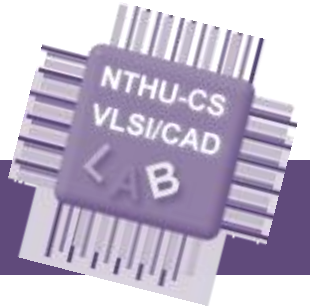
# Fault-tolerant by Using Scan-chain Test TSV

**Fu-Wei Chen, Hui-Ling Ting, and TingTing Hwang**

Computer Science Dept.
National Tsing Hua University, Taiwan





# Outline

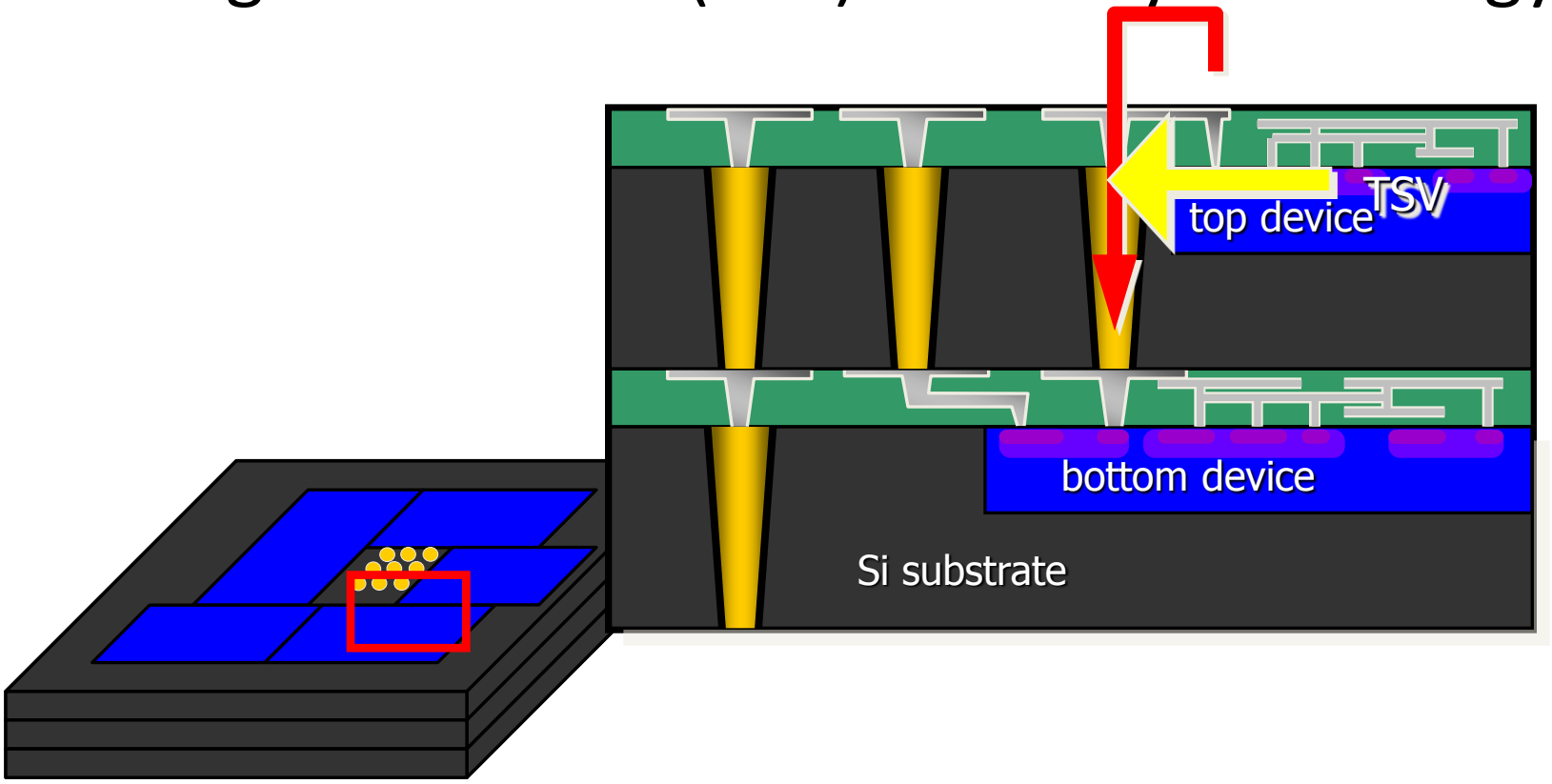
- Introduction
- Our Proposed Architecture
- Problem Definition & Our Proposed Algorithm
- Experiment
- Conclusions

# Outline

- Introduction
- Our Proposed Architecture
- Problem Definition & Our Proposed Algorithm
- Experiment
- Conclusions

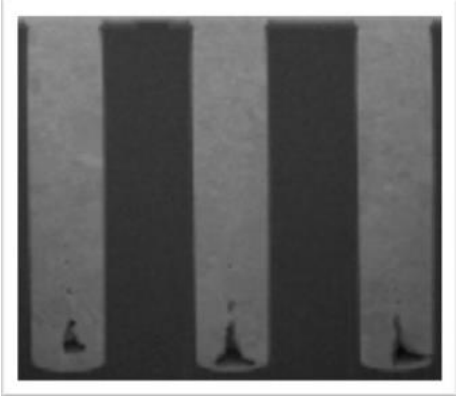
# Through-Silicon Via(TSV)

- Through-Silicon Via (TSV) – the key technology

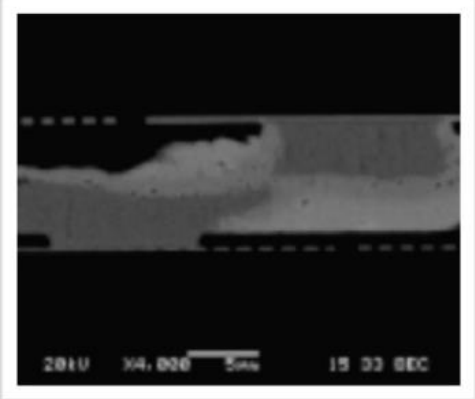


# Faulty TSV

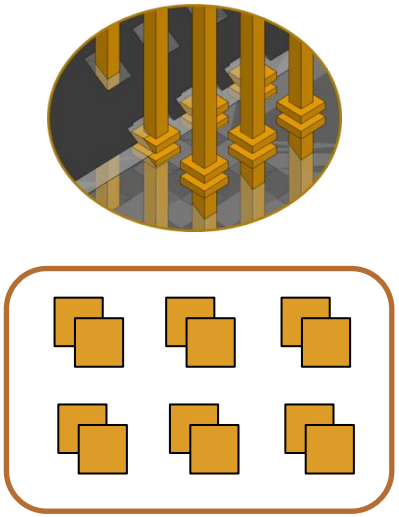
- TSV defects can happen in **fabrication process** and **bonding stage**, which can reduce the **yield** and increase the **cost**.



