# Application Specified Soft Error Failure Rate Analysis using Sequential Equivalence Checking Techniques

Tun Li, Dan Zhu, Sikun Li, Yang Guo
School of Computer
National University of Defense Technology
Speaker: Tun Li

国防科学技术大学
National University of Defense Technology

## Outline

## Introduction

- As technology scales and node capacitances decrease
  - Single Event Upsets (SEUs) induced by particle strikes from environmental radiation are increased
  - Process variations or aging effects which cause malfunctions under certain conditions are increased
  - Fault $\rightarrow$ Soft error $\rightarrow$ Failure

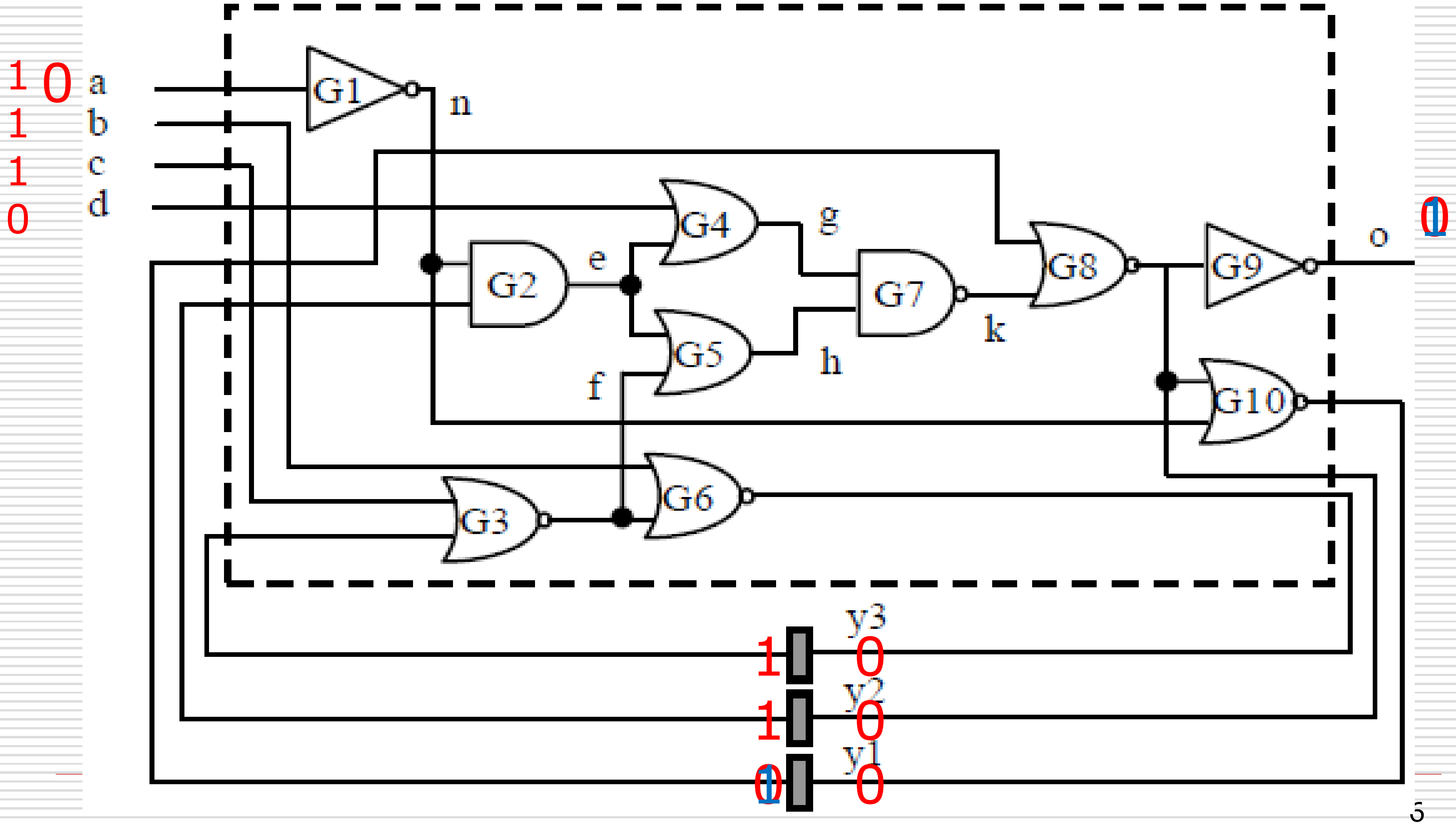
# Introduction

- Solution: hardening circuits to tolerate faults
- However:
  - Fault tolerance in circuits brings extra overhead
    - Area
    - Power
    - ……
- Selective protection:
  - Pin-point the most vulnerable components in the circuit to be hardened
  - Key technique is soft error failure rate analysis
- Our work: the probability of a soft error in a FF finally results as a failure.

