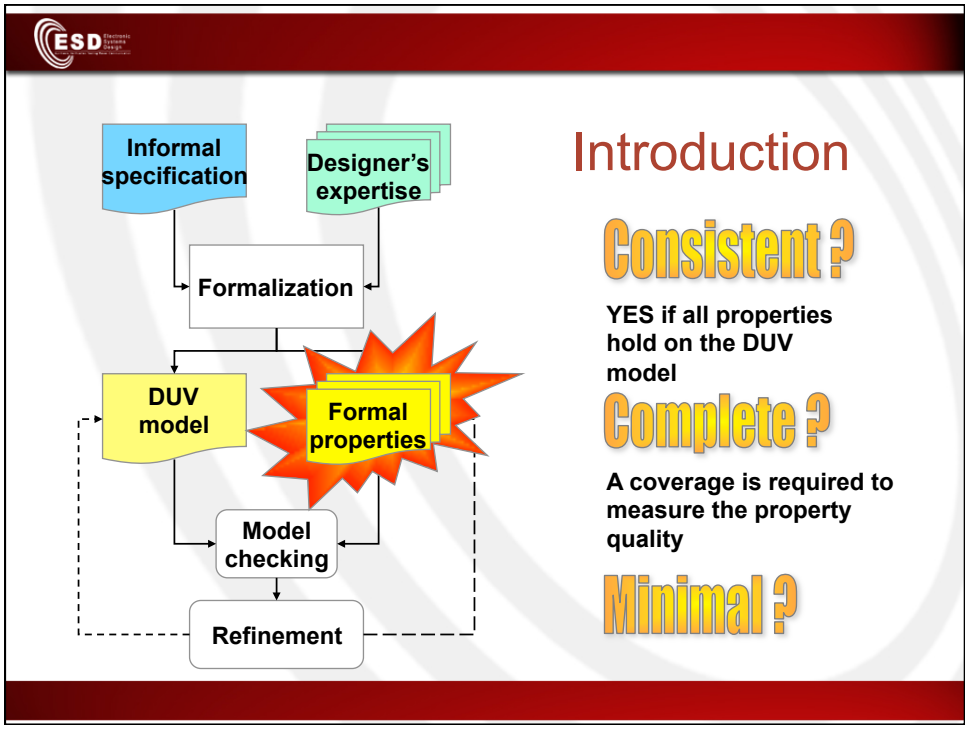
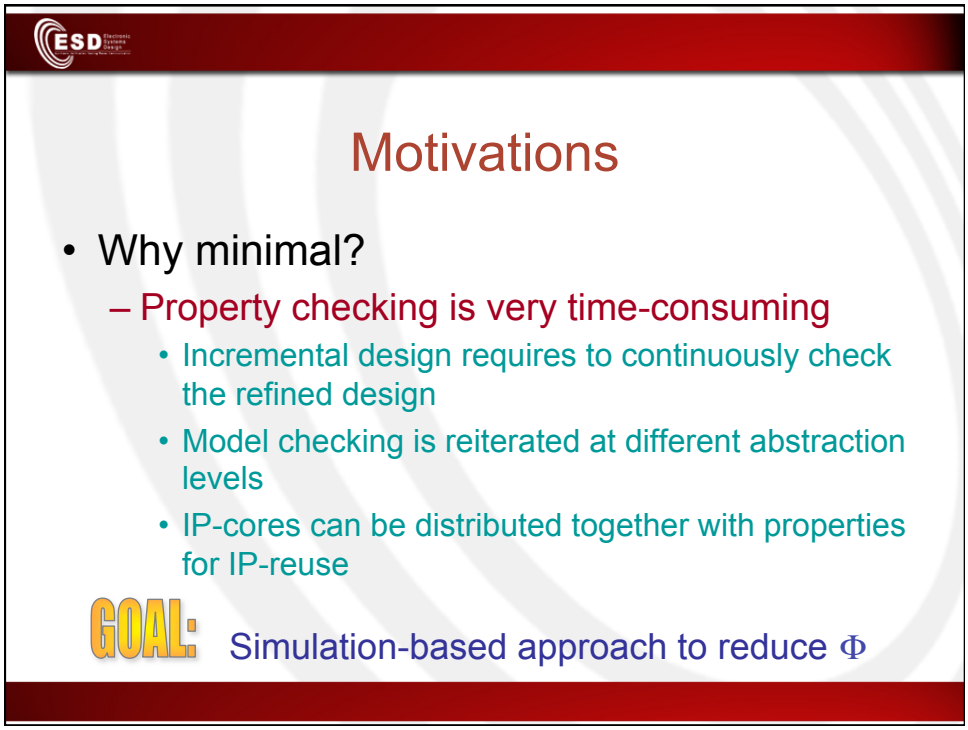
Overspecification Analysis

