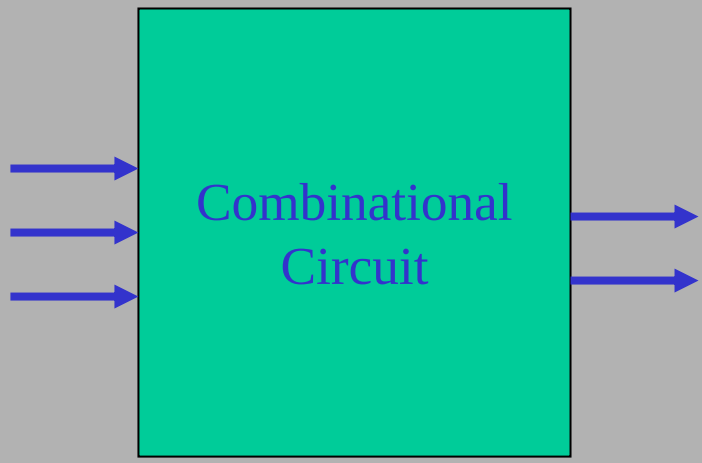
On the Existence of Hazard-Free Multi-Level Logic

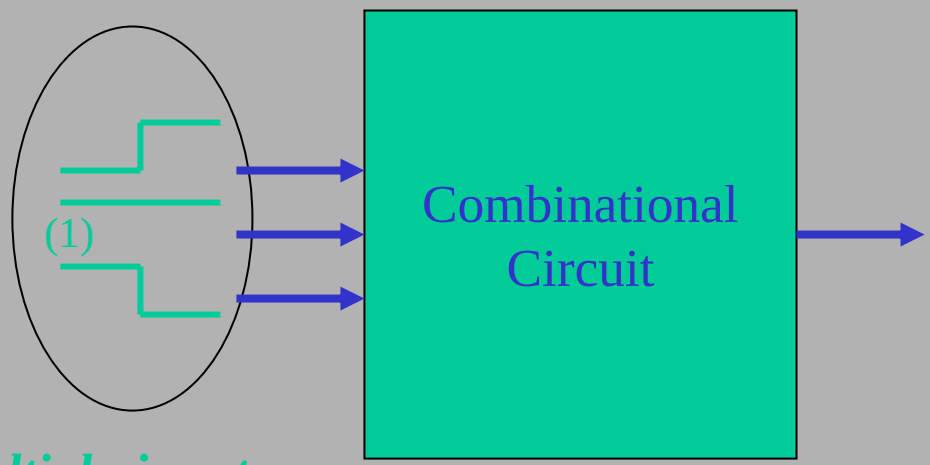
Steven M. Nowick
Charles W. O'Donnell

Department of Computer Science
Columbia University

Preliminaries



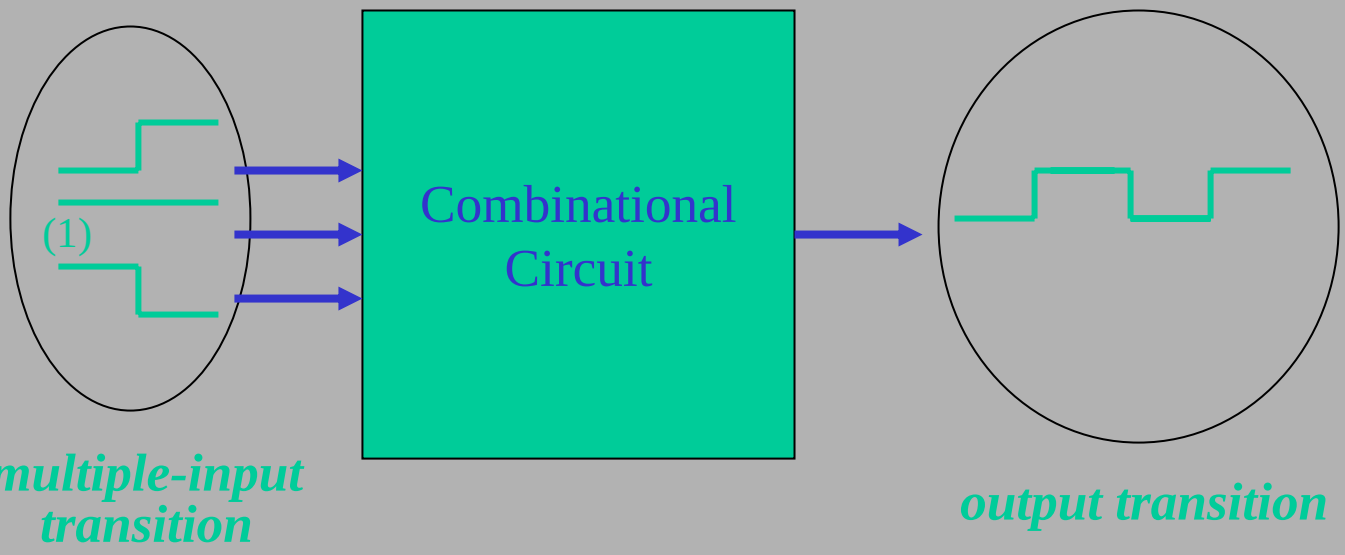
Preliminaries



multiple-input transition

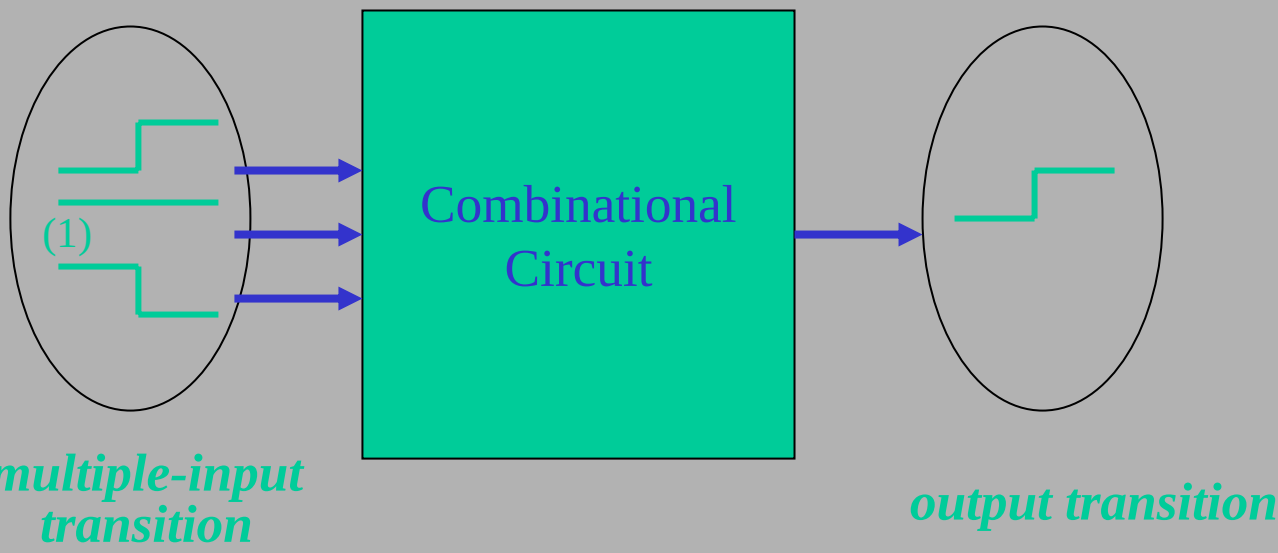
Preliminaries

hazard = ***potential for a glitch***



Preliminaries

hazard-free = *no potential for a glitch*

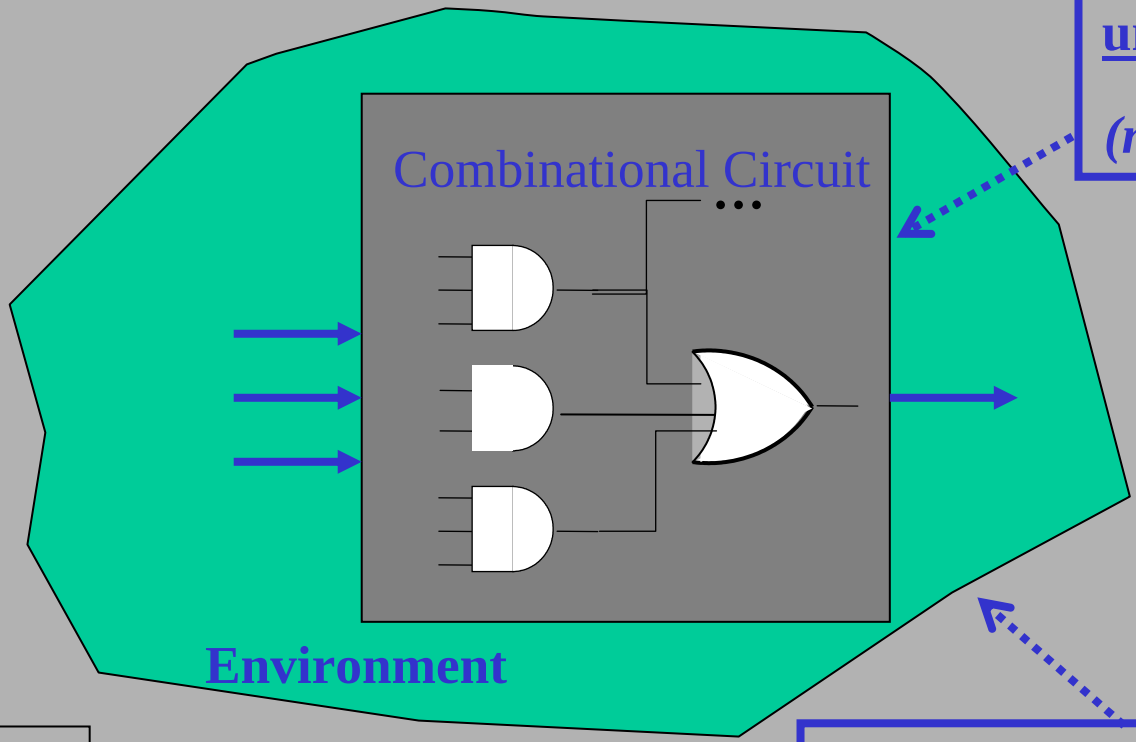


<u>Goal</u>: design hazard-free combinational circuits

Outline

- **Background:** Hazard-Free 2-Level Logic
- **Overview of Multi-Level Method**
  - Hazard-Free Decomposition
  - Algorithm + Examples
  - Standard 3-Level Circuit Structures
  - Canonicity
- **Related Work:** Bredeson [’74]
- **Experimental Results**
- **Conclusions**

Circuit & Environmental Models: #1



CIRCUIT MODEL:
unbounded gate & wire delays
(no timing assptns.)

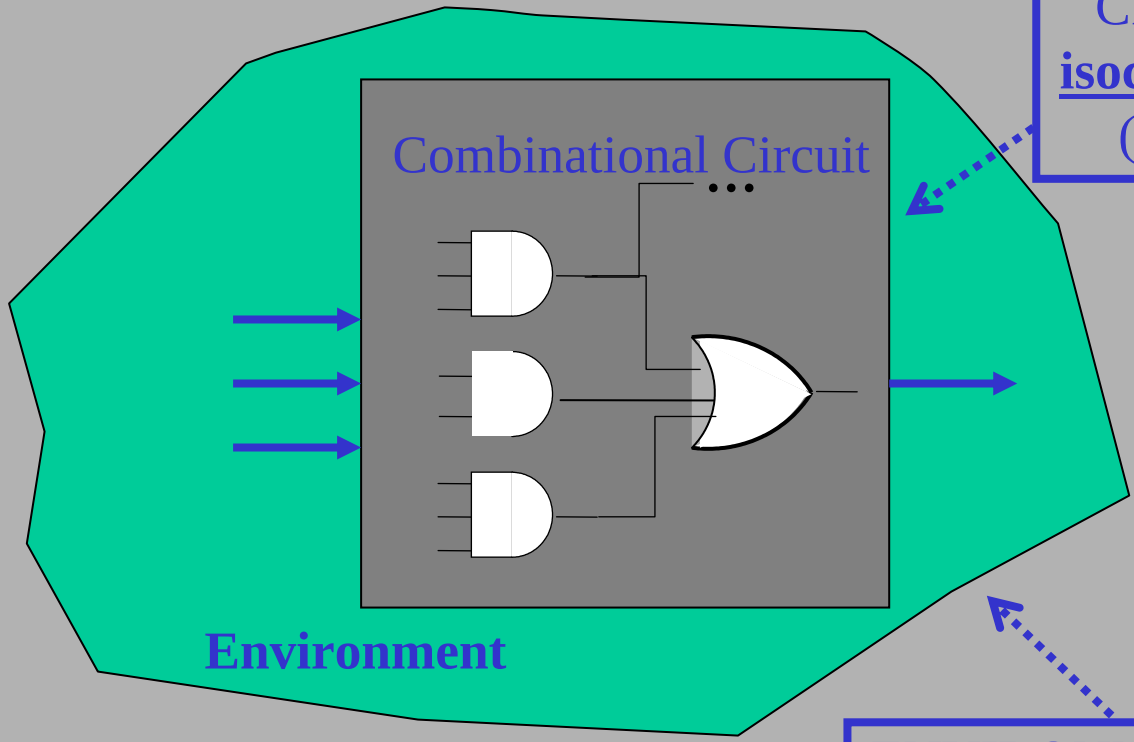
ENVIRONMENTAL MODEL:
"generalized fundamental mode"
(env. must wait until circuit stable)

Our Focus:

… used for "burst-mode" controllers, etc.

Goal: synthesize *hazard-free circuit* under this model

Circuit & Environmental Models: #2



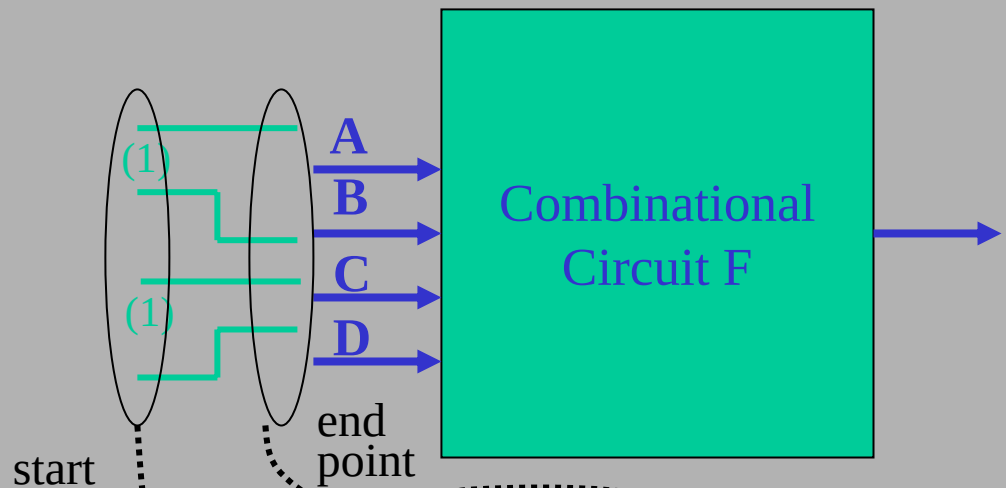
CIRCUIT MODEL:
isochronic fork assptn.
(*i.e. timing assptn.*)

ENVIRONMENTAL MODEL:
"input/output mode"
(no timing assptns.)

Alternative Model:

… used for "QDI" controllers, etc.

