SoC Design
ICE of silicon

![Graph showing computational efficiency vs. feature size for various processors like i386SX, i486DX, Pentium, 68040, microsparc, Ultra sparc, Turbosparc, 601, 604, 604e, 7400, 21164a, 21364, with labels for Roza's query by humming and 3D TV applications.](http://bwrc.eecs.berkeley.edu/cic)
Hardware Efficiency

- ASIC (high efficiency, low flexibility)
- ASIP (medium efficiency, medium flexibility)
- DSP (low efficiency, high flexibility)
- GP proc FPGA (high flexibility, low efficiency)
ASIC Style

- Highly efficient for fixed algorithms
- Ok only for large market volumes (100Ms for 32 nm)
- No changes after processing at all (no field upgrades, tuning to specific context, bug fixes, new standards)
- Irregular code leads to highly irregular floorplan with large wiring impact (Edyn) and large leakage (Estat)
- Difficult to efficiently include time multiplexing for irregular code
ASIC + microcontroller style

- Highly efficient for fixed algorithms that use μ-controller very seldom
- Ok only for large market volumes (100Ms for 32 nm)
- Limited changes after processing
- Changes only very locally in non-critical code (ok for some field upgrades, tuning to specific context, bug fixes, new standards)
- Irregular code leads to highly irregular floorplan with large wiring impact (Edyn) and large leakage (Estat)
- Difficult to efficiently include time multiplexing for irregular code
General-purpose microprocessors

- Highly flexible: easy field upgrades, tuning to specific context, bug fixes, new standards
- Easy to use and compiler friendly
- Large market due to combination of smaller markets
- Large A+E overhead: data cache hierarchy, multi-port register file, instr. hierarchy, very flexible data-path units (wide multiplier, ALU with many instr.)
GP CPUs + custom accelerators

- Highly flexible: easy field upgrades, tuning to specific context, bug fixes, new standards. But degraded when accelerators have to be used too much
- Easy to use and compiler friendly
- Large market due to combination of smaller markets, but not when accelerators used more
- Large A+E overhead: data cache hierarchy, multi-port register file, instr hierarchy, very flexible data-path units (wide multiplier, ALU with many instr). Partly mitigated when accelerators are used sufficiently
- Large overhead in communication between microproc and accelerators except when large code segments (not flexible!)
SoC Design

- Synthesis
- DFT Insertion
- Floorplanning
- Power Planning
- Clock tree insertion
- Place and Route
- RC extraction
- Timing check
Design Tools

• **System Architecture**
  – C/C++
  – SystemC
  – Matlab

• **Synthesis**
  – RC Compiler
  – Design Compiler

• **RTL**
  – Verilog-XL
  – NC-Verilog
  – NC-VHDL
  – Debussy

• **Physical Design**
  – SoC Encounter
  – Magma (Synopsys)
  – Mentor
Simplified Flow

Front End

- Test (ATPG)
- Logic Synthesis
- Static Timing Analysis

Back End

- RTL
- .lib LEF
- Timing Constraints
- Logic Simulation
- Floor planning
- Clock Tree Synthesis
- Place &Route
- RC Extraction
- DRC/LVS
- Static Timing Analysis

- Netlist
- GDSII
- SPEF, SDF
TSMC’s Design Flow

Design Challenges:
- Statistical Timing
- Enhanced DFM
- Enhanced Power Management
- Power Closure Flow
- SI Closure Flow
- Hierarchical Flow
- Timing Closure Flow