Micro-voids on TSV axis or large voids at bottom

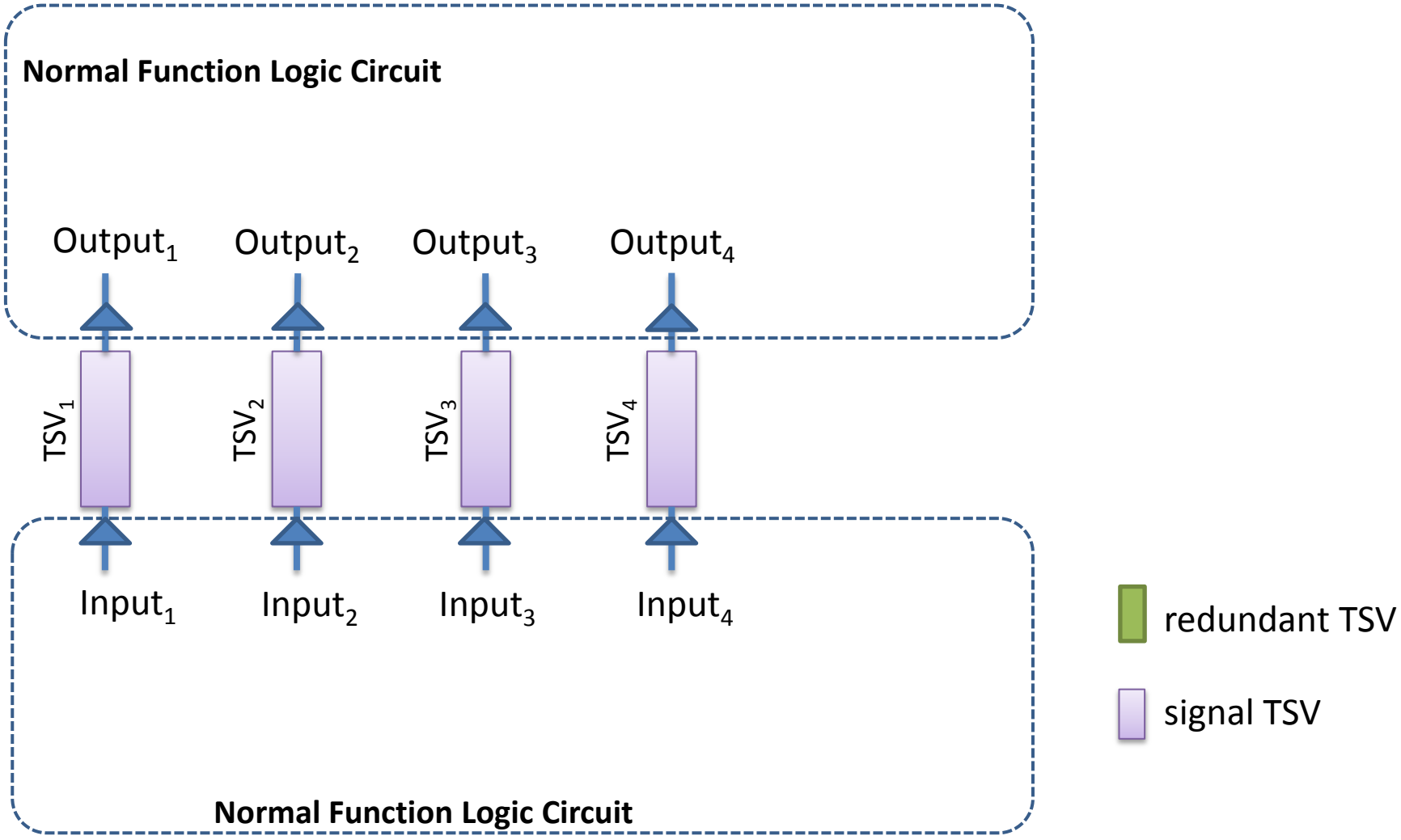


Misalignment



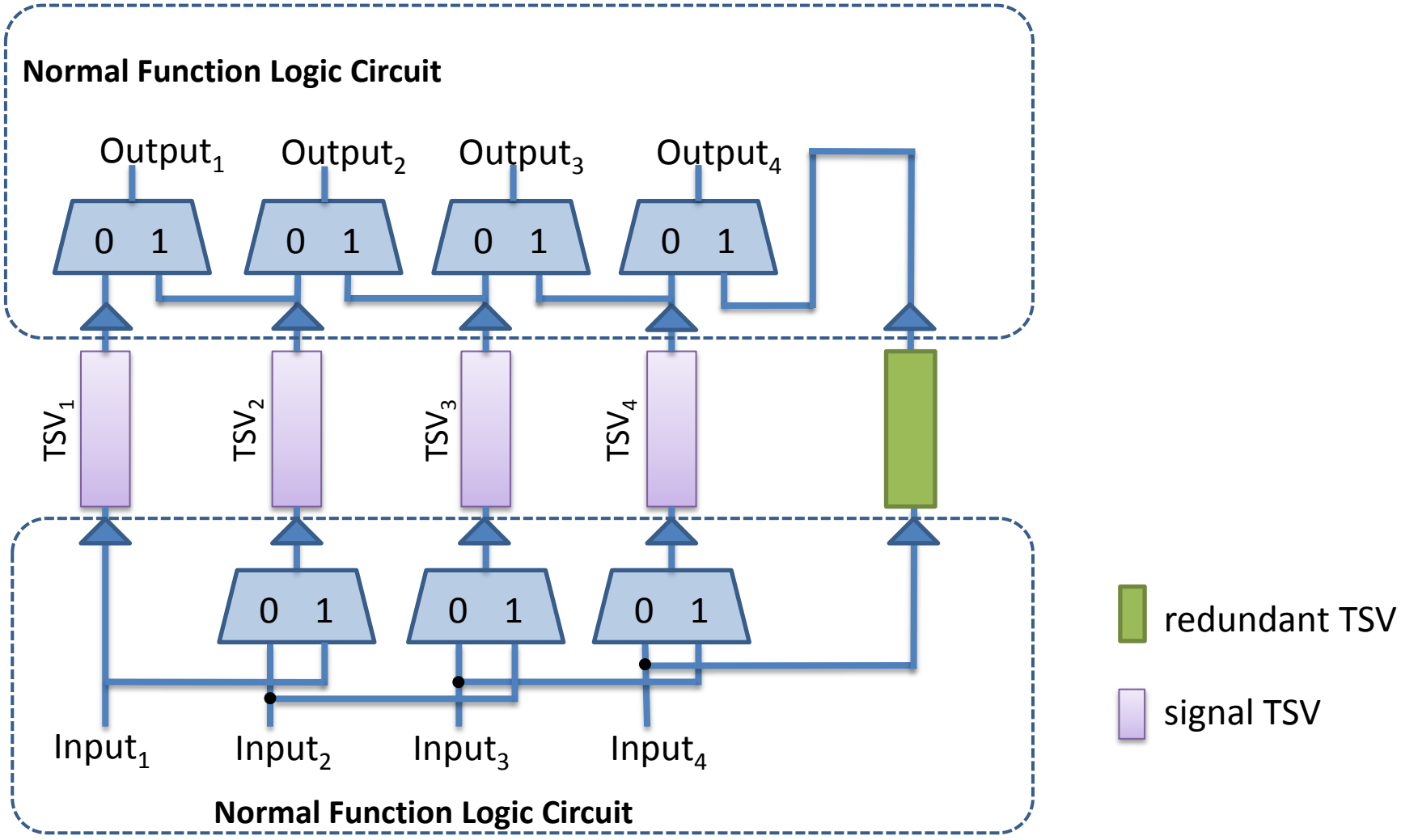
Fangming Ye et al., "TSV Open Defects in 3D Integrated Circuits: Characterization, Test, and Optimal Spare Allocation", DAC 2012

# Redundant TSV Repairing



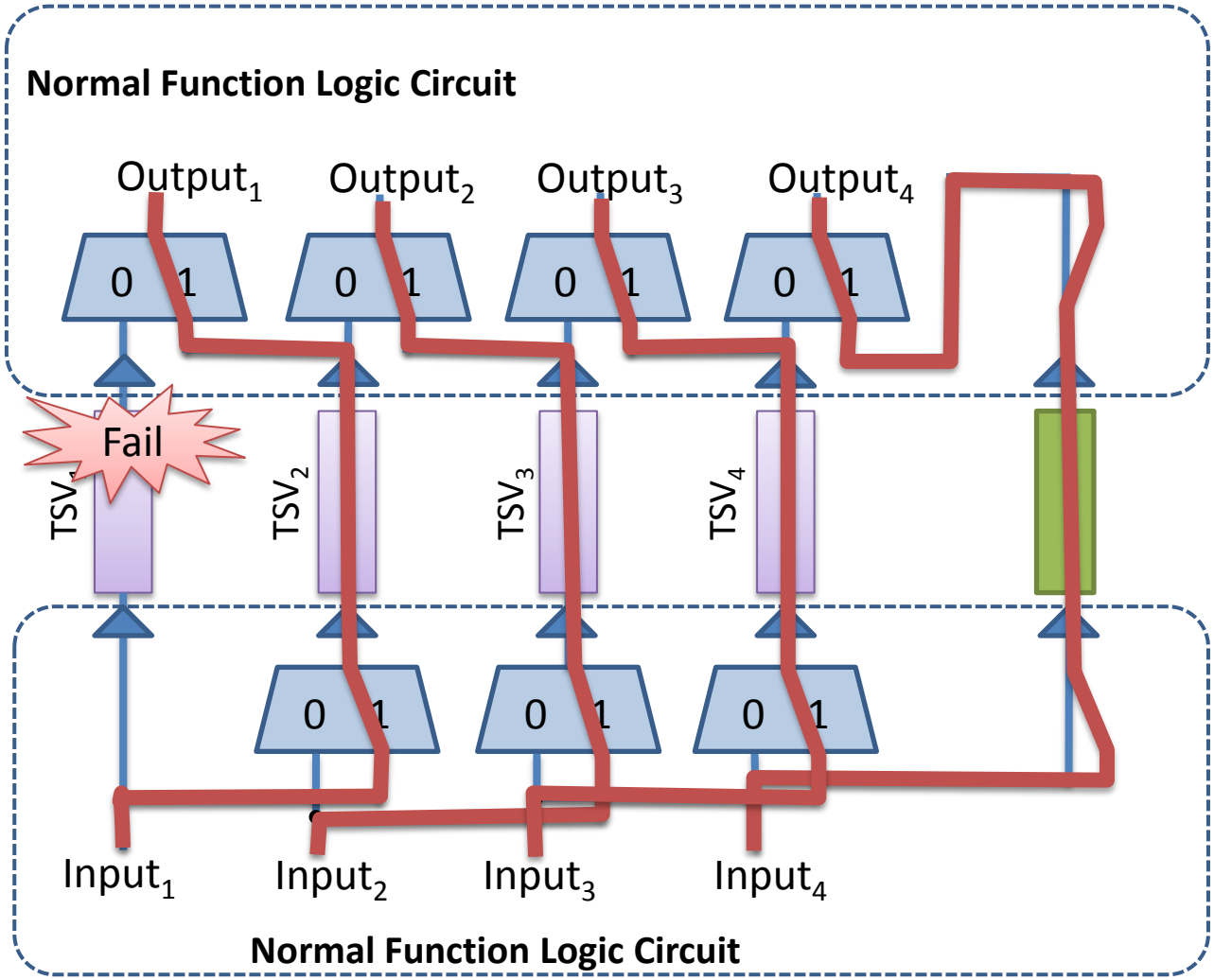

Ref: Ang-Chih Hsieh; TingTing Hwang, '' TSV Redundancy: Architecture and Design Issues in 3-D IC'', TVLSI 2012

# Redundant TSV Repairing




Ref: Ang-Chih Hsieh; TingTing Hwang, " TSV Redundancy: Architecture and Design Issues in 3-D IC", TVLSI 2012

