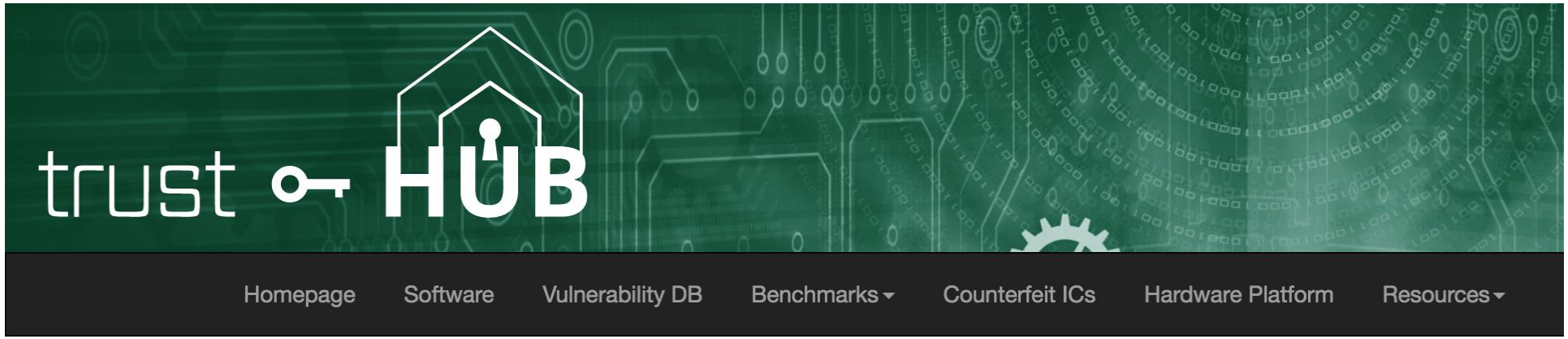
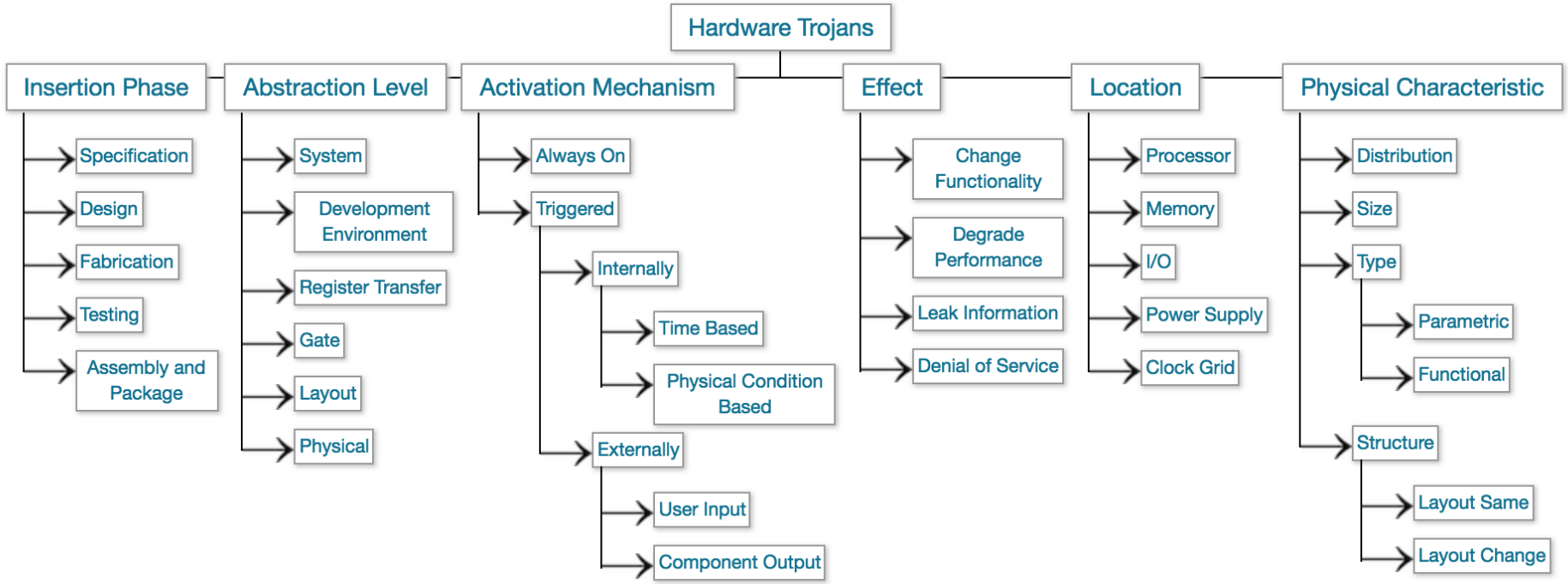
