SAND2011-5706P

Exploring Formal Verification Methodology for FPGA-based Digital Systems

LDRD
LABORATORY DIRECTED RESEARCH & DEVELOPMENT

Early Career R&D Program

Sandia National Laboratories

Yalin Hu (Org 8135)
Livermore, CA

Problem

- How to formally verify a FPGA-based digital system (Figure 1)?
- What are the specific requirements that are different than industrial applications?
- How confident are we after the verification?
- How to automate the verification with respect to specification?
- What is the best design/verification flow (Figure 2)?

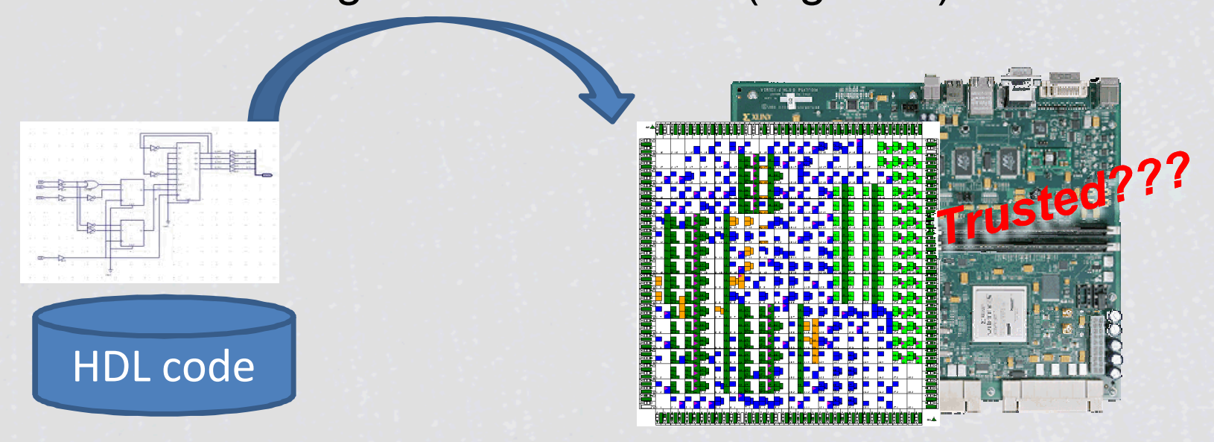


Figure 1. How to deliver trusted FPGA-based designs?

