Lecture 7
Overview of Design Flow

Xuan ‘Silvia’ Zhang
Washington University in St. Louis

http://classes.engineering.wustl.edu/ese566/
Application Specific Integrated Circuit (ASIC) Type

- **Full-custom**
  - transistors are hand-drawn
  - best performance (although almost extinct)
  - still using to optimized standard cell

- **Gate Array (for small volumes)**
  - use sea of gates (mask-programmable gate arrays)
  - FPGA (reconfigurable)

- **Standard Cell**
  - only use standard cell from the library
  - dominate design style for ASIC (what we are going to use)
Outline

Standard-Cell-Based Design

SoC-Platform-Based Design

Methodology Comparison
Standard-Cell-Based Design

- Product Requirement
  - Behavioral/Functional Specification
  - Behavioral (RTL) Synthesis
  - Structural Specification
  - Physical Synthesis
  - Physical Specification

Check

Front End

Back End

To CMOS Fab.

- $V_{DD}$ Bus Width
- Metal-to-Metal Spacing
- P Transistor Width
- N to P diffusion spacing or enough space for poly contact and metal routing for most complex cell (usually a flip-flop)

- Contact to Gate Spacing
- N Transistor Width
- Metal-to-Metal Spacing
- GND Bus Width
Example Standard Cell
Standard-Cell-Based Flow CAD Algorithms

Synthesis Algorithms

$x = a'bc + a'bc'$
$y = b'c' + ab' + ac$

RTL to Logic Synthesis

$x = a'b$
$y = b'c' + ac$

Technology Independent Synthesis

Physical Design Automation

Placement

Global Routing

Detailed Routing
Front-End Flow
Back-End Flow
Older Standard-Cell ASICs

Limited metal layers require dedicated routing channels and feedthrough cells.
Modern Standard-Cell ASICs

Increasing number of metal layers allow cells to be hidden under interconnect
# Standard-Cell Libraries

<table>
<thead>
<tr>
<th>Gate Type</th>
<th>Variations</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverter/Buffer/Tristate Buffers</td>
<td></td>
<td>Wide range of power options, 1x, 2x, 4x, 8x, 16x, 32x, 64x minimum size inverter</td>
</tr>
<tr>
<td>NAND/AND</td>
<td>2–8 inputs</td>
<td>High, normal, low power</td>
</tr>
<tr>
<td>NOR/OR</td>
<td>2–8 inputs</td>
<td>High, normal, low power</td>
</tr>
<tr>
<td>XOR/XNOR</td>
<td></td>
<td>High, normal, low power</td>
</tr>
<tr>
<td>AOI/OAI</td>
<td>21, 22</td>
<td>High, normal, low power</td>
</tr>
<tr>
<td>Multiplexers</td>
<td>Inverting/noninverting</td>
<td>High, normal, low power</td>
</tr>
<tr>
<td>Adder/Half Adder</td>
<td></td>
<td>High, normal, low power</td>
</tr>
<tr>
<td>Latches</td>
<td></td>
<td>High, normal, low power</td>
</tr>
<tr>
<td>Flip-Flops</td>
<td>D, with and without synch/asynch set and reset, scan</td>
<td>High, normal, low power</td>
</tr>
<tr>
<td>I/O Pads</td>
<td>Input, output, tristate, bidirectional, boundary scan, slew rate limited, crystal oscillator</td>
<td>Various drive levels (1–16 mA) and logic levels</td>
</tr>
</tbody>
</table>
Standard-Cell Libraries

Cell Library

- Cell logic models, containing cell functionality, pins, area, timing, power
  - Used for circuit constructions, timing, power and area optimizations

- Cell physical model (Abstract view): cell size, pin locations, pin directions
  - Used by physical synthesis

- Cell physical view: Original cell layout
  - Used to build layout of finished design

- Cell descriptions: Behavioral (Verilog) and transistor level (SPICE)
  - Used to simulate completed design at the required level

Cell description coding
- Specification
- Description simulation
- Logic Synthesis
- Formal Verification (RTL Vs Gates)
- Pre-layout STA

Timing OK?
- yes
  - Floorplanning, Placement & Routing
  - Formal Verification (Layout vs. synthesized Netlist)
  - Post-layout Simulation
  - Finished design

Timing OK?
- no
Standard-Cell Library Characterization

Extracted netlists → Characterization task → Characterization tool

- Timing Characterization (transition, delay, setup, hold, recovery, removal times)
- Power Characterization (dynamic power, leakage)
- Other Parameter Characterization (noise, variability, etc.)

