Testability of VLSI

Lecture 1: Introduction to VLSI Testing

By Dr. Sanjay Vidhyadharan

Why Testing is Important?

1994
Prof. Thomas Nicely reports bug in Pentium Restoring Division
Logic error not caught until > 1M units shipped
Recall cost $450M (!!!)

1997-2000
All major micro-processor manufacturers adopt formal verification.

Requirement of Testing

Verification

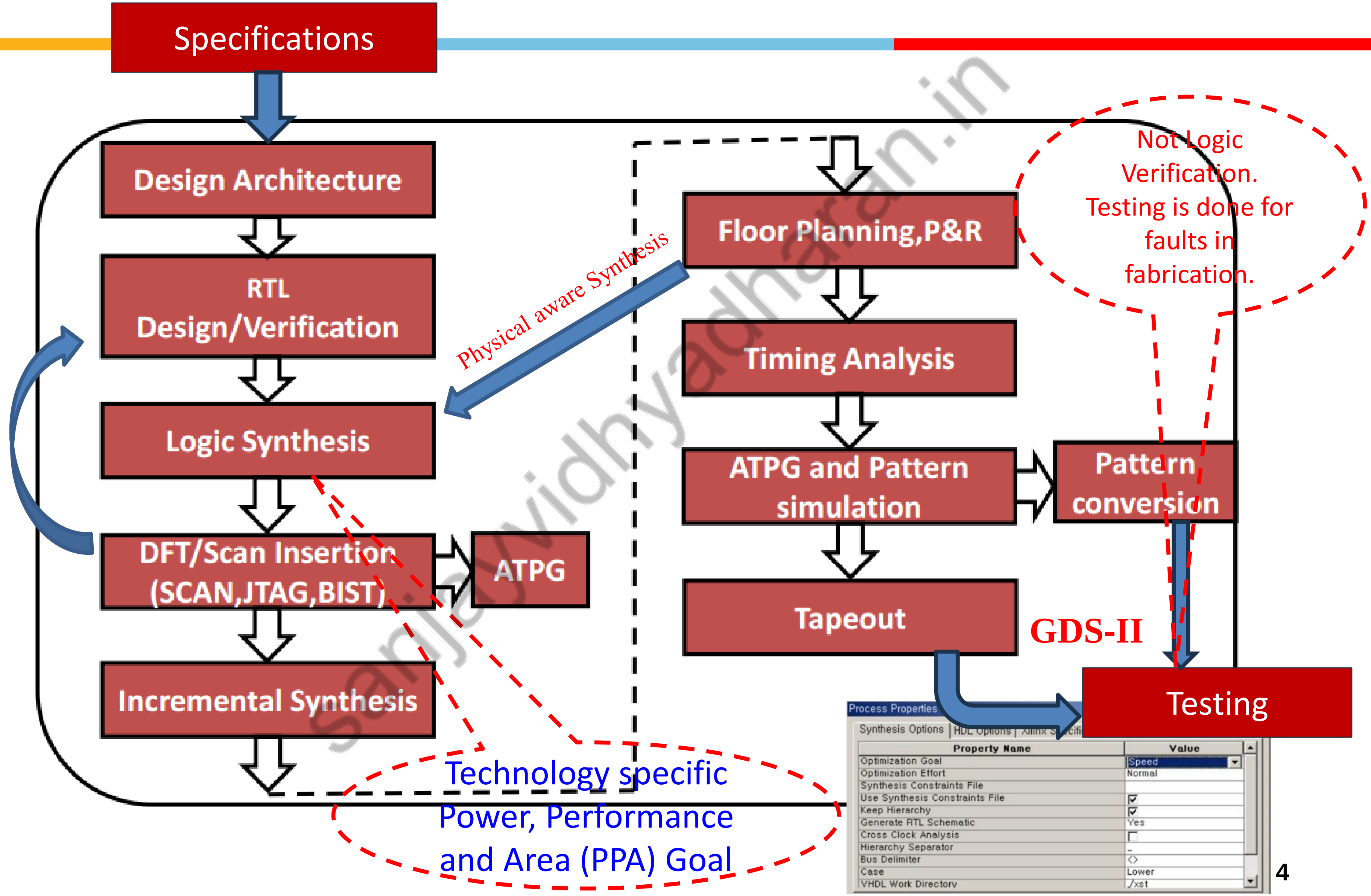
- Verifies correctness of design.
- Performed by simulation, hardware emulation, or formal methods.
- Performed prior to manufacturing.
- Responsible for quality of design.
- No limit on number of test points/test vectors

Testing

- Verifies correctness of manufactured hardware.
- Test generation: software process executed once during design
- Test application: electrical tests applied to hardware on every manufactured device.
- Responsible for quality of devices.
- Limited on number of test points/test vectors based on the I/O pins

- Manufacturing Defects : IC processing/ packaging (Nano Scale Devices ↑↑ Defects)
- PCB assembly and wiring errors
- Environment, Temperature, Humidity, Vibration
- Power supply fluctuations
- Wear and Tear : friction, corrosion

ASIC Design Flow

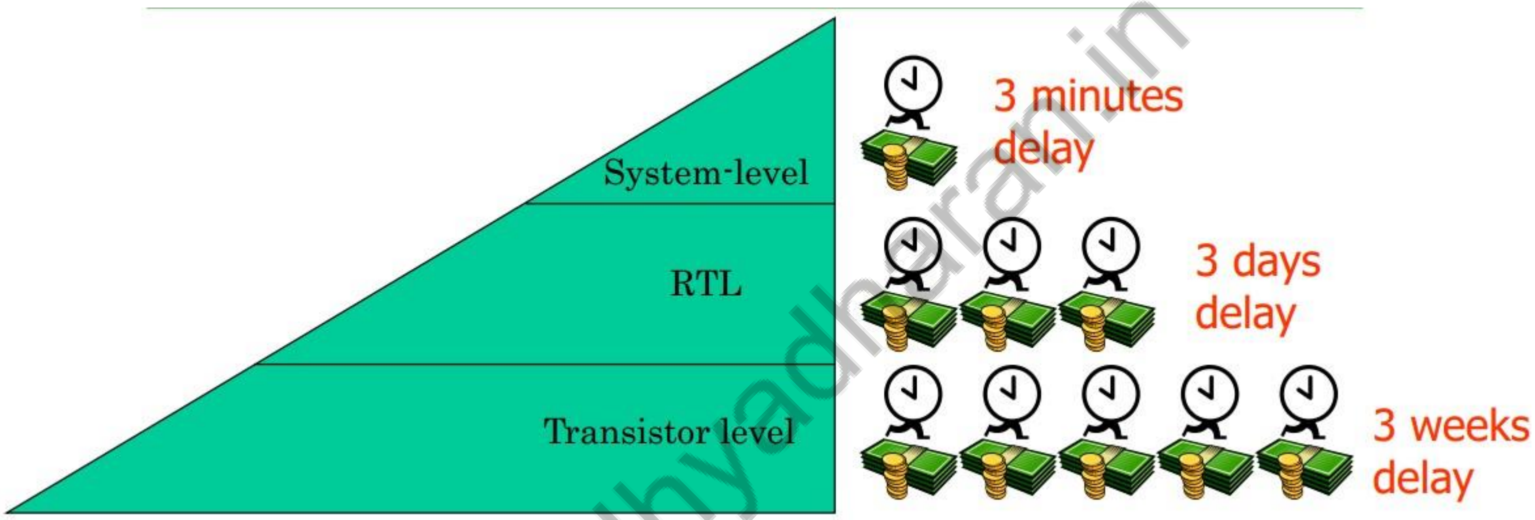


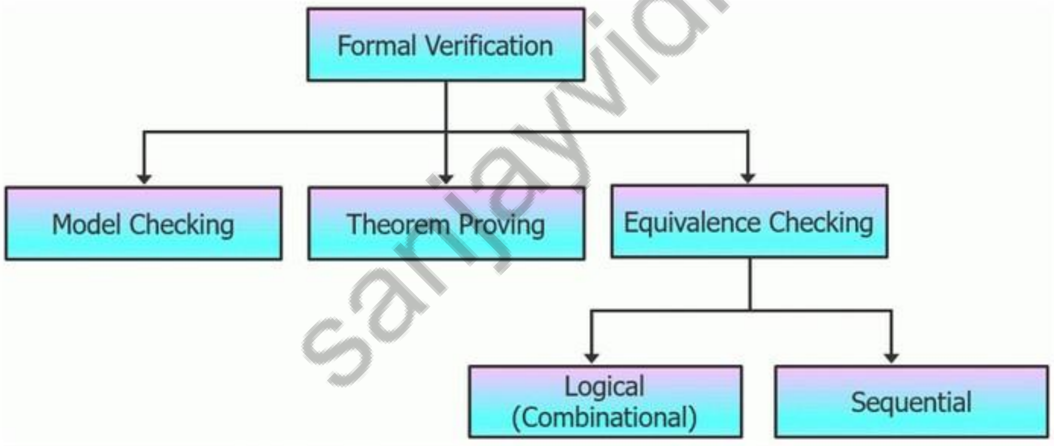
Not Logic Verification. Testing is done for faults in fabrication.