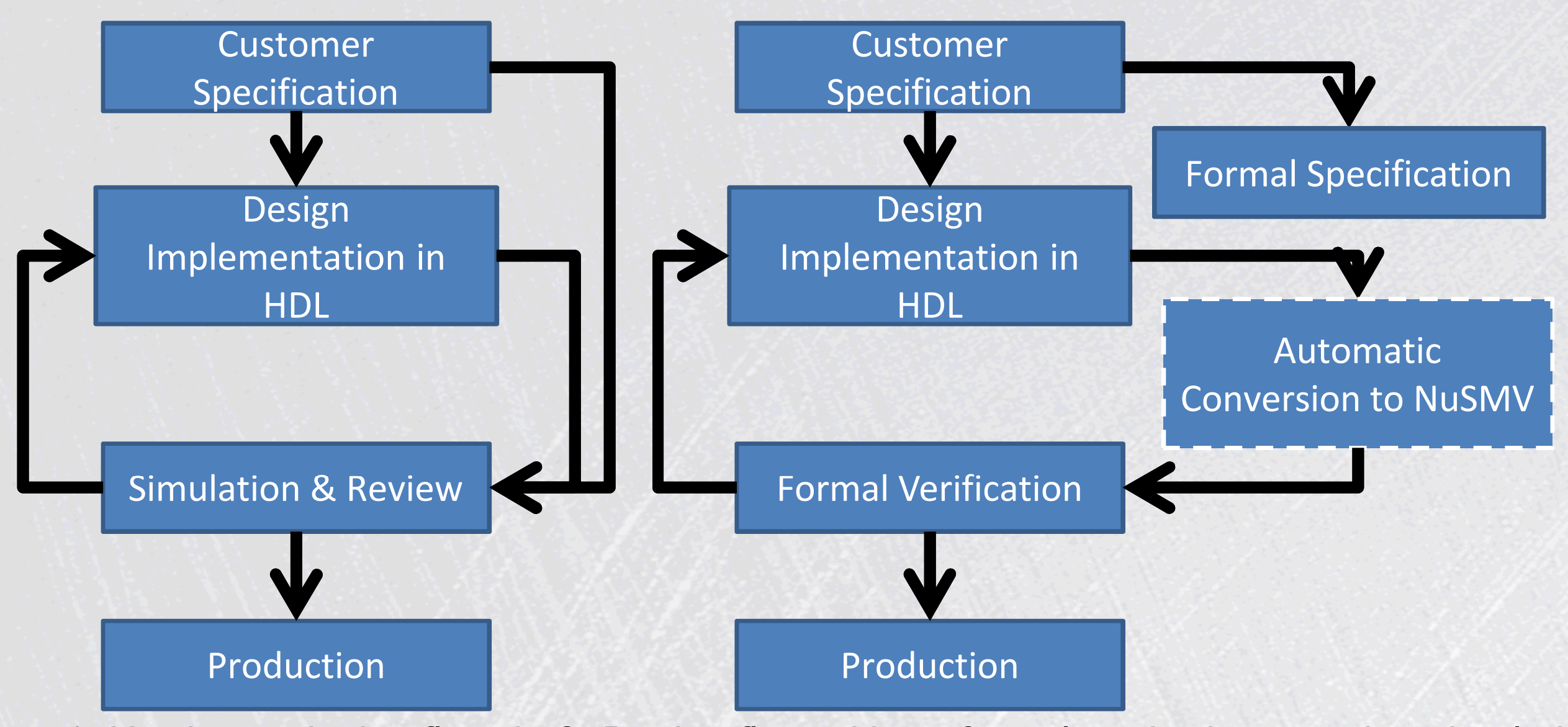


Figure 1. Hardware design flow. Left: Design flow without formal methods – passing simulation proves correctness. Right: Design flow augmented with formal verification – passing verification ensures that the formal specification is held.

Formal Verification – is the process of checking that the *intent* of the design was preserved in its *implementation*.