Combinational Hazards in 2-Level Circuits: Notation



F:

CD \ AB	00	01	11	10
00	1	1	1	1
01	0	1	1	0
11	0	1	1	0
10	1	1	0	0

t1

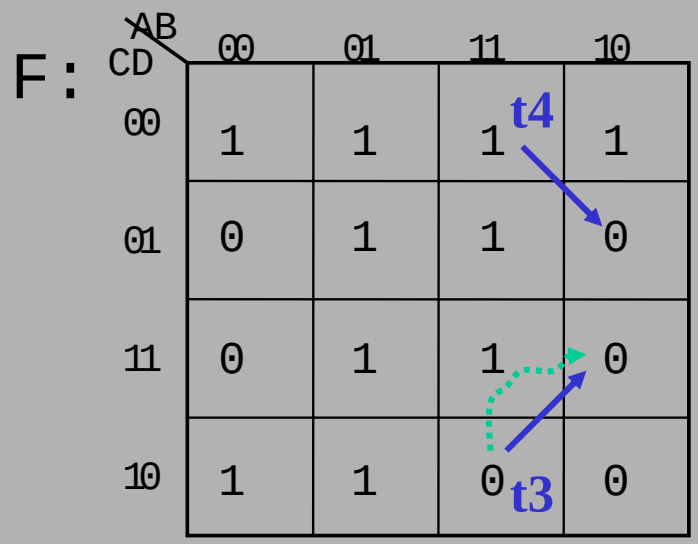
t2

input transition t1:
start point: 0100
end point: 1101

Function F: represented using Karnaugh Map (= truth table)

Function Hazards

- An input transition which is *not* monotonic *in function F itself* [Unger69]
- **Intuition:** occurs when user *specifies glitchy behavior*
  - circuit implementations cannot be guaranteed glitch-free!



t4: function hazard-free

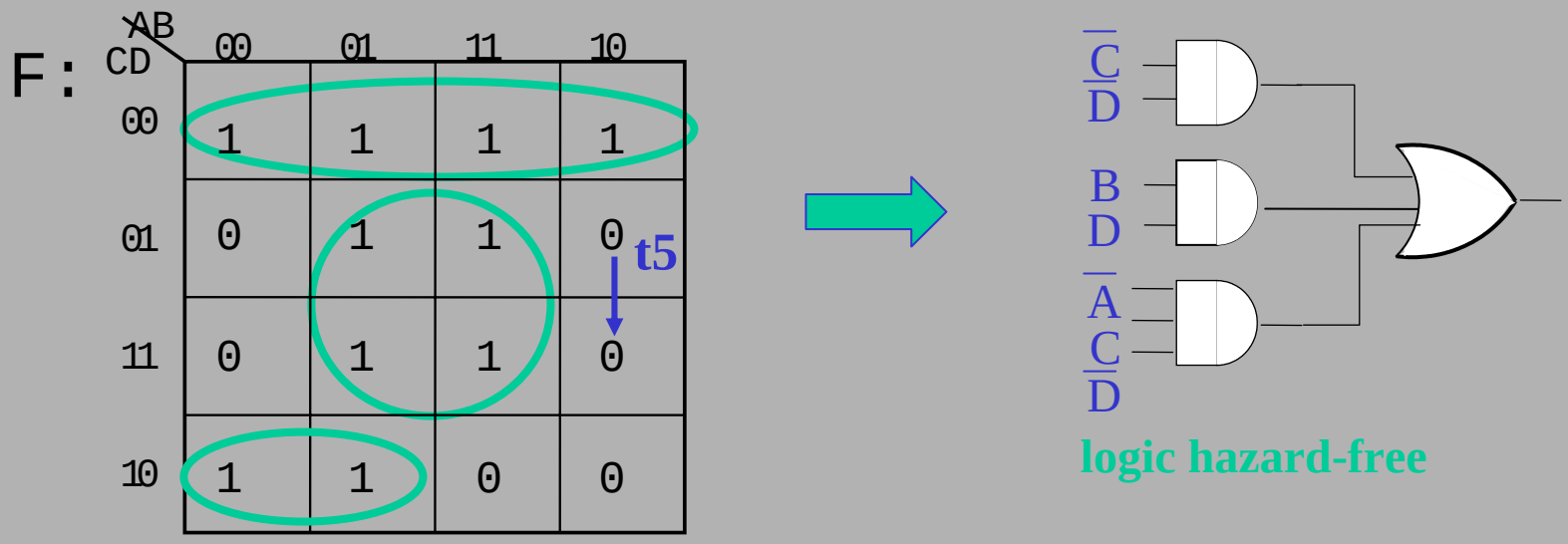
t3: function hazard

… only consider function hazard-free input transitions [Unger 69]

Logic Hazards: Conditions for Hazard-Free Transitions

- Property of a **given circuit implementation**
- An input transition has a **logic hazard** if *circuit implтn. may glitch*

Case 1: 0→0 Transition

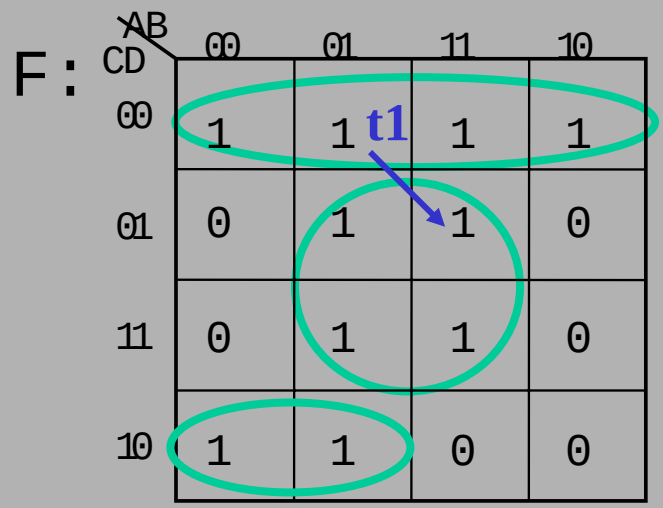


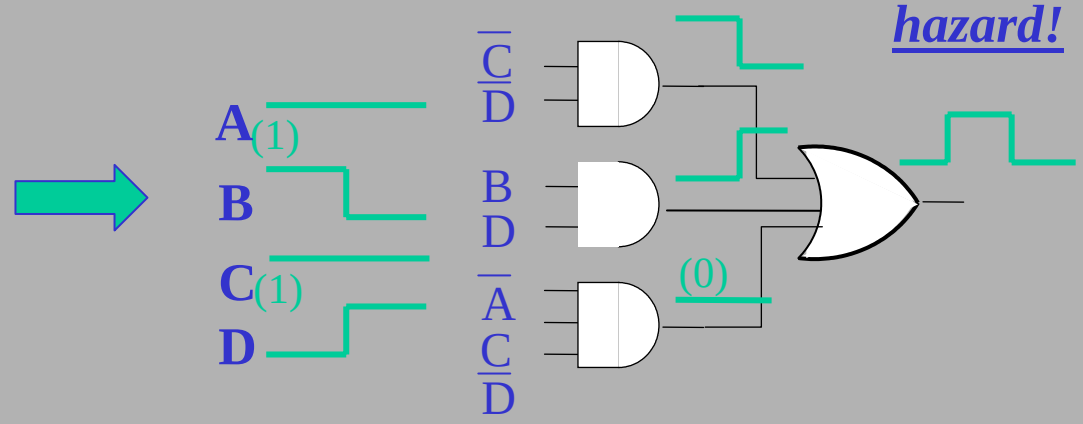
…every AND-OR circuit is logic hazard-free for 0→0 transitions [Unger 69]

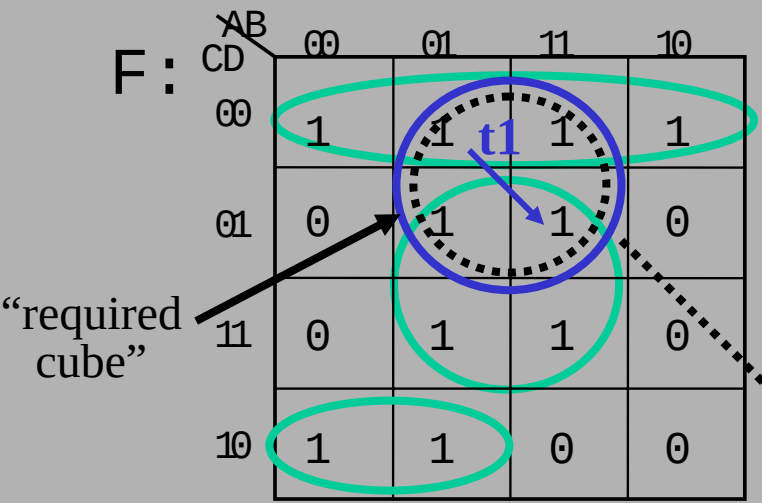
Logic Hazards (cont.)

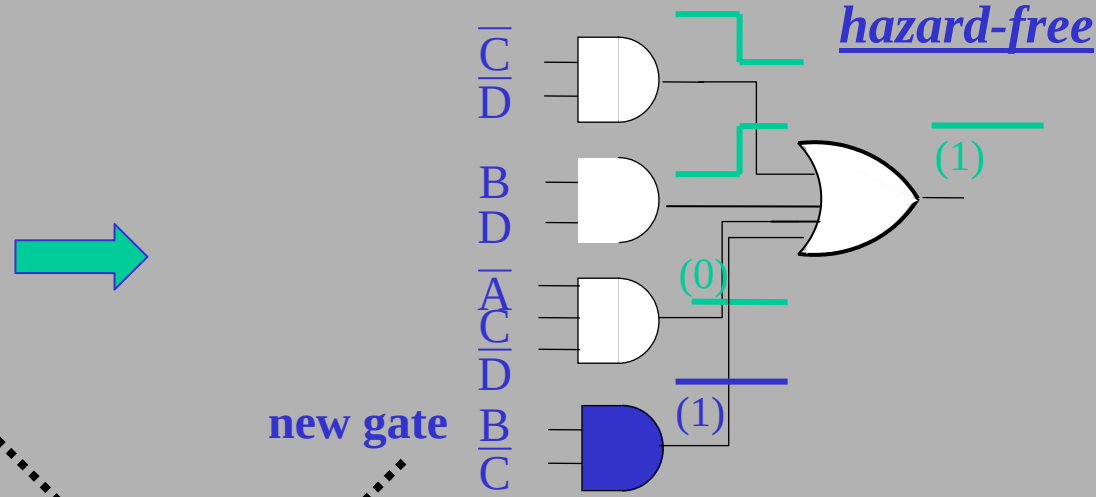
Case 2: 1→1 Transition

Example: input transition t1







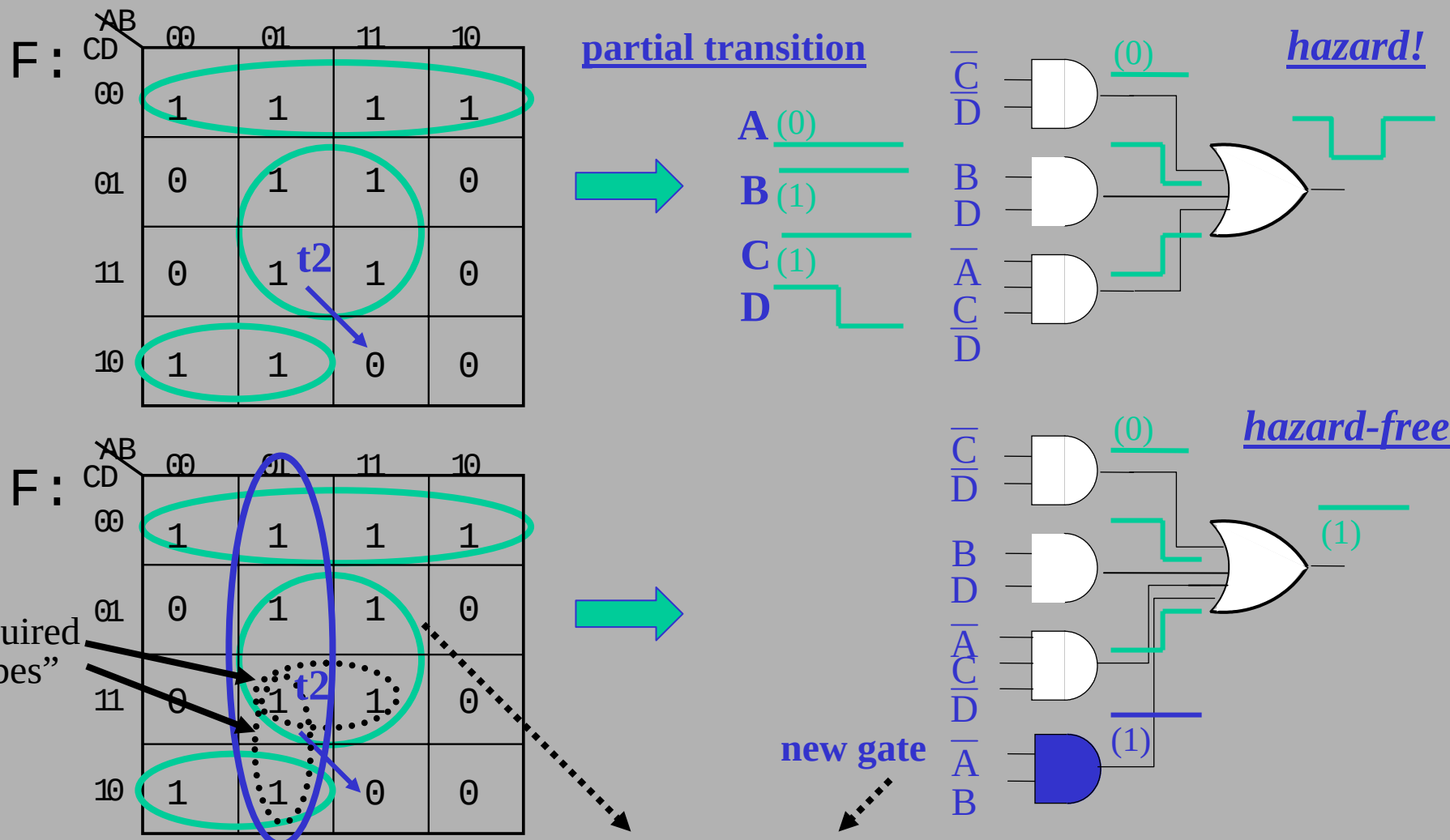


Transition t1's "required cube" must be covered

Logic Hazards (cont.)

Case 3: 1→0 (0→1) Transition

Example: input transition t2

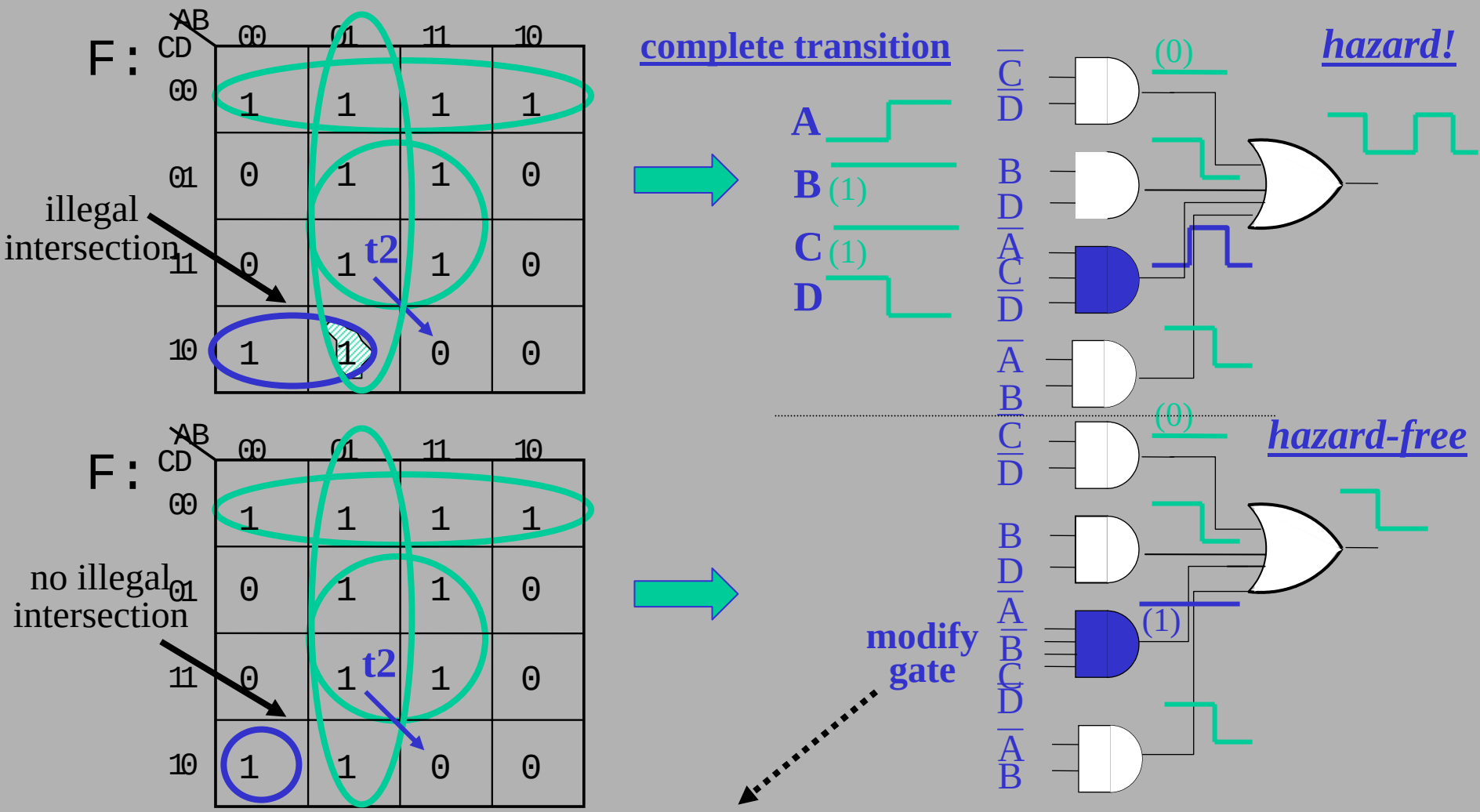


Transition t2: each max 1→1 subtransition = "required cube" which must be covered

Logic Hazards (cont.)

