Design Methodologies
The Design Problem

A growing gap between design complexity and design productivity

Source: sematech97
Design Methodology

• Design process traverses iteratively between three abstractions: behavior, structure, and geometry
• More and more automation for each of these steps
Design Analysis and Verification

- Accounts for largest fraction of design time
- More efficient when done at higher levels of abstraction - selection of correct analysis level can account for multiple orders of magnitude in verification time
- Two major approaches:
  » Simulation
  » Verification
Digital Data treated as Analog Signal

Circuit Simulation
Both Time and Data treated as Analog Quantities
Also complicated by presence of non-linear elements
(relaxed in timing simulation)
Representing Data as Discrete Entity

Discretizing the data using switching threshold

The linear switch model of the inverter
Circuit versus Switch-Level Simulation
Structural Description of Accumulator

Design defined as composition of register and full-adder cells ("netlist")

Data represented as \{0,1,Z\}

Time discretized and progresses with unit steps

Description language: VHDL
Other options: schematics, Verilog
Behavioral Description of Accumulator

entity accumulator is
  port (  
    DI : in integer;  
    DO : inout integer := 0;  
    CLK : in bit
  );
end accumulator;

architecture behavior of accumulator is
begin
  process(CLK)
    variable X : integer := 0;  -- intermediate variable
  begin
    if CLK = '1' then
      X <= DO + D1;
      DO <= X;
    end if;
  end process;
end behavior;

Design described as set of input-output relations, regardless of chosen implementation

Data described at higher abstraction level ("integer")
Behavioral simulation of accumulator

Discrete time

Integer data

(Synopsys Waves display tool)
Timing Verification

Critical path

Enumerates and ranks critical timing paths

No simulation needed!

(Synopsys-Epic Pathmill)
Issues in Timing Verification

False Timing Paths

In

4-bit adder

MUX

Out

bypass
Digital Circuit Implementation Approaches

Custom

Semi-custom

Cell-Based
- Standard Cells
- Compiled Cells
- Macro Cells

Array-Based
- Pre-diffused (Gate Arrays)
- Pre-wired (FPGA)
Custom Design – Layout Editor

Magic Layout Editor (UC Berkeley)
Symbolic Layout

- Dimensionless layout entities
- Only topology is important
- Final layout generated by “compaction” program

Stick diagram of inverter
Cell-based Design (or standard cells)

Routing channel requirements are reduced by presence of more interconnect layers.

Rows of Cells

Logic Cell

Feedthrough Cell

Routing Channel

Functional Module (RAM, multiplier, …)
Standard Cell — Example

[Brodersen92]
Standard Cell - Example

3-input NAND cell
(from Mississippi State Library)
characterized for fanout of 4 and
for three different technologies

<table>
<thead>
<tr>
<th>Fanout 4x</th>
<th>0.5 µm</th>
<th>1.0 µm</th>
<th>2.0 µm</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1_tphi</td>
<td>0.595</td>
<td>0.711</td>
<td>0.919</td>
</tr>
<tr>
<td>A1_tplh</td>
<td>0.692</td>
<td>0.933</td>
<td>1.360</td>
</tr>
<tr>
<td>B1_tphi</td>
<td>0.591</td>
<td>0.739</td>
<td>1.006</td>
</tr>
<tr>
<td>B1_tplh</td>
<td>0.620</td>
<td>0.825</td>
<td>1.181</td>
</tr>
<tr>
<td>C1_tphi</td>
<td>0.574</td>
<td>0.740</td>
<td>1.029</td>
</tr>
<tr>
<td>C1_tplh</td>
<td>0.554</td>
<td>0.728</td>
<td>1.026</td>
</tr>
</tbody>
</table>
Automatic Cell Generation

Random-logic layout generated by CLEO cell compiler (Digital)
Module Generators — Compiled Datapath

Advantages: One-dimensional placement/routing problem
Macrocell Design Methodology

**Floorplan:**
Defines overall topology of design, relative placement of modules, and global routes of busses, supplies, and clocks.
Macrocell-Based Design Example

Video-encoder chip [Brodersen92]
Gate Array — Sea-of-gates

- Rows of uncommitted cells
- Routing channel

Uncommitted Cell

Committed Cell (4-input NOR)
Sea-of-gate Primitive Cells

Using oxide-isolation

Using gate-isolation
Sea-of-gates

Random Logic

Memory Subsystem

LSI Logic LEA300K (0.6 µm CMOS)
Prewired Arrays

Categories of prewired arrays (or field-programmable devices):
- Fuse-based (program-once)
- Non-volatile EPROM based
- RAM based
Programmable Logic Devices

PLA

PROM

PAL

Digital Integrated Circuits

Design Methodologies

© Prentice Hall 1995
EPLD Block Diagram

Primary inputs

Macrocell

Courtesy Altera Corp.
Field-Programmable Gate Arrays
Fuse-based

Standard-cell like floorplan
Interconnect

Programmed interconnection

Antifuse

Horizontal tracks

Input/output pin

Vertical tracks

Programming interconnect using anti-fuses
Field-Programmable Gate Arrays

RAM-based

- CLB
- Horizontal routing channel
- Vertical routing channel
- Switching matrix
- Interconnect point
RAM-based FPGA

Basic Cell (CLB)

