathfrak{F}$ )
{
  (F,D,R)  $\leftarrow$  DECODE( $\mathfrak{F}$ )           //LASTGASP
  F  $\leftarrow$  EXPAND(F,R)               G  $\leftarrow$  REDUCE_GASP(F,D)
  F  $\leftarrow$  IRREDUNDANT(F,D)           G  $\leftarrow$  EXPAND(G,R)
  E  $\leftarrow$  ESSENTIAL_PRIMES(F,D)      F  $\leftarrow$  IRREDUNDANT(F+G,D)
  F  $\leftarrow$  F-E; D  $\leftarrow$  D+E         //LASTGASP
  do{                                  }while fewer terms in F
    do{                                F  $\leftarrow$  F+E; D  $\leftarrow$  D-E
      F  $\leftarrow$  REDUCE(F,D)           LOWER_OUTPUT(F,D)
      F  $\leftarrow$  EXPAND(F,R)           RAISE_INPUTS(F,R)
      F  $\leftarrow$  IRREDUNDANT(F,D)      error  $\leftarrow$  (Fold  $\subsetneq$  F) or (F  $\not\subset$  Fold + D)
    }while fewer terms in F          return (F,error)
  }
    
```

3

Heuristic Two-Level Logic Minimization ESPRESSO

Illustration



4

ESPRESSO IRREDUNDANT

Problem:

Given a cover of cubes C for some incompletely specified function (f,d,r) , find a minimum subset of cubes $S \subseteq C$ that is also a cover, i.e.

$$f \subseteq \sum_{c \in S} c \subseteq f + d$$

Idea 1:

We are going to create a function $g(y)$ and a new set of variables $y = \{y_i\}$, one for each cube c_i . A minterm in the y -space will indicate a subset of the cubes $\{c_i\}$.

Example

$y = (0,1,1,0,1,0)$, i.e. $y_1'y_2y_3y_4'y_5y_6'$, represents $\{c_2, c_3, c_5\}$

5

ESPRESSO IRREDUNDANT

Idea 2:

Create $g(y)$ so that it is the function such that:

$$g(y^*) = 1 \Leftrightarrow \sum_{y_i^*=1} c_i \text{ is a cover}$$

i.e. $g(y^*) = 1$ if and only if $\{c_i \mid y_i^* = 1\}$ is a cover.

Note: $g(y)$ can be made positive unate (monotone increasing) in all its variables.

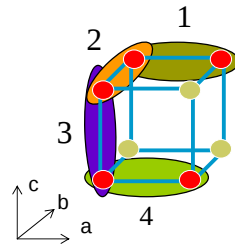
6

ESPRESSO IRREDUNDANT

Example

$$f = bc + \bar{a}c + \bar{a}\bar{b} + \bar{b}\bar{c}$$

$$g(y_1, y_2, y_3, y_4) = y_1 y_4 (y_2 + y_3)$$



Note:

We want a minimum subset of cubes that covers f , that is, the largest prime of g (least literals).

Consider g' : it is monotone decreasing in y (i.e. negative unate in y) e.g.

