COE 561
Digital System Design & Synthesis
Two-Level Logic Synthesis

Dr. Aiman H. El-Maleh

Computer Engineering Department

King Fahd University of Petroleum & Minerals

[Adapted from slides of Prof. G. De Micheli: Synthesis & Optimization of Digital Circuits]

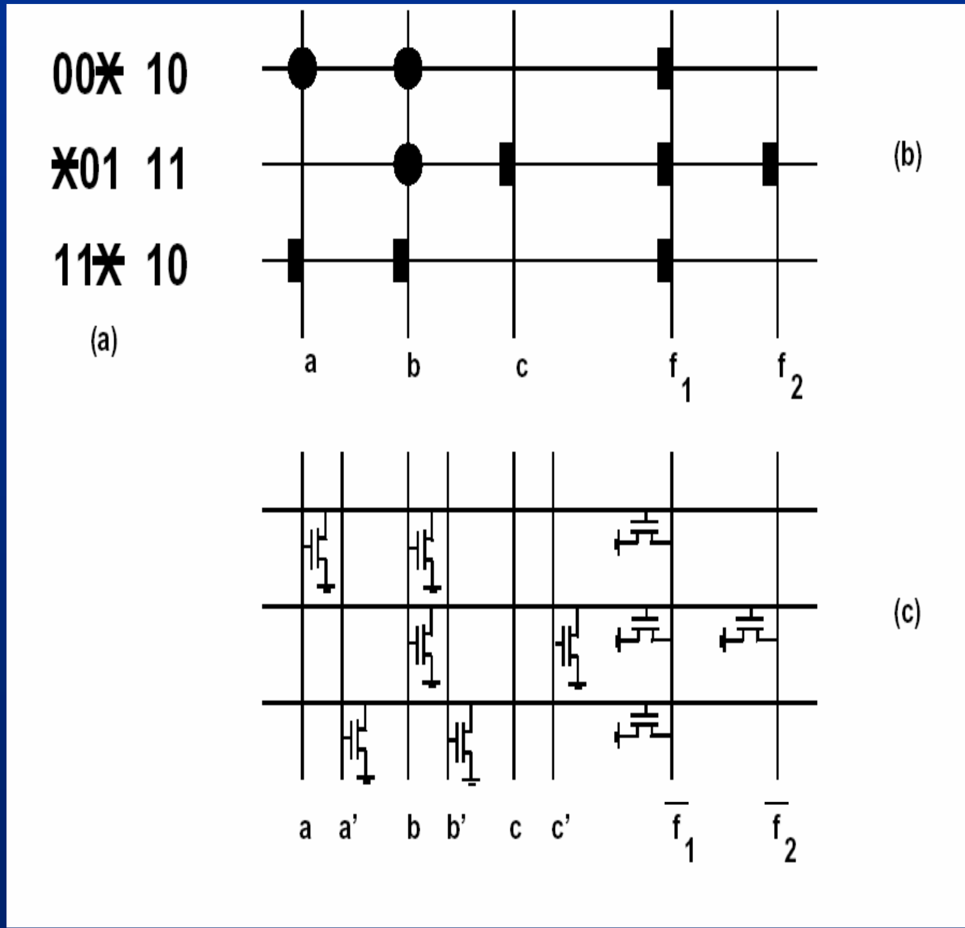
Outline

- **Programmable Logic Arrays**
- **Definitions**
- **Positional Cube Notation**
- **Operations on Logic Covers**
- **Exact Two-Level Optimization**
- **Heuristic Two-Level Optimization**
  - Expand
  - Reduce
  - Reshape
  - Irredundant
- **Espresso**
- **Testability Properties of Two-Level Logic**

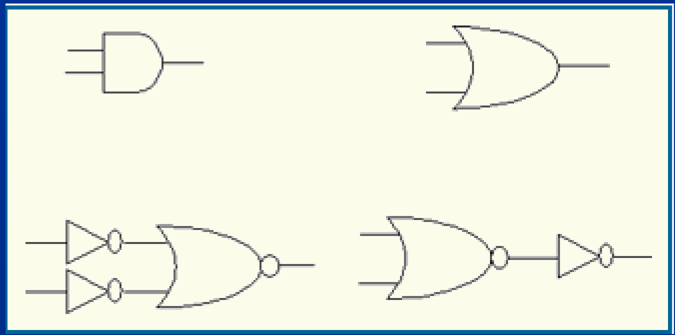
Programmable Logic Arrays …

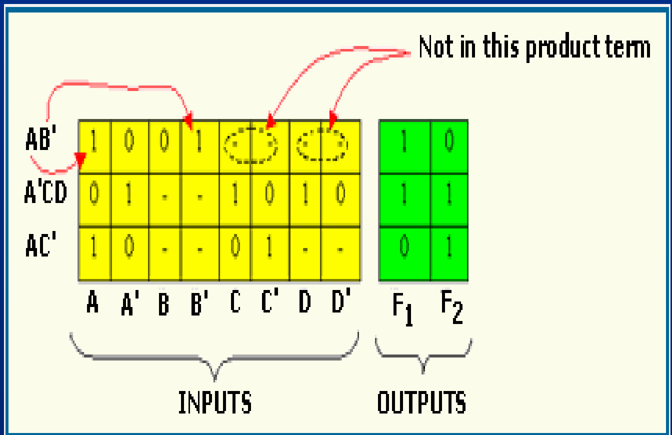
- Macro-cells with rectangular structure.
- Implement any multi-output function.
- Layout easily generated by module generators.
- Fairly popular in the seventies/eighties (NMOS).
- Still used for control-unit implementation.

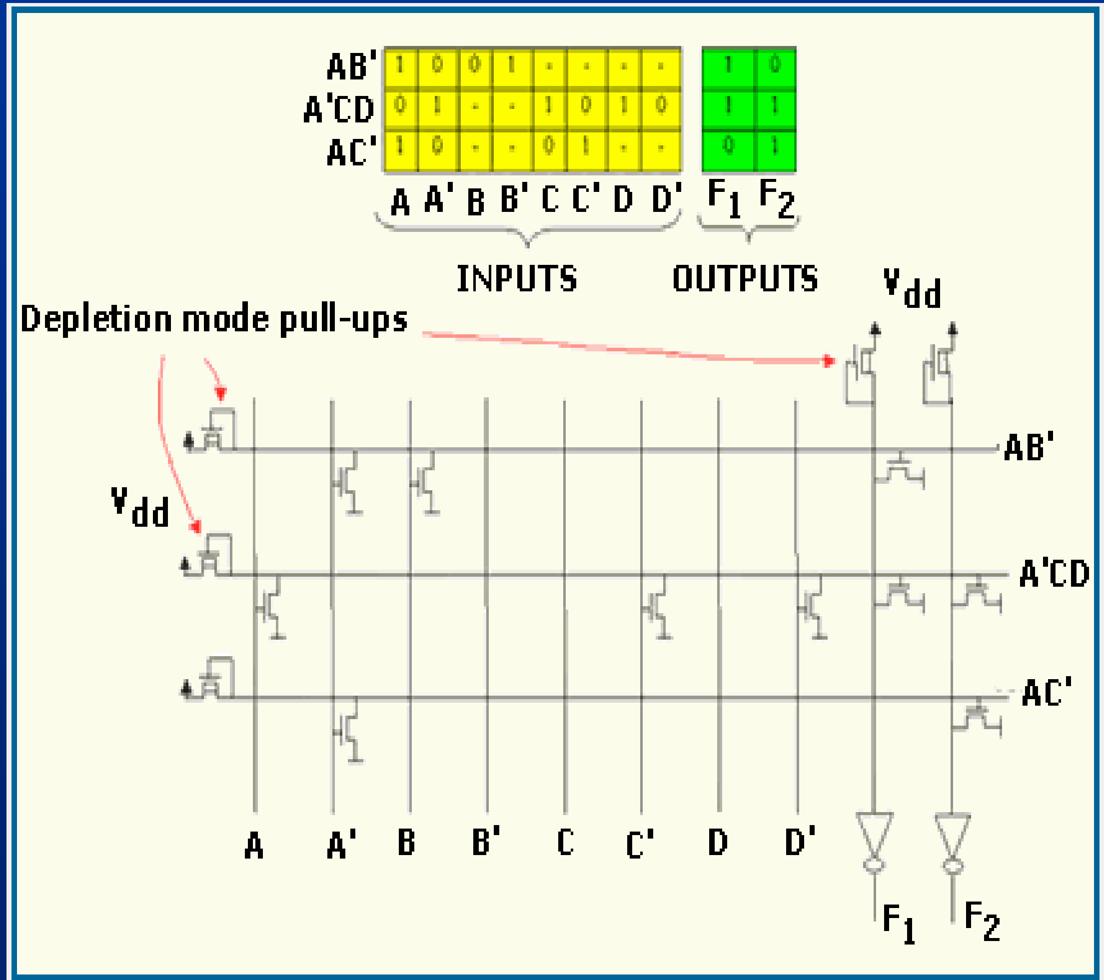
$f_1 = a'b' + b'c + ab$ $f_2 = b'c$



… Programmable Logic Arrays







Two-Level Optimization

- **Assumptions**
  - Primary goal is to reduce the number of implicants.
  - All implicants have the same cost.
  - Secondary goal is to reduce the number of literals.
- **Rationale**
  - Implicants correspond to PLA rows.
  - Literals correspond to transistors.

