Basics of VLSI Design and Test

M. Tehranipoor

Introduction to Hardware Security & Trust
University of Florida
Nanoelectronics

- Power will continue to increase
  - Low-leakage / leakage-tolerance
  - Dynamic power management
- Process variation will be a key issue
  - Design for Manufac. (DFM), Reliability
  - Arch. / design methods for new devices

- New Devices Emerge
  - SOI, FinFET, TriGate

- Carbon Nanotubes
- Quantum computing
- Molecular transistors etc.
Different applications have different power-performance demands.

VLSI Applications

• Ultralow power applications: medical, space, specific sensor network etc.

• Portable applications: mobile computing, wireless, multimedia etc.

• General purpose computing: internet server, database server, real-time jobs etc.
IC Design and Test Flow

1. Design Spec.
2. IC Design
3. Fab
4. Wafer test
5. Package test
6. Burn in
7. Assembly
8. Customer
Nanoscale VLSI System

VLSI System (e.g. Baseband Processor, DSP, ASIC)

Architectural blocks

Logic Blocks (Controller, Datapaths)

Interconnect

Memory

System -> Architecture -> Logic -> Transistor
Nanoscale Transistor

- Basic building block for the integrated circuit: both logic and memory
- Minimum feature size is now well below 100nm
Exponential Growth in Computing Power

What is the key to the growth in computing power?

Source: Intel
“The complexity for minimum component costs has increased at a rate of roughly a factor of two per year ... Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. ...”

*Electronics Magazine, 19 April 1965*
Technology Scaling

Dimensions scale down by 30%

<table>
<thead>
<tr>
<th>Description</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double transistor density</td>
<td>Rounds transistor density</td>
</tr>
<tr>
<td>Oxide thickness scales down</td>
<td>Faster transistor, higher</td>
</tr>
<tr>
<td>performance</td>
<td>Lower active power</td>
</tr>
<tr>
<td>Vdd &amp; Vt scaling</td>
<td>Lower active power</td>
</tr>
</tbody>
</table>
nMOS Transistor

- If the gate is “high”, the switch is on
- If the gate is “low”, the switch is off
pMOS Transistor

- If the gate is “low”, the switch is on
- If the gate is “high”, the switch is off
Solution: CMOS

- No static current flow
- Less current means less power
CMOS Inverter First-Order DC Analysis

- High noise margin
- Ratioless
- Low output impedance
- Extremely high input impedance
- No static power
CMOS Inverter: Transient Response

\[ V_{in} = V_{DD} \rightarrow 0 \]

\[ V_{in} = 0 \rightarrow V_{DD} \]

Output: Low-to-High

High-to-Low

To reduce delay:
- Reduce CL
- Reduce Rp, n
- Increase W/L ratio

CL is composed of the drain diffusion capacitances of the NMOS and PMOS transistors, the capacitance of connecting wires, and the input capacitance of the fan-out gates.
Performance Characterization

- Interconnect delay

![Diagram showing interconnect delay with voltage levels and time axis](image-url)
Boolean Algebra

- **Basic operators**
  - **AND**
    
    \[ f(A, B) = A \cdot B = A \cap B \]
    
    - Diagram of AND gate
  
    | A | B | A\cdot B |
    |---|---|---------|
    | 0 | 0 |   0     |
    | 0 | 1 |   0     |
    | 1 | 0 |   0     |
    | 1 | 1 |   1     |

  - **OR**
    
    \[ f(A, B) = A + B = A \cup B \]
    
    - Diagram of OR gate
  
    | A | B | A+B |
    |---|---|-----|
    | 0 | 0 |   0 |
    | 0 | 1 |   1 |
    | 1 | 0 |   1 |
    | 1 | 1 |   1 |
Boolean Algebra

- **Basic operators**
  - **NAND**
    
    \[
    f(A, B) = \overline{A \cdot B} = A \cap \overline{B}
    \]
    
    ![NAND Circuit Diagram]
    
    | A | B | \overline{A \cdot B} |
    |---|---|---------------------|
    | 0 | 0 | 1                   |
    | 0 | 1 | 1                   |
    | 1 | 0 | 1                   |
    | 1 | 1 | 0                   |
  
  - **NOR**
    
    \[
    f(A, B) = \overline{A + B} = A \cup \overline{B}
    \]
    
