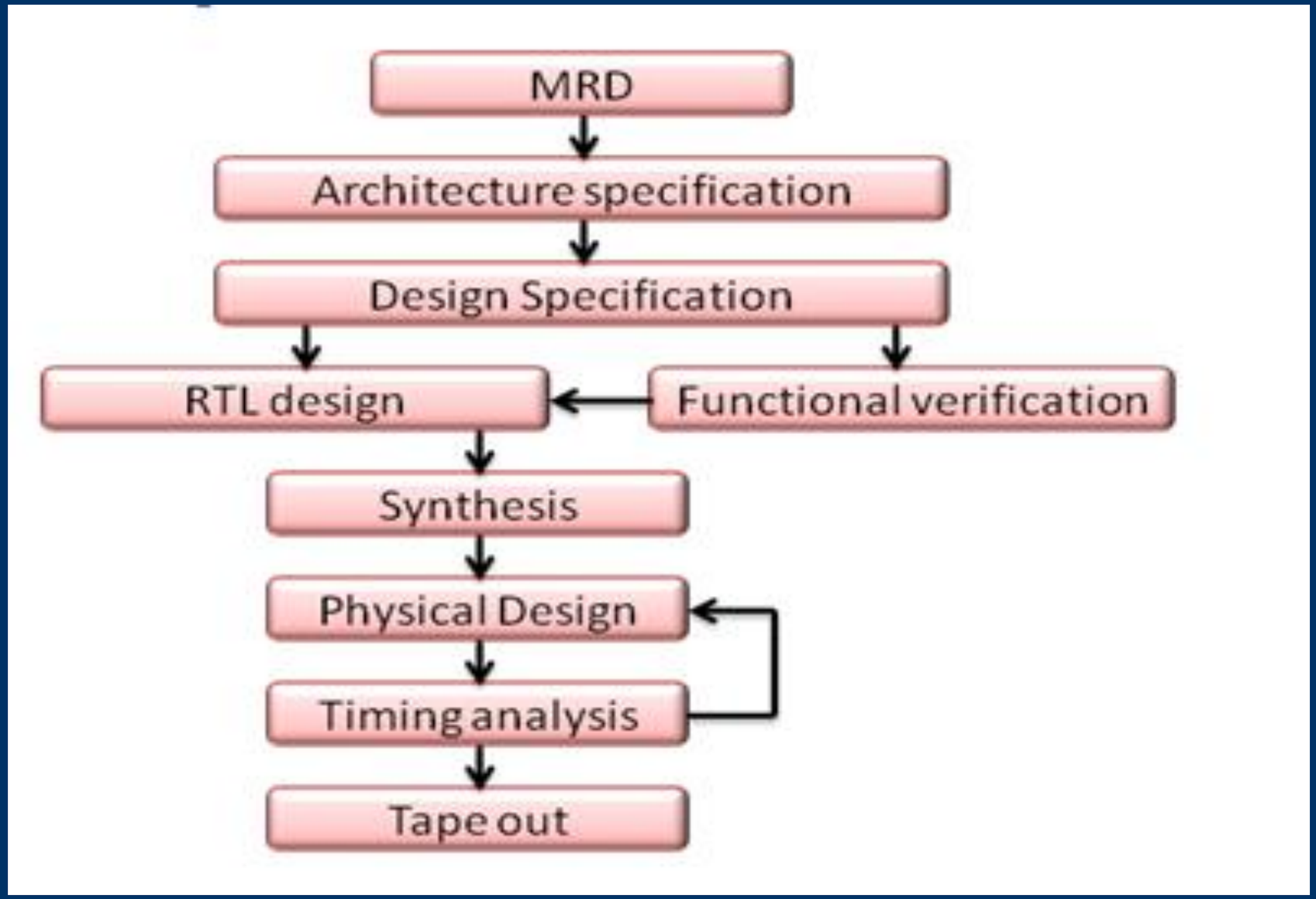
# *VERIFICATION*

# Contents

- Introduction
- Introduction to Verification
  - Describe Verification
- Verification Approaches
  - Identify differences between
    - Verification approaches
    - Testing and verification

# ASIC Design Flow



# *Familiar Bugs!*

- Pentium FDIV bug
  - Affecting the floating-point unit
- Therac-25
  - Computer-controlled radiation therapy machine

# *Verification*

- Verification is a process used to demonstrate that the intent of a design is preserved in its implementation.
  - Verification is not a testbench, nor is it a series of testbenches.