## Introduction

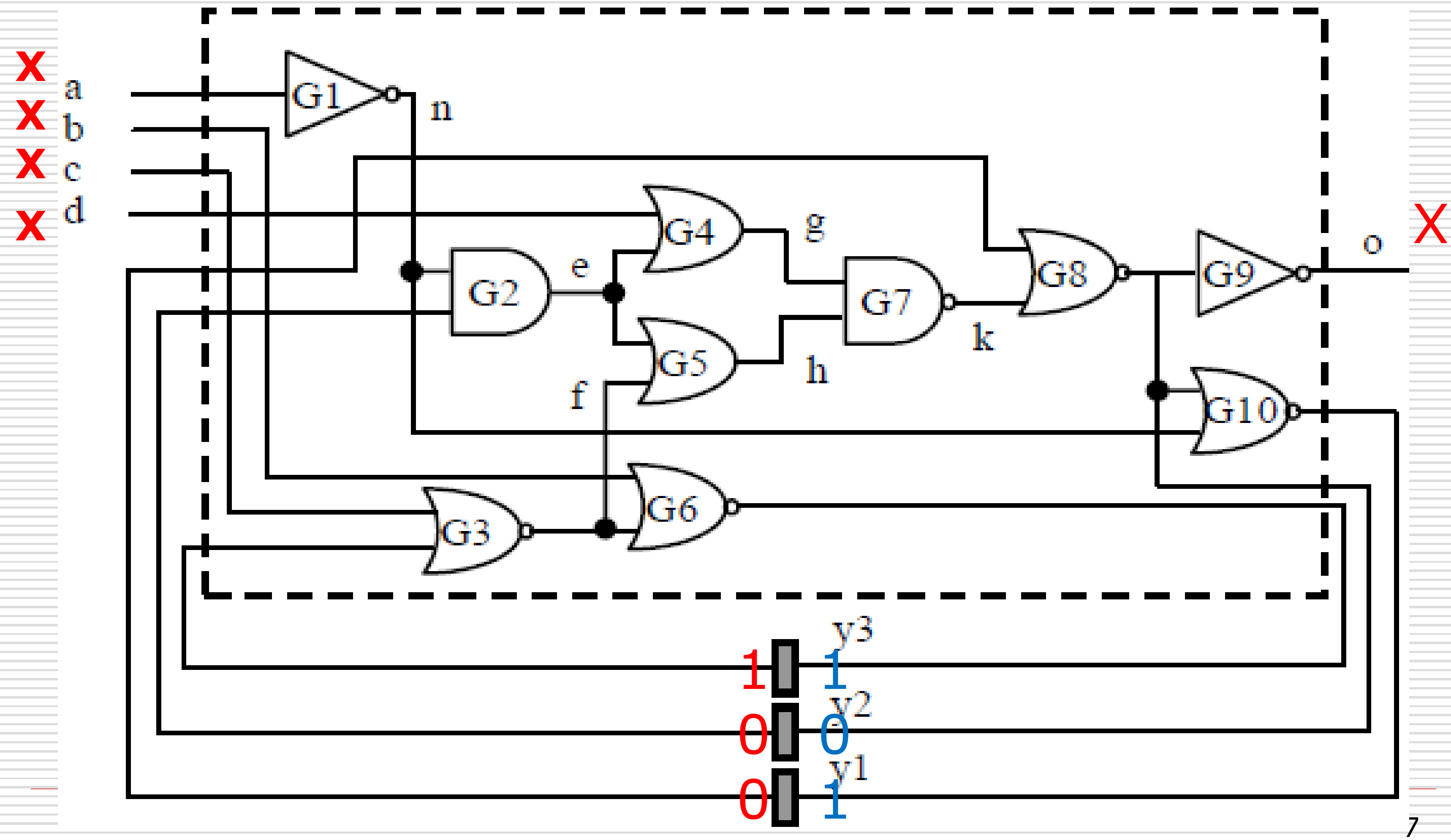
- Existing methods
  - Fault simulation based
    - Not complete
  - Formal verification based
    - Theorem Proving, Model Checking
      - Not scaled
      - Not completely automatic
  - Fault free simulation based
    - Suitable for processor designs
- Problem:
  - All methods do not take different application behaviors into account

