Test and Security

G. Di Natale, M.L. Flottes, B. Rouzeyre

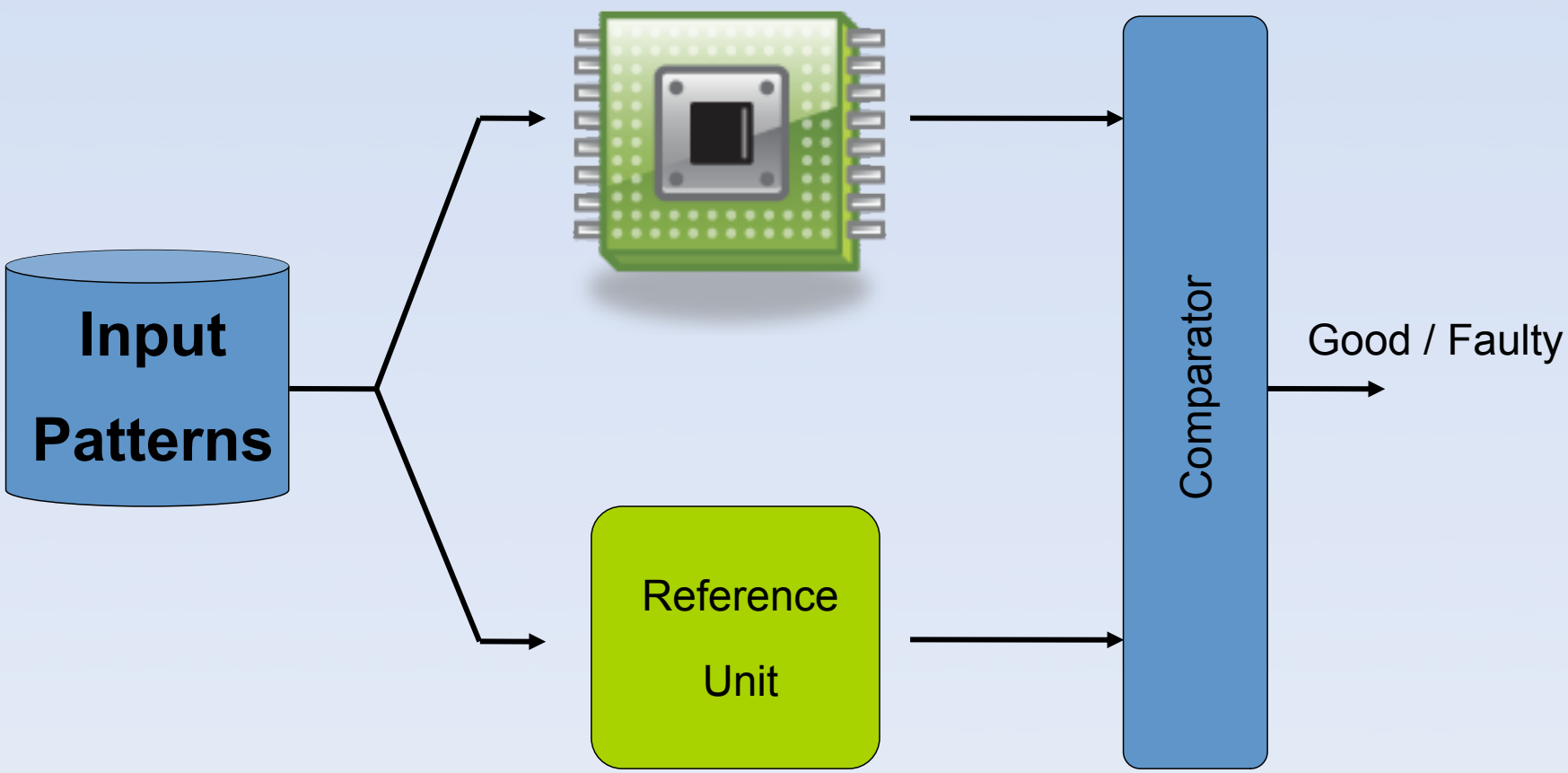
Outline

- Test
- Security vs Test
  - Scan-based attacks
  - Securing the scan chains
  - BIST as an alternative
- Conclusions

Test

- Test: set of operations aiming at checking whether a manufactured unit properly works w.r.t. its specifications or not
- Test cost: very high (~30% of the whole IC cost)

Test: basic principles

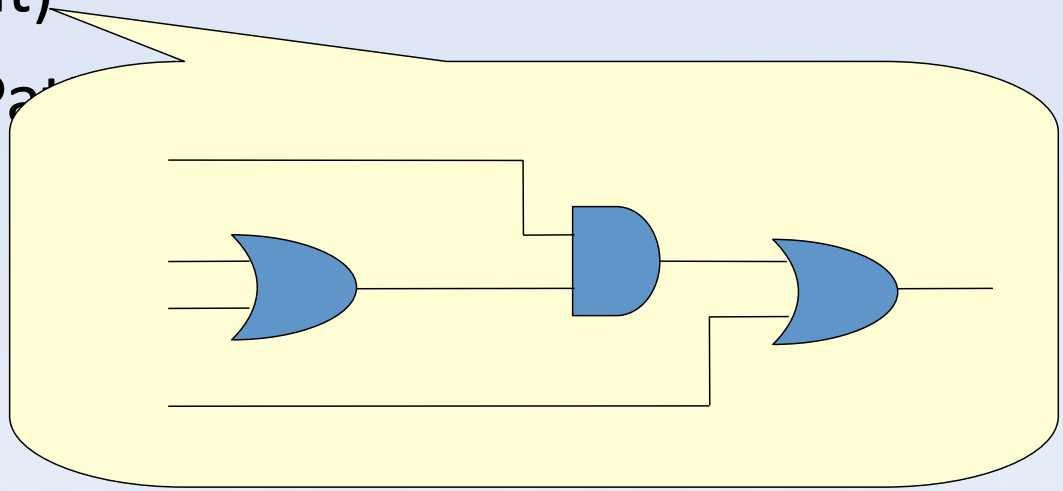


How to generate input patterns?

- Exhaustive test
- Functional test
- Random/Pseudo-Random test
- Structural test
  - Netlist
  - Fault model (e.g., stuck-at)
  - Deterministic TPG (Test Pattern Generation)

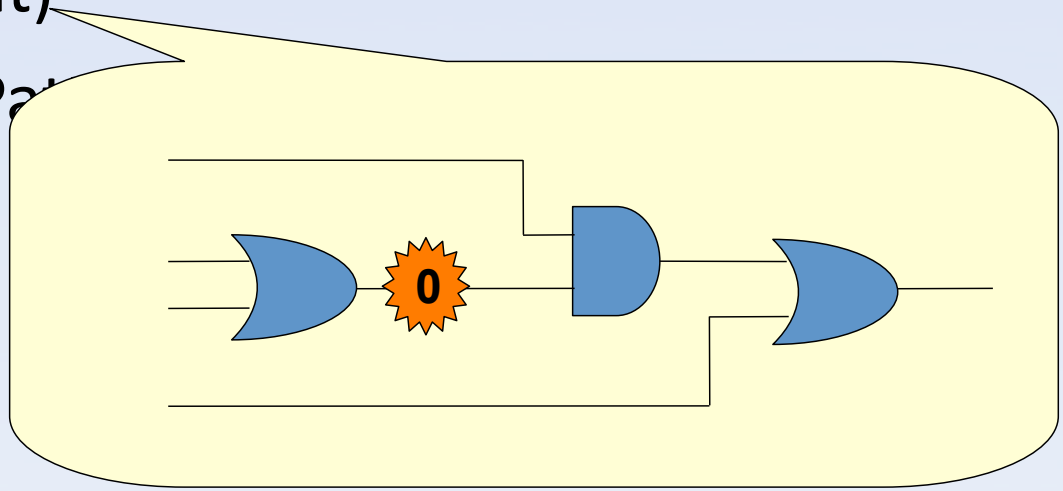
How to generate input patterns?

- Exhaustive test
- Functional test
- Random/Pseudo-Random test
- Structural test
  - Netlist
  - Fault model (e.g., stuck-at)
  - Deterministic TPG (Test Pa



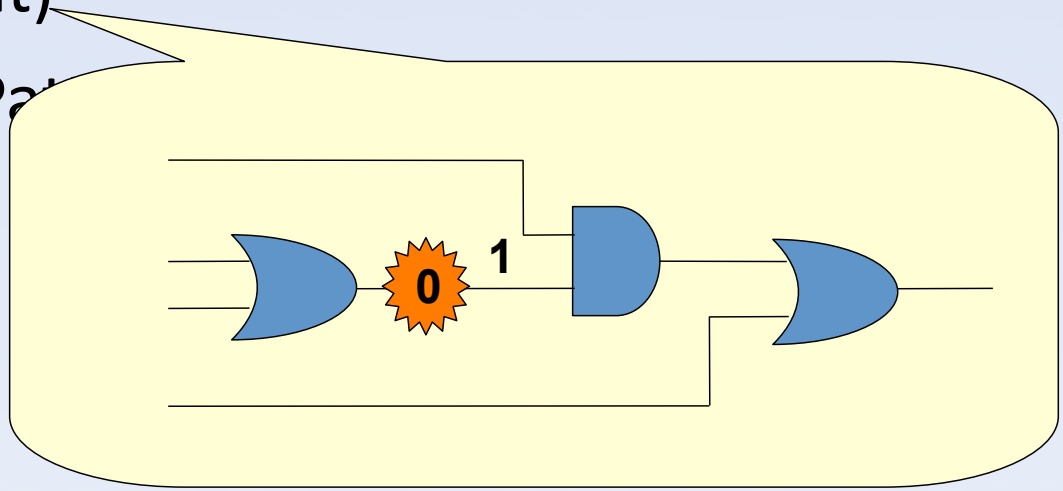
How to generate input patterns?

- Exhaustive test
- Functional test
- Random/Pseudo-Random test
- Structural test
  - Netlist
  - Fault model (e.g., stuck-at)
  - Deterministic TPG (Test Pa



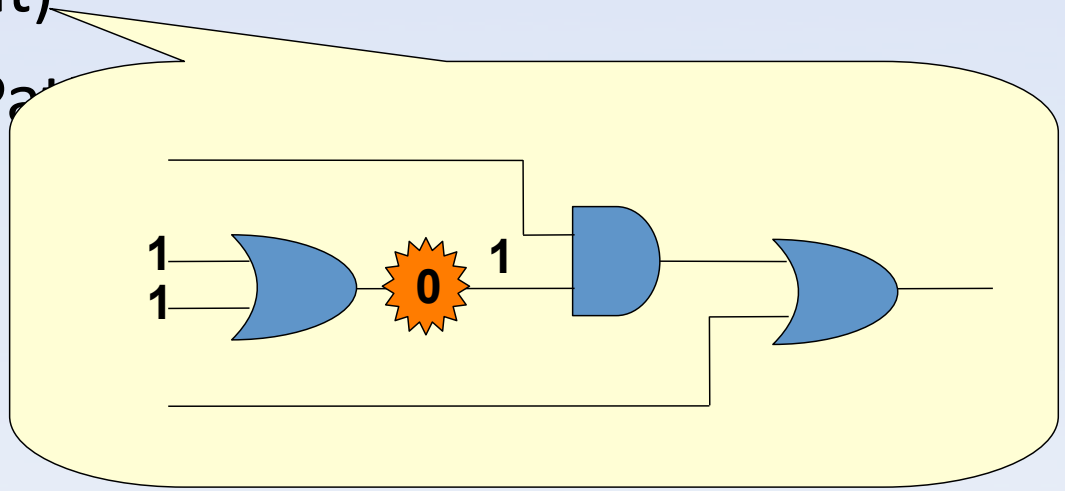
How to generate input patterns?

- Exhaustive test
- Functional test
- Random/Pseudo-Random test
- Structural test
  - Netlist
  - Fault model (e.g., stuck-at)
  - Deterministic TPG (Test Pa



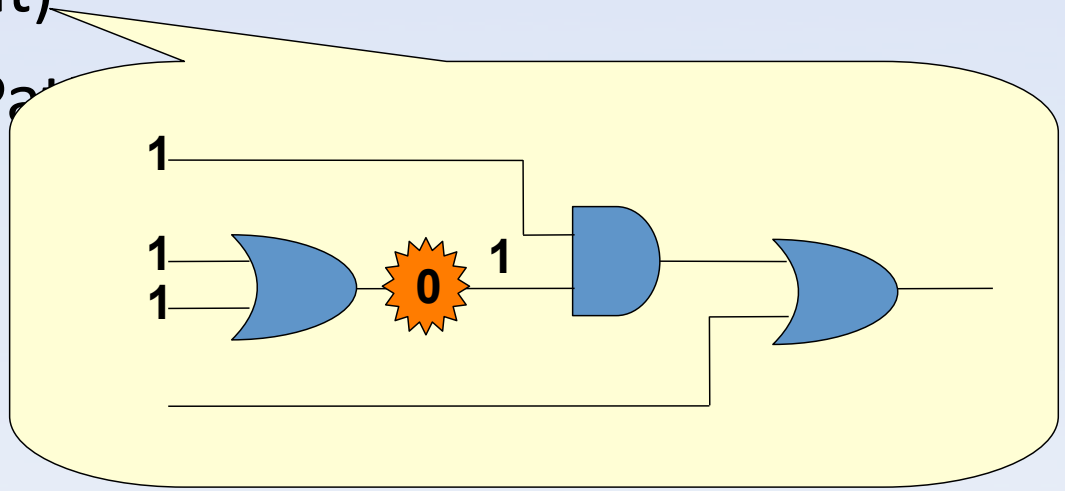
How to generate input patterns?

- Exhaustive test
- Functional test
- Random/Pseudo-Random test
- Structural test
  - Netlist
  - Fault model (e.g., stuck-at)
  - Deterministic TPG (Test Pa



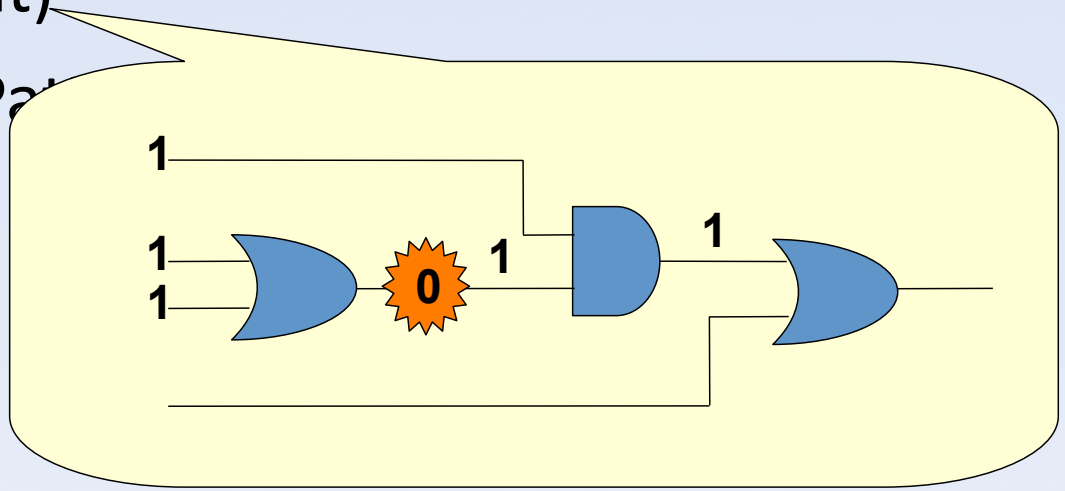
How to generate input patterns?

- Exhaustive test
- Functional test
- Random/Pseudo-Random test
- Structural test
  - Netlist
  - Fault model (e.g., stuck-at)
  - Deterministic TPG (Test Pa



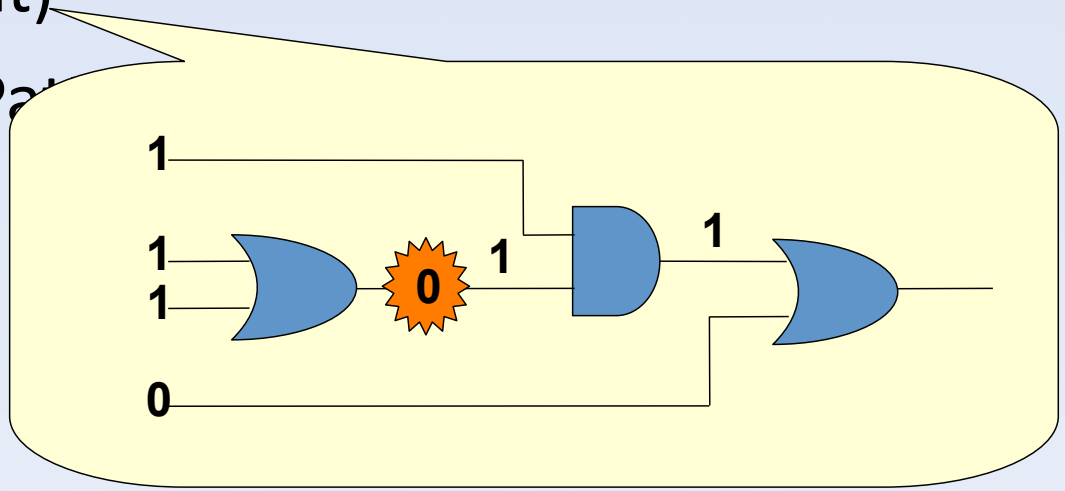
How to generate input patterns?

- Exhaustive test
- Functional test
- Random/Pseudo-Random test
- Structural test
  - Netlist
  - Fault model (e.g., stuck-at)
  - Deterministic TPG (Test Pa



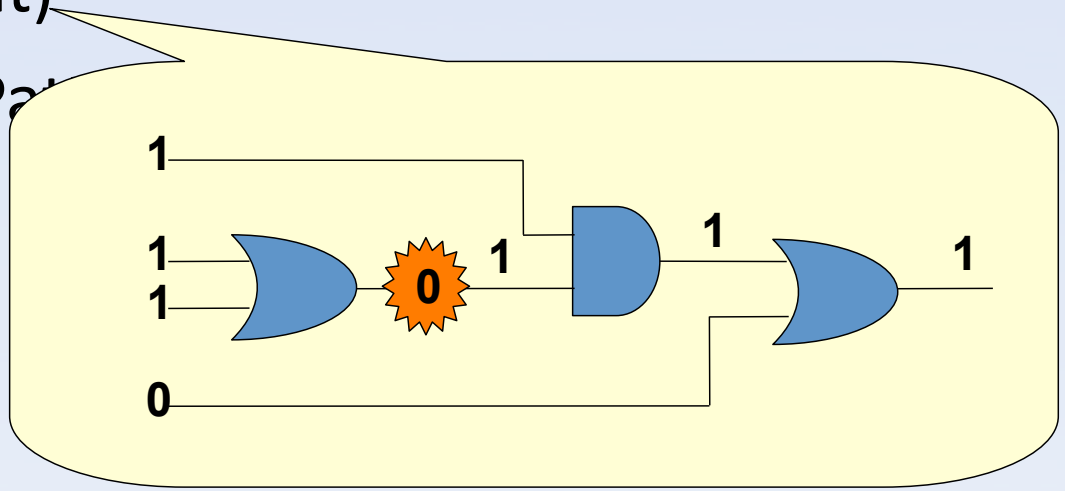
How to generate input patterns?

- Exhaustive test
- Functional test
- Random/Pseudo-Random test
- Structural test
  - Netlist
  - Fault model (e.g., stuck-at)
  - Deterministic TPG (Test Pa



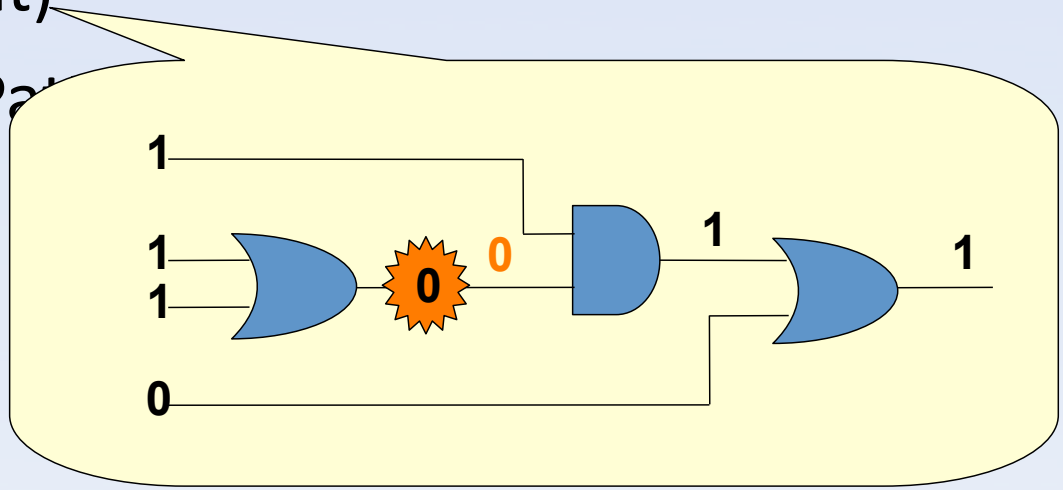
How to generate input patterns?

