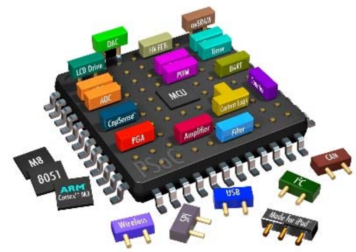
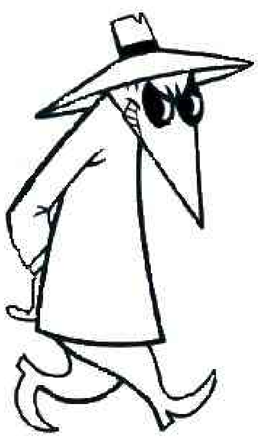
Detecting Hardware Trojans: A Tale of Two Techniques





Sharad Malik

sharad@princeton.edu

FMCAD 2015

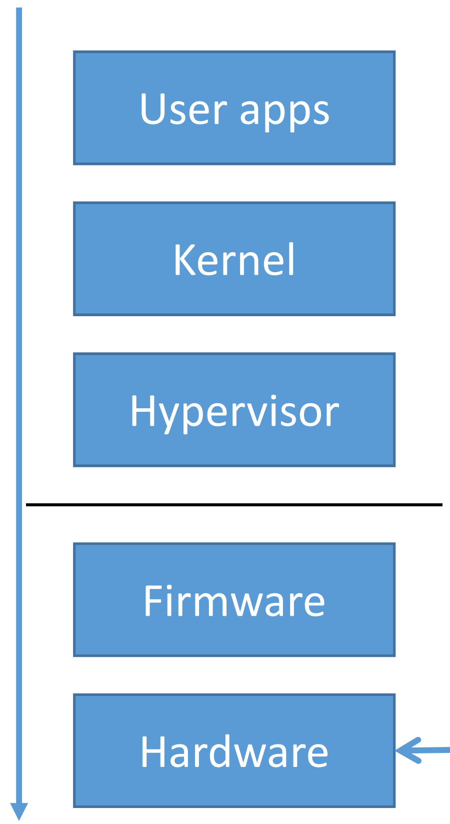
PRINCETON UNIVERSITY

Hardware Security and Hardware Trojans

Each layer trusts all layers below it

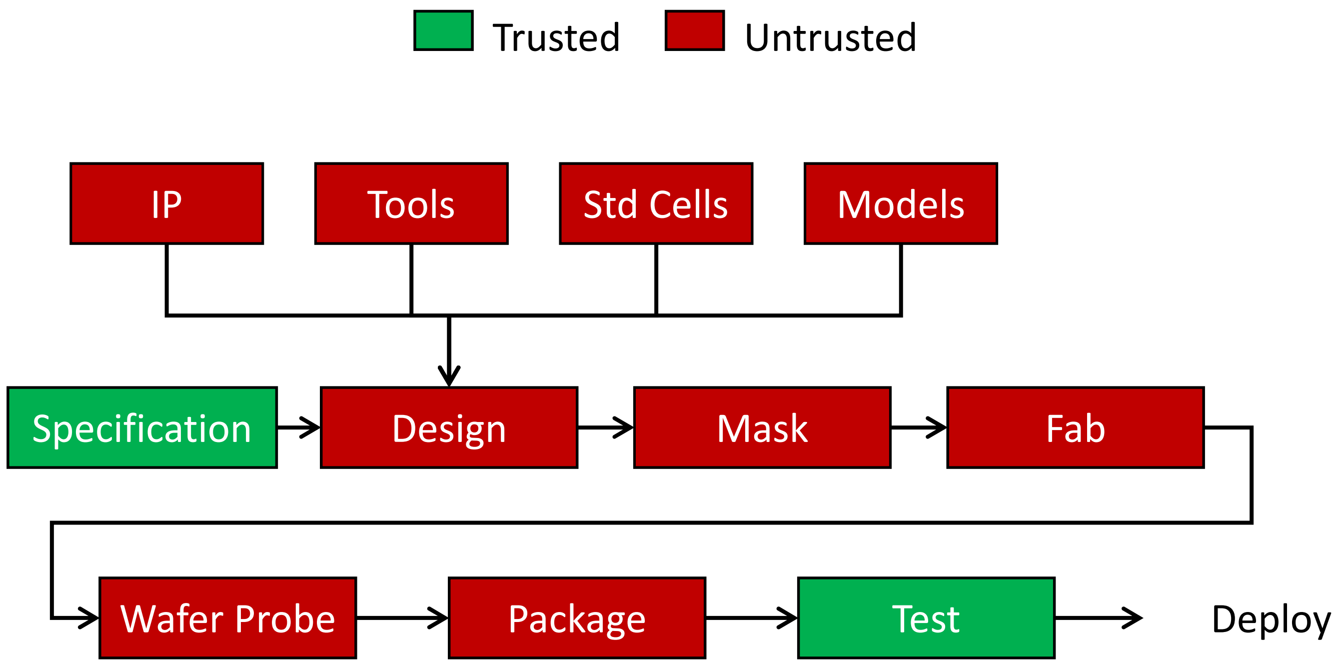
- More privilege
- Widely used platforms
- Difficult to patch

  ⇒ more damage



A Hardware Trojan is a **malicious intentional modification** of an electronic circuit or design, resulting in undesired behavior

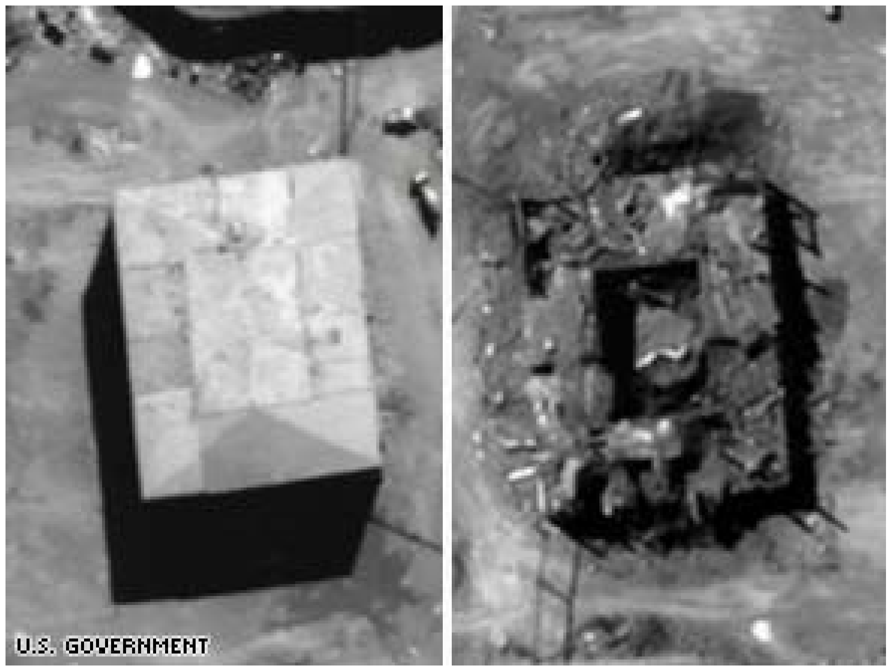
Where are the Vulnerabilities?



[Source: Brian Sharkey, *TRUST in Integrated Circuits Program: Briefing to Industry*, DARPA MTO, 26 March 2007]

A Real Threat?

Before/after pictures of a suspected nuclear reactor site



Suspicion that a hardware backdoor was exploited to disable the radar system

[Sally Adee, *The Hunt for the Kill Switch*, IEEE Spectrum May 2006]
[John Markoff, *Old Trick Threatens the Newest Weapons*, NY Times, 26 October 2009]

EXPLORE TOPICS

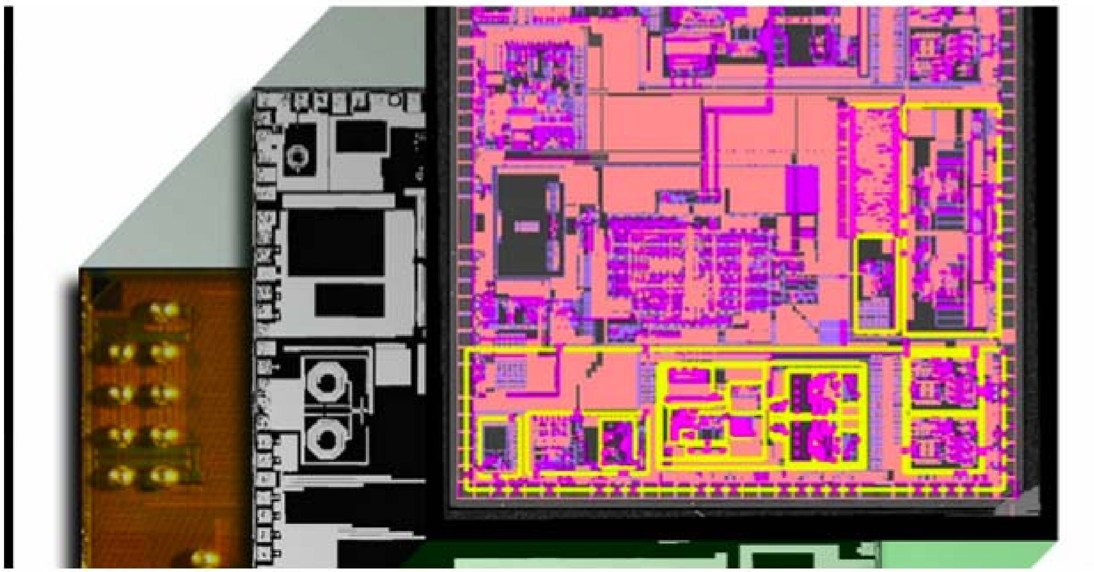
DARPA DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

ABOUT US / OUR RESEARCH / NEWS / EVENTS / WORK WITH US /

Defense Advanced Research Projects Agency > Program Information > Integrity and Reliability of Integrated Circuits (IRIS)

Integrity and Reliability of Integrated Circuits (IRIS)

Mr. Kerry Bernstein



Malicious circuits in a design

The integrated circuit (IC) is a core component of many electronic systems developed for the Department of Defense. However, the DoD consumes a very small percentage of the total IC production in the world. As a result of the globalization of the IC marketplace, much of the advanced IC production has moved to offshore foundries, and these parts make up the majority of ICs used in today's military systems.

Without the ability to influence and regulate the off-shore fabrication of ICs, there is a risk that parts acquired for DoD systems may not meet stated specifications for performance and reliability. This risk increases considerably with the proliferation of counterfeit ICs in the marketplace, as well as the potential for the introduction of malicious circuits into a design.

Acknowledgements

DARPA IRIS Project

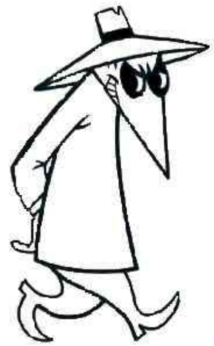
SRI
- Bruno Dutertre
- Adria Gascon
- Dejan Jovanovic
- Maheen Samad
- Natarajan Shankar
- Ashish Tiwari

Princeton
- Burcin Cakir
- Kanika Pasricha
- Dillon Reisman
- Pramod Subramanyan
- Adriana Susnea
- Nestan Tsiskaridze

UC Berkeley
- Wenchao Li
- Sanjit Seshia
- Wei Yang Tan

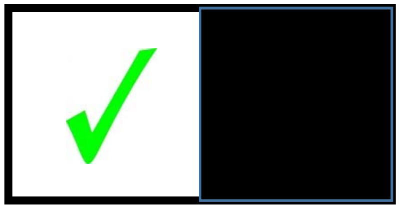
Center for Future Architectures Research (C-FAR)

- Burcin Cakir
- Pramod Subramanyan



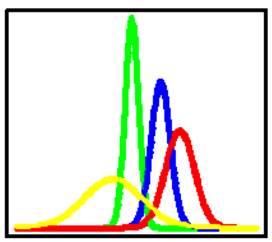
Logical Analysis

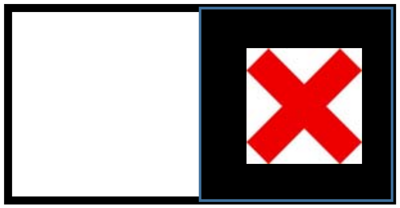
$\vdash$ $\exists$ $\forall$ $\vee$ $\neg$ $\wedge$



