Symbolic Hazard-Free Minimization and Encoding of Asynchronous Finite State Machines

Robert Fuhrer
Columbia University

Bill Lin
IMEC Laboratory

Steven Nowick
Columbia University

Contribution

State assignment method yielding hazard-free, low-cost 2-level implementations for asynchronous state machines.

Outline

- Introduction
- Background
- Optimal Encoding for Asynchronous FSM's
- Results
- Conclusions/Future Work

Introduction

Optimal State Encoding: Find encoding yielding optimal implementation of given FSM, according to cost metric.

Well-studied: DeMicheli [85], Devadas [88], Saldanha [91]

Dimensions:

- Area vs. speed vs. power
- 2-level vs. multi-level logic
- Exact vs. heuristic techniques
- input encoding vs. output encoding ...

Our focus: # of product terms in 2-level logic;
exact & heuristic techniques based on input encoding.

Requirements: hazard-free, critical race-free implementation.

Introduction (cont.)

Optimal State Encoding for Asynchronous FSM's

Several heuristic encoding techniques exist:

Tracey [66], Tan [67], Saucier [72], Fisher [93], Lam [94]

None provides systematic optimal state assignment.

Our research contribution:

- First such work for asynchronous FSM's
  - Deal directly with MIC hazards
  - Upper bound on overall logic
  - *Exactly optimal* result for output logic
- Leverage off of existing synchronous work [KISS]
  - Use input encoding formulation

Asynchronous Finite State Machines

Potential Benefits:

- Performance (utilize difference between avg & worst-case)
- Avoidance of clock skew
- Low power

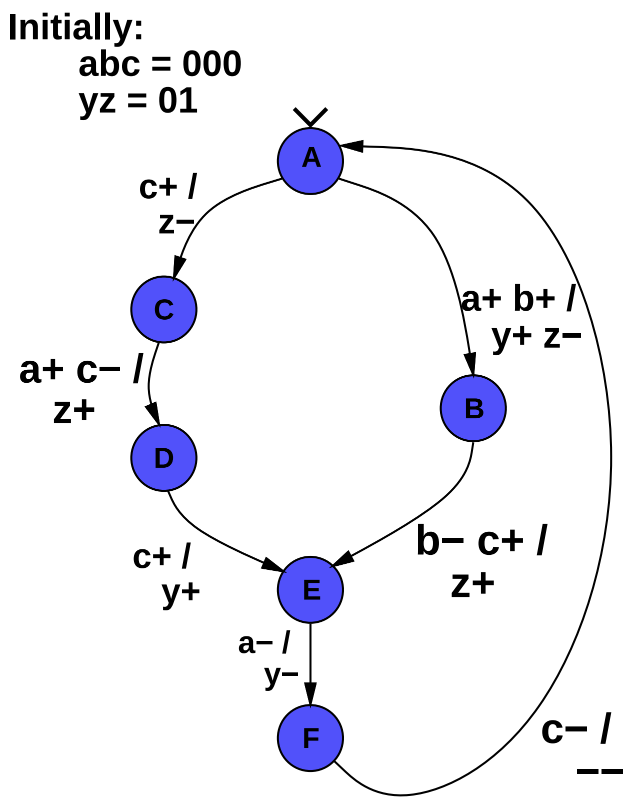
Recent successes:

- HP Stetson (Marshall et al. [94])
  - Low-power infrared communications chip
- Nowick et al. [93]
  - Cache controller
- Yun et al. [95]
  - AMD SCSI controller

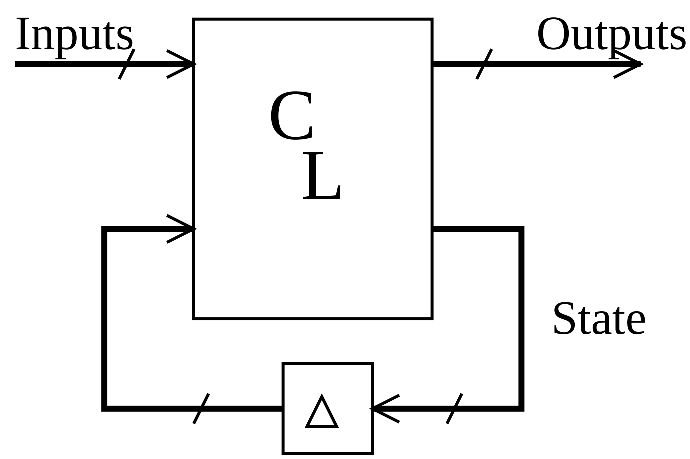
Recent work: [Nowick 91], [Yun 92], [Davis 93]

