

Introduction to Microelectronics

Prof. Hubert Kaeslin
Microelectronics Design Center
ETH Zürich

VLSI I: Architectures of VLSI Circuits

last update: June 25, 2012

Book chapter 1, pp1...43

Content

You will learn

the key terms and concepts behind digital integrated circuits

- ▶ The economic impact of microelectronics
- ▶ Microelectronics viewed from five different perspectives
 - ▶ Circuit complexity
 - ▶ Marketing
 - ▶ Fabrication
 - ▶ Design
 - ▶ Business
- ▶ Design flow in digital VLSI
- ▶ Field-programmable logic *To be introduced in a separate lesson.*
- ▶ CMOS compared to other logic families

Worldwide semiconductor market by vendors

Rank	Vendor	Revenue [GUSD]	Share [%]	Share [%]
1	Intel	41.4	13.8	with NEC
2	Samsung Electronics	28.3	9.4	
3	Toshiba	12.4	4.1	
4	Texas Instruments	12.4	4.1	
5	Renesas Electronics	10.4	3.5	
6	Hynix Semiconductor	10.4	3.4	
7	ST-Microelectronics	10.3	3.4	
8	Micron Technology	8.9	3.0	
9	Qualcomm	7.2	2.4	
10	Infineon	6.7	2.2	
...	others	152.2	50.7	
	Total	300.3	100	0.48
for comparison	World GDP (2010)	61 963		100

Source: Gartner and Wikipedia, preliminary figures as per Dec. 2010.

Economic leverage of semiconductors I

Observation

Semiconductors account for roughly 0.5% of the world gross domestic product.

Economic leverage of semiconductors I

Observation

Semiconductors account for roughly 0.5% of the world gross domestic product. 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

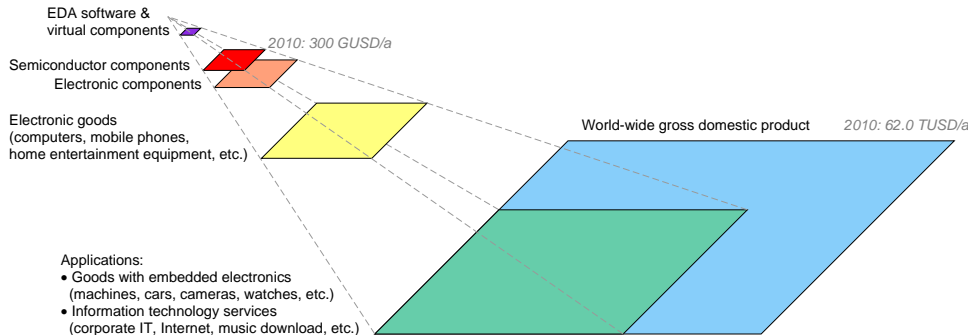


Figure: Impact of microelectronics on “downstream” industries and services.

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.
- ▶ The almost total penetration has been made possible by a long-running drop of cost per function at a rate of 25 to 29% per year.

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.
- ▶ The almost total penetration has been made possible by a long-running drop of cost per function at a rate of 25 to 29% per year.

Observation

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.

Subject

Microelectronics viewed from different perspectives

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 NAND \mapsto 4 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 NAND \mapsto 4 MOSFETs in static CMOS logic

circuit complexity	GEs of logic + bits of memory
small-scale integration (SSI)	1 ... 10
medium-scale integration (MSI)	10 ... 100
large-scale integration (LSI)	100 ... 10 000
very-large-scale integration (VLSI)	10 000 ... 1 000 000
ultra-large-scale integration (ULSI)	1 000 000 ...

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

Acronym	Meaning
MOS	Metal Oxide Semiconductor.
FET	Field Effect Transistor (n- or p-channel)
BJT	Bipolar Junction Transistor (nnp or pnp)
CMOS	Complementary MOS (circuit or technology)
static CMOS	data stored in bistable subcircuits and retained
dynamic CMOS	data stored as electrical charges to be refreshed
TTL	Transistor Transistor Logic (BJTs & passive devices)
ECL	Emitter-Coupled Logic (non-saturating logic)
BiCMOS	CMOS & bipolar devices on a single chip

2-input NAND gate in various techno- logies

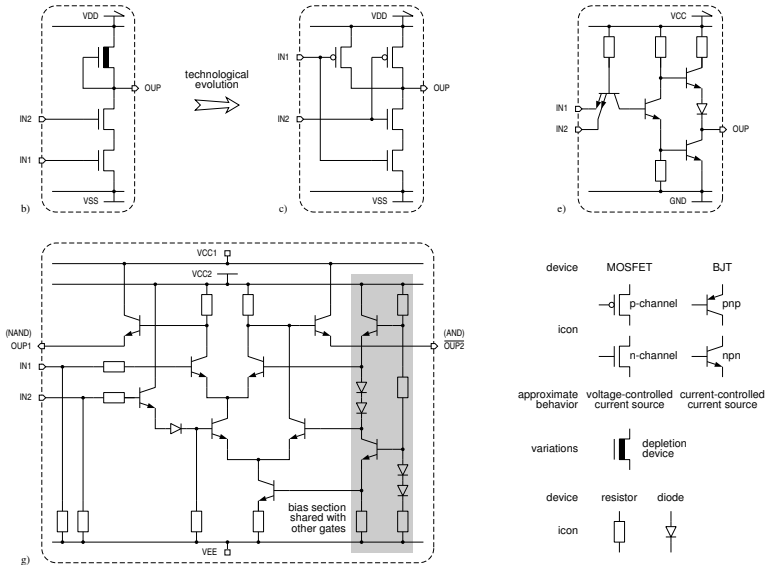


Figure: Static CMOS (c), NMOS (b), early TTL (e), and ECL circuit (g).

The marketing point of view

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

General-purpose IC Either very simple or of generic functionality.

Examples:

- simple: gates, flip-flops, counters, adders, etc.
- generic: RAMs, ROMs, microcomputers, many DSPs, etc.

The marketing point of view

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

General-purpose IC Either very simple or of generic functionality.

Examples:

- simple: gates, flip-flops, counters, adders, etc.
- generic: RAMs, ROMs, microcomputers, many DSPs, 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: Apple A4 SoC introduced with the iPad in 2010, various audio processors for hearing aids. Lower volume: USIC in Rhode & Schwarz oscilloscopes, 90 nm $14 \cdot 10^6$ GE.

A first IC classification scheme

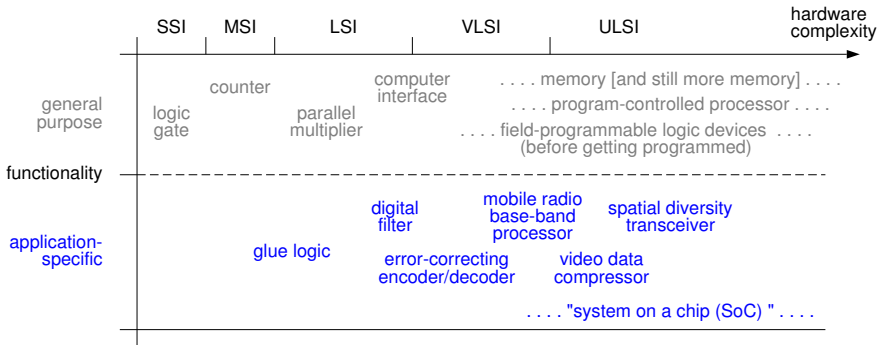


Figure: ICs classified as a function of functionality and hardware complexity.

