Cell-Based Design Flow Concept

Speaker: Y.-J. Huang and C.-S. Ho
Cell-Based Design Flow

System Level
- MATLAB/ C/ C++/ System C/ ADS/ Covergen (MaxSim)
- Memory Generator

RTL Level
- Verilog/ VHDL
- NC-Verilog/ ModelSim Debussy (Verdi)/ VCS
- Syntest

Logic Synthesis
- Design/ Power Compiler
- DFT Compiler/ TetraMAX
- NC-Verilog/ ModelSim Debussy (Verdi)/ VCS

Design for Test
- Conformal/ Formality
- SOC Encounter
- Physical Compiler/ Magma Blast Fusion

Gate Level
- Gate Level

Layout Level
- GDS II
- DRC/ LVS (Calibre)

Post-Layout Verification
- PVS: Calibre xRC/ NanoSim (Time/ Power Mill)

Tape Out

Advanced Reliable Systems (ARES) Lab.
Subjects

- Verilog and Simulation
- Design Compiler
- SOC Encounter
Verilog
Synthesizable Verilog

- Verilog Basis
  - parameter declarations
  - wire, wand, wor declarations
  - reg declarations
  - input, output, inout
  - continuous assignment
  - module instructions
  - gate instructions
  - always blocks
  - task statement
  - function definitions
  - for, while loop

- Synthesizable Verilog primitives cells
  - and, or, not, nand, nor, xor, xnor
  - bufif0, bufif1, notif0, notif1
Operators

- Binary bit-wise (~, &, |, ^, ~^)
- Unary reduction (&, ~&, |, ~|, ^, ~^)
- Logical (!, &&, ||)
- 2’s complement arithmetic (+, -, *, /, %)
- Relational (>, <, >=, <=)
- Equality (==, !=)
- Logic shift (>>, <<)
- Conditional (?:
- Concatenation ({}
Notice Before Synthesis

☐ Your RTL design
  ■ Functional verification by some high-level language
    ☐ Also, the code coverage of your test benches should be verified (i.e. VN)
  ■ Coding style checking (i.e. n-Lint)
    ☐ Good coding style will reduce most hazards while synthesis
    ☐ Better optimization process results in better circuit performance
    ☐ Easy debugging after synthesis

☐ Constraints
  ■ The area and timing of your circuit are mainly determined by your circuit architecture and coding style
  ■ There is always a trade-off between the circuit timing and area
  ■ In fact, a super tight timing constraint may be worked while synthesis, but failed in the Place & Route (P&R) procedure
Design Compiler

REF:
• CIC Training Manual – Logic Synthesis with Design Compiler, July, 2006
• TSMC 0.18um Process 1.8-Volt SAGE-X™ Stand Cell Library Databook, September, 2003
• TPZ973G TSMC 0.18um Standard I/O Library Databook, Version 240a, December 10, 2003
• Artisan User Manual
What is Synthesis

- Synthesis = translation + optimization + mapping

```
if(high_bits == 2'b10)begin
    residue = state_table[i];
    end
else begin
    residue = 16'h0000;
end
```

The synthesis is constraint driven and technology independent!!

Advanced Reliable Systems (ARES) Lab.
Logic Synthesis Overview

- RTL Design
  - HDL Compiler
    - Design Compiler
      - Optimized Gate-Level Netlist
  - Architecture Optimization
  - Logic Optimization
  - Design Library
  - Technology Library
  - Lib Compiler
  - DW Developer

Advanced Reliable Systems (ARES) Lab.
Compile

RTL code or netlist → Compile → Optimized Design (Gate-Level Netlist)

Attributes & Constraints → Compile → Schematic

Technology Library → Flatten Technology Library

Logic Level Optimization → Gate Level Optimization

Technology Library → Map

Advanced Reliable Systems (ARES) Lab.
Static Timing Analysis

- Main steps of STA
  - Break the design into sets of timing paths
  - Calculate the delay of each path
  - Check all path delays to see if the given timing constraints are met

- Four types of paths
  - Register - Register (Reg - Reg)
  - Primary Input - Register (PI - Reg)
  - Register - Primary Output (Reg - PO)
  - Primary Input - Primary Output (PI - PO)
Static Timing Analysis (Cont’)

- Setup Time

To meet the setup time requirement:

