COMP290-084
Clockless Logic

Prof. Montek Singh

Jan. 29, 2004

Acknowledgment

Michael Theobald and Steven Nowick,

for providing slides for this lecture.

An Implicit Method for Hazard-Free Two-Level Logic Minimization

Michael Theobald and Steven M. Nowick

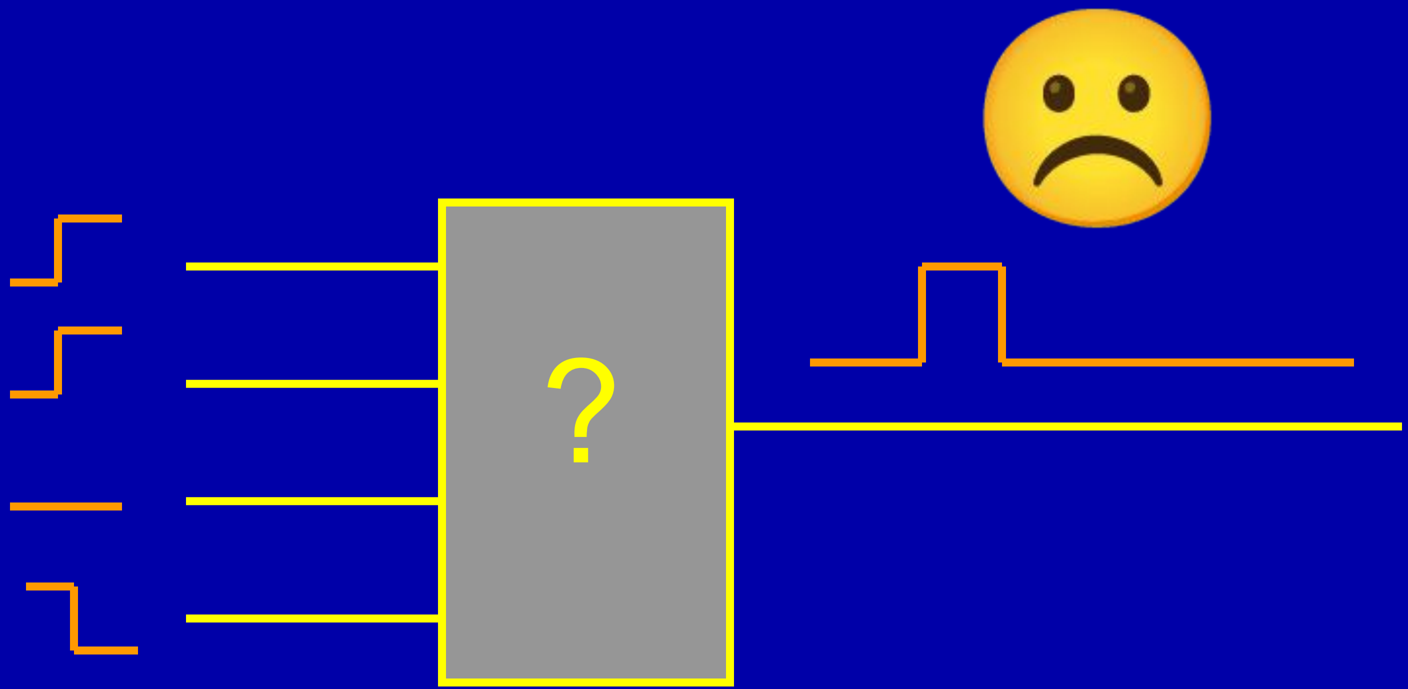
Columbia University, New York, NY

Paper appeared in Async-98

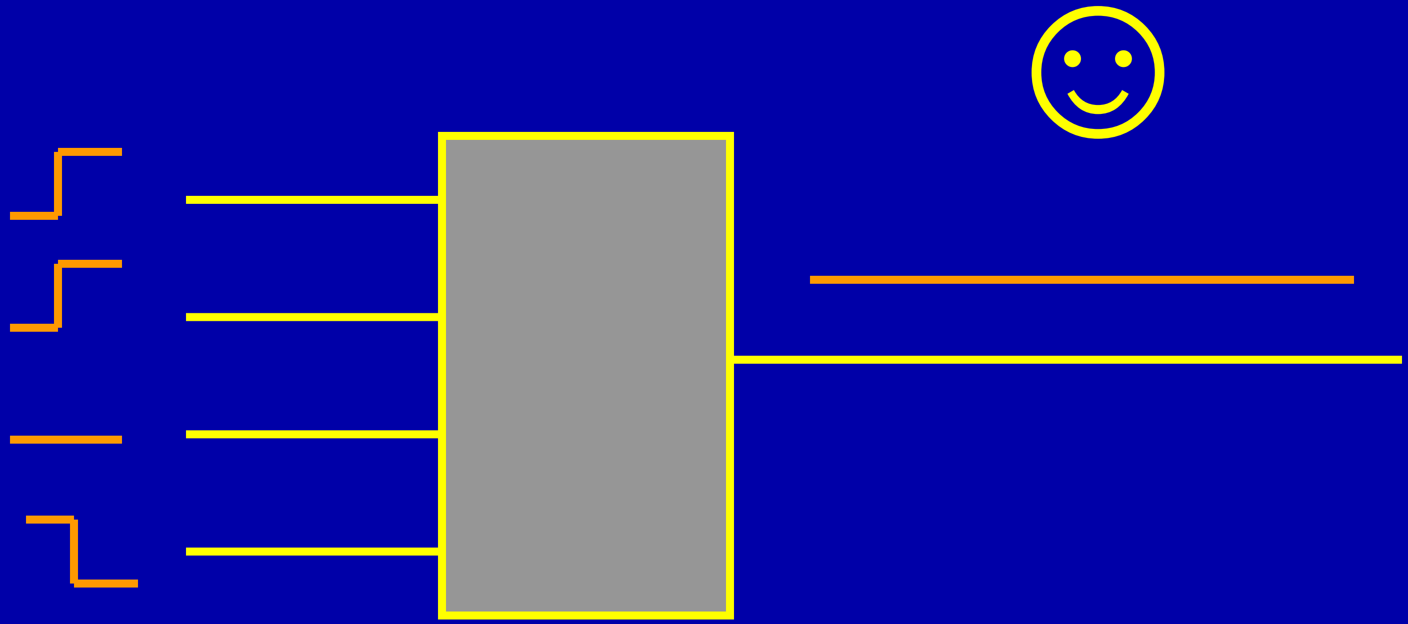
(Best Paper Finalist)

