Introduction to Microelectronics

Dr. Hubert Kaeslin
Microelectronics Design Center
ETH Zürich

VLSI I: Architectures of VLSI Circuits

last update: April 8, 2009
### Worldwide semiconductor market by vendors (2007)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Vendor</th>
<th>Revenue [GUSD]</th>
<th>Share [%]</th>
<th>Share [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Intel</td>
<td>33.80</td>
<td>12.3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Samsung Electronics</td>
<td>20.46</td>
<td>7.5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Toshiba</td>
<td>11.82</td>
<td>4.3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Texas Instruments</td>
<td>11.77</td>
<td>4.3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Infineon + Qimonda</td>
<td>10.20</td>
<td>3.7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>ST-Microelectronics</td>
<td>9.97</td>
<td>3.6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Hynix</td>
<td>9.10</td>
<td>3.3</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Renesas</td>
<td>8.00</td>
<td>2.9</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>AMD</td>
<td>5.88</td>
<td>2.1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>NXP</td>
<td>5.87</td>
<td>2.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>others</td>
<td>147.05</td>
<td>53.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Total</strong></td>
<td><strong>237.91</strong></td>
<td><strong>100</strong></td>
<td><strong>0.49</strong></td>
</tr>
</tbody>
</table>

For comparison

| World GDP (2006) | 48 462 | 100 |

Economic leverage of semiconductors

Microelectronics has a much larger impact on world economy, however, because it is acting as a technology driver for

- Computer and software industry
- Telecommunications and media industry
- Commerce, logistics and transportation
- Natural science and medicine
- Power generation and distribution
- Finance and administration
Economic leverage of semiconductors II

Figure: Impact of microelectronics on “downstream” industries and services.
Microelectronics drives the information age

- Microelectronics has an enormous economic leverage as any progress there spurs innovations in “downstream” industries and services.
- While computing, telecommunication, and entertainment products existed before the advent of microelectronics, today’s information society would not have been possible without.
Microelectronics drives the information age

- Microelectronics has an enormous economic leverage as any progress there spurs innovations in “downstream” industries and services.

- While computing, telecommunication, and entertainment products existed before the advent of microelectronics, today’s information society would not have been possible without.

⇒ Microelectronics is the enabler of information technology.
Impact of semiconductors on consumer goods I

Figure: Four products that take advantage of microelectronics.
Impact of semiconductors on consumer goods II

Figure: Similar products that include no large-scale integrated circuits.
Impact of semiconductors on consumer goods III

Figure: A product that has brought system integration to even higher levels.
The Guiness book of records point of view

“How large is that circuit?”

- Geometric chip size
- Transistor count
- Gate-equivalents
  
  $1 \text{ GE} \rightarrow 1 \text{ two-input NAND} \rightarrow 4 \text{ MOSFETs in static CMOS logic}$
The Guinness book of records point of view

“How large is that circuit?”

- Geometric chip size
- Transistor count
- Gate-equivalents
  1 GE $\mapsto$ 1 two-input $\text{NAND} \mapsto$ 4 MOSFETs in static CMOS logic

<table>
<thead>
<tr>
<th>circuit complexity</th>
<th>GE of logic + bits of memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>small-scale integration (SSI)</td>
<td>1 ... 10</td>
</tr>
<tr>
<td>medium-scale integration (MSI)</td>
<td>10 ... 100</td>
</tr>
<tr>
<td>large-scale integration (LSI)</td>
<td>100 ... 10 000</td>
</tr>
<tr>
<td>very-large-scale integration (VLSI)</td>
<td>10 000 ... 1 000 000</td>
</tr>
<tr>
<td>ultra-large-scale integration (ULSI)</td>
<td>1 000 000 ...</td>
</tr>
</tbody>
</table>

Hint: state storage capacities separately from logic complexity, e.g.
75 000 GE of logic + 32 Kibit SRAM + 512 bit flash $\approx$ 108 000 GE
What you ought to know about logic families

A logic family is a collection of digital subfunctions that
- assemble to arbitrary logic, arithmetic and storage functions
- are compatible among themselves electrically
- share a common fabrication technology