# Redundant TSV Repairing (Faulty)



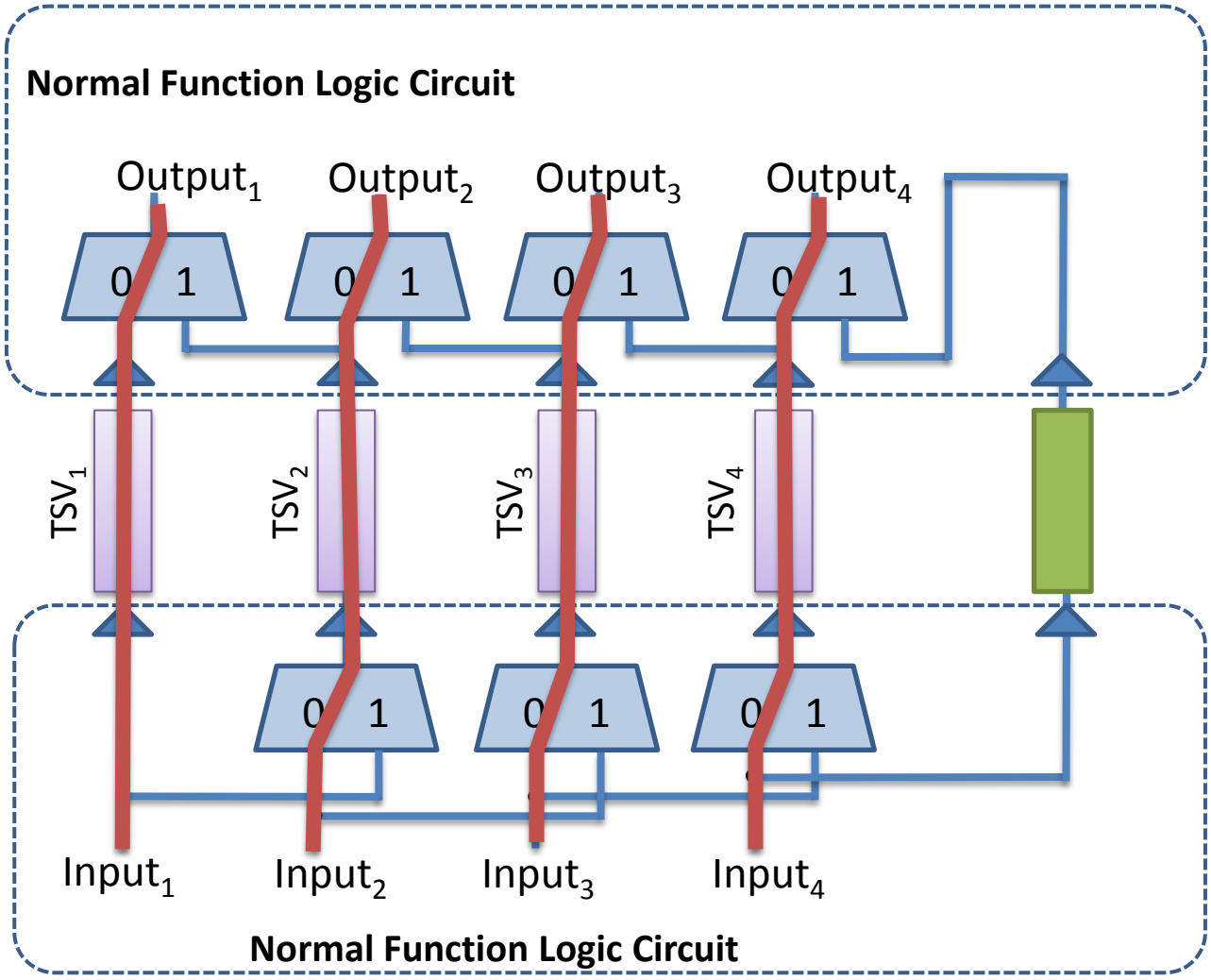

Normal Mode:
MUX configuration signal is 1

redundant TSV

signal TSV

Ref: Ang-Chih Hsieh; TingTing Hwang, " TSV Redundancy: Architecture and Design Issues in 3-D IC", TVLSI 2012

# Redundant TSV Repairing (Fault Free)




Normal Mode:
MUX configuration signal is 0

redundant TSV

signal TSV

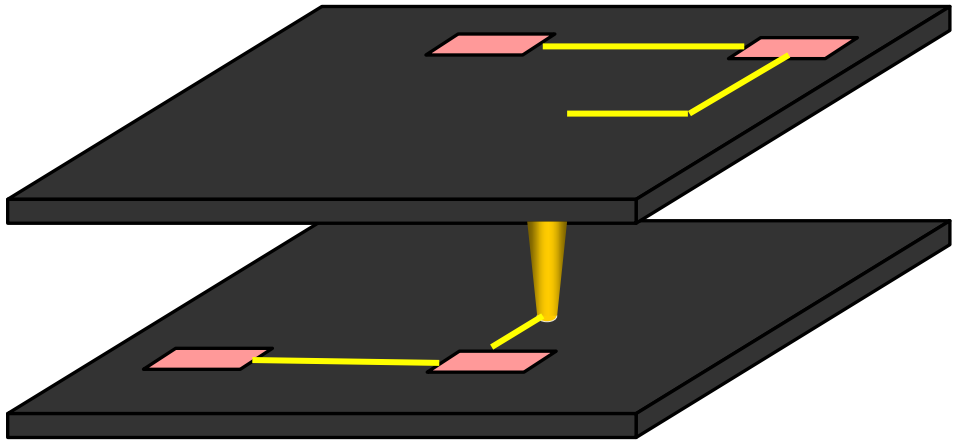
Ref: Ang-Chih Hsieh; TingTing Hwang, " TSV Redundancy: Architecture and Design Issues in 3-D IC", TVLSI 2012

# Outline

- Introduction
- Our Proposed Architecture
- Problem Definition & Our Proposed Algorithm
- Experiment
- Conclusions

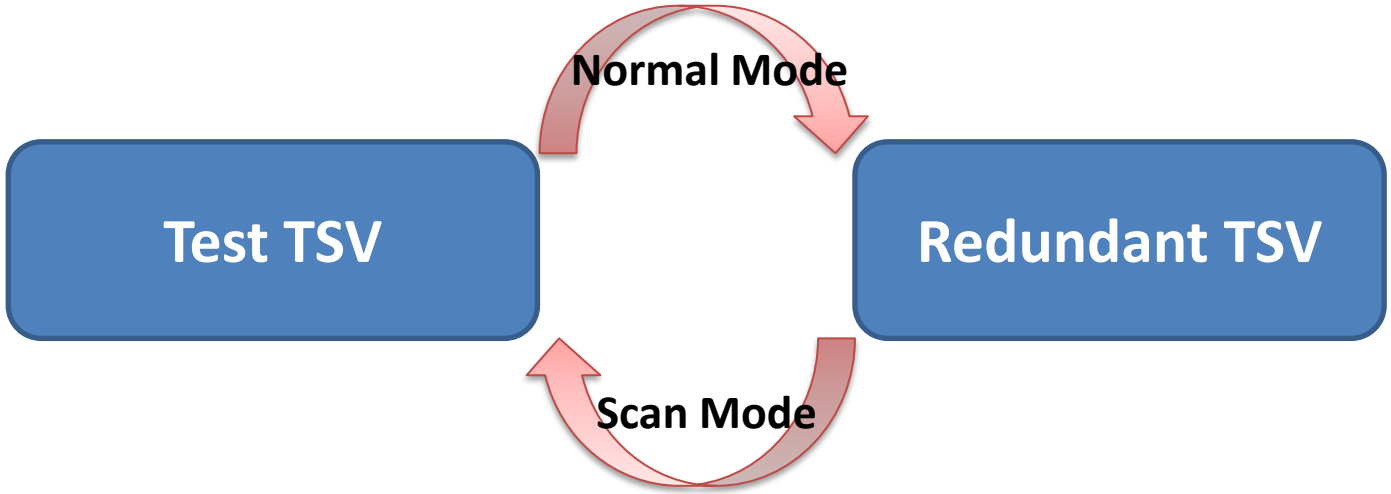
# Test TSV for 3-D Scan-chain

- Test TSVs are used to shift data in scan-in and scan-out phases during scan mode in **post-bond testing**

