

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

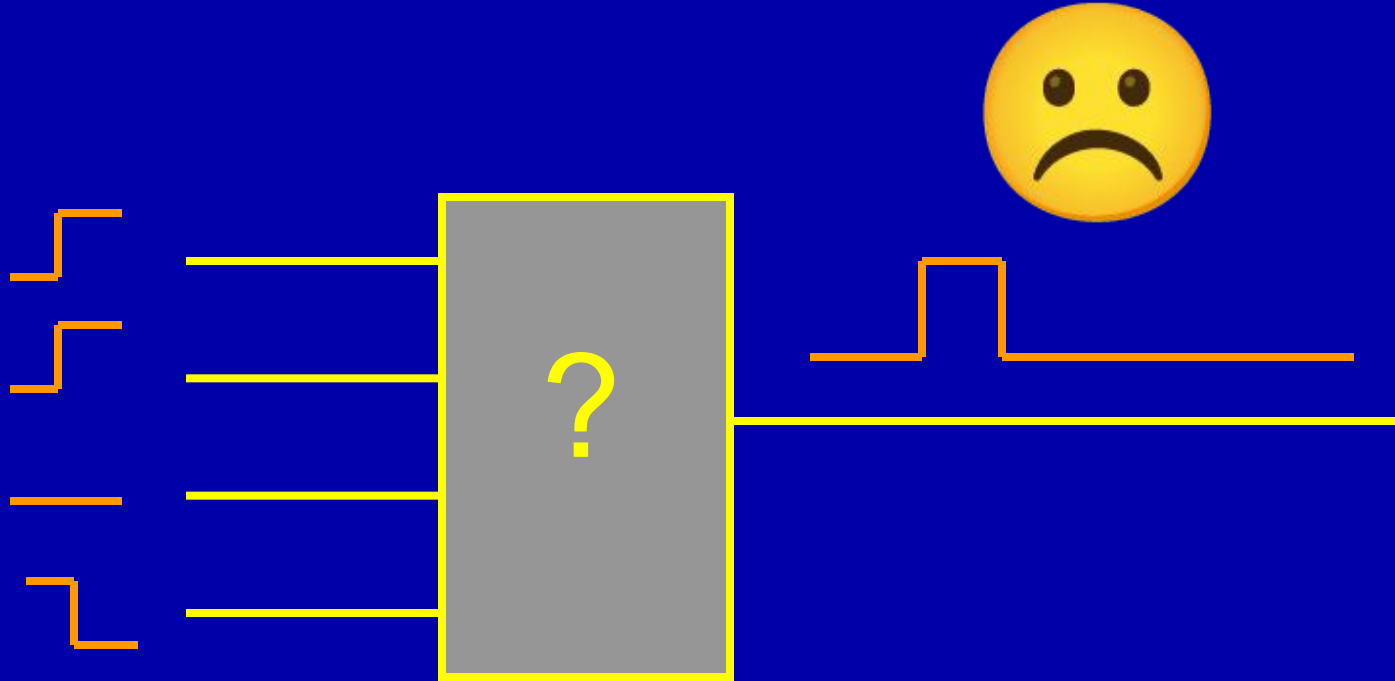
Columbia University, New York, NY

Paper appeared in Async-98