Case 3: 1→0 (0→1) Transition (cont.)

Example: input transition t2



Transition t2: avoid "illegal intersections" of 1→0 (0→1) transitions

Hazard-Free Two-Level Logic Minimization

Summary: "Hazard-Free Covering Rules"

In Karnaugh map for function F …:

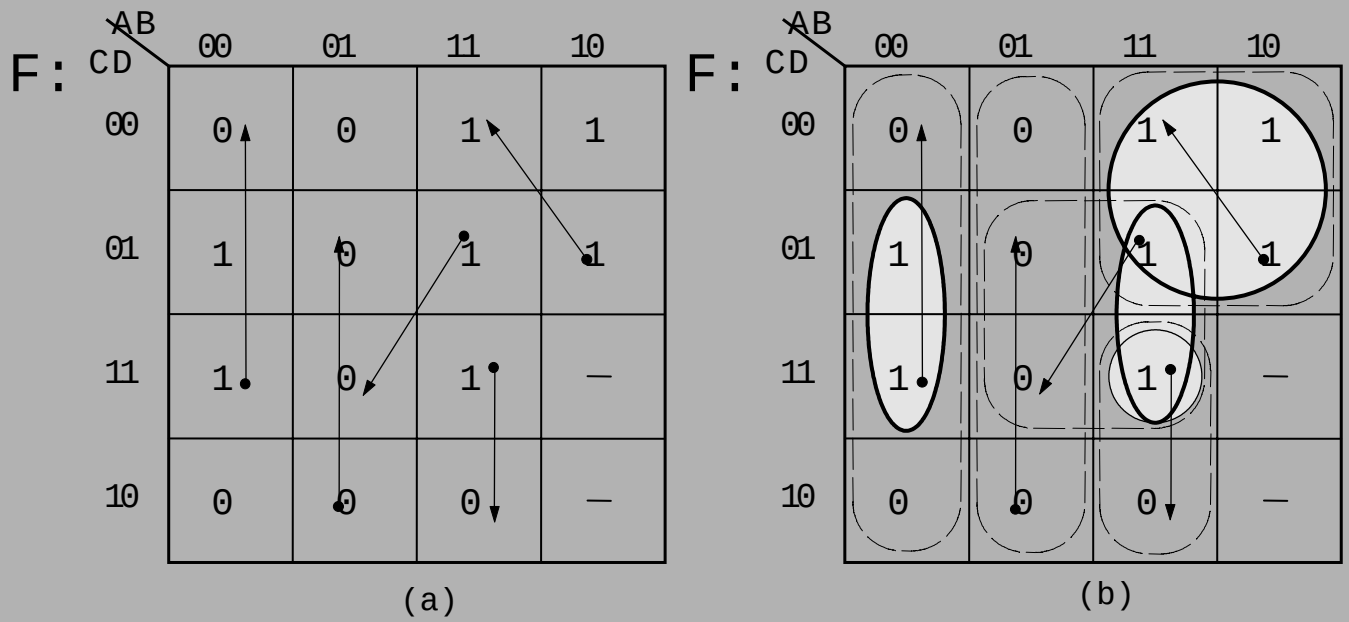
- each **required cube**: must be covered by some product
- no **illegal intersections:** with any 1→0 (0→1) transition

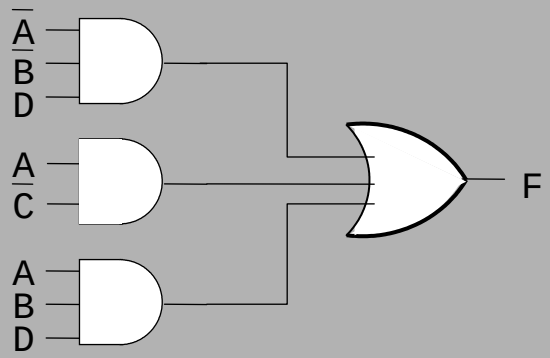
Hazard-Free Minimization Algorithm [Nowick/Dill ICCAD92,TCAD95]

- produce a minimum-cost hazard-free AND-OR circuit

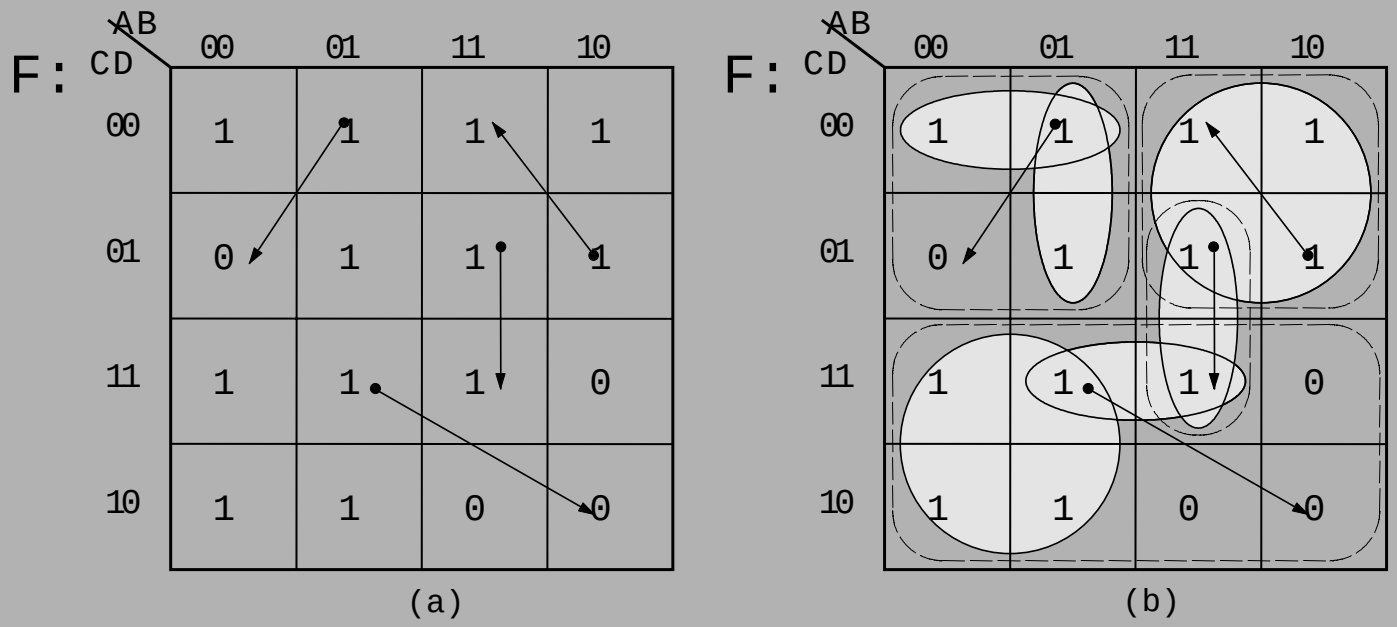
Existence of a Two-Level Solution: *not always!*

Example #1: Hazard-Free Two-Level Solution



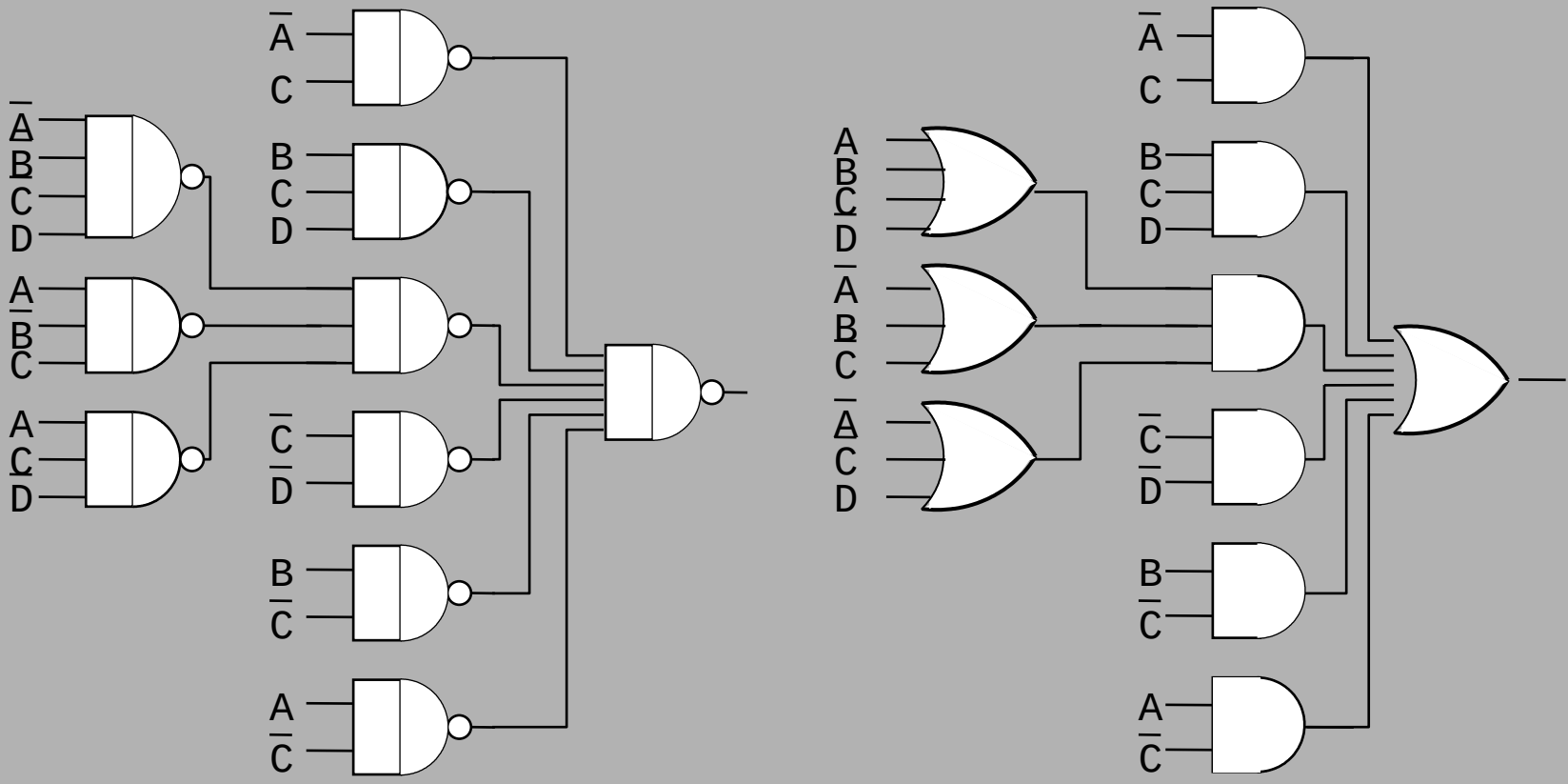


Example #2: No Hazard-Free 2-Level Solution Exists



[Nowick/Dill, ICCAD-92]

Example #2 (cont.)



However, several hazard-free multi-level implementations do exist!

Contribution

Problem. Given a Boolean function **F** and specified set of input transitions **T:**

Does any multi-level hazard-free implementation exist?

Assumptions. Consider only networks of simple combinational gates: AND, OR, INV, NAND, NOR

General Solution: ***2 new algorithms, always find a solution if one exists.***

1. Basic Recursive Method
2. Iterative Method

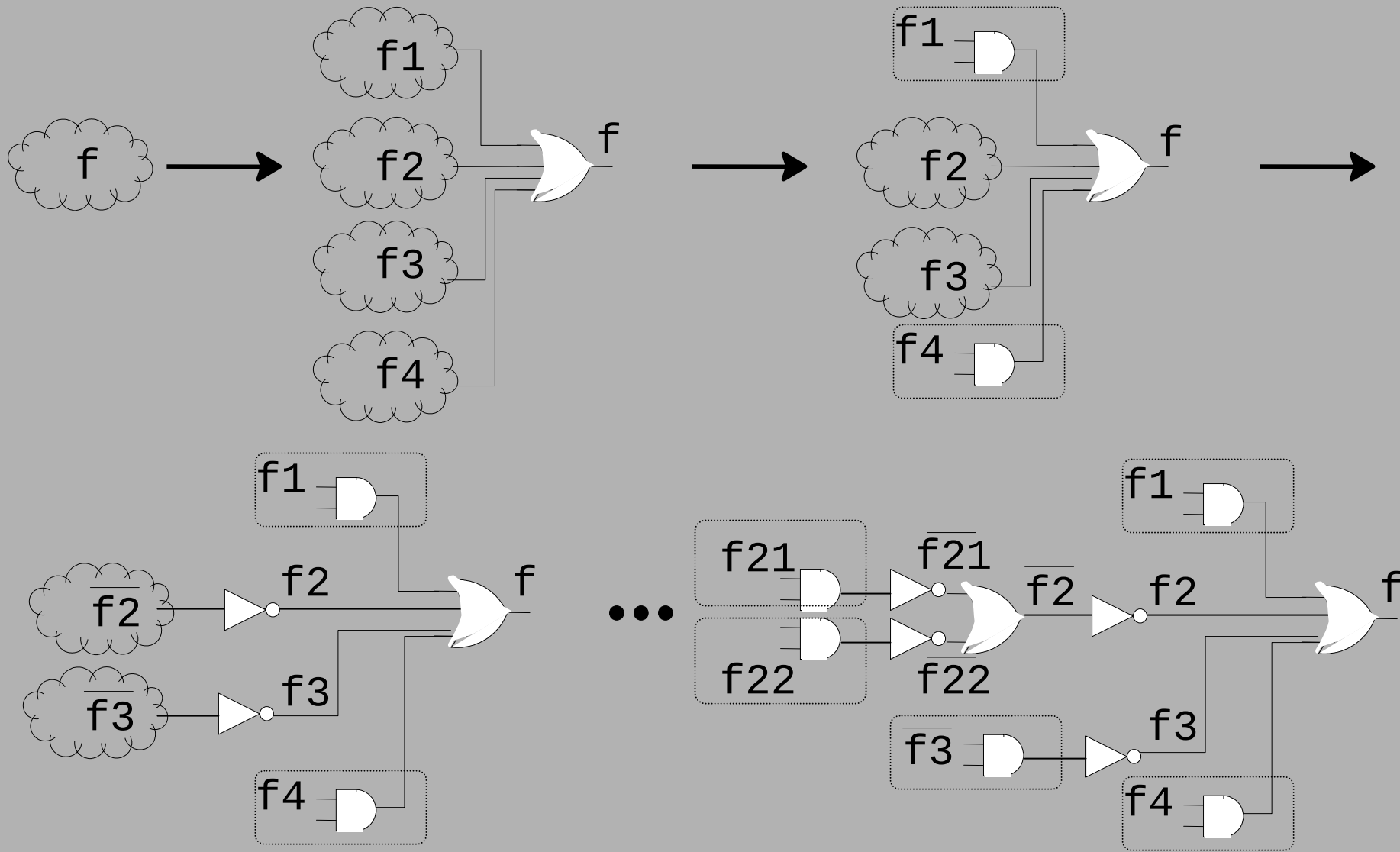
Key Contribution: show that, if some hazard-free multi-level implementation exists, then there is ***always a hazard-free 3-level implementation***:

- **3-Level NAND**
- **OR-AND-OR**

Outline

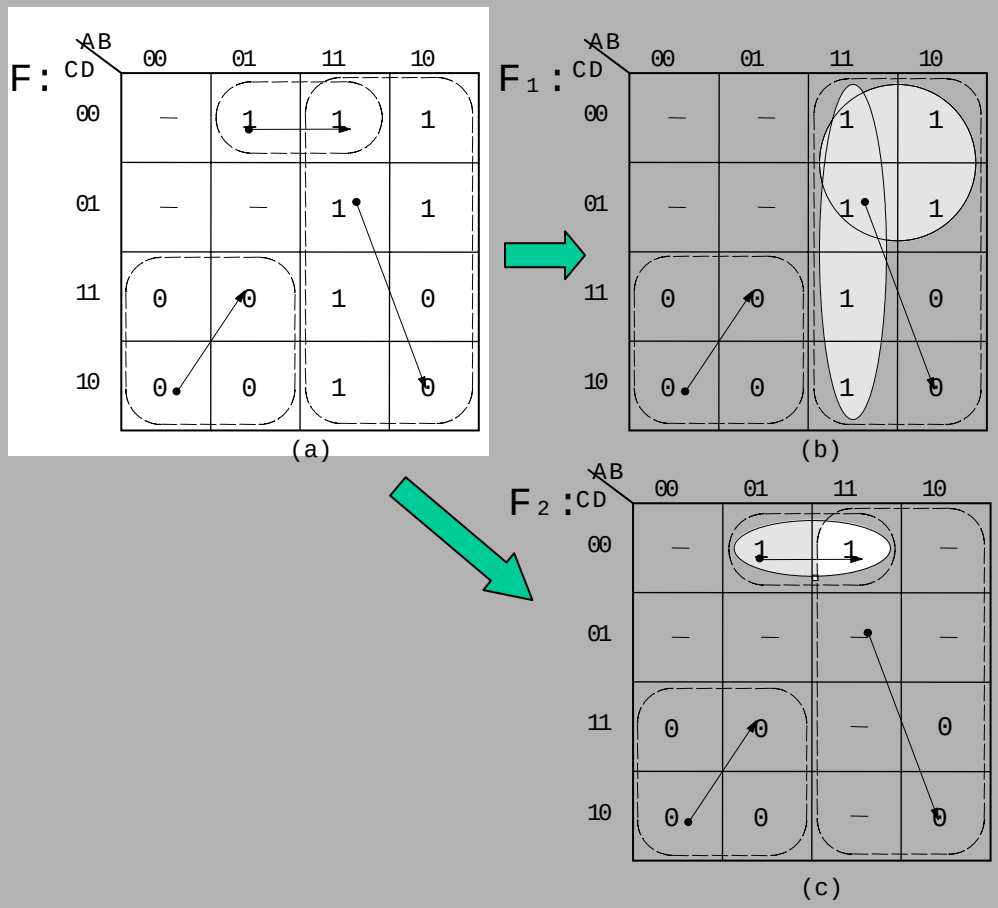
- **Background:** Hazard-Free 2-Level Logic
- **Overview of Multi-Level Method**
  - Hazard-Free Decomposition
  - Algorithm + Examples
  - Standard 3-Level Circuit Structures
  - Canonicity
- **Related Work:** Bredeson [’74]
- **Experimental Results**
- **Conclusions**

