Introduction to Microelectronics

Prof. Hubert Kaeslin
Microelectronics Design Center
ETH Zürich

Morgan Kaufmann "Top-Down Digital VLSI Design" Chapter 1

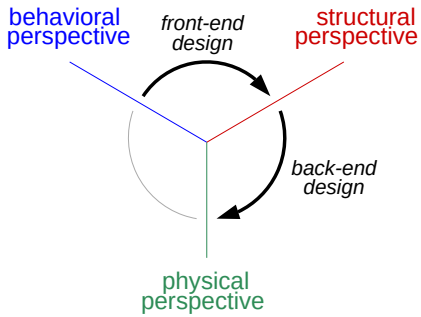
last update: July 18, 2014

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
  - Front-end (architecture and circuit design)
  - Back-end (layout design and verification)
- CMOS compared to other logic families



Subject

Economic impact

Worldwide semiconductor market by vendors

Rank	Vendor	Revenue [GUSD]	Share [%]	Share [%]
1	Intel	49.1	16.4	
2	Samsung Electronics	28.6	9.5	
3	Qualcomm	13.2	4.4	
4	Texas Instruments	11.1	3.7	
5	Toshiba	10.6	3.5	
6	Renesas Electronics	9.2	3.1	
7	SK Hynix	9.0	3.0	
8	ST-Microelectronics	8.4	2.8	
9	Broadcomm	7.8	2.6	
10	Micron Technology	6.9	2.3	
...	others	146.0	48.7	
	Total	299.9	100	0.42
for comparison	World GDP	71 830		100

Source: Gartner and Wikipedia, 2012 data.

Economic leverage of semiconductors I

Observation

Semiconductors account for roughly 0.4% of the world gross domestic product.

Economic leverage of semiconductors I

Observation

Semiconductors account for roughly 0.4% 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

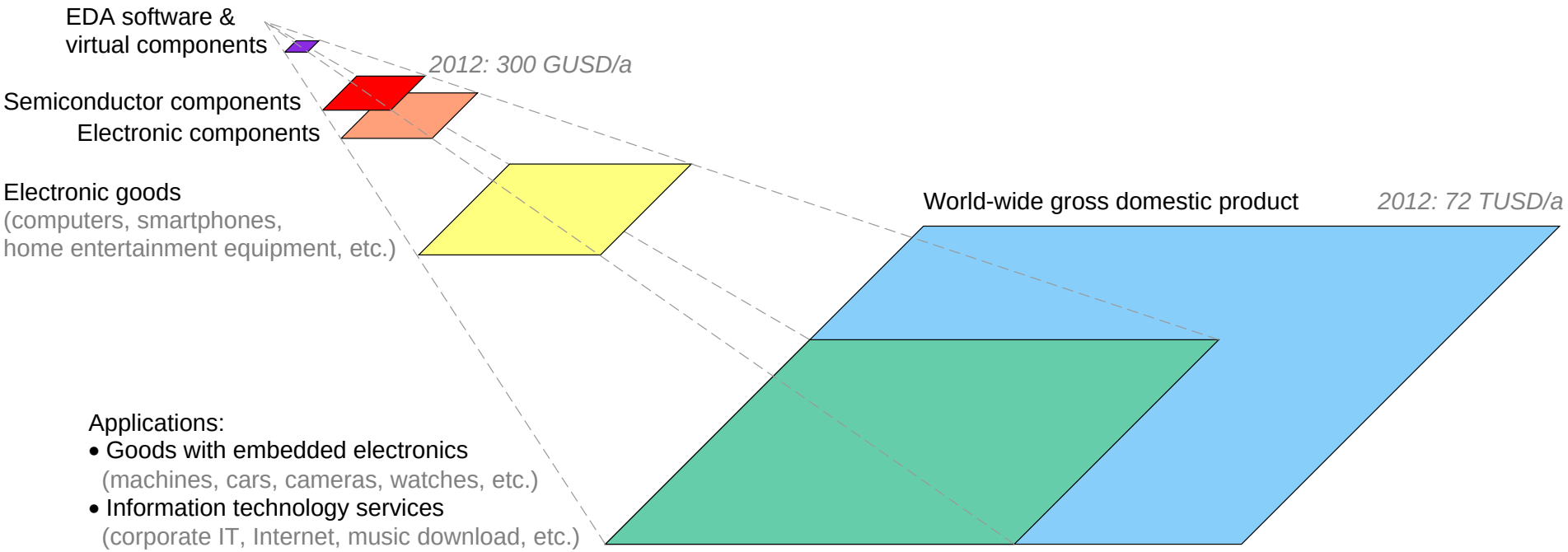


Figure: Impact of microelectronics on "downstream" industries and services.

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 microelectronics on consumer goods I

Figure: A TV player, a jukebox, a pocket calculator, a mobile phone, a .....

Impact of microelectronics on consumer goods II

Figure: The same four functions just a couple of years earlier (ca. 2005).

Impact of microelectronics on consumer goods III

Figure: Similar products that include no large-scale integrated circuits (1970s).

Subject

Microelectronics viewed from different perspectives

1st perspective: The Guiness book of records

"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

1st perspective: The Guiness book of records

"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 (npn 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

You may want to refer to appendix I for more details.

2-input NAND gate in various technologies

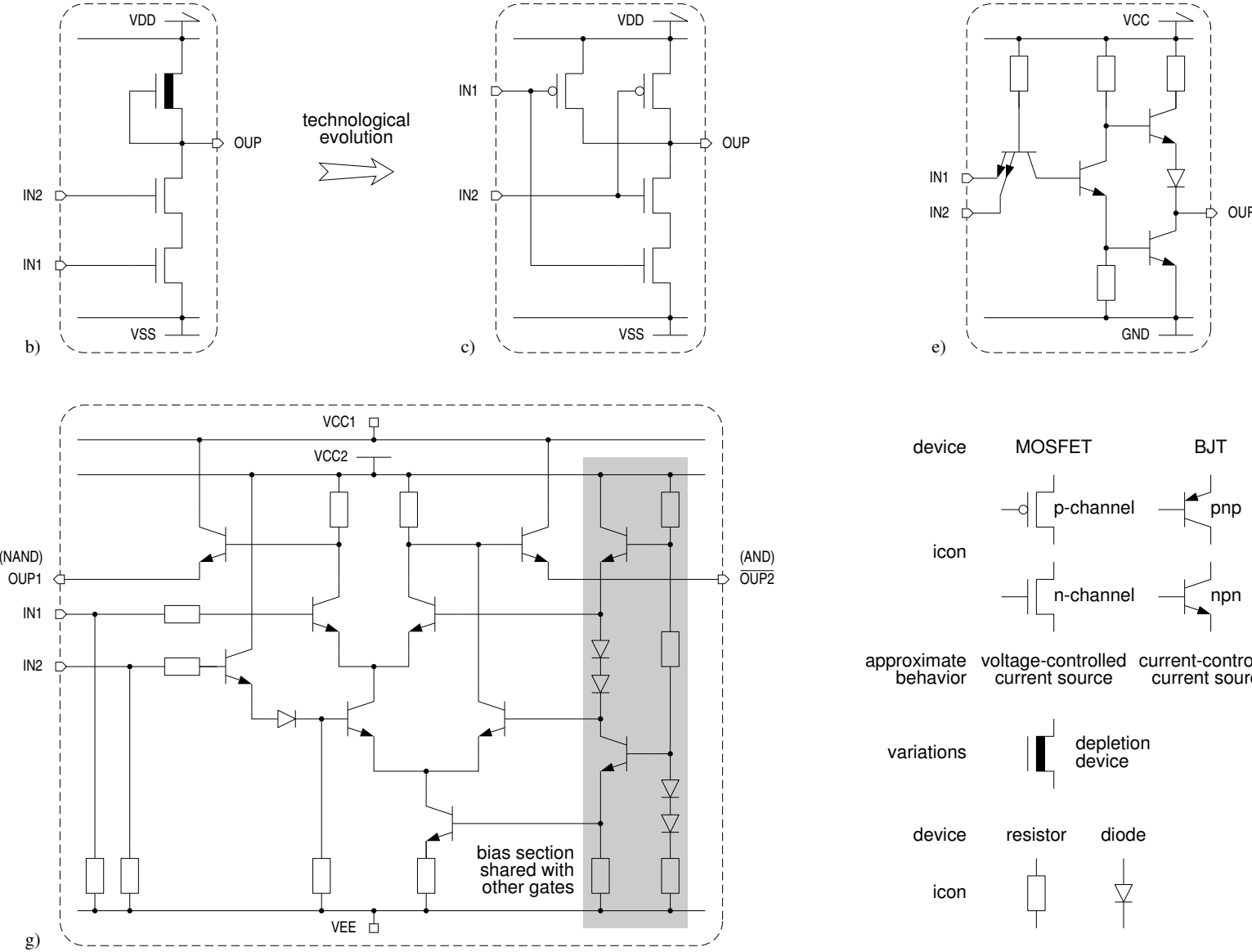


Figure: Static CMOS (c), NMOS (b), early TTL (e), and ECL circuit (g).

2nd perspective: Marketing

"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.