Technology specific Power, Performance and Area (PPA) Goal

Property Name	Value
Optimization Goal	Speed
Optimization Effort	Normal
Synthesis Constraints File	
Use Synthesis Constraints File	☑
Keep Hierarchy	☑
Generate RTL Schematic	Yes
Cross Clock Analysis	□
Hierarchy Separator	_
Bus Delimiter	<>
Case	Lower
VHDL Work Directory	./xst

Formal Verification





Formal verification techniques (Source: Aijaz Fatima)

Formal Verification

Formal verification

- Used in different Pre-fabrication stages in ASIC project
  - Two Types
    1.Formal Equivalence Checking
    2.Formal Property Checking

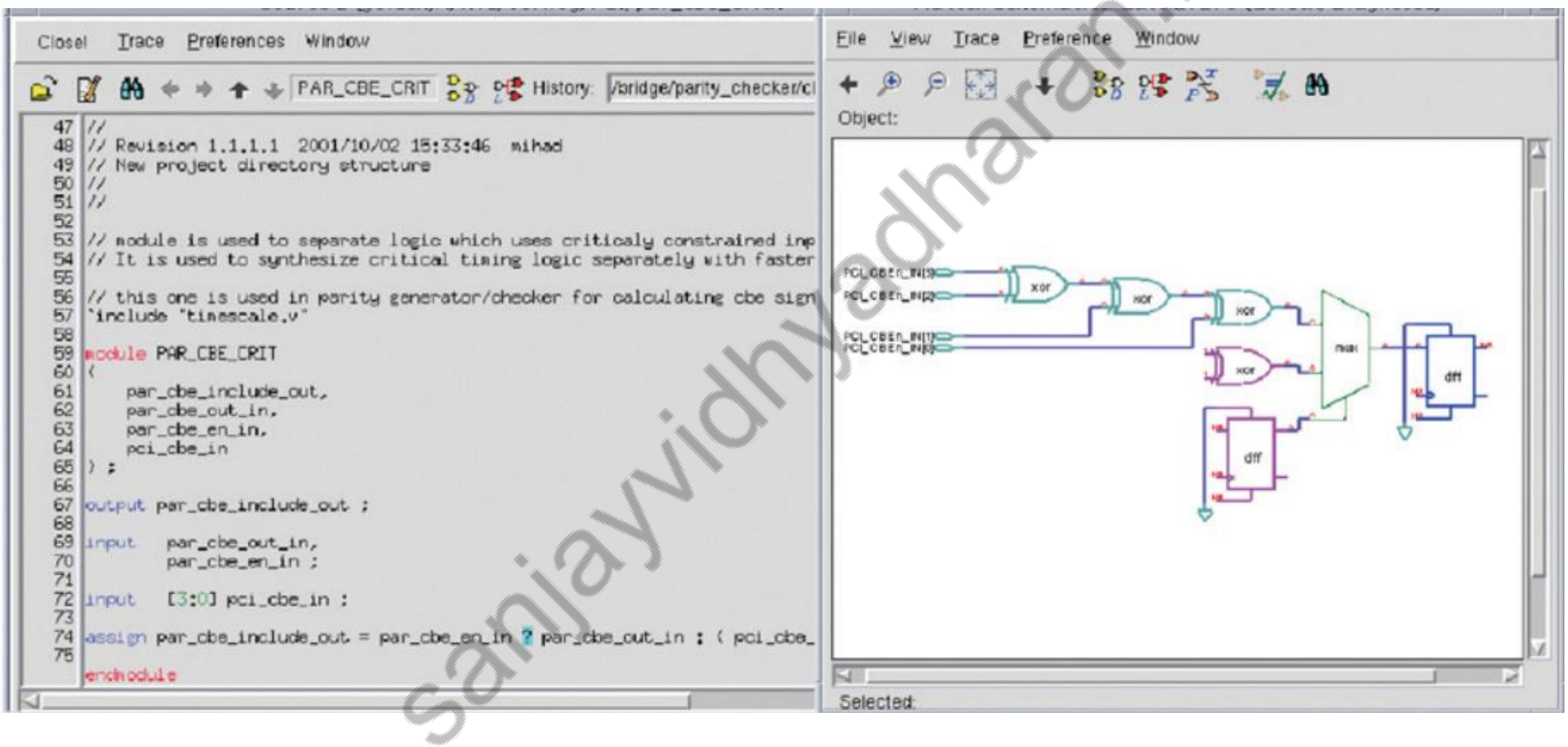
- Formal Equivalence Checking
  1.RTL vs Pre-Routed Netlist
  2.Pre-Routed Netlist vs Post Routed Netlist
  3.Netlist Vs ECO-Netlist (functional engineering change order (ECO) for optimization)

Cadence (Conformal LEC) and Synopsys (Formality).

RTL to gate-level netlist conversion is done using our synthesis tool called Genus.
Synthesized netlist can be imported using Cadence Composer

Formal Verification

- Formal Equivalence Checking

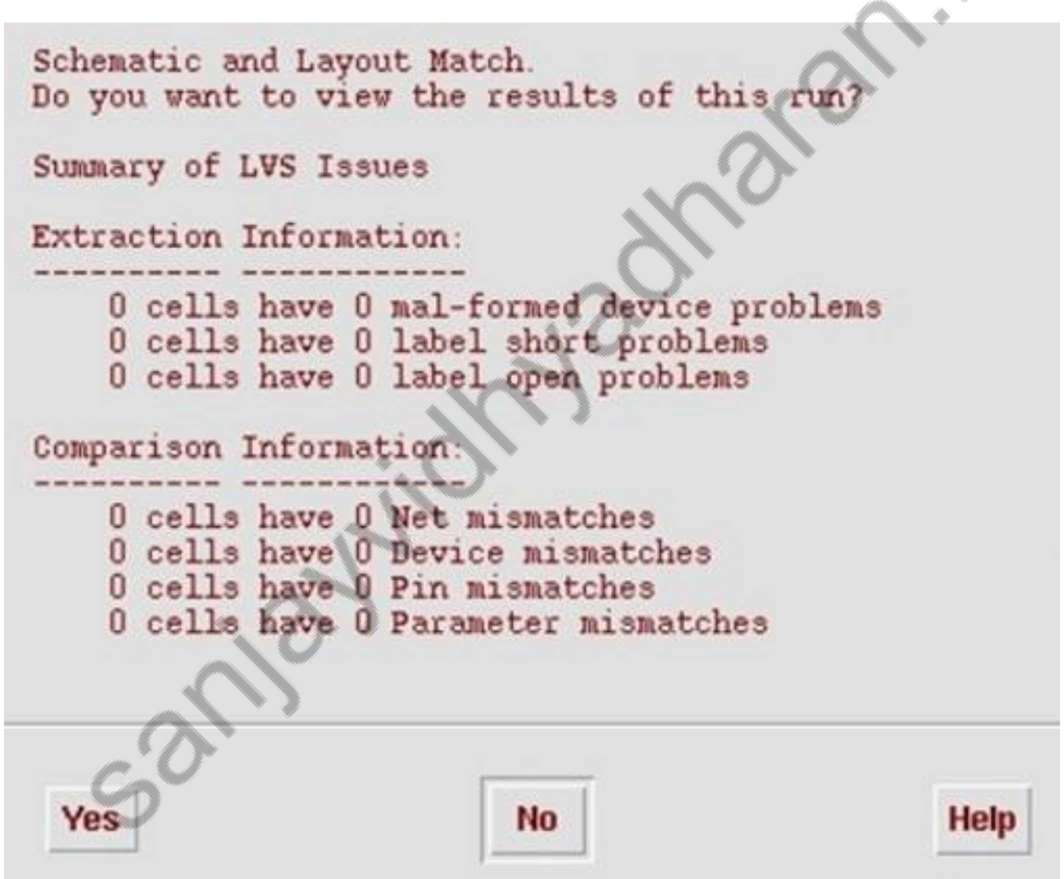


```
//
// Revision 1.1.1.1  2001/10/02 15:33:46  mihad
// New project directory structure
//
//

// module is used to separate logic which uses criticaly constrained inp
// It is used to synthesize critical timing logic separately with faster

// this one is used in parity generator/checker for calculating cbe sign
`include "timescale.v"

module PAR_CBE_CRIT
(
    par_cbe_include_out,
    par_cbe_out_in,
    par_cbe_en_in,
    pci_cbe_in
) ;

output par_cbe_include_out ;

input   par_cbe_out_in,
        par_cbe_en_in ;

input   [3:0] pci_cbe_in ;

assign par_cbe_include_out = par_cbe_en_in ? par_cbe_out_in : ( pci_cbe_

endmodule
```

1.RTL vs Pre-Routed Netlist

Formal Verification

- Formal Equivalence Checking



```
Schematic and Layout Match.
Do you want to view the results of this run?

Summary of LVS Issues

Extraction Information:
---------- -----------
    0 cells have 0 mal-formed device problems
    0 cells have 0 label short problems
    0 cells have 0 label open problems

Comparison Information:
---------- -----------
    0 cells have 0 Net mismatches
    0 cells have 0 Device mismatches
    0 cells have 0 Pin mismatches
    0 cells have 0 Parameter mismatches
```

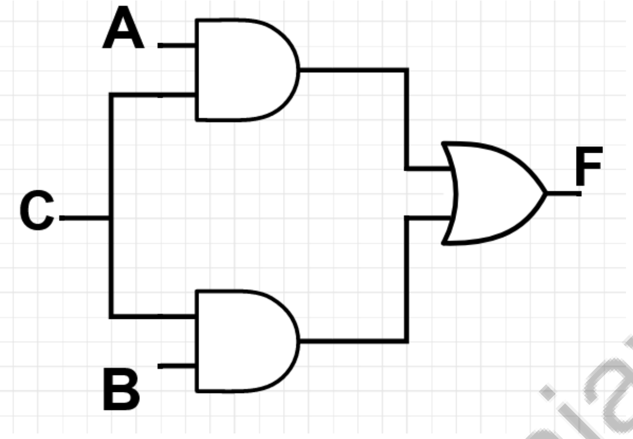
1.Pre-Routed Netlist vs Post Routed Netlist

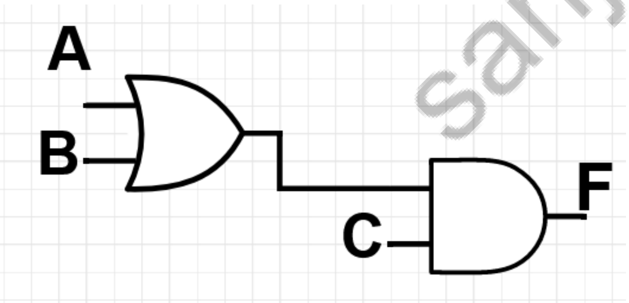
Formal Verification

- Formal Equivalence Checking

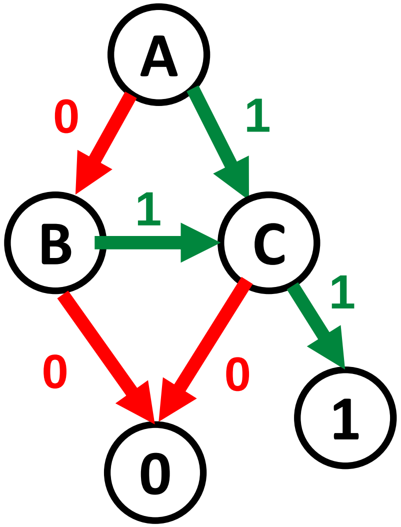
F= AC+BC

Assign F = (A&C)|(B & C)





Binary Decision Making



Formal Verification

- Two Types of Simulation
  - Exhaustive
  - Selective

- Formal Property Checking/Model Checking.

