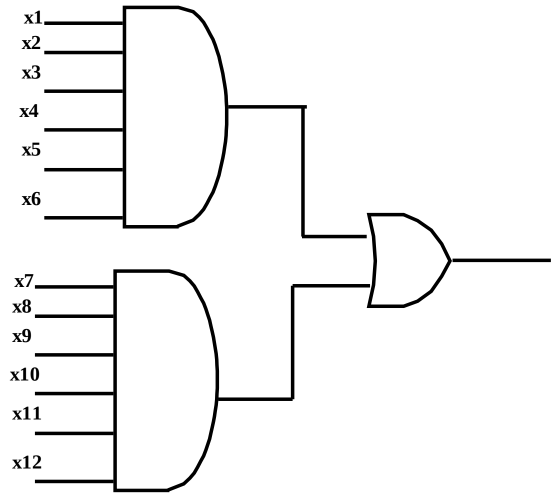
Heuristic Based two level Switching function minimization

Dr. Chandan Karfa

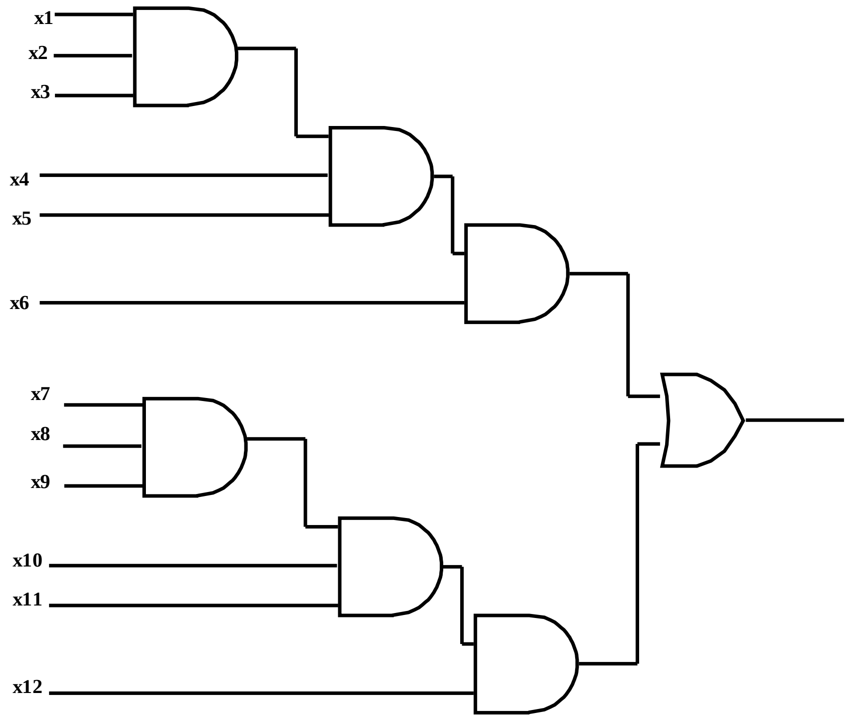
Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