A first glimpse at VLSI manufacturing

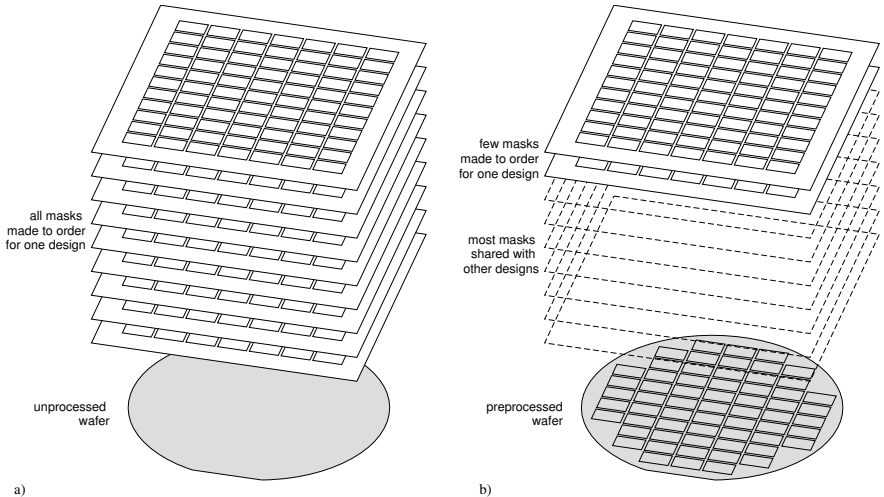


Figure: Full-custom (a) and semi-custom (b) masks sets compared.

Semi-custom fabrication I

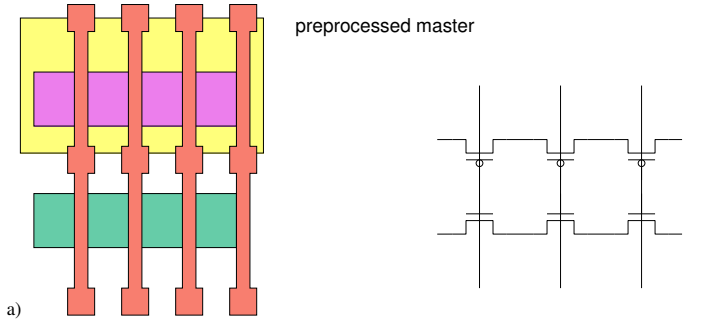


Figure: Gate array site with six prefabricated MOS transistors.

Semi-custom fabrication II

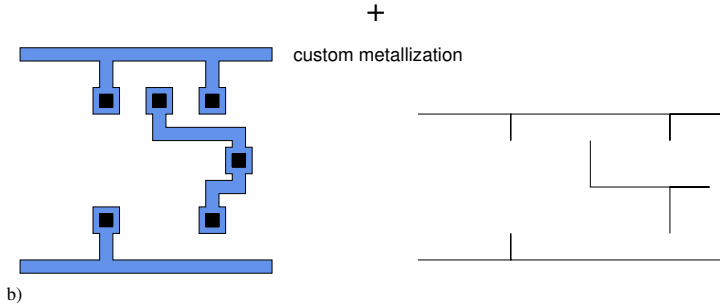


Figure: Custom-designed contact and metal masks.

Semi-custom fabrication III

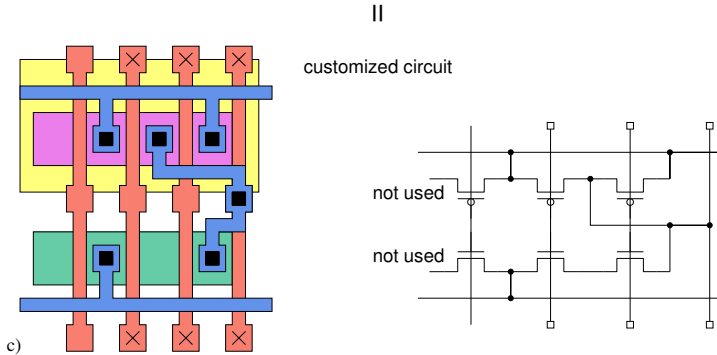


Figure: Site customized into a 2-input NAND gate.

Evolution of semi-custom floorplans

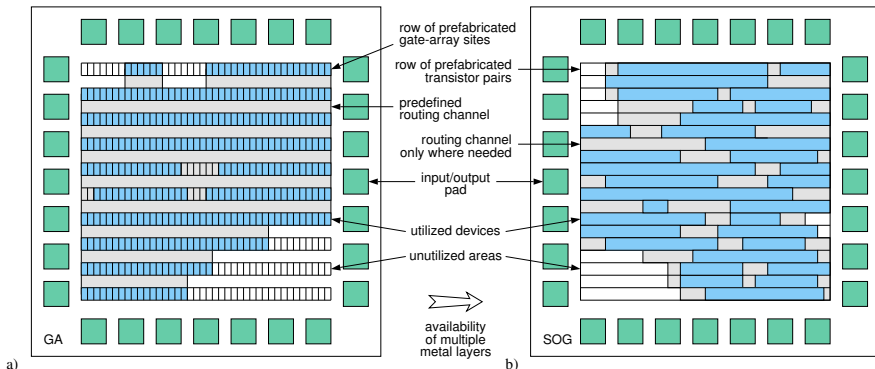


Figure: Channeled gate-array (a) versus channelless semi-custom circuits (b).

Field-programmable logic (FPL)

- ▶ Uses neither custom layout structures nor proprietary photomasks.
- ▶ The term “programmable” is a misnomer as there is no program, no instruction sequence to execute.
- ▶ Instead, pre-manufactured subcircuits get **configured** into the target circuit **via purely electrical means** such as programmable links and look up tables borrowed from memory technology.

Field-programmable logic (FPL)

- ▶ Uses neither custom layout structures nor proprietary photomasks.
- ▶ The term “programmable” is a misnomer as there is no program, no instruction sequence to execute.
- ▶ Instead, pre-manufactured subcircuits get **configured** into the target circuit **via purely electrical means** such as programmable links and look up tables borrowed from memory technology.
- ▶ Compared to mask-programmed ICs:
 - + Easy and extremely **fast to modify** (highly agile).
 - + I/O subcircuits, clock and power distribution, embedded memories, testability, etc. come at no extra effort shut in the component.
 - **Large overhead** in terms of area, delay and energy.

Field-programmable logic (FPL)

- ▶ Uses neither custom layout structures nor proprietary photomasks.
- ▶ The term “programmable” is a misnomer as there is no program, no instruction sequence to execute.
- ▶ Instead, pre-manufactured subcircuits get **configured** into the target circuit **via purely electrical means** such as programmable links and look up tables borrowed from memory technology.
- ▶ Compared to mask-programmed ICs:
 - + Easy and extremely **fast to modify** (highly agile).
 - + I/O subcircuits, clock and power distribution, embedded memories, testability, etc. come at no extra effort shut in the component.
 - **Large overhead** in terms of area, delay and energy.

Observation

FPL devices can be thought of as “soft hardware”.

We recommend to introduce FPL in a separate lesson after the introductory presentation!

