

Power Reduction in Digital Circuits

Ph.D. Defense

Jan Láník

16th June 2016

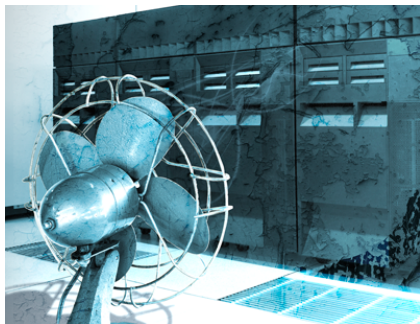


- CIFRE Ph.D. Thesis
- Verimag (University of Grenoble)
- Industrial partner: Atrenta/Synopsys
- Supervisors: Oded Maler (Verimag), Fahim Rahim (Synopsys)
- 2012-2016

Introduction

Motivation

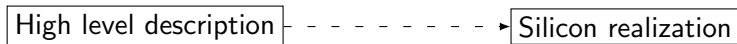
Power consumption of integrated chips is an issue



Our work: yet another attempt to reduce power consumption

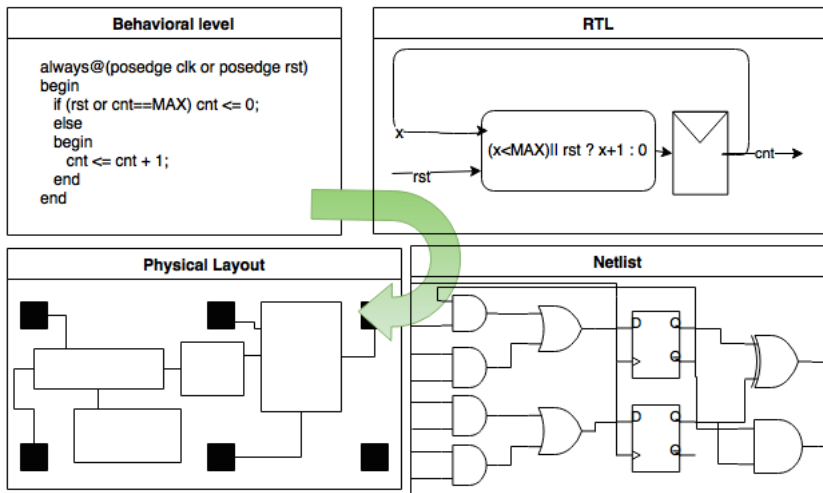
Hardware synthesis

Hardware analog of a compilation in software



- Crucial step in hardware production
- Optimizations for speed, space and power
- Many intermediate steps
- Many degrees of freedom

Some Steps in the Hardware Synthesis



Switching power dissipation at a gate

$$P = \frac{1}{2} V_{dd}^2 C_i E_i f$$

V_{dd} ... supply voltage

C_i ... capacitance connected to the output of gate i

E_i ... switching activity (number of switches per cycle)
of gate i

f ... clock frequency

Our methods

Two methods for switching activity reduction :

- 1 **Power Aware Synthesis** Optimization of combinatorial logic synthesis
- 2 **Activity Triggers** Optimization of sequential logic blocks by clock gating (industrial project)

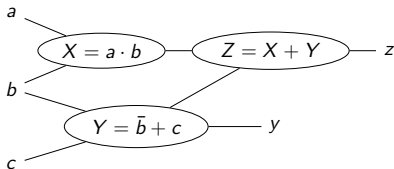
The two methods are independent.

Common points: RTL/Netlist level, modeling input to achieve statistical switching reduction