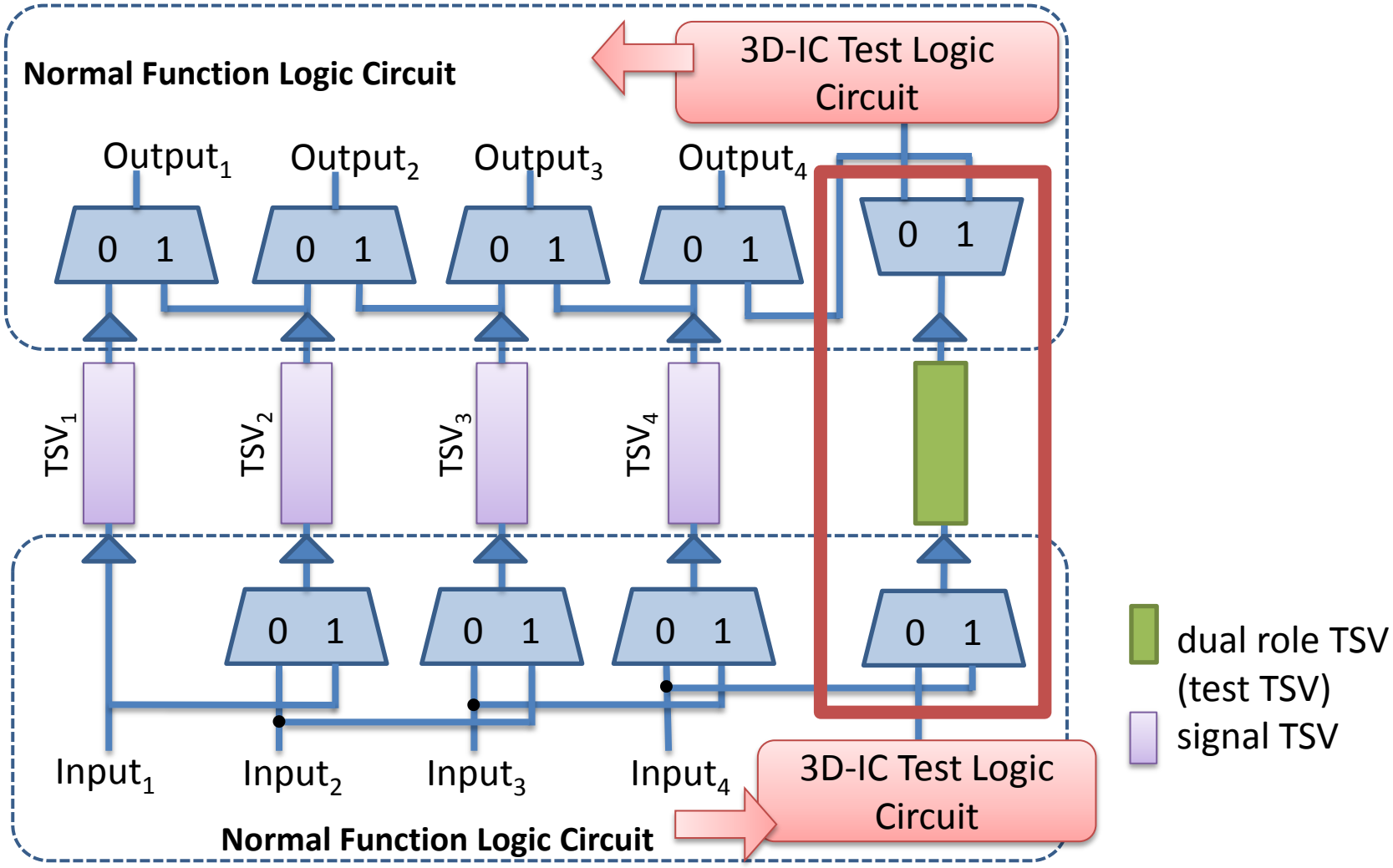


# Dual Role of TSV

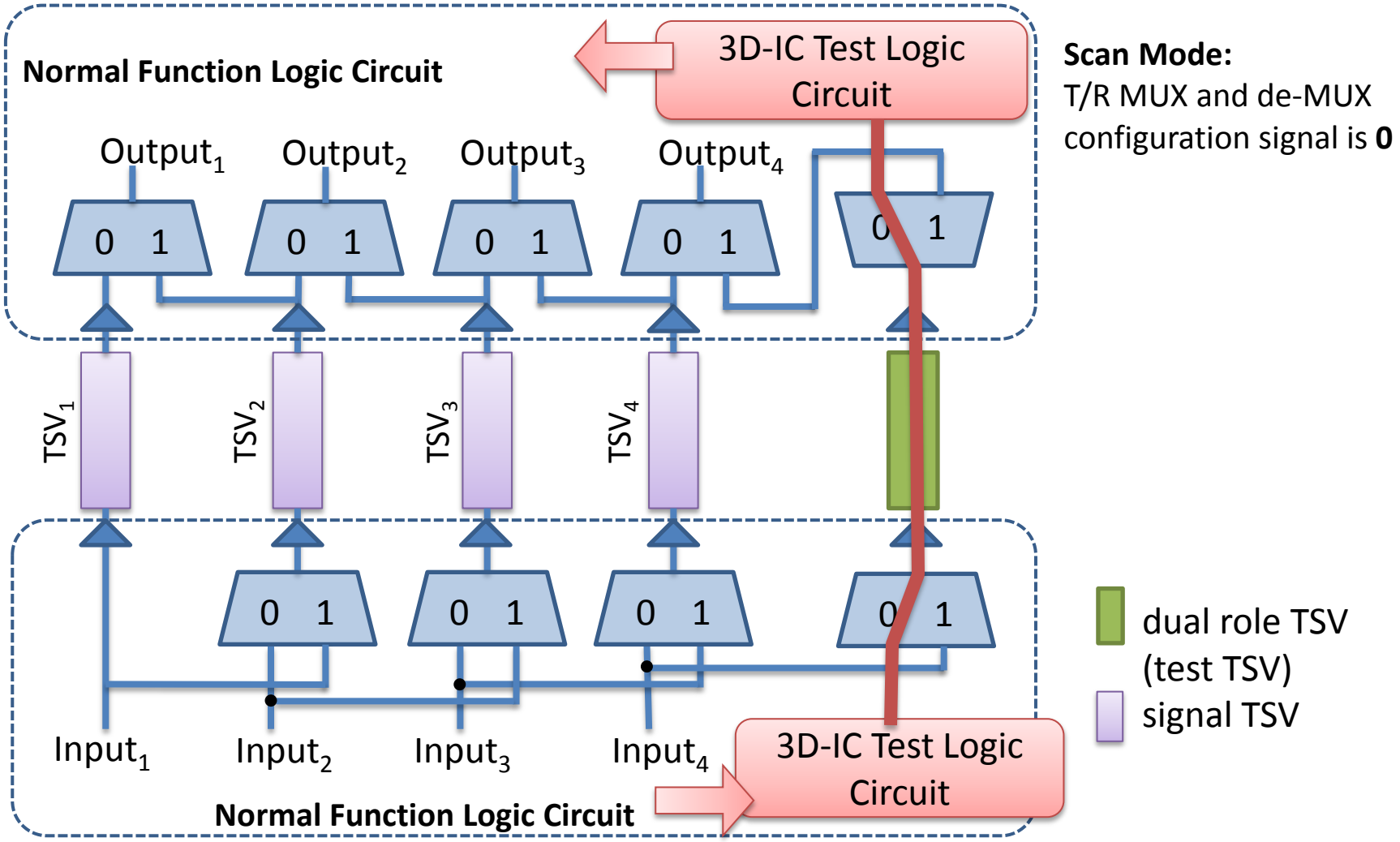
- **Test TSVs** are used to shift data in scan-in and scan-out phases during scan mode in post-bond testing but **not used** in normal mode
- In scan mode: TSV as **test TSV**
- In normal mode: TSV as **redundant TSV**



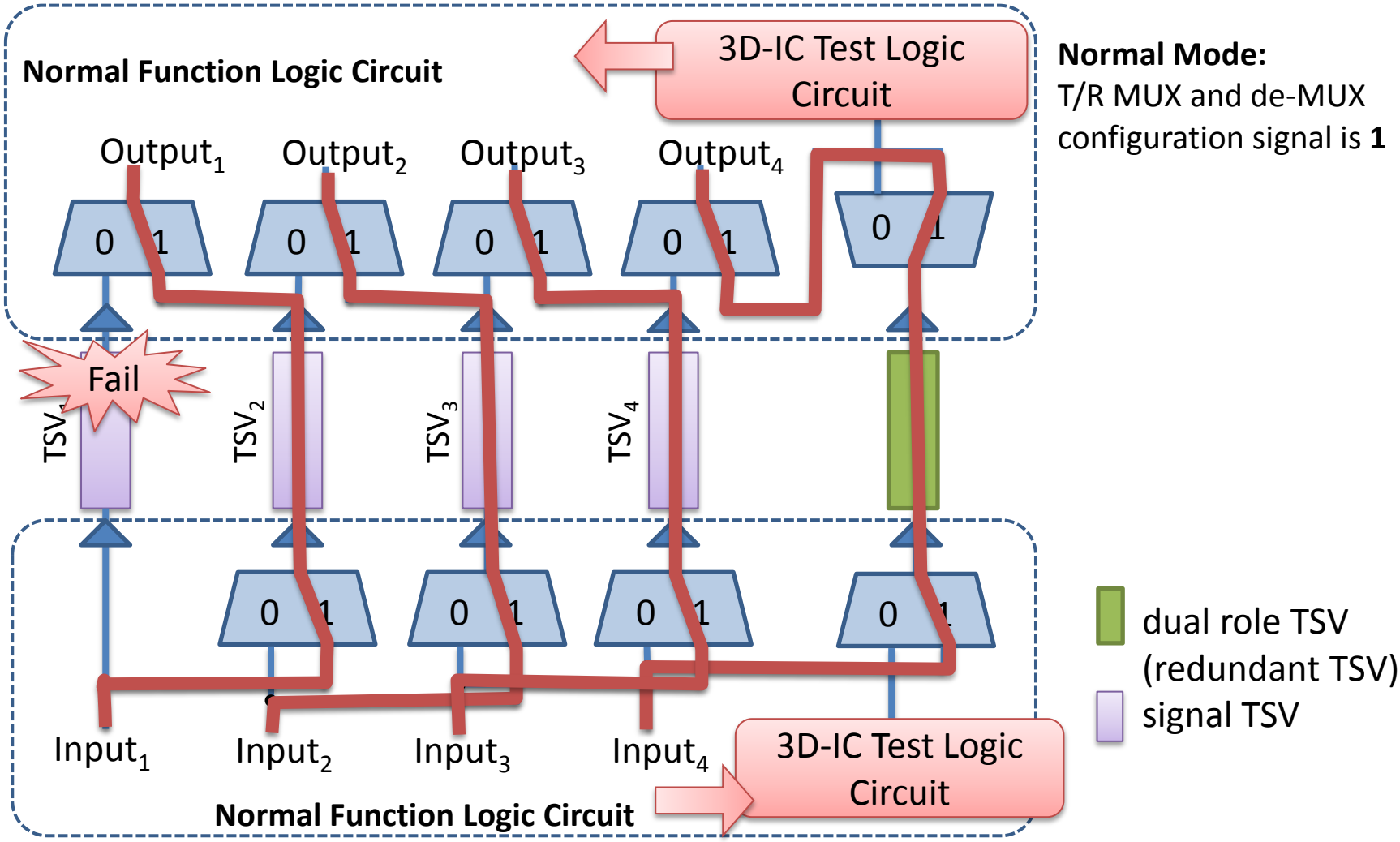
# Test TSV Repairing



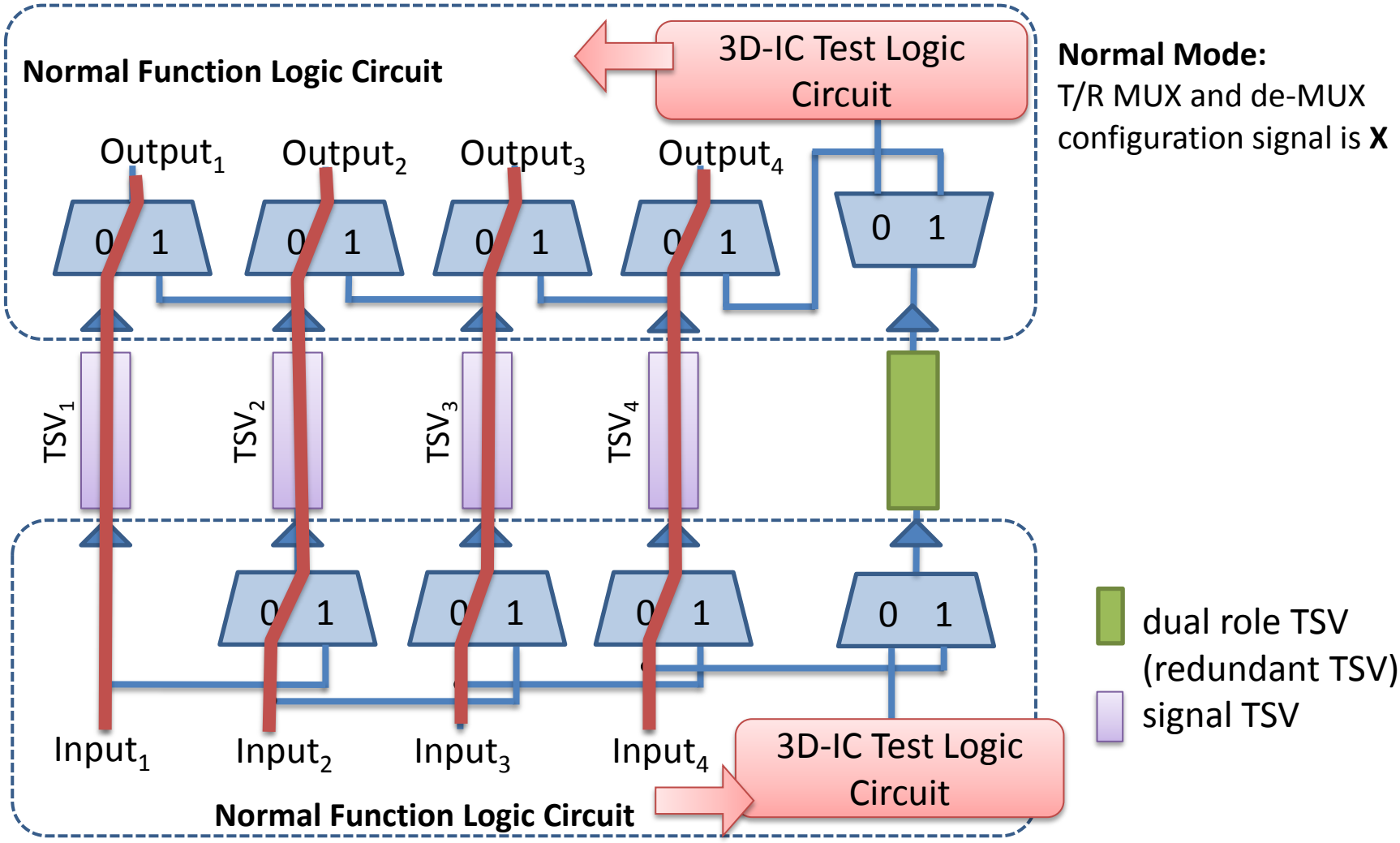
# Test TSV Repairing (Scan Mode)