$$\bar{g}(y_1, y_2, y_3, y_4) = \bar{y}_1 + \bar{y}_4 + \bar{y}_2 \bar{y}_3$$

7

ESPRESSO IRREDUNDANT

Example

■ Create a Boolean matrix B for g' :

$\bar{g}$	$\rightarrow$	$B =$	1000
			0001
			0110

$$f = bc + \bar{a}c + \bar{a}\bar{b} + \bar{b}\bar{c}$$

$$\bar{g}(y_1, y_2, y_3, y_4) = \bar{y}_1 + \bar{y}_4 + \bar{y}_2 \bar{y}_3$$

■ Recall a minimal column cover of B is a prime of $g = (g')'$

■ We want a *minimum* column cover of B

■ E.g., $\{1, 2, 4\} \Rightarrow y_1 y_2 y_4$ (cubes 1, 2, 4) $\Rightarrow \{bc, a'c, b'c'\}$

8

ESPRESSO IRREDUNDANT

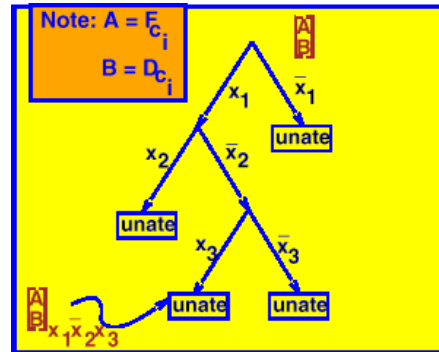
Deriving $g'(y)$

- Modify tautology algorithm:

$F = \text{cover of } \mathfrak{Z} = (f, d, r)$
 $D = \text{cover of } d$

- Pick a cube $c_i \in F$
 (Note: $c_i \subseteq F \Leftrightarrow F_{c_i} \equiv 1$)
 Do the following for each cube $c_i \subseteq F$:

$$\begin{bmatrix} A \\ B \end{bmatrix} \equiv \begin{bmatrix} F_{c_i} \\ D_{c_i} \end{bmatrix}$$

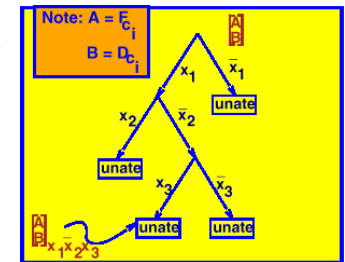


9

ESPRESSO IRREDUNDANT

Deriving $g'(y)$

- All leaves must be tautologies
- g' means how can we make it **not** a tautology
 - Must exactly delete **all** rows of all '-'s that are not part of D
- Each row came from some row of A/B
- Each row of A is associated with some cube of F
- Each cube of B is associated with some cube of D
 - Don't need to know which, and cannot delete its rows
- Rows that must be deleted are written as a cube
 - E.g. $y_1y_2y_7 \Rightarrow$ delete rows 1,3,7 of F



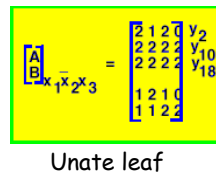
10

ESPRESSO IRREDUNDANT

Deriving $g'(y)$

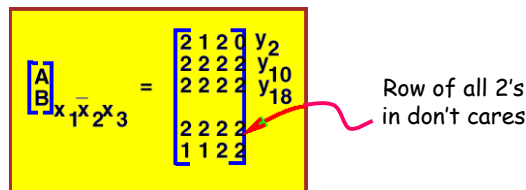
- Example

Suppose unate leaf is in subspace $x_1x_2x_3$:
 Thus we write down: $\bar{y}_{10}\bar{y}_{18}$ (actually, $\bar{y}_i$ must be one of $\bar{y}_{10}, \bar{y}_{18}$). Thus, F is **not a cover** if we leave out cubes c_{10}, c_{18} .



Note:

If a row of all 2's is in don't cares, then there is no way not to have tautology at that leaf.



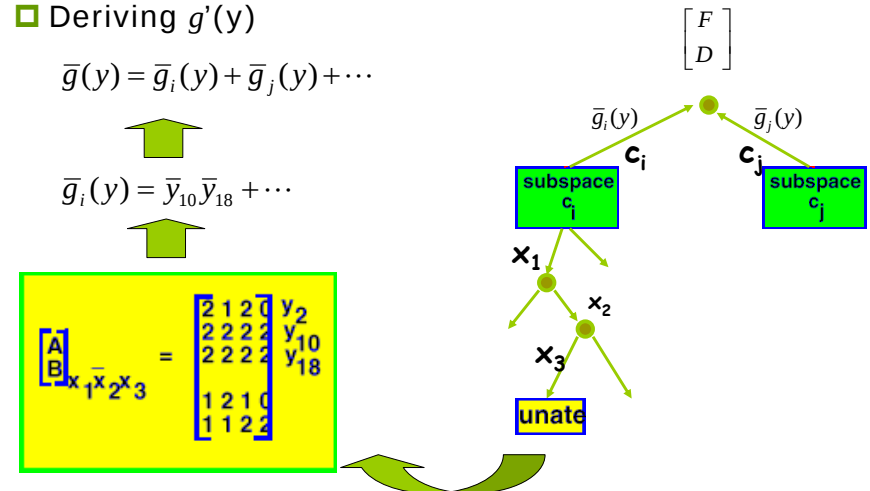
11

ESPRESSO IRREDUNDANT

Deriving $g'(y)$

$$\bar{g}(y) = \bar{g}_i(y) + \bar{g}_j(y) + \dots$$

$$\bar{g}_i(y) = \bar{y}_{10}\bar{y}_{18} + \dots$$



12

ESPRESSO IRREDUNDANT

□ Summary

1. Convert $g'(y)$ into a Boolean matrix B

□ Note that $g(y)$ is unate

2. Find a minimum column cover of B

□ E.g., if $y_1 y_3 y_{18}$ is a minimum column cover, then the set of cubes $\{c_1, c_3, c_{18}\}$ is a minimum sub-cover of $\{c_i \mid i=1, \dots, k\}$. (Recall that a minimal column cover of B is a prime of $g(y)$, and $g(y)$ gives all possible sub-covers of F).