Definitions …

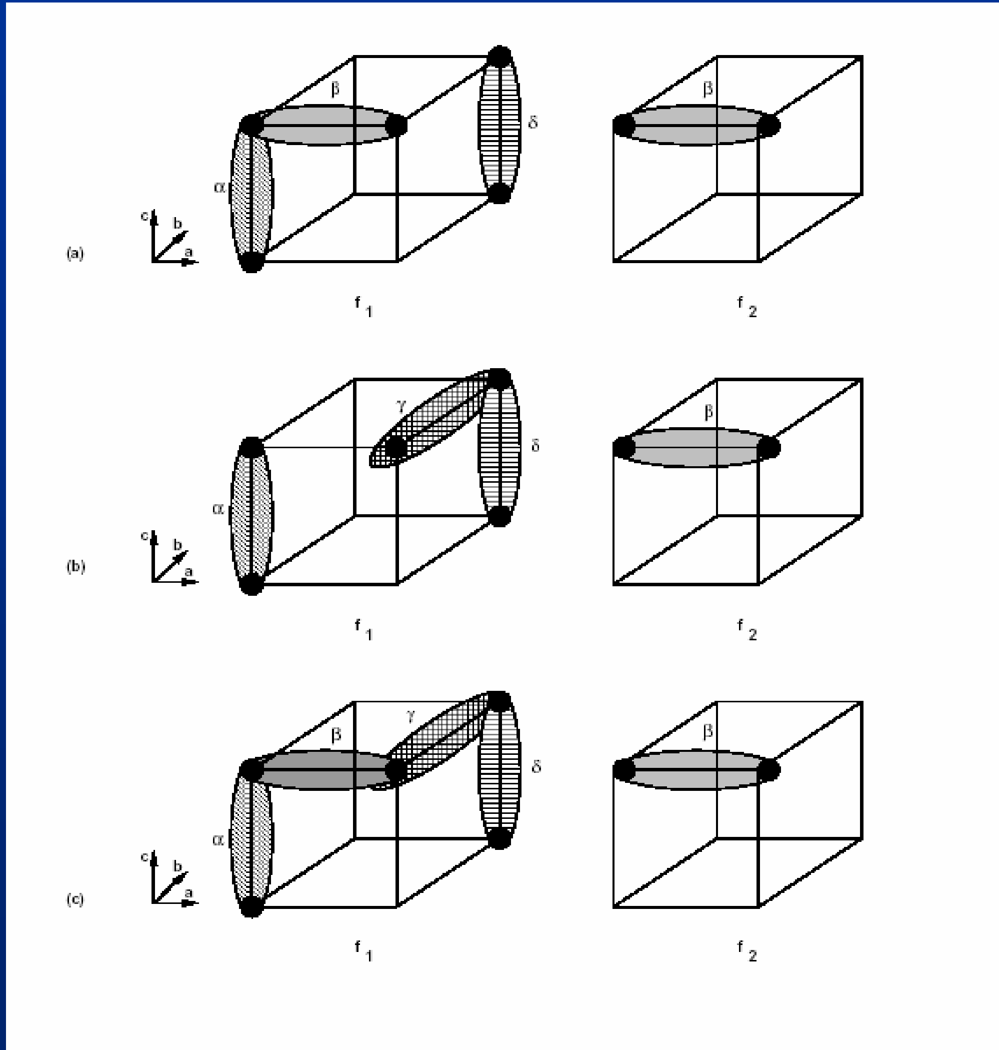
- **A cover of a Boolean function is a set of implicants that covers its minterms.**
- **Minimum cover**
  - Cover of the function with minimum number of implicants.
  - Global optimum.
- **Minimal cover or irredundant cover**
  - Cover of the function that is not a proper superset of another cover.
  - No implicant can be dropped.
  - Local optimum.
- **Minimal cover w.r.t. 1-implicant containment**
  - No implicant is contained by another one.
  - Weak local optimum.

… Definitions …

$f_1 = a'b'c' + a'b'c + ab'c$
$+ abc + abc'$

$f_2 = a'b'c + ab'c$

- ***(a) cover is minimum.***
- ***(b) cover is minimal.***
- ***(c) cover is minimal* w.r.t. 1-implicant containment.**



… Definitions …

- **Prime implicant**
  - Implicant not contained by any other implicant.
- **Prime cover**
  - Cover of prime implicants.
- **Essential prime implicant**
  - There exist some minterm covered only by that prime implicant.

The Positional Cube Notation

- Encoding scheme
  - **One column for each variable.**
  - **Each column has 2 bits.**
- *Example: f = a'd' + a'b + ab' + ac'd*

∅	00
0	10
1	01
*	11

a'd'	10	11	11	10
a'b	10	01	11	11
ab'	01	10	11	11
ac'd	01	11	10	01

- Operations
  - **Intersection: AND**
  - **Union: OR**

Operations on Logic Covers

- The intersection of two implicants is the largest cube contained in both. (bitwise AND)
- The supercube of two implicants is the smallest cube containing both. (bitwise OR)
- The distance between two implicants is the number of empty fileds in their intersection.
- An implicant covers another implicant when the bits of the former are greater than or equal to those of the latter.
- Recursive paradigm
  - Expand about a variable.
  - Apply operation to cofactors.
  - Merge results.
- Unate heuristics
  - Operations on unate functions are simpler.
  - Select variables so that cofactors become unate functions.

Cofactor Computation

- **Let $\alpha = a_1 a_2 \ldots a_n$ and $\beta = b_1 b_2 \ldots b_n$**
- **Cofactor of α w.r. to β**
  - Void when α does not intersect β (i.e. distance is ≥ 1)
  - $a_1 + b_1'$ $a_2 + b_2'$ . . . $a_n + b_n'$
- **Cofactor of a set C = $\{\gamma_i\}$ w.r. to β**
  - Set of cofactors of γ_i w.r. to β.
- **Example: f = a'b'+ab**
  - a'b'   10 10
  - ab   01 01
  - Cofactor w.r. to (a) 01 11
    - First row: void.
    - Second row: 11 01.
  - Cofactor $f_a = b$

Sharp Operation

- The **sharp operation** α # β returns the sets of implicants covering all minterms covered by α and not by β.
- Let $\alpha = a_1 a_2 \ldots a_n$ and $\beta = b_1 b_2 \ldots b_n$

$$\alpha \# \beta = \begin{cases} a_1.b'_1 \quad a_2 \quad \ldots \quad a_n \\ a_1 \quad a_2.b'_2 \quad \ldots \quad a_n \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ a_1 \quad a_2 \quad \ldots \quad a_n.b'_n \end{cases}$$

- **Example: compute complement of cube ab**
  - 11 11 # 01 01 = {10 11; 11 10}= a'+b'

Disjoint Sharp Operation ⓐ#

- The **disjoint sharp operation** $\alpha \circledast \beta$ returns the sets of implicants covering all minterms covered by α and not by β such that all implicants are **disjoint**.
- Let $\alpha = a_1 a_2 \ldots a_n$ and $\beta = b_1 b_2 \ldots b_n$

$$\alpha \circledast \beta = \begin{cases} a_1.b'_1 & a_2 & \ldots & a_n \\ a_1.b_1 & a_2.b'_2 & \ldots & a_n \\ \ldots\ldots\ldots\ldots\ldots\ldots & & & \\ a_1.b_1 & a_2.b_2 & \ldots & a_n.b'_n \end{cases}$$

- **Example: compute complement of cube ab**
  - 11 11 ⓐ# 01 01 = {10 11; 01 10}= a'+ab'

Consensus

- Let $\alpha=a_1a_2\ldots a_n$ and $\beta=b_1b_2\ldots b_n$

$$\text{Consensus}(\alpha,\beta)=\begin{cases} a_1+b_1 & a_2.b_2 & \ldots & a_n.b_n \\ a_1.b_1 & a_2+b_2 & \ldots & a_n.b_n \\ \ldots\ldots\ldots\ldots & & & \\ a_1.b_1 & a_2.b_2 & \ldots & a_n+b_n \end{cases}$$

- Consensus is void when two implicants have distance larger than or equal to 2.
- Yields a single implicant when distance is 1.
- Example: α=01 10 01 and β =01 11 10
  - Consensus(α,β)= {01 10 00, 01 11 00, 01 10 11}=01 10 11=ab'

Computation of all Prime Implicants …

- **Let f= x f_x + x' $f_{x'}$**
- **There are three possibilities for a prime implicant of f**
  - It is a prime of x f_x i.e. a prime of f_x
  - It is a prime of x' $f_{x'}$ i.e. a prime of $f_{x'}$
  - It is the consensus of two implicants one in x f_x and one in x' $f_{x'}$

$$P(f) = SCC(\,(x \cap P(F_x)) \cup (\overline{x} \cap P(F_{\overline{x}}))$$
$$\cup\, CONSENSUS(\,(x \cap P(F_x)), (\overline{x} \cap P(F_{\overline{x}}))\,)\,)$$

- **A unate cover, F, with SCC contains all primes.**
  - P(F)=SCC(F)
  - Each prime of a unate function is essential.

… Computation of all Prime Implicants

- **Example: f=ab + ac + a’**
  - Let us choose to split the binate variable a
  - Note that $f_{a'}$ is tautology; $P(f_{a'})$=U; C(a’) ∩ $P(f_{a'})$ =10 11 11=P1=a’
  - $P(f_a)$= {11 01 11; 11 11 01}=b+c; C(a) ∩ $P(f_a)$={01 01 11; 01 11 01}=P2={ab, ac}
  - Consensus(P1,P2)= {11 01 11; 11 11 01}={b,c}
  - P(F)=SCC{10 11 11; 01 01 11; 01 11 01; 11 01 11; 11 11 01}

    = {a’, ab, ac, b, c}

    = {10 11 11; 11 01 11; 11 11 01}

    = {a’, b, c}

Tautology ...

