Topic IV

Verification Techniques Overview (II)

Static Analysis, Physical Verification & Testing

系統晶片驗證

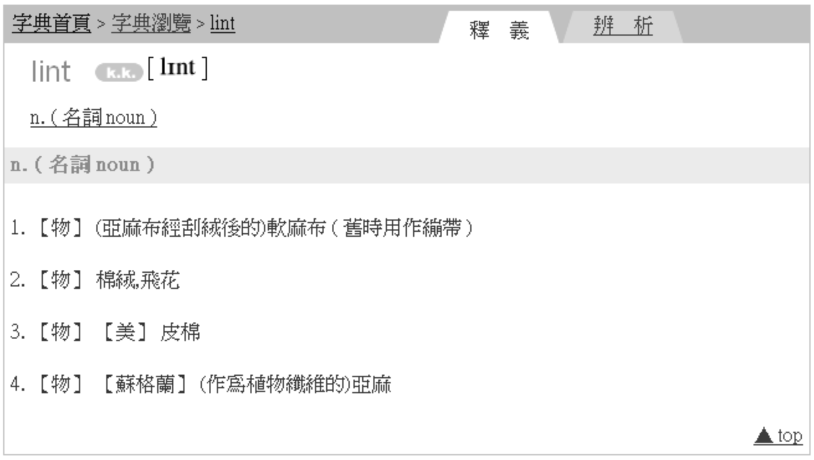
SoC Verification

Sep, 2004

What we will cover in this topic…

- ◆Static analysis
  - ● Linting
  - ● Clock domain checking
  - ● Static timing analysis
  - ● Signal Integrity
- ◆Physical verification
  - ● DRC
  - ● LVS
  - ● Design for Manufacturability (DfM)
- ◆Testing

What is Linting?



字典首頁 > 字典瀏覽 > lint

釋 義 | 辨 析

lint k.k. [lɪnt]

n. (名詞 noun)

n. (名詞 noun)

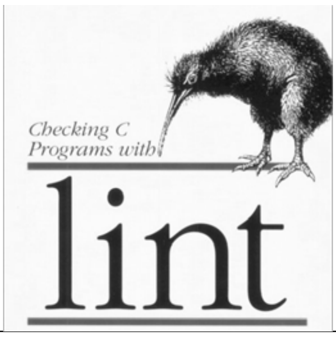
1. 【物】 (亞麻布經刮絨後的)軟麻布 (舊時用作繃帶)
2. 【物】 棉絨,飛花
3. 【物】 【美】 皮棉
4. 【物】 【蘇格蘭】 (作爲植物纖維的)亞麻

▲ top

Source: Yahoo! 奇摩字典

The Origin of Linting

◆ Named after the Unix programming tool lint
- Picks fluff/lint out of your programming code.
- Seeks out common errors, missing attributes, poor formatting that may be technically correct



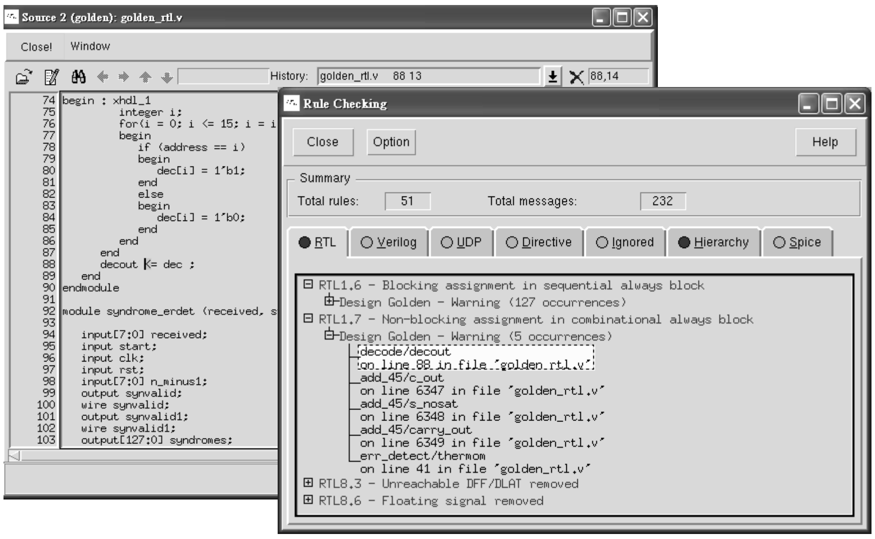
HDL Linting

- Also called “RTL Design Rule Checking”
- Gives warnings/errors to:
  - Bad/ambiguous coding styles
  - Potential simulation/synthesis inconsistency
- Tool support
  - Special linting tools
  - Synthesis tools
  - Simulation/Verification tools

- It’s a good practice to pay attention to the linting errors, however ---

  (Problem) User may ignore the warnings/errors because there are usually too many…

Linting Tool Example



```
 74 begin : xhdl_1
 75         integer i;
 76         for(i = 0; i <= 15; i = i
 77         begin
 78            if (address == i)
 79            begin
 80               dec[i] = 1'b1;
 81            end
 82            else
 83            begin
 84               dec[i] = 1'b0;
 85            end
 86         end
 87      end
 88      decout <= dec ;
 89   end
 90 endmodule
 91
 92 module syndrome_erdet (received, s
 93
 94    input[7:0] received;
 95    input start;
 96    input clk;
 97    input rst;
 98    input[7:0] n_minus1;
 99    output synvalid;
100    wire synvalid;
101    output synvalid1;
102    wire synvalid1;
103    output[127:0] syndromes;
```

Total rules: 51 Total messages: 232

```
⊟ RTL1.6 - Blocking assignment in sequential always block
   ⊞Design Golden - Warning (127 occurrences)
⊟ RTL1.7 - Non-blocking assignment in combinational always block
   ⊟Design Golden - Warning (5 occurrences)
      decode/decout
      on line 88 in file "golden_rtl.v"
      add_45/c_out
      on line 6347 in file "golden_rtl.v"
      add_45/s_nosat
      on line 6348 in file "golden_rtl.v"
      add_45/carry_out
      on line 6349 in file "golden_rtl.v"
      err_detect/thermom
      on line 41 in file "golden_rtl.v"
⊞ RTL8.3 - Unreachable DFF/DLAT removed
⊞ RTL8.6 - Floating signal removed
```

