Design experience and verification methods using the latest Microsemi RTAX FPGAs

SEFUW Workshop 2014 – Noordwijk

O. Ried, T. Helfers, M. Plintovic – Airbus DS, Germany

17. September 2014

AIRBUS DEFENCE & SPACE

Agenda

- RTAX Heritage
- Applications
- Challenges
- Solution
- Formal Verification Method
  - What is Formal Verification
  - Example
  - Experience with Formal Verification

Heritage

Microsemi RTAX FPGAs are used in almost all flight programs:

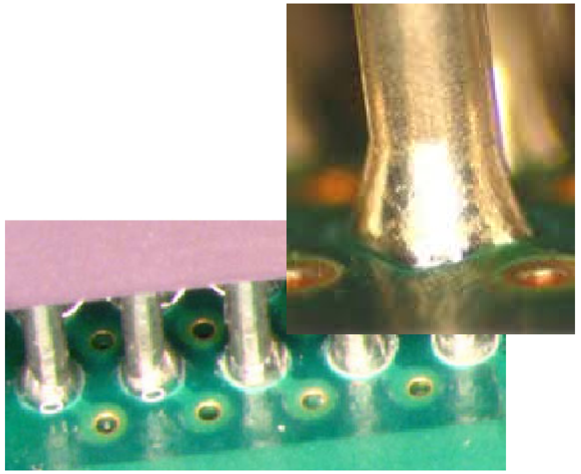
- Mass Memories
- PCDUs
- Instrument Control Units

Developments with Microsemi (Actel) FPGAs for about 20years, Usage of RTAX FPGAs for >10 years

Complex Units contain more than 20 RTAX devices

Used RTAX Devices are mainly RTAX2000, recently also RTAX4000

Used Packages are mainly CQ352, but also CG624 and recently the 1272 pin Package



High Pin Count Package CCGA Assembly and Repair Process

Applications

Functions, among others:

- Mass Memory SDRAM Controller with File Management Support, Input/output rate: (RTAX2000)
- Mass Memory SDRAM/Flash Data Recorder Controller, Input/output rate: (RTAX4000)
- Input/Output Data Formatter (RTAX2000)
- Control Interface FPGAs for providing CAN bus, MilBus, SPI, SpaceWire and customised interfaces (RTAX2000)
- Image Compression (RTAX2000)

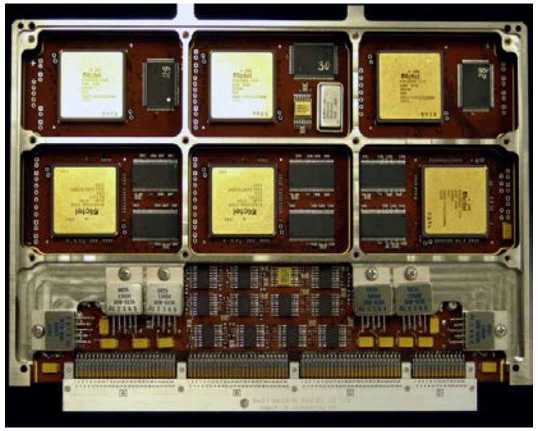
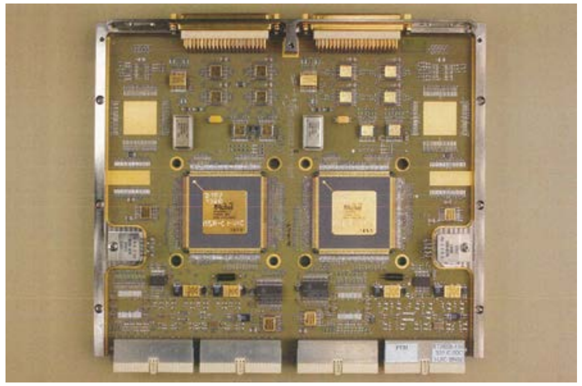
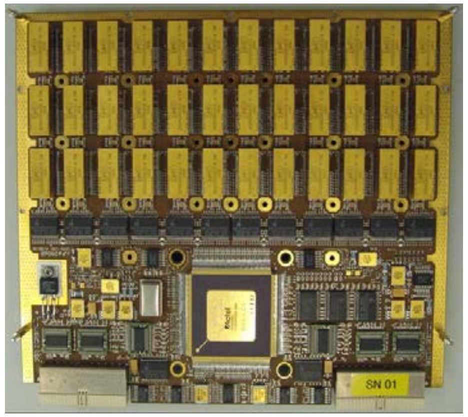


