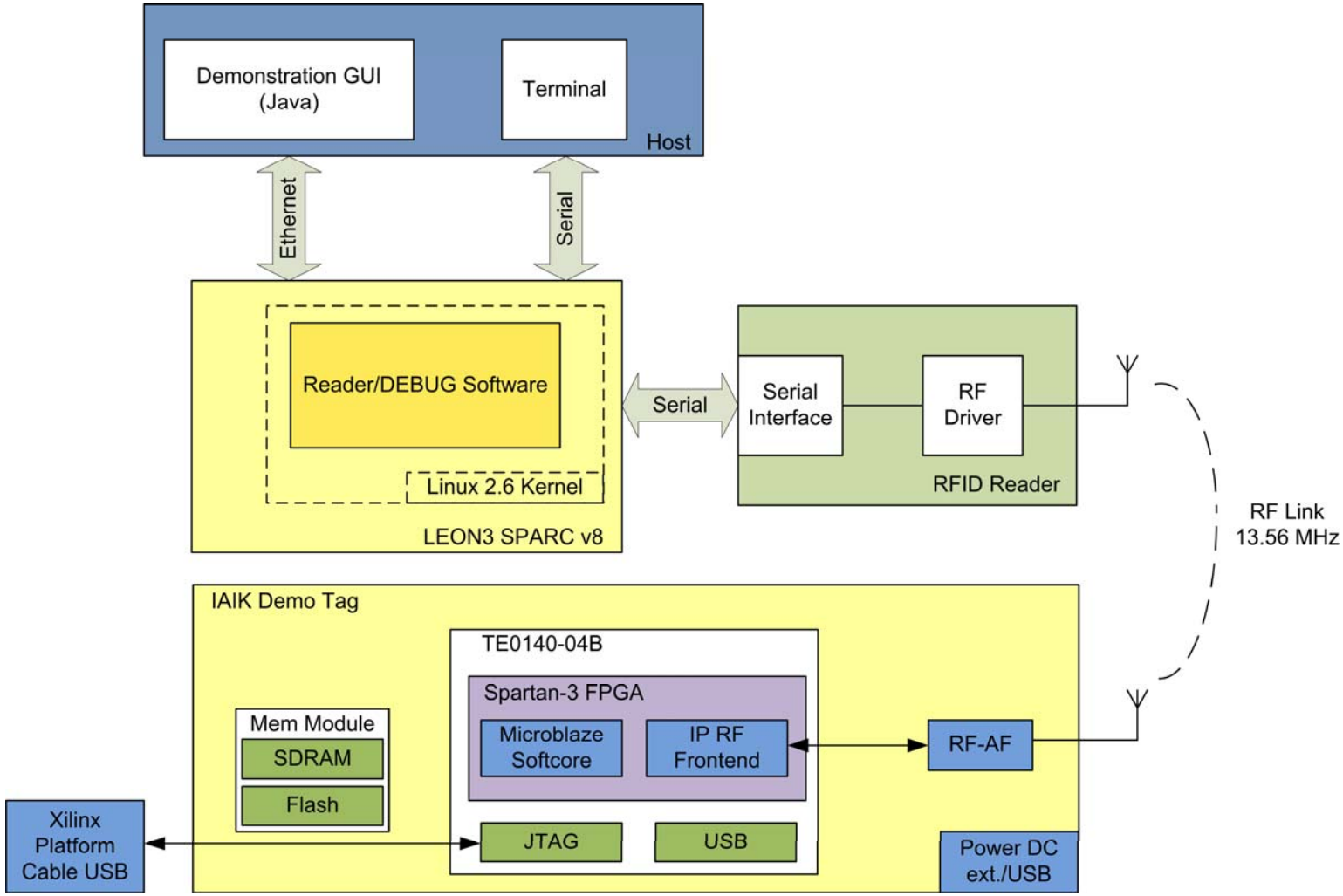
System on Chip (SoC) Architectures and Modelling 2010

Michael Hutter
Alexander Szekely
Walter Bell
Krassimir Duschkov
Johannes Eichler
Michael Hemetsberger
Michael Langreiter
Arnold Loidl
Daniel Mandler
Alexander Melzer
Thomas Raggl
Thomas Tanzer
Viktor Unterberger
Georg Wagner

The SoC Project 2010



Human Resources

- 1 Project manager: Michael Hutter
- 1 External consultant: Alexander Szekely
- 12 Students