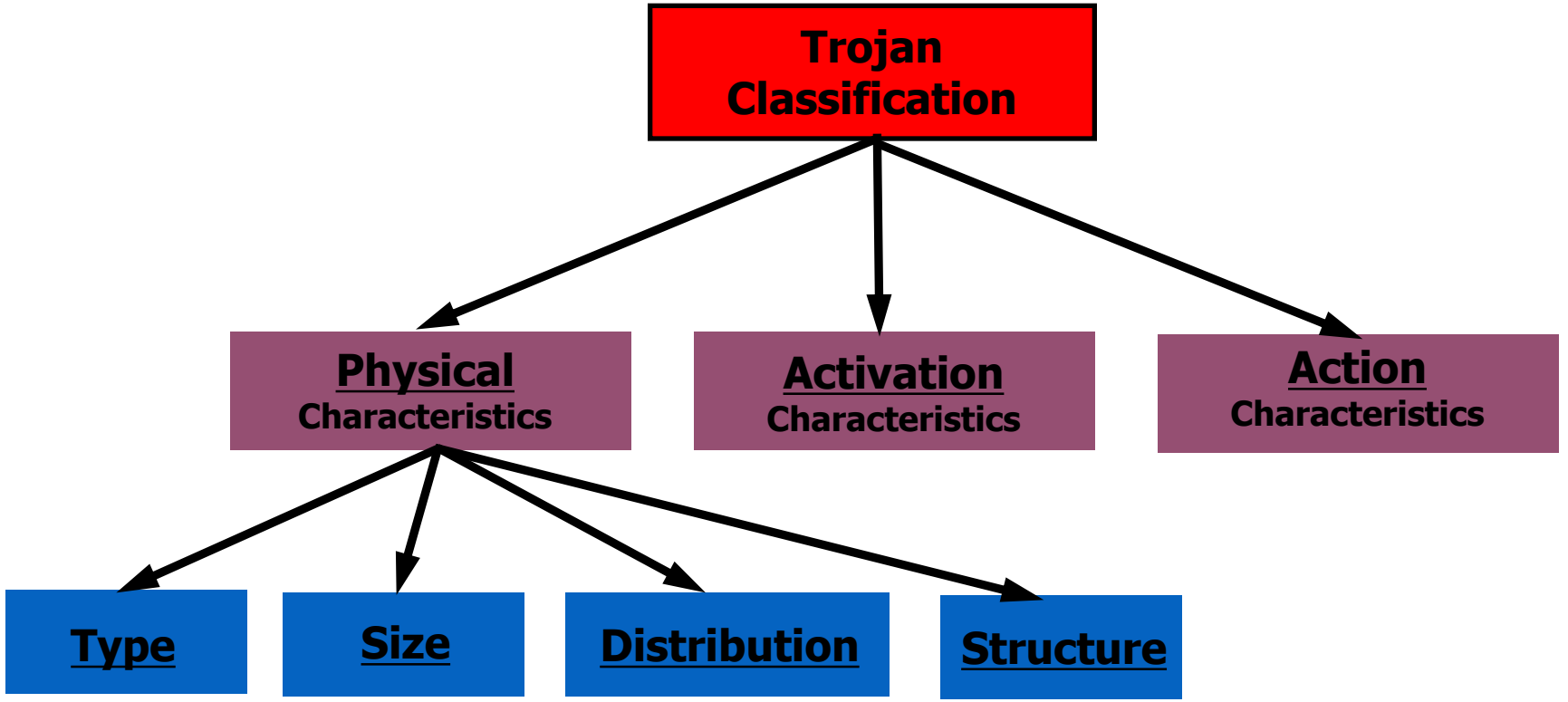
Trojan Taxonomy



