Testing and Verification – *An Introduction*

Testing and Verification of Circuits (CS60089)

Dept. of CSE, IIT Kharagpur

Dr. Aritra Hazra
Assistant Professor,
Department of Computer Science & Engineering,
Indian Institute of Technology Kharagpur,
Paschim Medinipur, West Bengal, India – 721302.

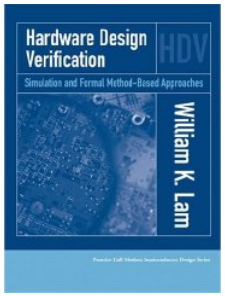
Course Details

- **Class Timings:** (Slot: F4)
  - Wednesday: 10:00 AM – 10:55 AM
  - Thursday: 09:00 AM – 09:55 AM
  - Friday: 11:00 AM – 12:55 AM
- **Venue:** Room No: CSE-120 (Ground Floor, CSE Building)
- **Course Credits:** 4 (L-T-P: 3-1-0)
- **Course Web:** http://cse.iitkgp.ac.in/~aritrah/course/theory/TVC/Autumn2018/CS60089_TVC_Autumn2018.html
- **Examinations:**
  - Tutorials (10 x 10 marks = 100 marks): 10%
  - Term-Projects (20+10+20 = 50 marks): 25%
  - Mid-Semester (50 marks): 25%
  - End-Semester (80 marks): 40%
- **Attendance:** Mandatory (> 80%)
  - failure lead to de-registration from course

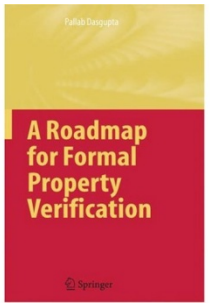
Course Agenda

- ***Design and Simulation Part***
  - Design Verification: Introduction and Overview
  - Design Entry: *Verilog Basics*
  - Simulators: *Working Principle*
- ***Design Verification Part***
  - Test Scenarios and Coverage: *Test-bench, Simulation Coverage*
  - Design Specifications: *LTL, CTL, SVA*
  - Symbolic Representation of Logic and State Spaces: *BDDs, SAT*
  - Equivalence Checking
  - Formal Verification: *Model Checking, Bounded Model Checking*
  - Abstraction-Refinement: *CEGAR*
  - Formal Verification Coverage
- **Circuit Testing Part**
  - Test Modeling: Logic and Fault Simulation
  - Testability Measures and Analysis
  - Test Pattern Generation: Combinational + Sequential ATPG
  - Design for Testability: Scan Chain based DFT
  - Built-in-Self-Test (BIST) and Memory Testing

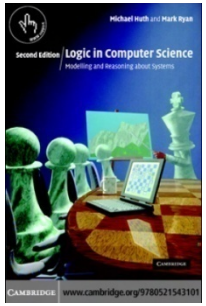
Related References (Verification)



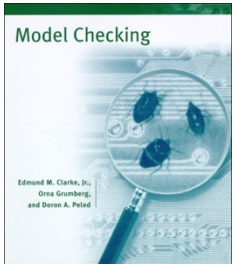
Hardware Design Verification:
Simulation and Formal Method-Based Approaches
William K Lam
Prentice Hall Modern Semiconductor Design Series



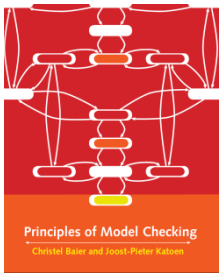
A Roadmap for Formal Property Verification
Pallab Dasgupta
Springer



Logic in Computer Science
Michael Huth and Mark Ryan
Cambridge University Press

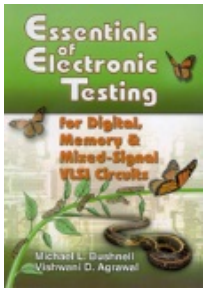


Model Checking
Edmund M. Clarke, Orna Grumberg and Doron A. Peled
The MIT Press

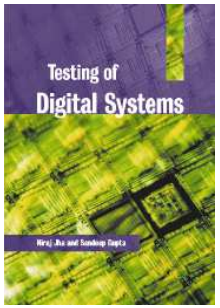


Principles of Model Checking
Christel Baier and Joost-Pieter Katoen
The MIT Press

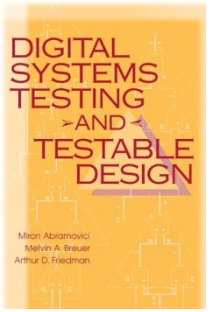
Related References (Testing)



Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits
Michael L. Bushnell and Vishwani D. Agrawal
Kluwer Academic Publishers

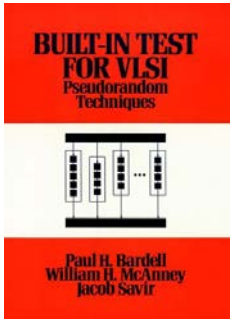


Testing of Digital Systems
Niraj K. Jha and Sandeep Gupta
Cambridge University Press



Digital Systems Testing and Testable Design
Miron Abramovici, Melvin A. Breuer and Arthur D. Friedman
Wiley-IEEE Press

VLSI Test Principles and Architectures
Laung-Terng Wang, Cheng-Wen Wu and Xiaoqing Wen
Morgan Kaufman Publishers



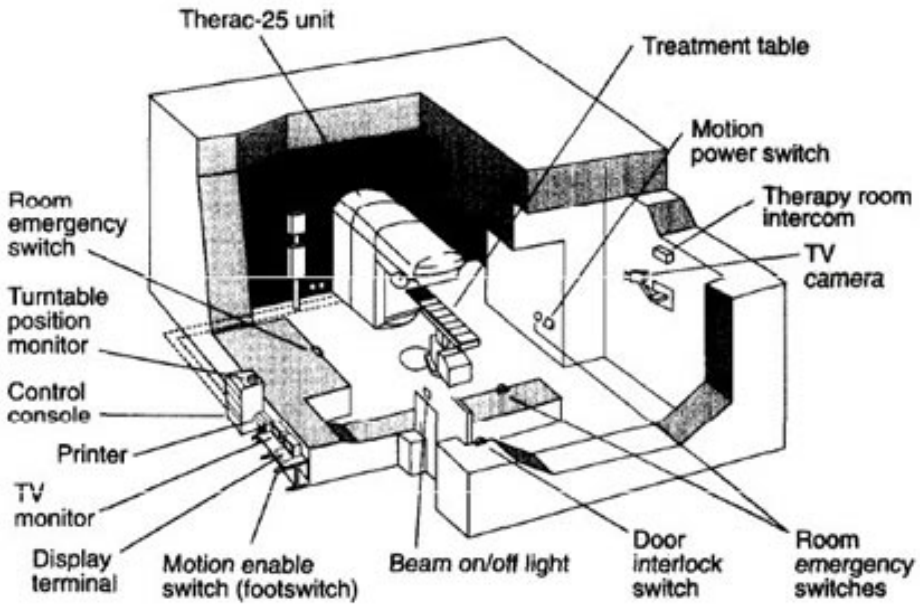
Built-in Test for VLSI: Pseudorandom Techniques
Paul H. Bardell, William H. McAnney and Jacob Savir
Wiley Interscience

Fault Tolerant and Fault Testable Hardware Design
Parag K. Lala
Prentice-Hall International

Why Testing and Verification?