→ Characterized library (*.LIB, *.DB, ...)

Cell functionality data
Standard-Cell Electrical Characterization

- Characterization computes cell parameters
  - e.g. delay, output current
  - depend on input variables, i.e. output load, input skew, etc.

- Characterization is performed under conditions
  - combination of process, voltage, temperature (PVT)
ST Microelectronics: 3-Input NAND Gate

- $C$ = load capacitance
- $T$ = input rise/fall time

<table>
<thead>
<tr>
<th>Path</th>
<th>1.2V - 125°C</th>
<th>1.6V - 40°C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$In1\rightarrow t_{pLH}$</td>
<td>$0.073 + 7.98C + 0.317T$</td>
<td>$0.020 + 2.73C + 0.253T$</td>
</tr>
<tr>
<td>$In1\rightarrow t_{pHL}$</td>
<td>$0.069 + 8.43C + 0.364T$</td>
<td>$0.018 + 2.14C + 0.292T$</td>
</tr>
<tr>
<td>$In2\rightarrow t_{pLH}$</td>
<td>$0.101 + 7.97C + 0.318T$</td>
<td>$0.026 + 2.38C + 0.255T$</td>
</tr>
<tr>
<td>$In2\rightarrow t_{pHL}$</td>
<td>$0.097 + 8.42C + 0.325T$</td>
<td>$0.023 + 2.14C + 0.269T$</td>
</tr>
<tr>
<td>$In3\rightarrow t_{pLH}$</td>
<td>$0.120 + 8.00C + 0.318T$</td>
<td>$0.031 + 2.37C + 0.258T$</td>
</tr>
<tr>
<td>$In3\rightarrow t_{pHL}$</td>
<td>$0.110 + 8.41C + 0.280T$</td>
<td>$0.027 + 2.15C + 0.223T$</td>
</tr>
</tbody>
</table>
Standard-Cell Electrical Characterization (.lib)

/* Characterization for a 3-input NAND gate */
cell  ( NAND3x0 ) {

    /* Overall characterization */
cell_footprint : "nand3x0";
area            : 7.3728;
cell_leakage_power : 9.151417e+04;

    /* Characterization for input pin IN1 */
pin  ( IN1 ) {
        direction : "input";

        /* Fixed input capacitance */
capacitance : 2.190745;

        /* Transient capacitance values */
fall_capacitance : 2.212771;
rise_capacitance : 2.168719;
cell ( NAND3X0 ) {
pin ( IN1 ) {

    /* Short-circuit and internal switching power when IN2 and IN3 are zero */
    internal_power () {
        when : "!IN2&!IN3"
    }

    /* 1D lookup tables to calculate power as function of input slew */
    rise_power ( "power_inputs_1" ) {
        index_1( " 0.0160000,  0.0320000,  0.0640000" );
        values( "-1.2575404, -1.2594251, -1.2887053" );
    }

    fall_power ( "power_inputs_1" ) {
        index_1( " 0.0160000,  0.0320000,  0.0640000" );
        values( " 1.9840914,  1.9791286,  2.0696119" );
    }
}
}
cell ( NAND3X0 ) {
  pin ( QN ) {
    direction : "output";

    /* Boolean logic eq for QN as function of inputs */
    function : "((IN3*IN2*IN1)')";

  timing () {
    related_pin : "IN1";
    cell_rise ( "del_1_7_7" ) {

      index_1( "0.016, 0.032, 0.064" ); /* Input slew */
      index_2( "0.1, 3.75, 7.5" ); /* Load cap */

      /* Lookup table to calculate delay as non-linear function of input signal slew and load cap */
      values( "0.0178632, 0.0275957, 0.0374970", \
              "0.0215562, 0.0316225, 0.0414275", \
              "0.0261721, 0.0387623, 0.0496870" )
    }
  }
}
Standard-Cell Physical Characterization

Layout View

Abstract Physical View

Metal Pins

NAND_1
### Table 10.11. NAND Truth Table (n=2,3,4)