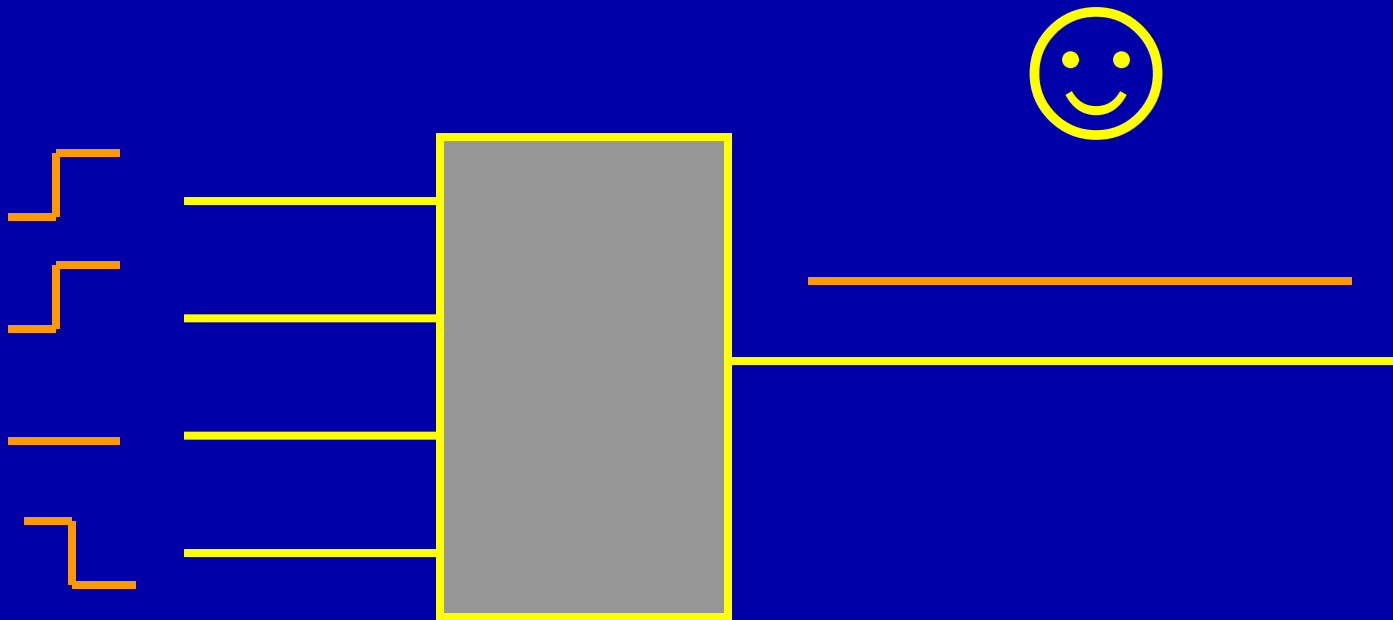
(Best Paper Finalist)

Hazard-Free Logic Minimization

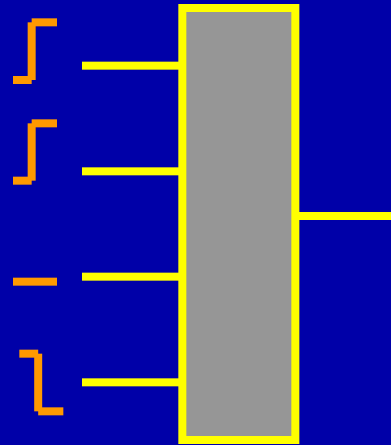
Given: Boolean function and multi-input change