Whitelist

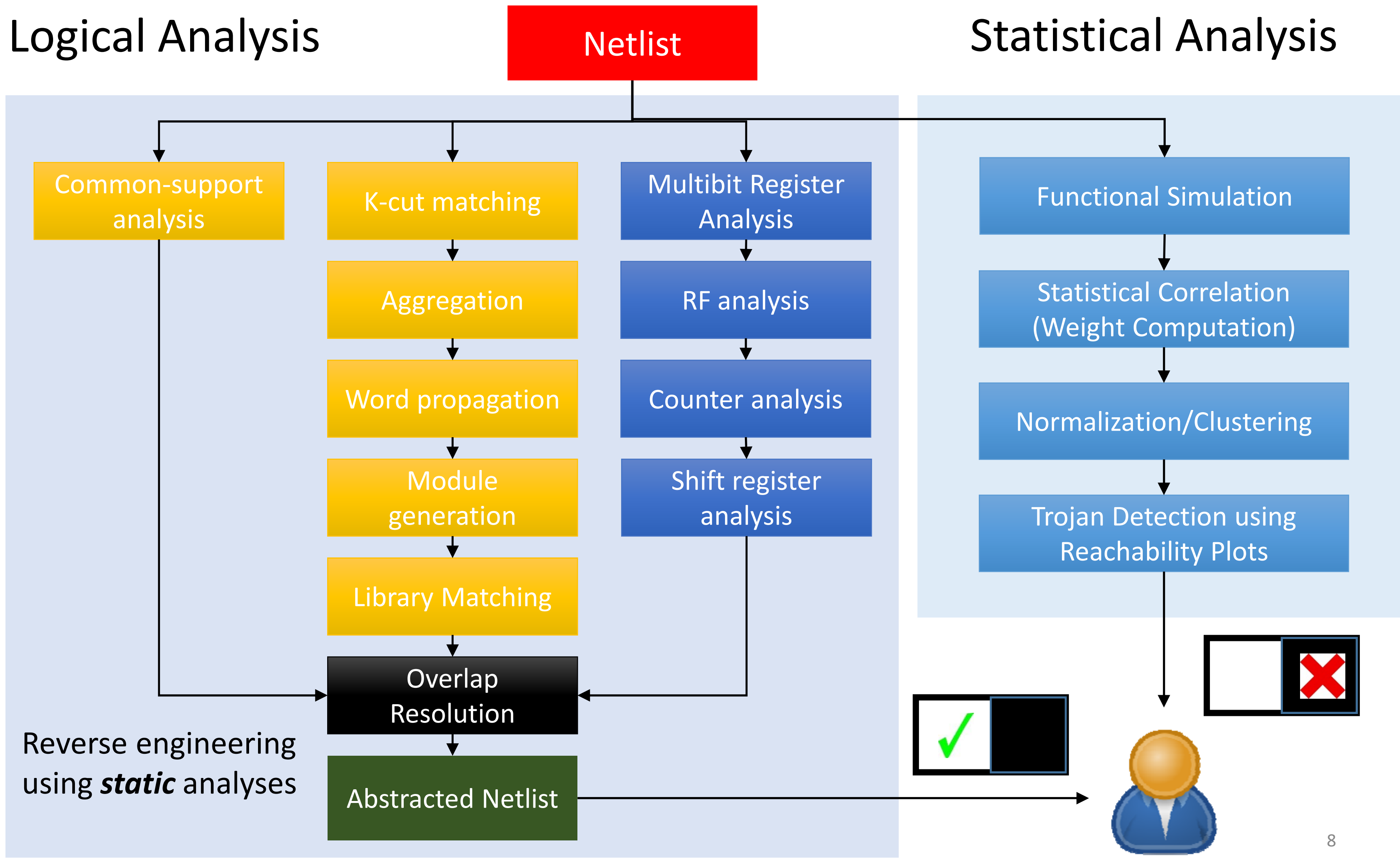
Statistical Analysis

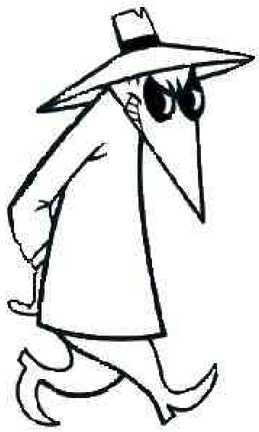




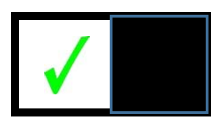
Blacklist

Netlist Analysis Portfolio

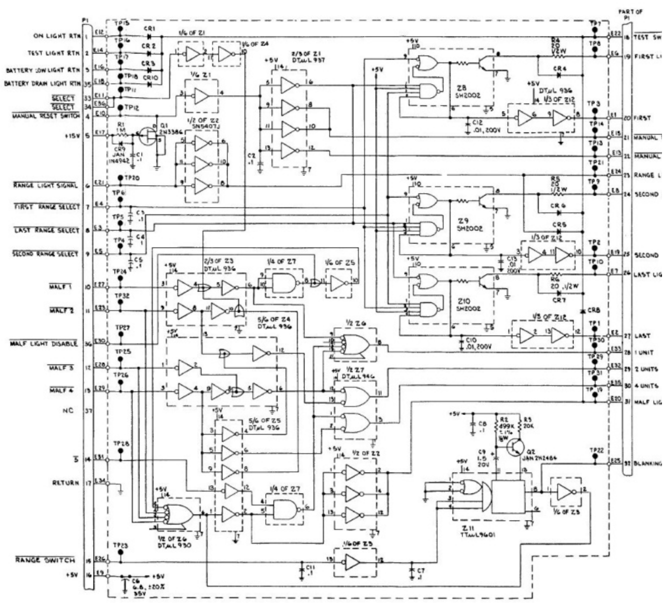




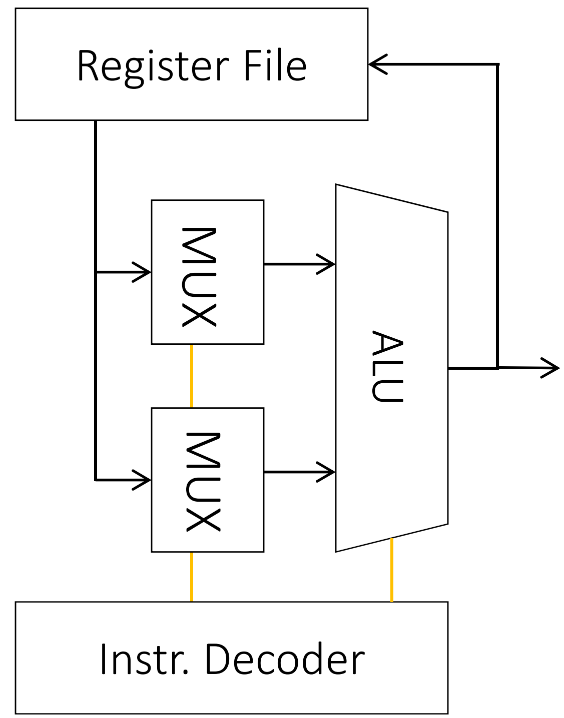
Logical Analysis for Reverse Engineering



Reverse Engineering Objective

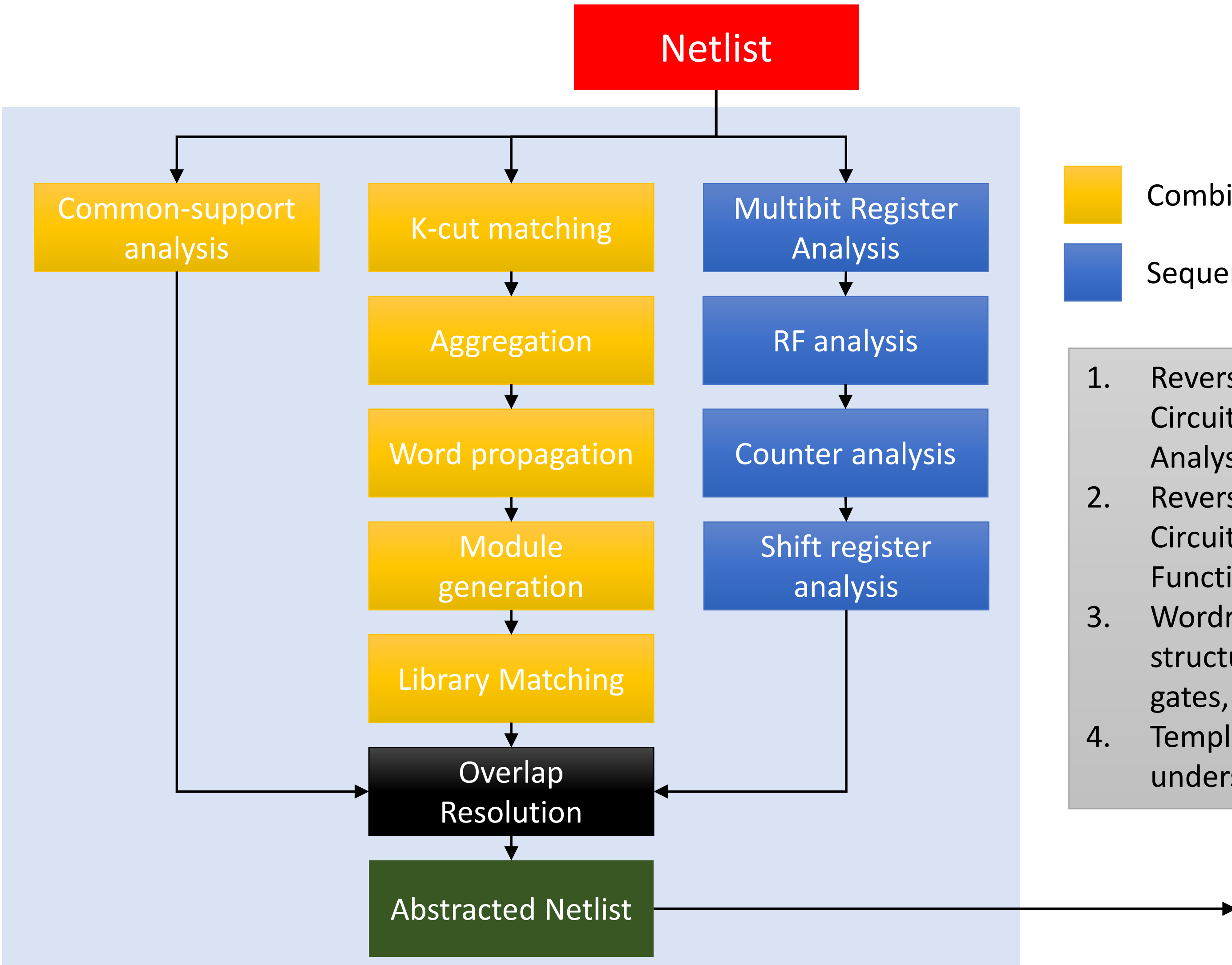


Source: http://miscpartsmanuals2.tpub.com/TM-9-1240-369-34/TM-9-1240-369-340115.htm



Extract high-level components from an unstructured and flat netlist

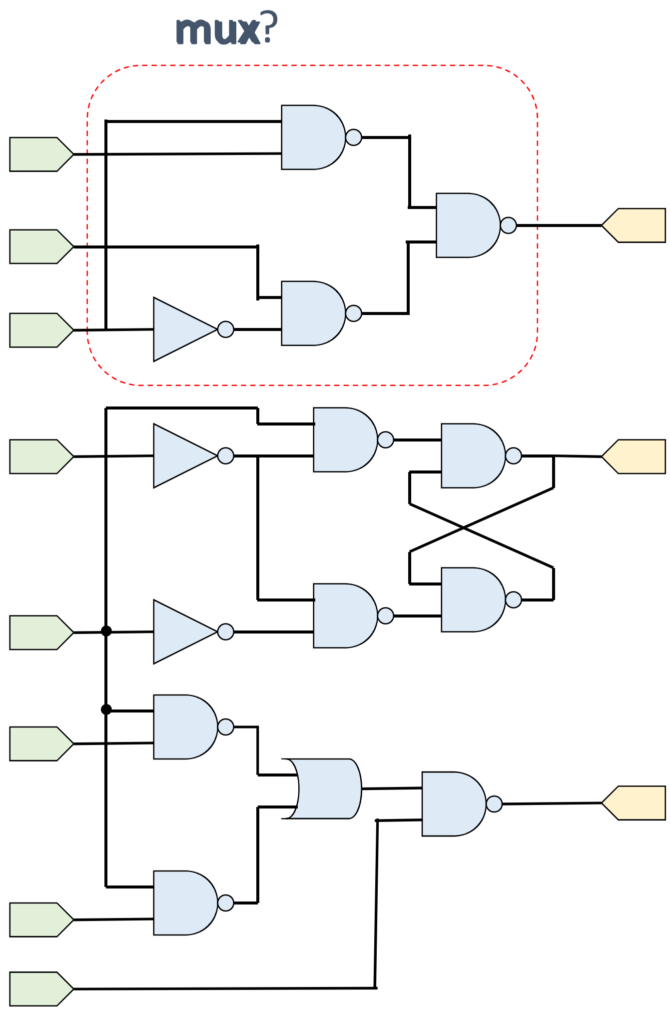
Reverse Engineering Portfolio



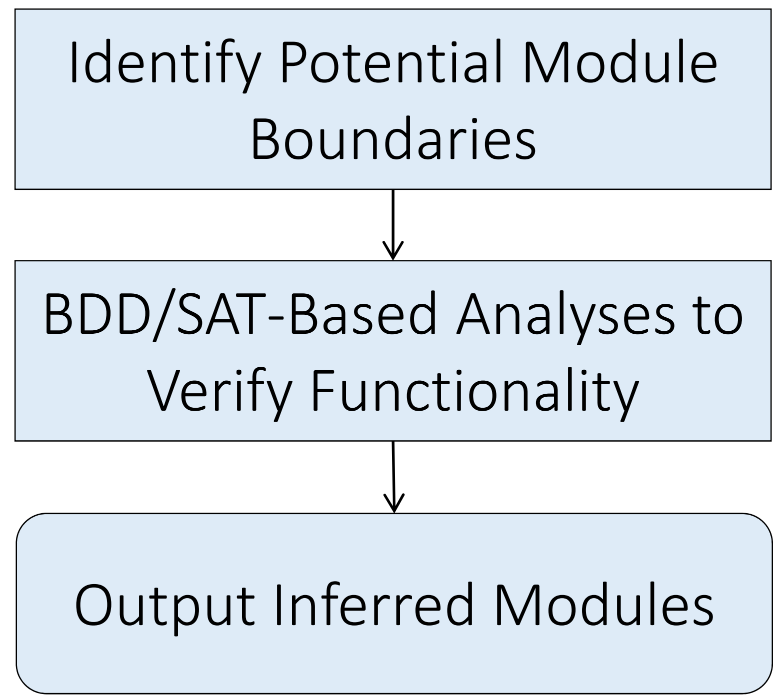
- Combinational component analyses
- Sequential component analyses

1. Reverse Engineering Digital Circuits Using Functional Analysis, [DATE'13]
2. Reverse Engineering Digital Circuits Using Structural and Functional Analysis, [TETC'14]
3. Wordrev: Finding word-level structures in a sea of bit-level gates, [HOST'13]
4. Template-based circuit understanding, [FMCAD'14]

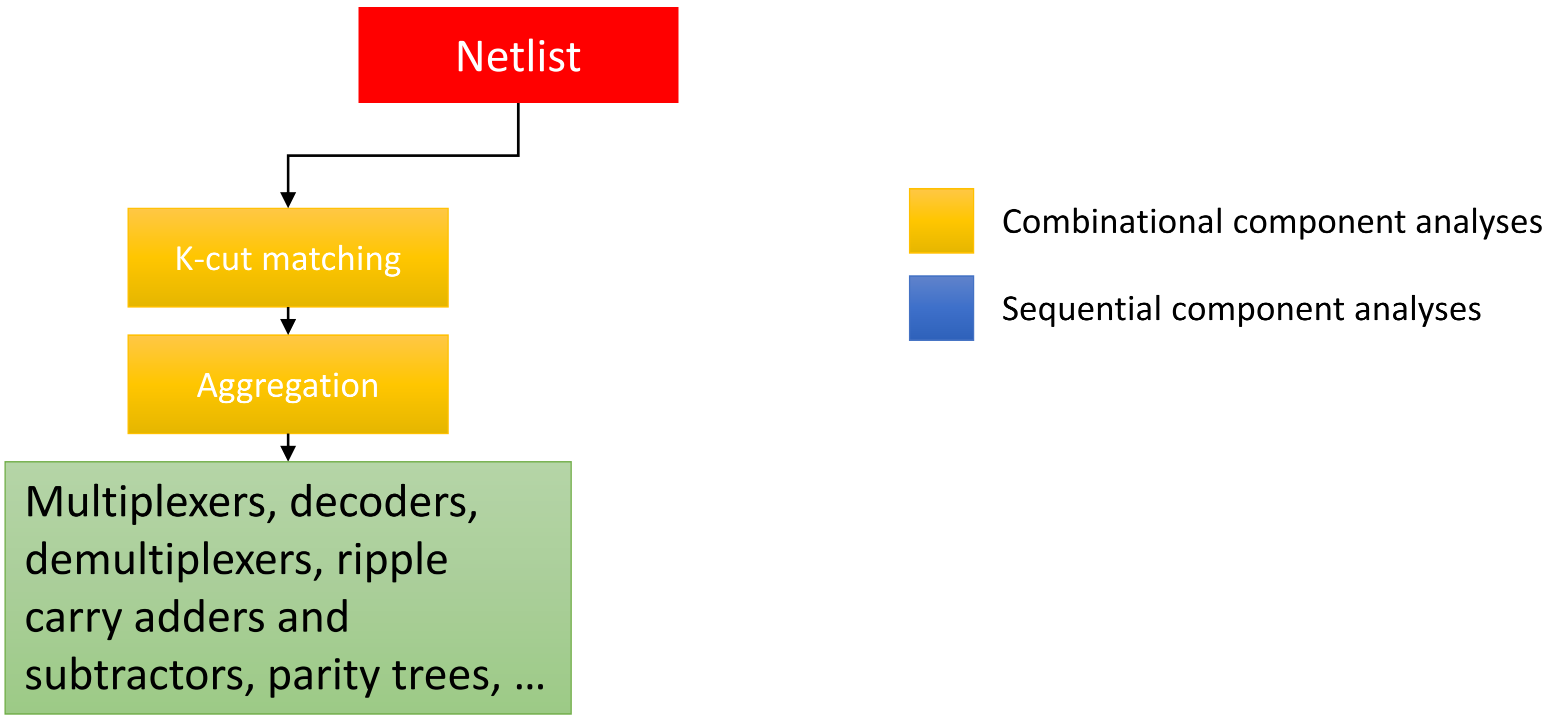
General Strategy



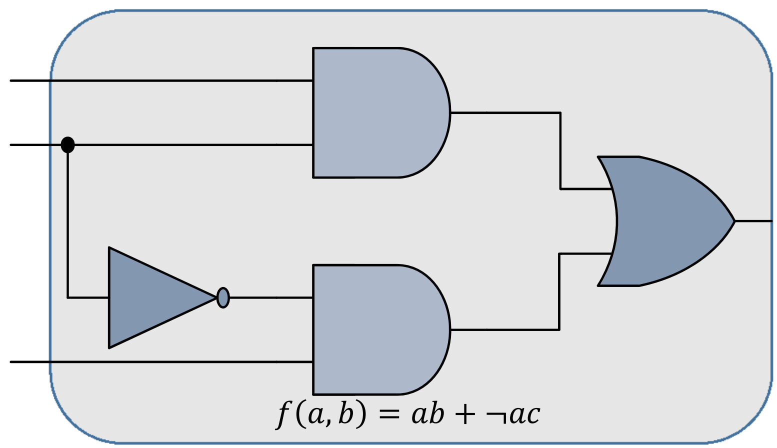
Main Challenge: Netlist is a sea of gates! No information about the boundaries of modules inside it!