- \( T_{\text{require}} \geq T_{\text{arrival}} \)

Reg to Reg

- \( T_{\text{arrival}} = T_{\text{clk1}} + T_{\text{DFF1}(\text{clk} \rightarrow \text{Q})} + T_{\text{PATH}} \)
- \( T_{\text{require}} = T_{\text{clk2}} - T_{\text{DFF2}(\text{setup})} \)
- \( T_{\text{slack}} = T_{\text{require}} - T_{\text{arrival}} \) (\( T_{\text{slack}} > 0 \) denotes “no timing violation”)

Advanced Reliable Systems (ARES) Lab.
Memory Compiler

REF:
• CIC Training Manual – Logic Synthesis with Design Compiler, July, 2006
• TSMC 0.18um Process 1.8-Volt SAGE-X™ Stand Cell Library Databook, September, 2003
• TPZ973G TSMC 0.18um Standard I/O Library Databook, Version 240a, December 10, 2003
• Artisan User Manual
Memory Compiler

- **Artisan SRAM Types:**

<table>
<thead>
<tr>
<th>Generator</th>
<th>Product Name</th>
<th>Executable</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-Speed/Density Single-Port SRAM</td>
<td>SRAM-SP</td>
<td>ra1sh</td>
</tr>
<tr>
<td>High-Speed/Density Dual-Port SRAM</td>
<td>SRAM-DP</td>
<td>ra2sh</td>
</tr>
<tr>
<td>High-Density Single-Port SRAM</td>
<td>SRAM-SP-HD</td>
<td>ra1shd</td>
</tr>
<tr>
<td>High-Density Dual-Port SRAM</td>
<td>SRAM-DP-HD</td>
<td>ra2shd</td>
</tr>
<tr>
<td>Low-Power Single-Port SRAM</td>
<td>SRAM-SP-LP</td>
<td>ra1shl</td>
</tr>
</tbody>
</table>

- Only **ra1shd** and **ra2sh** are supported in school

- Generated files:
  - Memory Spec. (i.e. used for layout-replacement procedure in CIC flow)
  - Memory Data Sheet
  - Simulation models: Verilog Model & VHDL Model
  - Memory Libraries for P&R: Synopsys Model & VCLEF Footprint
  - Timing Files: TLF Model & PrimeTime Model

[REF: Artisan User Manual]

Advanced Reliable Systems (ARES) Lab.
**Single-Port SRAM**

![SRAM Diagram]

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic Pins</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLK</td>
<td>Input</td>
<td>Clock</td>
</tr>
<tr>
<td>WEN[*]</td>
<td>Input</td>
<td>Write enable, active low. *If word-write mask is enabled, this becomes a bus</td>
</tr>
<tr>
<td>CEN</td>
<td>Input</td>
<td>Chip enable, active low</td>
</tr>
<tr>
<td>OEN</td>
<td>Input</td>
<td>Tri-state output enable</td>
</tr>
<tr>
<td>A[m-1:0]</td>
<td>Input</td>
<td>Address (A[0]=LSB)</td>
</tr>
<tr>
<td>D[n-1:0]</td>
<td>Input</td>
<td>Data inputs (D[0]=LSB)</td>
</tr>
<tr>
<td>Q[n-1:0]</td>
<td>Output</td>
<td>Data outputs (Q[0]=LSB)</td>
</tr>
</tbody>
</table>
Waveforms for Single-Port SRAM

- Read Cycle

![Waveform diagram for Single-Port SRAM showing waveforms for CLK, CEN, WEN', A[j], and Q[i]. The diagram includes timing notations such as t_cyc, t_ckh, t_ckl, t_cs, t_ch, t_ws, t_wh, t_as, and t_ah.](image-url)
Waveforms for Single-Port SRAM (Cont’)

- Write Cycle

Advanced Reliable Systems (ARES) Lab.
Mask-Write SRAM

- Word Width: 64 bits
  - Word Partition Size: 32 bits
  - Mask Width = WEN Width = 2
  - WEN[1:0]
    - 11: No write
    - 10: Write to LSB part
    - 01: Write to MSB part
    - 00: Write to the whole word