2nd perspective: Marketing

"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 A4,5,6,... SoC introduced with the iPad, 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

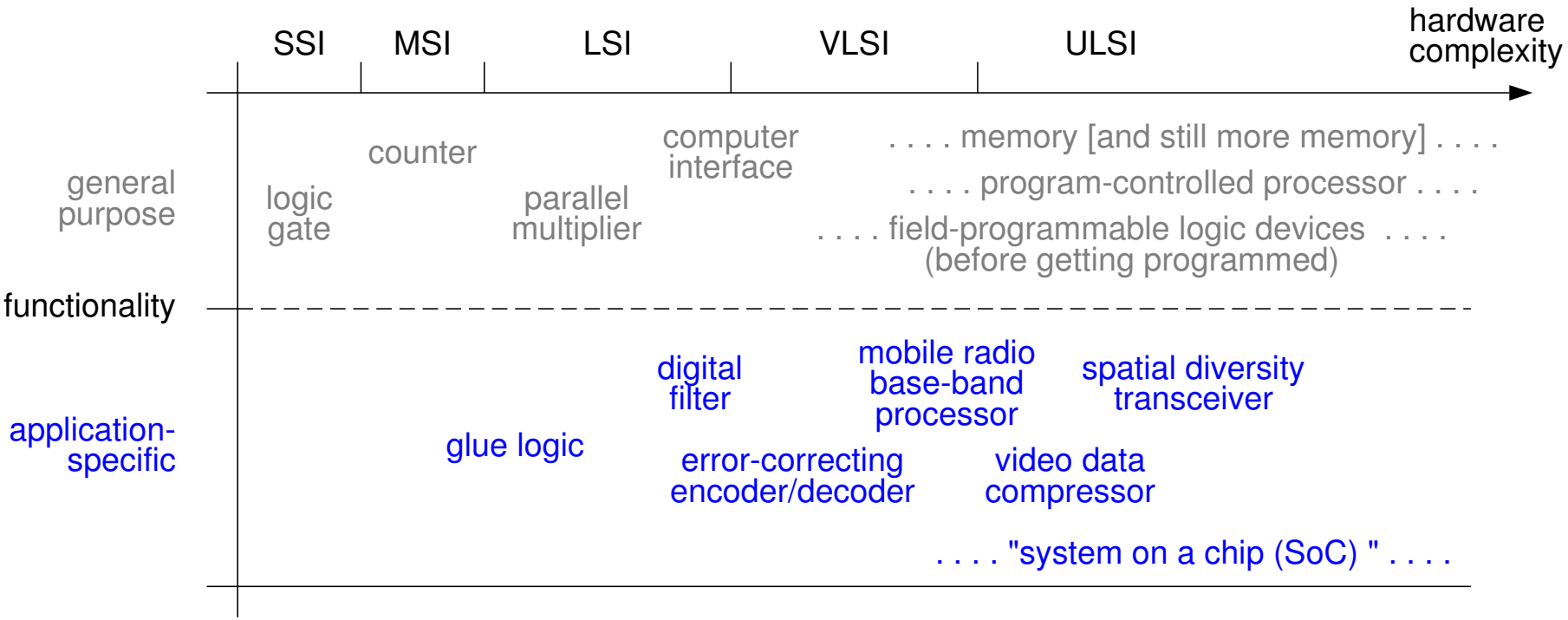


Figure: ICs classified as a function of functionality and hardware complexity.

3rd perspective: Manufacturing

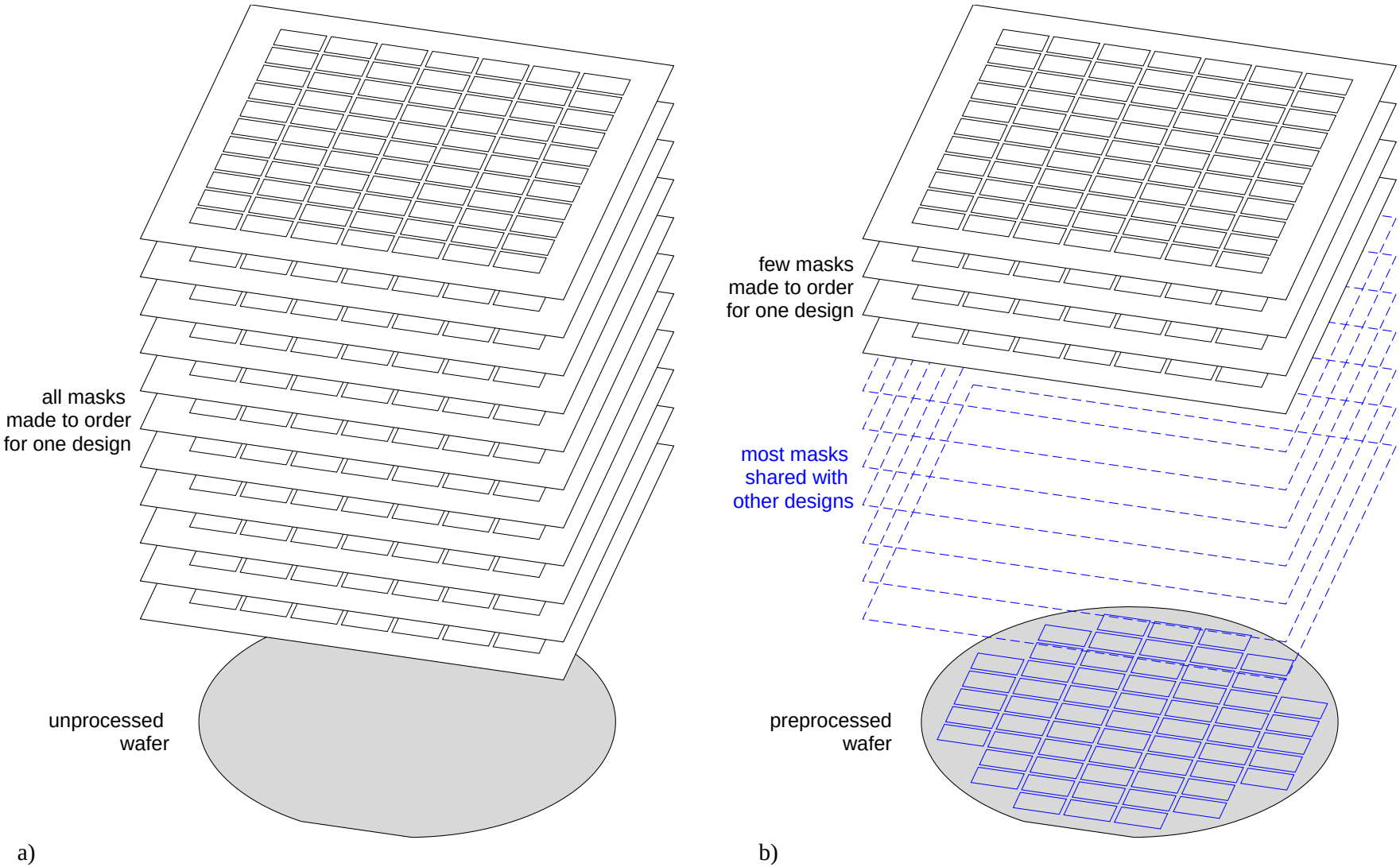
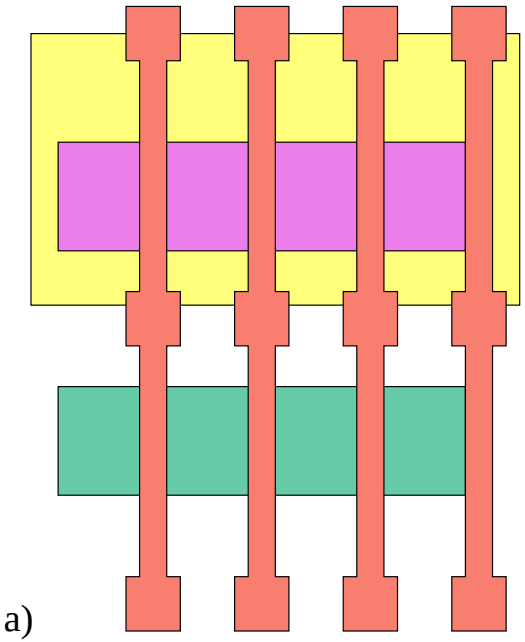


Figure: Full-custom (a) and semi-custom (b) masks sets compared.

Semi-custom fabrication I



a)

preprocessed master

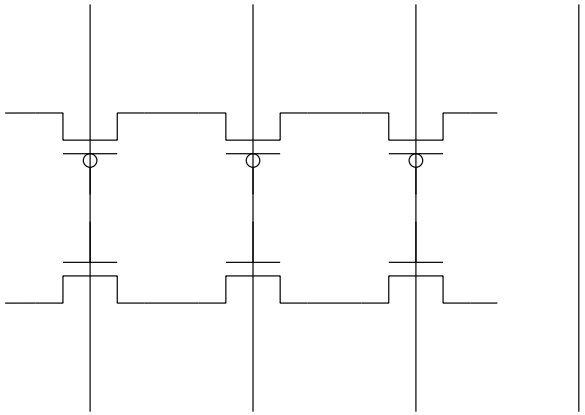


Figure: Gate array site with six prefabricated MOS transistors.