Bitslice Identification and Aggregation



Bitslice Identification using Cut-based Matching



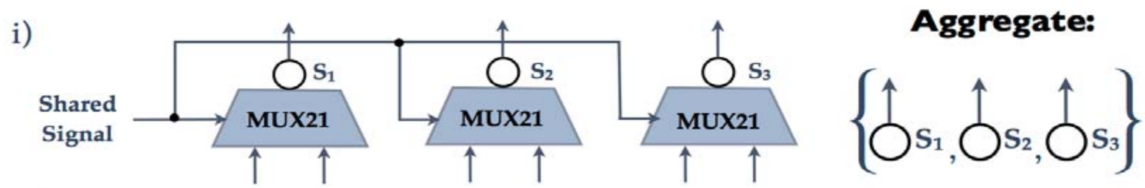
- Cuts are computed recursively
- Made tractable by enumerating cuts with $k \leq 6$ inputs
- Group cuts into equivalence classes using permutation independent comparison
- BDDs used to represent Boolean functions during matching

Cong and Ding, *FlowMap*, [TCAD'94]
Chatterjee et al., *Reducing Structural Bias in Technology Mapping*, [ICCAD'05]

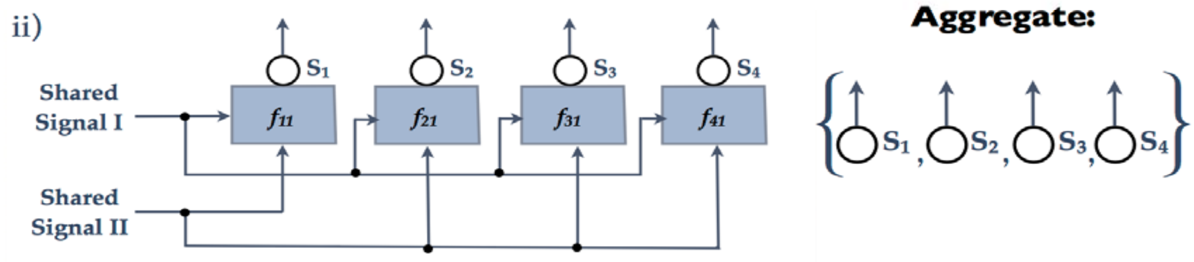
Bitslice Aggregation

Group Bitslices With Shared Signals

Multi-bit Multiplexer 2-to-1

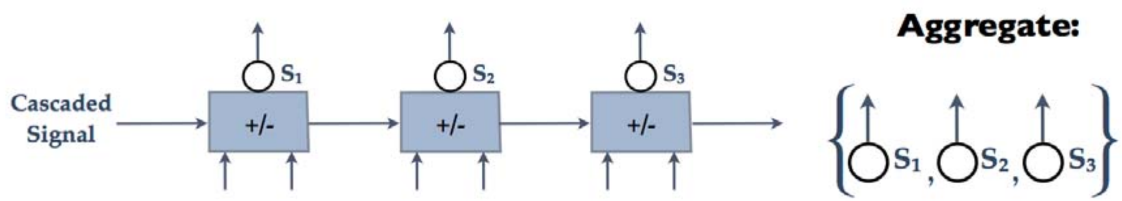


Decoder 2-to-4

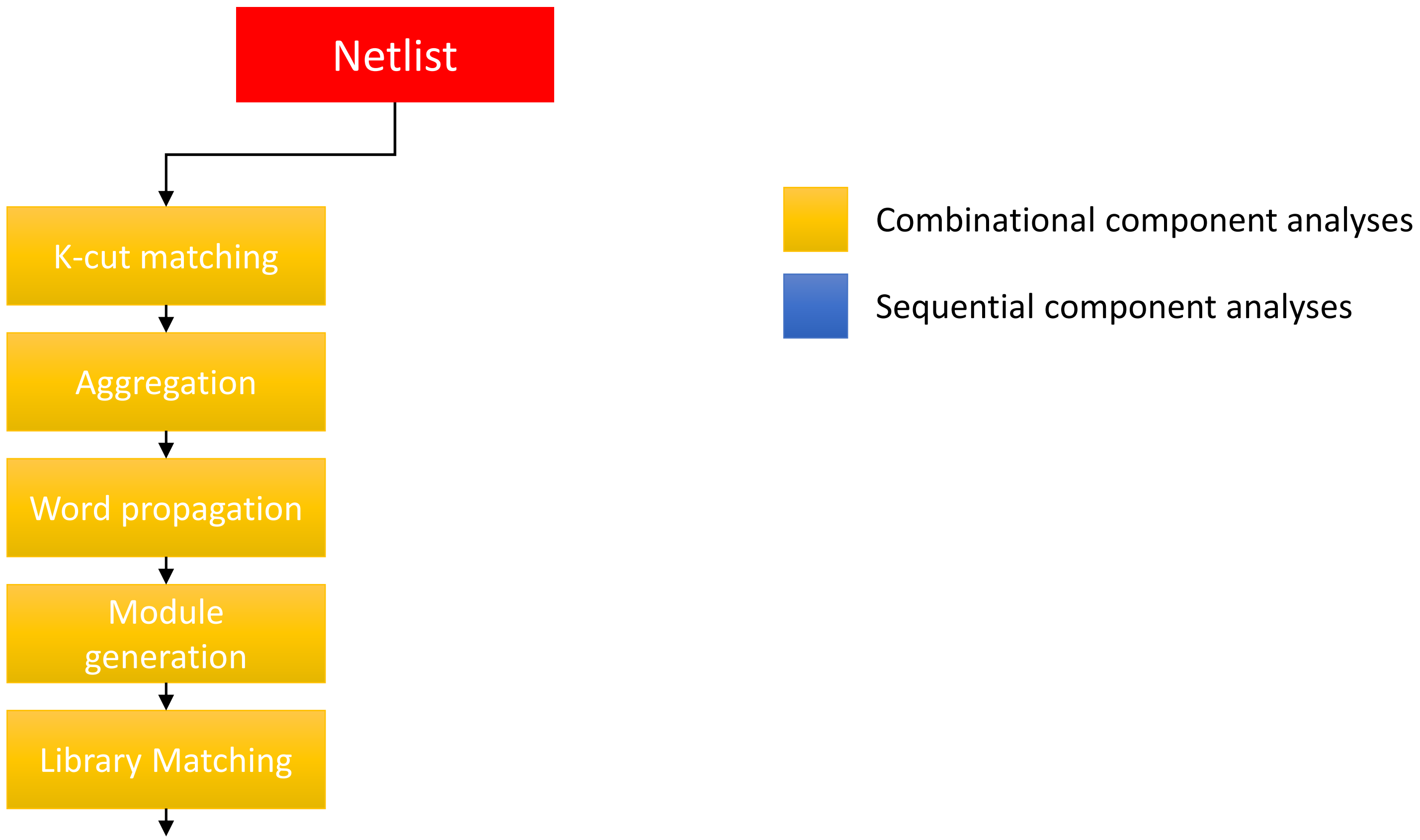


Group Bitslices With Cascading Signals

Ripple Carry Adder/Subtractor



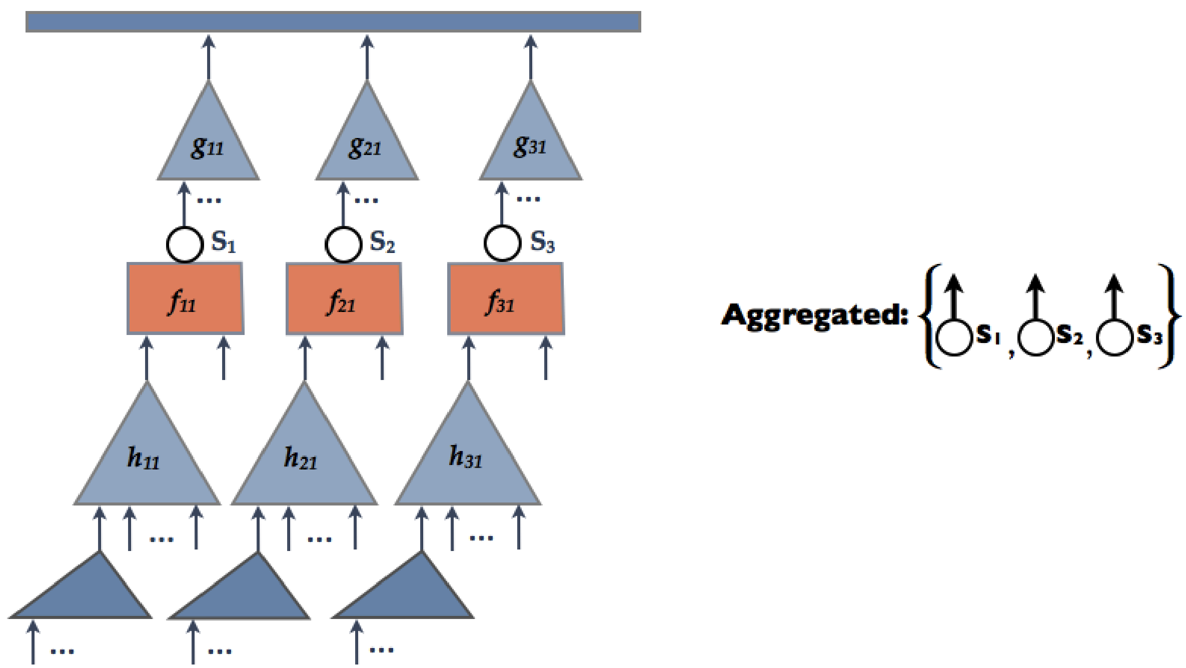
Word Propagation and Module Matching



Word Propagation and Module Generation

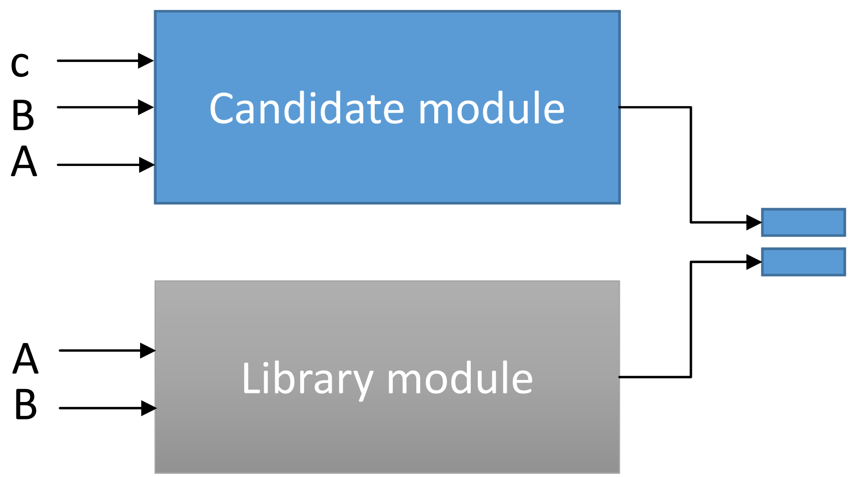
Once multibit structures blocks are found, larger bit slices can be identified by forward and backward traversal of the circuit.

Given an "output" word, we can traverse backwards to closely-related words to find ***candidate modules***



Library Matching

[FMCAD '14]



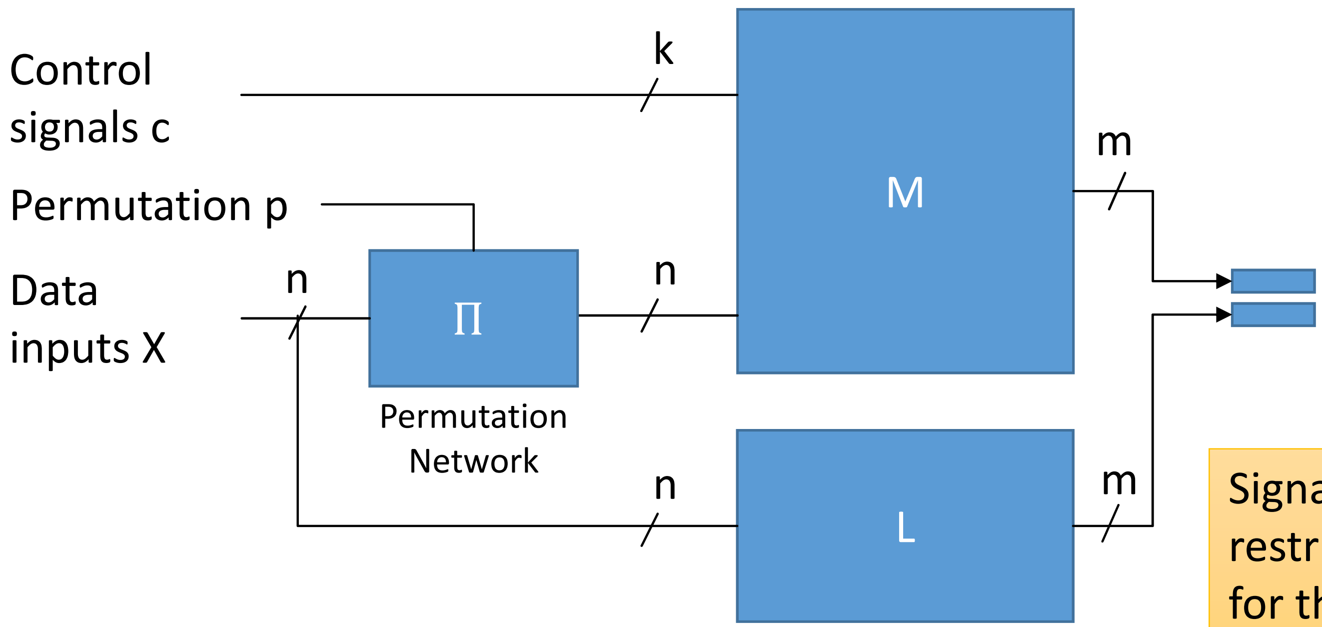
Match candidate modules against a library of common modules such as adders, ALUs, …

Challenges
- Permutation and polarity of inputs
- Setting of control inputs

QBF Formulation:
Does there exist some setting of the control inputs, and some ordering of the inputs such that for all input values, the candidate and the library module produce the same outputs?