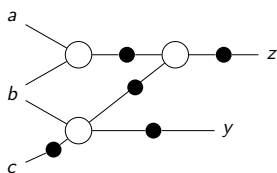
Power Aware Synthesis

Our place in the synthesis flow

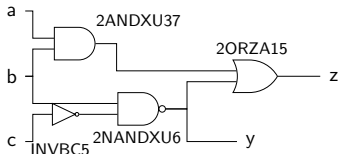
1) multilevel logic specification



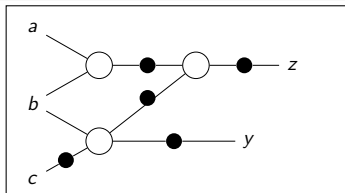
2) AIG



3) Technology dependent representation



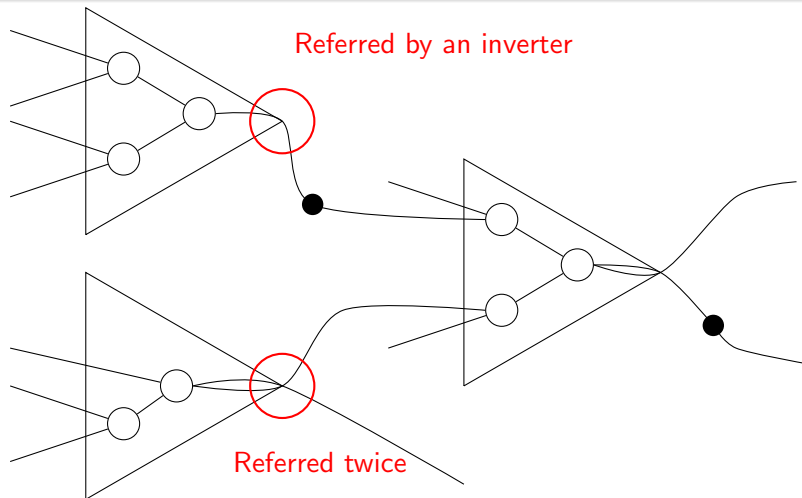
AIG (AND-Inverter graph)



- Acyclic directed graph
- Nodes: AND and NOT gates
- Efficient representation
- Not canonical
- Many optimizations

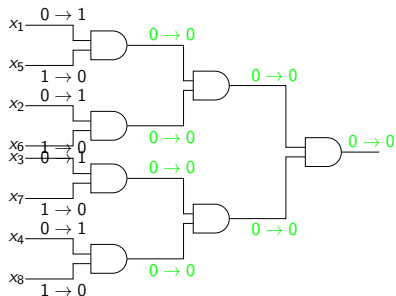
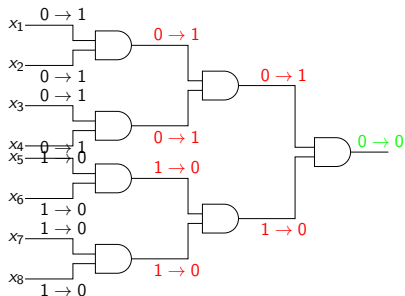
Our method: yet another optimization on the AIG level

AND cones in AIG



We want to optimize AIGs by re-arranging AND cones

2 ways to realize 8AND by 2ANDs



- we assume synchronized design, 0 time delay
- 1 switch = change of value at a gate output
- gate values determined by input values

Input stream and switching

- A circuit sees more than one transition during its lifetime
- Input stream : sequence of values as they are applied to the circuit inputs

$$\boxed{\text{Input stream}} + \boxed{\text{Internal structure}} = \boxed{\text{Actual switching}}$$

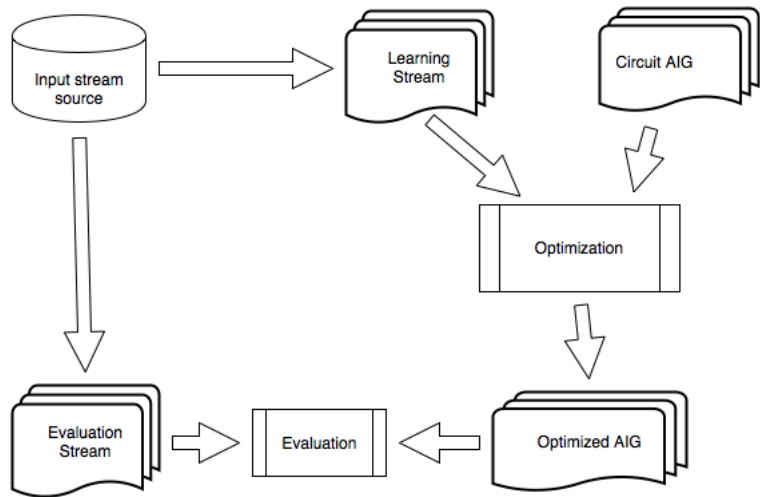
What is a 'typical' input stream?

