Complete Functional Verification

Joerg Bormann

joerg.d.bormann@web.de

Contents

- Characterisation of Complete Functional Verification
- Technical Components
- Methodology
- Application Experience

Literature: Bormann: Vollständige funktionale Verifikation (Complete functional Verification), Dissertation, University of Kaiserslautern, 2009

Overview

Complete Functional Verification is a self-dependent verification approach.

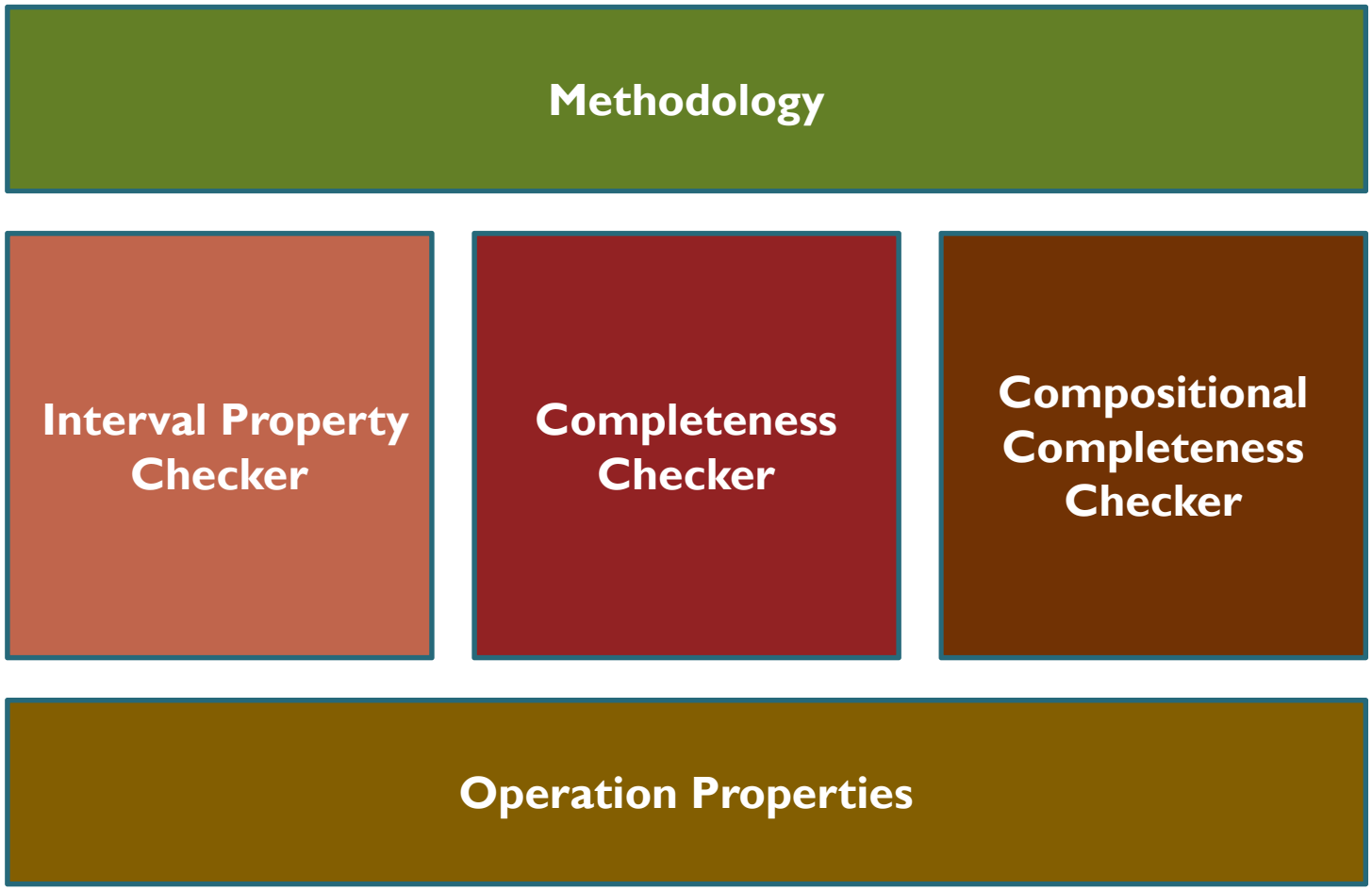
- Industrially applied
- For digital synchronous modules (up to ~ 200 k LOC)
- Formal only
- Verifies entire functionality of a module
- Proves that functionality is completely verified

- Alternative to simulation based verification approaches for modules, e.g., coverage driven random pattern simulation

Quality/Cost Metrics For Verification

- Final circuit quality
- Verification cost
  - Human effort
  - Hardware and software usage
- Integration into the industrial environment
  - Processes
  - Mindset

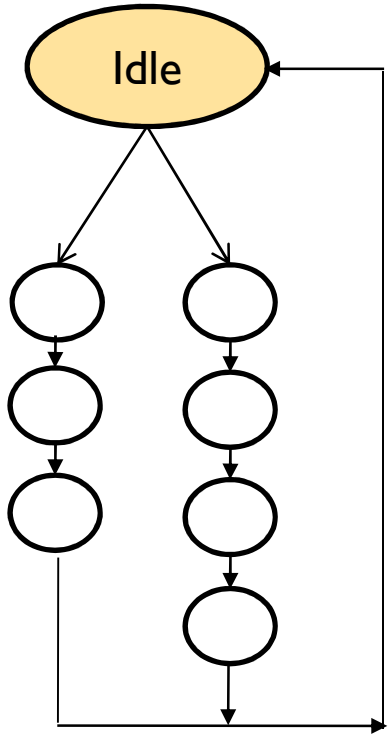
Components of Complete Functional Verification



OPERATION PROPERTIES

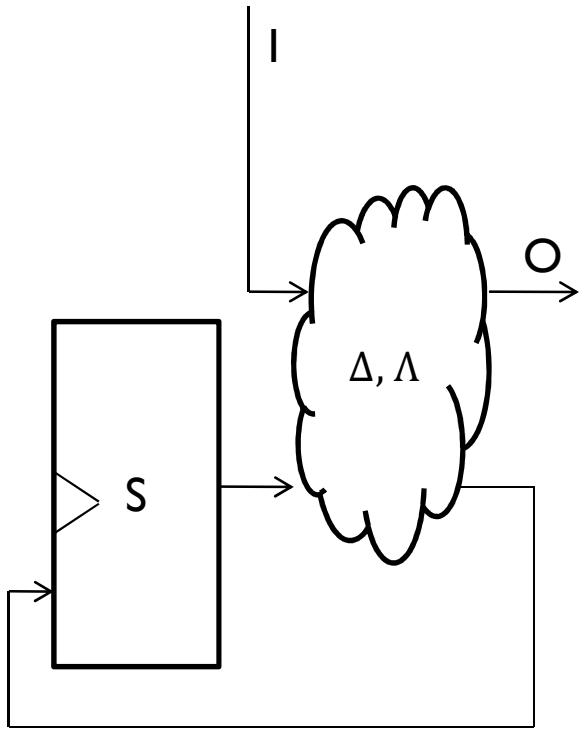
J. Bormann, S. Beyer, A. Maggiore, M. Siegel, S. Skalberg, T. Blackmore, F. Bruno, "Complete Formal Verification of TriCore2 and Other Processors," *DVCon* 2007