# Motivation



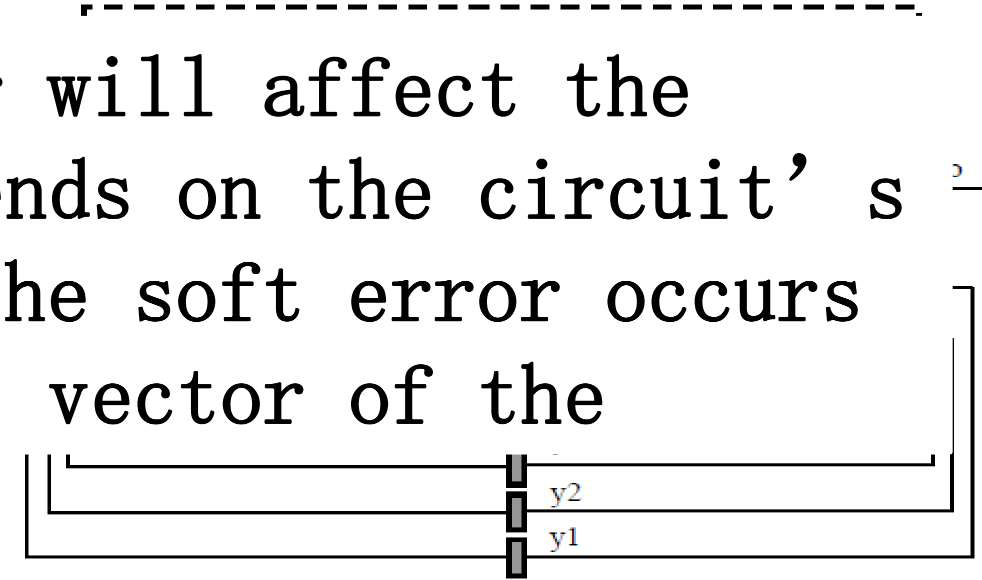
1 0 a
1 b
1 c
0 d
G1
n
G2
e
G3
f
G4
g
G5
h
G6
G7
k
G8
G9
o
G10
0
y3
1 0
y2
1 0
y1
0 0

## Motivation



## Motivation

- "000" is a *vulnerable state* of y1
- *vulnerable state set* (*VSS*)
  - $VSS(y1) = \{y1y2y3|X00;X11;X10\}$
  - $VSS(y2) = \{y1y2y3|0XX\}$
  - $VSS(y3) = \{y1y2y3|XXX\}$



whether a soft error will affect the circuit outputs depends on the circuit's current state when the soft error occurs as well as the input vector of the circuit

## Our method

- Several Definitions
  - **FF soft error Failure Rate (FFR)**
  - **Vulnerable State Vulnerability Factor (VSVF)**

$$\mathrm{FFR}(y) = (\text{Total Suspicious Runtime}(y) / \text{Total Runtime of Circuit}) \times \underline{R(y)}$$