Graziano Pravadelli
Dipartimento di Informatica Università di Verona

Agenda

- Introduction & Motivations
- State of the art
- Background
- Methodology
  - Redundancy analysis
- Fault model
- Experimental results


Informal specification
Designer's expertise
Formalization
DUV model
Formal properties
Model checking
Refinement
Introduction
Consistent ?
YES if all properties hold on the DUV model
Complete ?
A coverage is required to measure the property quality
Minimal ?


Motivations
• Why minimal?
– Property checking is very time-consuming
• Incremental design requires to continuously check the refined design
• Model checking is reiterated at different abstraction levels
• IP-cores can be distributed together with properties for IP-reuse
GOAL: Simulation-based approach to reduce Φ

State of the Art

- Some papers address property completeness and vacuity analysis by
  - formal methods
  - simulation-based techniques

Property minimization not yet investigated

Background

- A realization $\sigma = (M, a)$ is a couple where
  - M is a model
  - a is a function which assigns values to M inputs
- Given a realization σ and a property φ, the interpretation of φ in σ can be T or F
- φ is valid ($\models \varphi$) if it is T in all the interpretations
  - Axioms: a set of valid properties

Background

- A set of properties Φ is satisfiable if there exists an interpretation where all properties in Φ are T
- φ is a logical consequence of ψ ($\psi \models \varphi$) if φ is T in all the interpretations where ψ is also T
- Modus ponens (MP)
  - if φ and $\varphi \rightarrow \psi$ are T in σ then ψ is T in σ
  - Modus ponens preserves logical consequence

Background

- A deduction from Φ is a finite succession of properties which are axioms, or are in Φ or are obtained from previous properties by MP
- φ can be deducted from Φ ($\Phi \vdash \varphi$) if there is a deduction from Φ where the last property is φ
- Given a set of properties Φ and a property φ, a logic is complete if it always happens that $\Phi \models \varphi \leftrightarrow \Phi \vdash \varphi$

Methodology

Main Idea

Removing redundant properties

Given a set of properties Φ, $\varphi \in \Phi$ is redundant if $\Phi \setminus \{\varphi\} \models \varphi$

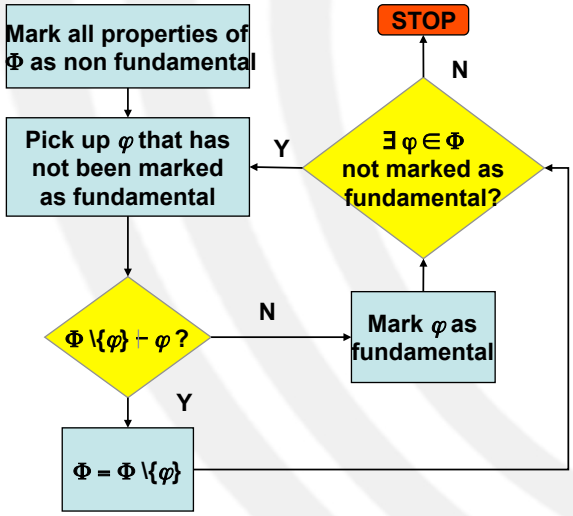
Property coverage

~~**Theorem proving**~~

- It identifies redundant properties by deduction
  - Too time-consuming
  - No automatic

- It Identifies "essential properties"
  - Based on high-level fault simulation
  - Automatic & fast