Operation Properties



- Observation: Inputs at specific points in time determine circuit behavior over several succeeding clock cycles.
  - Controller: In the idle state the inputs determine which type of transaction is to be executed next.
  - Pipelines: Inputs to first pipe stage determine behavior of other stages at later clock cycles.
- Example operations:
  - Requests of a bus bridge
  - Arbitration cycle
  - Instruction execution of processor
- In general: Incoming transaction makes circuit switch from one conceptual state to another while producing outgoing transaction(s).

Hardware Model



- Next state function Δ
- Output function Λ of primary outputs
- Set Σ of reset states

- Traces:
  - Inputs I
  - States S — $S_0 \in \Sigma, S_{n+1} = \Delta(S_n, I_n)$
  - Outputs O — $O_n = \Lambda(S_n, I_n)$

- Automaton M is predicate M(I,S, O) about traces

Operation Properties

A ***timed Boolean expression*** is a LTL formula using only the operators

$$P \wedge Q \mid P \vee Q \mid \sim P \mid \mathbf{X}(P)$$

An ***operation property*** is a LTL formula of the following form

$$\mathbf{G}\,(SC \wedge IC \Rightarrow OC \wedge \mathbf{X}^{t_{end}}\, EC)$$

with SC, IC, OC and EC being timed Boolean expressions.

The sub-formulas of an operation property specify the following conditions:

SC:	Start State Condition	EC:	End state Condition
IC:	Input Condition	OC:	Output Condition
t_{end}:	duration of the operation		

An operation property specifies a behavior where the design makes a transition between conceptual states specified by SC and EC within t_{end} clock cycles. This transition is triggered by inputs fulfilling IC and produces outputs fulfilling OC.

User-level Language

Syntactic sugar: bounded operators, such as

Time points

t	(universally quantified time point)
$T_i = T_{i-1} + n_i \,..\, m_i$ awaits p_i	(Time of external event p_i, bounded)

Bounded operators

at $T_i + k$: p;	(p holds at time Ti + k)
during $[T_i + k, T_j + v]$: p;	(p holds always in the interval)
within $[T_i + k, T_j + v]$: p;	(p holds once in the interval)
prev(expr), next(expr)	(evaluate expr at previous / next time point)

Can be combined with additional techniques for unbounded operators introduced by

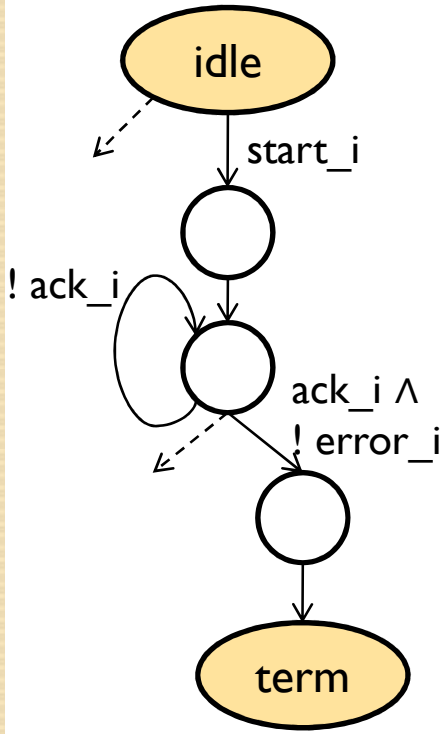
$T_i \geq T_{i-1} + n_i$ awaits p_i	(Time of external event p_i, potentially ∞)

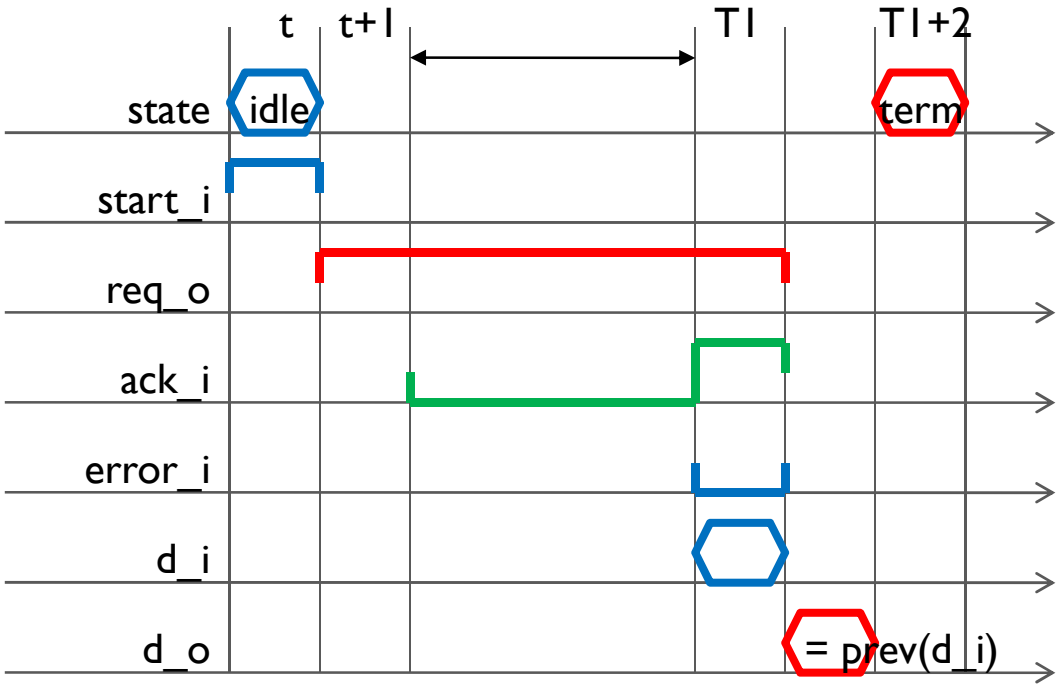
Example

timepoints $T1 \geq t + 2$, awaits ack_i = 1;

