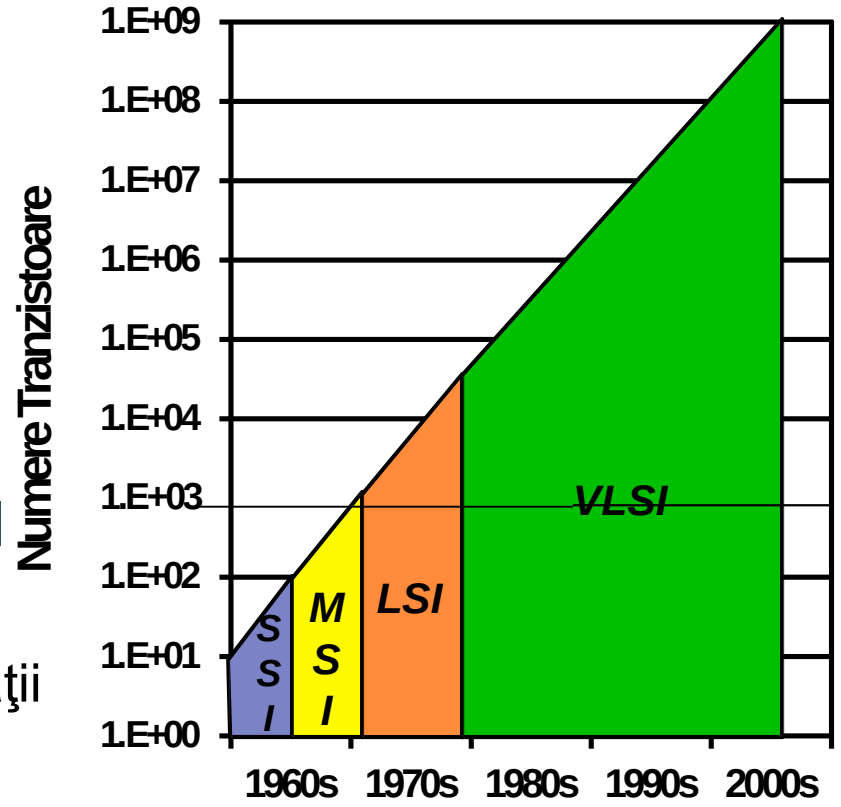
Testarea şi Fiabilitatea Sistemelor de Calcul Curs 1

Ce conţine acest capitol?

- Diferenţierea noţiunilor de testare - verificare
- Analiză a următoarelor aspecte:
  - Importanţa testării în procesul de proiectare şi manufactură a circuitelor
  - Necesitatea testării
  - Testarea pe durata de viaţă a unui circuit (la general)
  - Verificarea functionala (detaliat)

Introducere

- Circuitele Integrate (ICs) au crescut în mărime şi complexitate după anii 1950
  - Integrare pe scală mică (SSI)
  - Integrare pe scală medie (MSI)
  - Integrare pe scală largă (LSI)
  - Integrare pe scală foarte largă (VLSI)
- *Legea lui Moore*: densitatea CI se dublează la fiecare 18 luni
  - Creşterea densităţii şi complexităţii aduce noi probleme în testare



Importanţa Testării

- Legea lui Moore porneşte de la scăderea lărgimii canalului unui tranzistor
  - Scăderea a avut loc de la zecimi de μm la zecimi de nm pentru tranzistori şi interconexiuni
- Frecvenţele de lucru au crescut de la sute de KHz la GHz
- Scăderea dimensiunilor tranzistoarelor au făcut să crească probabilitatea generării de defecte în timpul manufacturii
  - Este suficient un tranzistor cu defect pentru a fi defect circuitul integrat -> testarea este esenţială

Importanţa Testării

- *Creştere exponenţială*: costul detectării unui CI defect creşte exponenţial în domeniile:
  - dispozitiv → PCB → sistem → sistem în funcţiune
    - Testarea este necesară la fiecare din aceste nivele
- Testarea are loc şi în timpul:
  - Manufacturării pentru creşterea productivităţii
    - Metodă folosită: Failure mode analysis (FMA)
  - Funcţionării în timp real, pentru a reduce rata de defecte:
    - În această fază defectele se repară

Necesitatea Testării

- Tendinţele tehnologice VLSI şi impactul asupra testării ["Design Verification & Testing" UMBC, CMPE 418]

Perioada	1997 - 2001	2003 - 2006	2009 - 2012
Dimensiunea tranzistorului (μm)	0.25 – 0.15	0.13 – 0.10	0.25 – 0.15
Milioane tranzistori / cm^2	4 - 10	18 - 39	84 - 180
Numărul de straturi tehnologice	6 - 7	7 - 8	8 - 9
Mărimea *dye*-ului, în mm^2	50 - 385	60 - 520	70 - 750
Numărul de pini	100 - 900	160 - 1475	260 - 2690
Frecvenţa folosită, în MHz	200 - 730	530 – 1100	840 - 1830
Tensiune, în V	1,2 – 2,5	0,9 – 1,5	0,5 – 0,9
Putere, în W	1,2 - 61	2 - 96	2,8 - 109

