

Hardware-Software-Codesign

Felix Leidl

8. September 2025

Inhaltsverzeichnis

Architectures	2
General-Purpose Processors	2
Reasons for High Performance	2
General-Purpose Processors and Real-Time	3
Multimedia - Extensions and Sub-words	3
Microcontrollers	4
Digital Signal Processors	4
DSP - Arithmetics	5
DSP - Program Development	5
DSP - Trends	6
Application-specific instruction set processors (ASIP)	6
Integrated Circuit	6
Phases of IC Design	6
Design Styles for ICs	7
Comparison of Design Styles	7
FPGA (field programmable gate arrays)	8
FPGA Design Steps	8
FPGA - Application Areas	8

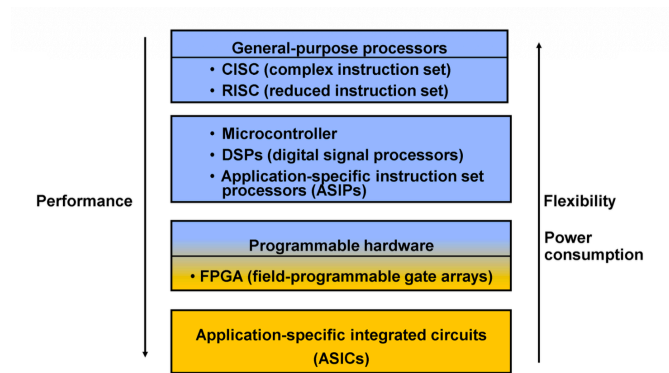


Abbildung 1: Implementation Styles

Architectures

See fig. 1

General-Purpose Processors

- Properties
 - high performance for a broad range of applications, not optimized for any particular application
 - high power consumption
- Design
 - highly optimized circuits
 - design times > 100 person years
 - cost/device only cheap when fabricated in huge numbers
- Application areas
 - PCs
 - Smartphones

Reasons for High Performance

- Exploitation of parallelism
 - multiple scalar units (superscalar)
 - * Functional units: integer units, floating-point unit, load/store-unit
 - * Dynamic scheduling
 - * instruction compatibility

- * complex control unit
- deep instruction pipeline
 - * fetch | decode | read | execute | write back
 - * each scalar unit may have individual pipeline depth
 - * branch prediction
- Multi-level memory hierarchy
 - \uparrow Speed \Rightarrow \downarrow Size \Leftrightarrow \uparrow Size \Rightarrow \downarrow Speed
 - Register < Cache < Main Memory

General-Purpose Processors and Real-Time

- Execution time of programs is not well predictable
 - Dynamic scheduling
 - Caching
 - Branch prediction
- Complex I/O and memory interface

Multimedia - Extensions and Sub-words

- Multimedia Applications
 - 8 / 16 Bit data types
 - many arithmetic operations
 - huge data sets \Rightarrow Huge I/O-bandwidth
 - high data parallelism
- Sub-word execution
 - split 32/64-bit registers and ALUs into smaller sub-units
 - instructions execute in parallel on these sub-units
 - compromise between usage of parallelism and available data paths
 - currently most often based on hand-written assembly routines
 - Depends on language with defined bit width per data type and diverse overflow semantics
 - Compiler needs to automatically detect sub-word parallelism

Microcontrollers

- For control-dominant applications
 - control flow dominant programs (many branches and jumps)
 - few arithmetic operations
 - low throughput requirements
 - multitasking
- Microcontrollers are optimized for
 - Bit- and logic-level operations
 - Registers often realized in RAM: context switch through pointer operation, therefore shortest interrupt latencies
 - integrated peripheral units (e.g. A/D, D/A, CAN, Timer)
- Systems with 4/8 bit processors
- Microcontrollers with high performance
 - 16 - 64 bit processors
 - Application areas
 - * Systems with high control-dominant parts and additional demand
 - * High throughput (telecommunication, automotive)
 - * Computational power (industrial control, signal processing)
 - * As a part of an SOC

Digital Signal Processors

- Signal processing applications
 - data-flow dominant
 - many arithmetic operations, less branches and jumps
 - high degree of parallelism
 - high throughput requirements
- DSPs are optimized for
 - parallel instruction processing (MAC fig. 2)
 - Harvard Architecture, simultaneous access of multiple operands
 - zero-overhead loops
 - special addressing modes (circular, bit-revers)