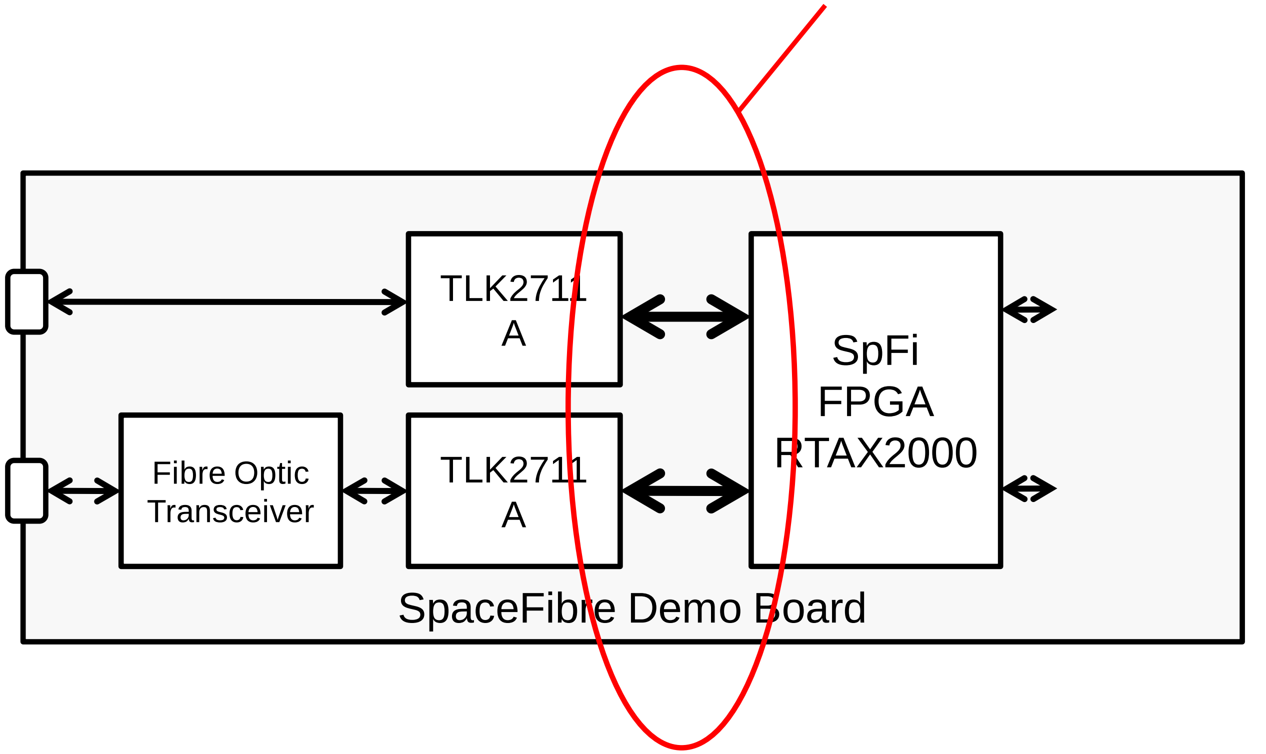
Image Compression Module


Data Formatting Module

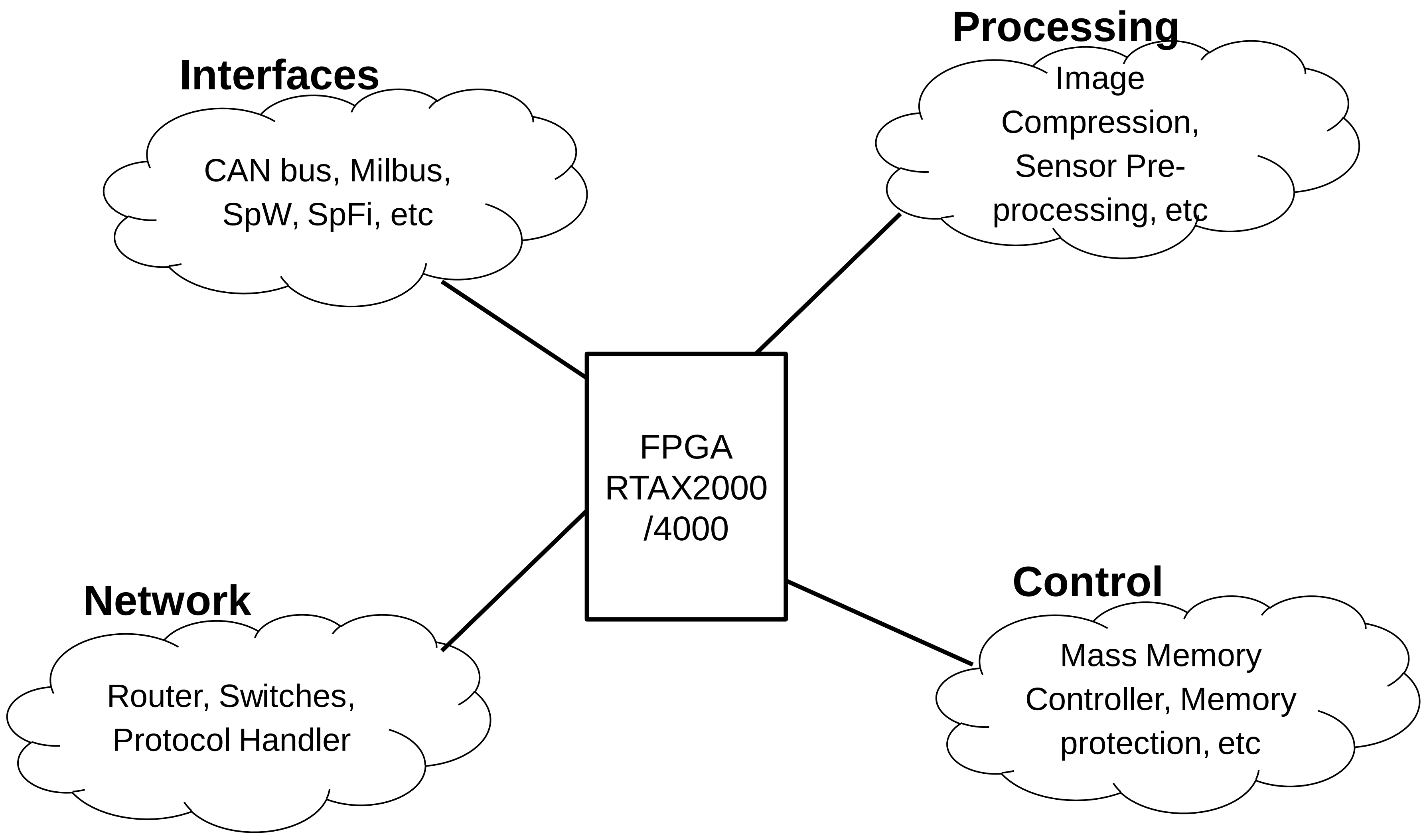

Mass Memory Module

Challenges (1): Speed and timing accuracy

High Speed Serial Link
Wizard Link Physical Layer:
Clock Frequency higher than 100 MHz



Challenges (2): Functionalities, Complexity, Pin count


Interfaces
CAN bus, Milbus, SpW, SpFi, etc
Processing
Image Compression, Sensor Pre-processing, etc
FPGA RTAX2000 /4000
Network
Router, Switches, Protocol Handler
Control
Mass Memory Controller, Memory protection, etc

Challenges (Summary)

Device complexity and complexity of required functions increases