Source: Cadence Conformal/LEC

Linting vs. Super Linting

- Linting
  - Structural, syntactical, semantics checking
- Super linting
  - Structural + functional, predefined property checking
  - Properties are automatically extracted during the RTL compilation process
  - Need to call formal verification engine

Super Linting Properties

- Also called “predefined properties”, “static properties”, etc.
- Examples
  - Bus contention (one-hot checks)
  - Design don’t-care
    - Synopsys full/parallel case
    - X assignment
    - Array bound check (e.g. assign a[y] = data;)
  - Finite State Machine (FSM) checks
    - State reachability
    - State transition
    - Dead lock checks
  - Register racing condition
  - Branch enabling condition
  - Floating point error

What we will cover in this topic…

- ◆Static analysis
  - Linting
  - Clock domain checking
  - Static timing analysis
  - Signal Integrity
- ◆Physical verification
  - DRC
  - LVS
  - Design for Manufacturability (DfM)
- ◆Testing

What is Clock Domain?

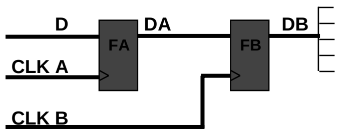
- ◆Clock domain
  - Signals that share the same clock (tree) source (i.e. have the synchronized timing)

Modern designs usually have multiple clocks with very high frequencies

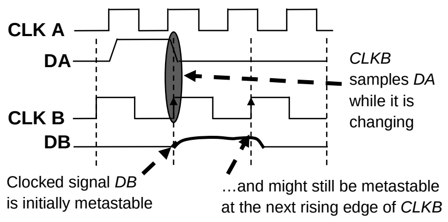
Clock Domain Crossing Problem

◆Signals crossing from one clock domain to another clock domain are considered asynchronous

What's the problem??



◆Signals crossing asynchronous domains create metastability


CLK A
DA
CLK B
DB
CLKB samples DA while it is changing
Clocked signal DB is initially metastable
…and might still be metastable at the next rising edge of CLKB

Source: Cadence Design Systems

Handling Asynchronous Crossings

☞ Using synchronizers

◆Metastable signals need time to settle to a stable value

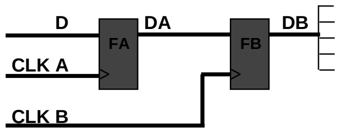
◆By adding synchronizers, the metastable signal can have additional clock cycles to become stable

◆Synchronizer examples:

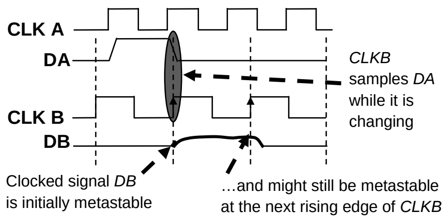
- Flip-Flops
- Mux
- Multi-cycle path

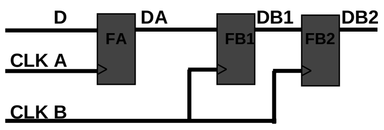
Example: Flip-Flops as Synchronizers

◆Adding adequate flip-flops

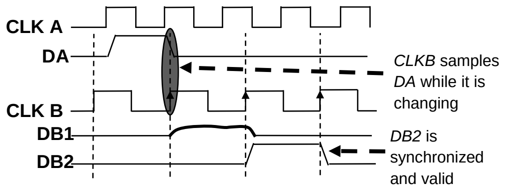


Potential *metastability* occurs due to setup or hold time violation in flip-flop "FB"


CLK A
DA
CLK B
DB
CLKB samples *DA* while it is changing
Clocked signal *DB* is initially metastable
…and might still be metastable at the next rising edge of *CLKB*



Two flip-flop synchronizer solution


CLK A
DA
CLK B
DB1
DB2
CLKB samples *DA* while it is changing
DB2 is synchronized and valid

Source: Cadence Design Systems

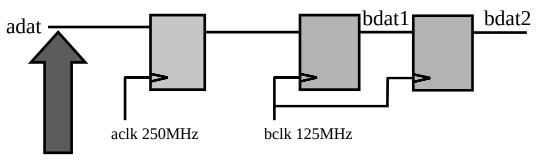
Clock Domain Crossing Checks

◆However, given a multi-million-gate design, which may have *billions or trillions of paths*, how do you make sure you have properly added in synchronizers and have no domain crossing problems?

➔ Finding Synchronization errors could be very expensive by STA or simulation

➔ Clock Domain Checker (CDC) performs a structural check and can find clock domain crossing violations more quickly

Another Clock Domain Crossing Problem: Data Stability



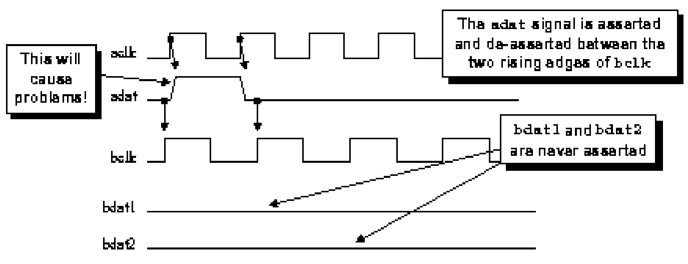

This will cause problems!
aclk
adat
bclk
bdat1
bdat2
The adat signal is asserted and de-asserted between the two rising edges of bclk
bdat1 and bdat2 are never asserted

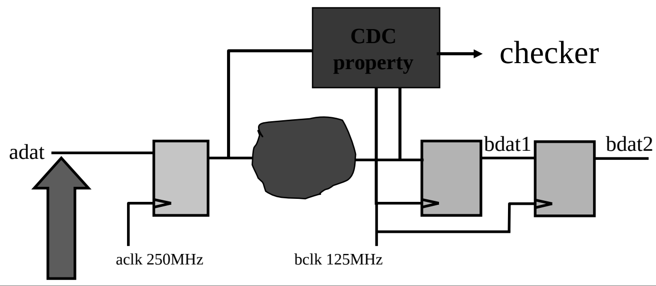
Figure 5 - Short control signal pulse missed during synchronization

◆No metastability problem; but “adat” is lost!!