    ![NOR Circuit Diagram]
    
    | A | B | \overline{A + B} |
    |---|---|------------------|
    | 0 | 0 | 1                |
    | 0 | 1 | 0                |
    | 1 | 0 | 0                |
    | 1 | 1 | 0                |
Boolean Algebra

- Basic operators
  - XOR

\[ f(A, B) = A \oplus B \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A \oplus B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
DeMorgan’s Theorem

\[\overline{A + B} = \overline{A} \cdot \overline{B} \]

\[\overline{A \cdot B} = \overline{A} + \overline{B} \]
# Boolean Algebra

## Truth Tables

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Representation of the function to be realized**
- **Sum of Products representation**
  - **Sum of minterms**
    \[ F = \overline{A}BC + \overline{A}B\overline{C} + \overline{A}BC + \overline{A}BC + AB\overline{C} \]
- **Product of Sums representation**
  - **Product of maxterms**
    \[ F = (A + B + C) \cdot (A + \overline{B} + \overline{C}) \cdot (\overline{A} + \overline{B} + \overline{C}) \]
CMOS Logic Implementations

- Inverter
CMOS Logic Implementations

- NOR

![NOR Circuit Diagram]
CMOS Logic Implementations

- NAND
CMOS Logic Implementations

- Multi-input NOR
CMOS Logic Implementations

- General CMOS combinational logic

![Diagram of CMOS logic circuit](image)
What is VLSI design?

- The process of creating an integrated circuit from specifications to fabrication

What is an integrated circuit?

- A single integrated component that contains all the primary elements of an electrical circuit: transistors, wiring, resistors, capacitors, etc.
VLSI Design Automation

- Large number of components
- Optimize requirements for **higher performance**
  - Performance relates to speed, power and size.
- Time to market competition
- Cost
  - Using computer makes it cheaper by reducing time-to-market.
VLSI Design Cycle

System Specifications

Functional Design

Logic Design

Circuit Design

X = (AB*CD) + …
VLSI Design Cycle

1. Physical Design
2. Fabrication
3. Packaging
4. IC Test
Semiconductor Processing

- How do we make a transistor?

- How do you control where the features get placed?
  - Photo lithography masks
Wafer Processing
Intel 4004

- First microprocessor
- Designed in 1971
- 2300 transistors
- 10-um process
- ~100 KHz
Intel Itanium Processor

- Released in 2005
- 1.72 Billion transistors
- 90-nm process
- 2 GHz
Design Methodology

- Functional specification
  - What does the chip do?
- Behavioral specification
  - How does it do it? (abstractly)
- Logic design
  - How does it do it? (logically)
- Layout
  - How does it do it? (physically)
Design Constraints

- **Budget**
  - Total cost

- **Silicon area**

- **Power requirements**
  - Dynamic
  - Static

- **Speed**
  - Performance

- **Schedule**
  - Time to market
Functional Specification

- Full adder

```
  Cin

X

Y

Cout

S
```
Behavioral Specification

- VHDL
- Verilog

entity adder is
  -- i0, i1 and the carry-in ci are inputs of the adder.
  -- s is the sum output, co is the carry-out.
  port (i0, i1 : in bit; ci : in bit; s : out bit; co : out bit);
end adder;
architecture rtl of adder is
  begin -- This full-adder architecture contains two concurrent assignment.
    -- Compute the sum. s <= i0 xor i1 xor ci;
    -- Compute the carry. co <= (i0 and i1) or (i0 and ci) or (i1 and ci);
  end rtl;
module fulladder (a,b,cin,sum,cout);
    input a,b,cin;
    output sum,cout;

    reg sum,cout;
    always @ (a or b or cin)
    begin
        sum <= a ^ b ^ cin;
        cout <= (a & b) | (a & cin) | (b & cin);
    end
endmodule
Logic Design

Full Adder Truth Table

<table>
<thead>
<tr>
<th>CARRY IN</th>
<th>input B</th>
<th>input A</th>
<th>CARRY OUT</th>
<th>SUM digit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Transistor Schematic
Layout
Design Process is Iterative

- Behavioral
- Structural
- Synthesis
- PNR

Simulation-Based Verification
Simulation-Based Verification
Simulation-Based Verification
Simulation/Emulation-Based Verification

PNR: Placement and routing
VLSI Design Methodologies

- Full custom
  - Design for performance-critical cells
  - Very expensive
- Standard cell
  - Faster
  - Performance is not as good as full custom
- Gate array
- Field Programmable Gate Array
# Comparison of Design Styles