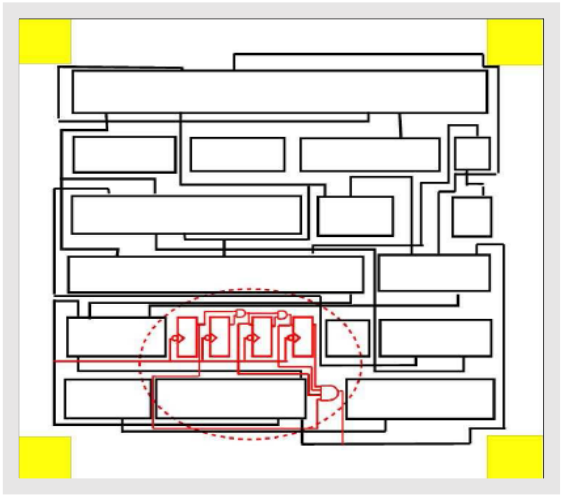


Trojan Taxonomy

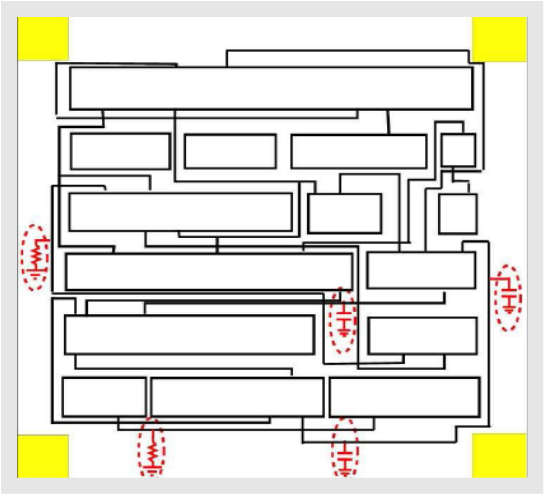


Example: Type

Functional



Parametric

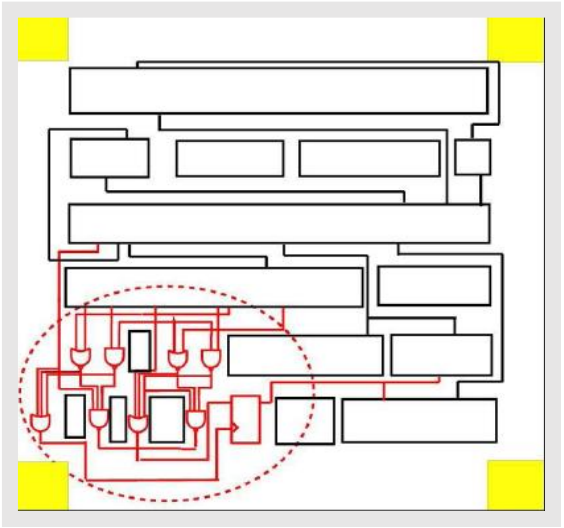


- **Functional**
  - **Addition or deletion of components**
  - **Sequential circuits**
  - **Combinational circuits**
  - **Modification to function or no change**

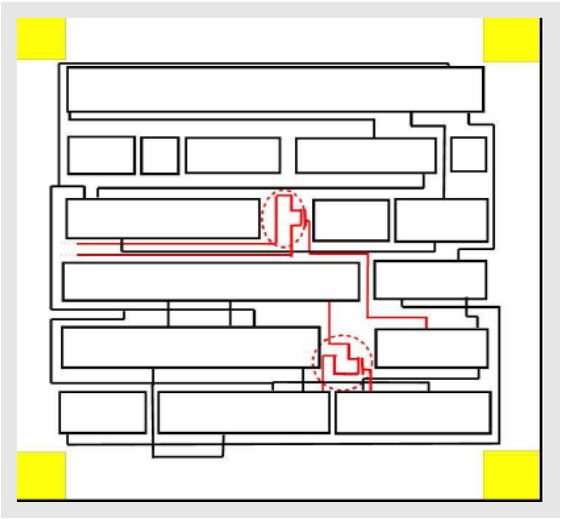
- **Parametric**
  - **Modifications of existing components**
    - **Wire: e.g. thinning of wires**
    - **Logic: Weakening of a transistor, modification to physical geometry of a gate**
    - **Modification to power distribution network**
  - **Sabotage reliability or increase the likelihood of a functional or performance failure**