Functional CDC Checker

◆It’s a predefined property checking

- Data stability property
  - Make sure the data latched at source FF will stay stable until the clock ticks at the destination FF
- ➔ Use formal engine to prove it



In short,

- Linting and clock domain checking allow you to identify potential RTL design problems up front
  - Structural checks are very fast
  - Functional checks are mostly local, and thus usually can work for large designs too

  ➔ can be integrated into design flow easily

What we will cover in this topic…

- Static analysis
  - Linting
  - Clock domain checking
  - Static timing analysis
  - Signal Integrity
- Physical verification
  - DRC
  - LVS
  - Design for Manufacturability (DfM)
- Testing

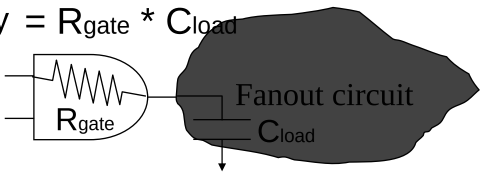
How fast can your design run?

- Depend on the *critical delay* of your circuit
  - Max clock frequency = 1 / (critical delay)
- Timing (delay) paths of your circuit
  - Combinational paths (i.e. no FFs). 4 types:
    1. Register to register
    2. PI to register
    3. Register to PO
    4. PI to PO
- Critical delay path
  - The timing path that has the longest delay

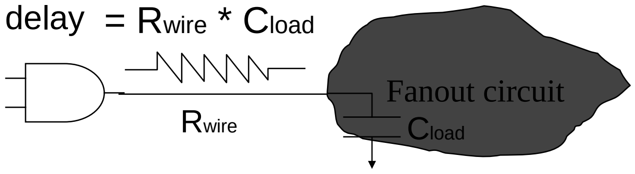
How to calculate the delay?

R * C = (d V / d I) * (d Q / d V) = d T

- Gate delay $= R_{gate} * C_{load}$



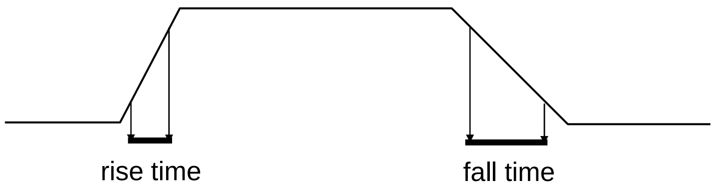
- Wire delay $= R_{wire} * C_{load}$



- R & C (and sometimes Inductance) are extracted from layout netlist

Delay Calculation

- ◆ Delay is pattern dependent
  - ● rise time != fall time (aka. setup time != hold time)



- ◆ Delay also depends on
  - ● Process
  - ● Voltage
  - ● Temperature
  - ➔ Linear or Table-Lookup model

Issues about Delay Calculation

- ◆ In deep submicron, wire delay > gate delay
  - ● Need physical (placement & routing) information
- ◆ Since delay is pattern dependent
  - ● Given a path, different input patterns will result in different delays
  - ➔ (Dynamic) Timing analysis is very expensive!!

Static Timing Analysis (STA)

- For each node (gate or wire) in a circuit, compute its ---
  - (Best, worst) x (rise time, fall time)
- Perform topological traversals
  - Propagate forward the arrival time
  - Compute backward the required time
  - Create the slack (i.e. required - arrival time) graph and obtain the critical delay path(s)

Advantages of Static Timing Analysis

- Exhaustive timing analysis
- No input vectors required
- Much more efficient than dynamic methods
  - Faster operations
  - Capacity of millions of gates

Source: CIC course

Disadvantages of Static Timing Analysis

- For synchronous logic only
- Much more setup efforts
- Tricky constraints needed for design that is NOT single-clock flip-flop-based
  - Multiple clocks
  - False paths
  - Latches
  - Multi-cycle paths
- Lack of consistent conventions

Source: CIC course

Do you see a niche market?

What we will cover in this topic…

- Static analysis
  - Linting
  - Clock domain checking
  - Static timing analysis
  - Signal Integrity
- Physical verification
  - DRC
  - LVS
  - Design for Manufacturability (DfM)
- Testing

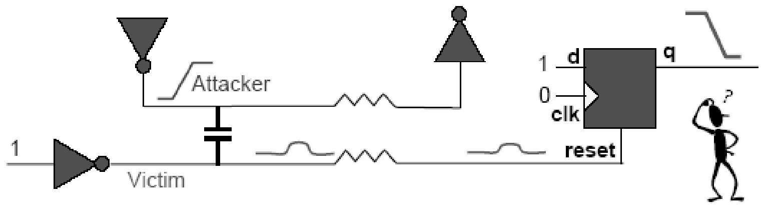
Signal and Design Integrity Problem

- As processes shrink and frequencies increase
  ➔ signal integrity problems created by 2nd order physical effects
  - i.e. Capacitance coupling, capacitance + inductance
  - Signal integrity problems
    - Cross talk induced functional errors
    - Cross talk induced timing violations
    - Power (IR) drop or Ground bounce
  - Design integrity problems (discussed later)
    - Electromigration
    - Hot electron effects
    - Wire self-heating

Source: Cadence S&DI White Paper

Cross Talk Induced Functional Errors

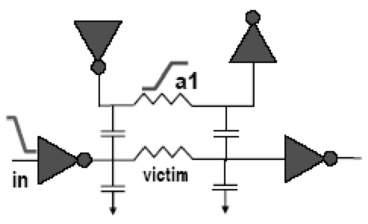
Coupling noise can cause functional failures.