- Exhaustive test
- Functional test
- Random/Pseudo-Random test
- Structural test
  - Netlist
  - Fault model (e.g., stuck-at)
  - Deterministic TPG (Test Pa



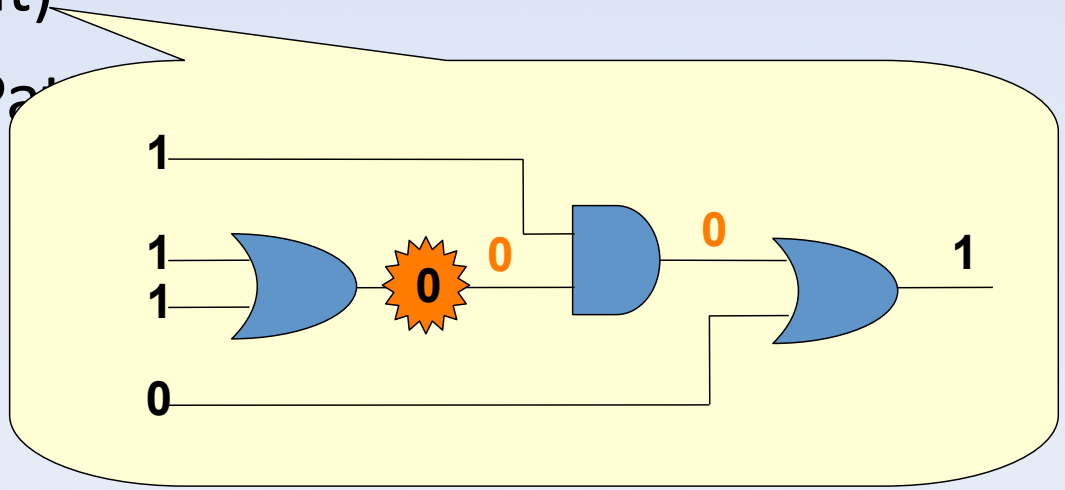
How to generate input patterns?

- Exhaustive test
- Functional test
- Random/Pseudo-Random test
- Structural test
  - Netlist
  - Fault model (e.g., stuck-at)
  - Deterministic TPG (Test Pa



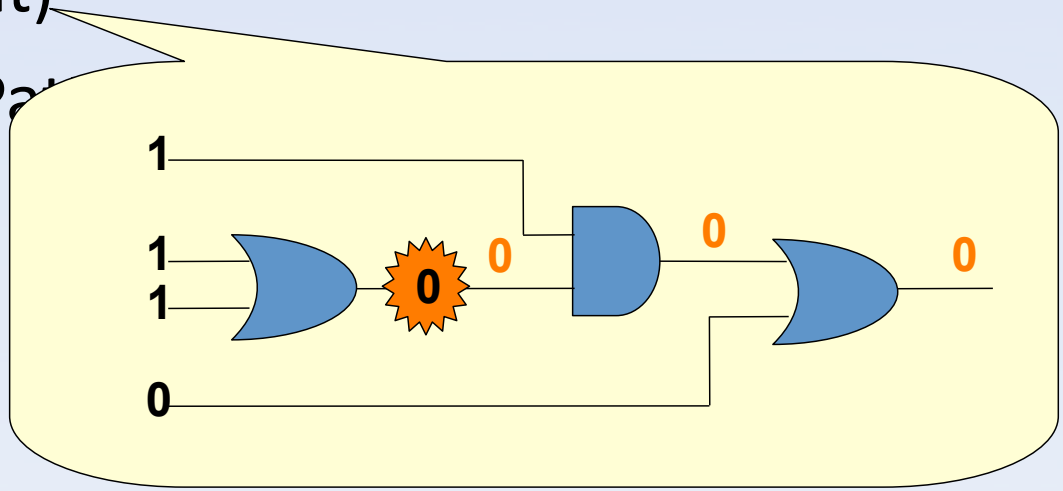
How to generate input patterns?

- Exhaustive test
- Functional test
- Random/Pseudo-Random test
- Structural test
  - Netlist
  - Fault model (e.g., stuck-at)
  - Deterministic TPG (Test Pa



How to generate input patterns?

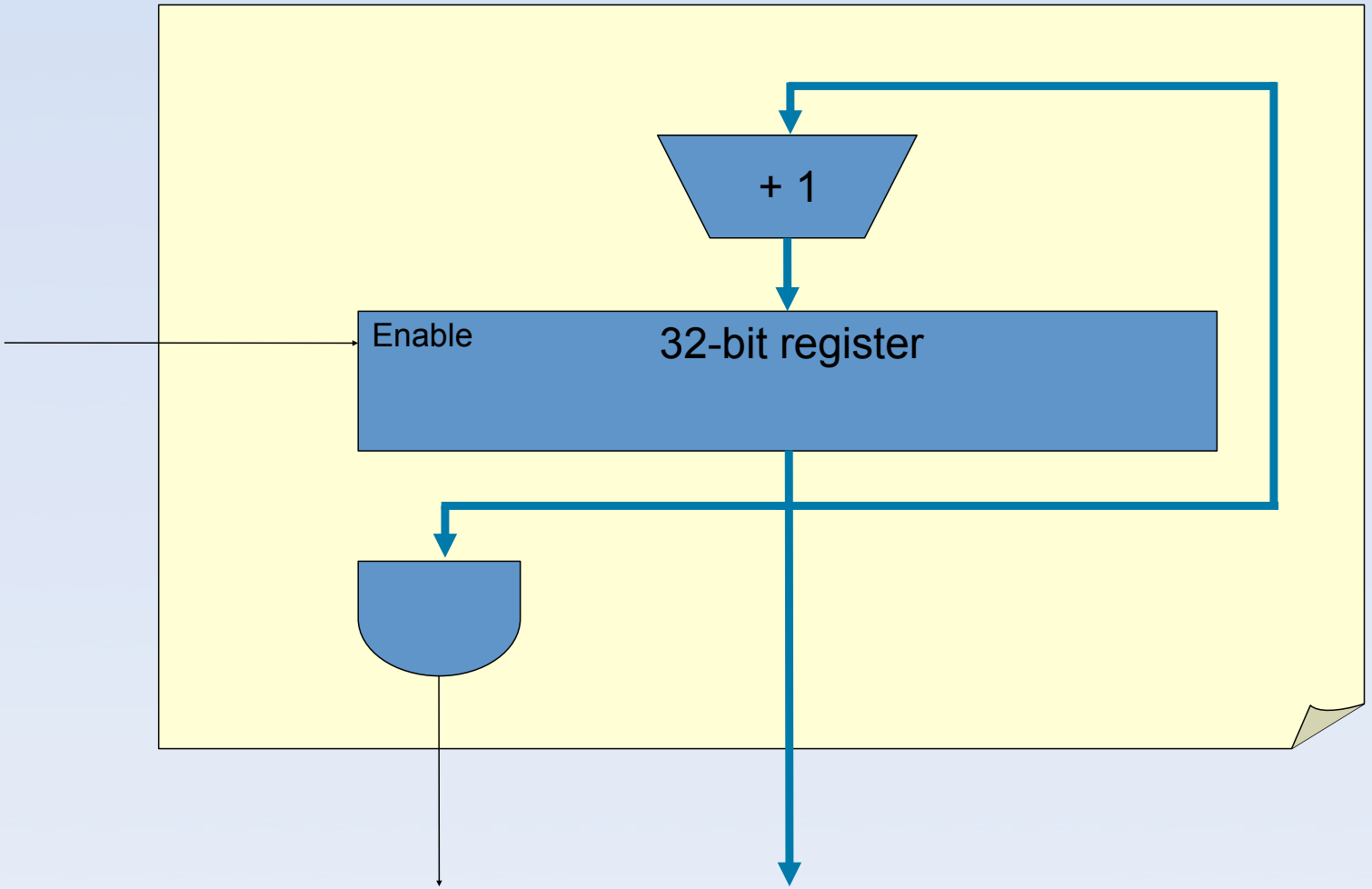
- Exhaustive test
- Functional test
- Random/Pseudo-Random test
- Structural test
  - Netlist
  - Fault model (e.g., stuck-at)
  - Deterministic TPG (Test Pa



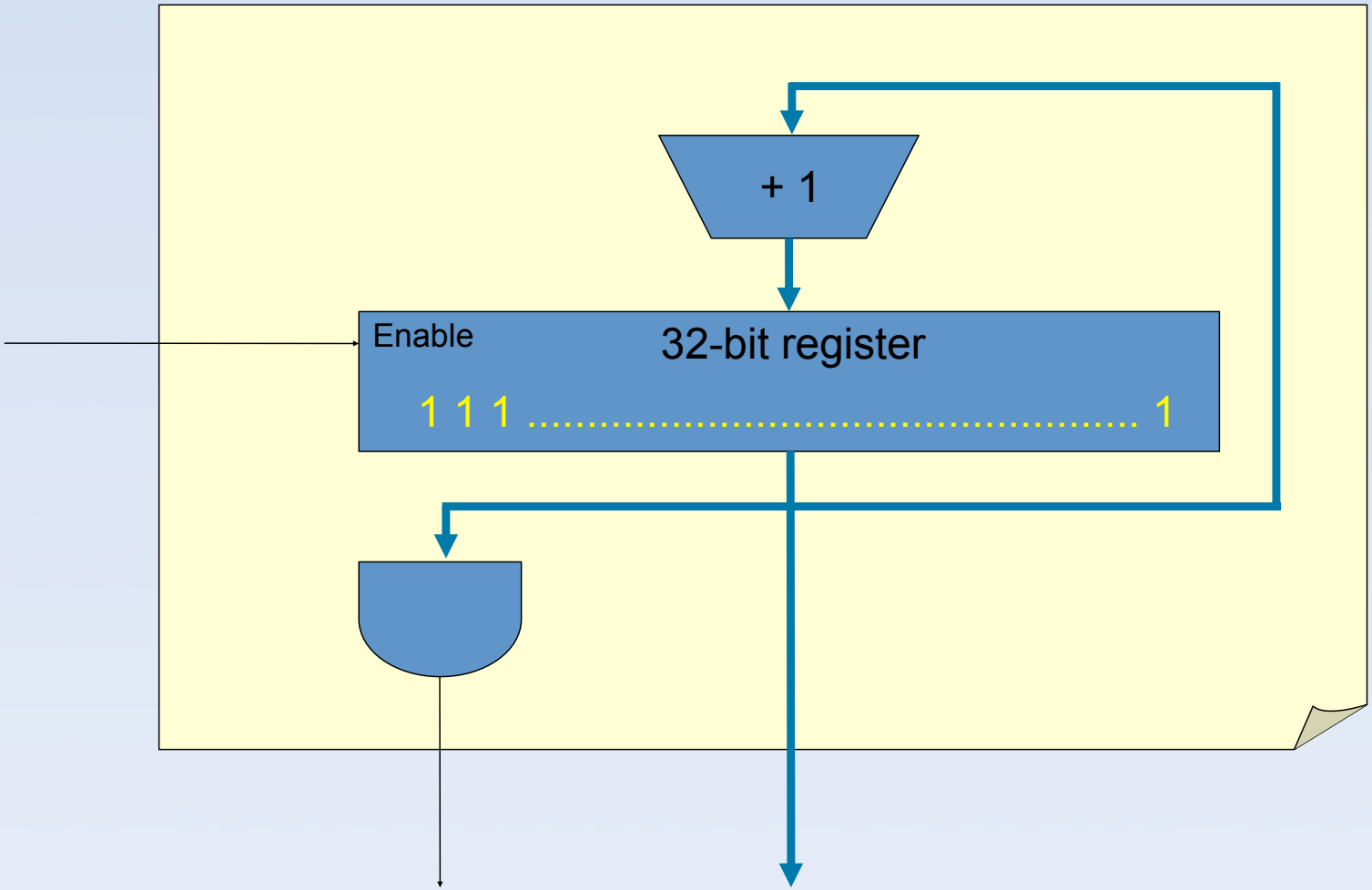
How to generate input patterns?

- Exhaustive test
- Functional test
- Random/Pseudo-Random test
- Structural test
  - Netlist
  - Fault model (e.g., stuck-at)
  - Deterministic TPG (Test Pattern Generation)