Example: Size

Big



Small

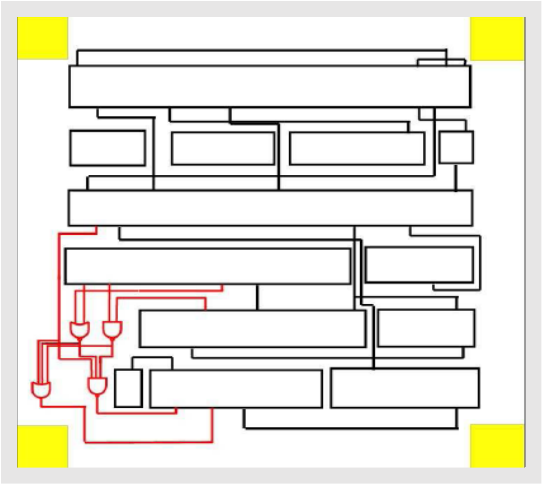


- **Size:**
  - **Number of components added to the circuit**
    - **Small transistors**
    - **Small gates**
    - **Large gates**

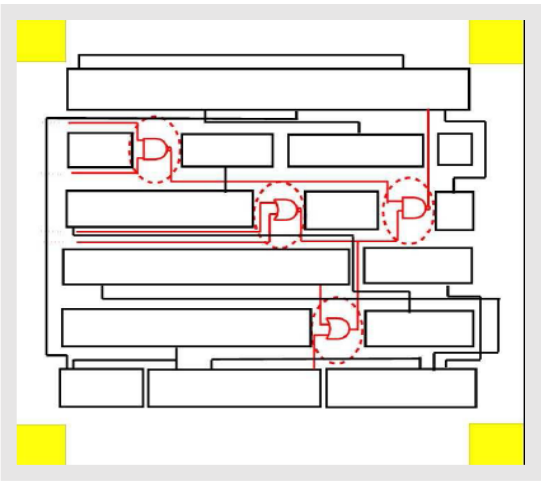
- **In case of layout, depends on availability of:**
  - **Dead spaces**
  - **Filler cells**
  - **Decap cells**
  - **Change in the structure**

Example: Distribution

Tight



Loose



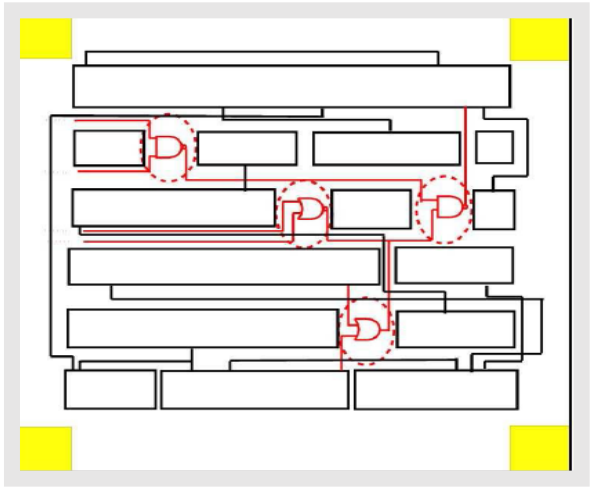
- **Tight Distribution**
  - **Trojan components are topologically close in the layout**

- **Loose Distribution**
  - **Trojan components are dispersed across the layout of a chip**

- **Distribution of Trojans depends on the availability of dead spaces on the layout**