- **Check if a function is always TRUE.**
- **Plays an important rule in all algorithms for logic optimization.**
- **Recursive paradigm**
  - Expand about a variable.
  - If all cofactors are TRUE then function is a tautology.
- **TAUTOLOGY**
  - The cover has a row of all 1s (Tautology cube).
  - The cover depends on one variable only, and there is no column of 0s in that field.
- **NO TAUTOLOGY**
  - The cover has a column of 0s (A variable that never takes a certain value).
- **When a cover is the union of two subcovers that depend on disjoint subsets of variables, then check tautology in both subcovers.**

… Tautology

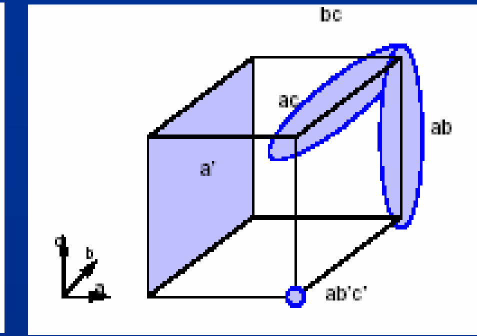
- **Unate heuristics**
  - If cofactors are unate functions, additional criteria to determine tautology.
  - Faster decision.
- **If a function is expanded in a unate variable, only one cofactor needs to be checked for tautology**
  - Positive unate in variable xi, $f_{xi} \supseteq f_{xi'}$; only $f_{xi'}$ needs to be checked for tautology.
  - Negative unate in variable xi, $f_{xi} \subseteq f_{xi'}$; only f_{xi} needs to be checked for tautology.
- **A cover is not tautology if it is unate and there is not a row of all 1's.**

Tautology Example

- **f = ab+ac+ab'c' +a'**
- **Select variable a.**
  - Cofactor w.r.to a'
    - 11 11 11 => Tautology.
  - Cofactor w.r.to a is:

11	01	11
11	11	01
11	10	10

01	01	11
01	11	01
01	10	10
10	11	11



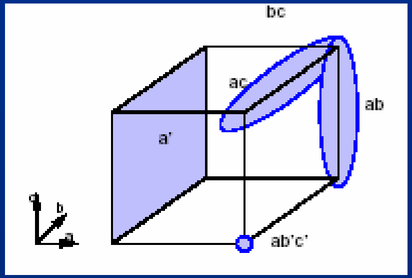
- **Select variable b.**
  - Cofactor w.r. to b' is:

11	11	01
11	11	10

    - Depends on a single variable, no column of 0's => Tautology.
  - Cofactor w.r. to b is: 11 11 11 => Tautology
- **Function is a TAUTOLOGY.**

Containment

- **Theorem**
  - A cover F contains an implicant α iff F_α is a tautology.
- **Consequence**
  - Containment can be verified by the tautology algorithm.
- **Example**
  - f = ab+ac+ab'c'+a'
  - Check covering of *bc: C(bc)* 11 01 01
  - Take the cofactor



01	11	11
01	11	11
10	11	11

01	01	11
01	11	01
01	10	10
10	11	11

  - Tautology; *bc* is contained by *f*

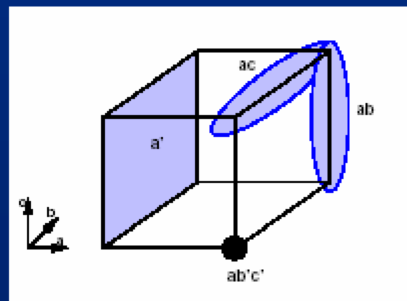
Complementation

- **Recursive paradigm**
  - $f = x \cdot f_x + x' \cdot f_{x'} \Rightarrow f' = x \cdot f'_x + x' \cdot f'_{x'}$
- **Steps**
  - Select a variable.
  - Compute cofactors.
  - Complement cofactors.
- **Recur until cofactors can be complemented in a straightforward way.**
- **Termination rules**
  - The cover *F* is void. Hence its complement is the universal cube.
  - The cover *F* has a row of 1s. Hence *F* is a tautology and its complement is void.
  - All implicants of *F* depend on a single variable, and there is not a column of 0s. The function is a tautology, and its complement is void.
  - The cover *F* consists of one implicant. Hence the complement is computed by De Morgan's law.

Complement of Unate Functions…

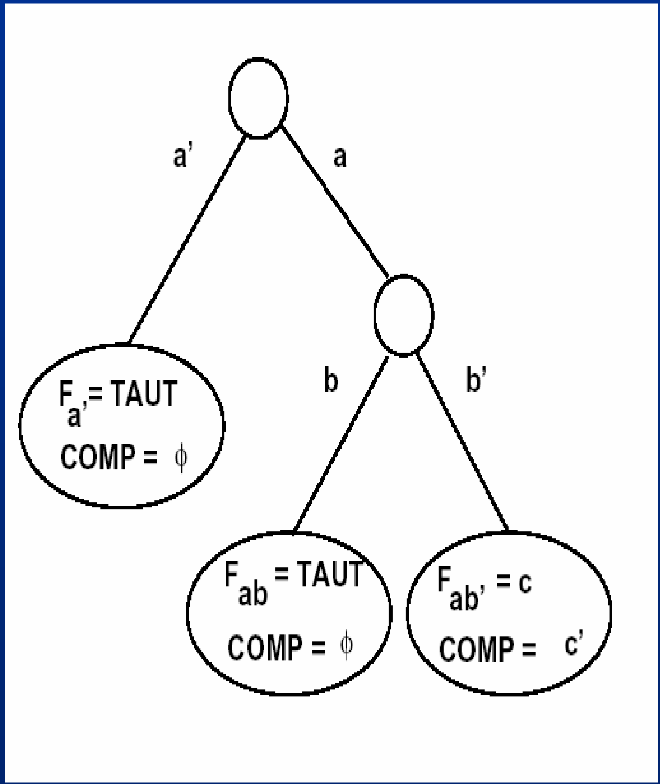
- **Theorem**
  - If f is positive unate in variable x: $f' = f'_x + x' \cdot f'_{x'}$.
  - If f is negative unate in variable x: $f' = x \cdot f'_x + f'_{x'}$.
- **Consequence**
  - Complement computation is simpler.
- **Heuristic**
  - Select variables to make the cofactors unate.
- **Example: *f = ab+ac+a'***
  - Select binate variable *a.*
  - Compute cofactors
    - $F_{a'}$ is a tautology, hence $F'_{a'}$ is void.
    - F_a yields:

11	01	11
11	11	01



… Complement of Unate Functions

- Select unate variable *b*.
  - Compute cofactors
    - F_{ab} is a tautology, hence F'_{ab} is void.
    - $F_{ab'}$ = 11 11 01 and its complement is 11 11 10.
  - Re-construct complement
    - **11 11 10 intersected with $C(b')$ = 11 10 11 yields 11 10 10.**
    - **11 10 10 intersected with $C(a)$ = 01 11 11 yields 01 10 10.**
- Complement: F' = 01 10 10.



Two-Level Logic Minimization

- Exact methods
  - Compute minimum cover.
  - Often impossible for large functions.
  - Based on derivatives of Quine-McCluskey method.
  - Many minimization problems can be now solved exactly.
  - Usual problems are memory size and time.
- Heuristic methods
  - Compute minimal covers (possibly minimum).
  - Large variety of methods and programs
    - MINI, PRESTO, ESPRESSO.

Exact Two-Level Logic Minimization

- Quine's theorem
  - There is a minimum cover that is prime.
- Consequence
  - Search for minimum cover can be restricted to prime implicants.
- Quine McCluskey method
  - Compute prime implicants.
  - Determine minimum cover.
- Prime implicant table
  - Rows: minterms.
  - Columns: prime implicants.
  - Exponential size
    - 2^n minterms.
    - Up to $3^n/n$ prime implicants.

Remark:
• Some functions have much fewer primes.
• Minterms can be grouped together.

Prime Implicant Table Example

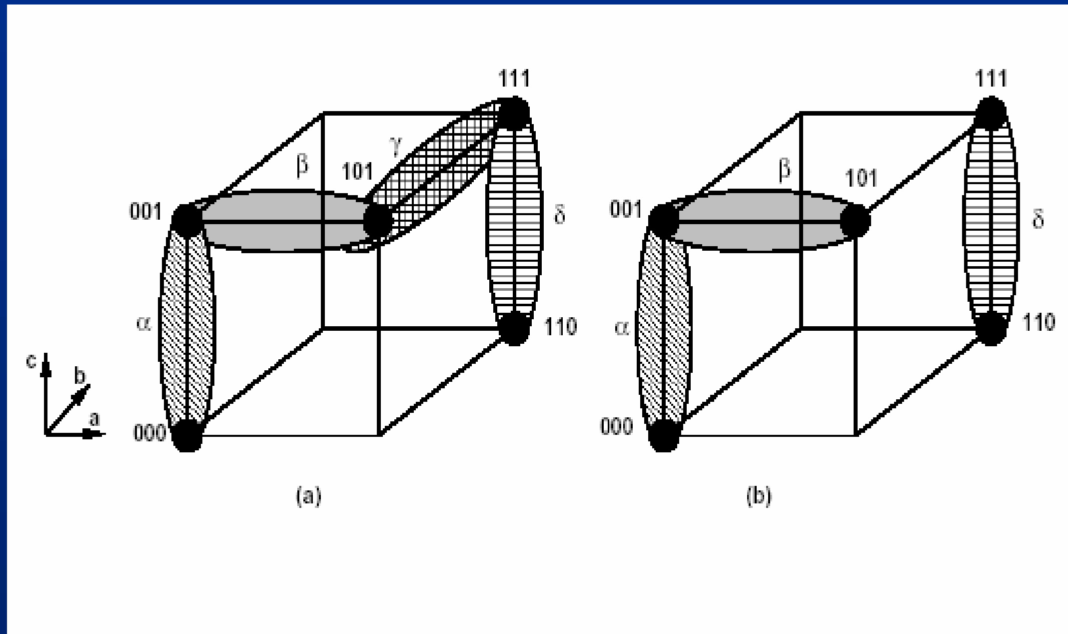
Function: f = a’b’c’+a’b’c+ab’c+abc’+abc

Prime Implicants

α	00*	1
β	*01	1
γ	1*1	1
δ	11*	1

Implicant Table

	α	β	γ	δ
000	1	0	0	0
001	1	1	0	0
101	0	1	1	0
111	0	0	1	1
110	0	0	0	1



Minimum Cover: Early Methods

- Reduce table
  - Iteratively identify essentials, save them in the cover, remove covered minterms.
  - Use row and column dominance.
- Petrick's method
  - Write covering clauses in POS form.
  - Multiply out POS form into SOP form.
  - Select cube of minimum size.
  - Remark
    - Multiplying out clauses is **exponential**.
- Petrick's method example
  - POS clauses: $(\alpha)(\alpha+\beta)(\beta+\gamma)(\gamma+\delta)(\delta) = 1$
  - SOP form: $\alpha\,\beta\,\delta + \alpha\,\gamma\,\delta = 1$
  - Solutions
    - $\{\alpha, \beta, \delta\}$
    - $\{\alpha, \gamma, \delta\}$