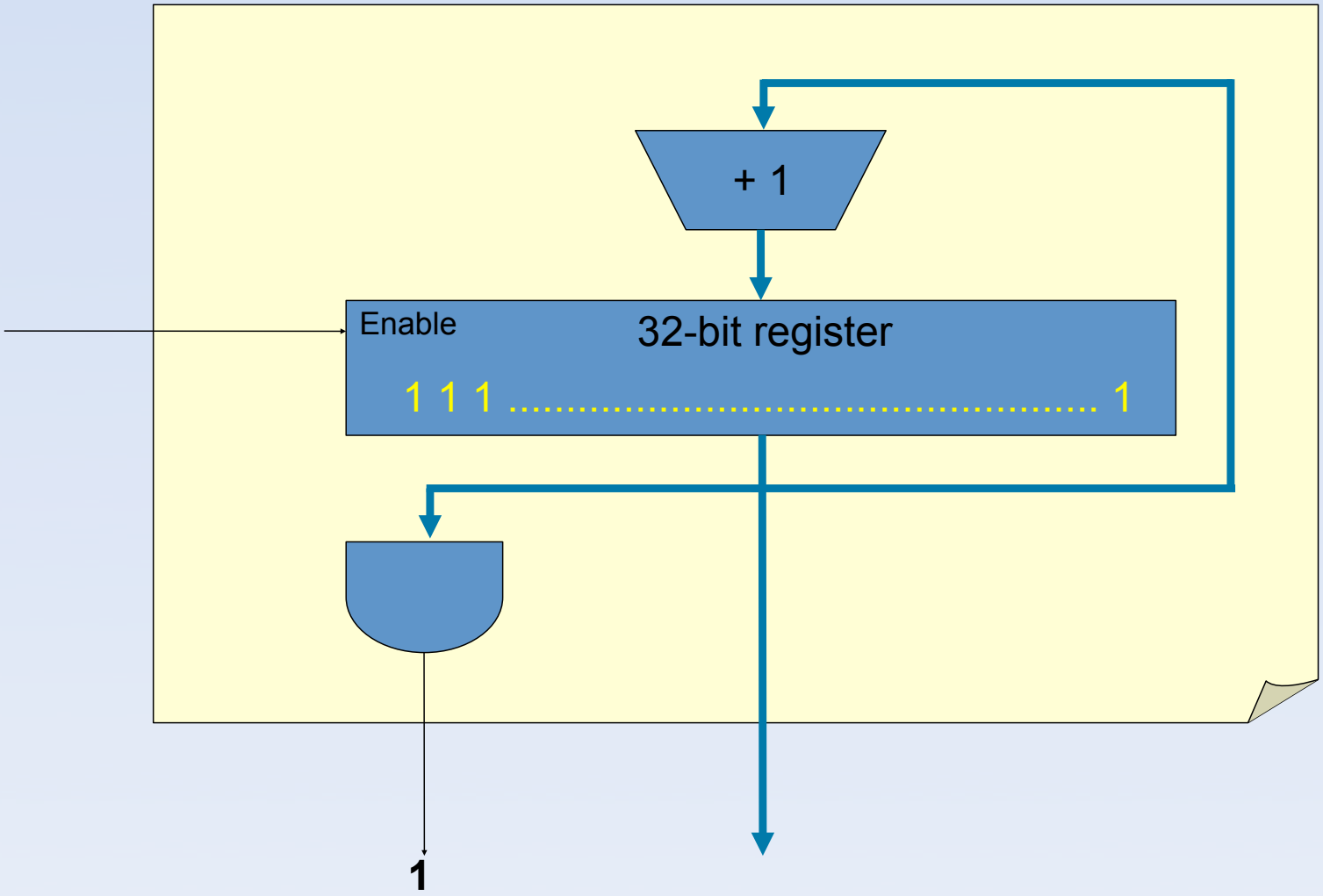
Sequential circuits



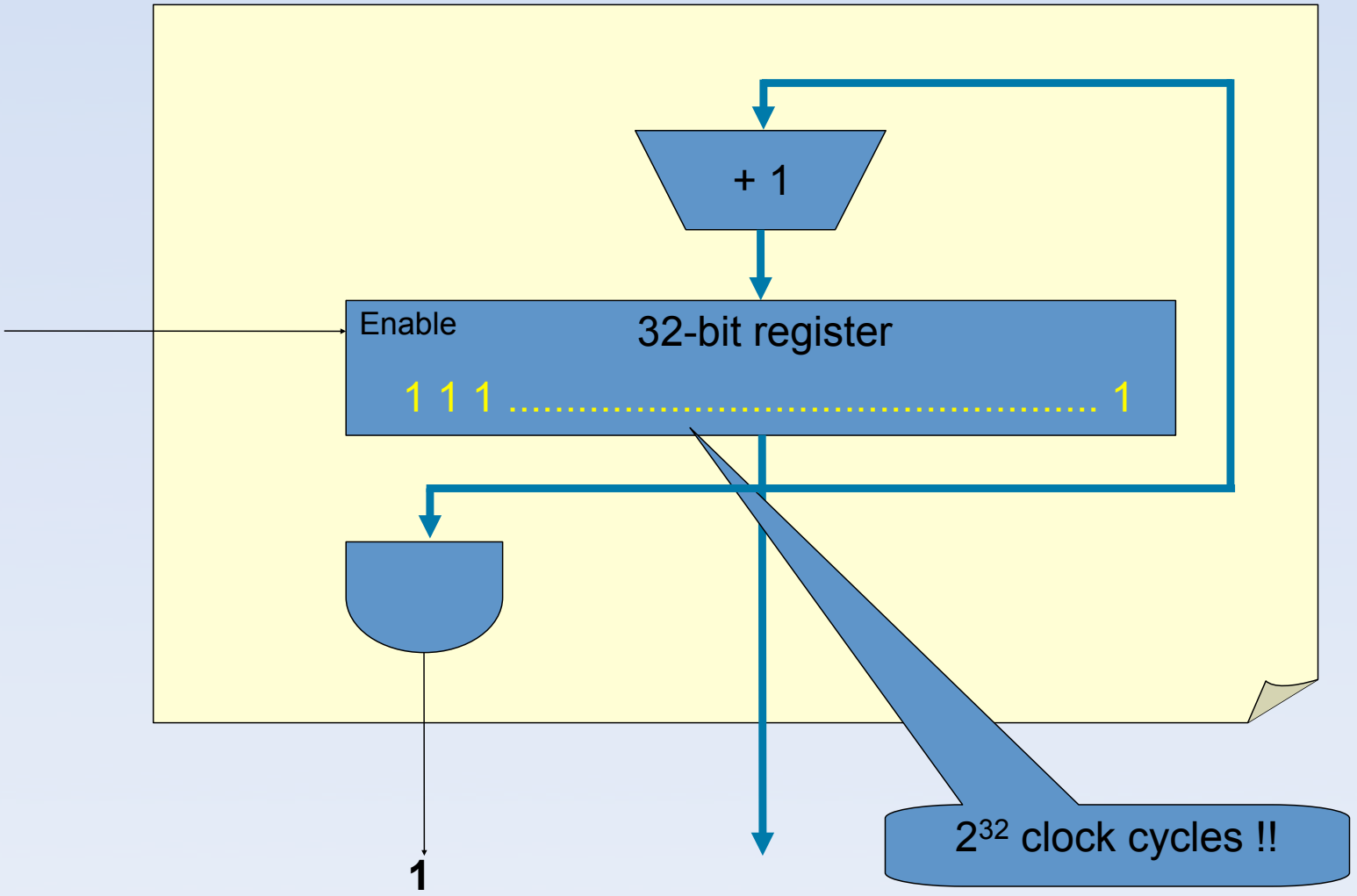
Sequential circuits



Sequential circuits



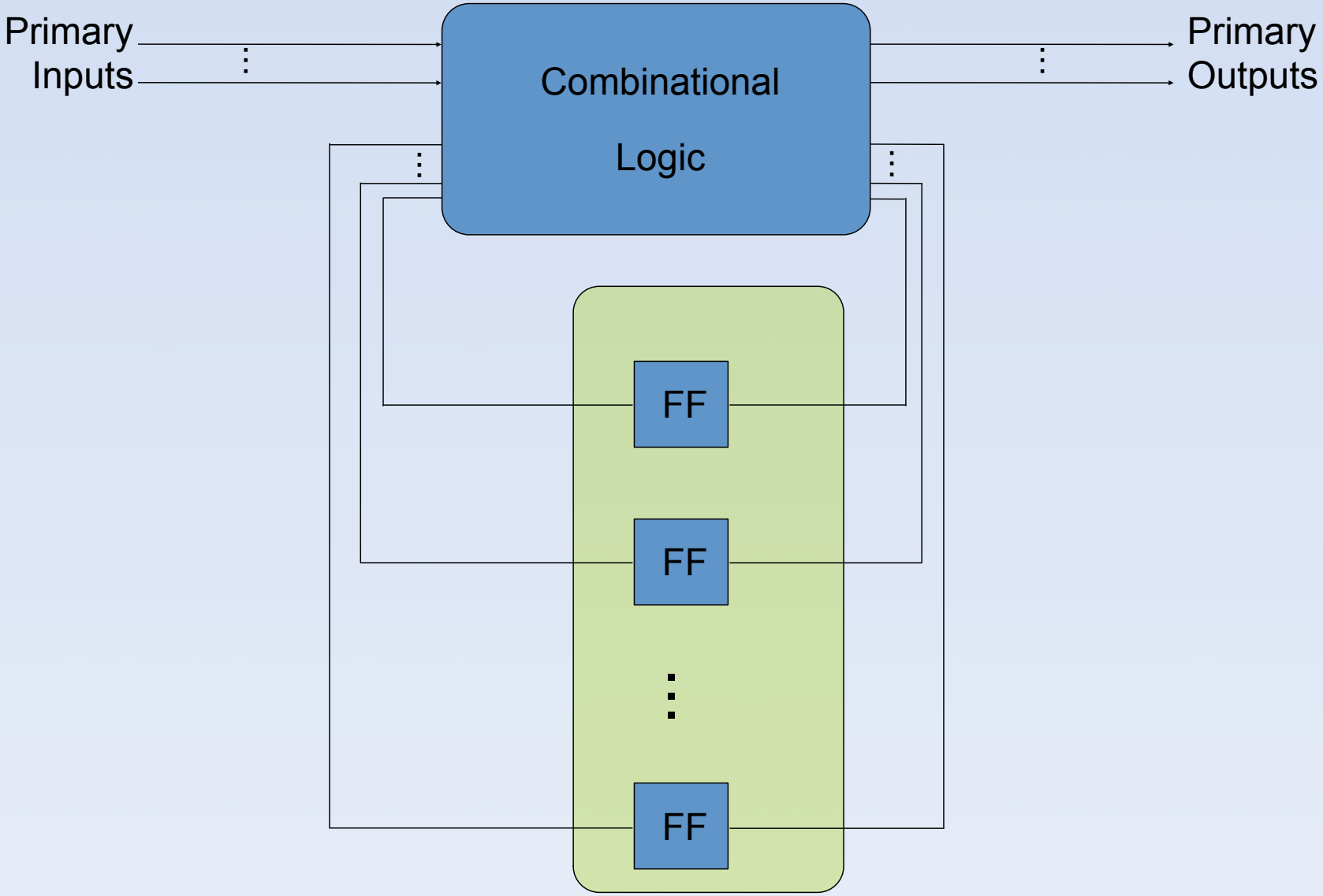
Sequential circuits



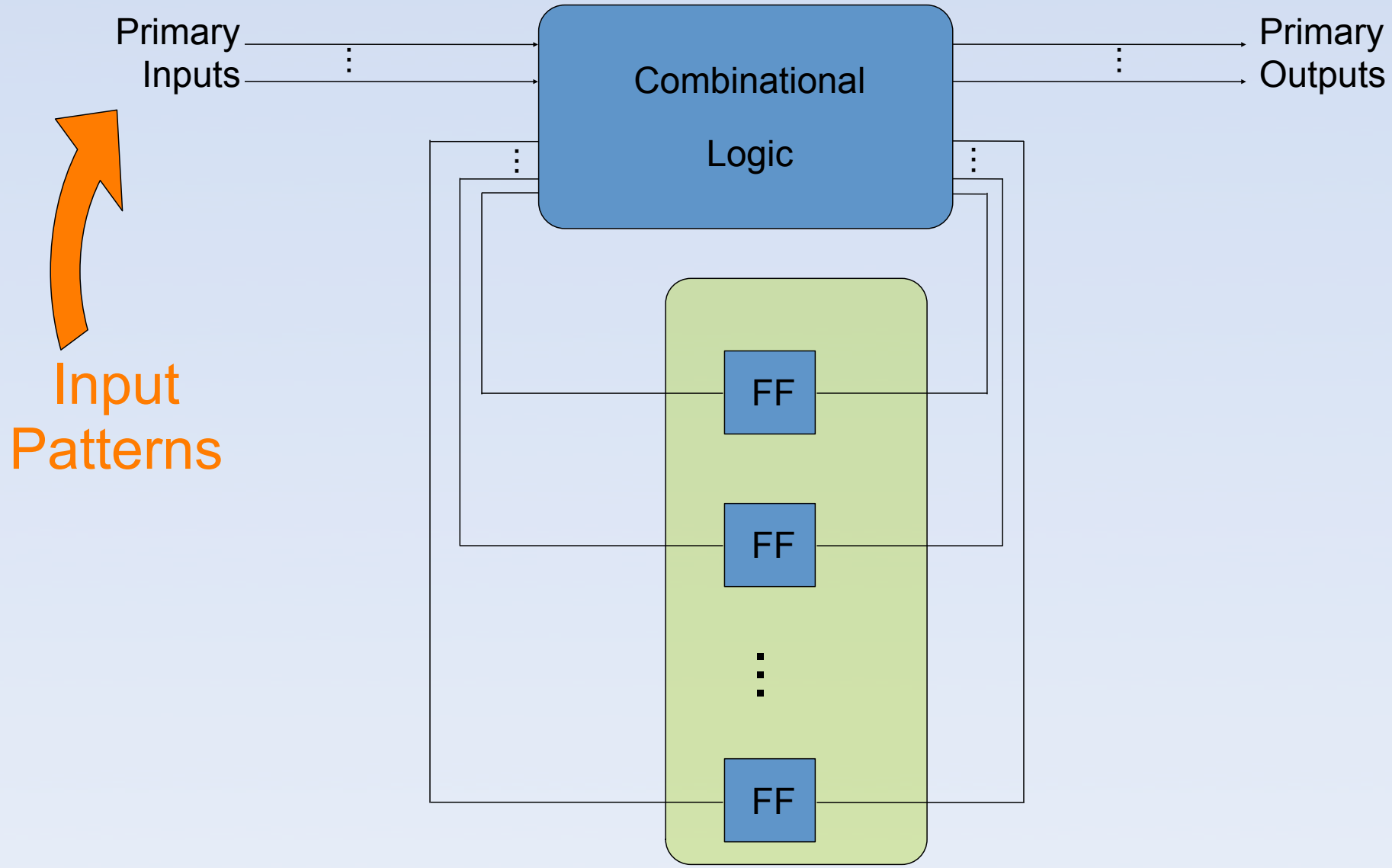
Design for Testability (DfT)

- Design techniques that add testability features to a device
- Goal: to increase controllability and observability
- Techniques:
  - Scan chains for sequential circuits
  - Built-In Self-Test