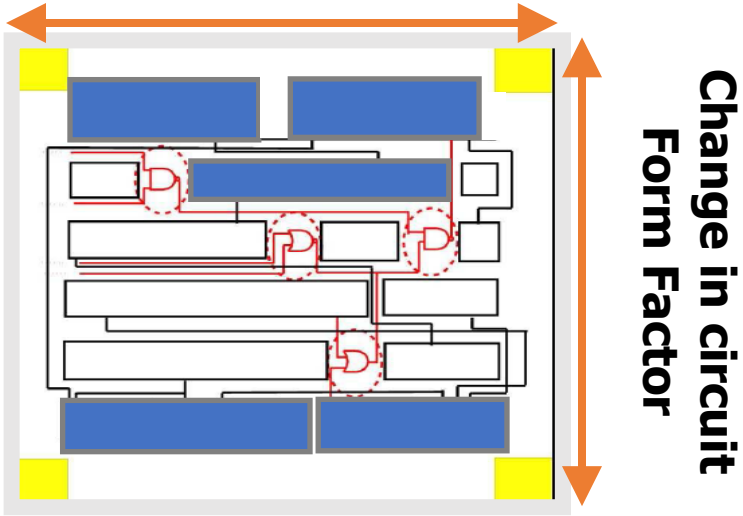
Example: Structure

No-change



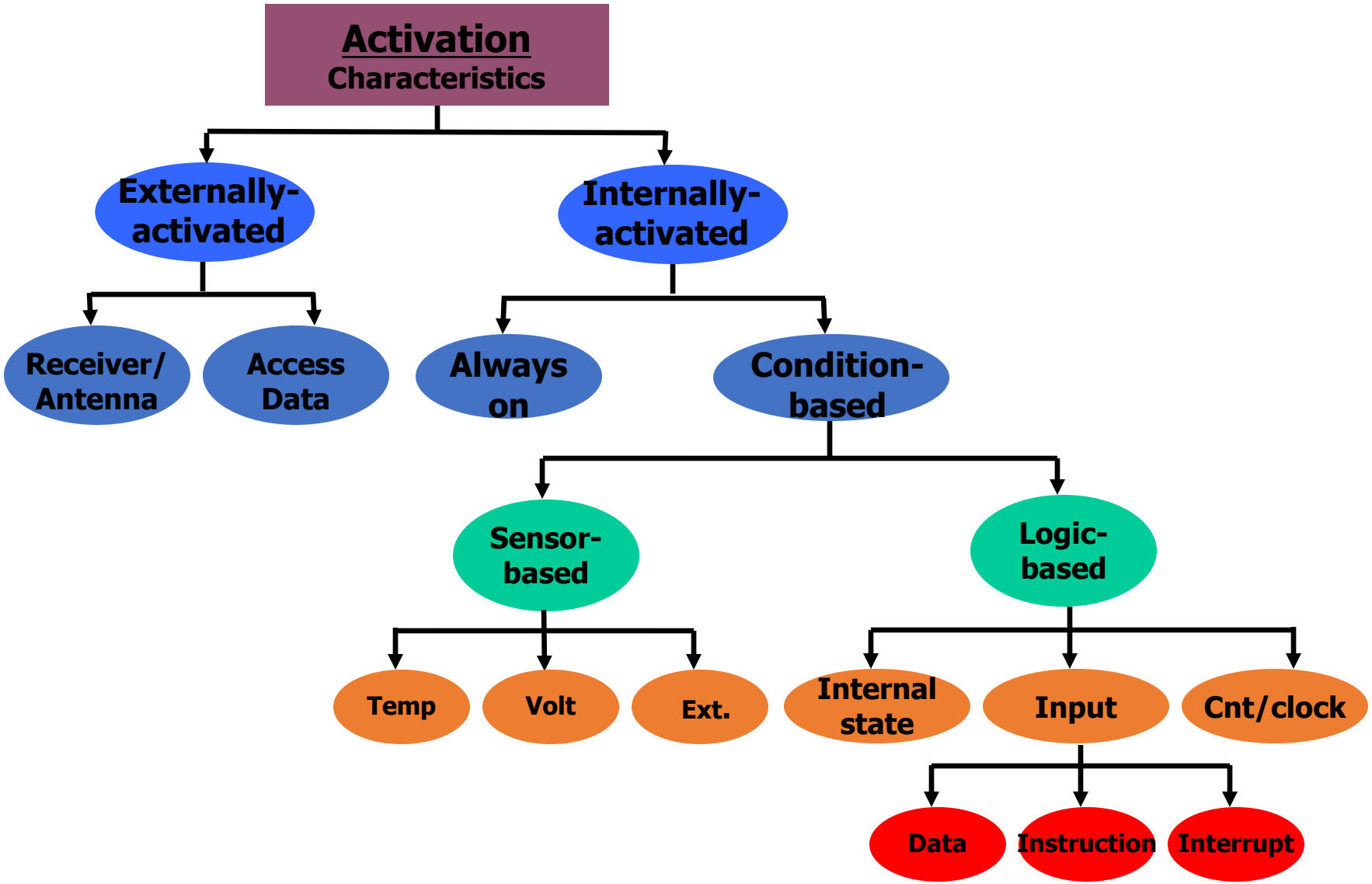
- The adversary may be forced to regenerate the layout to be able to insert the Trojan, then the chip dimensions change
  - It could result in different placement for some or all the design components