1. \( WEN = 2'b11 \)  
   - No write

2. \( WEN = 2'b10 \)  
   - Write to LSB part

3. \( WEN = 2'b01 \)  
   - Write to MSB part

4. \( WEN = 2'b00 \)  
   - Write to the whole word

Advanced Reliable Systems (ARES) Lab.
Synthesis Flow

- Design Import
- Setting Design Environment
- Setting Clock Constraints
- Setting Design Rule Constraints
- Compile the Design

- DFT Insertion
- Compile After DFT
- Assign Violation Avoidance
- Naming Rule Changing
- Save Design

Advanced Reliable Systems (ARES) Lab.
VLSI Testing

- VLSI Testing
  - Introduction
  - Fault modeling
  - Test generation
- Design for Testability (DFT)
- Fault Simulation
Definitions

- **Design synthesis**
  - Give an I/O function, develop a procedure to manufacture a device using known materials and processes.

- **Verification**
  - Predictive analysis to ensure that the synthesized design will perform the given I/O function.

- **Testing**
  - A manufacturing step that ensures that the physical device, manufactured from the synthesized design, has no manufacturing defect.
Why Testing?

☐ Economy
  ■ Product quality
  ■ Product reliability

☐ The purpose of testing is to weed out all bad products before they are shipped to users
  ■ The number of bad products heavily affect the price of good products

☐ A profound understanding of the principles of manufacturing and test is essential for a designer to design a quality product
Defect, Fault, and Error

- **Defect**
  - A defect is an unintended difference between the implemented hardware and its intended design
  - It is caused during manufacture or the use of devices

- **Fault**
  - A representation of a physical defect at the abstracted function level

- **Error**
  - A wrong output signal produced by a defective circuit
  - It is caused by a fault or a design error

*Advanced Reliable Systems (ARES) Lab.*
Modern IC Testing

Devise under test (DUT)

Automatic test equipment (ATE)

Test program

Passed the test

Failed the test

Advanced Reliable Systems (ARES) Lab.
Cost of Test

- Design for testability (DFT)
  - Area overhead and yield reduction
  - Performance overhead

- Software processes of test
  - Test generation and fault simulation
  - Test programming and debugging

- Manufacturing test
  - Automatic test equipment (ATE) capital cost
  - Test center operational cost
Real defects are too numerous and hard to be analyzed
Fault

- Fault is a physical defect in a circuit/system
  - Permanent Fault: a fault that is continuous and stable, whose nature do not change before, during, and after testing
    - Hard fault or solid fault
  - Temporary fault: a fault that is present only part of the time, occurring at random moments and affecting the system for finite, but unknown, intervals of time
    - Transient fault, soft error
  - Intermittent fault: caused by non-environmental conditions
Fault Model and Error

- Fault model is a logic effect of a fault
  - Structural fault
    - Stuck-at-faults
    - Bridging fault
    - Open fault
    - Transition fault
    - Delay fault
  - Functional fault
    - RAM coupling and pattern-sensitive faults
- Error is manifestation of a fault that results in an incorrect module output or system state
Defect Level, Fault Coverage and Yield

- **Defect Level**
  - The fraction of devices that pass all the tests but still contain faults
  - \[ DL = 1 - Y(1 - FC) \]

- **Fault Coverage (FC)**
  - The measure of the ability of a test set \( T \) to detect a given set of faults
  - \[ FC = \frac{\text{No. of detected faults}}{\text{No. of possible faults}} \]
  - Can be determined by fault simulation

- **Yield (Y)**
  - \[ Y = \frac{\text{No. of good dies per wafer}}{\text{No. of dies per wafer}} \]
Defect Level and Fault Coverage

- DL is measured in terms of DPM (defects per million), and typical values claimed are less than 200DPM, or 0.02%
- Required FC for DL = 200DPM

<table>
<thead>
<tr>
<th>Y(%)</th>
<th>10</th>
<th>50</th>
<th>90</th>
<th>99</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC(%)</td>
<td>99.99</td>
<td>99.97</td>
<td>99.8</td>
<td>98</td>
</tr>
</tbody>
</table>
Stuck-at-fault

- Single stuck-at-fault: line has a constant value
- Multiple stuck-at-fault: several single SAFs occur at the same time

```
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>c(a/0)</th>
<th>c(a/1)</th>
<th>c(b/0)</th>
<th>c(b/1)</th>
<th>c(c/0)</th>
<th>c(c/1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
```
Bridging Faults