Redundancy Analysis



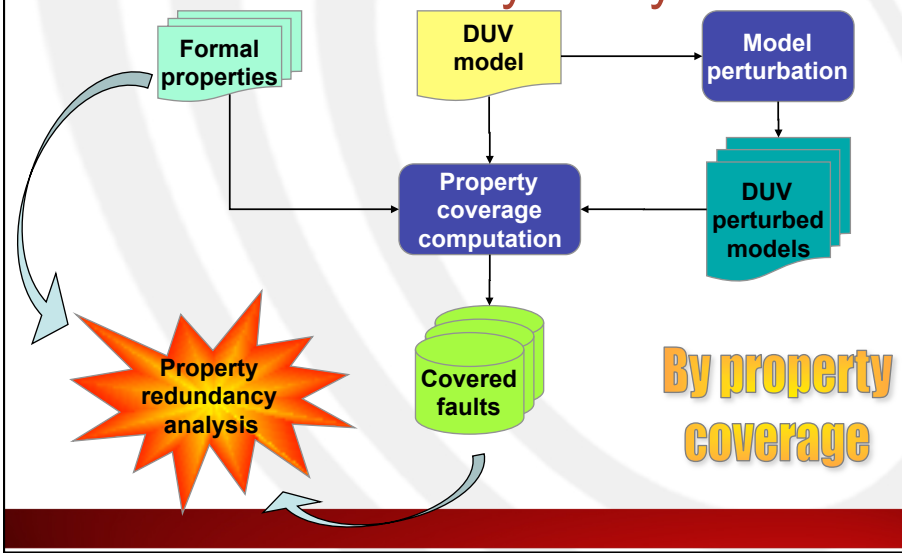
By theorem proving

$|\Phi|=N$

N proofs

N*exptime

Redundancy Analysis



Property Coverage

- Is there a relation between fault detection and property coverage?
  - Properties represent the golden model
    - They hold on the design implementation
  - Implementation is perturbed by using a high-level fault model

Do all properties hold on the perturbed implementations?

Properties are not able to distinguish between faulty and fault-free implementation
The set of properties is ***INCOMPLETE*** !

Redundancy Analysis

- Consider
  - M, the model of the DUV
  - $\Phi = \{\varphi_1, \ldots, \varphi_m\}$, the set of properties

  $\} \; M \models \Phi$
  - $F = \{f_1, \ldots, f_n\}$, the set of detectable faults
  - $M_F = \{M_{f1}, \ldots, M_{fn}\}$, the set of perturbed models
  - $F_1, \ldots, F_m$, the set of faults detected by $\varphi_1, \ldots, \varphi_m$

Conjecture 1 $\quad \forall i \left(\Phi \setminus \{\varphi_i\} \models \varphi_i\right) \Rightarrow \left(F_i \subseteq \bigcup_{i \neq j} F_j\right)$

Conjecture 2 $\quad \forall i \left(F_i \subseteq \bigcup_{i \neq j} F_j\right) \Rightarrow \left(\Phi \setminus \{\varphi_i\} \models \varphi_i\right)$

Proof of Conjecture 1

Conjecture 1 $\quad \forall i \left(\Phi \setminus \{\varphi_i\} \models \varphi_i\right) \Rightarrow \left(F_i \subseteq \bigcup_{i \neq j} F_j\right)$

If $\Phi \setminus \{\varphi_\iota\} \models \varphi_\iota$ then $M_{fj} \models \Phi \setminus \{\varphi_\iota\} \Rightarrow M_{fj} \models \varphi_\iota$

$\Downarrow$

$M_{fj} \not\models \varphi_\iota \Rightarrow M_{fj} \not\models \Phi \setminus \{\varphi_\iota\}$

$\Downarrow$

If a fault is detected by φ_ι then
it is also detected by $\Phi \setminus \{\varphi_\iota\}$

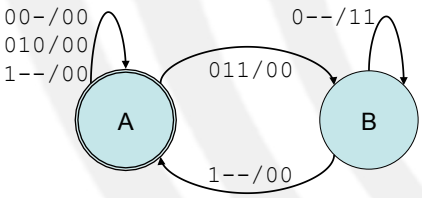
Consequence

$$\forall i \left(\mathrm{F}_i \not\subset \bigcup_{i \neq j} \mathrm{F}_j \right) \Rightarrow \left(\Phi \setminus \{\varphi_i\} \not\models \varphi_i \right)$$

If a property covers a fault
that is not covered by any other property then
it is not logical consequence of the others,
thus it is not redundant

Counterexample of Conjecture 2

Conjecture 2 $$\forall i \left(\mathrm{F}_i \subseteq \bigcup_{i \neq j} \mathrm{F}_j \right) \Rightarrow \left(\Phi \setminus \{\varphi_i\} \models \varphi_i \right)$$