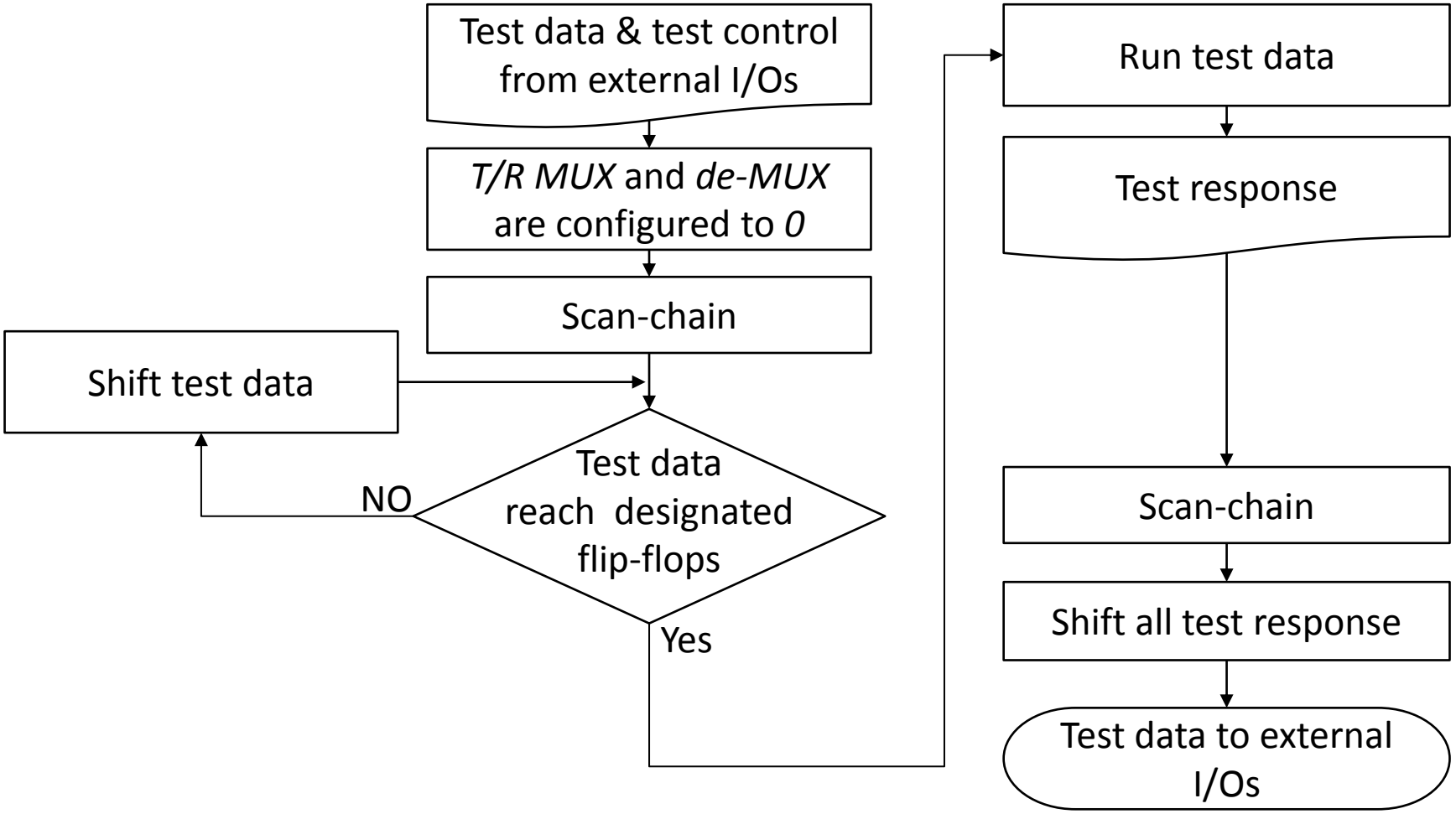
# Test TSV Repairing (Normal Mode: Faulty)



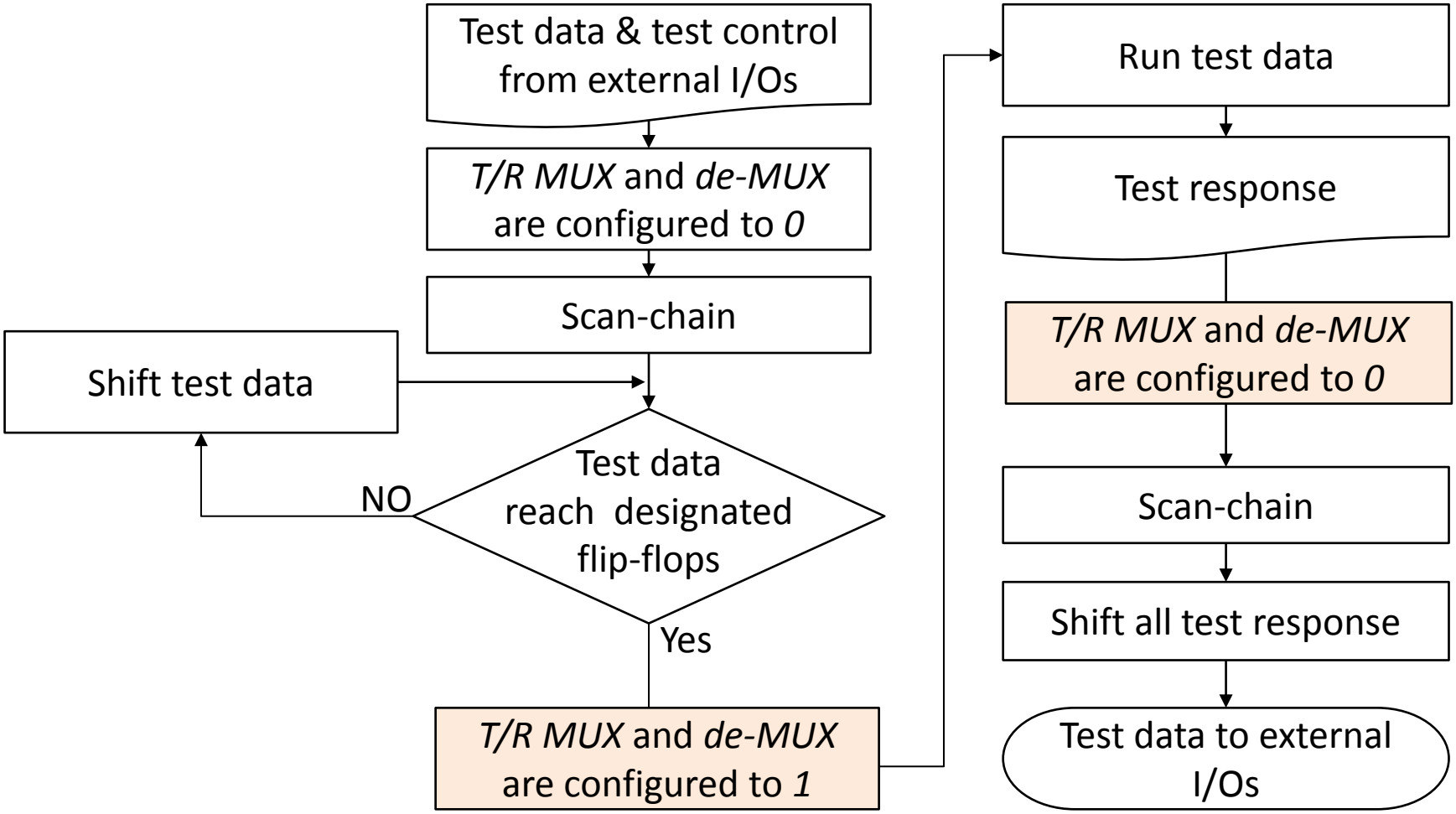
# Test TSV Repairing (Normal Mode: Fault Free)