Two-level vs Multi-level Logic



(a) Two level Implementation



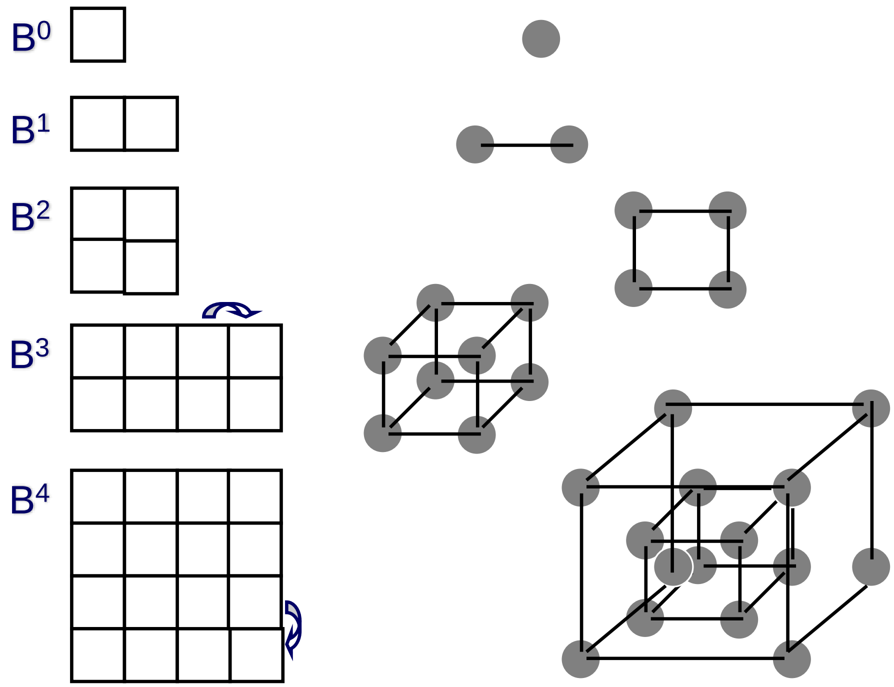
(a) Multi-level Implementation

we have a Boolean function as $f = x1.x2.x3.x4.x5.x6 + x7.x8.x9.x10.x11.x12$.

The Boolean Space B^n

$B = \{0,1\}$, $B^2 = \{0,1\} \times \{0,1\} = \{00, 01, 10, 11\}$, etc.

Karnaugh Maps: Boolean Spaces:



Boolean Functions

Boolean Function: $f(x): B^n \to B$

$B = \{0,1\}$

$x = (x_1, x_2, \ldots, x_n) \in B^n; x_i \in B$

$$\begin{array}{c|cc} & & x_2 \\ \hline x_1 & 0 & 1 \\ & 1 & 0 \end{array}$$

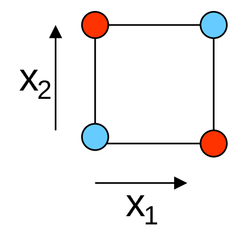
- $x_1, x_2, \ldots$ are variables
- $x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots$ are literals
- essentially: f maps each vertex of B^n to 0 or 1

● = on-set minterm ($f = 1$)

○ = off-set minterm ($f = 0$)

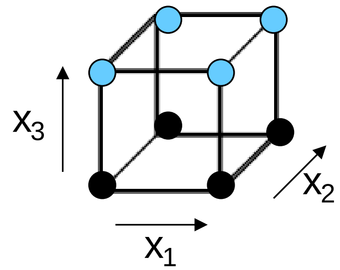
Example:

$$f = \{((x_1 = 0, x_2 = 0), 0), ((x_1 = 0, x_2 = 1), 1),$$
$$((x_1 = 1, x_2 = 0), 1), ((x_1 = 1, x_2 = 1), 0)\}$$



Set of Boolean Functions

- Truth Table or Function Table:



$X_1x_2x_3$	F
0 0 0	1
0 0 1	0
0 1 0	1
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	1
1 1 1	0

- There are 2^n vertices in input space B^n
- There are 2^{2^n} distinct logic functions.
  - Each subset of vertices is a distinct logic function: $f \subseteq B^n$

Representations of Boolean Functions

- Forms to represent Boolean Functions
  - Truth table
  - List of cubes: Sum of Products, Disjunctive Normal Form (DNF)
  - List of conjuncts: Product of Sums, Conjunctive Normal Form (CNF)
  - Binary Decision Tree, Binary Decision Diagram
  - Boolean formula
  - Boolean network

Cube

- A <u>cube</u> is defined as the product (AND) of a set of literal functions ("conjunction" of literals).

  Example:

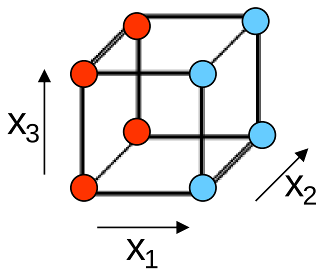
$$C = x_1 x'_2 x_3$$

Other examples of cubes:

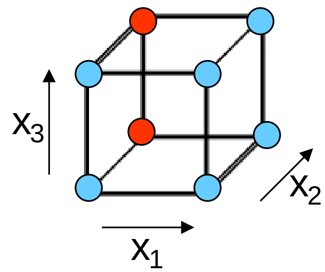
● = on-set minterm ($f = 1$)

○ = off-set minterm ($f = 0$)