Semi-custom fabrication II

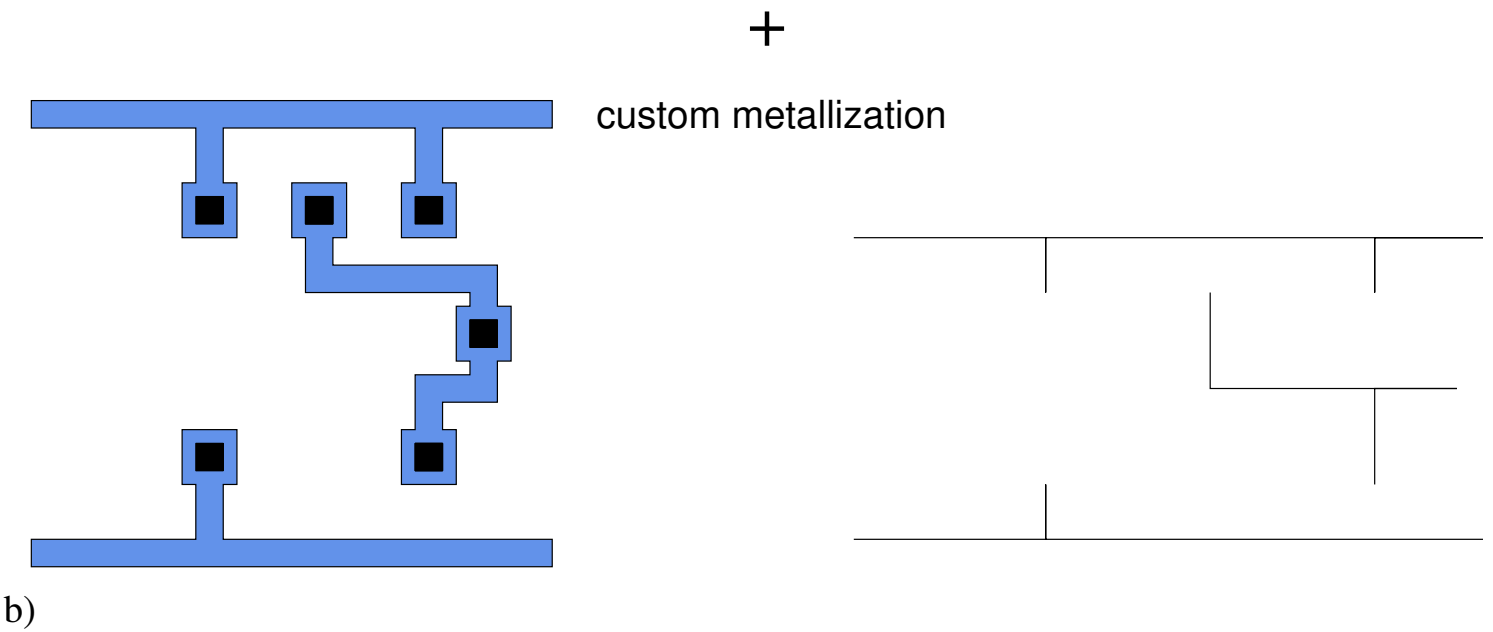


Figure: Custom-designed contact and metal masks.

Semi-custom fabrication III

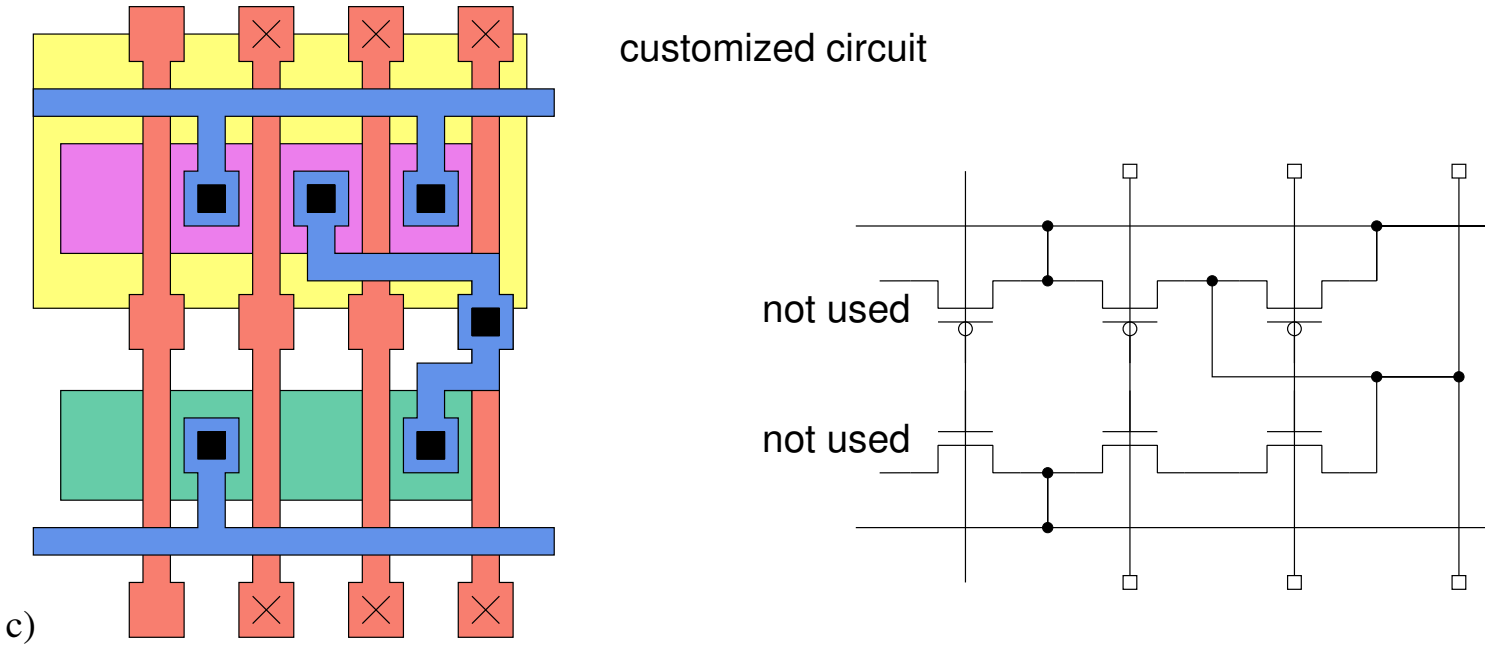


Figure: Site customized into a 2-input NAND gate.

Evolution of semi-custom floorplans

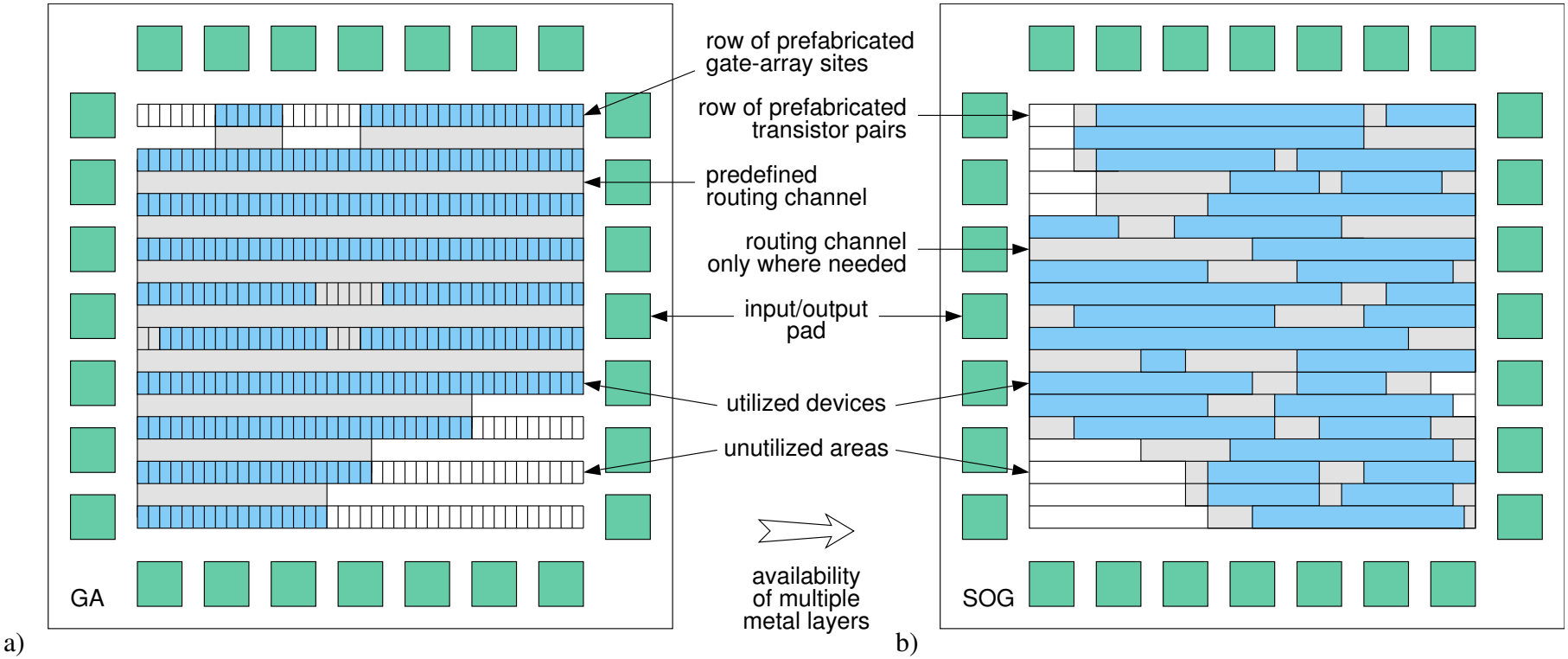


Figure: Channeled gate-array (a) versus channelless semi-custom circuits (b).

Field-programmable logic (FPL)

- Uses neither custom layout structures nor proprietary photomasks.
- 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.

To be explained in chapter 2 "Field Programmable Logic".

Field-programmable logic (FPL)

- Uses neither custom layout structures nor proprietary photomasks.
- 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.

To be explained in chapter 2 "Field Programmable Logic".

- 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.
- 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.

To be explained in chapter 2 "Field Programmable Logic".