- Divided the work into 4 Work Packages
  - Each with 3 students

Work Packages

- WP 0
  - Linux on Demotag
- WP 1
  - HF Driver for Demotag
- WP 2
  - Linux booting, communication with flash
  - Leon + Demotag
- WP 3
  - Linux on Power–Trust Leon3

WP 0

“Linux on DemoTag”

Team Members

- Walter Bell
- Michael Langreiter
- Georg Wagner

Outline

- Work package goals
- Hardware description
- Problems & solutions
- Final project status of WP0

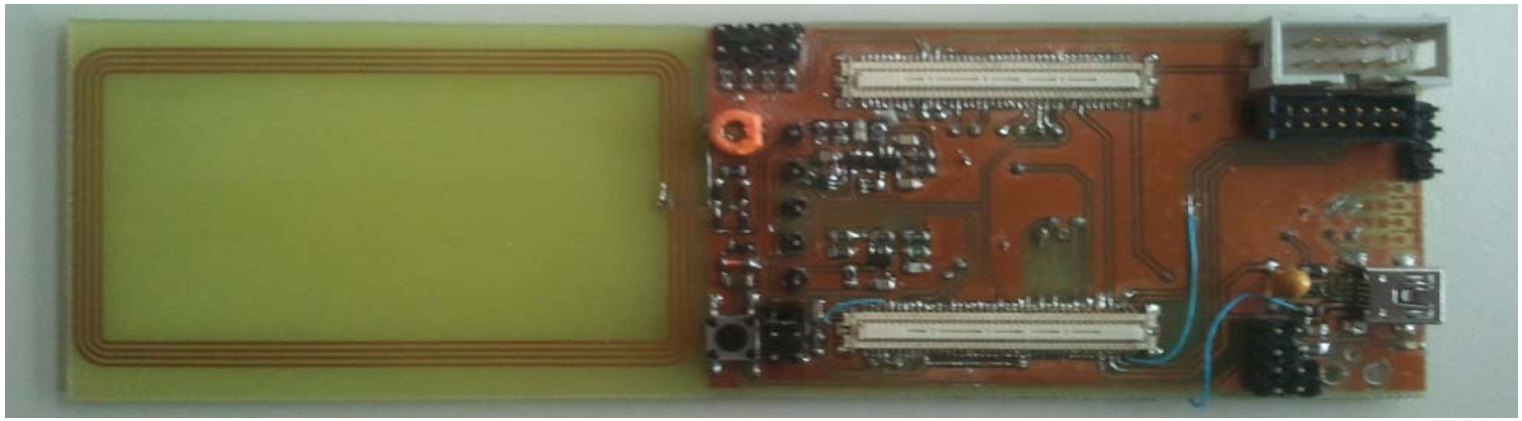
Work Package Goals

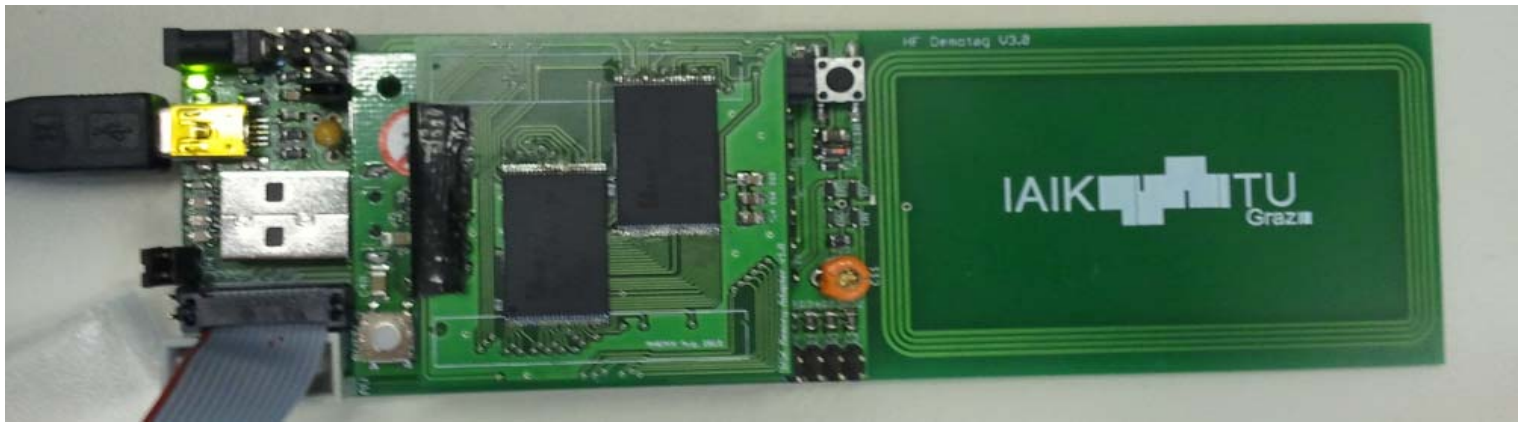
- Running Linux on the DemoTag
- Get SDRAM running
  - Memory test program
- Synthesize Microblaze system
  - Multi Port Memory Controller (MPMC)
  - Bus system
  - Clock generator
  - Serial port