<table>
<thead>
<tr>
<th>Production Volume:</th>
<th>Mass Production Volume</th>
<th>Medium Production Volume</th>
<th>Medium Production Volume</th>
<th>Low Production Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity:</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Area</th>
<th>Full Custom</th>
<th>Standard Cell</th>
<th>Gate Array</th>
<th>FPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compact</td>
<td>Moderate</td>
<td>Moderate</td>
<td>Low</td>
<td></td>
</tr>
</tbody>
</table>

1. **Full Custom**:
   - High complexity
   - Compact area
   - High performance

2. **Standard Cell**:
   - Moderate complexity
   - Moderate area
   - Moderate performance

3. **Gate Array**:
   - Moderate complexity
   - Moderate area
   - Moderate performance

4. **FPGA**:
   - Low complexity
   - Large area
   - Low performance

---

23 January 2018
VLSI Chip Yield

- A manufacturing defect in the fabrication process causes electrically malfunctioning circuitry.
- A chip with no manufacturing defect is called a good chip.
  - The defective ones are called bad chips.
- Percentage of good chips produced in a manufacturing process is called the yield.
- Yield is denoted by symbol $Y$.

\[
Y = \frac{\text{# of good die}}{\text{# total manufactured die}}
\]

- How to separate bad chips from the good ones?

TEST ALL CHIPS
In simple terms, TEST identifies the defective chips

Some bad chips (■) are easy to find

Some other are difficult (■)

Test is associated with

- Cost
- Return On Investment (ROI)
  - ¥ € $ - Money
Testing Principle

Input Patterns

0 0 1
0 1 1
0 1 0
0 1 1
1 0 0

Output Patterns

1 1 1
0 1 0
1 0 0
1 0 1
1 0 0

Digital Circuit

Stored Correct Response

Comparator

Test Result

Functional Test Method – Not very efficient
Contract between design house and fab vendor

- Design is complete and checked (verified)
- Fab vendor: How will you test it?
- Design house: I have checked it and ...
- Fab vendor: But, how would you test it?
- Design house: Why is that important?
- **complete the story**

- That is one reason for design-for-testability, test generation etc.
Contract between design …

Hence:
- “Test” must be comprehensive
- It must not be “too long”

Issues:
- Model possible defects in the process
  - Understand the process
- Develop simulator and fault simulator
- Develop test generator
- Methods to quantify the test efficiency
  - Fault coverage
Ideal Tests

- Ideal tests detect **all** defects produced in the manufacturing process.
- Ideal tests pass all functionally good devices.
- Very large numbers and varieties of possible defects need to be tested.
- Difficult to generate tests for some real defects. *Defect-oriented testing is an open problem.*
Real Tests

- Based on analyzable fault models, which may not map on real defects.
- Incomplete coverage of modeled faults due to high complexity.
- Some good chips are rejected. The fraction (or percentage) of such chips is called the yield loss.
- Some bad chips pass tests. The fraction (or percentage) of bad chips among all passing chips is called the defect level.
Level of testing (1)

- Levels
  - Chip
  - Board
  - System
    - Boards put together
    - System-on-Chip (SoC)
  - System in field

- Cost – Rule of 10
  - It costs 10 times more to test a device as we move to higher level in the product manufacturing process
VLSI Defects

Unclustered defects
Wafer yield = 12/22 = 0.55

Clustered defects (VLSI)
Wafer yield = 17/22 = 0.77
Scan Flip-Flop

Flip-Flop

D Q 
CK

Scan Flip-Flop (SFF)

D Q 
CK
SI
TC

 CK
     Master open | Slave open     

 TC
     Normal mode, D selected | Scan mode, SD selected 

23 January 2018
Adding Scan Structure

Combinational logic

PI → SFF → SFF → SFF → SCANOUT

PO

Not shown: CK or MCK/SCK feed all SFFs (scan Flip-flops).

Scan Path

Also called Scan Chain

TC or TCK

SCANIN
Scan Design

Primary Inputs

Scan-in (SI)

0 1 0 0 1 0 1 1

Scan Flip-Flop

Circuit-Under-Test (CUT)

1 1 0 1 0 1 1

Primary Outputs

Scan-out (SO)

1 0 1 0 0 0 1 0

Structural Test Method – Extremely efficient
ADVANTEST Model T6682 ATE

Testers are very expensive ($150K – $20M)
Sub-Wavelength

WYSINWYG

What You See Is Not What You Get

Process variations

No two transistors have the same parameters