# Flow of Testing 3-D Designs (Fault Free)



# Flow of Testing 3-D Designs (Faulty)



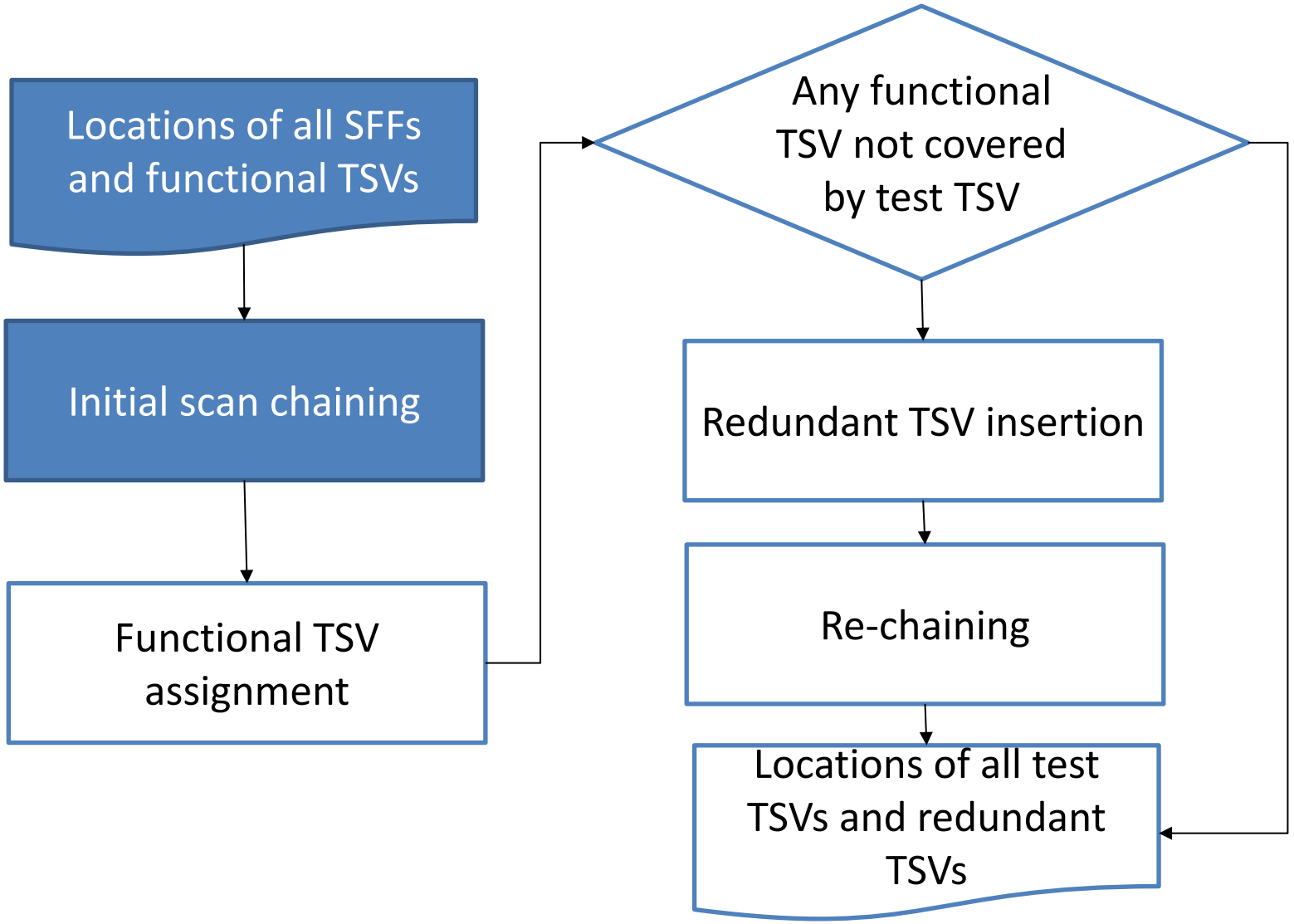
# Outline

- Introduction
- Our Proposed Architecture
- Problem Definition & Our Proposed Algorithm
- Experiment
- Conclusions

# Problem Definition

- **Input:** a set of placed module blocks and functional TSVs, logic gates, and scan flip-flops
- **Objective:**
  - **minimize WL of 3-D IC scan-chain**
  - **minimize WL of TSV repairing chain**
  - **minimize #redundant TSVs**

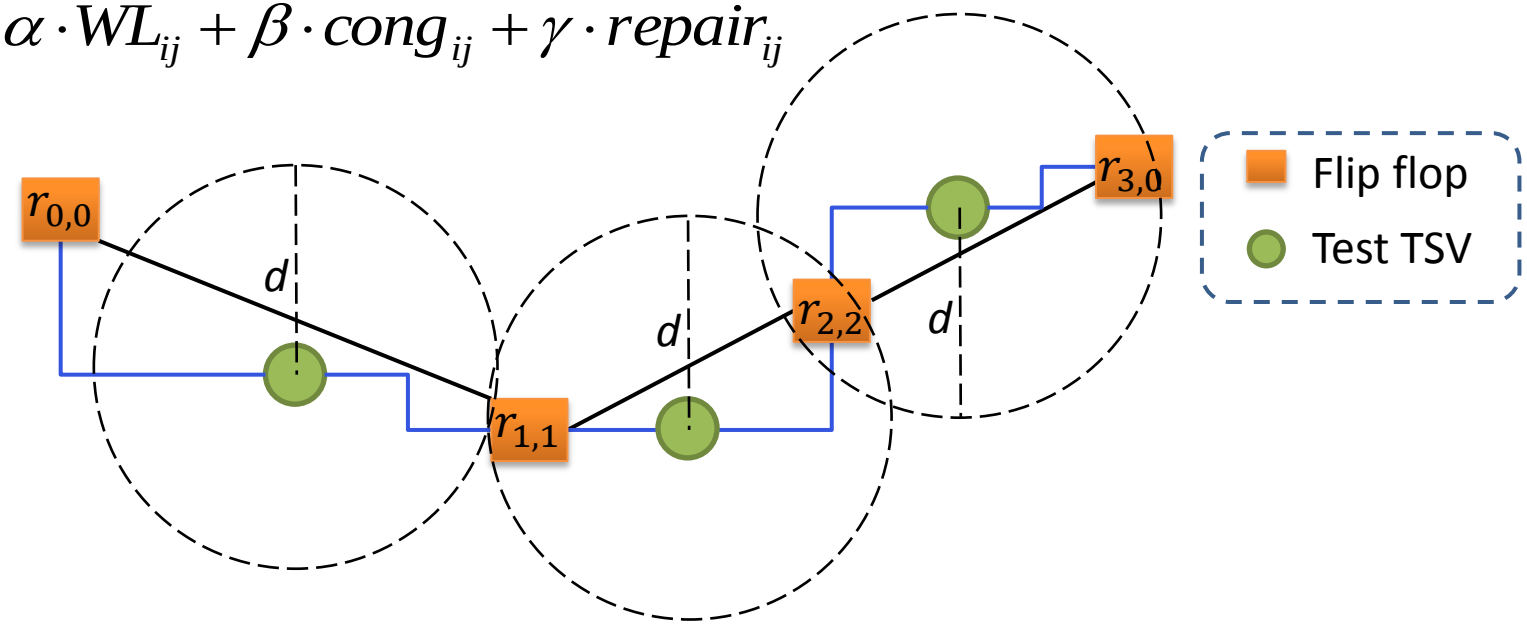
# Functional TSV Repairing Algorithm



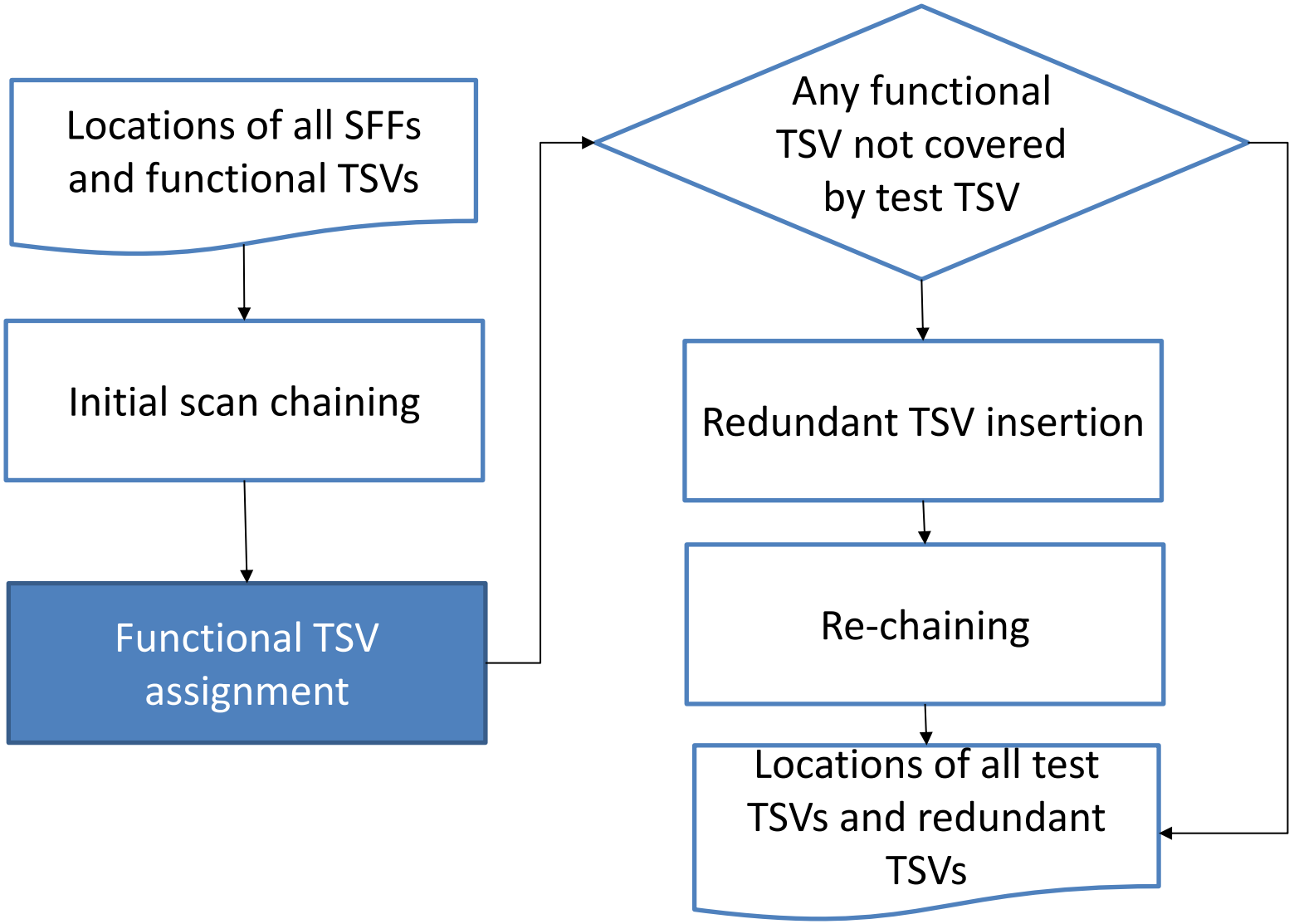
# Initial Scan Chaining

- The chaining of scan FFs is modeled as a **complete graph**, $G = (V,E,C)$
- The **minimum cost scan-chain** solution is to find a minimum distance *Traveling Salesman Problem* (TSP)
- The cost of each edge between $\mathrm{FF_i}$ (Vi) and $\mathrm{FF_j}$ ($\mathrm{V_j}$) in $G$

$$c_{ij} = \alpha \cdot WL_{ij} + \beta \cdot cong_{ij} + \gamma \cdot repair_{ij}$$