# *Verification Challenges*

Reference

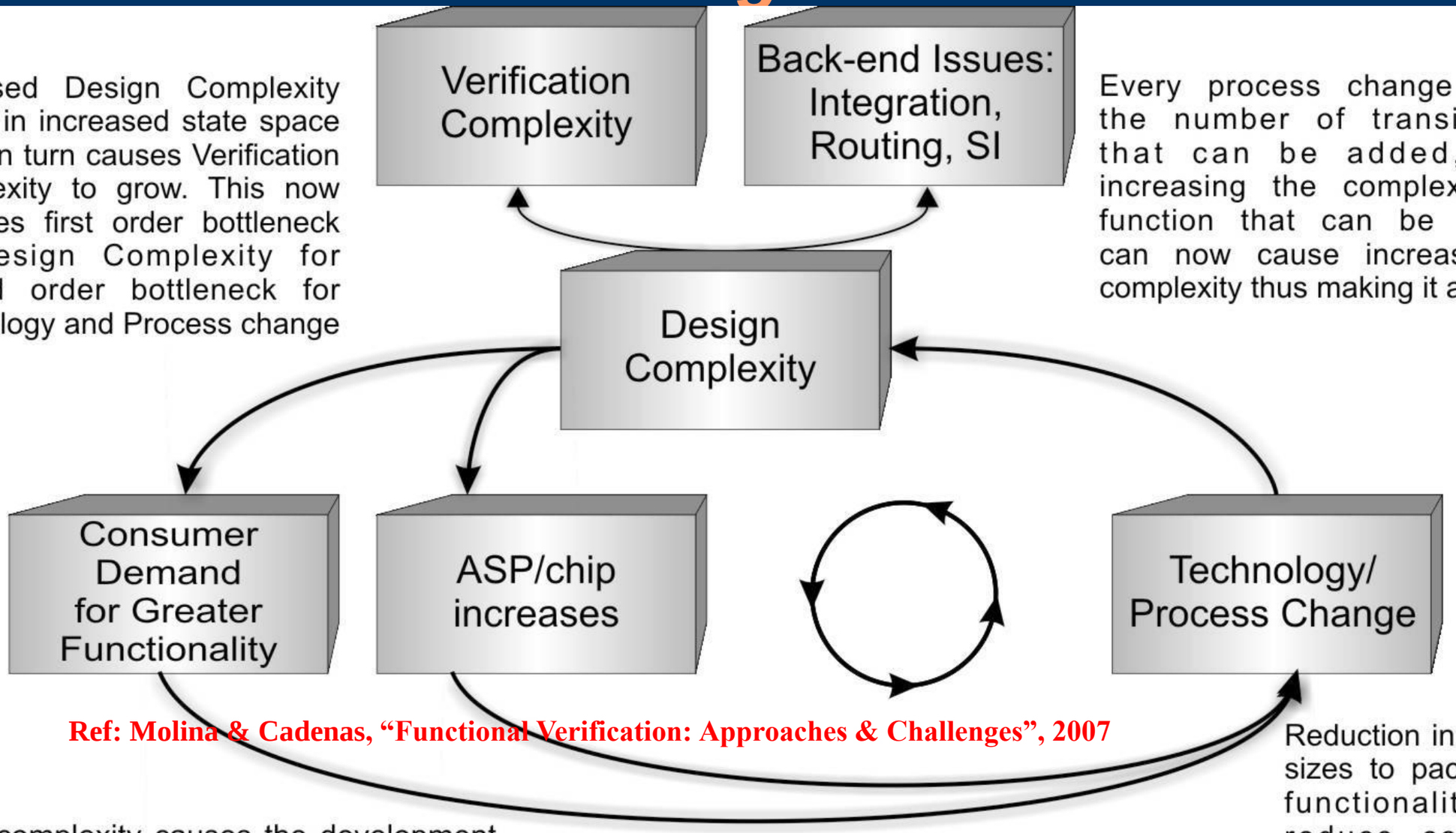
- The Profession of Verification by Paul Wilcox
- Writing Testbenches using System Verilog by Janick Bergeron

# *Verification Challenges*

- Every Development team faces Verification issues due to
  a. Size or number of designs
  b. Complexity of the design
  c. Verification Process being used
- Verification teams continually attempt to address these issues to find new problems that may arise.
- These new problems can be more complex than original ones

# Verification Challenges

Increased Design Complexity results in increased state space which in turn causes Verification Complexity to grow. This now becomes first order bottleneck for Design Complexity for second order bottleneck for Technology and Process change