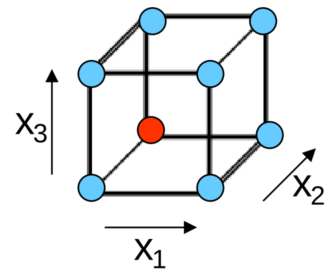
$c = \overline{x}_1$



$f = \overline{x}_1 x_2$



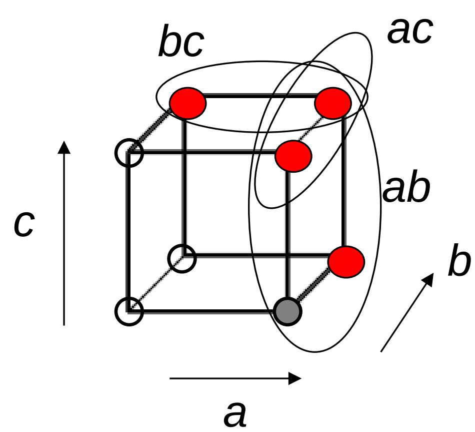
$f = \overline{x}_1 x_2 \overline{x}_3$



Definitions

- Minterm: Product term with all n variables presenet either in its tru or complement form.
  - abc, abc’
- Implicant: Two or more terms may be combined to get a term of reduced size that is covered by the function
  - F = abc + abc’. An implicant of f is ab
- Prime Implicant: If a term cannot be reduced further, then it is prime implicant
- Essential Prime Implicant: A prime implicant is essential if covers minterms which are not covered by others.

Cover minimization



= on-set minterm (*f = 1*)
= off-set minterm (*f = 0*)
= don't care-set minterm (*f = x*)

Note: each onset minterm is "covered" by at least one of the cubes and none of the offset minterms is covered.

- Two-level minimization seeks a *minimum size cover* (least number of cubes). Reason: *minimize number of product terms in PLA*

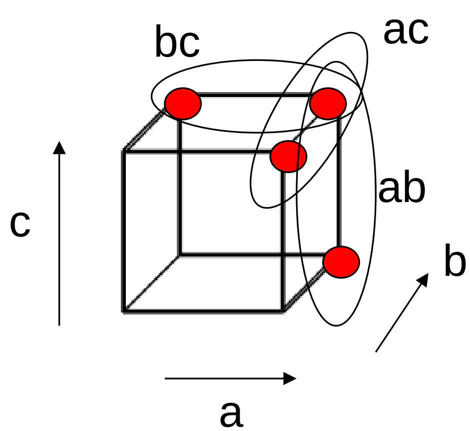
Irredundant Cubes

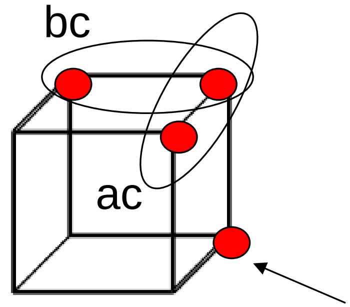
- Definition: Let $F = \{c_1, c_2, \ldots, c_k\}$ be a cover for f, i.e. $f = \sum_{i=1}^{k} c_i$

  A cube $c_i \in F$ is *irredundant* if $F \setminus \{c_i\} \neq f$

  A cover is *irredundant* if all its cubes are irredundant.

