Design for Hardware Trust

- Since detecting Trojan is extremely challenging, design for hardware trust approaches are proposed to

  - **Improve hardware Trojan detection methods**
    - Improve sensitive to power and delay
    - Rare event removal

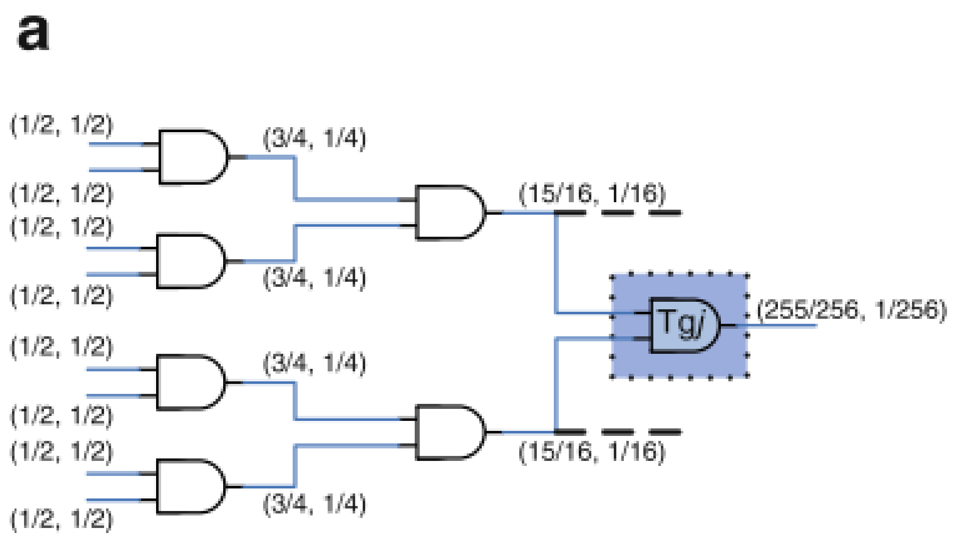
  - **Prevent hardware Trojan insertion**
    - Design obfuscation

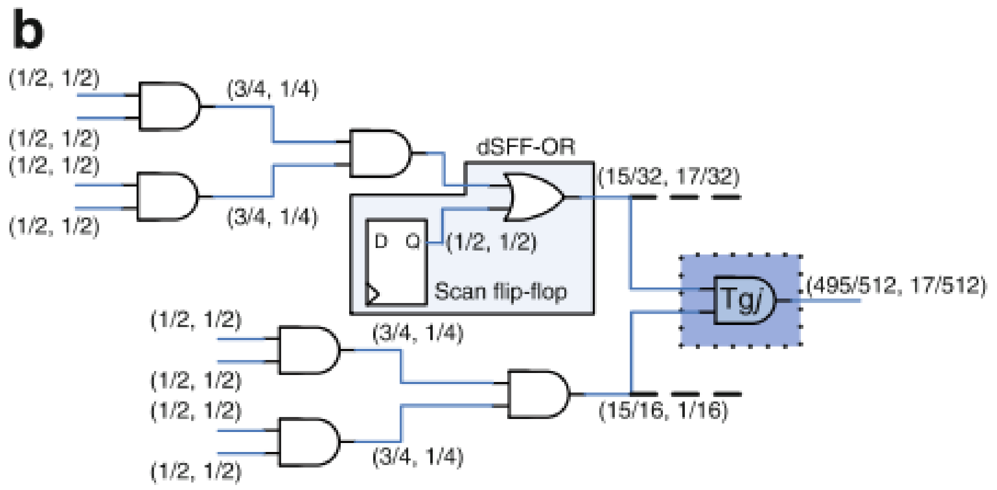
Rare Event Removal

- Intelligent attackers will choose low-frequency events to trigger the inserted Trojans.
- Improving controllability or observability can make rare events scarce, thereby facilitating detecting Trojans inside the design.
  - Design for Trojan test: inserting probing points
  - Inserting dummy scan flip-flops

Increasing Probability of Partial/Full Activation

- Inserting dummy FFs on path with very low activation probability
- To increase transition probability of nets whose transition probability is lower than a sepcific probability threshold, some efficient dummy flip-flops are inserted at these nets.
- In the figure, for each net the first value is probability of “0” and the second value is probability of “1”.
- In figure a, the transition probability of Trojan gate is low because of low transition probability on two inputs.
- If one dummy filp-flop is inserted at the first input in figure b, the transition probability of Trojan gate increases, so that it can be activated easier.





Increasing Probability of Partial/Full Activation

- Dummy scan flip-flops are inserted to control hard-to-excite nodes.
- Dummy flip-flops insertion technique is useful in full activation, power-based, and delay-based hardware Trojan detection methods.