Basic Overview of Approach:
Use Recursive “Hazard-Free Decomposition”



First Hazard-Free Decomposition: **Disjunction**

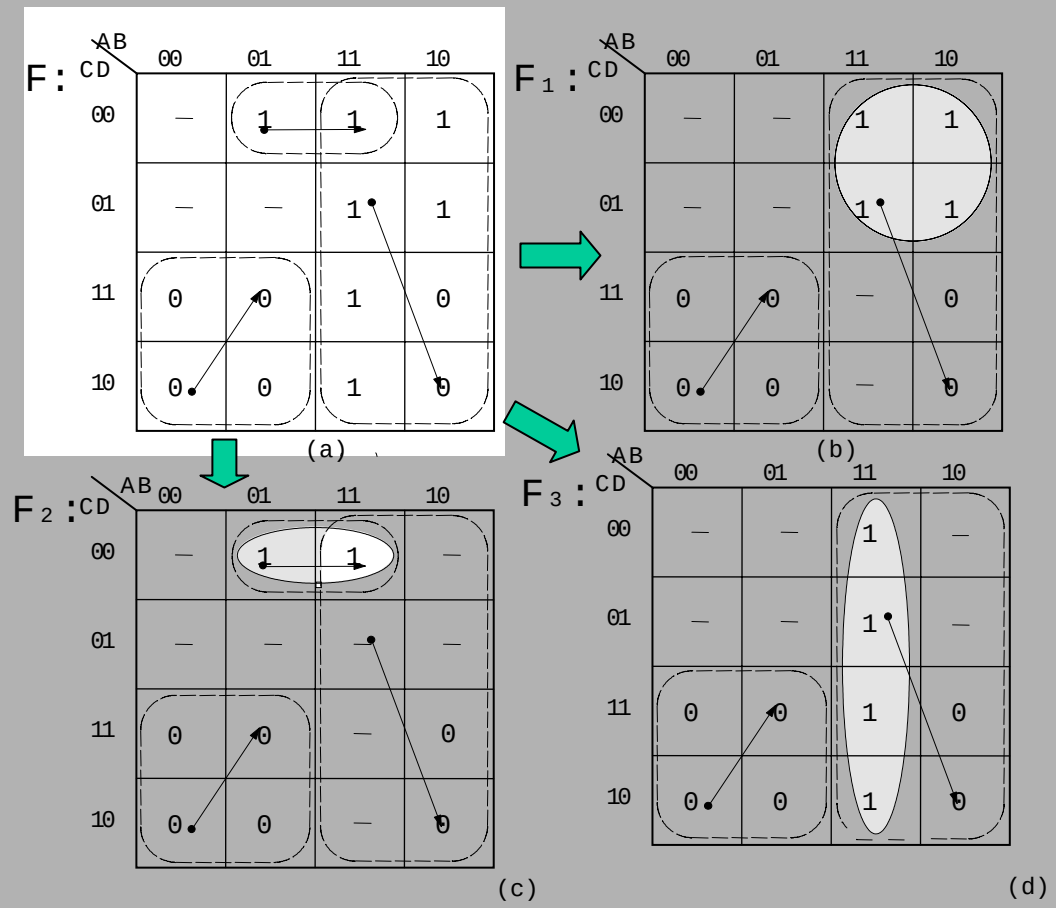
General disjunctive decomposition:



"Required cubes" are divided among the new subfunctions

First Hazard-Free Decomposition: **Disjunction** (cont.)

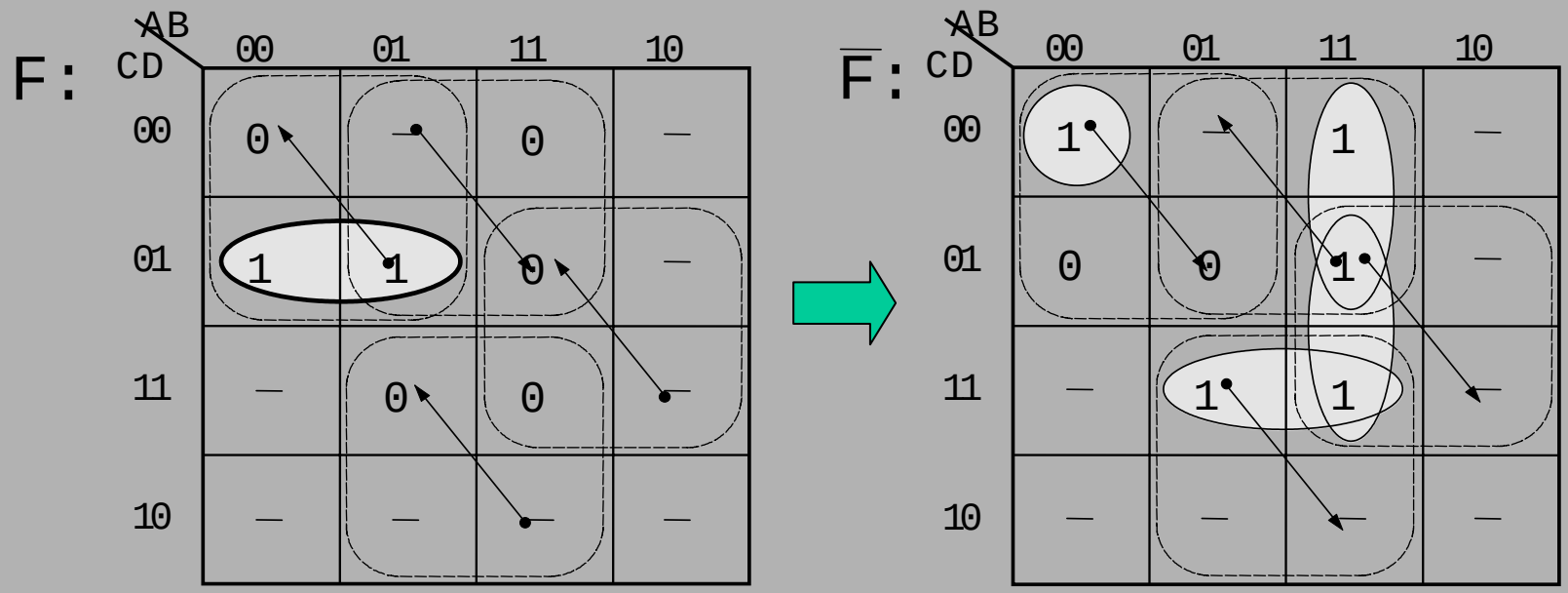
Maximal disjunctive decomposition: a special case



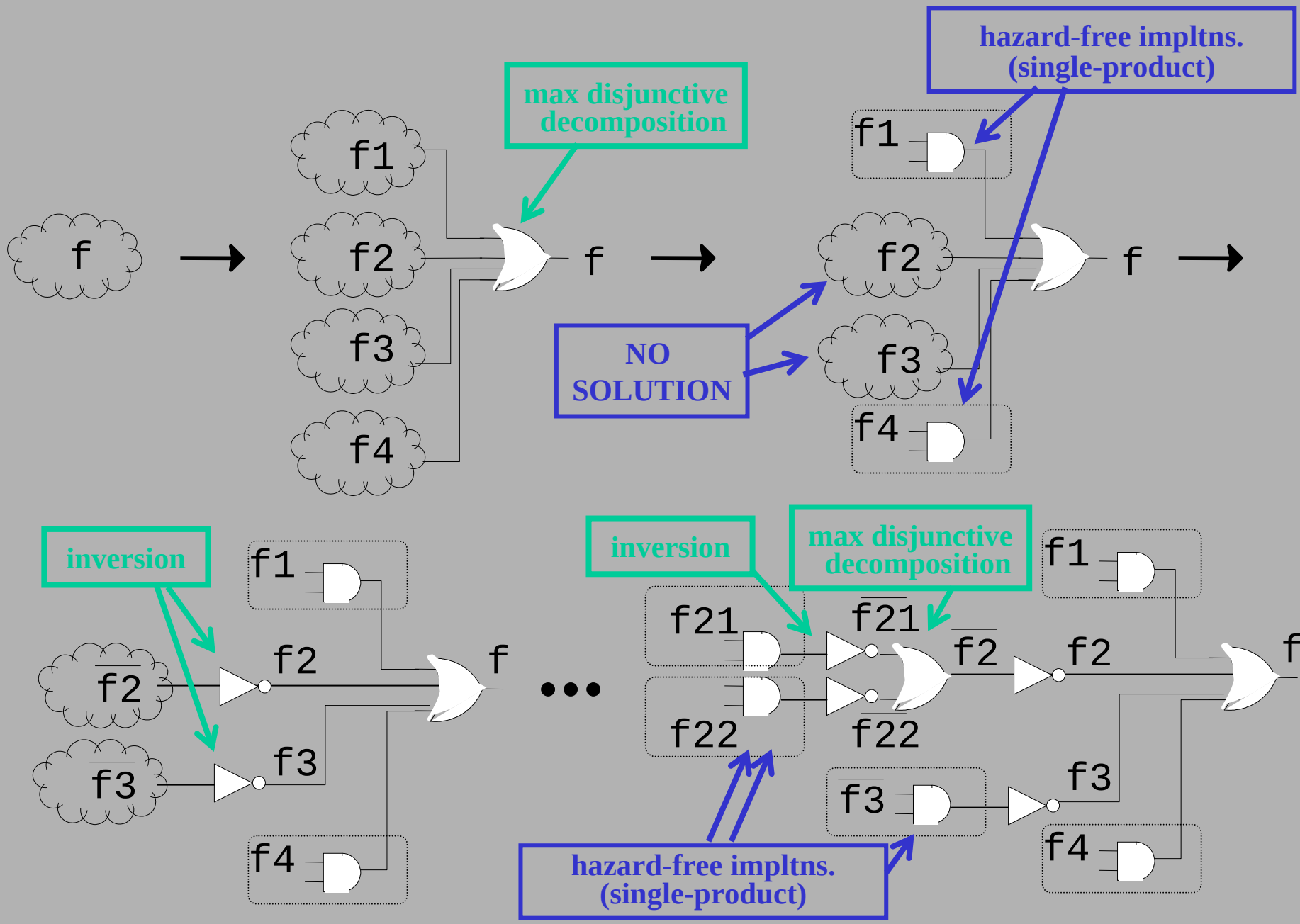
Each new subfunction gets *exactly one* (distinct) "required cube"

Second Hazard-Free Decomposition: **Inversion**

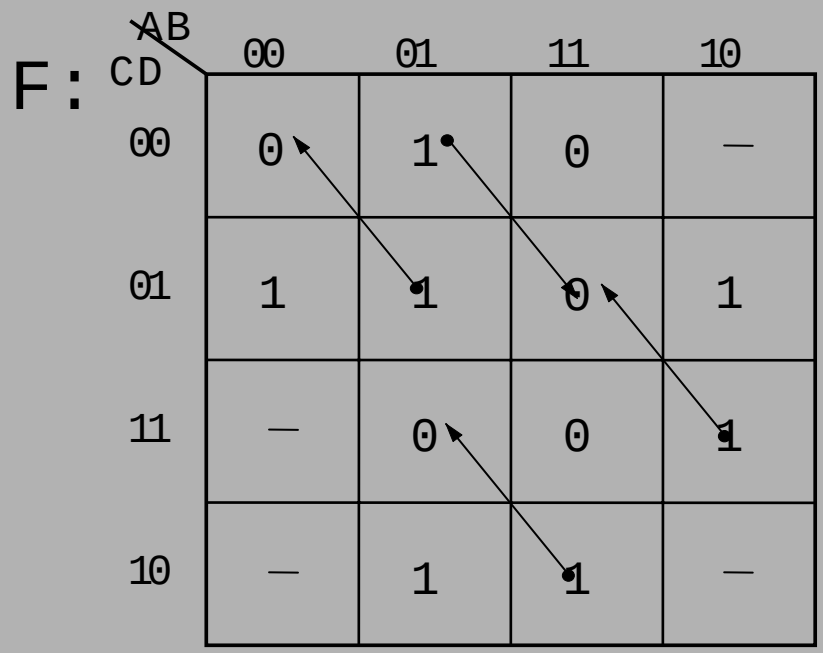
Swap OFF-set and required cubes

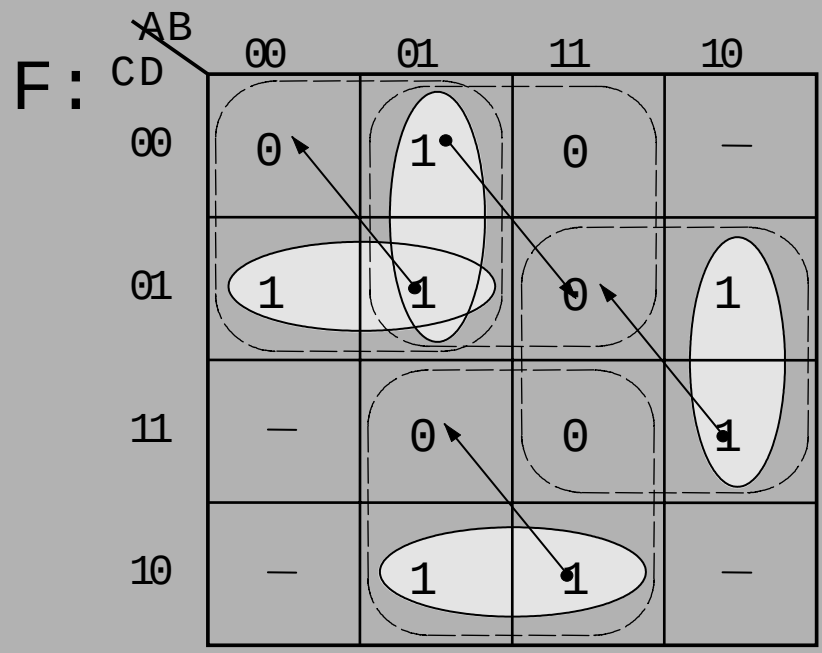


Basic Summary of Approach: *alternate* disjunction + inversion



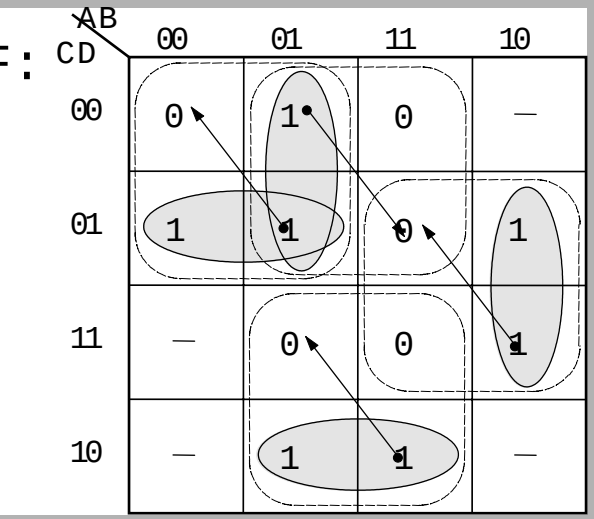
First Complete Example:
Finding a Hazard-Free Multi-Level Implementation