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 I

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

- Hand layout** Desired geometric shapes manually drawn to scale.
- + optimum density, performance and device matching.
 - slow, cumbersome and prone to errors.

The design engineer's point of view I

"Which levels of detail are being addressed during a part's design?"

Hand layout Desired geometric shapes manually drawn to scale.

- + optimum density, performance and device matching.
- slow, cumbersome and prone to errors.

Cell-based design by means of schematic entry Manual drawing of cell-level circuits followed by automatic place & route.

Standard cells small universal building blocks, preestablished layout, fully characterized. Examples: logic gates, flip-flops, adder slices, etc.

The design engineer's point of view I

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

Hand layout Desired geometric shapes manually drawn to scale.

- + optimum density, performance and device matching.
- slow, cumbersome and prone to errors.

Cell-based design by means of schematic entry Manual drawing of cell-level circuits followed by automatic place & route.

Standard cells small universal building blocks, preestablished layout, fully characterized. Examples: logic gates, flip-flops, adder slices, etc.

Megacells much larger size and complexity than standard cells. Examples: microprocessor cores and peripherals, A/D and D/A converters.

Macrocells layout put together on a per case basis according to specs from a limited collection of layout tiles. Examples: RAMs and ROMs.

Cell library views

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).

Example: 3-input NOR gate

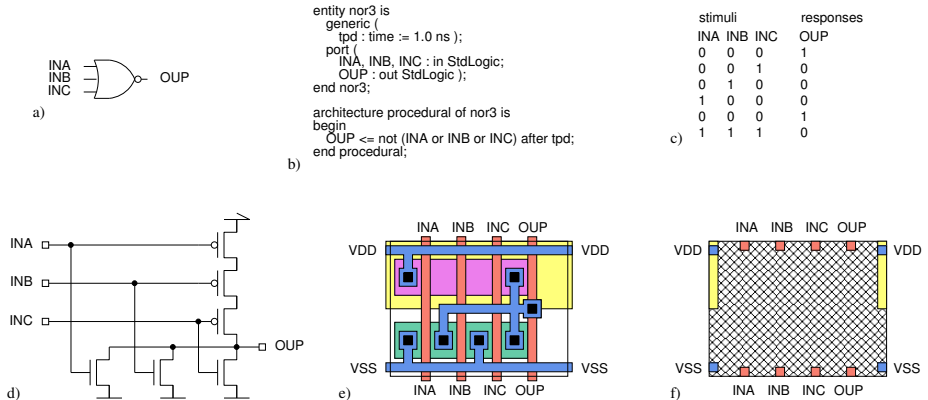


Figure: Icon (a), simulation model (b), test vector set (c), transistor-level schematic (d), detailed layout (e), and cell abstract (f).

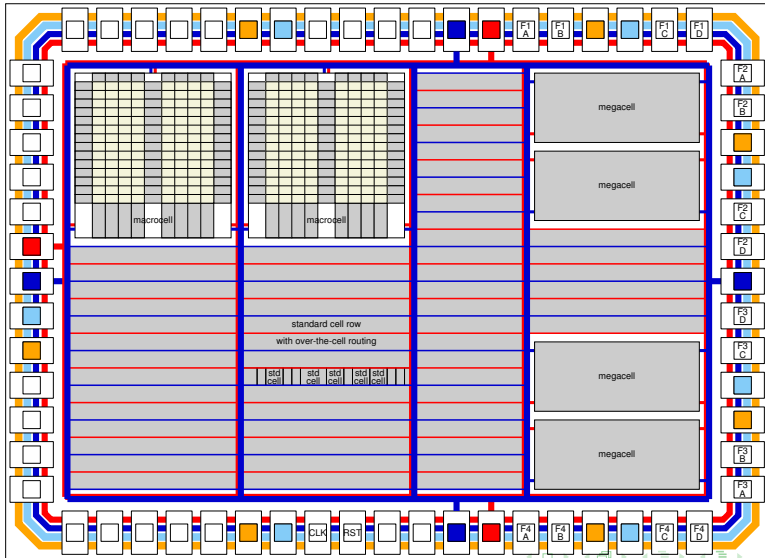
Cell library abstracts

- ▶ 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.

Observation

Many digital VLSI circuits are being designed without the design engineers knowing all circuit and layout details of the library elements they rely upon.

Typical cell mix in a full-custom IC



The design engineer's point of view II

Automatic circuit synthesis

Logic synthesis: accepts logic equations, truth tables, and state graphs; generates gate-level netlists for combinat. logic and for finite state machines (FSM) \rightsquigarrow absorbed in today's EDA flows.

The design engineer's point of view II

Automatic circuit synthesis

Logic synthesis: accepts logic equations, truth tables, and state graphs; generates gate-level netlists for combinat. logic and for finite state machines (FSM) \rightsquigarrow absorbed in today's EDA flows.

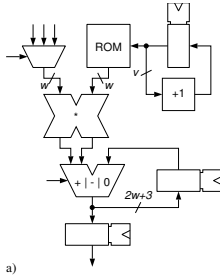
Register transfer level (RTL) synthesis:

- ▶ Circuit viewed as a network of storage elements — registers and also RAMs — that are held together by combinational building blocks.
- ▶ Behavioral specs allowed to include arithmetic functions, string operations, arrays, enumerated types, etc.

\rightsquigarrow universally adopted since the early 1990s.

+ technology-independent, supports parametrization, favors reuse.

RTL views of small circuits



b)

```
architecture procedural of patternmatch is
  signal PREST : Std_Logic_Vector(0 to 5);
begin
  allbits : for i in 1 to 5 generate
    process (CLK,CLR) is
    begin
      if CLR='1' then
        PREST(i) <= '0';
      elsif CLK'event and CLK='1' then
        PREST(i) <= PREST(i-1);
      end if;
    end process;
  end generate;

  PREST(0) <= INP;

  OUP <= true when PREST(1 to 5)="11011"
    else false;

end architecture procedural;
```

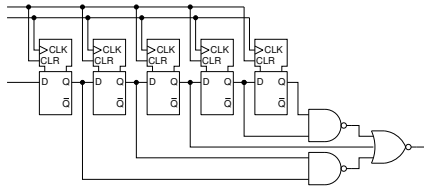


Figure: RTL diagram (a), RTL synthesis model (b), and gate-level schematic (c) (simplified, note that (a) and (b) refer to different circuits).

The design engineer's point of view 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.

The design engineer's point of view 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.

Observation

For arbitrary applications, architecture synthesis represents a formidable optimization problem. Automation does not always yield optimal results and continues to be an active field of research.

The design engineer's point of view IV

Design with 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 & route, 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.

The design engineer's point of view IV

Design with 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 & route, 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.

Observation

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

The design engineer's point of view (summary)

Options for capturing a design with EDA tools

Hand layout Confined to niches (such as memories, analog subcircuits, library cells, and high-performance datapaths).

Schematic entry Confined to high-level block diagrams and a few out-of-the-ordinary subcircuits (neither available from cell libraries nor amenable to synthesis).

RTL synthesis HDL code (VHDL or Verilog) $\xrightarrow{\text{automatic}}$ gate-level netlist.
Universally adopted.

Architecture synthesis SW code $\xrightarrow{\text{automatic}}$ RTL synthesis model.
Research goal, viable for specific fields of applications.

Incorporation of VCs Purchased HDL code $\xrightarrow{\text{automatic}}$ gate-level netlist.
Routine practice today.