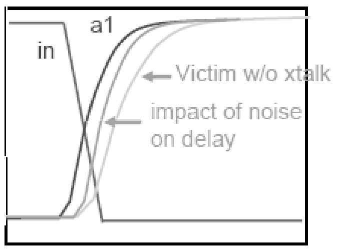


- ◆ Two wires are very close to each other
- ◆ Attacker (aggressor) wire switching affects the victim wire

Source: Cadence CeltIC

Cross Talk Induced Hold Time Violations

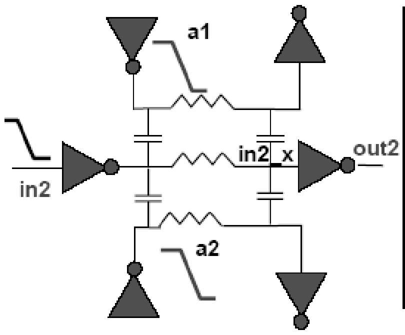


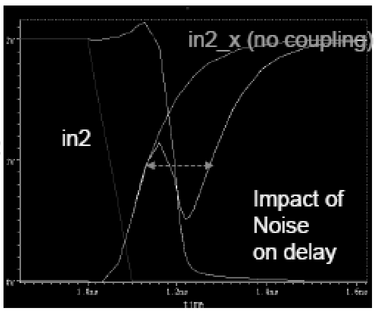


- ◆ Helpful glitch: a fast transition on a aggressor net that switches in the same direction as a victim net
- ◆ Helpful glitch causes a victim net to transition faster ➔ hold time violation

Source: Cadence CeltIC

Cross Talk Induced Setup Time Violations





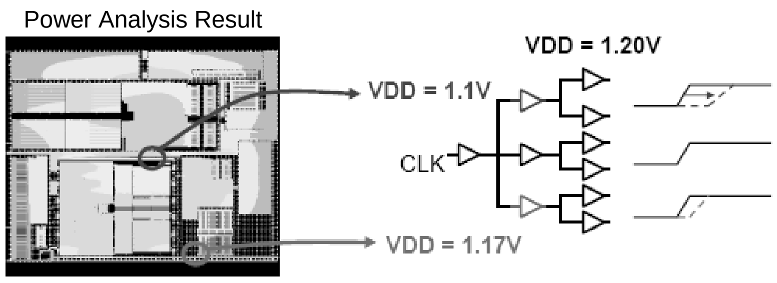
- ◆ Unhelpful glitch: a fast transition on a aggressor net that switches in the opposite direction from a victim net
- ◆ Unhelpful glitch causes a victim net to transition slower
  ➔ setup time violation

Source: Cadence CeltIC

Voltage (IR) Drop or Ground Bounce Problems

- ◆The fact
  - ● The wires from chip input power pin(s) to power nodes (Vdd) circuitry inside have resistance
  - ➔ The voltage that goes to internal circuitry is less than that applied to the chip
- ◆The problems
  - ● The circuit may not get enough voltage and can malfunction or not meet the timing specifications

Voltage Drop Impacts on Timing



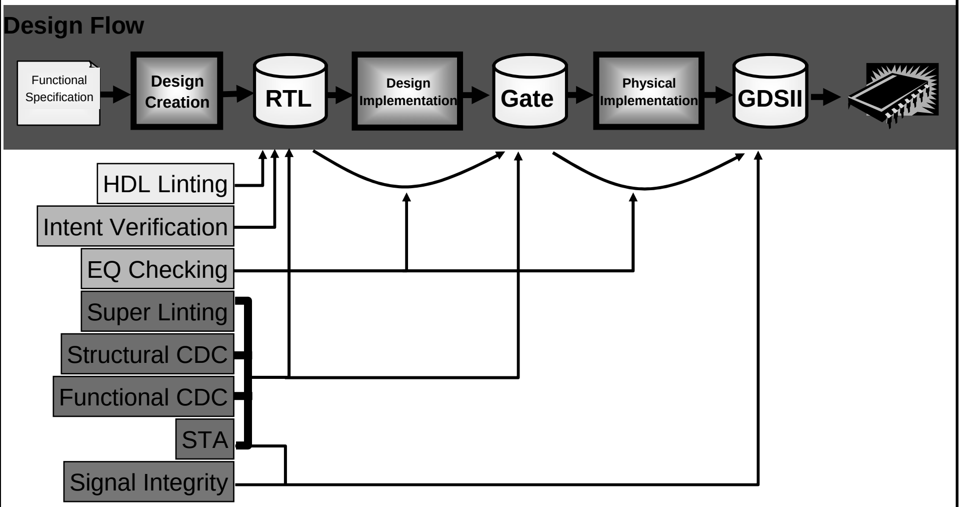
- ◆ IR Drop (ground bounce) increases clock skew
  ➔ Hold time violations
- ◆ IR Drop (ground bounce) increases signal skew
  ➔ Setup time violations

Source: Cadence VoltageStorm

Signal Integrity Problem Prevention and Correction

- ◆ Needs to take cross-coupling effects and peak voltage drops into consideration
- ◆ Different Tools' Approaches
  1. Post-layout SI analysis
  2. Combined STA and SI approaches
  3. Integrated physical implementation and analysis engine

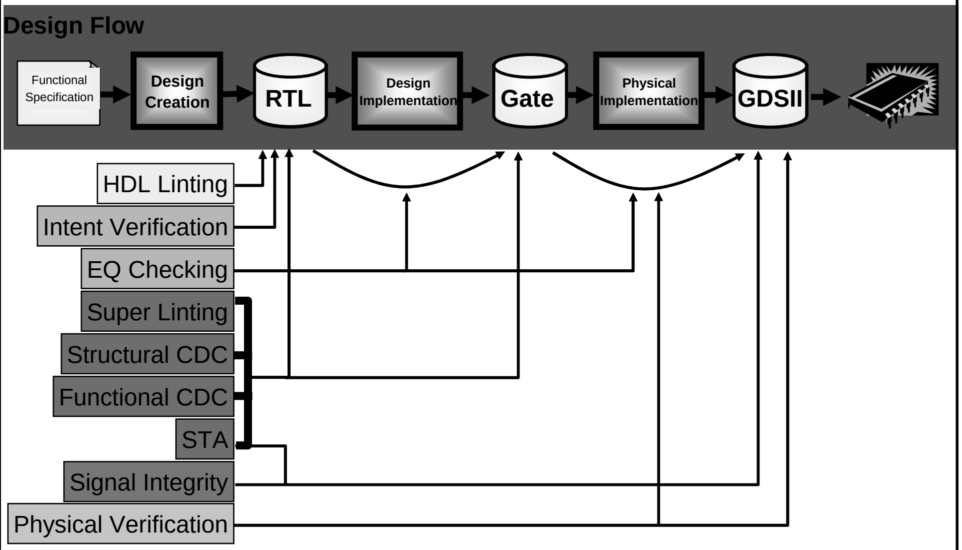
Static Analysis in the Design Flow



What we will cover in this topic…

- Static analysis
  - Linting
  - Clock domain checking
  - Static timing analysis
  - Signal Integrity
- Physical verification
  - DRC
  - LVS
  - Design for Manufacturability (DfM)
- Testing

What is Physical Verification?