SC	at t:	state = idle;			
IC	at t:	start_i = 1;	during [t+1,T1]:	req_o = 1;	OC
	at T1:	error_i = 0;	at T1+1:	d_o = prev(d_i);	
			during [T1+1,T1+2]:	req_o = 0;	
			at T1+2:	state = term;	EC

t_{end}

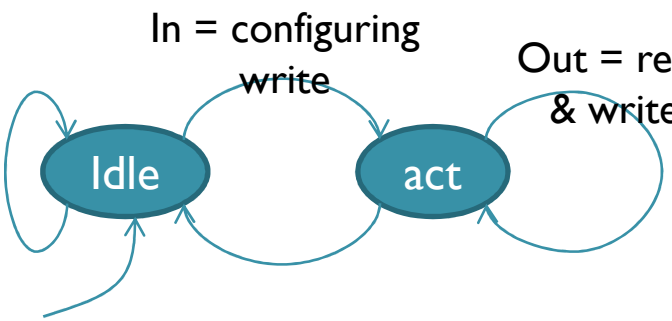


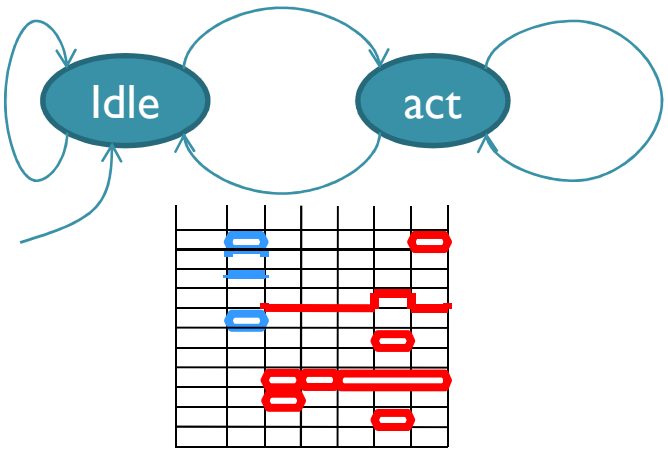


COMPLETENESS CHECKER

Literature: Bormann, Busch: Method For The Determination of the Quality of a Set of Properties, Usable for the Verification And Specification of Circuits,
US Patent 7571398, priority Sep. 2005, granted Aug 4th, 2009

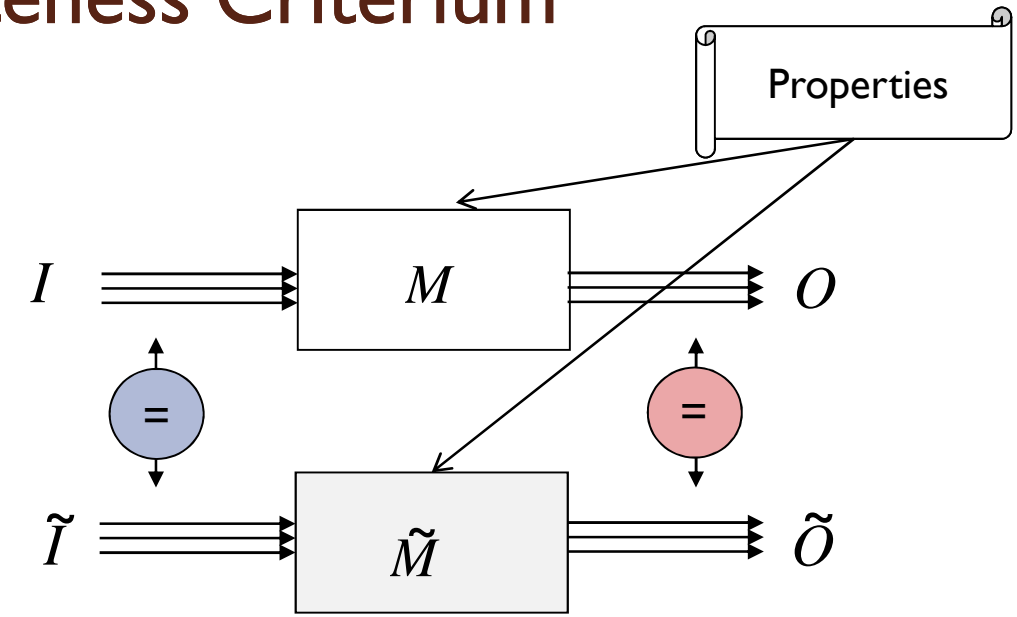
Transaction and Operation Automata





- Automata with transitions formed by operations
  - Conceptual states = start and end states
  - The conceptual reset state contains Σ
- Transaction Automaton
  - Transactions treated atomically
  - Similar to transaction level models in simulation
- Operation Automaton
  - Transitions = operation properties
  - Cycle accurate representation
  - Less abstraction but more structure
- Complete Functional Verification is an equivalence verification between RTL model and the operation automaton

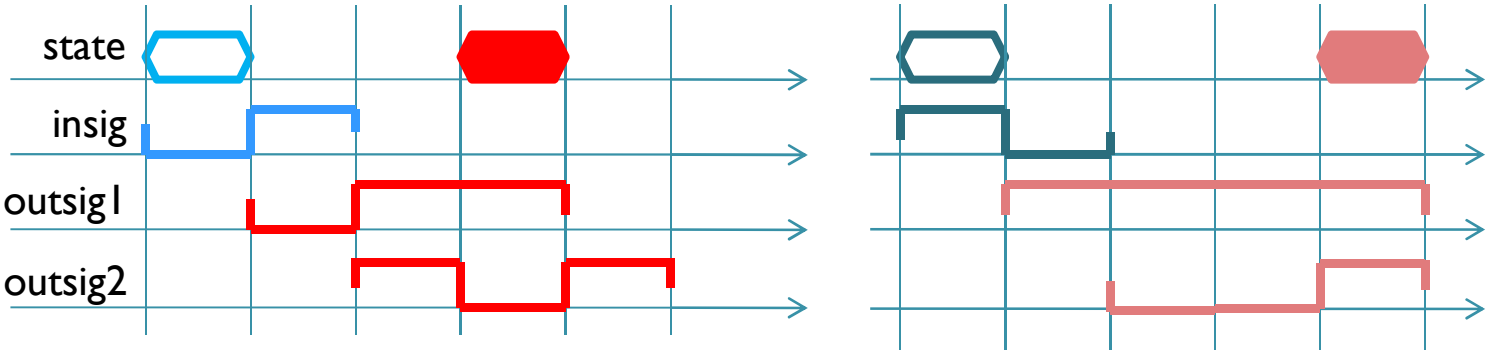
Completeness Criterium



$$\left(\mathbf{G}(I = \tilde{I}) \wedge \mathbf{G}\, P_{free} \wedge \mathbf{G}\, \tilde{P}_{free}\right) \Rightarrow \mathbf{G}(O_{free} = \tilde{O}_{free})$$

- A set of properties is called complete, if any two circuits fulfilling the properties are sequentially equivalent.
- Sequential equivalence check is computationally hard, exacerbated by the indirect description of the circuits by properties.
- Operation properties allow to use inductive argument that can be checked quickly and implies completeness.