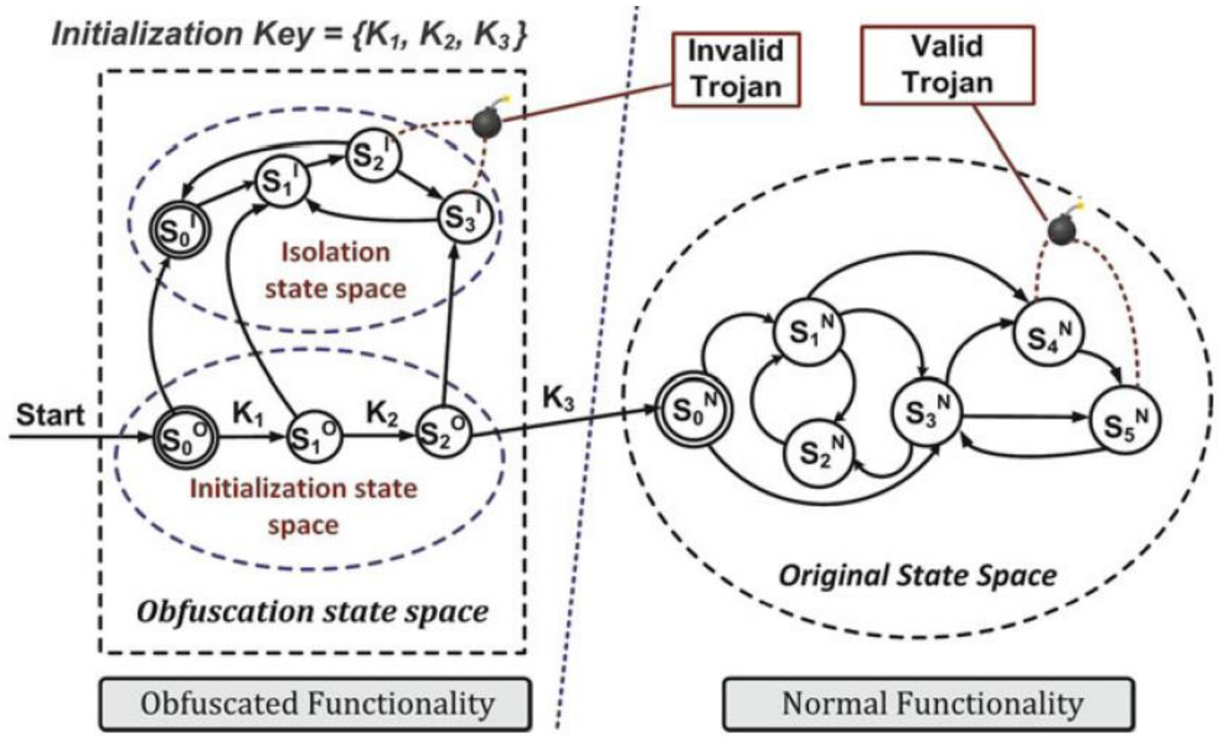
- Usage:
  - **Full activation**: increase controllability
  - **Power-based**: generate switching activities
  - **Delay-based**: activate more paths to improve coverage

Trojan Prevention-Design obfuscation

- The objective is deterring attackers from inserting Trojans inside the design.
- Design obfuscation means that a design will be transformed to another one which is functionally equivalent to the original, but in which it is much harder for attackers to obtain complete understanding of the internal logic, making reverse engineering much more difficult to perform.
- It obfuscates the state transition function to add an obfuscated mode on top of the original functionality (called normal mode).
- For example, you can add an obfuscated mode on top of the original functionality of a state transition function.

