Reducing Power with Activity Trigger Analysis

Jan Láník*+, Julien Legriel*, Erwan Piriou#, Emmanuel Viaud*, Fahim Rahim*, Oded Maler+, Solaiman Rahim*

*ATRENTA

+VERIMAG – University of Grenoble,

#CEA-LIST

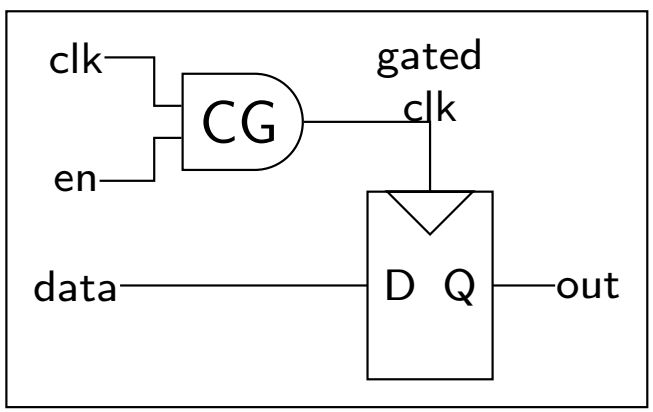
23rd September 2015

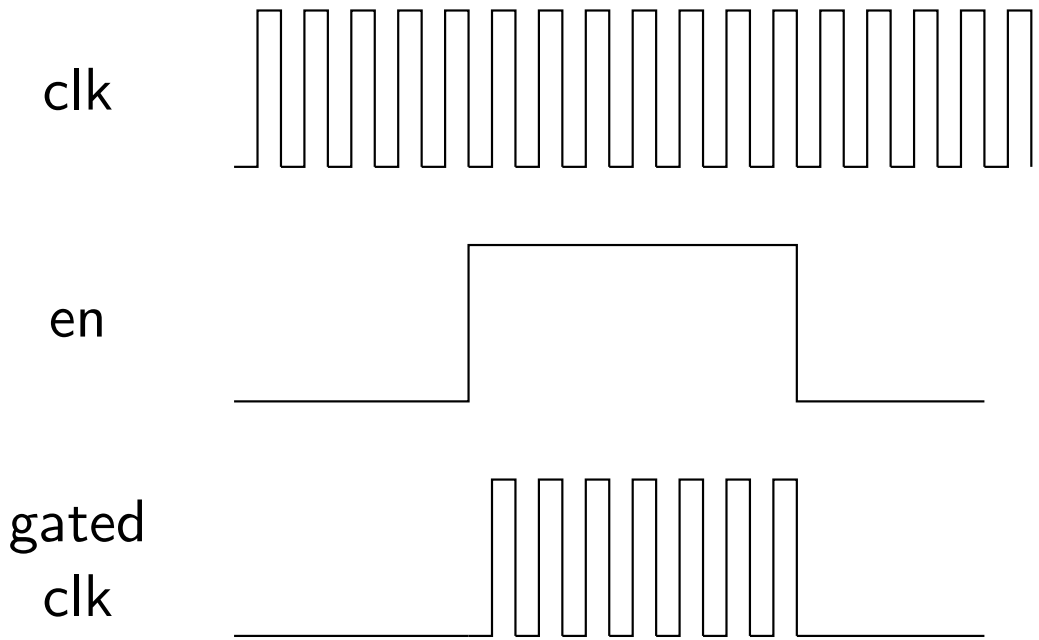
Motivation


Nokia 3310
Smartphone
Charge once a month
It will never break
Charge everyday
It will break asap
www.rascojet.com/blog

Clock gating

Disabling registers when not needed by 'gating the clock' to save power





Problem: How to compute the enabling condition?

Clock gating conditions - granularity

Global

- Design decision.
- On the high level - whole functional blocks.
- Handcrafted enable conditions.
- Efficient, easy to implement, high in the clock tree.

Local

- Small register groups deep in the designs.
- Complex, not intuitive enable conditions.
- Need tools to find the conditions.
- Expensive, but can be efficient.