Basic Idea



- Basic idea: For every input trace, there must be a chain of operation properties $P_0, P_1, \ldots$ that uniquely determines the output trace:

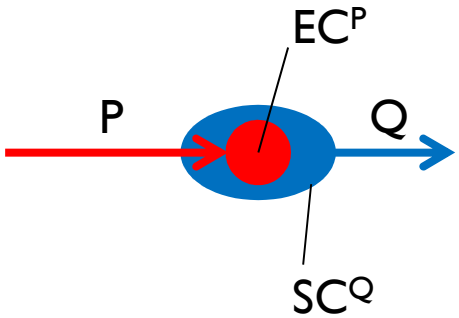
$$\Sigma \wedge \prod_{i \geq 1} \mathbf{X}^{t_{i-1}} P_i, \text{where } t_0 = 0, t_i = t_{i-1} + t^i_{end}$$

- User input for ensuring the existence of chains:
  - Properties P_i and their duration t^i_{end}
  - Operation Graph

Chain Building Checks

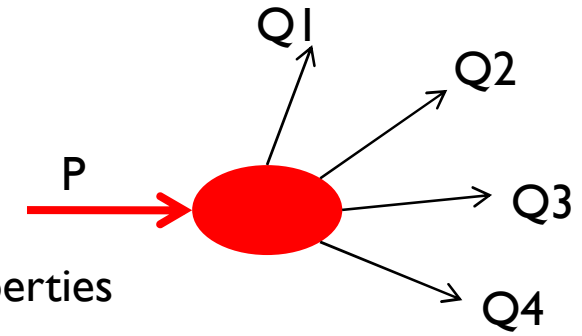
- **Successor Test:** For every property P that is followed by Q in the Operation Graph

$$EC^{P}_{free} \Rightarrow SC^{Q}_{free}$$



- **Case Split Test:** For every property P and all properties Q1, Q2, Q3, … that follow P in the Operation Graph check that the input conditions of Q1, Q2, Q3, … cover all possible input traces, i.e.,

$$IC^{Q1}_{free} \vee IC^{Q2}_{free} \vee IC^{Q3}_{free} \vee IC^{Q4}_{free} \vee \ldots$$



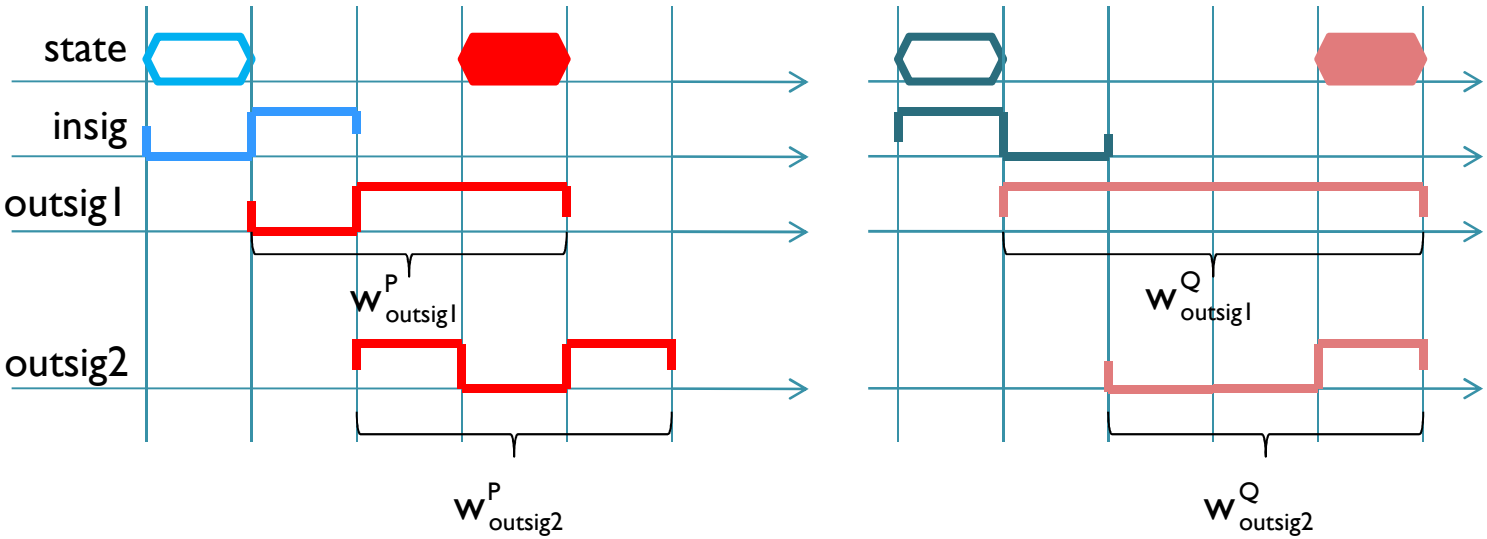
- It follows by induction that a chain of properties exists for every input trace.

Determination Check

- User specifies determination window w_{outsig} for every output signal and for every property.
- For every property P in the Operation Graph and every output signal, prove on two sets of free variables

$$I = \tilde{I} \wedge OC^{P}_{free} \wedge \widetilde{OC}^{P}_{free} \Rightarrow \forall \tau \in w^{P}_{outsig} : \mathbf{X}^{\tau}\left(outsig_{free} = \widetilde{outsig}_{free}\right)$$

- Prove that the determination windows are adjacent for every property P and every property Q that can follow P in the operation automaton.