Therac-25 Radiation Overdosing (1985-1987)

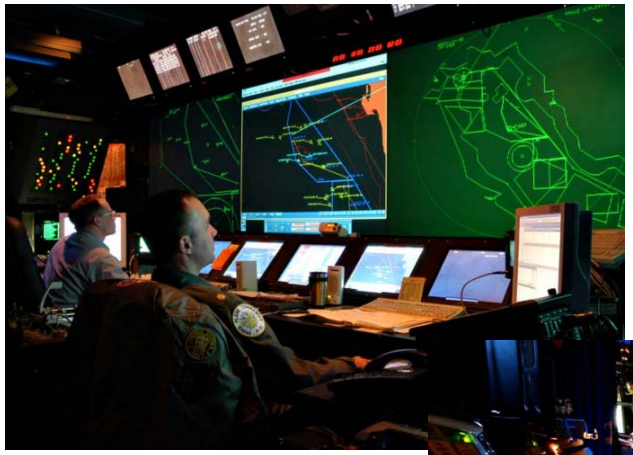
- Radiation machine for treatment of cancer patients
- At least 6 cases of overdoses in 1985-1987 (~100-times doses)
- Three cancer patients died
- Source: Design error in the control software (race condition)



Why Testing and Verification?

Shooting Down Airbus 320 (1988)

- US Vincennes shot down Airbus 320
- Mistook Airbus 320 for a F-14
- 290 people died



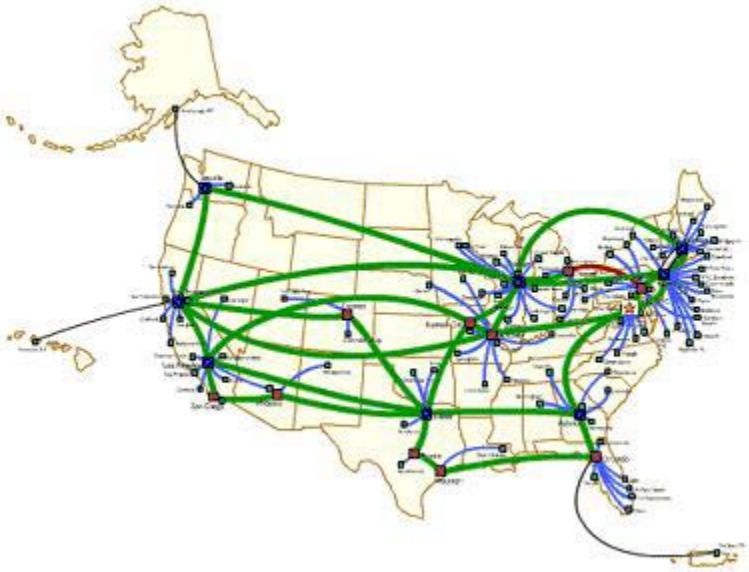
Source:

Software Bug – cryptic and misleading output displayed by the traking software

Why Testing and Verification?

❑ AT&T Telephone Network Outage (1990)

- January 1990: problem in New York City leads to 9-hour outage of large parts of U.S. telephone network
- Costs: several 100 million US$
- Source: Software Flaw
  (wrong interpretation of break statement in C)



Why Testing and Verification?

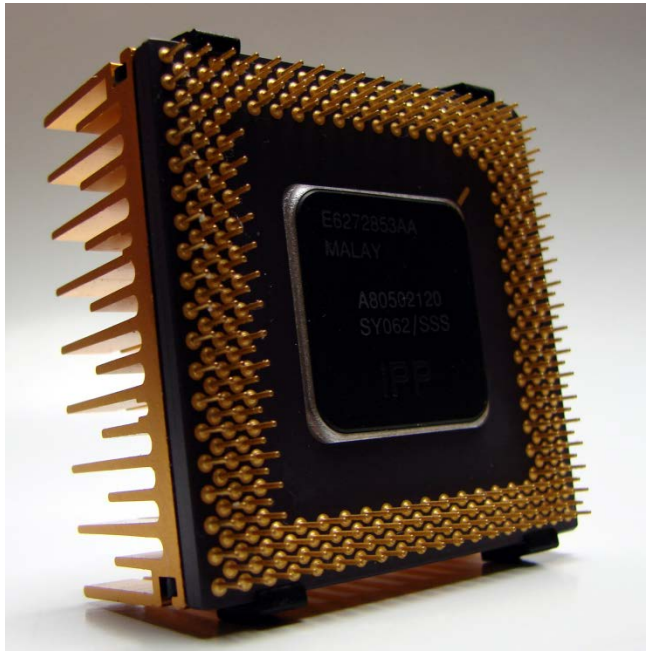
Computer-Aided Ambulance Service in London (1992)

- London Ambulance Service computer aided despatch system
  - Area 600sq miles
  - Population 6.8million
  - 5000 patients per day
  - 2000-2500 calls per day
  - 1000-1200 emergency calls

- Costs: 20-30 people estimated to have died as a result

- Source: position of vehicles incorrectly recorded and multiple vehicles sent to the same location

Why Testing and Verification?



❑ Pentium FDIV Bug (1994)

- FDIV = Floating-point DIVision unit
- Certain floating-point division operations performed produced incorrect results
- Byte: 1 in 9 billion floating point divides with random parameters would produce inaccurate results
- Loss: 500 million US$ (all awed processors were replaced) + enormous image loss of Intel Corp.

- Source: Flawless realization of floating-point division

Why Testing and Verification?

❑ Ariane 5 Crash (1996)

- Crash of the European Ariane-5 missile in June 1996
- Crashed 40 seconds after Launch
- Costs: more than 500 million US$

- Source:
  - Flaw in the control software
  - A data conversion from a 64-bit floating-point to 16-bit signed integer
  - Efficiency considerations had led to the disabling of the software handler (in Ada)

Why Testing and Verification?

❑ Anti-lock Braking Problem in Toyota Prius (2010)

- Toyota Prius : first mass-produced hybrid vehicle
- Numerous complaints/accidents
- Source: Software "glitch" found in anti-lock braking system
- Costs: Eventually fixed via software update
  - in total 625,000 cars recalled
  - handling of the incident prompted much criticism, bad publicity

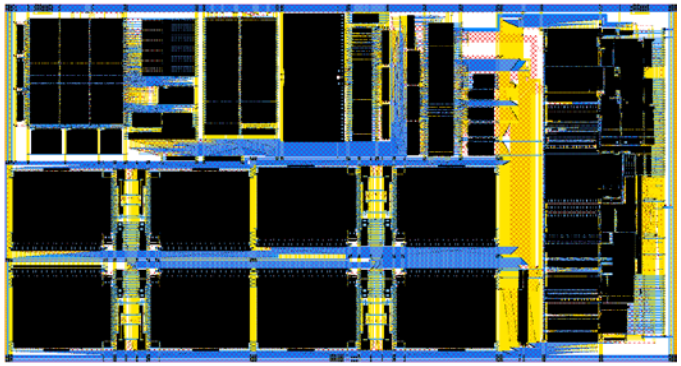
We must Test and Verify Automatically!

"Testing can only show the presence of errors, not their absence."

"In their capacity as a tool, computers will be but a ripple on the surface of our Culture. In their capacity as intellectual challenge, computers are without precedent in the cultural history of mankind."