Package pin count increases, making assembly more difficult and especially repair hardly possible

High Speed serial links are required in nowadays equipments and need to be implemented using RTAX FPGAs, which is difficult , because

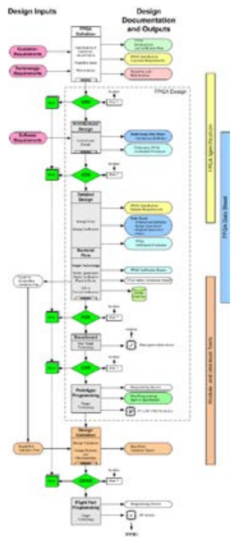
- no internal PLLs available
- device speed decreases with larger device types (e.g. RTAX4000 vs RTAX2000)

Commercial prototypes (AX) not available for RTAX4000 in the same way as for RTAX2000

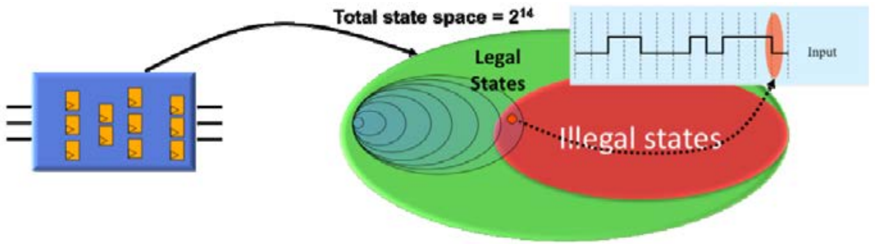
No reprogrammability -> repair necessary in case of a modification after RTAX soldering