Proof of Assertions

- Assertions capture important verification goals about all operations.
- Use complete property set as design-specific induction scheme.
- Prove assertion A by

$$IC^P \wedge OC^P \Rightarrow \underset{\tau \in W^P}{\forall} \mathbf{X}^\tau A$$

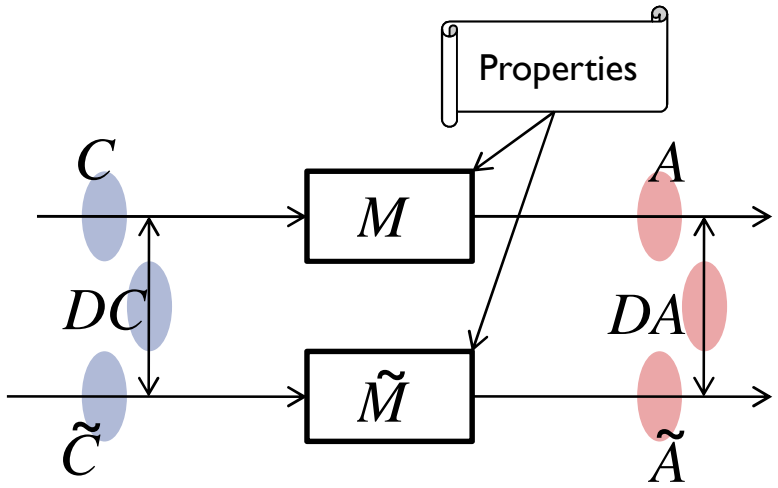
for user-defined time windows W^P that are adjacent in every property chain

Refined Completeness Criterion

- Use constraints to restrict completeness checking to practically relevant input stimuli.
- Use determination constraints (DC) and determination assumptions (DA) to specify valid data inputs or valid data outputs.
- Notation:

  `if g then determined(expr); end if;` = $((\neg g \wedge \neg \tilde{g}) \vee expr = \widetilde{expr})$

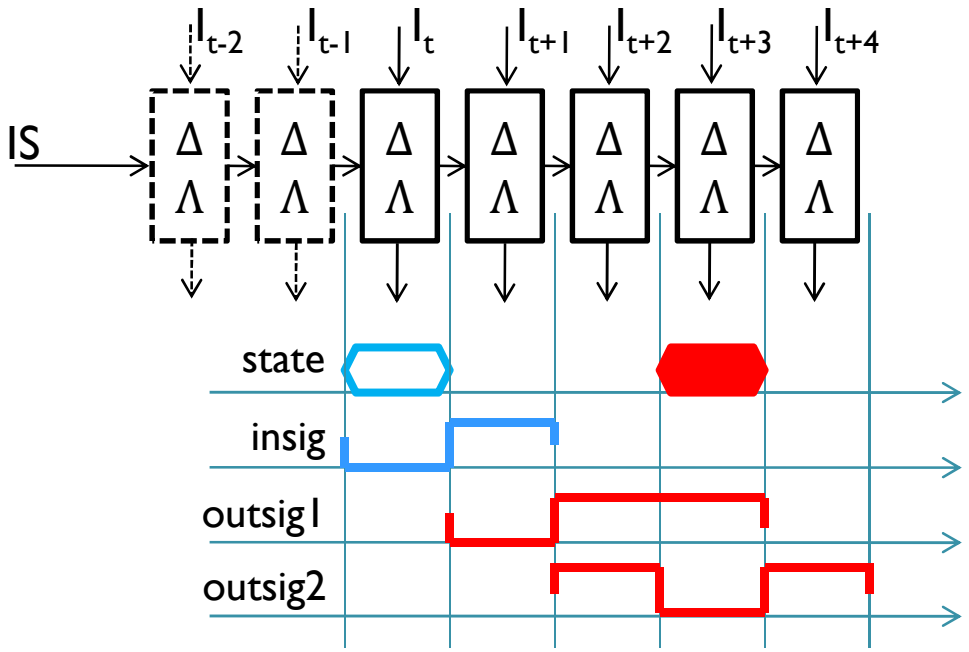
- Protocol description by constraints and determination constraints / assertions and determination assertions can describe data transport.



INTERVAL PROPERTY CHECKING

Literature: M. D. Nguyen, M. Thalmaier, M. Wedler, J. Bormann, D. Stoffel, W. Kunz: Unbounded Protocol Compliance Verification Using Interval Property Checking With Invariants", *IEEE TCAD,* Nov. 2008

„Interval Property Checking (IPC)“: Proving Operation Properties on Bounded Circuit Model



IPC handles large designs (e.g. complete processor core)

- Operation properties can be proven efficiently by SAT/SMT
- Operation properties lead to less difficult reachability problems

Characteristics of IPC

- Operation properties can be proven efficiently by SAT/SMT
  - Proof with arbitray initial state guarantees unbounded validity
  - Size of iterative circuit model depends on inspection window only
  - Conceptual start state and input conditions simplify SAT model significantly

- Operation properties lead to less difficult reachability problems
  - Reachability information related to conceptual states is intuitive and known to the designer
  - Approximate reachability computation can exploit conceptual states
  - User may safely provide additional reachability information
  - … but there are doomed circuits!

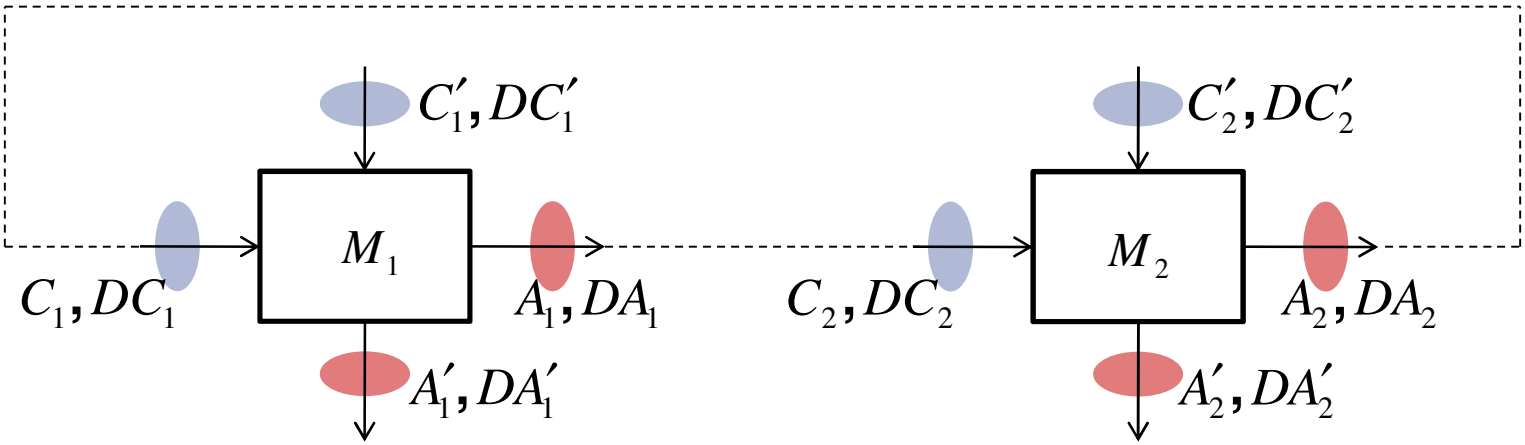
COMPOSITIONAL COMPLETENESS CHECKER

Literature: Bormann: Vollständige funktionale Verifikation (Complete functional Verification), Dissertation, University of Kaiserslautern, 2009