<table>
<thead>
<tr>
<th>IN1</th>
<th>IN2</th>
<th>...</th>
<th>INn</th>
<th>QN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>...</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>...</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>...</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

### Logic Symbol of NAND

![Logic Symbol of NAND](image)

**Figure 10.6. Logic Symbol of NAND**

### Operating Conditions:
- VDD = 1.2 V DC
- Temp = 25 Deg.C
- Operating Frequency: 300 MHz
- Capacitive Standard Load: Csl = 13 fF

<table>
<thead>
<tr>
<th>Cell Name</th>
<th>Prop Delay (Avg)</th>
<th>Leakage (VDD=1.2 V DC, Temp=25 Deg.C)</th>
<th>Dynamic</th>
<th>Area (um²)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ps</td>
<td>nW</td>
<td>nW/MHz</td>
<td>(um²)</td>
</tr>
<tr>
<td>NAND2X0</td>
<td>0.5 x Csl</td>
<td>140</td>
<td>38</td>
<td>3583</td>
</tr>
<tr>
<td>NAND2X1</td>
<td>1 x Csl</td>
<td>132</td>
<td>78</td>
<td>5208</td>
</tr>
<tr>
<td>NAND2X2</td>
<td>2 x Csl</td>
<td>126</td>
<td>157</td>
<td>9191</td>
</tr>
<tr>
<td>NAND2X4</td>
<td>4 x Csl</td>
<td>125</td>
<td>314</td>
<td>17902</td>
</tr>
<tr>
<td>NAND3X0</td>
<td>0.5 x Csl</td>
<td>128</td>
<td>91</td>
<td>5331</td>
</tr>
<tr>
<td>NAND3X1</td>
<td>1 x Csl</td>
<td>192</td>
<td>102</td>
<td>12200</td>
</tr>
<tr>
<td>NAND3X2</td>
<td>2 x Csl</td>
<td>212</td>
<td>155</td>
<td>19526</td>
</tr>
<tr>
<td>NAND3X4</td>
<td>4 x Csl</td>
<td>241</td>
<td>260</td>
<td>44937</td>
</tr>
<tr>
<td>NAND4X0</td>
<td>0.5 x Csl</td>
<td>147</td>
<td>106</td>
<td>5357</td>
</tr>
<tr>
<td>NAND4X1</td>
<td>1 x Csl</td>
<td>178</td>
<td>161</td>
<td>15214</td>
</tr>
</tbody>
</table>
Outline

Standard-Cell-Based Design

SoC-Platform-Based Design

Methodology Comparison
System-on-Chip (SoC) Platform-Based Design

- Integration
  - standard cell block
  - custom analog
  - processor
  - memory

- Standardized Bus
  - AMBA, Sonics, ...

- IP Business Model
  - hard or soft IP from 3rd party provider
  - e.g. ARM
SoC Hardware/Software Co-Design

Parallel Development of Hardware IP Blocks

- Generic Hardware IP Blocks

- Architecture Platform

- Select Architecture Platform

- Integrate Application-specific IP Blocks into Architecture Platform

- Functional simulation

- Hardware Design Flow

- Hardware/Software Co-simulation

Electronic System Level Design

Specify System-on-Chip

- Partition Hardware/Software

- Select Software IP Modules

- Integrate Software IP Modules

- Low-level Software Simulation

- Software Design Flow

Parallel Development of Software IP Modules

- Generic Software IP Models

Application-specific Software Modules

- Operating System
SoC Hardware/Software Co-Design

Hardware and Low-level Software Emulation on FPGA-based Emulation Platform

- Physical Design
- Prototype IC Fabrication
- Volume IC Fabrication

Application Software Development

Hardware/Software Verification on Application Prototype or Development Board

Software Test

Ship ICs and Software to Clients
SoC Platform-Based Design

- Platform allows restriction on design space
  - limit implementation choice
  - provide well-defined abstraction of the underlying technology for app developers

- New platform
  - at architecture and micro-architecture boundary

- Representation of communication
  - key to the platform design approach
Outline

Standard-Cell-Based Design

SoC-Platform-Based Design

Methodology Comparison
Operation Binding Time

- Earlier the operation is bound, the less area, delay, and energy required for implementation.
- Later the operation is bound, the more flexible the device.