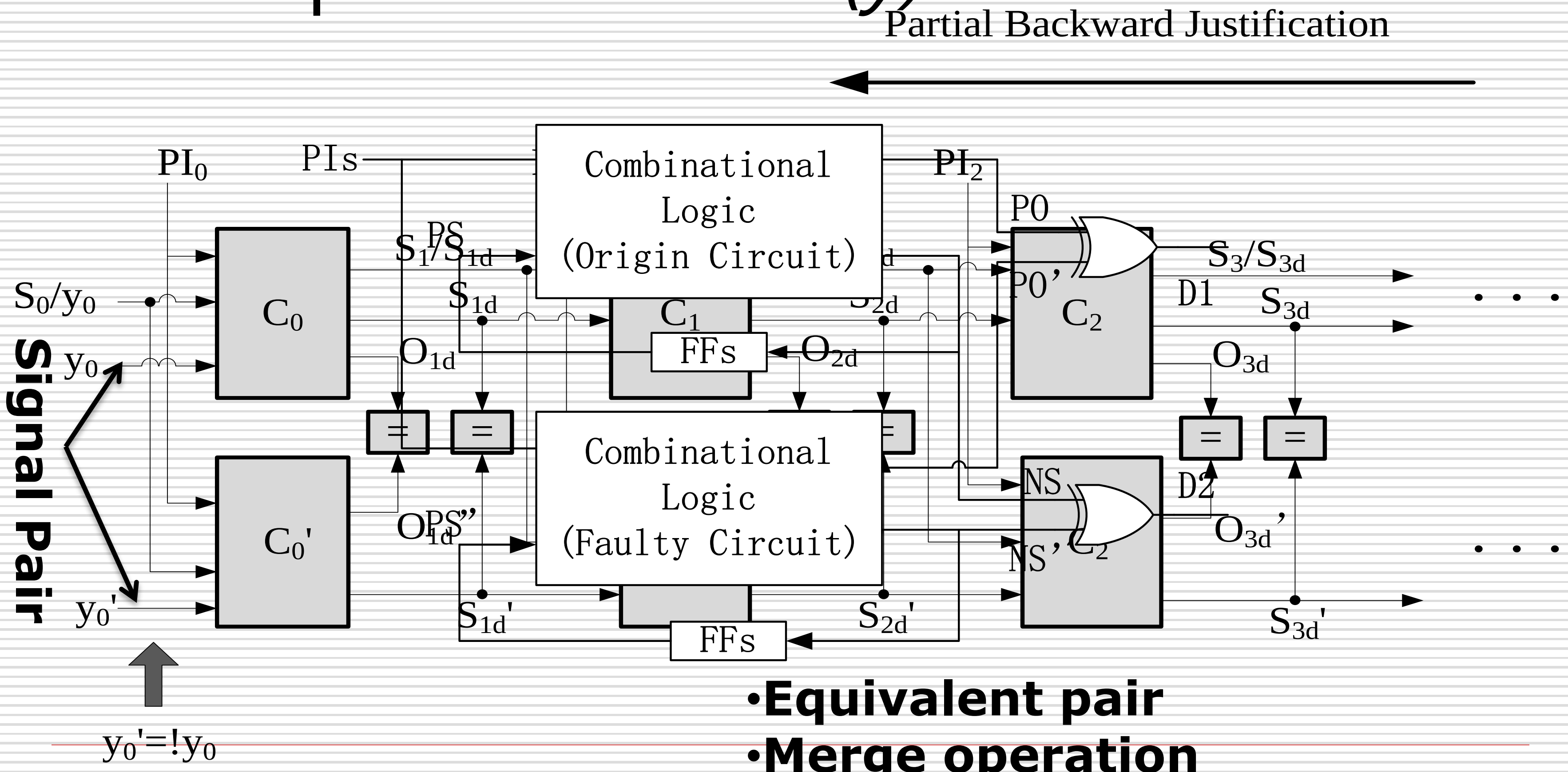
$$= \underline{\mathrm{VSVF}_i\,(y)} \times \sum_{i=1}^{|\mathrm{VSS}(y)|} \underline{((T(S_i))/(\sum_{j=1}^{n} T(S_j)))} = \mathrm{VSVF}_i\,(y) \times \sum_{i=1}^{|\mathrm{VSS}(y)|} \underline{f_i}$$

## Our method

- To compute *FFR(y)*
  - *VSS(y)*
  - $VSVF_i(y)$
  - the steady-state probability distribution of *y*

# Our method

- Computation of *VSS(y)*



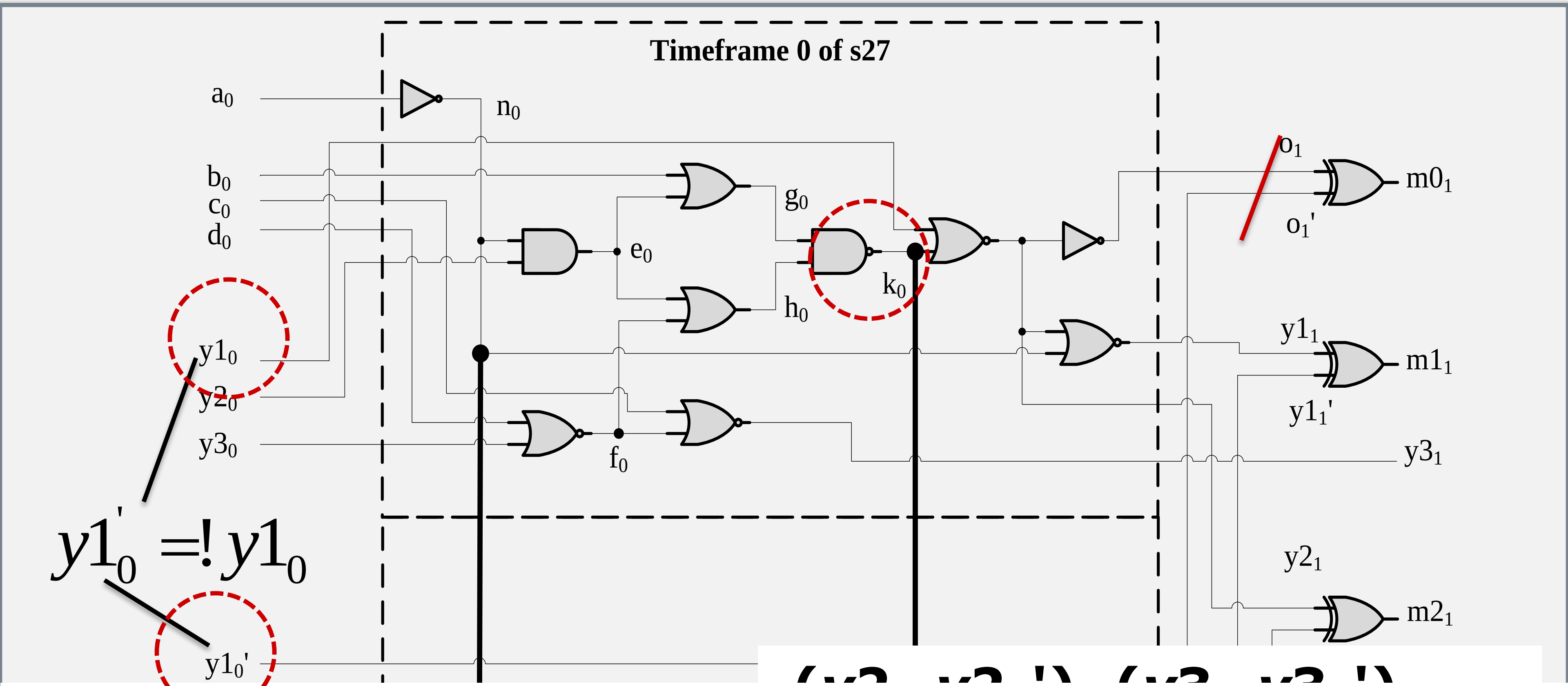
- **Equivalent pair**
- **Merge operation**
- **Discrepancy function**