↪ Several approaches are typically combined.

A second IC classification scheme

Fabricat. depth	Electrical configuration	Semi-custom fabrication	Full-custom fabrication	
Design level	Cell-based as obtained from <ul style="list-style-type: none"> ○ synthesis with VCs in HDL form, ○ synthesis from captive HDL code, ○ schematic entry, or a mix of these 		Hand layout	
Product name	Field- programmable logic device (FPGA, CPLD)	Gate-array, sea-of-gates, or structured ASIC	Standard cell IC	Full-custom IC

IC families as a function of fabrication depth and design abstraction level with focus of this course **highlighted**.

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.
⇒ LISA, SystemVerilog, SystemC, etc.
- ▶ Make it possible to start software development before hardware design is completed.
- ▶ Improve the coverage and efficiency of functional verification.
 - ▶ deal with system-level transactions
 - ▶ take advantage of formal verification

⇒ Vera, e, SystemVerilog, PSL, etc.

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, SMIC, etc.

Fabless chip vendor Develops and markets proprietary semiconductor components but has their manufacturing commissioned to an independent silicon foundry. (≈ 1300 worldwide @ 2008)
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 business).
- 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 & smartphones), 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?

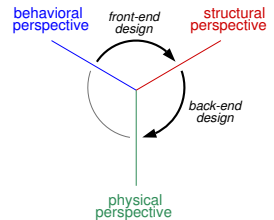
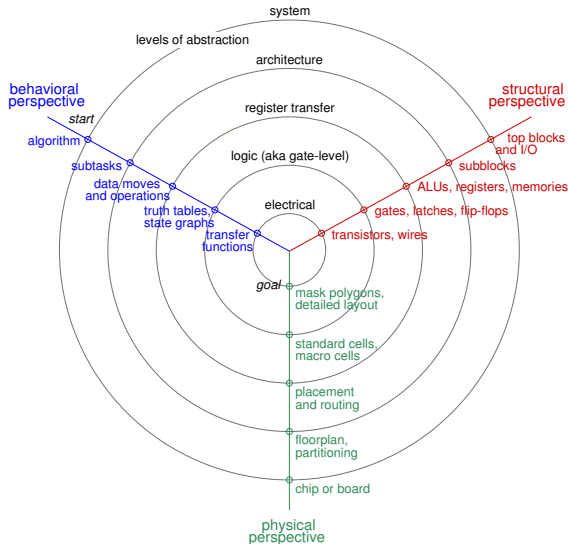
Three factors came together to make fabless operation possible:

- ▶ 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.

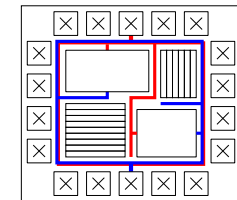
Subject

Design flow in digital VLSI

The Y-chart



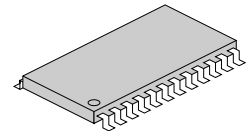
More design views



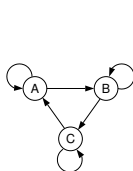
a)

architecture seriesexpansion of cosine is
begin
 process (theta) is
 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 seriesexpansion;

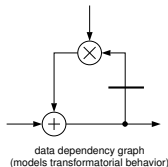
b)



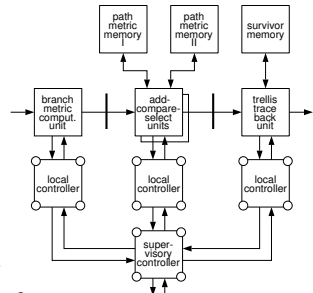
c)



d)



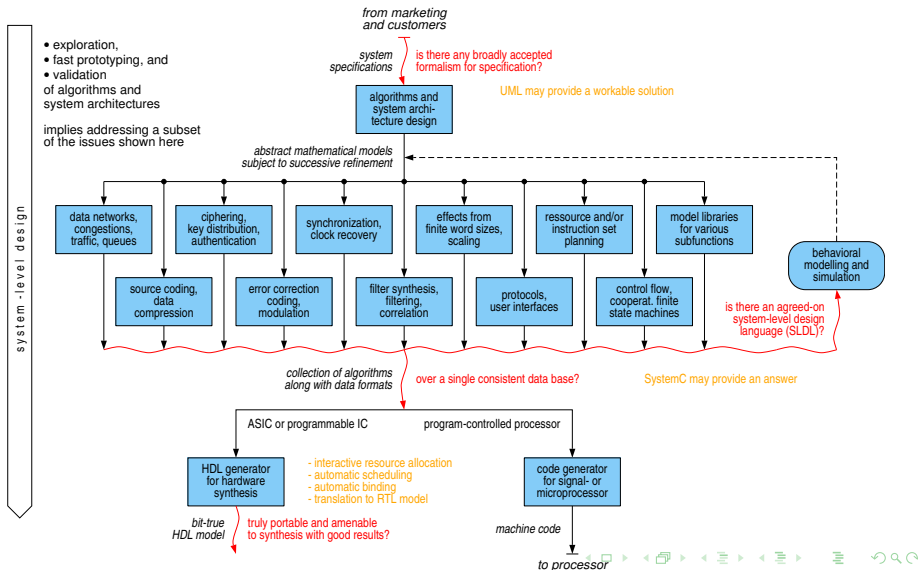
e)



f)

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



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.**

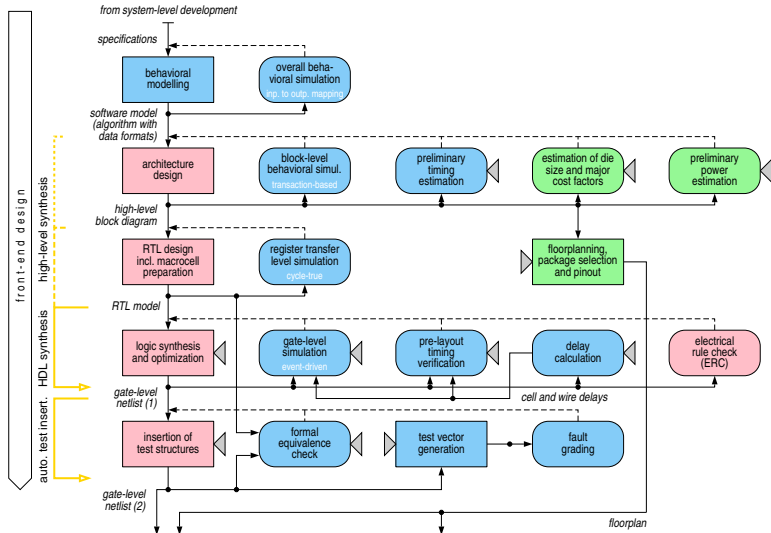
Algorithm design

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)



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.