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOS</td>
<td>Metal Oxide Semiconductor.</td>
</tr>
<tr>
<td>FET</td>
<td>Field Effect Transistor (n- or p-channel)</td>
</tr>
<tr>
<td>BJT</td>
<td>Bipolar Junction Transistor (nnp or pnp)</td>
</tr>
<tr>
<td>CMOS static CMOS</td>
<td>Complementary MOS (circuit or technology) data stored in bistable subcircuits and retained</td>
</tr>
<tr>
<td>dynamic CMOS</td>
<td>data stored as electrical charges to be refreshed</td>
</tr>
<tr>
<td>TTL</td>
<td>Transistor Transistor Logic (BJTs &amp; passive devices)</td>
</tr>
<tr>
<td>ECL</td>
<td>Emitter-Coupled Logic (non-saturating logic)</td>
</tr>
<tr>
<td>BiCMOS</td>
<td>CMOS &amp; bipolar devices on a single chip</td>
</tr>
</tbody>
</table>
2-input NAND gate in various technologies

Figure: Static CMOS (c), NMOS (b), early TTL (e), and ECL circuit (g).
The marketing point of view

“How do functionality and target markets relate to each other?”

**General-purpose IC.** Examples are either very simple or very generic.
Simple circuit: gates, flip-flops, counters, etc.
Generic functionality: RAMs, ROMs, microcomputers, FPL, etc.
The marketing point of view

“How do functionality and target markets relate to each other?”

General-purpose IC. Examples are either very simple or very generic.
Simple circuit: gates, flip-flops, counters, etc.
Generic functionality: RAMs, ROMs, microcomputers, FPL, etc.

Application-specific integrated circuit (ASIC).

► Application-specific standard product (ASSP): designed for a specific task and sold to various customers.
Examples: graphics accelerators, cellular radio chip sets, smart card chips, etc.

► User-specific integrated circuit (USIC): designed and produced for a single company.
Examples: audio processor for hearing aids, etc.
A first IC classification scheme

Figure: ICs classified as a function of functionality and hardware complexity.
A first glimpse at VLSI manufacturing

**Figure:** Full-custom (a) and semi-custom (b) masks sets compared.
Semi-custom fabrication I

Figure: Prefabricated gate array site.
Semi-custom fabrication II

Figure: Custom contact and metal masks.
Figure: Site customized into a 2-input NAND gate.
Evolution of semi-custom floorplans

Figure: Channeled gate-array (a) versus channelless semi-custom circuits (b).
Field-programmable logic

▶ No dedicated layout structures, no dedicated photomasks. Customization is via purely electrical means.

▶ “Programmable” is a misnomer as there is no instruction sequence to execute. “Configurable” is more accurate as pre-manufactured subcircuits are made to form the target circuit.

▶ All configuration technologies today have their roots in semiconductor memory technology.

▶ Benefits compared to mask-programmed ASICs:
  ▶ Easy and extremely fast to modify (highly agile).
  ▶ Solutions for testability, I/O subcircuits, clock and power distribution, embedded memories, etc. all come at no extra effort shut in the FPL component.

⇒ FPL can be thought as “soft hardware”.

Dr. Hubert Kaeslin Microelectronics Design Center ETH Zürich
The fabrication point of view (summary)

“To what extent is a circuit manufactured to user specs?”

**Full-custom IC:** all fabrication layers, full set of photomasks.

**Semi-custom IC** (gate array, sea-of-gates, structured ASIC):
- a few metal layers only, subset of photomasks.

**Field-programmable logic (SPLD, CPLD, FPGA):**
- customization occurs electrically, no masks involved.

**Standard part:** catalog part with no customization whatsoever
- aka commercial off-the-shelf (COTS) component.
The design engineer’s point of view

“Which levels of detail are being addressed during a part’s design?”

Hand layout: Desired geometric shapes manually drawn to scale.

Cell-based design by means of schematic entry:
Manual schematic entry $\leadsto$ automatic place & route.

Automatic circuit synthesis:
Manual HDL or SW code writing $\leadsto$ automatic netlist generation.