Edsger Dijkstra (1930-2002)

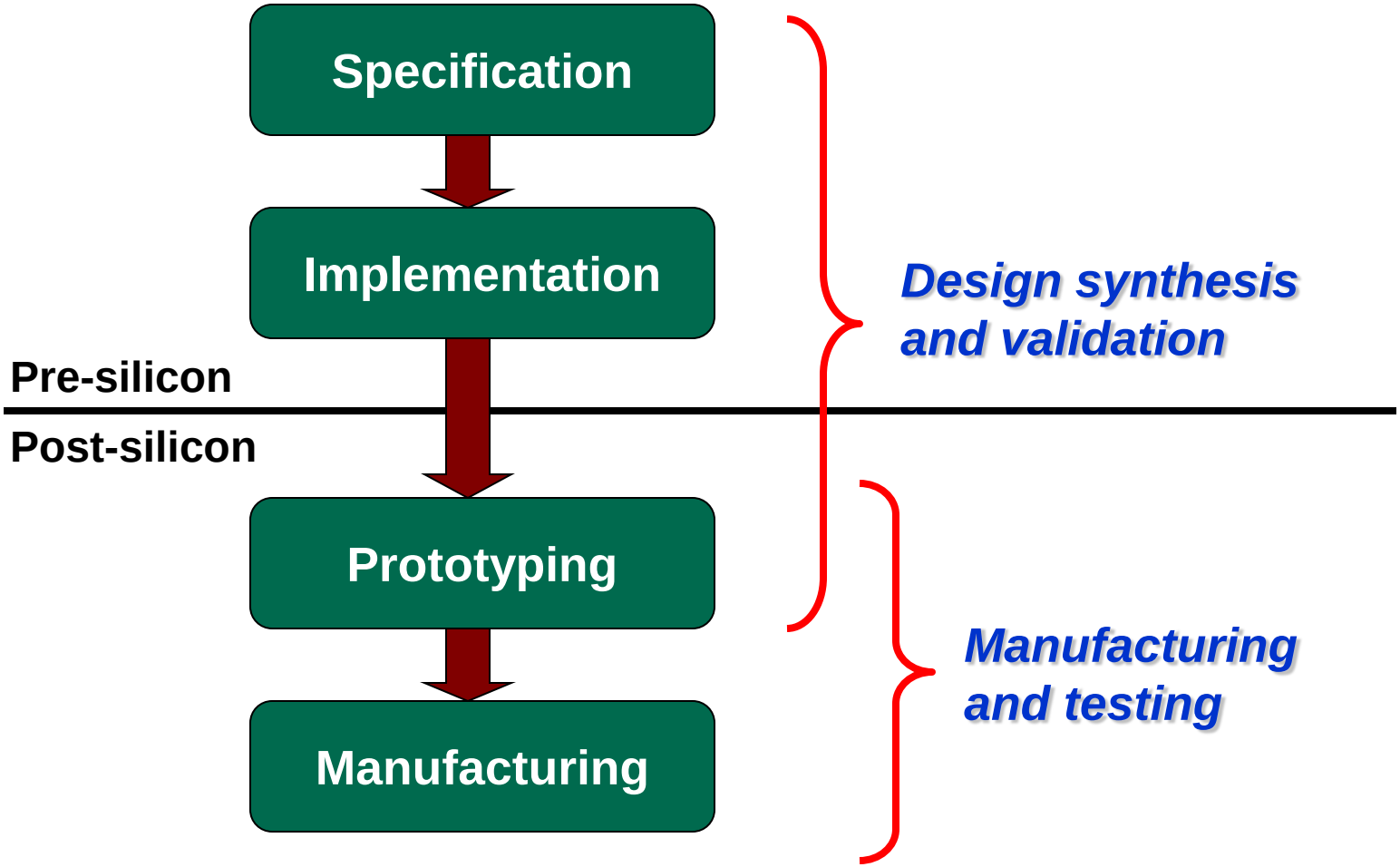
To rule out errors must consider all possible executions – often not feasible mechanically!