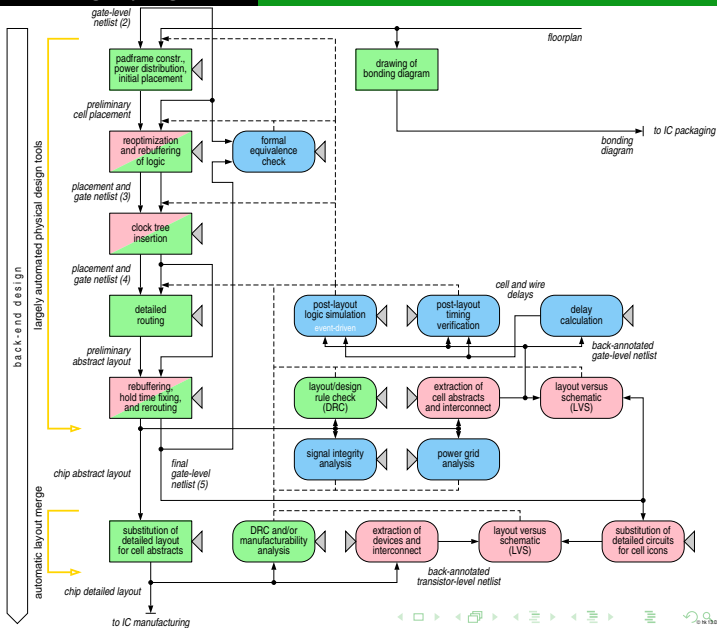
Architecture design II

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, padframe generation, etc.)
- ▶ 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.**

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.**

Common theme throughout the design flow

Many analysis steps and corrective loops (rounded boxes, backward arrows).

Why?



To avoid this!

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.

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

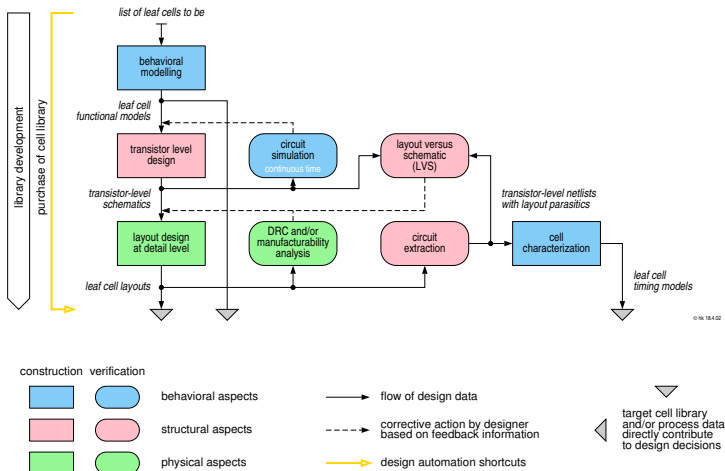
“First time right” design as a guiding principle.

Consequence

VLSI engineers typically spend much more time verifying a circuit than actually designing it.

Cell library design

occurs quite separately from actual IC design in specialized companies.



© 1998 IBM Corp.

Subject

Field-programmable logic

Overview

Key properties of an FPL device depend on choices along two dimensions.

Configuration technology

What are the programmable links?

How is the configuration stored?

How many times can it be changed?

Can this be done without removing the device from the board?

Organization of hardware resources

What prefabricated hardware resources are made available to customers? ¹

How can they be made to form a larger circuit?

¹Customers = designers who want to implement their own circuits in an FPL device.

Overview

Key properties of an FPL device depend on choices along two dimensions.

Configuration technology

What are the programmable links?

How is the configuration stored?

How many times can it be changed?

Can this be done without removing the device from the board?

Organization of hardware resources

What prefabricated hardware resources are made available to customers? ¹

How can they be made to form a larger circuit?

- All configuration techniques in use today have their roots in semiconductor memory technology (SRAM, flash, PROM).

¹Customers = designers who want to implement their own circuits in an FPL device.

Configuration technologies for field-programmable logic

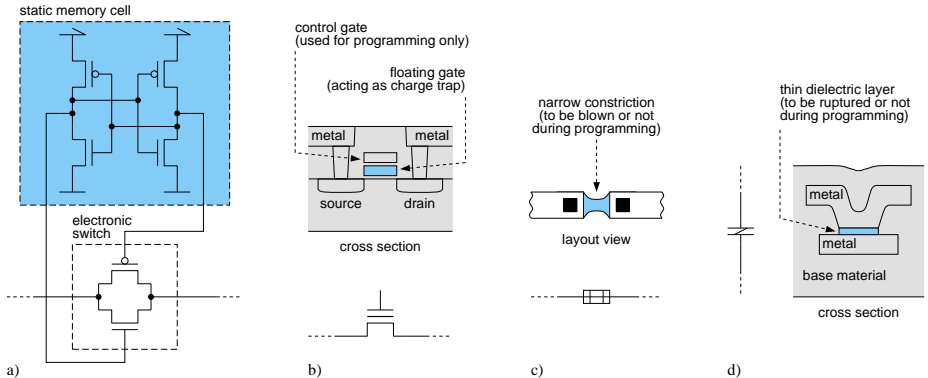


Figure: Electrical connections that can be done by electrical means.

- SRAM: Switch steered by static memory cell (a),
- Flash: MOSFET controlled by trapped charge (b),
- PROM: fuse (c) and antifuse (d).

The pros and cons of storing configuration data in SRAM-type circuits

- + Unlimited in-circuit reprogrammability
- Huge overhead in terms of transistor count and area
- Volatile (configuration gets lost whenever circuit is powered down)

The pros and cons of storing configuration data in SRAM-type circuits

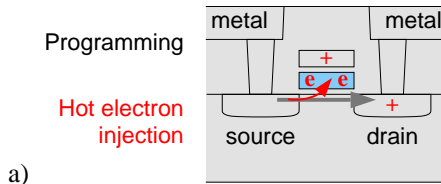
- + Unlimited in-circuit reprogrammability
- Huge overhead in terms of transistor count and area
- Volatile (configuration gets lost whenever circuit is powered down)

The need to (re-)obtain the configuration from outside at power-up is solved in one of three possible ways:

- o by reading from a dedicated off-chip ROM (bit-serial or bit-parallel),
- o by downloading a bit stream from a host computer, or
- o by long-term battery backup.

Operation principles of flash memory I

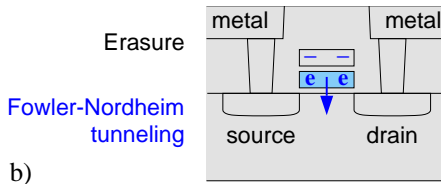
- ▶ A floating gate is sandwiched between bulk material and a control gate.
- ▶ The electrical charge trapped there turns the MOSFET “on” or “off”.



Programming occurs by way of **hot electron injection** from the channel. A strong **lateral field** accelerates electrons to the point where they get injected through the thin dielectric layer into the floating gate. The necessary programming voltage on the order of 5 to 20 V is generated by an on-chip charge pump.

Operation principles of flash memory II

- ▶ A floating gate is sandwiched between bulk material and a control gate.
- ▶ The electrical charge trapped there turns the MOSFET “on” or “off”.



Erasure removes trapped electrons by having them tunnel through the oxide layer underneath by way of **Fowler-Nordheim tunneling** that occurs when a strong **vertical field** ($\approx 8 \dots 10 \text{ MV/cm}$) is applied across the gate oxide.