- Logic synthesis
- Register transfer level (RTL) synthesis
- Architecture or high-level synthesis

Design with virtual components:
Purchase of HDL code $\leadsto$ automatic netlist generation.
Views of a library cell (or of any other subcircuit)

entity nor3 is
    generic (tpd : time := 1.0 ns);
    port (INA, INB, INC : in StdLogic;
          OUP : out StdLogic);
end nor3;
architecture procedural of nor3 is
begin
    OUP <= not (INA or INB or INC) after tpd;
end procedural;

stimuli responses
INA INB INC OUP
0 0 0 1
0 0 1 0
0 1 0 0
1 0 0 0
0 0 0 1
1 1 1 0

Figure: Icon (a), simulation model (b), test vector set (c), transistor-level schematic (d), detailed layout (e), and cell abstract (f).
Typical cell mix in a full-custom IC
Automatic circuit synthesis I

Logic synthesis accepts logic equations, truth tables, and state graphs. Generates gate-level netlists for combinational logic and for finite state machines (FSM).
⇒ Absorbed in today’s EDA flows.

Register transfer level (RTL) modelling:

► Circuit viewed as a network of storage elements — registers and possibly also RAMs — that are held together by combinational building blocks.
► Behavioral specifications allowed to include arithmetic functions, string operations, arrays, enumerated types, and other more powerful constructs.

⇒ Introduced in the early 1990s, universally adopted. Parametrized and portable designs favor reuse.
Automatic circuit synthesis II

Figure: RTL diagram (a), RTL synthesis model (b), and gate-level schematic (c) (simplified, note that (a) and (b) refer to different circuits).
Automatic circuit synthesis III

Architecture synthesis starts from a purely behavioral data processing algorithm. Source code includes no explicit indications for how to marshal data processing operations and hardware resources. Works in five major phases:

1. Identify the computational and storage requirements.
2. From a virtual library, select a suitable building block for each kind of processing and storage operation.
3. Establish a cycle-based schedule for carrying out the algorithm.
4. Decide on a hardware organization able to execute the resulting work plan.
5. Keeping track of data moves and operations for each clock cycle, translate into the necessary instructions for RTL synthesis.

⇒ Does not always yield optimal results, active field of research.
Virtual components

VCs (aka intellectual property modules or cores) are HDL synthesis packages made available to others on a commercial basis:

▸ Vendor develops a major function into a synthesis model for sale.
▸ Licensee buys VC, incorporates it into his design, then carries out all the rest, i.e. synthesis, place and route (P&R), and overall verification.
▸ VCs are portable across fabrication technologies (soft modules), standard/macro/megacells are process-specific (hard modules).
▸ Most VCs implement fairly common subfunctions, parametrization is sought to cover more applications.

⇒ VCs have given rise to a new industry since the late 1990s.
Virtual components

VCs (aka intellectual property modules or cores) are HDL synthesis packages made available to others on a commercial basis:

- Vendor develops a major function into a synthesis model for sale.
- Licensee buys VC, incorporates it into his design, then carries out all the rest, i.e. synthesis, place and route (P&R), and overall verification.
- VCs are portable across fabrication technologies (soft modules), standard/macro/megacells are process-specific (hard modules).
- Most VCs implement fairly common subfunctions, parametrization is sought to cover more applications.
- Examples: processor cores, all sorts of filters, audio and/or video en/decoders, cipher functions, error correction en/decoders, USB, FireWire, and other interfaces.

⇒ VCs have given rise to a new industry since the late 1990s.
## A second IC classification scheme

<table>
<thead>
<tr>
<th>Fabrication depth</th>
<th>Electrical configuration</th>
<th>Semi-custom fabrication</th>
<th>Full-custom fabrication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design level</td>
<td>Cell-based as obtained from</td>
<td>Hand layout</td>
<td></td>
</tr>
<tr>
<td></td>
<td>◦ synthesis with VCs in HDL form,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>◦ synthesis from captive HDL code,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>◦ schematic entry, or a mix of these</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product name</td>
<td>Field-programmable logic device (FPGA, CPLD)</td>
<td>Gate-array, sea-of-gates, or structured ASIC</td>
<td>Std. cell IC</td>
</tr>
</tbody>
</table>

IC families as a function of fabrication depth and design abstraction level.
Electronic system-level (ESL) design automation

Pressure towards better design productivity has incited the industry to look at design automation from a wider perspective.