Background: Asynchronous FSM's

Multiple Input Change (MIC) Machines: Burst-Mode



- Only *specified* transitions
- Inputs arrive in any order



Huffman Machine

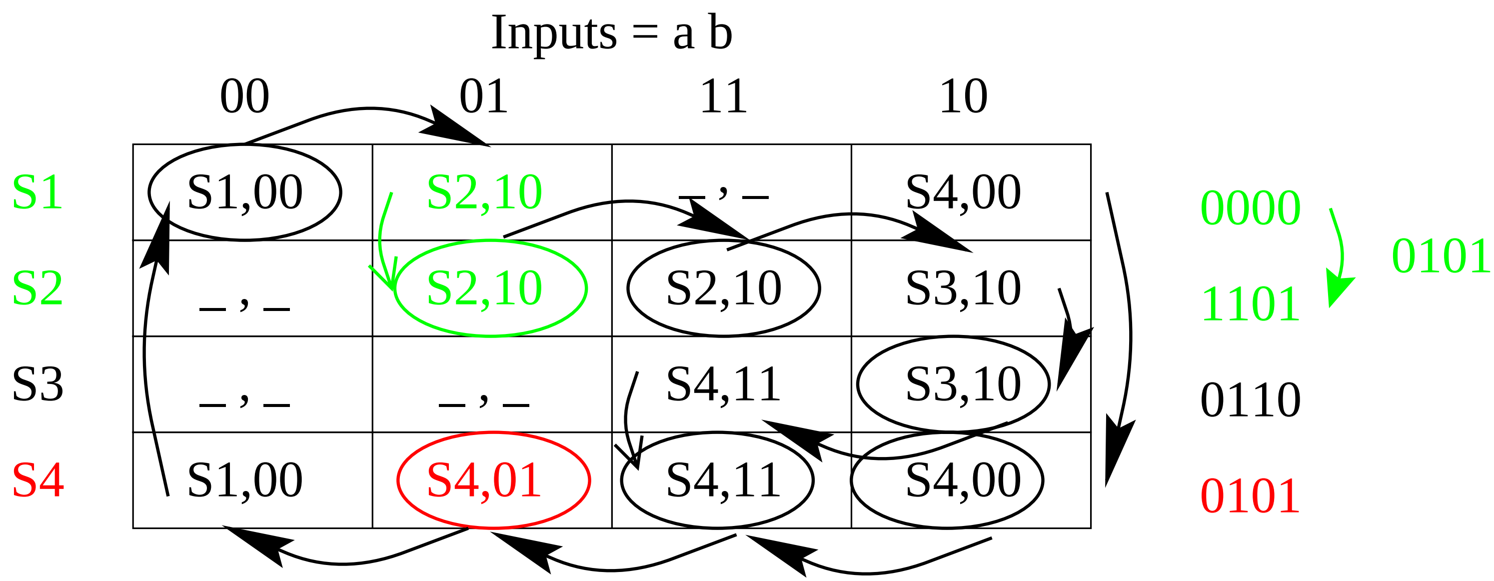
- *Generalized fundamental-mode* of operation.
- *Hazard-free* logic.

Background: Asynchronous FSM's (cont.)

Two basic issues confronting asynchronous FSM synthesis:

1. Critical Races

Can cause FSM to settle in wrong state [Tracey 66].