  Example: $f = ab + ac + bc$



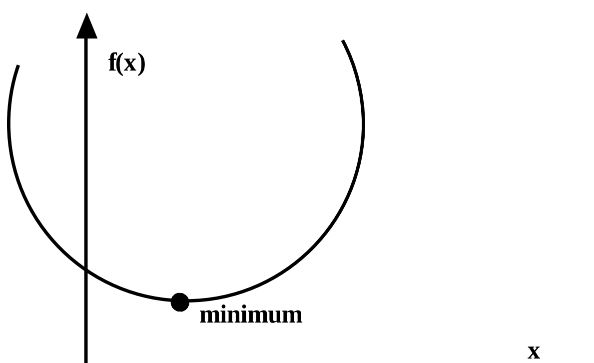


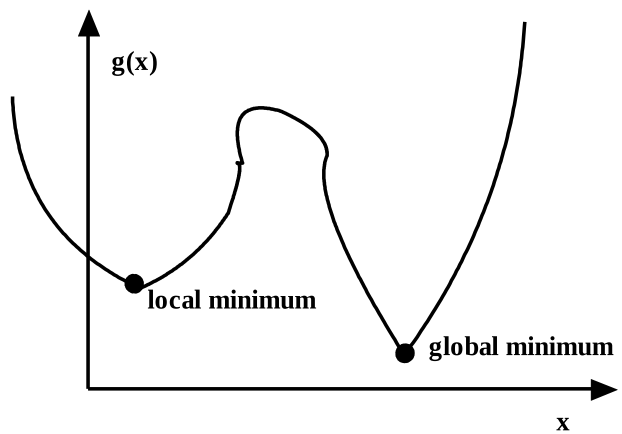
$F \setminus \{ab\} \neq f$

not covered,
so *ab* is irredundant

Two-level Logic Optimizations: Methods

- Find prime implicants and try to cover them using minimal number of covers
  - Karnaugh Map based method
  - Quine–McCluskey method
  - Heuristic based approach
    - ESPRESSO
- ESPRESSO is based on simulated annealing method





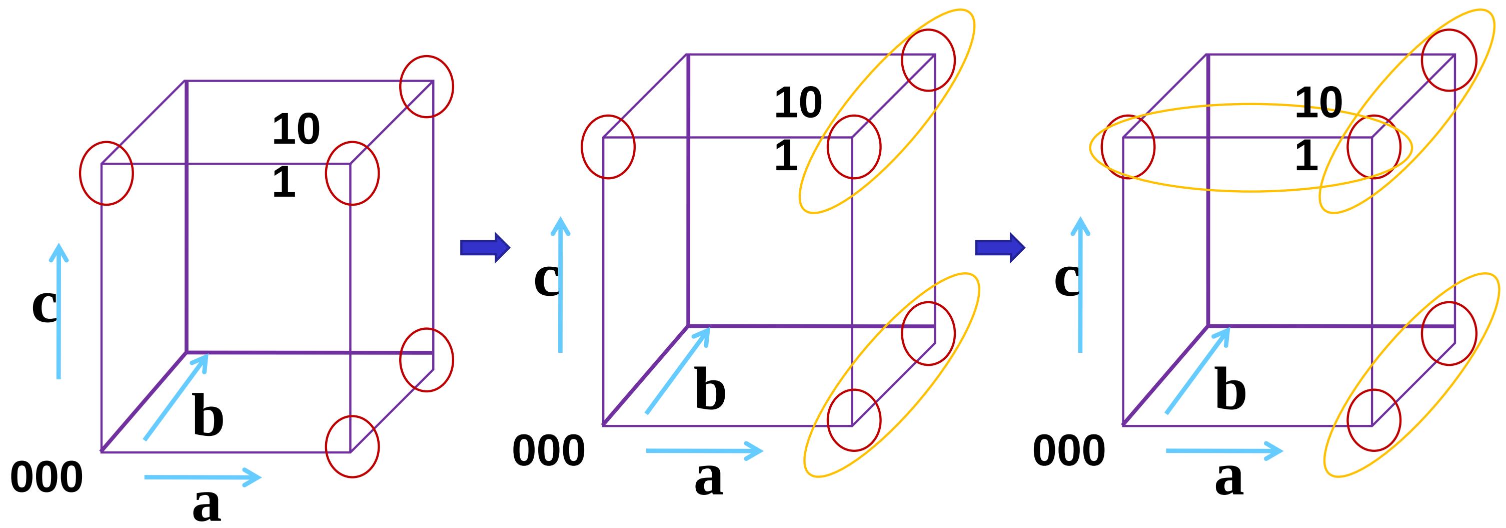
ESPRESSO

- Start from initial cover
  - May be the minterm itself
- Modify cover
  - Make it prime and irredundant,
  - Reduce, expand, irredundant operations
- Stop when no further improvement is possible or timed out