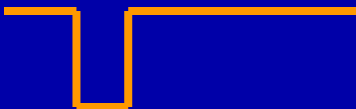




Hazard-Free Logic Minimization

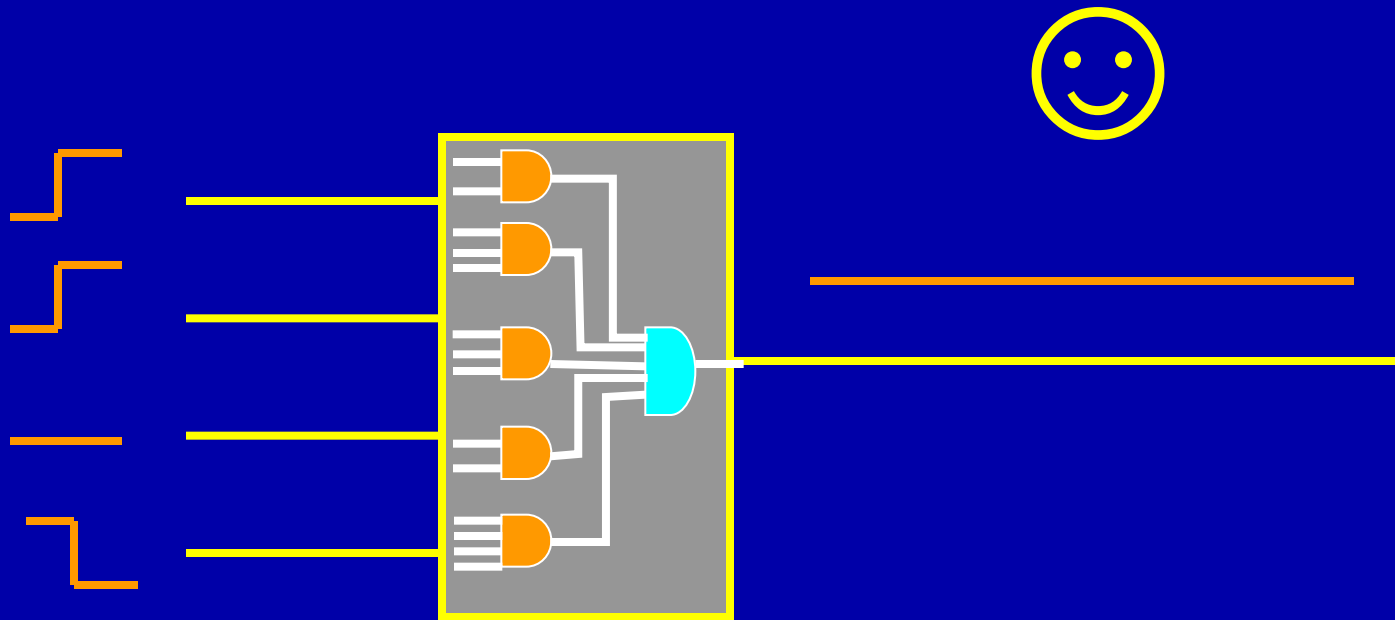


Hazard-Free Logic Minimization



$f(A)$ <input type="checkbox"/> $f(B)$		
0 <input type="checkbox"/> 0		
0 <input type="checkbox"/> 1		
1 <input type="checkbox"/> 1		
1 <input type="checkbox"/> 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:

A 4x2 Karnaugh Map with the following values:

0	0
0	1
1	1
1	0

The map shows two prime implicants circled: a yellow oval around the top-right 2x2 area (cells (1,1), (1,2), (2,1), (2,2)) and a cyan oval around the bottom-left 2x2 area (cells (2,1), (2,2), (3,1), (3,2)).

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 glitch-free
- Required cubes = sets of minterms
- DHF-Prime implicants =
maximal implicants 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
must be covered

Hazard-Freedom Conditions: 1 -> 0 transition

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

The diagram shows a 4x4 Karnaugh map with the following values:

- Row 1: 0, 0, 0, 0
- Row 2: 0, 1, 1, 0
- Row 3: 0, 1, 0, 0
- Row 4: 0, 1, 0, 0

Yellow circles highlight the 1s at positions (1,1), (1,2), (2,1), and (3,1). A red arrow points from the 1 at (1,1) to the 0 at (2,2), indicating a potential hazard during a 1 to 0 transition.

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

The image shows a 4x4 Karnaugh map with the following values:

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

Yellow circles highlight the 1s in the second column (rows 2, 3, 4) and the 1s in the second and third rows (columns 1, 2). A red arrow points from the 1 in the second row, second column to the 0 in the second row, third column, illustrating a 1-to-0 transition.

Hazard-Freedom Conditions: 1 -> 0 transition

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

The diagram shows a 4x4 grid representing a Karnaugh map. The values in the cells are as follows:

- Row 1: 0, 0, 0, 0
- Row 2: 0, 1, 1, 0
- Row 3: 0, 1, 0, 0
- Row 4: 0, 1, 0, 0

Annotations:

- A yellow oval encircles the 1s in the second column (rows 2 and 3).
- A red arrow points from the 1 in the second row, second column to the 0 in the third row, third column.