Problem

Given two circuits M_1 and M_2, each completely verified with

- a property set Π_1 and Π_2,
- constraints $C_i \wedge C_i'$, determination constraints $DC_i \wedge DC_i'$, $i = 1,2$
- assertions $A_i \wedge A_i'$ and determination assertions $DA_i \wedge DA_i'$

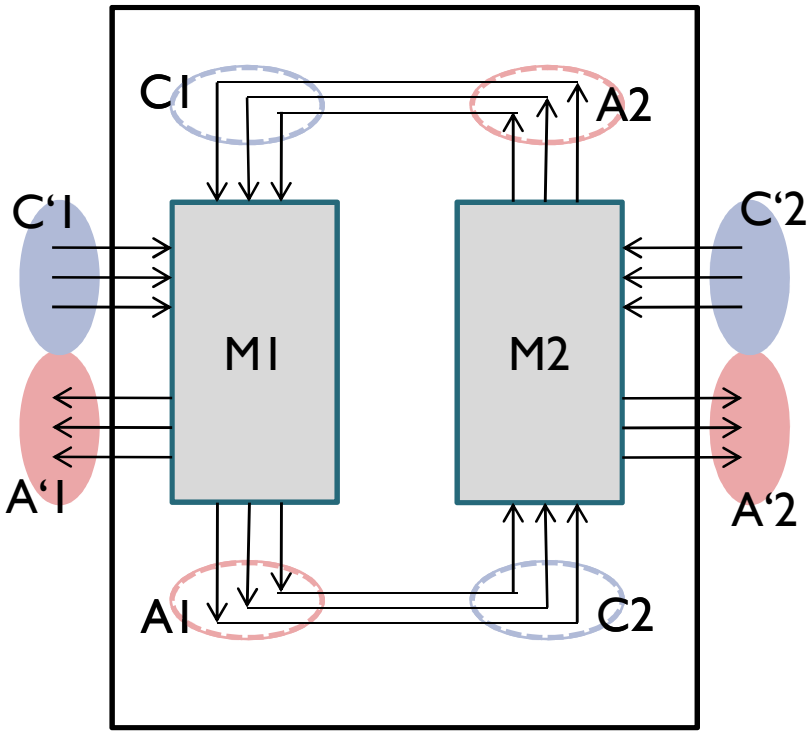


Question: When is the composite circuit completely verified by

- the property set $\Pi_1 \cup \Pi_2$
- constraints $C_1' \wedge C_2'$, determination constraints $DC_1' \wedge DC_2'$
- assertions $A_1' \wedge A_2'$, determination assertions $DA_1' \wedge DA_2'$?

Assume/Guarantee Basics

- Subcircuits verified.
  - Constraints C1, C‘1, C2, C‘2
  - Assertions A1, A‘1, A2, A‘2
- When is the composite circuit completely verified?
  - Constraints C‘1, C‘2
  - Output Assertions A‘1, A‘2
- Problem: Cyclic reasoning
- Assume-Guarantee-Theory allows cyclic reasoning
  - If constraints depend only on input signals
  - No combinatorial loops
  - A1 ⇒ C2 and A2 ⇒ C1

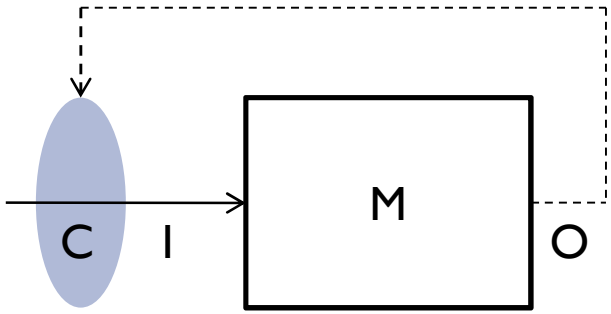


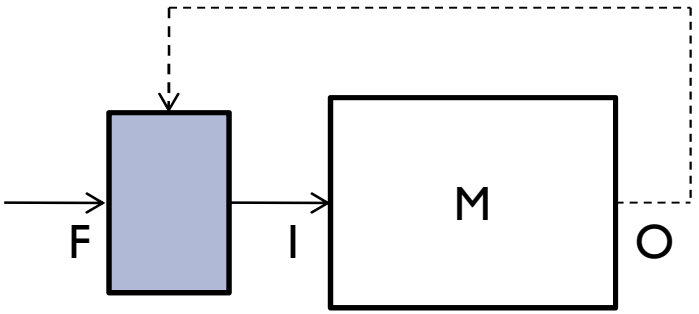
How to deal with reactive constraints (depend on outputs)?

How to deal with Completeness?