- Two or more normally distinct points (lines) are shorted together

- Two types of bridging faults
  - Input bridging
    - Can form wired logic or voting model
  - Feedback bridging
    - Can introduce feedback
Testing

- Testing = test generation + test application + output evaluation
- FC can be determined by fault simulation
- Cost of test generation (TG) depends on
  - Complexity of the fault model
  - Complexity of the TG algorithm
  - Complexity of the DUT
- A test set for a class of faults $F$ is a set of tests $T$ such that for any fault $f \in F$, there exists $t \in T$ such that $t$ detects $f$
### Test Generation by Truth Table

![Logic Diagram](attachment:image.png)

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>c(a/0)</th>
<th>c(a/1)</th>
<th>c(b/0)</th>
<th>c(b/1)</th>
<th>c(c/0)</th>
<th>c(c/1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- $T_{a/0} = \{11\}; T_{a/1} = \{01\}; T_{b/0} = \{11\}; T_{b/1} = \{10\}$
- $T_{c/0} = \{11\}; T_{c/1} = \{00\}$ or $\{01\}$ or $\{10\}$
- $T = \{01, 10, 11\}$

Advanced Reliable Systems (ARES) Lab.
Design for Testability (DFT)

- A fault is testable if there is a well-specified procedure to expose it, which can be implemented with a reasonable cost using current technique

- DFT
  - A class of design methodologies which put constraints on the design process to make test generation and diagnosis easier
Sequential Logic

- Harder to test
  - Sequential circuit has memory in addition to combinational logic
  - It takes more clock cycles to activate the fault and propagate the fault effect

- Example

\[
\begin{array}{cccc}
\times & \times & 1 & a \\
\times & \times & 0 & b \\
\times & \times & 0 & c \\
\times & \times & 0 & d \\
\end{array}
\]

- Example circuit diagram

---

Advanced Reliable Systems (ARES) Lab.
Yu-Jen Huang
Scan Cell

Combinational Logic

PI -> mux -> PO

PPI -> mux

SI -> ...... -> SO

T -> ...... -> clk

Advanced Reliable Systems (ARES) Lab.
Yu-Jen Huang
Automation Layout

REF:
• CIC Training Manual – Cell-Based IC Physical Design and Verification with SOC Encounter, July, 2006
Placement & Routing Flow

- Netlist (Verilog)
  - Timing Constraints (sdc)
  - IO Constraints (ioc)

1. IO, P/G Placement
2. Specify Floorplan
3. Timing Analysis
4. Pre-CTS Optimization
5. Power Planning
6. Power Analysis
7. Clock Tree Synthesis
8. Timing Analysis
9. Post-CTS Optimization
10. Power Route
11. SI Driven Route
12. Timing/SI Analysis

Advanced Reliable Systems (ARES) Lab.
Standard Cell

- The placement site gives the placement grid of a family of macros
IO, P/G Placement

- Determine the positions of the PADs
  - Functional IO PAD
  - Power/Ground PAD
  - Corner PAD
- Just for the connection of PAD power rings

Advanced Reliable Systems (ARES) Lab.
Floorplan

- Determine the related positions of Hard Blocks
  - The performance is highly affected
Power Planning

- Plan the power ring & power stripe
  - IR-drop consideration
Power Analysis

- IR-drop & electron migration
Power Route

- Connect the power pins of standard cells to the global power lines
Add IO Filler

- Fill the gap between PADs
  - Connect the PAD power rings
Routing

- Construct the final interconnections
Post-Layout Verification

- Post-Layout Verification do the following jobs:
  - DRC (Design Rule Check)
  - ERC (Electrical Rule Check)
  - LVS (Layout versus Schematic)
  - LPE/PRE (Layout Parasitic Extraction/ Parasitic Resistance Extraction) and Post-Layout Simulation
**DRC Flow**

- **Prepare Layout:**
  - Stream in GDSII
  - Add power PAD text
  - Stream out GDSII

- **Prepare DRC command file**
- **Run DRC**
- **View DRC error**

---

(Virtuosso Layout View)
Transistor-Level Post-Layout Simulation

Layout → Netlist/ Parasitic Extraction → SPICE Netlist → Post-Layout Simulation → Simulation Result

Calibre LPE/ PRE → Nanosim

Advanced Reliable Systems (ARES) Lab.