Hazard-Free Logic Minimization

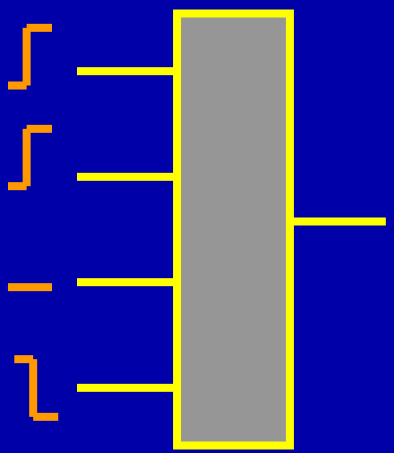
Given: Boolean function and multi-input change



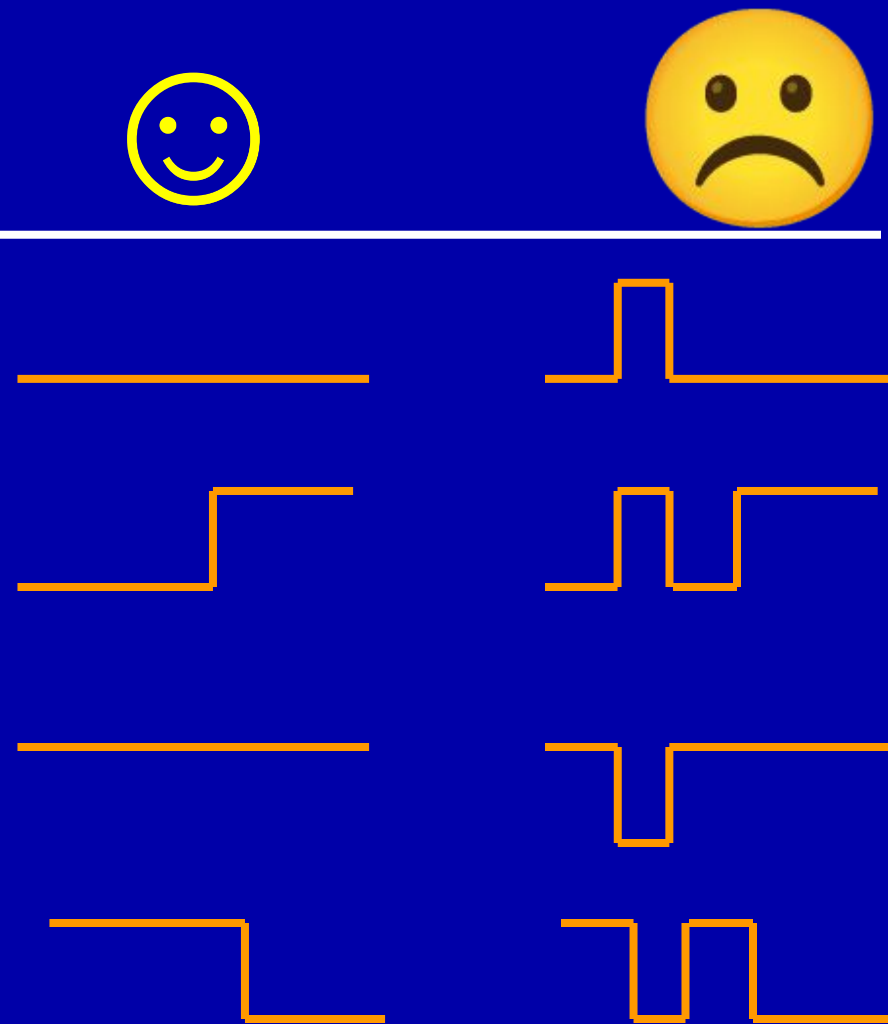
Hazard-Free Logic Minimization



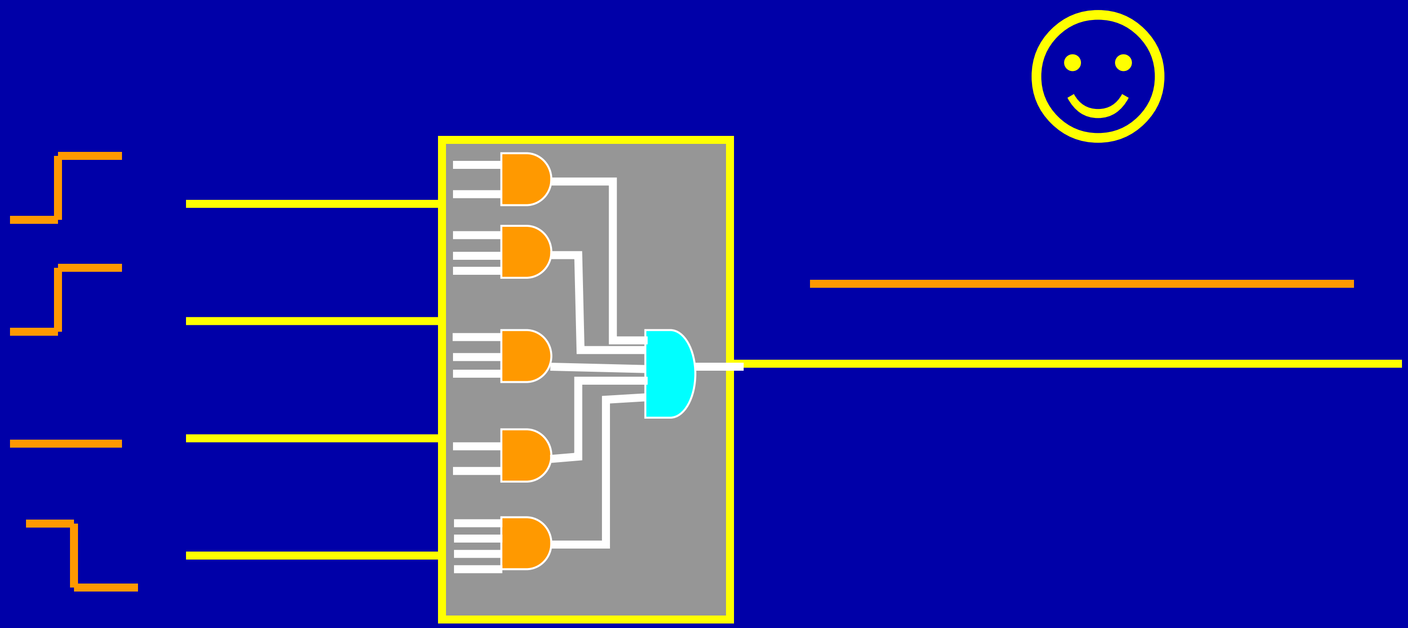
Hazard-Free Logic Minimization



f(A) □ f(B)
0 □ 0
0 □ 1
1 □ 1
1 □ 0



Hazard-Free 2-Level Logic Minimization



Classic 2-Level Logic Minimization

Quine-McCluskey Algorithm

Step 1. Generate Prime Implicants

1's: "Minterms"

