# Processor Design
# –
# Processor Design Flow

Professor Jari Nurmi
Institute of Digital and Computer Systems
Tampere University of Technology, Finland
email jari.nurmi@tut.fi

## Processor Design Flow – 1

- Capturing requirements
  - Functional requirements
    - of the application(s)/workload
    - of the operating environment
  - Non-functional requirements
- Functional requirements of the applications
  - Identification of (typical, representative, all?) algorithms to be executed
  - Finding instruction primitives to support these algorithms
  - Profiling of algorithms
    - full algorithms (e.g. GSM half-rate speech codec, MPEG-4, Viterbi)
    - algorithm kernels, inner loops (e.g. codebook search, motion estimation)
    - ”forehead-profiling” of the application
    - pitfall: profiling with an existing processor may be misleading!

## Misleading Profiling?

- E.g. ADPCM encoder - decoder
  - Profiling with old Sun workstation with no support for
    - exponent extraction
    - multibit shifting based on register value

    indicated a huge number of cycles for scaling of results
  - On the other hand, profiling with Analog Devices 2151 DSP with
    - hardware resources for exponent extraction
    - barrel shifter

    yielded completely different results and indicated better the requirements of the algorithm when implemented efficiently
- Profiling a real source code with an existing processor
  - Function of the instruction set of the old processor
  - Function of the compiler ability to optimize for the target

## Processor Design Flow – 2

- Functional requirements of the operating environment
  - Requirements of the (real-time) operating system
  - Requirements of the memory subsystem
  - Requirements of I/O to be attached to the processor
  - Requirements of the software tools
  - Clock cycle, throughput, latency requirements by the environment
- Non-functional issues
  - Cost, power consumption
  - Design tool compatibility
  - SW tool compatibility, backwards compatibility
  - EMC restrictions
  - Manufacturability, testability, maintainability
  - Acceptable failure rate, etc.

## Preparing for the Future?

- Maintaining a certain degree of general-purposity
  - Elementary support for rare operations (not judging them impossible)
  - Elementary support for rare addressing modes
- The old Patterson – Hennessy truth!
  - Make the common case fast
  - Make the rare case correct
- Future requirements
  - There will be more applications than we can think of
  - We should be wizards to foresee all of them!
  - Or leave some room for extensions – or configurability – in the core

## Processor Design Flow – 3

- We know the required
  - operations
  - addressing modes
  - data types
- Now we need to design the actual instruction set and its coding
  - Iterative task
  - Restricted by the cost/performance requirements
  - Has a major effect on the organizational architecture as well
- This is so important that we will devote a complete lecture on this later on!
- Needs hands-on experience ❍ exercises

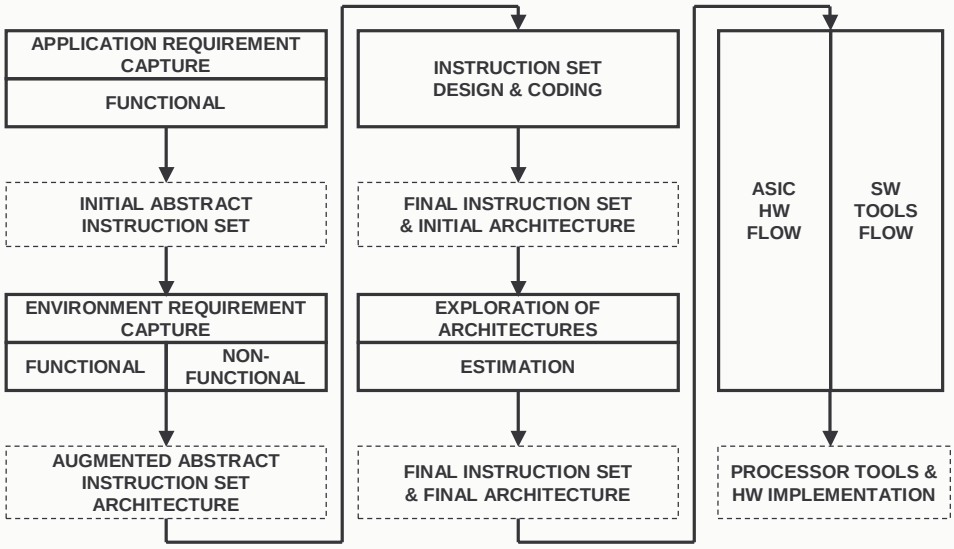
## Processor Design Flow – 4

- Implications of the instruction set to the resources needed
  - Arithmetic and address calculations
  - Data registers
  - Data transfers, buses
  - Control structures and registers
- Exploration of organizational architectures
  - Pen-and-paper methods
  - Spread-sheet calculations of cycle counts etc.
  - Using specification languages WITH automatic performance estimator or simulator generation (strongly recommended)
- Estimation of the foreseen implementation(s)
  - Fitting to the non-functional specification (e.g. cost, power)?
  - Achieving the target cycle time? critical paths? severe bottlenecks?

## Processor Design Flow – 5

- Development of the HW
  - ASIC design flow for the HW
  - Accompanying HW models for developing application systems
- Software tools and libraries
  - Essential for the adoption of the processor
  - Egg – chicken problem (which first, HW or SW?)
  - Ungrateful co-development and co-debugging of
    - program
    - SW tools
    - HW platform
  - Assembler, linker, disassembler, ISS/debugger
  - High-level language compiler (e.g. C/C++)
  - RTOS
  - application examples and libraries

## Processor Design Flow




## End of processor design flow

next we will look at instruction coding issues