Expand

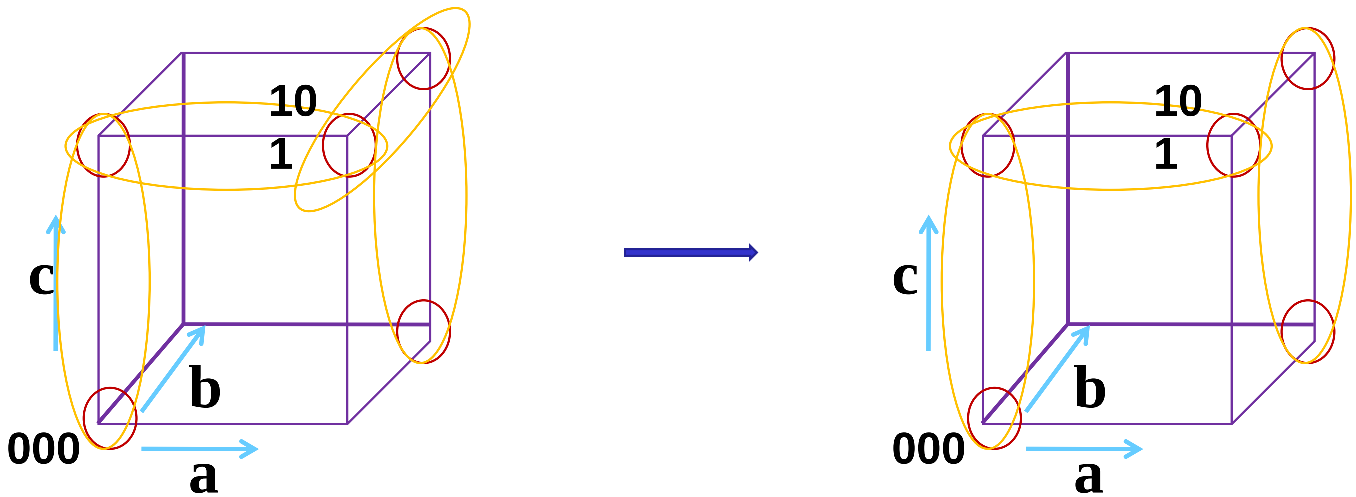
- Make each cube as large as possible without covering a point in the OFF-set.
- It tries to expand the cubes in F with neighboring cubes with nodes in the DC-set to form larger cubes.
- It takes essential sub-cubes and tries to expand them till they become prime sub-cubes
- Each non prime implicant is expanded to a prime and it is replaced by a prime implicant that contains it.

Expand



Irredundant

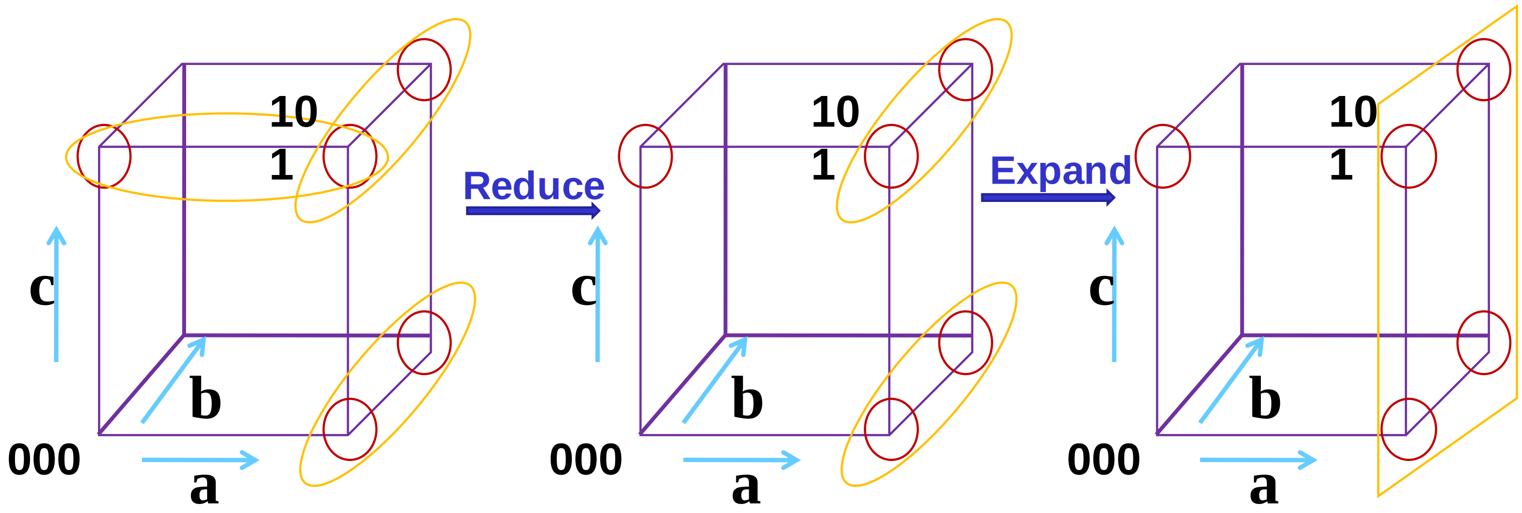
- Throw out redundant cubes
- Irredundant: Makes a cover minimal/irredundant covering the ON-set.
- Irredundant Cover: no proper subset is also cover