The pros and cons of flash-based FPL devices

- + Non-volatile (no need to be configured following power-up)
- + Reconfigurable through package pins (no need for UV exposure)
- + Data retention times 10 to 40 years
- Endurance 100 to 1000 configure-erase cycles (less than for flash memory)

The pros and cons of antifuse-based FPL devices

- ± Programming is permanent
 - ▶ No need to be configured following power-up
 - ▶ New part required for each bug fix or design update
- + Higher layout densities than with reprogrammable links
(antifuses are only about the size of a contact or via)
- + Less sensitive to radiation
- + Superior protection against unauthorized cloning

FPL configuration technologies compared

Configuration technology	Non volatile	Live at power up	Reconfigurable	Unlimit. endurance	Area occupation per link	Extra fabr. steps
SRAM	no	no	in ckt	yes	large	0
UV-erasable EPROM	yes	yes	out of circuit	no	small in array	3
Electr. erasable EEPROM Flash memory	yes	yes	in ckt	no no	2·EPROM ≈EPROM	> 5
Antifuse PROM	yes	yes	no	n.a.	small	3
Ideal	yes	yes	in ckt	yes	zero	0

Complex programmable logic devices (CPLD)

- Overall organization has evolved from purely combinational devices.

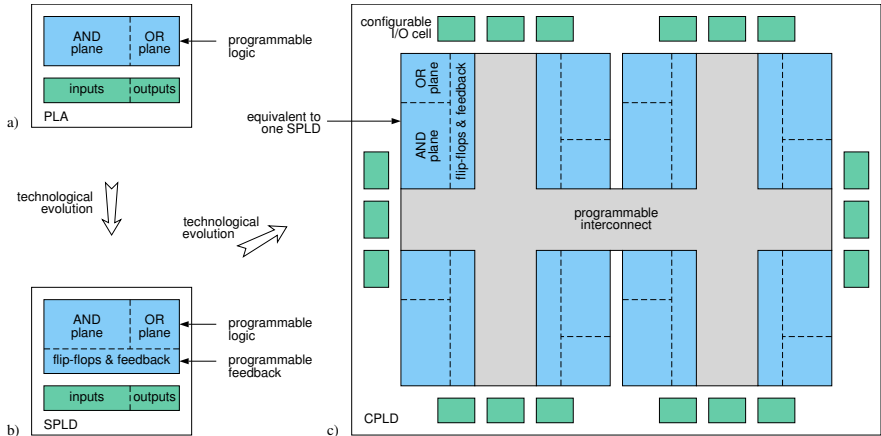


Figure: General architecture of CPLDs (c) along with precursors (a,b).

Field-programmable gate arrays (FPGA)

- Overall organization patterned after mask-programmed gate-arrays.

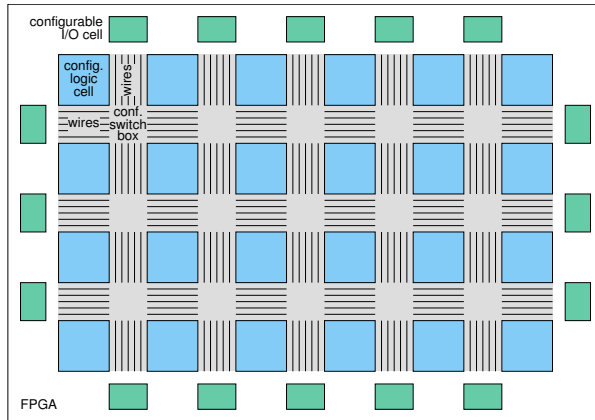


Figure: General architecture of FPGAs.

Fine-grained FPGAs

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

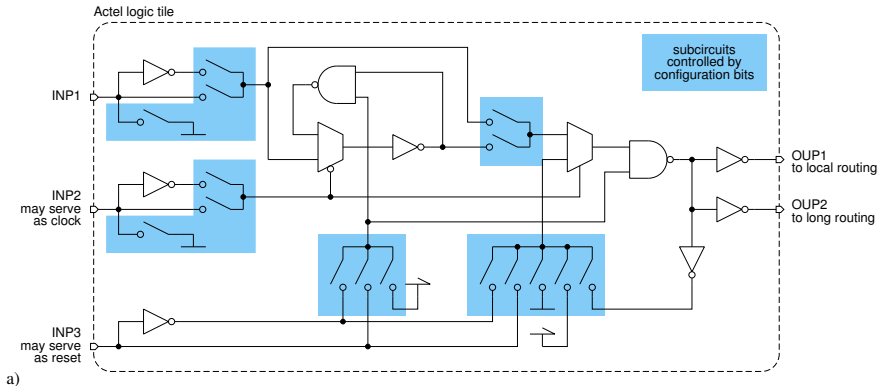


Figure: Example: logic tile from Actel ProASIC.

Coarse-grained FPGAs

- Combinational functions of four or more variables.
- Two or more bits stored per configurable logic cell.

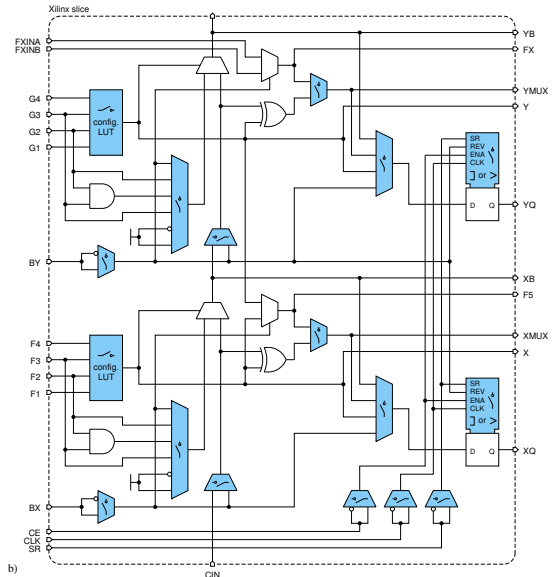


Figure: Example: logic slice from Xilinx Virtex-4 (2 4-input LUTs, 2 bistables).

Introduction to Microelectronics

There has been a trend towards coarser granularities

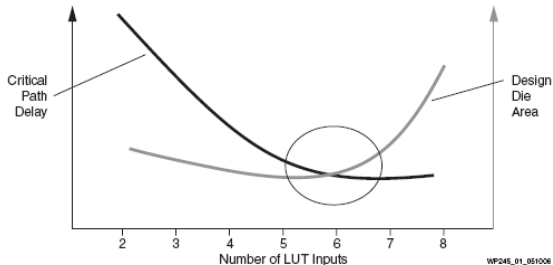
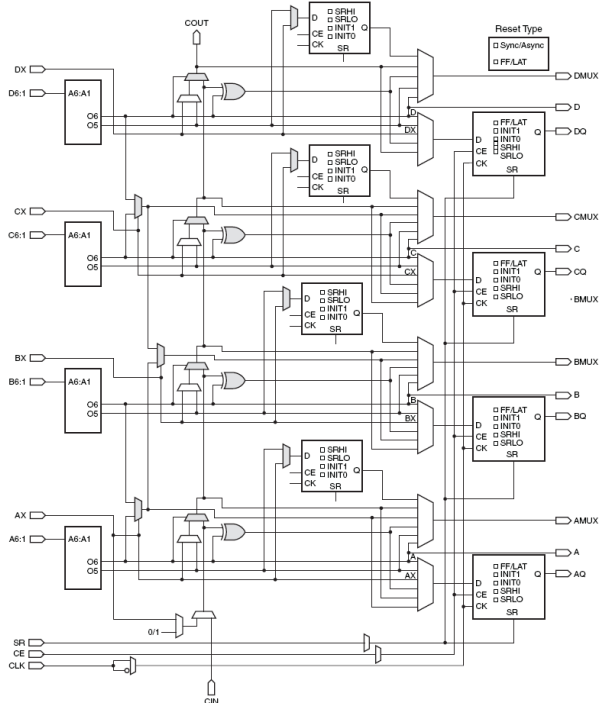


Figure: LUT granularity trade-offs at the 65 nm technology node.