Modified Layout

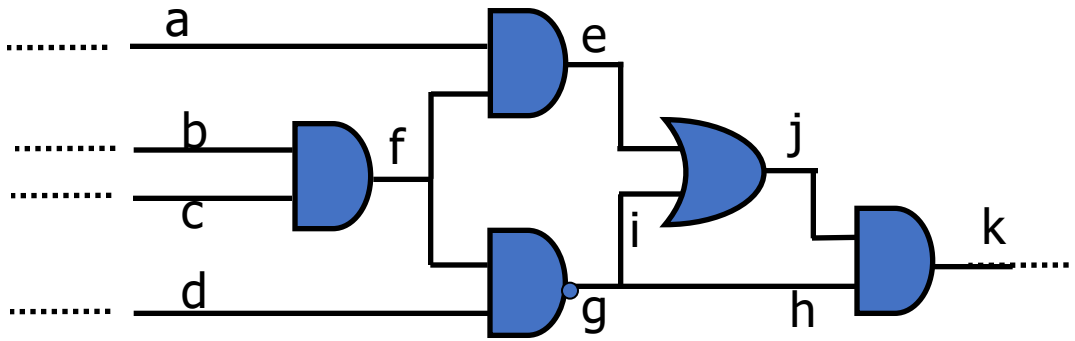


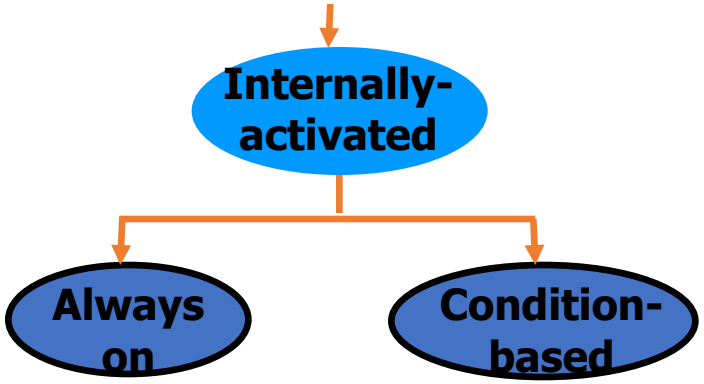
- A change in physical layout can change the delay and power characteristics of chip
  - It is easier to detect the Trojan