$$\boxed{\text{Input model}} + \boxed{\text{Internal structure}} = \boxed{\text{Expected switching}}$$

Input model

- Ideally: Markov chain
- Realistically: Long input stream provided by designers

Optimization and evaluation flow



AND Cone optimization

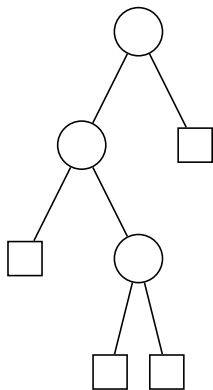
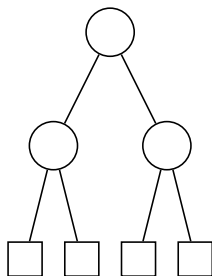
- An AND cone is semantically equivalent to an n -input AND gate
- **Goal: find 2AND realization for the given cone with a minimal switching w.r.t. the typical (training) stream**
- Constrained to minimal-depth 2AND (timing)

AND Cone optimization methods

Solution:

- 1 Enumerative
 - Problem space growing fast
 - Efficient up to approximately 8 inputs (symmetry reduction)
 - Sufficient for most of the cones in real designs
- 2 Layer based approximation
 - Optimal on 'layers'
 - Locally optimal
 - Efficient for larger cones

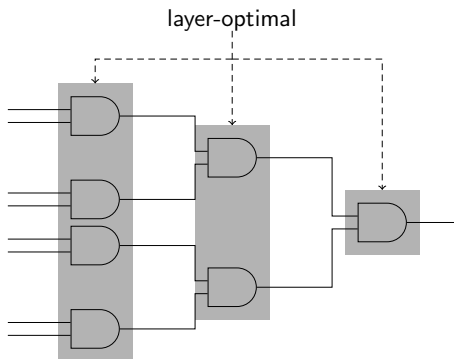
Balanced and unbalanced trees



Number of trees to check

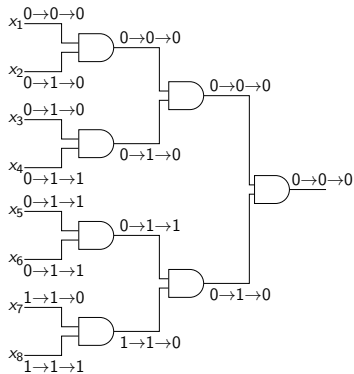
in	all trees	canonical	balanced	canonical & balanced
1	1	1	1	1
2	2	1	2	1
3	12	3	12	3
4	120	15	24	3
5	1680	105	480	30
6	3.0240e+04	945	4320	135
7	6.6528e+05	1.0395e+04	2.0160e+04	315
8	1.7297e+07	1.3514e+05	4.0320e+04	315
9	5.1892e+08	2.0270e+06	2.9030e+06	1.1340e+04
10	1.7643e+10	3.4459e+07	1.0161e+08	1.9845e+05
11	6.7044e+11	6.5473e+08	2.2353e+09	2.1830e+06
12	2.8159e+13	1.3749e+10	3.3530e+10	1.6372e+07
13	1.2953e+15	3.1623e+11	3.4871e+11	8.5135e+07

Layer based cone synthesis

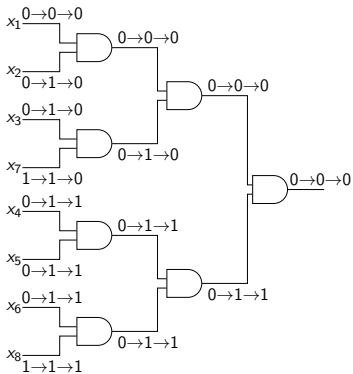


Each pairing of input signals into an AND gate produces certain switching number. Minimizing the switchings in the first level corresponds to minimal perfect matching in a weighted graph $O(n^3)$, Edmonds65, Lawrer76.