Ovals: "Prime Implicants"

Karnaugh-Map:

0	0
0	1
1	1
1	0

Step 2. Select Minimum # of Primes ...to cover all Minterms

	Prime implicants
Minterms	

2-level Logic Minimization: Classic vs. Hazard-Free

- Classic (Quine-McCluskey):

  <On-set minterms, Prime implicants>

- Hazard-Free:

  <Required cubes, DHF-Prime implicants>

  - Given: Boolean function & set of "multi-input" changes
  - Find: *min-cost* 2-level implementation guaranteed to be <u>glitch-free</u>
  - Required cubes = <u>sets of minterms</u>
  - DHF-Prime implicants =

    <u>maximal implicants</u> that do not intersect privileged cubes illegally

Hazard-Free Logic Minimization

Multi-Input Changes:

- Non-monotonic
  - *function hazard*
  - no implementation hazard-free
- Monotonic
  - *function-hazard-free*

0	0	0	0
0	1	1	0
1	1	0	0
1	0	0	0

☹ ☺

Restriction to monotonic changes

Hazard-Freedom Conditions: 1 -> 1 transition

0	0	0	0
0	1	0	0
0	1	0	0
0	1	_	0

☹

0	0	0	0
0	1	0	0
0	1	0	0
0	1	_	0

☺ Required Cube <u>must be covered</u>

Hazard-Freedom Conditions: 1 -> 0 transition

0	0	0	0
0	1	1	0
0	1	0	0
0	1	0	0

Hazard-Freedom Conditions: 1 -> 0 transition

0	0	0	0
0	1	1	0
0	1	0	0
0	1	0	0

Hazard-Freedom Conditions: 1 -> 0 transition

0	0	0	0
0	1	1	0
0	1	0	0
0	1	0	0

Hazard-Freedom Conditions: 1 -> 0 transition

0	0	0	0
0	1	1	0
0	1	0	0
0	1	0	0

Hazard-Freedom Conditions: 1 -> 0 transition

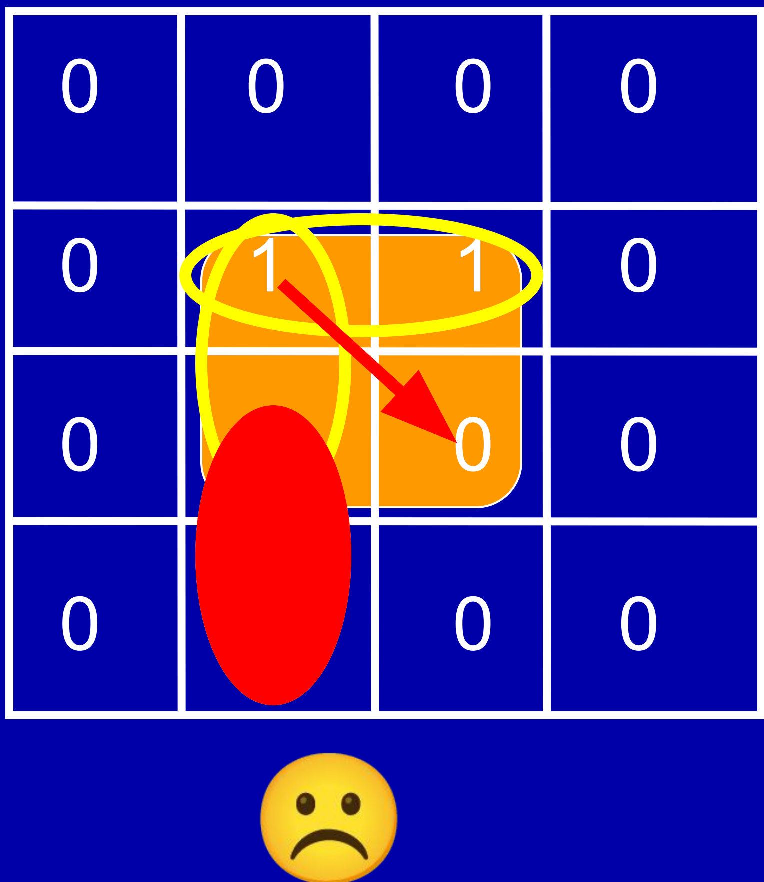
0	0	0	0
0	1	1	0
0	1	0	0
0	1	0	0

Hazard-Freedom Conditions: 1 -> 0 transition

0	0	0	0
0	1	1	0
0	1	0	0
0	1	0	0

☹️

Hazard-Freedom Conditions: 1 -> 0 transition



illegal intersection

Hazard-Freedom Conditions: 1 -> 0 transition

0	0	0	0
0	1	1	0
0		0	0
0		0	0

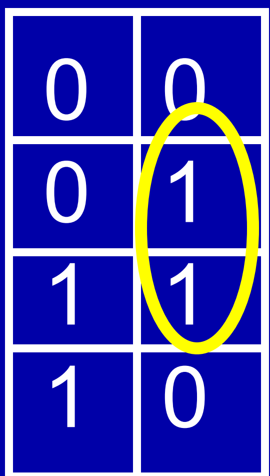
☹

illegal intersection

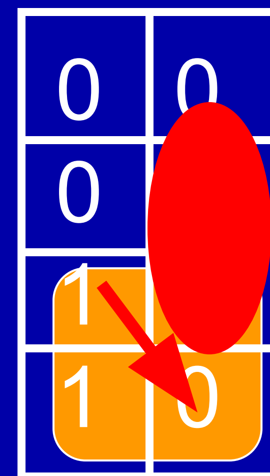
0	0	0	0
0	1	1	0
0	1	0	0
0	1	0	0

☺ No illegal intersection of privileged cube

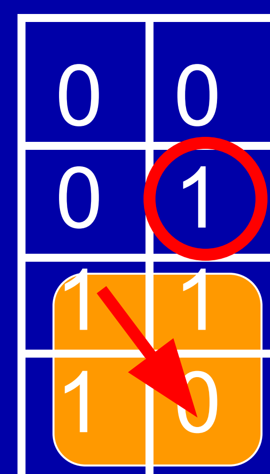
Dynamic-Hazard-Free Prime Implicants



Prime



NO DHF-Prime illegal intersection



DHF-Prime

2-level Logic Minimization: Classic vs. Hazard-Free

- Classic (Quine-McCluskey):