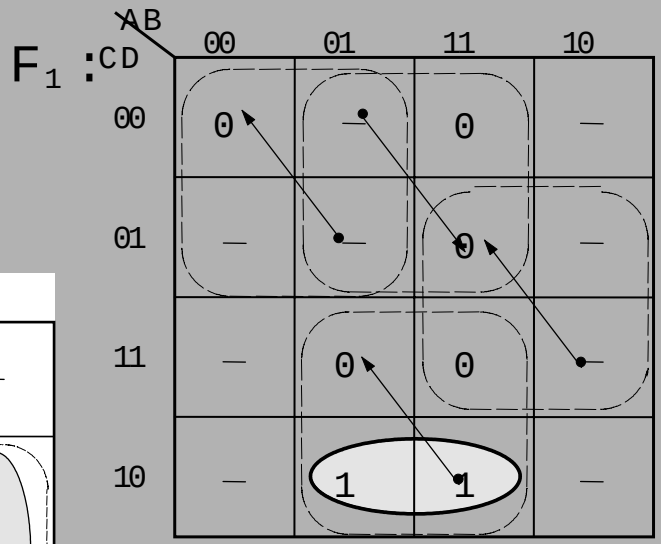


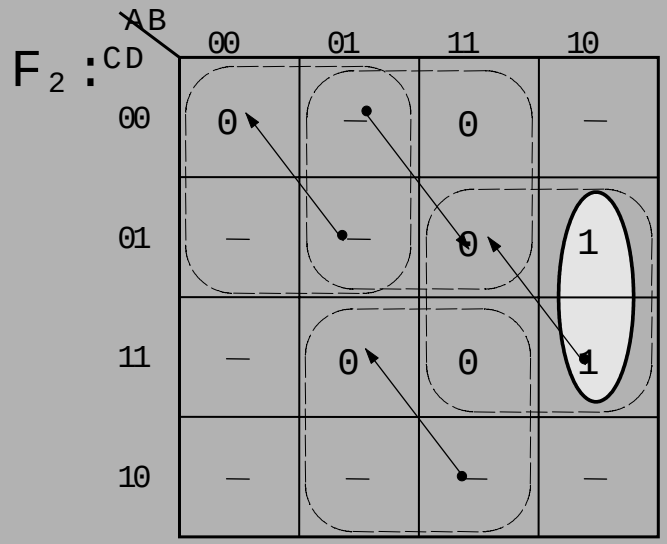


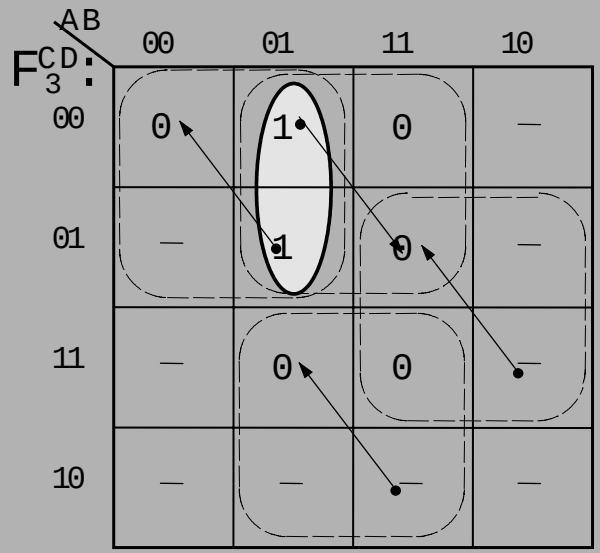
Given Boolean Function + Specified Transitions

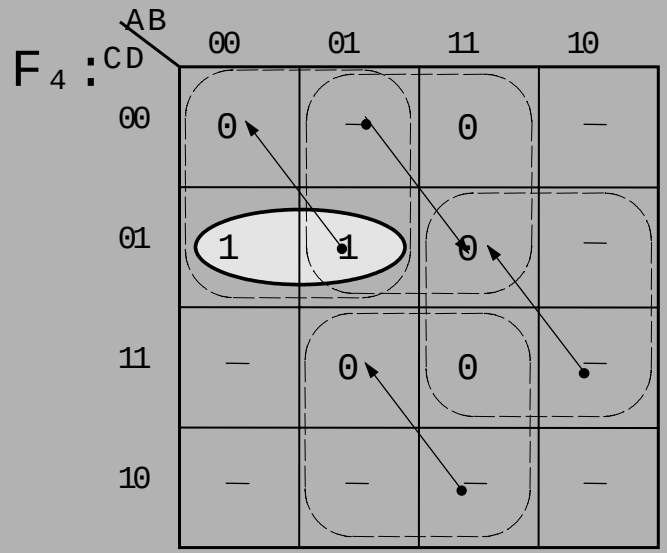
Level-0: Maximal Disjunctive Decomposition



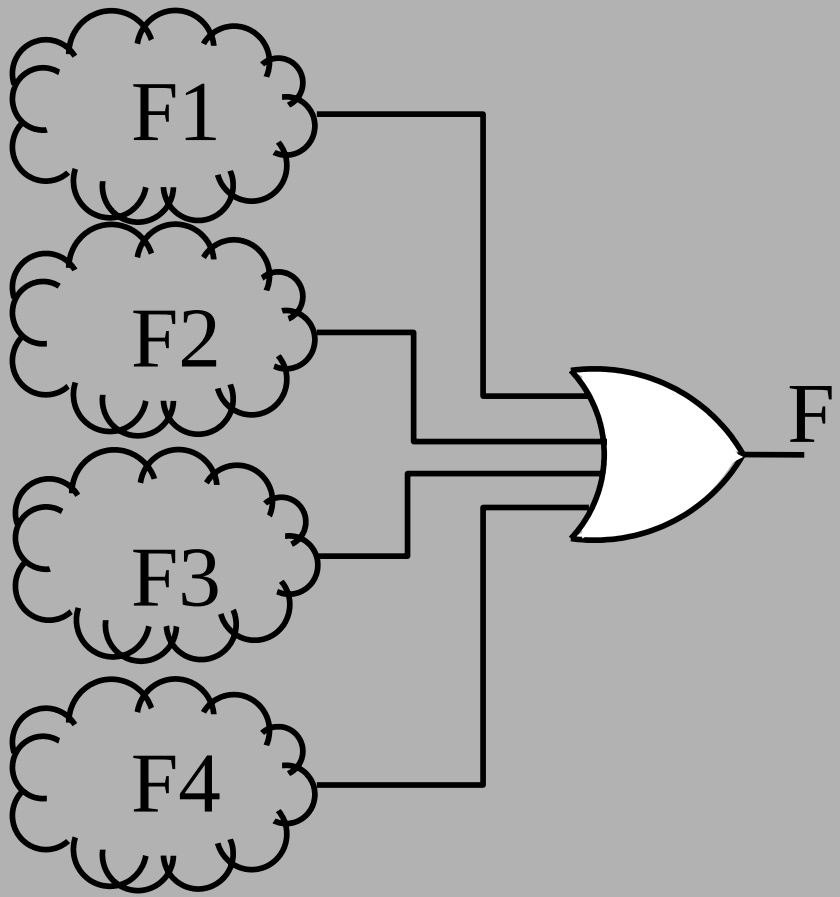




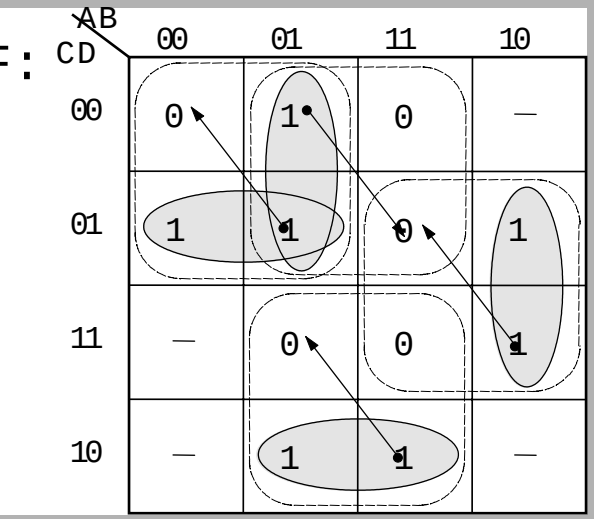


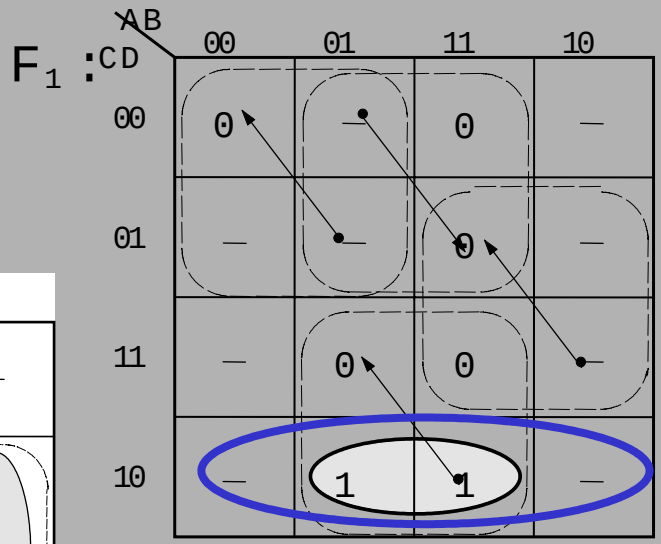


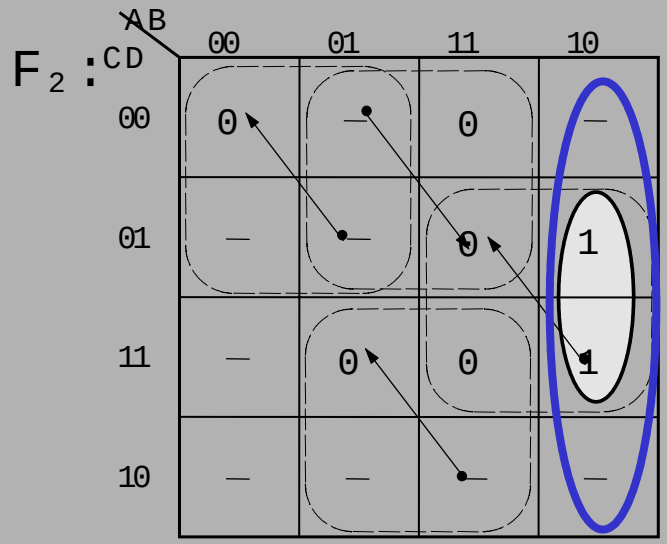
Level-0: Maximal Disjunctive Decomposition (cont.)

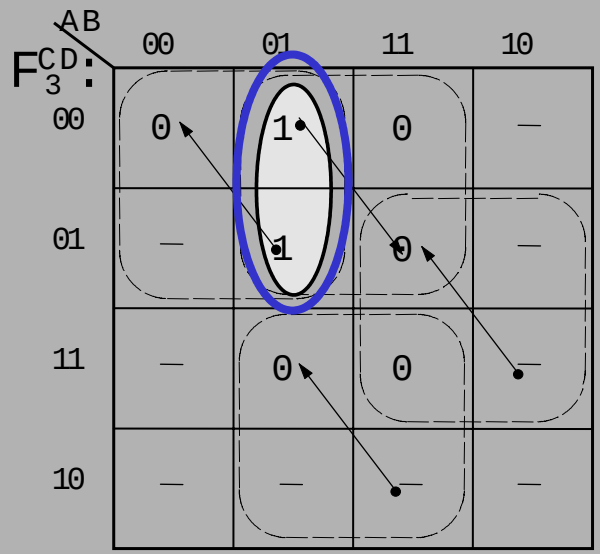


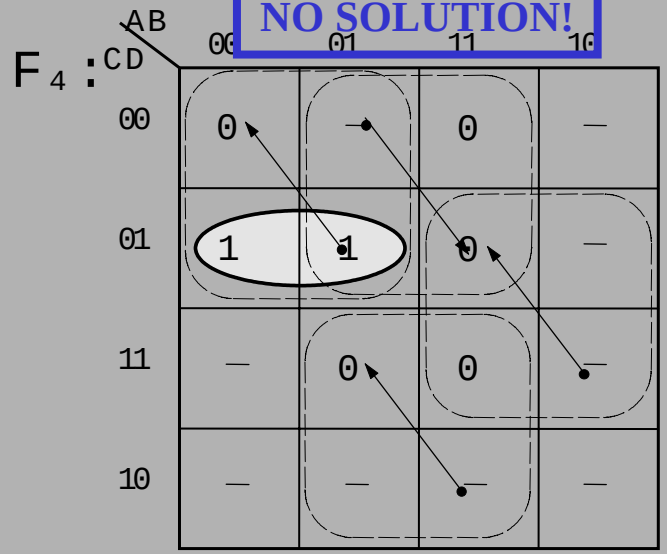
Level-0: Maximal Disjunctive Decomposition (cont.)





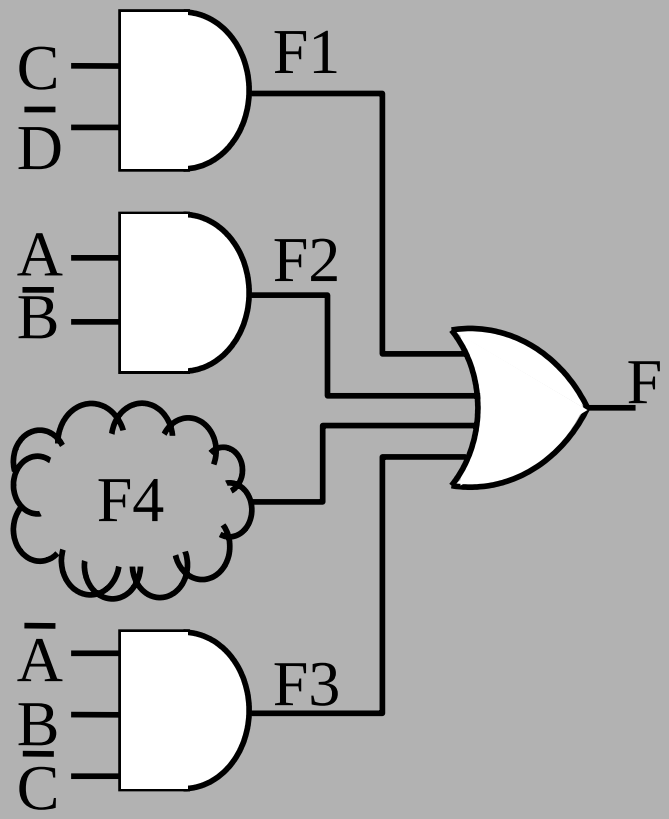




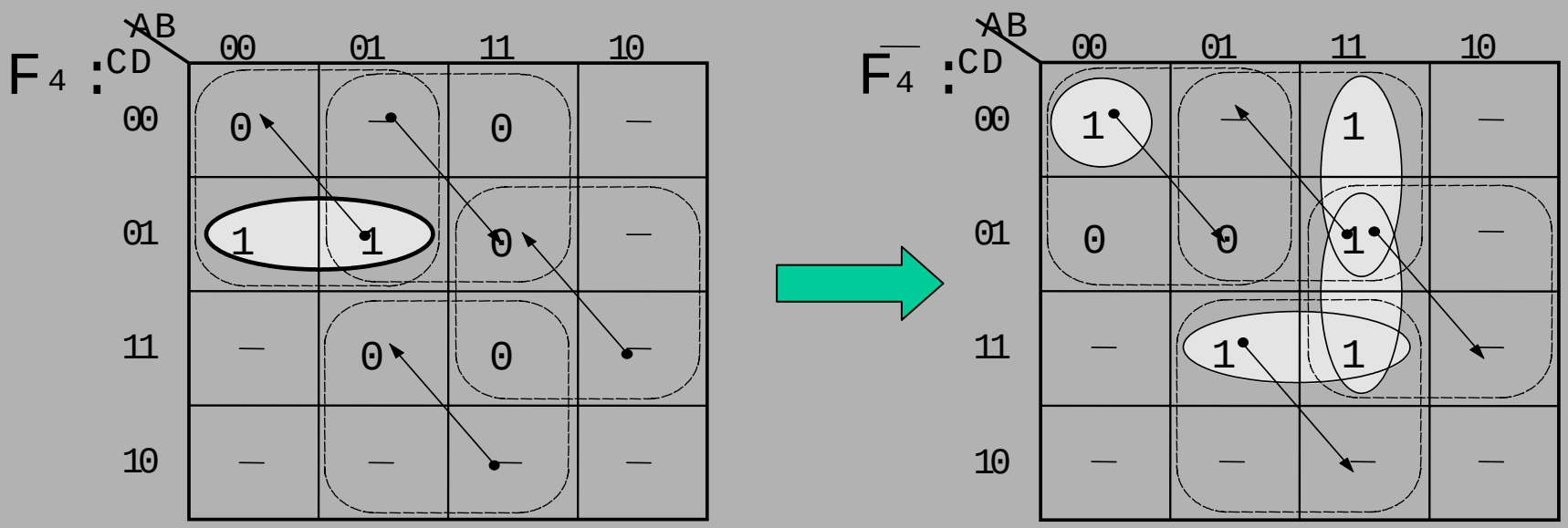


NO SOLUTION!

