EGC 455

SOC Design &Verification

Functional Verification of Hardware

As presented by IBM

Baback Izadi
Division of Engineering Programs
bai@engr.newpaltz.edu

1

Speakers

- IBM-Z Hardware Verification
  - Luke Buschmann – Unit Verification Co-lead
  - Shaun Uldrikis – Core Verification Co-lead

SUNY – New Paltz
Elect. & Comp. Eng.

2

References

- *Writing Testbenches using SystemVerilog (2006)*
  By Janick Bergeron
- Slide decks
  - John Goss (IBM)
  - Gerrit Koch (IBM)
  - Bruce Wile (IBM)

SUNY – New Paltz
Elect. & Comp. Eng.

3

Outline

- Verification overview
  - Definition
  - Why is it necessary
  - Block diagrams
  - Verification vs. Testing
- Types of verification
  - Functional vs Formal
- Verification in industry
  - Industry examples

SUNY – New Paltz
Elect. & Comp. Eng.

4

What is Verification?

- Verification is a process used to demonstrate the functional correctness of a design.
  - Does it work as defined in the spec?
  - What happens if something goes wrong? (ie. bad inputs, or internal problem)
- Primarily, verification happens before the design is fabricated, utilizing a software model of the product
- Also called logic verification or simulation.

SUNY – New Paltz
Elect. & Comp. Eng.

5

What is Verification? (cont.)

An electric car design

- Does the regenerative braking controls work under all conditions?
- If you turn on the radio, does it go to the last setting?
- What if I use the gas and brake at the same time?

Cell phone

- Can it connect to all versions of Wifi?
- Do multiple apps work at the same time? Any limit?
- If you get many texts at once, will they all be received?

SUNY – New Paltz
Elect. & Comp. Eng.

6

What is Verification? (cont.)

An electric car design

- Does the regenerative braking controls work under all conditions?
- If you turn on the radio, does it go to the last setting?
- Does the display board work?

Cell phone

- Can it connect to all versions of Wifi?
- Do multiple apps work at the same time? Any limit?
- If you get many texts at once, will they all be received?

All Tested using software design models, and Verification environments

SUNY – New Paltz
Elect. & Comp. Eng.

7

Importance of verification

- Most books focus on syntax, semantics and **register transfer level** (RTL) subset
  - Given the amount of literature on writing synthesizeable code vs writing verification testbenches, one would think that the former is a more daunting task. Experience proves otherwise.

Fabricating a chip is EASY

Fabricating a chip that works as designed is HARD

- 70% of design effort goes to verification
  - Properly staffed design teams have **dedicated** verification engineers.
  - Verification Engineers usually outweigh designers 2-1 (ideally)
- 80% of all written code is in the verification environment

SUNY – New Paltz
Elect. & Comp. Eng.

8

Importance of verification 2

- Objective: Deliver products and make money
- Cost of bugs over time: Longer a bug goes undetected, the more expensive it is
  - $ Bug found early (designer sim) has little cost
  - $$ Finding a bug in chip/system has moderate cost
    - ✓ Requires more debug time and isolation time
    - ✓ Could require new algorithm, which could effect schedule and cause circuit board rework
  - $$$$$ Finding a bug in System Test (test floor) requires new 'spin' of a chip
  - $$$$$$$$ Finding bug in customer's environment can cost hundreds of millions and worst of all - Reputation

SUNY – New Paltz
Elect. & Comp. Eng.

9

Importance of verification 3

- Economics
  - Product time-to-market
    - Hardware turn-around time
    - Volume of "bugs"
  - Development costs
    - "Early User Hardware" (EUH)
- Impact of a functioning product
  - That satellite needs to work for immediate scientific research
- Impact of a malfunctioning product
  - That car must operate correctly, or someone could get injured.

SUNY – New Paltz
Elect. & Comp. Eng.

10

Why is verification hard?

60% - 80% time spent in verification – WHY??

Why can't you just test all combinations of inputs and state?