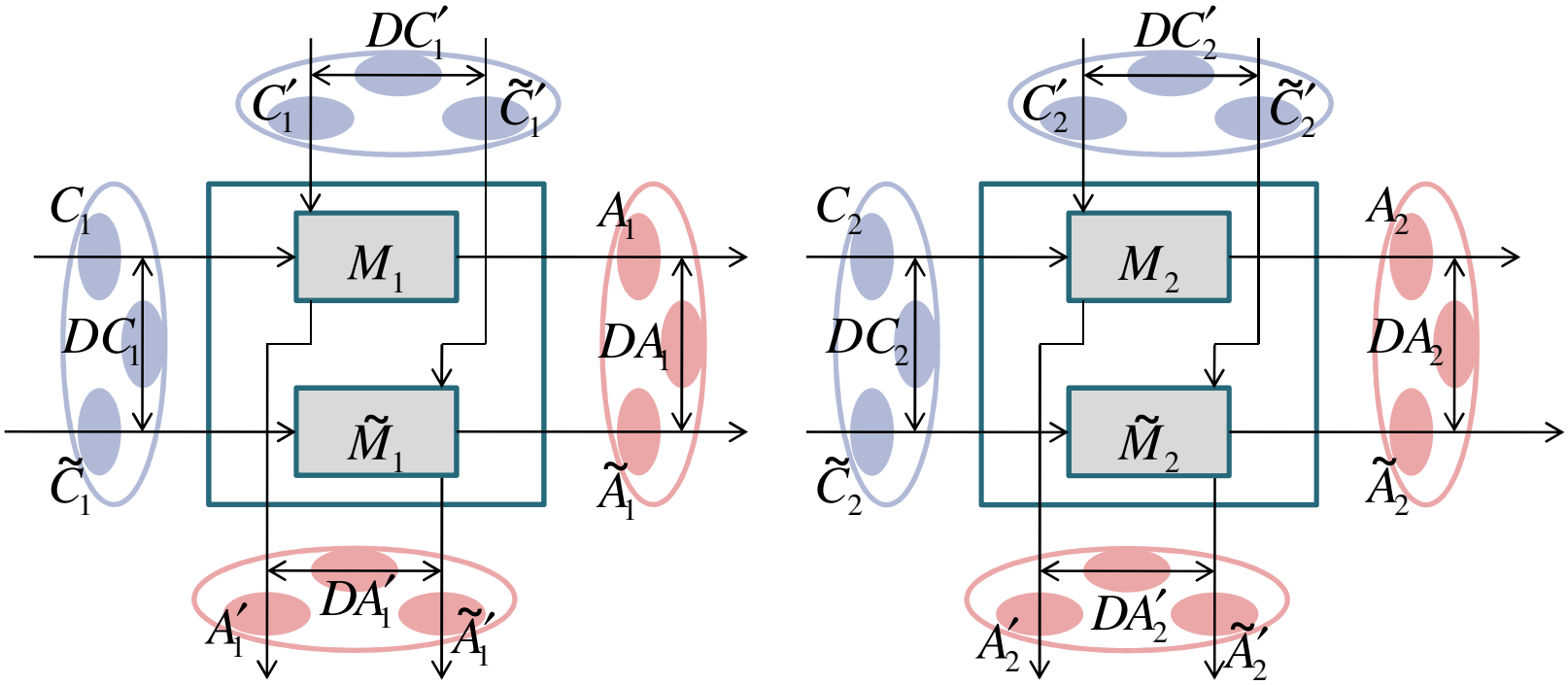
Implementable Constraints

- Many constraints restrict inputs depending on outputs.
  - e.g. protocol constraints
- Assume-Guarantee-Reasoning allows constraints that depend on outputs, but the constraints must satisfy some requirements:
  - No restriction of outputs
  - No examination of output values in the future
- Application specific characterization: Constraint must be implementable
  - Existence of reference circuit that implements the constraint
  - No combinatorial feedback loop through real and reference circuit





Compositional Complete Verification



- Reduce to the usual Assume-Guarantee-Problem
- Consequence: $C_i \wedge \tilde{C}_i \wedge DC_i$ must be implementable (i = 1,2)
- No combinatorial loops through reference circuit and M_j
- Moreover, $A_i \wedge \tilde{A}_i \wedge DA_i \Rightarrow C_i \wedge \tilde{C}_i \wedge DC_i, \; i = 1,2$

METHODOLOGY

Literature: Beyer, Bormann, Schönherr: Method for Verifying, European Patent Application, priority Feb. 2008, publication number EP2088521

Operation Properties and the Methodology

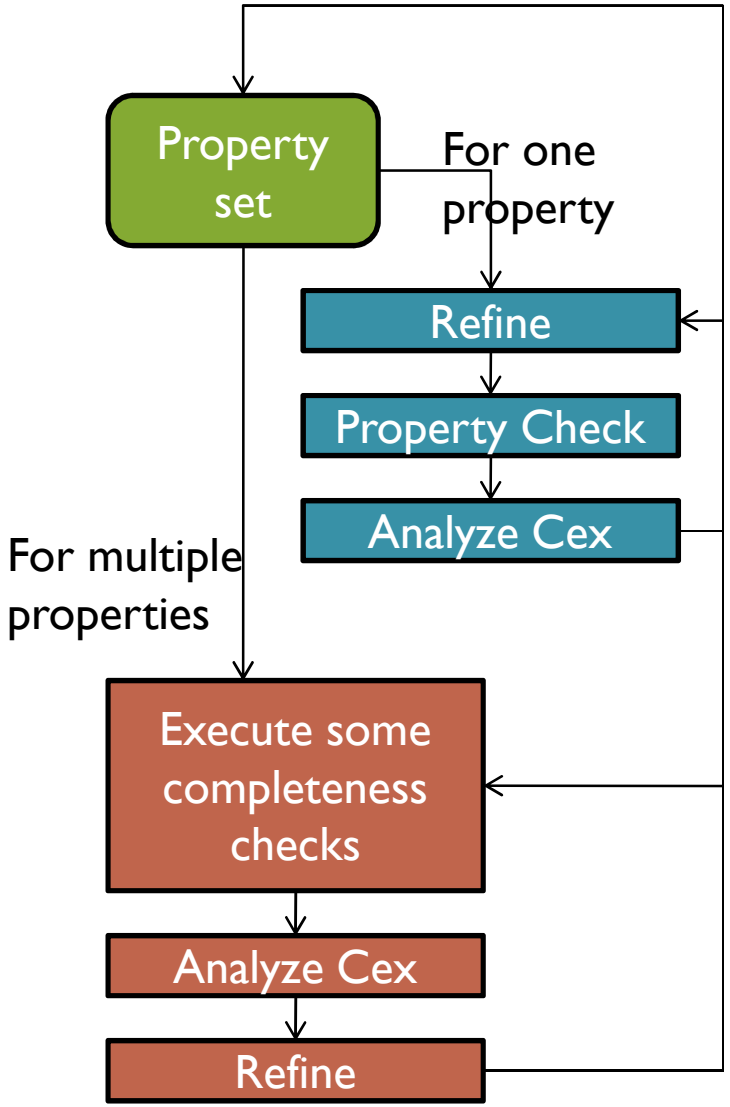
- Simple structure
  - Simple language eases formalization of intuition
  - Guidance by the 4 timed predicates
- General applicability
  - Only a focussed set of skills required
- Structuring of the verification task
  - Detailed general examination of one operation at a time
  - Counter examples show different aspects of the same mechanism
- Tool advantages
  - Proof times of 5 min. or less allow interactive use
  - Partitioned completeness check allows to obtain partial results

How a Verification Proceeds

For a representative subset of functionality

- Identify sequences of states in central controller
  - Ideas for start and end state conditions
- Identify conditions to primary inputs for these sequences
  - Input conditions
- Examine the output behavior
  - Output conditions

Use already developed properties as templates for the verification of the remaining functionality