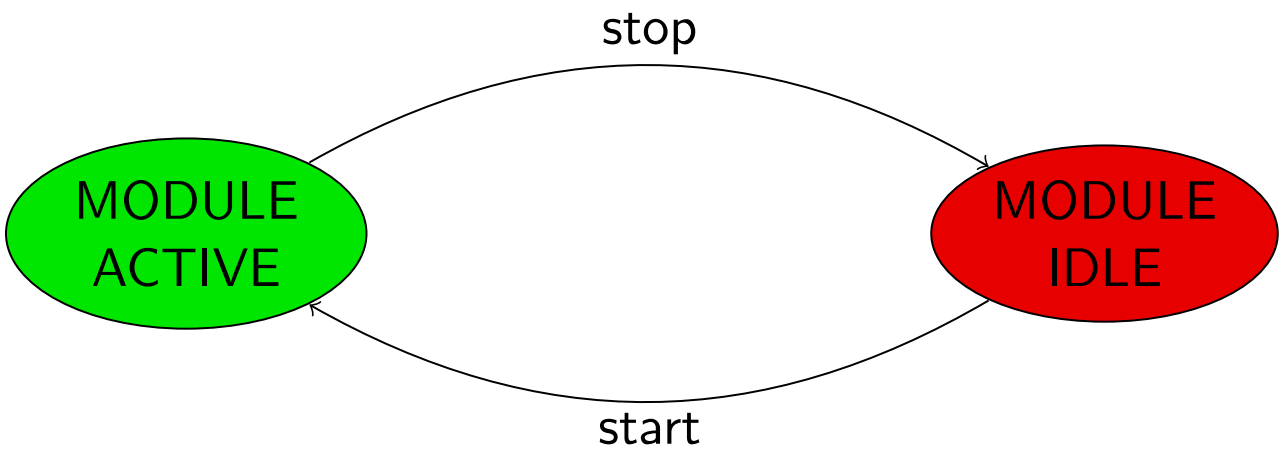
Clock gating conditions - granularity

Intermediate

- Missing link.
- Medium sized blocks.
- Understandable, but not necessarily obvious.
- Human designer should be able to find them if he did a time consuming detailed analysis. Tools in demand.

Activity Triggers

Events related to a change of activity status of a design block.