Blame Math

Example:

- 800: X86-64 Architecture has about 800 instruction types, not including variants (functional variations per instr.)
- 19: AMD ZEN2 architecture has 19 pipeline stages
- How many different combinations of those insns could happen, in a pipeline that deep? insn ^ pipeline depth
- 800^19 ~= **1,400,000,000,000,000 * 10^40**
- This doesn't account for residual states tracked by the CPU

SUNY – New Paltz
Elect. & Comp. Eng.

11

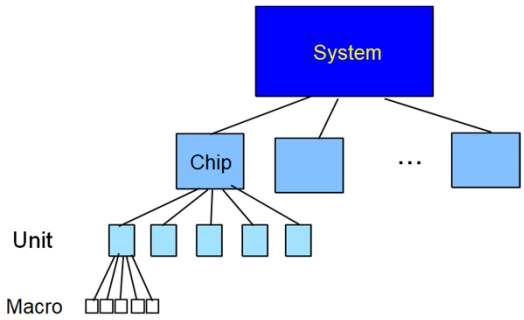
Attempting to verify a large state space

- Divide and conquer
  - Designer Sim, Unit Sim
  - Element Sim, Chip/System Sim
- Multiple verification types
  - Functional Simulation (Event or Cycle Sim)
  - Formal Verification
- Coverage Metrics
  - Tracking occurrences of model states (events)
  - Frequency of defects found

SUNY – New Paltz
Elect. & Comp. Eng.

12

Hierarchical Design



- Allows design team to break system down into logical and comprehensible components.
- Also allows for repeatable components.

SUNY – New Paltz
Elect. & Comp. Eng.