  Increased Complexity of modern-day chips makes exhaustive simulation impractical

  Formal Verification is done at abstract model, Need to have

  System Model (Behavior Model using Verilog. VHDL or Software d sing C++)

  Specifications (Property)

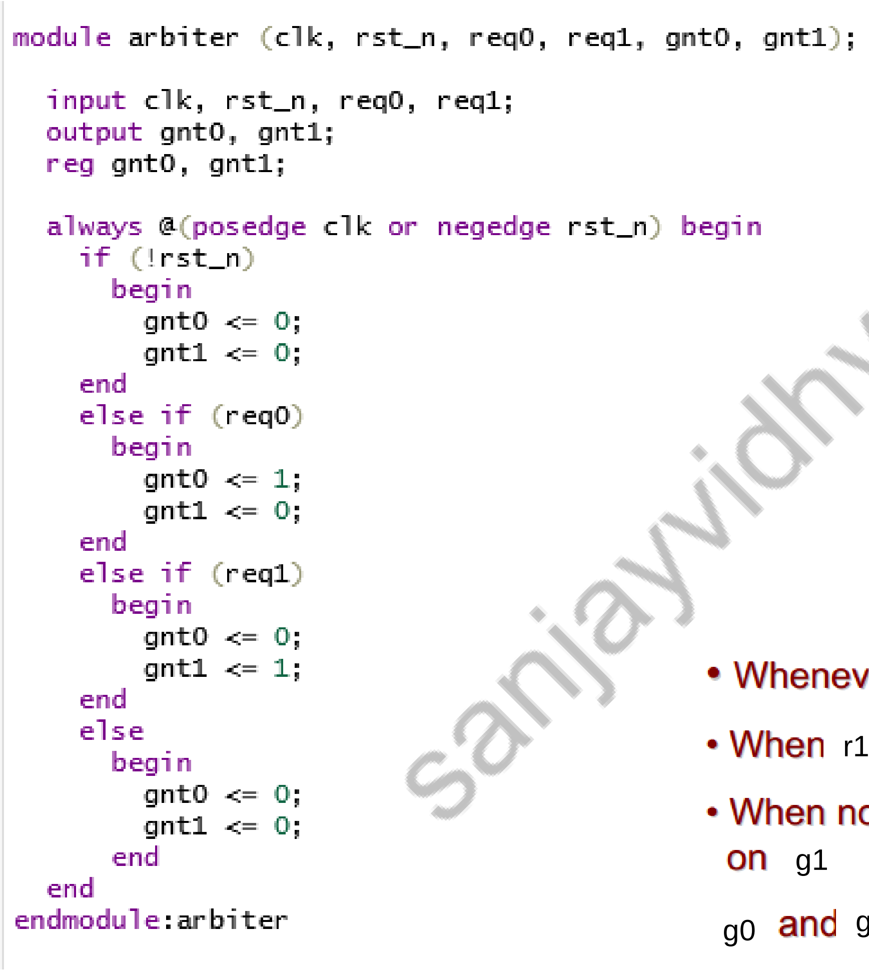
  Verification Method

System Verilog, Cadence Jasper

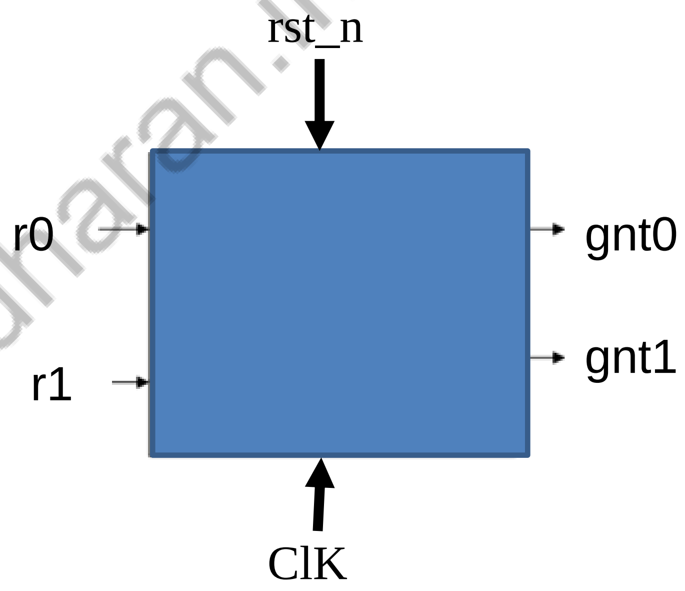
Tool takes the DUT and Assertion file as inputs

Formal Verification

- Formal Property Checking/Model Checking.



```
module arbiter (clk, rst_n, req0, req1, gnt0, gnt1);

  input clk, rst_n, req0, req1;
  output gnt0, gnt1;
  reg gnt0, gnt1;

  always @(posedge clk or negedge rst_n) begin
    if (!rst_n)
      begin
        gnt0 <= 0;
        gnt1 <= 0;
    end
    else if (req0)
      begin
        gnt0 <= 1;
        gnt1 <= 0;
    end
    else if (req1)
      begin
        gnt0 <= 0;
        gnt1 <= 1;
    end
    else
      begin
        gnt0 <= 0;
        gnt1 <= 0;
      end
  end
endmodule:arbiter
```

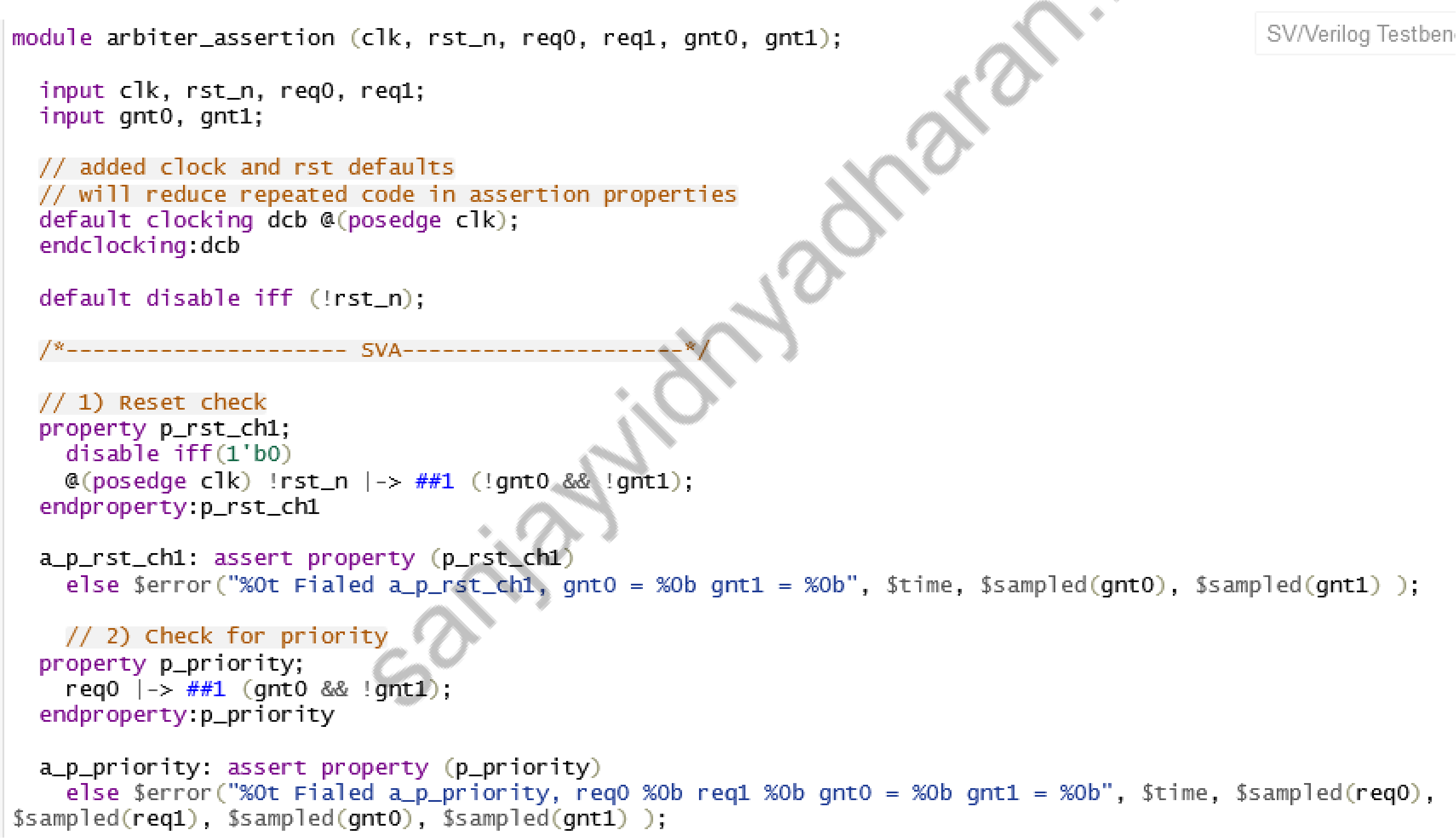


- Whenever r0 is asserted g0 is given in the next cycle
- When r1 is the sole request, g1 comes in the next cycle
- When none of them are requesting, the arbiter parks the grant on g1

g0 and g1 can not be true at the same time (mutual exclusion)