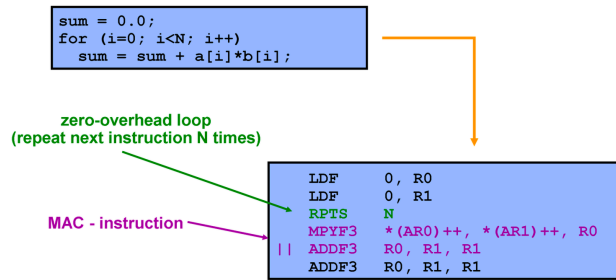


Abbildung 2: Multiply & accumulate

DSP - Arithmetics

- Number formats
 - Mantissa determines precision
 - Exponent determines dynamic range
- Fixed point
 - in case of equal mantissa length smaller (cheaper) and faster as floating-point
 - application design requires consideration of rounding and scaling problems
 - sufficient for many DSP applications
- Floating point
 - high dynamic range of numbers
 - easier application program development

DSP - Program Development

- Assembler, Compiler
 - Many DSP features still only usable if code is written in assembly
 - Programming in C → profiling → time-critical parts in assembly code
- Libraries
 - Programming in C, call of hand-written optimized functions
- Code generation
 - Usage of design environments for modeling, simulation and code generation
 - e.g. Synopsis

DSP - Trends

- Multi-DSP Systems
 - for applications requiring highest performance
 - interfaces available to connect multiple processors
 - Multi-Processor SoC (MPSoC)
- VLIW (Very long instruction word)
 - multiple functional units
 - Compiler detects parallelism and creates a program that schedules multiple instructions jointly
 - Only useful for applications with high degree of instruction parallelism
 - suffers often from low code density
 - compilation difficult

Application-specific instruction set processors (ASIP)

- Specialization
 - instruction set (Operand chaining)
 - functional units (pixel operations, $1/\sqrt{x}$)
 - memory architecture (parallel access onto multiple memory banks)
- Advantages
 - higher performance
 - lower cost
 - smaller code images
 - lower power consumption

Integrated Circuit

Phases of IC Design

1. Design
 - (a) Modeling
 - (b) Synthesis & Optimization
 - (c) Verification
2. Fabrication
 - (a) Masks
 - (b) Wafer
3. Testing

4. Packaging
 - (a) Slicing
 - (b) Packaging

Design Styles for ICs

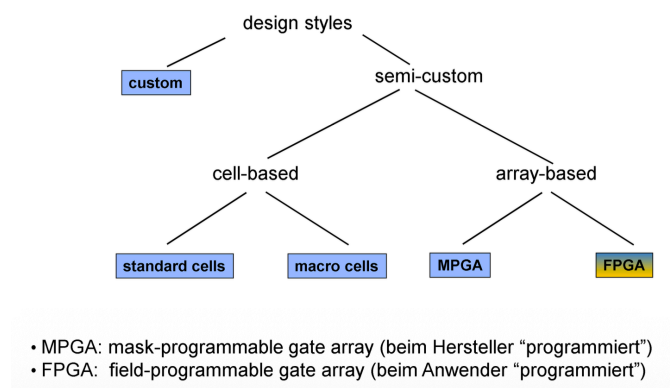


Abbildung 3: Design Styles for ICs

Comparison of Design Styles

	Custom	Cell-based	MPGA	FPGA
Density	very high	high	high	medium-low
Performance	very high	high	high	medium-low
Design time	very long	short	short	very short
Manufacturing time	medium	medium	short	very short
Cost-low volume	very high	high	high	low
Cost-high volume	low	low	low	high

Abbildung 4: Comparison of Design Styles

FPGA (field programmable gate arrays)

- Logic blocks
- I/O-blocks
- Interconnect

FPGA Design Steps

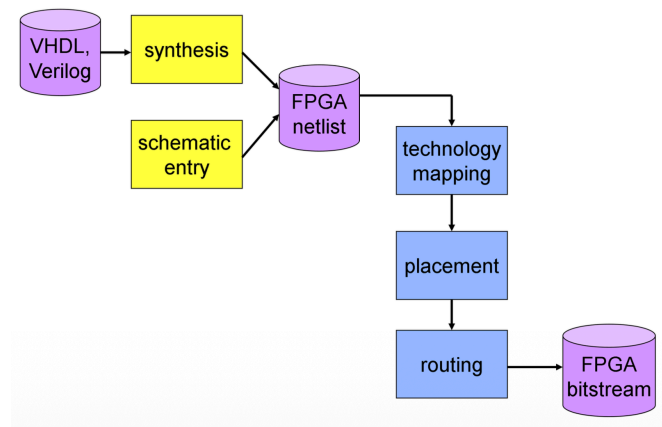


Abbildung 5: FPGA design steps

FPGA - Application Areas

- Glue Logic
- Rapid Prototyping, Emulation
- Embedded Systems
 - when processors are too slow or too power-inefficient and
 - * flexibility is a must
 - * the sale volumes are too low to afford ASIC
- Custom Computing
 - Goal: Combine flexibility of processors with efficient advantages of ASICs
 - e.g. Outsourcing a complex task from SW to HW