Examples of suboptimality

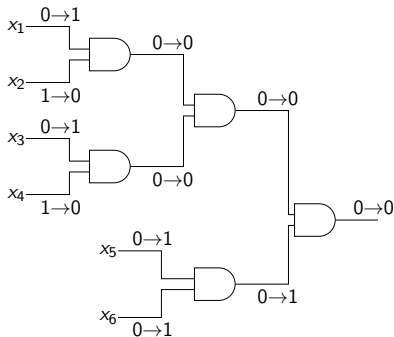


A level-greedy pairing
 with 6 switches

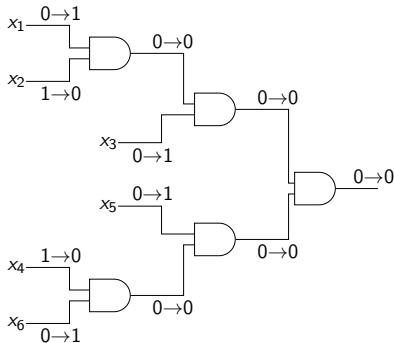


An optimal pairing
 with 5 switches

Examples of suboptimality: missing topologies



(a)



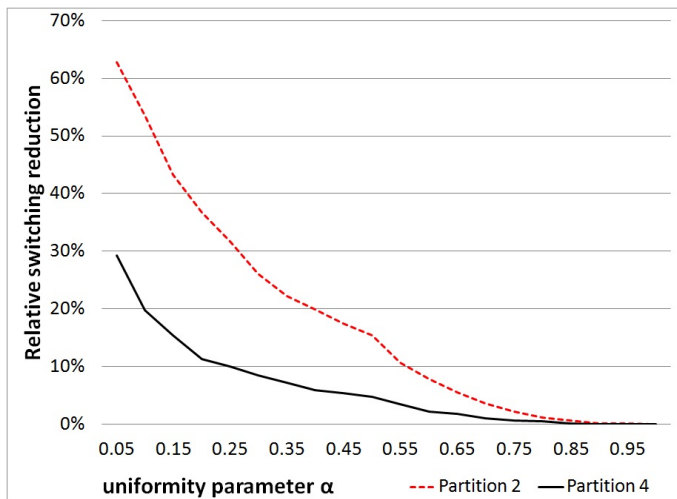
(b)

Evaluation scenarios

We evaluate on 2 classes of examples:

- 1 Synthetic products of Markov chains
 - Different forms of interaction/correlation between variables
 - Another parameter characterizes the amount of randomness/determinism
- 2 Verilog models of 2 small designs
 - A simple decoder for a hand held calculator
 - A Serial Peripheral Interface from Opencores

Synthetic examples



Small realistic circuits

Net effect of our method on AIG level

	maximum	minimum	level-greedy
Mini Instruction Decoder	250338	128118	158726
Core SPI (opencores)	20577	19681	19681

Interference due to other optimization methods

	maximum	minimum	level-greedy
Mini Instruction Decoder	73976	72288	72288
Core SPI (opencores)	19555	19233	19233

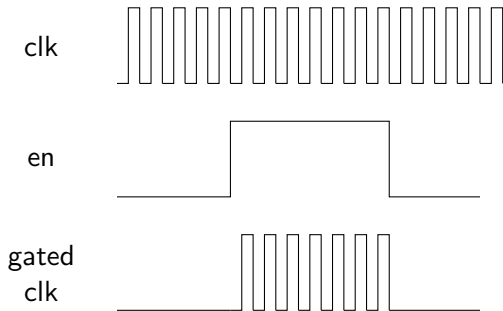
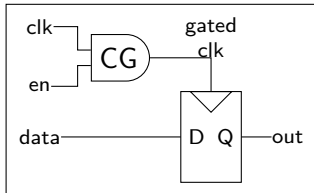
Shortcomings

- Preprocessing diminishes the savings
- Unclear if reduction preserved on mapped netlist

Activity Triggers

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

Coarse grained

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

Fine grained

- Small register groups deep in the designs
- Complex, not intuitive
- Need tools to find them
- Can be expensive