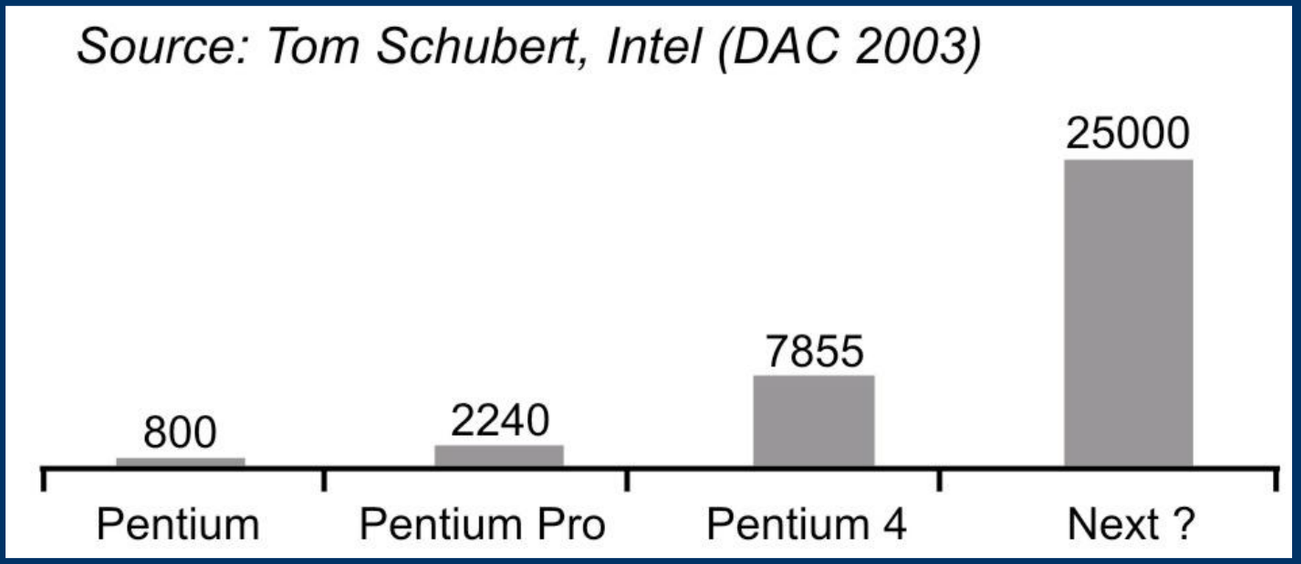

Every process change increases the number of transistors/area that can be added, thereby increasing the complexity of the function that can be built. This can now cause increased design complexity thus making it a bottleneck.

**Ref: Molina & Cadenas, "Functional Verification: Approaches & Challenges", 2007**

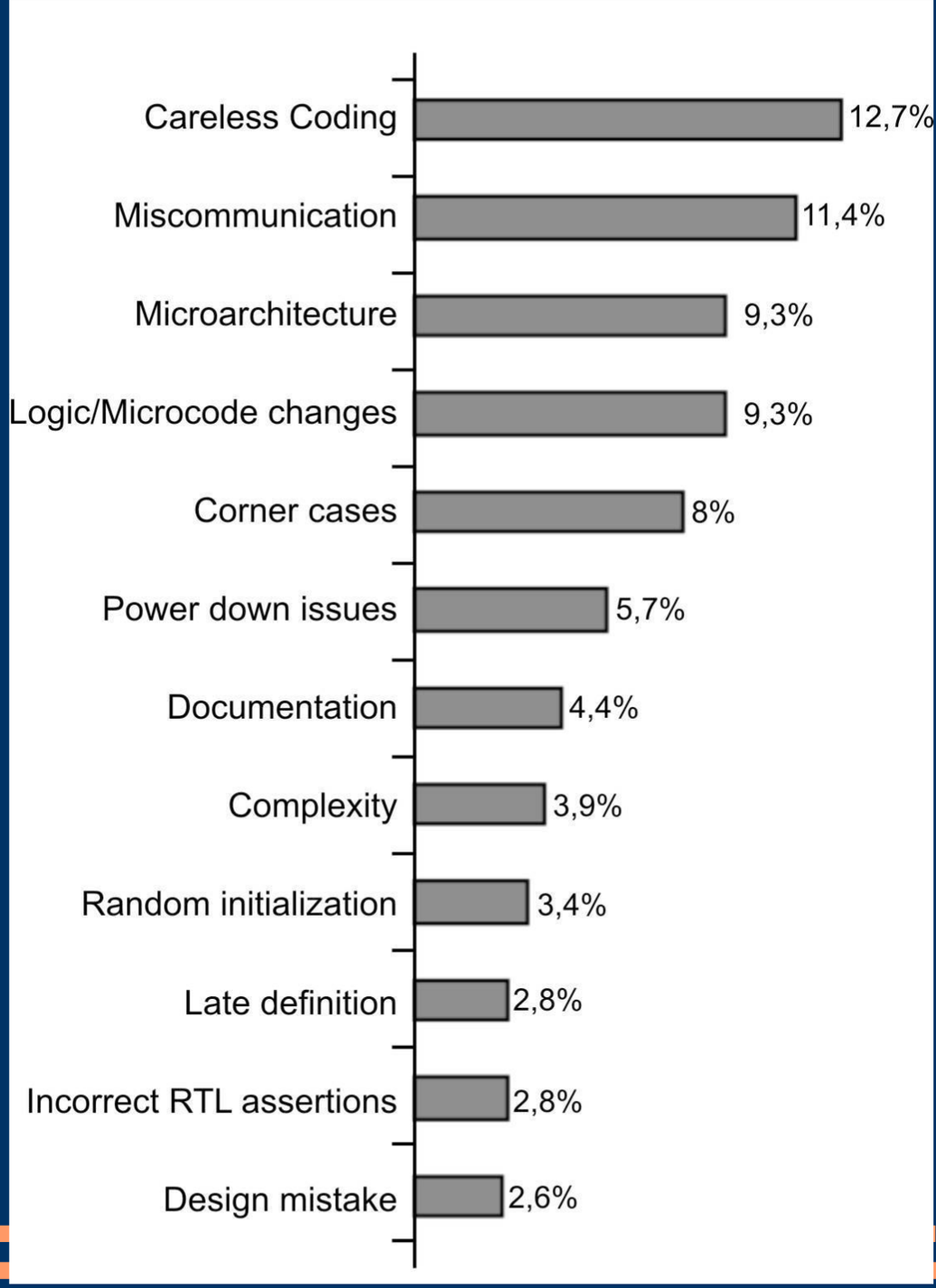
Increased design complexity causes the development cost to go up and effect yield and die areas. In addition, they fuel the desire for more functionality by consumers