➔ **Challenge**: Verification of FPGA design

Solutions



„ASIC similar“
Development Process

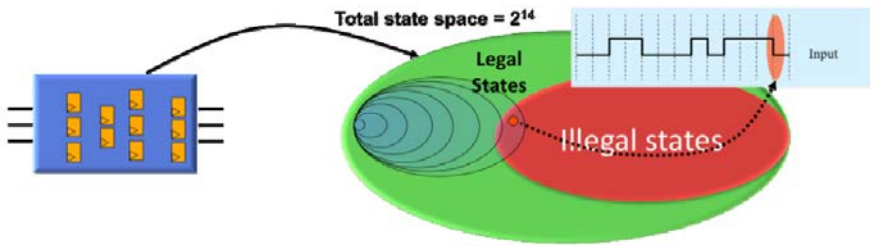


Employment of Formal Verification Methods

ChipIt Emulation System
(Synopsys)

Prototype Adapter Boards
with same footprint as RTAX FPGAs

AIRBUS DEFENCE & SPACE



Formal Verification Methods as a solution to the RTAX design challenges

Referenz: White Paper „Understanding Formal Methods for use in DO-254 Programs“
Harry Foster, David Landoll, Michelle Lange, Mentor Graphics Corporation

Understanding Formal Methods: One Analogy

Problem: $2X^2 + 8X + 5 = 0$

One Solution: Try various solutions for X;
Analogy: Simulation based tests

Alternative Solution: Calculate x;
Analogy: Formal, mathematical method

	X	Y
Test #1	0	5
Test #2	1	-1
Test #3	2	-3
Test #4	3	-1
Test #5	4	5
Test #6	5	15
…		

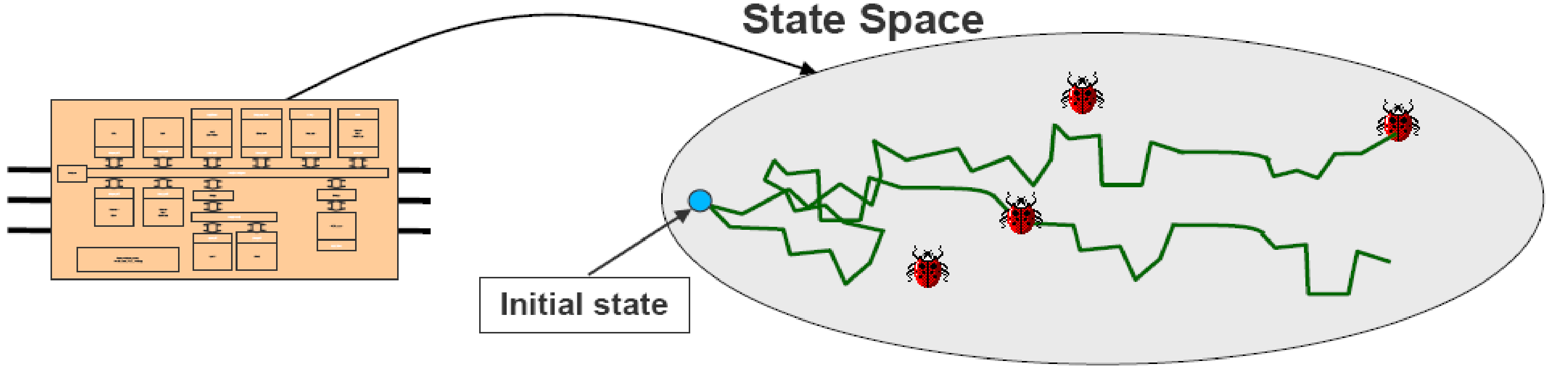
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Result: X has to be somewhere between 0 and 1 and between 3 and 4