UART example

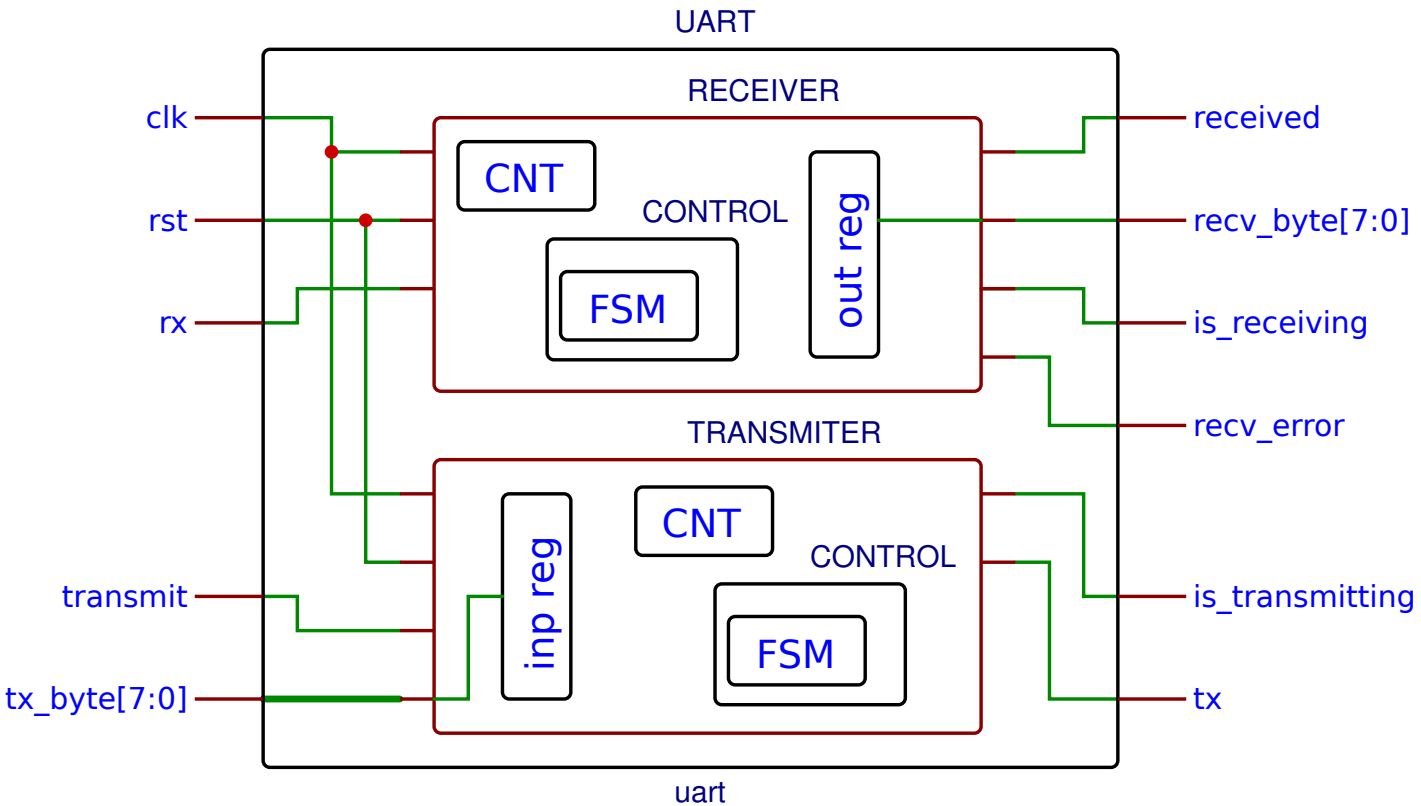


Figure: Schema of a simple UART design