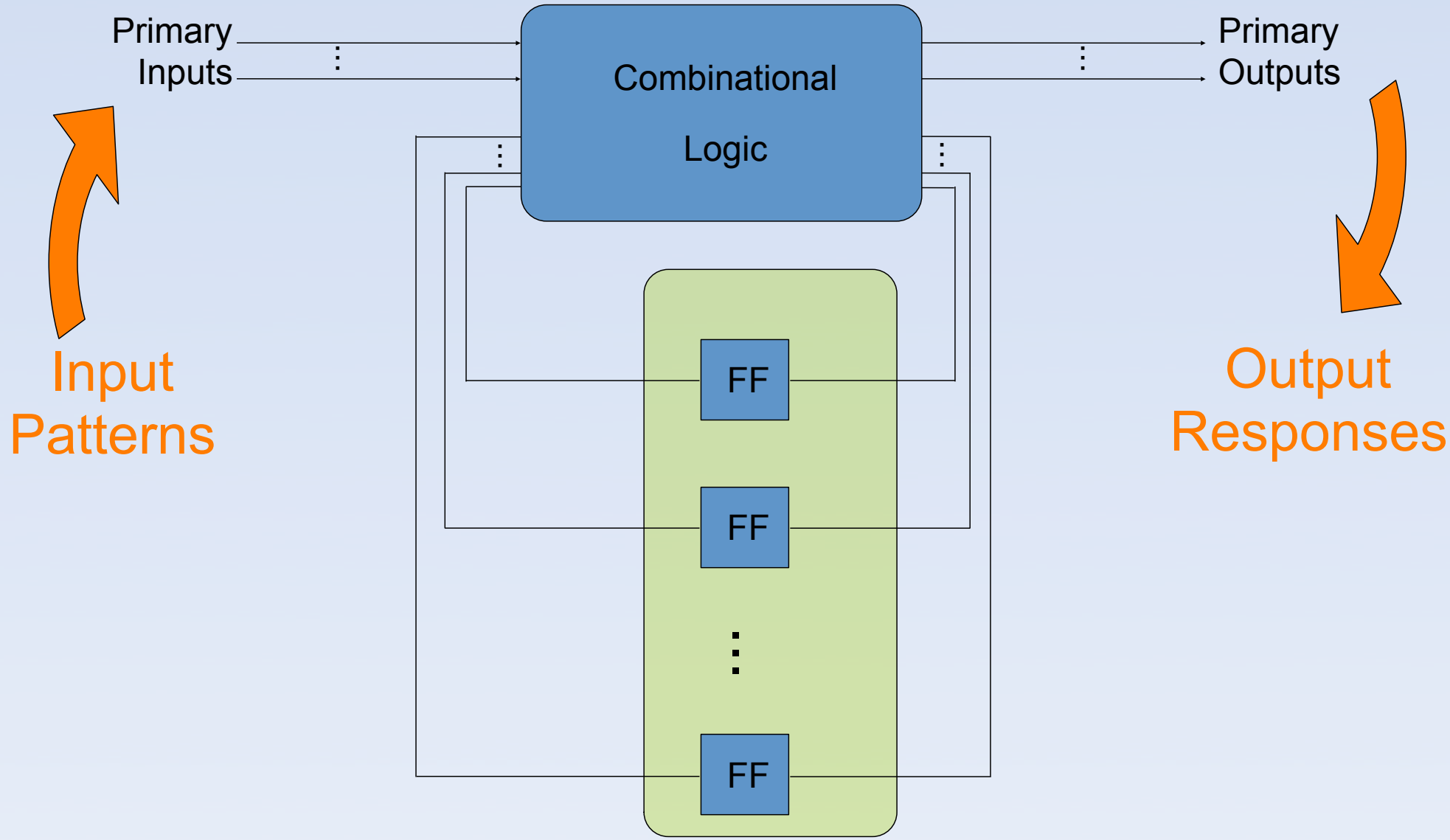
Scan-based Design



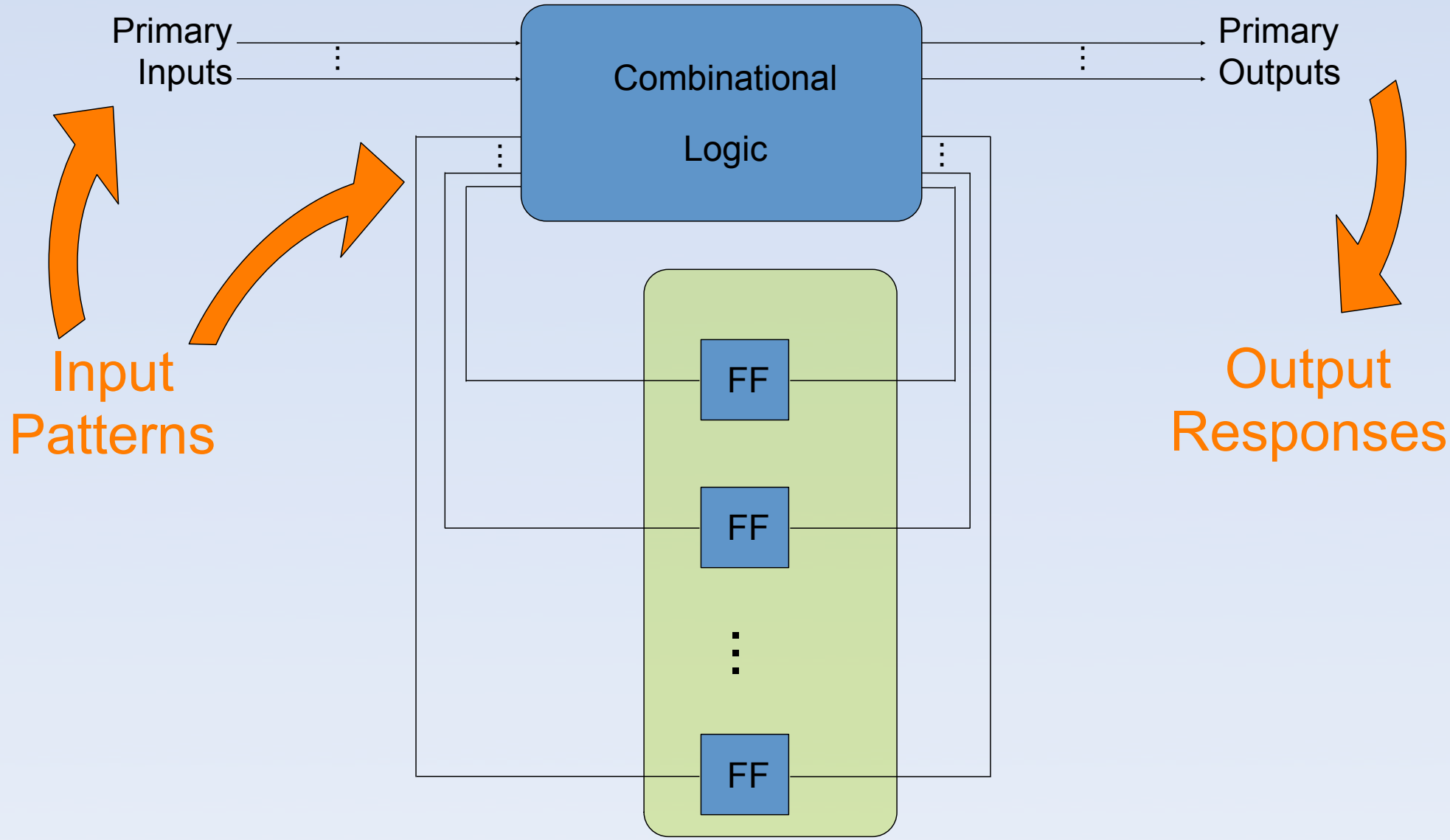
Scan-based Design



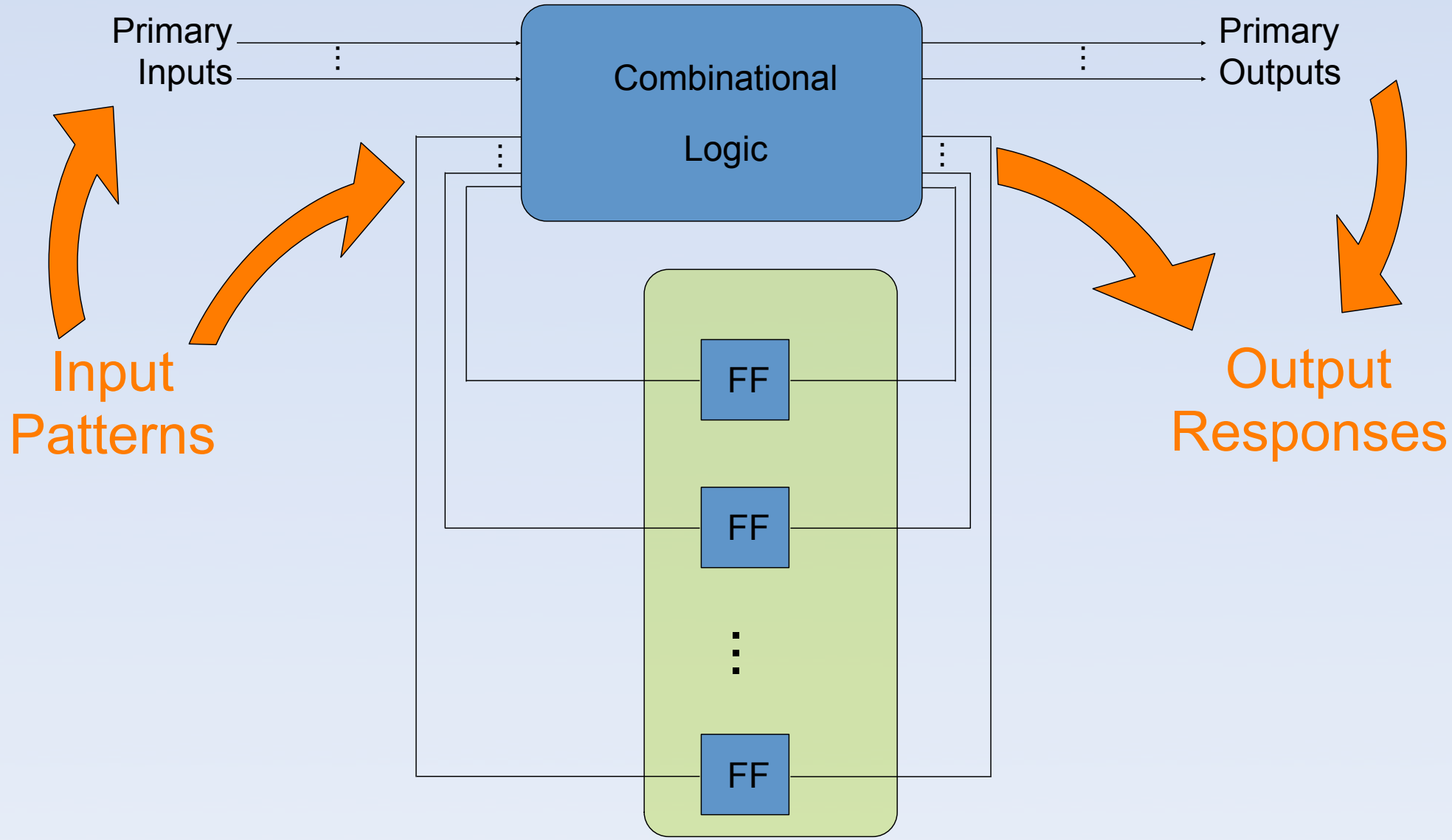
Scan-based Design



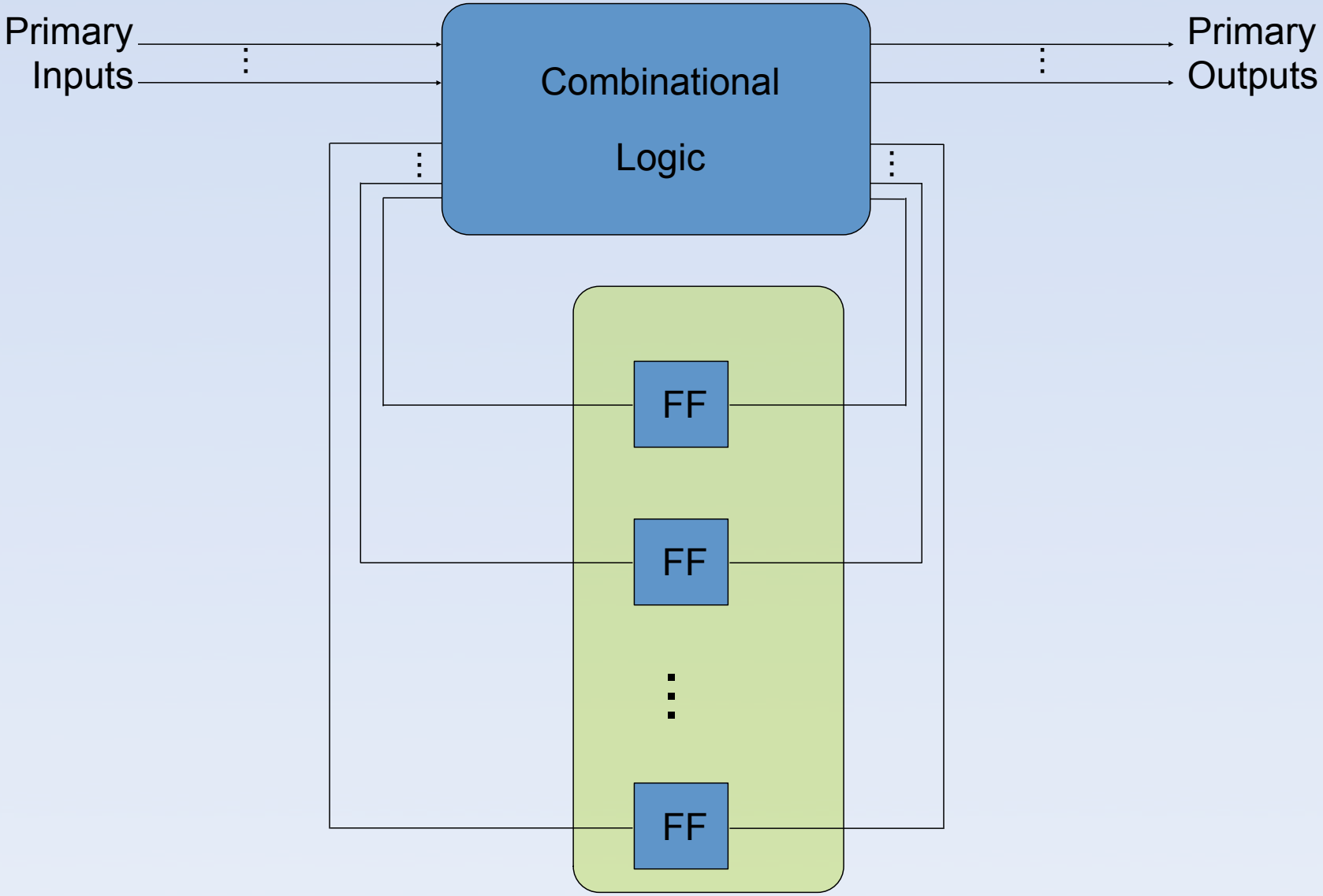
Scan-based Design



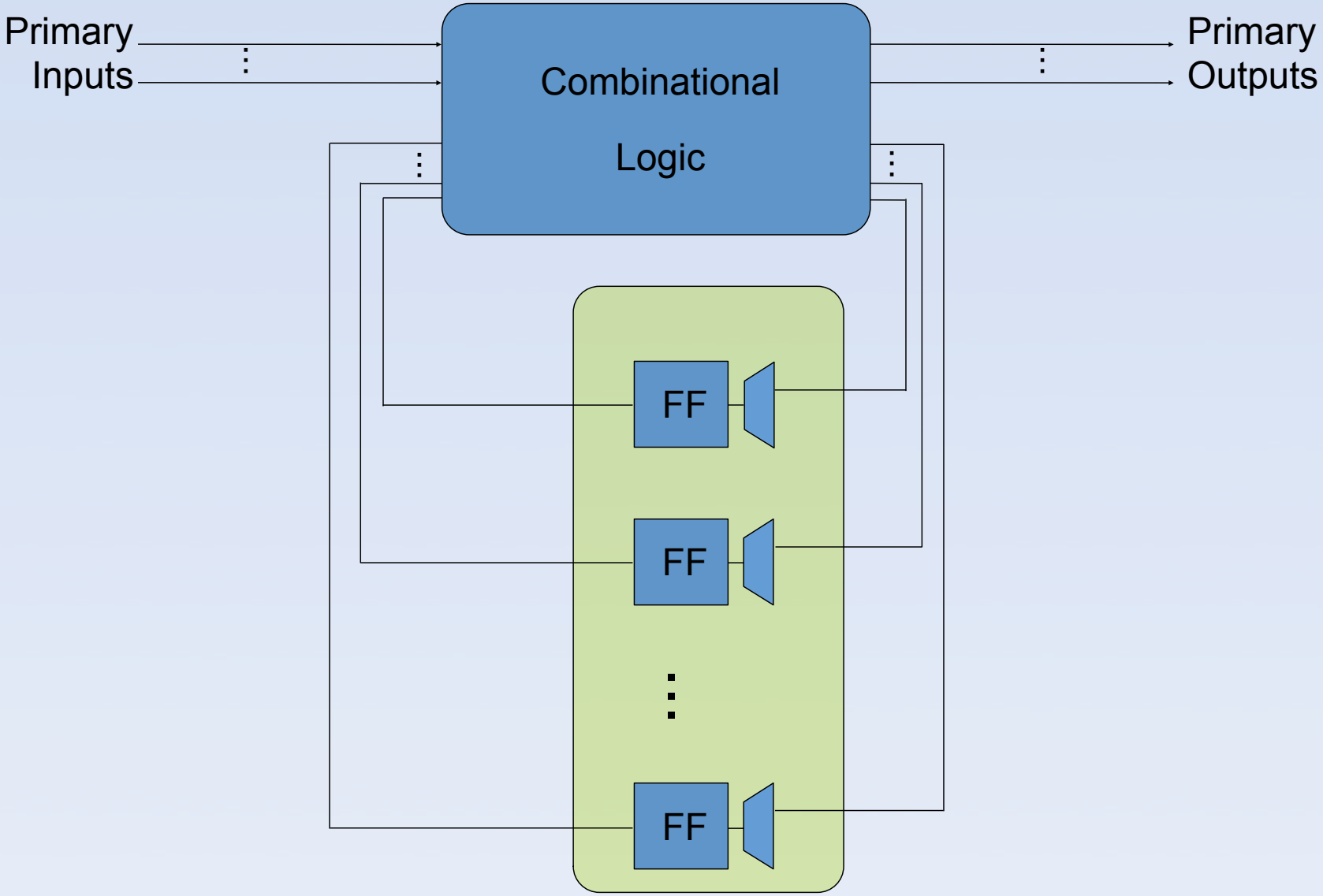
Scan-based Design



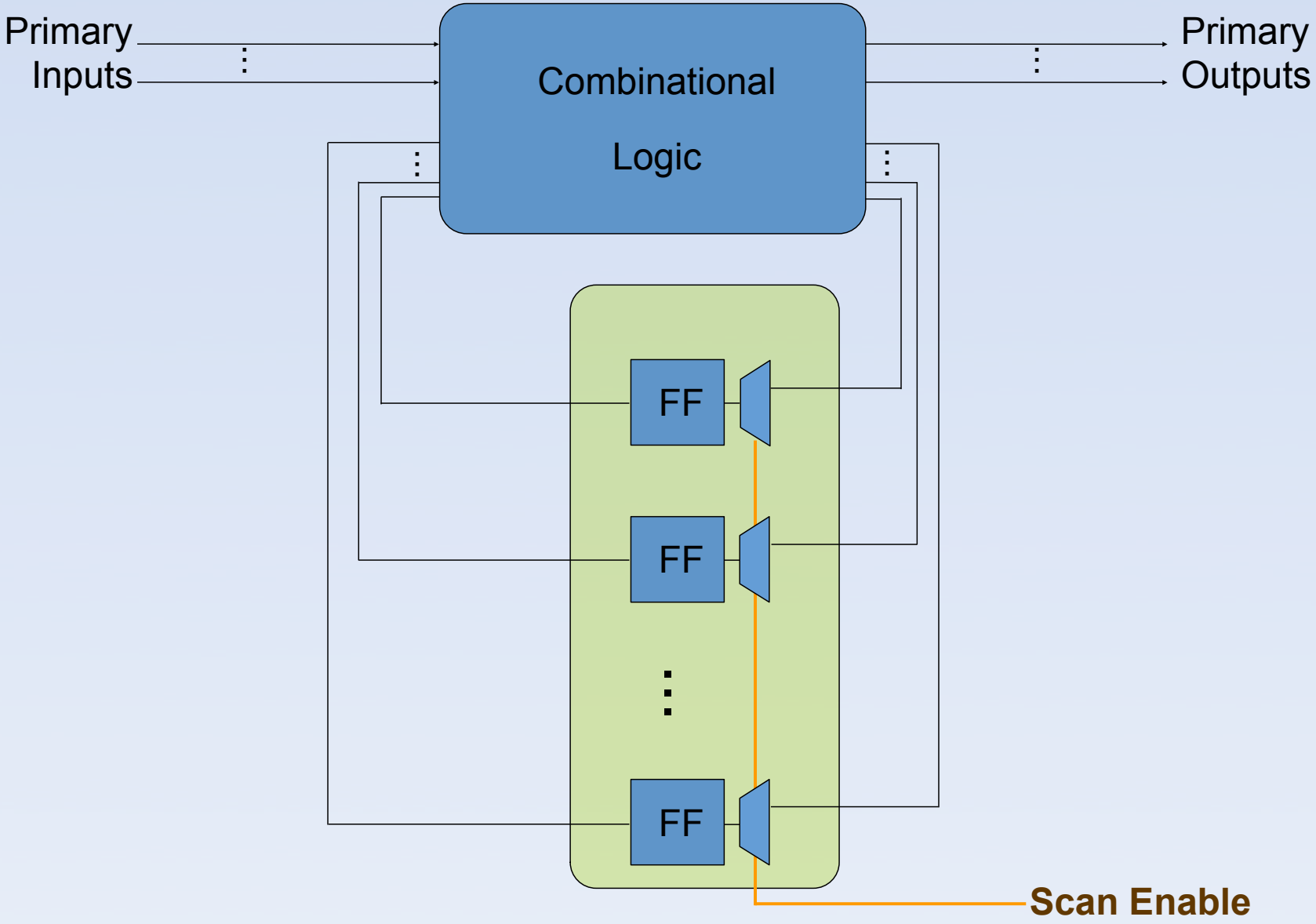
Scan-based Design (2)



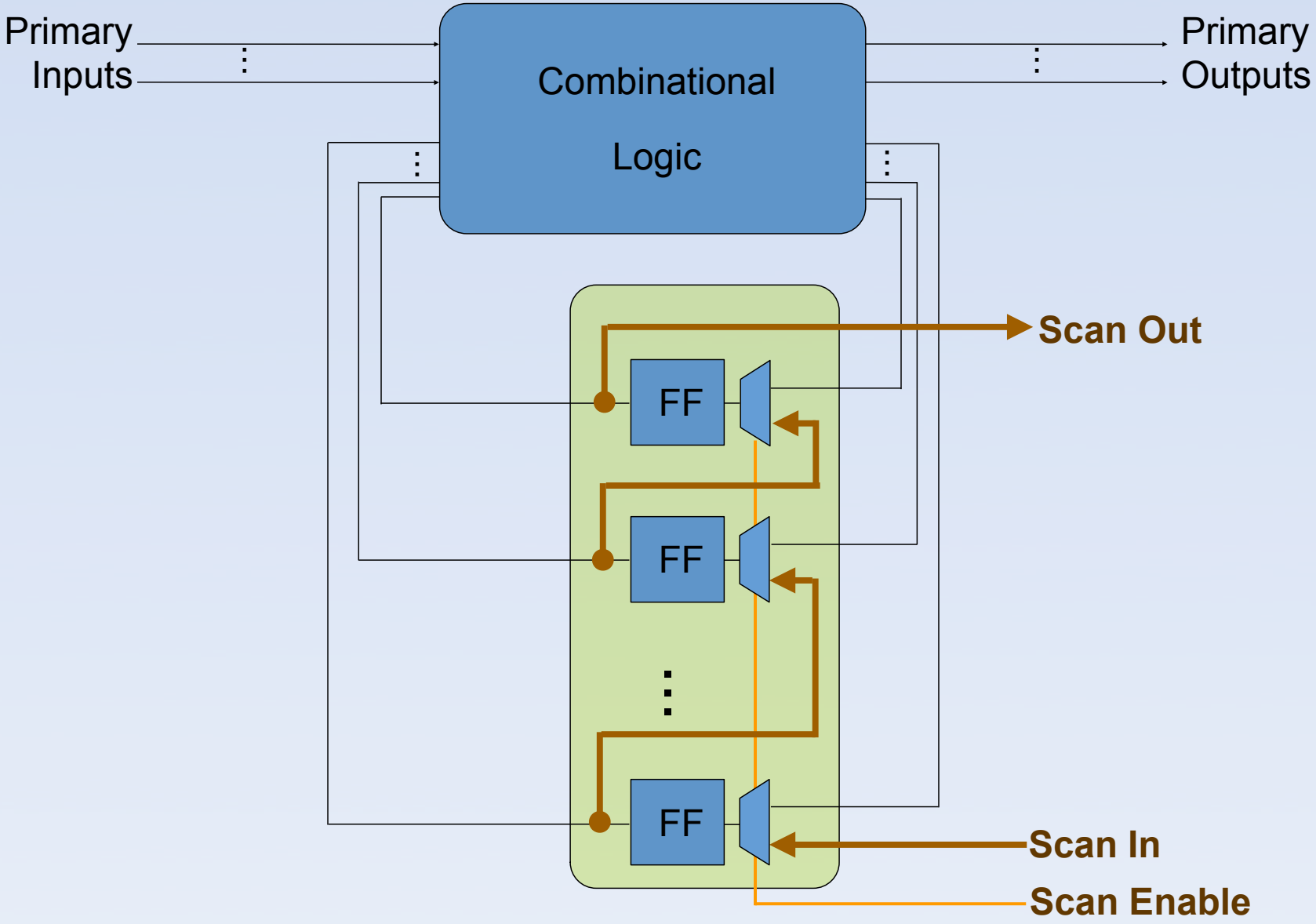
Scan-based Design (2)



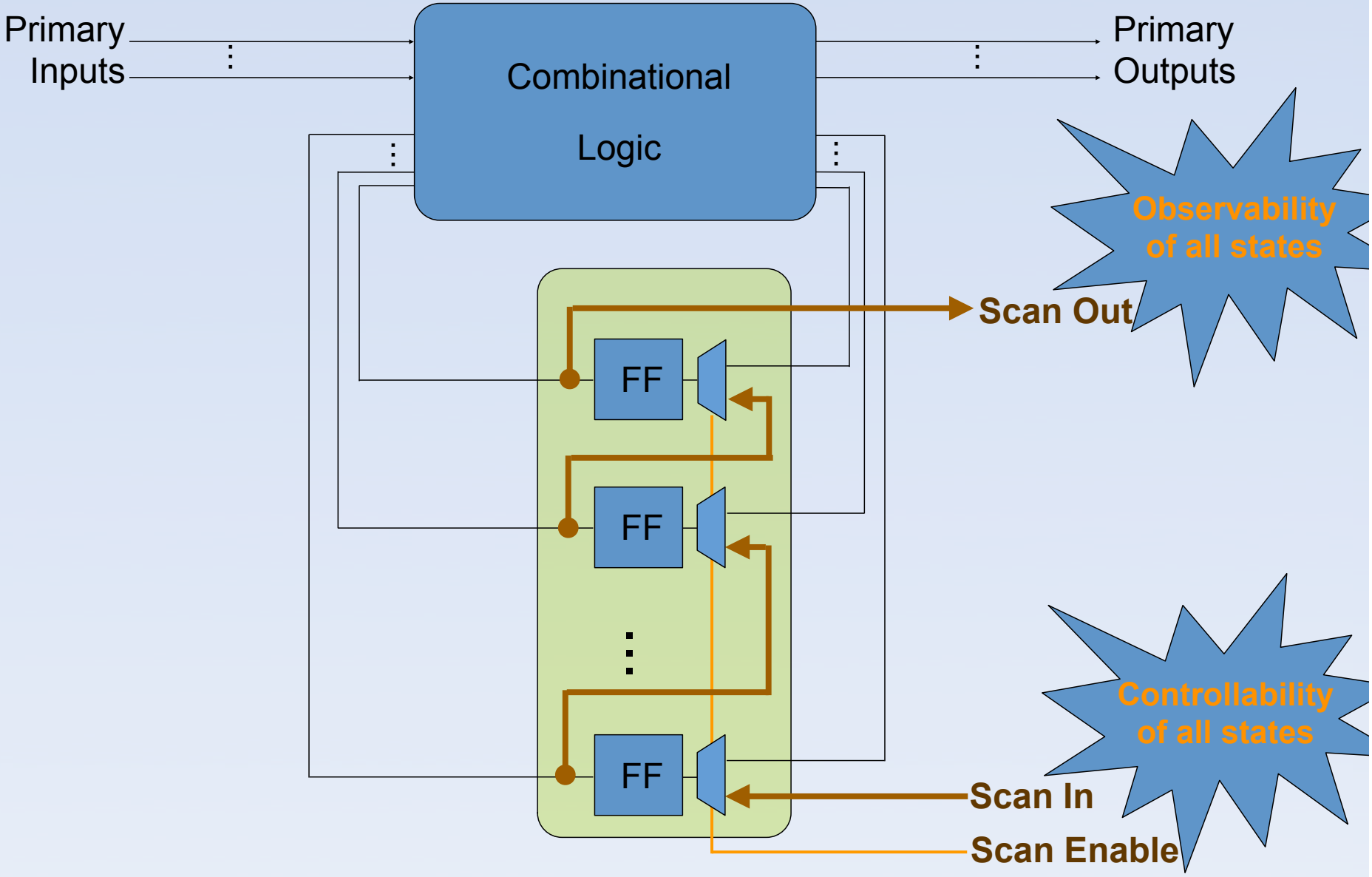
Scan-based Design (2)



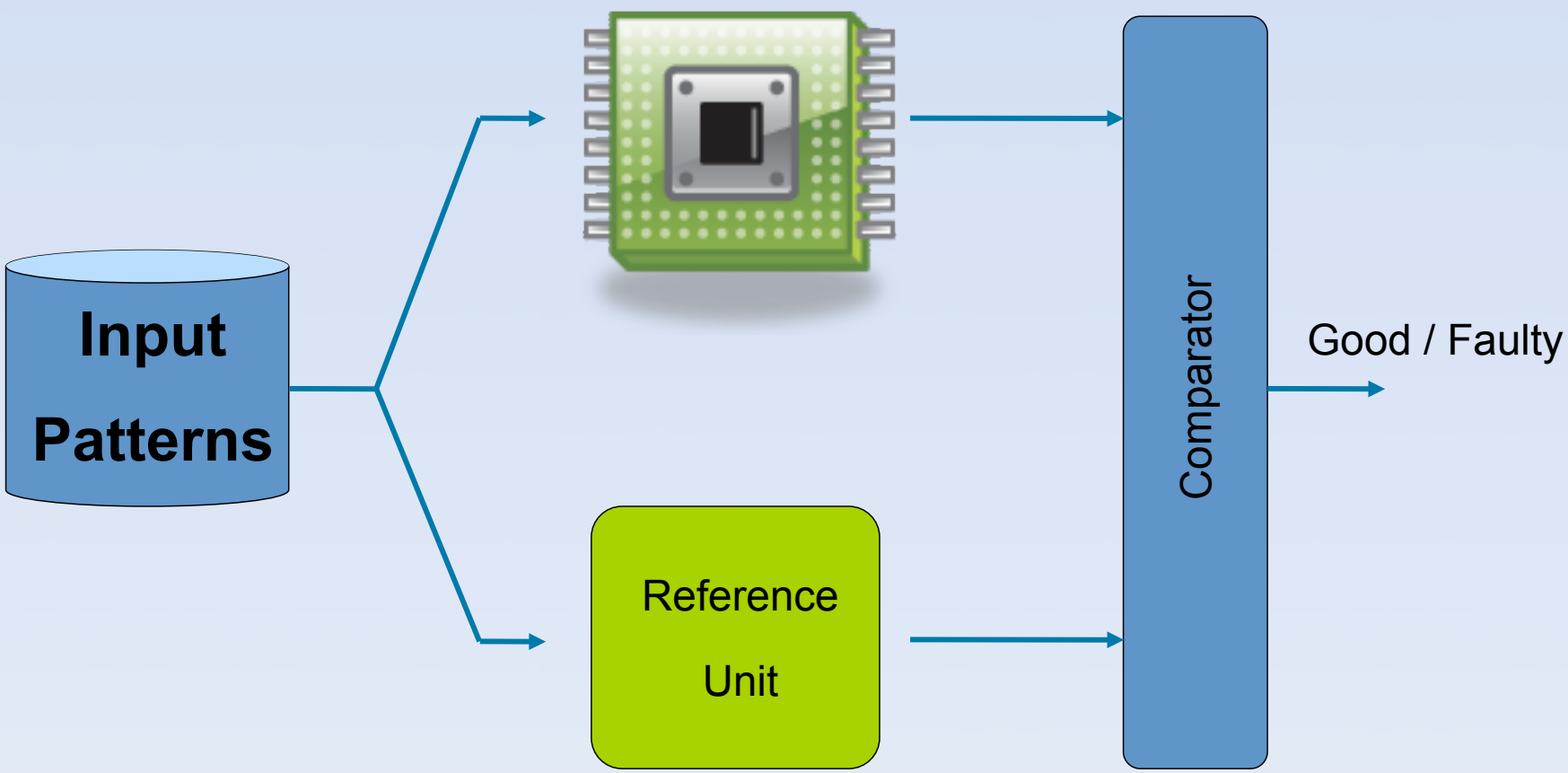
Scan-based Design (2)



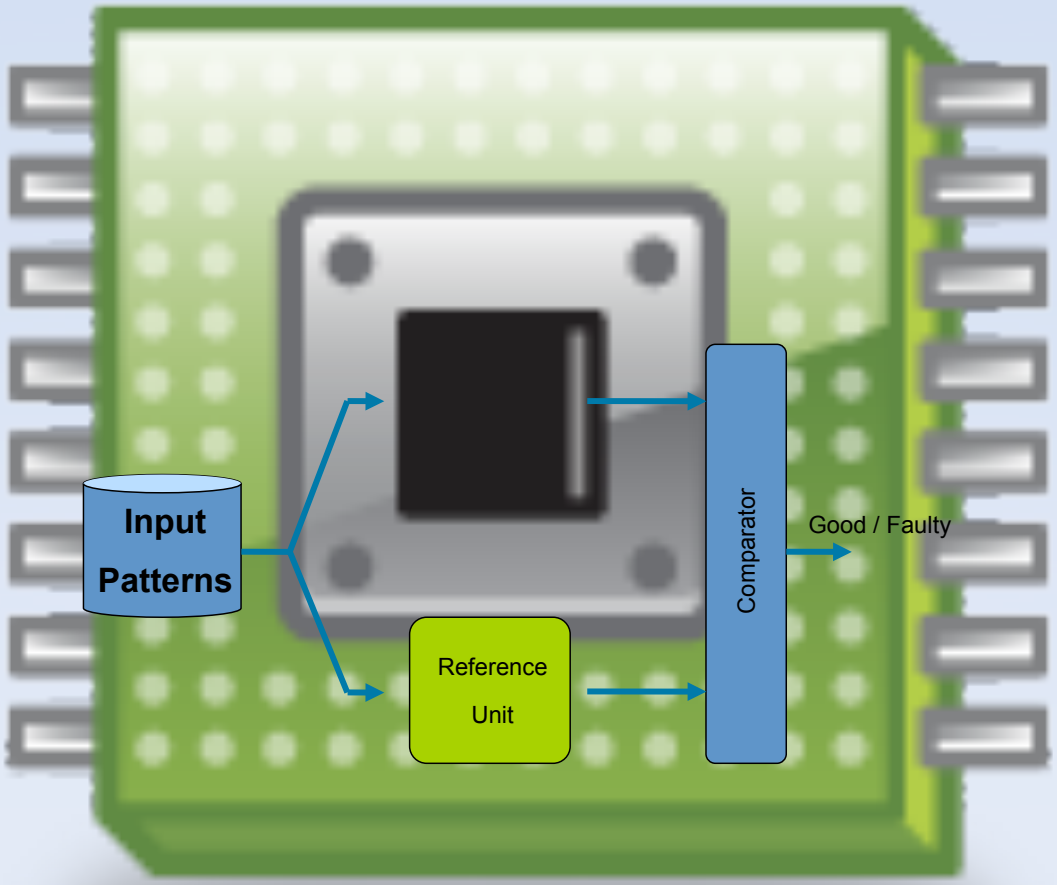
Scan-based Design (2)