- 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".

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.

4th perspective: 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.

4th perspective: 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.

4th perspective: 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.

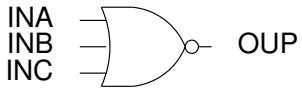
Macrocells layout put together on a per case basis according to specs from a limited collection of layout tiles. Examples: RAMs and ROMs.

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



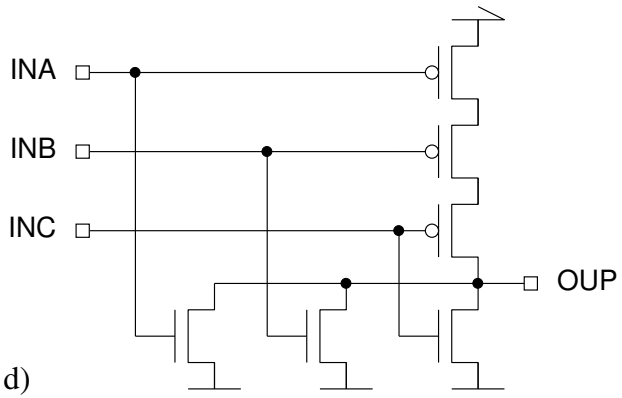
a)

```
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;
```

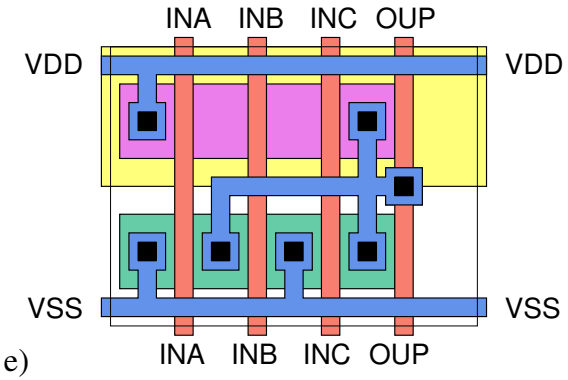
b)

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

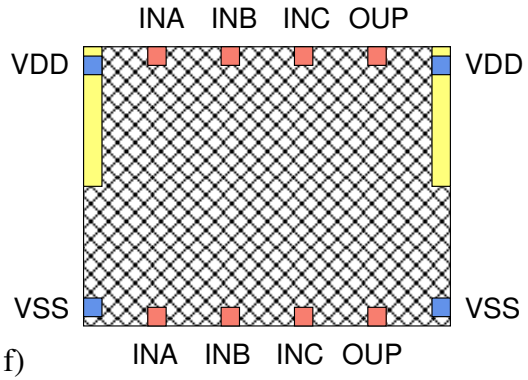
c)



d)



e)



f)

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

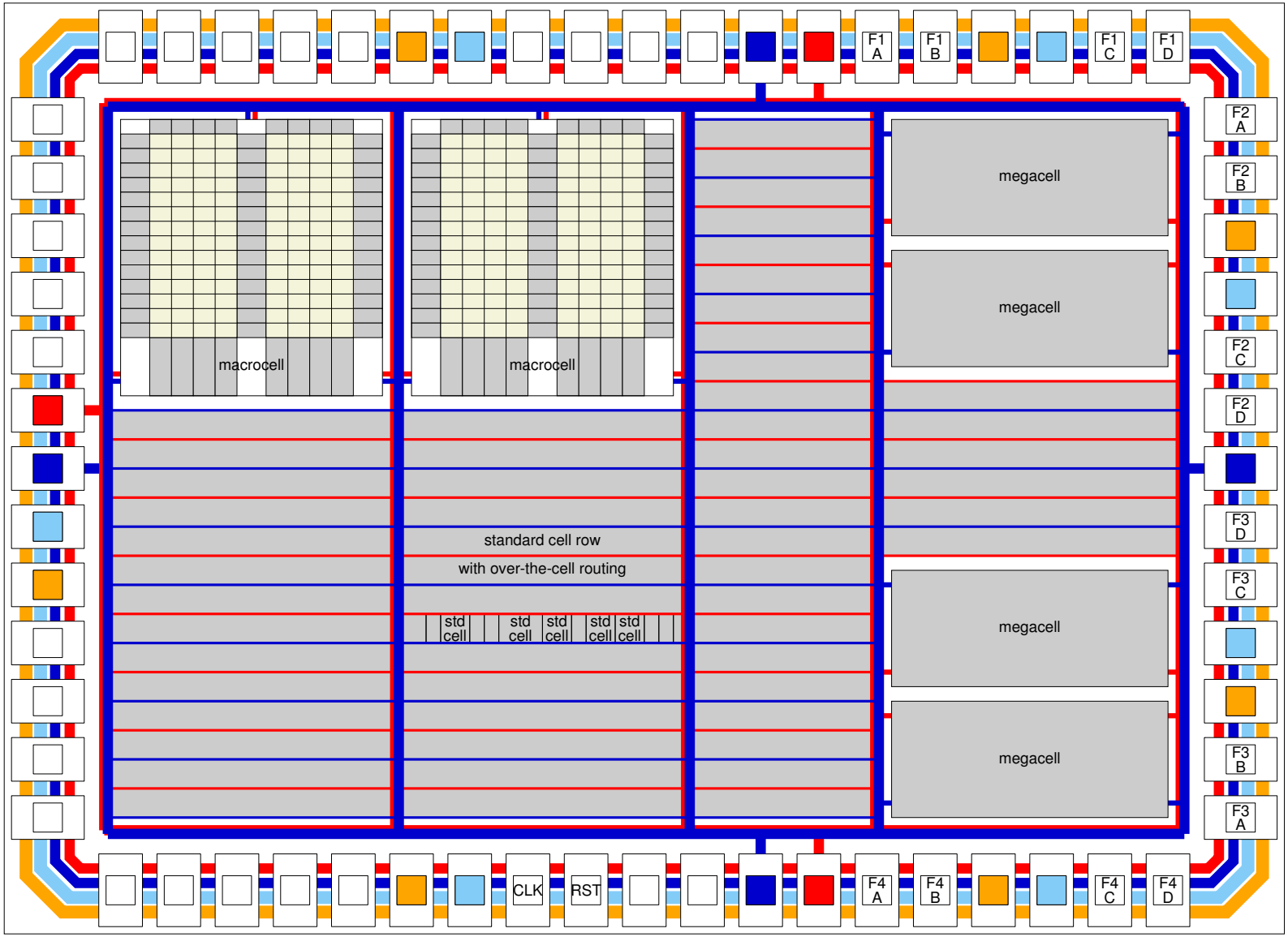
Cell library abstracts

- 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:

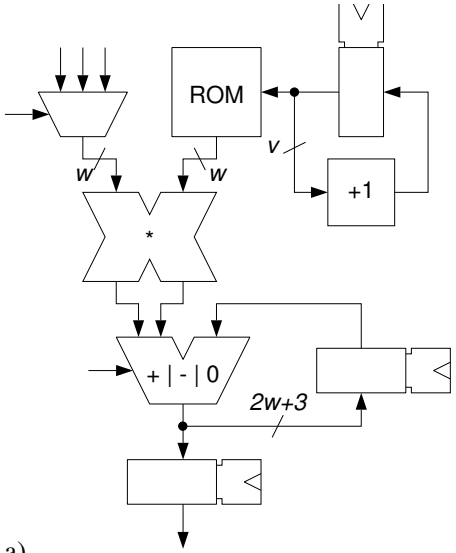
- Circuit viewed as a network of storage elements — registers and also RAMs — held together by combinational logic.
- Behavioral specs allowed to include arithmetic functions, string operations, arrays, enumerated types, etc.

$+$ technology-independent, supports parametrization, favors reuse.

$+$ dispenses with manual logic design, fair productivity.

$\rightsquigarrow$ universally adopted since the early 1990s.

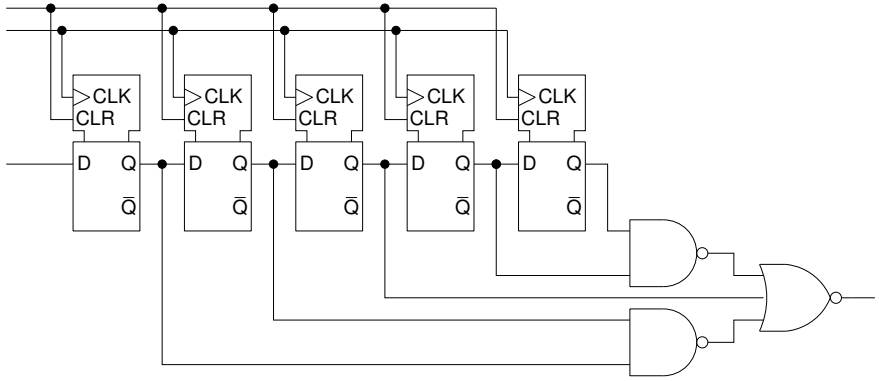
RTL views of small circuits



a)

```
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;
```

b)



c)

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 algorithmic description. Source code includes no explicit indications for how to marshal data processing operations and hardware resources.

1. Identify the computational and storage requirements.
2. Select a suitable building block for each 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 algorithmic description. Source code includes no explicit indications for how to marshal data processing operations and hardware resources.

1. Identify the computational and storage requirements.
2. Select a suitable building block for each 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.

$\mapsto$ Formidable optimization problem. Pros and cons:

$+$ Rapid exploration of design space. Good results in specialized areas.

$-$ Does not normally yield optimal results for arbitrary applications.

$\rightsquigarrow$ 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 (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 (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 Important at its time, largely confined to analog circuit design today.