Hazard-Freedom Conditions: 1 -> 0 transition

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



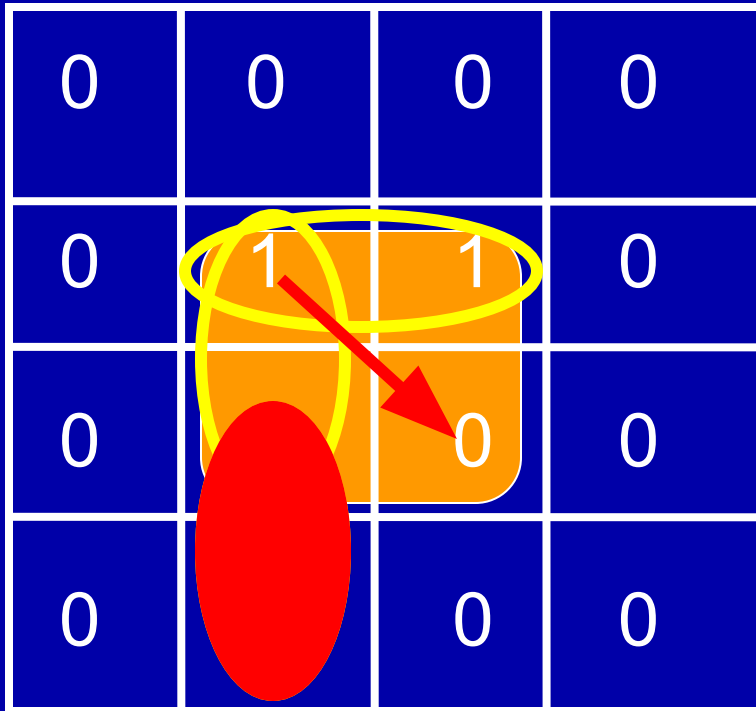
Hazard-Freedom Conditions: 1 -> 0 transition

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

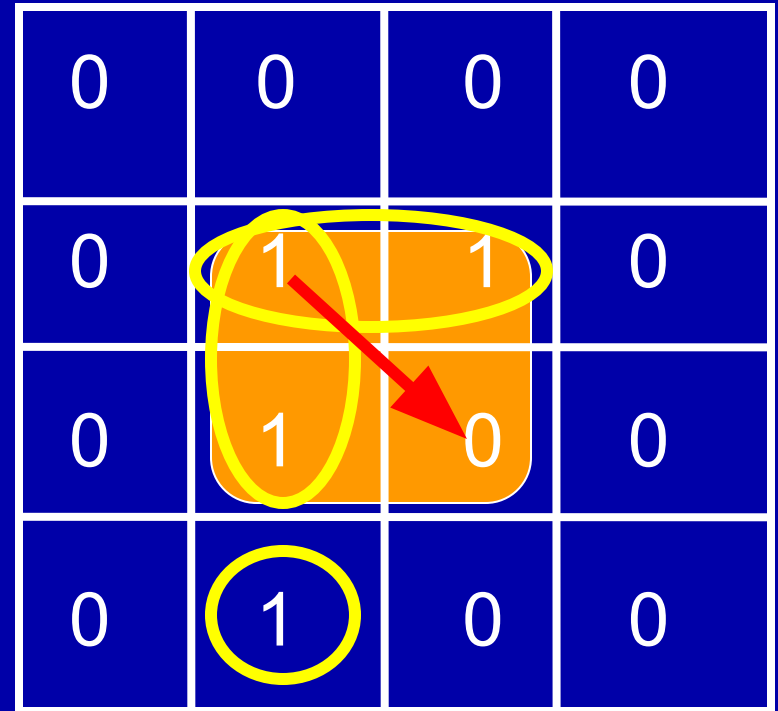


illegal intersection

Hazard-Freedom Conditions: 1 -> 0 transition



illegal intersection



No illegal
intersection
of privileged cube

Dynamic-Hazard-Free Prime Implicants

0	0
0	1
1	1
1	0

Prime

0	0
0	0
1	1
1	0

**NO DHF-Prime
illegal
intersection**

0	0
0	1
1	1
1	0

DHF-Prime

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

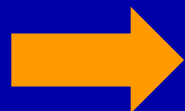
- Classic (Quine-McCluskey):

<On-set minterms, Prime implicants>

- Hazard-Free:

<Required cubes, DHF-Prime implicants>

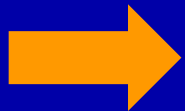
- Given: Boolean function & set of “multi-input” changes
- Find: *min-cost* 2-level implementation guaranteed to be glitch-free
- Required cubes = sets of minterms
- DHF-Prime implicants =
maximal implicants that do not intersect privileged cubes illegally



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):

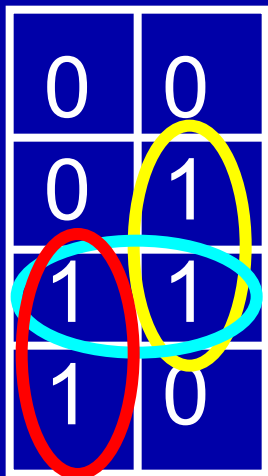
<On-set minterms, **Prime implicants**>

Hazard-Free:

<Required cubes, **DHF-Prime implicants**>

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

0	0
0	1
1	1
1	0

A 2x2 Karnaugh map with cells containing 0, 0, 1, 1, 1, 0. Three prime implicants are circled: a red circle around the two 1s in the left column, a yellow circle around the two 1s in the right column, and a cyan circle around the two 1s in the middle row.