Formal Verification

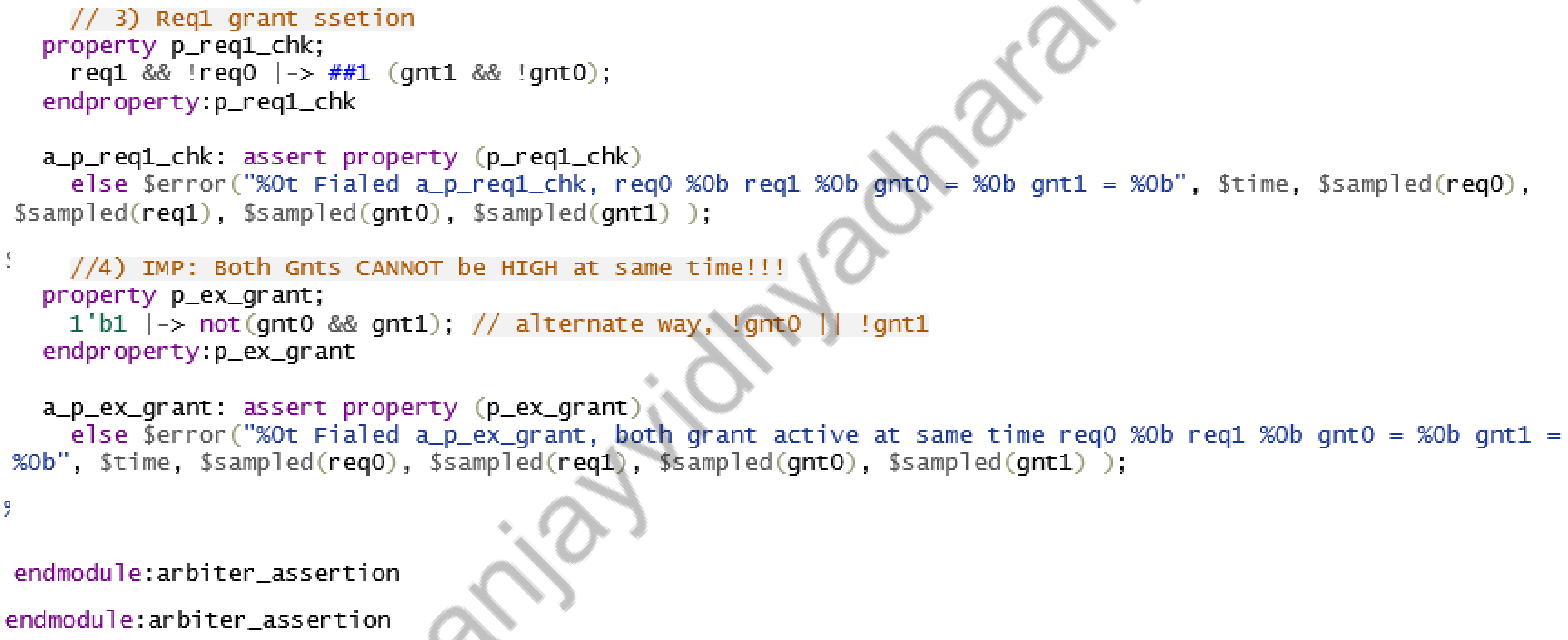
- Formal Property Checking/Model Checking.



```
module arbiter_assertion (clk, rst_n, req0, req1, gnt0, gnt1);

  input clk, rst_n, req0, req1;
  input gnt0, gnt1;

  // added clock and rst defaults
  // will reduce repeated code in assertion properties
  default clocking dcb @(posedge clk);
  endclocking:dcb

  default disable iff (!rst_n);

  /*--------------------- SVA---------------------*/

  // 1) Reset check
  property p_rst_ch1;
    disable iff(1'b0)
    @(posedge clk) !rst_n |-> ##1 (!gnt0 && !gnt1);
  endproperty:p_rst_ch1

  a_p_rst_ch1: assert property (p_rst_ch1)
    else $error("%0t Fialed a_p_rst_ch1, gnt0 = %0b gnt1 = %0b", $time, $sampled(gnt0), $sampled(gnt1) );

    // 2) Check for priority
  property p_priority;
    req0 |-> ##1 (gnt0 && !gnt1);
  endproperty:p_priority

  a_p_priority: assert property (p_priority)
    else $error("%0t Fialed a_p_priority, req0 %0b req1 %0b gnt0 = %0b gnt1 = %0b", $time, $sampled(req0),
$sampled(req1), $sampled(gnt0), $sampled(gnt1) );
```

Formal Verification

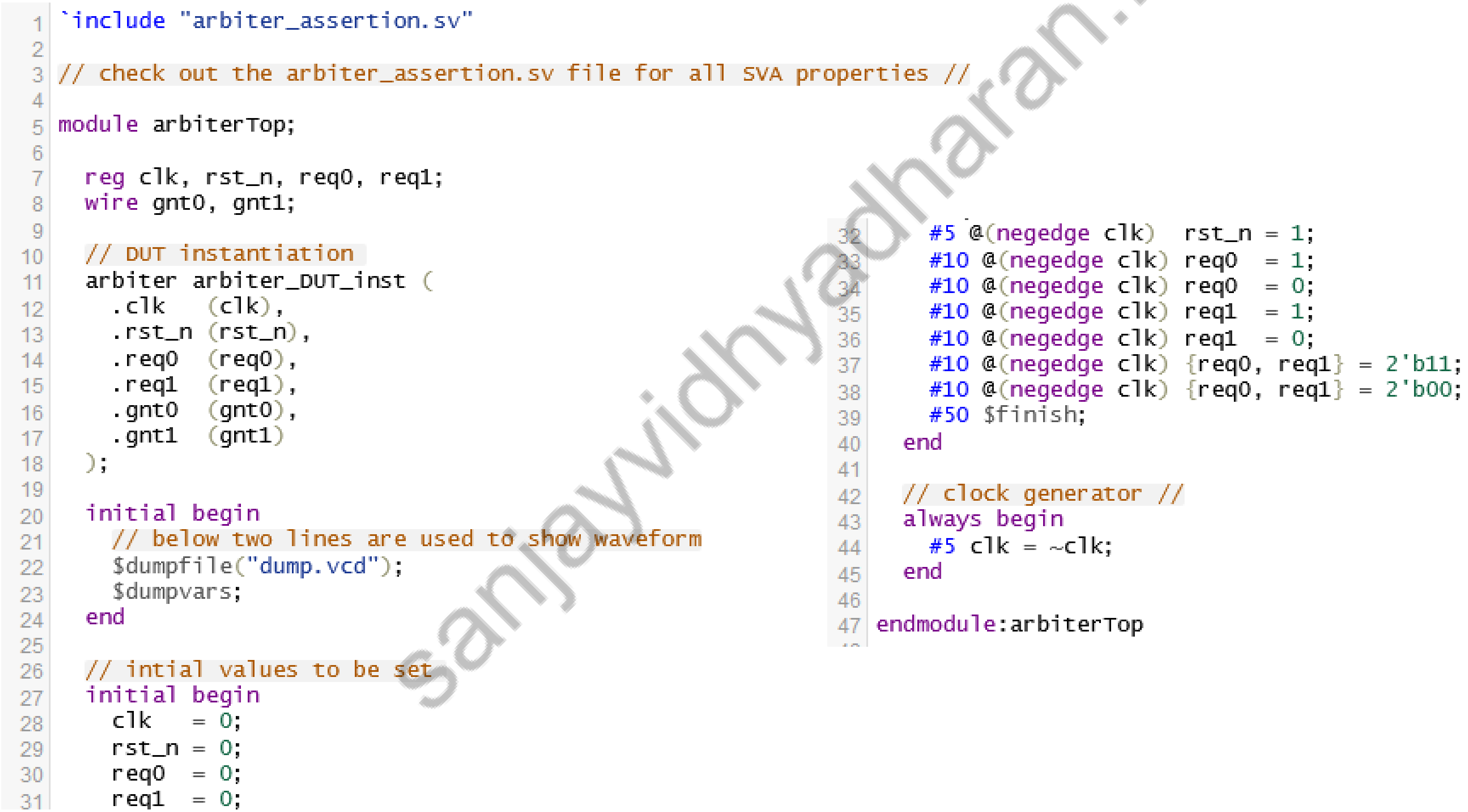
- Formal Property Checking/Model Checking.



```
    // 3) Req1 grant ssetion
  property p_req1_chk;
    req1 && !req0 |-> ##1 (gnt1 && !gnt0);
  endproperty:p_req1_chk

  a_p_req1_chk: assert property (p_req1_chk)
    else $error("%0t Fialed a_p_req1_chk, req0 %0b req1 %0b gnt0 = %0b gnt1 = %0b", $time, $sampled(req0),
$sampled(req1), $sampled(gnt0), $sampled(gnt1) );

    //4) IMP: Both Gnts CANNOT be HIGH at same time!!!
  property p_ex_grant;
    1'b1 |-> not(gnt0 && gnt1); // alternate way, !gnt0 || !gnt1
  endproperty:p_ex_grant

  a_p_ex_grant: assert property (p_ex_grant)
    else $error("%0t Fialed a_p_ex_grant, both grant active at same time req0 %0b req1 %0b gnt0 = %0b gnt1 =
%0b", $time, $sampled(req0), $sampled(req1), $sampled(gnt0), $sampled(gnt1) );


 endmodule:arbiter_assertion

endmodule:arbiter_assertion
```

Formal Verification

- Formal Property Checking/Model Checking.



```
`include "arbiter_assertion.sv"

// check out the arbiter_assertion.sv file for all SVA properties //

module arbiterTop;

  reg clk, rst_n, req0, req1;
  wire gnt0, gnt1;

  // DUT instantiation
  arbiter arbiter_DUT_inst (
    .clk    (clk),
    .rst_n  (rst_n),
    .req0   (req0),
    .req1   (req1),
    .gnt0   (gnt0),
    .gnt1   (gnt1)
  );

  initial begin
    // below two lines are used to show waveform
    $dumpfile("dump.vcd");
    $dumpvars;
  end

  // intial values to be set
  initial begin
    clk    = 0;
    rst_n  = 0;
    req0   = 0;
    req1   = 0;
    #5  @(negedge clk)  rst_n = 1;
    #10 @(negedge clk)  req0  = 1;
    #10 @(negedge clk)  req0  = 0;
    #10 @(negedge clk)  req1  = 1;
    #10 @(negedge clk)  req1  = 0;
    #10 @(negedge clk)  {req0, req1} = 2'b11;
    #10 @(negedge clk)  {req0, req1} = 2'b00;
    #50 $finish;
  end

  // clock generator //
  always begin
    #5 clk = ~clk;
  end

endmodule:arbiterTop
```

Formal Verification

- Formal Property Checking/Model Checking.