## Our Method

- Computation of *VSS(y)*
  - Basic idea:
    - Checking the equivalence of each primary output (PO) pair at time step *d*.
    - If not, PBJ is used to compute the necessary state requirements set (*SRS*) to differentiate each nonequivalent pair that should be satisfied by the present state of *the circuit* at time step 0.
    - For a nonequivalent PO pair, each state in its *SRS* is a vulnerable state of *y*. Thus, add the *SRS* into set *V*.

## Our Method

- Computation of *VSS(y)*
  - When to stop backward justification
    - The initial state is included in *SRS* of step *d* ×
    - SRS is empty or it reaches a fixpoint √
  - When to stop unrolling
    - *V* is equal to the reachable state set *R* of the *circuit*, and *VSS(y) = R*
    - Every NS pair at time step *d* is an equivalent pair



Checking equivalence of ($o_1$, $o_1$'):

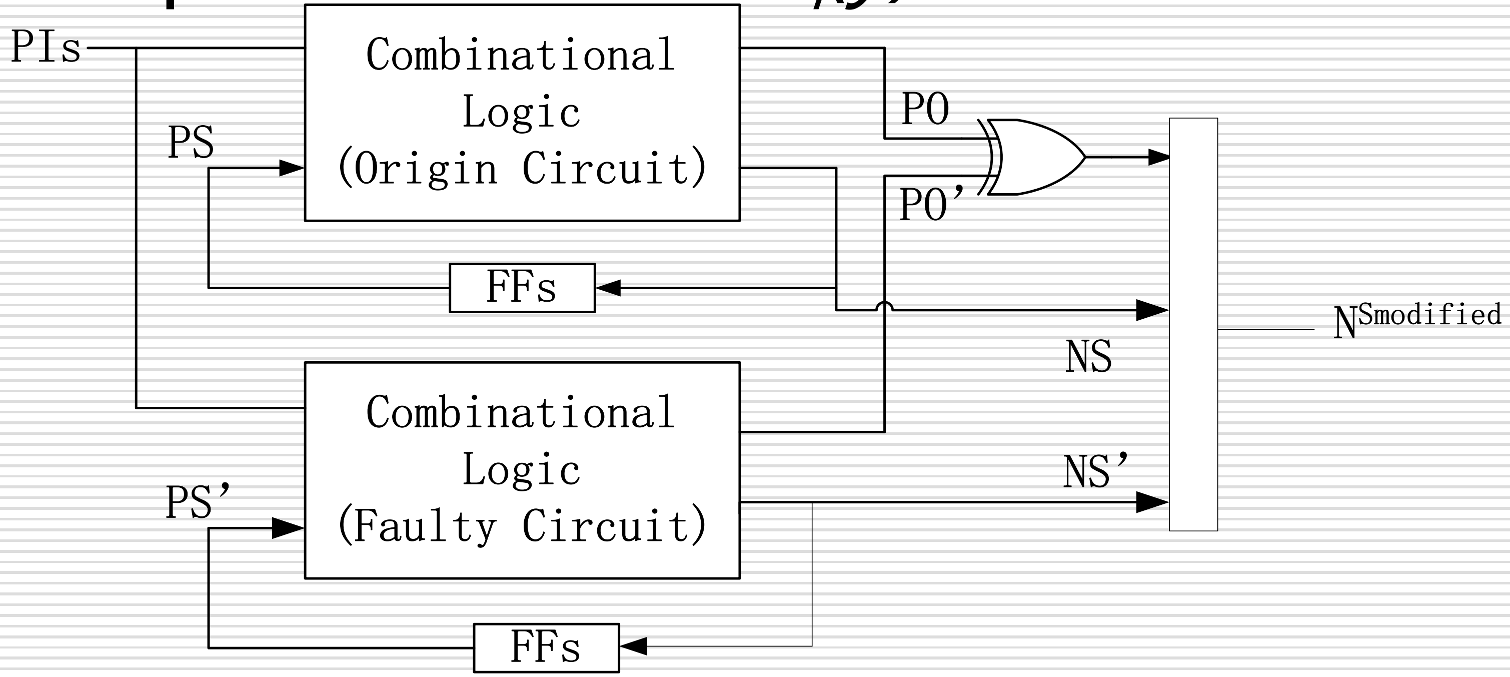
- Cutset is $\lambda_0$={ $y2_1$, $y2_1$'}
- Backward enlarge cutset to $\lambda_{02}$={ $y1_0$ , $k_0$ , $y1_0$' }
- Backward traversal until encounter PIs and PSs
- PIs are existentially quantified
- The set of State to Diff ($o_1$, $o_1$') is