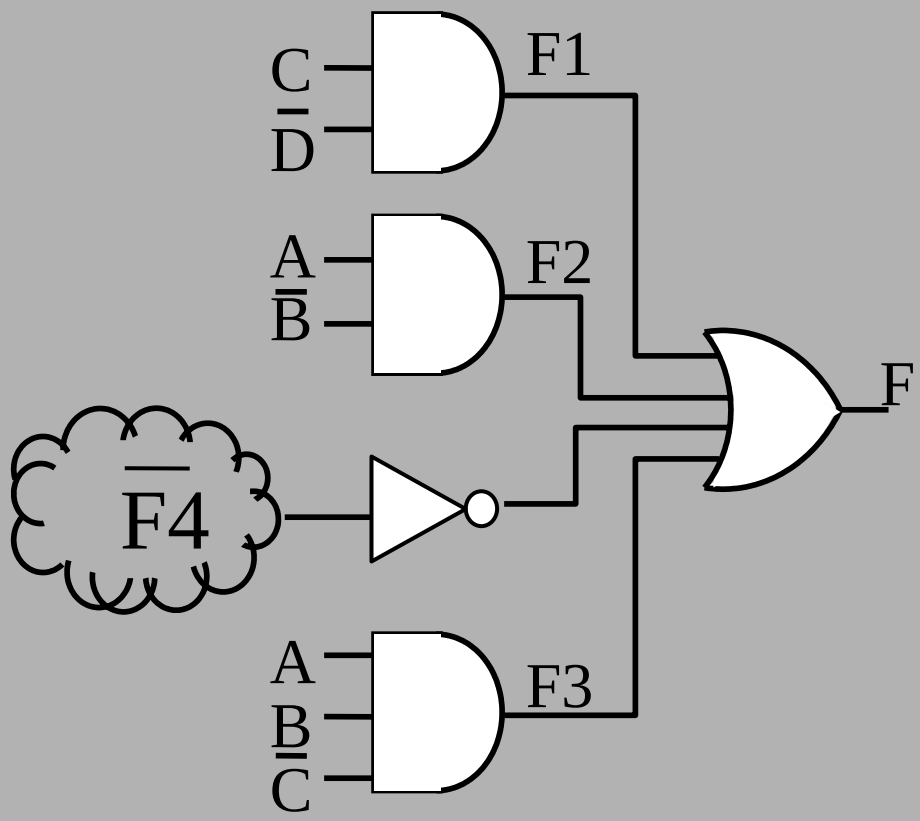
Level-0: Maximal Disjunctive Decomposition (cont.)



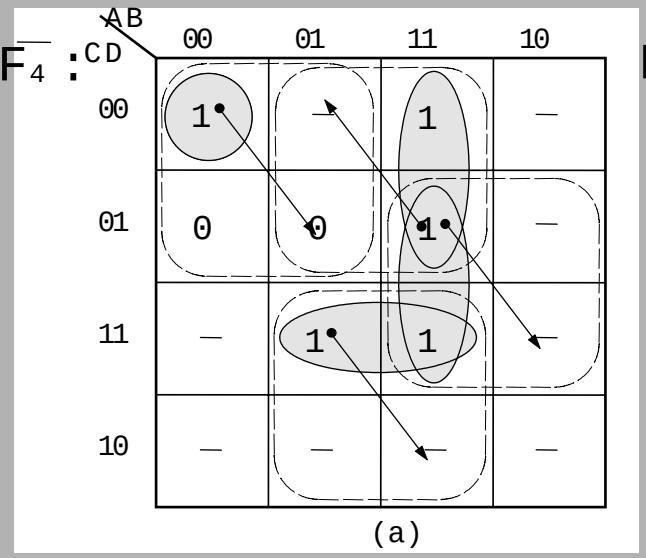
Level-1: Inversion

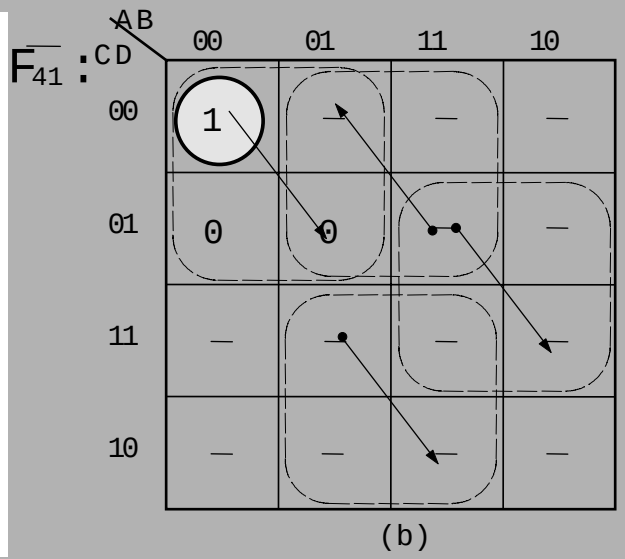


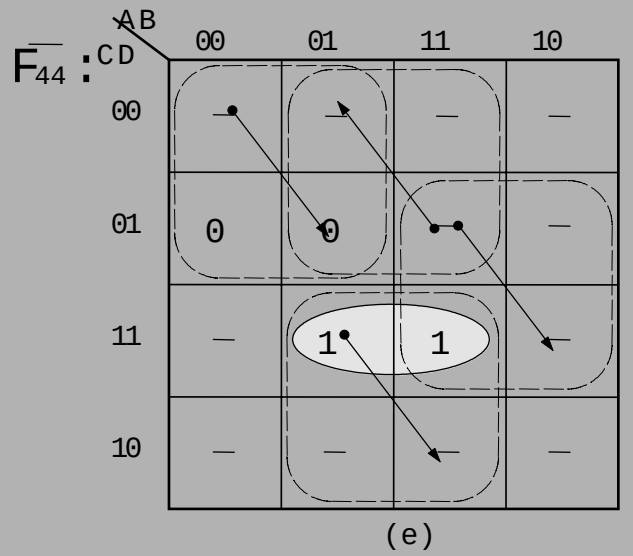
Level-1: Inversion (cont.)

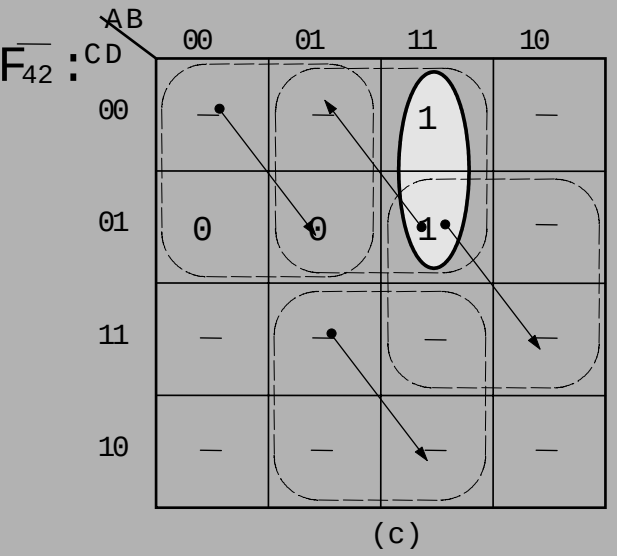


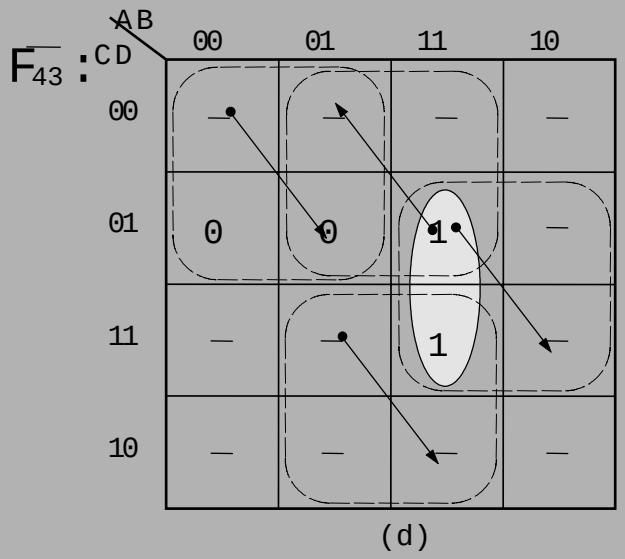
Level-1: Maximal Disjunctive Decomposition



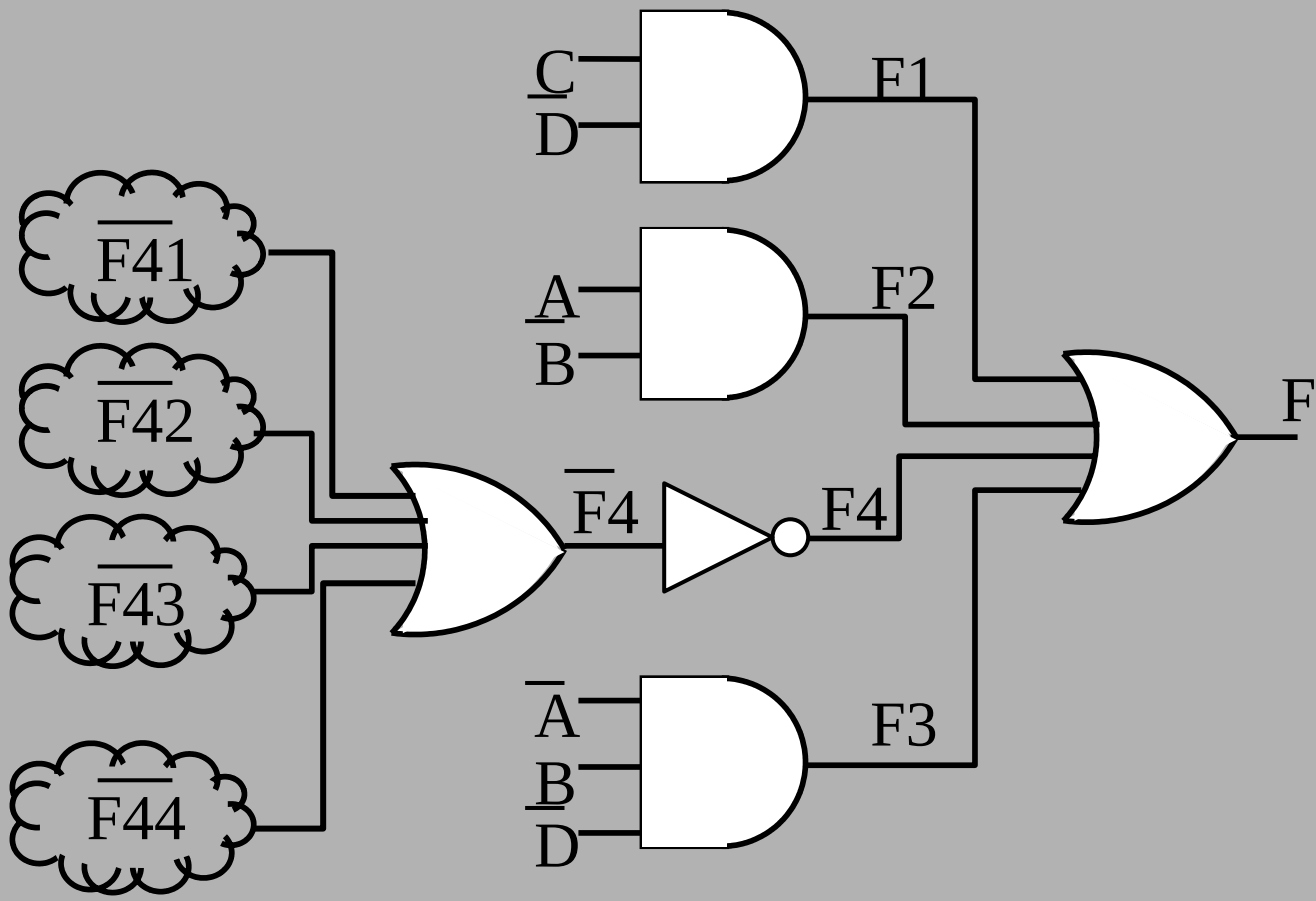




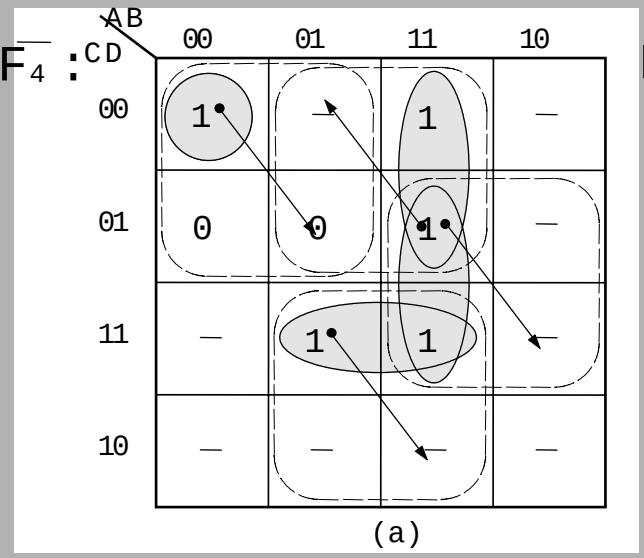


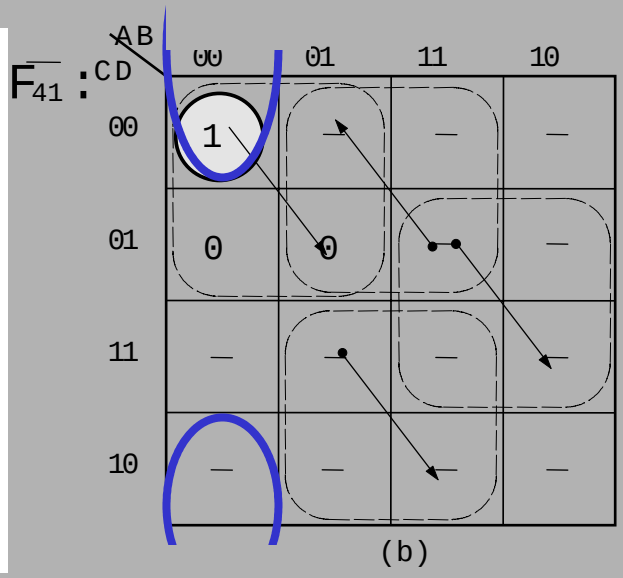


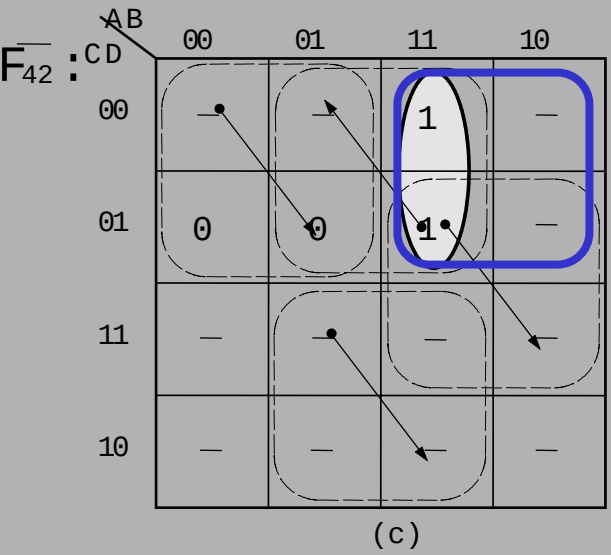
Level-1: Maximal Disjunctive Decomposition (cont.)

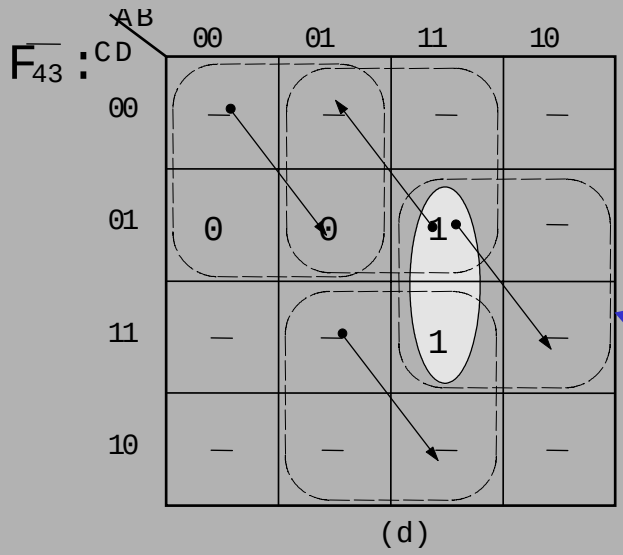


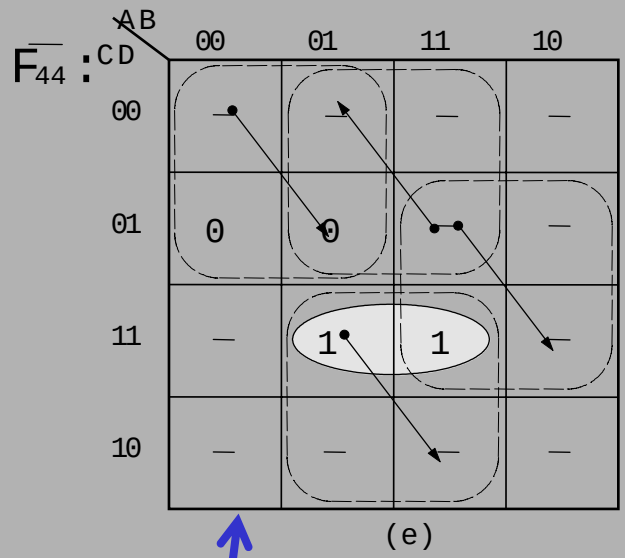
Level-1: Maximal Disjunctive Decomposition (cont.)