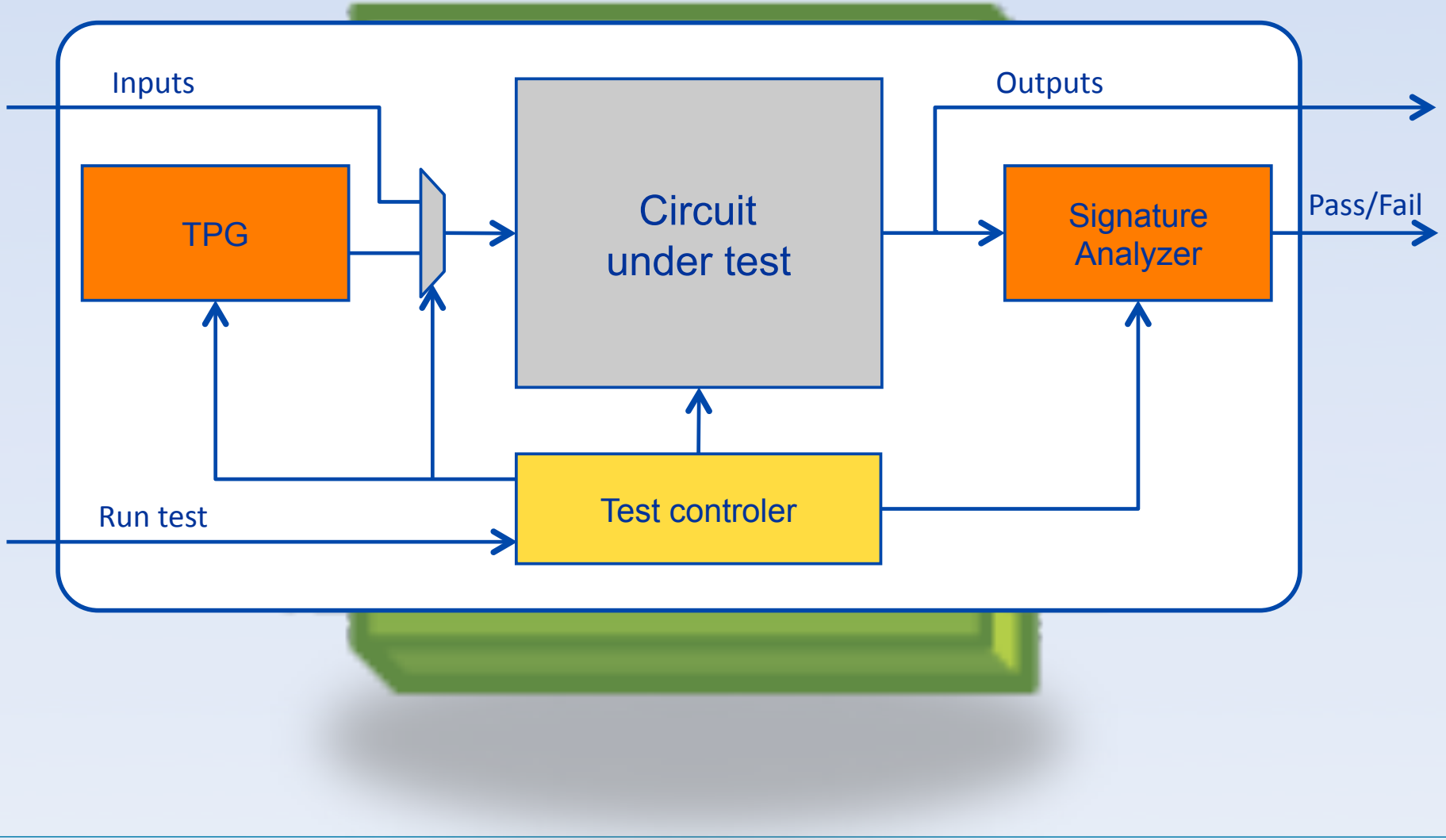
Built In Self-Test



Built In Self-Test



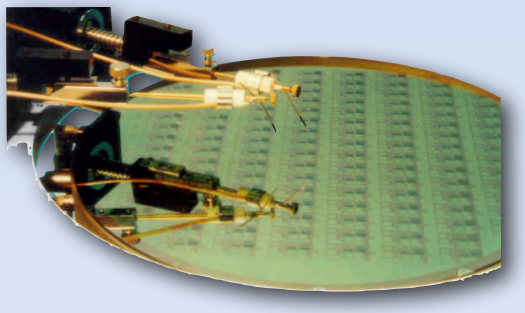
Built In Self-Test



Test vs Security

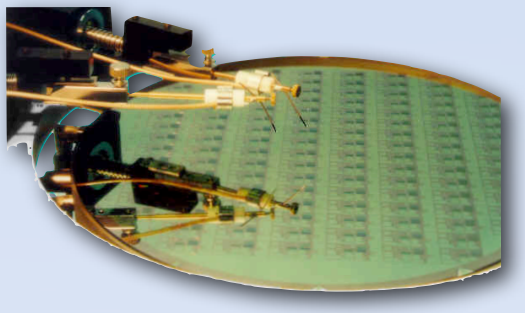
Test vs Security

- Circuit testing is mandatory to guarantee high quality of digital ICs
  - A hardware defect may induce some security vulnerability

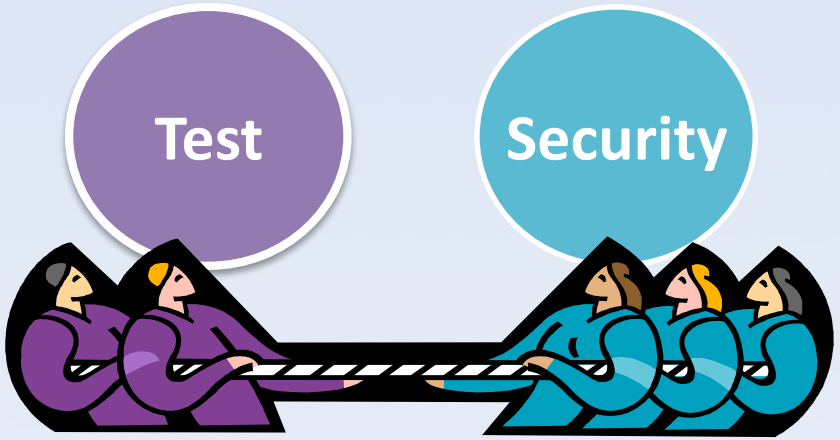


Test vs Security

- Circuit testing is mandatory to guarantee high quality of digital ICs
  - A hardware defect may induce some security vulnerability



- On the contrary, security fears testability
  - Test infrastructure used for attacks



Outline

- Test
- **Security vs Test**
  - **Scan-based attacks**
  - **Securing the scan chains**
  - **BIST as an alternative**
- Conclusions

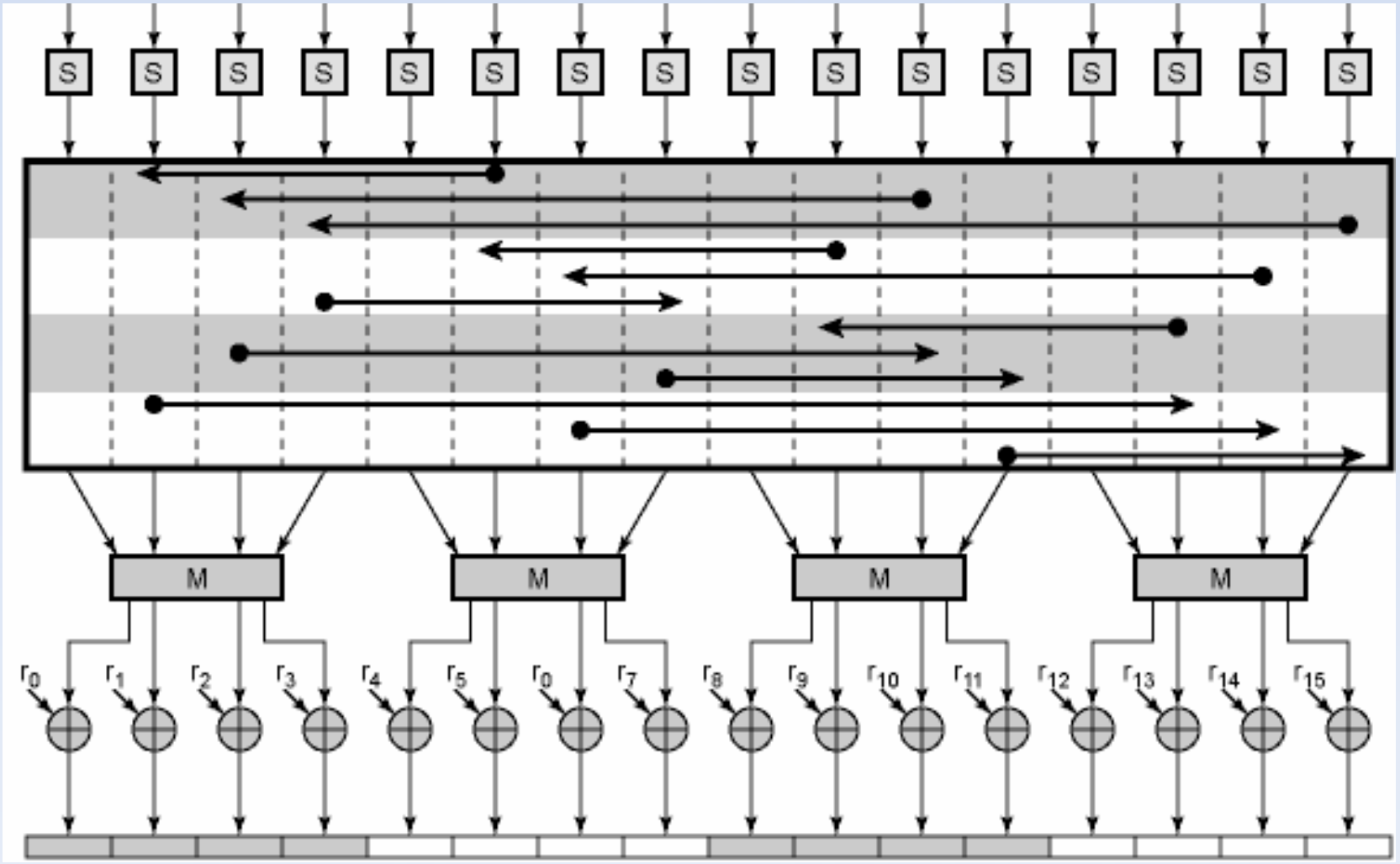
Scan-Based attacks

- Hypothesis
  - Free access to Scan-in, Scan-out, Scan enable signals
  - Test-key and Secret-key are different
  - No register storing the Secret-key in the scan path
- Hazard
  - Possibility for hackers to shift out the scan chain content while the circuit contains data related to the secret key
- Principle
  - Switch from System mode to Test mode and scan out sensible data

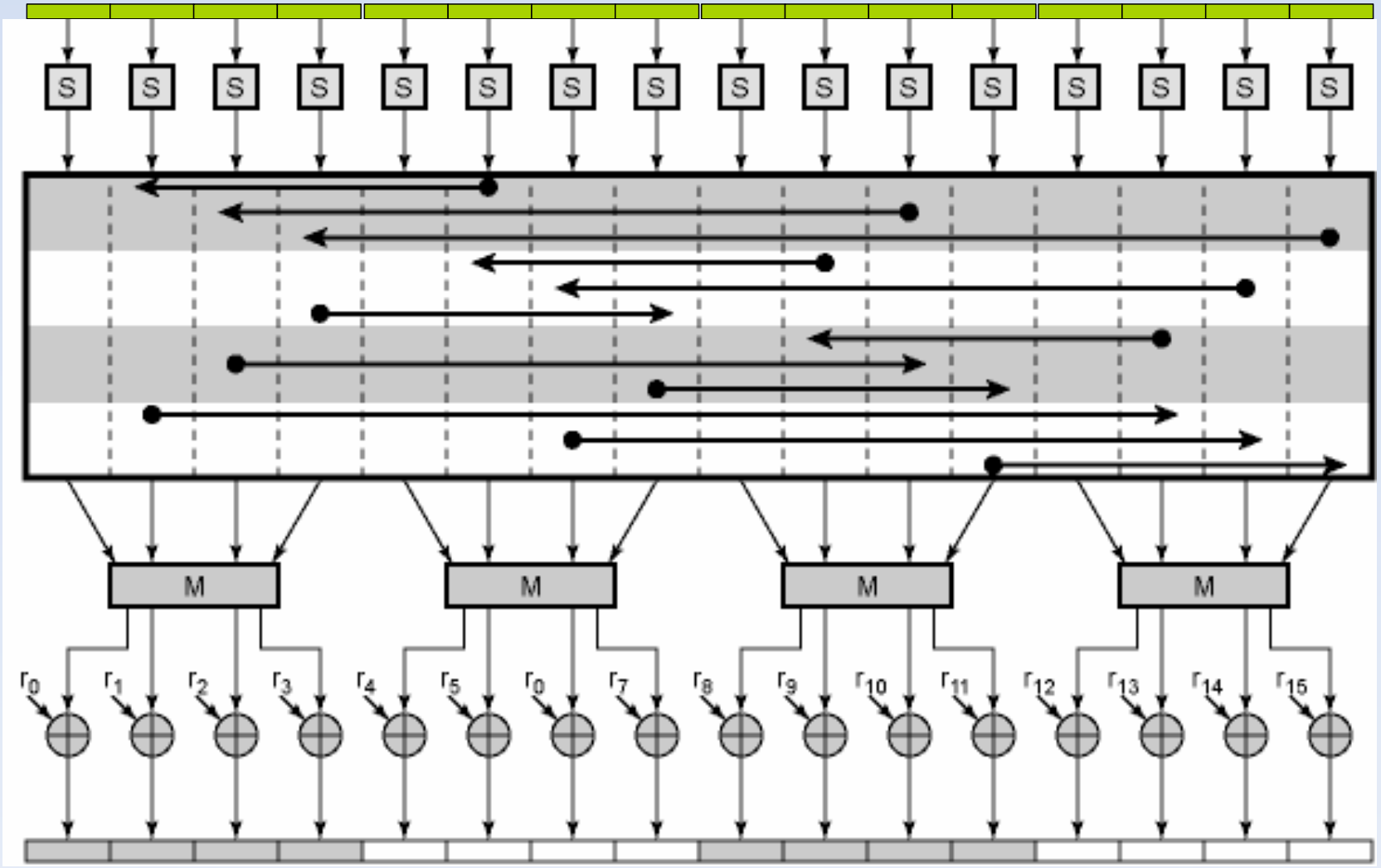
Scan-Based attacks [Yang, DAC05]

- Target: AES
- Goal: to retrieve the secret key (K)
- Method
  1. Retrieve the scan FFs storing the cipher text (among all the scan FFs)
  2. Read the FFs content after 1 encryption round and mathematically compute the key