▶ Correct-by-construction methodology by supporting progressive refinement starting with a virtual prototype

▶ Explore the architectural solution space more systematically and more rapidly than with RTL synthesis methods.

▶ Make it possible to start software development before hardware design is completed.

▶ Improve the coverage and efficiency of functional verification by dealing with system-level transactions and by taking advantage of formal verification.
Players in semiconductor markets I

Our final question relates to business.
“How are the industrial activities shared between business partners?”

Traditional business model:

**Integrated device manufacturer (IDM):** a chip vendor who operates his own wafer processing facilities.
Examples: Intel, Toshiba, Samsung, ST-Microelectronics, IBM semiconductors, austriamicrosystems, etc.
Players in semiconductor markets II

More recent business models support more narrow specialization:

**Silicon foundry:** a company that operates a wafer processing line and that offers its manufacturing services to others.
Examples: TSMC, UMC, etc.

**Fabless chip vendor:** develops and markets proprietary semiconductor components but has their manufacturing commissioned to an independent silicon foundry.
Examples: Altera and Xilinx (FPL), Broadcom (networking), Cirrus Logic/Crystal (audio and video chips), Nvidia (graphics chips), Ramtron (non-volatile memories).

**Fab-lite chip vendor:** retains just the limited and specialized manufacturing capabilities to integrate sensors, actuators, RF components, or photonic devices, in a silicon substrate along with electronic circuitry.
Examples: Sensirion, Luxtera.
Players in semiconductor markets III

**Virtual component vendor:** a company that develops synthesis packages and licenses them to others for incorporation into their own ICs. Examples: ARM, Sci-worx, Synopsys (former InSilicon).

**System house:** a company that integrates both hardware and software into their products. Hardware is based on microprocessors, memories, ASSPs and FPGAs. USICs are being designed iff they provide a competitive advantage. Examples: Apple (media players), Cisco (network equipment), Landis+Gyr (energy meters), Valeo (automotive).

Many small and medium-sized electronics companies (typical for Europe) operate as system houses.
What has made these new business models possible?

Three factors came together to make fabless operation possible:

- Generous integration densities at low costs.
- Proliferation of high-performance engineering workstations and EDA software
- Availability of know-how in VLSI design outside IC manufacturing companies (this course).
The Y-chart, a map of digital electronic systems

Dr. Hubert Kaeslin Microelectronics Design Center ETH Zürich

Introduction to Microelectronics
More design views

architecture series expansion of cosine is
begin
process (theta) is
begin
variable sum, term : real;
variable n : natural;
begin
sum := 1.0;
term := 1.0;
n := 0;
while abs term > abs (sum / 1.0E6) loop
n := n+2;
term := (-term)*theta**2 / real(((n-1)*n));
sum <= sum+term;
end loop;
result <= sum;
end process;
end architecture series expansion;

Figure: Floorplan (a), software model (b), encapsulated chip (c), graphical formalisms (d), transfer characteristic (e), and block diagram (f) (simplified).
Electronic system-level design flow

- exploration, fast prototyping, and validation of algorithms and system architectures implies addressing a subset of the issues shown here.

Is there any broadly accepted formalism for specification?

- behavioral modeling and simulation

SystemC may provide an answer

- is there an agreed-on system-level design language (SLDL)?

UML may provide a workable solution

- formalism for specification

exploration, fast prototyping, and validation of algorithms and system architectures implies addressing a subset of the issues shown here.

The Y-chart, a map of digital electronic systems

Major stages in VLSI design

Field-programmable logic

Appendix I: A brief glossary of logic families

Cell libraries

Electronic system-level design flow

collection of algorithms along with data formats

abstract mathematical models subject to successive refinement

from marketing and customers

algorithms and system architecture design

UML may provide a workable solution

behavorial modeling and simulation

is there an agreed-on system-level design language (SLDL)?