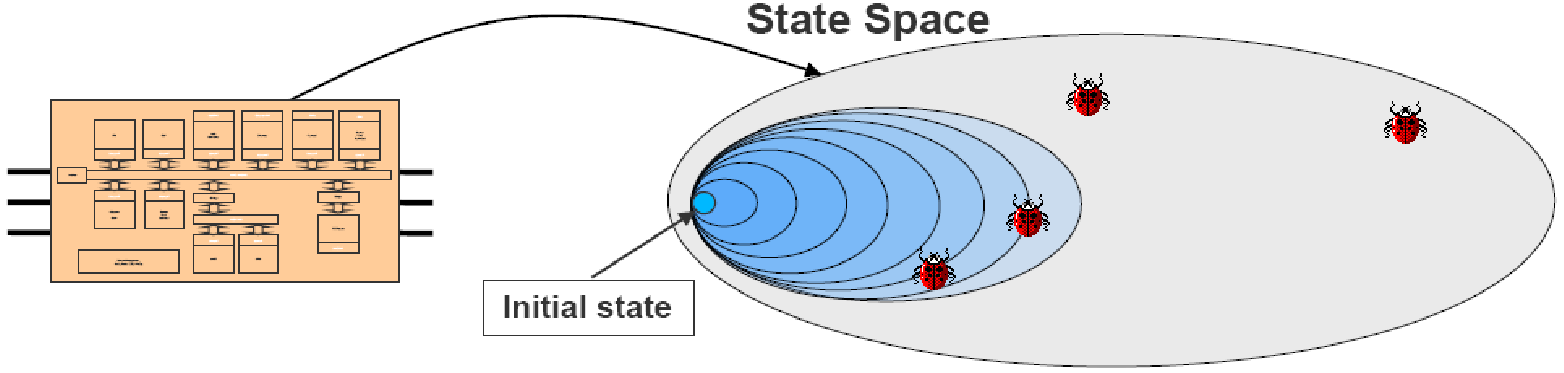
Result: X=0.775 and X=3.225

Simulation vs. Formal analysis in the digital domain

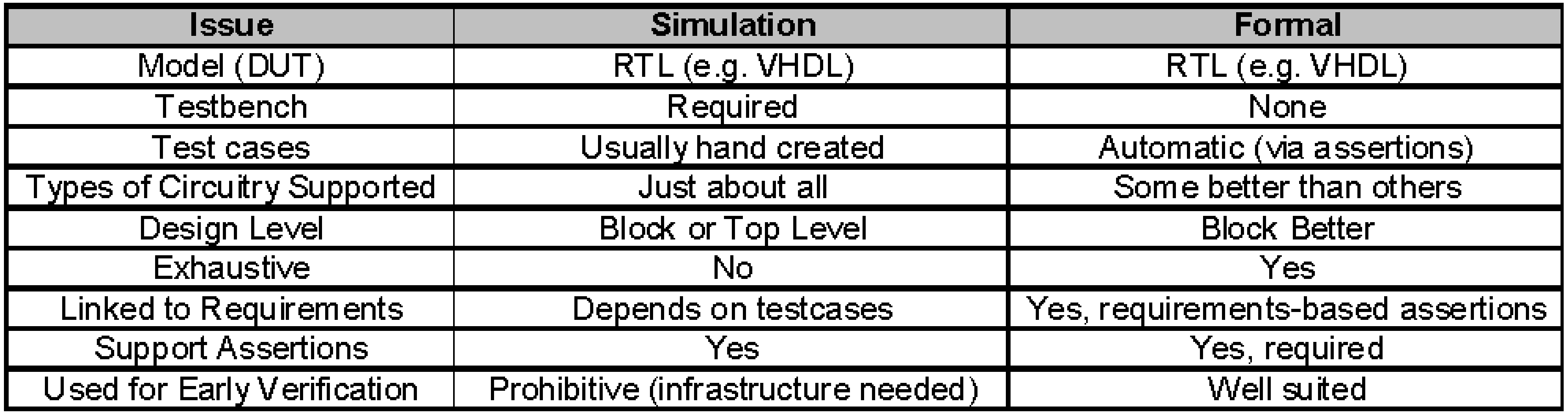
SIMULATION



FORMAL



What is the difference to simulation?

