Development of Complex Curricula for Molecular Bionics and Infobionics Programs within a consortial* framework**

Consortium leader

PETER PAZMANY CATHOLIC UNIVERSITY

Consortium members

SEMMELWEIS UNIVERSITY, DIALOG CAMPUS PUBLISHER

The Project has been realised with the support of the European Union and has been co-financed by the European Social Fund ***

**Molekuláris bionika és Infobionika Szakok tananyagának komplex fejlesztése konzorciumi keretben

***A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

VLSI Design Methodologies

(VLSI tervezési módszerek)

Digital design flow

(A digitális tervezési folyamat)

PÉTER FÖLDESY

The topics are covered in this chapter:

- Overview
- Design Description
- Logic Simulation
- Logic Synthesis
- Verification
- Gate-level Simulation and Extraction
- Similarity of ASIC and FPGA design flows

Section I

Overview of the digital design, role, position and types of digital systems from designer point of view

Where can we find digital (or better say, binary) leveled circuits?

- Are they really digital from every aspects?
- DRAM
- MPU/ASIC
  - Logic, analog, RF for wireless communication
  - High-voltage for display driver, LCD, OLED, MEMs driver
- Flash
  - NAND, NOR flash memories

System that are digital also in functionality, but multi-leveled at the physical form:

- Signal transmission to enhance throughput (all DSL lines, cable TV)
- Flash memories to compress information to smaller physical volume (up to 8-16 levels, 2-3 bits including error correction).
- In simulation, the circuits are handled usually as ternary systems (low, high, unknown levels).

The digital systems shows a large variety and only a part of them could be handled, from the designer point of view, as pure digital (even the simulation is ternary).

The goal of the digital design flow is to separate different steps and provide simplified, modular view for the designers.

The systems that are designed by Hardware Description Language (HDL) and built from standard cells and IPs are usually called ASIC or ASSP:

- The *ASIC* is an acronym for Application Specific Integrated Circuit
- *ASSP* means Application Specific Standard Product

- Any integrated chip are called ASIC,
  - that contains an entire systems on a single chip
  - not mass product, designed for a specific reason,
  - regardless of the technology or design technique.
- The ASSP is the same for wide markets
  - like MP3 decoder, mobile phone camera ICs, etc.
- Today, the ASIC is usually a digital systems that are built of standard cells and other blocks, which have been standardized by fabrication houses.

Areas of the digital ASIC design

- System design and verification
  - High level synthesis,
  - simulation acceleration,
  - HW/SW codesign, verification IP integration,
  - low-power verification
- Functional verification
  - Equivalence checking,
  - simulation with multi-language, mixed-signal codes,
  - code coverage

Areas of the digital ASIC design

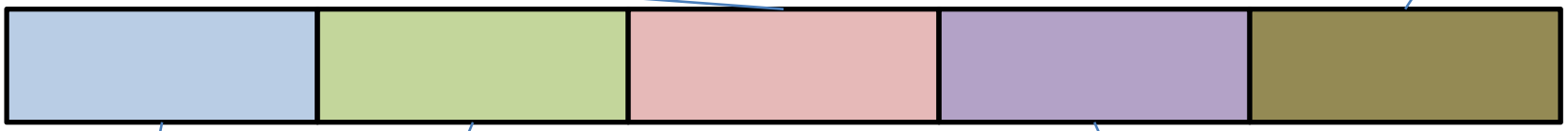
- Logic implementation
  - Chip planning,
  - block implementation,
  - low-power implementation,
  - low level synthesis, placement and route
- Manufacturability sign-off analysis
  - Litho-aware, CMP-aware (chemical-mechanical planarization (CMP) can lead to physical or electrical failures),
  - yield analysis

Section II
Description of digital systems

Description of digital systems

Standardized cell and IP level, consists of useful modules, and used for random logic and moderated speed digital systems description

HW/SW codesign of processor cores and software on it



System level (RTL), expressed in hardware description languages like Verilog, VHDL, system-C

Circuit level, provides very detailed operation for sensitive modules, like high speed ALUs. Also used for characterizing cells that are used later as black boxes with timing, etc. information.

Physical level, technology and material sciences
Provides physical models for simulation

At different levels, different description are used.

Languages:

- Verilog, SystemVerilog, VHDL
- System-C, Handle-C
- C++, ANSI C subsets

Way of coding to perform different tasks (using the above languages):

- To synthesize circuits
- To model and simulate, may be verify other codes
- Describe existing, gate level networks

Important different categories:

- To synthesize circuits we use RTL codes
  - *Register Transfer Level* helps to think about the functionality of the design rather than its implementation.
- To model and simulate: behavioral level
  - Can contain any file operation, implicit timing, delay, symbolic checks, floating point description, mixed analog and digital behavioral
- Describe existing, gate level networks
  - Called *gate level netlist*, barely readable, contains only connecting wire descriptions and gates, IP blocks.