  <On-set minterms, Prime implicants>

- Hazard-Free:

  <Required cubes, DHF-Prime implicants>

  – Given: Boolean function & set of "multi-input" changes
  – Find: *min-cost* 2-level implementation guaranteed to be glitch-free
  – Required cubes = sets of minterms
  – DHF-Prime implicants =

    maximal implicants that do not intersect privileged cubes illegally

Main challenge: Computing DHF-prime implicants

Hazard-Free 2-level Logic Minimization: Previous Work

- Early work (1950s-1970s):
  - Eichelberger, Unger, Beister, McCluskey
- Initial solution: Nowick/Dill [ICCAD 1992]
- Improved approaches:
  - HFMIN: Fuhrer/Nowick [ICCAD 1995]
  - Rutten et al. [Async 1999]
  - Myers/Jacobson [Async 2001]

→ No approach can solve large examples

IMPYMIN: an exact 2-level minimizer

- Two main ideas:
  - novel reformulation of hazard-freedom constraints
    - used for dhf-prime generation
    - recasts an asynchronous problem as a synchronous one
  - uses an "implicit" method
    - represents & manipulates large # of objects simultaneously
    - avoids explicit enumeration
    - makes use of BDDs, ZBDDs
- Outperforms existing tools by orders of magnitude

Review: Primes vs. DHF-Primes

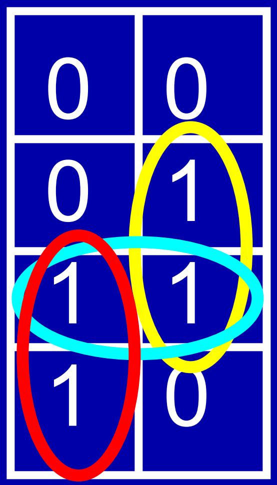
Classic (Quine-McCluskey):

<On-set minterms, Prime implicants>

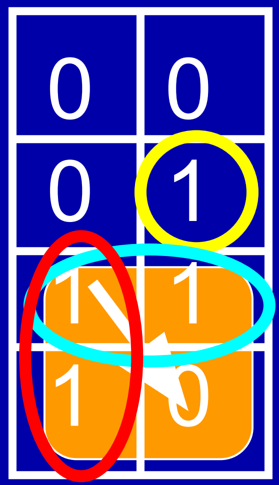
Hazard-Free:

<Required cubes, DHF-Prime implicants>

DHF-Prime Implicants = maximal implicants that do not intersect "privileged cubes" illegally



Primes



DHF-Primes

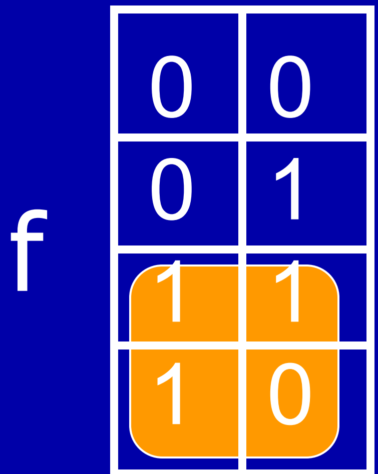
DHF-Prime Generation

- Challenge: Two types of constraints
  - maximality constraints: "we want maximally large implicants"
  - avoidance constraints: "we must avoid illegal intersections"

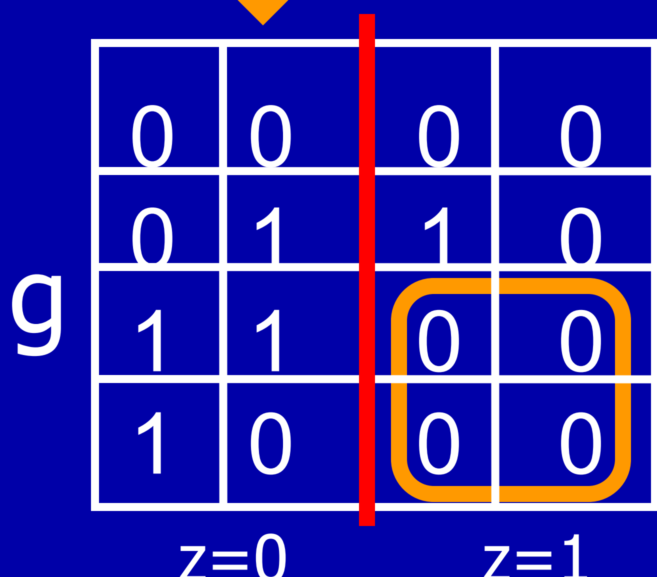
- New Approach: <u>Unify constraints</u> by "lifting" the problem into a higher-dimensional space:

f(x1,…,xn), T	maximality & avoidance constraints
↓	
g(x1, …, xn, z1, …, zl)	maximality

