Do we need more chips (ASICs)?

ASIC = Application-Specific Integrated Circuit
Wide Variety of Products Rely on ASICs

Robots
Supercomputers
Automobiles
Sensor Nets
Games
Set-top boxes
Cameras
Media Players
Laptops
Servers
Routers
Smart phones
Wide Variety of Products Rely on ASICs


What’s required?
ICs with dramatically higher performance, optimized for applications and at a size and power to deliver mobility and cost to address mass consumer markets
Let’s take a look at current CMOS technology...

Chip = Transistors + Wires

Cross-section through IBM 90nm process, 10 metal layers

Transistors fabricated first on original surface of wafer

"Vias" connect one layer to another

Wiring added in layers on top of wafer

Thicker wires on higher layers used for power and ground, and long range signals

"Glass" on top seals chip

Thinner wires on lower layers used for dense local wiring

Bulk of wafer

[ISSCC 2004]
**FET = Field-Effect Transistor**
A four terminal device (gate, source, drain, bulk)

- **Inversion:** A vertical field creates a channel between the source and drain.
- **Conduction:** If a channel exists, a horizontal field causes a drift current from the drain to the source.

---

**Simplified FET Model**

- PFET connects S and D when G="low"=0V
- NFET connects D and S when G="high"=V\_DD

- Supply Voltage = V\_DD
- Ground = GND = 0V

Binary logic values represented by voltages:
"High" = Supply Voltage, "Low" = Ground Voltage
NAND Gate

- When both A and B are high, output is low
- When either A or B is low, output is high

NAND Gate Layout

- Series NMOS Transistors
- Parallel PMOS Transistors
- Metal 1-Diffusion Contact
- P-Diffusion (in N-well)
- Output on Metal-1
- Poly wire connects PMOS & NMOS gates
### Exponential growth: Moore’s Law

<table>
<thead>
<tr>
<th>Year</th>
<th>Processor</th>
<th>Area (mm²)</th>
<th>Frequency (MHz)</th>
<th>Transistors</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>1974</td>
<td>Intel 8080A</td>
<td>33</td>
<td>3</td>
<td>6K</td>
<td>6u</td>
</tr>
<tr>
<td>1978</td>
<td>Intel 8086</td>
<td>33</td>
<td>33</td>
<td>29K</td>
<td>3u</td>
</tr>
<tr>
<td>1982</td>
<td>Intel 80286</td>
<td>47</td>
<td>12.5</td>
<td>134K</td>
<td>1.5u</td>
</tr>
<tr>
<td>1985</td>
<td>Intel 386DX</td>
<td>43</td>
<td>33</td>
<td>275K</td>
<td>.8u</td>
</tr>
<tr>
<td>1989</td>
<td>Intel Pentium</td>
<td>43</td>
<td>50</td>
<td>1.2M</td>
<td>.8u</td>
</tr>
<tr>
<td>1993/1</td>
<td>Intel Pentium II</td>
<td>295/147/90</td>
<td>66/33/33</td>
<td>7.5M</td>
<td>.35u/.25u</td>
</tr>
<tr>
<td>1997</td>
<td>Intel Pentium II</td>
<td>203/104/72</td>
<td>200/330/73</td>
<td>410M</td>
<td>.25u/.35u</td>
</tr>
</tbody>
</table>


---

### Intel Penryn (2007)

- Dual core
- Quad-issue out-of-order superscalar processors
- 6MB shared L2 cache
- 45nm technology
  - Metal gate transistors
  - High-K gate dielectric
- 410 Million transistors
- 3+ GHz clock frequency

Could fit over 500 486 processors on same size die.
.. But Design Effort Growing

Nvidia Graphics Processing Units

<table>
<thead>
<tr>
<th>Year</th>
<th>Design Effort (M)</th>
<th>Relative staffing on back-end</th>
<th>Relative staffing on front-end</th>
</tr>
</thead>
<tbody>
<tr>
<td>1993</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>1995</td>
<td>80</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>1996</td>
<td>120</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>1997</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>1998</td>
<td>90</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>1999</td>
<td>80</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>2000</td>
<td>70</td>
<td>70</td>
<td>70</td>
</tr>
<tr>
<td>2001</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>2002</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

- Front-end is designing the logic (RTL)
- Back-end is fitting all the gates and wires on the chip; meeting timing specifications; wiring up power, ground, and clock

Design Cost Impacts Chip Cost

- 90nm ASIC cost breakdown, $30M total (Altera study):
  - 59% chip design (architecture, logic & I/O design, product & test engineering)
  - 30% software and applications development
  - 11% prototyping (masks, wafers, boards)

- If we sell 100,000 units, Non-Recurring Engineering (NRE) costs add $30M/100K = $300 per chip!