Technology:
- Release 7.0
- Release 8.0
- Release 9.0
- Release 10.0

New Feature:
System-in-Pkg. (SiP)
Flow with Multi-Vendor Tools

<table>
<thead>
<tr>
<th>Function</th>
<th>EDA Vendor</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Synthesis*</td>
<td>Synopsys</td>
<td>Design Compiler</td>
</tr>
<tr>
<td>Physical Synthesis*</td>
<td>Synopsys</td>
<td>Physical Compiler</td>
</tr>
<tr>
<td>Logic Simulation</td>
<td>Cadence, Mentor Graphics, Synopsys</td>
<td>NC Verilog, ModelSim, VCS</td>
</tr>
<tr>
<td>Timing Analysis</td>
<td>Cadence, Synopsys</td>
<td>Primetime</td>
</tr>
<tr>
<td>Place &amp; Route*</td>
<td>Cadence, Synopsys</td>
<td>SOC Encounter, Astro</td>
</tr>
<tr>
<td>3D-Extraction*</td>
<td>Cadence</td>
<td>Fire and Ice QX</td>
</tr>
<tr>
<td>Power*</td>
<td>Synopsys</td>
<td>Power Compiler</td>
</tr>
<tr>
<td>Test*</td>
<td>Synopsys</td>
<td>DFT Compiler, Tetramax</td>
</tr>
<tr>
<td>Co-Verification</td>
<td>Mentor Graphics</td>
<td>Seamless</td>
</tr>
<tr>
<td>Verification*</td>
<td>Synopsys, Cadence</td>
<td>Vera, Conformal</td>
</tr>
<tr>
<td>Emulation</td>
<td>EVE</td>
<td>Zebu</td>
</tr>
<tr>
<td>FPGA</td>
<td>Synplicity</td>
<td>Synplify Pro</td>
</tr>
</tbody>
</table>
Design Abstraction Levels
Conceptual level
high level
RT level
gate level
transistor level

complexity

impact of a design decision
At higher levels the impact of a design decision is larger.

Vendors concentrate on lower levels (more general solutions).
Logic Synthesis

Synthesis is the process by which an abstract description (known as RTL) of the circuit behaviour (generally in VHDL) is mapped to a set of primitive standard cells in a library for a particular process technology.

- **Translation** of RTL description into an intermediate format
- **Optimization** of logic
- **Mapping** of the optimized netlist to the gates of target library.

**Synthesis tool requires**
- RTL code
- Target ASIC cell library
- User Constraints
  - Timing and Area
  - Environmental
  - Power, Load etc.

- **Output of the synthesis is a gate level netlist in the target technology**
RTL Coding

- RTL stands for Register Transfer Level
- RTL description of a design describes the design in terms of registers and logic that resides between them
- This captures the timing constraints of the design efficiently
- Verilog and VHDL are two most popular hardware description languages that are commonly used to write RTL description
- RTL description captures the change in data at each clock cycle
- All the registers are updated at the same time in a clock cycle
- RTL captures the data flow
- Logic synthesis tools translate an RTL model more efficiently compared to behavioral model

Sample RTL code

```
if IR(3) = 0' then'
   PC := PC + 1;
else
   DBUF := MEM(PC);
   MEM(SP) := PC + 1;
   SP := SP - 1;
   PC := DBUF;
end if;
```
Logic Synthesis

RTL

Process (CLK, RST)
  if (RST = ‘1’) then
    Q <= ‘0’;
  else
    if rising_edge (CLK) then
      Q <= A and B and !(C and D);

Gate level netlist

ASIC cell library

User constraints

Logic Synthesis Tool
Logic Synthesis: Technology Mapping

\[ Z = (\text{not } S \text{ and } A) \text{ or } (S \text{ and } B) \]

Generic Gates

Standard Cells

ANDOR-001
DfT Insertion

- Testable Flip-Flops
- Scan chain generation
- Chain propagation from core to output pin
Backend Design

- Technology Information and Physical Libraries
  - Corelib.lef
  - IOlib.lef
  - Rams.vclef
- Timing libraries
  - Corelib_slow.lib
  - Corelib_fast.lib
  - Corelib_typ.lib
  - IOlib_slow.lib
  - RAM timing libraries
- Timing constraints (user defined)
- Design Netlist
  - Add IO pads, power pads
  - Verilog design netlist
- IO pad location file
Floorplanning

- Floor planning is the task of deciding how the chip area is to be utilized by the leaf modules taking care of wiring considerations.
- Two methods of floorplanning:
  - Top Down: Here the chip is partitioned up during the development of the RTL level modelling. Area is assigned on the basis of estimated block areas and shapes, and blocks are placed relative to each other depending on connectivity.
  - Bottom up: Here the design is first synthesised and then the resultant gates are clustered together into blocks on the basis of connectivity.
- Most designs use a combination of both of the above techniques, but the emphasis is increasingly on the first.
Floorplanning