	α	β	γ	δ
000	1	0	0	0
001	1	1	0	0
101	0	1	1	0
111	0	0	1	1
110	0	0	0	1

Matrix Representation

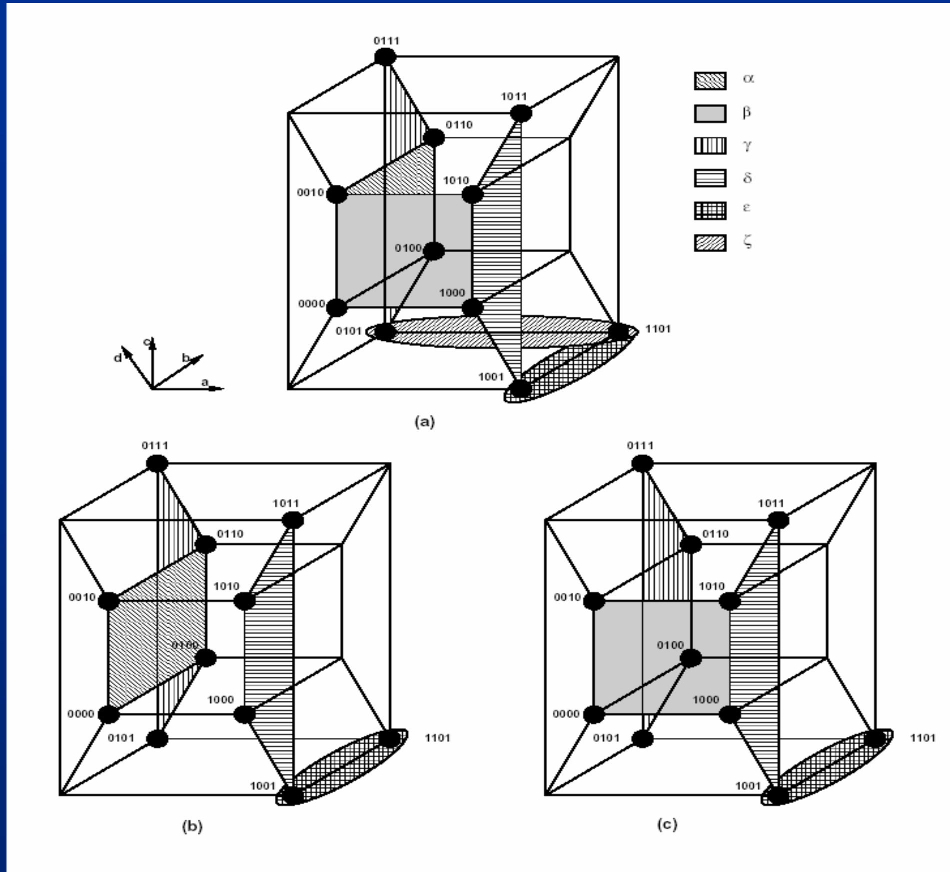
- View table as Boolean matrix: A.
- Selection Boolean vector for primes: x.
- Determine x such that
  - **A x ≥ 1.**
  - Select enough columns to cover all rows.
- Minimize cardinality of x
  - Example: $x = [1101]^T$
- Set covering problem
  - **A set S. (Minterm set).**
  - **A collection C of subsets. (Implicant set).**
  - **Select fewest elements of C to cover S.**

	α	β	γ	δ
000	1	0	0	0
001	1	1	0	0
101	0	1	1	0
111	0	0	1	1
110	0	0	0	1

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

ESPRESSO-EXACT

- Exact minimizer [Rudell].
- Exact branch and bound covering.
- Compact implicant table
  - Group together minterms covered by the same implicants.
- Very efficient. Solves most problems.



0000	1
0010	1
0100	1
0110	1
1000	1
1010	1
0101	1
0111	1
1001	1
1011	1
1101	1

α	0**0	1
β	*0*0	1
γ	01**	1
δ	10**	1
ϵ	1*01	1
ζ	*101	1

	α	β	ϵ	ζ
0000,0010	1	1	0	0
1101	0	0	1	1

Implicant table after reduction

Minimum Cover: Recent Developments

- Many minimization problems can be solved exactly today.
- Usually bottleneck is table size.
- Implicit representation of prime implicants
  - Methods based on BDDs [COUDERT]
    - to represent sets.
    - to do dominance simplification.
  - Methods based on signature cubes [MCGEER]
    - Represent set of primes.
    - A signature cube identifies uniquely the set of primes covering each minterm.
    - It is the largest cube of the intersection of corresponding primes.
    - The set of maximal signature cubes defines a minimum canonical cover.

Heuristic Minimization Principles

- Provide irredundant covers with 'reasonably small' cardinality.
- Fast and applicable to many functions.
- Avoid bottlenecks of exact minimization
  - Prime generation and storage.
  - Covering.
- Local minimum cover
  - Given initial cover.
  - Make it prime.
  - Make it irredundant.
  - Iterative improvement
    - Improve on cardinality by 'modifying' the implicants.

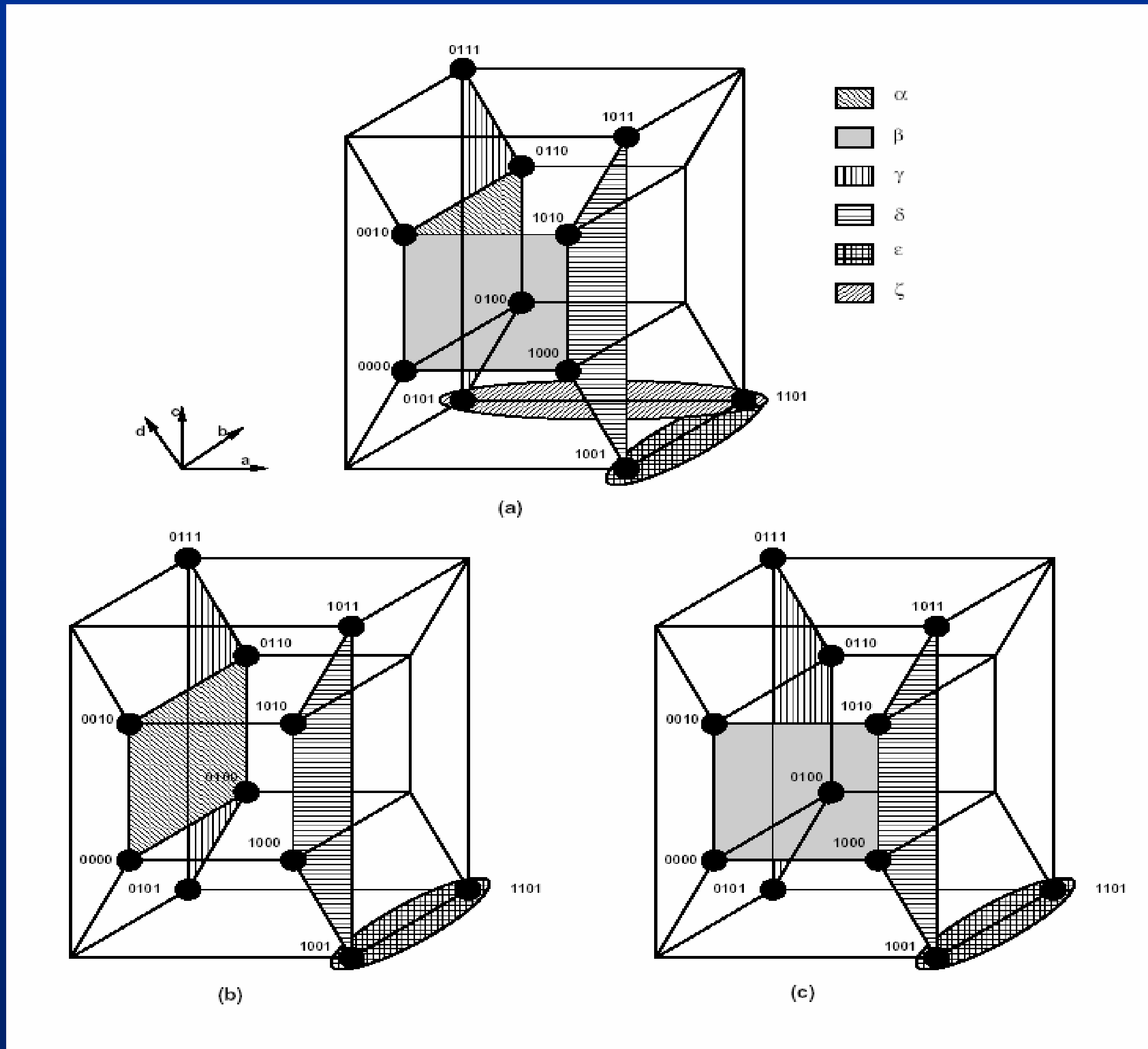
Heuristic Minimization Operators

- Expand
  - Make implicants prime.
  - Remove covered implicants w.r.t. single implicant containment.
- Irredundant
  - Make cover irredundant.
  - No implicant is covered by the remaining ones.
- Reduce
  - Reduce size of each implicant while preserving cover.
- Reshape
  - Modify implicant pairs: enlarge one and reduce the other.