UART transmission

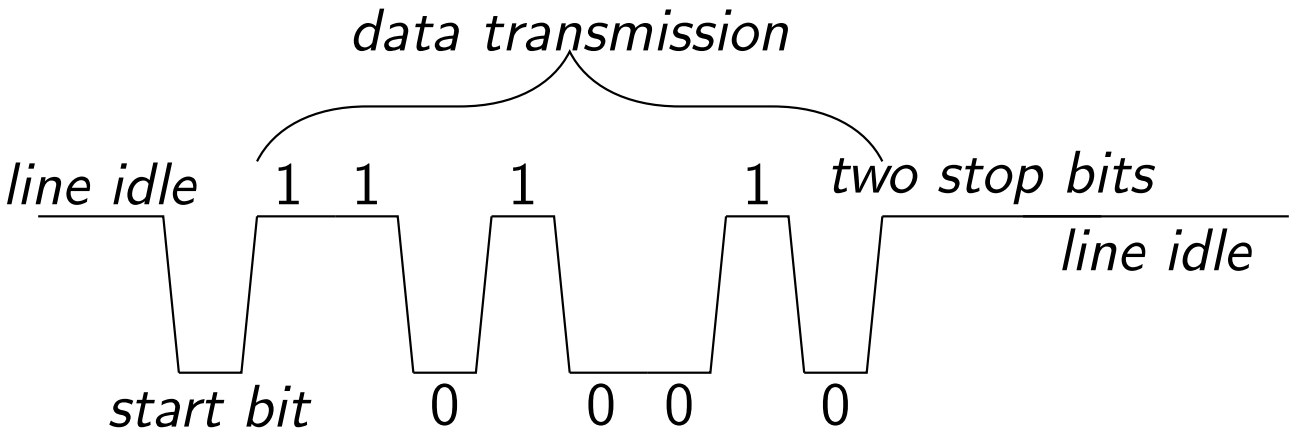
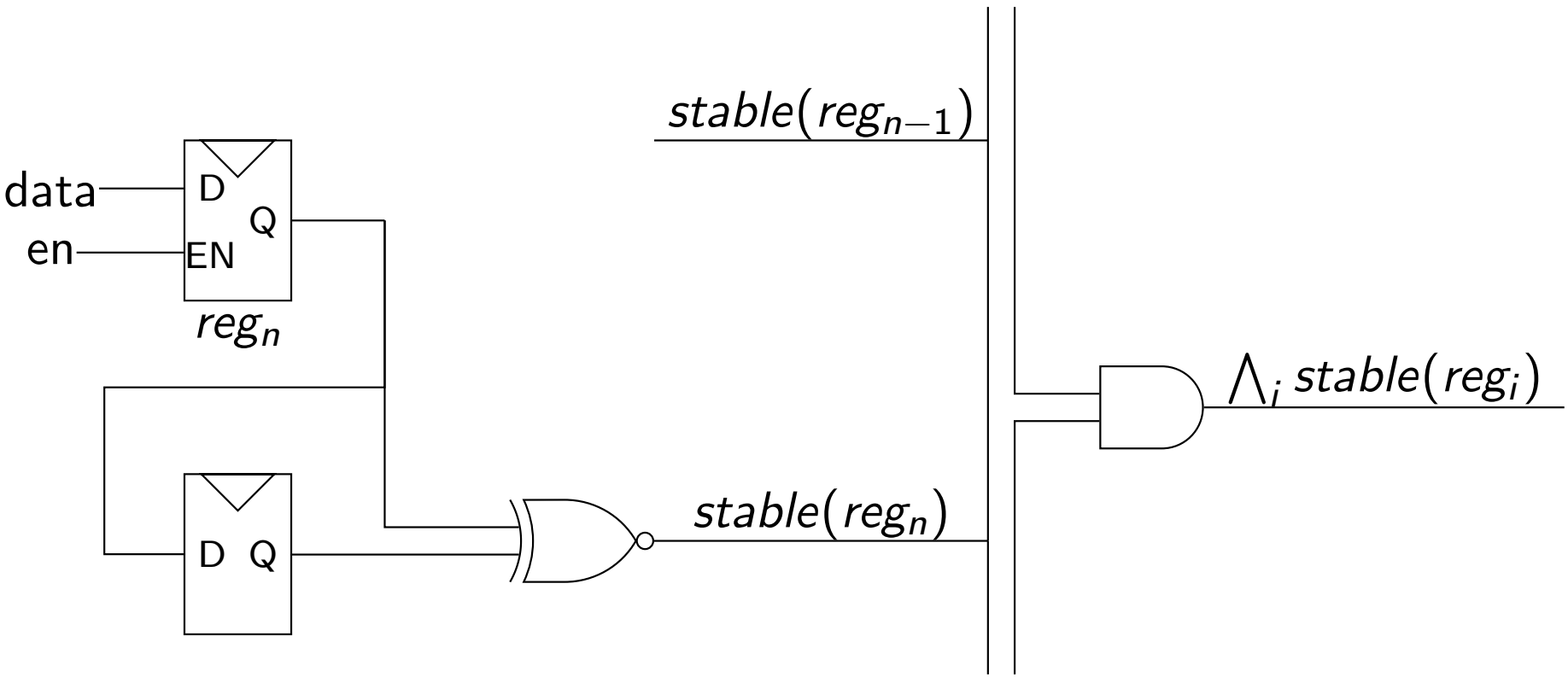