  {($y1_0$, $y2_0$, $y3_0$) | (X, 0, 0), (X, 1, 0), (X, 1, 1)}

{( $y1_0$, $k_0$) | (X, 0)}

## Our method

- Computation of *VSVF_i(y)*


PIs
PS
Combinational Logic (Origin Circuit)
PO
PO'
FFs
NS
NSmodified
Combinational Logic (Faulty Circuit)
PS'
NS'
FFs

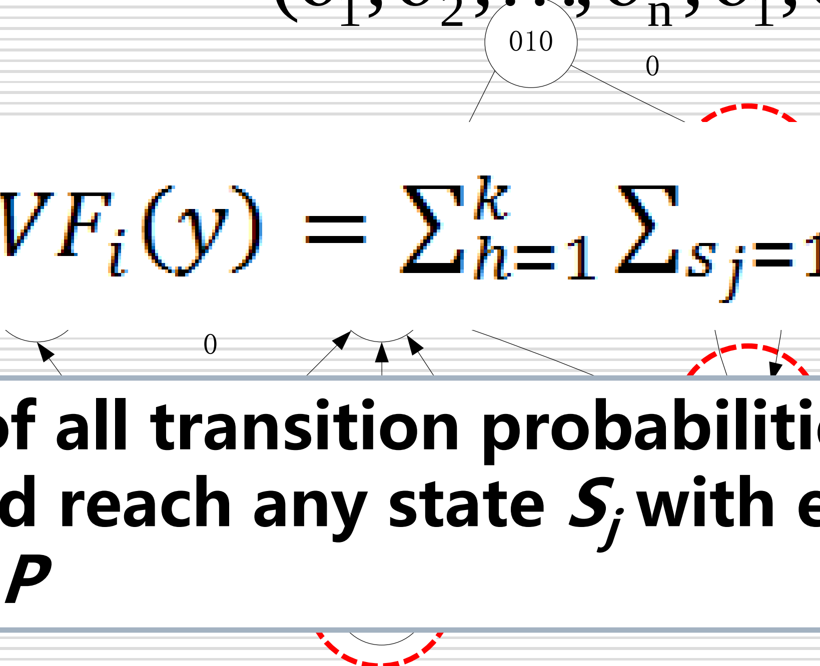
$$NS^{modified} = (\delta_1, \delta_2, \ldots, \delta_n, \delta_1', \delta_2', \ldots, \delta_n', \varepsilon)$$

**$\varepsilon$ : the state bit for erroneous outputs (==1)**

## Our method

$$P = \begin{bmatrix} p_{00} & p_{01} & \dots & p_{06} & p_{07} \\ p_{10} & p_{11} & \dots & p_{16} & p_{17} \\ \cdots & \cdots & \vdots & \cdots & \cdots \\ p_{60} & p_{61} & \dots & p_{66} & p_{67} \\ p_{70} & p_{71} & \dots & p_{76} & p_{77} \end{bmatrix} = \begin{bmatrix} p_{00} & p_{01} & \dots & p_{06} & p_{07} \\ 0 & 0 & \dots & 0 & 0 \\ \cdots & \cdots & \vdots & \cdots & \cdots \\ p_{60} & p_{61} & \dots & p_{66} & p_{67} \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}$$

$$NS^{modified}_{(a)} = (\delta_1, \delta_2, \ldots, \delta_n, \delta'_1, \delta'_2, \ldots, \delta'_n, \varepsilon)$$