Primes

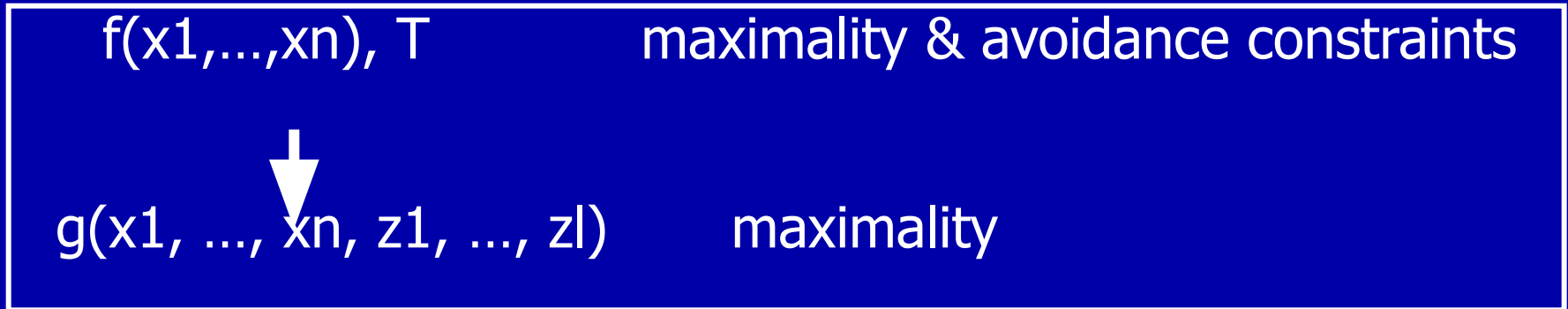
0	0
0	1
1	1
1	0

A 2x2 Karnaugh map with cells containing 0, 0, 1, 1, 1, 0. Three prime implicants are circled: a red circle around the two 1s in the left column, a yellow circle around the two 1s in the right column, and a cyan circle around the two 1s in the middle row. The two 1s in the bottom row are highlighted with an orange square. Two white arrows point from the 1 in the bottom-left cell to the 1 in the bottom-right cell and from the 1 in the middle-right cell to the 0 in the bottom-right cell.

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:** Unify constraints by “lifting” the problem into a higher-dimensional space:



Auxiliary Synchronous Function g

f

0	0
0	1
1	1
1	0



Add one new dimension per privileged cube

g

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

$z=0$ $z=1$

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

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

Expansion in z-dimension
guarantees avoidance
of priv-cube in original domain

Prime Implicants 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

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 $\text{Primes}(g)$
- after filtering step, retrieve $\text{dhf-primes}(f)$

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

f

0	0
0	1
1	1
1	0

Lifting

g

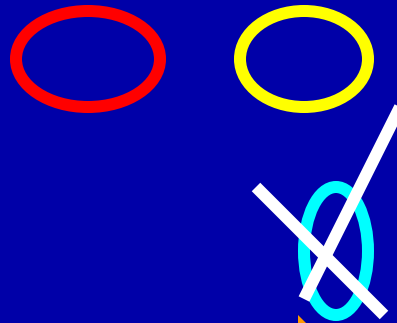
0	0	0	0
0	1	1	0
1	1	0	0
1	0	0	0

Prime implicants 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



Filter

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

Projection

f

0	0
0	1
1	1
1	0

Lifting

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

Prime implicants of g

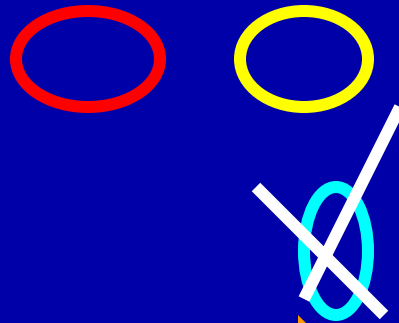
0	0
0	1
1	1
1	0

DHF-Prime(f,T)