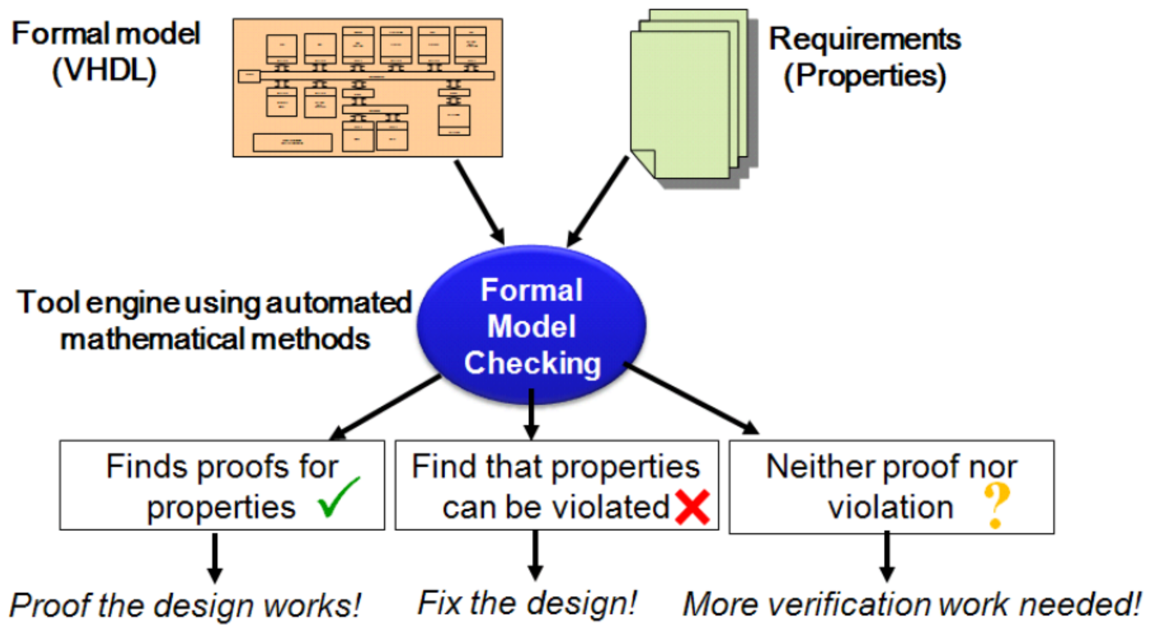


Issue	Simulation	Formal
Model (DUT)	RTL (e.g. VHDL)	RTL (e.g. VHDL)
Testbench	Required	None
Test cases	Usually hand created	Automatic (via assertions)
Types of Circuitry Supported	Just about all	Some better than others
Design Level	Block or Top Level	Block Better
Exhaustive	No	Yes
Linked to Requirements	Depends on testcases	Yes, requirements-based assertions
Support Assertions	Yes	Yes, required
Used for Early Verification	Prohibitive (infrastructure needed)	Well suited

What is needed?

Three items required to perform formal verification:

- Formal Engine: This is in fact the tool to run the mathematical analysis. In Airbus DS, Mentor Questa Formal tool suite is used.
- Design Model: This is the VHDL code, which shall be verified. Note that the source code is used as designed for the target hardware, no intermediate model to be generated as it may be required for software code formal verification
- Requirement: The requirement, normally stated in the specification, is to be written in a formal specification language, e.g. Property Specification Language (PSL, IEEE1850) or System Verilog Assertions (SVA, IEEE1800)

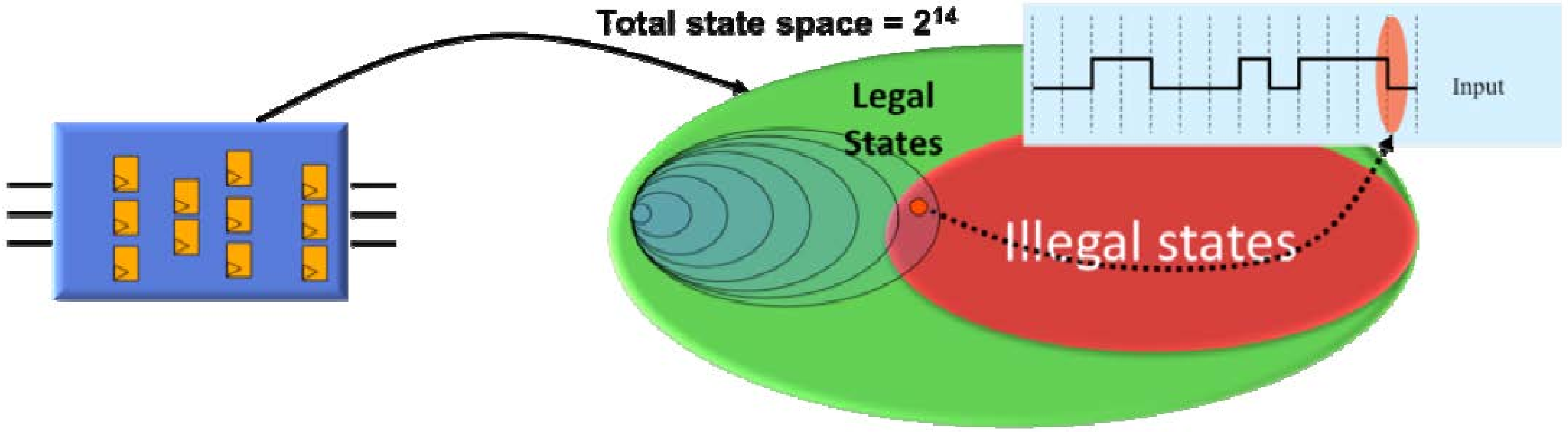

Formal model (VHDL)
Requirements (Properties)
Tool engine using automated mathematical methods
Formal Model Checking
Finds proofs for properties ✓
Find that properties can be violated ✗
Neither proof nor violation ?
Proof the design works!
Fix the design!
More verification work needed!