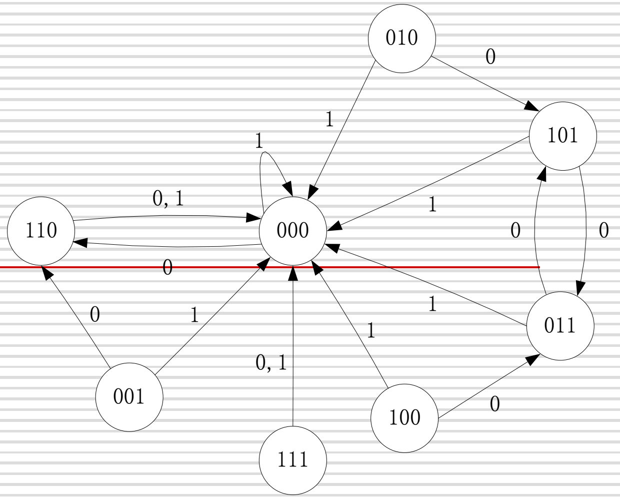


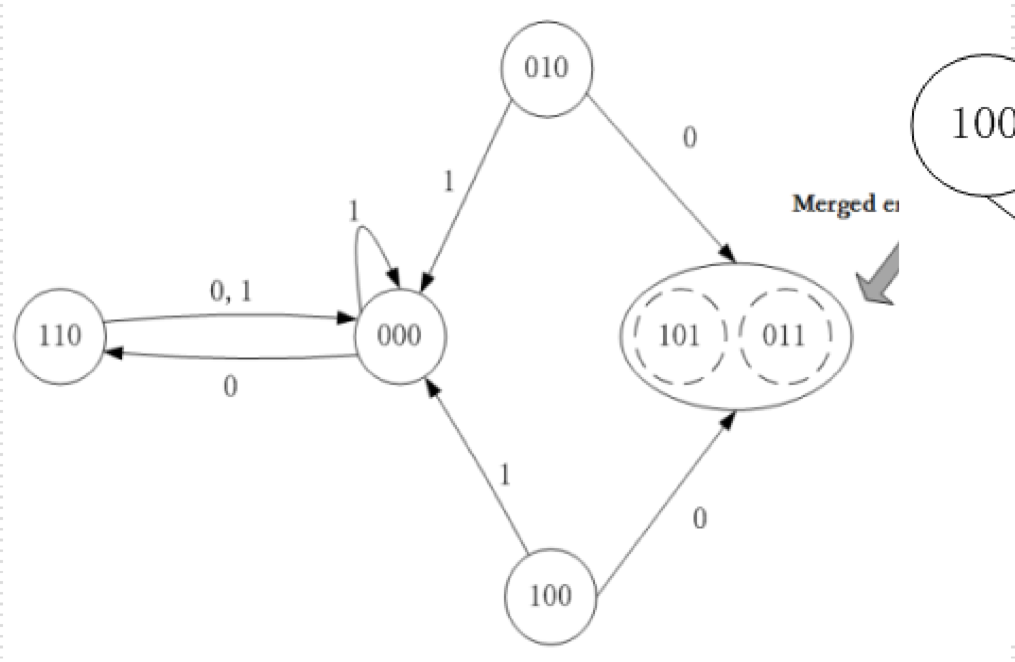
$$VSVF_i(y) = \sum_{h=1}^{k} \sum_{s_j=1} p_{ij}^{(h)}$$