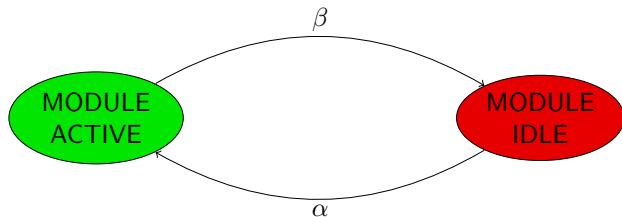
Clock gating conditions - granularity

Intermediate

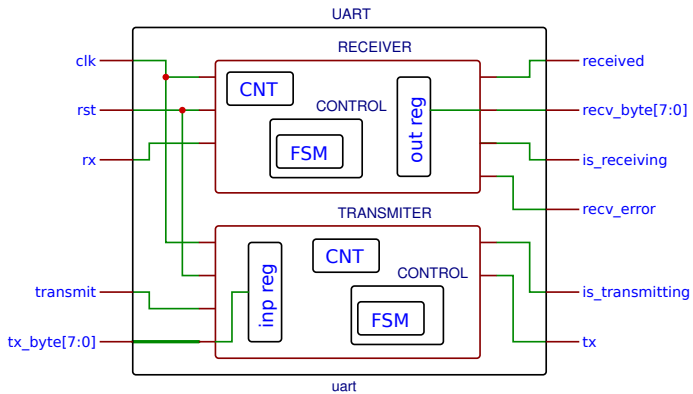
- Missing link
- Medium sized blocks
- Understandable, but not necessarily obvious
- Human designer should be able to find them if he did a time consuming detailed analysis
- Tools in demand

Activity Triggers

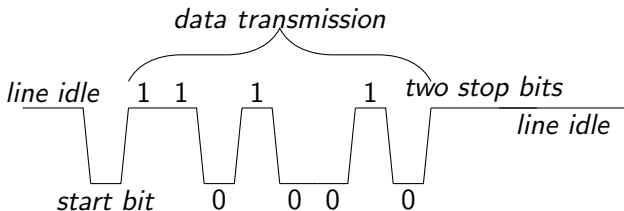
Events related to a change of activity status of a design block.



Example



UART - serial line transmission



Tool components

- 1 Detection of potential triggers
- 2 Formal verification (my contribution)

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)

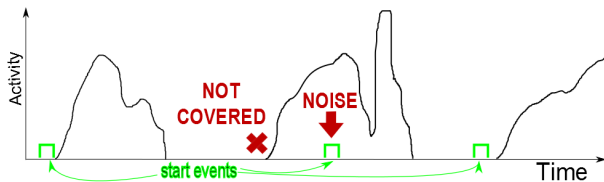
Potential start and stop signals location

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



Ranking of events

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

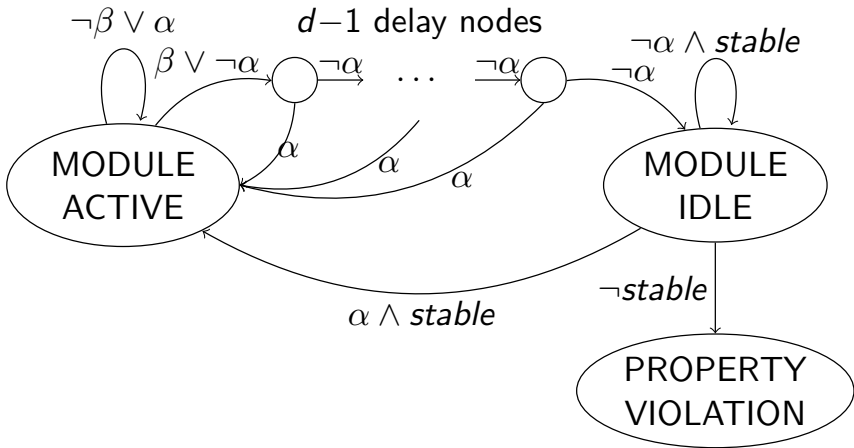


Formalization

An activity trigger is a triple (α, β, d)

- α ... start condition
- β ... stop condition
- d ... shutdown delay

Monitor automaton



Formalization (PLTL)

Stability of a signal:

$$stable(x) = (x \wedge \ominus(x)) \vee (\neg x \wedge \ominus(\neg x))$$

Stability of a block (set of signals):

$$stable(M) = \bigwedge_{x \in M} stable(x)$$

Formalization (PLTL)

Stop Condition:

$$\ominus^d \beta \wedge \bigwedge_{i=0}^d \ominus^i \neg \alpha$$

Start condition:

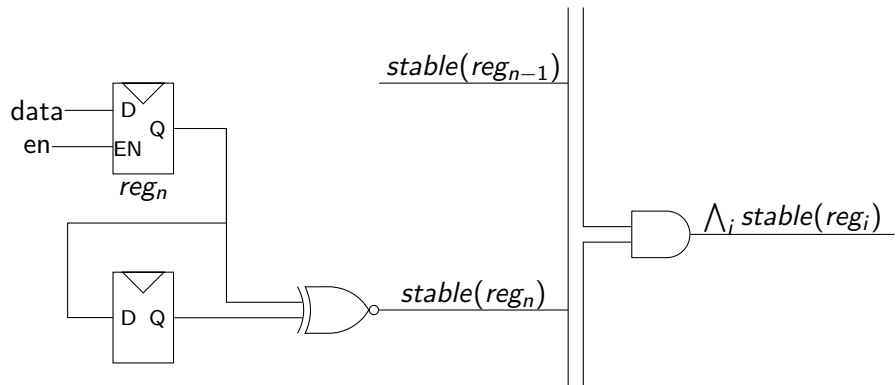
$$\alpha$$

Formal Verification

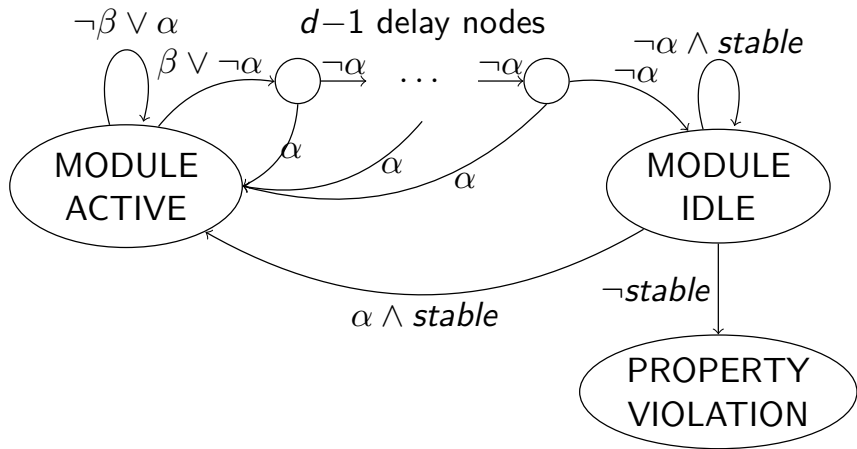
We need to prove:

$$\neg\alpha\mathcal{S} \left(\ominus^d\beta \wedge \bigwedge_{i=0}^d \ominus^i\neg\alpha \right) \Rightarrow \text{stable}(M)$$

Stability modeling



Monitor automaton



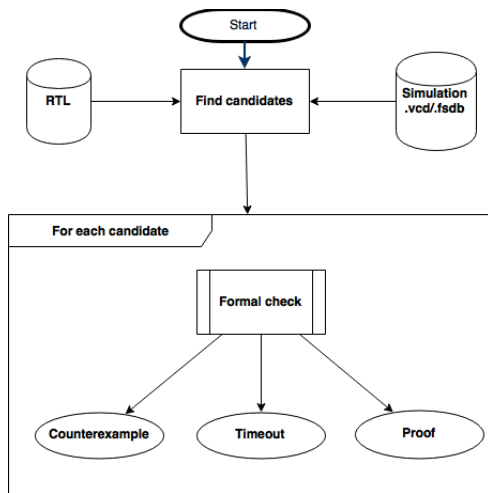
Formal verification flow

- 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)