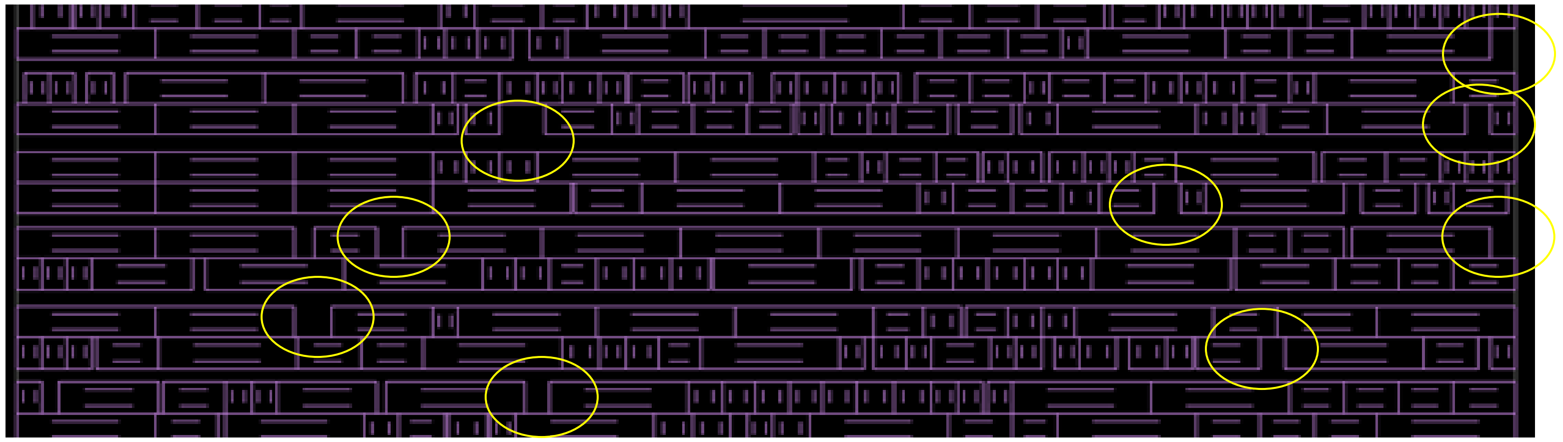
Design obfuscation

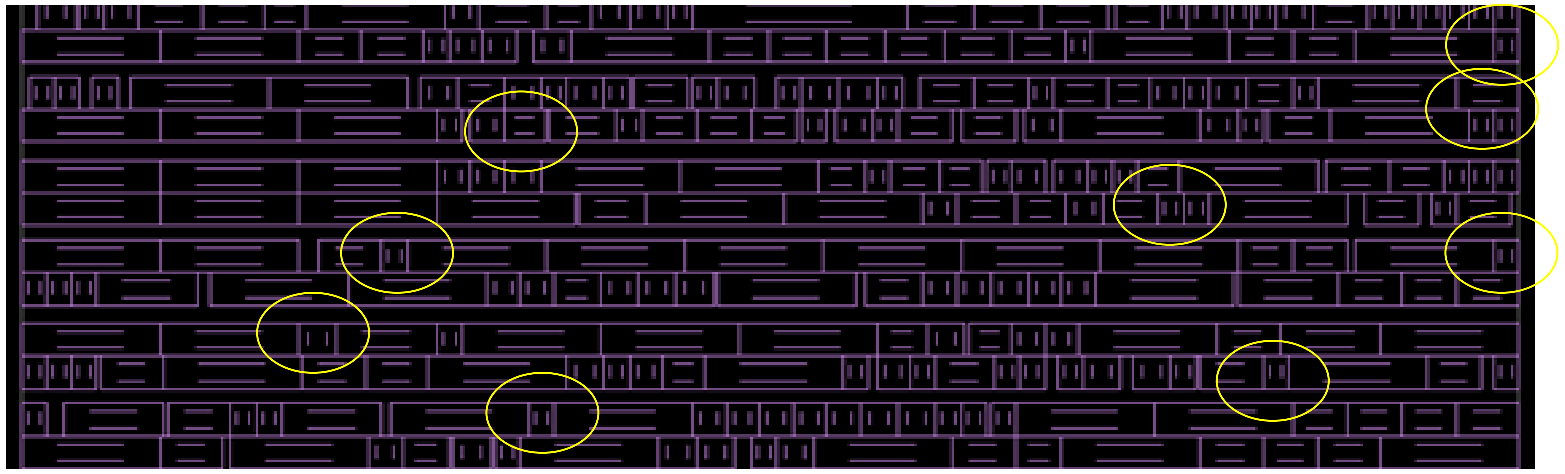
- Specified pattern is able to guide the circuit into its normal mode.
- The transition arc K3 is the only way the design can enter normal operation mode from the obfuscated mode.
- Only designer knows the specified pattern (initialization key) to access normal function.
- If a Trojan is inserted in the obfuscated circuits, it will not affect the original functionality. This is an invalid Trojan in the figure.
- If a Trojan is inserted in the original circuits, it will threaten the original circuits.



BISA: Built-In Self-Authentication

- Filling all unused spaces with a circuit that can easily test itself





What is BISA

- BISA stands for Built-In Self-Authentication. It refers to a mechanism integrated into hardware systems, particularly chips and processors, to enable self-authentication. BISA ensures that the hardware can authenticate itself without relying on external systems, improving security by preventing unauthorized access or tampering.
- BISA helps in mitigating security risks in hardware, especially in environments where components may be vulnerable to attacks, counterfeiting, or unauthorized modifications.
- Key Features of BISA:
  - Self-contained authentication: The system can verify its own integrity and authenticity, ensuring that only authorized components are in use.
  - Security against tampering: BISA is designed to protect hardware from malicious attacks, counterfeit components, or tampering in the supply chain.
  - Low overhead: Since the authentication is built into the hardware itself, it minimizes the need for external security checks, making it efficient.

Common Examples of BISA

1. Hardware Security Modules (HSMs): Certain chips and devices use BISA to authenticate themselves when connecting to a network or communicating with other devices, ensuring they haven’t been tampered with.
2. IoT Devices: In Internet of Things (IoT) systems, BISA can ensure that only authorized devices are connected to a network, preventing malicious or counterfeit devices from communicating with the system.
3. Secure Boot Process in Processors: Many modern processors use BISA in their secure boot processes to verify that the firmware and hardware configurations haven’t been altered before allowing the system to boot.
4. Authentication of Cryptographic Modules: Chips with embedded cryptographic functions often use BISA to verify the integrity of encryption keys and cryptographic algorithms stored in hardware.