Projection

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

Filter



Formal Characterization of DHF-Prime(f, T)

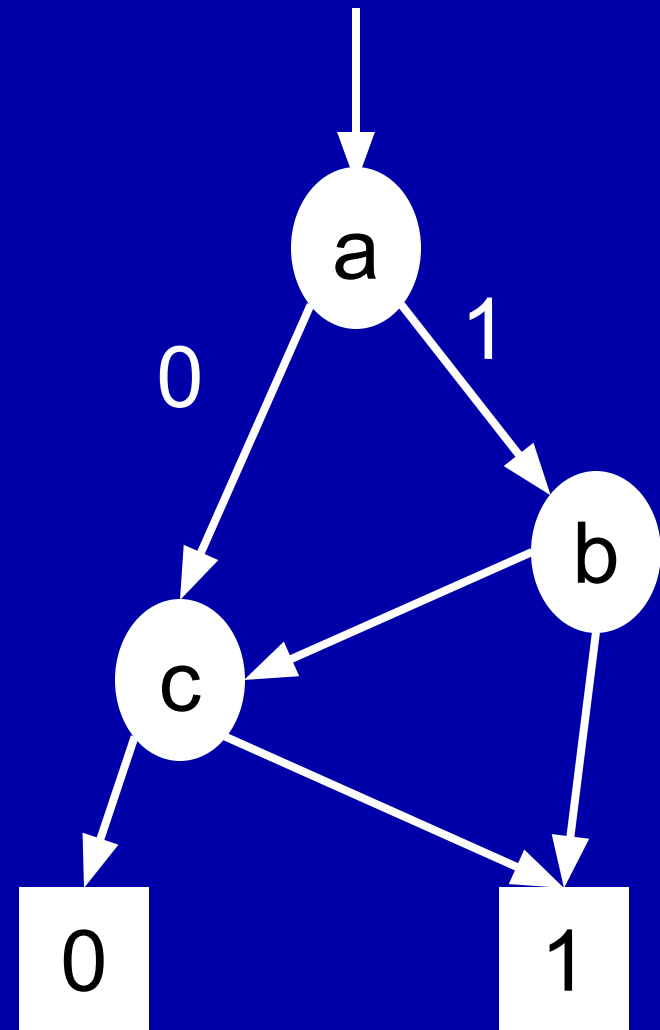
$$g(x_1, \mathbb{N}, \dots, x_n, z_1, \mathbb{N}, \dots, z_l) = f \bullet \prod_{1 \leq i \leq l} (\overline{z_i} + \overline{p_i})$$

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

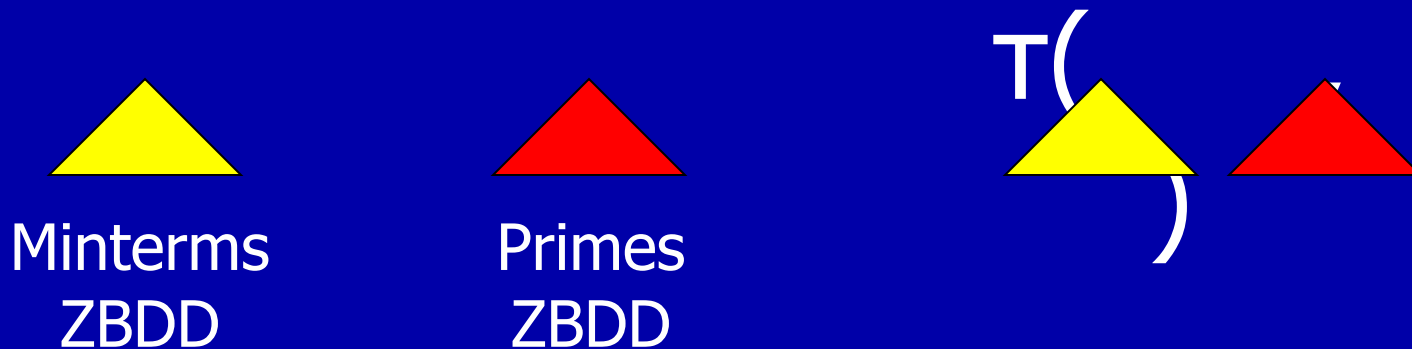


What is implicit logic minimization?