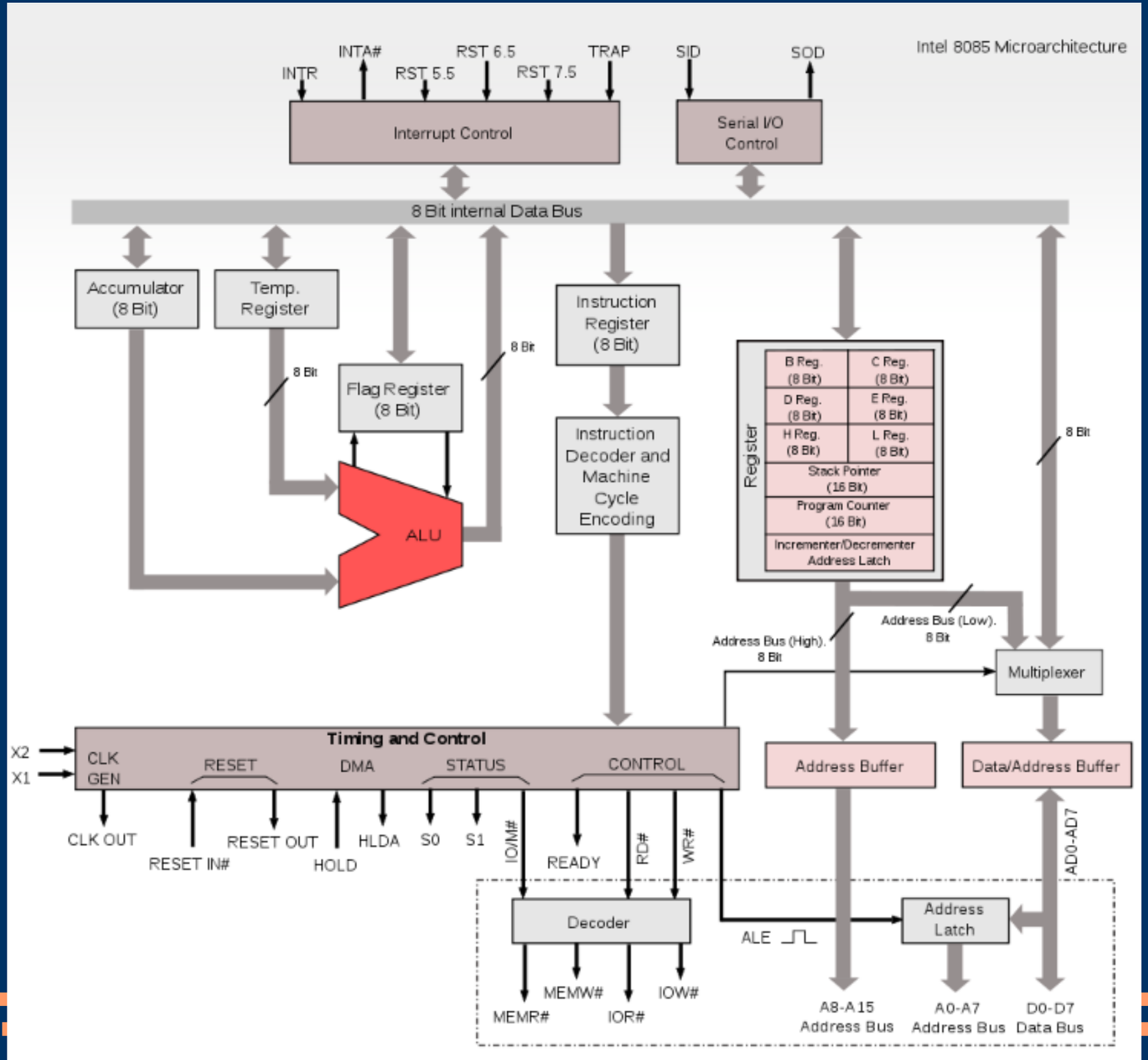
Reduction in feature sizes to pack more functionality and reduce cost/die. Follows Moore´s law