AIRBUS DEFENCE & SPACE

What is the output of the Formal Engine?

There are three different results from Formal Verification

- **Proof**: A proof provides evidence that exhaustive analysis reveals that a model will always operate according to the requirement (no exceptions).
- **Counter Example**: The formal engine outputs a counter example if it has found an illegal state in respect to the specified property
- **Inconclusive**: This situation indicates that given the current conditions, a tool is unable to come up with one of the previous options. The property might need to be adapted or the amount of circuitry to be analysed need to be de-scoped.

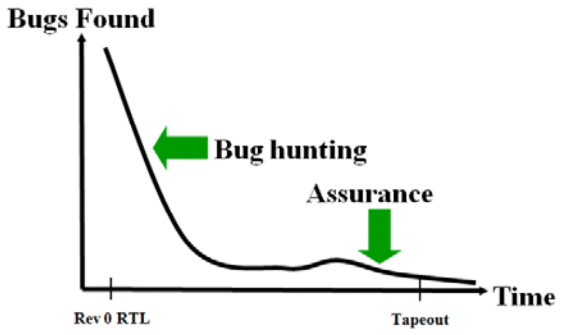


AIRBUS DEFENCE & SPACE

When should it be used?

Three areas of use have been established

- Enhanced verification on block level at an early stage of the design process to deliver robust, pre-verified blocks for system integration and reuse
- Searching for bugs (i.e bug hunting)
- Exhaustively prove that no bug exist (i.e. design assurance)



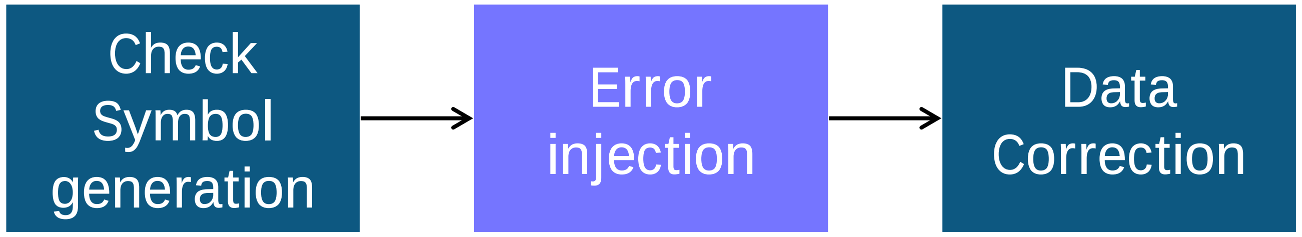
Formal Verification does generally not take the place of simulation, but should rather be used alongside of it.

When/Where to Use	When/Where to Avoid
Control or datapath circuitry with high concurrency (and no data transformation)	**Datapaths with data transformations**
Arbiters, On-chip bus bridges	Floating Point Units
Power management units	MPEG decoder
DMA, Interrupt, memory controllers	Convolution unit in DSP
Bus interfaces, Schedulers, Standard Interfaces, Protocol handler	Graphics shading unit

