Hardware/Software Codesign

SS 2016

Prof. Dr. Christian Plessl

High-Performance IT Systems group
University of Paderborn
Motivating Example

• how to design a "digital TV set top box"

• main functions
  – reception of (possibly encrypted) digital video data
  – decompression of video and forwarding to TV
  – enhancement of image quality
  – storing video to attached hard disk (re-compression)

• auxiliary functions
  – electronic program guide
  – digital teletext
  – multimedia hub for home network (video, music, photos, ...)
  – games
  – downloadable apps
Many Choices for Implementation

- how to chose the processing resources?
  - general purpose CPU
  - specialized processor (micro-controller, DSP, ...)
  - programmable logic (FPGA)
  - dedicated, fixed-function chip (ASIC)

- how to divide the processing among resources?
  - all processing on same resource (powerful CPU)
  - system built from dedicated processors (e.g. micro-controller for control and user interface, ASIC accelerator for decompression and decryption)
  - build a highly integrated system-on-chip

- how to interface the processing resources?
  - point-to-point communication links
  - on-board/on-chip networks
Do Our Choices Matter? Example Decryption (1)

- performance of different target architectures varies widely
- example
  - soft- and hardware implementations of different cryptography algorithms
  - illustrates performance vs. complexity tradeoffs
Do Our Choices Matter? Example Decryption (2)

- the energy efficiency varies even more than performance
- example: energy efficiency of AES encryption for different architectures

![Graph showing efficiency encrypted bits/energy for different platforms]

- software on CPU
  - max flexibility
  - min efficiency
- special purpose hardware
  - min flexibility
  - max efficiency

Schaumont 2012
Most Systems Use a Mix of Target Architectures

- economical and technological reasons prohibit a "one-size-fits-all" solution
  - different user and application requirements (performance, power, cost)
  - cost savings through reuse of hard- and software implementations
  - need to amortize development cost over a variety of products

- most computer systems use a variety of HW and SW technologies
  - programmable processing cores (RISC, DSP, microcontroller, …)
  - fixed co-processors/accelerators (e.g., ASIC for cryptography, video, …)
  - reconfigurable accelerators (FPGA, coarse-grained arrays)
  - integration of several technologies in system-on-chip (SoC)

- functional partitioning depending on requirements
  - fixed functions that are performance or power critical → specialized accelerators
  - variable or non-critical functions → programmable processors
Example: SoC for Mobile Computer System

TI OMAP 4430 system-on-chip
Example: SoC for Mobile Computer System

TI OMAP 4430 system-on-chip
HW/SW Codesign-related Questions

- **hardware platform**
  - what processing elements are available?
  - how to chose the ones that satisfy performance, cost, power budget, ... ?
  - what is the communication demand between components?

- **software/application**
  - how to decompose the application into different parts?
  - what parts shall be executed on which resource?

- **tool support**
  - how to build automated tools to support the system design process?
  - how to build development tools for generating software and hardware?

The goal of HW/SW codesign is to develop methods and tools to solve these questions in an integrated, systematic and computer supported way.
General Approach in HW/SW Codesign

- **abstract and model**
  - treat design of HW/SW system as optimization problem
  - abstract and formalize problem and design options (e.g., using graph problems, integer linear programming, dynamic programming, …)

- **then optimize …**
  - exact or heuristic methods
  - determine optimal parameter settings and their sensitivity

- **… or explore**
  - explore design space and trade-offs
  - computer aided design automation guided by human decisions
Example: System Design

- Application specification
- System synthesis
- Exploration, optimization
- Estimation

- SW-synthesis
- Machine code

- I/F-synthesis

- HW-synthesis
- Net lists
generally speaking: the synthesis process transforms a behavioral description into a structural description

synthesis is comprised of 3 tasks

- allocation: select components
- binding: assign functions to components
- scheduling: determine execution order

in the following: illustration of these steps in system-level design
Application Specification at the System Level

task graph

constraints

< 200 ms
Allocation at the System Level

- components to be allocated
  - processors, dedicated hardware
  - memory, I/O
  - interconnection structures
Binding at the System Level

A

B

C

D

E

MIPS

DSP

ASIC

Mem

Mem
Scheduling on the System Level

- **A** (DSP)
- **B** (MIPS)
- **C** (MIPS)
- **D** (ASIC)
- **E** (ASIC)

Diagram showing the scheduling of tasks on different hardware components over time.
Scheduling on the System Level

alternative schedule, reduces execution time
1, 2, 4, 6 are Pareto points

Exercise 1.1: Design Space, Pareto Points
Course Synopsis

• introduction
  – target architectures
  – introduction to compilers

• high-level hardware architecture synthesis

• partitioning
  – hardware/software partitioning
  – system partitioning

• design space exploration
  – evolutionary algorithms
  – estimation of design parameters

• optional topics
  – instruction set extension
  – emulation
  – case studies
Relation to Other Lectures in CS/CE Curriculum

• Bachelor level courses
  – Foundations of Computer Architecture/ Engineering (GRA/GTI)
    ▪ fundamentals of digital logic and computer architecture
  – Embedded processors
    ▪ architecture of embedded processors, micro controllers and DSPs
    ▪ compilers for special purpose processors

• Master level courses
  – Reconfigurable Computing
    ▪ FPGA technology (architecture, methods and applications)
  – Advanced Computer Architecture
    ▪ memory hierarchy design, instruction level parallelism, data-level parallelism, thread-level parallelism
  – Architektur Paralleler Rechnersysteme
    ▪ parallel programming models and languages, high-performance computing
Benefits From Attending This Course

• learn about
  – challenges and approaches in modern system design
  – target architectures
  – useful optimization methods
  – a current and active research area

• prerequisites, fundamentals of:
  – digital design (combinational and sequential logic, finite state machines)
  – computer architecture (CPU architecture, instruction sets)
  – graphs
  – C programming language
Course Organization

• lecture & exercises
  – Room O1.258
  – Exercises are integrated in lecture, your participation is expected
  – Additional programming exercises

• contact: Christian Plessl
  email: christian.plessl@uni-paderborn.de
  office: O2.167, phone: 60-5399

• web page
  http://homepages.uni-paderborn.de/plessl/teaching/2016-Codesign

• course Materials on the web
  – lecture slides, exercise sheets, selected papers
• this course is based on materials from the following books…

• … and on lecture materials from the Bachelor-level lecture “HW/SW Codesign”
  – taught by Christian Plessl (2009-2010) and previously by Marco Platzner
Changes

- V2.2.0 (2016-04-08)
  - updated for SS2016
- v2.1.1 (2015-04-16)
  - updated after 1st lecture with concrete details of organization
Changes

• v2.1.0 (2015-04-13)
  – updated for SS2015, cosmetics
• v2.0.2 (2014-04-09)
  – updated for SS2014, minor additions
• v2.0.1 (2013-04-10)
  – cosmetics and minimal additions
• v2.0.0 (2013-04-10)
  – completely revised and updated for SS2013
• v1.0.3 (2012-04-14)
  – corrected minor typos
• v1.0.2 (2012-04-10)
  – updated for SS2012
• v1.0.1 (2011-04-01)
  – added more slides illustrating the FPGA toolflow