Register transfer level description

- Roughly speaking anything that can be automatically synthesized for standardized cells, blocks, registers and logic.
- The restricted style of the RTL coding allows the synthesis tool to optimize the structure of the design and how it fits in order to create the optimal implementation. The style of coding required for synthesis tools is known as RTL coding.

How the RTL code generated?

- Usually text based, easier to port, modify
- Most simulation environment helps in it
  - Online checking, code coloring, etc.
- There are many template to fill in with actual names
- Graphical design (typically state machines)
- Code generation
  - Automatic testbench generation
  - Matlab, simuling, C/C++ translators (Mentor CatapultC, Cadence), they are better and better

Reasons for the behavioral level

- Its goal to describe operation,
  - the behavior of the system. No reason to be tightly bonded to its real speed, technology, area.
- Higher abstraction level operation
  - Floating point multiplication, division, etc.
  - Objects, structures are described
- Test and verification codes
  - Not correct situations are detectable during simulation
  - Series of signal level checks, Protocol verification
  - Verification of the coverage of the tests

Reasons for the behavioral level

- Algorithmic experiences
  - Similar handling and capabilities like Simulink
  - Experiment of new architectures and simulate them in fast simulators at almost timing true levels.
- Specification form
  - Easier to describe the expected behavior, that will be used later as the golden reference for the differently matured solution.

Gate level description

- Important form of a synthesized design.
- Could be in any low level HDL (historically verilog for ICs, VHDL for FPGAs)
- Its structure is hierarchical, no functions, no tasks, no operators
- Network of instances of logic gates and registers described by a technology library
- Used for post synthesis, post routing simulation, transfer from tool to tool.

Functional verification: simulator types

- Compiled code logic simulators
  - Slow, used for debug
- Event-driven simulators (most of the logic simulators)
  - It evaluates blocks and gates if its input changes
  - Handles timing by setting events in the future to be evaluated
- Native compiled simulators
  - It generates a system native machine code to run
- Emulators using FPGA-s performing the system itself

Importance of constraint driven design

- The applications are usually constrained in many aspects: budget, design time, manpower, timing, size, power consumption
- The technology oriented constraints should be mapped to the design environment. The way to do it is the constraints.
  - Timing constraints, like clock domains, input/output setup and hold times, protocols
  - Area constraints: area of the design, pad limits
  - Power constraints