Necesitatea Testării

- Exemplu practic ["Design Verification & Testing" UMBC, CMPE 418]

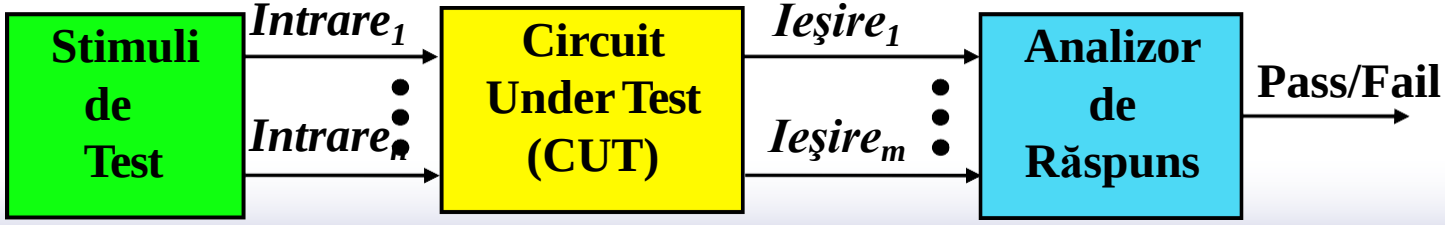
Costul echipamentului automatizat (ATE) estimat pentru un CI cu frecvenţă > 250 MHz:

- Necesar un tester pentru 500 MHz, al cărui cost este:

  1.2 M $ + (1024 pini x 3000 $ / pin) = 4272 M $
- Costul de întreţinere: 1439 M$/an (4.5 centi/s)
- Timpul de testare a unui ASIC: 6 s -> 27 centi
- Pentru un randament de 65%, preţul de testare: 27 / 0.65 = 41.5 centi

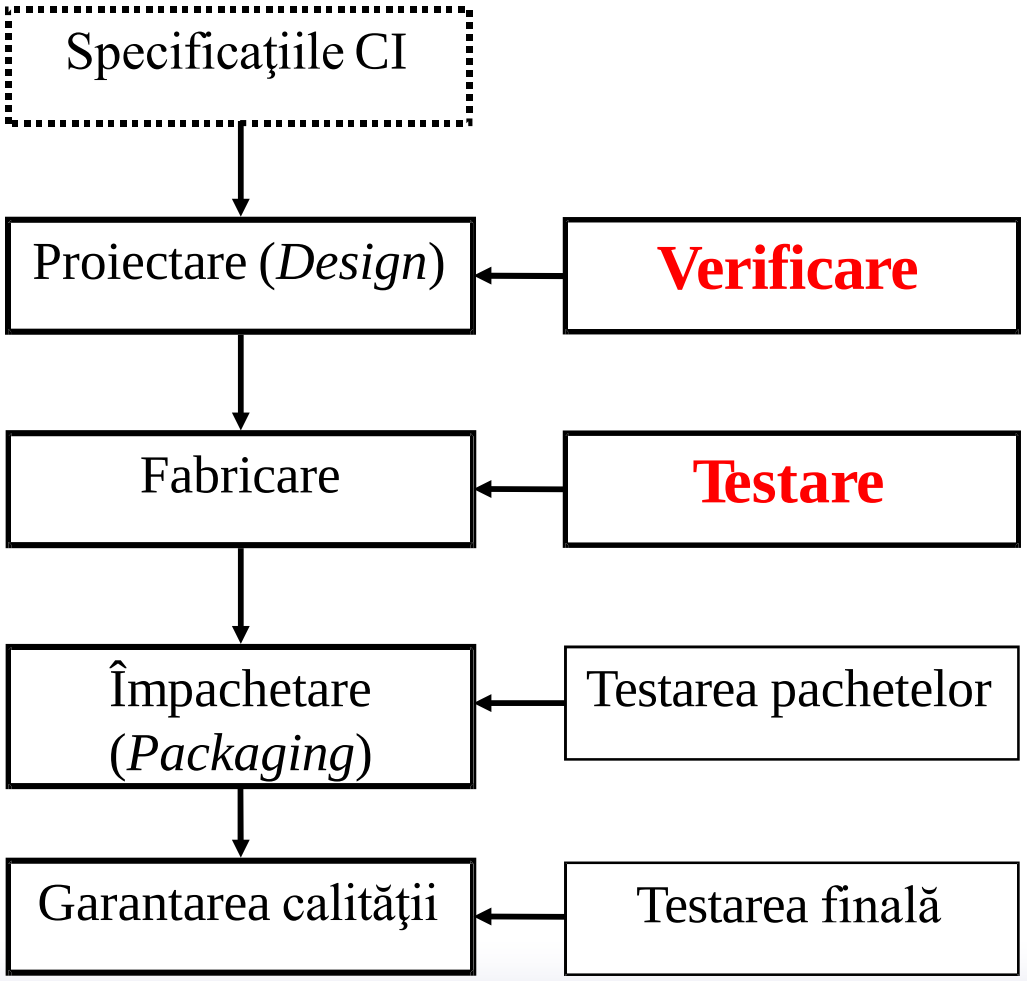
Testarea în Cursul Duratei de Viaţă a unui CI