  Correctness of design checked by rigorous mathematical procedures

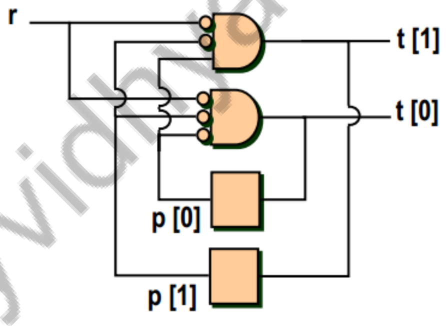
  It does not require test benches or stimuli and turnaround time is very less

  Boolean equivalence, Binary decision diagram (BDD)

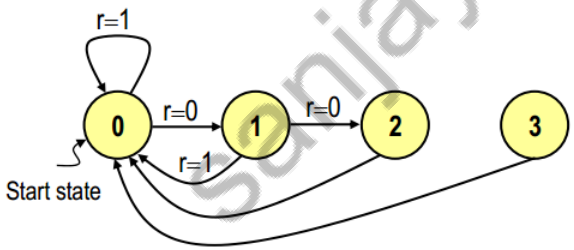
  System Verilog, Cadence Jasper

  Tool takes the DUT and Assertion file as inputs

❑ RTL Design
- Modulo-3 counter



❑ FSM Model



❑ CTL Property
- EF (p = 2)

❑ Starting from start state, can the counter eventually count upto 2

Functional Verification

1.Static Verification

- Against some predefined rules
- Verify your design at an early stage, without any stimulus
- Reduce the verification effort at the RTL level.

2.Functional Simulation

- Verifying the functional behavior
- Timing delays of the internal logic or interconnects are not considered

Timing Analysis

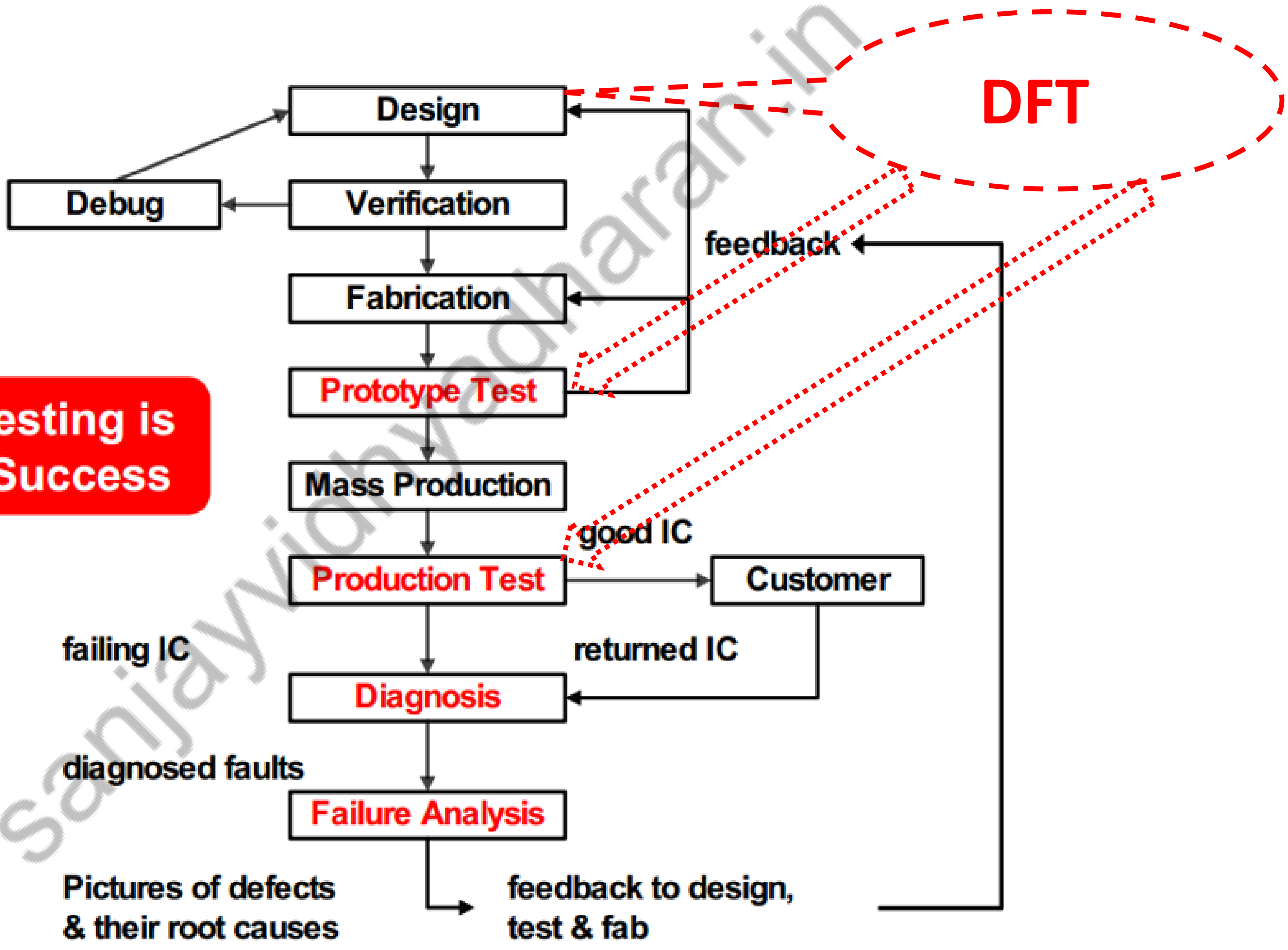
Static Timing Analysis

Does Static delay requirements without any input or output vectors

Dynamic Timing Analysis

Verifies functionality by applying input vectors and checking for correct output vectors

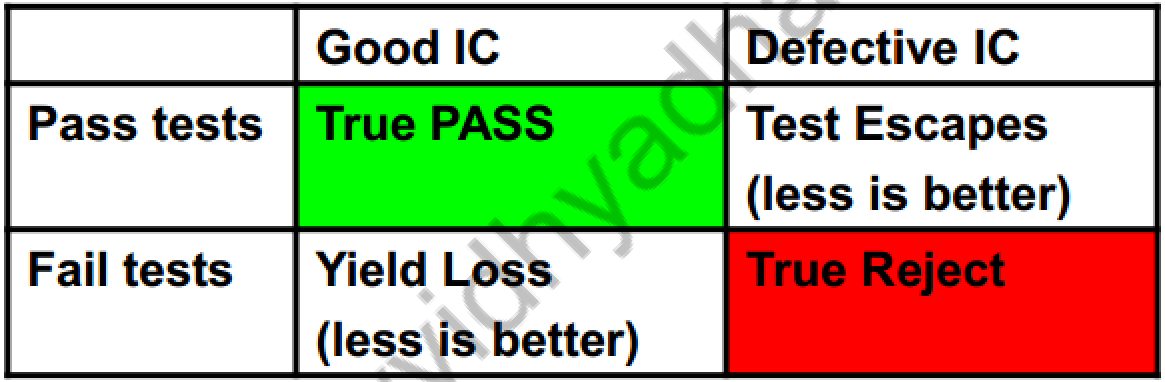
VLSI Production Flow



Good Testing is Key to Success

https://www.youtube.com/watch?v=yfcoKOUV5DM&list=PLvd8d-SyI7hjk_Ci0zpTqImAtpEjdK5JF&index=1

Testing



	Good IC	Defective IC
Pass tests	True PASS	Test Escapes (less is better)
Fail tests	Yield Loss (less is better)	True Reject

https://www.youtube.com/watch?v=yfcoKOUV5DM&list=PLvd8d-SyI7hjk_Ci0zpTqImAtpEjdK5JF&index=1

Types of Testing

1. Characterization (Verification)

- Verify that the design is correct, and the device will meet all specifications.
- Functional tests are run, and comprehensive AC and DC measurements are made.
- Probing of internal nodes of the chip can be done on all PVT corners
- Silicon debug/ Basic DC/AC tests (V_{OL}, V_{OH}, t_{pd} etc post fab is also called characterization.

2. Production (Testing)

- Quality check on produced chips
- The vectors may not cover all possible functions and data patterns but must have a high coverage of modeled faults.
- The main driver is cost, since every device must be tested. Test time (and therefore cost) must be absolutely minimized.

3. Burn-in

- Testing, either continuously or periodically, over a long period of time.
- Accelerated Life test

4. Incoming Inspection

- Inspection on the purchased devices before integrating them into the system. Depending upon the context, this testing can be either similar to production testing, or more comprehensive than production testing, or even tuned to the specific systems application

Types of Testing

1. Wafer sort or *probe*

Wafer Sort is a process where a die is tested electrically while still in wafer form.
Wafer Sort process done with the presence of equipment called wafer prober and Tester.

Types of Testing

2. Parametric Tests

- DC parametric tests include shorts test, opens test, maximum current test, leakage test, output drive current test, and threshold levels test.
- AC parametric tests include propagation delay test, setup and hold test, functional speed test, access time test, refresh and pause time test, and rise and fall time test.

3. Functional Tests.

- These consist of the input vectors and the corresponding responses. They check for proper operation of a verified design by testing the internal chip nodes.
- Functional tests cover a very high percentage of modeled (e.g., stuck type) faults in logic circuits and their generation is the main topic of this course.

