Design Methodologies and Tools

• Design styles
  – Full-custom design
  – Standard-cell design
  – Programmable logic
    • Gate arrays and field-programmable gate arrays (FPGAs)
    • Sea of gates
      – System-on-a-chip (embedded cores)
• Design tools

Full-Custom Design

• Every gate is carefully designed and optimized
  – Hierarchical design may be used, place-and-route tools typically used
• Advantages:
  – High density, ideal for high volumes
  – Performance optimization
• Disadvantages:
  – High design time, cumbersome
• Applications: Datapaths in microprocessors
• High-paying jobs!
Full-Custom Design (Contd.)

- Simple CAD tools suffice
  - Design entry
  - Schematic editor
  - Layout editor
  - DRC, LVS, Spice
- No need for sophisticated synthesis and design partitioning tools

Standard Cell Design

- Use pre-optimized SSI or MSI library cells
  - NAND gates, multiplexers, adder slices, decoders, comparators, RAM, ROM
- Reduces design time
- Lower density and lower performance
- Standardized at the logic or function level

Pitch-matched cells
Routing channel
Standard Cells (Contd)

- CAD tools needed for partitioning design
- Technology mapping
- Design entry, DRC, LVS, simulations tools needed
- Place and route tools
- Good for moderate volumes, as in ASICs, typically used for non-critical portions of a CPU

Programmable Logic

- Programmable logic blocks
  - PLDs, PALs
  - Fusible links (fuses) blown when current is exceeded
- Programmable interconnects
  - Mask-programmed gate-arrays (MPGAs)
  - Field-programmable gate-arrays (FPGAs)
  - Sea of gates
Programmable Logic

• PLDs
  - Wide fan-in, 2-level SOP, optional flip-flops on output
  - Best known: 22V10 with 22 inputs, 10 outputs, from AMD
  - Programmed by users
  - Fusible links

• MPGAs
  - Also called gate arrays
  - More dense than PLDs
  - Predesigned transistors with customized wiring
  - Wiring done during manufacture (not user)

FPGAs: complex designs, user programmable

MPGAs vs FPGAs
FPGAs

- Advantages
  - Low design cost (custom masks not needed)
  - Rapid turnaround
  - Low risk
  - Effective design verification
  - Low testing costs-test program same for all designs

- Disadvantages
  - Programming circuitry: area penalty, ten times larger for same gate capacity as MPGA
  - Speed: 2-3 times slower than MPGA
  - Design methodology: “too easy” to use, encourages “try-it-and-see-what-happens” methodology

Analog FPGAs (FPAAs) now commercially available
Programmable Interconnects

- Program the routing
- PLICE: Programmable Low-Impedance Circuit Element, “antifuse”
  - Normally high resistance (> 100MΩ)
  - Antifuse can be permanently changed to a low-resistance (200-500Ω) structure, i.e. form links
- Low gate utilization
- E.g. 32-bit adder using Actel FPGAs:
  - 160 logic modules, 65 ns for addition
  - 3.5 32-bit adders on an FPGA chip

Comments on Gate Utilization for FPGAs

- 32-bit adder using Actel FPGAs:
  - 160 logic modules, 65 ns for addition
  - 3.5 32-bit adders on an FPGA chip
  - Design time: one afternoon, $5-$10 design cost
  - Lot of wasted logic
- 32-bit full-custom adder (1 micron)
  - 1300 adders on one chip, 33 ns addition time
  - Design time: 6 months, $200,000
Actel Logic Cell

- All 2-input functions are realized (select inputs carefully)
- All 3-input functions are realized (if?)
- Which 4-input functions?
- How to implement a latch?