Figure: Serial line transmission of the character 'K' in the ACSCII encoding

Stability modeling



Monitor automaton

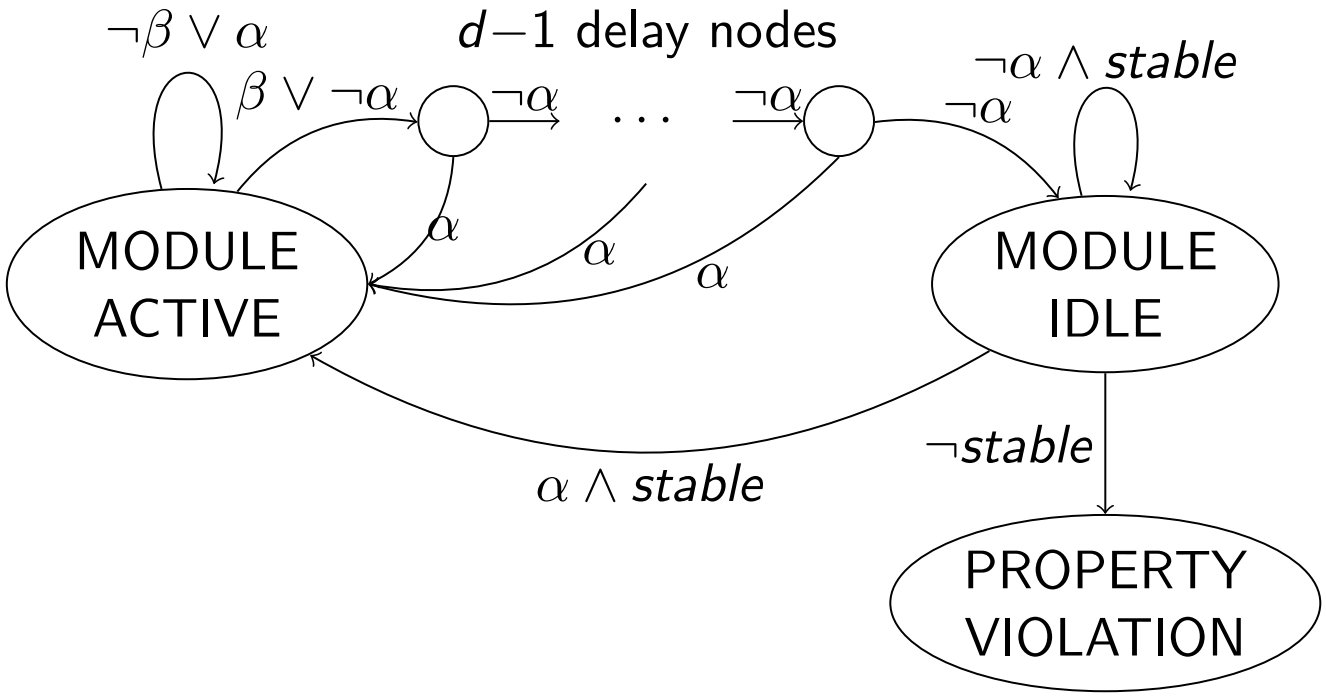


Figure: An automaton for checking the validity of activity triggers

Formal verification flow

1. Original RTL $\Rightarrow$ circuit representation
2. Stability modeling and monitor automaton added to the circuit
3. Verification using reachability engines from ABC (BMC + PDR)