- Testarea constă în:
  - Aplicarea unui set de stimuli de intrare asupra
  - Intrărilor IC numit *circuit under test* (CUT) şi
  - Analiza răspunsurilor circuitului testat
    - Dacă este incorect (eroare), CUT este defect
    - Dacă este corect (fără eroare), CUT nu are defecte



Testarea în Contextul Realizării unui CI

- Verificarea circuitului detectează erorile de proiectare
  - Corecţiile sunt făcute înainte de fabricare
- Testarea circuitului detectează erorile de fabricaţie
  - Un defect reprezintă o imperfecţiune fizică care generează funcţionare incorectă



Verificarea functionala

(urmeaza pagini in engleza)

What this course is about?

- To teach necessary concepts for tools of verification
- Describe a process for carrying out effective functional verification
- Present techniques for applying stimulus and monitoring the response of a design utilizing bus functional models
- Present the importance of behavioral modeling

Prior Knowledge

- This course focuses on functional verification of hardware design using either Verilog or System Verilog
- Expect students to have a basic knowledge of one of these languages
- Expect students to have basic understanding of digital hardware design
- Course will focus more on System Verilog

Course structure

- 5/6 weeks of theoretical knowledge
- 10 weeks of practical knowledge(Lab)
- Final course grade: 50% course, 50% laboratory (TBD)

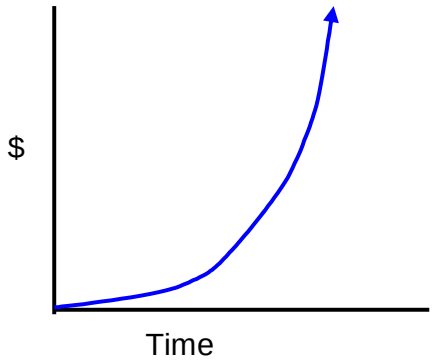
Introduction

60% – 80% of effort in digital design is dedicated to verification

Why HDL Verification?

- I mentioned 60% - 80% time spent in verification – WHY??
  - Product time-to-market
    hardware turn-around time
    volume of "bugs"
  - Development costs

Why HDL Verification? (cont)



- Cost of bugs over time
  - Longer a bug goes undetected, the more expensive it is
    - Bug found early has little cost
    - Finding a bug at chip/system has low/moderate cost
      - Could require new algorithm, which could effect schedule and cause board rework
    - Finding a bug in System Test requires new ‘spin’ of a chip
      - Requires more debug time and isolation time
    - Finding bug in customer’s environment can cost hundreds of millions and worst of all - Reputation

Why HDL Verification? (cont)

- Thomas Nicely, a professor of mathematics, had written code to enumerate primes. Nicely noticed some inconsistencies in the calculations on June 13, 1994, shortly after adding a Pentium system to his group of computers. On October 24, 1994, he reported the issu Intel.

  4.195.835 / 3.145.727 = 1,333820449136241 0
  4.195.835 / 3.145.727 = 1,333739068902037 5

- On January 17, 1995, Intel announced "a pre-tax charge of $475 million against earnings, ostensibly the total cost associated with replacement of the flawed processors.

What is Verification?

What is Verification?

Verification is a process used to demonstrate the functional correctness of a design.

Verification Challenge

- How do we know that a design is correct?
- How do we know that the design behaves as expected?
- How do we know we have checked everything?
- How do we deal with size increases of designs faster than tools performance?
- How do we get correct Hardware for the first delivery?

Importance of Verification

- Most books focus on syntax, semantics and RTL subset
  - Given the amount of literature on writing synthesizable code vs.. writing verification testbenches, one would think that the former is a more daunting task.
  - Experience proves otherwise.
- 70% of design effort goes to verification
  - Properly staffed design teams have dedicated verification engineers.
  - Verification Engineers usually outweigh designers 2-1
- 80% of all written code is in the verification environment

Importance of Verification

Verification is on critical path

What is being verified?