Types of Physical Verification

- Design Rule Check (DRC)
- Layout Versus Schematic (LVS)
- Resolution Enhancement Technology (RET)
  - Optical Proximity Correction (OPC) (or Optical and Process Correction)
  - Phase-Shifting Mask (PSM)
- Others
  - Electrical Rule Check
  - Device-level Resistance, Capacitance, Induction Extraction

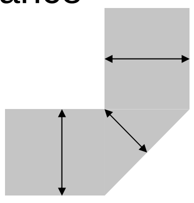
Design Rule Check (DRC)

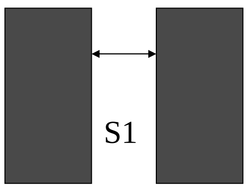
- Check if a *physical implementation* is in conformance with a given *IC manufacturing process*
  - Inputs:
    - Layout file --- geometrical shapes (e.g. GDS-II)
    - IC manufacturing rule document
      (e.g. TSMC 0.18um Mixed Signal (1P6M) CMOS)
  - Outputs:
    - Layout geometrical violations that may affect
      1. Manufacturability
      2. Yield

      of the chip
- [Note] Don't confused with HDL Design Rule Checking

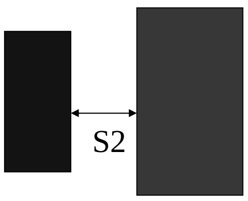
What does DRC Check?

1. Distance


Width: distance of interior-facing edge for a single layer (W)



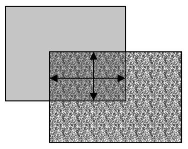
Space: distance of exterior-facing edge for one layer (S1)

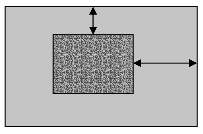


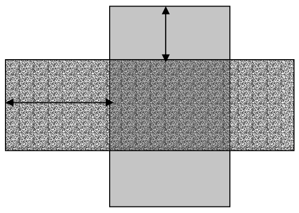
Space: distance of exterior-facing edge for two layers (S2)

What does DRC Check?

2. Overlap

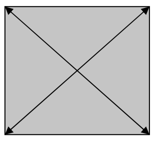

Overlap: distance of interior-facing layers

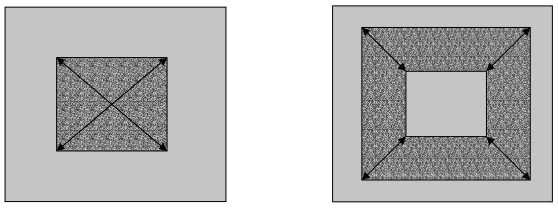

Enclosure: Distance of inside edge To outside edge (fully inside)


Extension: distance of inside edge to outside edge

What does DRC Check?

3. Area

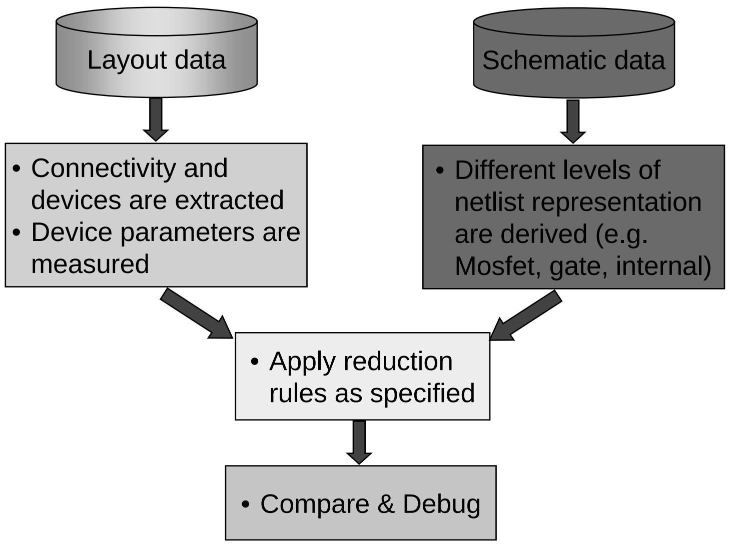

Area of the geometry


Enclosed Area

Layout Versus Schematic (LVS)

- ◆Compare the connectivity of *a physical layout* design to the *schematic* from which it was designed (e.g. Spice, Verilog)
  - ● Perform design matching of nets, devices, and device parameters
  - ● In addition, even Layout vs Layout (LVL), or Schematic vs Schematic (SVS) can be checked

LVS Operational Flow


Layout data
Schematic data
• Connectivity and devices are extracted
• Device parameters are measured
• Different levels of netlist representation are derived (e.g. Mosfet, gate, internal)
• Apply reduction rules as specified
• Compare & Debug

LVS Rules Files

[Using Cadence Assura as example]