13

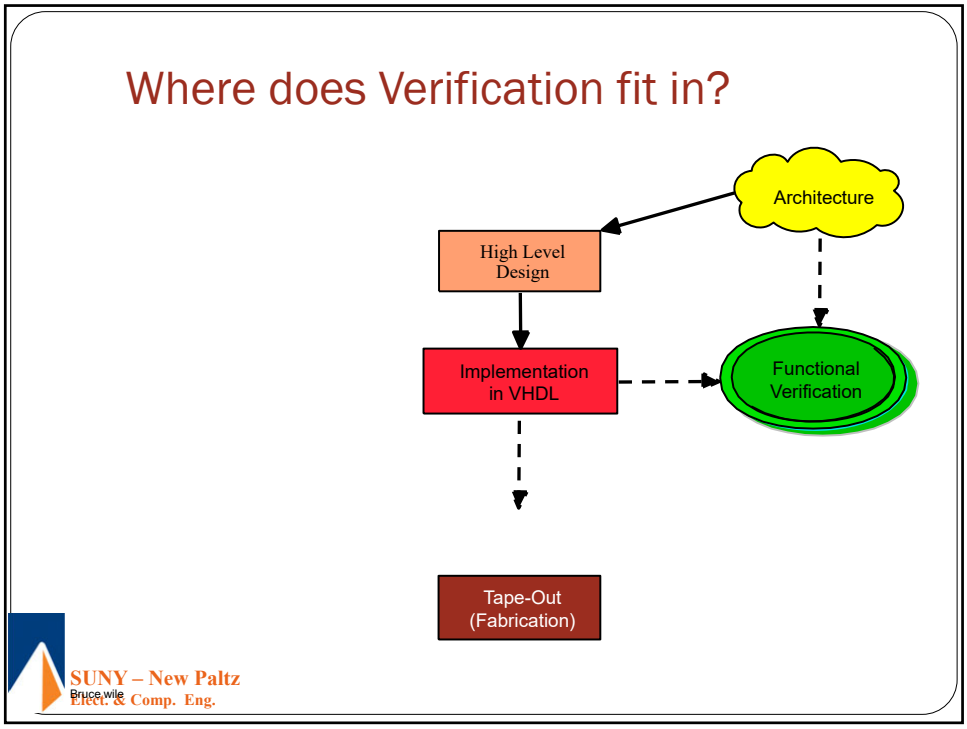
Divide and Conquer

Verify smaller blocks of logic, working up to larger environments

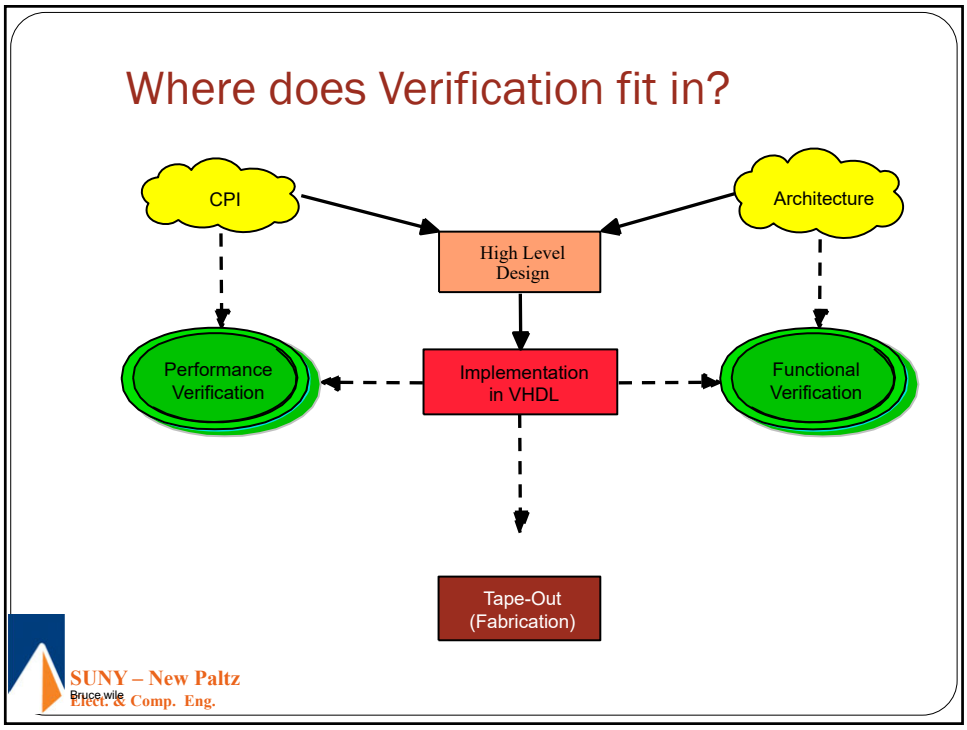
1. Designer Sim – small env testing 1 (or a few) HDL files
2. Unit Sim – Small env, stressing several design files together. Typically divided from other units by clearly defined hardware interfaces.
3. Element Sim – Larger env, including several units. This validates the interaction between units.
4. Chip / System Sim – Model including all design files. Large and slow, but closely represents the end product.

SUNY – New Paltz
Elect. & Comp. Eng.

14

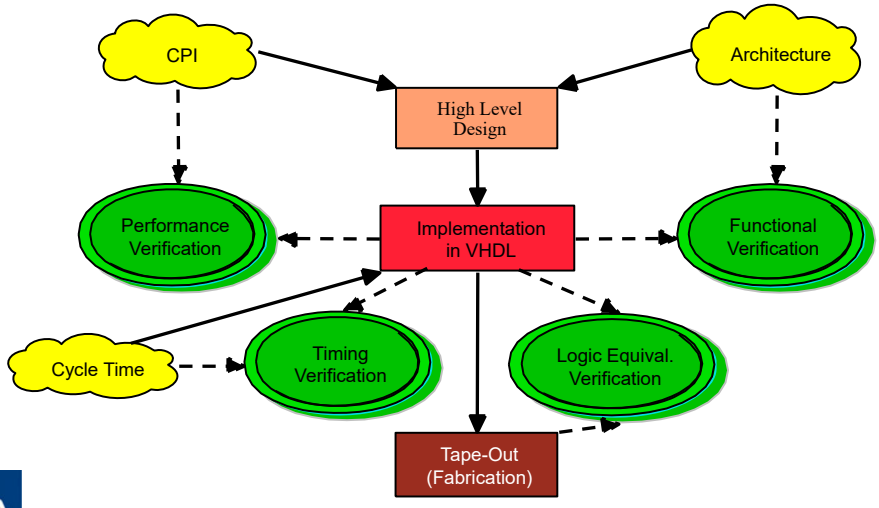

Where does Verification fit in?
Architecture
High Level Design
Implementation in VHDL
Functional Verification
Tape-Out (Fabrication)
SUNY – New Paltz
Bruce.wile
Elect. & Comp. Eng.