# Verification Challenges

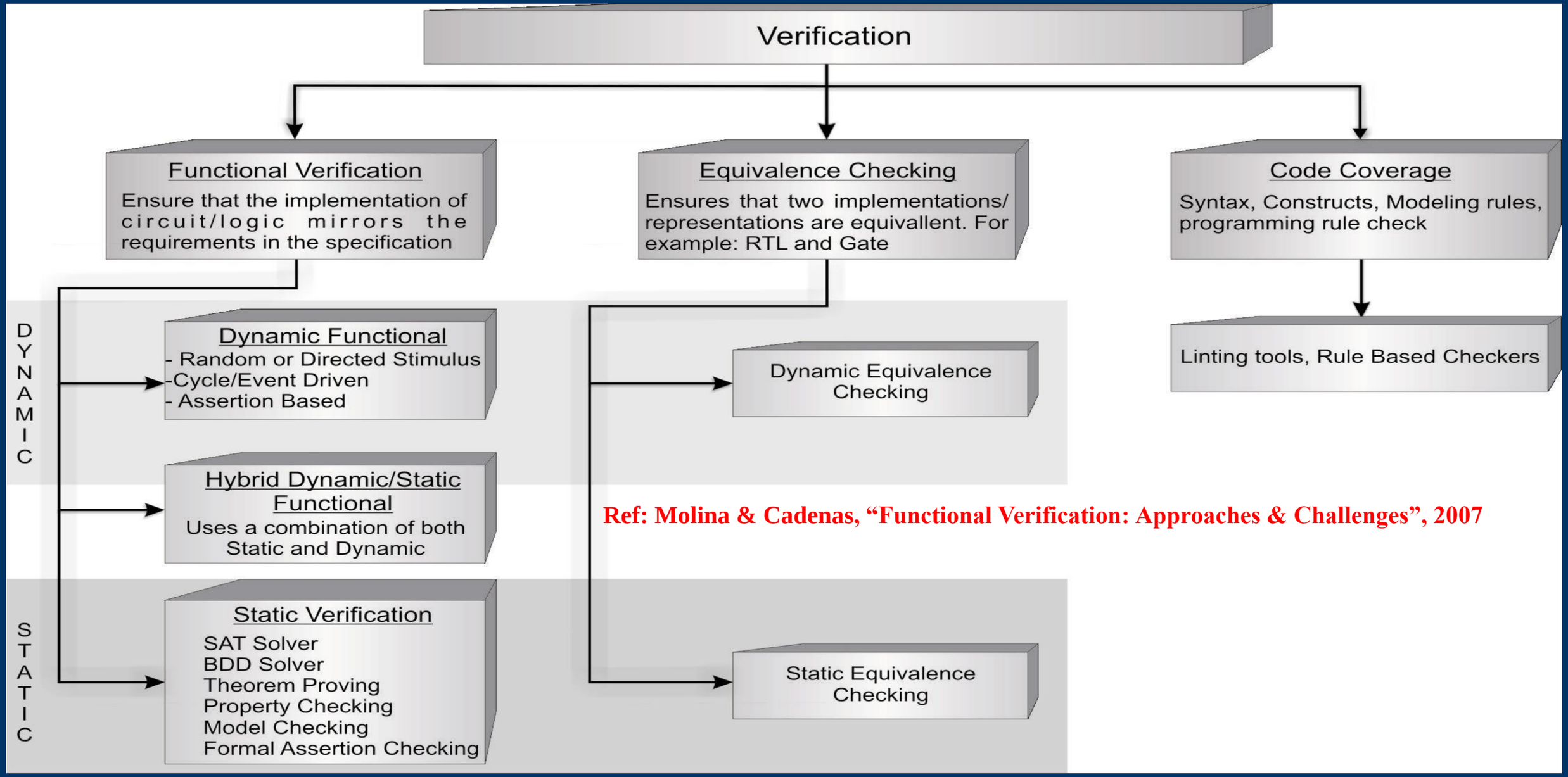



- The bug rate is linearly proportional to the number of lines of structural RTL code in each design




Intel 8085 Microarchitecture
INTR
INTA#
RST 5.5
RST 6.5
RST 7.5
TRAP
SID
SOD
Interrupt Control
Serial I/O Control
8 Bit internal Data Bus
Accumulator (8 Bit)
Temp. Register
Flag Register (8 Bit)
ALU
Instruction Register (8 Bit)
Instruction Decoder and Machine Cycle Encoding
Register
B Reg. (8 Bit)
C Reg. (8 Bit)
D Reg. (8 Bit)
E Reg. (8 Bit)
H Reg. (8 Bit)
L Reg. (8 Bit)
Stack Pointer (16 Bit)
Program Counter (16 Bit)
Incrementer/Decrementer Address Latch
8 Bit
Address Bus (High). 8 Bit
Address Bus (Low). 8 Bit
Multiplexer
Timing and Control
X2
X1
CLK GEN
RESET
DMA
STATUS
CONTROL
Address Buffer
Data/Address Buffer
CLK OUT
RESET IN#
RESET OUT
HOLD
HLDA
S0
S1
IO/M#
READY
RD#
WR#
AD0-AD7
Decoder
ALE
Address Latch
MEMR#
MEMW#
IOR#
IOW#
A8-A15 Address Bus
A0-A7 Address Bus
D0-D7 Data Bus

# Verification Challenges - Methodology




**Ref: Molina & Cadenas, "Functional Verification: Approaches & Challenges", 2007**

# *Verification Challenges*

- Almost every issue in verification can be placed in one of the three categories