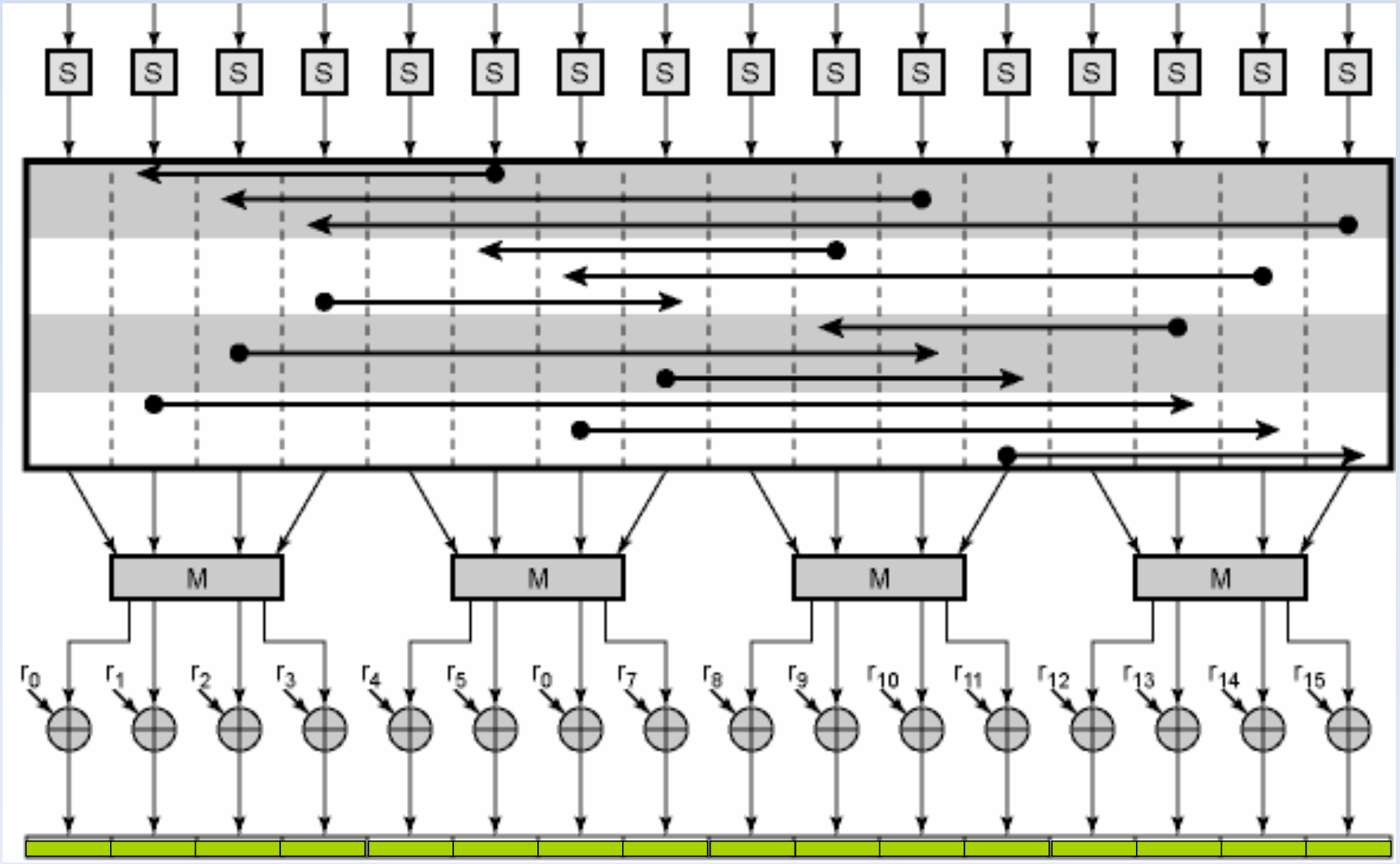
1. Retrieve the FFs



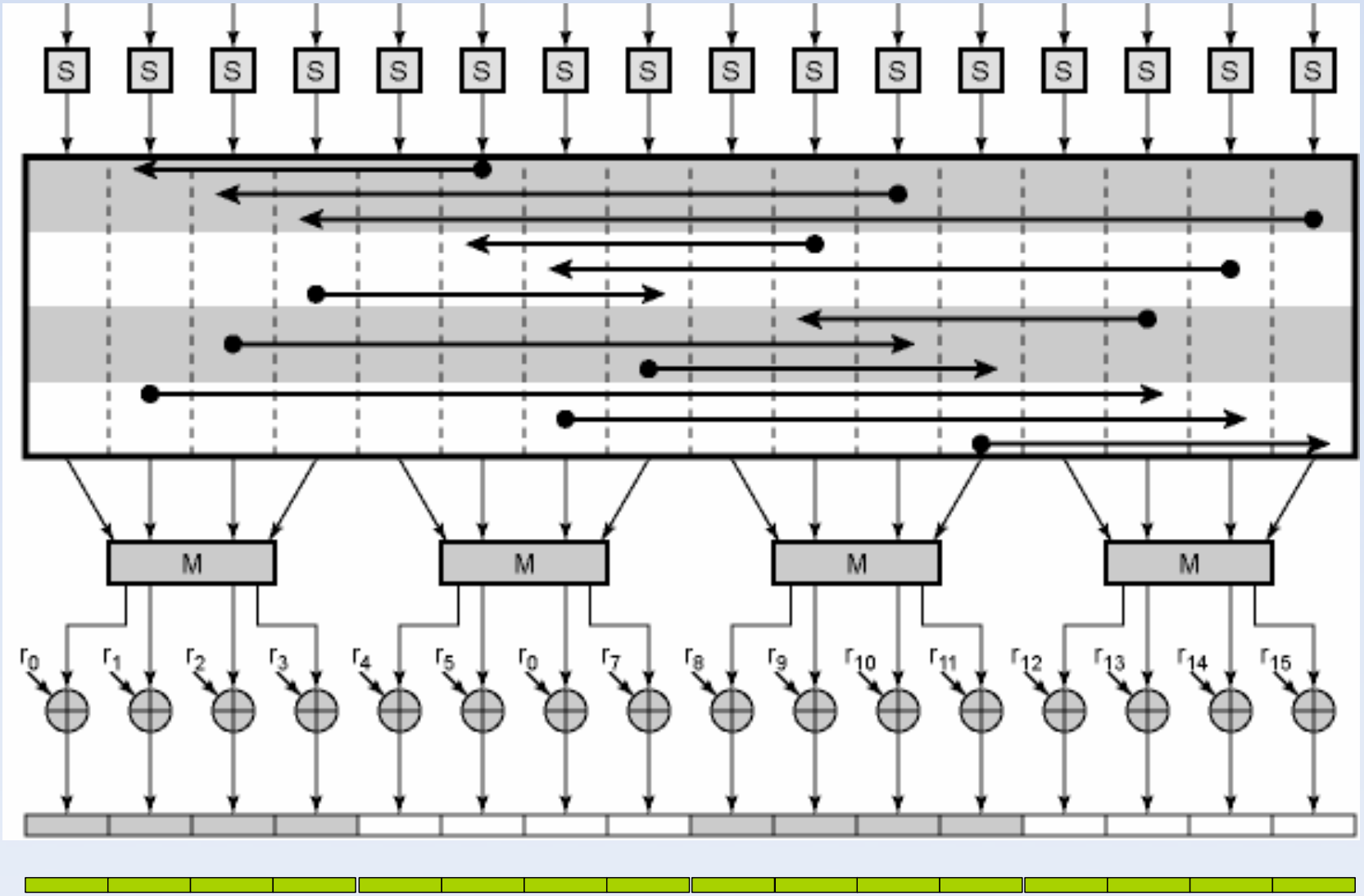
1. Retrieve the FFs



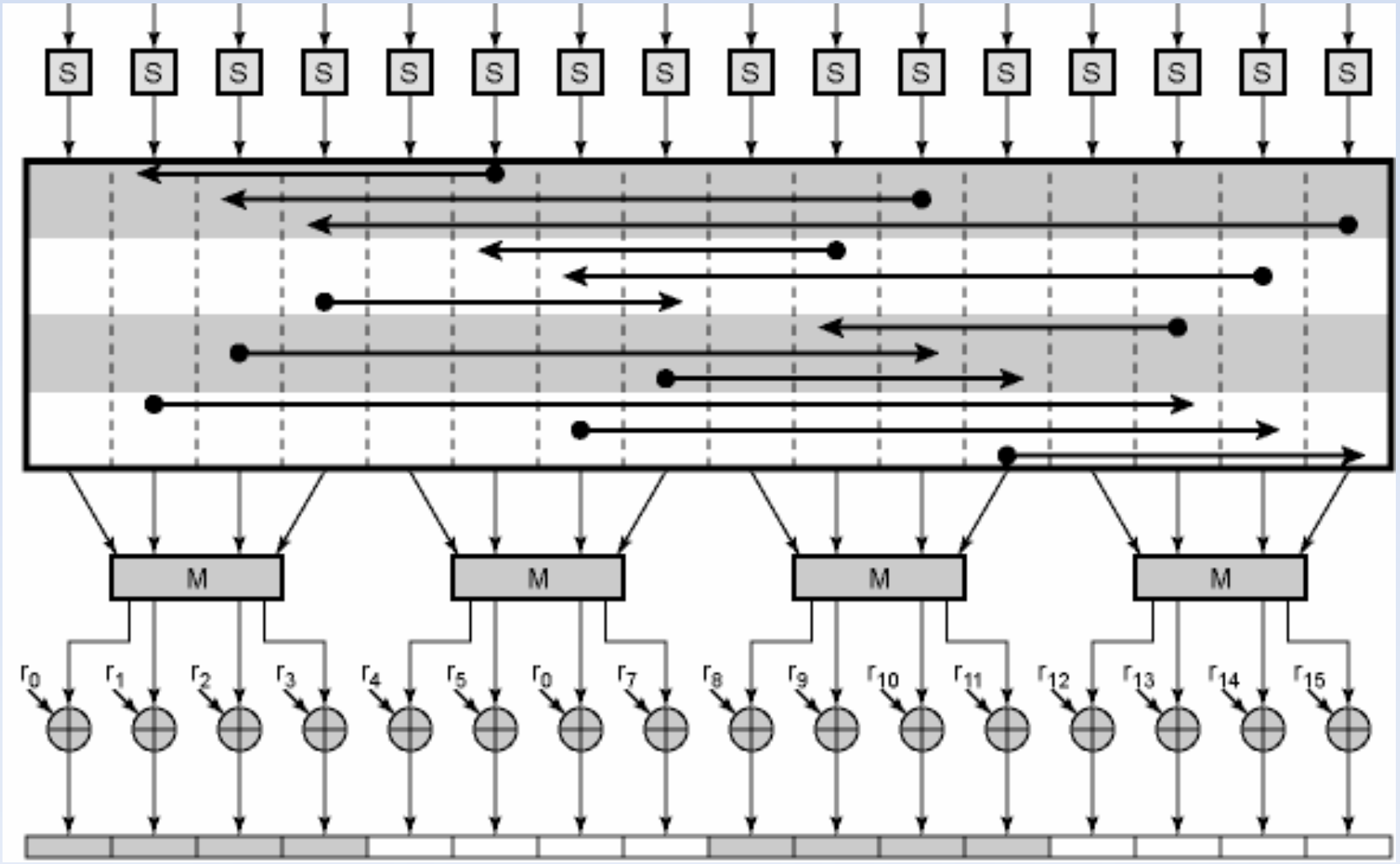
1. Retrieve the FFs



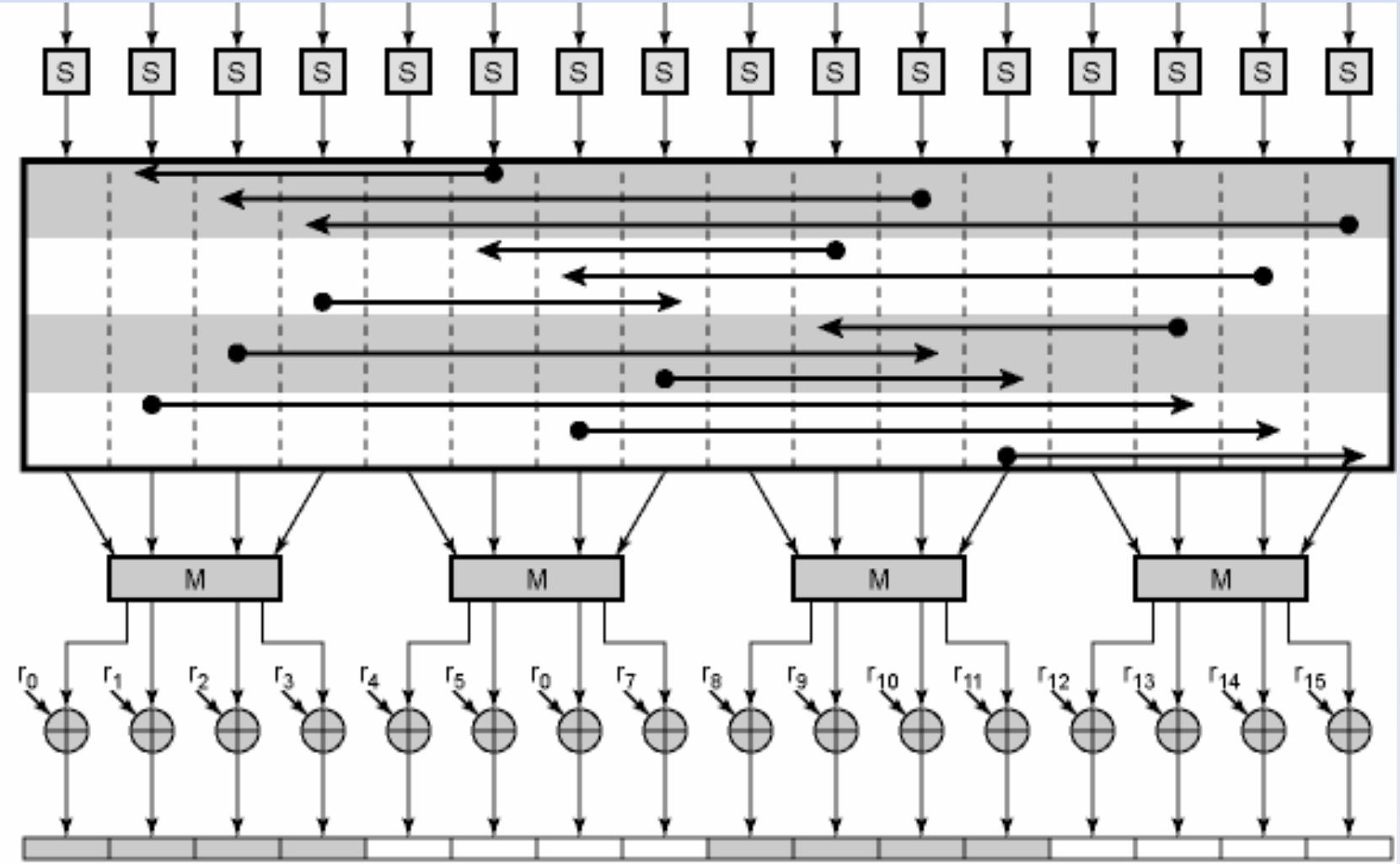
1. Retrieve the FFs



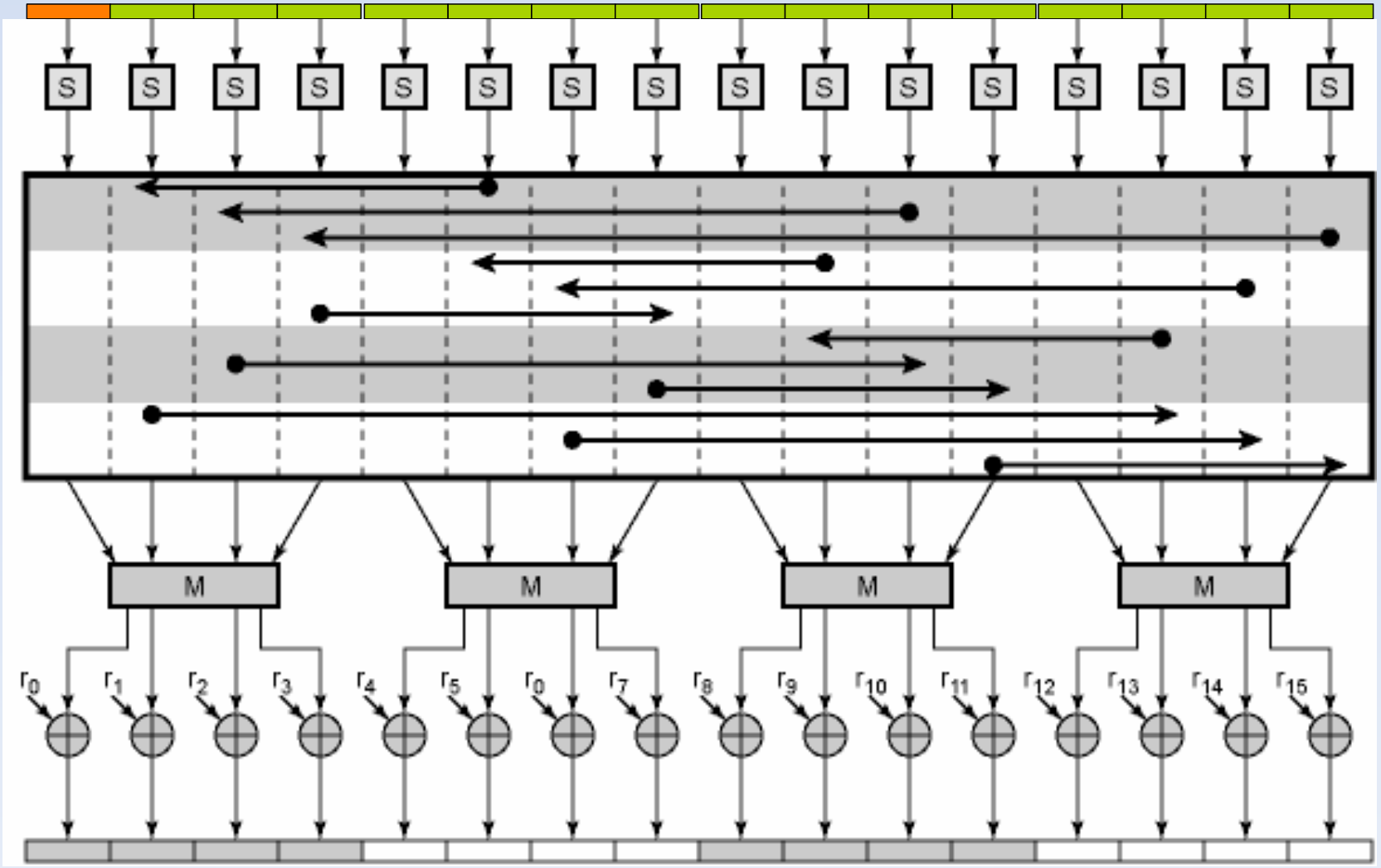
1. Retrieve the FFs



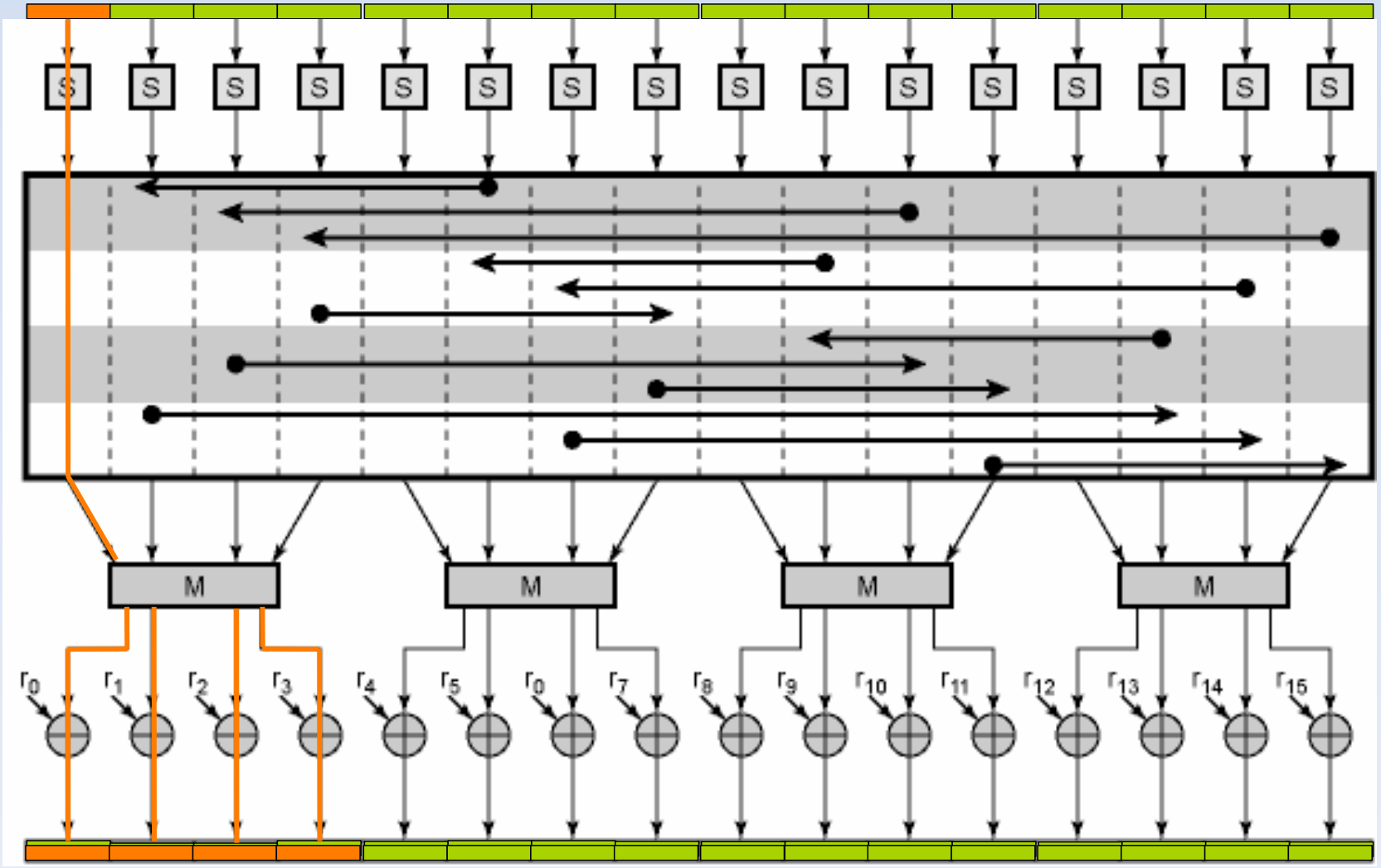
1. Retrieve the FFs



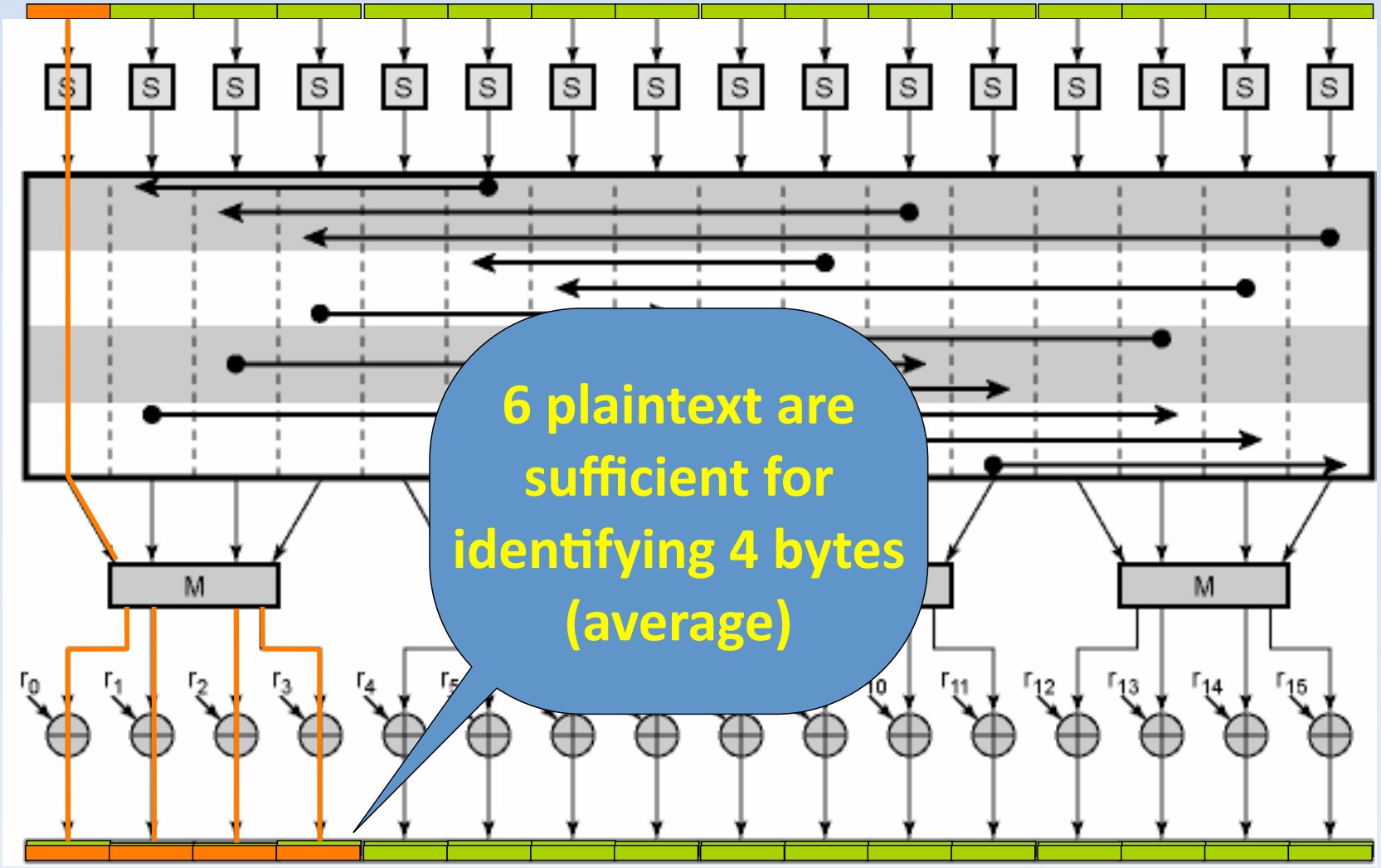
1. Retrieve the FFs



1. Retrieve the FFs



1. Retrieve the FFs



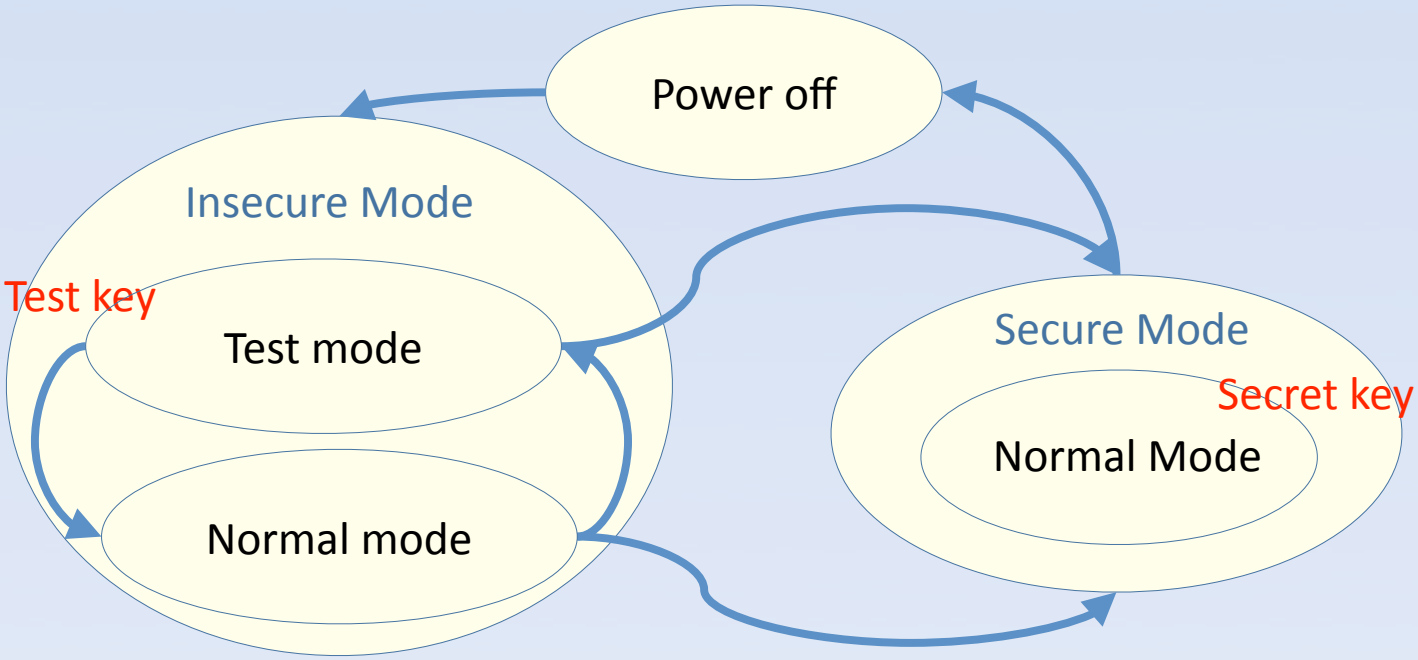
2. Compute the secret key

- R_i = Round (A_0 ... A_i ... A_{15})
- R_i' = Round (A_0 ... (A_i+1)... A_{15})
- C_i = Hamming Weight ($R_i \oplus R_i'$)
- $$X_i = \begin{cases} 226 & if \quad C = 9 \\ 242 & if \quad C = 12 \\ 122 & if \quad C = 23 \\ 130 & if \quad C = 24 \end{cases}$$
- $K_i = A_i \oplus X_i$