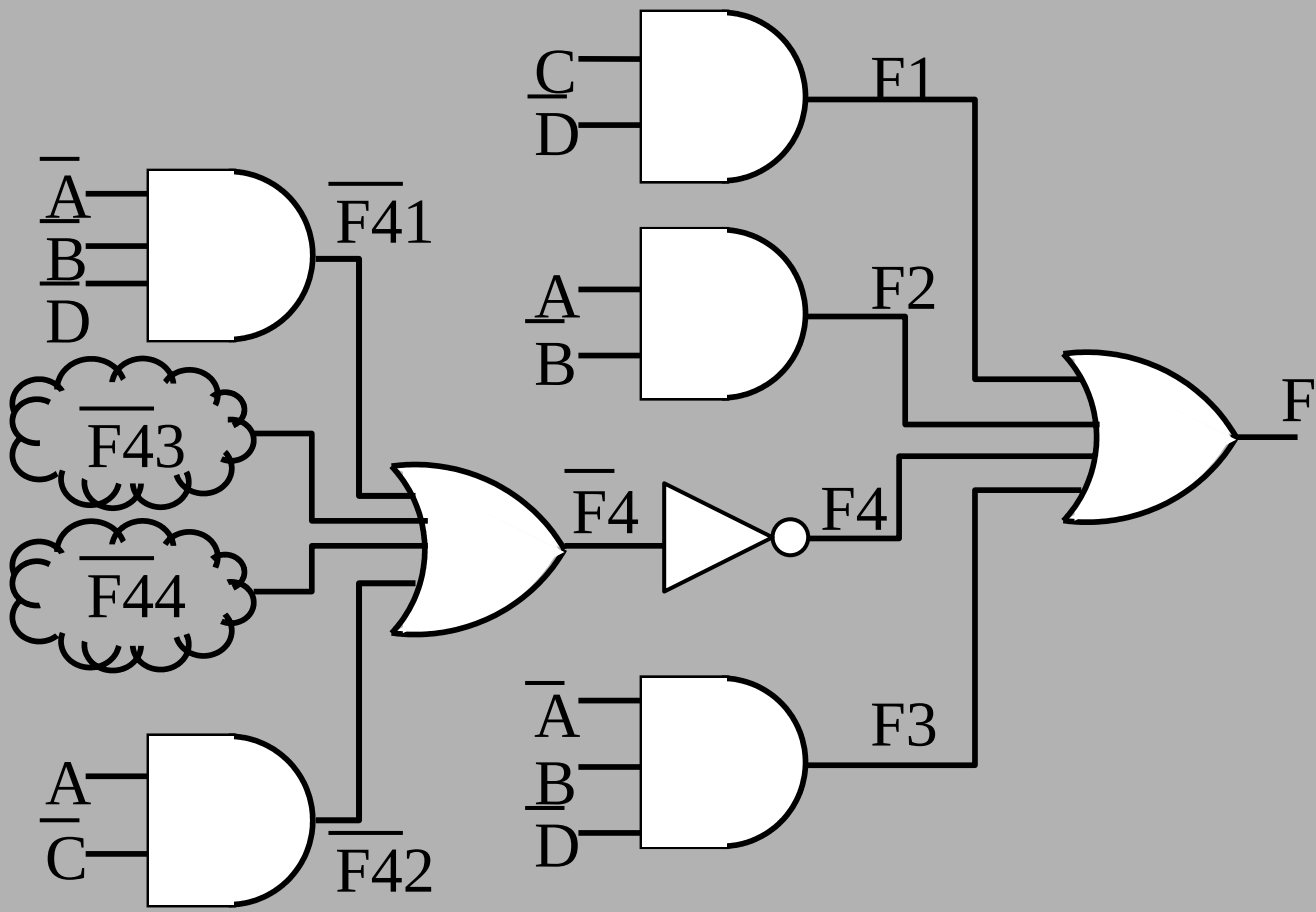




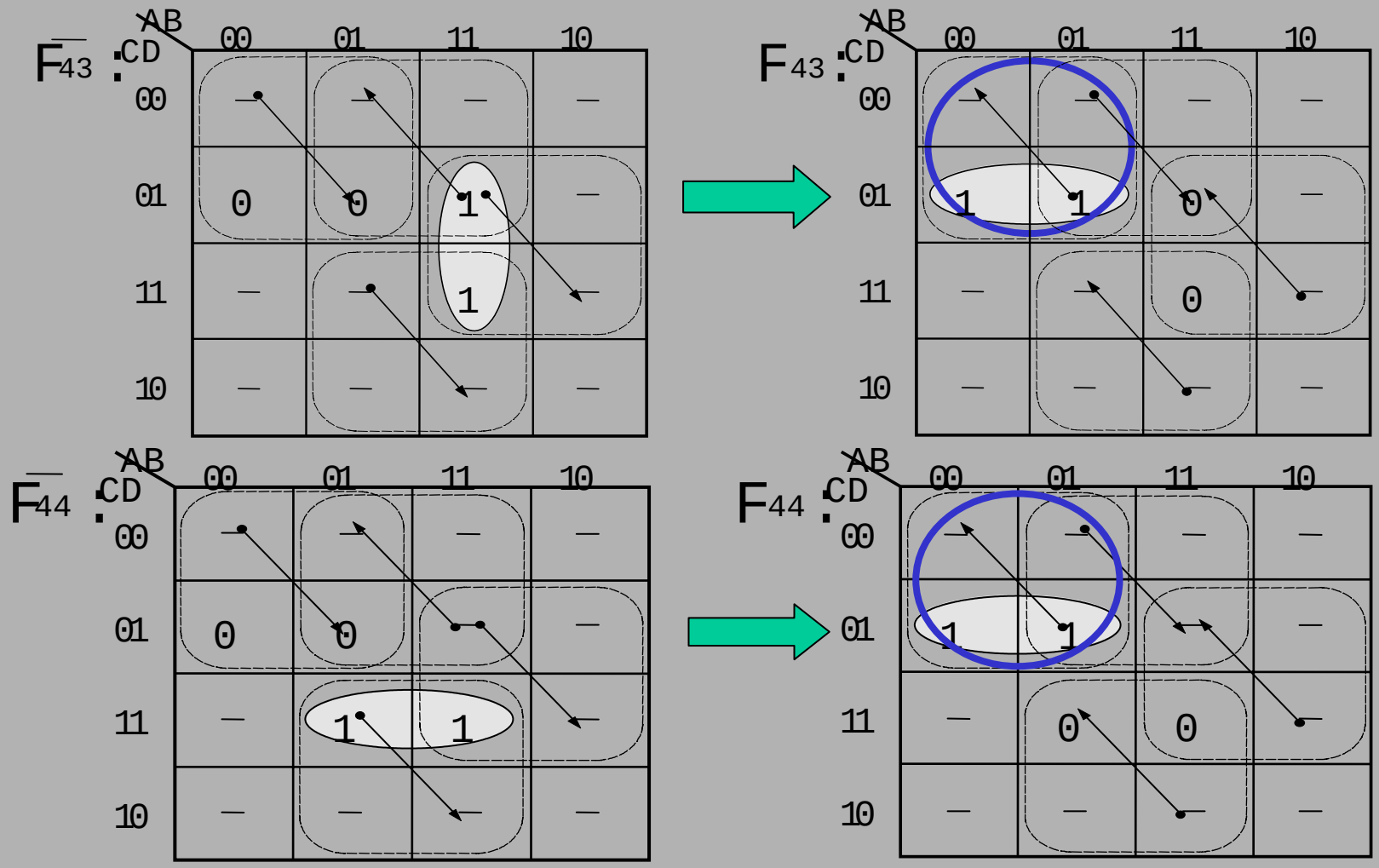


NO SOLUTION

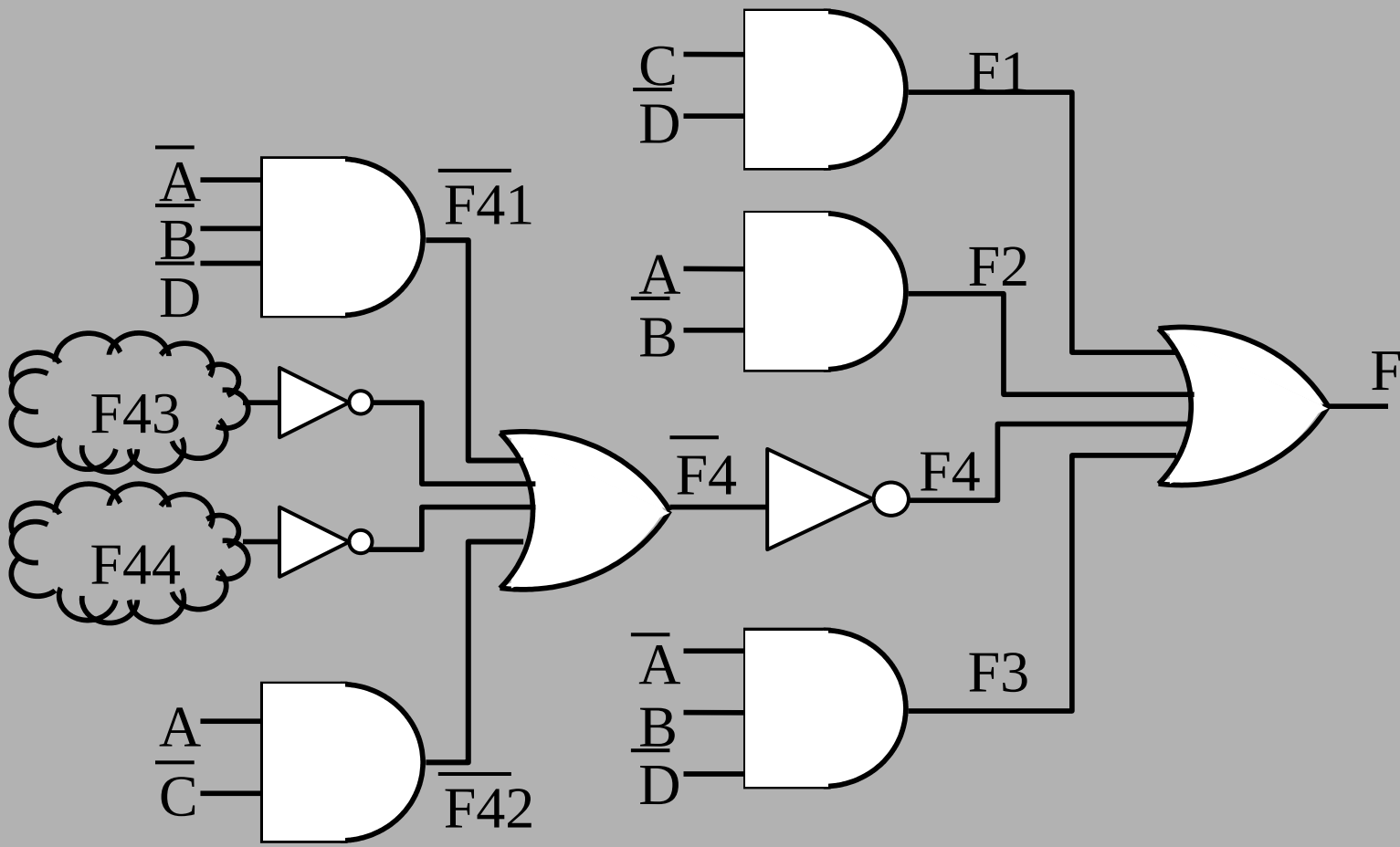
Level-1: Maximal Disjunctive Decomposition (cont.)



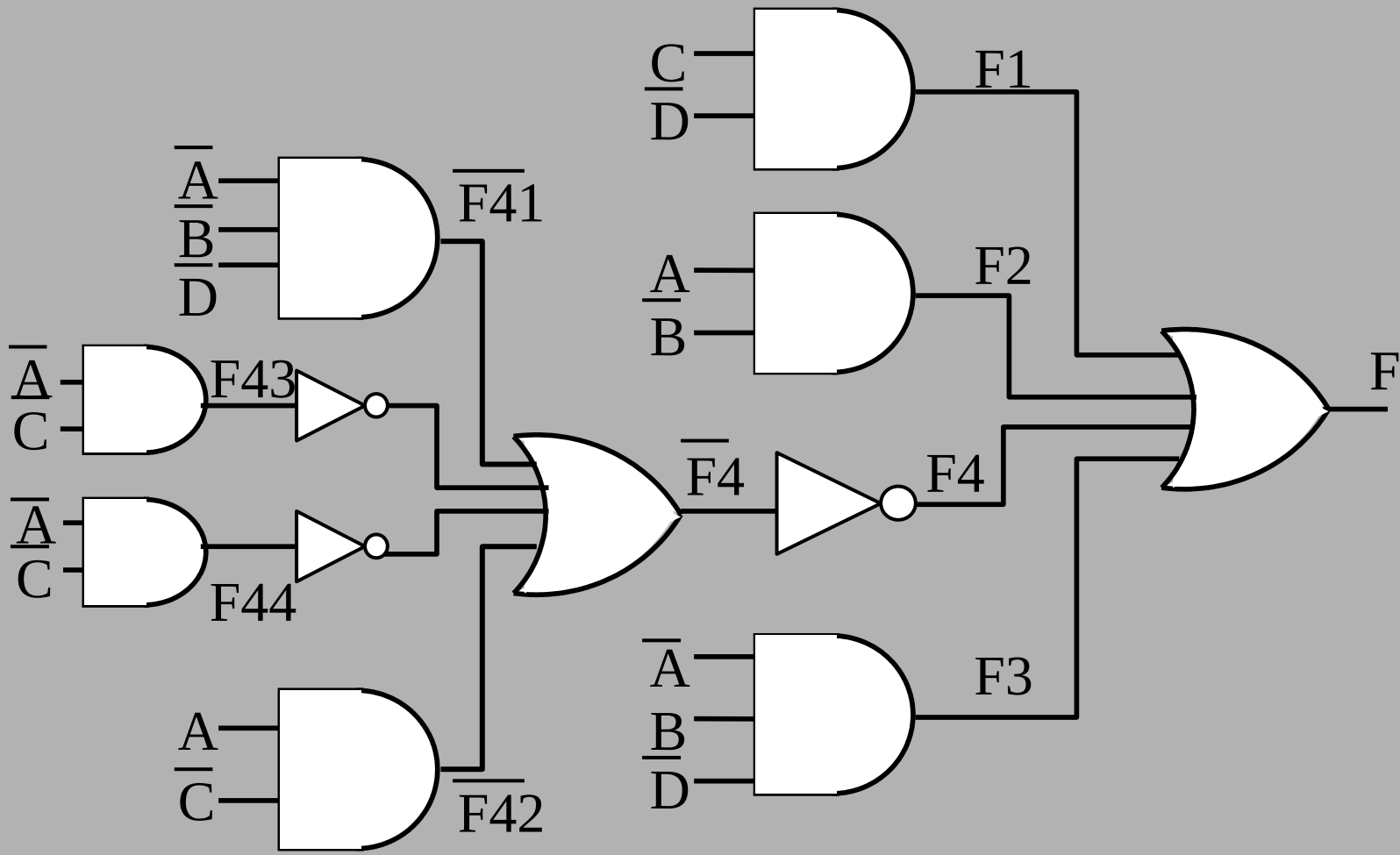
Level-2: Inversion



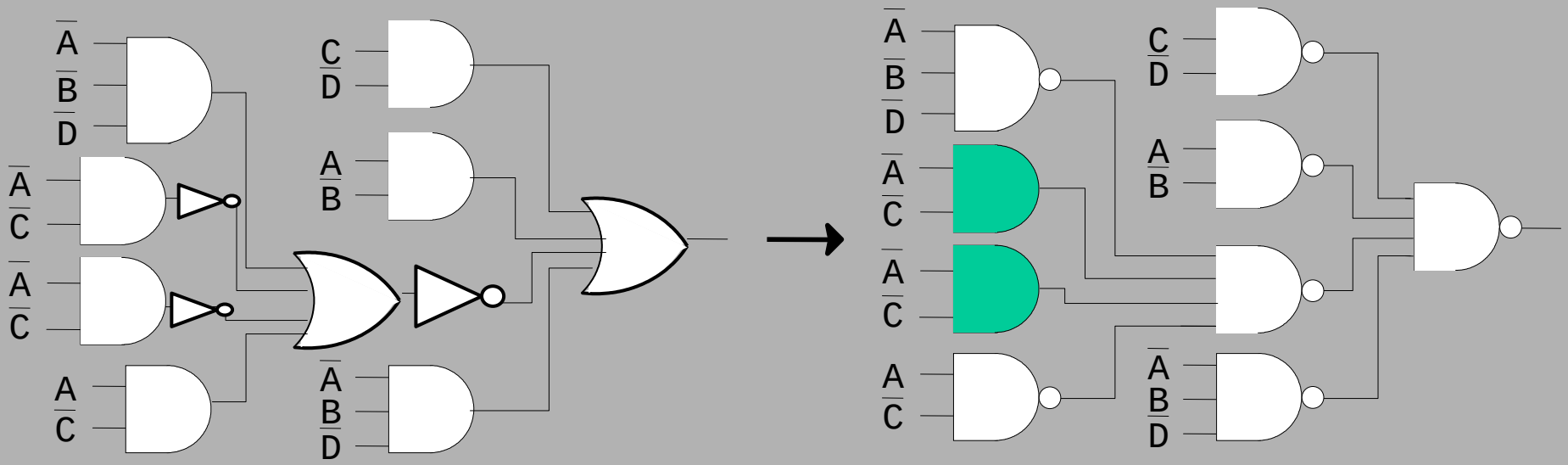
Level-2: Inversion (cont.)