Auxiliary Synchronous Function *g*



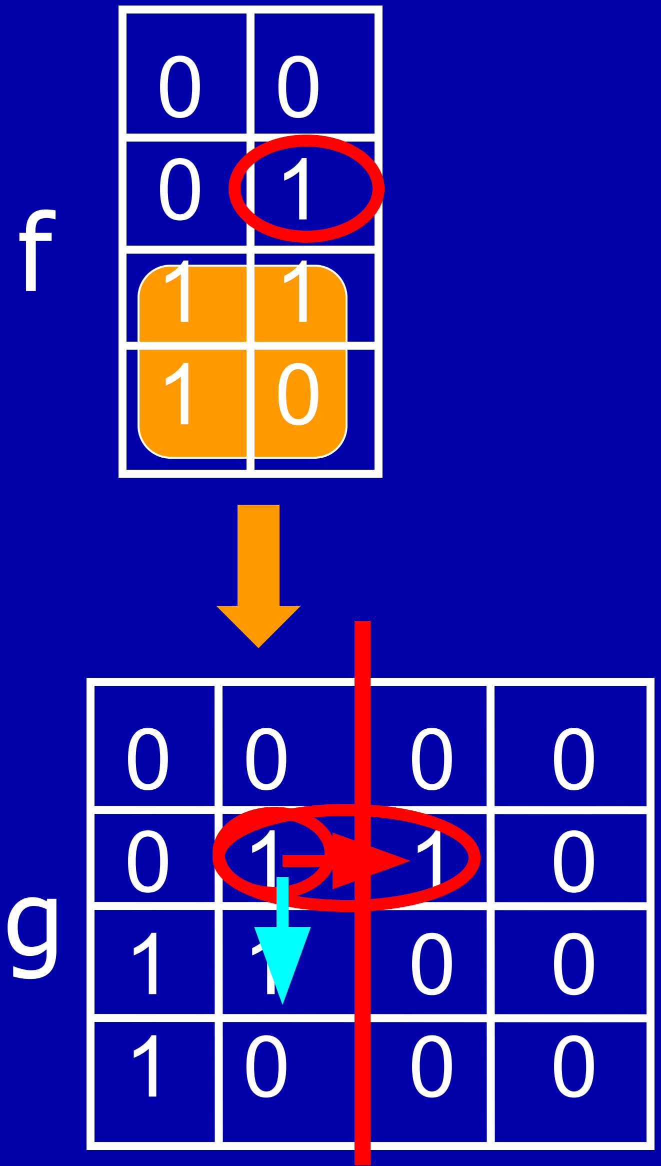
Add one new dimension per privileged cube



0-half-space: g is defined as f

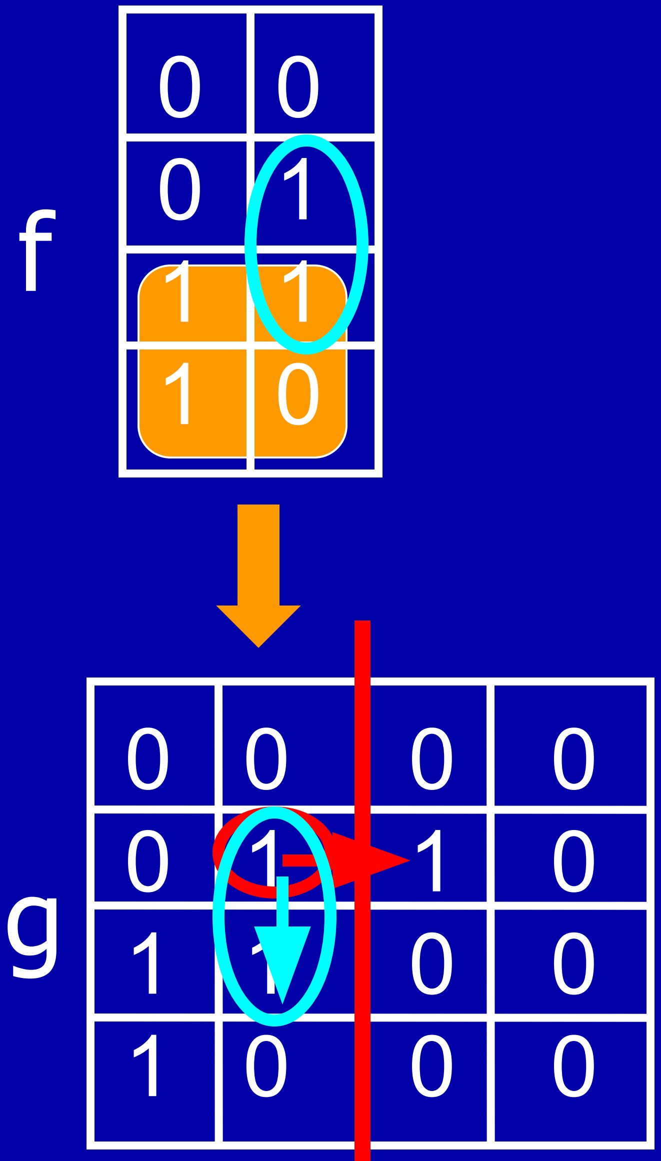
1-half-space: g is defined as f
BUT priv-cube is
filled with 0’s

Prime Implicants of g



Expansion in z-dimension
<u>guarantees avoidance</u>
of priv-cube in original domain

Prime Implicants of g



Expansion in x-dimension corresponds to enlarging cube in original domain.

Summary: Auxiliary Synchronous Function g

The definition of auxiliary function g **exactly** ensures :

- Expansion in a z-dimension corresponds to avoiding the privileged cube in the original domain.
- Expansion in a x-dimension corresponds to enlarging the cube in the original domain.

New approach: DHF-Prime Generation

Goal: Efficient new method for DHF-Prime generation

Approach:

- translate original function f into synchronous function g
- generate Primes(g)
- after filtering step, retrieve dhf-primes(f)