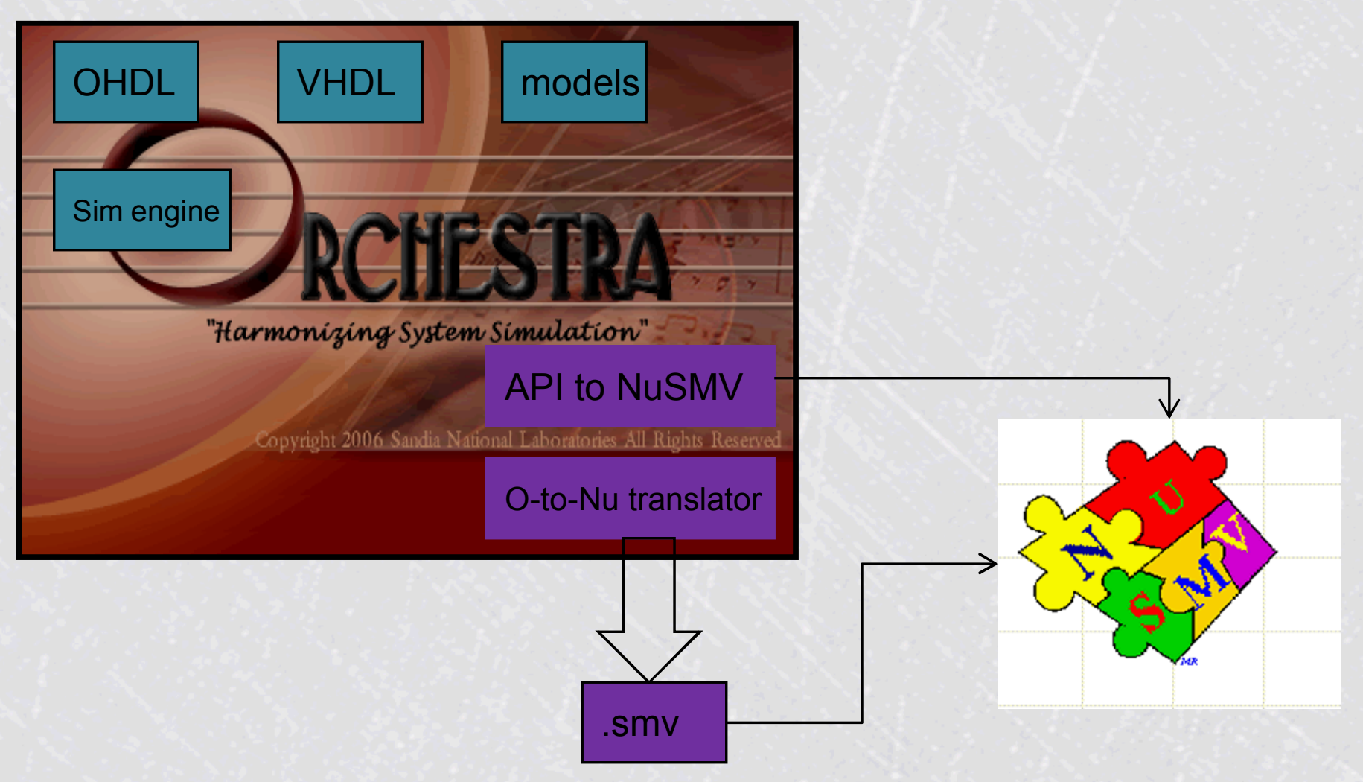
Approach

Investigation of various formal approaches

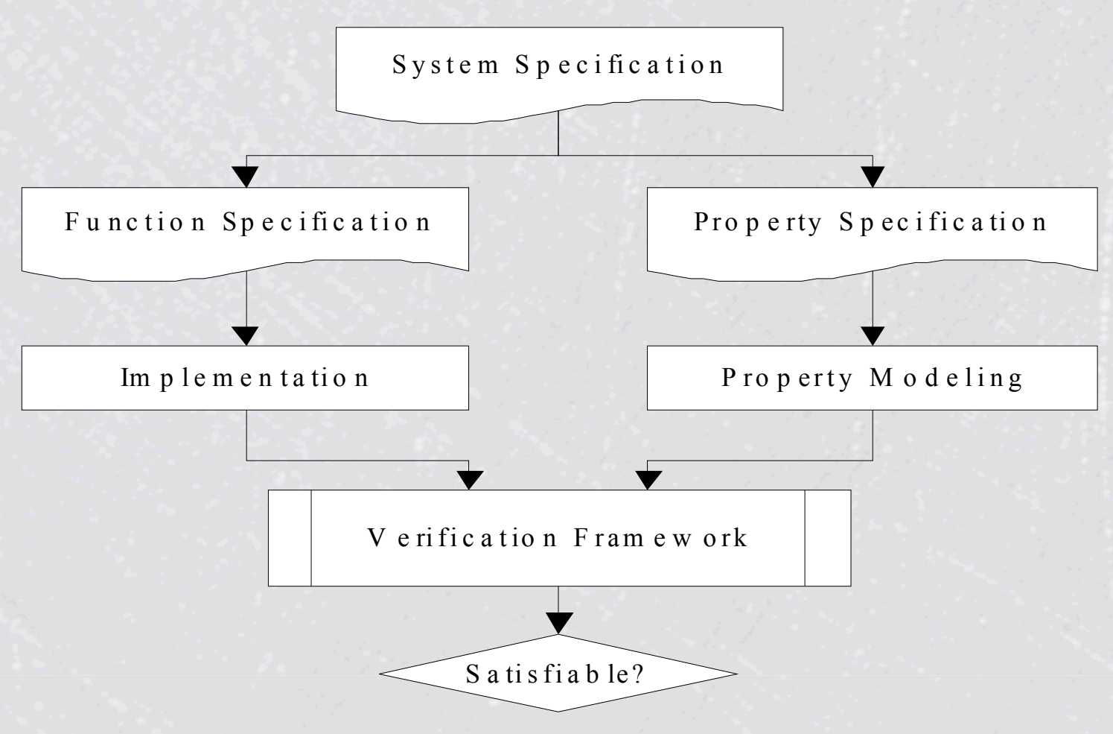
- Equivalence checking -- whether two representations of a design is equivalent
- Model checking -- whether a modeled design meets specification
- Symbolic model checking -- build propositional logic instead of graph for FSM
- Automated theorem proving -- proving of mathematical theorems with computer programs
- Integrated approaches

Integration of event-driven simulator with model checker

- What is the appropriate abstraction level?
- What is the appropriate application programming interface (API)?


OHDL
VHDL
models
Sim engine
ORCHESTRA
"Harmonizing System Simulation"
API to NuSMV
Copyright 2006 Sandia National Laboratories All Rights Reserved
O-to-Nu translator
.smv
NuSMV

Introduction of the concept of "systems are designed to be verified and are verified by design"



Results

Potential development framework has been identified
Potential application has been identified
Several prototype models have been developed for NuSMV

- Simple ALU
- Shift register
- Single port random access memory (RAM)
- Open source Advanced Encryption System (AES)

Proved challenge with existing model checking tools for designs involving memory

- Small sized RAM experienced state explosion problem during model checking with NuSMV (Figure 3)
- An implementation of AES (computational intensive) is too complex to model efficiently with NuSMV