- Note: We are just doing tautology in constructing $g'(y)$, so **unate reduction** is applicable

$$F = \begin{bmatrix} A & C \\ T & F^* \end{bmatrix}$$

13

ESPRESSO IRREDUNDANT

□ Summary

- In Q-M, we want a maximum prime of $g(y)$

$$B = \begin{matrix} \text{Minterms} \\ \text{of } f \end{matrix} \begin{matrix} \text{All primes} \\ \begin{bmatrix} 1011010 \\ \dots \\ \dots \\ \dots \\ \dots \end{bmatrix} \end{matrix} \quad B \cong \bar{g}(y) = \bar{y}_1 \bar{y}_3 \bar{y}_4 \bar{y}_6 + \dots$$

Note: A row of B says if we leave out primes $\{p_1, p_3, p_4, p_6\}$, then we **cease** to have a cover

- So basically, the only difference between Q-M and IRREDUNDANT is that for the latter, we just constructed a $g'(y)$ where we did not consider all primes, but only those in **some** cover: $F = \{c_1, c_3, \dots, c_k\}$

14

ESPRESSO EXPAND

□ $F \leftarrow \text{EXPAND}(F, R)$

- Problem: Take a cube c and make it prime by removing literals

- Greedy way: (uses D and not R)

□ Remove literal l_i from c (results in, say c^*)

□ Test if $c^* \subseteq f+d$ (i.e. test if $(f+d)_{c^*} \equiv 1$)

□ Repeat, removing valid literals in order found

- Better way: (uses R and not D)

□ Want to see all possible ways to remove maximal subset of literals

□ Idea: Create a function $g(y)$ such that $g(y)=1$ iff literals $\{l_i \mid y_i = 0\}$ can be removed (or $\{l_i \mid y_i = 1\}$ is a subset of literals such that if kept in c , will still make $c^* \subseteq f+d$, i.e. $c^* \wedge r \equiv 0$)

15

ESPRESSO EXPAND

□ Main idea

Outline:

1. Expand one cube, c_i , at a time
2. Build "blocking" matrix $B = B^{c_i}$
3. See which other cubes c_j can be **feasibly** covered using B
4. Choose expansion (**literals to be removed**) to cover most other c_j

Note: • $g(y)$ is monotone increasing

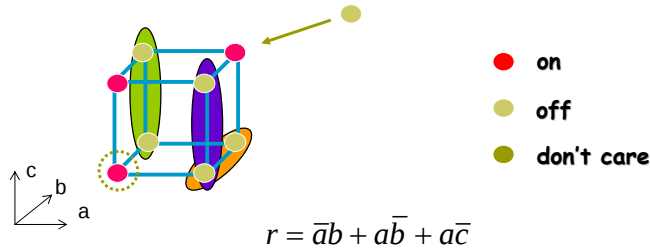
• $B \cong \bar{g}(y)$ is easily built if we have R, a cover of r .