- Example above is for design using automated tools
  - Similar to what we’ll be using in 6.375

- Hand-crafted IBM-Sony-Toshiba Cell microprocessor achieves 4GHz in 90nm, but development cost was >$400M
Topics to address in 6.375

- How can we design complex billion transistor ASICs with reasonable effort?
- How good are our designs?
  - Performance, area, power

Designer’s Dilemma

ASIC Complexity
- 2000: 1M+ logic gates
- 2005: 10M+ logic gates
- 2010: 100M+ logic gates

Designer must take shortcuts
- Conservative design
- No time for exploration
- Educated guess & code
- Gates are free mentality

<table>
<thead>
<tr>
<th>Pipeline Type</th>
<th>Speed (ns)</th>
<th>Area (gates)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td>8.88</td>
<td>6.60</td>
</tr>
<tr>
<td>Linear</td>
<td>4.79</td>
<td>99.9</td>
</tr>
<tr>
<td>Circular</td>
<td>3.67</td>
<td>99.9</td>
</tr>
<tr>
<td>Static (2)</td>
<td>8.88</td>
<td>63.5</td>
</tr>
</tbody>
</table>

[ICCAD’04]

What happens when a designer must implement a 1M gate block?

Sub-optimal implementations!

Alternatives?
6.375 Course Philosophy

- Effective abstractions to reduce design effort
  - High-level design language rather than logic gates
  - Control specified with Guarded Atomic Actions rather than with finite state machines
  - Guarded module interfaces automatically ensure correctness of composition of existing modules
- Design discipline to avoid bad design points
  - Decoupled units rather than tightly coupled state machines
- Design space exploration to find good designs
  - Architecture choice has largest impact on solution quality

A unified view of languages, disciplines and tools that supports rapid design space exploration to find best area, power, and performance point with reduced design effort

6.375 Objectives

By end of term, you should be able to:

- Decompose system requirements into a hierarchy of sub-units that are easy to specify, implement, and verify, and which can be reused
- Develop efficient verification and test plans
- Select appropriate microarchitectures for a unit and perform microarchitectural exploration to meet price, performance, and power goals
- Use industry-standard tool flows
- Complete a working million gate chip design!
- Make millions $$ at a new chip startup

(Don’t forget your alma mater!)
6.375 Prerequisites

- You must be familiar with undergraduate (6.004) logic design:
  - Combinational and sequential logic design
  - Dynamic Discipline (clocking, setup and hold)
  - Finite State Machine design
  - Binary arithmetic and other encodings
  - Simple pipelining
  - ROMs/RAMs/register files
- Additional circuit knowledge (6.002, 6.374) useful but not vital
- Architecture knowledge (6.823) helpful for projects

6.375 Structure

- First half of term (before Spring Break)
  - Lecture or tutorial MWF, 2:30pm to 4:00pm in 32-124
  - Three labs (on Athena, lab machines in 38-301)
  - Form project teams (2-3 students); prepare project proposal (watch website for project ideas)
  - Closed-book 90 minute quiz (Friday before Spring Break)
- Second half of term (after Spring Break)
  - Weekly project milestones, with 1-2 page report
  - Weekly project meeting with the instructors and TAs
  - Final project presentations in last week of classes
  - Final project report (~15-20 pages) due Thursday May 17 (no extensions)
6.375 Grade Breakdown

- Three labs: 30%
- Quiz: 20%
- Five project milestones: 25%
- Final project report: 25%
  (including presentation)

6.375 Collaboration Policy

- We strongly encourage students to collaborate on understanding the course material, BUT:
  - Each student must turn in individual solutions to labs
  - Students must not discuss quiz contents with students who have not yet taken the quiz
  - If you’re inadvertently exposed to quiz contents before the exam, by whatever means, you must immediately inform the instructors or TA
ASIC Design Styles