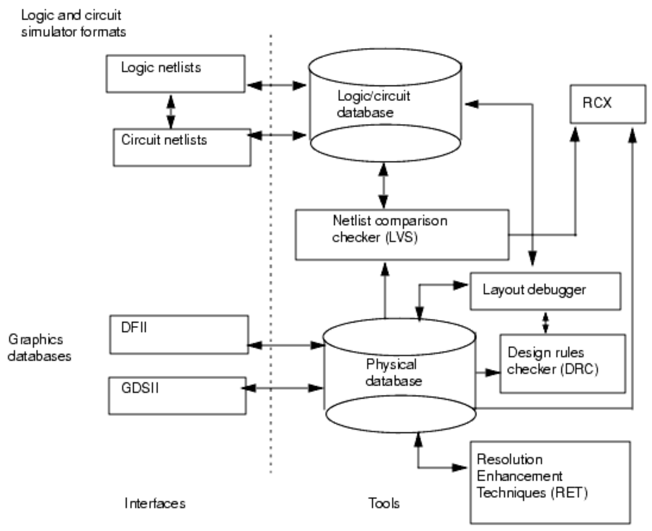
- ◆ extract.rul
  - Contains rules for defining and extracting devices and nets from the layout data
- ◆ compare.rul
  - Contains rules for comparing the schematic netlist to the extracted layout netlist
- ◆ binding.rul
  - Optional binding rules to help Assura LVS match the schematic to layout
- ◆ deviceinfo.rul
  - A support file that can be used to import a schematic netlist

Source: Cadence Assura Developer Guide

LVS vs. Functional Equivalence Checking

- ◆ LVS
  - Structurally compare layout and schematic netlists
- ◆ Functional Equivalence Checking
  - Can ompare netlists from RTL to GDS-II (layout)
  - Use formal engine for functional comparison
  - For layout circuit, need to extract logic functions from transistor netlist

Typical Physical Verification Flow

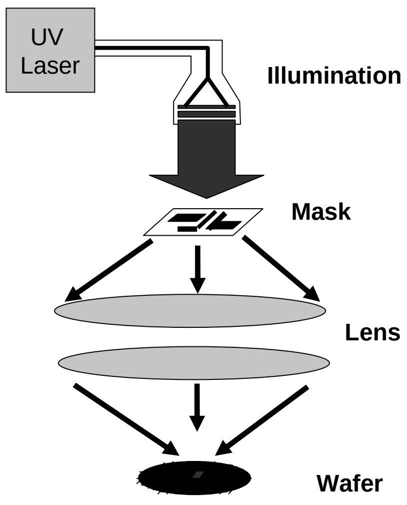


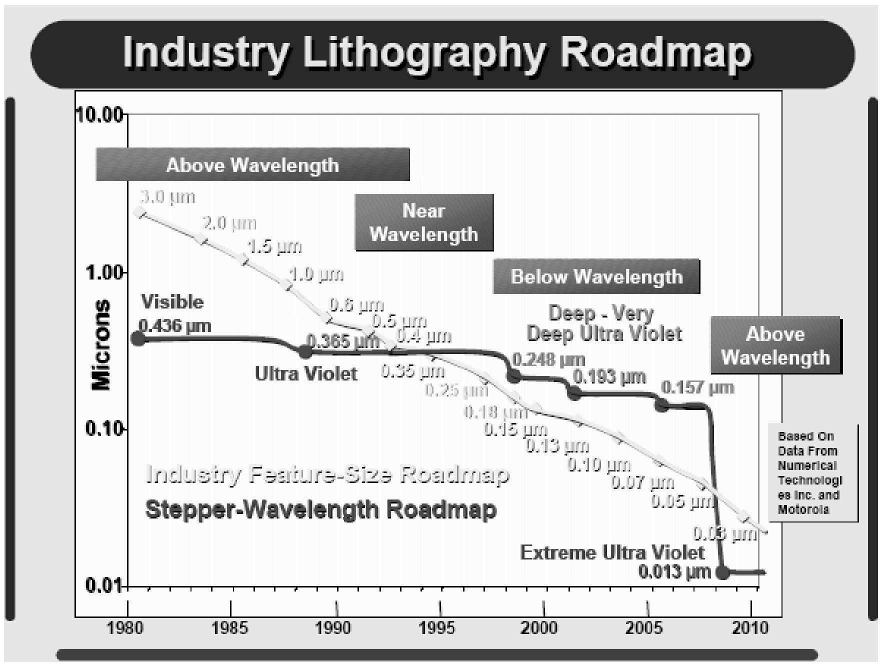
Logic and circuit simulator formats
Logic netlists
Circuit netlists
Logic/circuit database
RCX
Netlist comparison checker (LVS)
Layout debugger
Graphics databases
DFII
GDSII
Physical database
Design rules checker (DRC)
Resolution Enhancement Techniques (RET)
Interfaces
Tools

Source: Cadence Assura Developer Guide

What is Resolution Enhancement Technology (RET)? Why?

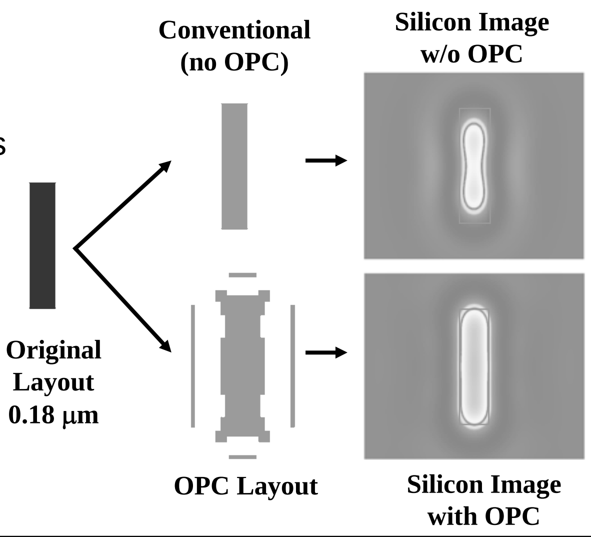
- All IC layers are formed by *Lithography*
- The light interacts with a reticle (aka mask) and the lens to form a reduced image in photoresist
- Sub-Wavelength Gap
  - Feature Size < Optical Wavelength
  - Problem:
    - Critical Dimension Variation





Optical Proximity Correction (OPC)

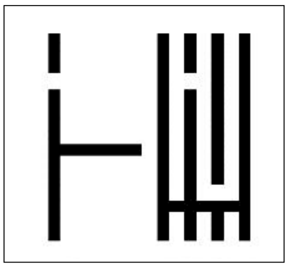
- ◆ Corrective modifications to improve process control
  - improve yield
  - improve device performance