  a. Missed bugs

  b. Lack of Time

  c. Lack of resources

# Verification Challenges

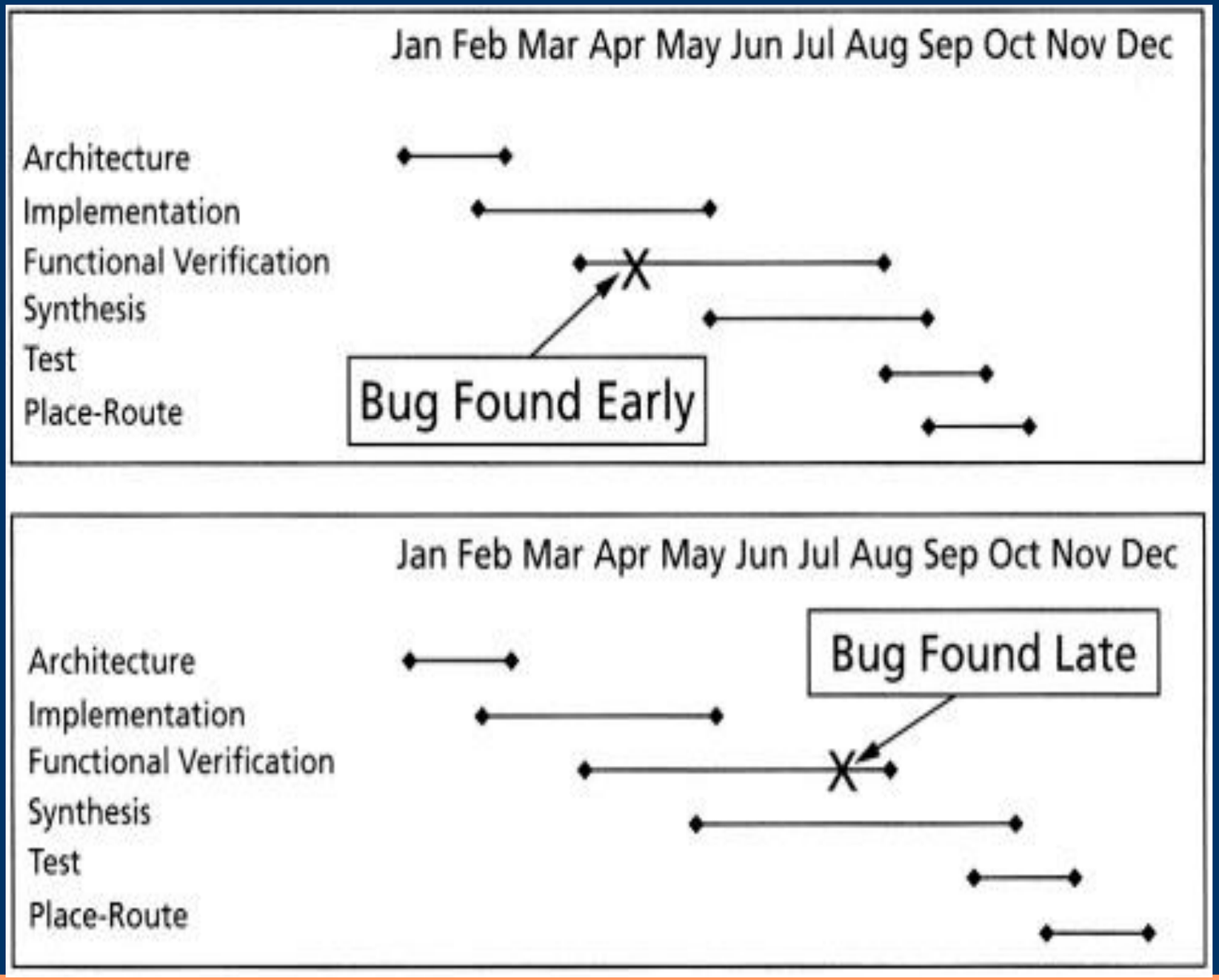
- Greater effort required than the design itself
  - Multi-million ASICs & FPGA
  - System-On-Chip
  - Reusable IPs
- A time consuming process
  - Use of tools and methodologies reduce the verification time
  - Write and debug testbenches in parallel with the design implementation
  - Use of higher abstraction levels – reduces control over low level details
- Must accommodate transition between higher and lower level of abstraction
- Use of automation
  - Variety of functions, interfaces, protocols and transformations
  - Randomization

# Missed Bugs

- The highest priority for verification teams has always been to find bugs
- The farther down the supply chain a bug is found, the more costly it is
- Most common sources of functional bugs found in silicon are
    a. Design errors
    b. Incorrect or incomplete specifications
    c. Changing specifications

# *Lack of Time*

- Time to market pressures forces the completion of IC development in less time.
- Size and complexity of designs continue to increase
- “when are we done??”
- Verifying large complex designs is an exercise in risk management
- But teams usually know when they are not done


Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Architecture
Implementation
Functional Verification
Synthesis
Test
Place-Route
Bug Found Early
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Bug Found Late
Architecture
Implementation
Functional Verification
Synthesis
Test
Place-Route

- Finding 98% of the bugs – normal process
- Finding remaining 2% (or tough bugs) is