Reduce

- The cubes in the cover are reduced in size.
- Each implicant is reduced to a smaller one that is contained in.
- Reduced one and the remaining one are still cover the ON-set.
- Might exist another cover with fewer terms of fewer literals.
- This allow the newly formed cubes to expand in the different direction
- New larger cube can possibly be obtained by exploring the new direction

Reduce



ESPRESSO

```
Forig = ON-set; /* vertices with expression TRUE */
R = OFF-set; /* vertices with expression FALSE */
D = DC-set; /* vertices with expression DC */
F = expand(Forig, R); /* expand cubes against OFF-set */
F = irredundant(F, D); /* remove redundant cubes */
do {
        do {
                F = reduce(F, D); /* shrink cubes against ON-set */
                F = expand(F, R);
                F = irredundant(F, D);
        } until cost is “stable”;
        /* perturb solution */
        G = reduce_gasp(F, D); /* add cubes that can be reduced */
        G = expand_gasp(G, R); /* expand cubes that cover another */
        F = irredundant(F+G, D);
} until time is up;
ok = verify(F, Forig, D); /* check that result is correct */
```

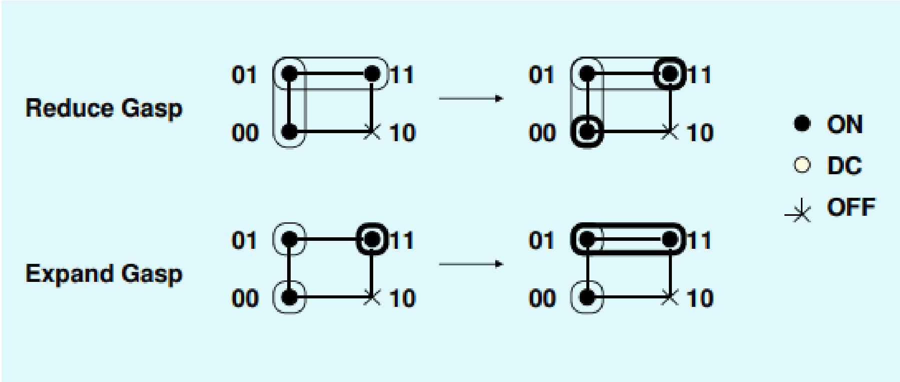
Reduce_gasp and expand_gasp

Reduce_gasp:

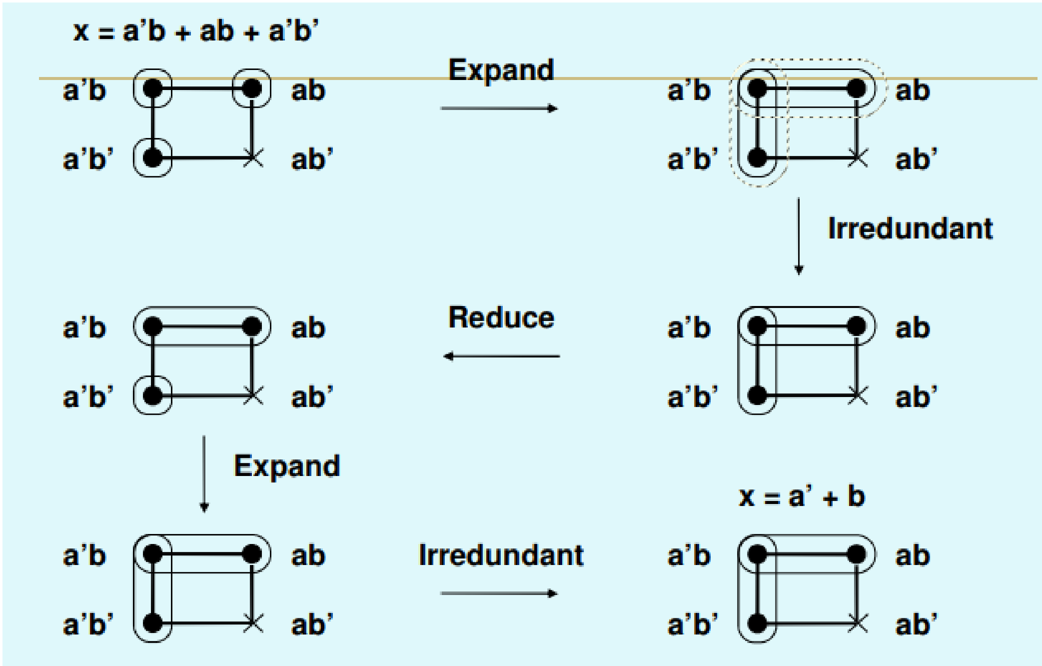
- For each cube in F, add those sub-cubes of cubes that are not covered by other cubes
- It uses D to ensure that new sub-cubes are not produces for just some DC nodes.

Expand_gasp

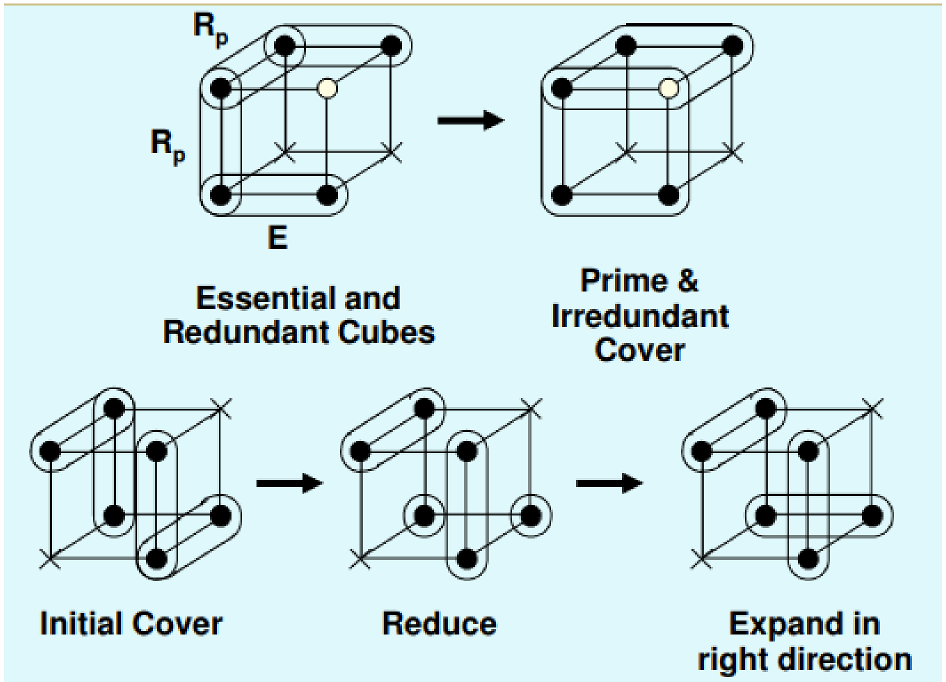
- Expand subcubes and add them if they cover another cube.
- Later use “irredundant” to discard redundant cubes.


Reduce Gasp
01
11
00
10
Expand Gasp
ON
DC
OFF

An example



Cost stable



R_p
R_p
E
Essential and Redundant Cubes
Prime & Irredundant Cover
Initial Cover
Reduce
Expand in right direction

ESPRESSO Conclusions

- The algorithm successively generates new covers until no further improvement is possible.
- Produces near-optimal solutions.
- Used for PLA minimization, or as a sub-function in multilevel logic minimization.
- Can process very large circuits. – 10,000 literals, 100 inputs, 100 outputs
- Less than 15 minutes on a high-speed workstation