Example: MINI

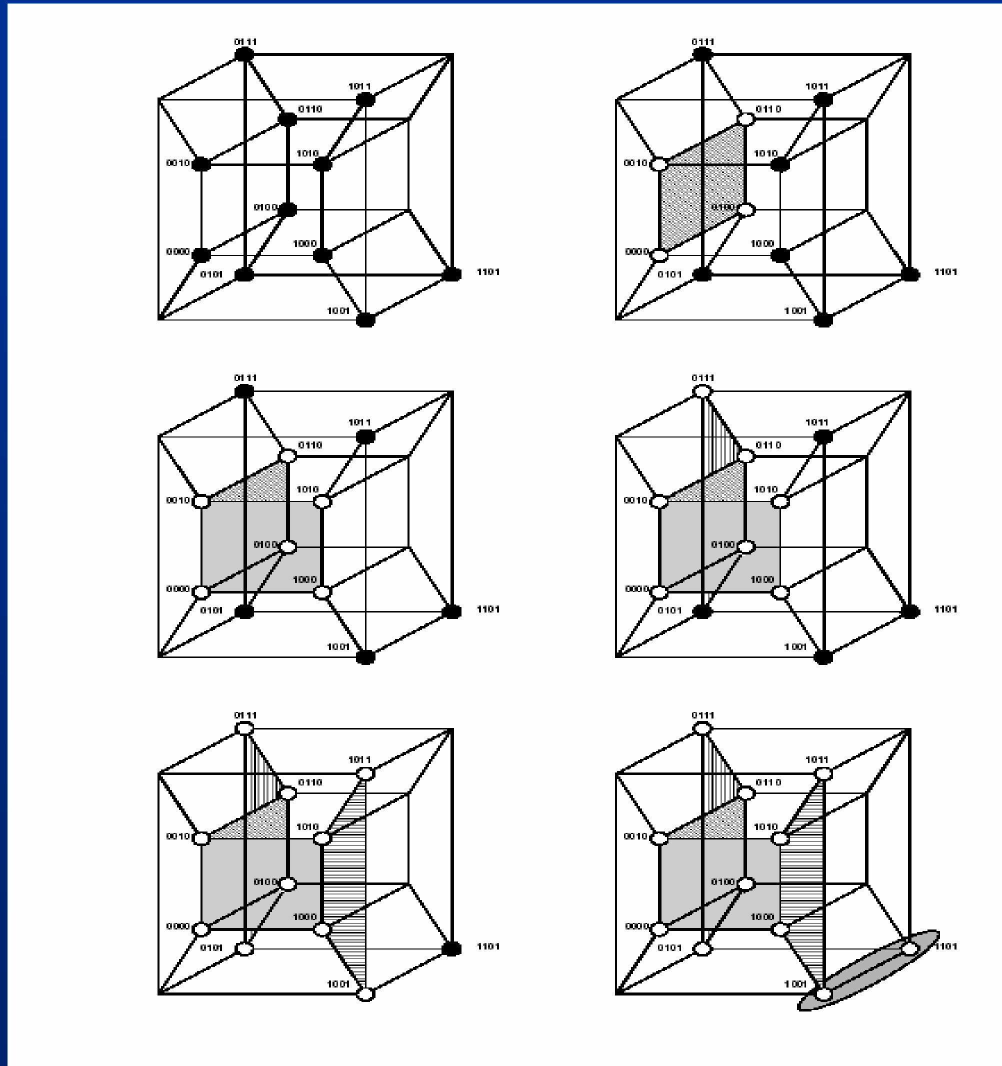
0000	1
0010	1
0100	1
0110	1
1000	1
1010	1
0101	1
0111	1
1001	1
1011	1
1101	1

α	0**0	1
β	*0*0	1
γ	01**	1
δ	10**	1
ϵ	1*01	1
ζ	*101	1



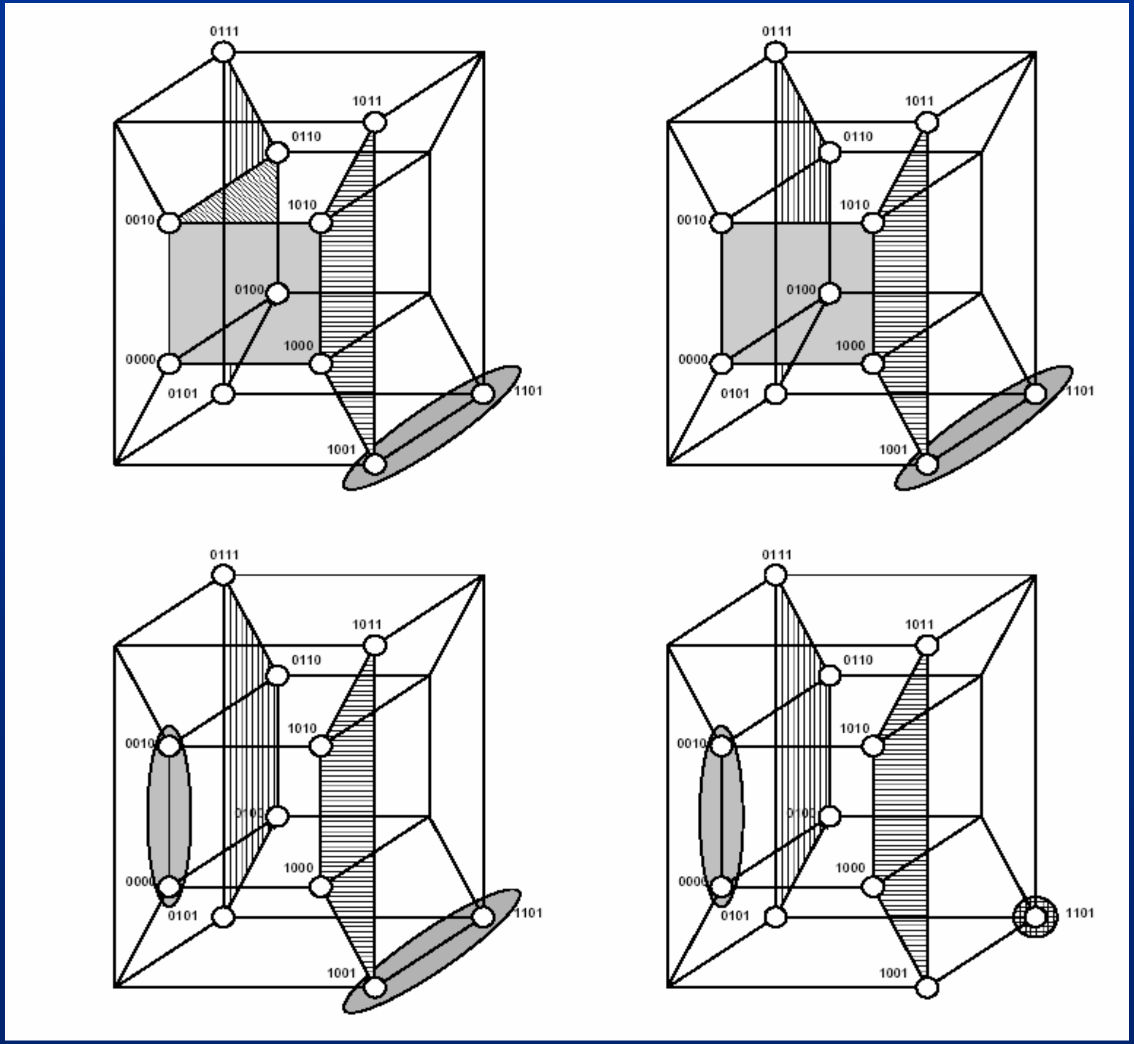
Example: Expansion

- Expand 0000 to α=0∗∗0.
  - Drop 0100, 0010, 0110 from the cover.
- Expand 1000 to β= ∗0∗0.
  - Drop 1010 from the cover.
- Expand 0101 to γ= 01∗∗ .
  - Drop 0111 from the cover.
- Expand 1001 to δ= 10∗∗.
  - Drop 1011 from the cover.
- Expand 1101 to ε= 1∗01.
- Cover is: {α, β, γ, δ, ε}
  - Prime.
  - Redundant.
  - Minimal w.r.t. scc.



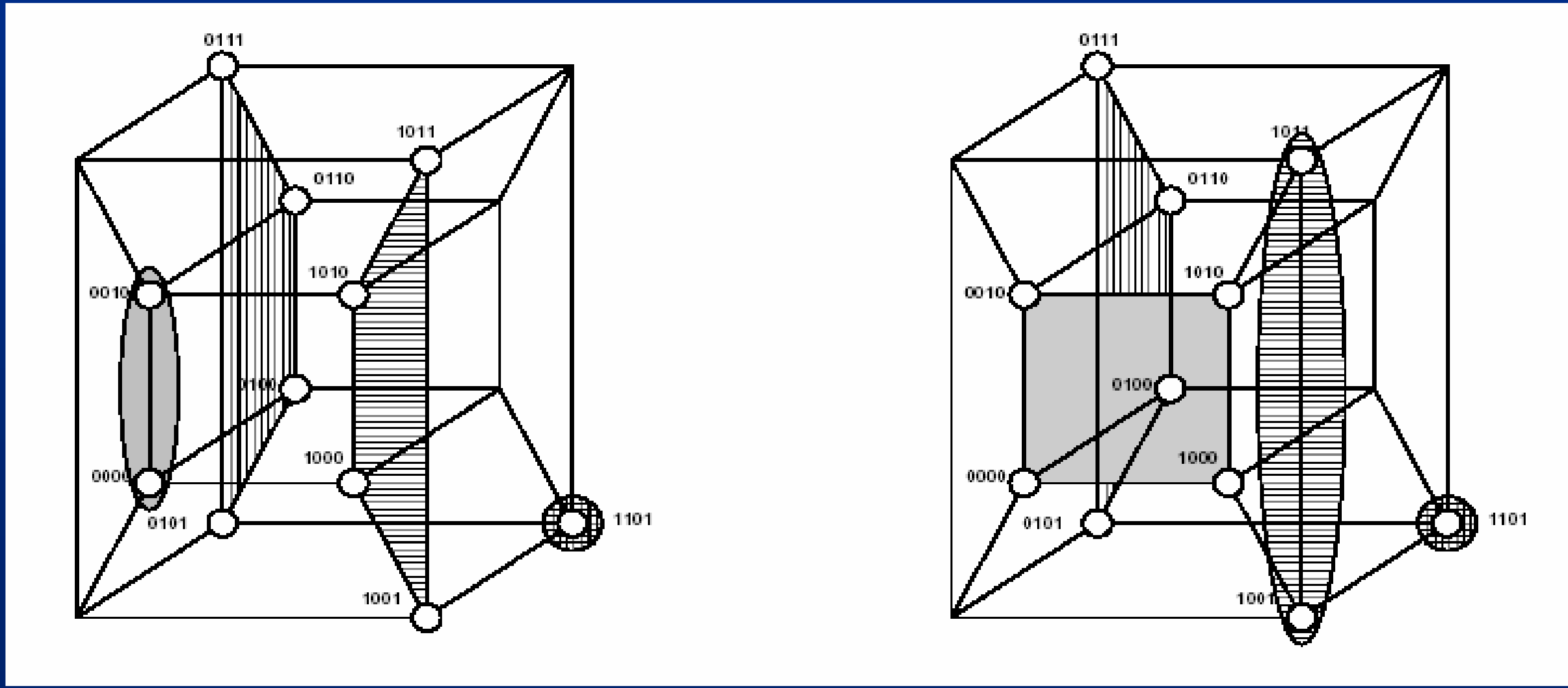
Example: Reduction

- Reduce α=0∗∗0 to nothing.
- Reduce β=∗0∗0 to $\tilde{\beta}$=00∗0
- Reduce ε=1∗01 to $\tilde{\varepsilon}$=1101
- Cover={$\tilde{\beta}$, γ, δ, $\tilde{\varepsilon}$}