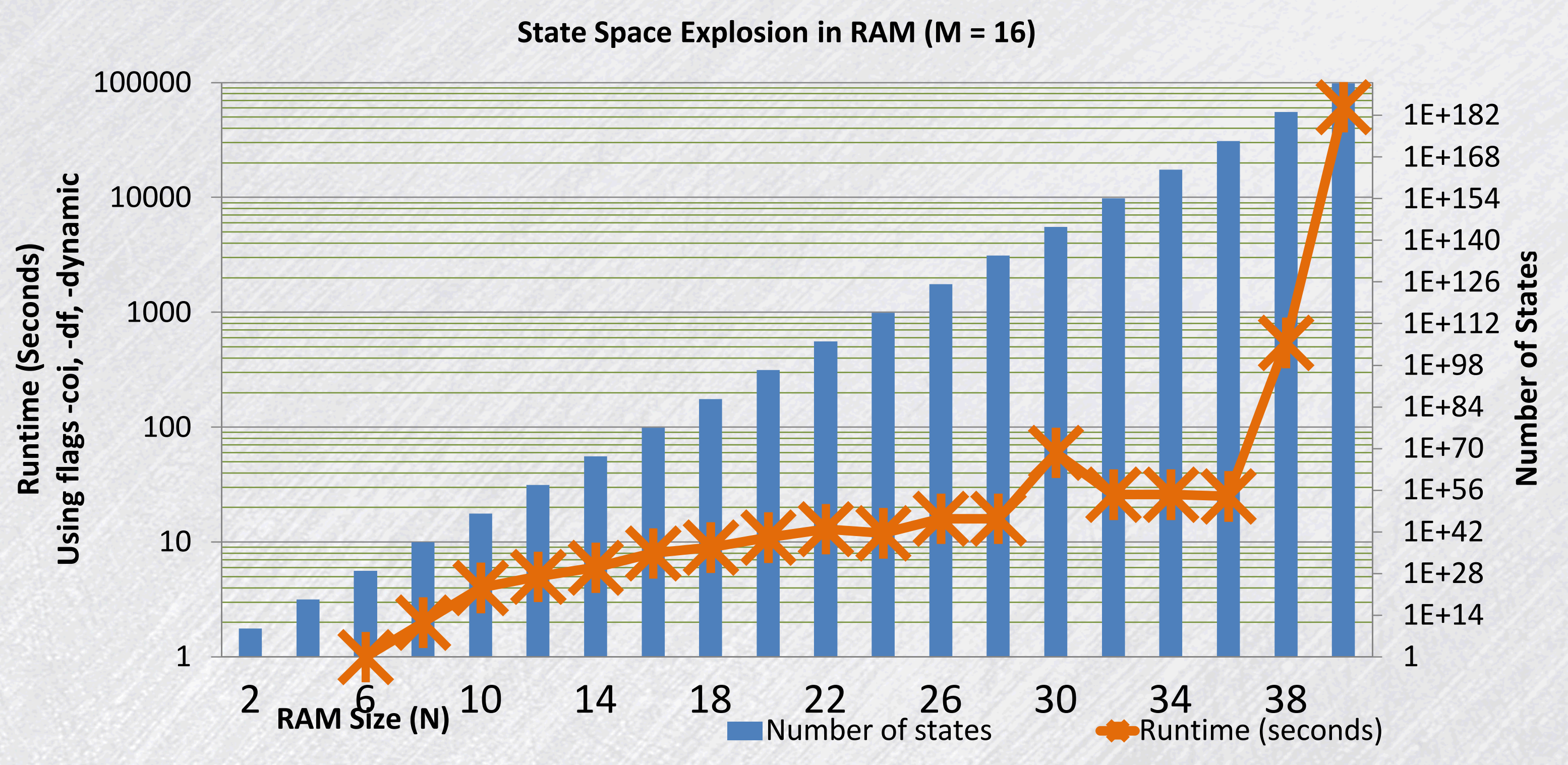


Figure 3. State space explosion for RAM with fixed word size (M) and variable memory size (N).

Identified the need for a hybrid approach to compensate for the limitations of model checking and theorem proving

Significance

Bring new S&T into mission-critical, high-consequence digital system design/verification to compliment the traditional simulation based testing

- Simulation: dynamically demonstrate that the design intent is preserved in implementation → proves correctness
  - Fact: *potentially* identify the presence of a bug
  - Challenge: does not ensure the absence of a bug
- Formal verification: statically proves that the implementation satisfies the requirements → catches fault
  - Fact: exhaustive explore all state space to uncover all incorrect behaviors
  - Challenge: identify *enough* properties to check

Distinguish Sandia's specific needs from industrial applications, which can be handled with commercial tools – how many "9"s will be "good enough"?