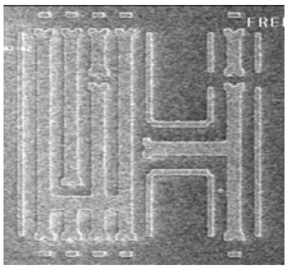


Source: Prof. Martin Wang, DAC 2004

Impact on Physical Design: OPC

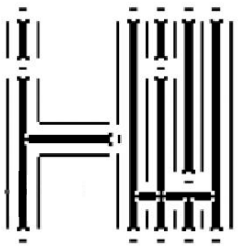
Designed Layout

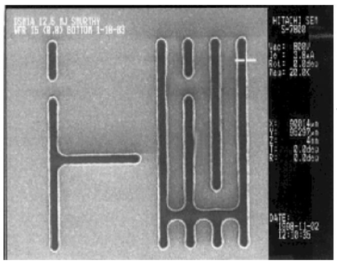




Mask

Final Layout





Wafer

Source: Mentor Graphics, ISPD 2001

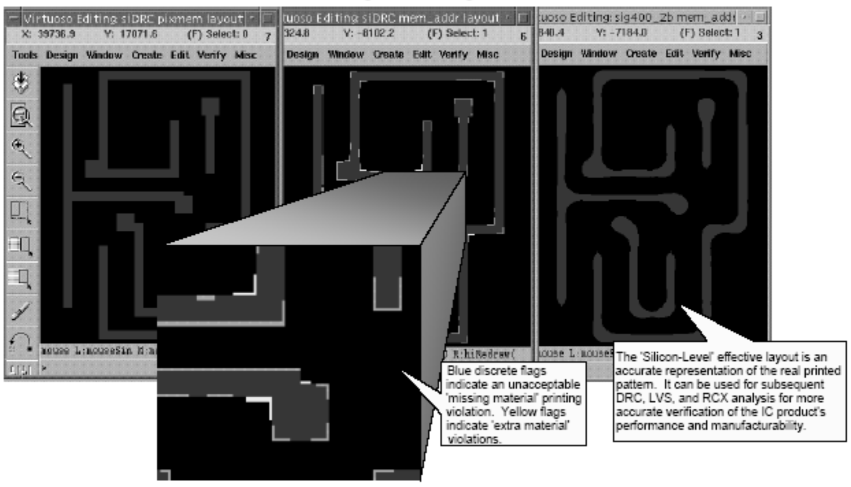
cadence

Assura 'Silicon-Level' Verification

Layout Editor Output of As Designed Layout

Output of 'Silicon-Level' Verification Printing Violation Flags

Output of 'Silicon-Level' Layout


Virtuoso Editing: siDRC pixmem layout
Tools Design Window Create Edit Verify Misc
tuoso Editing: siDRC mem_addr layout
Design Window Create Edit Verify Misc
tuoso Editing: sig400_2b mem_addr
Design Window Create Edit Verify Misc

Blue discrete flags indicate an unacceptable 'missing material' printing violation. Yellow flags indicate 'extra material' violations.

The 'Silicon-Level' effective layout is an accurate representation of the real printed pattern. It can be used for subsequent DRC, LVS, and RCX analysis for more accurate verification of the IC product's performance and manufacturability.

Phase Shifting Mask (PSM)

◆ Uses phase-modulation at the mask level to further the resolution capabilities of optical lithography

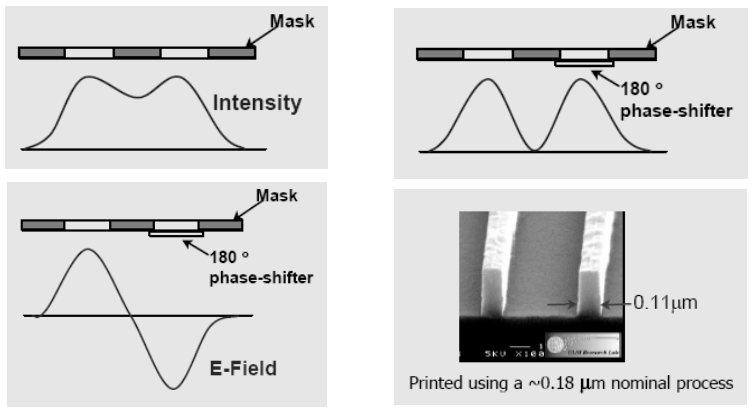


Figure Courtesy of Numerical Technologies, Inc.

Advantages of PSM

◆ Smaller feature sizes

◆ Improved yield (process latitude)

◆ Dramatically extends useful life of current equipment

◆ Performance Boost

◆ Chip Area/Cost Advantage for Embedded Systems

- Modified Architecture Based on Higher Speed Busses

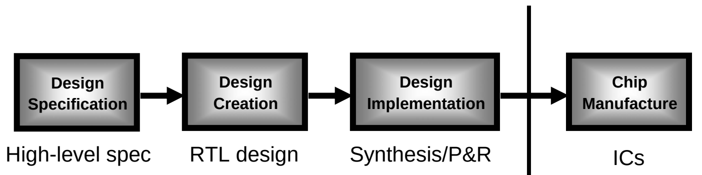
Source: Motorola

What we will cover in this topic…

- ◆ Static analysis
  - Linting
  - Clock domain checking
  - Static timing analysis
  - Signal Integrity
- ◆ Physical verification
  - DRC
  - LVS
  - Design for Manufacturability (DfM)
- ◆ Testing

Remember…

- ◆ Functional Verification vs. Testing



Verification

- Object → design (SW)
- Methodologies
  - ➢ Simulation
  - ➢ Emulation
  - ➢ Formal techniques

Testing

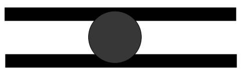
- Object → chip (HW)
- Methodologies
  - ➢ ATPG
  - ➢ Fault Simulation
  - ➢ Scan / BIST