Constraint support

- Constraining behavior is often crucial for thae success of a formal proof
- Typical cases: configuration registers or input following specific pattern
- We support behavioral constraints expressed as System Verilog Assertions (SVA)

Statistical detection

Correlation analysis performed on vcd or fsdb traces generated from simulation
For detection we consider only events that are bit/bus transitions.
E.g. a signal x going from 0 to 1 or a 4-bit bus Y going from $4'b0001$ to $4'b0010$

1. design decomposition
2. idle periods detection
3. potential events filtering (based on size and sequential distance)
4. ranking potential events (coverage and ran measures)

Potential start and stop signals location

- *Stop events* $\cdots$ in a short window before the beginning of stable periods
- *Start events* $\cdots$ in a short window before the end of stable periods

Figure: Idle periods in a simulation

Ranking of events

- *Coverage* $\cdots$ the ratio of idle periods that are correlated with the event
- *Noise* $\cdots$ the ratio of events that are 'out of place'

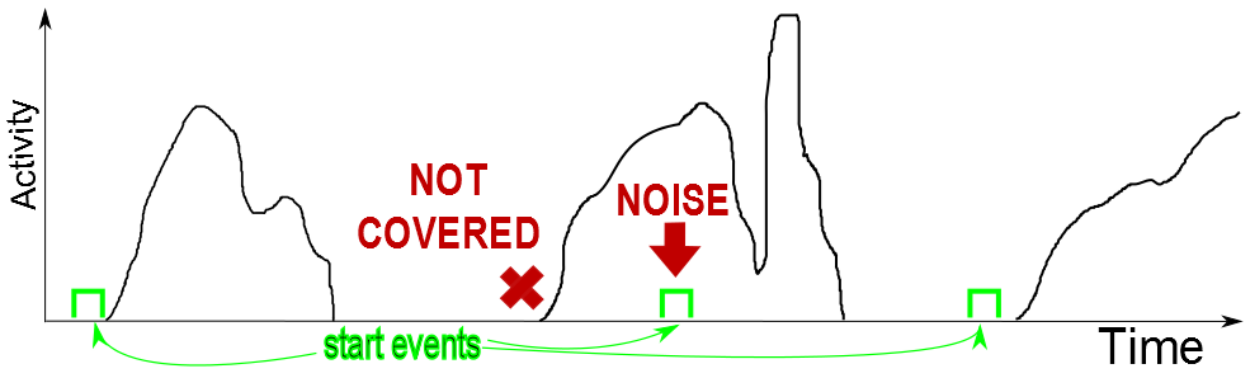
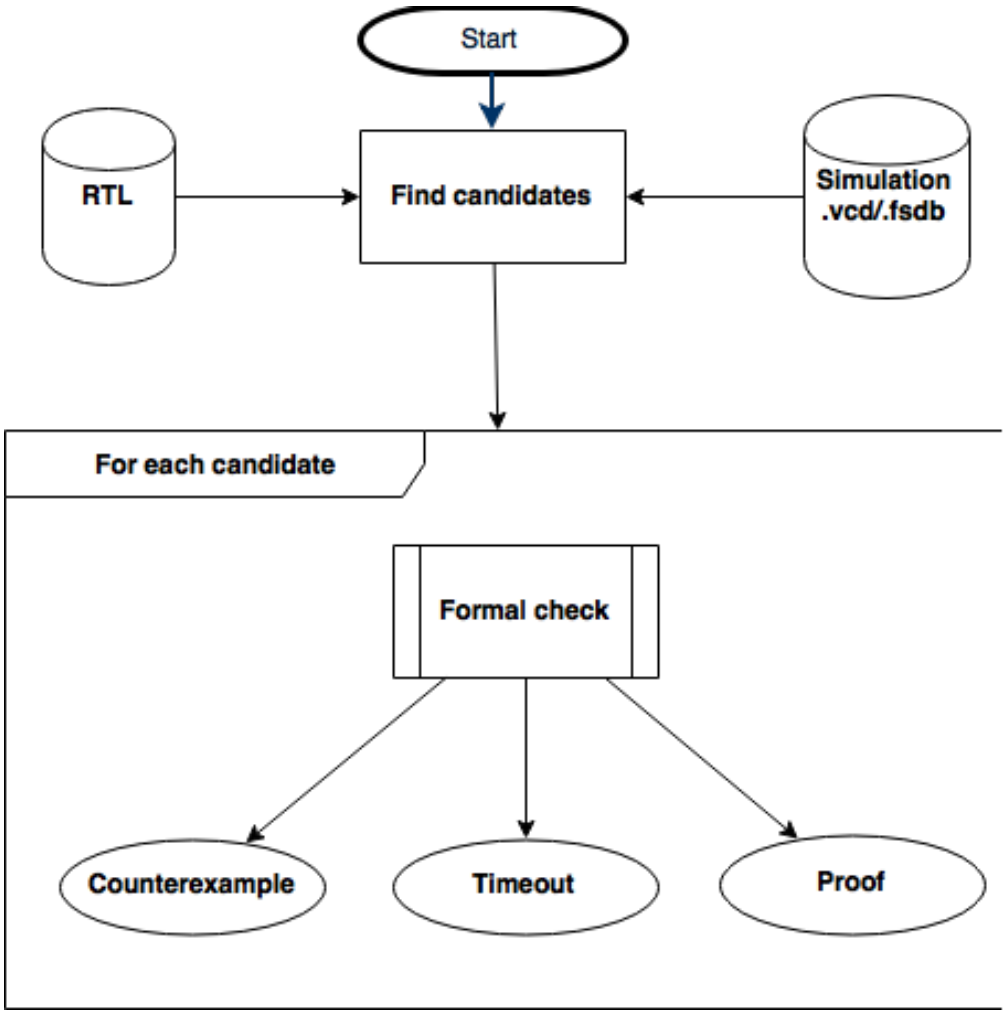
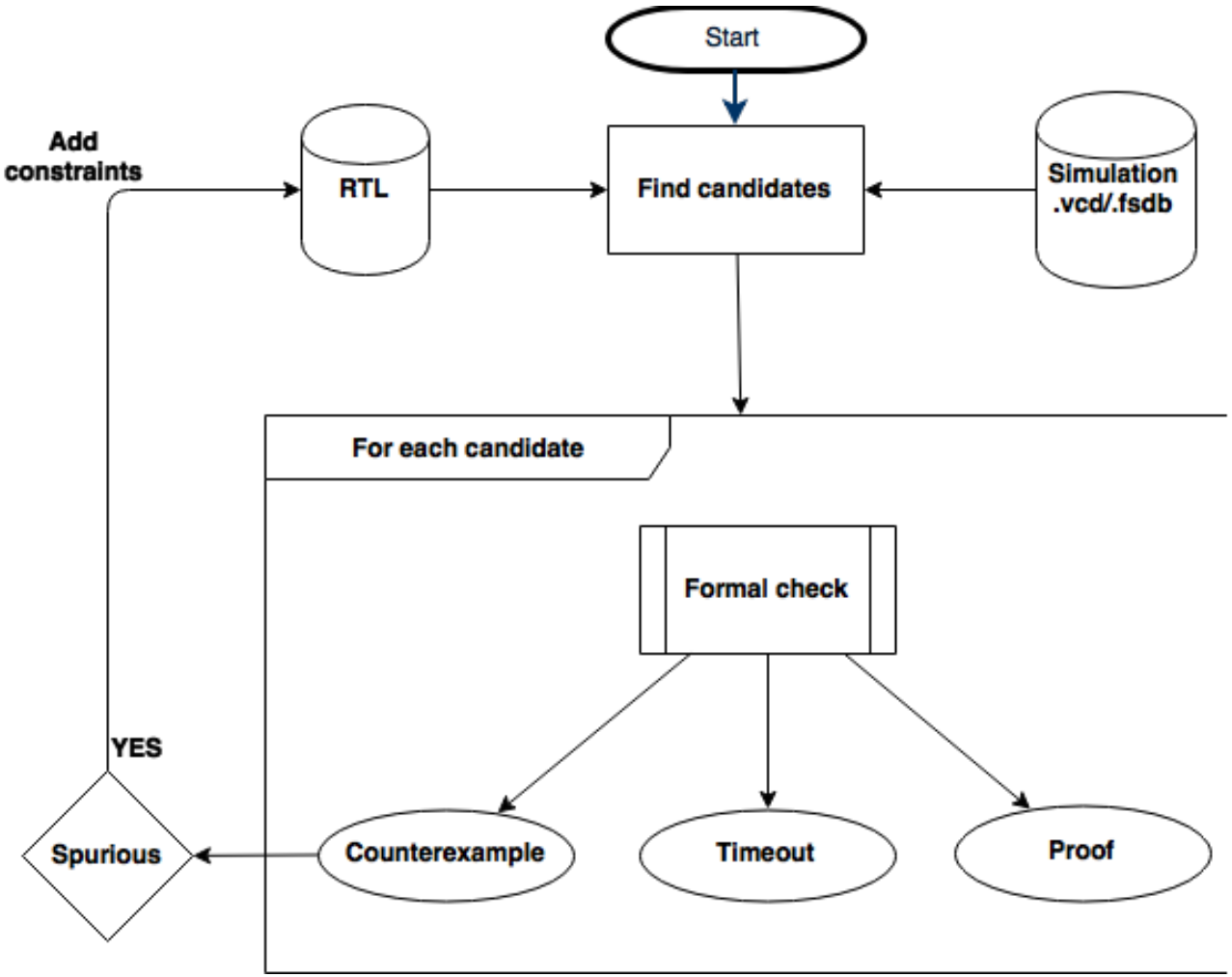