• We do not need all of R. (**reduced offset**)

16

ESPRESSO EXPAND

Reduced offset



Make r unate by adding (1,1,1) to offset. Then the new offset $R_{\text{new}} = a + b \cong g'(y)$. This is simpler and easier to deal with.

17

ESPRESSO EXPAND

Blocking matrix B (for some cube c)

Given $R = \{r_i\}$, a cover of r . [$\mathfrak{Z} = (f, d, r)$]

$$B_{ij} = 1 \Leftrightarrow \begin{cases} l_j \in c \text{ and } \bar{l}_j \in r_i \\ \bar{l}_j \in c \text{ and } l_j \in r_i \end{cases}$$

B: rows indexed by offset cubes, columns indexed by literals of c

What does row i of B say?

- It says that if literals $\{j \mid B_{ij} = 1\}$ are removed from c , then $c^* \wedge r_i \neq 0$, i.e., $B_{ij} = 1$ is one reason why c is orthogonal to offset cube r_i
- Thus $B \rightarrow g'(y) = y_1'y_3'y_{10}' + \dots$ gives all ways that literals of c can be removed to get $c^* \not\subseteq f + d$ (i.e. $c^* \wedge r \neq 0$)

18

ESPRESSO EXPAND

Example

$$\begin{aligned} c &= ab\bar{d} \\ r_i &= \bar{a}b\bar{d}\bar{e} \\ y_1y_2y_3 &\propto a, b, \bar{d} \end{aligned}$$

$$\begin{aligned} y_1 = 1 &\Leftrightarrow \text{keep } a \\ y_2 = 1 &\Leftrightarrow \text{keep } b \\ y_3 = 1 &\Leftrightarrow \text{keep } \bar{d} \\ (B_i) = 101 &= \bar{y}_1\bar{y}_3 + \dots = \bar{g}_i(y) \end{aligned}$$

Suppose $g(y) = 1$

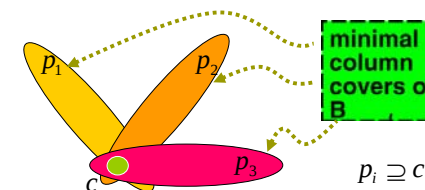
- If $y_1 = 1$, we keep literal a in cube c .
- B_i means do not keep literals 1 and 3 of c (implies that subsequent c^* is not an implicant)
 - If literals 1, 3 are removed we get $c \rightarrow c^* = b$. But $c^* \wedge r_i \neq 0$: $b \wedge \bar{a}b\bar{d}\bar{e}' = \bar{a}b\bar{d}\bar{e}' \neq 0$. So b is not an implicant.

19

ESPRESSO EXPAND

Example (cont'd)

- Thus **all minimal column covers** ($\cong g(y)$) of B are the minimal subsets of literals of c that must be kept to ensure that $c^* \subseteq f + d$ (i.e. $c^* \wedge r_i = 0$)
- Thus each minimal column cover is a prime p that covers c , i.e. $p \supseteq c$



20

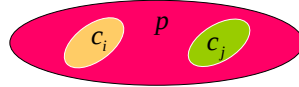
ESPRESSO EXPAND

Expanding c_i

$$F = \{ c_i \}, \mathfrak{F} = (f, d, r) \quad f \subseteq F \subseteq f + d$$

Q: Why do we want to expand c_i ?

A: To cover some other c_j 's



Q: Can we cover c_j ?

A: If and only if (SCC = "smallest cube containing" also called "supercube")

$$\text{equivalent to: } SCC(c_i \cup c_j) \subseteq f + d$$

$$\text{equivalent to: } SCC(c_i \cup c_j) \wedge r = 0$$

literals "conflicting" between c_i, c_j can be removed and still have an implicant

21

ESPRESSO EXPAND

Expanding c_i

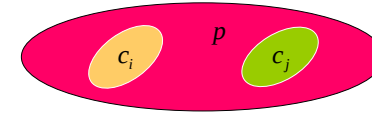
Can check $SCC(c_i, c_j)$ with blocking matrix:

$$c_i = 12012$$

$$c_j = 12120$$

implies that literals 3 and 4 must be removed for c_i^* to cover c_j

Check if columns 3, 4 of B can be removed without causing a row of all 0's



22

ESPRESSO EXPAND

Covering function

- The objective of EXPAND is to expand c_i to cover as many cubes c_j as possible. The blocking function $g'(y) = 1$ whenever the subset of literals $\{l_i \mid y_i = 1\}$ yields a cube $c^* \not\subseteq f + d$.