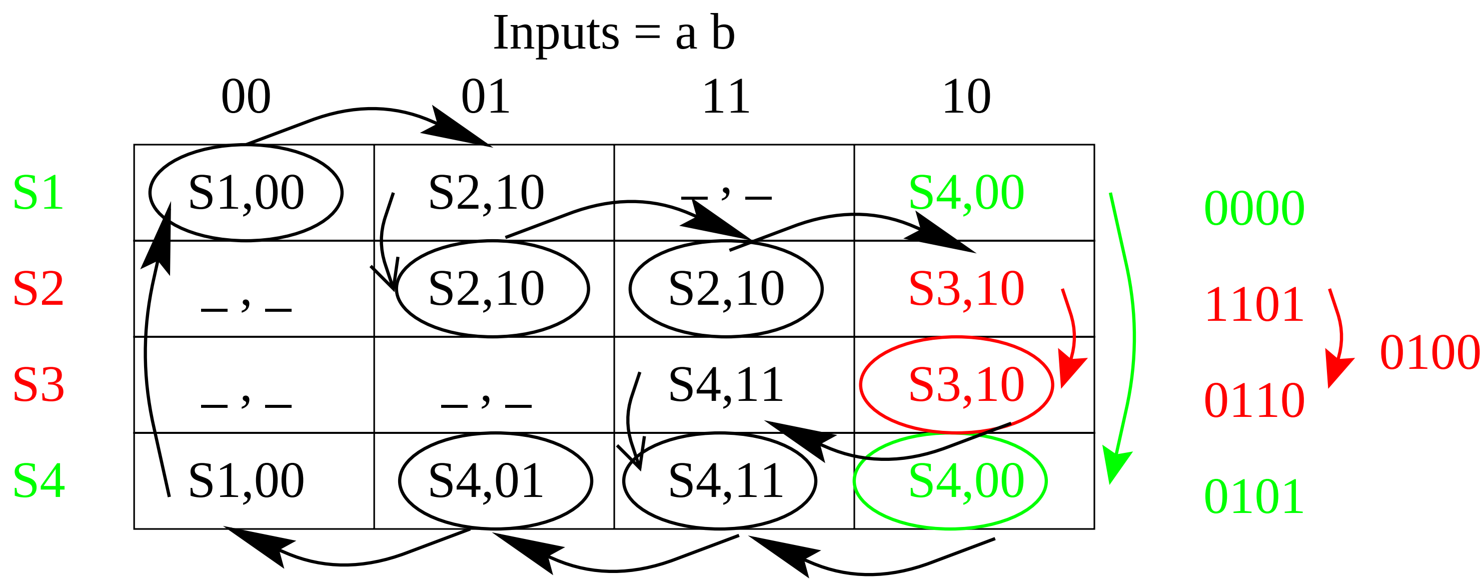


Cause: interference in given input column between:

I. Unstable transition and stable state ($\{s_1, s_2;\ \ s_4\}$)

Background: Asynchronous FSM's (cont.)

1. Critical Races (cont.)



Cause: interference in given input column between:

I. Unstable transition and stable state $(\{s_1, s_2;\ \ s_4\})$

II. Unstable transition and unstable transition $(\{s_1, s_4;\ \ s_2, s_3\})$

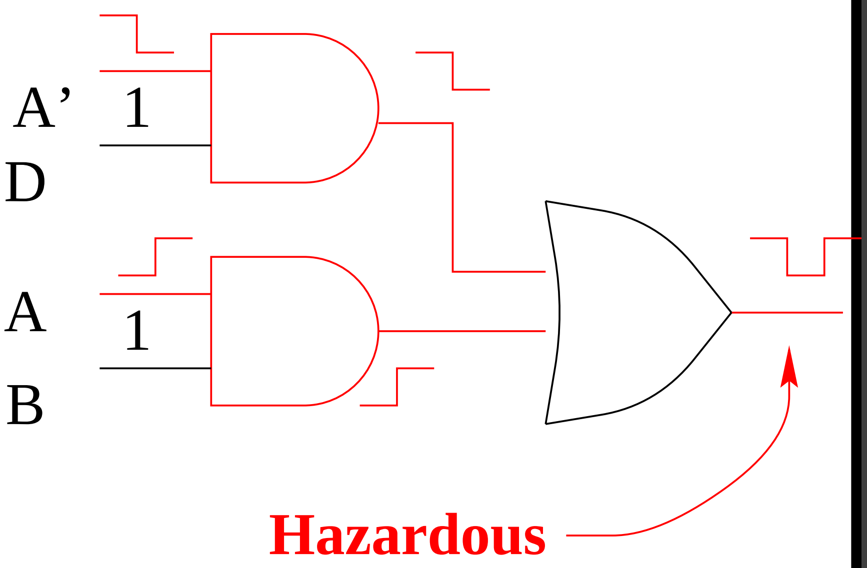
Solution: judicious state encoding.

Background: Asynchronous FSM's (cont.)