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

© Hubert Kaeslin Microelectronics Design Center ETH Zürich



Commercial products

configuration technology	overall organization of hardware resources		
	CPLD	FPGA coarse grained	fine grained
static memory (SRAM)		Xilinx Virtex, Spartan. Lattice SC, EC, ECP. Altera Stratix, Arria, Cyclone. eASIC Nextreme SL. Achronix Speedster.	Atmel AT6000, AT40K.
UV-erasable (EPROM)	Cypress MAX340 (discontinued)		
electrically erasable (flash)	Xilinx XC9500, CoolRunner-II. Altera MAX. Lattice MACH 1,...,5. Cypress Delta39K, Ultra37000.	Lattice XP MACH XO.	Actel ProASIC3, ProASIC3 nano, Igloo, Fusion.
antifuse (PROM)		QuickLogic Eclipse II, PolarPro.	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.

$$GE_{manuf} > GE_{usable} > GE_{actual}$$

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.

$$GE_{manuf} > GE_{usable} > GE_{actual}$$

Hint

Carry out benchmarks with representative designs as this helps to

- ▶ make better cost calculations,
- ▶ obtain realistic timing figures,
- ▶ avoid misguided choices.

Interim assessment of field-programmable logic

Observation

Field-programmable devices can be thought of as “soft hardware”.

There are limitations, though.

Interim assessment of field-programmable logic

Observation

Field-programmable devices can be thought of as “soft hardware”.

There are limitations, though. Configurable logic cells are

- + designed to implement small LUTs and random logic functions,
- unsuitable for implementing analog subfunctions,
- extremely wasteful (in terms of area, delay and energy) when used to implement
 - ▶ memories of non-trivial size,
 - ▶ datapaths (that include multiplications and related arithmetic-logic operations) on wide data words,
 - ▶ instruction-set processors (where the software affords flexibility), or
 - ▶ fixed functions (that do not ask for flexibility).

Interim assessment of field-programmable logic

Observation

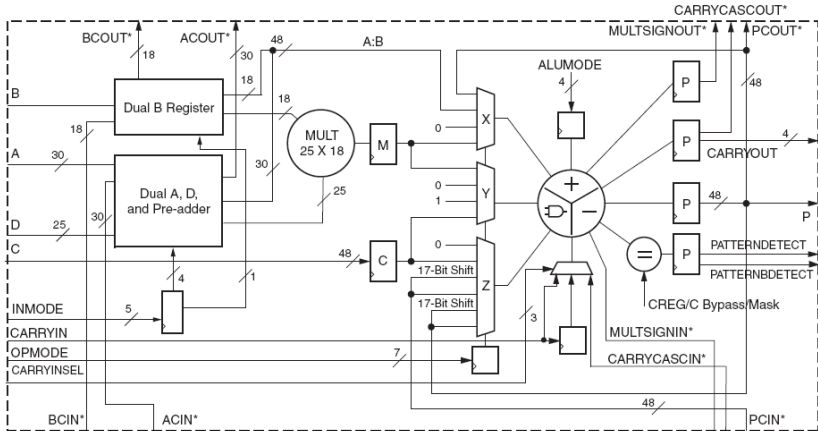
Field-programmable devices can be thought of as “soft hardware”.

There are limitations, though. Configurable logic cells are

- + designed to implement small LUTs and random logic functions,
- unsuitable for implementing analog subfunctions,
- extremely wasteful (in terms of area, delay and energy) when used to implement
 - ▶ memories of non-trivial size,
 - ▶ datapaths (that include multiplications and related arithmetic-logic operations) on wide data words,
 - ▶ instruction-set processors (where the software affords flexibility), or
 - ▶ fixed functions (that do not ask for flexibility).

↪ FPL often gets combined with less malleable but more efficient resources.

Some FPGAs also include wide datapath units (MAC)



*These signals are dedicated routing paths internal to the DSP48E1 column. They are not accessible via fabric routing resources.

Figure: Example: DSP48E slice from Xilinx Virtex-6
(25x18bit multiply, 48bit accumulate).

Further extensions

On top of FPL per se, most commercial parts include

Hardwired subblocks SRAMs, FIFOs, clock recovery circuits, SerDes, etc.

Industry-standard functions and interfaces such as PCI, USB, FireWire, Ethernet, WLAN, JTAG, LVDS, etc.

Analog-to-digital and digital-to-analog converters

Hardwired microprocessor and DSP cores (e.g. PowerPC, ARM)

Weakly configurable analog subfunctions such as filters or PLLs
in various combinations.

Further extensions

On top of FPL per se, most commercial parts include

Hardwired subblocks SRAMs, FIFOs, clock recovery circuits, SerDes, etc.

Industry-standard functions and interfaces such as PCI, USB, FireWire, Ethernet, WLAN, JTAG, LVDS, etc.

Analog-to-digital and digital-to-analog converters

Hardwired microprocessor and DSP cores (e.g. PowerPC, ARM)

Weakly configurable analog subfunctions such as filters or PLLs
in various combinations.

And then there exist

Field-programmable analog arrays (FPAA) built from OpAmps, capacitors, resistors and switchcap elements.

Example: Intel Stellarton

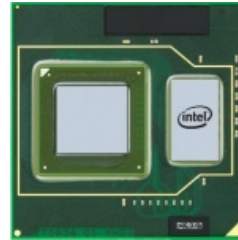


Figure: FPGA (left) and CPU (right) dies mounted on a common carrier.

System-in-Package (SiP) combining

- ▶ Intel Atom CPU (0.6, 1.0 or 1.3 GHz) providing x86 instruction set, DDR2 memory controller, PCIe ports, GPU, display interfaces, etc.
- ▶ Altera FPGA (\approx Arria II EP2AGX65) providing \approx 25 000 adaptive logic modules (ALM), 312 18x18 bit multipliers, 350 customizable pins, etc.
- ▶ Connected with each other via two PCIe-x1-links.
- ▶ Thermal design power 2.7...3.6 W.
- ▶ Unit price 61...106 USD (@ 2011 for orders of 1000).

Example: Intel Stellarton

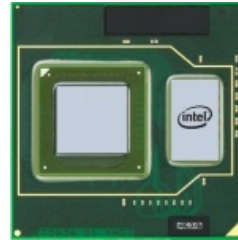


Figure: FPGA (left) and CPU (right) dies mounted on a common carrier.

System-in-Package (SiP) combining

- ▶ Intel Atom CPU (0.6, 1.0 or 1.3 GHz) providing x86 instruction set, DDR2 memory controller, PCIe ports, GPU, display interfaces, etc.
- ▶ Altera FPGA (\approx Arria II EP2AGX65) providing \approx 25 000 adaptive logic modules (ALM), 312 18x18 bit multipliers, 350 customizable pins, etc.
- ▶ Connected with each other via two PCIe-x1-links.
- ▶ Thermal design power 2.7...3.6 W.
- ▶ Unit price 61...106 USD (@ 2011 for orders of 1000).
- ▶ Missing as yet: Monolithic integration \rightsquigarrow platform-IC.