Section III

Digital design flow overview and steps

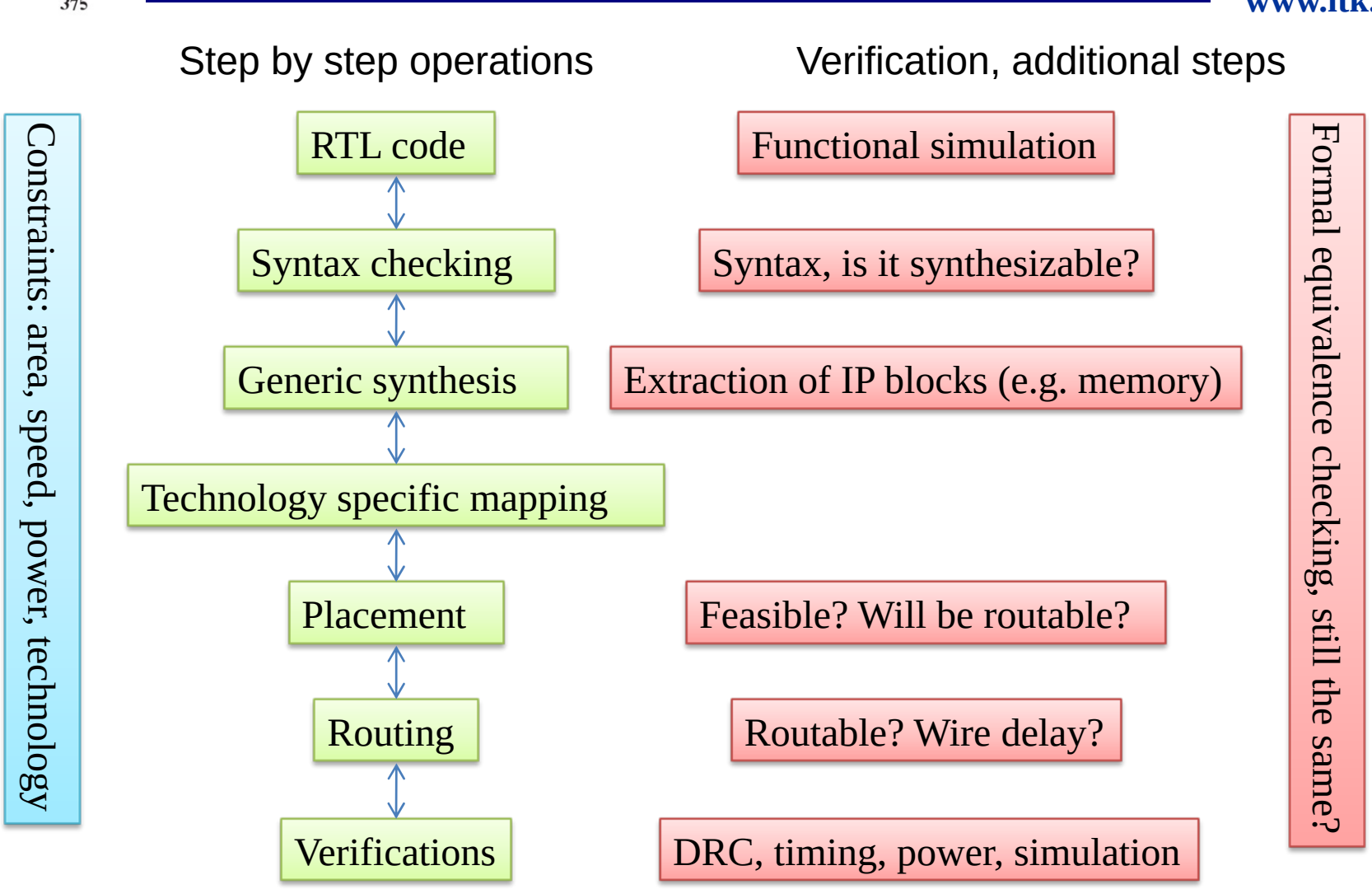
What happens between our chip comes back from manufacturing and the composition of the RTL code?

1. Syntax checking of the code,
2. Filter out non synthesizable parts.
3. Synthesis of code to a metalevel of non-physical gates and IP blocks. This step is called *elaboration* or generic synthesis.

What happens between our chip comes back from manufacturing and the composition of the RTL code?

4. Mapping of the metalevel gates to physical ones
5. Placement of the gates with constraints like area, placement blockades
6. Routing of the gates
7. Verification, physical, connectivity, timing, IR drop, temperature, functional

VLSI Design Methodologies: Digital design flow



Synthesis methods

- Goal of the synthesis:
  - Transition from RTL description into gates, flip-flops, IP blocks (memory, multiplier, pads)
  - Optimization of logic (prune unused paths, registers, merge arithmetical units, insert cloned registers, etc).
  - Placement and routing of optimized netlist creation.
  - Timing constraints aware implementation (if the clock speed is high, the tool creates a larger, but faster adder, multiplier, makes multiplexers parallel instead of serial ones, etc.)

Synthesis methods

- The basis of technology mapping is the standard cell libraries.
- These libraries contains a variety of logic gates, like and, or, not, nand, nor, xor, buffer, complex gates, and sequential elements like latches, flip-flops and memories

Quality Metrics for the synthesis are:

- Area, Timing, Power, Routablity

Placement and route tasks in order:

- Floorplan generation, mostly iterative, including prototype level placement and routing for extraction of early congestion, timing, power problems
- Macroblocks are place first (large memories)
- Padring
- Random logic (gates)
- Routing
  - Power lines (special wires)
  - Clock tree synthesis with inserting buffers, delays.
  - Regular wiring (classic interconnections)
- Verification (DRC, electrical, power, timing, etc.)

Final tasks:

- *Parasitic extraction* and cell and interconnect delay calculation (so called back-annotation) and following functional simulation
- *Formal equivalence checking*: formal comparison of the final logic functionality with the original RTL code
- Design closure: a buzzword for design exploration and timing check for different modes and corners against the constraints
- *Signoff analysis*: final verification of timing, signal integrity, power consumption, statistical static timing, electro-migration, and thermal characteristics

Section IV
Similarity of ASIC and FPGA design flows

Similarity of the ASIC flow to the FPGA flow

ASIC and FPGAs have different value propositions. The design flow is very similar in the two cases, but, there are differences:

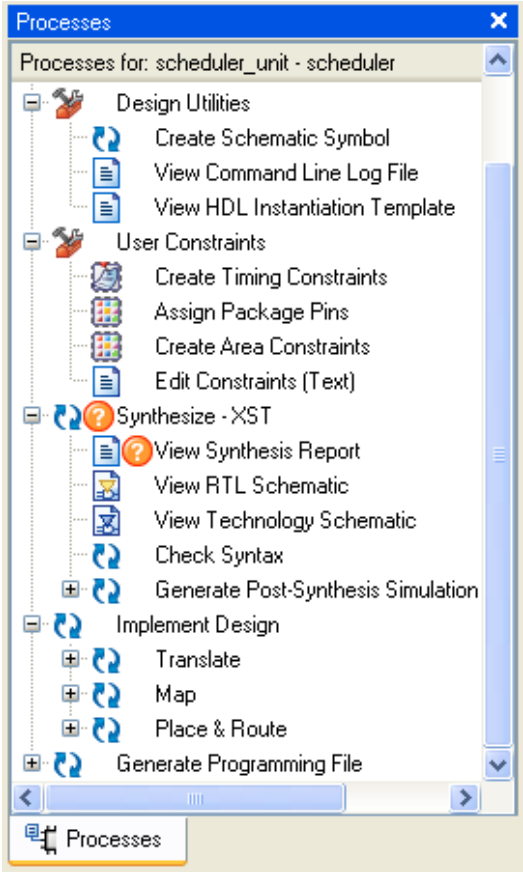
- The FPGA design flow eliminates the complex and time-consuming floorplanning, place and route, timing analysis, and mask / re-spin stages of a design.
- The design logic is already synthesized to be placed onto an already verified, characterized FPGA device.
- Much easier to reach complex IPs on FPGA (if exists).
- An ASIC is more compact, as there is no routing fabric and programming infrastructure.

Comparison of the ASIC flow to the FPGA flow

FPGA Design Advantages	ASIC Design Advantages
Faster time-to-market - no layout, masks or other manufacturing steps are needed	**Full custom capability** - for design since device is manufactured to design specs
No upfront NRE (non recurring expenses) - costs typically associated with an ASIC design, like CAD tools	**Lower unit costs** - for very high volume designs
Simpler design cycle - due to software that handles much of the routing, placement, and timing	**Smaller form factor** - since device is manufactured to design specs, e.g. no routing fabric included
Field reprogrammability - a new bitstream can be uploaded remotely	**Higher raw internal clock speeds**

In the Xilinx ISE (as a typical logic implementation tool) we can find familiar items:

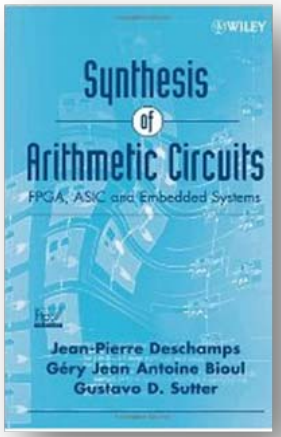
- the HDL creation helpers (e.g. templates)
- Constraints editors, timing and placement
- Synthesis (= generic architecture map)
- Implementation (translate = generic synthesis, map = technology mapping, place and route)
- Instead of mask generation, generation of programming file and download tools



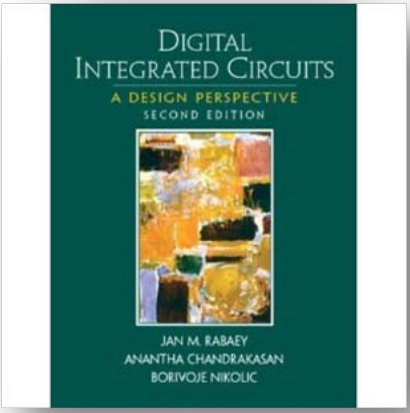
Conclusions

- The digital flow is simple in theory, but for advanced nodes and technologies, becomes extremely complex
- We saw the basic steps:
  - Design,
  - Synthesis to logic gates and connections,
  - Placement and routing
  - Verification both functional and manufacturability
- Worth to remembering that these basic steps requires similar time frame and very different mindset and profession

Recommended literature



Synthesis of Arithmetic Circuits: FPGA, ASIC and Embedded Systems
Jean-Pierre Deschamps, Gery J.A. Bioul, Gustavo D. Sutter
Publisher: Wiley-Interscience (March 10, 2006)



Digital Integrated Circuits (2nd Edition)
Jan M. Rabaey, Anantha Chandrakasan
Publisher: Prentice Hall; 2 edition (January 3, 2003)

Comprehension questions:

I. What are the description methods of digital systems?

II. Why they are used?

III. Describe the digital design flow.