Automatic Test Equipment

Advantest Model T6682 ATE

- The instrument electronics 0.35 VLSI chips.
- 1024 channels, so it can independently control and observe 1024 chip pins simultaneously.
- Test speed is either 250 MHz, 500 MHz, or 1 GHz.
- Drive busses between –2.5 V to 6.0 V,
- Can drive small amplitude 200 mV signals. clock/strobe timing accuracy is 870ps

Automatic Test Equipment

Semiconductor Test

with

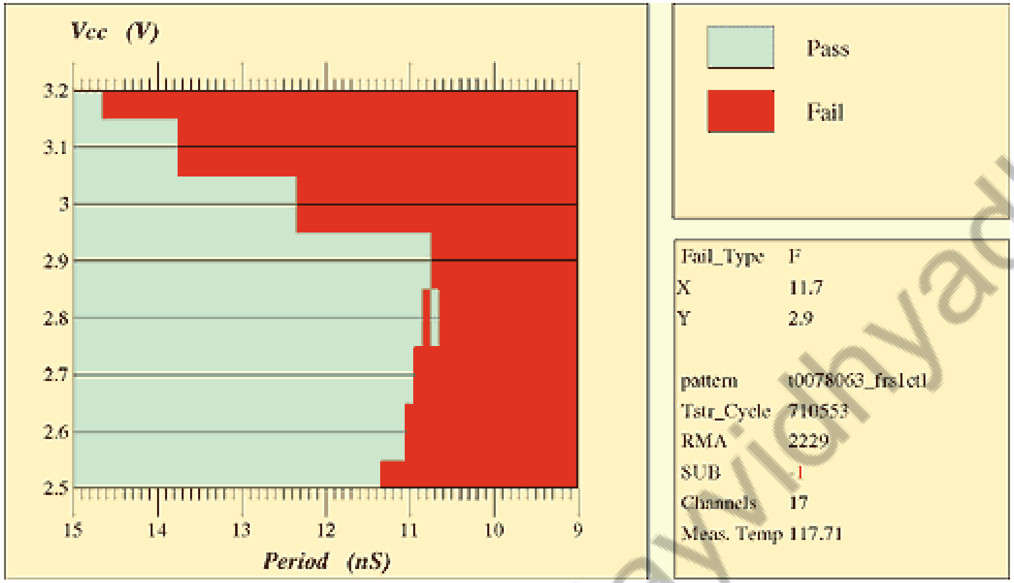
Software-Defined Modular Instrumentation

NI PXIe-6544/45/47/48
High-Speed Digital I/O

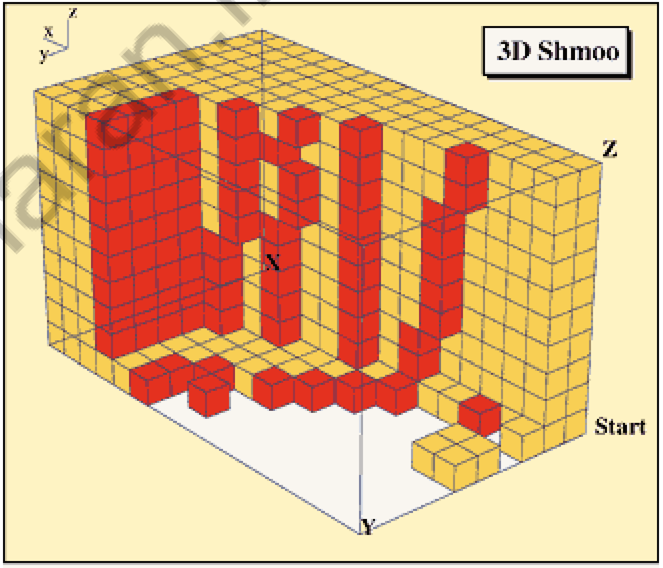
NI PXI-4132 High-Precision
Source Measure Unit

NI PXIe-5663E/5673E
6.6 GHz VSA & VSG

Shmoo Plot



Fail_Type	F
X	11.7
Y	2.9
pattern	t0078063_frslctl
Tstr_Cycle	710553
RMA	2229
SUB	-1
Channels	17
Meas. Temp	117.71



Axes	Param	Units	Start	End	Step
X	Period	nS	14	12.4	0.1
Z	VCCq	V	1.3	2.2	0.1
Y	VCC	V	1.3	2.2	0.1

VCCQ is the I/O supply of the emc interface

https://www.semiconductoronline.com/doc/shmooplot-0001

Test Economics

Tradeoff : Quality level vs. Cost

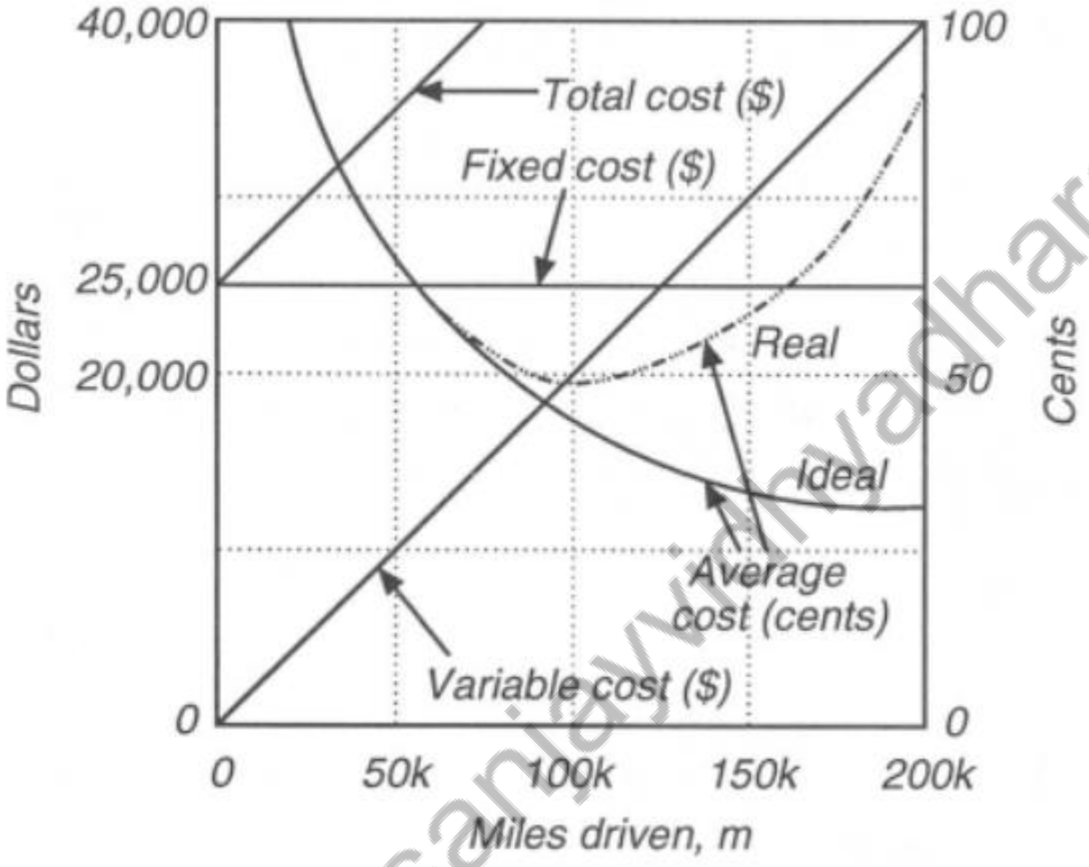
1. **Fixed Costs (FC):** These are the costs of things that are necessary but do not change with use. Example machinery. Fixed cost per product reduces with increase in product output.

2. **Variable Costs (VC):** These costs increase with production output. E.g., Labor, raw material energy etc. Variable cost per product may remain constant reduces with increase in product output.

3. **Total Costs (TC):** Sum of FC and VC

4. **Average Cost :** These are obtained by dividing the total costs by the number of units produced.

Test Economics



If aging factor is taken into account, the average cost might be as shown by the rising curve (shown as real), called a *bathtub* curve.

$$\text{Average cost} = \frac{25,000}{m} + 0.2 \ \textit{dollars per mile}$$

Test Economics

Production output Q (x) where x is inputs

$$\text{Average product} = \frac{Q}{x}$$

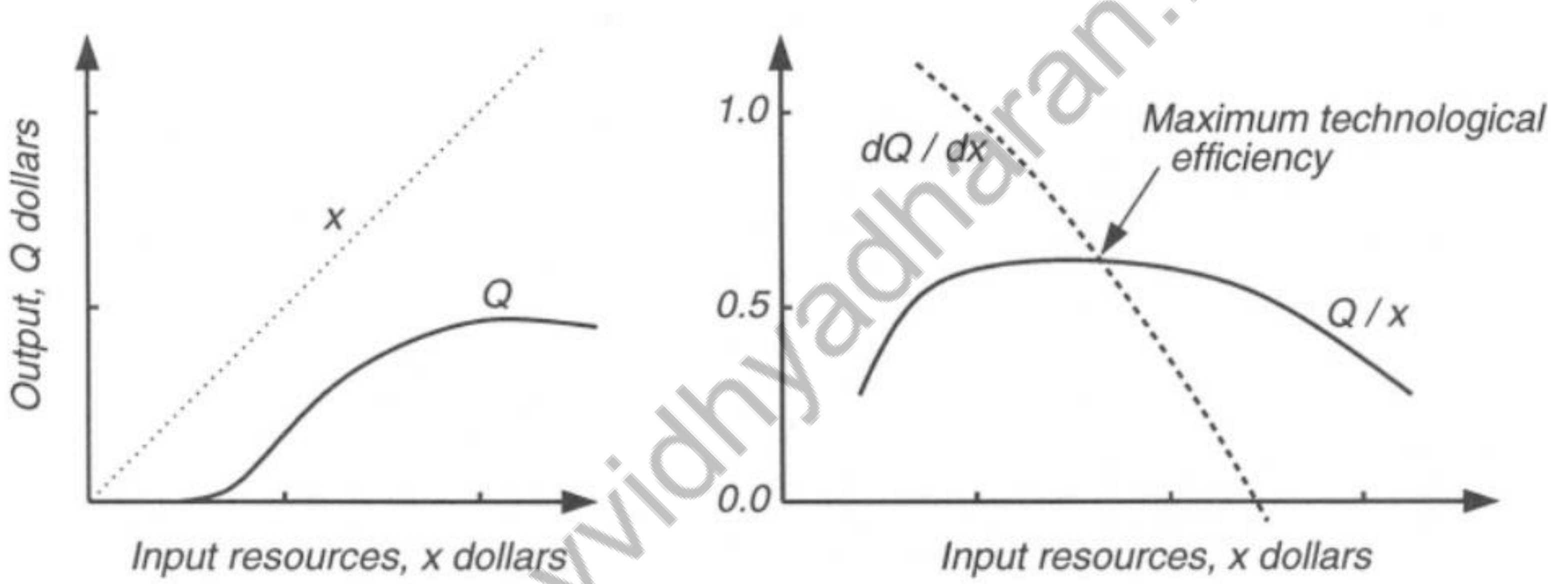
$$\text{Marginal product} = \frac{dQ}{dx}$$