15


Where does Verification fit in?
CPI
Architecture
High Level Design
Performance Verification
Implementation in VHDL
Functional Verification
Tape-Out (Fabrication)
SUNY – New Paltz
Bruce.wile
Elect. & Comp. Eng.

16

Where does Verification fit in?



SUNY – New Paltz
Bruce.wile
Elect. & Comp. Eng.

17

Result of all the Above

- Verification is on the Critical Path
- Need to minimize verification time!
- How? By using special tools and methodologies

SUNY – New Paltz
Elect. & Comp. Eng.

18

When is verification complete?

- Never truly done on complex designs
  - Verification can only show presence of errors, not their absence
  - Given enough time, errors will be uncovered
- Question – Is the error likely to be severe enough to warrant the effort spent to find the error?
- Verification is similar to statistical hypothesis.
  - Hypothesis – Is the design functionally correct?

Metrics are necessary for the Verification Team to sign-off, and agree the design is ready for the fab.

SUNY – New Paltz
Elect. & Comp. Eng.

19

Verification vs. Testing

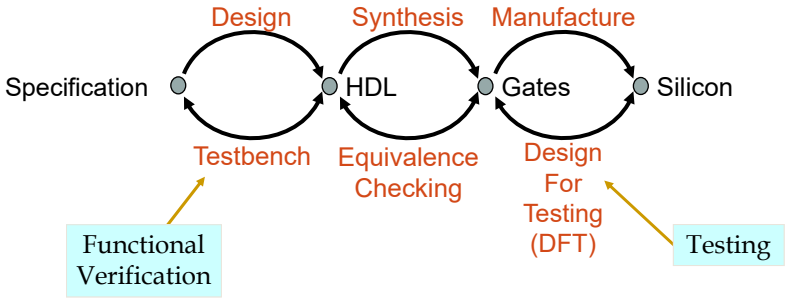
The two often confused

- Purpose of *test* is to verify that the design was manufactured properly (Quality Assurance)
- *Verification* is to ensure that the design meets the intended functionality

SUNY – New Paltz
Elect. & Comp. Eng.

20

Verification vs. Testing



SUNY – New Paltz
Elect. & Comp. Eng.

21

What this course is about?

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

SUNY – New Paltz
Elect. & Comp. Eng.

22

Verification challenges

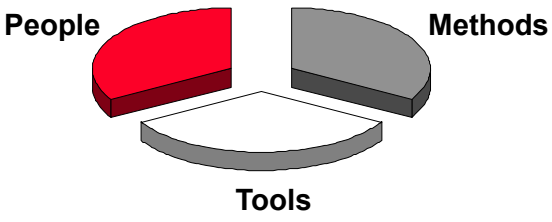
- How do we know that a design is correct?
- How do we know that the design behaves as expected?
- How do we know we have checked everything?
- How do we deal with size increases of designs faster than tools performance?
- How do we get correct hardware for the first tape out?

SUNY – New Paltz
Elect. & Comp. Eng.

23

Verification process

- Involves

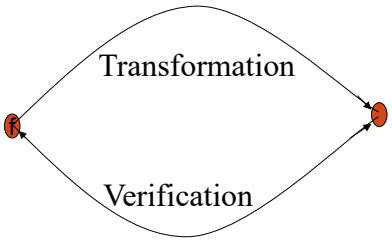


SUNY – New Paltz
Elect. & Comp. Eng.

24

Reconvergence model

- Conceptual representation of the verification process
- The purpose of verification is to ensure that the result of some transformation is as intended or expected.
- **Verification can only be accomplished through a re-convergent path to a common source**
  - A written specification is interpreted and written into RTL code



SUNY – New Paltz
Elect. & Comp. Eng.