Library Matching as QBF

[FMCAD '14]

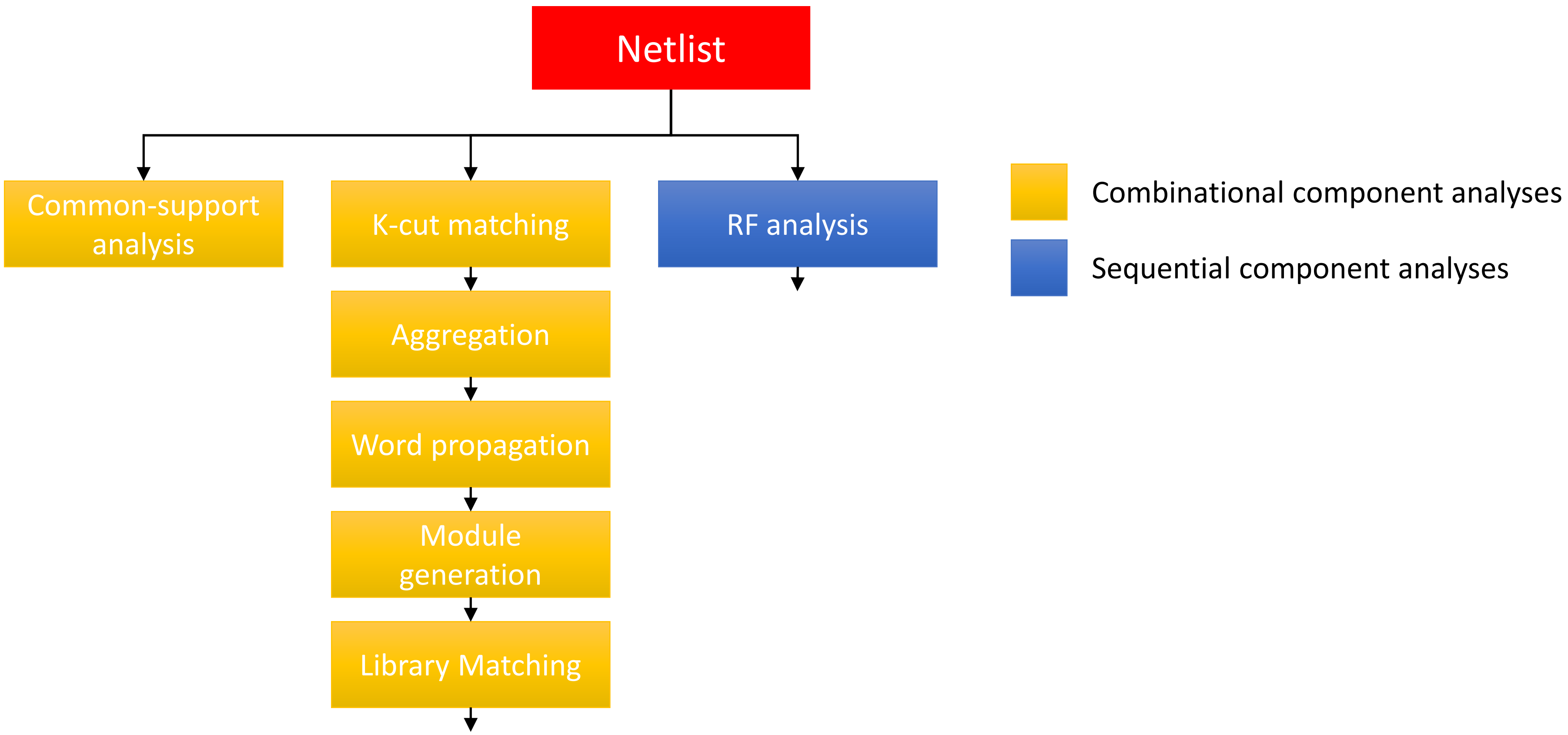


Signatures are used to restrict the search space for the permutations

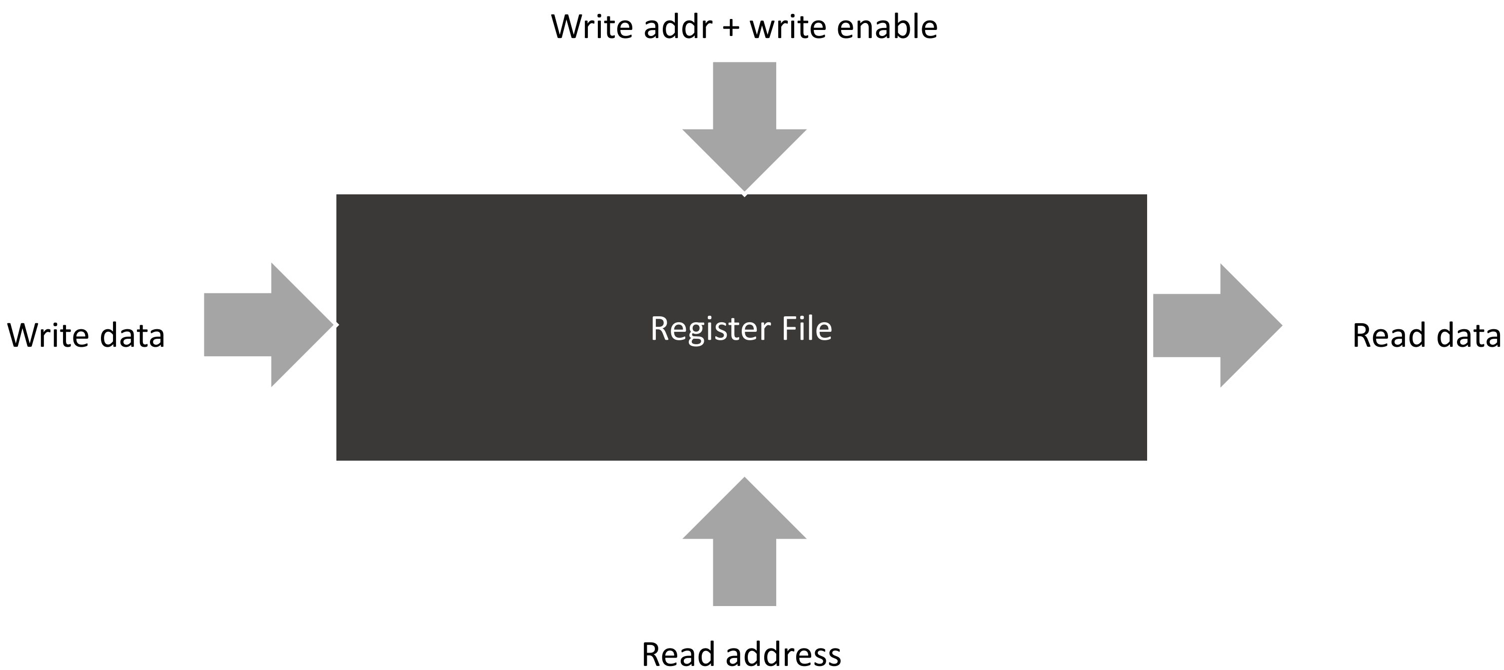
$$\exists c, p\ \forall X\colon\ M(\Pi(p, \mathrm{X}), \mathrm{c}) \equiv \mathrm{L}(\mathrm{X})$$

Mohnke and Malik, Permutation and Phase Independent Boolean Comparison, [Integration '93]

Identifying Register Files



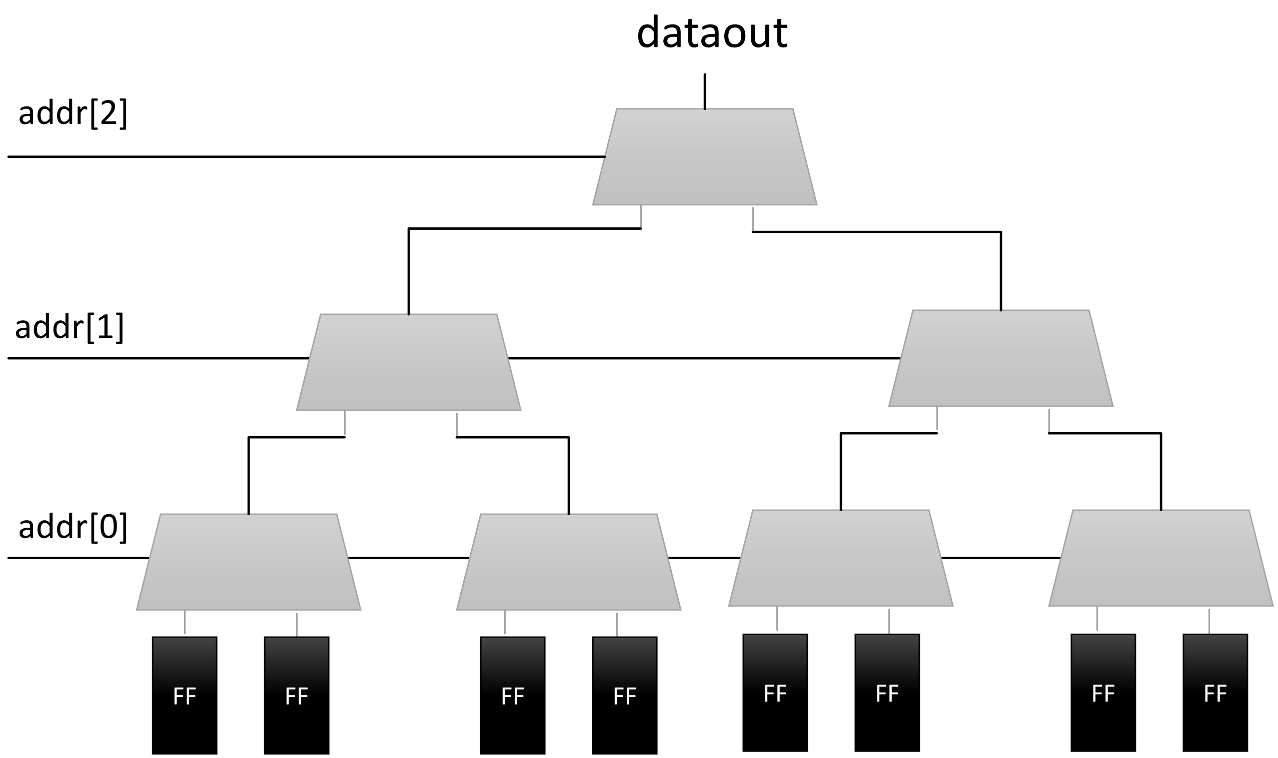
The Structure of a Register File



Register file consists of:

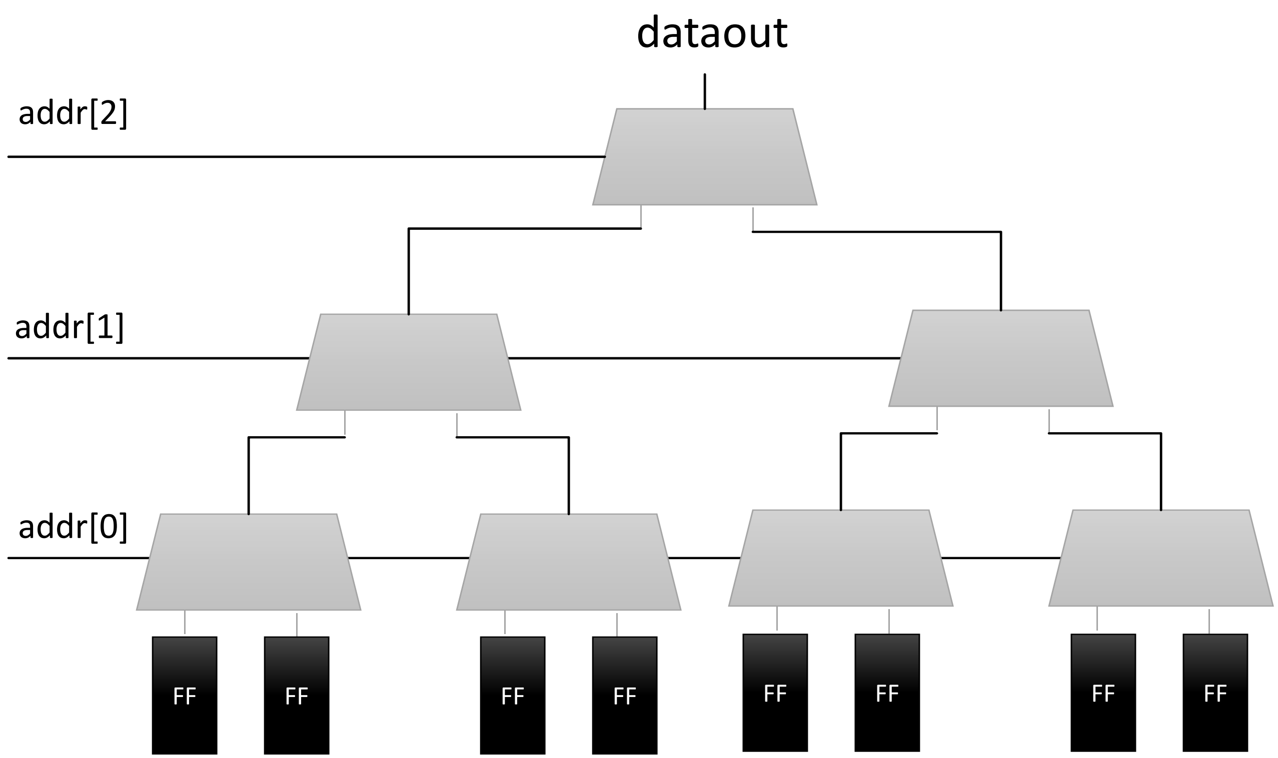
- Flip-flops that store information
- Read logic: takes a read address and outputs stored data
- Write logic: *stores* data in the register file

Identifying Read Logic



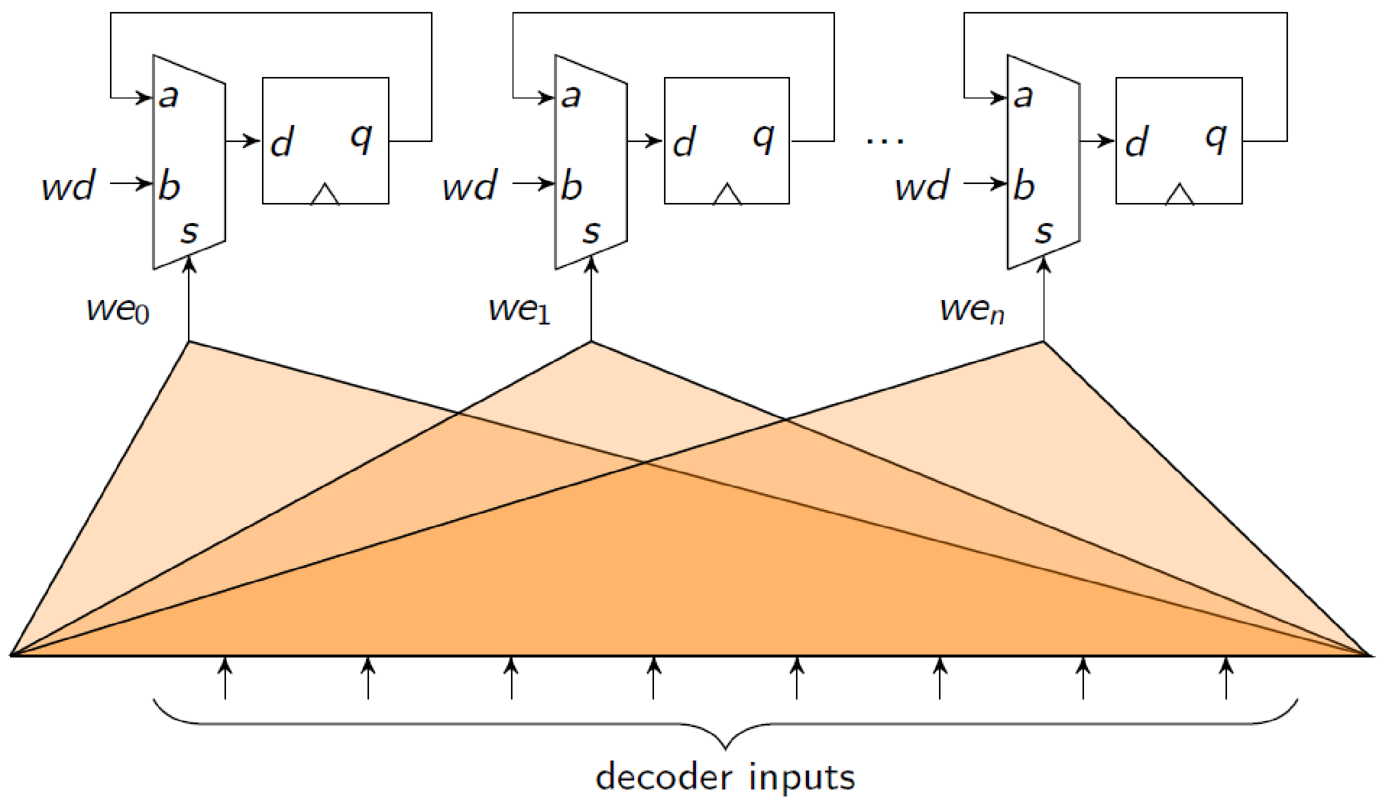
Insight: look for trees of logic where the leaves of the tree are flip-flops