Trojan Taxonomy: Activation

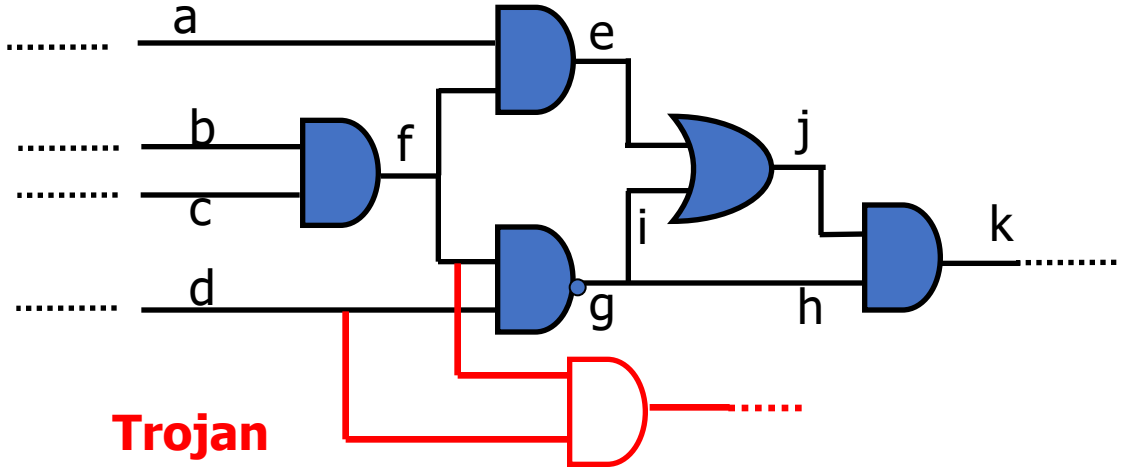


Activation: Internally Activated



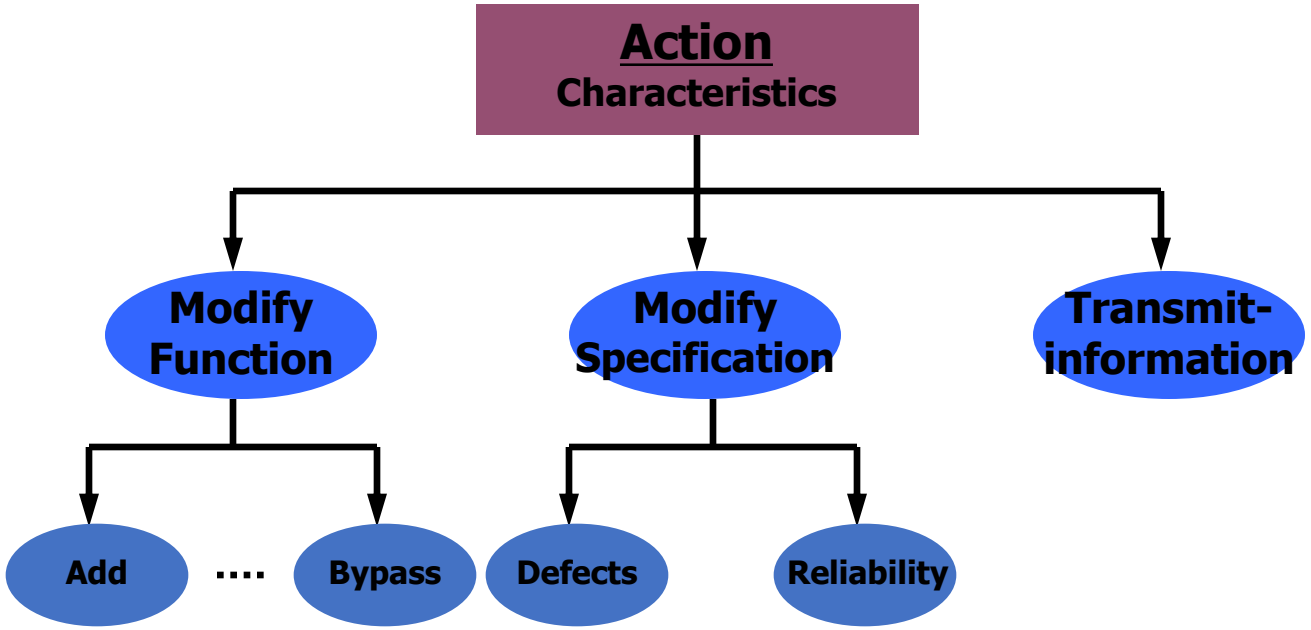


Trojan

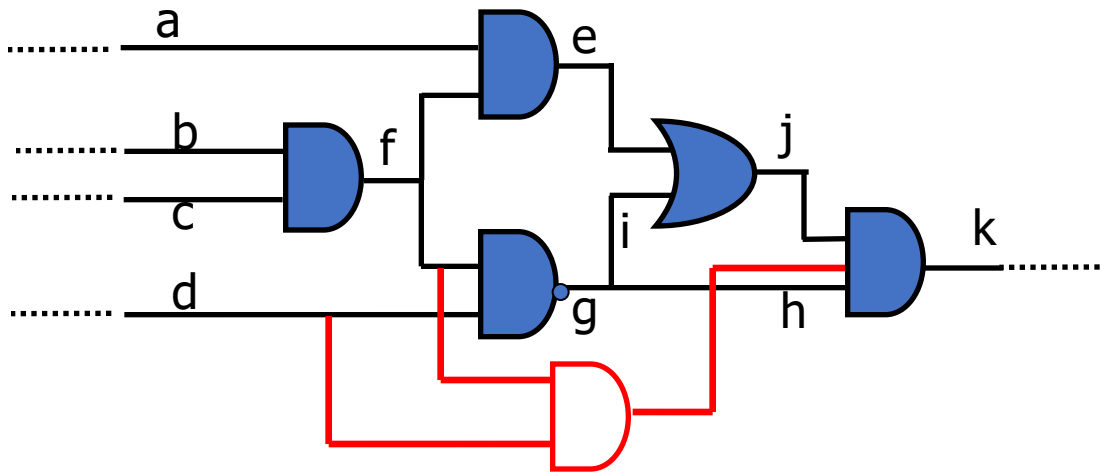


Trojan

Trojan Taxonomy: Action



Example: Action

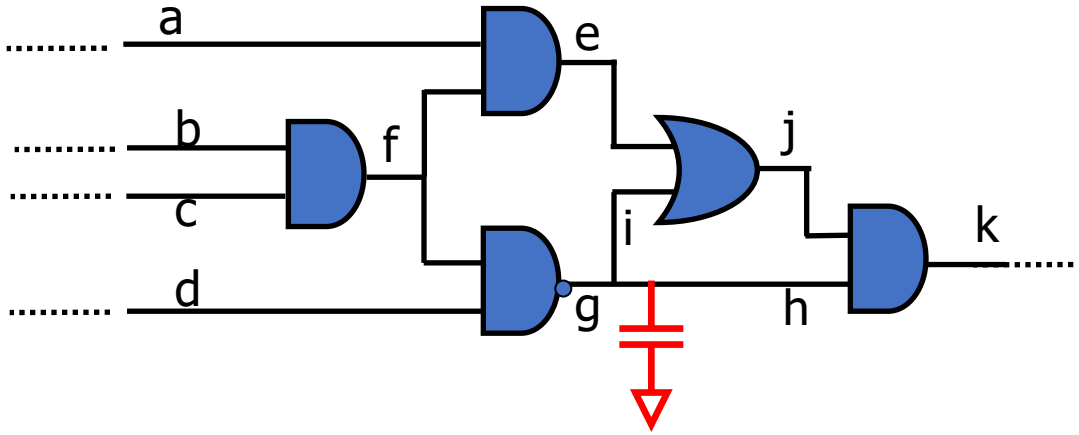


Modify Function

Trojan

Modify Specification:

Noise, Delay and Temperature



IP Trust & IP Security

- **IP Trust**
  - Detect ***malicious*** circuits inserted by IP designers
    - **Goal to Verify Trust**: Protect IP buyers, e.g., SoC integrators
  - Focus of this lecture

- **IP Security**
  - Information leakage, side-channel leakage, backdoors, functional bugs and flaws, illegal IP use/overuse, etc.
    - **Goal to Verify Security**: Protect application

IP Trust

- IPs from untrusted vendors need to be verified for trust before use in a system design
- **Problem statement:** How can one establish that the IP does exactly as the specification, nothing less, nothing more?

- **IP Cores**:
  - Soft IP, firm IP and hard IP

- **Challenges**:
  - No known golden model for the IP
    - Spec could be assumed as golden
  - Soft IP is just a code so that we cannot read its implementation