SystemC may provide an answer

- interactive resource allocation
- automatic scheduling
- automatic binding
- translation to RTL model

HDL generator for hardware synthesis

HDL model

truly portable and amenable to synthesis with good results?

ASIC or programmable IC

program-controlled processor

code generator for signal- or microprocessor

machine code

to processor

source coding, data compression

generating, key distribution, authentication

synchronization, clock recovery

filter synthesis, filtering, correlation

protocols, user interfaces

control flow, cooperat. finite state machines

model libraries for various subfunctions

resource and/or instruction set planning

effects from finite word sizes, scaling

filter synthesis, filtering, correlation

error correction coding, modulation

bit-true HDL model

- interactive resource allocation
- automatic scheduling
- automatic binding
- translation to RTL model

HDL generator for hardware synthesis

machine code

to processor
System-level design

Decisions taken at this stage determine the final outcome more than anything else:

- Specify the functionality and characteristics of the system to be
- Partition the system’s functionality into subtasks
- Explore alternative hardware and software tradeoffs
- Decide on make or buy for all major building blocks
- Decide on interfaces and protocols for data exchange
- Decide on data formats, operating modes, exception handling, etc.
- Define, model, evaluate and refine the various subtasks

Result: System-level model.
Algorithm design

Streamline computations in view of their implementation in hardware:

▶ Cut down computational burden and memory requirements
▶ Find compromises between computational complexity and accuracy
▶ Contain effects due to finite word-length computation
▶ Decide on number representation schemes
▶ Evaluate alternatives and selecting the one best suited
▶ Quantify the minimum required computational resources

Result: Bit-true software model.
Digital VLSI design flow (front-end)

1. **From system-level development**
   - Specifications

2. **Specification**
   - Behavioral modeling
   - Overall behavioral simulation (esp. to output mapping)

3. **Architecture Design**
   - High-level block diagram (or HDL code)

4. **RTL Design**
   - Code of regs and arith./logic ops (or schematics)

5. **Logic Design and Optimization**
   - Gate-level netlist (1)

6. **Insertion of Test Structures**
   - Gate-level netlist (2)

7. **Formal Equivalence Check**
   - Pre-layout timing verification

8. **Test Vector Generation**
   - Delay calculation

9. **Fault Grading**
   - Electrical rule check (ERC)

10. **Floorplanning, Package Selection, and Pinout**
    - Floorplan

**Front End Design**

**HDL Synthesis**

**Automatic Test Insertion**
Architecture design I

Take important high-level decisions:

▶ Partition a computational task in view of a hardware realization.
▶ Organize the interplay of the various subtasks.
▶ Allocate hardware resources to each subtask (allocation).
▶ Define datapaths and controllers.
▶ Decide between off-chip RAMs, on-chip RAMs and registers.
▶ Decide on communication topologies and protocols (parallel, serial).
▶ Define how much parallelism to provide in hardware.
▶ Decide where to opt for pipelining and to what degree.
▶ Decide on a circuit style and fabrication process.
▶ Get a first estimate of the circuit’s size and cost.

Results: High-level block diagram and preliminary floorplan.
Architecture design II

Work out lower-levels details of an architecture by deciding:

- How to implement arithmetic and logic units?
- Whether to use hardwired logic or microcode for a controller?
- When to use a ROM rather than random logic?
- What operations to perform during which clock cycle (scheduling)?
- What operations to carry out on which processing unit (binding)?
- What clocking discipline to adopt?
- What time interval to use as the basic clock period?
- Where to prefer a bidirectional bus over two unidirectional ones?
- By what test strategy to ensure testability?
- How to initialize the circuit?