- Calculating core size, width and height
- When calculating core size of standard cells, the core utilization must be decided first. Usually the core utilization is higher than 85%
- The core size is calculated as follows

\[
\text{Core Size of Standard Cell} = \frac{\text{standard cell area}}{\text{core utilization}}
\]

- The recommended core shape is a square, i.e. Core Aspect Ratio = 1.
- Width = Height = \((\text{Core Size of Standard Cells})^{0.5}\)

Example
- Standard cell area = 2,000,000um\(^2\)
- Core utilization demanded = 85%
- No macros
- Core Size of Standard Cells = \(2,000,000 / 0.85 = 2,352,941\)um\(^2\)
- Width = Height = \((2,352,941)^{0.5} = 1534\)um
Floorplanning

- **Core Margins**
  - Space for power and ground routing
- **Core limited / Pad limited designs**
  - When pad width > (core width + core margin), die size is decided by pads. And it is called pad limited design
  - When pad width < (core width + core margin), die size is decided by core. And it is called core limited design
Power Planning

- Metal migration (also known as electro-migration)
  - Under high currents, electron collisions with metal grains cause the metal to move. The metal wire may be open circuit or short circuit.
    - Prevention: sizing power supply lines to ensure that the chip does not fail
    - Experience: make current density of power ring < 1mA/m

- IR drop
  - IR drop is the problem of voltage drop of the power and ground due to high current flowing through the power-ground resistive network
  - When there are excessive voltage drops in the power network or voltage rises in the ground network, the device will run at slower speed
  - IR drop can cause the chip to fail due to
    - Performance (circuit running slower than specification)
    - Functionality problem (setup or hold violations)
    - Unreliable operation (less noise margin)
    - Power consumption (leakage power)
    - Latch up
    - Prevention: adding stripes to avoid IR drop on cell’s power line
Power Planning: IR Drop

- Number of counts inversely proportional to DSP clock frequency
- $F_C = 10, 20$ and $25$ MHz
- Ringo frequency $\approx 115$ MHz @ $V_{DD} = 1.8$V
- DSP induced PSN is clearly detected

Average PSN = 6 counts $\times$ 2.4 mV/count = 14.4 mV

Source: J. Rius, UPC
Voltage Drop Verification

VoltageStorm (Cadence)

SoC Encounter

Results displayed in SoC Encounter Interface

Block-level Analysis
- Encounter Power Analysis
  - Block Power Consumption
  - Voltage Storm

Top-level Analysis
- Encounter Power Analysis
  - Instance Power Consumption
  - Voltage Storm

Power Grid View Library

CELL PGV

BLOCK PGV
Power Grid Design
Power Ring Width

Experience
- Gate count = 70 k
- 4000 Flip-Flops
- 80% FF with dynamic gated clock
- Current needed = 0.2mA/MHz
  - Note: the value should multiply with 1.8~2 for no gated design

Example:
- Gate count = 200 k
- No gated clock
- Clock frequency = 20 MHz
- Current needed = \((200/70) \times 0.2 \times 20 \times 2 = 22.86\ mA\)
- Current density < 1mA/m
- The Width of P/G Ring > 22.86 um
- In order to avoid the slot rule of wide metal, the largest width is 20 um (process dependent)
- Use two sets of P/G ring for this case
Power Stripe Calculation

Experience
• Add one strap set per 100 um

Example
• Core width = height = 1600
• Stripe set added = 15

Core/IO power pad selection
• Core power pad
  – One set core power pad (PVDDC along with PVSSC) can provide 40~50mA current
• IO power pad
  – One set IO power pad (PVDDDR along with PVSSR) can provide the power for
    • 3~4 output pads, or
    • 6~8 input pads
Placement

- Placement decides the positions of components within allocated blocks.
- One cannot route until the components have been placed.
- The quality of placement is decided solely on the basis of the quality of routing it allows.
- Placement is performed using simple estimates of final routing.
- Timing driven P&R is the state of the art.
- Gates, flip-flops/latches are the common placement objects.
  - Smaller elements like logic gates are placed in single row.
  - Larger blocks are placed in multiple-rows.