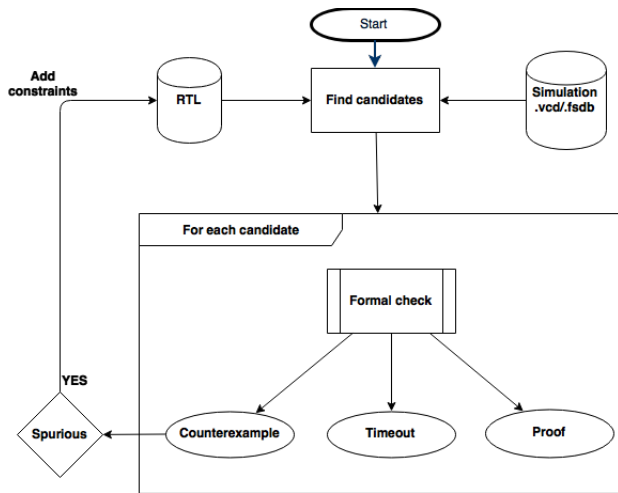
Constraint support

- Implicit assumptions often not present in the design
- Can be added in form of assertions
- Typical cases: configuration registers or input following specific pattern
- This is crucial to avoid spurious counter examples
- We support System Verilog Assertions (SVA)

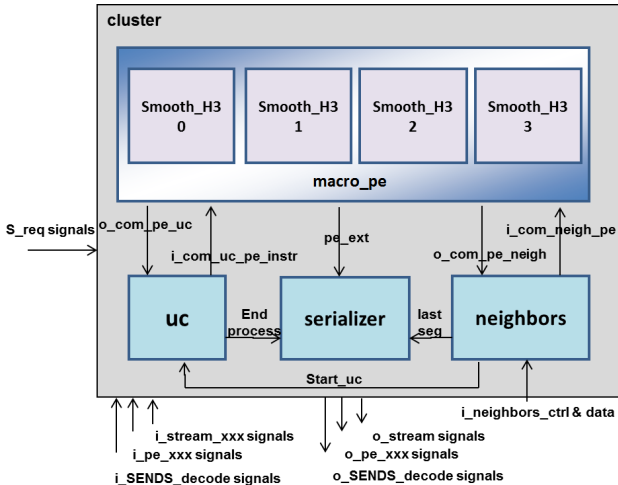
Automatic flow



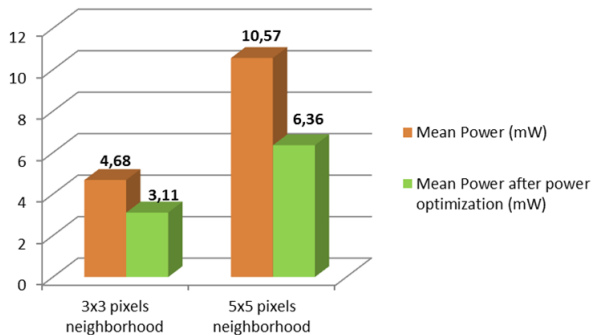
Semi-automatic flow



SENDS – A video processing architecture



SENDS results



Automatic mode results

design	power	power covered
1	4.169 mW	75.84%
2	7.163 mW	54.93%
3	0.479 mW	49.62%
4	7.145 mW	49.47%
5	5.314 μ W	31.04%
6	0.606 mW	16.30%
7	8.891 mW	15.58%
8	92.491 mW	6.77%
9	55.851 mW	4.54%
10	92.444 mW	2.87%
11	1.430 μ W	1.61%
12	4.079 mW	0.70%
13	149.955 mW	0.61%

Conclusions

Activity Triggers: Summary

- New class intermediate-block-size clock gating conditions
- Heuristic detection based on trace analysis
- **Formal proof of validity**
- Semi-automatic and automatic methodology
- Integrated within a commercial tool

Activity Triggers: Limitations

- Constraints
- Room for improvement in scalability
- Room for improvement in detection



Power Aware Synthesis: Summary

- Power Aware Synthesis
- Optimizing switching in combinational logic
- Average case optimization
- Implemented as prototype within ABC
- Experimented on synthetic as well as realistic models
- Works well on AIG level

Power Aware Synthesis: Limitations

- Interference from other optimization techniques
- Unclear if switching reduction can be preserved in the next synthesis steps

References I

-  Jan Lanik and Oded Maler. “On Switching Aware Synthesis for Combinational Circuits”. In: *Hardware and Software: Verification and Testing*. Ed. by Nir Piterman. Vol. 9434. Lecture Notes in Computer Science. Springer International Publishing, 2015, pp. 276–291.
-  Jan Lanik et al. “Reducing power with activity trigger analysis”. In: *Formal Methods and Models for Codesign (MEMOCODE), 2015 ACM/IEEE International Conference on*. IEEE. 2015, pp. 169–178.

Thank you!