FPL design flow I

Front-end flow is essentially the same as for ASICs:

1. Architecture design.
2. HDL coding.
3. Functional verification (mostly by way of simulations).
4. HDL synthesis \mapsto Gate-level netlist.

FPL design flow I

Front-end flow is essentially the same as for ASICs:

1. Architecture design.
2. HDL coding.
3. Functional verification (mostly by way of simulations).
4. HDL synthesis \mapsto Gate-level netlist.

Particularities of coarse-grained FPGAs:

- ▶ Look-up tables cheap.
- ▶ Routing dominates over gate delay due to conf. switches and larger die.
- ▶ Flip-flops come in generous numbers \rightsquigarrow pipelining is essentially free.
- ▶ On-chip clock preparation circuits (nets, drivers, PLLs).
- ▶ Sophisticated input/output circuits (adjustable, LVDS, synchronization).
- ▶ On-chip block RAMs (depending on product).
- ▶ Available with on-chip microcontroller (depending on product).

FPL design flow II

Back-end flow differs considerably.

5. Gate-level netlist is mapped onto the configurable cells available in the target device.
6. Interconnect gets implemented using the wires, switches and drivers available there.
7. Result is converted into a **configuration bit stream** for download into the FPL device.

FPL vendors make available proprietary tools for the above procedure.

FPL design flow II

Back-end flow differs considerably.

5. Gate-level netlist is mapped onto the configurable cells available in the target device.
6. Interconnect gets implemented using the wires, switches and drivers available there.
7. Result is converted into a **configuration bit stream** for download into the FPL device.

FPL vendors make available proprietary tools for the above procedure.

Hierarchy of required skill sets

Field-programmable logic \subset Semi-custom ICs \subset Full-custom ICs

- The VLSI I course covers those topics that matter independently of fabrication depth (architectures, functional verification, HDL coding, synchronous clocking, etc.).

Pros and cons of field-programmable logic

- + Easy and extremely fast to modify (in minutes instead of months).
- + Designers can focus on functionality right away.
No need to care about subordinate details such as
 - ▶ I/O subcircuits,
 - ▶ clock distribution,
 - ▶ power distribution,
 - ▶ embedded memories,
 - ▶ testability, etc.
- + Low initial effort, lower than any other hardware alternative.
- + Affordable design tools.

Pros and cons of field-programmable logic

- + Easy and extremely fast to modify (in minutes instead of months).
- + Designers can focus on functionality right away.
No need to care about subordinate details such as
 - ▶ I/O subcircuits,
 - ▶ clock distribution,
 - ▶ power distribution,
 - ▶ embedded memories,
 - ▶ testability, etc.
- + Low initial effort, lower than any other hardware alternative.
- + Affordable design tools.
- ± Devices come in thousands of variations, may be confusing.
- Huge overhead in terms of area, delay (performance), and energy.
- **Cost-effective for small volumes**, not economic for large quantities.

Conclusions

Field-programmable logic is best for

- ▶ Prototyping and other
- ▶ Situations where specs are subject to change at any time
- ▶ Products that sell in modest quantities
- ▶ Products that need to be reconfigured or updated from remote

Cost structure to be examined in chapter 13 "VLSI Economics and Project Management"

Appendix

A brief glossary of logic families

Major semiconductor technologies and logic families

The alphabet soup explained.

Acronym	Meaning
MOS	Metal Oxide Semiconductor.
FET	Field Effect Transistor (n- or p-channel)
BJT	Bipolar Junction Transistor (nnp or pnp)
NMOS	n-channel MOS (transistor, circuit or technology)
PMOS	p-channel MOS (transistor, circuit or technology)
CMOS	Complementary MOS (circuit or technology)
static CMOS	data stored in bistable subcircuits and retained
dynamic CMOS	data stored as electrical charges to be refreshed
TTL	Transistor Transistor Logic (BJTs & passive devices)
ECL	Emitter-Coupled Logic (non-saturating logic)
BiCMOS	CMOS & bipolar devices on a single chip

2-input NAND gate in TTL technology

TTL invented 1961 as an improvement over DTL and RTL.

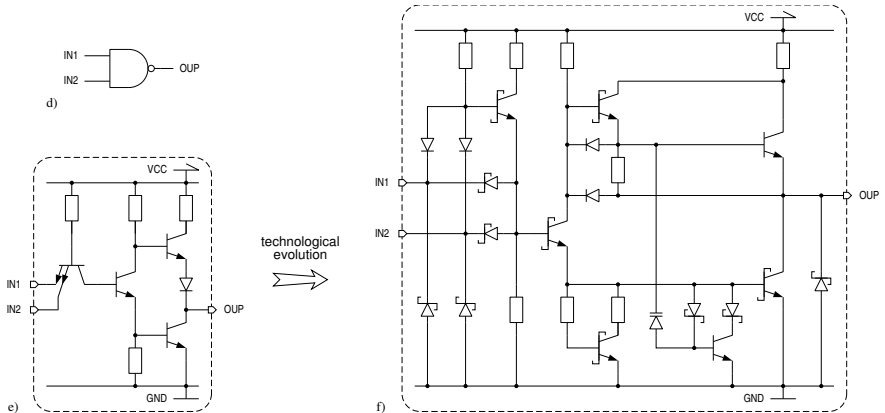
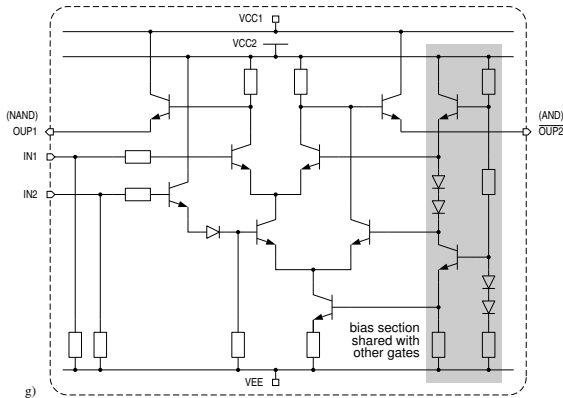


Figure: Icon (d), original multi-emitter circuit (e), and more recent F generation circuit (f). The auxiliary devices serve clamping and speed-up purposes.

2-input NAND gate in ECL technology

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












device	MOSFET	BJT	
icon	 p-channel	 pnp	
	 n-channel	 npn	
approx. behavior	voltage-controlled current source	current-controlled current source	
variations	 depletion device	 Schottky device	
device	resistor	diode	varicap
icon			

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.

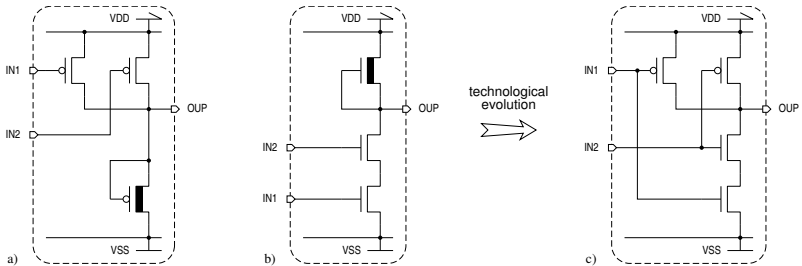


Figure: PMOS (a), NMOS (b), and static CMOS (c) circuits.

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.

Result

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