Xilinx FPGAs

- Configurable logic blocks (contain SRAMs), CLBs
- Programmable interconnect
- 500 CLBs on a chip
- 100K bits of RAM per chip
- 32-bit adder example:
  - 62 CLBs, 8 adders per chip
  - Speed: 20-50 MHz
Sea of Gates Design

- Goal: reduce design cost of IC, rapid prototyping, fast turnaround time
  - Not useful for high-volume production (more area per IC)
- Core of chip (base array) contains continuous array of n and p transistors
  - Polysilicon laid a-priori (transistors formed in advance)
- Personalization done by using design-specific metallization and contacts
- Highly automated process: sophisticated CAD tools used

Sea of Gates

Routing possible over unused rows
**Sea of Gates**

- **Design Decisions**
  - How many n and p rows per strip?
  - Size and ratio of transistors
  - Number, direction of routing tracks

**Embedded Cores**

- Complex “off-the-shelf”, optimized, pre-designed circuits: processors, ASICs, memories, controllers
- Used for system-on-a-chip (SOC) designs
- Hot intellectual property (IP)
- IC consists of several embedded cores along with custom-designed blocks
The Billion Transistor Era

Reliability
Wire Delay
Power
Variability

Roadblocks Ahead for IC Designers

Images courtesy of Intel Corporation
Vertical (3D) Integration isn’t new!
We can also do heterogeneous stacks!

Vertical (3D) Integration isn’t new!

when real estate became pricey,
architects built 3D high-rise

When interconnect / memory bandwidth became the bottleneck for performance,

IC designers built 3D architectures!
“Space on the ground is running out…
The only way is up…
The question is not whether this reach for the sky will happen –
it’s happening now –
but what it portends.”

Hugh Pearman, May 2004
(The Sunday Times, UK)
A TSV-based 3D IC

- Most of the current ICs have
  - Only one layer of active devices (N/P MOS)
  - multi-layer metal interconnects
- A TSV-based 3D IC has
  - Multiple layers of active devices (N/P MOS)
  - With many Through-Silicon-Vias (TSV) between layers
Benefits of TSV based 3D ICs

- Wire Length Reduction!

- Performance/Power Improvement
- High Throughput (Bandwidth)
- Mix-technology integration
- Cost

Design Tools

- Synthesis tools
  - Behavioral synthesis
  - Logic synthesis
- Design capture
- Design verification
**Synthesis Tools**

- Specify system behavior without implementation details
- Allows fast simulation/verification
- Technology-independent
- Esp. successful for signal processing architectures
- Silicon compilers

**Behavioral Synthesis**

- Decide upon and assign resources (scheduling and binding)
- Insert pipeline registers to meet timing constraints
- Create microcode and control logic
Logic Synthesis

- Technology-independent: use Boolean and/or algebraic techniques
  - Network optimization, two-level and multi-level minimization
- Technology-mapping phase: cell binding

Logic Optimization Example

- Input
- Extract
- Netlist
- Netlist generation

Logic equation

- Optimization

Literal count = 17

\[ f_1 = aef + bef + ceg \]
\[ f_2 = aeg + bg + def \]

Literal count = 17

Extract common subexpression ef

Literal count = 15

\[ f_1 = (a+b)t1 + ceg \]
\[ f_2 = aeg + bg + dt1 \]
Design Capture Tools

- How to describe the behavior/structure of a system?
- HDLs: VHDL, Verilog
- High-level languages (HLLs): C, Pascal, Lisp
- HDLs differ from HLLs by catering for hardware concepts: bit vectors, signals, timing
  - Like HLLs, HDLs provide structure, parameterization, conditionals, looping, hierarchy
- Structure specification: HDL, netlist editor
- Behavior specification: HDL, finite-state machines, Petri nets

Schematic Entry

- Textual editor
- Easy to modify, make changes
- Suitable for more complex designs
- Interactive graphics editor
- Example: Mentor DA
- Diagrams are easily understood (“A picture is worth a thousand words”)
Layout Editors

- Captured via code: (cell/layout generators), silicon compilation
- Interactive graphics editors (Mentor IC)
- Means for turning off detail or zooming
- Design rule checking programs (DRC)
- Layout extraction programs
  - Determine netlist from layout

Design Verification

```
Functional specification ➔ Equivalence check

RTL/logic ➔ Equivalence check

Layout ➔ Silicon
```

Verify the tool?
Design Verification

Simulation
  Gate-level
  Transistor, schematic-level (Qsim)
  Circuit level (HSpice): high accuracy but high
  simulation times, O(N^m), N transistors,
  0<m<1
  Formal verification: Mathematical models