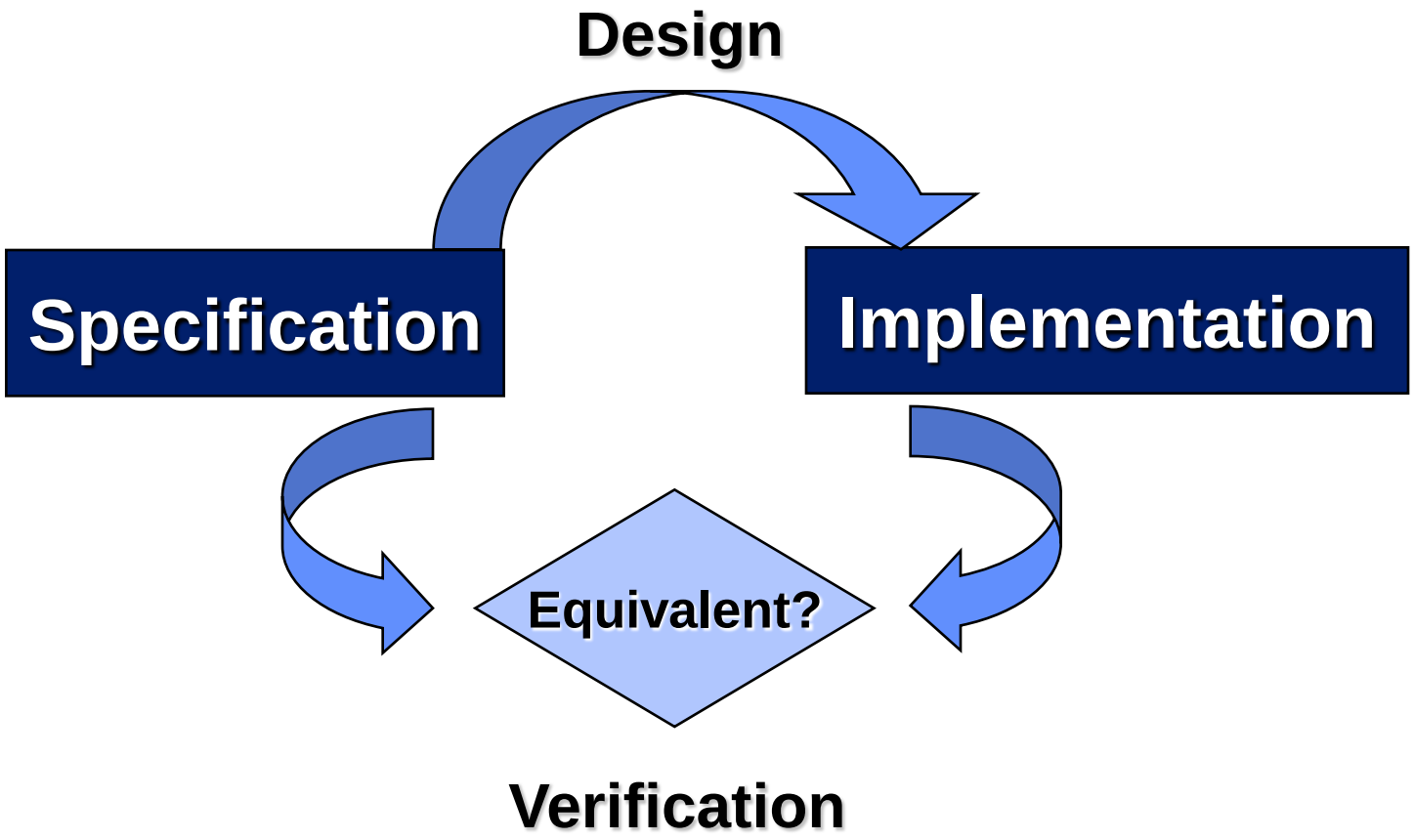
$10^{500,000}$ states

10^{70} atoms

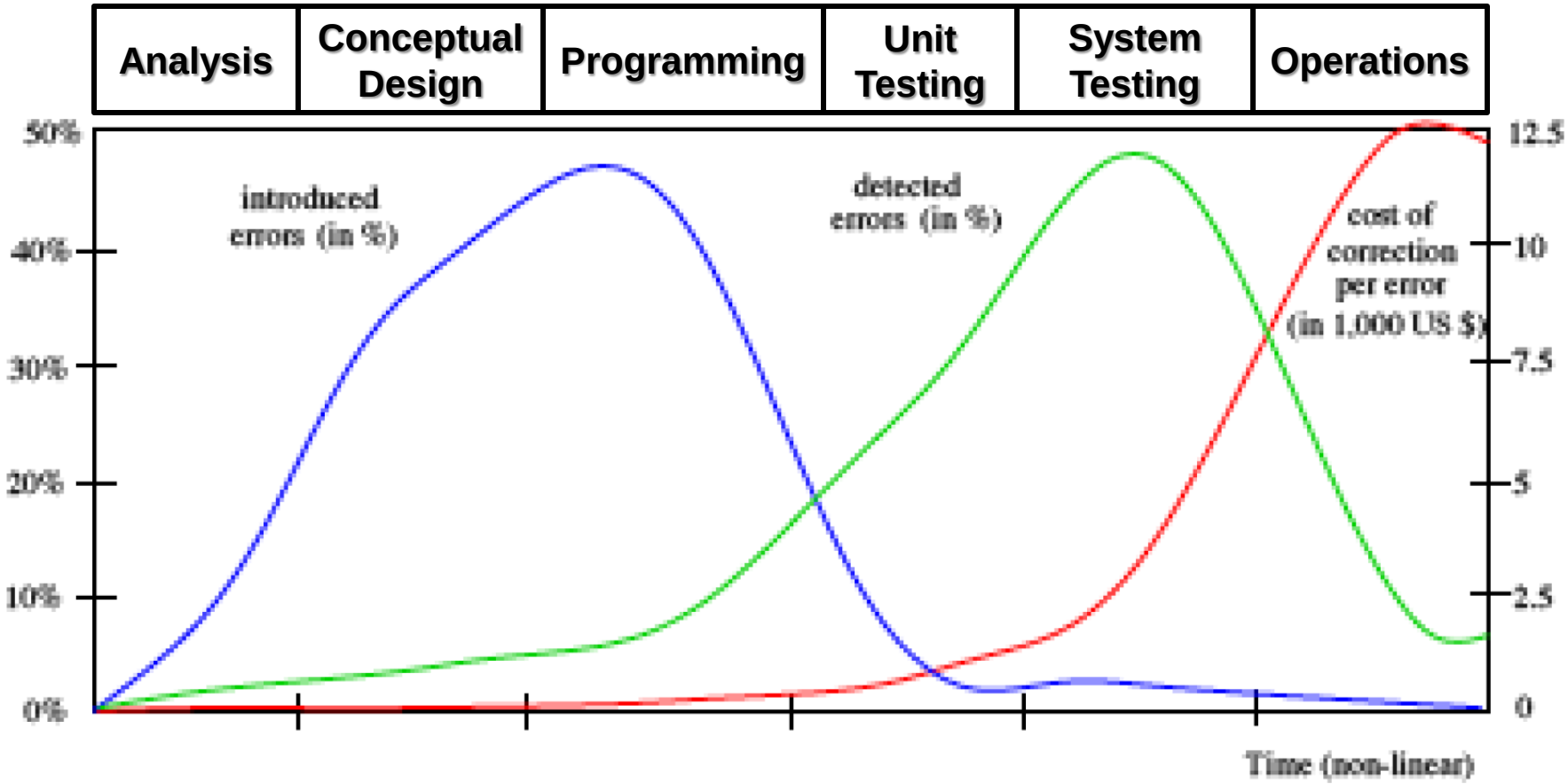
Design, Validation and Testing



Design and Verification



Design Verification: the Sooner, the Better!



Design Challenges

Suppose we have to design a pacemaker

Resource optimization → **Reduce Cost**

Test automation and Design-for-Test → **Improve Testability**

Algorithms for better yield → **Improve Reliability**

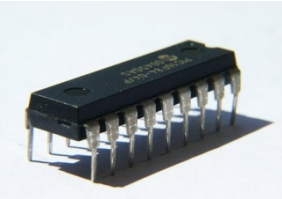
Improve Verifiability → ***Verification reliability and coverage***

Minimize Power → ***Power optimization during synthesis***

Minimize Area → ***Area optimization during synthesis***

Minimize Delay → ***Delay optimization during synthesis***

Digital Design: Abstraction Levels



Exponential growth in circuit size (Moore's Law)

Formalisms introduced at the Entry-Level

Register Transfer Level

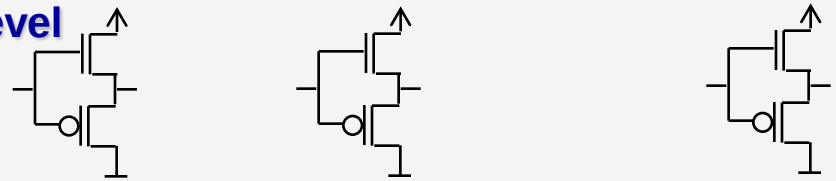
```
always @( posedge clk )
begin
 if (!rst) begin a1 <= a2;
      a2 <=  ~a1; end;
end
```

Restricted semantics of Programming Languages, Communicating Concurrent State Machines (CSM)