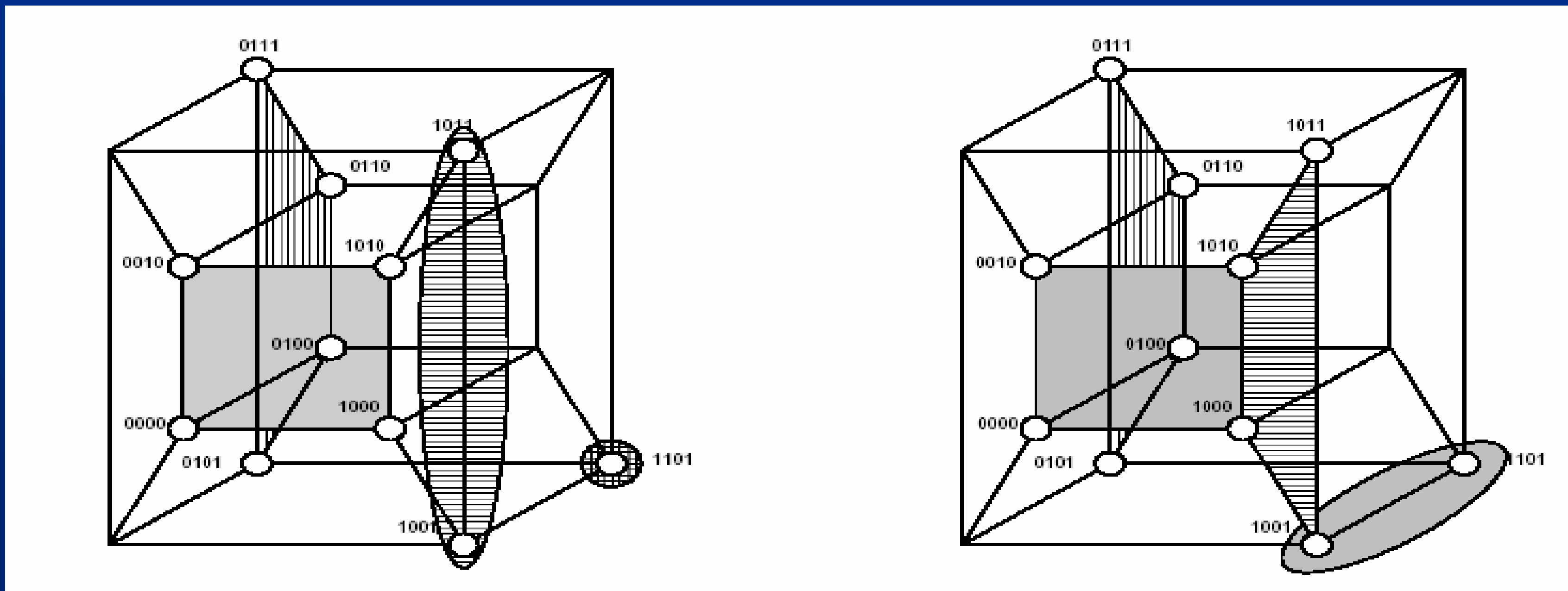
Example: Reshape

- Reshape {β~, δ} to {β, δ~}
  - δ~=10∗1
- Cover={β, γ, δ~, ε~}



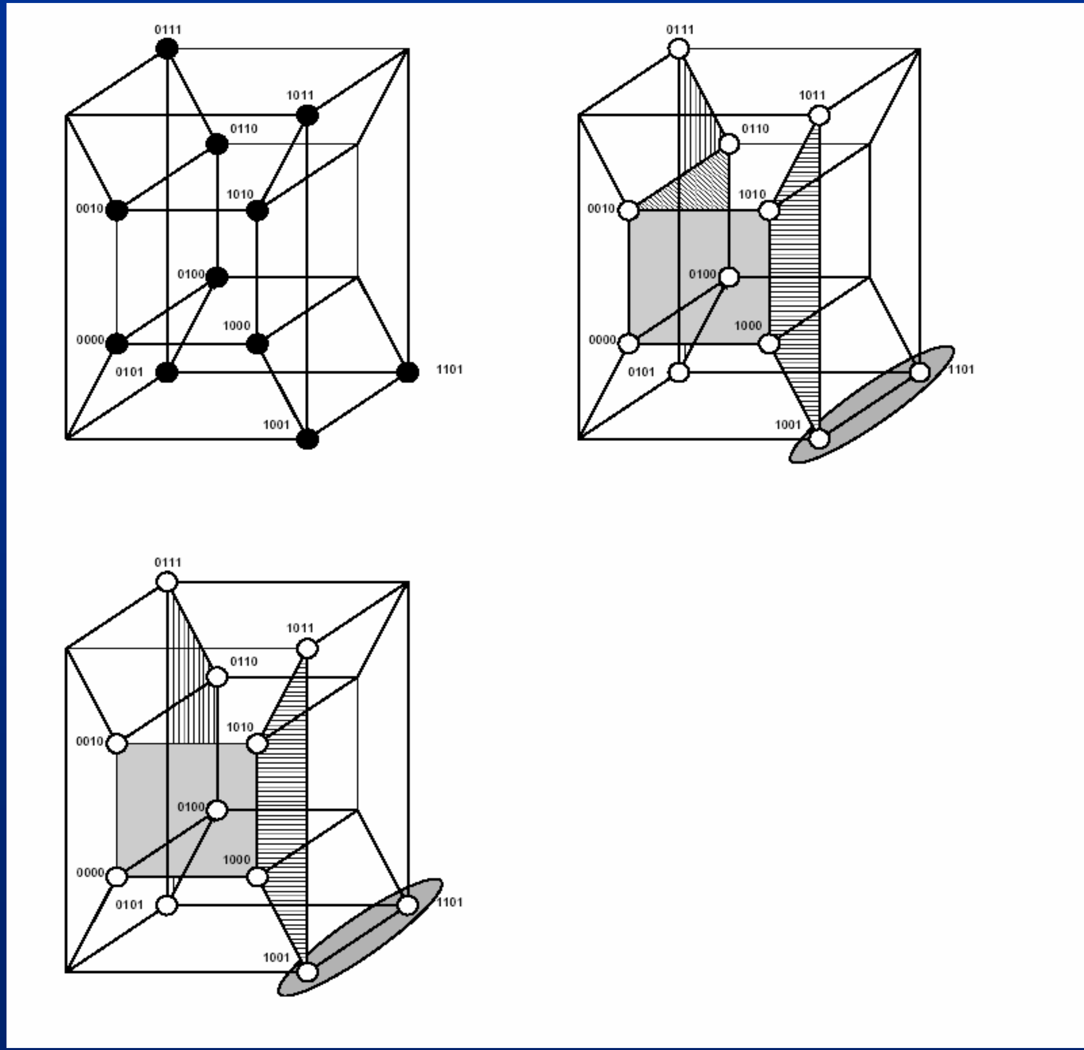
Example: Second Expansion

- Cover={β, γ, δ~, ε~}
- Expand δ~=10*1 to δ= 10∗∗.
- Expand ε~=1101 to ε= 1∗01.
- Cover={β, γ, δ, ε}; prime and irredundant



Example: ESPRESSO

- Expansion
  - Cover is: {α, β, γ, δ, ε}.
  - Prime, redundant, minimal w.r.t. scc.
- Irredundant
  - Cover is: {β, γ, δ, ε}
  - Prime, irredundant



Expand: Naive Implementation

- **For each implicant**
  - For each care literal
    - Raise it to don't care if possible.
  - Remove all covered implicants.
- **Problems**
  - Validity check.
  - Order of expansions.
- **Validity Check**
  - Espresso, MINI
    - Check intersection of expanded implicant with OFF-set.
    - Requires complementation of {ON-set ∪ DC-Set}
  - Presto
    - Check inclusion of expanded implicant in the union of the ON-set and DC-set.
    - Can be reduced to recursive tautology check.

Expand Heuristics …

- **Expand first cubes that are unlikely to be covered by other cubes.**
- **Selection**
  - Compute vector of column sums.
  - Implicant weight: inner product of cube and vector.
  - Sort implicants in ascending order of weight.
- **Rationale**
  - Low weight correlates to having few 1's in densely populated columns.

Example …

- f = a'b'c' +ab'c' +a'bc' +a'b'c
- DC-set = abc'
- Ordering
  - Vector: $[313131]^T$
  - Weights: (9, 7, 7, 7).
- Select second implicant.