<table>
<thead>
<tr>
<th>&quot;Hardware&quot;</th>
<th>&quot;Software&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Custom</td>
<td>Reprog. Logic</td>
</tr>
<tr>
<td>Standard Cell</td>
<td>μproc</td>
</tr>
<tr>
<td>SoC Platform</td>
<td>DSP</td>
</tr>
<tr>
<td>Gate Array</td>
<td>Load Program</td>
</tr>
<tr>
<td>Prog. Logic</td>
<td>Config Program</td>
</tr>
<tr>
<td>&quot;First Mask&quot;</td>
<td>Load</td>
</tr>
<tr>
<td>&quot;First Mask&quot;</td>
<td>&quot;First Mask&quot;</td>
</tr>
<tr>
<td>Metal Masks</td>
<td>&quot;Fuse Program&quot;</td>
</tr>
</tbody>
</table>

Later Binding Time
## Comparison of Specific Design Methodologies

<table>
<thead>
<tr>
<th>Design Method</th>
<th>NRE</th>
<th>Unit Cost</th>
<th>Power Disp</th>
<th>Impl Compl</th>
<th>Time to Market</th>
<th>Perf</th>
<th>Flex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Custom</td>
<td>VHigh</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>VHigh</td>
<td>Low</td>
</tr>
<tr>
<td>Standard Cell</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>SoC Platform</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>Med</td>
<td>High</td>
<td>High</td>
<td>Med</td>
</tr>
<tr>
<td>Prog Logic</td>
<td>Low</td>
<td>High</td>
<td>Med</td>
<td>Low</td>
<td>Low</td>
<td>Med</td>
<td>Med</td>
</tr>
<tr>
<td>Reprog Logic</td>
<td>Low</td>
<td>High</td>
<td>Med</td>
<td>Med</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>µProc/DSP</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>VHigh</td>
</tr>
</tbody>
</table>
ASIC vs. FPGA

- **Traditional Argument**
  - ASIC: high NRE ($2M for 0.35µm), low marginal cost, best efficiency
  - FPGA: low NRE, high marginal cost, lower efficiency
  - crossover point: ~10,000

- **Current Trends**
  - ASIC: increasing NRE ($40M for 90nm) due to design, verification, and mask costs, etc.
  - FPGA: better able to track Moore’s law integrating fixed function blocks
  - crossover point: ~100,000
Scale: ASIC with Pre-Placement & SRAMs
T0: Full Custom with Standard Cells

- Control Logic
- MIPS
- DP
- VMP
- VP1
- Vector Registers
- VPO
- I$
ASIC and Full Custom with Standard Cell

Standard cell: predefined gates, automatically placed and routed. In .5u → 10K fets/mm²

Full custom: custom “cells” meant to be stacked in columns to create N-bit wide datapath. Signals between columns routed across cells. In .5u → 25K/mm²

RAM Generator: one cell iterated many times perhaps surrounded by driver/sensing logic. Basic structure stays the same, only dimensions change. In .5u → 45K/mm² for multiport regfile
Xilinx Vertex-II Pro: FPGA with Hard Processor

- FPGA Fabric
- Embedded memories
- Embedded PowerPc
- Hardwired multipliers
- High-speed I/O
Altera HardCopy: FPGA to Gate-Array-Like Tapeout

- RTL code
  - Timing constraints
    - Synthesis
      - Stratix placement and routing
        - Stratix database
          - Formal verification
          - HardCopy placement and global routing
            - HardCopy database
              - Design handoff
              - HardCopy Design Center
                - Synopsys Astro
                - Cadence Conformal
                - Synopsys PrimeTime SI
                - Synopsys TetraMax
              - Tape-out
                - HardCopy fab, assembly, and test
                - System available
                  - System bring-up
                  - Software/hardware co-design

Multi-Chip System: BrainSoC and PEU in RoboBee
Design Principle in Automated Methodologies

• Modularity
  - to enable mixing different custom and automated methods

• Hierarchy
  - to efficiently handle automatically transforming large design

• Encapsulation
  - significant higher emphasis on encapsulation in all domains (behavioral, structural, physical) at all level of abstraction (architecture, RTL, and gate-level)

• Regularity
  - to create automated tools to generate structures like datapaths and memories

• Extensibility
  - automated methodologies enable more highly parameterized and flexible implementation improving extensibility
Questions?

Comments?

Discussion?
Acknowledgement

Cornell University, ECE 5745