Manufacturing Defects

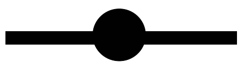
- ◆ Defects in chip manufacturing will lead to failure silicon or severe reliability problems
- ◆ Defects occur at
  - ● Front end of the line (FEOL), i.e., devices
  - ● Back end of the line (BEOL), i.e., interconnect and vias
  - ● BEOL defects are increasingly dominant
- ◆ Main BEOL failure mode: spot defects

Spot Defect Classification

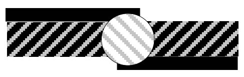
- Intra-layer extra material
  - ➔ short-circuit faults



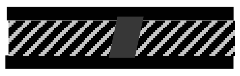
- Intra-layer missing material
  - ➔ open circuit faults



- Inter-layer extra material
  - ➔ open faults (blocked vias)



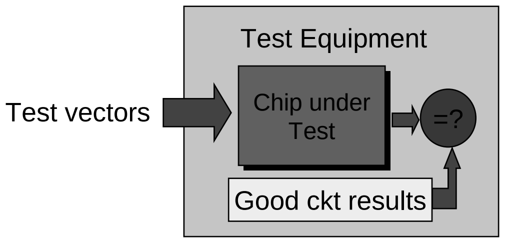
- Inter-layer missing material
  - ➔ short-circuit faults (oxide pinholes)



Source: Fujitsu Labs

Testing is ---

◆Apply test vectors on real chip and check if there is any unexpected result or the chip becomes malfunctioned



Test Vector (Pattern) Generation

◆Methods

- Random
- Automatic (Deterministic) Test Pattern Generation (ATPG)

◆Goal: to generate quality test vectors that can detect as many defects as possible

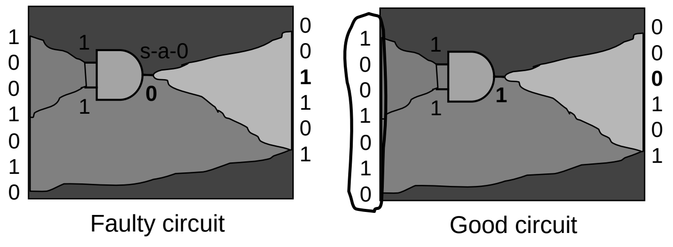
➔ Need a quantitative metric to measure the quality of tests

Test Measurement

- *How good is your test?*
- Fault model
  - To characterize defects (faults) into different categories
  - Short / Open faults
    - ➔ For each node in the circuit, it has either short or open fault
    - ➔ Cover > 90% common defects
    - ➔ Correspond to stuck-at-1 / stuck-at-0 faults
  - ➔ With fault model, defects can then be enumerated and test quality can then be measured
- Fault coverage
  - (#detected faults) / (# total faults)

For a test vector to detect a fault…

1. It must generate different responses at the fault location for the good and faulty circuits
2. The above difference must be propagated to primary outputs for observation



Automatic Test Pattern Generation (ATPG)

- Given a fault list, generate as few vectors and detect as many faults as possible
  - Usually use stuck-at-1/0 (s-a-0/1) fault model
  - Assume single-fault model
  - Branch-and-bound algorithms
  - (Details to cover later)

Remember ---

- For a test vector to detect a fault…

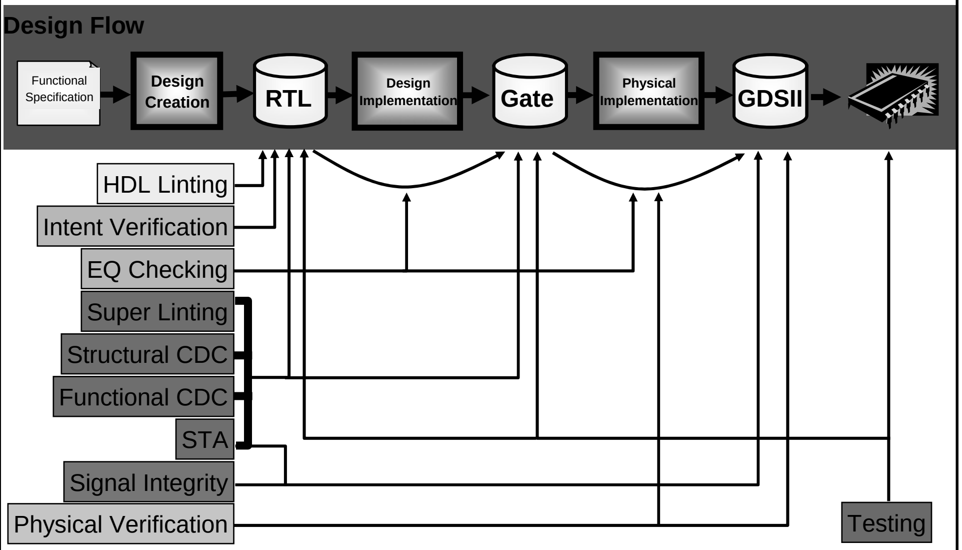
1. It must generate different responses at the fault location for the good and faulty circuits
   ➔ Controllability problem
2. The above difference must be propagated to primary outputs for observation
   ➔ Observability problem

- Registers are neither controllable nor observable in a chip
  ➔ Sequential ATPG is very difficult (rarely used)
- Use *combinational ATPG + DfT* techniques

Design for Testability (DfT) Techniques

- Scan Design
  - Add a “scan chain” on registers in order to “scan in” test vectors and “scan out” simulation results
  - Scan and functional modes
- Test Point Insertions
  - Insert circuitry in critical internal points of the circuit to increase controllability or observability

Various Verification Techniques



What we have learned about various verification techniques

- Simulation
- Emulation / hardware acceleration / rapid prototyping
- Formal verification
  - Property checking
  - Equivalence checking
- Static analysis
  - Linting
  - Clock domain checking
  - Static timing analysis
  - Signal Integrity
- Physical verification
  - DRC
  - LVS
  - Design for Manufacturability (DfM)
- Testing

- What types of verification techniques have you used?
- What kinds of new verification techniques will be useful for your next design?
- How efficient is your current verification practice?
- Do you expect any new verification techniques? In what area?