RTL synthesis HDL code (VHDL or SystemVerilog) $\overset{automatic}{\longmapsto}$ gate-level netlist. Universally adopted.

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

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

$\rightsquigarrow$ Two or three 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 ◦ 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

Competitive pressure 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.
  $\rightsquigarrow$ LISA, 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

  $\rightsquigarrow$ e, PSL, Vera, SystemVerilog, etc.

5th perspective: Business model

"How are the industrial activities shared among business partners?"

5th perspective: Business model

"How are the industrial activities shared among business partners?"

Players in semiconductor markets I

Original business model:

Integrated device manufacturer (IDM) A chip vendor who operates his own wafer processing facilities.

Examples: Intel, Toshiba, Samsung, ST-Microelectronics, Texas Instruments, Cypress Semiconductor, AMS, etc.

Players in semiconductor markets II

Later 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.
Examples: Altera and Xilinx (FPL), Broadcom (networking), Cirrus Logic-Crystal (audio and video chips), Nvidia (graphics chips), Qualcomm (wireless telecommunication).

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: Luxtera, Sensirion.

Players in semiconductor markets III

Intellectual property (IP) vendor A company that develops hardware subfunctions (std/macro/megacells, virtual components) and licenses them to others for incorporation into their own ICs.
Examples: ARM, Faraday, Sci-worx.

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 (smartphones & tablets), 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?

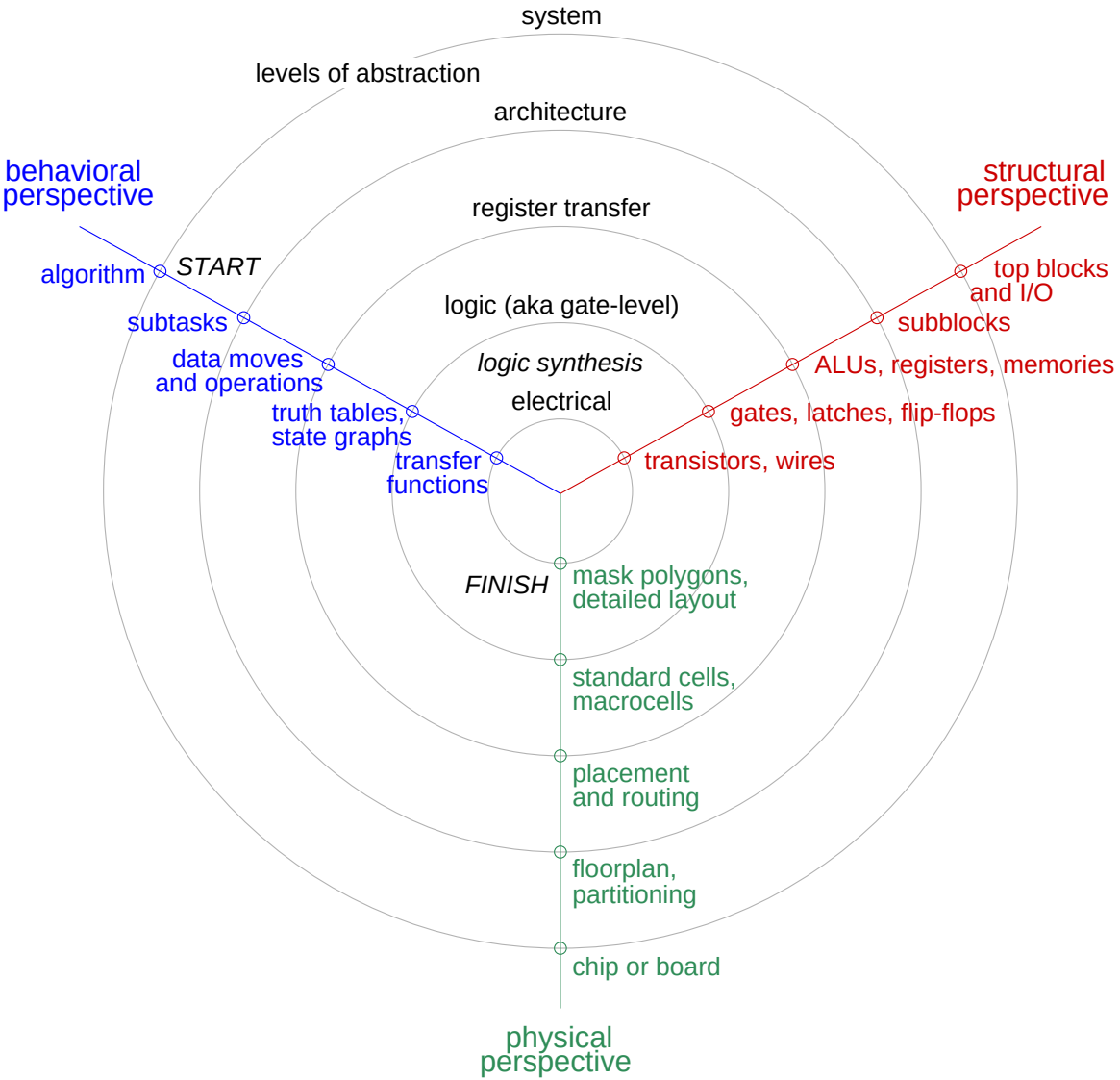
Three factors came together to make fabless operation possible:

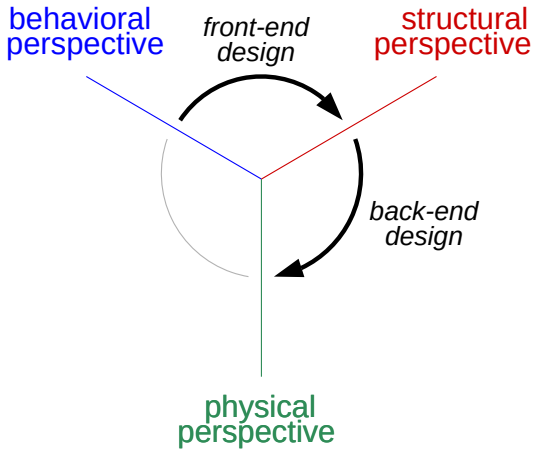
- Generous integration densities at low costs.
- Proliferation of high-performance engineering workstations and EDA software
- Availability of know-how in VLSI design outside IC manufacturing companies. ← *This course wants to contribute here.*

Subject

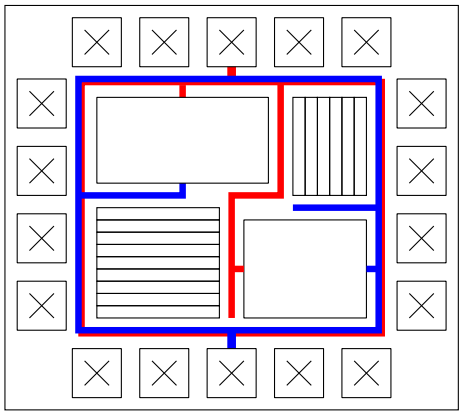
The VLSI design flow

The Y-chart


system
levels of abstraction
architecture
behavioral perspective
structural perspective
register transfer
algorithm
START
top blocks and I/O
logic (aka gate-level)
subtasks
subblocks
data moves and operations
logic synthesis
ALUs, registers, memories
electrical
truth tables, state graphs
gates, latches, flip-flops
transfer functions
transistors, wires
FINISH
mask polygons, detailed layout
standard cells, macrocells
placement and routing
floorplan, partitioning
chip or board
physical perspective


behavioral perspective
front-end design
structural perspective
back-end design
physical perspective

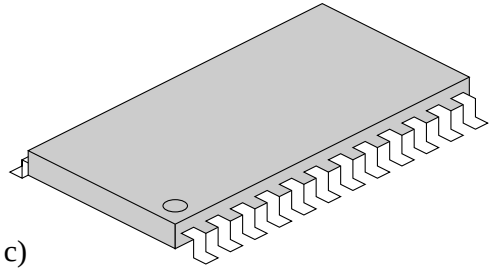
More design views



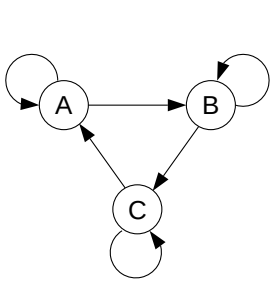
a)

```
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;
```

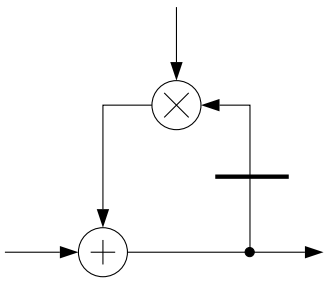
b)



c)

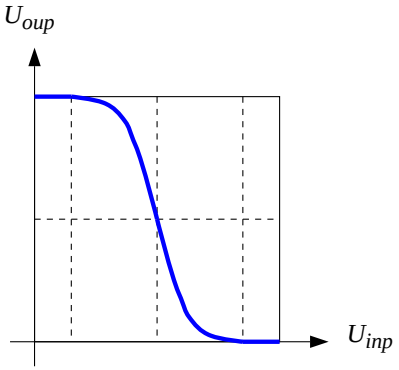


state graph
(models reactive behavior)