2. Combinational Hazards

Type I: MIC Static Hazards

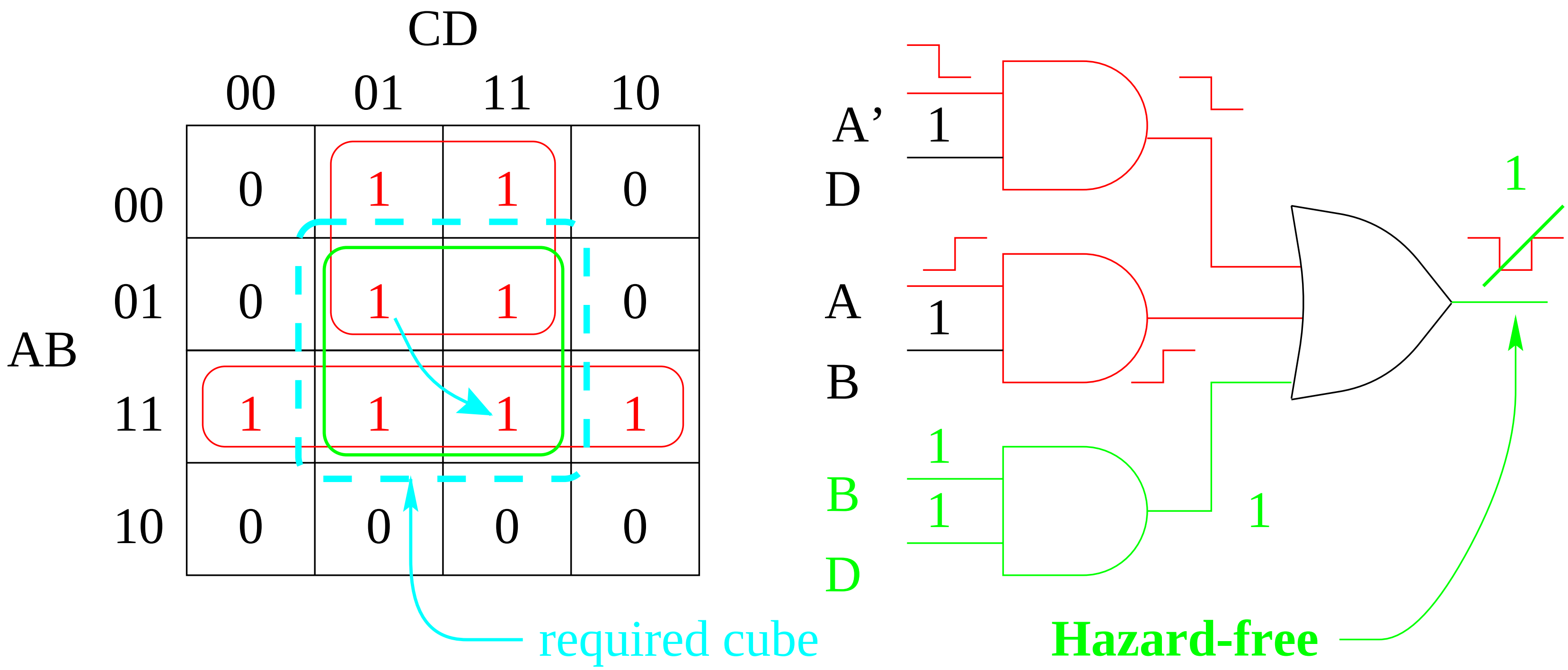
AB \ CD	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	1	1	1
10	0	0	0	0



Background: Asynchronous FSM's (cont.)