Figure: Coverage and noise

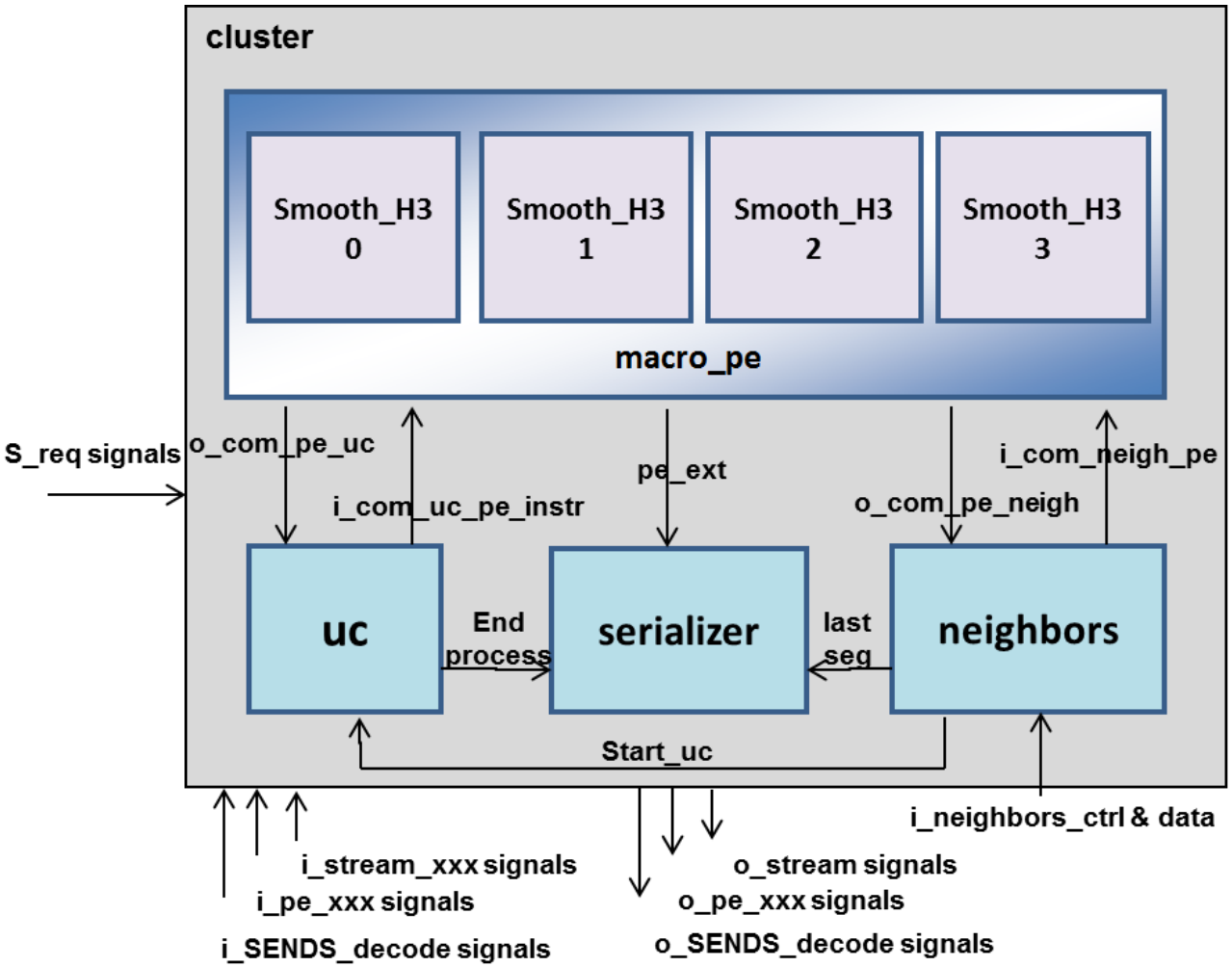
Automatic flow



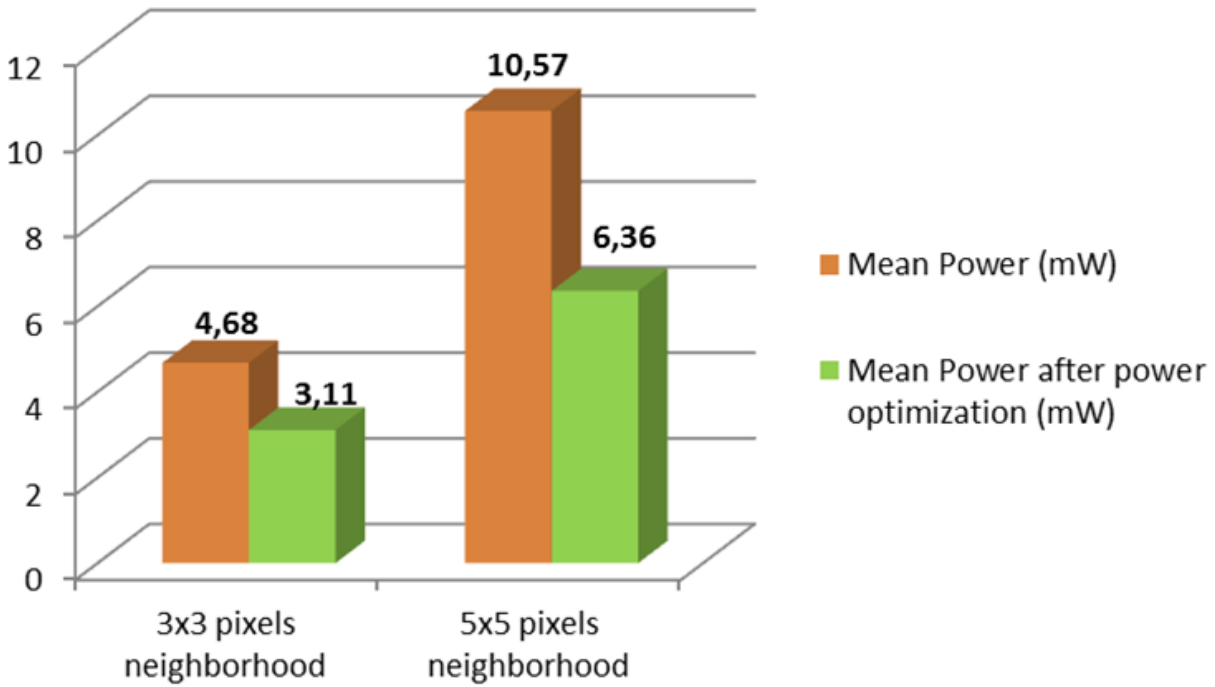
Semi-automatic flow



SENDS – A video processing architecture



SENDS results



Automatic mode results

design	power	power covered	# registers	reg-covered
1	4.169 mW	75.84%	26680	37.17%
2	7.163 mW	54.93%	9128	56.38%
3	0.479 mW	49.62%	1352	82.84%
4	7.145 mW	49.47%	9128	13.56%
5	5.314 μW	31.04%	326	33.74%
6	0.606 mW	16.30%	2070	6.96%
7	8.891 mW	15.58%	690	28.70%
8	92.491 mW	6.77%	30520	4.65%
9	55.851 mW	4.54%	107848	8.14%
10	92.444 mW	2.87%	114546	1.56%
11	1.430 μW	1.61%	162	14.81%
12	4.079 mW	0.70%	5292	0.15%
13	149.955 mW	0.61%	111012	1.11%

Main contributions

- Activity triggers = New class intermediate-block-site clock gating conditions
- Heuristical detection of activity triggers based on RTL simulation trace analysis
- Formal method to prove validity
- Semiautomatic and automatic methodology integrated within a commercial tool

Thank you!