Characteristics of the Methodology

- Follows a common process to familiarize with RTL code
- Quick validation of guesses by property checking
- Counter examples help refining guesses
- Unambiguous yet intuitive description of aspects of functionality

- Automated examination of coverage
- Provision of unexamined situations by completeness checker
- Automated non-heuristic termination criterion

- Allows reliable planning
- Verification planning is usual project planning. No need for comprehensive lists of verification goals

PRACTICAL EXPERIENCE

Literature: J. Bormann, C. Blank, K. Winkelmann, "Technical and Managerial Data About Property Checking With Complete Functional Coverage," *Euro DesignCon*, Munich 2005

Application Experience

Processors

TriCore2 (superskalar, 32 Bit)

Multithreaded network processor

IEEE floating point processor

Weakly programmable IP

Peripherals

USB master interface, Counter/Timer

UART, Interrupt Controllers, A/D Converter

Controller, Flash Card Data Port

configurable Arbiter, DMA Controller

Memory Interfaces

SDRAM Controller, SATA, Caches

Flash Memory Interface

Bus Interfaces

AHB (master IF, slave IF, bridges, multilayer)

CAN, LIN, Flex Ray, AXI, SRC Audio bus IF

Network-On-Chip

HDLC Controller

Telecom

AAL2 Termination Element

Address management in ATM Switch

Sonet / SDH Frame Alignment

Path Overhead Processing of Multi-Gigabit-Switch

DSP coprocessor ASIC for correlation computation

Quality & Cost

Bugs missed because	Simulation	Complete FV
not stimulated	yes	no
no checker	yes	no
duplication in RTL and verification code (function or constraints)	yes	yes

- Human effort: 2-4 kLOC RTL code per person month for an expert.
  - May require 2 years to become expert.
- Hardware / Software cost:
  - Simulation: Software System consisting of simulator, testbench automation tool, bus functional models, application specific software, …
  - Simulation: Occupies compute farms over weeks
  - Complete functional verification: 1 property Checker, 1 completeness checker, possibly a debugging support tool

Integration Into Industrial Processes

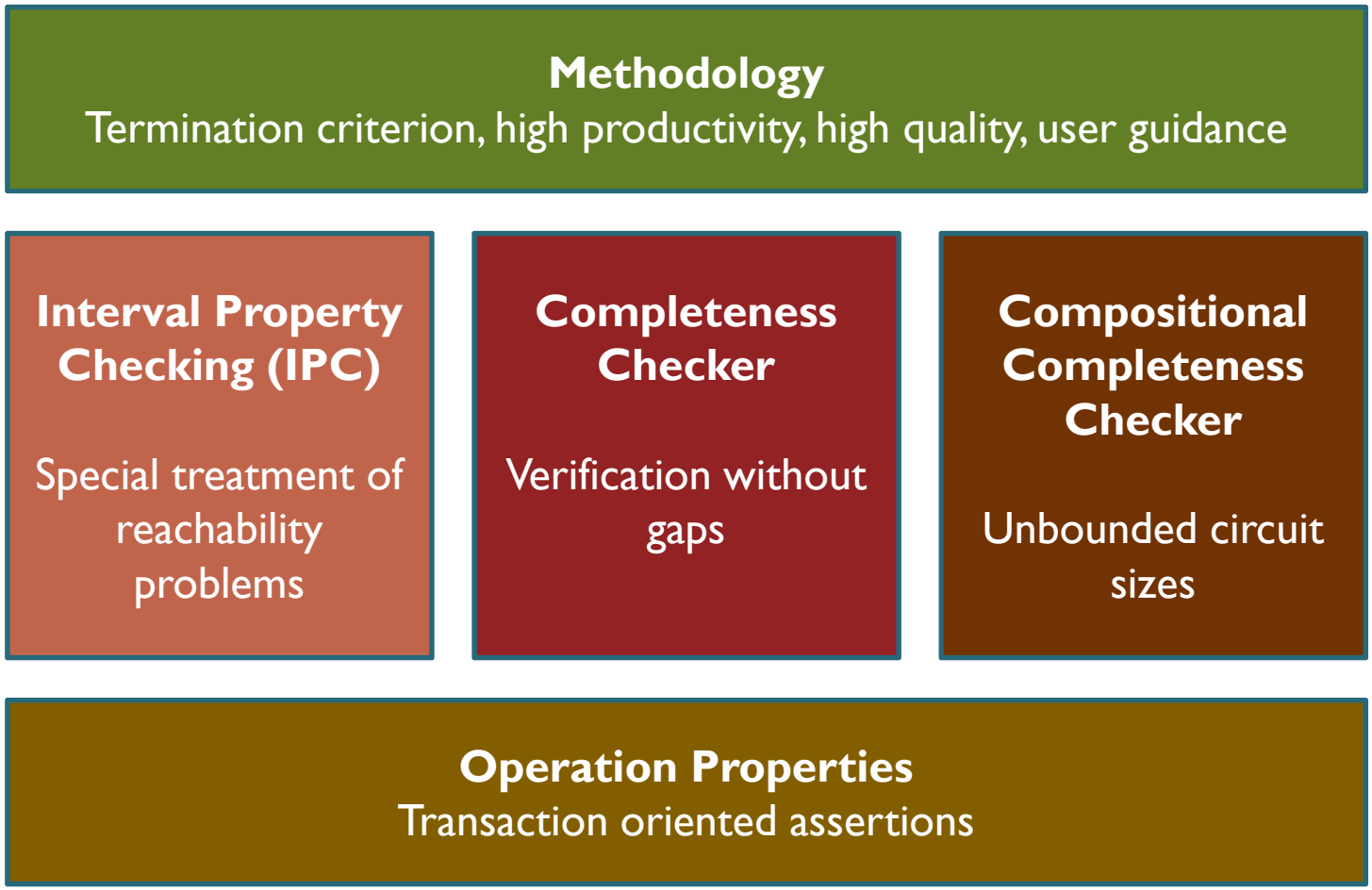
Approach provides:

- Good error localization
- Non-heuristic termination criterion – sign off documentation
- Tolerance wrt. specification quality
- Verification planning / monitoring is usual project planning / monitoring
- Integration to system level verification by checking constraints during system level simulation
- Operations provide coverage base for system verification.
- Provable Design Documentation

Approach demands:

- Provision of white box information, e.g., by designers

Summary


Methodology
Termination criterion, high productivity, high quality, user guidance
Interval Property Checking (IPC)
Special treatment of reachability problems
Completeness Checker
Verification without gaps
Compositional Completeness Checker
Unbounded circuit sizes
Operation Properties
Transaction oriented assertions