2. Combinational Hazards

Type I: MIC Static Hazards

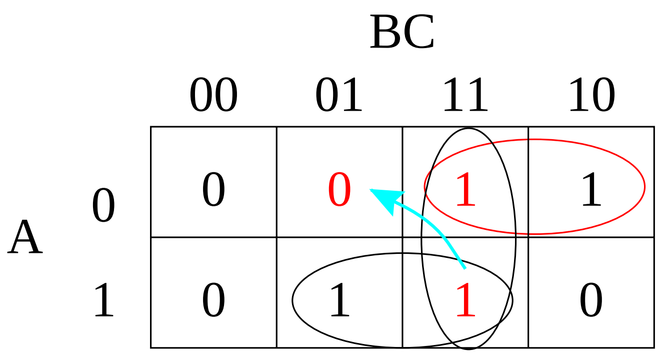


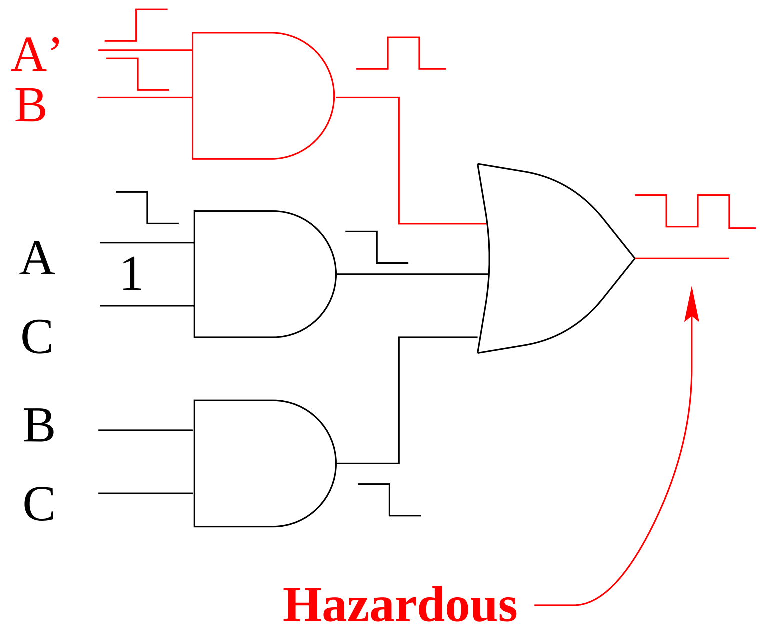
Solution: Some product term must cover each **required cube**.

Background: Asynchronous FSM's (cont.)