a'b'c'	10	10	10
ab'c'	01	10	10
a'bc'	10	01	10
a'b'c	10	10	01
	31	*31*	*31*

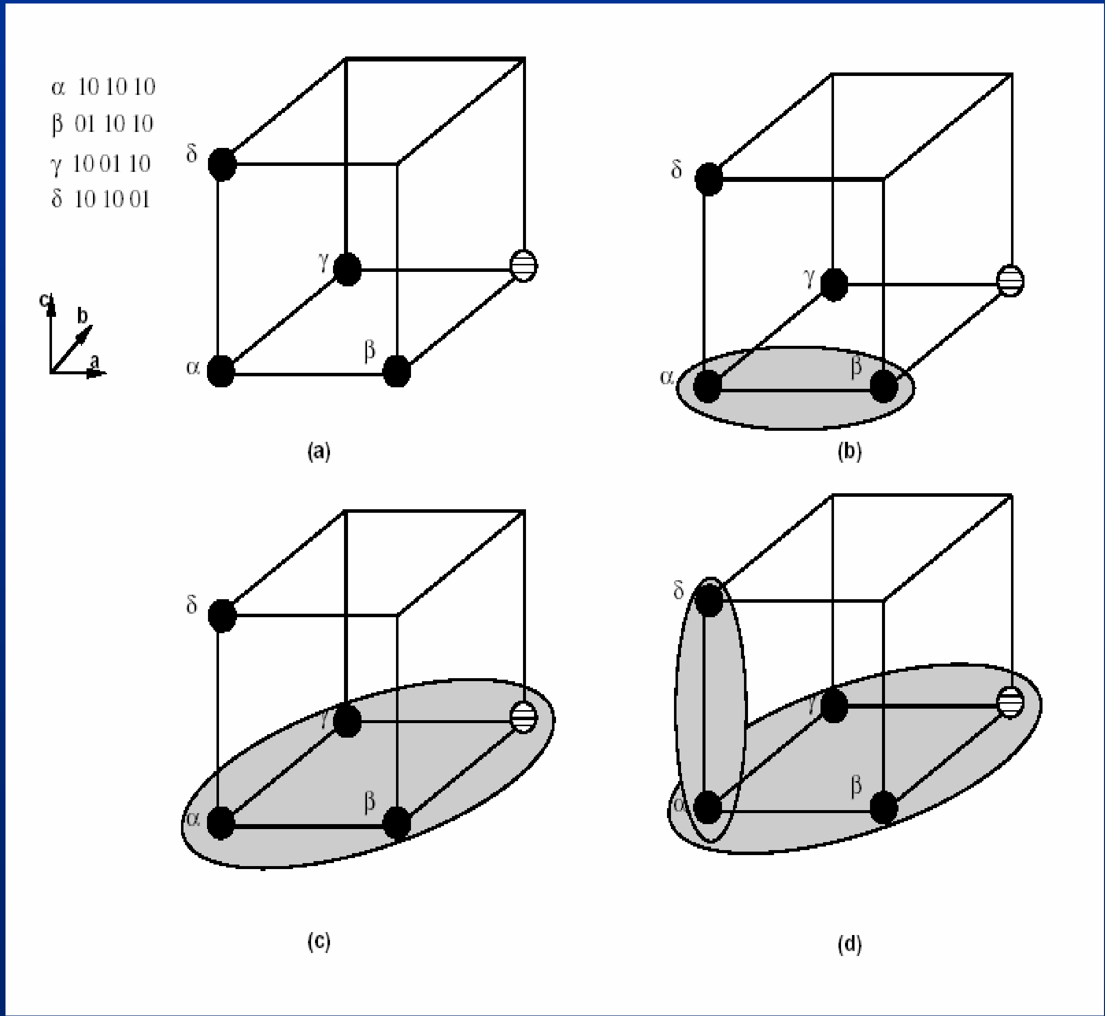
$$\begin{bmatrix} 10 & 10 & 10 \\ 01 & 10 & 10 \\ 10 & 01 & 10 \\ 10 & 10 & 01 \end{bmatrix} * \begin{pmatrix} 3 \\ 1 \\ 3 \\ 1 \\ 3 \\ 1 \end{pmatrix} = [\,9\ 7\ 7\ 7\,]$$

OFF-set:

01 11 01
11 01 01

… Example

- Expand **01 10 10**
  - 11 10 10 valid.
  - 11 11 10 valid.
  - 11 11 11 invalid.
- Update cover to
  - 11 11 10
  - 10 10 01
- Expand **10 10 01**
  - 11 10 01 invalid.
  - 10 11 01 invalid.
  - 10 10 11 valid.
- Expanded cover
  - 11 11 10
  - 10 10 11



Expand in ESPRESSO …

- **Smarter heuristics for choosing literals to be expanded.**
- **Four-step procedure in Espresso.**
- **Rationale**
  - Raise literals so that expanded implicant
    - Covers a maximal set of cubes.
    - As large as possible.
- **Definitions: For a cube α to be expanded**
  - *Free*: Set of entries that can be raised to 1.
  - Overexpanded cube: Cube whose entries in *free* are simultaneously raised.
  - Feasibly covered cube: A cube $\beta \in F^{ON}$ is feasibly covered iff supercube with α is distance 1 or more from each cube of F^{OFF} (i.e. does not intersect with offset).

… Expand in ESPRESSO …

- **1. Determine the essential parts.**
  - Determine which entries can never be raised, and remove them from *free* .
    - Search for any cube in F^{OFF} that has distance 1 from α (corresponding column cannot be raised)
  - Determine which parts can always be raised, raise them, and remove them from *free* .
    - Search for any column that has only 0's in F^{OFF}
- **2. Detection of feasibly covered cubes.**
  - If there is an implicant $\beta \in F^{ON}$ whose supercube with α is feasible repeat the following steps.
    - Raise the appropriate entry of α and remove it from *free*.
    - Remove from *free* entries that can never be raised or that can always be raised and update α.
  - Each cube remaining in the cover F^{ON} is tested for being feasibly covered.
  - α is expanded by choosing feasibly covered cube that covers the most other feasibly covered cubes.

… Expand in ESPRESSO

- Only cubes $\in F^{ON}$ that are covered by the overexpanded cube of α need to be considered.
- Cubes $\in F^{OFF}$ that are 1 distance or more from the overexpanded cube of α do not need to be checked.

- **3. Expansion guided by the overexpanded cube.**
  - When there are no more feasibly covered cubes while the overexpanded cube of α covers some other cubes of F^{ON}, repeat the following steps.
    - **Raise a single entry of α as to overlap a maximum number of those cubes**.
    - Remove from *free* entries that can never be raised or that can always be raised and update α.
  - This has the goal of forcing α to overlap with as many cubes as possible in F^{ON} .

- **4. Find the largest prime implicant covering α**
  - When there are no cubes $\in F^{ON}$ covered by the over-expanded cube of α
    - Formulate a covering problem and solve it by a heuristic method.
    - Find the largest prime implicant covering α.

Example

- **β = 01 10 10 is selected first for expansion**
  - *Free* set includes columns {1,4,6}
  - Column 6 cannot be raised
    - Distance 1 from off-set 01 11 01
  - Supercube of β and α is valid
    - β = 11 10 10
  - Supercube of β and γ is valid
    - β = 11 11 10
  - Supercube of β and δ is invalid
  - Select γ since the expanded cube by γ covers that one by α
    - Delete implicants α and γ; β' = 11 11 10
- **Next, expand δ = 10 10 01**
  - Free set is {2, 4, 5}
  - Columns 2 and 4 cannot be raised
  - Column 5 of F^{OFF} has only 0's. The 0 in column 5 can be raised
    - δ' = 10 10 11
- **Final cover is {β', δ' }**

α	10	10	10
β	01	10	10
γ	10	01	10
δ	10	10	01

OFF-set:

01 11 01

11 01 01

Another Expand Example …

- **F^{ON}= a'b'cd + a'bc'd + a'bcd + ab'c'd' + ac'd**
- **F^{DC}= a'b'c'd + abcd + ab'cd'**
- **Let assume that we will expand the cube a'b'cd**
  - We can see that variables a and d cannot be raised.
  - Overexpanded cube is a'd.
  - Note that only cubes a'bc'd and a'bcd need to be considered for being feasibly covered.
  - None of the offset cubes need to be checked as they are all distance 1 or more from the overexpanded cube.
  - Supercube of a'b'cd and a'bc'd is a'd.
  - Supercube of a'b'cd and a'bcd is a'cd.
  - So, a'bc'd is selected and the cube is expanded to **a'd**.

… Another Expand Example

- **Next, let us expand cube ab'c'd'.**
  - We can see that variables a and b cannot be raised.
  - Overexpanded cube is ab'.
  - None of the remaining cubes can be feasibly covered.
  - None of the remaining cubes is covered by ab'.
  - Expansion is done to cover the largest prime implicant.
  - So, variable d is raised and the cube is expanded to **ab'c'.**
- **Finally, cube ac'd is expanded.**
  - Variables c and d cannot be raised.
  - Overexpanded cube is c'd.
  - No remaining cubes covered with overexpanded cube.
  - Find the largest prime implicant covering the cube.
  - Largest prime implicant is **c'd**.
- **Final Expanded Cover is: a'd + ab'c' + c'd**

Reduce Heuristics …

- **Goal is to decrease size of each implicant to smallest size so that successive expansion may lead to another cover with smaller cardinality.**
- **Reduced covers are not prime.**
- **Sort implicants**
  - First process those that are large and overlap many other implicants
  - Heuristic: sort by descending weight (weight like expand)
- **For each implicant**
  - Lower as many * as possible to 1 or 0.
- **Reducing an implicant α**
  - Can be computed by intersecting α with complement of $F-\{\alpha\}$.
  - May result in multiple implicants.
  - Must ensure result yields a single implicant.
- **Theorem**
  - Let $\alpha \in F$ and $Q = \{F \cup F^{DC}\} - \{\alpha\}$
  - Then, the maximally reduced cube is: $\alpha\sim = \alpha \cap \text{supercube}(Q'_{\alpha})$

… Reduce Heuristics …

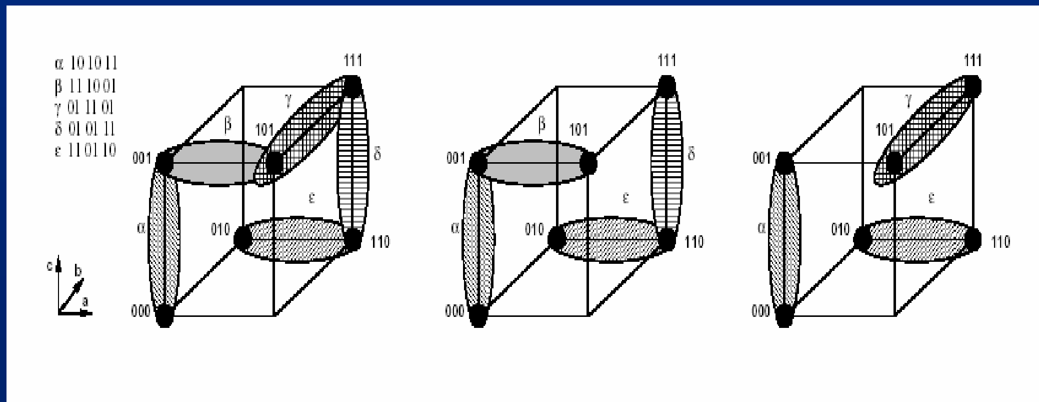
- Expanded cover
  - **11 11 10**
  - **10 10 11**
- Select first implicant 11 11 10 = c’
  - **Complement of 10 10 11 (a’b’) is {01 11 11; 11 01 11} (a+b)**
  - **C’ intersected with 1 is c’.**
  - **Cannot be reduced.**
- Select second implicant 10 10 11 (a’b’)
  - **Complement of c’ is c.**
  - **a’b’ intersected with c is a’b’c.**
  - **Reduced to 10 10 01 (a’b’c).**
- Reduced cover
  - **11 11 10**
  - **10 10 01**