Results: Set of more detailed diagrams and verified RTL code.
Digital VLSI design flow (back-end)
Physical design

Steps

- Floorplanning (begins during front-end design)
- Padframe generation and power distribution
- Initial placement of cells
- Reoptimization and rebuffering
- Clock tree insertion
- Detailed routing
- Rebuffering and hold time fixing
- Chip assembly (global routing)
- Substitution of detailed layout for cell abstracts

Result: Polygon layout data for mask preparation (GDS II).
Physical design verification

Prior to fabrication, all layout data need to be checked to protect against fatal mishaps. The set of instruments available includes:

- Check conformity of layout with geometric rules (DRC)
- Search for patterns likely to be detrimental to yield
- Layout extraction [re-]obtains the actual circuit netlist
- Layout-versus-schematic (LVS)
- Post-layout timing verification
- Post-layout simulation

Result: Either proof of geometric integrity or error list.
The leitmotiv of VLSI design

▶ Any design flaw found after tapeout or, even worse, after prototype fabrication wastes important amounts of time and money.
The leitmotiv of VLSI design

- Any design flaw found after tapeout or, even worse, after prototype fabrication wastes important amounts of time and money.
- Redesigns are so devastating that the entire semiconductor industry is committed to “first-time-right” design as a guiding principle.
- VLSI engineers typically spend much more time verifying a circuit than actually designing it.
Cell libraries I

Figure: Library design flow.

- list of leaf cells to be
- leaf cell functional models
- transistor-level design
- transistor-level schematics
- layout design at detail level
- leaf cell layouts
- leaf cell timing models
- behavioral modelling
- circuit simulation
- continuous time
- DRC and/or manufacturability analysis
- circuit extraction
- layout versus schematic (LVS)
- transistor-level netlists with layout parasitics
- cell characterization
- leaf cell models
- construction verification
  - behavioral aspects
  - structural aspects
  - physical aspects
- flow of design data
- corrective action by designer based on feedback information
- design automation shortcuts
- target cell library and/or process data directly contribute to design decisions
Cell libraries II

The views required for each cell in a library include:

- Datasheet with functional, electrical and timing specs.
- Graphical icon or symbol.
- Accurate behavioral models for simulation and timing analysis.
- Set of simulation and test vectors.
- Transistor-level netlist or schematic.
- Detailed layout.
- Simplified layout showing only cell outline and connector locations (known as cell abstract, floorplanning abstract, or phantom cell).
Cell libraries III

- Designing, characterizing, documenting, and maintaining a cell library is a considerable effort.
- To protect their investments, library vendors are not willing to disclose how their cells are constructed internally.
- Vendors thus supply only cell abstracts.
- Detailed layouts are to be substituted for all abstracts by the vendor before mask preparation can begin.
Field-programmable logic (FPL) configuration technologies

Figure: Configuration storage is adapted from semiconductor memories.
- SRAM: Switch steered by static memory cell (a),
- Flash: MOSFET controlled by trapped charge (b),
- PROM: fuse (c) and antifuse (d).
### Field-programmable logic (FPL) configuration technologies (continued)

<table>
<thead>
<tr>
<th>Configuration technology</th>
<th>Non volatile</th>
<th>Live at power up</th>
<th>Reconfigurable</th>
<th>Unlimit. endurance</th>
<th>Area occupation per link</th>
<th>Extra fabr. steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM</td>
<td>no</td>
<td>no</td>
<td>in ckt</td>
<td>yes</td>
<td>large</td>
<td>0</td>
</tr>
<tr>
<td>UV-erasable EPROM</td>
<td>yes</td>
<td>yes</td>
<td>out of circuit</td>
<td>no</td>
<td>small in array</td>
<td>3</td>
</tr>
<tr>
<td>Electr. erasable EEPROM</td>
<td>yes</td>
<td>yes</td>
<td>in ckt</td>
<td>no</td>
<td>≈EPROM</td>
<td>&gt; 5</td>
</tr>
<tr>
<td>Flash memory</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>n.a.</td>
<td>small</td>
<td>3</td>
</tr>
<tr>
<td>Antifuse PROM</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>n.a.</td>
<td>small</td>
<td>3</td>
</tr>
<tr>
<td>Ideal</td>
<td>yes</td>
<td>yes</td>
<td>in ckt</td>
<td>yes</td>
<td>zero</td>
<td>0</td>
</tr>
</tbody>
</table>
Unlimited reprogrammability has its drawback

- Storing the FPL configuration is an SRAM-type memory implies that the configuration gets lost whenever the circuit is powered down.