The average product, or the product per unit of input, is called the *technological efficiency*. We maximize this efficiency by setting:

$$\frac{d}{dx}\frac{Q}{x} = 0 \text{ or } \frac{1}{x}\frac{dQ}{dx} - \frac{Q}{x^2} = 0$$

$$\frac{Q}{x} = \frac{dQ}{dx}$$

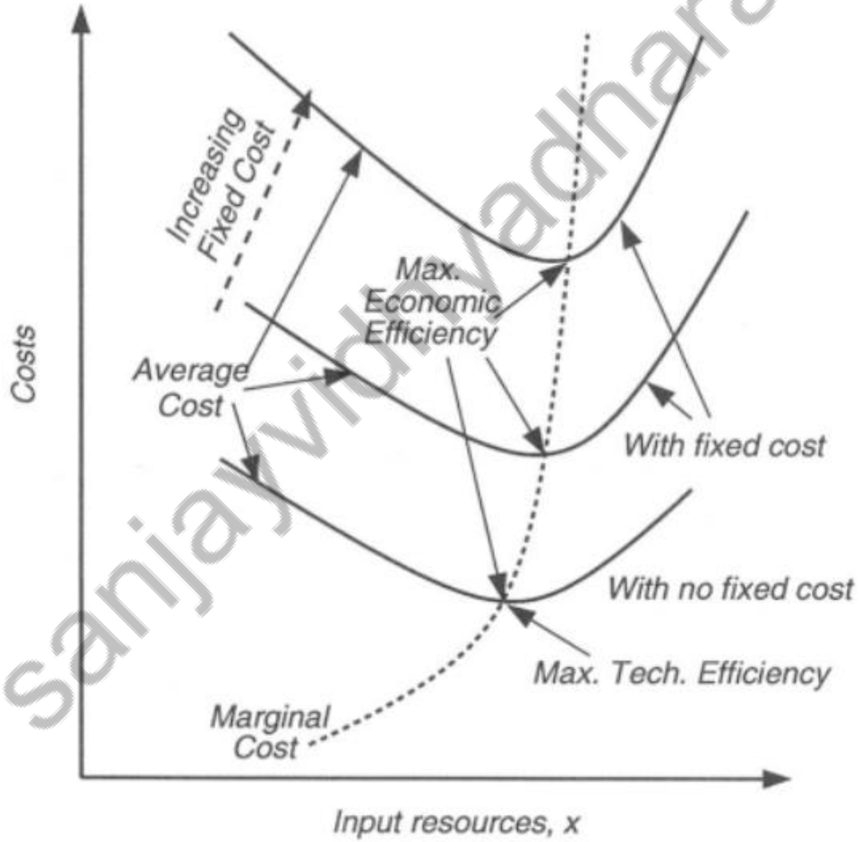
Test Economics



Maximizing technological efficiency.

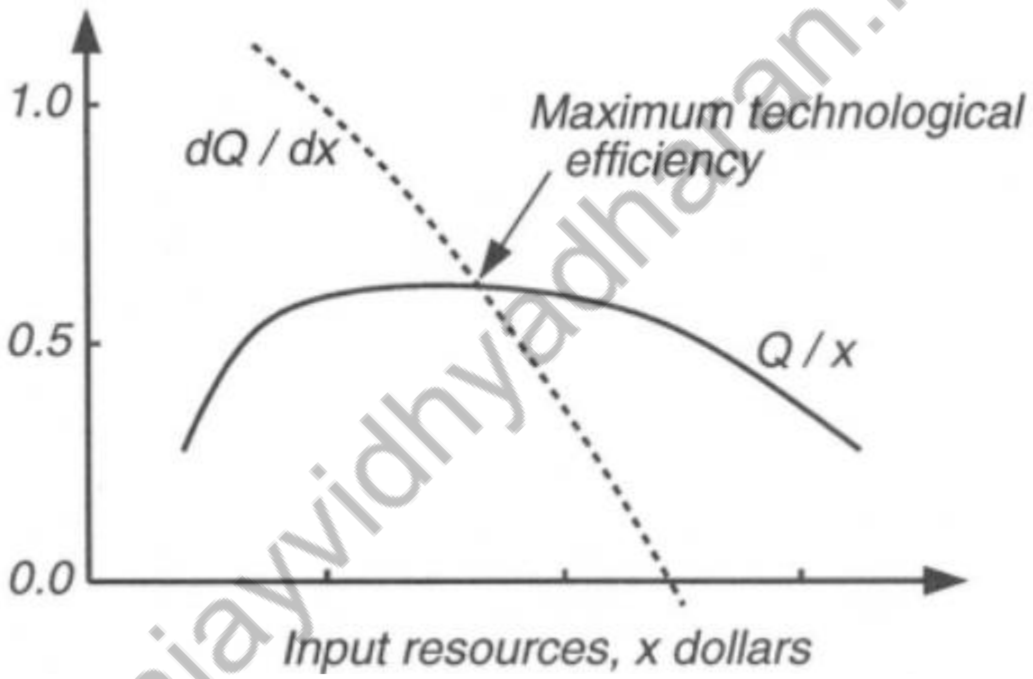
Test Economics

Economic Efficiency. Engineers are good at optimizing the technological efficiency, but often ignore the total cost of the product. Economic efficiency is related to the *total cost* of production, which includes both fixed and variable costs



Maximum economic efficiency

Test Economics



The Law of Diminishing Returns: *If one input of production is increased keeping other inputs constant, then the output may increase, eventually reaching a point beyond which increasing the input will cause progressively less increase in output.*

Test Economics

Increasing Returns to Scale. The case of *mass production* is worth considering. Production often increases faster than the increase of inputs, which is called *increasing returns to scale*. Some of the reasons are:

(1) Technological factors and

(2) Specialization.

In the long run, however, the law of diminishing returns prevails.

Test Economics

Benefit-Cost Analysis

Benefits include income from sale of products or services, savings in cost and time, etc. *Costs* refer to the costs of labor, machinery, energy, finances, risks, etc. All items are normally quantified and expressed in the same units (e.g., dollars.) We then define the *benefit-cost ratio* as follows:

$$\text{B/C ratio} = \frac{\text{annual benefits}}{\text{annual costs}}$$

For buying a car, the benefits could include convenient transportation to work or school and saving time.

Test Economics

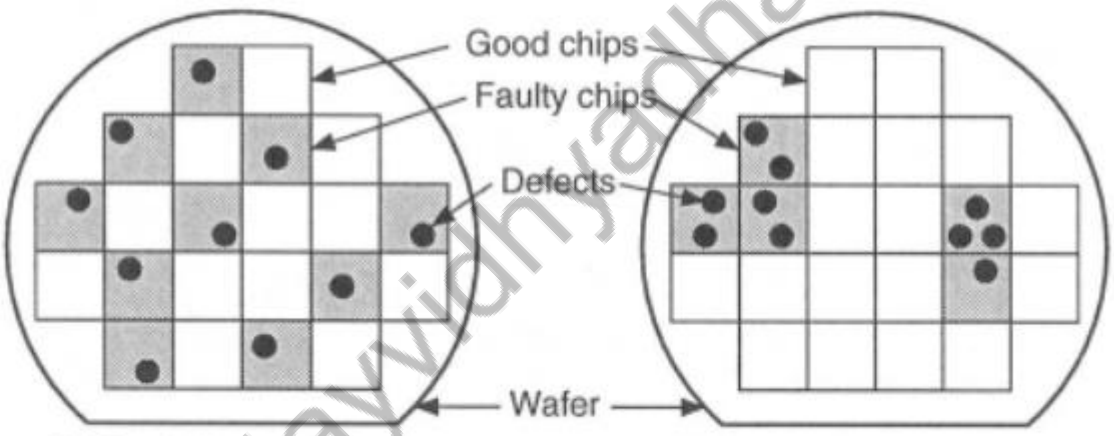
Testing can be 50 to 60% of their equipment manufacturing cost
Test hardware onto the chip enable at speed testing
"Any attempt to observe a system will perturb the system behavior."
Techniques such as *scan* design, BIST, and *boundary scan* simplify the test problem of electronic systems.

The Rule of Ten

It is widely accepted in the electronics industry that chips must be tested before they are assembled onto printed circuit boards (PCBs), which, in turn, must be tested before they are assembled into systems. This is because experience has shown that the *rule of ten* holds. If a chip fault is not caught by chip testing, then finding the fault costs 10 times as much at the PCB level as at the chip level. Similarly, if a board fault is not caught by PCB testing, then finding the fault costs 10 times as much at the system level as at the board level.

Test Economics

Yield The *process yield* of a manufacturing process is defined as the fraction (or percentage) of acceptable parts among all parts that are fabricated.



Wafer yield = 12/22 = 0.55

(a) Unclustered defects.

Wafer yield = 17/22 = 0.77

(b) Clustered defects.

The term *wafer yield* is sometimes used to refer to the average number of good chips produced per wafer.

Test Economics

A good testing procedure can reject all (or most) defective parts. Testing, however, cannot improve the process yield. There are two ways of improving the process yield:

(1) *Diagnosis and Repair.* The parts that are found defective after test are diagnosed for specific failures which are then repaired. Although the yield is improved, this procedure increases the cost of manufacturing. The reason is that we first allow the process to make errors which are then corrected. A more economical procedure is to eliminate the source errors. (Fault Tolerant Design).

(2) *Process Diagnosis and Correction.* The defects found in the failed parts are traced to specific causes, which may be defective material, faulty machines, incorrect human procedures, etc. Once the cause is eliminated, the yield improves. Process diagnosis is the preferred method of yield improvement.

Test Data Analysis

Defects versus faults.