Approaches for Pre-synthesis

- **Formal verification**
  - Property checking
  - Model checking
  - Equivalence checking
- **Coverage analysis**
  - Code coverage
  - Functional coverage

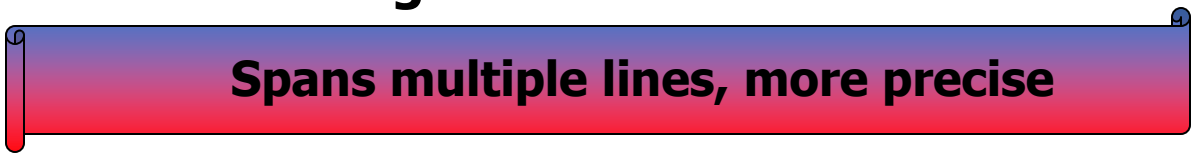
Formal Verification

- **Formal verification**
  - Ensuring IP core is exactly same as its specification
  - Three types of verification methods
    - **Property checking**: Every *requirement* is defined as assertion in testbench and is checked
    - **Equivalence checking**: Check the equivalence of RTL code, gate-level netlist and GDSII file
    - **Model checking**
      - System is described in a formal model (C, HDL)
      - The desired behavior is expressed as a set of properties
      - The specification is checked against the model

Coverage Analysis

- **Code coverage**
  - **Line coverage**

  - **Statement Coverage**



  - **FSM Coverage**

  - **Toggle**

- **Function coverage**
  - **Assertion**

Suspicious Parts

- **If one of the assertions fails, the IP is assumed untrusted.**
- **If coverage is not 100%, *uncovered* parts of the code (RTL, netlist) are assumed suspicious.**