Gate Level

Boolean Logic
Finite State Machines

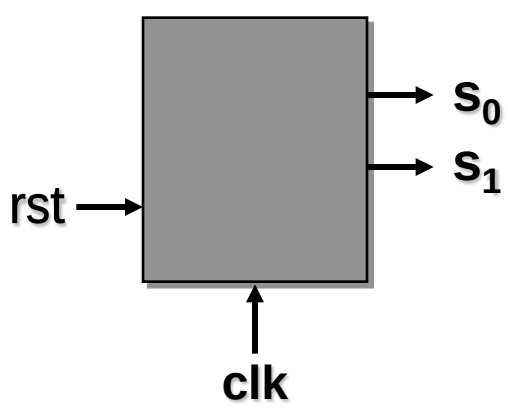
Transistor Level

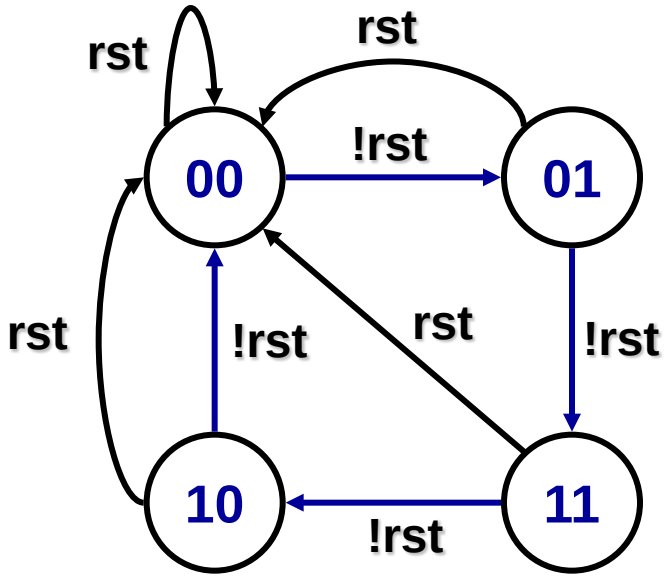


Schematic

Design Example: 2-bit Gray Counter

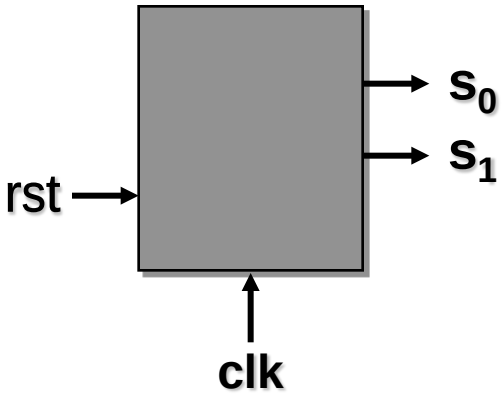
Gray Counter: Successive values should differ only in one bit. Reset signal resets the counter to zero.

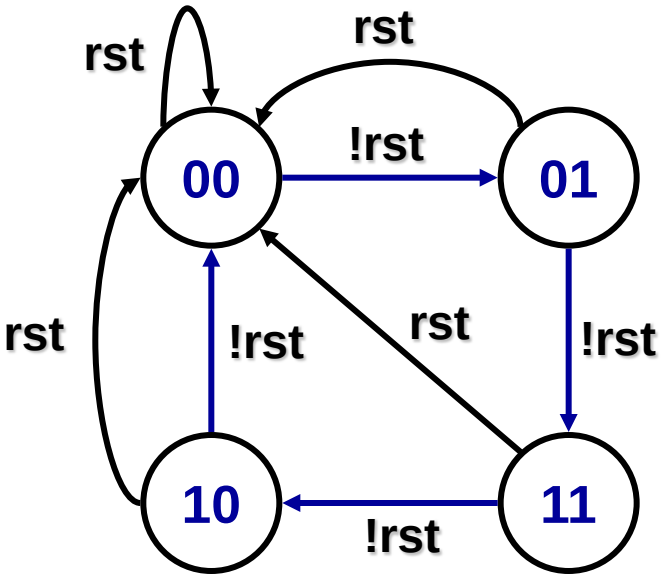




State m/c Representation

Design Example: 2-bit Gray Counter





State m/c Representation

State Transition Table

(s_0s_1)	rst	(n_0n_1)
00	0	01
00	1	00
01	0	11
01	1	00
10	0	00
10	1	00
11	0	10
11	1	00

Design Example: 2-bit Gray Counter

State Transition Table

$(s_0 s_1)$	rst	$(n_0 n_1)$
00	0	01
00	1	00
01	0	11
01	1	00
10	0	00
10	1	00
11	0	10
11	1	00

State Transition Functions:

$$n_0 = s_0' s_1 r' + s_0 s_1 r'$$
$$n_1 = s_0' s_1' r' + s_0' s_1 r'$$

After Logic Minimization:

$$n_0 = s_1 r'$$
$$n_1 = s_0' r'$$

Design Example: 2-bit Gray Counter

State Transition Functions:

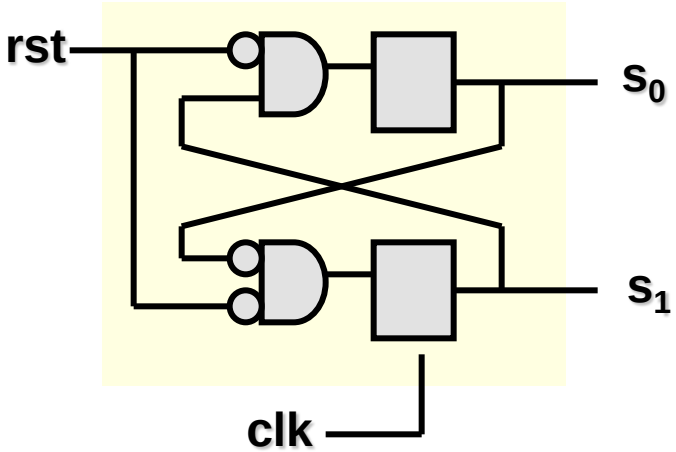
$$n_0 = s_1 r'$$
$$n_1 = s_0' r'$$

Verilog Code (RTL):

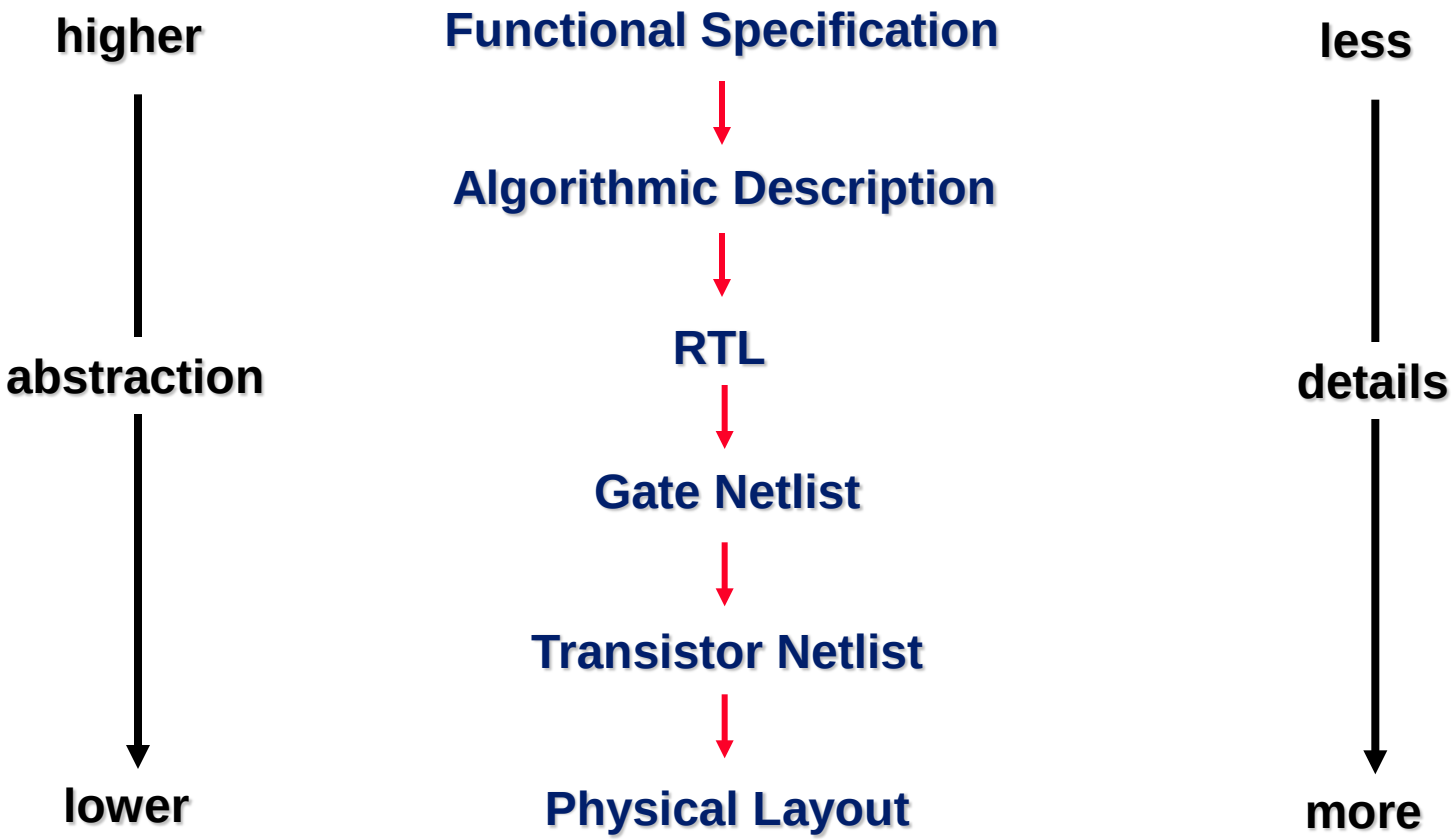
```
module GrayCounter(s0, s1, rst)
input rst;
reg s0, s1;

always @ (posedge clk)
begin
    s0 <= s1 & ~rst;
    s1 <= ~s0 & ~rst;
end
endmodule
```