$$\text{Note: } c^* = \prod_{(y_j=1)} l_j$$

- We now build the covering function h , such that:

$h(y) = 1$, whenever the cube $c^* \supseteq c_i$ covers another cube $c_j \subseteq F$

Note: $h(y)$ is easy to build

- Thus a minterm m of $g(y) \wedge h(y)$ is such that it gives $c^* \subseteq f + d$ ($g(m) = 1$) and covers at least one cube ($h(m) = 1$). In fact every cube $c^*_m \supseteq c_i$ is covered. We seek m which results in the most cubes covered.

23

ESPRESSO EXPAND

Covering function

Define $h(y)$ by a set of cubes where $d_k = k^{\text{th}}$ cube is:

$$d_k = \begin{cases} \emptyset & \text{if } SCC[c_i \cup c_k] \not\subseteq f + d \\ \bar{y}_j & \text{if } c_k^j \not\subseteq c_i^j \text{ i.e.} \\ 2 & \text{otherwise} \end{cases} \quad \begin{cases} 2 \not\subseteq 1 \\ 2 \not\subseteq 0 \\ 0 \not\subseteq 1 \\ 1 \not\subseteq 0 \end{cases}$$

d_k^j : j^{th} literal of k^{th} cube

Every d_k indicates the minimal expansion to cover c_k , that is, which literals that we have to leave out to minimally cover c_k . Essentially $d_k \neq \emptyset$ if cube c_k can be feasibly covered by expanding cube c_i .

Note that $h(y) = d_1 + d_2 + \dots + d_{|F|-1}$ (one for each cube of F , except c_i) is monotone decreasing.

24

ESPRESSO EXPAND

□ Covering function

- We want a minterm m of $g(y) \wedge h(y)$ contained in a **maximum** number of d_k 's
- In Espresso, we build a Boolean covering matrix C (note that $h(y)$ is **negative unate**) representing $h(y)$ and solve this problem with greedy heuristics

Note:

$$B \cong \bar{g}(y)$$

$$\text{but } C \cong \tilde{h}(y) \supseteq h(y)$$

$\tilde{h}(y)$ is an over-approximation of $h(y)$, e.g., by removing the $d_k = \emptyset$ rule in the previous slide

$$C = \begin{Bmatrix} \dots \\ 010110 \\ 101011 \\ 100101 \\ \dots \end{Bmatrix} \quad B = \begin{Bmatrix} \dots \\ 110110 \\ 101010 \\ 101001 \\ \dots \end{Bmatrix}$$

25

ESPRESSO EXPAND

□ Covering function

$$C = \begin{Bmatrix} \dots \\ 010110 \\ 101011 \\ 100101 \\ \dots \end{Bmatrix} \quad B = \begin{Bmatrix} \dots \\ 110110 \\ 101010 \\ 101001 \\ \dots \end{Bmatrix}$$

- Want a set of columns such that if eliminated from B and C results in **no empty rows of B** and a **maximum of empty rows in C**
- Note: A "1" in C can be interpreted as a reason why c^* does **not** cover c_j

26

ESPRESSO EXPAND

□ Endgame

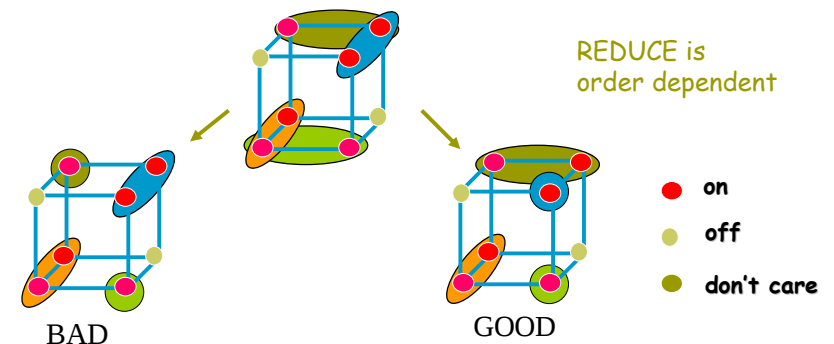
- What do we do if $h(y) \equiv 0$?