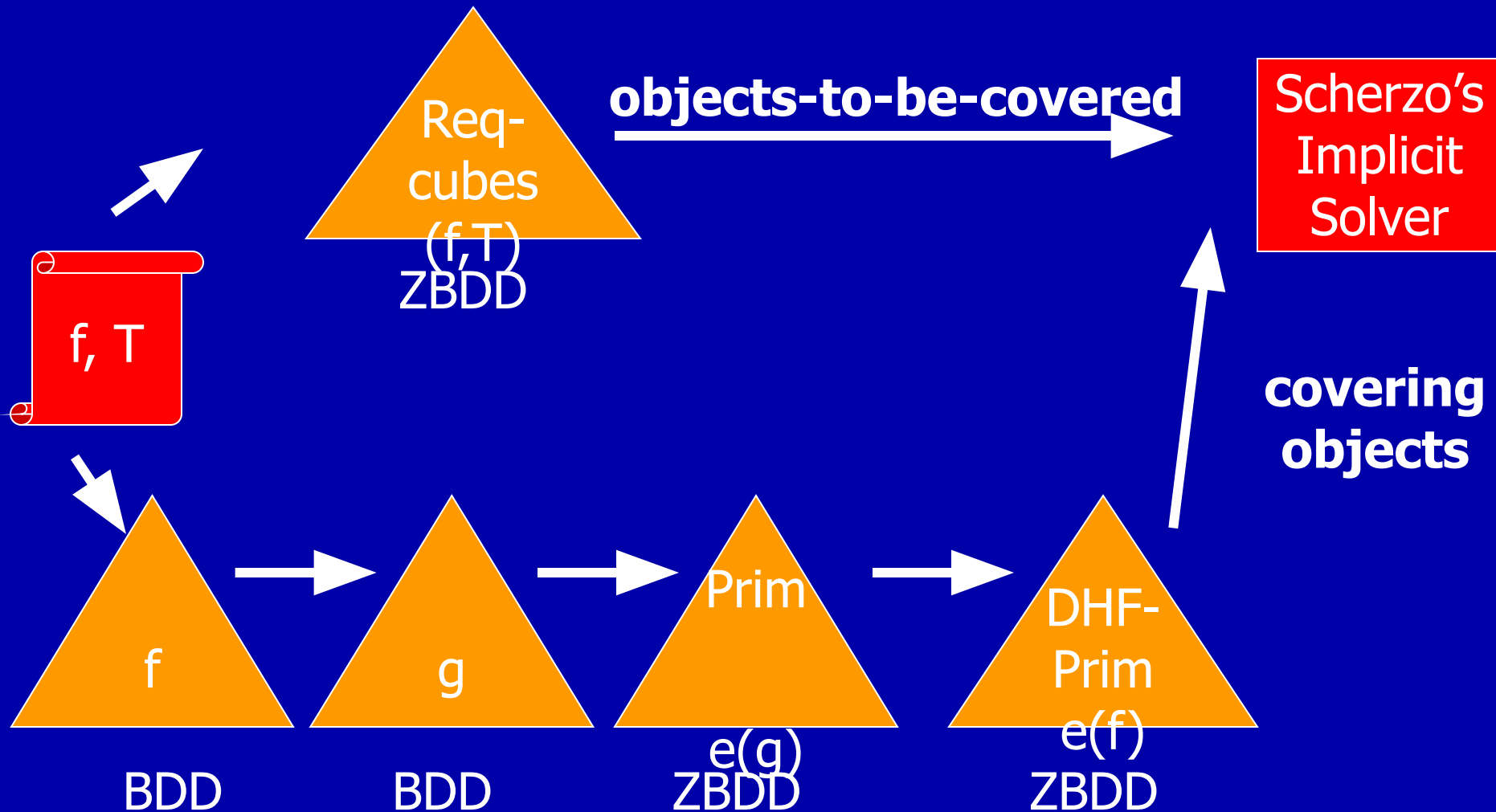
- Classic Quine-McCluskey:

	Prime implicants
Minterms	

- Scherzo [Coudert] (implicit logic minimization):



IMPYMIN Overview: Implicit Hazard-free 2-Level Minimizer



Impymin 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