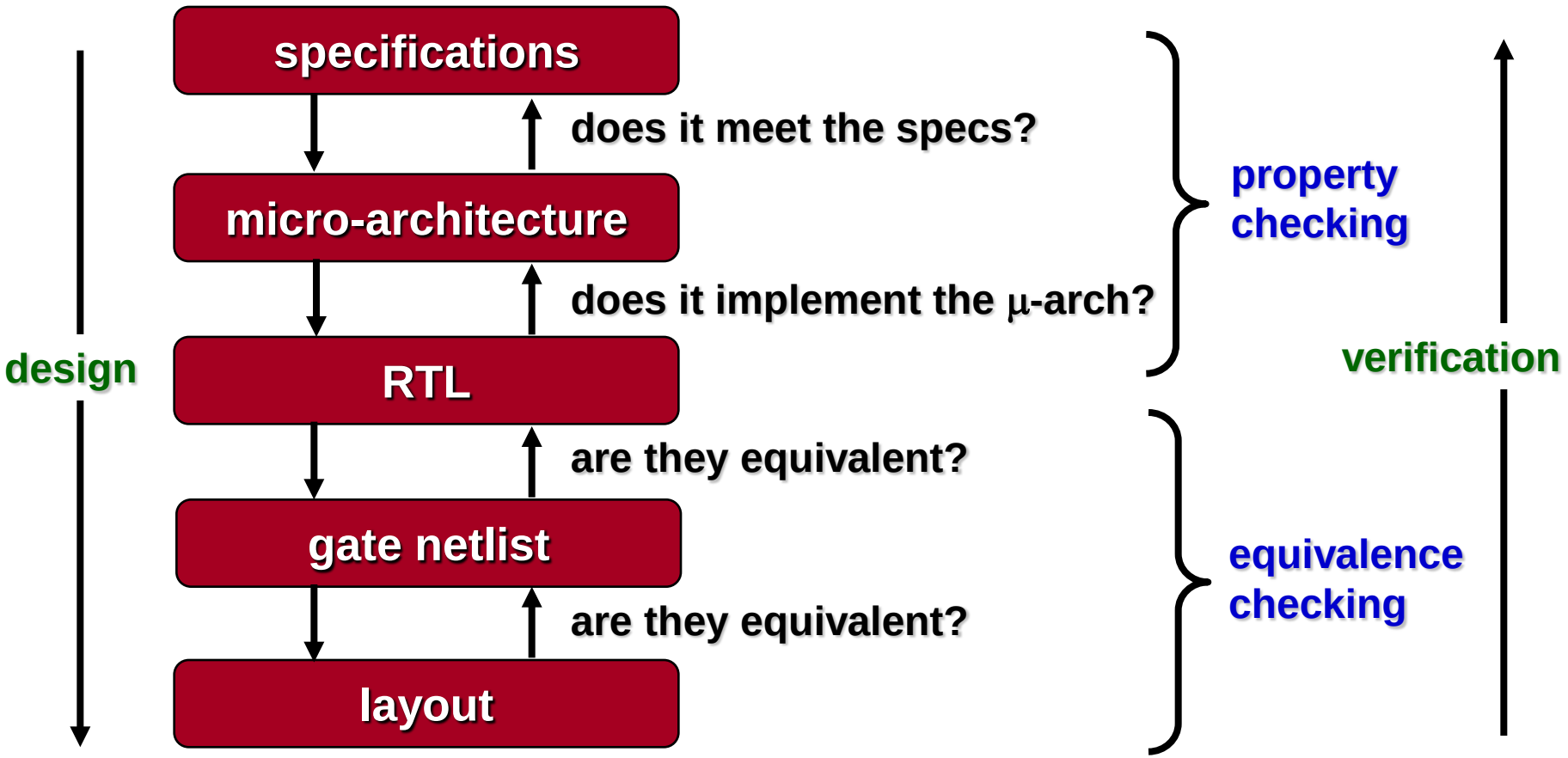
Synthesis



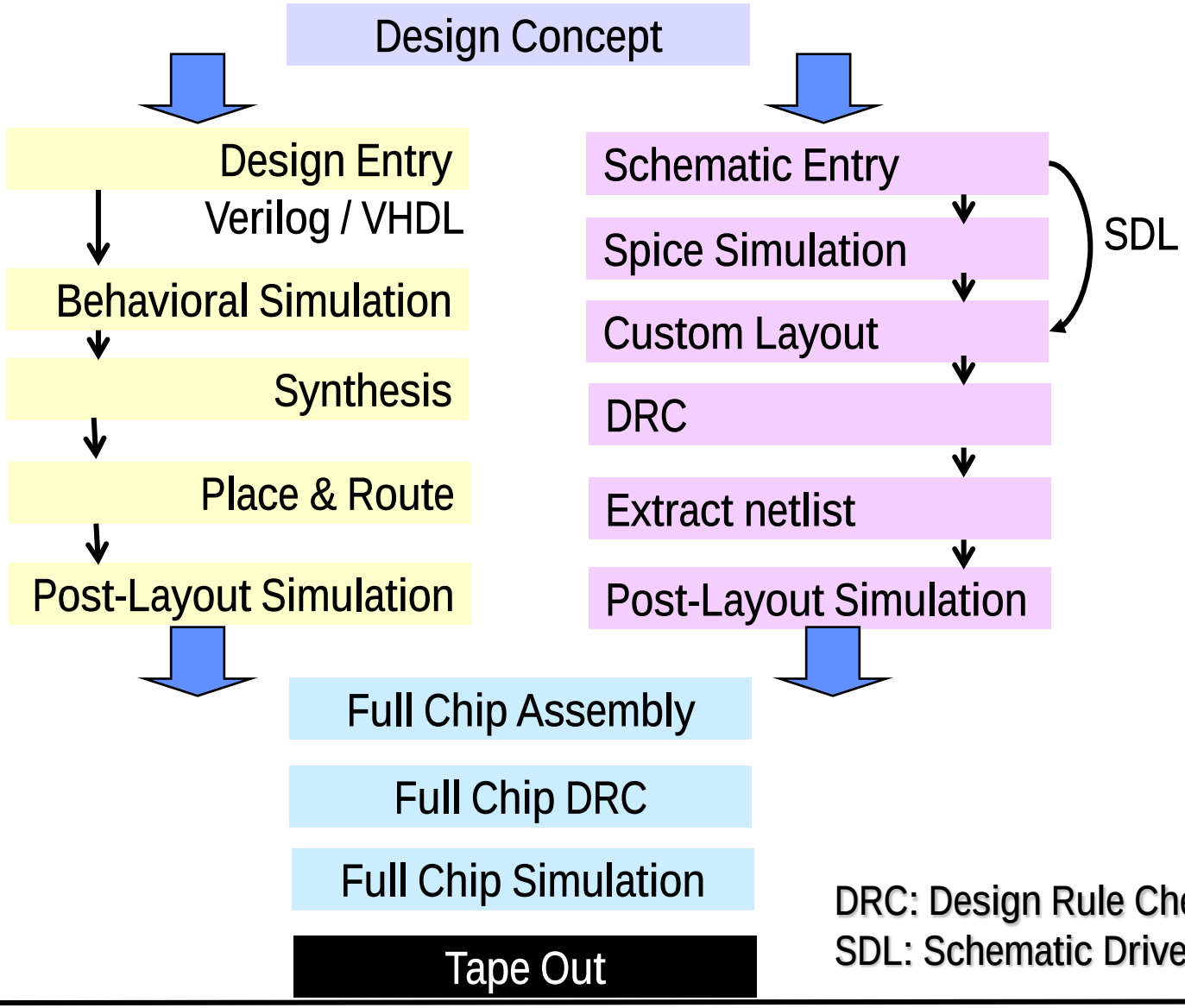
Abstractions in Design Flow



Design and Verification

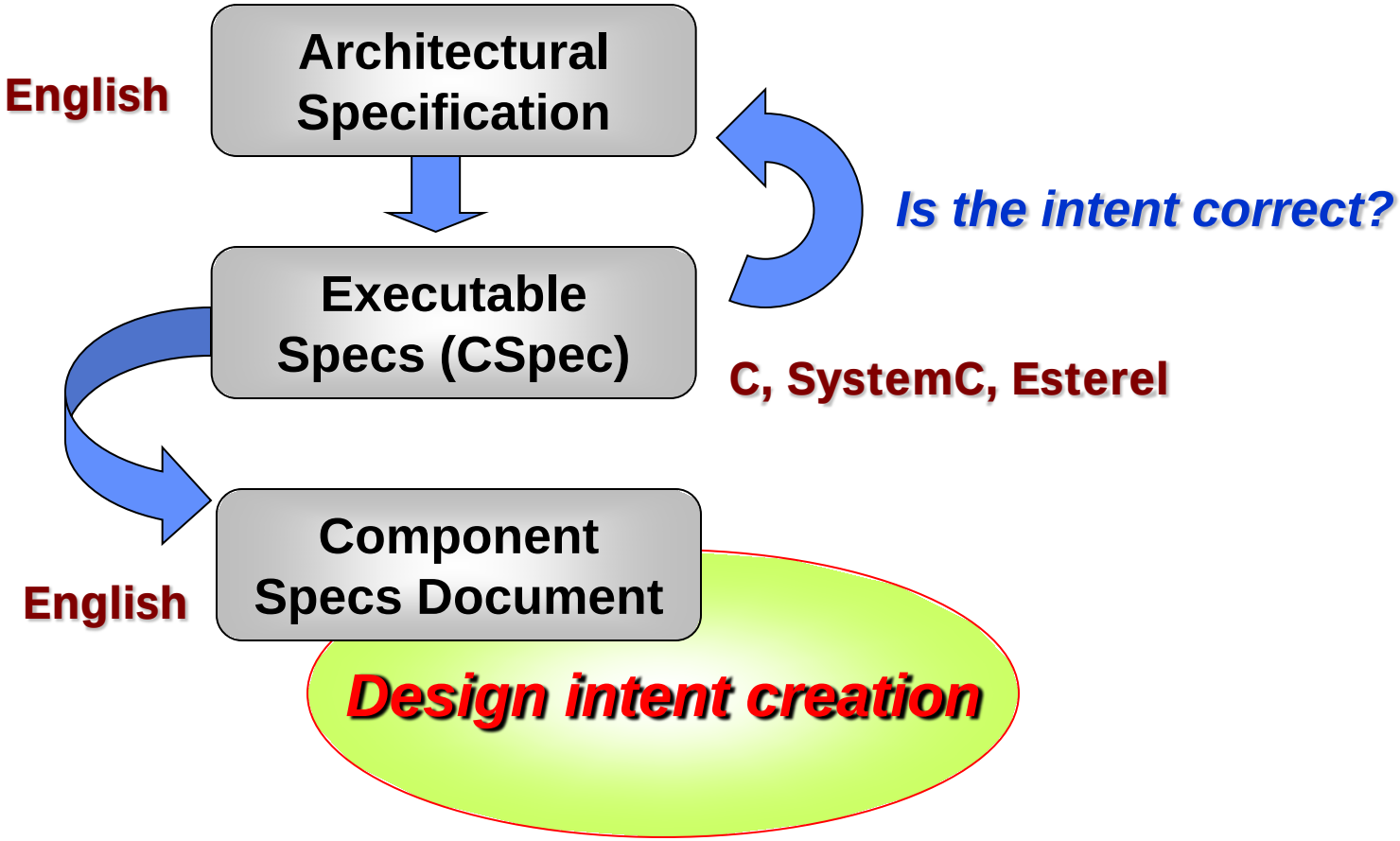


Design Flows: *Digital versus Analog*

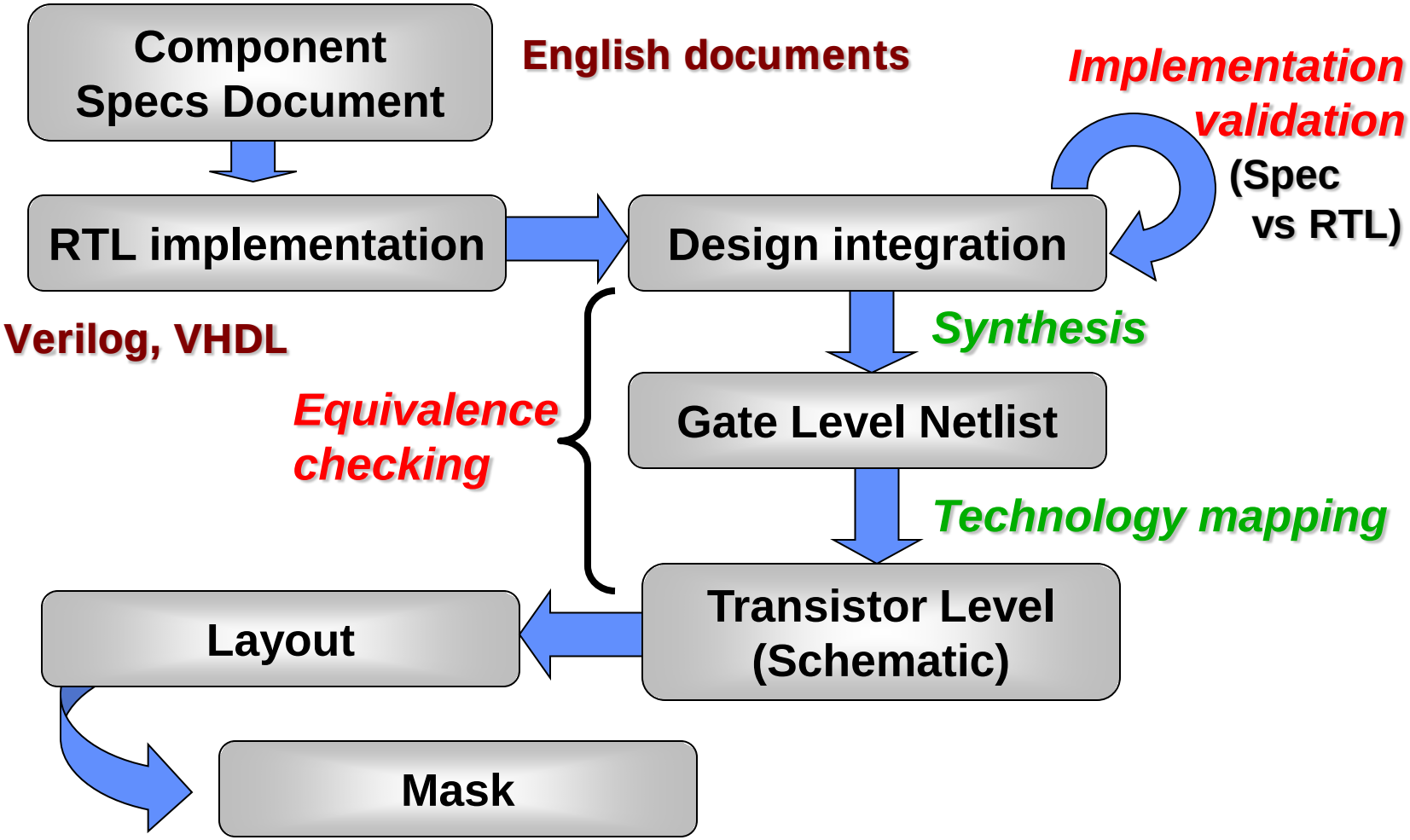