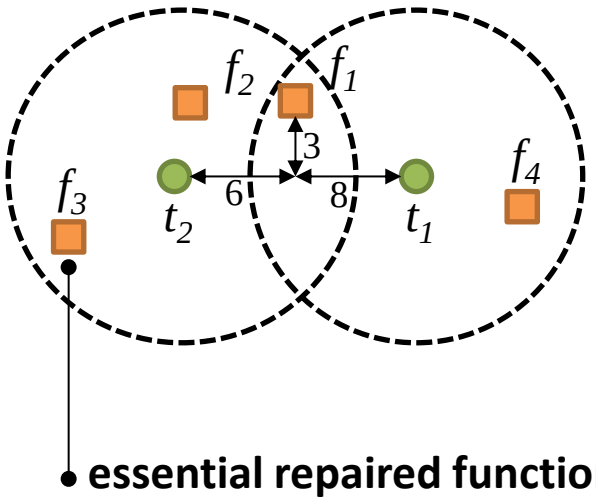
# Functional TSV Repairing Algorithm

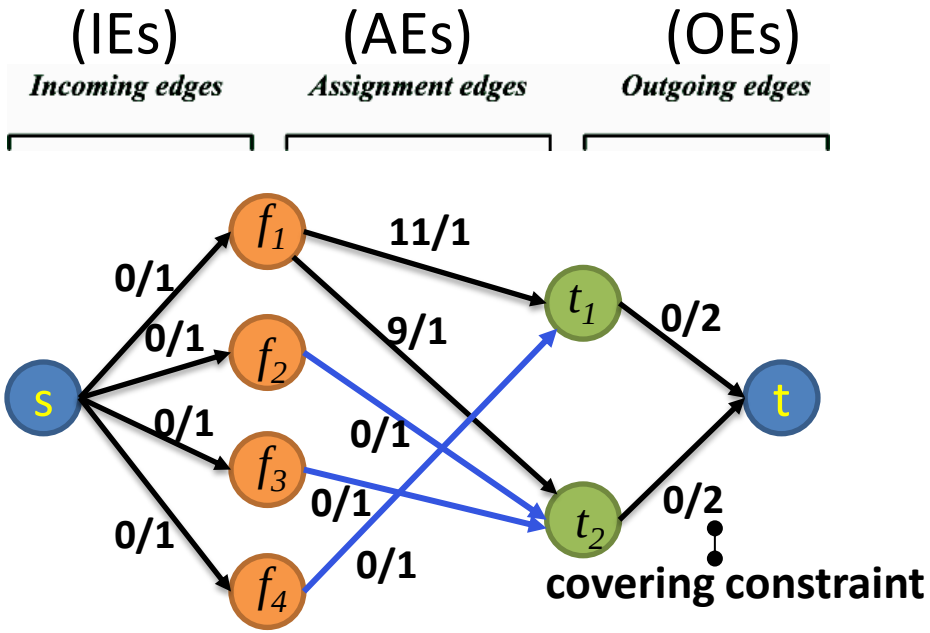


# Functional TSV Assignment

- Each functional TSV should be assigned.
- We model the problem as a **network-flow** optimization problem.
  - consider TSV yield (covering constraint)
    - one *test TSV* can cover less than *N* functional TSVs
  - consider repairing wire length
    - assign functional TSV to nearest test TSV
- Solution: minimum cost maximum flow

# Functional TSV Assignment (Example)




Edge: cost / capacity

**Cost**

**IEs** and **OEs** are set to 0

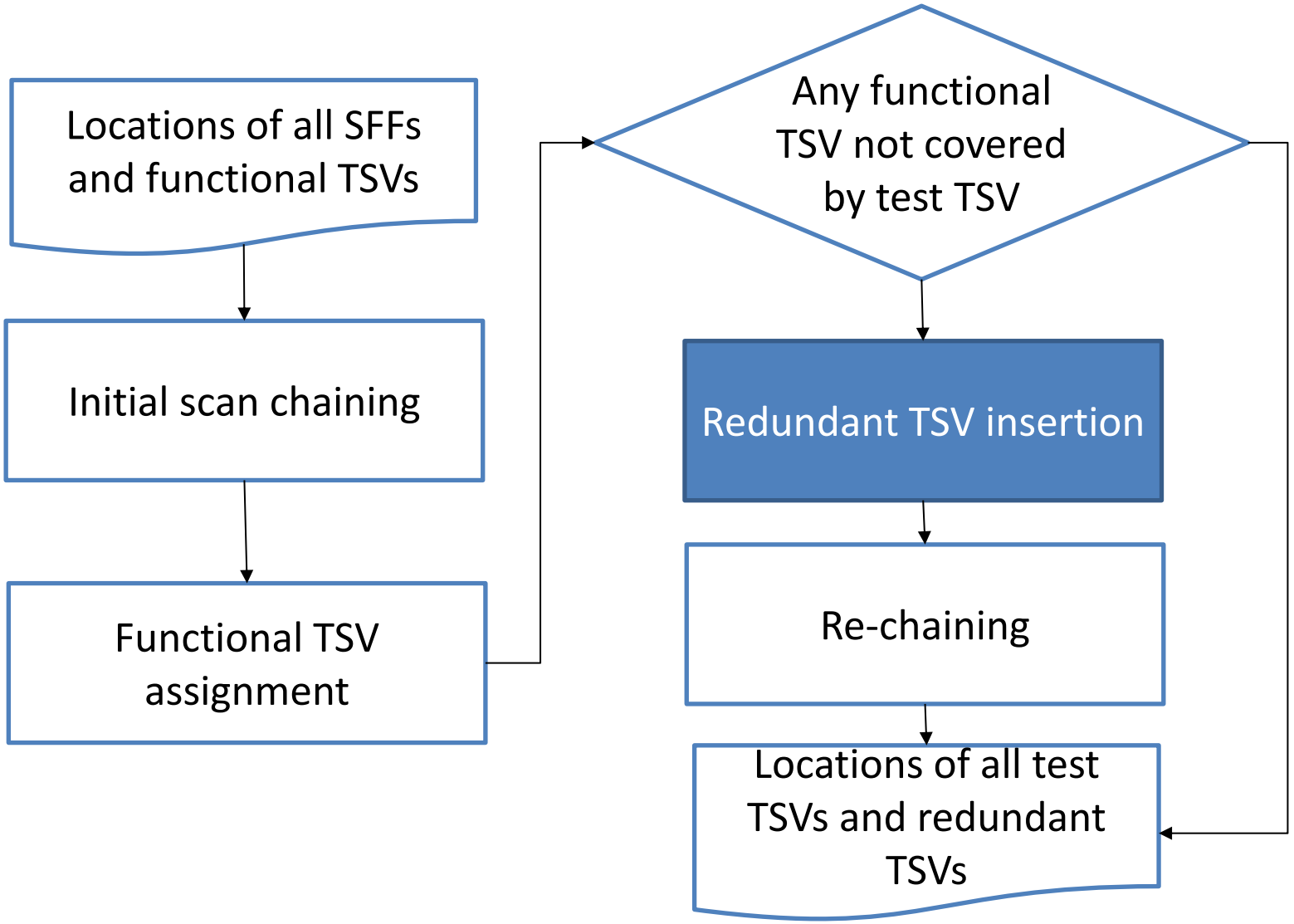
**AEs:** **if functional TSV is essential** set to 0
**else** set to HPWL between functional TSV and test TSV

**Capacity**

**IEs** and **AEs** are set to 1

**OEs** are set to covering constraint

# Functional TSV Repairing Algorithm



# Redundant TSV Insertion

- Clustering algorithm is performed by *K*-means tier by tier
  - cluster all unassigned functional TSVs into groups
- Do *K*-means recursively
- One redundant TSV is inserted for one cluster
- The setting of *K*
  - if there are $n$ functional TSVs and covering constraint = M
  - $K = \left\lceil \frac{n}{M} \right\rceil$