Securing the scan chains

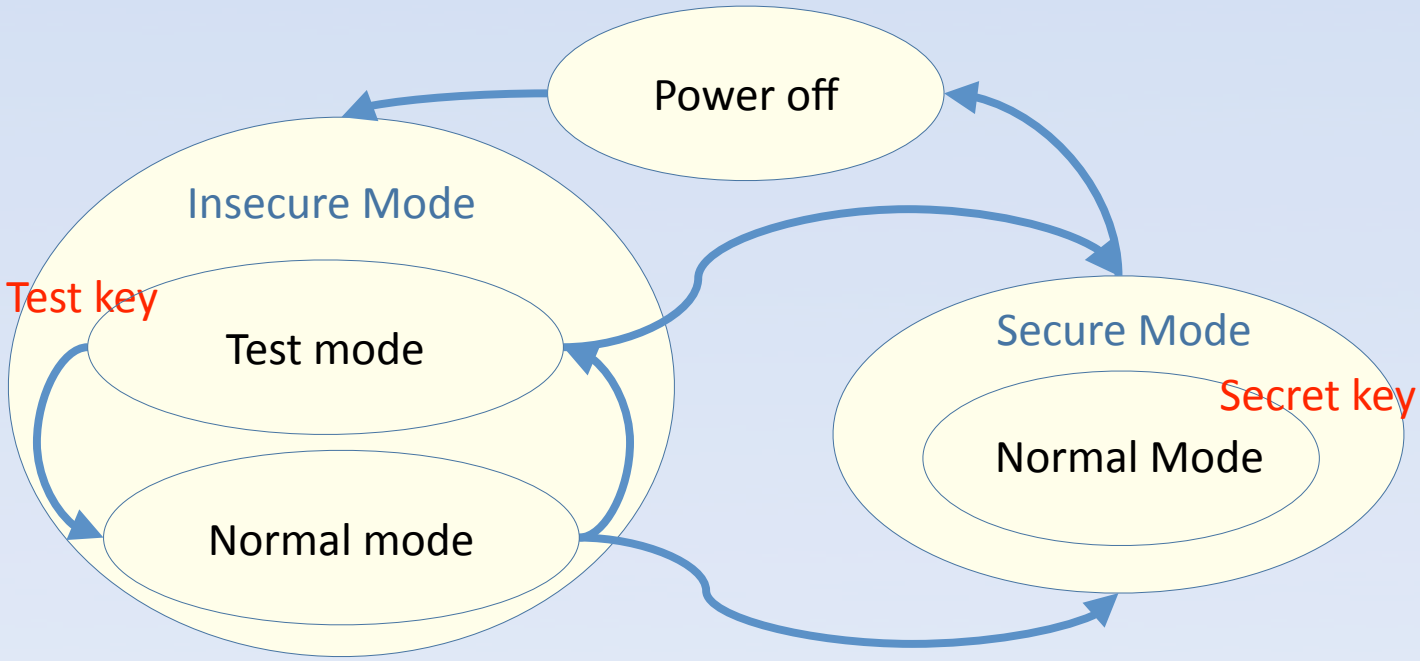
- Basic Countermeasure:
  - Leave the scan chain unbound
  - But
    - Scan chains may still be accessed
    - Compromise maintenance capability in field
- Alternatives
  - Securing the test protocol
  - Securing the chains

Test protocol



[B. Yang et al., IEEE TRANS. ON CAD, oct. 2006]

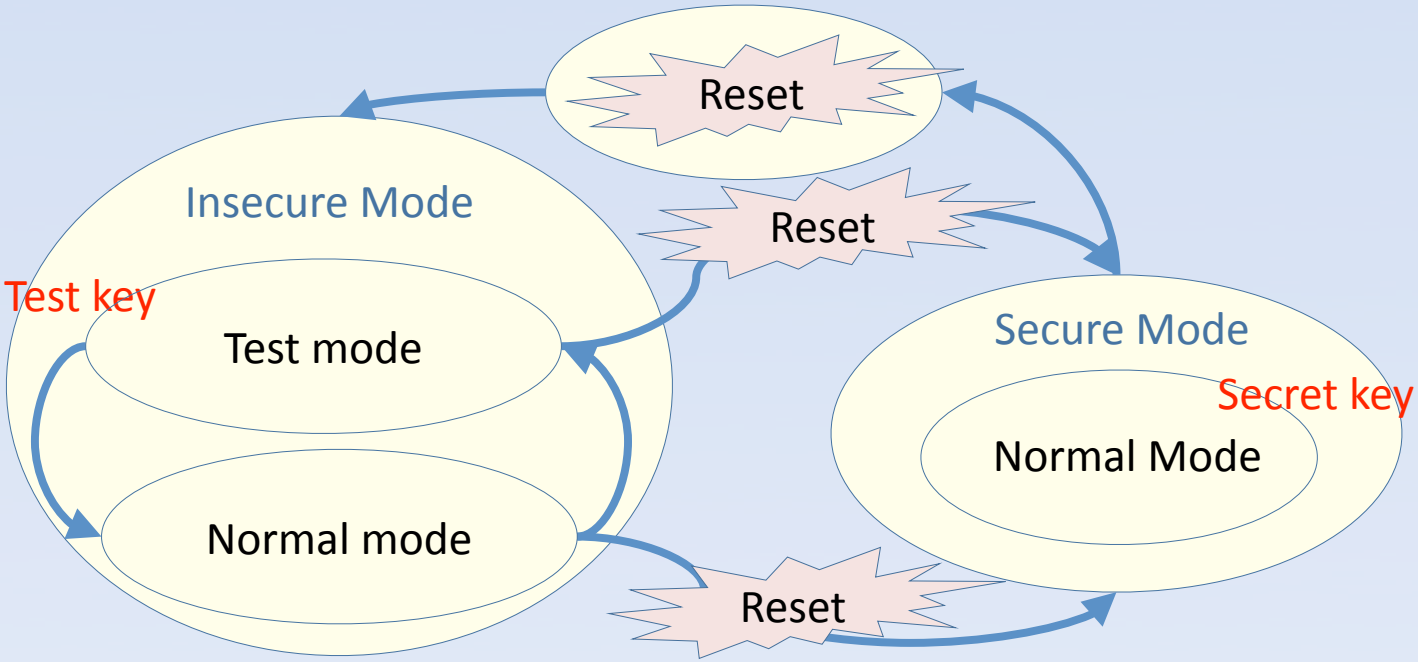
Test protocol



[B. Yang et al., IEEE TRANS. ON CAD, oct. 2006]

[D. Hély et al., Securing Scan Control in Crypto Chips, JETTA 2007]

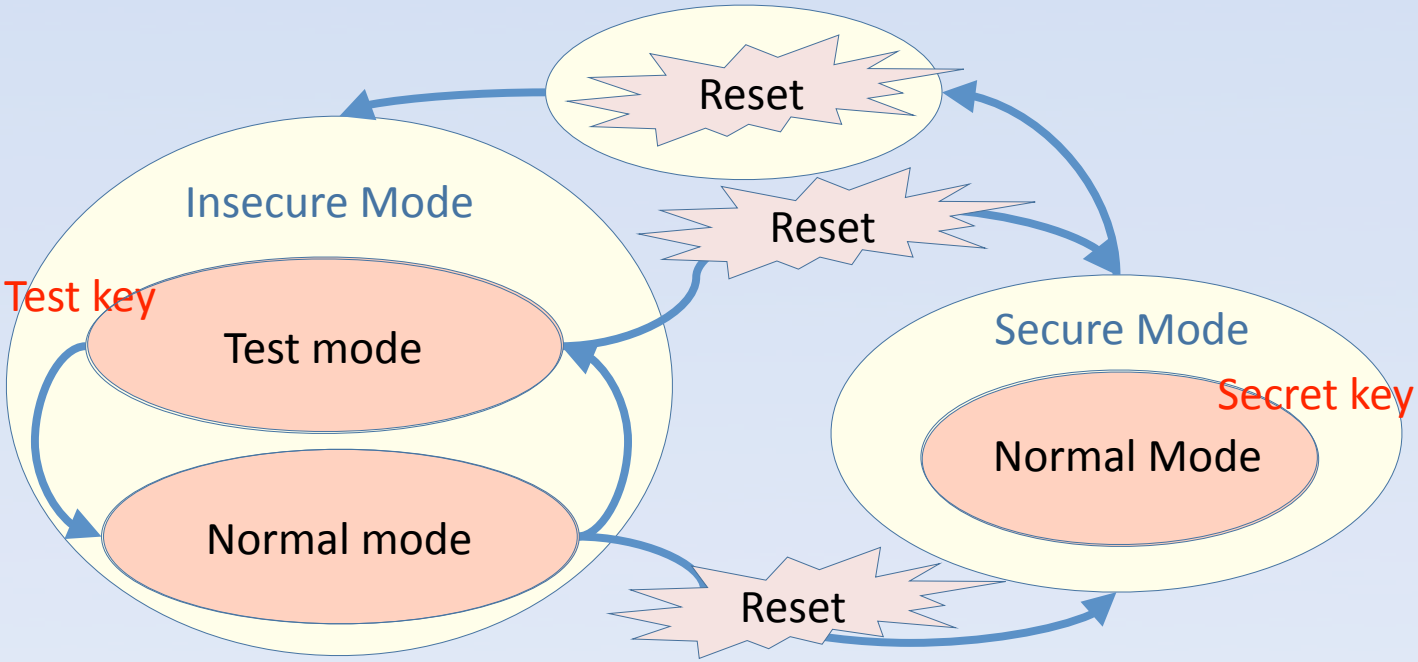
Test protocol



[B. Yang et al., IEEE TRANS. ON CAD, oct. 2006]

[D. Hély et al., Securing Scan Control in Crypto Chips, JETTA 2007]

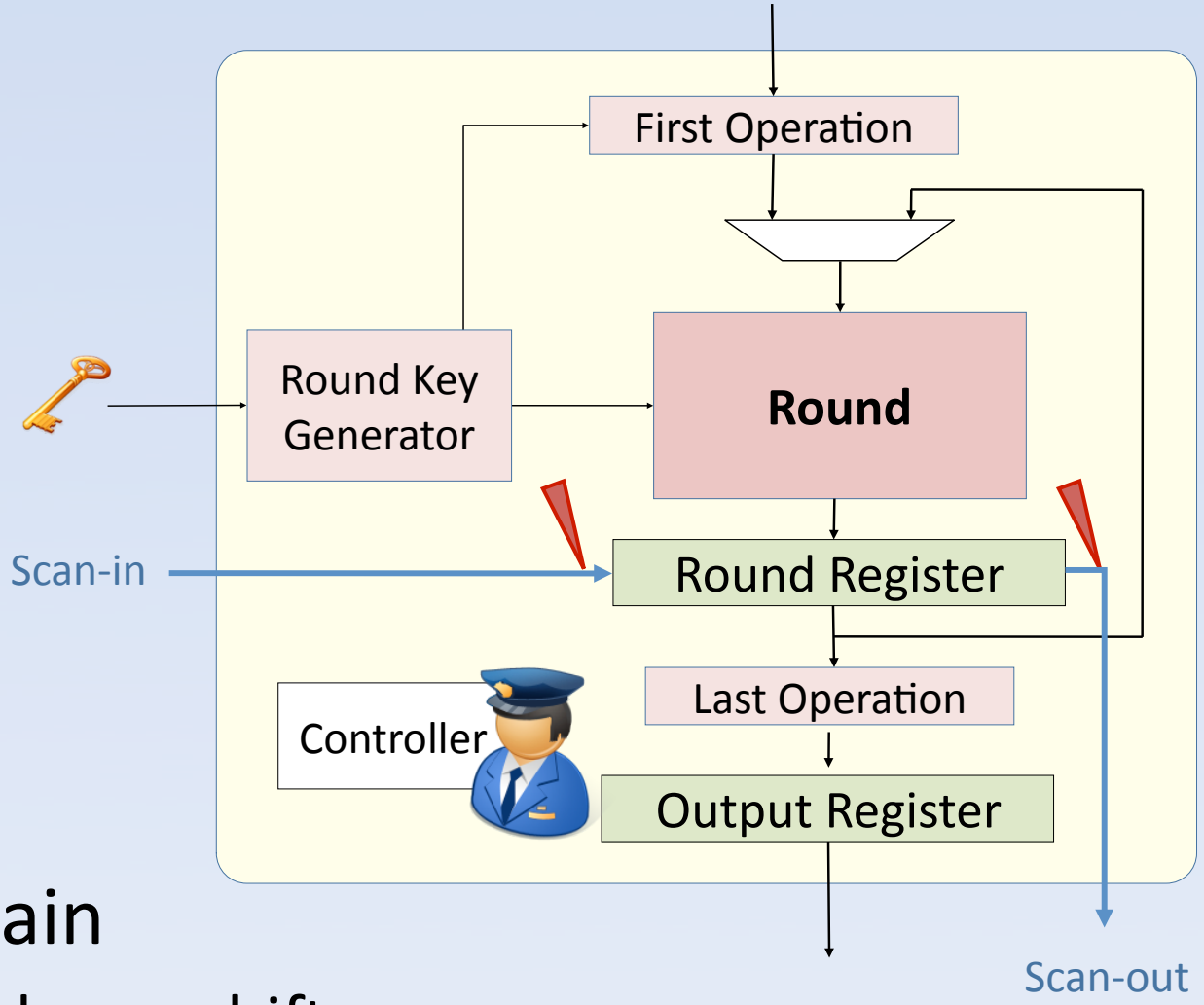
Test protocol



[B. Yang et al., IEEE TRANS. ON CAD, oct. 2006]

[D. Hély et al., Securing Scan Control in Crypto Chips, JETTA 2007]

Securing the scan chain

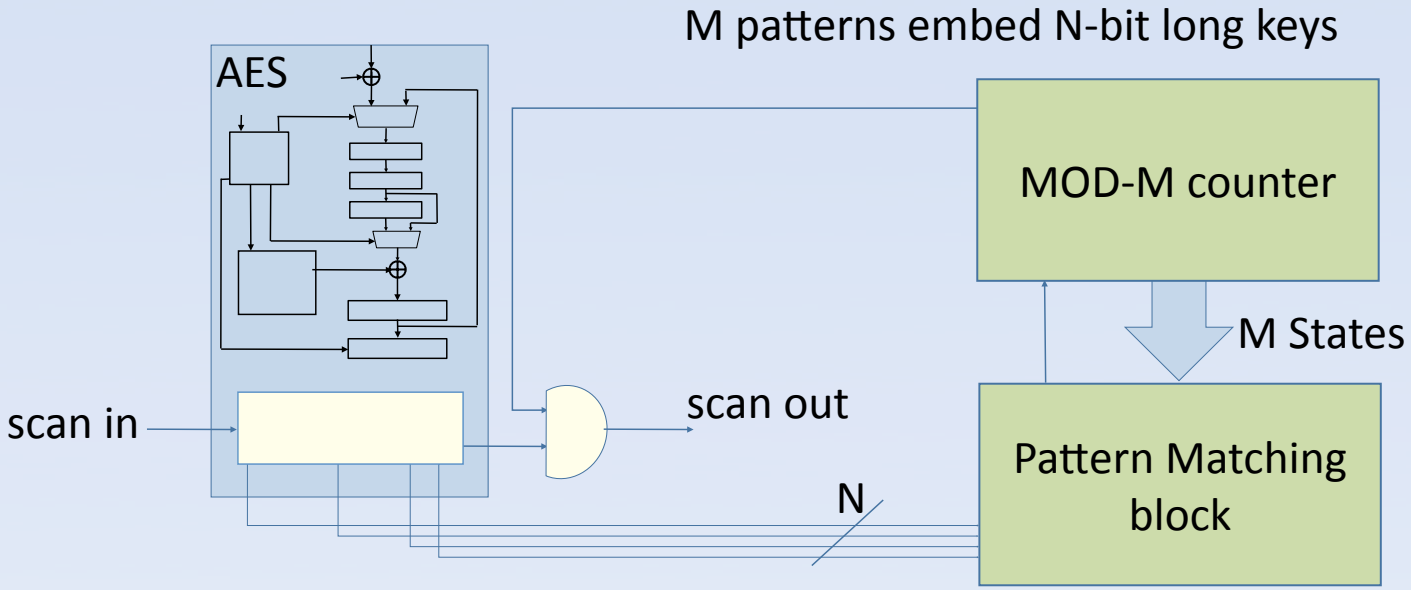


- Secure the scan chain
  - Detect unauthorized scan shifts
  - Data confusion

Detect unauthorized scan shift

1. Test Pattern Watermarking

– Test patterns include authentication keys

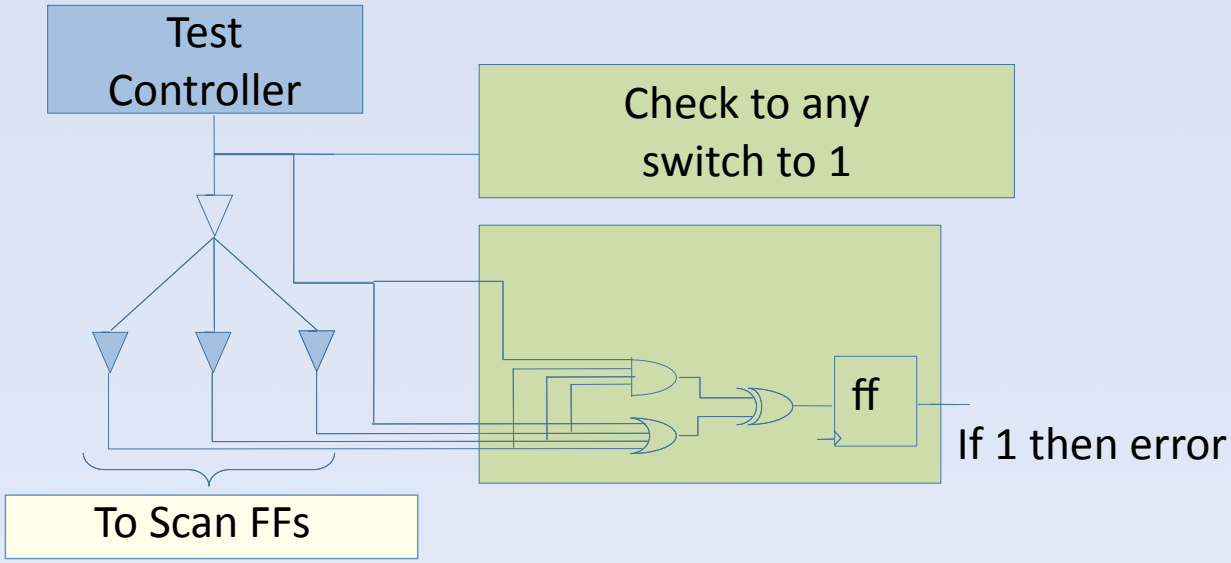


[D. Hély et al., Scan Pattern Watermarking, LATW 2006]
[J. Lee et al., A Low-Cost Solution for Protecting IPs Against Scan-Based Side-Channel Attacks, VTS 2006]
[S. Paul et al., Vim-Scan: A low Overhead Scan Design Approach for Protection of Secret Key in Scan-Based Secure Chips, VTS 2007]

Detect unauthorized scan shift

2. Scan-enable tree inspection

Compare Scan_Enable signals at different locations

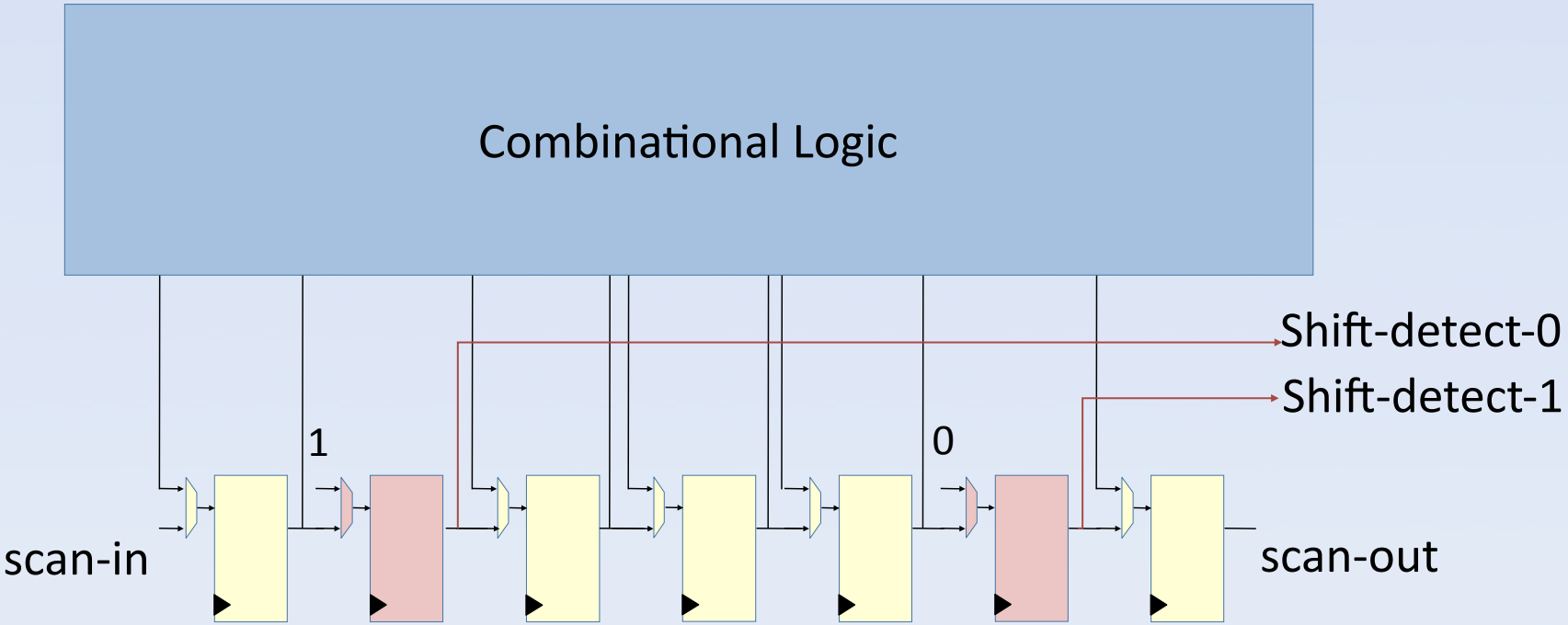


[D. Hély et al., Test Control for Secure Scan Designs, ETS 2005]