- Choosing a common origin and reconvergence points determines what is being verified and what type of method to use.
- Following types of verification all have different origin and reconvergence points:
  - Formal Verification
  - Model Checking
  - Functional Verification

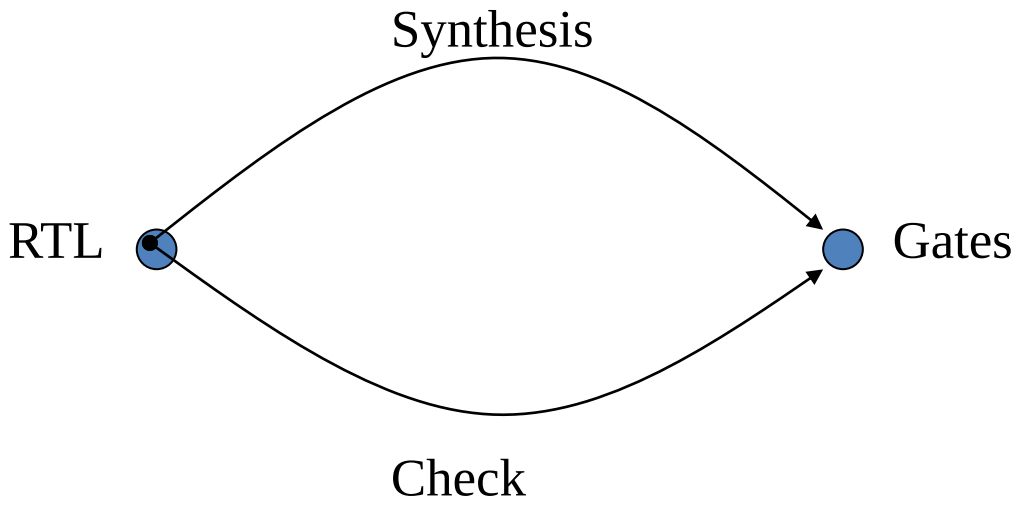
Formal Verification

- Once the end points of formal verification reconvergence paths are understood, then you know exactly what is being verified.
- 2 Types of Formal:
  - Equivalence
  - Model Checking

Equivalence Checking

- Compares two models to see if equivalence
- Proves mathematically that the origin and output are logically equivalent
- Examples:
  - RTL to Gates (Post Synthesis)
  - Post Synthesis Gates to Post PD Gates

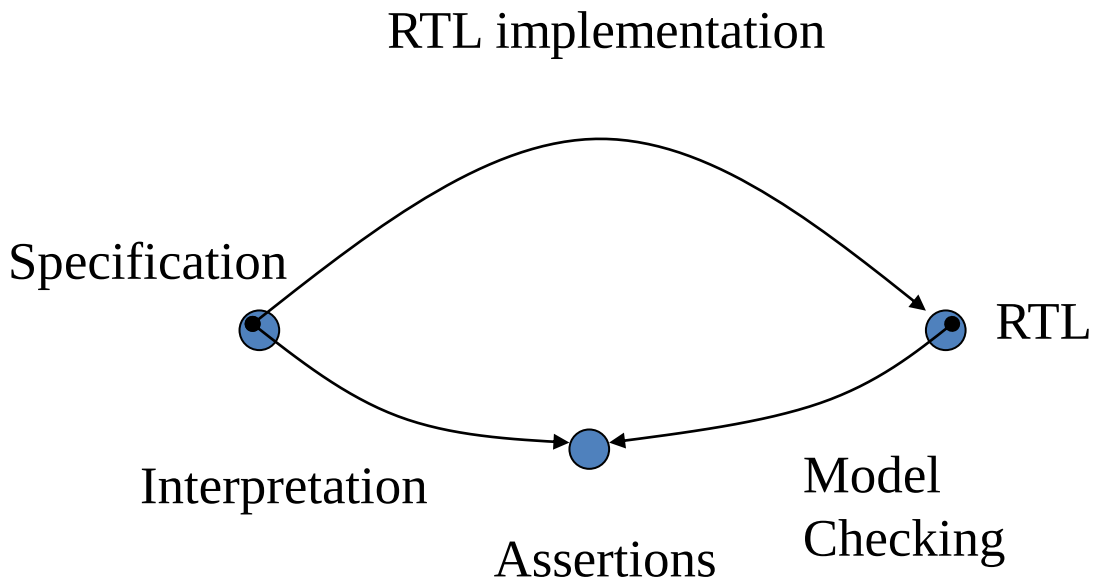
Equivalence Reconvergence Model



Model Checking

- Form of formal verification
- Characteristics of a design are formally proven or disproved
- Looks for generic problems or violations of user defined rules about the behavior of the design

Model Checking Reconvergence Model



Functional Verification

- Verifies design intent
  - Without, one must trust that the transformation of a specification to RTL was performed correctly
- Prove presence of bugs, but cannot prove their absence

Functional Reconvergence Model