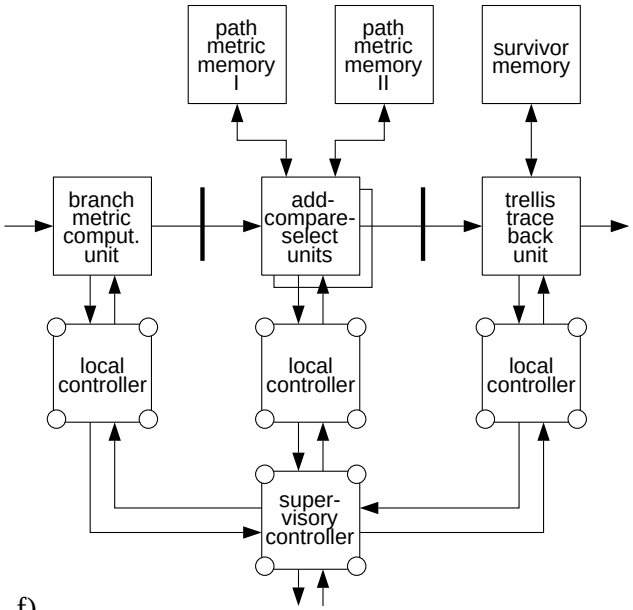


data dependency graph
(models transformatorial behavior)

d)



e)



f)

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

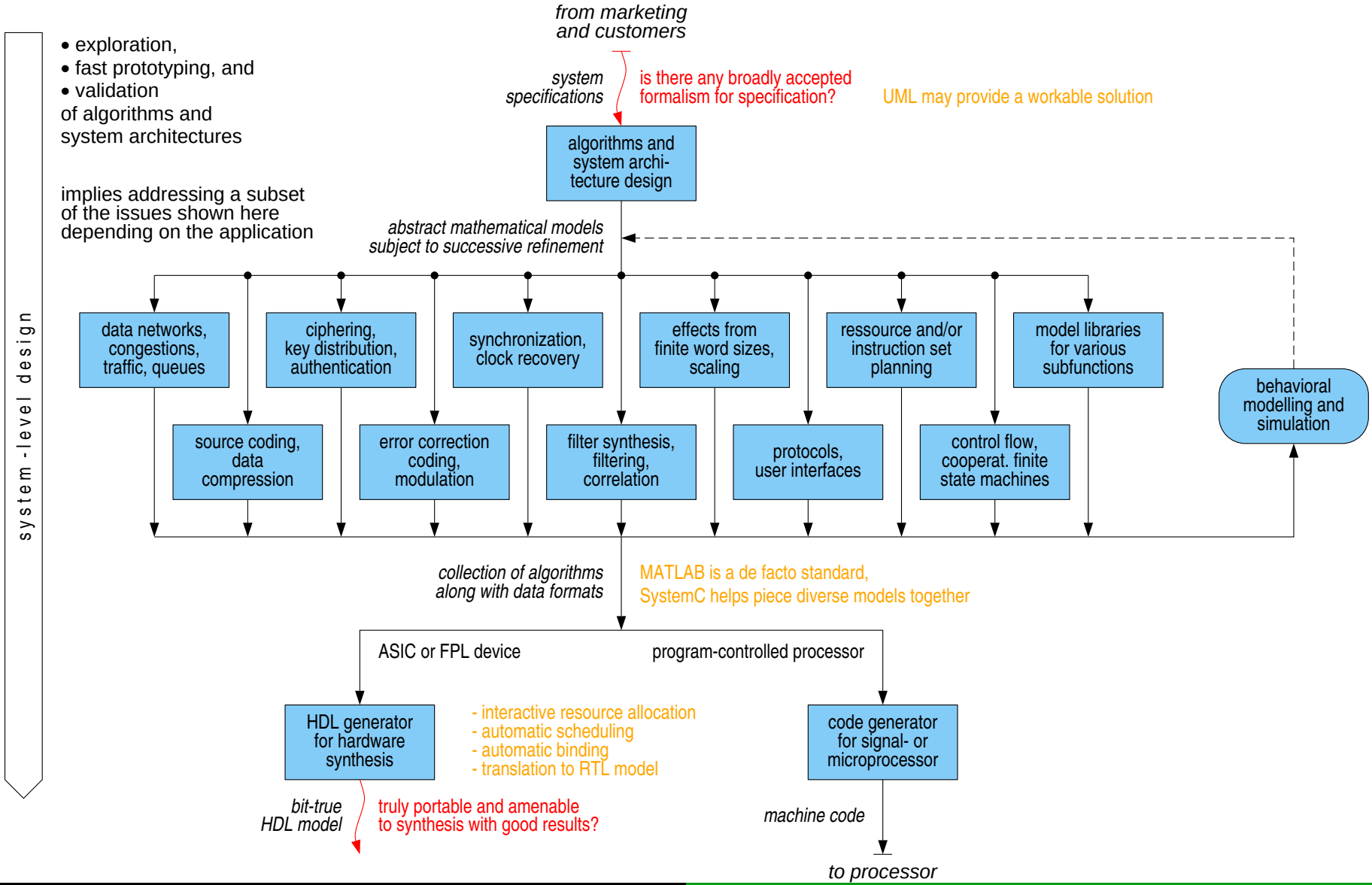
System-level design

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.

Electronic system-level design flow



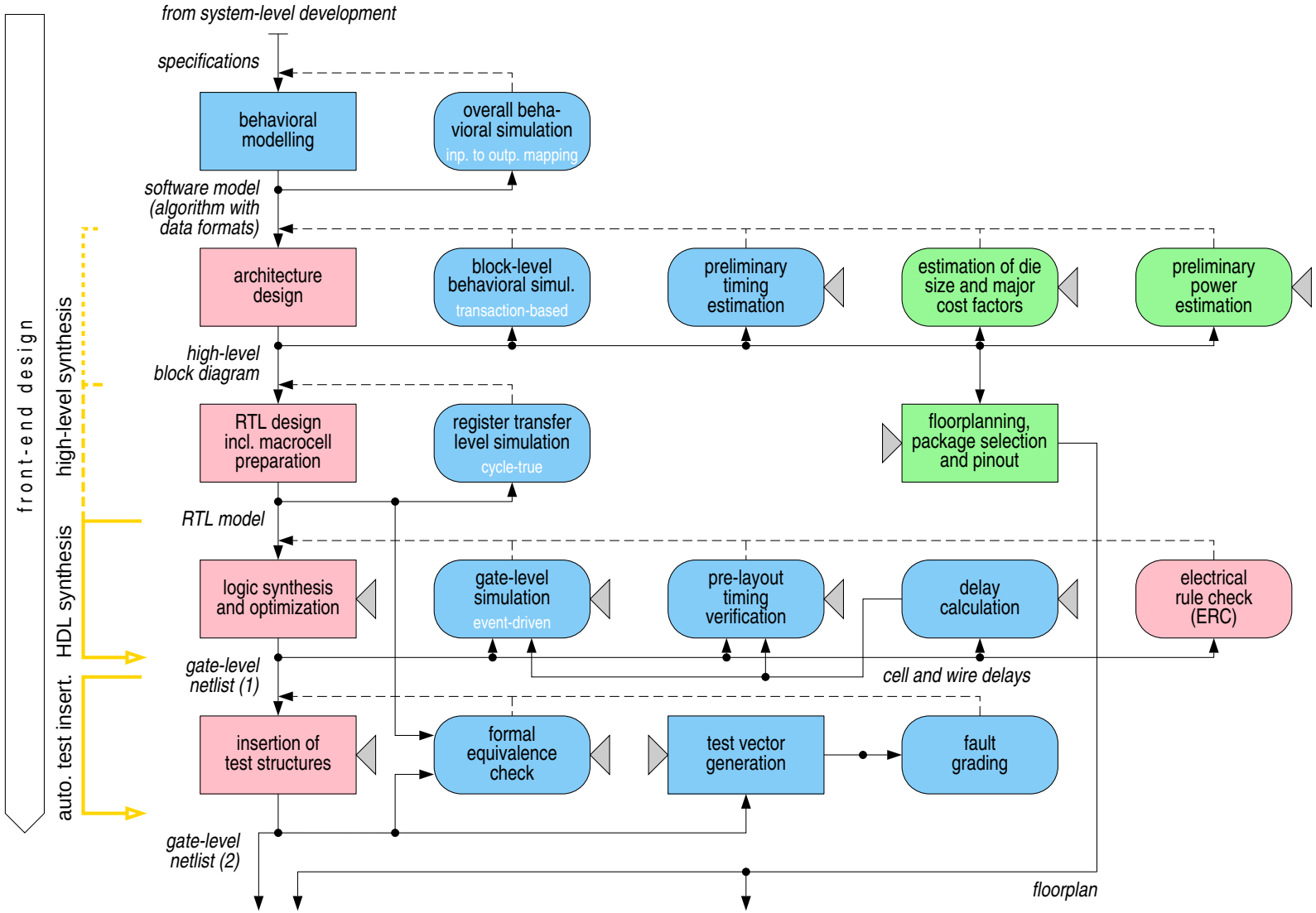
Algorithm design

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. $\mapsto$ 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.