a. Partly an art

b. Partly luck and lots of hard work

# *Lack of resources*

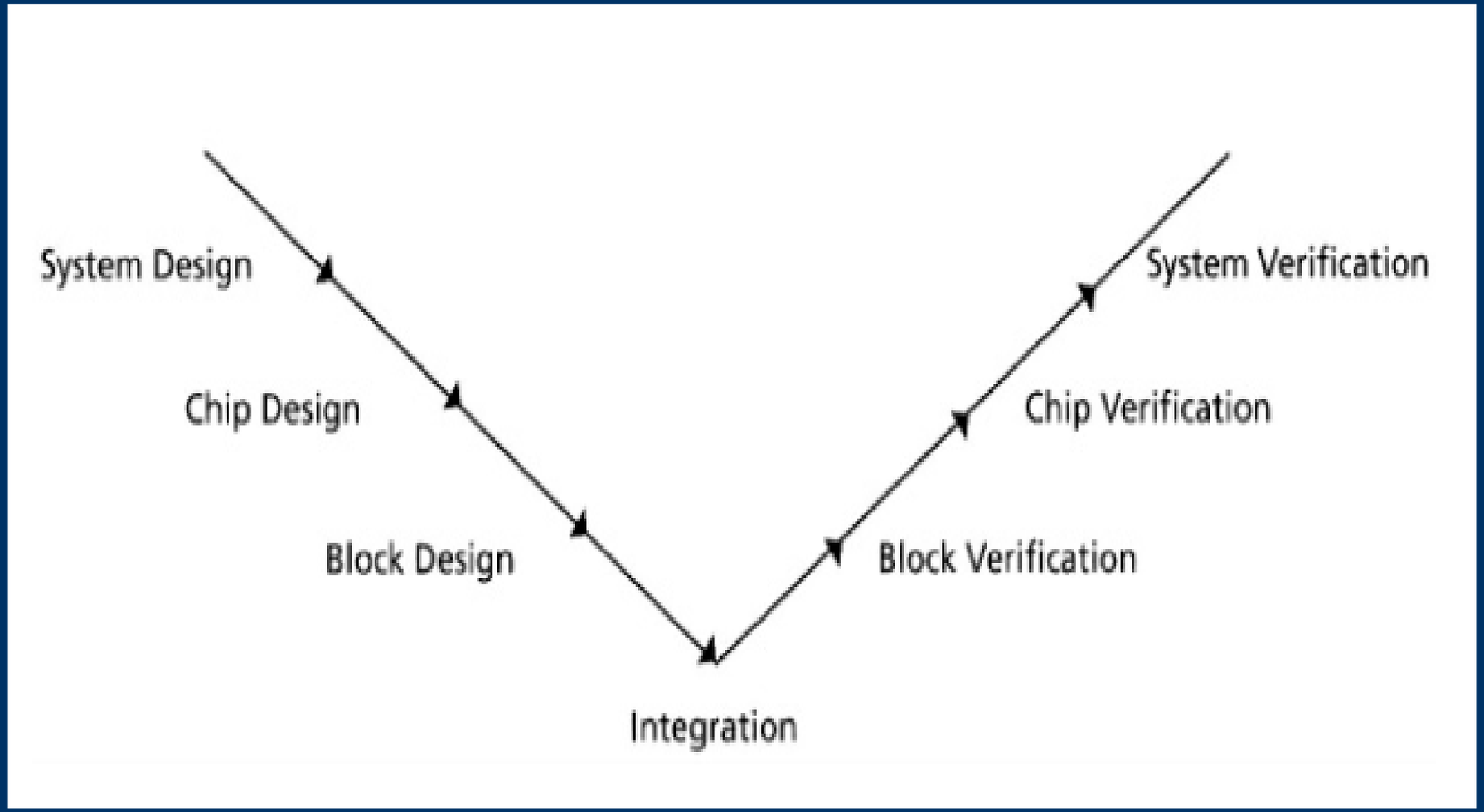
- Given enough time and resources, most verification teams can meet their goals

- Resources:
  - Experienced verification engineer – hard to find
  - Specialists + inexperienced  = inefficiency
  - S/W licenses and verification tools are costly
  - Tools often will have narrow focus
  - Verification reuse – write once and use often