- The problem is solved in one of three possible ways:
  (a) by reading from a dedicated off-chip ROM (bit-serial or bit-parallel),
  (b) by downloading a bit stream from a host computer, or
  (c) by long-term battery backup.
Complex programmable logic devices (CPLD)

Figure: General architecture of CPLDs (c) along with precursors (a,b).
Field-programmable logic devices (FPGA)

Overall organization patterned after mask-programmed gate-arrays.

Figure: General architecture of FPGAs.
Fine-grained FPGAs

A few logic gates and/or one bistable per configurable logic cell.

Figure: Example: logic tile from Actel ProASIC.
Coarse-grained FPGAs

Combinational functions of four or more variables, two or more bits stored per configurable logic cell.

Figure: Example: logic slice from Xilinx Virtex-4 (4-input LUTs, 2 bistables).
There is a general trend towards coarser granularities

- The optimum trade-off for LUTs has shifted from 4 to 6 inputs over the last couple of process generations.

**Figure:** LUT granularity trade-offs at the 65 nm technology node.
Example 1: Logic slice from Xilinx Virtex-5
Example II: Adaptive logic module from Altera Stratix II
## Commercial products

<table>
<thead>
<tr>
<th>Configuration technology</th>
<th>Overall organization of hardware resources</th>
<th>CPLD</th>
<th>FPGA coarse grained</th>
<th>FPGA fine grained</th>
</tr>
</thead>
<tbody>
<tr>
<td>static memory (SRAM)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UV-erasable (EPROM)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>electrically erasable (flash)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>antifuse (PROM)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Configuration technologies
- **CPLD**
  - Xilinx Spartan, Virtex.
  - Lattice SC, EC, ECP.
  - Altera FLEX, APEX, Stratix, Cyclone.
  - eASIC Nextreme SL

### UV-erasable (EPROM)
- Cypress MAX340 (discontinued)

### Electrically erasable (flash)
- Xilinx XC9500, CoolRunner-II.
- Altera MAX3000, 7000.
- Lattice MACH 1,...,5.
- Cypress Delta39K, Ultra37000.

### Antifuse (PROM)
- QuickLogic Eclipse II, PolarPro.

### FPGA-coarse grained
- Lattice XP MACH XO.

### FPGA-fine grained
- Actel ProASIC PLUS
- Fusion
- Igloo.

### UV-erasable EPROM
- Atmel AT6000, AT40K.

### Electrically erasable flash
- Actel ProASIC PLUS
- Fusion
- Igloo.

### Antifuse PROM
- Actel MX, Axcelerator AX.
Watch out!

Capacity figures of FPL and of semi-custom ICs may be confusing.

**Manufactured gates:** Total number of GEs physically present on a die.

**Usable gates:** Maximum number of GEs that are usable under typical or best case conditions. The exact percentage depends on the application, advertisements tend to exaggerate.

**Actual gates:** GEs that are indeed put to service by a given design, corresponds to the GEs for a cell-based full-custom IC.

\[ \text{GE}_{\text{manufactured}} > \text{GE}_{\text{usable}} > \text{GE}_{\text{actual}} \]
Watch out!

Capacity figures of FPL and of semi-custom ICs may be confusing.

**Manufactured gates**: Total number of GEs physically present on a die.

**Usable gates**: Maximum number of GEs that are usable under typical or best case conditions. The exact percentage depends on the application, advertisements tend to exaggerate.

**Actual gates**: GEs that are indeed put to service by a given design, corresponds to the GEs for a cell-based full-custom IC.

\[
\text{GE}_{\text{manufactured}} > \text{GE}_{\text{usable}} > \text{GE}_{\text{actual}}
\]

⇒ Carry out benchmarks with representative designs as this helps to

- make better cost calculations,
- obtain realistic timing figures,
- avoid misguided choices.
Some FPGAs also include wide datapath units (MAC)

*These signals are dedicated routing paths internal to the DSP48E column. They are not accessible via fabric routing resources.