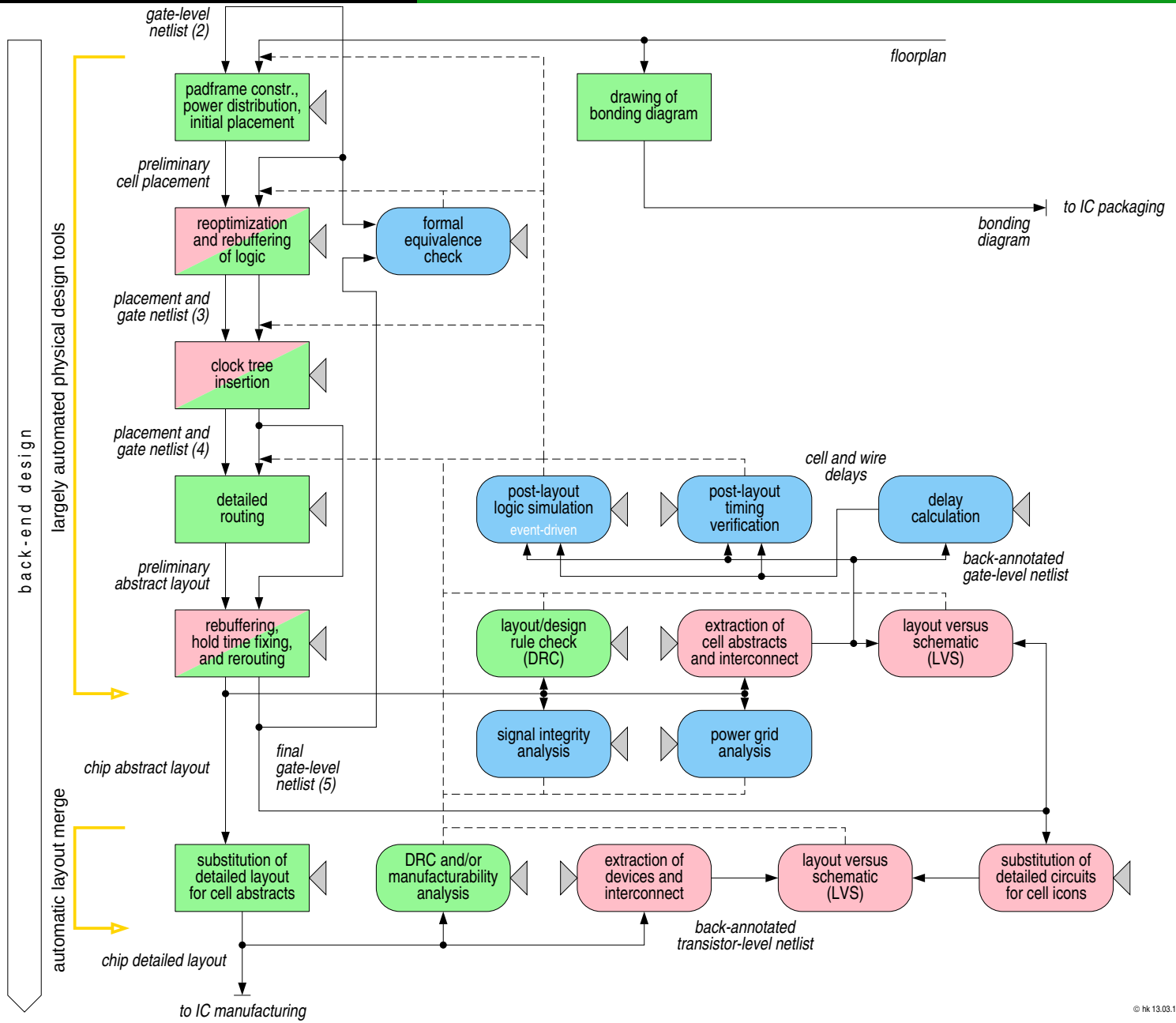
Architecture design II

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? $\mapsto$ scheduling
- What operations to carry out on which processing unit? $\mapsto$ binding
- What clocking discipline to adopt?
- What time interval to use as the basic clock period?
- Where to prefer a bidirectional bus over a unidirectional one?
- How to control the access to a bus with multiple drivers?
- 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
- Intial 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

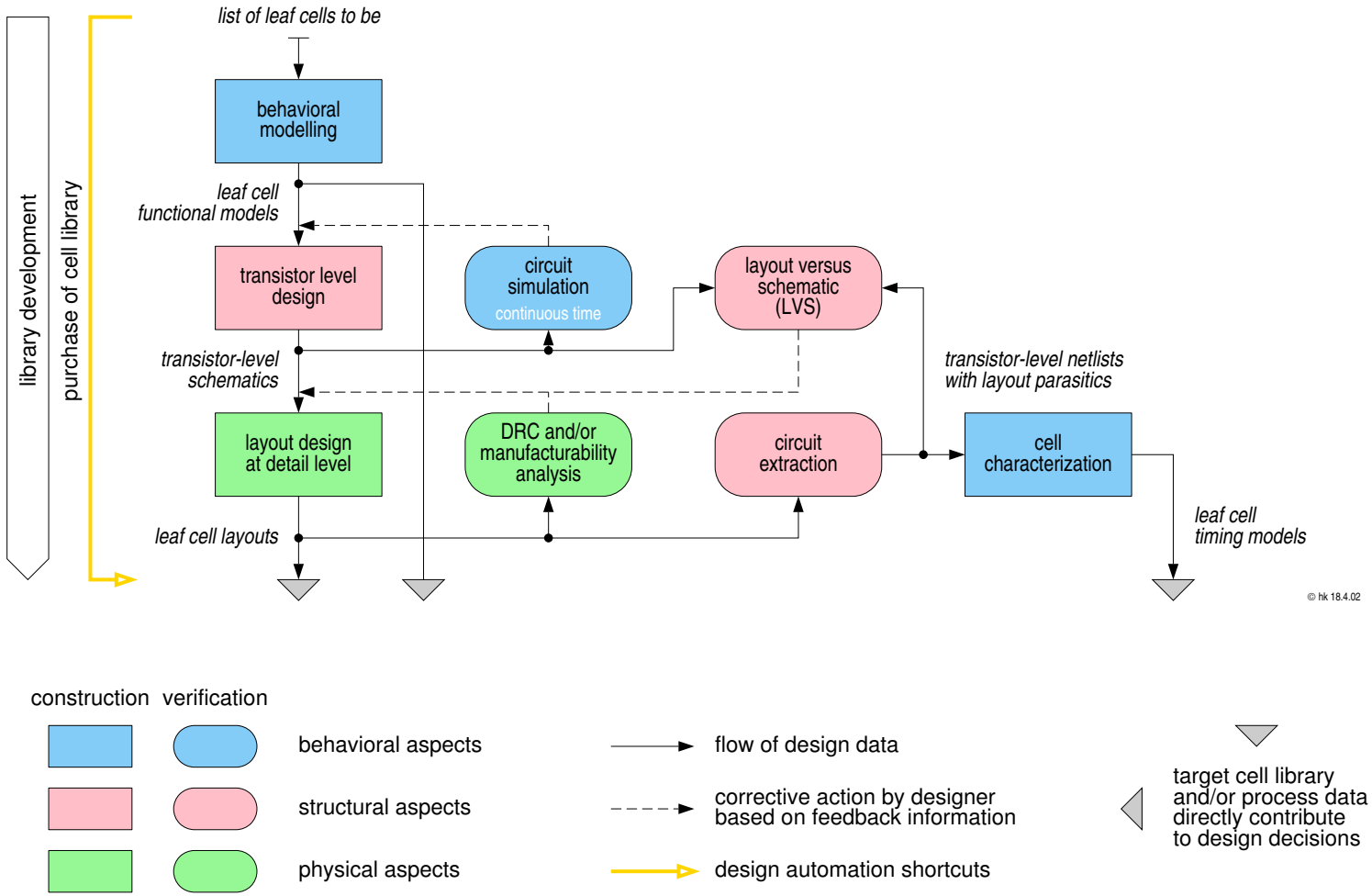
"First time right" design as a guiding principle.

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.



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 (npn or pnp)
NMOS	n-MOS (transistor, circuit, or technology)
PMOS	p-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 BJTs)
ESCL	Enhancement Source-... (similar with MOSFETs)
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.