25

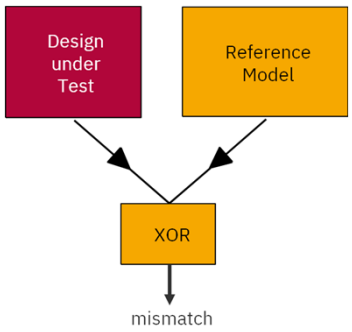
What is being verified?

- Choosing 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 & verify different things:
  - Formal Verification
  - Functional Verification
    - Testbench Generators
    - Constrained Random

SUNY – New Paltz
Elect. & Comp. Eng.

26

Formal verification



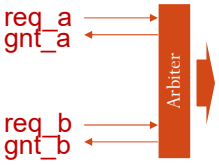
- Formal verification tools attempt to mathematically prove correctness of properties (“assertions”)
- Can also by expressed using reference model

  // assure dut and refmodel match
  never mismatch
- Tool will attempt to find counter-example by doing state space exploration

SUNY – New Paltz
Elect. & Comp. Eng.

27

Formal verification



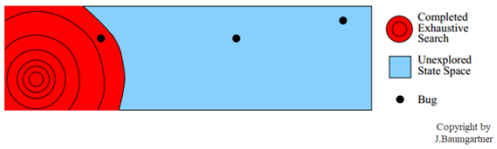
- Formal verification tools attempt to **mathematically prove correctness** of properties (“assertions”)
- Expressed using specific languages such as PSL (Property Specification Language) or SVA (SystemVerilog Assertions)
- ```
  // assure arbitration liveliness
  (no deadlocks)
  always (req_a -> eventually! gnt_a)
  ```
- Tool will attempt to **find counter-example** by doing state space exploration

SUNY – New Paltz
Elect. & Comp. Eng.