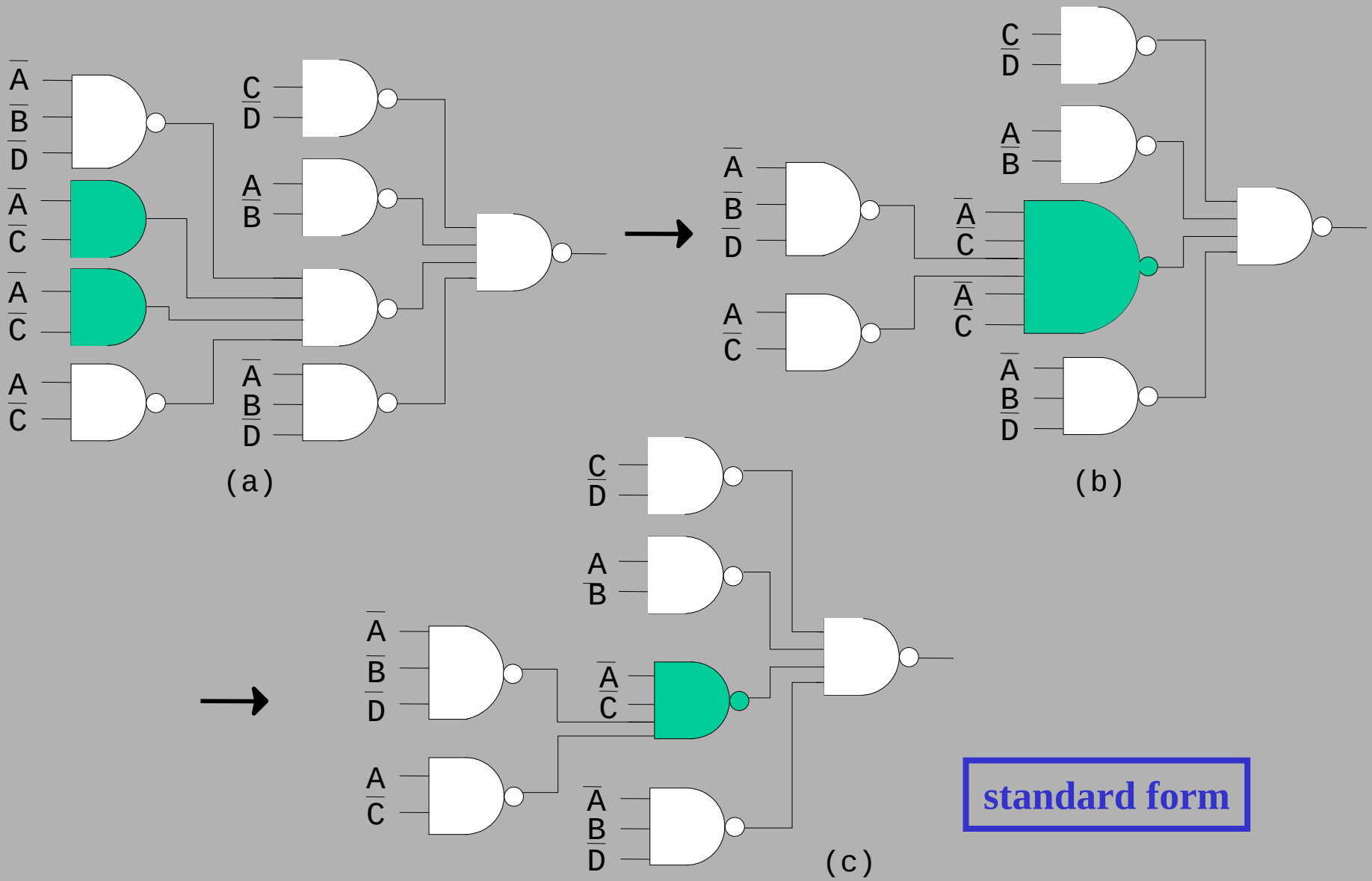
Final Hazard-Free Multi-Level Implementation



Deriving a 3-Level NAND Implementation: First Step

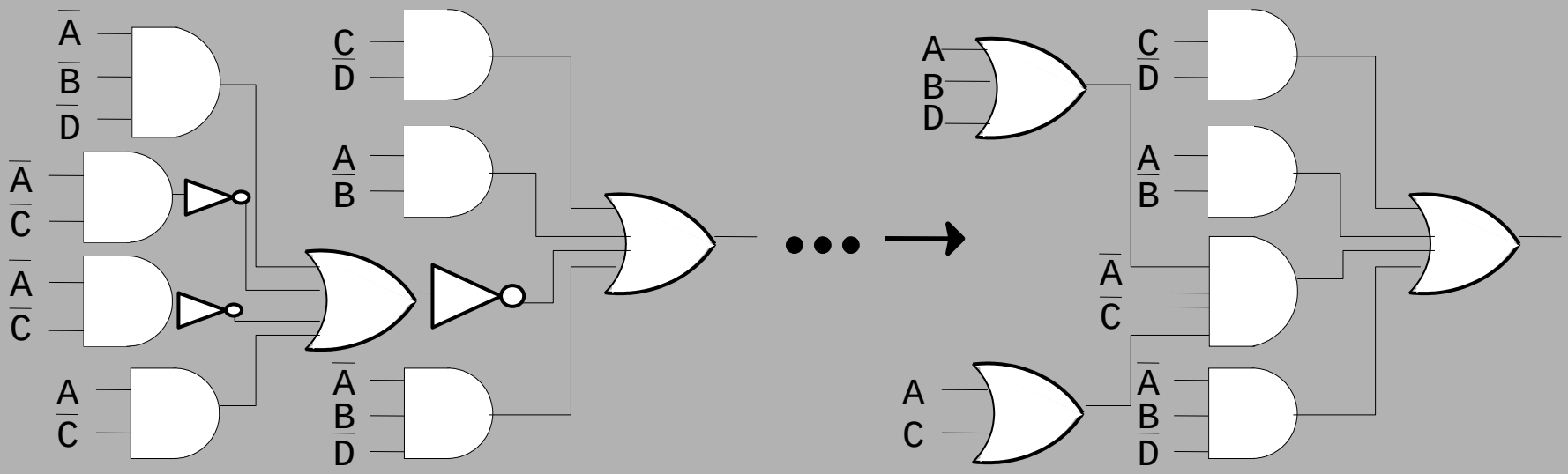


Deriving a 3-Level NAND Implementation: Final Step



standard form

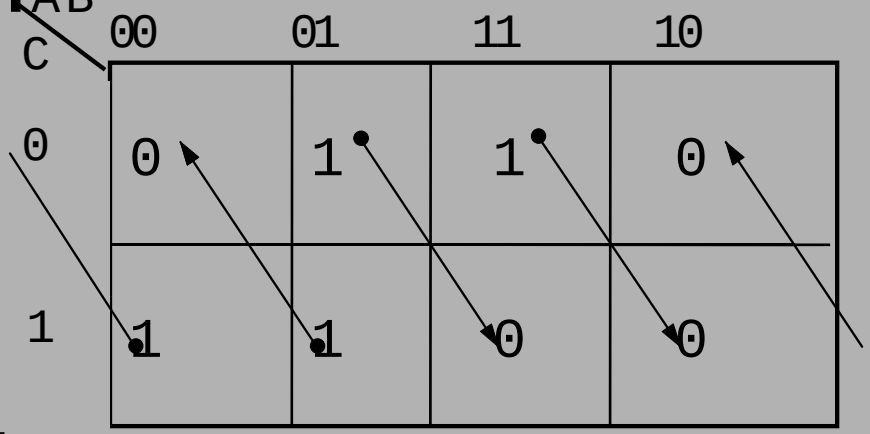
Deriving a 3-Level OR-AND-OR Implementation



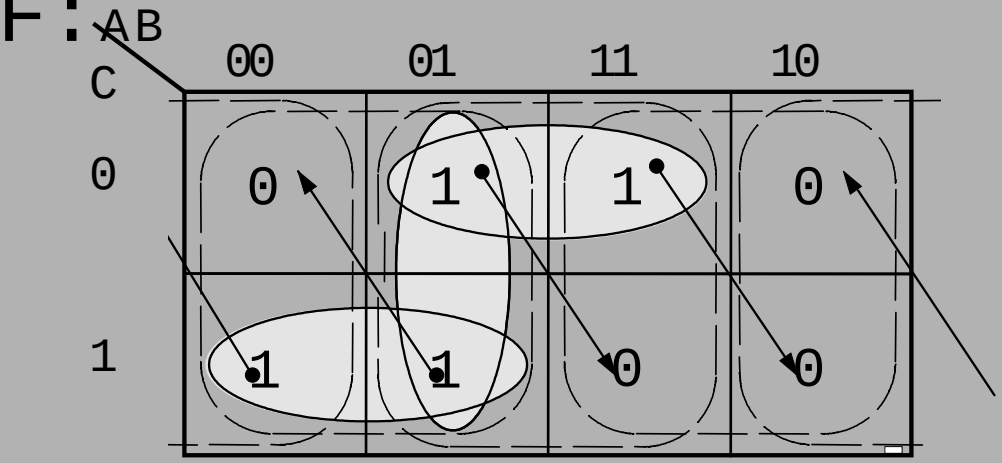
standard form

Second Example: No Hazard-Free Multi-Level Solution

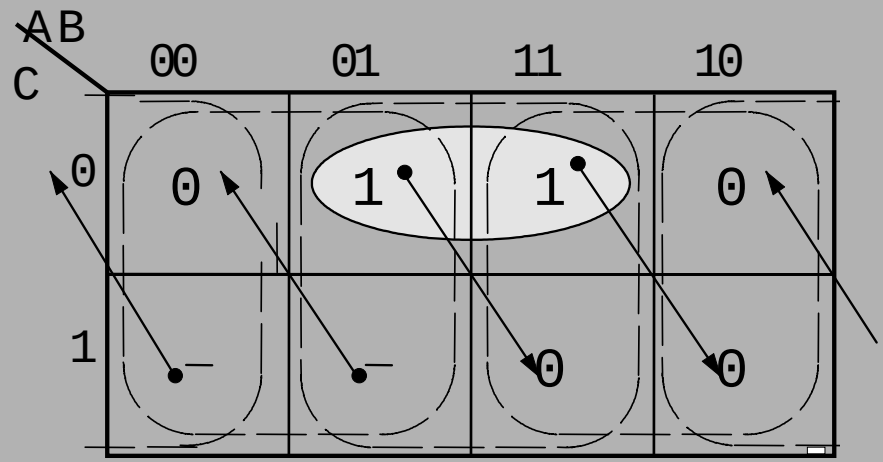
F:



F:



F_1:

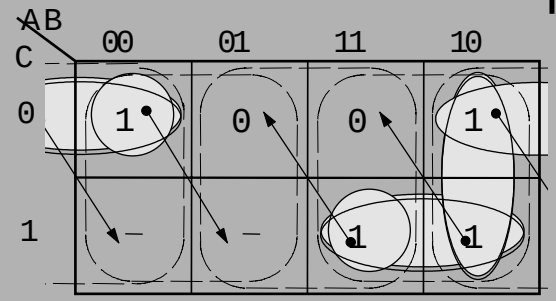


One of the subfunctions:
after max disjunctive decomposition

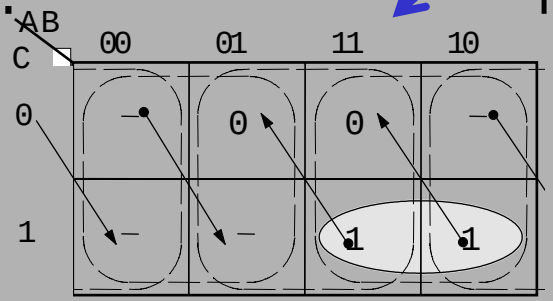
Second Example: No Multi-Level Solution (cont.)

no (single-product) hazard-free implementation

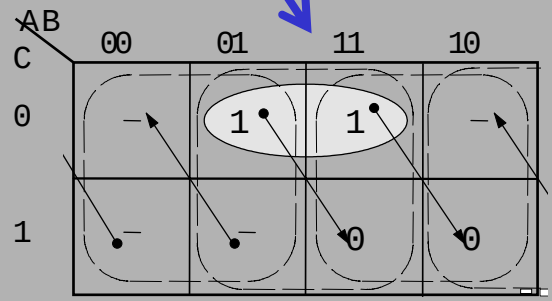
$\bar{F}_1$:



$\bar{F}_{11}$:



F_{11}:



after inversion → One of the subfunctions: after max disjunctive decomposition → after inversion

Conclusion: no hazard-free multi-level solution exists!

Other Contributions

Correctness of Method:

- **Soundness:** only produces hazard-free implementations
- **Completeness:** if *any* hazard-free multi-level solution exists, method will always produce a solution

Canonicity:

- variant of algorithm: identifies unique "representative" solution

Iterative Algorithm: (…see paper)

- **Observation:** after 3 recursive levels, ***sub-problems repeat***
  → can terminate recursion
- Propose ***3-level iterative algorithm***

Summary

Given: a Boolean function F + specified set of transitions T.

If (F,T) has *any* hazard-free multi-level solution, then it *always* will have a *hazard-free 3-level implementation*:

- 3-Level NAND
- OR-AND-OR

The proposed iterative method will always find such a solution… if one exists.

Outline

- **Background:** Hazard-Free 2-Level Logic
- **Overview of Multi-Level Method**
  - Hazard-Free Decomposition
  - Algorithm + Examples
  - Standard 3-Level Circuit Structures
  - Canonicity
- **Related Work:** Bredeson [’74]
- **Experimental Results**
- **Conclusions**

Related Work: Bredeson [‘74]

First known approach to tackle the problem:

- **uses recursive algorithm:** alternates disjunction + inversion

Limitations:

- **does not identify that solutions are always 3-level**
- **does not address canonicity**
- **highly-restricted formulation:**
  - tight restrictions on allowed input transitions
  - limited notion of hazard-free decomposition
  - cannot handle incompletely-specified functions

Experimental Results

Prototype CAD Tool:

- Reads in Berkeley PLA format + input transition list
- Produces hazard-free multi-level implementation

  -- or indicates if no solution exists

Function Name	# Inputs	Solution Exists?	Total # Subfunctns	Total # Inversions	Max Inverters per Level	Recursn Depth	CPU Time (ms)
bredeson	3	No	4	3	1	2	0.36
simple_ex	4	Yes	3	0	0	0	0.11
nowick_dill	4	Yes	8	1	1	1	0.59
z1	4	Yes	19	3	3	1	2.28
z2	4	No	12	10	4	2	1.42
z3	4	Yes	8	2	2	1	0.68
z4	4	Yes	4	0	0	0	0.18
z5	4	No	11	14	2	2	2.23
z6	4	Yes	7	3	2	2	0.99

Conclusions

- **General Constructive Method:**
  - given function **F** and set of input transitions **T**, determine if any hazard-free multi-level solution exists
  - based on "hazard-free decomposition": alternate **disjunction** + **inversion**
- **Key Observation:**
  - if *any solution exists*, then *so does a 3-level solution*
  - target two forms: **(a) 3-level NAND, (b) OR-AND-OR**
- **Potential Practical Benefits:**
  - can use as starting point of hazard-free logic synthesis
  - can simplify conservative restrictions in sequential synthesis
  - provides deeper understanding of nature of hazard-free logic….