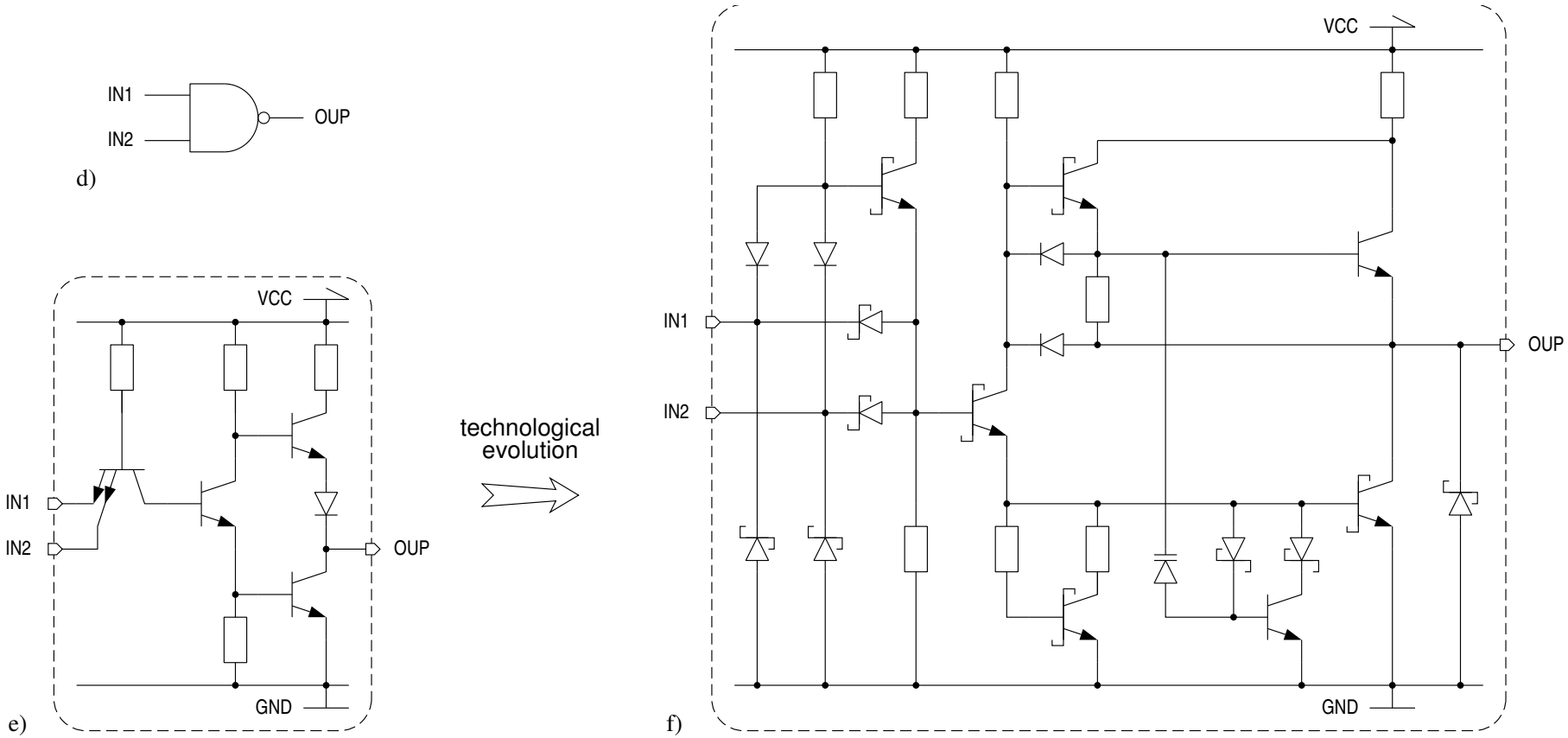
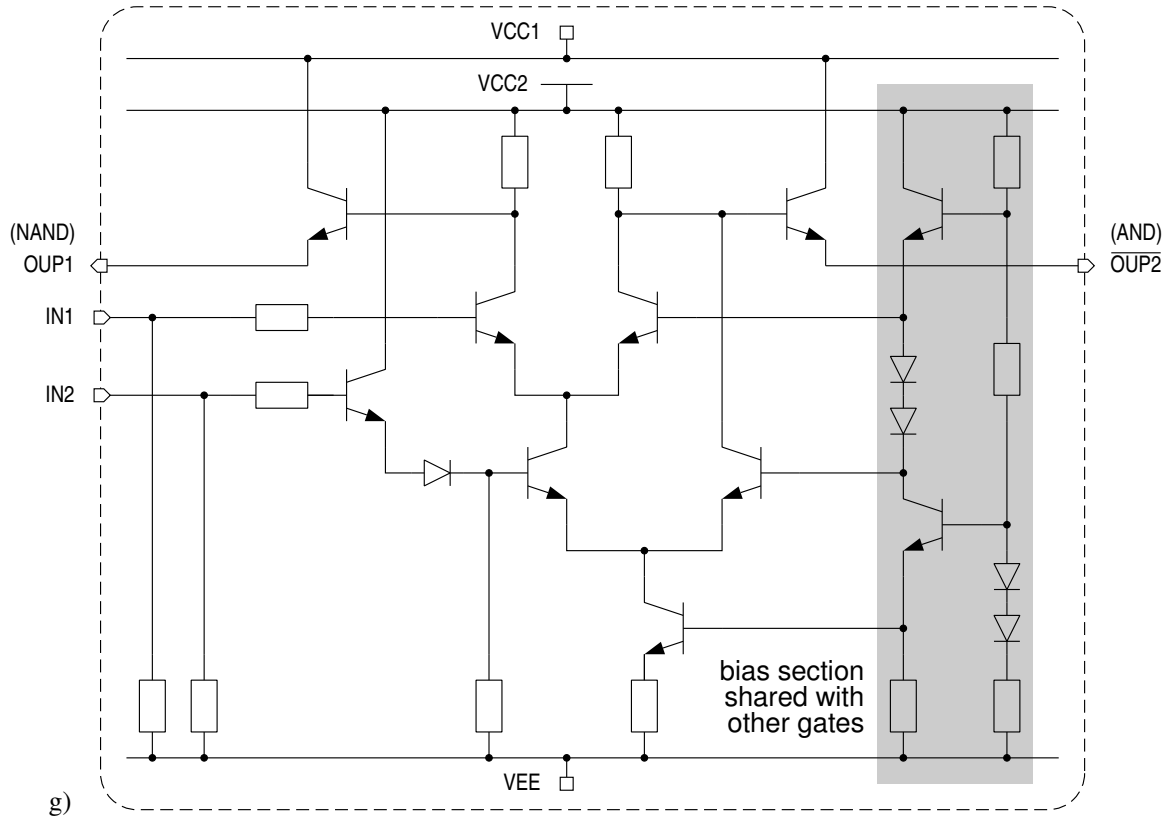


Figure: Icon (d), original multi-emitter circuit (e), and more recent F generation circuit (f). The auxiliary devices serve clamping and speedup purposes.

2-input NAND gate in ECL technology

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



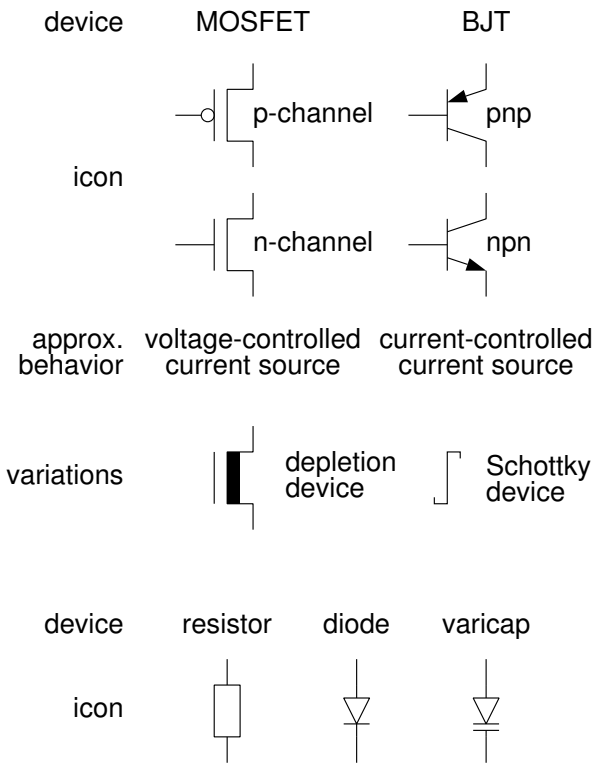


Figure: Circuit (g) with schematic symbols used.
Switching is by current steering without transistors entering saturation.

2-input NAND gate in MOS technologies

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

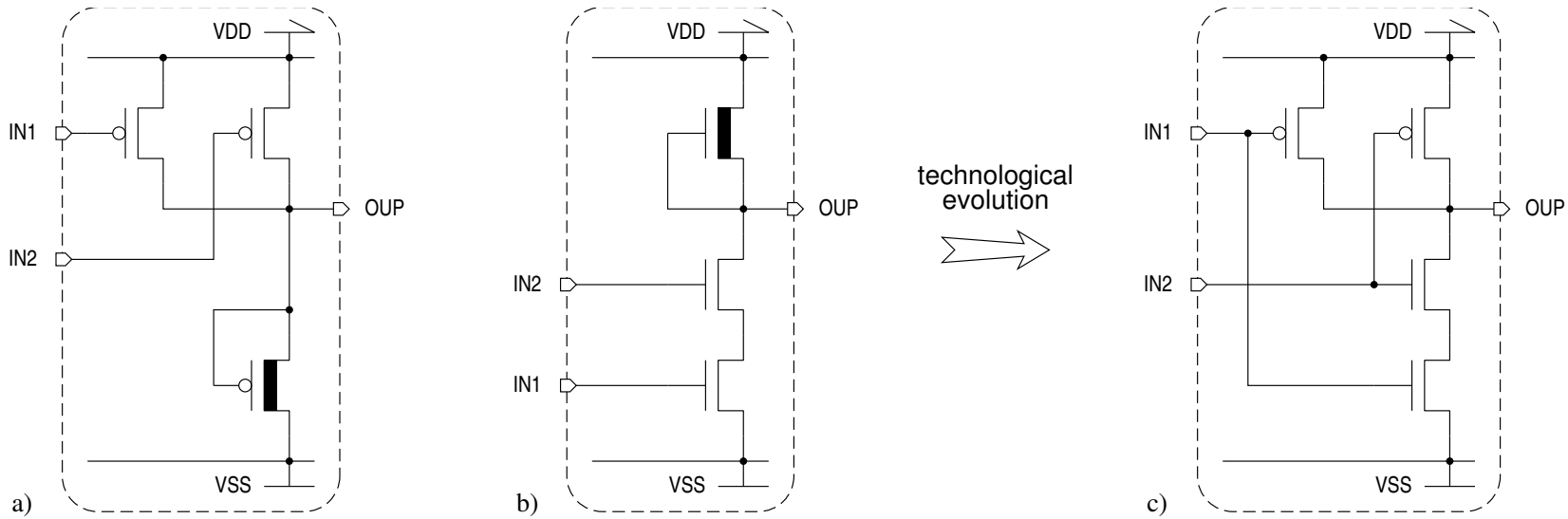


Figure: PMOS (a), NMOS (b), and static CMOS (c) circuits.

Why does CMOS technology dominate VLSI today?

As first observed in 1974 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.

The future prospects for CMOS will be discussed in chapter 18 "Outlook".