- High-reliability → high functional correctness
- High-availability → low system downtime

Availability (%)	Downtime		
	Weekly	Monthly	Annually
90% "one 9"	16.8 hours	72 hours	36.5 days
99% "two 9s"	16.8 hours	7.2 hours	3.65 days
99.9% "three 9s"	10.1 minutes	43.2 minutes	8.76 hours
99.99% "four 9s"	1.01 minutes	4.32 minutes	52.56 minutes
99.999% "five 9s"	6.05 seconds	25.9 seconds	5.256 minutes
99.9999% "six 9s"	0.605 seconds	2.59 seconds	31.5 seconds

Leverage this S&T in an improved design/verification flow

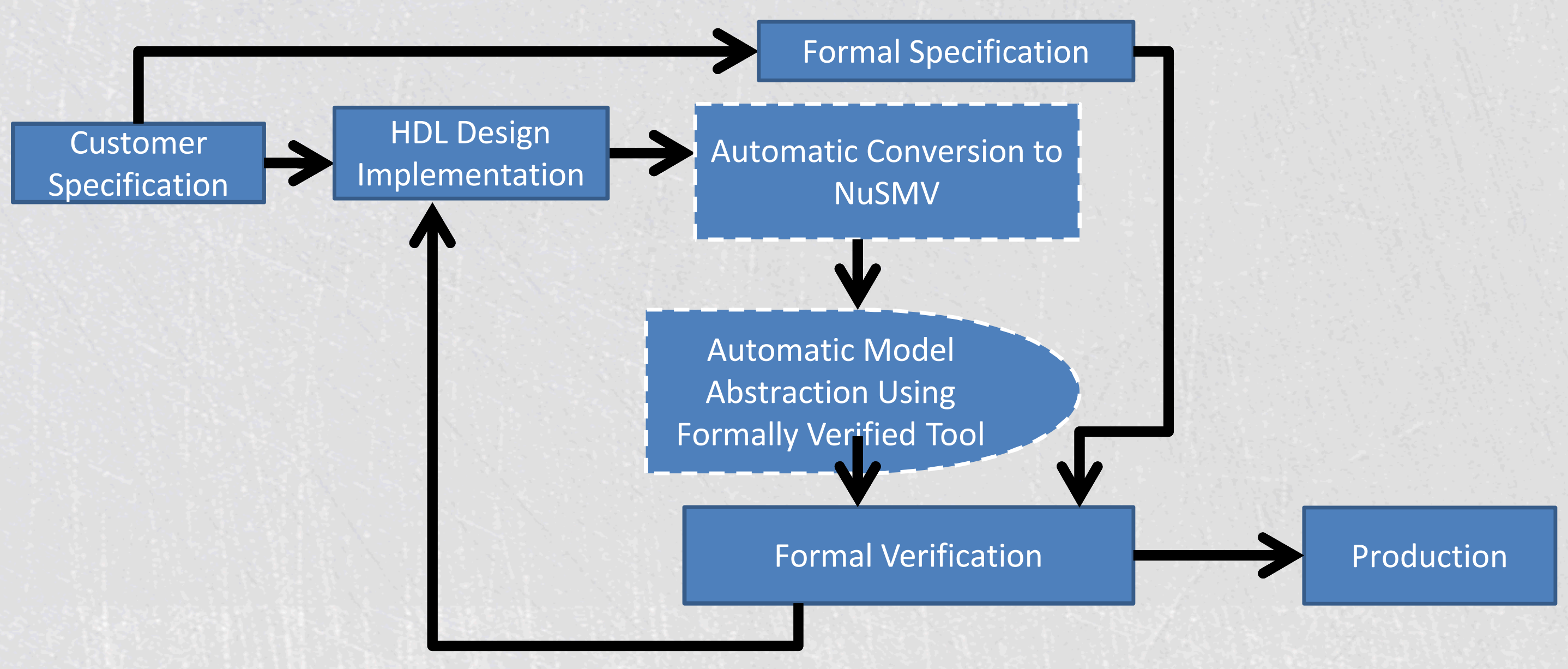

Customer Specification
HDL Design Implementation
Formal Specification
Automatic Conversion to NuSMV
Automatic Model Abstraction Using Formally Verified Tool
Formal Verification
Production

Figure 4. Improved hardware design flow combines model checking and theorem proving