Verifying Identified Read Logic



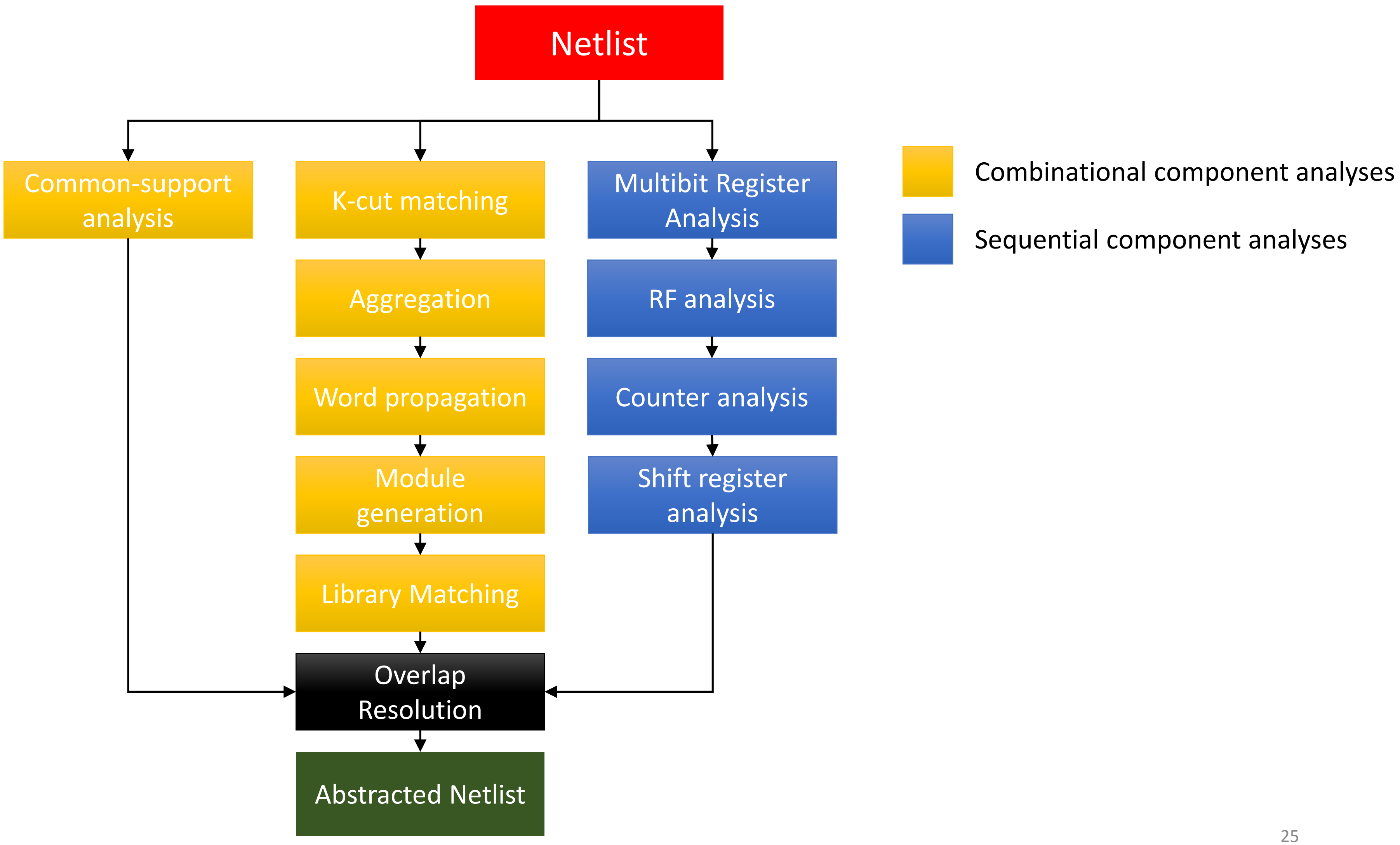
- Verify there exists some address which propagates each flip-flop output to the data output
- This is done using a BDD-based analysis

Identifying Write Logic

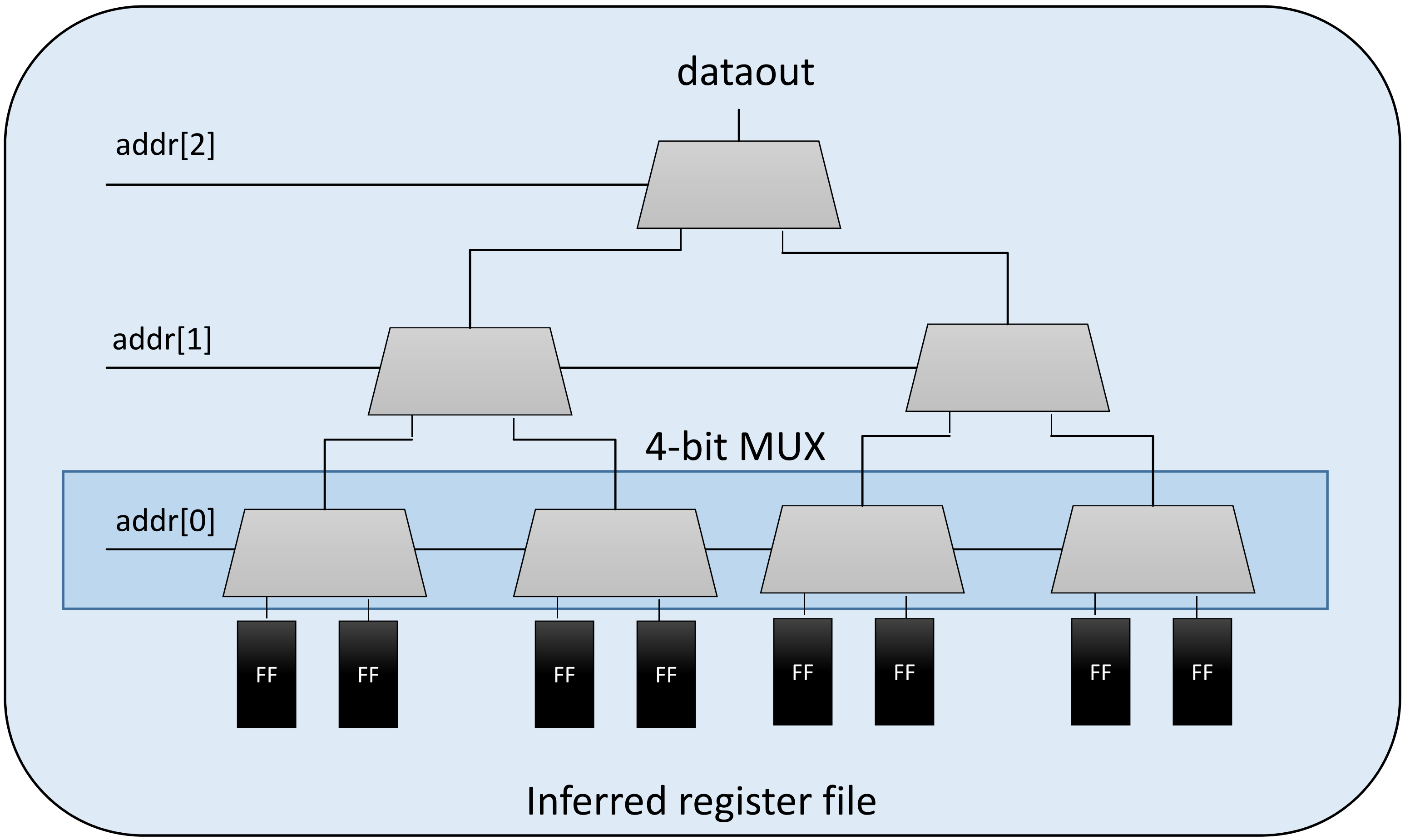


- Muxes select between current value and write data
- Decoders select the location that is being written to
- Easy to find muxes and decoders after we find the flip-flops

Overlap Resolution



Problem: Inferred Modules Overlap



Resolving Overlaps

Formulate an **Integer-Linear Program**

1. Constraints specify that modules must not overlap
2. Objective is one of the following
   - **Maximize** the number of **covered** gates OR
   - **Minimize** the number of modules given a coverage target

Experimental Setup

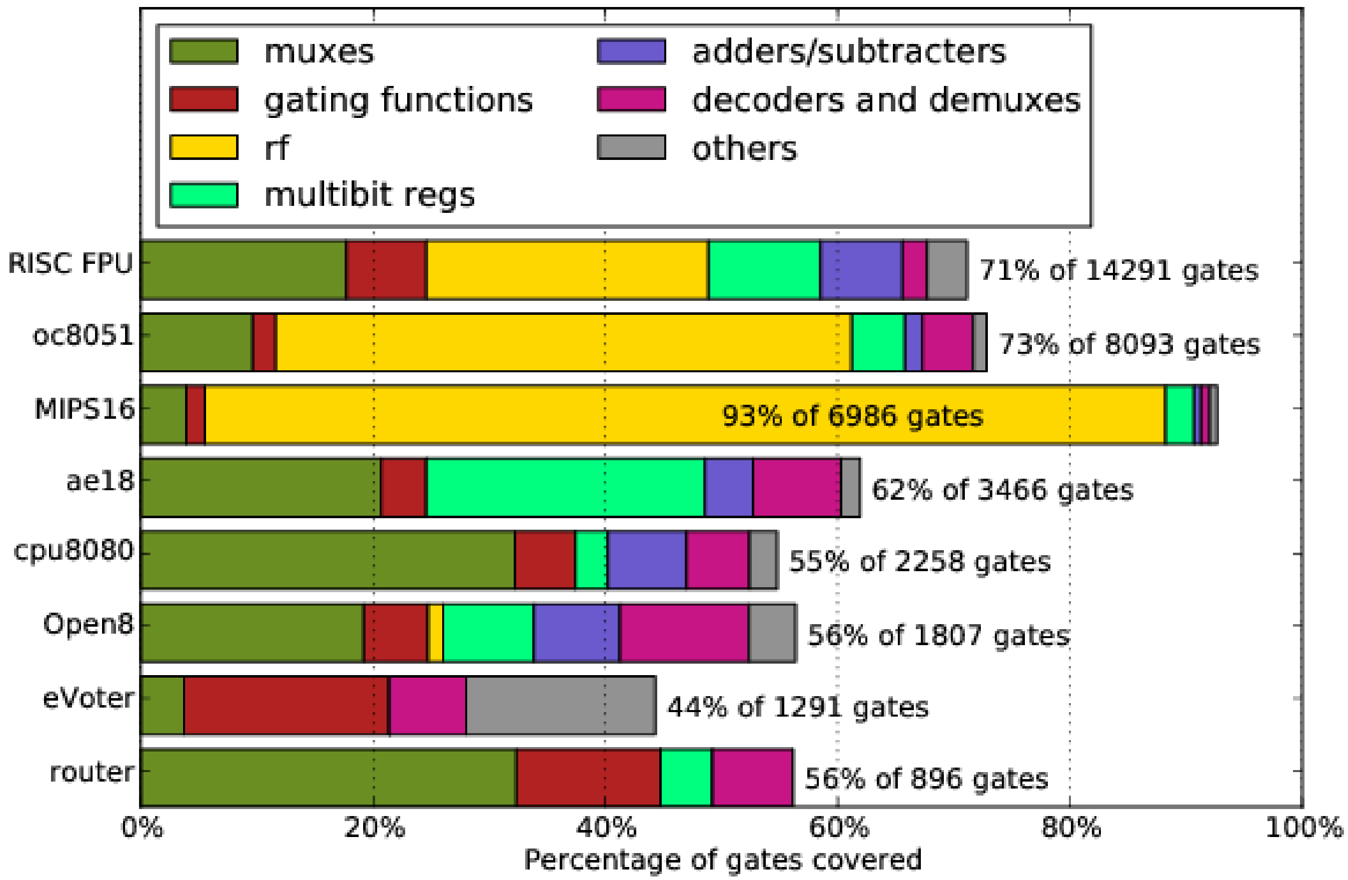
Toolchain

- Implemented in C++
- MiniSAT 2.2
- CUDD 2.4
- CPLEX 12.5

Designs

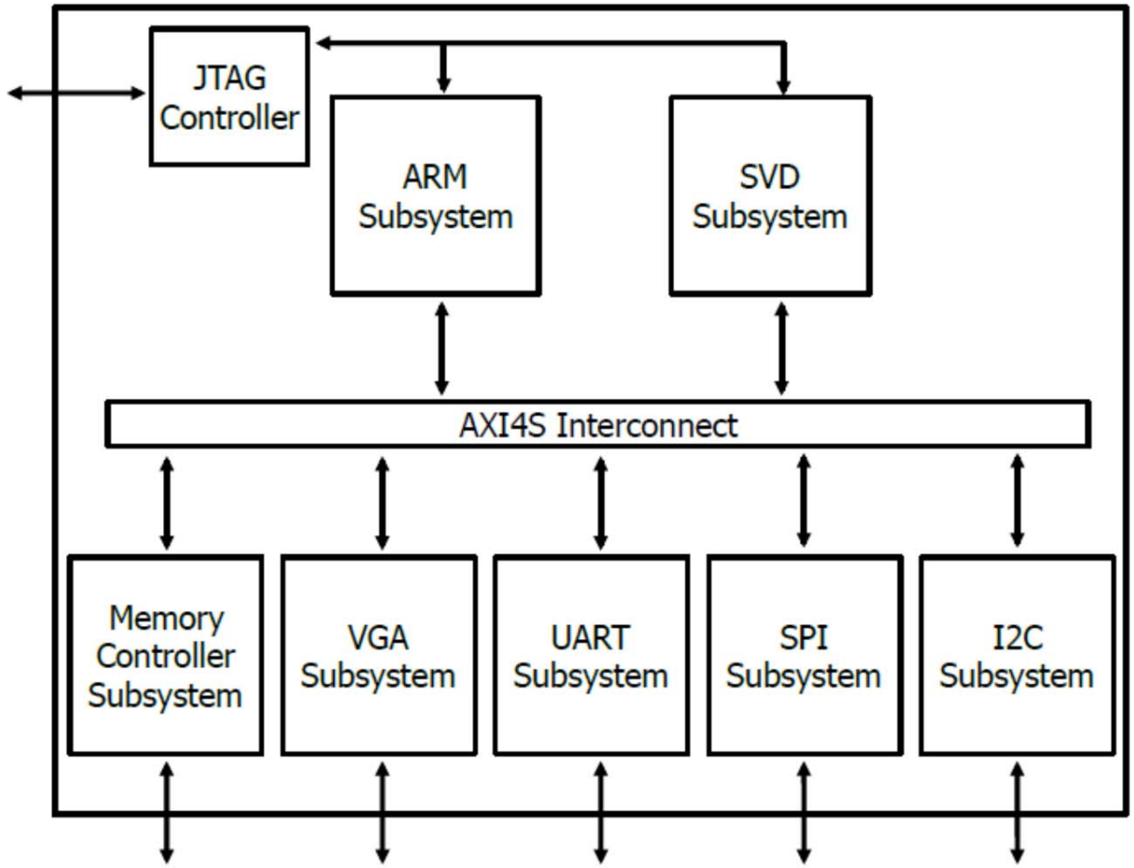
- Many from OpenCores.org
- Size ranges from few hundred to several thousand gates
- ITAG1B: 375k gate test case from DARPA