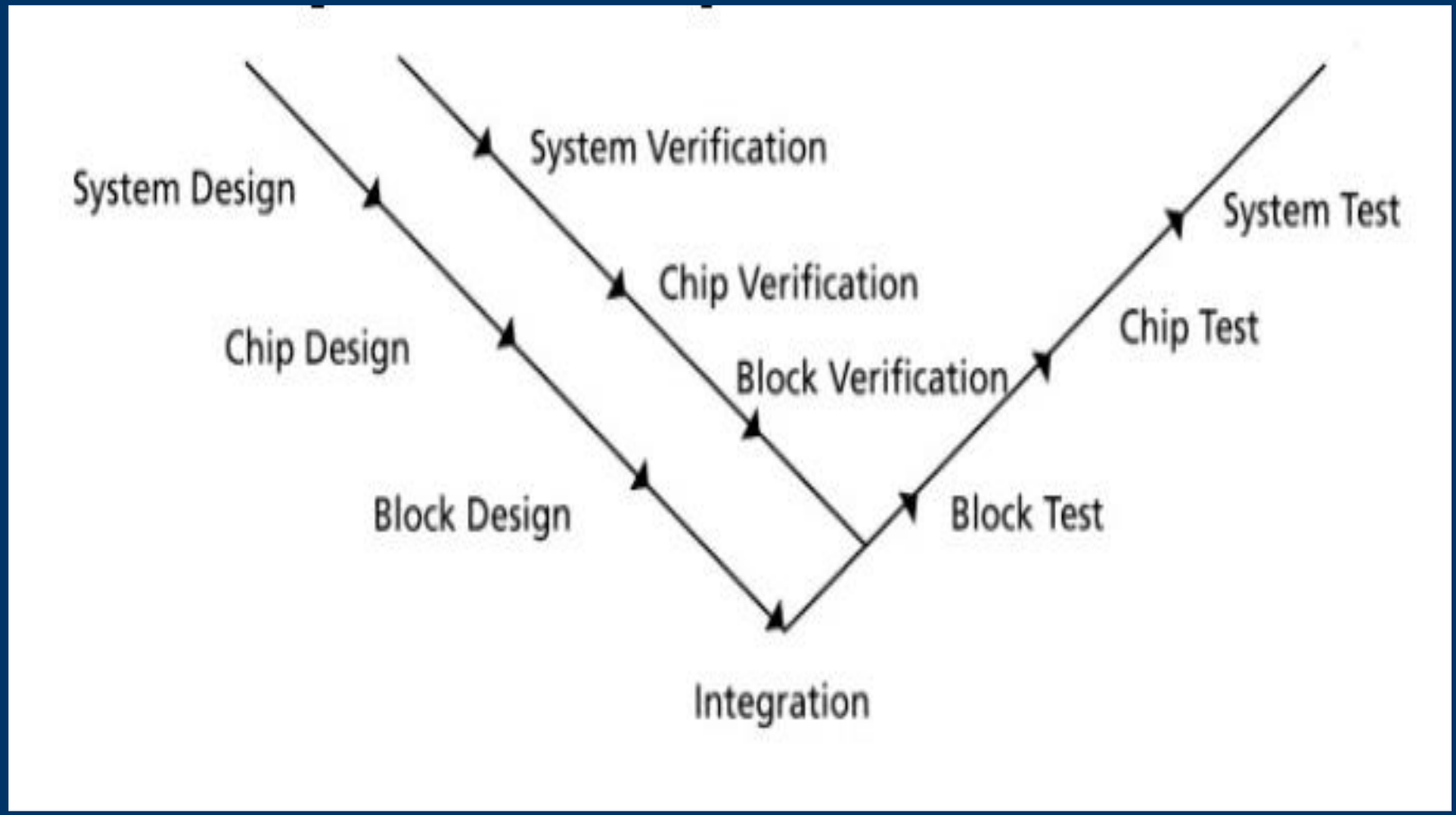
# *Basic V design*

- Verification should be separate task
- Verification requires a different mind set than implementation
- Separation results in improved efficiency and quality of results (QOR)

# Basic V design an Test Process



# *Modified V design and Test process*

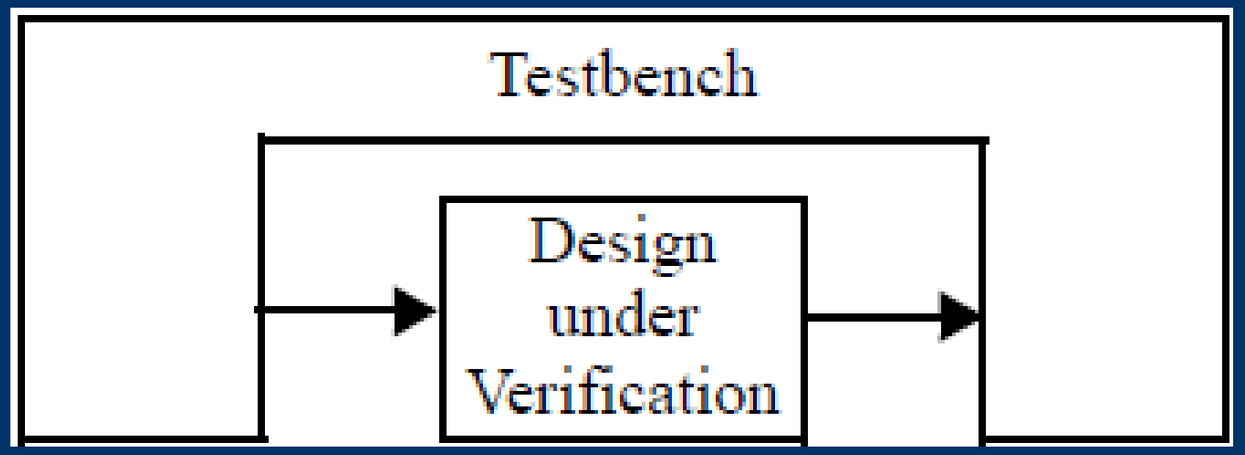


# *Verification Approach*

-Reference: Writing Testbenches using SystemVerilog

# TESTBENCH

- Simulation code used to create a predetermined input sequence to a design, then optionally to observe the response.
- Implemented using SystemVerilog, may also include external data files or C routines.




- Provides inputs to the design and watches any outputs
- Completely a closed system - no inputs or outputs go in or out
- Challenge is to determine input patterns to supply and the expected output

# *What is being verified?*

- Transformation Process: Specification -> Interpretation -> RTL Code
- Formal Verification – Equivalence Checking & Property Checking

  Verifying the properties of design using assertions
- Functional Verification – Verifying the functionality of the design
- Rule Checkers – Linting

# *Testing v/s Verification*

- Purpose of the Testing is to verify that the design was manufactured correctly.
  - Scan-Based Testing
- Purpose of the Verification is to ensure that a design meets its functional intent.