2. Combinational Hazards (cont.)

Type II: MIC Dynamic Hazards

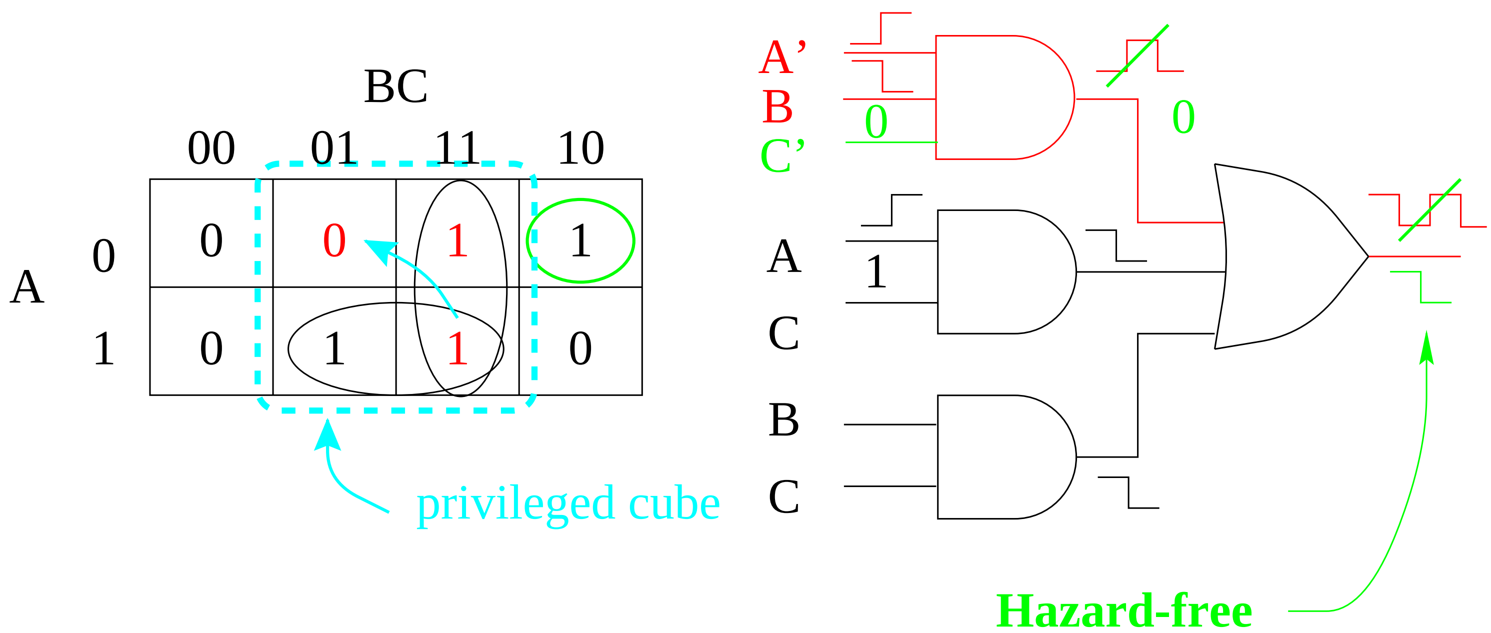




Background: Asynchronous FSM's (cont.)

2. Combinational Hazards (cont.)

Type II: MIC Dynamic Hazards



Solution: Implicants must not **illegally intersect** the **privileged cube** of any dynamic transition.

Use only **dynamic-hazard-free** (DHF) prime implicants.

Background: Asynchronous FSM's (cont.)

Exact Hazard-Free 2-Level Logic Minimization [Nowick 91]

Given: *incompletely specified* Boolean function +
set of **specified** input transitions:

1. Generate required cubes
2. Generate DHF prime implicants
3. Solve *unate covering problem*

Covered objects: required cubes
Covering objects: DHF prime implicants

Limitations:

- BVI only
- Single-output only

Background: KISS Optimal State Encoding

Step # 1: Symbolic Logic Minimization

Goal: Find optimal symbolic cover.

input I

	0	1
s_1	$s_3, 1$	$s_1, 0$
s_2	$s_2, 0$	$s_1, 0$
s_3	$s_3, 1$	$s_1, 0$

Minimal Symbolic Cover

	input	present	next	output
p_1:	0	s_1, s_3	s_3	1
p_2:	0	s_2	s_2	0
p_3:	1	s_1, s_2, s_3	s_1	0

Key Idea: perform **symbolic** minimization using multi-valued input (**mvi**) minimization techniques [Sasao 84].

Caveat: Only an *approximation* for next-state logic.

Background: KISS Optimal State Encoding (cont.)

Can instantiate symbolic cover with an encoding.

Problem: Instantiated cover **incorrect** for certain encodings.

Solution: Step # 2: Encoding Constraints

Constraints allow direct instantiation of minimal symbolic cover.

Derive **face embedding constraints**: set of $N \rightarrow 1$ dichotomies.

Minimal Symbolic Cover

	input	present	next	output
p_1:	0	s_1, s_3	s_3	1
p_2:	0	s_2	s_2	0
p_3:	1	s_1, s_2, s_3	s_1	0

Example: product p_1 yields *seed dichotomy* $\{s_1, s_3;\ \ s_2\}$.

Background: KISS Optimal State Encoding (cont.)

Step # 3: Solve Constraints

Solution *always* exists.

Various exact & heuristic solution methods:

kiss	[DeMicheli 85],	dichot	[Saldanha 91],
nova	[Villa 89],	duet	[Cieselski 91]

Key Result:

Exact constraint solution yields *minimum cardinality output cover* (if outputs minimized separately).

However: Approximate solution for next-state logic.

Optimal Encoding for Asynchronous FSM's

Step # 1: Symbolic Hazard-Free 2-Level Logic Minimization

Unlike KISS: need *hazard-free* symbolic cover.