DRC: Design Rule Checking
SDL: Schematic Driven Layout

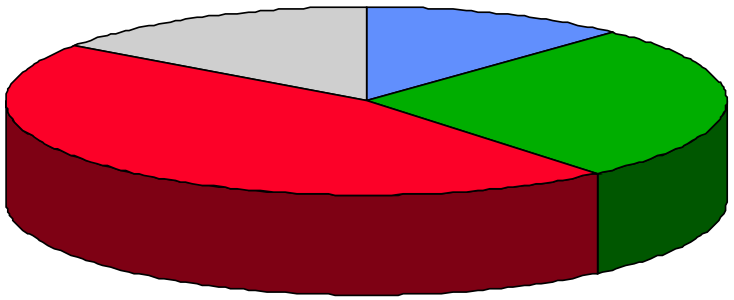
Design Cycle: *Intent Creation*



Design Cycle: *Implementation*



Verification Dominates Design



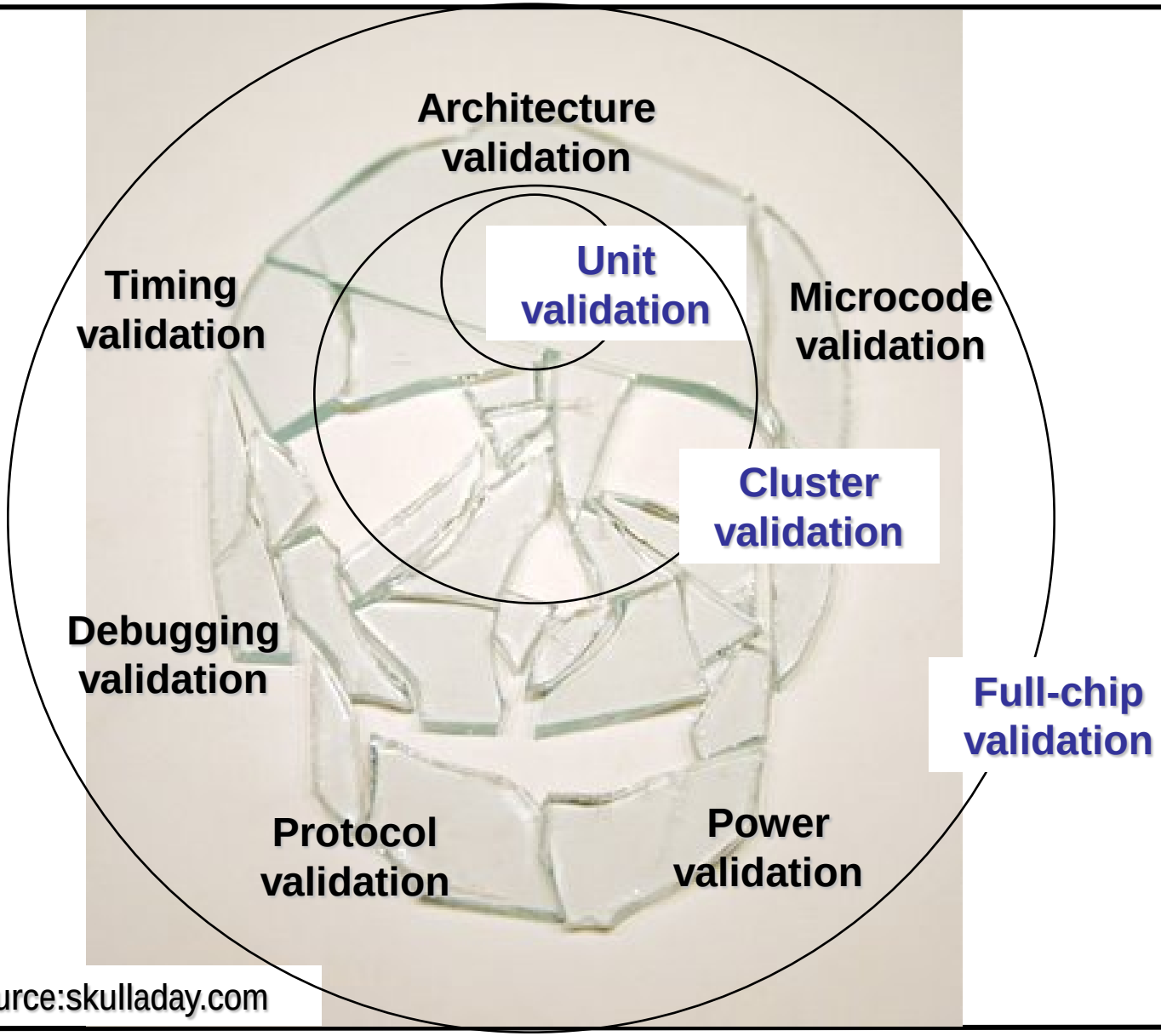
- Synthesis
- Timing analysis
- Equivalence checking
- DFT

- High-level design
- RTL coding
- Block-level simulation

- Behavioral modeling
- Architecture level simulation
- System level simulation

Source: 0-In Design Automation

Pieces of the verification puzzle



Picture source:skulladay.com