Detect unauthorized scan shift

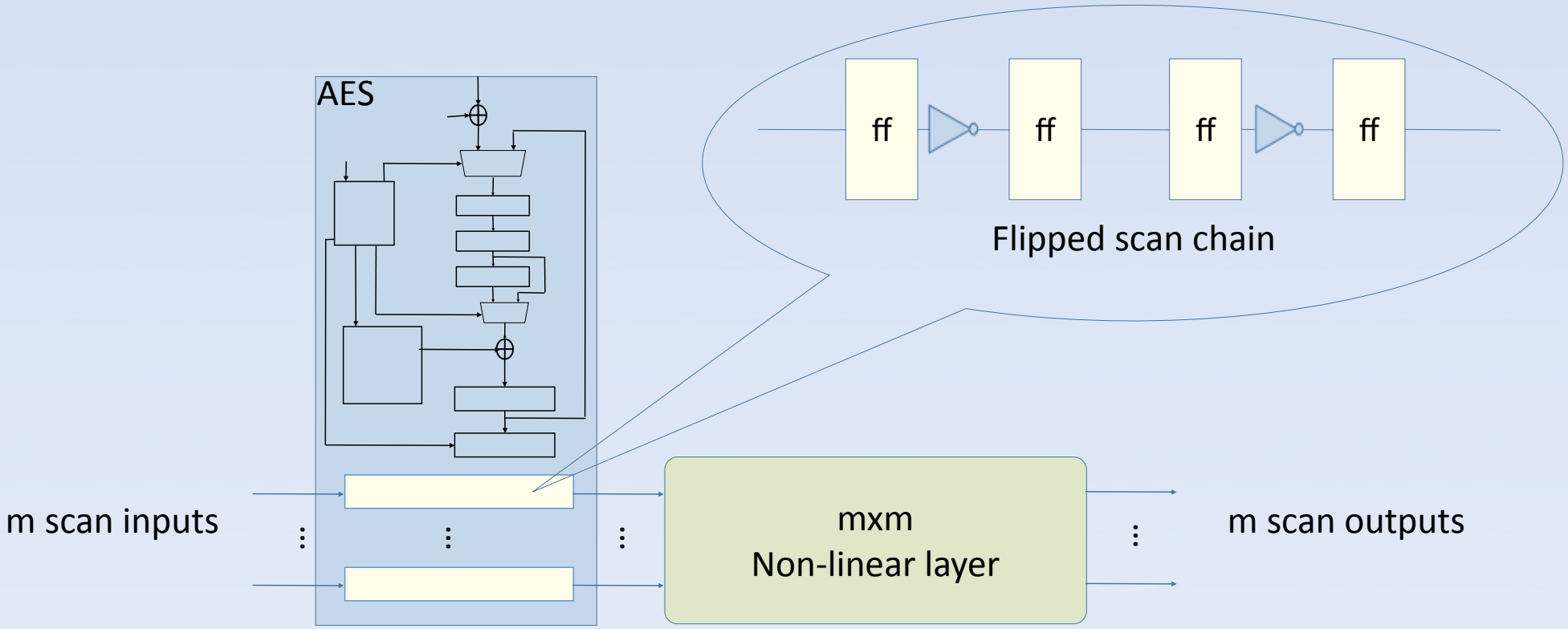
3. Spy on shift operations

- Include spy FFs in the scan chain, control their states to a fixed value and observe their states



Data confusion

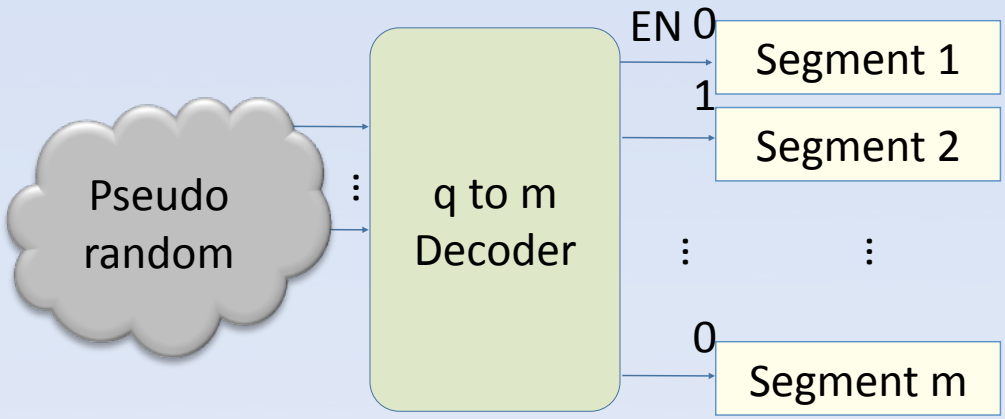
1. Flipped scan chain + nonlinear mapping



[G. Sengar et al., Secured Flipped Scan-Chain Model for Crypto-Architecture, TCAD 2007]

Data confusion

2. Scramble the scan chain structure



[D. Hély et al., Scan Design and Secure Chip, IOLTS 2004]

[J. Lee et al., Securing Scan Design Using Lock & Key Technique, DFT'05]

Built-In Self-Test (BIST)

- Motivations:
  - Allow at-speed testing
  - Reduced ATE cost
  - Avoid scan-based testing

- But:
  - Area overhead ?
  - Fault coverage ?

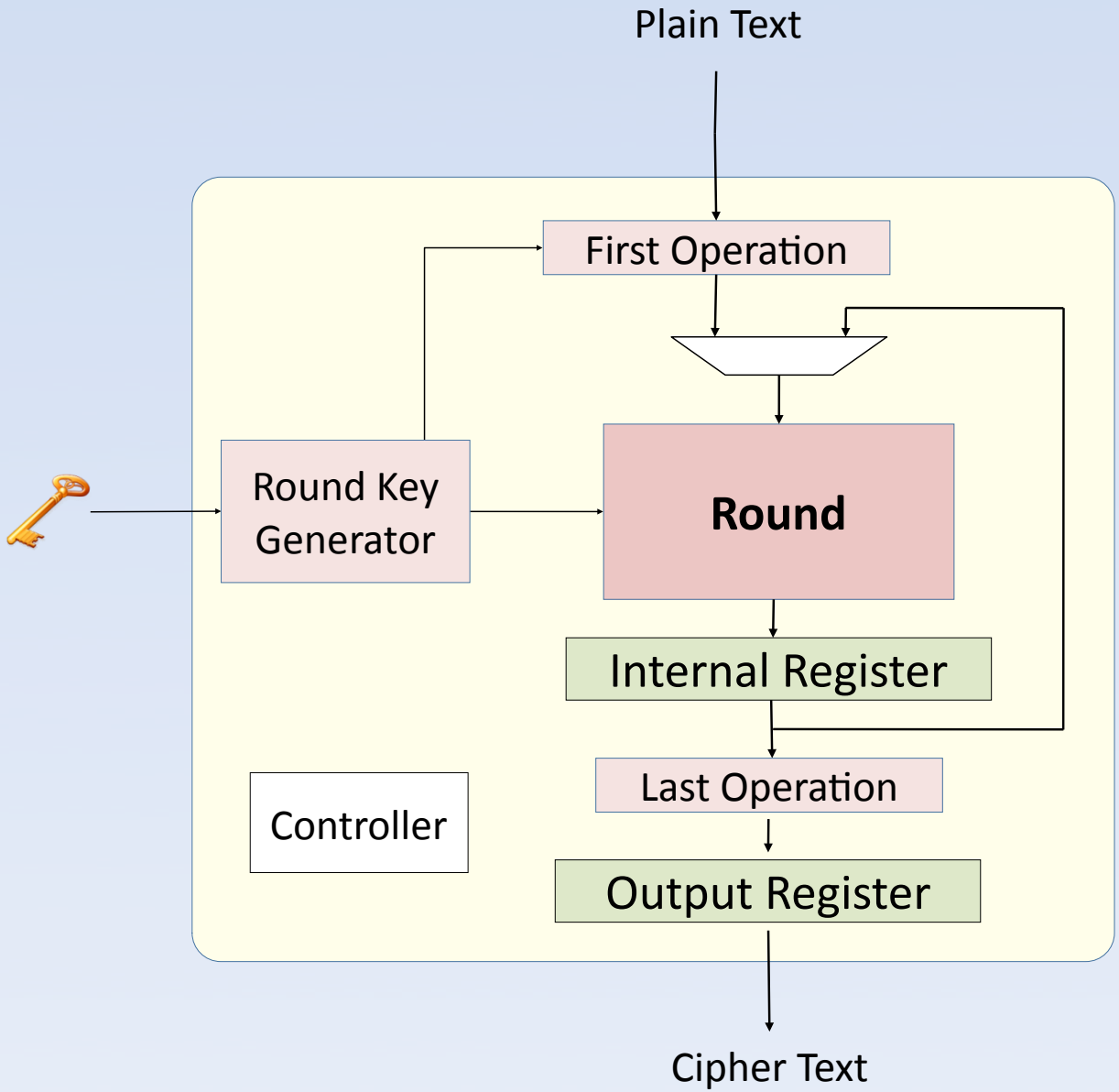
Properties of Crypto-Algorithms

- Confusion
  - making the relationship between the key and the ciphertext as complex and involved as possible
- Diffusion
  - a change in a single bit of the plaintext should result in changing the value of many ciphertext bits

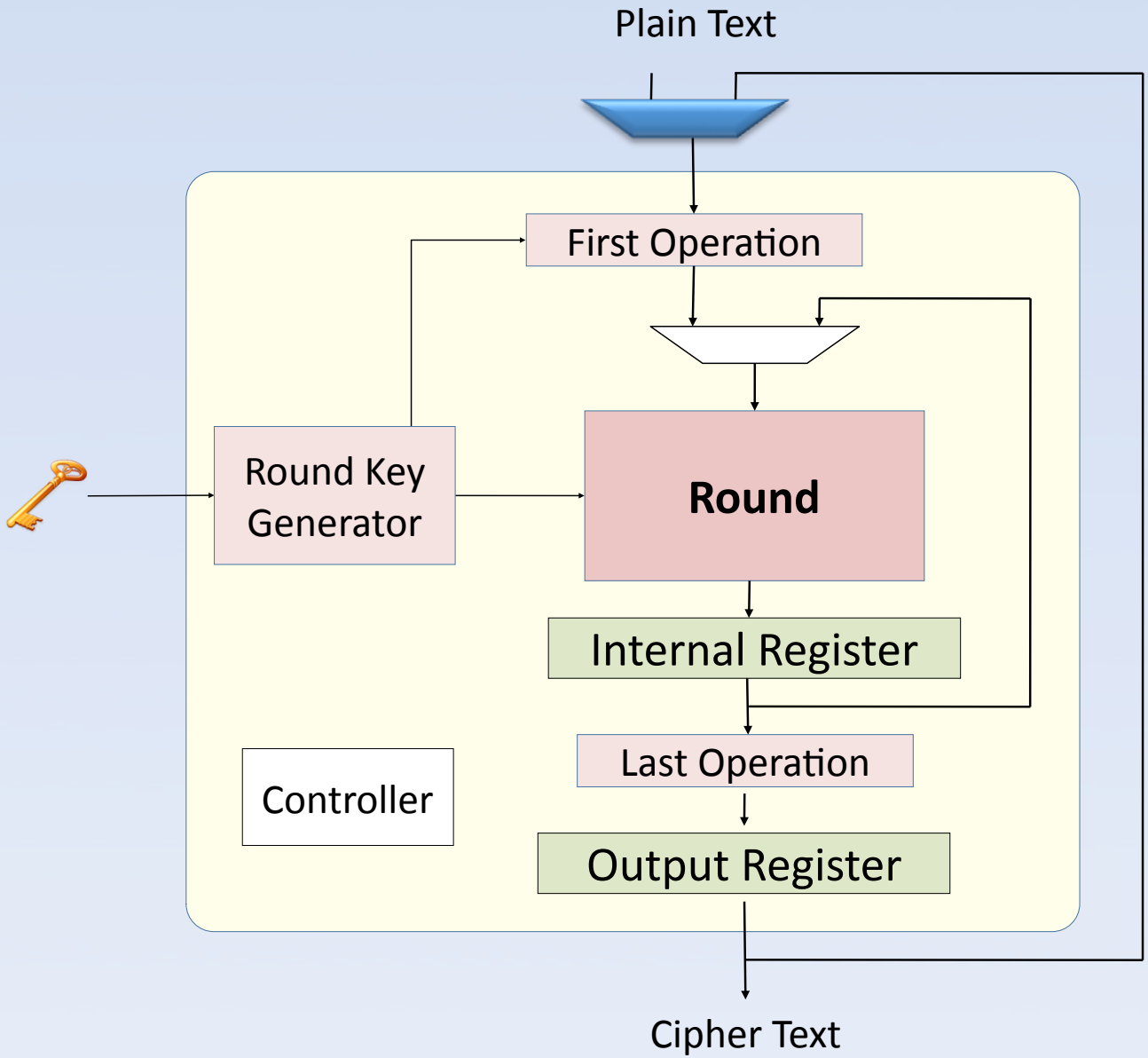
Testability

- Diffusion vs testability:
  - every input bit influences many output bits
  - ➡ Each fault is easily controllable and observable!
- Diffusion and confusion de-correlate the output values from the input ones
- It has been demonstrated that by feeding back the output to the input, the generated output sequence has random propertie

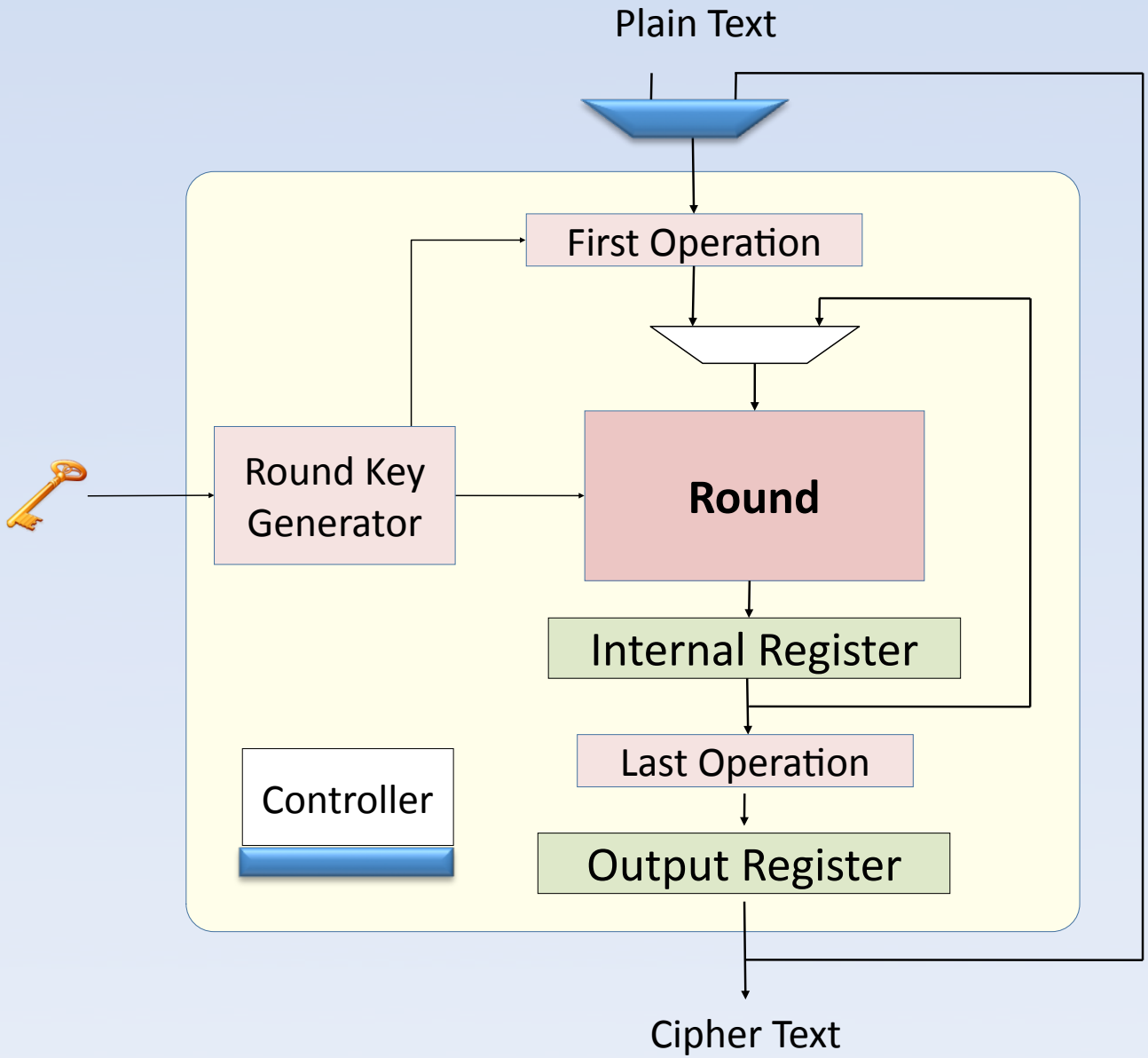
BIST Architecture



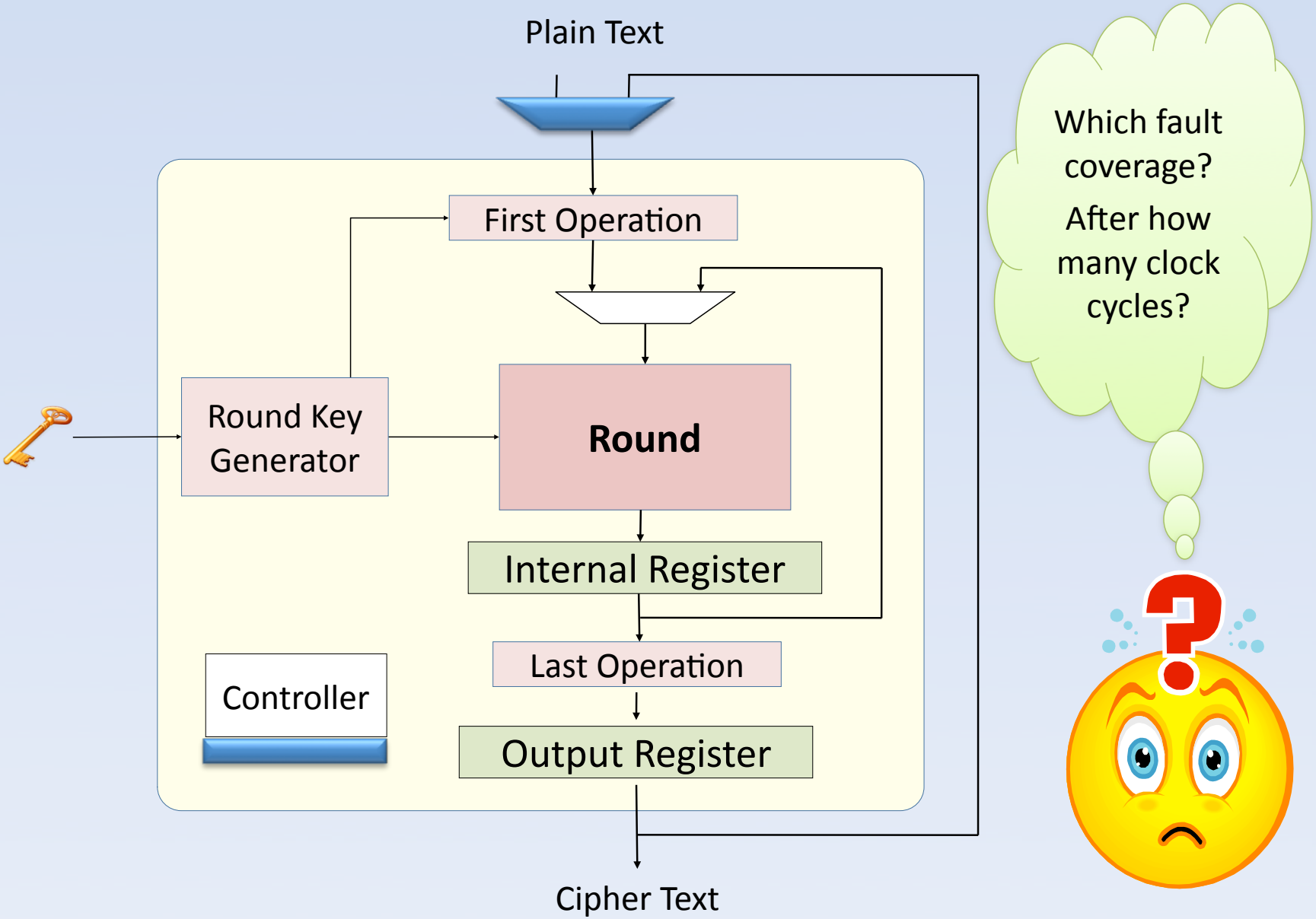
BIST Architecture



BIST Architecture



BIST Architecture



AES Self-test - Results

[M.-L. Flottes, at al. “AES-based BIST: Self-test, Test Pattern Generation and Signature Analysis”, DELTA'08]

AES Self-test - Results

- Result :
  - Fault coverage: 100% always before 2400 clock cycles (several keys, several plaintexts)

[M.-L. Flottes, at al. “AES-based BIST: Self-test, Test Pattern Generation and Signature Analysis”, DELTA'08]

AES Self-test - Results

- Result :
  - Fault coverage: 100% always before 2400 clock cycles (several keys, several plaintexts)
  - Area overhead:

		AES	BIST AES
Round	SubBytes ShitRow MixColumn AddRoundKey	10192 0 301 423	10192 0 301 423
Control Unit		67	121
Key generator		3409	3444
AES logic		932	1351
Total		**15324**	**15832**

Overhead : 507 cells (3.31%)

[M.-L. Flottes, at al. “AES-based BIST: Self-test, Test Pattern Generation and Signature Analysis”, DELTA'08]

Outline

- Test
- Security vs Test
  - Scan-based attacks
  - Securing the scan chains
  - BIST as an alternative
- **Conclusions**

Conclusions

- Scan based testing
  - Efficient but subject to scan-based attacks
  - It requires additional design efforts (secure DfT)
- BIST solutions
  - No attacks
  - Good fault coverage
  - No diagnosis

Thank you