28
  ```

State Space Exploration

"Tool will attempt to **find counter-example** by doing state space exploration"

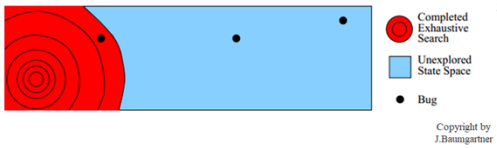


SUNY – New Paltz
Elect. & Comp. Eng.

29

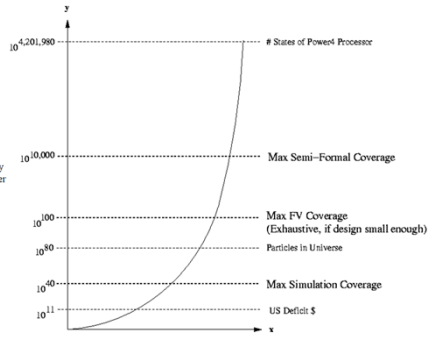
State Space Exploration

"Tool will attempt to **find counter-example** by doing state space exploration"



State space quickly grows beyond boundary of what's computationally possible
"**State Space Explosion**"

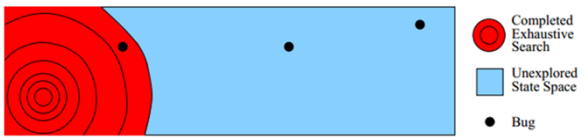
Application of formal verification **methods limited to smaller designs** or dedicated components of larger designs or requires design abstraction (such as limiting width of operands…)



SUNY – New Paltz
Elect. & Comp. Eng.

30

Formal vs. Functional Coverage Comparison

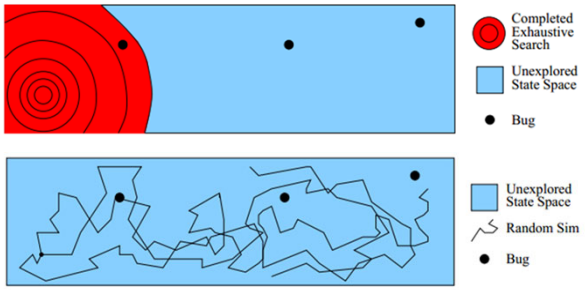


formal verification

SUNY – New Paltz
Elect. & Comp. Eng.

31

Formal vs. Functional Coverage Comparison



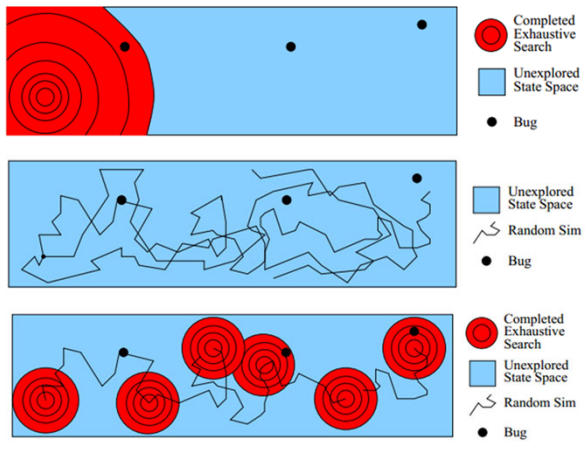
formal verification

(constrained) random verification

SUNY – New Paltz
Elect. & Comp. Eng.

32

Formal vs. Functional Coverage Comparison



formal verification

(constrained) random verification

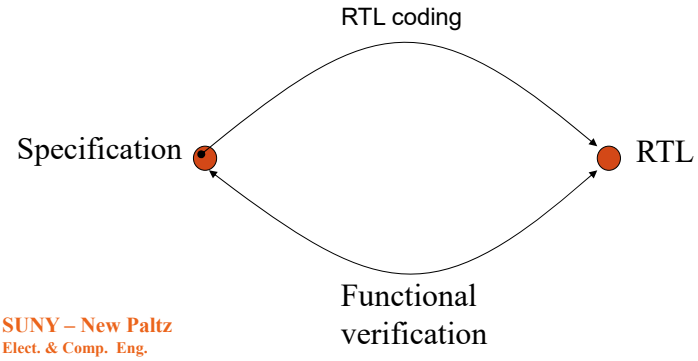
semi-formal verification

SUNY – New Paltz
Elect. & Comp. Eng.

33

Functional verification

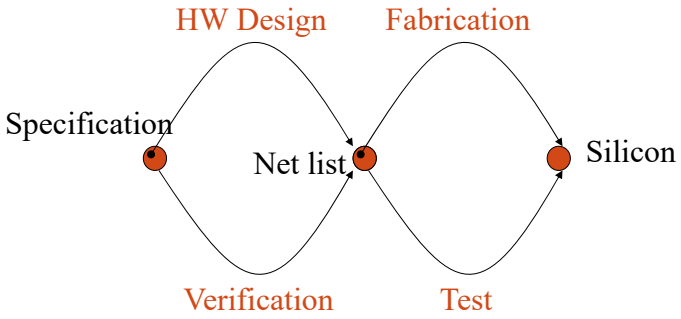
- Verifies design intent
  - Without, one must trust that the transformation of a specification to RTL was performed correctly
- Prove presence of bugs, but cannot prove their absence



SUNY – New Paltz
Elect. & Comp. Eng.

34

Verification and test reconvergence model

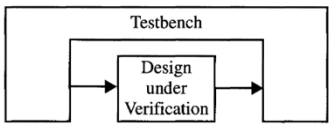


SUNY – New Paltz
Elect. & Comp. Eng.

35

What is a testbench?

- A "testbench" usually refers to the code used to create a pre-determined input sequence to a design, then optionally observe the response.
  - Generic term used differently across industry
- Completely closed system
  - No inputs or outputs out of the whole system
  - effectively a model of the universe as far as the design is concerned.
- Verification challenge:
  - What input patterns to supply to the Design Under Verification (DUV) and what is ***expected*** for the output for a ***properly working design***



SUNY – New Paltz
Elect. & Comp. Eng.

36

Logic Verification Industry Perspective

SUNY – New Paltz
Elect. & Comp. Eng.

37

Server Design Process

1 Customer

2 High Level Design/ System Structure

7 Bring Up Test

High Level Chip Design

Substrate/MCM Design

VHDL Chip Implementation

Microcode Design

Service Code Design

Logic Verification

Chip Timing, Circuit Design, Integration

6 Manufacturing

3 Chip Design Team

You are here

4 Hardware Fabrication

5 Systems Test

SUNY – New Paltz
Elect. & Comp. Eng.

38

Integration Levels & Verification Focus

	Integration Level	Verification Focus
	Macro Level / Designer sim	Implementation Algorithm