![Image of placement settings window and a diagram of a chip with labeled Std cells and Low utilization core.]
Placement

Source: Magma
Clock Tree Synthesis

• Clock signal is used as a timing reference in a synchronous digital system for the movement of data within that system.

• The Clock Tree or clock distribution network distributes the clock signal(s) from a common point to all the elements that need it.

• Properties of clock signals
  – They are loaded with the greatest fanout,
  – travel over the greatest distances
  – operate at the highest speeds

• The goal of clock tree synthesis includes
  – Creating clock tree spec file
  – Building a buffer distribution network

• In automatic CTS mode, Encounter will do the following things
  – Build the clock buffer tree according to the clock tree specification file
  – Balance the clock phase delay with appropriately sized, inserted clock buffers
Clock Tree Synthesis

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoCTSRootPin</td>
<td>clk</td>
</tr>
<tr>
<td>MaxDelay</td>
<td>10ns</td>
</tr>
<tr>
<td>MinDelay</td>
<td>0ns</td>
</tr>
<tr>
<td>SinkMaxTran</td>
<td>400ps</td>
</tr>
<tr>
<td>BufMaxTran</td>
<td>400ps</td>
</tr>
<tr>
<td>MaxSkew</td>
<td>400ps</td>
</tr>
<tr>
<td>NoGating</td>
<td>NO</td>
</tr>
<tr>
<td>Buffer</td>
<td>CLKBUX1, CLKBUX2, CLKBUX3, CLKBUXN</td>
</tr>
<tr>
<td>PostOpt</td>
<td>YES</td>
</tr>
</tbody>
</table>
Routing

- Routing is the process of building the physical connections between blocks as defined by the logical connections.
- Routing takes place in more than one layer, the exact number available depending on the process and design conventions.
- Layers are connected together using vias
- Global Routing
  - Assigns wires to channels defined during the floor planning phase
- Detailed Routing
  - Assigns nets to individual tracks in the channel

Routing and Final Optimisation

- Signal Routing
- Antennas
- Decap, Fillers
- Crosstalk Fixing
- Post Route Fix
- Editing
Routing: Signal Integrity Cross-talk

- Parallel repeater insertion *does not* reduce the cross-talk peak noise
  - For a 10mm communication bus, the delay noise is lowered by about 77%
- Staggered repeaters reduce delay noise by about 88%

Source: M. Meijer and A. Katoch, Philips
Routing: SI Prevention

- Reduce crosstalk glitch and delay variation
- Reduce coupling capacitance
- Wiring Spacing
- Layer Switching
- Parallel Wires Reducing
- Net Re-ordering

Verification Signoff
- Timing & Crosstalk Analysis
- Power Distribution Analysis
- Parasitic Extraction
Static Timing Analysis

• This involves three main steps:
  – Design is broken down into sets of timing paths
  – The delay of each path is calculated
  – All path delays are checked to see if timing constraints have been met

Path delay calculations

\[
\text{path\_delay} = (1.0 + 0.54 + 0.32 + 0.66 + 0.23 + 0.43 + 0.25) = 3.43 \text{ ns}
\]
Physical Verification

- DRC
  - Design Rule Checking
- LVS
  - Layout vs. Schematic verifications
Chip Finishing

• Seal-ring & Artefact Generation
  – helps to make the circuit moisture resistant and prevents the generation of cracks in the die during sawing the wafer
  – Sometimes this step is simply called ‘Design Chip Finishing’
  – critical dimensions structures, mask ids, fuse markers, etc

• Tiling - dummy fill/pattern fill
  – Fabs stringent min and rules on layer densities on active, poly and metal must be met by all designs
  – Currently back-end operation

• Each step is followed by Physical Verification step
Package Fitting

- Selection of appropriate package
- Route pads to pins
  - Wire length is important
  - Rule checking
- GDS2 minimum required information is the nitride or pad opening layer or the pad boundary layer
Packaging