Combinational logic

Storage elements

Any function of up to 4 variables

Any function of up to 4 variables

Courtesy of Xilinx
RAM-based FPGA

Xilinx XC4025
Taxonomy of Synthesis Tasks

<table>
<thead>
<tr>
<th>Architectural Level</th>
<th>Logic Level</th>
<th>Circuit Level</th>
</tr>
</thead>
</table>
| (i: 1..16) :: sum = sum*\(z^{-1}\) + coeff[i]*ln*\(z^{-1}\)| state | \(a\) \(b\) \(c\) 
| Behavioral View     | Logic Synthesis | Circuit Synthesis |
| Structural View     | mem \(\cdot\) fsm | \(D\) \(c\) \(x\) | \(t_p\) |
Design for Test
Validation and Test of Manufactured Circuits

Goals of Design-for-Test (DFT)
Make testing of manufactured part swift and comprehensive

DFT Mantra
Provide controllability and observability

Components of DFT strategy
• Provide circuitry to enable test
• Provide test patterns that guarantee reasonable coverage
Test Classification

- **Diagnostic test**
  - used in chip/board debugging
  - defect localization

- **“go/no go” or production test**
  - Used in chip production

- **Parametric test**
  - $x \in [v, i]$ versus $x \in [0, 1]$
  - check parameters such as $NM, V_t, t_p, T$
Design for Testability

(a) Combinational function

\[ 2^N \text{ patterns} \]

(b) Sequential engine

\[ 2^{N+M} \text{ patterns} \]

Exhaustive test is impossible or unpractical
Problem: Controllability/Observability

- **Combinational Circuits:**
  controllable and observable - relatively easy to determine test patterns

- **Sequential Circuits:** State!
  Turn into combinational circuits or use self-test

- **Memory:** requires complex patterns
  Use self-test
Test Approaches

- Ad-hoc testing
- Scan-based Test
- Self-Test

Problem is getting harder

- increasing complexity and heterogeneous combination of modules in system-on-a-chip.
- Advanced packaging and assembly techniques extend problem to the board level
Generating and Validating Test-Vectors

- Automatic test-pattern generation (ATPG)
  - for given fault, determine excitation vector (called test vector) that will propagate error to primary (observable) output
  - majority of available tools: combinational networks only
  - sequential ATPG available from academic research

- Fault simulation
  - determines test coverage of proposed test-vector set
  - simulates correct network in parallel with faulty networks

- Both require adequate models of faults in CMOS integrated circuits
Fault Models

Most Popular - “Stuck - at” model

Covers almost all (other) occurring faults, such as opens and shorts.

\[ \alpha, \gamma : x_1 \text{ sa1} \]
\[ \beta : x_1 \text{ sa0 or } x_2 \text{ sa0} \]
\[ \gamma : Z \text{ sa1} \]
Problem with stuck-at model: CMOS open fault

Sequential effect
Needs two vectors to ensure detection!

Other options: use stuck-open or stuck-short models
This requires fault-simulation and analysis at the switch or transistor level - Very expensive!
Problem with stuck-at model: CMOS short fault

Causes short circuit between Vdd and GND for A = C = 0, B = 1

Possible approach: Supply Current Measurement (IDDQ)
but: not applicable for gigascale integration
Path Sensitization

Goals: Determine input pattern that makes a fault controllable (triggers the fault, and makes its impact visible at the output nodes)

Techniques Used: D-algorithm, Podem
Ad-hoc Test

Inserting multiplexer improves testability
Scan-based Test

ScanIn → Register → Combinational Logic A → Register → Combinational Logic B → ScanOut

In → Logic A

Out
Polarity-Hold SRL (Shift-Register Latch)

Introduced at IBM and set as company policy
Scan-Path Register

Diagram of a scan-path register with inputs and control signals.
Scan-based Test — Operation

\[ \text{ScanIn} \quad \text{Test} \quad \text{Latch} \quad \text{ScanOut} \]

\[ \text{Latch} \quad \text{ScanIn} \quad \text{Test} \quad \text{Latch} \quad \text{ScanOut} \]

\[ \text{Test} \quad \phi_1 \quad N \text{ cycles} \quad \text{scan-in} \]

\[ \phi_2 \quad \text{1 cycle evaluation} \]

\[ \phi_1 \quad \text{N cycles scan-out} \]
Scan-Path Testing

Partial-Scan can be more effective for pipelined datapaths
Boundary Scan (JTAG)

Board testing becomes as problematic as chip testing
Rapidly becoming more important with increasing chip-complexity and larger modules
Linear-Feedback Shift Register (LFSR)

Pseudo-Random Pattern Generator
Signature Analysis

Counts transitions on single-bit stream
≡ Compression in time
BILBO

\[ \begin{array}{c|c|c|c|c|c|c|c}
B_0 & B_1 & D_0 & D_1 & D_2 & S_0 & S_1 & S_2 \\
\hline
1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array} \]

Operation mode

- Normal
- Scan
- Pattern generation or Signature analysis
- Reset
BILBO Application

ScanIn

In

BILBO-A

Combinational Logic

BILBO-B

Combinational Logic

ScanOut

Out
Memory Self-Test

Patterns: Writing/Reading 0s, 1s, Walking 0s, 1s, Galloping 0s, 1s