Summarizing Inference Results (1/2)



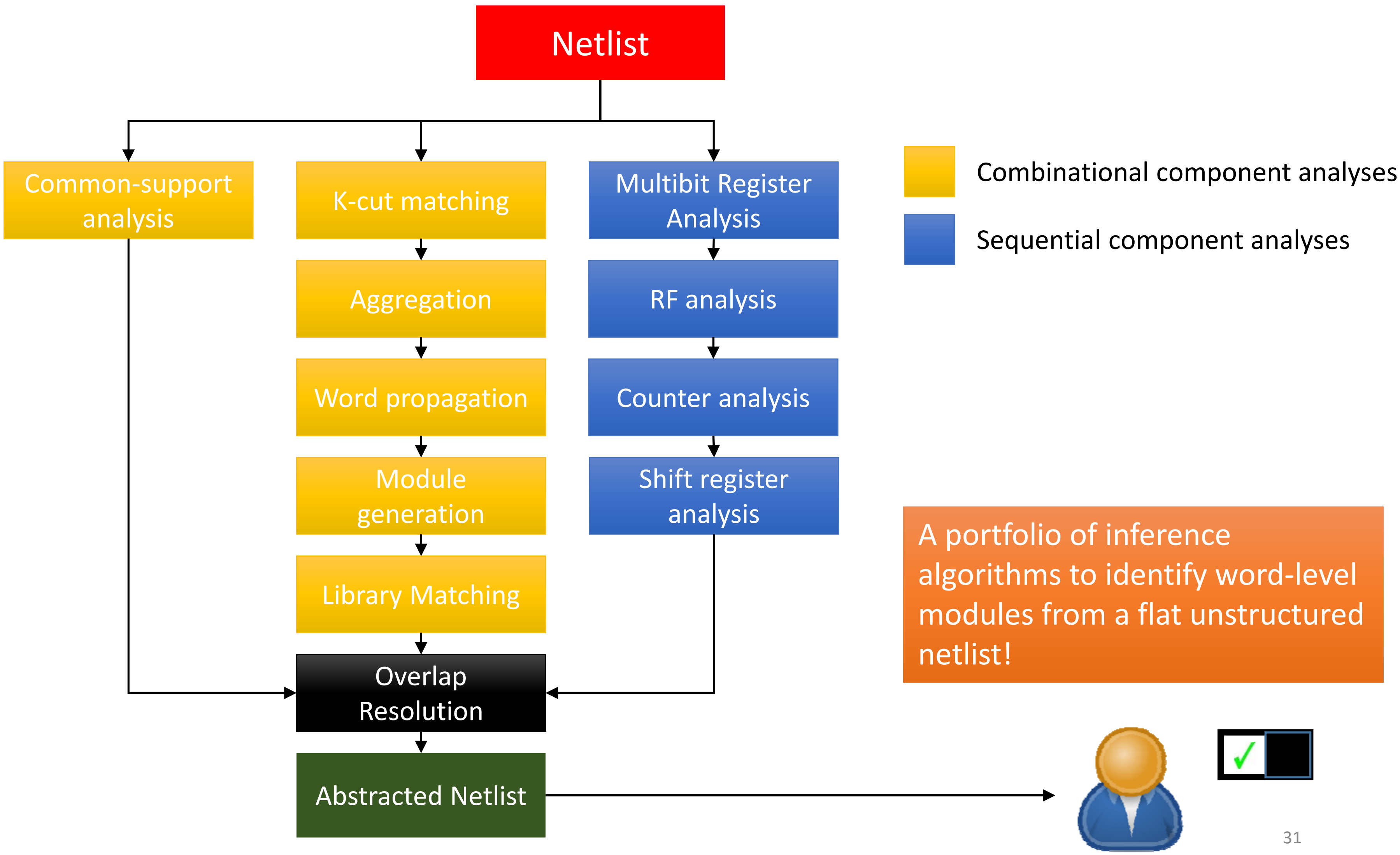
- 45-90% of the gates in these are covered
- Runtime is a maximum of a several minutes

Summarizing Inference Results (2/2)



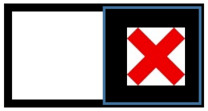
- Covered ~70% of the large test article (375k gates)
- Split the up big design into 7 subcomponents using reset tree; Covered 60-87%
- Entire analysis terminates in an hour

Summarizing the Reverse Engineering Efforts



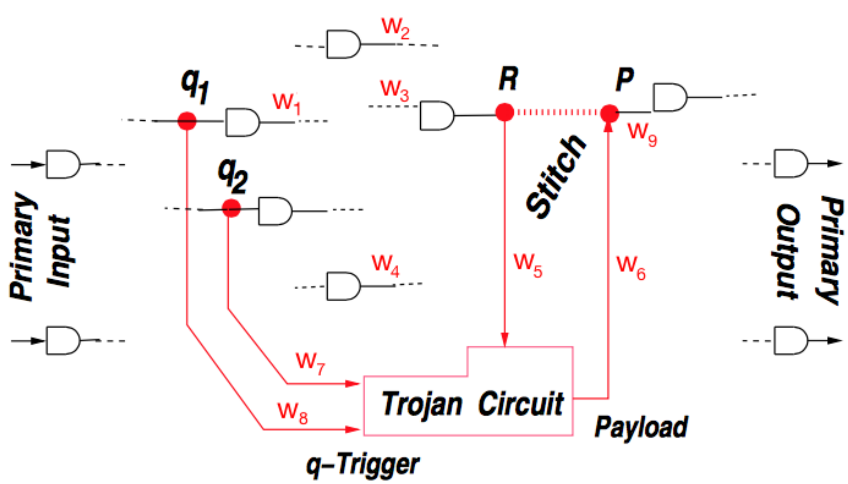
A portfolio of inference algorithms to identify word-level modules from a flat unstructured netlist!

Statistical Analysis of Suspicious Logic



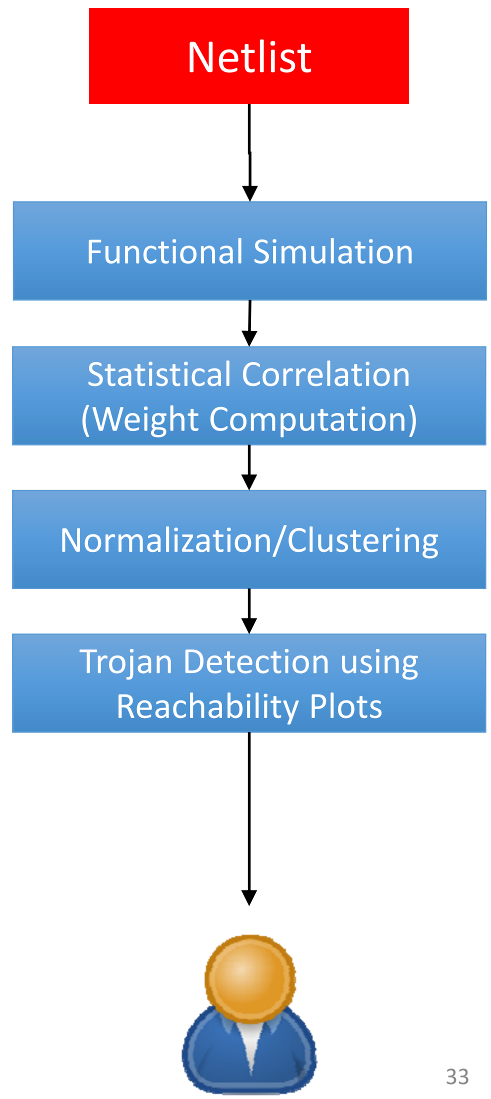
Signal Correlation-Based Clustering: Overview

An information-theoretic approach for Trojan detection

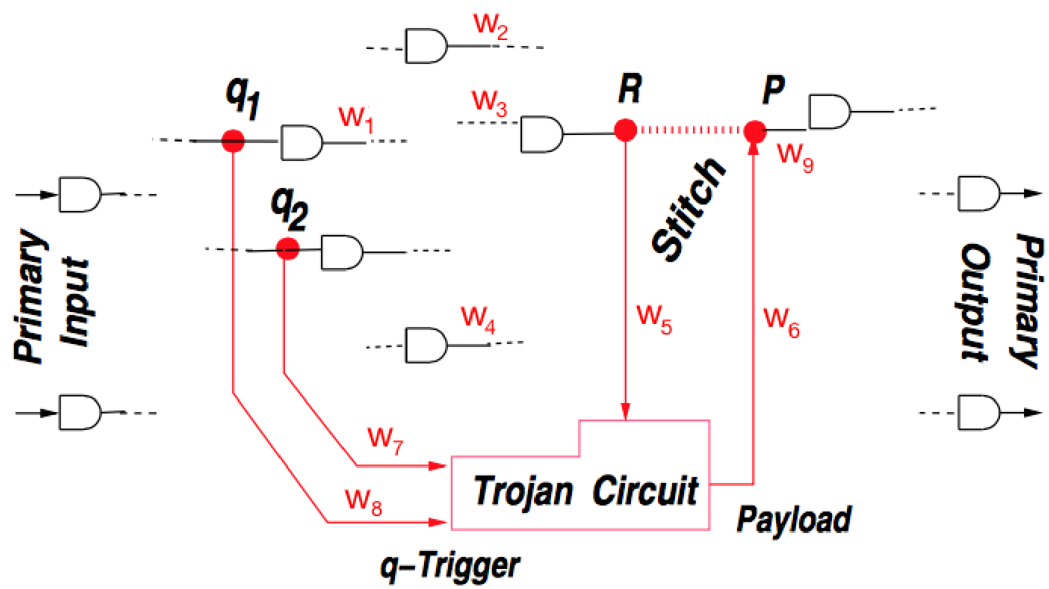


- Estimate **statistical correlation** between signals in a design using simulation data
- Use this estimate in a clustering algorithm to isolate Trojan logic

Cakir and Malik, *"Hardware Trojan Detection for Gate-level ICs Using Signal Correlation Based Clustering,"* DATE 2015 ***[Best Paper Award]***



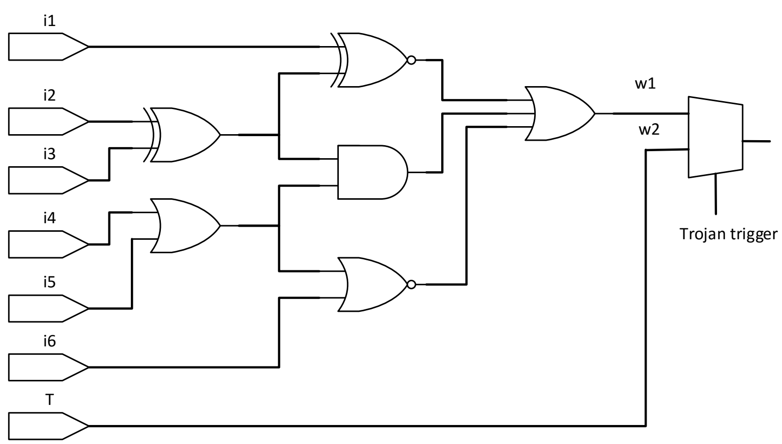
Intuition



Trojan has weak statistical correlation with the rest of the circuit

Functional Simulation-based Statistical Correlation

Example Trojan Circuit

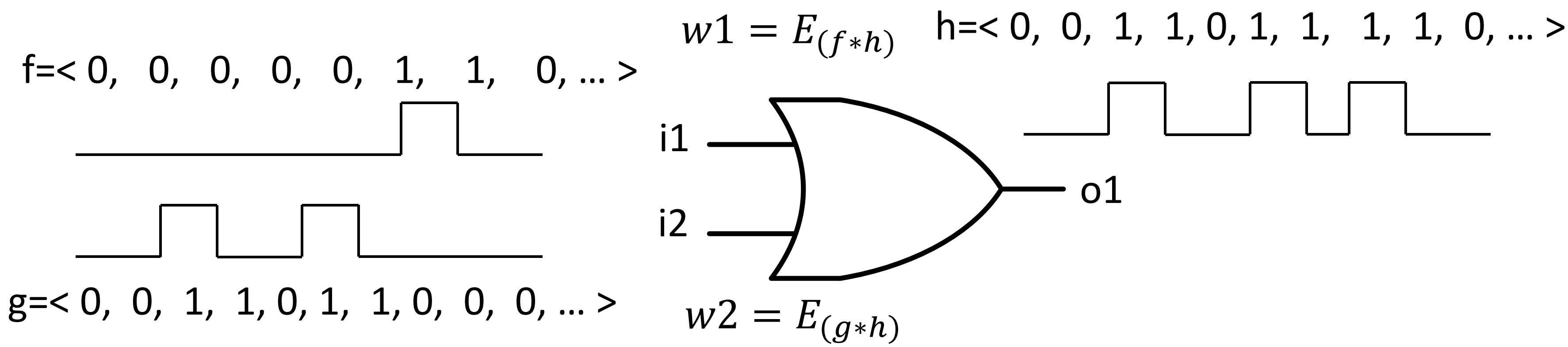


Weight Computation

- Use existing/new testbenches for functional tests
- Generate digital stimuli on different regions of the circuit

Target: **excite the circuit as much as possible** to estimate the statistical correlation between neighboring nodes in the circuit

Functional Simulation-based Statistical Correlation



Simulation waveforms generated with functional tests

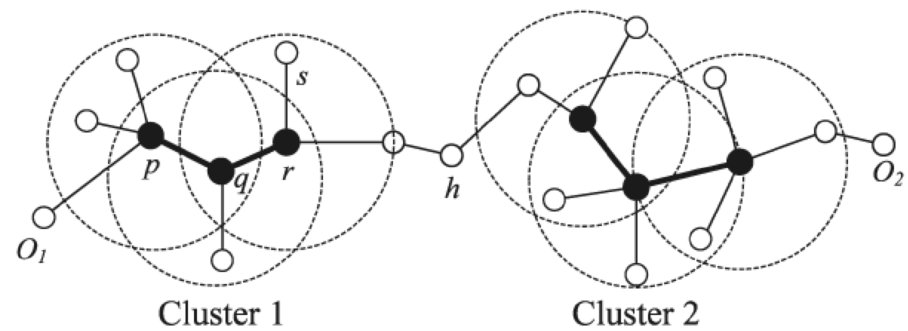
Obtaining new signals from simulation waveforms