Process variations, such as impurities in wafer material and chemicals, dust particles on masks or in the projection system, mask misalignment, incorrect temperature control, etc., can produce defects on wafers. The term *defect* generally refers to a physical imperfection in the processed wafer

The term *fault* is used to refer to electrical, Boolean, or The term *fault* is used to refer to electrical, Boolean, or functional malfunctions functional malfunctions.

In general, a physical defect in a chip can produce multiple faults. Thus, the spatial distribution of faults on a wafer is also clustered, sometimes even more so than the defects.

Test Quality

Defect level is measured as Defect per Million (DPM)

< ***200 DPM is acceptable for most IC***
>***1,000 DPM is very bad for most IC***

System DPM
Chip DPM X Number of chips in the system

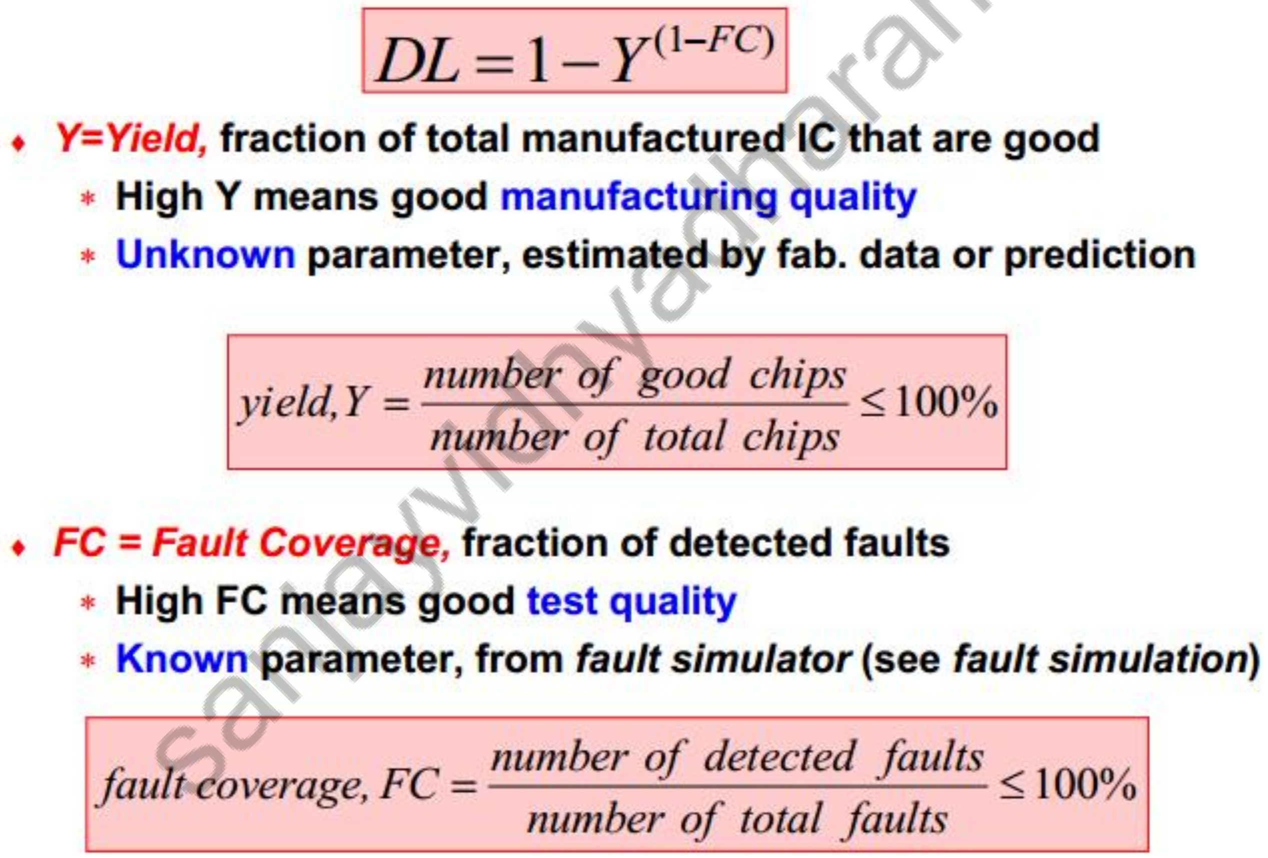
E.g.
If system has 10 Chips and Chip has DPM of 1000 (0.1% IC is defective)
System DPM = 1%
If 1 Million system is manufactured 10,000 will be defective

Defect Level (DL)

- **Fraction of bad IC passing the test (test escapes)**

Models to Predict DPM

Brown & Williams (IBM, 1981), Binomial distribution



$$\boxed{DL = 1 - Y^{(1-FC)}}$$

- *Y=Yield,* fraction of total manufactured IC that are good
  - High Y means good manufacturing quality
  - Unknown parameter, estimated by fab. data or prediction

$$\boxed{yield, Y = \frac{number\ of\ good\ chips}{number\ of\ total\ chips} \leq 100\%}$$

- *FC = Fault Coverage,* fraction of detected faults
  - High FC means good test quality
  - Known parameter, from *fault simulator* (see *fault simulation*)

$$\boxed{fault\ coverage, FC = \frac{number\ of\ detected\ faults}{number\ of\ total\ faults} \leq 100\%}$$

https://www.youtube.com/watch?v=yfcoKOUV5DM&list=PLvd8d-SyI7hjk_Ci0zpTqImAtpEjdK5JF&index=2

Models to Predict DL

Brown & Williams (IBM, 1981), Binomial distribution

$$FC = \frac{d}{t}, where\ d\ is\ faults\ detected\ and\ t\ is\ total\ faults$$

$$Yeild = Probability\ that\ IC\ is\ good = (1-q)^{t}$$

Where each fault occurrence probability is q (uniform independent)

$$Fraction\ of\ \ Good\ IC\ passing\ test\ = (1-q)^{t-d} = (1-q)^{t\left(1-\frac{d}{t}\right)} = Y^{(1-FC)}$$

$$Fraction\ of\ \ Bad\ IC\ passing\ test\ (DL) = 1 - Y^{(1-FC)}$$

In the Williams-Brown model, dies are assumed to have equal faults to model the impact of actual defects.

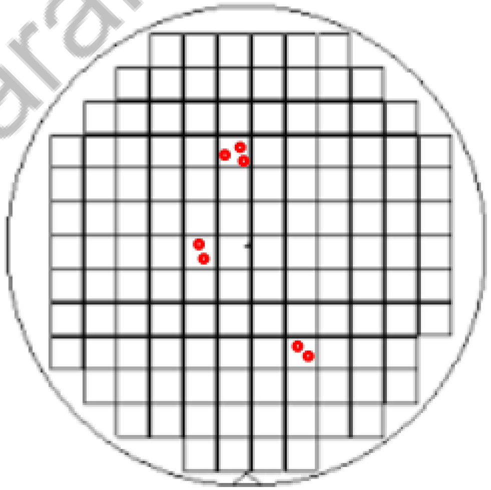
https://www.youtube.com/watch?v=yfcoKOUV5DM&list=PLvd8d-SyI7hjk_Ci0zpTqImAtpEjdK5JF&index=2

Models to Predict DL

Agarwal Model, Poisson distribution

$$DL = \frac{(1-FC)(1-Y)e^{-(n-1)FC}}{Y+(1-FC)(1-Y)e^{-(n-1)FC}}$$

- Actual data show defects are clustered
  - *n* =average num. of defects on a bad die
- We need two unknown parameters
  - *n*: from experiment
  - *Y*: from fab. or prediction



7 defects
3 bad die
n=7/3=2.3

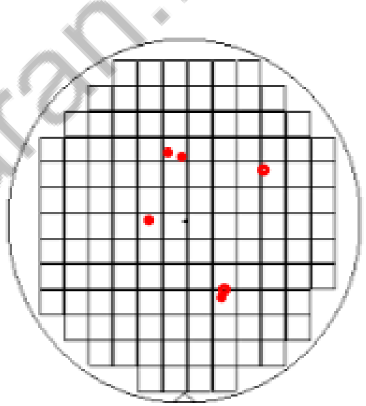
https://www.youtube.com/watch?v=yfcoKOUV5DM&list=PLvd8d-SyI7hjk_Ci0zpTqImAtpEjdK5JF&index=2

Yield Estimation

- Number of defects *k* on a die obeys *Poisson distribution* with mean *λ*
  - *Prob(k)* = prob exactly *k* defects on a die
  - A die is good when *k*=0
  - Yield = *Prob(k=0)*

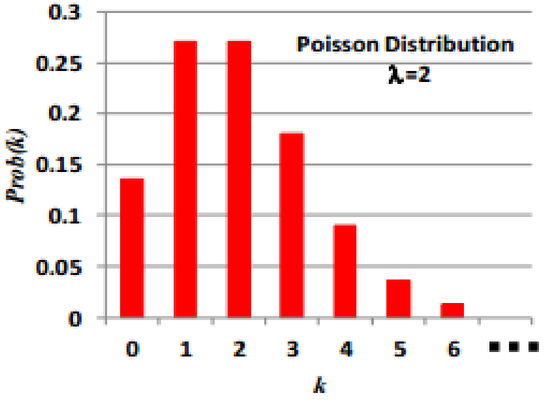
$$Prob(k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

$$Y = Prob(k=0) = e^{-\lambda}$$



- *A* = die area, *D* = defect density
- *λ* = *AD* = ave. number of defects per die

$$\boxed{Y = e^{-AD} \quad \textit{Largerarea,lower yield}}$$



https://www.youtube.com/watch?v=yfcoKOUV5DM&list=PLvd8d-SyI7hjk_Ci0zpTqImAtpEjdK5JF&index=2

References

1. "Essentials of Electronic Testing, for Digital, Memory and Mixed-Signal VLSI Circuits", Michael L. Bushnell and Vishwani D. Agrawal, Kluwer Academic Publishers (2000).

2. Video lectures by Professor James Chien-Mo Li
Lab. of Dependable Systems Graduate Institute of Electronics Engineering.
National Taiwan University
https://www.youtube.com/watch?v=yfcoKOUV5DM&list=PLvd8d-SyI7hjk_Ci0zpTqImAtpEjdK5JF&index=1

Thankyou