Prime Generation of g

f

0	0
0	1
1	1
1	0

g

0	0	0	0
0	1	1	0
1	1	0	0
1	0	0	0

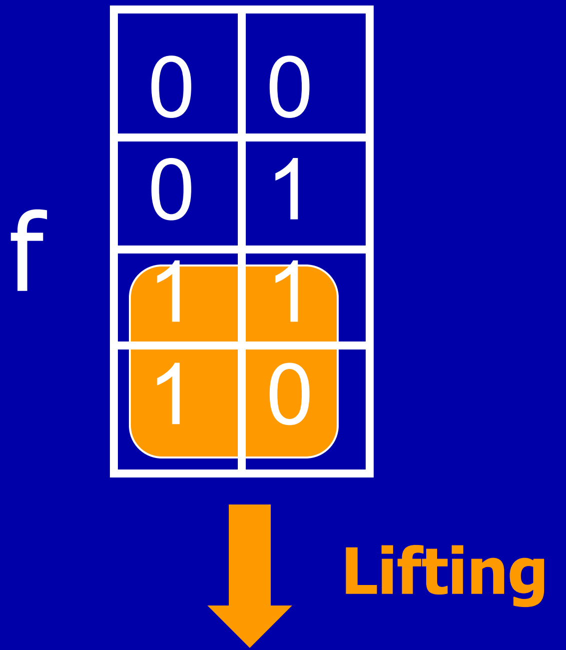
Prime implicants of g

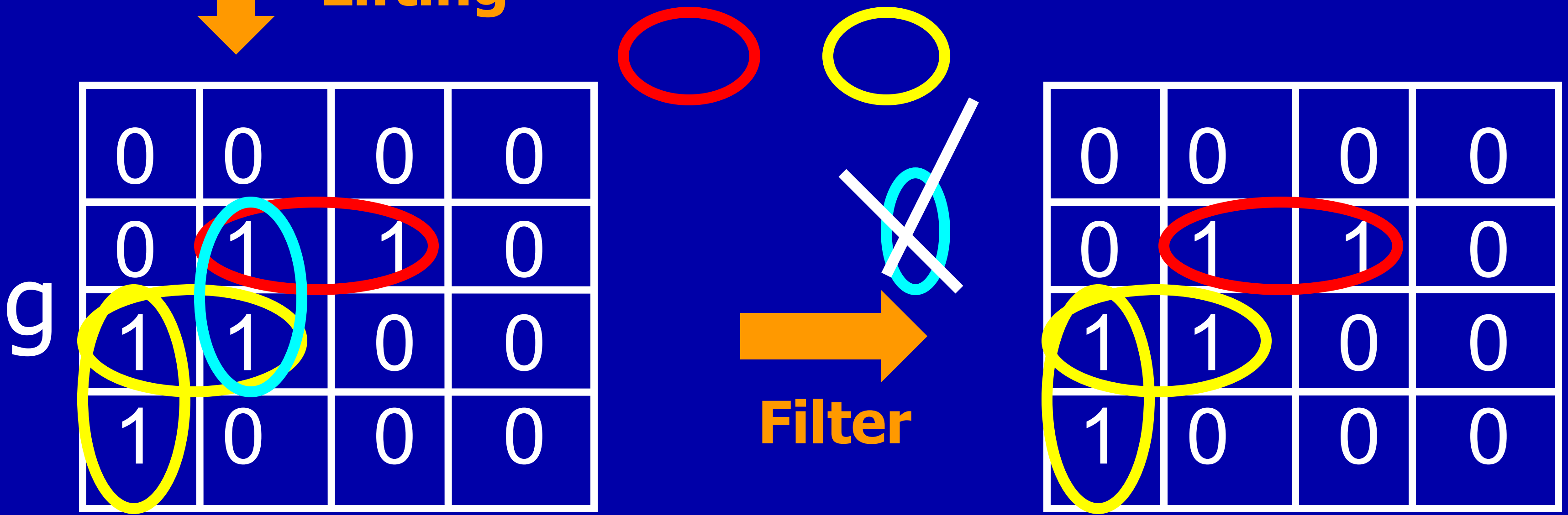
Filtering Primes of g

Transforming Prime(*g*) into DHF-Prime(*f*,*T*):

3 classes of primes of synchronous fct g:

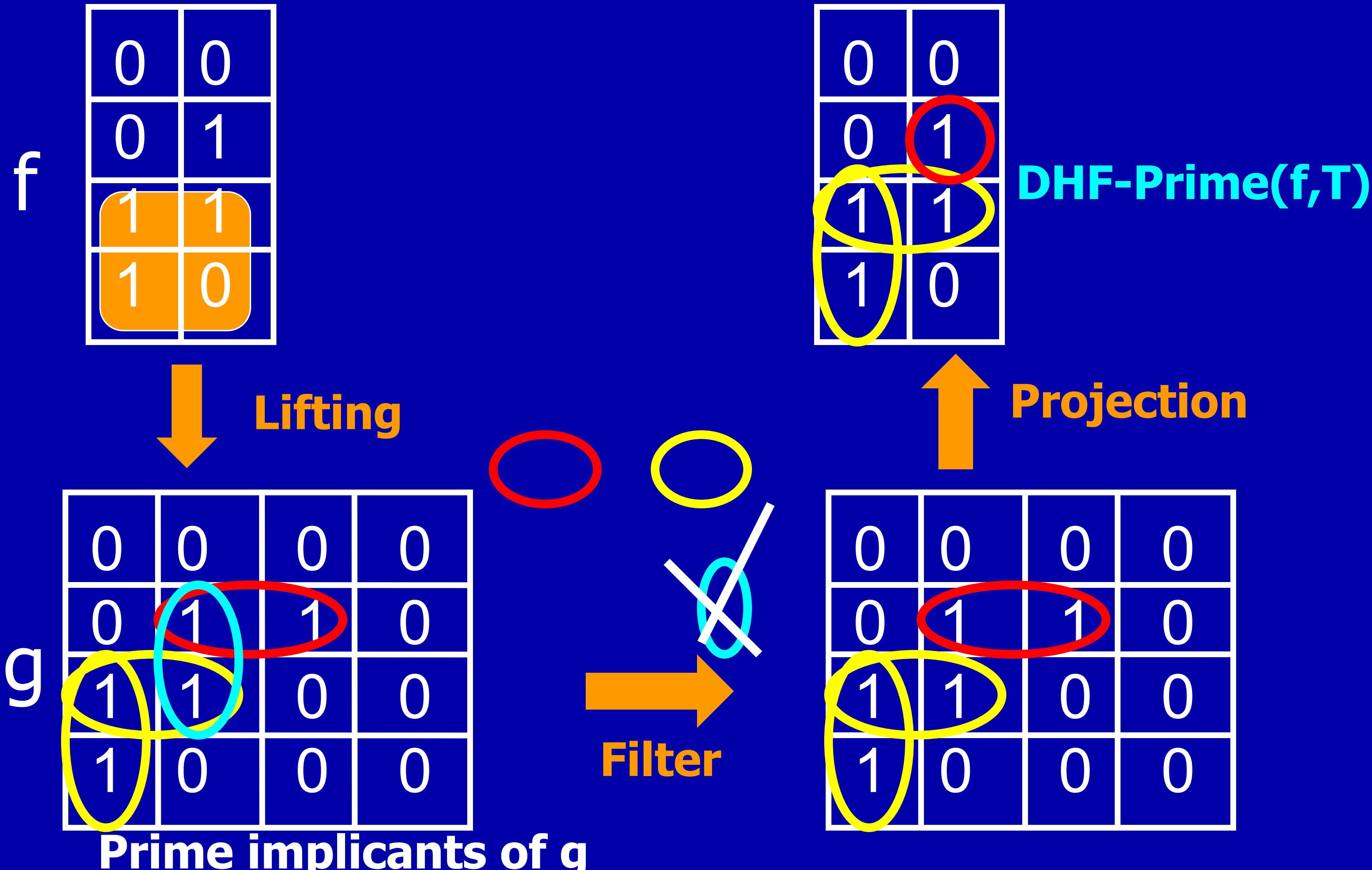
- **1. do not intersect priv-cube (in original domain)**
- **2. intersect legally**
- **3. intersect illegally**