… Reduce Heuristics

- **Another Reduce Example**
  - $F = a'b' + c'$
  - $F^{DC} = bc'$
- **Consider reducing c'**
  - $Q = \{a'b', bc'\}$
  - $Q_{c'}=\{a'b', b\}$
  - $Q'_{c'}=\{ab'\}$, $SC(Q'_{c'})=\{ab'\}$
  - Thus, $c' \cap SC(Q'_{c'})=ab'c'$
- **Note that if F^{DC} is not included in Q, we will not get the correct result**
  - $Q = \{a'b'\}$
  - $Q_{c'}=\{a'b'\}$
  - $Q'_{c'}=\{a+b\}$, $SC(Q'_{c'})=\{1\}$
  - Thus, $c' \cap SC(Q'_{c'})=c'$

Irredundant Cover …

- **Relatively essential set E^r**
  - Implicants covering some minterms of the function not covered by other implicants.
  - $\alpha \in F$ is in E^r if it is not covered by $\{F \cup F^{DC}\} - \{\alpha\}$
- **Totally redundant set R^t**
  - Implicants covered by the relatively essentials.
  - $\alpha \in F$ is in R^t if it is covered by $\{E^r \cup F^{DC}\}$
- **Partially redundant set R^p**
  - Remaining implicants.
  - $R^p = F - \{E^r \cup R^t\}$



… Irredundant Cover …

- **Find a subset of R^p that, together with E^r, covers the function.**
- **Modification of the tautology algorithm**
  - Each cube in R^p is covered by other cubes in E^r and R^p.
  - Determine set of cubes when removed makes function non-tautology.
  - Find mutual covering relations.
- **Reduces to a covering problem**
  - Heuristic algorithm.

… Irredundant Cover

- $E^r = \{\alpha, \varepsilon\}$
- $R^t = \{\}$
- $R^p = \{\beta, \gamma, \delta\}$
- Covering relations
  - **β is covered by {α, γ}.**
    - $(\alpha + \gamma + \delta + \varepsilon)_\beta$
    - $(a'b'+ac+ab+bc')_{b'c}$
    - $(a' + a + 0 + 0)_{b'c}$
  - **γ is covered by {β, δ}.**
  - **δ is covered by {γ, ε}**
- Minimum cover: $\gamma \cup E^r = \{\alpha, \varepsilon, \gamma\}$

α	10	10	11	*a'b'*
β	11	10	01	*b'c*
γ	01	11	01	*ac*
δ	01	01	11	*ab*
ϵ	11	01	10	*bc'*

	β	γ	δ
β	1	1	0
γ	1	1	1
δ	0	1	1

Essentials …

- **Essential prime implicants are part of any cover.**
- **Theorem**
  - Let $F=G\cup\alpha$, where α is a prime disjoint from G. Then, α is an essential prime iff Consensus(G,α) does not cover α.
- **Corollary**
  - Let F^{ON} be a cover of the on-set and F^{DC} be a cover of the dc-set and α is a prime implicant. Then, α is an essential prime implicant iff $H\cup F^{DC}$ does not cover α, where $H=Consensus(((F^{ON}\cup F^{DC})\#\alpha), \alpha)$
- **Example**

α	*10*	*10*	*11*	***a'b'***
β	*11*	*10*	*01*	***b'c***
γ	*01*	*11*	*01*	***ac***
δ	*01*	*01*	*11*	***ab***

Test α :

$F\#\alpha$={ab'c, ab, ac}={ab, ac}

H= {b'c}

H_α={c}; not tautology

α not contained in H and essential

… Essentials

- **Another Example**
  - F = a'b' + c'
  - F^{DC} = bc' + ac'
- **Let us consider if c' is essential prime implicant**
  - F#c'=a'b'c
  - H=a'b'
  - H∪ {F^{DC}}={a'b',bc',ac'}
  - {a'b',bc',ac'}$_{c'}$= {a'b',b,a}=Tautology
  - Thus, c' is not essential prime implicant
  - Note that if you do not include F^{DC}, you will get the incorrect result

ESPRESSO Algorithm …

- **Compute the complement.**
- **Find a prime cover: Expand.**
- **Find a prime and irredundant cover: Irredundant.**
- **Extract Essentials.**
- **Iterate**
  - Reduce, Expand, irredundant.
- **Cost functions**
  - Cover cardinality $\varnothing_1$.
  - Weighted sum of cube and literal count $\varnothing_2$.

… ESPRESSO Algorithm

$espresso(F, D)\{$
$R = complement(F \cup D);$
$F = expand(F, R);$
$F = irredundant(F, D);$
$E = essentials(F, D);$
$F = F - E;$
$D = D \cup E;$
repeat $\{$
$\phi_2 = cost(F);$
repeat $\{$
$\phi_1 = |F|;$
$F = reduce(F, D);$
$F = expand(F, R);$
$F = irredundant(F, D);$
$\}$ **until** $(|F| \geq \phi_1);$
$F = last\_gasp(F, D, R);$
$\}$ **until** $(cost(F) \geq \phi_2);$
$F = F \cup E;$
$D = D - E;$
$F = make\_sparse(F, D, R);$
$\}$

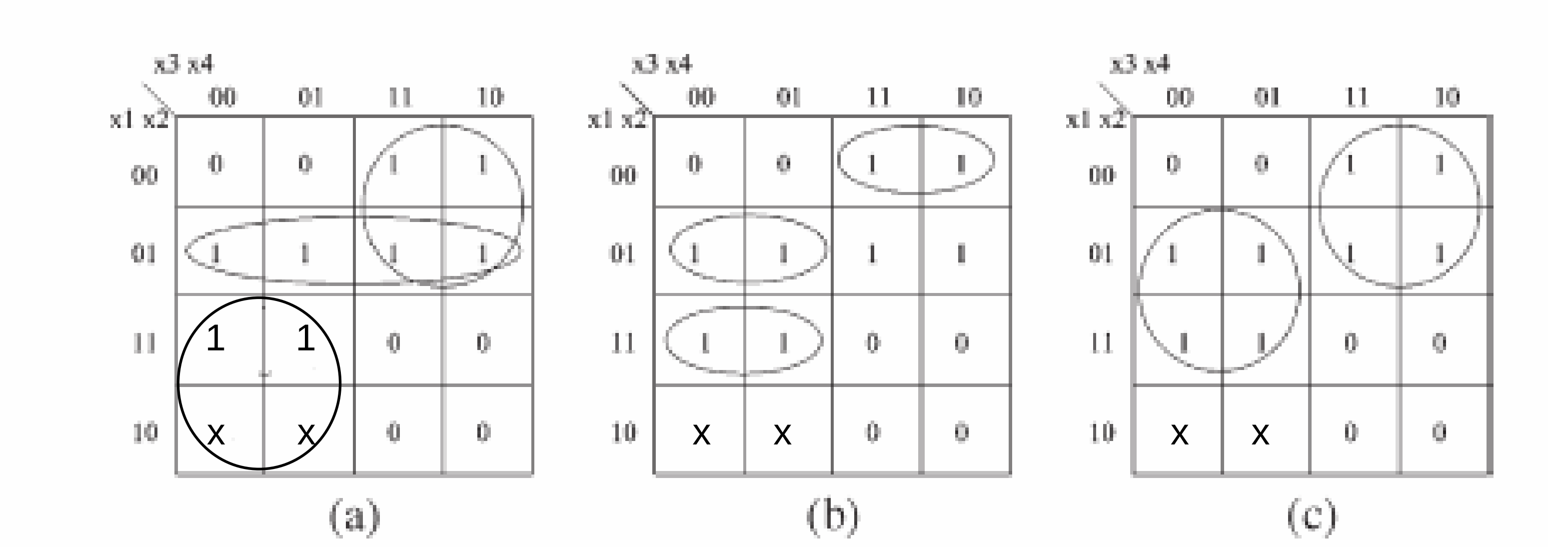
last_gasp: uses different heuristics for reduce and expand to get out of local minimum.

- Reduce each cube independently to cover only minterms not covered by other implicants
- The generated cover after reduce may not cover the function
- Expand only those cubes that were reduced to cover reduced cubes
- Call irredundant on the primes in the original cover and the newly generated primes

make_sparse: attempts to reduce the number of literals in the cover. Done by:

- reducing the "sparse" variables (using a modified version of irredundant rather than reduce),
- followed by expanding the "dense“ variables (using modified version of expand).

Last_gasp Example



- Original cover = $\{x_1x_3', x_1'x_2, x_1'x_3\}$
- Reduced cover=$\{x_1x_2x_3', x_1'x_2x_3', x_1' x_2'x_3\}$
- Cover after expansion= =$\{x_2x_3', x_1'x_3\}$
- Make irredundant of $\{x_1x_3', x_1'x_2, x_1'x_3, x_2x_3', x_1'x_3\}$ =$\{x_2x_3', x_1'x_3\}$

Espresso Format

Example Input	*Espresso Output*
.i 3 *.o 2* *.ilb a b c* *.ob f1 f2* *.p 6* *00- 10* *-01 11* *1-1 10* *11- 10* *110 11* *100 0-* *.e*	*.i 3* *.o 2* *.ilb a b c* *.ob f1 f2* *.p 4* *1-0 01* *11- 10* *00- 10* *-01 11* *.e*

Testability Properties of Two-Level Logic Circuits

- **Single stuck-at fault model**
  - Assumes a single line in the circuit faulty.
  - Faulty line is either stuck-at-0 or stuck-at-1.
- **Theorem**
  - A two-level circuit is fully single stuck-at fault testable iff it is **PRIME** and **IRREDUNDANT**.
- **An untreatable stuck-at fault corresponds to redundancy in the circuit**
  - Redundant stuck-at-0 in any of the products indicates product term is redundant
  - Redundant stuck-at-1 in any of the products inputs indicates product term is not prime
  - Redundancy can be removed by injecting the redundant faulty value in the circuit and propagating constants