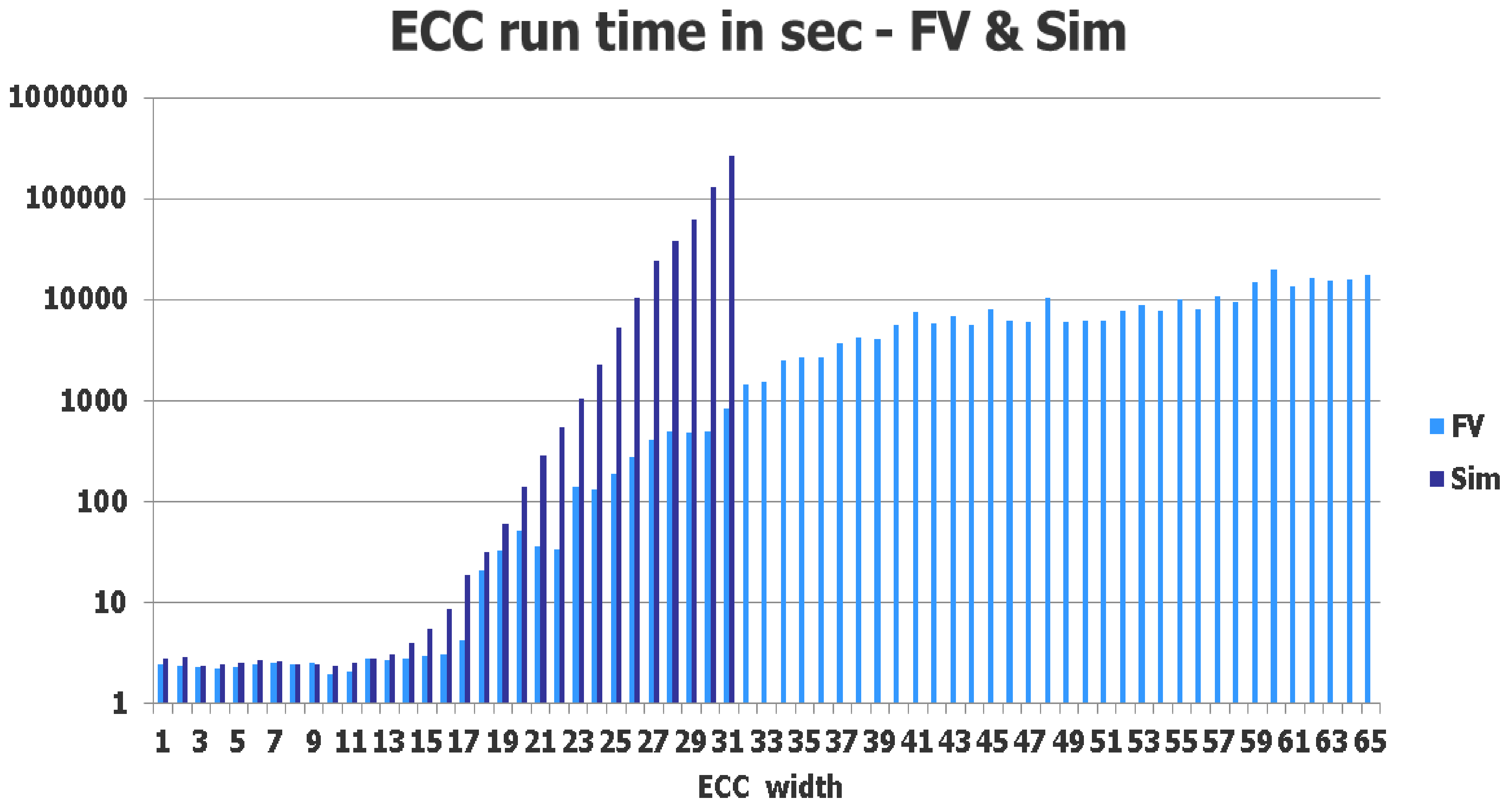
Formal Verification & Simulation Comparison

Exhaustive verification of an EDAC circuit, capable of correcting one bit
Simulation uses brute force method
FV properties are described by one assertion and one assumption:

```
-- psl default clock is rising_edge(clk_sys);
-- psl property p_check_val is always dataout = prev(datain) abort not reset_n;
-- psl assert p_check_val;
-- psl assume always onehot0(err_inj);
```



Comparison result



Other Formal Methods

- Equivalence checking
- Automatic design checks

  → Basic design checks e.g.:

  Latch inferred

  Register has no reset

  Illegal index value

  FSM deadlock

  … and many others

- Clock Domain Crossing (CDC) check

  → Several synchronization schemas could be proofen automatically.

  → Unknown synchronization schemas are detected

➔ These methods are quick to setup and to execute.

➔ Equivalence, Automatic design and CDC checks are now mandatory for all designs.

Experience with Formal Verification at Airbus DS

Resulting Benefits

- Formal Verification has been employed on multiple design blocks of two FPGAs
- Both of these FPGA designs were bug free at delivery to the system level integration team

Challenges:

- Learning curve for assertion languages and formal methodology.
  - To enable formal verification, designers must write assertions, constraints and cover properties to describe the design intent.
  - The designers also have to become accustomed to using formal verification tools and methodology
- Formal Verification is not the best solution any type of design. It has to be analyzed whether Formal Verification or timing simulation is the best solution for a block.
- Formal Tools are expensive

Outlook

- Broader deployment of the formal verification methodology is envisaged.
- Guideline for writing assertions and a pool of standard assertions for common design blocks or interfaces

AIRBUS DEFENCE & SPACE

Thank You!