 - This could be important in many hard problems, since it is often the case that $h(y) \equiv 0$
- Some things to try:
 - Generate largest prime covering c_i
 - Generate largest prime covering cover most care points of another cube c_k
 - Coordinate two or more cube expansions, i.e. try to cover another cube by a combination of several other cube expansions

27

ESPRESSO REDUCE

□ Problem:

- Given a cover F and $c \in F$, find the **smallest** cube $\underline{c} \subseteq c$ such that $F \setminus \{c\} + \{\underline{c}\}$ is still a cover
- $\underline{c}$ is called the **maximally** reduced cube of c

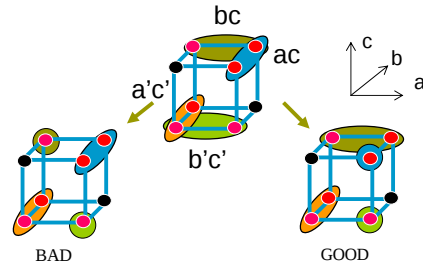


28

ESPRESSO REDUCE

Example

$$F = ac + bc + \bar{b}\bar{c} + \bar{a}\bar{c}$$



Two orders:

1. $\text{REDUCE}(F = \{ac, bc, \bar{b}\bar{c}, \bar{a}\bar{c}\}) = \bar{a}\bar{b}c + bc + \bar{a}\bar{b}\bar{c} + \bar{a}\bar{c}$
2. $\text{REDUCE}(F = \{bc, \bar{b}\bar{c}, ac, \bar{a}\bar{c}\}) = \bar{a}\bar{b}c + ac + \bar{a}\bar{b}\bar{c} + \bar{a}\bar{c}$

REDUCE is **order dependent** !

29

ESPRESSO REDUCE

Algorithm $\text{REDUCE}(F, D)\{$

```

F ← ORDER(F)
for (1 ≤ j ≤ |F|) {
    c_j ← MAX_REDUCE(c, F, D)
    F ← (F ∪ {c_j}) \ {c_j}
}
return F
}
```

30

ESPRESSO REDUCE

Main Idea: Make a prime not a prime but still maintain cover:

$$\{c_1, \dots, c_i, \dots, c_k\} \rightarrow \{c_1, \dots, \underline{c}_i, c_{i+1}, \dots, c_k\}$$

But

$$f \subseteq \sum_{j=0}^{i-1} c_j + \underline{c}_i + \sum_{j=i+1}^k c_j \subseteq f + d$$



To get out of a local minimum (prime and irredundant is local minimum)

After reduce, have non-primes and can expand again in different directions

Since EXPAND is "smart", it may know best direction

31

ESPRESSO REDUCE

$$F = \{c_1, c_2, \dots, c_k\}$$

$$F(i) = (F + D) \setminus \{c_i\} = \{c_1, c_2, \dots, c_{i-1}, c_{i+1}, \dots, c_k\}$$

Reduced cube:

$\underline{c}_i$ = smallest cube containing $(c_i \cap \bar{F}(i))$

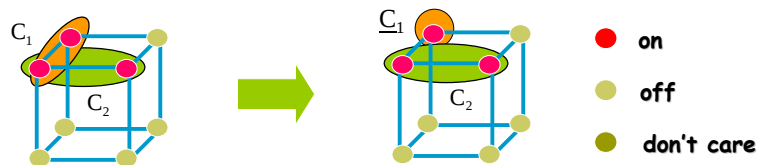
Note that $c_i \cap \bar{F}(i)$ is the set of points uniquely covered by c_i (and not by any other c_j or D).

Thus, $\underline{c}_i$ is the smallest cube containing the minterms of c_i which are not in $F(i)$.