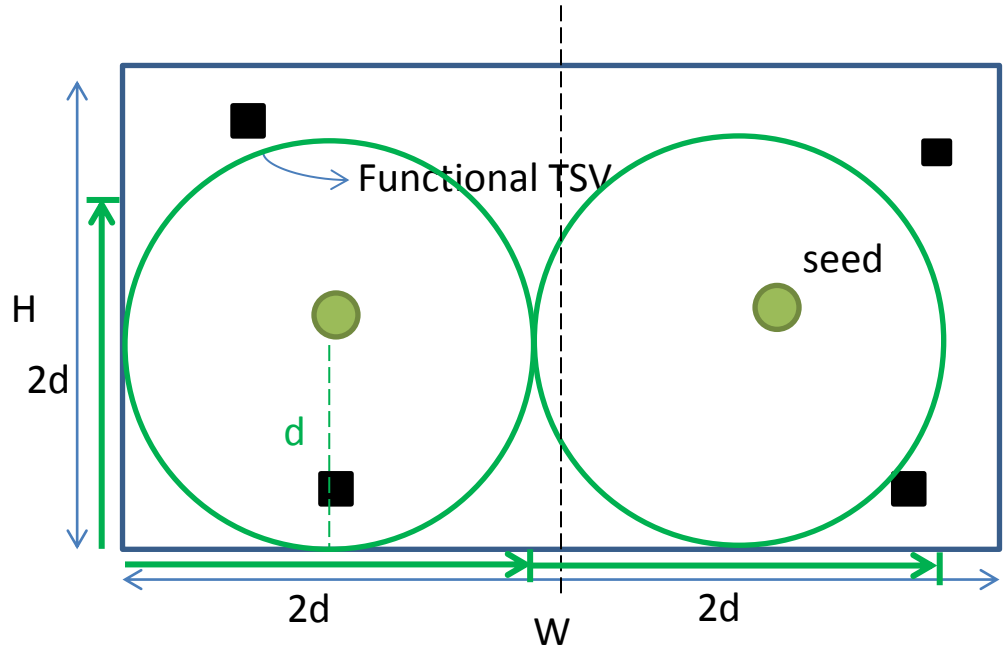
# One Problem for Setting of K

- Covering constraint = 10

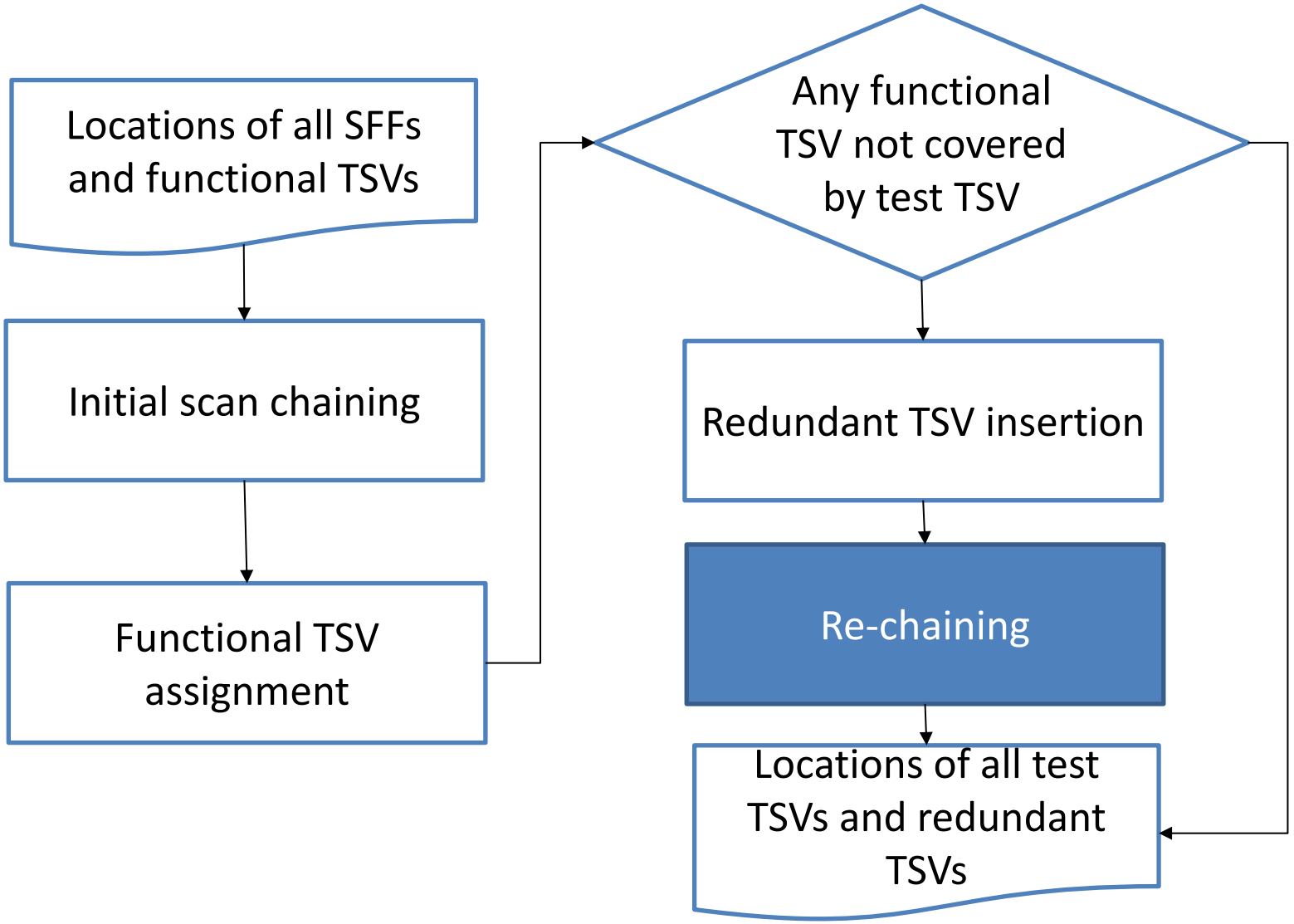
~~K = 1~~ ?

Repairing WL can not control

$$K = \left\lfloor \frac{W}{2d} \right\rfloor \times \left\lfloor \frac{H}{2d} \right\rfloor = 2$$



# Functional TSV Repairing Algorithm



# Re-chaining

- All the functional TSVs can be repaired by a *test TSV* or a redundant TSV
- To **reduce wire length** of the scan chain, re-chaining the scan chain
- Only the wire length is modeled as the cost
- Update the role of *test TSV* and redundant TSV

# Outline

- Introduction
- Our Proposed Architecture
- Problem Definition & Our Proposed Algorithm
- Experiment
- Conclusions

# Experiment Setup

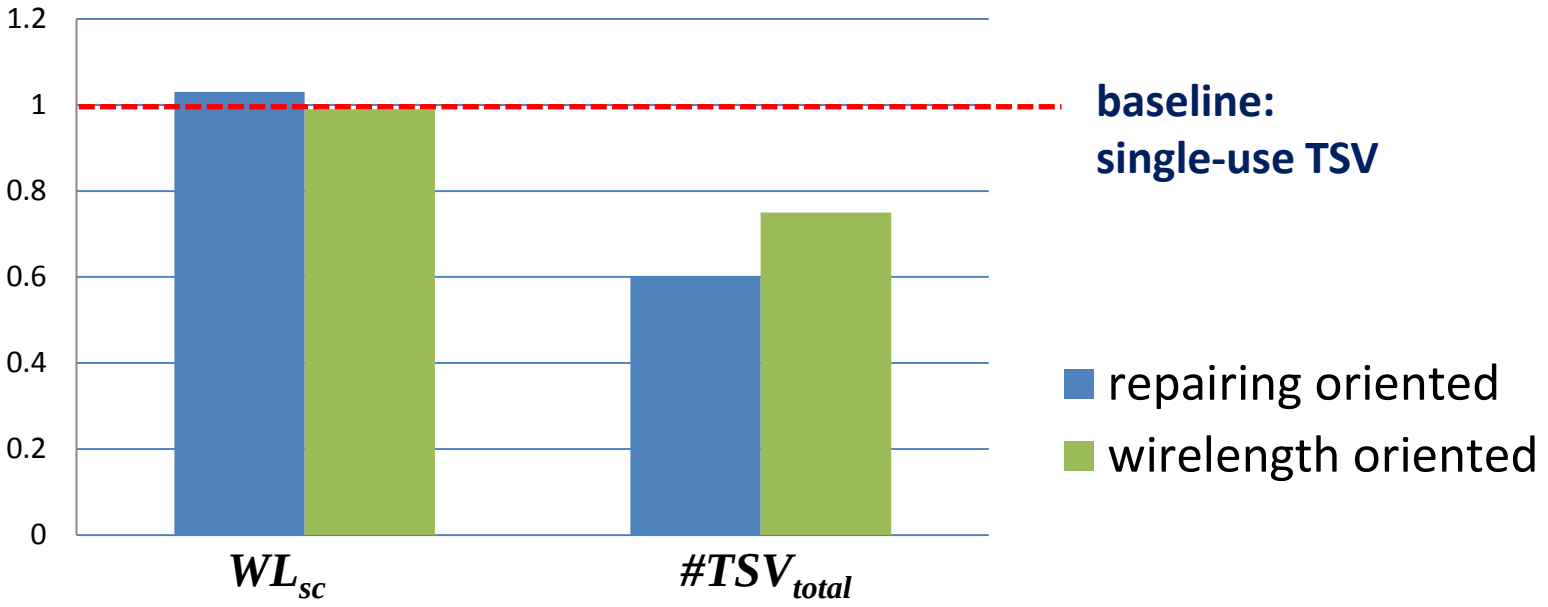
- Using C++/STL programming language with LEDA library
- **OpenCore** benchmark circuits are synthesized by using DesignCompiler™ with **45-*nm*** technology library
- 3-D IC placement tool from UCLA laboratory
  - 4-tiers
- Scan-chain odering (TSP-based)
  - K. D. Boese et. al., ``Scan Chain Optimization: Heuristic and Optimal Solutions``, 1994
- Weight factors, $\alpha$, $\beta$, $and$ $\gamma$, are set to $1$, $1$, and $0.01$, respectively
- Covering constraint = 50 (maximum number of functional TSVs are covered by one test TSV)

# Comparisons of Architecture

| circuit name | single-use TSV | | | | | dual-use TSV | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $WL_{SC}$ | $\#TSV_r$ | $\#TSV_t$ | $\#TSV_{r/t}$ | $\#TSV_{total}$ | $WL_{SC}$ | $\#TSV_r$ | $\#TSV_t$ | $\#TSV_{r/t}$ | $\#TSV_{total}$ |
| **openMSP430** | 8680261 | 19 | 8 | 0 | 27 | 9034431 | 10 | 0 | 8 | 18 |
| **tv80_core** | 5008458 | 31 | 13 | 0 | 44 | 5151862 | 20 | 0 | 9 | 29 |
| **fft** | 10486698 | 12 | 10 | 0 | 22 | 10852080 | 5 | 0 | 10 | 15 |
| **lcd_ctrl** | 10512128 | 34 | 19 | 0 | 53 | 10580498 | 21 | 0 | 9 | 30 |
| **pci_controller** | 11795747 | 13 | 10 | 0 | 23 | 12735565 | 1 | 1 | 9 | 11 |
| **ac97** | 20376028 | 18 | 12 | 0 | 30 | 20415944 | 9 | 0 | 8 | 17 |
| **usb2.0** | 18473980 | 23 | 9 | 0 | 32 | 18554764 | 13 | 0 | 7 | 20 |
| **sub_x86_cpu** | 4610518 | 22 | 16 | 0 | 38 | 4825646 | 10 | 0 | 10 | 20 |
| **ratio** | 1 | | | | 1 | **1.03** | | | | **0.6** |

- almost all test TSVs can be used as redundant TSVs
- avg. improvement of the total number of TSVs is 40%
- avg. routing resource overhead of 3-D IC scan-chain is 3%

# Comparisons of Algorithm




- Repairing oriented
  - consider repairing cost
- Wire length oriented
  - optimizes the wire length w/o repairing cost
  - step1: clustering all functional TSVs
  - step2: find a test TSV nearby or insert redundant TSV

# Outline

- Introduction
- Our Proposed Architecture
- Problem Definition & Our Proposed Algorithm
- Experiment
- Conclusions

# Conclusions

- We have proposed an architecture of TSV recovery by using scan-chain test TSV
- 40% less number of total TSVs with 3% wiring overhead

# Thank you