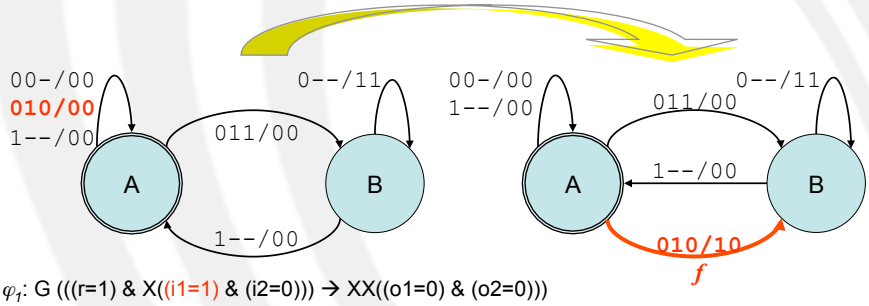
- Fault model flips one by one the output of M

- φ_1: G (((r=1) & X((i1=1) & (i2=0))) → XX((o1=0) & (o2=0)))
- φ_2: G (((r=1) & X((i1=0) & (i2=0))) → XX((o1=0) & (o2=0)))
- They cover the same set of faults, but neither $\varphi_1 \models \varphi_2$ nor $\varphi_2 \models \varphi_1$

Consequences

- Conjecture 1
  - The methodology provides a necessary condition for property minimization
  - Faster than theorem proving
- Conjecture 2
  - The methodology does not provide a sufficient condition
  - It depends on the adopted fault model

Fault Model Dependency



φ_1: G (((r=1) & X((i1=1) & (i2=0))) → XX((o1=0) & (o2=0)))
φ_2: G (((r=1) & X((i1=0) & (i2=0))) → XX((o1=0) & (o2=0)))

- f cannot be modeled by the *"flipping fault model"* F
- φ_1 or φ_2 is redundant for F

- φ_1 detects f while φ_2 does not ➡ F U {f} shows that $\varphi_2 \not\models \varphi_1$
- Analogously, there exists $f' \notin$ F such that F U {f'} ➡ $\varphi_1 \not\models \varphi_2$

Final Methodology

1. Computing the property coverage
2. Compare the sets of faults detected by properties (N properties → N comparisons)
3. Mark the set of properties that are surely non redundant
4. Use a Theorem Prover to analyze remaining properties

Which fault model?

Fault Model

Bit Coverage ∪ **Transition Fault**

- Bit failure
  - a=b ➡ a=f(b)
  - Each bit can be stuck-at 0 or stuck-at 1
- Condition failure
  - If (a==b) ➡ if(f(a==b))
  - Each condition can be stuck-at T or stuck-at F
- Single fault

- Output failure
  - Like bit failure, but only on POs
- Transition failure
  - It changes the destination state of a transition
- Multiple faults
  - Output failure or transition failure or output + transition failure

Experimental Results

Design	In	Out	#Gates	#FF	Φ	#BC	#TF	#BC+TF
b01	4	2	1319	5	13	201	536	737
b02	3	1	296	4	9	52	161	213
b03	6	4	629	30	18	159	81	240
b06	4	6	179	9	23	120	812	932

		BC		TF		BC+TF			
Design	Φ	NR	Time(s)	NR	Time(s)	NR	Time(s)	R	NR
b01	13	7	0.2	4	0.3	7	0.5	4	9
b02	9	3	0.3	4	0.3	4	0.3	2	7
b03	18	5	22.4	3	22.5	5	23.0	5	13
b06	23	9	8.9	13	9.0	13	9.2	1	22

Experimental Results

		Time (s)				Saving (%)		
Design	Ψ	BC	TF	BC+TF	T.P.	BC	TF	BC+TF
b01	13	8.3	12.6	8.6	20.2	58.9	37.6	57.4
b02	9	0.5	0.4	0.6	0.8	37.5	50.0	25.0
b03	18	65.9	76.1	66.4	98.5	33.1	22.7	32.6
b06	23	52.0	44.5	44.7	69.7	25.4	36.1	35.9

Conclusions

- Simulation-based approach…
  - …to analyze logical consequence of properties
  - …to remove redundant properties
- It provides
  - a necessary condition for logical consequence
  - > 30% of saving time with respect to TP
- It does not provide
  - sufficient condition, it depends on the fault model

References

- S. Brait, F. Fummi and G. Pravadelli *"On the Use of a High-Level Fault Model to Analyze Logical Consequence of Properties"*, Proc. of ACM/IEEE MEMOCODE 2005