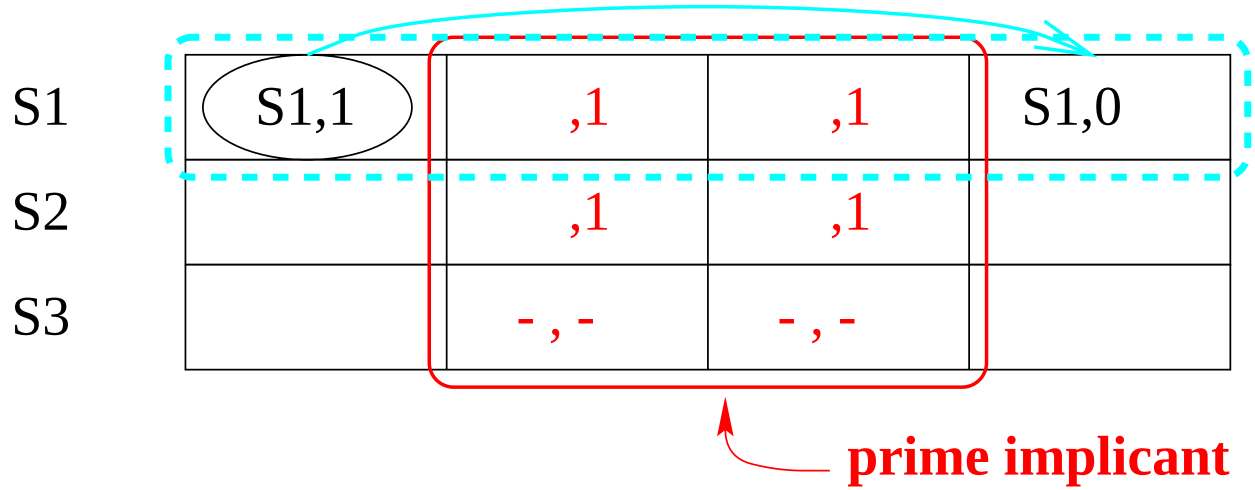
Generalize 2-level hazard-free **bvi** algorithm to **mvi** functions.

- Defined **mvi multiple-input transitions**
- Defined *static* and *dynamic* output transitions
- Extended notion of privileged cubes, illegal intersections, etc.
- Generalized hazard-free conditions for mvi functions.

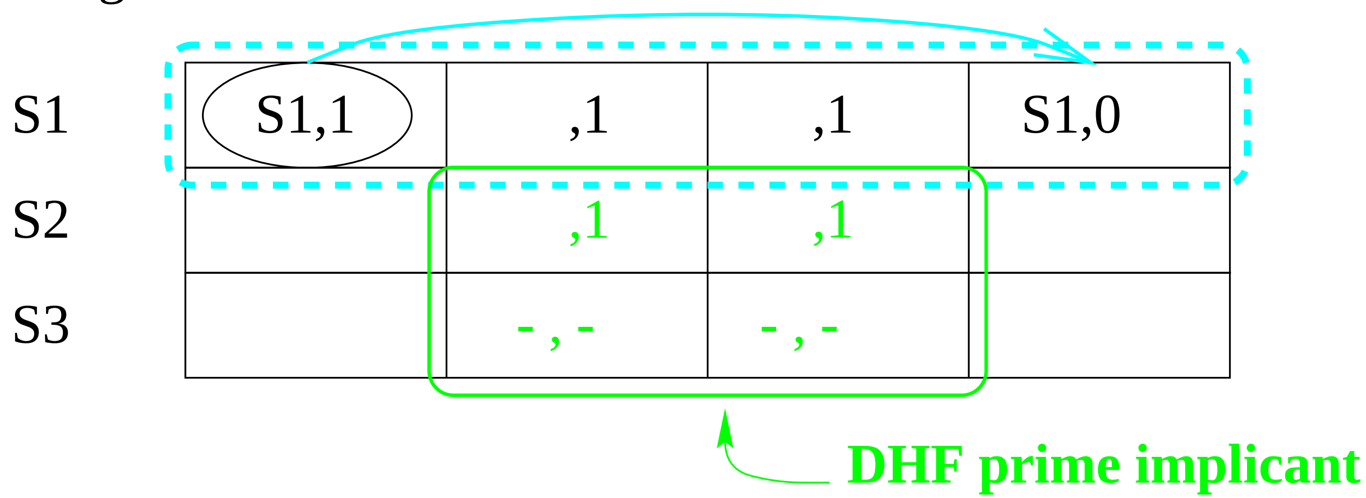
Extended previous algorithm to **multiple-output** minimization.

Step # 1: Hazard-Free Symbolic Logic Minimization

Illegal Intersection:



No Illegal Intersection:



Optimal Encoding for Asynchronous FSM's (cont.)

Step # 2: Encoding Constraints

KISS face embedding constraints *inadequate.*

Asynchronous FSM defines OFF-set **during state transitions**.

Inputs = a b

	00	01	11	10	
S1	S1,00	S2,10	– , –	S4,00	0000
S2	– , –	S2,10	S2,10	S3,10	1101
S3	– , –	– , –	S4,11	S3,10	0110
S4	S1,00	S4,01	S4,11	S4,00	0101

Goal: Prevent instantiated implicants from hitting OFF-set.