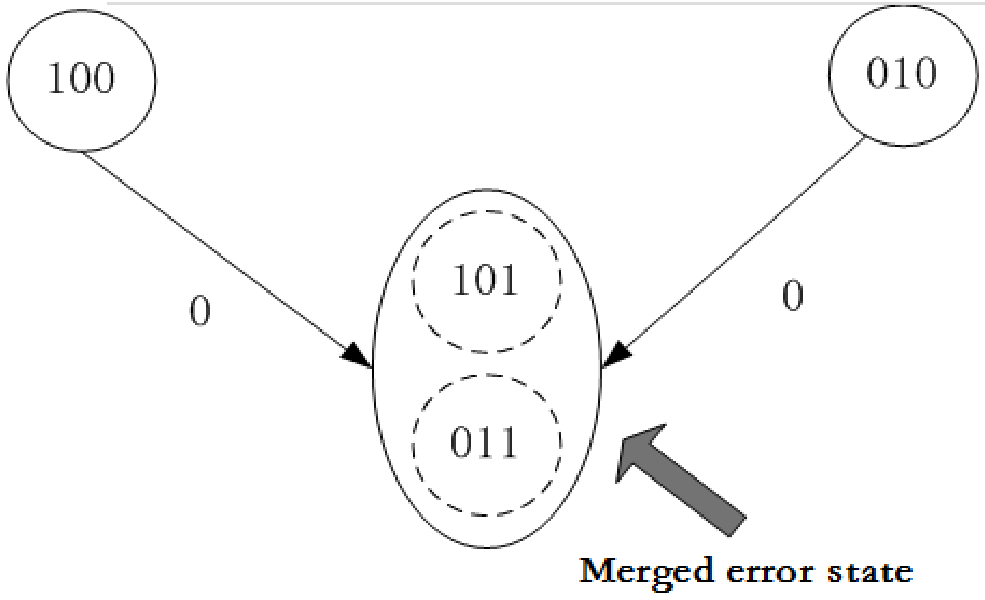
the sum of all transition probabilities that start from $S_i$ and reach any state $S_j$ with error outputs in $h$ steps in $P$

# Our method

- Computation of *VSVF$_i$(y)*







## Our method

- Computation of Steady-State Probabilities
  - Traditional MC based method

## Experimental results

- SpaceWire end node
  - 145 FFs
    - 110 FFs are robust
    - 35 FFs are vulnerable
      - 16 FFs (11%) are protected – error coverage 77%
        - 6.5% power and 0.15% area overhead
      - 35 FFs (24%) are protected – error coverage ~100%
        - 14.8% power and 0.26% area overhead

$$\text{Error Coverage} = 1 - (\text{Error outputs detected after protection})/(\text{Error outputs detected without protection}) \times 100\%$$

# Experimental results – ISCAS' 89

| Circuit | #FFs | #robust FFs | Error Coverage | | | | Time (s) | Memory (MB) | Error Coverage [14] | | | | Times[14] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 20% | 40% | 60% | 80% | | | 20% | 40% | 60% | 80% | |
| s4863 | 104 | 0 | *0.261* | *0.463* | *0.670* | *0.868* | ***202.2*** | 37.319 | *0.165* | *0.361* | *0.560* | *0.732* | ***129*** |
| s5378 | 179 | 23 | *0.762* | *0.863* | *0.972* | *0.990* | ***338.6*** | 40.205 | *0.632* | *0.754* | *0.851* | *0.928* | ***241*** |
| s3384 | 183 | 0 | *0.333* | *0.459* | *0.527* | *0.805* | ***679.8*** | 67.801 | *0.229* | *0.349* | *0.413* | *0.648* | ***417*** |
| s9234 | 211 | 33 | *0.296* | *0.453* | *0.778* | *0.981* | ***1441.9*** | 94.493 | *0.179* | *0.371* | *0.690* | *0.893* | ***614*** |
| s15850 | 534 | 0 | *0.459* | *0.573* | *0.778* | *0.986* | ***2055.8*** | 137.180 | *0.353* | *0.568* | *0.754* | *0.916* | ***1231*** |
| s38584 | 1426 | 0 | *0.422* | *0.617* | *0.760* | *0.959* | ***5561.6*** | 371.201 | *0.316* | *0.501* | *0.659* | *0.877* | ***4305*** |
| s38417 | 1636 | 72 | *0.463* | *0.645* | *0.781* | *0.972* | ***7242.3*** | 402.729 | *0.362* | *0.540* | *0.694* | *0.901* | ***6276*** |
| s35932 | 1728 | 0 | *0.512* | *0.755* | *0.919* | *0.997* | ***10023.7*** | 401.815 | *0.421* | *0.635* | *0.819* | *0.920* | ***9164*** |

$$\text{Error Coverage} = 1 - (\text{Error outputs detected after protection})/(\text{Error outputs detected without protection}) \times 100\%$$

## Case study

- Estar2 – Instruction decoder
  - 369 FFs
  - 3439 comb. gates
  - 485 outputs
- QGIS application
  - Collect input distribution - SimpleScalar
  - Analyze: 1329 seconds
  - 189 FFs to be protected, the error coverage is about 91%
    - power and area overhead do not exceed 22.7% and 0.59%

## Conclusion

- A novel methodology for performing soft error failure rate analysis of arbitrary sequential circuit designs
  - A novel failure rate measurement – VSS
  - A novel methodology - combines circuit states and application behaviors
  - A really automatic formal method – SEC based
- Future
  - combinational components
  - multiple soft error
  - SAT-based techniques

Thank you!

Q & A