Figure: Example: DSP48E slice from Xilinx Virtex-5.
Further extensions

Many commercial parts include field-programmable logic plus
- hardwired SRAMs, FIFOs, clock recovery circuits, etc.
- hardwired microprocessor and DSP cores (e.g. PowerPC, ARM),
- hardwired standard interface circuits (PCI, USB, FireWire, Ethernet, WLAN, JTAG, LVDS, etc.)
- hardwired analog-to-digital and digital-to-analog converters,
- configurable analog building blocks (such as filters, for instance),
- field-programmable analog arrays (FPAA) built from OpAmps, capacitors, resistors and switchcap elements,
- combinations of the above.
FPL design flow

- Front-end flow (architecture design, HDL coding, functional verification) is much the same for FPGAs and CPLDs as for ASICs.

- Back-end flow differs to some extent.
  1. Netlist obtained from HDL synthesis is mapped onto configurable blocks available in the target device.
  2. Interconnect gets implemented using the wires, switches and drivers available.
  3. Result is converted into a configuration bit stream for download into the FPL device.
  4. FPL vendors make available proprietary tools for the above procedure.
FPL design flow

▶ Front-end flow (architecture design, HDL coding, functional verification) is much the same for FPGAs and CPLDs as for ASICs.

▶ Back-end flow differs to some extent.

1. Netlist obtained from HDL synthesis is mapped onto configurable blocks available in the target device.
2. Interconnect gets implemented using the wires, switches and drivers available.
3. Result is converted into a configuration bit stream for download into the FPL device.
4. FPL vendors make available proprietary tools for the above procedure.

⇒ Hierarchy of required skills:
Field-programmable logic ⊂ Semi-custom ICs ⊂ Full-custom ICs.
Let us begin with topics that matter independently of fabrication depth.
### Major semiconductor technologies and logic families

The alphabet soup explained.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOS</td>
<td>Metal Oxide Semiconductor.</td>
</tr>
<tr>
<td>FET</td>
<td>Field Effect Transistor (n- or p-channel)</td>
</tr>
<tr>
<td>BJT</td>
<td>Bipolar Junction Transistor (npn or pnp)</td>
</tr>
<tr>
<td>NMOS</td>
<td>n-channel MOS (transistor, circuit or technology)</td>
</tr>
<tr>
<td>PMOS</td>
<td>p-channel MOS (transistor, circuit or technology)</td>
</tr>
<tr>
<td>CMOS</td>
<td>Complementary MOS (circuit or technology)</td>
</tr>
<tr>
<td>static CMOS</td>
<td>data stored in bistable subcircuits and retained</td>
</tr>
<tr>
<td>dynamic CMOS</td>
<td>data stored as electrical charges to be refreshed</td>
</tr>
<tr>
<td>TTL</td>
<td>Transistor Transistor Logic (BJTs &amp; passive devices)</td>
</tr>
<tr>
<td>ECL</td>
<td>Emitter-Coupled Logic (non-saturating logic)</td>
</tr>
<tr>
<td>BiCMOS</td>
<td>CMOS &amp; bipolar devices on a single chip</td>
</tr>
</tbody>
</table>
2-input NAND gate in TTL technology

TTL invented 1961 as an improvement over DTL and RTL.

Figure: Icon (d), original multi-emitter circuit (e), and more recent F generation circuit (f). The auxiliary devices serve clamping and speed-up purposes.
2-input NAND gate in ECL technology

ECL invented 1956 as fast non-saturating current switching logic.

Figure: Circuit (g) with schematic symbols used. Switching is by current steering without transistors entering saturation.
2-input NAND gate in MOS technologies

CMOS invented 1963 as an improvement over NMOS and PMOS with (close-to) zero quiescent power.

Figure: PMOS (a), NMOS (b), and static CMOS (c) circuits.
Why does CMOS technology dominate VLSI today?

As first observed in 1972 by Robert Dennard

- **Geometric down-scaling benefits**
  - layout density,
  - operating speed,
  - energy efficiency, and
  - manufacturing costs per function.

- Simplicity and comparatively low power dissipation have allowed for integration densities not possible on the basis of BJTs.

⇒ After a start as a low-power but slow circuit alternative, CMOS has gradually displaced competing technologies and logic families.