32

ESPRESSO REDUCE

- SCC: “smallest cube containing”, i.e., supercube
- SCCC: “smallest cube containing complement”



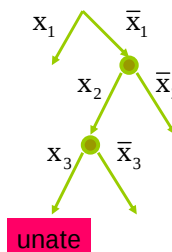
$$\begin{aligned}
 \underline{c}_i &= SCC(c_i \cap \overline{F(i)}) \\
 &= SCC(c_i \overline{F_{c_i}(i)}) \\
 &= c_i SCC(\overline{F_{c_i}(i)}) \\
 &= c_i SCCC(F_{c_i}(i))
 \end{aligned}$$

33

ESPRESSO REDUCE

■ SCCC computation

- Unate recursive paradigm
 - Select most binate variable
 - Cofactor until unate leaf



What is SCCC (unate cover) ?

- Note that for a cube c with at least 2 literals, $SCCC(c)$ is the **universe**:

$$\text{cube} = 01222 \longrightarrow \overline{\text{cube}} = \begin{matrix} 12222 \\ 20222 \end{matrix} \quad \text{Hence, } SCCC(\text{cube}) = 22222$$

- Implies only need to look at 1-literal cubes

34

ESPRESSO REDUCE

■ SCCC computation

- $SCCC(U) = \gamma$ for a unate cover U

Claim

- If unate cover has row of all 2's except one 0, then complement is in x_i , i.e. $\gamma_i = 1$
- If unate cover has row of all 2's except one 1, then complement is in x_i' , i.e. $\gamma_i = 0$
- Otherwise, in both subspaces, i.e. $\gamma_i = 2$

Finally

$$\begin{aligned}
 SCCC(c_1 + c_2 + \dots + c_k) &= SCC(\overline{c_1} \overline{c_2} \dots \overline{c_k}) \\
 &= SCC(\overline{c_1}) \cap \dots \cap SCC(\overline{c_k})
 \end{aligned}$$

35

ESPRESSO REDUCE

■ SCCC computation

Example 1: $f = a + bc + \overline{d} \Rightarrow \overline{f} = \overline{a}(\overline{b} + \overline{c})d \subseteq \overline{a}d$

- **Note:** 0101 and 0001 are both in $\overline{f}$. So SCCC could not have literal b or $\overline{b}$.

Example 2:

	2	2	2	2	0
	0	2	2	2	2
$U(\text{unate}) =$	2	1	1	2	2
	2	1	2	1	0
	↑			↑	

- Note that columns 1 and 5 are essential: they must be in every minimal cover. So $\neg U = x_1 x_5 (\dots)$. Hence $SCCC(U) = x_1 x_5$

36

ESPRESSO REDUCE

■ SCCC computation

Example 2 (cont'd):

$$\begin{aligned}
 U &= \bar{x}_1 + \bar{x}_5 + x_2(x_3 + x_4) \\
 \bar{U} &= x_1x_5(\bar{x}_2 + \bar{x}_3\bar{x}_4) \\
 \bar{U}(\text{unate}) &= 1 \quad 2 \quad 0 \quad 0 \quad 1 \subseteq 12221 \\
 &\quad \uparrow \quad \uparrow \quad \uparrow
 \end{aligned}
 \quad \text{minterms of } \bar{U} = \begin{array}{ccccc} 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{array}$$

The marked columns contain both 0's and 1's. But every prime of $\bar{U}$ contains literals x_1, x_5

37

ESPRESSO REDUCE

■ SCCC computation

■ At unate leaves

$$n = \text{SCCC}(\text{unate}) = \emptyset \quad \text{if row of all 2's}$$

$$n_j = \begin{cases} x_j & \text{if column } j \text{ has a row singleton with a 0 in it} \\ \bar{x}_j & \text{if column } j \text{ has a row singleton with a 1 in it} \\ 2 & \text{otherwise} \end{cases}$$

■ Hence unate leaf is **easy** !