Hardware Design Abstraction Levels

- Application
- Algorithm
- Unit-Transaction Level (UTL) Model
- Guarded Atomic Actions (Bluespec)
- Register-Transfer Level (Verilog RTL)
  - Gates
  - Circuits
  - Devices
  - Physics
ASIC Design Styles

- Full-Custom (every transistor hand-drawn)
  - Best possible performance: as used by Intel μPs
- Semi-Custom (Some custom + some cell-based design)
  - Reduced design effort: AMD μPs plus recent Intel μPs
- Cell-Based ASICs (Only use cells in standard library)
  - High-volume, moderate performance: Graphics chips, network chips, cellphone chips
  - This is what we’ll use in 6.375
- Mask-Programmed Gate Arrays/Structured ASICs
  - Medium-volume, moderate performance applications
- Field-Programmable Gate Arrays
  - Low-volume, low-moderate performance applications, and prototyping

Comparing styles:
- how many design-specific mask layers per ASIC?
- how much freedom to develop own circuits?
- what design methods and tools are needed?

Custom and Semi-Custom

- Usually, in-house design team develops own libraries of cells for commonly used components:
  - memories
  - register files
  - datapath cells
  - random logic cells
  - repeaters
  - clock buffers
  - I/O pads
- In extreme cases, every transistor instance can be individually sized ($$$$)
  - approach used in Alpha microprocessor development
- The trend is towards greater use of semi-custom design style
  - use a few great circuit designers to create cells
  - redirect most effort at microarchitecture and cell placement to keep wires short
Custom Designer works with Low-Level Design Rules

- An abstraction of the fabrication process that specifies various geometric constraints on how different masks can be drawn.
- Design rules can be absolute measurements (e.g., in nm) or scaled to an abstract unit, the lambda. The value of lambda depends on the manufacturing process finally used.

Standard Cell ASICs
aka Cell-Based ICs (CBICs)

- Fixed library of cells + memory generators, often provided by fabrication foundry or third-party library providers.
- Cells can be synthesized from HDL, or entered in schematics.
- Cells placed and routed automatically.
- Requires complete set of custom masks for each design.
- Currently the most popular hard-wired ASIC type (6.375 will use this).
Standard Cell Library Components

- Cells have standard height but vary in width
- Designed to connect power, ground, and wells by abutment

6.375 Standard Cell Design Flow

Bluespec SystemVerilog source → Bluespec Compiler → Blueview → C

Legend:
files
Bluespec tools
3rd party tools

http://csg.csail.mit.edu/6.375/
Standard Cell Design Examples

- Channel routing for 1.0mm 2-metal stdcells
- Over cell routing for 0.18mm 6-metal stdcells

Mask-Programmed Gate Arrays

- Can cut mask costs by prefabricating arrays of fixed size transistors on wafers
- Only customize metal layer for each design

Two kinds:
- Channeled Gate Arrays
  - Leave space between rows of transistors for routing
- Sea-of-Gates
  - Route over the top of unused transistors
Gate Array Personalization

- Isolating transistors by shared GND contact
- Isolating transistors with "off" gate

Gate Array Pros and Cons

- Cheaper and quicker since less masks to make
  - Can stockpile wafers with diffusion and poly finished
- Memory inefficient when made from gate array
  - Embedded gate arrays add multiple fixed memory blocks to improve density (=>Structured ASICs)
  - Cell-based array designed to provide efficient memory cell (6 transistors in basic cell)
- Logic slow and big due to fixed transistors and wiring overhead
  - Advanced cell-based arrays hardwire logic functions (NANDs/NORs/LUTs) which are personalized with metal
Field-Programmable Gate Arrays (FPGAs)

- Arrays mass-produced and programmed by customer after fabrication
  - Can be programmed by blowing fuses, loading SRAM bits, or loading FLASH memory
- Each cell in array contains a programmable logic function
- Array has programmable interconnect between logic functions
- Overhead of programmability makes arrays expensive and slow but startup costs are low, so much cheaper than ASIC for small volumes

Xilinx Configurable Logic Block
FPGA Pros and Cons

Advantages
- Dramatically reduce the cost of errors
- Remove the reticle costs from each design

Disadvantages (as compared to an ASIC) [Kuon & Rose, FPGA2006]
- Switching power around ~12X worse
- Performance up 3-4X worse
- Area 20-40X greater

Still requires tremendous design effort at RTL level