Optimal Encoding for Asynchronous FSM's (cont.)

Step # 2: Encoding Constraints (cont.)

KISS face embedding constraints *inadequate*.

Asynchronous FSM defines OFF-set **during state transitions**.

Inputs = a b

State	Code	00	01	11	10
S1	0000	, 0	, 1	, -	S4, 0
S2	1101	, -	, 1	, 1	, 1
S3	0110	, -	, -	, 1	, 1
S4	0101	, 0	, 0	, 1	S4, 0

0100

Goal: Prevent instantiated implicants from hitting OFF-set.

Example: column `10` $\Rightarrow$ dichotomy $\{s_2, s_3; \; s_1, s_4\}$.

Optimal Encoding for Asynchronous FSM's (cont.)

Step # 2: Encoding Constraints (cont.)

Observation:

- **Critical race-free** dichotomies: $2 \to 2$, $2 \to 1$
- **Face embedding** dichotomies: $N \to 1$

Asynchronous optimality dichotomies: $N \to 2$, $N \to 1$.

Subsume both KISS optimality and critical race-free constraints.

Optimal Encoding for Asynchronous FSM's (cont.)

Step # 3: Constraint Solution

Exact solution: used *dichot* [Saldanha 91].

Observation: More constraints than in synchronous case.

Increased code length $\Rightarrow$ increased logic complexity.

Heuristic solution: Fixed code length via **simulated annealing**.

Idea: Satisfy maximum # of constraints possible.

Successfully used in synchronous methods (e.g. Villa, Lin).

Optimal Encoding for Asynchronous FSM's (cont.)

Step # 3: Constraint Solution (cont.)

Asynchronous Correctness Requirement:

Critical race-free constraints must be satisfied.

Solution: Partition constraints into 2 classes:

1. **compulsory** constraints (for **correctness**)
2. **optional** constraints (for **optimality**)

Assign sufficiently large weights to compulsory constraints to ensure satisfaction.

Theoretical Results

1. **Overall logic:**

   - Unlike KISS: instantiated cover may be incorrect

     $\Rightarrow$ need added cubes to avoid next-state hazards
   - Cardinality $|\tilde{\mathcal{C}}| = \mathcal{O}(|\mathcal{C}| + u)$

     $u = \#$ of unstable state transitions
   - **Upper bound** on logic complexity, due to input encoding

2. **Output logic: exactly optimal** over all CRF codes

   if outputs minimized separately

   Important for burst-mode, where input/output latency determines operating speed.

Experimental Results

Up to 17% improvement in **overall logic** with no increase in code length, using heuristic constraint satisfaction.

DESIGN	I/S/O	*heuristic* bits	*heuristic* cubes	*exact* bits	*exact* cubes	*base-crf* bits	*base-crf* cubes
sbuf-read-ctl	3/3/3	2	7	3	9	2	8
pscsi-ircv	4/4/3	2	9	4	12	2	10
pscsi-trcv	4/4/3	3	9	4	13	2	11
sscsi-trcv-csm	5/3/4	2	12	3	12	2	12
pscsi-trcv-bm	4/4/4	2	12	4	15	2	14
rf-control	6/6/5	3	13	6	15	3	15
sscsi-tsend-csm	5/4/4	2	14	5	15	2	14
it-control	5/5/7	3	15	6	15	3	15
pe-send-ifc	5/5/3	3	18	7	27	3	21
pscsi-isend	4/6/3	3	17	7	23	3	19
sscsi-trcv-bm	5/4/4	2	18	5	24	2	18
sscsi-tsend-bm	5/5/4	3	17	6	20	3	18
sscsi-isend-bm	5/4/4	2	21	5	22	2	24
sd-control	8/13/12	5	29	10	34	4	35
stetson-p2	8/13/12	4	31	10	37	4	36
stetson-p1	13/12/14	4	53	19	-	4	55

Conclusions

- First systematic optimal encoding method for asynchronous FSM's
- Symbolic hazard-free 2-level logic minimization
- Extended encoding constraints for asynchronous FSM's
- Constraint solution:
  - Compulsory vs. optional constraints
- Exactly optimal output logic
- Significant improvement on industrial examples

Future Work

- Improve results of annealing algorithm
- Extend to output encoding formulation
- Generalized symbolic transitions
  - larger machine class: extended burst-mode
  - relax operating constraints