	Unit Level	Microarchitecture
	Core Level Element Level	Architecture Integration
	Chip Level	Architecture Multi-core interaction Cache coherency
	System Level	HW / FW interaction Performance IO Interaction

SUNY – New Paltz
Elect. & Comp. Eng.

39

Typical Hardware Bugs

- Manufacturing Bug
  - Particle on wafer or mask during production
  - Chemical reaction took too long
- Timing Bug
  - Fanout too big, leads to extended rise times
- Logic Bug
  - Write to single Register Bit doesn‘t work
  - Array access only works for first 63 of 64 rows
  - While buffer full and async interrupt pending a cache miss is initiated by wrong branch prediction
- Performance Bug
  - One of 2 issue queues is not used by the instr. dispatcher for certain opcodes

SUNY – New Paltz
Elect. & Comp. Eng.

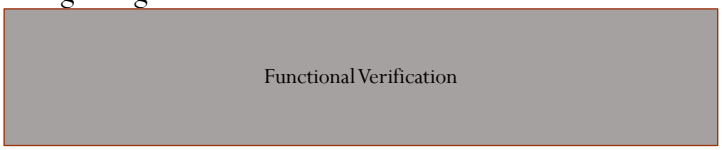
40

Typical Hardware Bugs

- Manufacturing Bug

- Timing Bug

- Logic Bug



- Performance Bug

SUNY – New Paltz
Elect. & Comp. Eng.

41

What a great time to be an engineer!

- Exciting work
- Major effect on culture
- Compensation
  - Industry's word for "money, morale, and benefits"

SUNY – New Paltz
Elect. & Comp. Eng.

42

Why all the big bucks?

- Basic business principle:
  Company that gets a product to the market first gets an inordinate share of the market revenue

SUNY – New Paltz
Elect. & Comp. Eng.

43

Triple Constraints

- Schedule
- Costs
- Quality

SUNY – New Paltz
Elect. & Comp. Eng.

44

Why is verification so important to the chip industry?

- Verification is the single biggest lever to positively effect the triple constraints
  - Fewer revs through the fabrication process means lower costs and faster time-to-market
  - Re-spinning a chip costs:
    - Hundreds of thousands of dollars
    - 6-8 weeks
- So if you can get it right in fewer "passes", you WIN!!!

SUNY – New Paltz
Elect. & Comp. Eng.

45

Biggest challenges are in Verification

- Circuit design process has been "fixed"
- Industry-wide shortage of "good" verification engineers

SUNY – New Paltz
Elect. & Comp. Eng.

46

Famous Examples of Failed Verification

- Intel FP Divide bug
  - Problem uncovered on Pentium chips in 1994
  - Bug discovered by math professor investigating mathematical theory
  - FP Unit returns erroneous values for certain digits beyond the 8th significant digit
  - Despite 1 in a million users ever being affected, confidence in Intel dropped
  - Opened the door for AMD market growth
  - NASA's Jet Propulsion Lab was worried that the Pentium chip would cause errors.....

SUNY – New Paltz
Elect. & Comp. Eng.

47

Famous Examples of Failed Verification

- NASA.....
  - 1999 Mars Climate Orbiter
  - System level verification failure caused Orbiter to fly too close to Mars atmosphere and burn up
  - Problem was a mix up between metric and English units of measurement causing a miscalculation in trajectory

SUNY – New Paltz
Elect. & Comp. Eng.

48