Prime implicants of g

Projection


f
Lifting
g
Filter
Projection
DHF-Prime(f,T)
Prime implicants of g

Formal Characterization of DHF-Prime(*f*,*T*)

$$g(x_1, \ldots, x_n, z_1, \ldots, z_l) = f \bullet \prod_{1 \le i \le l} (\overline{z_i} + \overline{p_i})$$

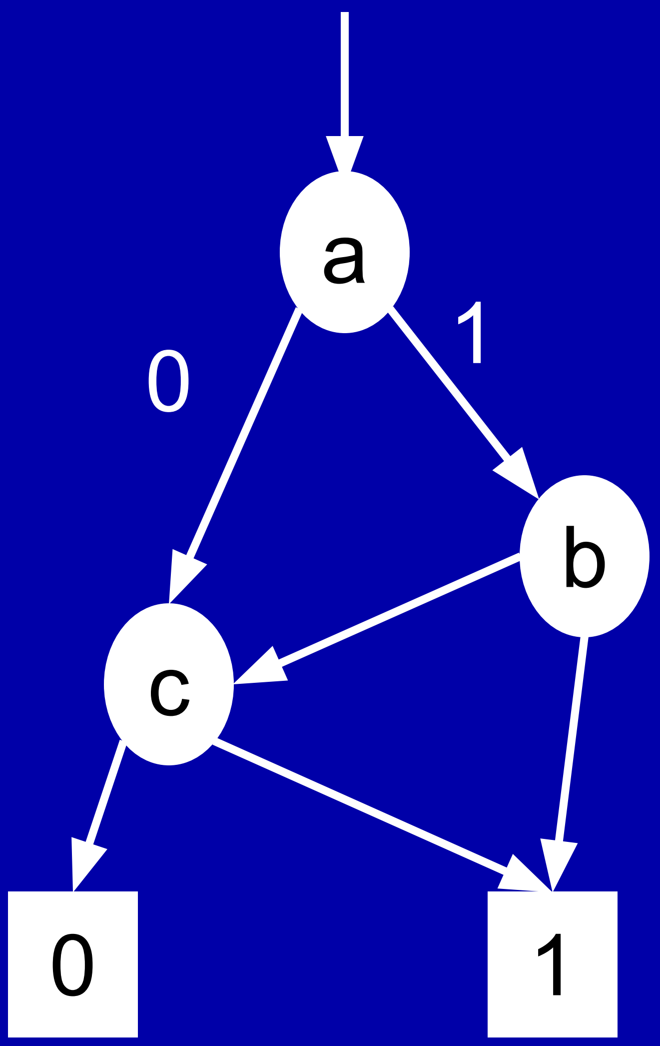
IMPYMIN

- CAD tool for Hazard-Free 2-Level Logic
- Two main ideas:
  - Computes DHF-Primes in higher-dimension space
  - Implicit Method: makes use of BDDs, ZBDDs

What is a BDD ?