38

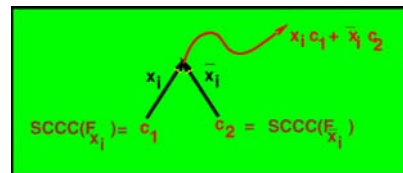
ESPRESSO REDUCE

■ SCCC computation

■ Merging

■ We need to produce $\text{SCCC}(f) = \text{SCC}(x_1c_1 + \bar{x}_1c_2) = \gamma$

$$\begin{aligned}
 \gamma &= I_1 I_2 \dots I_k \\
 x_i \in \gamma &\Leftrightarrow c_2 = \emptyset \\
 \bar{x}_i \in \gamma &\Leftrightarrow c_1 = \emptyset \\
 I_{j \neq i} \in \gamma &\Leftrightarrow (I_j \in c_1) \wedge (I_j \in c_2)
 \end{aligned}$$



- If $c_1 \wedge c_2 \neq \emptyset$, then $\gamma_i = 2$
 - because minterms with x_i and $\neg x_i$ literals both exist, and thus $(\text{SCC}(x_i c_1 + x_i c_2))_i = 2$
- If $I_j \notin c_1$ or $I_j \notin c_2$, then $\gamma_j = 2$ (where $I_j = x_j$ or $\neg x_j$)
 - because minterms with x_j and $\neg x_j$ literals both exist
- If $I_j \in c_1$ and $\neg I_j \in c_2$, then $\gamma_j = 2$.

39

ESPRESSO

ESPRESSO($\mathfrak{F}$)

```

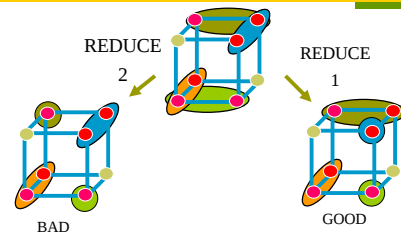
{
  (F,D,R) ← DECODE( $\mathfrak{F}$ )
  F ← EXPAND(F,R)
  F ← IRREDUNDANT(F,D)
  E ← ESSENTIAL_PRIMES(F,D)
  F ← F-E; D ← D+E
  do{
    do{
      F ← REDUCE(F,D)
      F ← EXPAND(F,R)
      F ← IRREDUNDANT(F,D)
    }while fewer terms in F
    F ← F+E; D ← D-E
    LOWER_OUTPUT(F,D)
    RAISE_INPUTS(F,R)
    error ← (Fold ⊂ F) or (F ⊄ Fold + D)
    return (F,error)
  }
  //LASTGASP
  G ← REDUCE_GASP(F,D)
  G ← EXPAND(G,R)
  F ← IRREDUNDANT(F+G,D)
  //LASTGASP
}
    
```

40

ESPRESSO LASTGASP

Reduce is order dependent:

E.g., expand can't do anything with that produced by REDUCE 2.



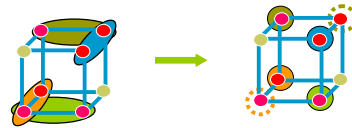
Maximal Reduce:

$$\underline{c}_i^M = SCC(c_i \cap \overline{F(i)}) = c_i \cap SCCC(F(i)_{c_i}) \quad \forall i$$

i.e., we reduce all cubes as if each were the first one.

Note:

$\{\underline{c}_1^M, \underline{c}_2^M, \dots\}$ is not a cover



41

ESPRESSO LASTGASP

Now EXPAND, but try to cover only $\underline{c}_j^M$'s.

■ We call $EXPAND(G, R)$, where $G = \{\underline{c}_1^M, \underline{c}_2^M, \dots, \underline{c}_k^M\}$

■ If a covering is possible, take the resulting prime:

$$f + d \supseteq p_i \supseteq \underline{c}_i^M \cup \underline{c}_j^M$$

and add to F:

$$\tilde{F} = F \cup \{p_i\}$$

Since F is a cover, so is $\tilde{F}$. Now apply IRREDUNDANT on $\tilde{F}$.

What about "supergasp" ?

Main Idea: Generally, think of ways to throw in a few more primes and then use IRREDUNDANT. If all primes generated, then just Quine-McCluskey



42