Weight of an input/output pair is the energy of the cross-correlation signal

Weight Normalization and Clustering

Weight normalization

- Degree of a node is important to identify hubs and outliers
- Normalize weights based on node degrees
  - obtain **new metric σ**
- Hubs have high degrees
  - Keeps σ across a cluster small

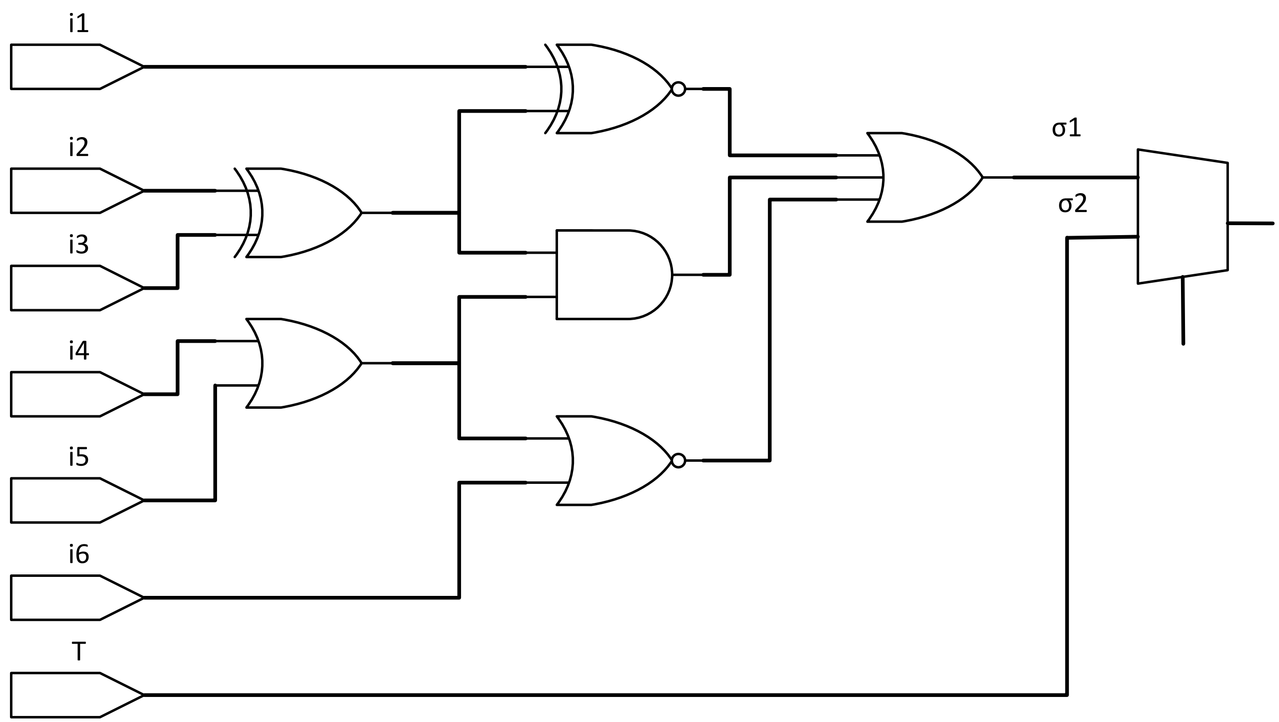


Two structure-connected clusters, with one hub and two outliers

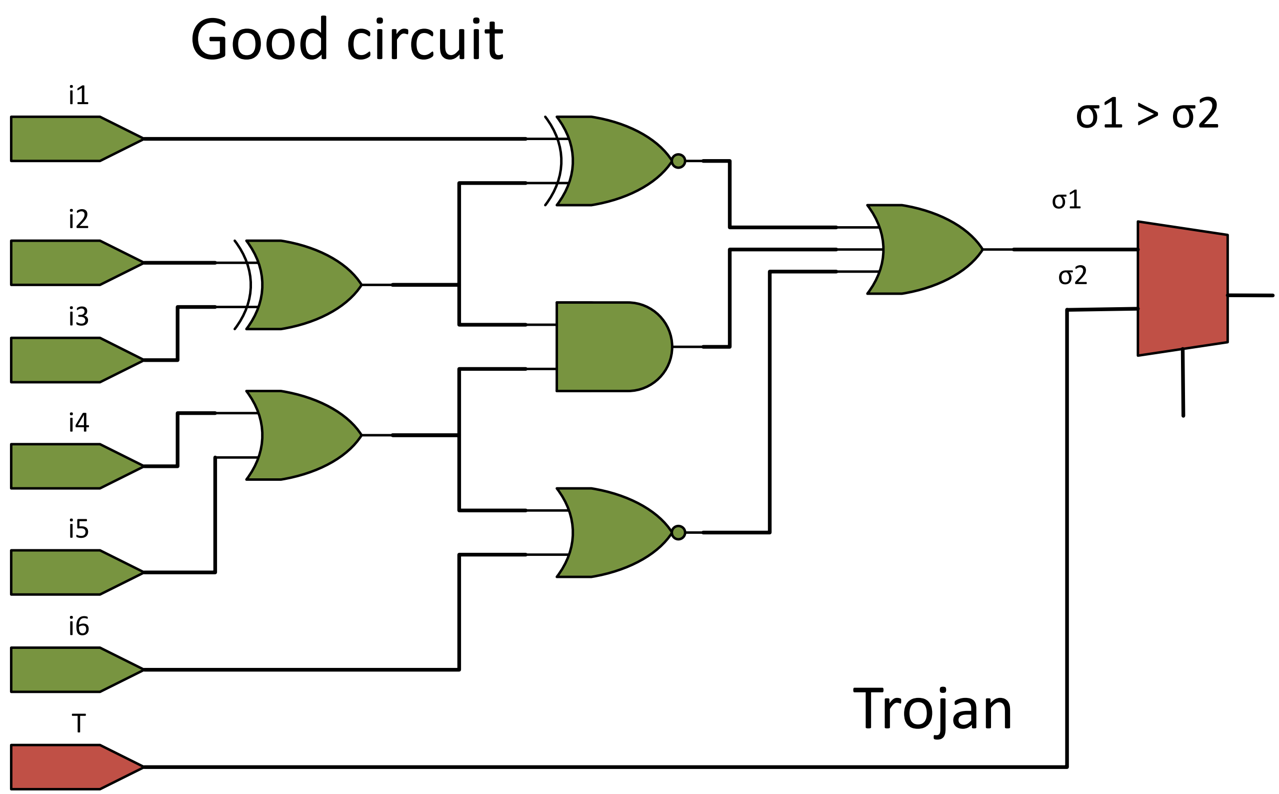
[Jianbin Huang et al., IEEE Transactions on Knowledge and Data Engineering, Aug. 2013]

Weight Normalization and Clustering

Example Trojan



Weight Normalization and Clustering

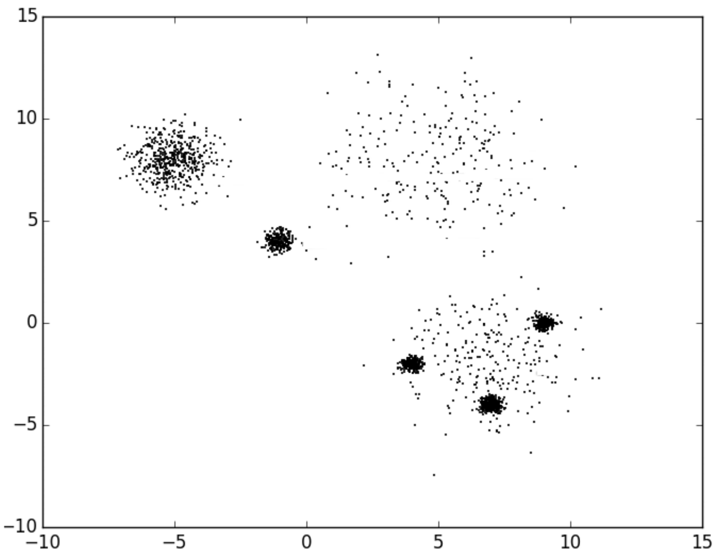


How does clustering help detect Trojans?

- Use OPTICS algorithm in practice, used in learning

Clustering with Reachability Plots

2D Data Set



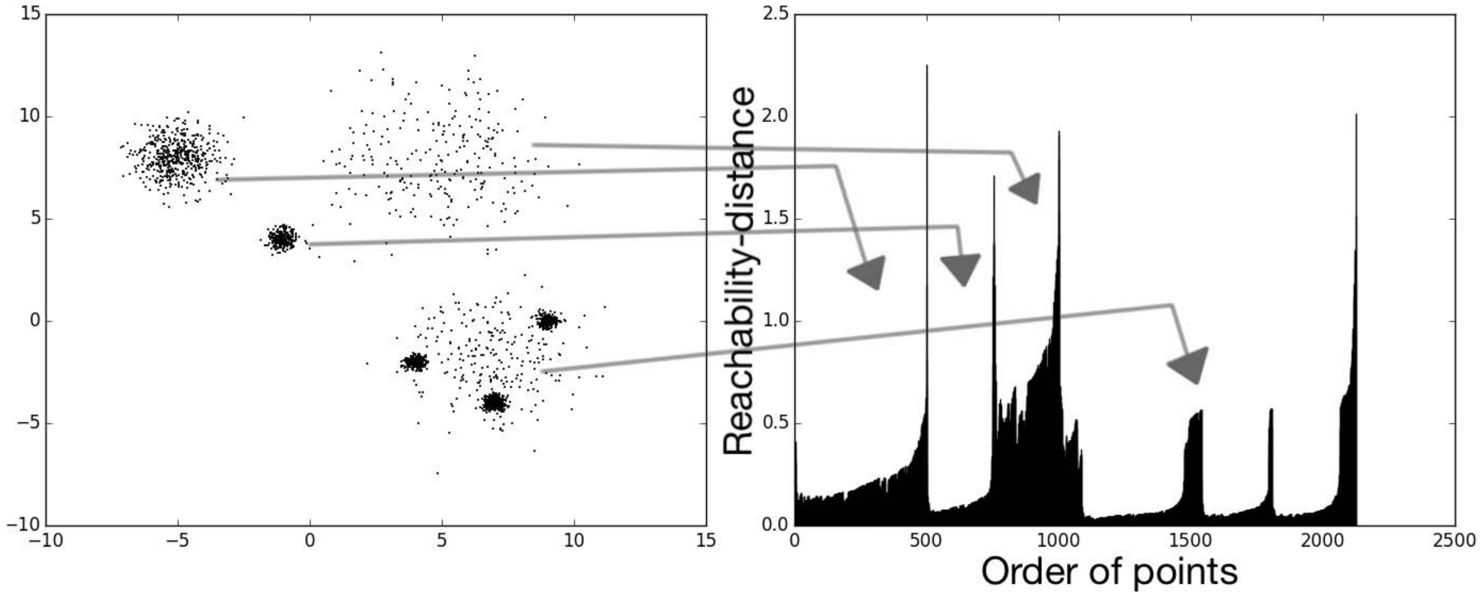
Walk on dataset: An augmented order of dataset to reflect the clustering structure

Example data set:

- Hierarchical clusters of different sizes, densities and shapes

Clustering with Reachability Plots

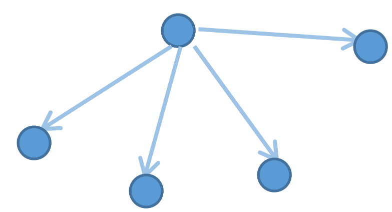
2D Data Set | **Reachability Plot**



Walk on dataset: An augmented order of dataset to reflect the clustering structure

Reachability distance: measure of proximity to dense regions

- Starting point arbitrary
- Order points in increasing distance from current point

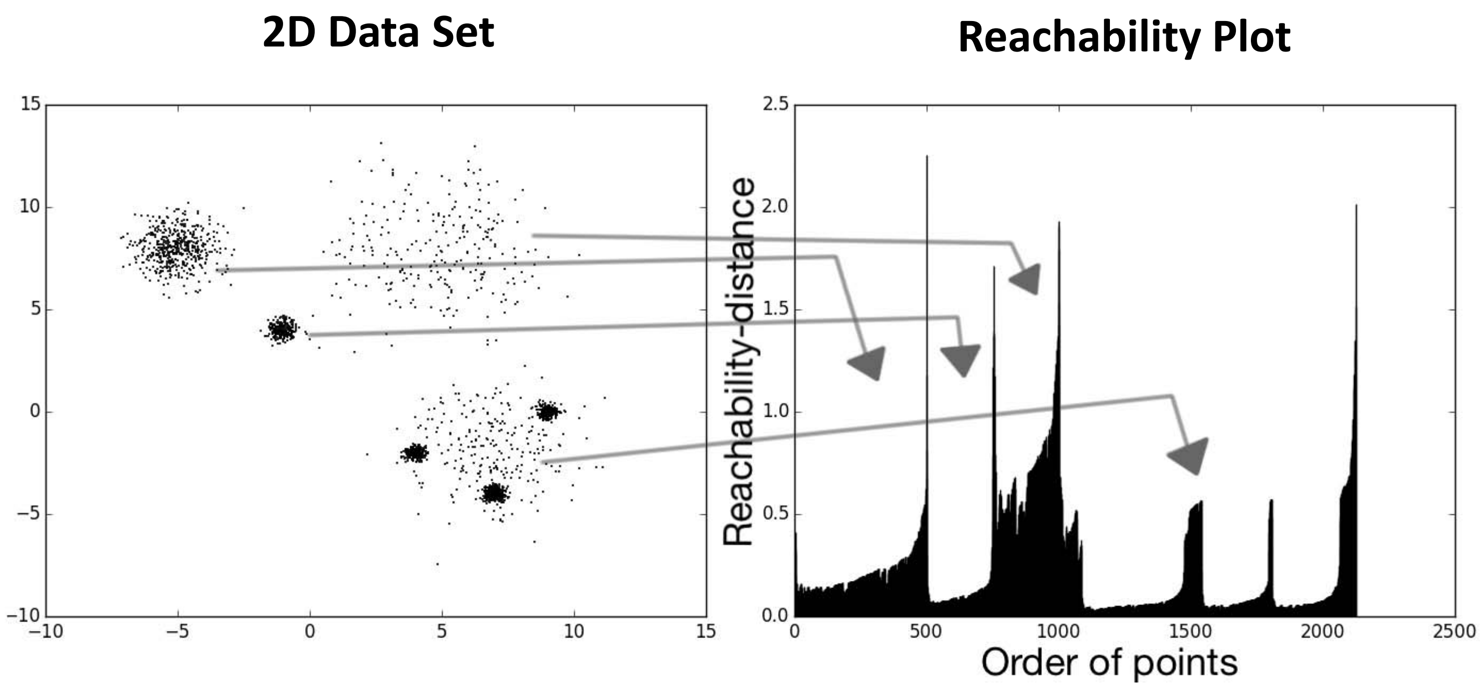


Our Application:

Distance based on 1/σ

- High correlation, smaller distance
- Across hub, larger distance

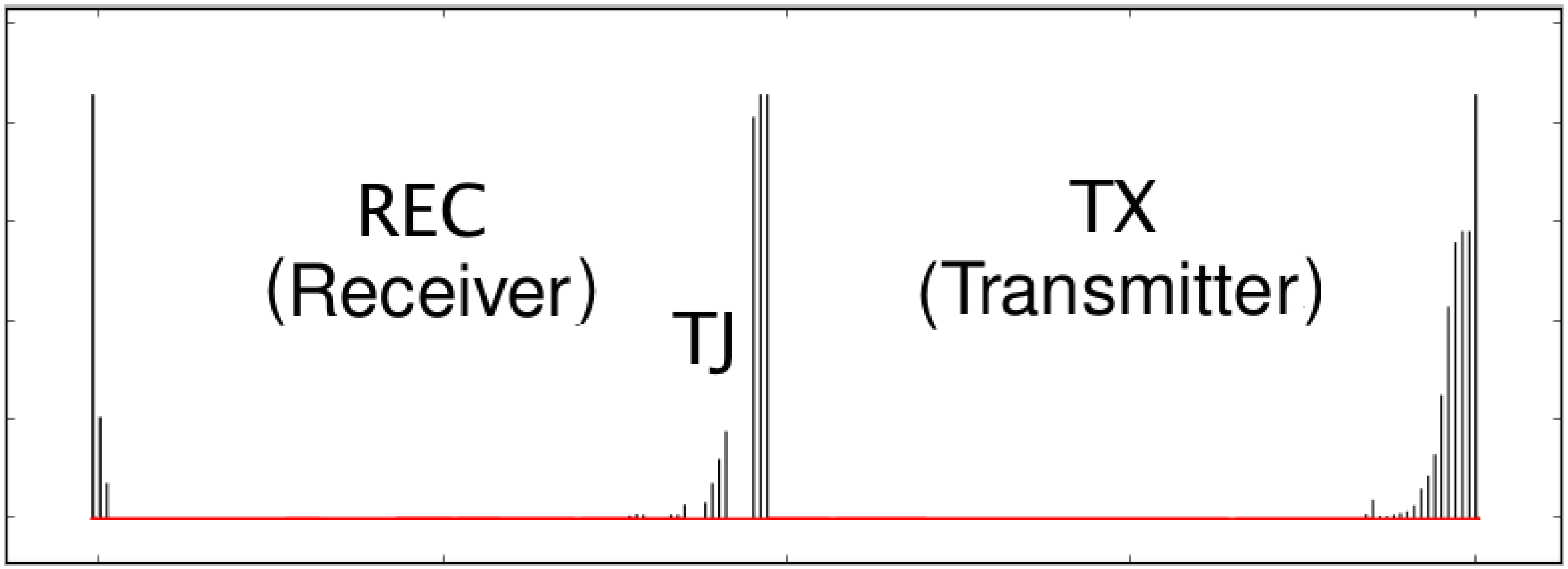
Clustering with Reachability Plots



How useful is this for Trojan detection?

Trojan Detection based on Reachability Plots

<u>RS232-800: UART core</u>
Trojan: Comparator in receiver circuit. Manipulates output signal.

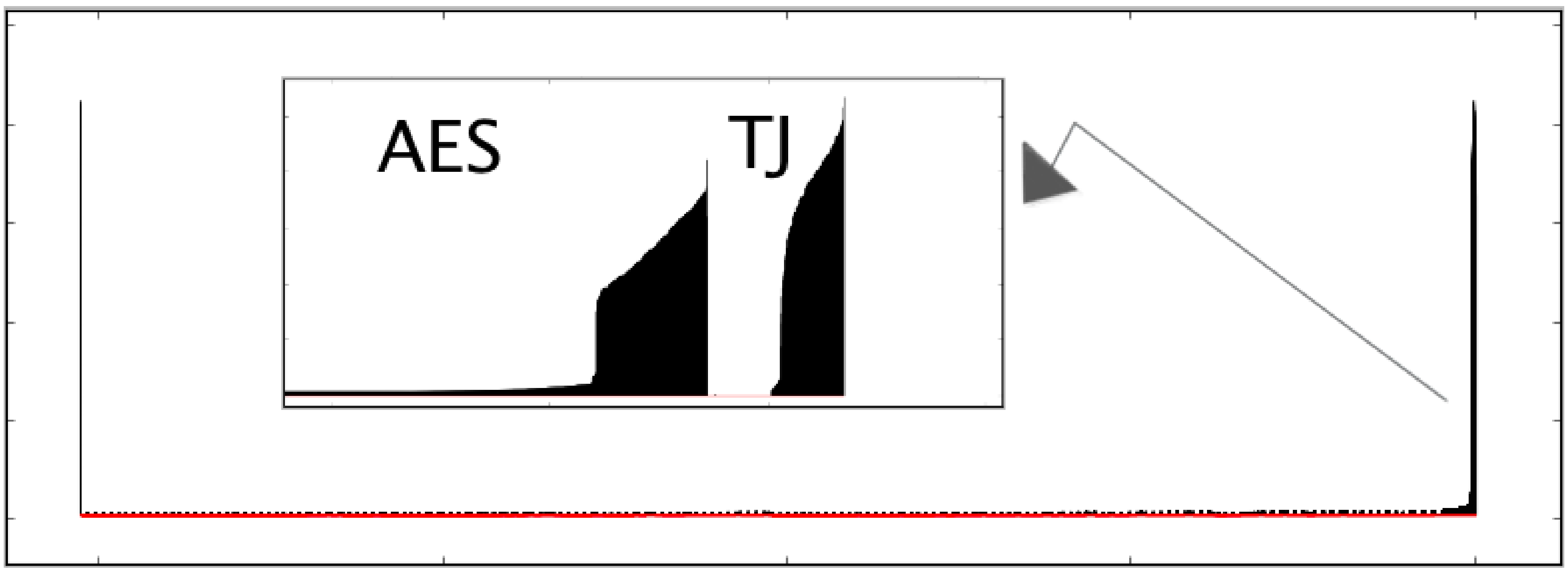


Trojan (TJ) logic distinguished from TX and REC

Trojan Detection based on Reachability Plots

AES-1800: Encryption circuit

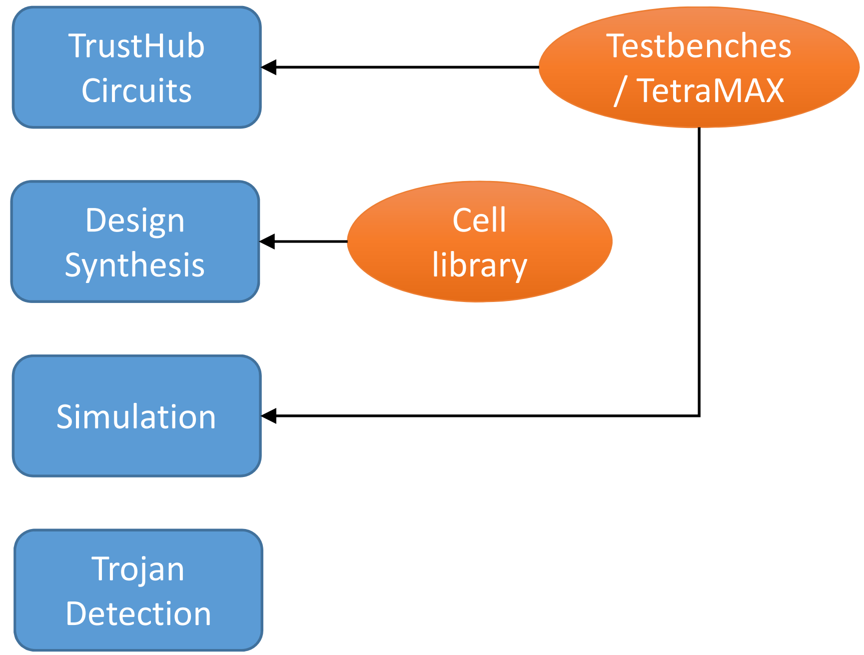
Trojan: Drains the battery after observing a predefined input plaintext.



Trojan (TJ) logic appearing as a separate cluster

Evaluation Methodology

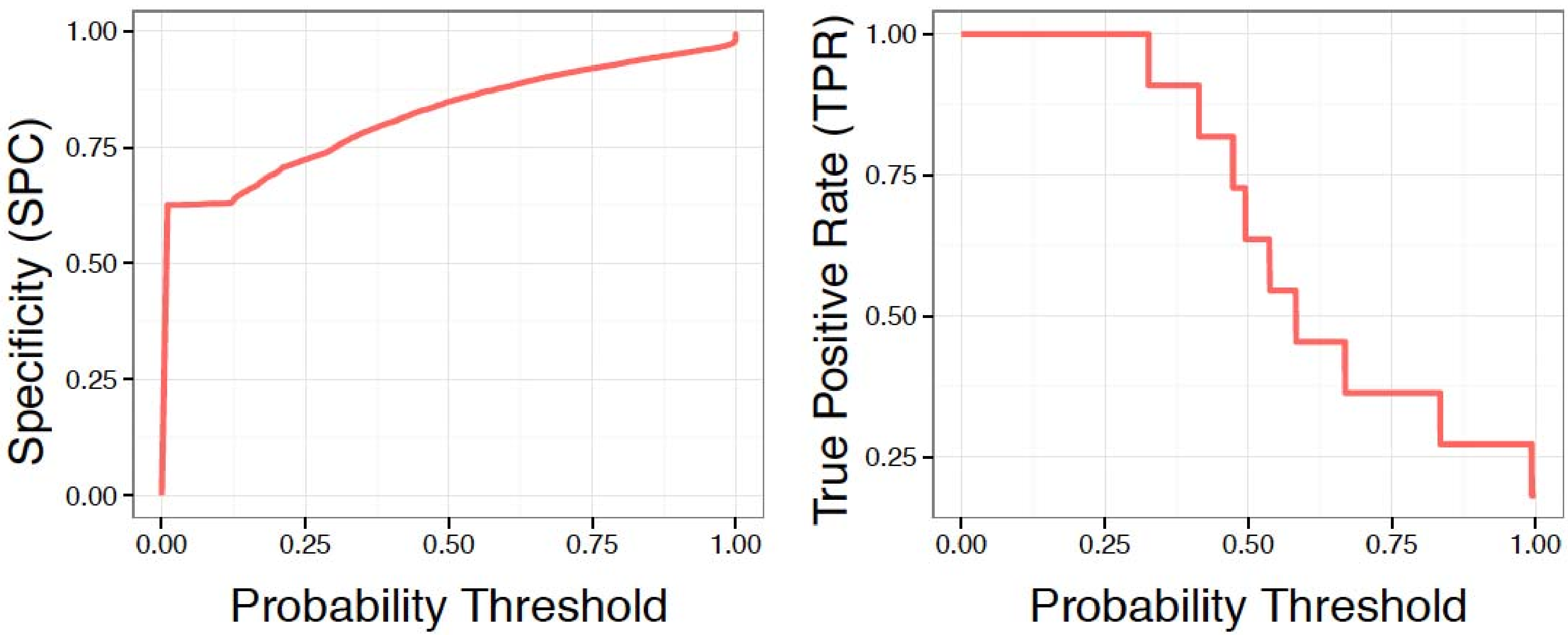
- Eight TrustHub groups of Verilog circuits
  - Synthesized using Synopsys Design Compiler
- IBM/ARM cell library
- Synopsys TetraMAX ATPG tool
  - Used if testbenches not available



Trusthub benchmarks [http://www.trust-hub.org/resources/benchmarks]

Sensitivity and Specificity Analysis

s35932-200: ISCAS’89 benchmark



Specificity: 1 - False positive ratio, TPR: True positive ratio (Sensitivity), Probability Threshold: Confidence-level parameter

Sensitivity and Specificity Analysis

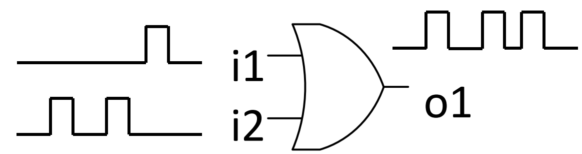
Design Information		Trojan Detection	
Name	**Gate/Latch**	**SPC (%)**	**TPR (%)**
s15850-100	3478	99	61
s35932-200	8107	99	27
s38417-100	8422	99	100
s38584-200	9548	99	99
AES-1800	164800	98	92
wb-conmax-200	20224	96	28
PIC16F84-100	1616	96	75
RS232-800	205	94	80

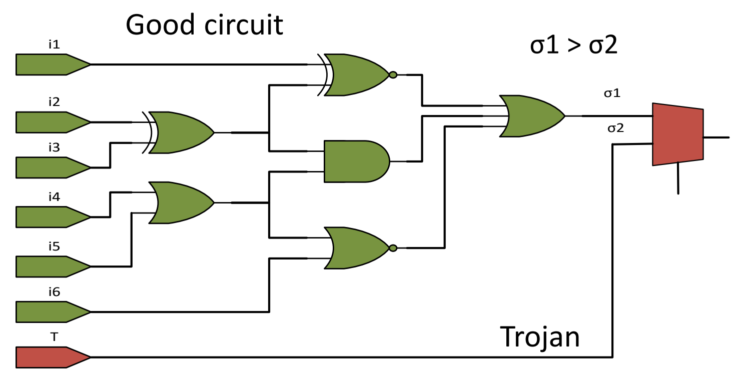
At least a quarter of the nodes of *each Trojan* is identified

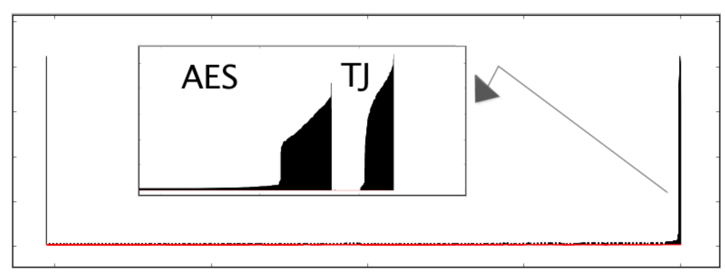
Specificity: 1 - False positive ratio, TPR: True positive ratio,

Summary: Signal Correlation-Based Clustering

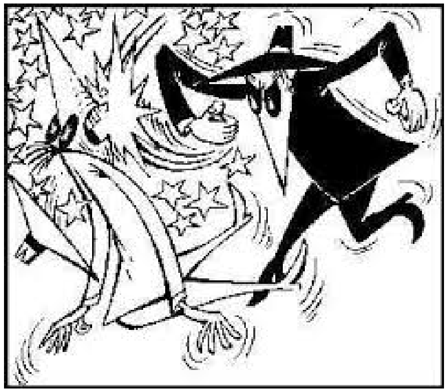
- Simulation-based clustering technique to detect hardware Trojans in gate-level circuits
- Methodology to find weakly-correlated nodes or functionally isolated sections in the netlist
- Identify Trojan-related nodes with low false positive rates
- Key observations
  - Do not attempt to find all Trojan logic but flag a small subset of gates
  - Extensive test sets lead to higher coverage and better statistics $\Rightarrow$ Better results







Conclusions



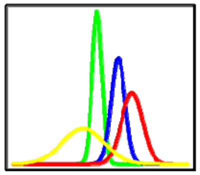
Logical Analysis

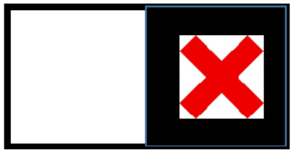
$\vdash$ $\exists$ $\forall$ $\vee$ $\neg$ $\wedge$

Whitelist

- Portfolio of matching algorithms for reverse engineering
- Went much further than we expected
- Simulation data-based clustering very powerful
- Applications beyond Trojan detection?

Statistical Analysis





Blacklist