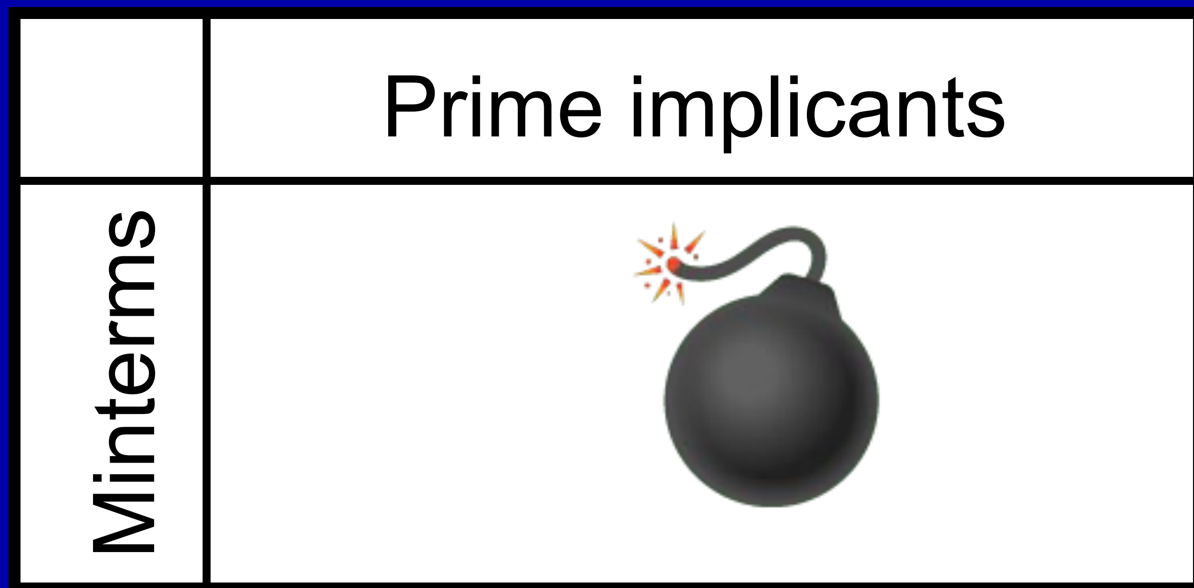
- Compact representation for

  Boolean function

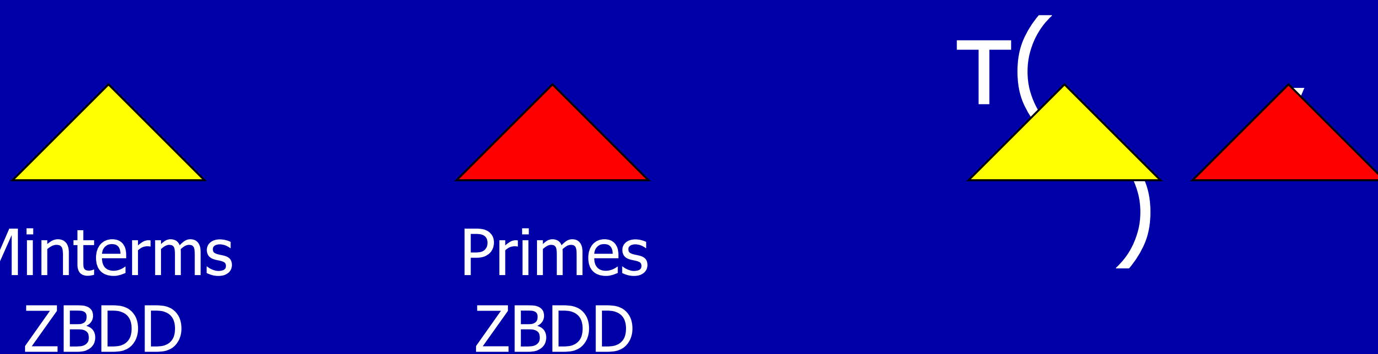


What is implicit logic minimization?

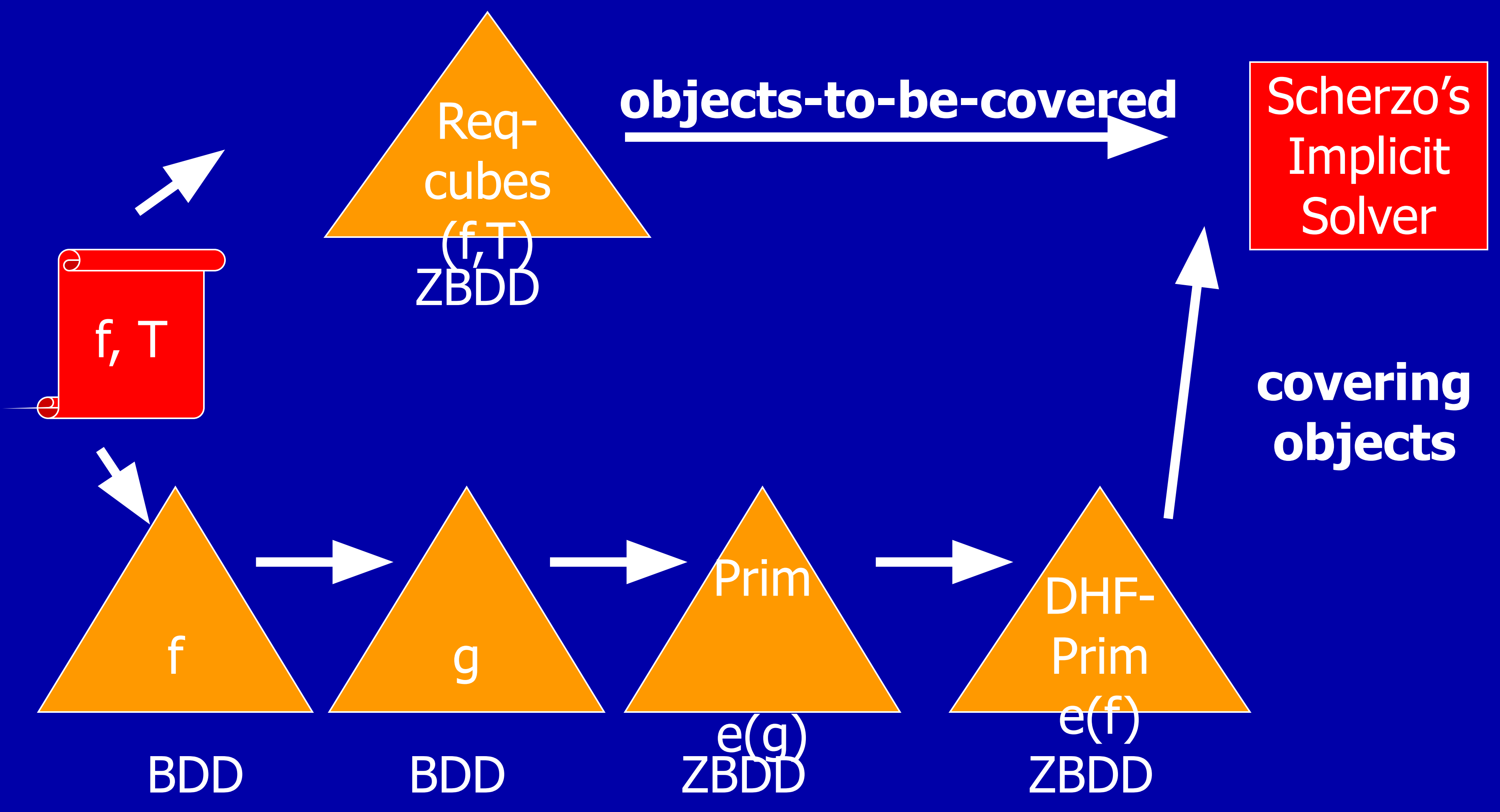
- Classic Quine-McCluskey:



- Scherzo [Coudert] (implicit logic minimization):



IMPYMIN Overview:
Implicit Hazard-free 2-Level Minimizer



Impympin vs. HFMIN: Results

added variables

	I/O	#C	HFMIN	IMPYMIN	#z
cache	20/23	97	impossible	301	**39**
pscsi	16/11	77	1656	105	**23**
sd	18/22	34	172	52	**0**
Stetson1	32/33	60	>72000	813	**9**
Stetson2	18/22	37	151	49	**0**

IMPYMIN: Conclusions

- New idea: incorporate hazard-freedom constraints
  - transformed asynchronous problem into synchronous problem
- Presented implicit minimizer IMPYMIN:
  - significantly outperforms existing minimizers
- Idea may be applicable to other problems, e.g. testing