Hardware Description (1)

- IAIK DemoTag
- Trenz Elektronic Spartan-3 1000K Mikromodule
- IAIK Memory Module
  - SDRAM
  - Flash

Hardware Description (2)





Problems

- Synthesize Microblaze system
  - ISE-Flow not working
  - Configuring the system to run C programs
  - „Segmentation faults“ during synthesis

Problems

- Get SDRAM running
- @20MHz: about 50% errors
- @18MHz: about 5% errors

- Problem of lowering the frequency beneath 18MHz
  - Clock generator does not support clocks under 18 MHz

Solution

- Clock Divider
  - Self written clock divider module
  - Input frequency of 120MHz
  - Output clocks 7,5MHz 0˚/90˚ phase shift

Problems

- Running Linux on the DemoTag
  - Toolchain
  - Configuration of Linux
  - Adapting devicetree
  - How to get Linux running on the Spartan-3
  - Boottime: about 3 minutes

- Linux is not booting from flash memory (WP2)

Solution

- Downloading Toolchain
  - Crosscompiler
- Configuring Linux Kernel
  - Removing all unused components
  - Syncing the configuration of the processor
- Devicetree
  - Exporting devicetree from XPS

Final Project Status

- Running Linux on the DemoTag ✓
  - Running on SDRAM using RAMFS
- Get SDRAM running ✓
  - SDRAM works at a frequency of 7,5MHz
  - Higher frequencies tested (not working/working when its cold)
- Synthesize Microblaze system ✓

WP 1

“Communication HF Reader ↔ DemoTag”

Team Members

- Johannes Eichler
- Michael Hemetsberger
- Daniel Mandler

Outline

- Work package goals
- Hardware description
- Framing Logic
- PLB to AMBA bridge
- Linux driver and software application
- Final project status of WP1

Work Package Goals

- Synthesize Framing Logic
  - Adapt IP module according to the requirements
- Synthesize Microblaze system
  - Connect PLB to custom IP core
  - Add GPIO module
- Linux Driver to write to PLB
- Software application to generate random UIDs

Framing Logic (I)

- IP module designed by Thomas Plos (IAIK)
- Get familiar with IP module
  - Simulation to verify functionality
  - Which signals are needed
- Adapt Framing Logic regarding the requirements
  - Import IP module to ISE
  - Change anticollision sequence for 4 byte UIDs
  - Connect AMBA interface to top module

Framing Logic (II)

- Create corresponding *ucf* file
- Testing generated bitfile on DemoTag
  - Problems with defect analog front-end
  - Problems with malfunctioning switches
  - Problems with defect JTAG interface

PLB to AMBA Bridge (I)

- Synthesize Microblaze system
  - Problems with project files of WP0
- Automatically generated PLB slave
  - State machine to translate PLB commands to AMBA
- Connecting Framing Logic to Microblaze
  - Done by PLB to AMBA bridge
- Testing communication with Framing Logic
  - C program to write to PLB

PLB to AMBA Bridge (II)

- Debugging PLB to AMBA bridge
  - Problems with byte order
  - Problems with state transition

Linux Driver (I)

- General Description
  - Character Driver
  - Device file /dev/framing_logic
  - User application uses:
    - open()
    - close()
    - write()
    - read() – implemented but not needed
  - Memory mapped I/O

Software Application

- Generates a random UID, when DemoTag gets in the field of a HF reader
- Polling of the carrier signal on the analog front-end
  - Done by reading from a GPIO module connected to the PLB
- Working with interrupts would be a better solution but couldn’t be achieved so far

Final Project Status

- Working implementation of Framing Logic ✓
  - All commands of ISO14443 layer 3 can be performed
- Working interface between Microblaze and Framing Logic ✓
  - Implemented as PLB to AMBA bridge
- Working Linux driver for communication with Framing Logic ✓

WP 2

“Linux Booting and Flash Communication”

Team Members

- Krassimir Duschkov
- Thomas Raggl
- Arnold Loidl

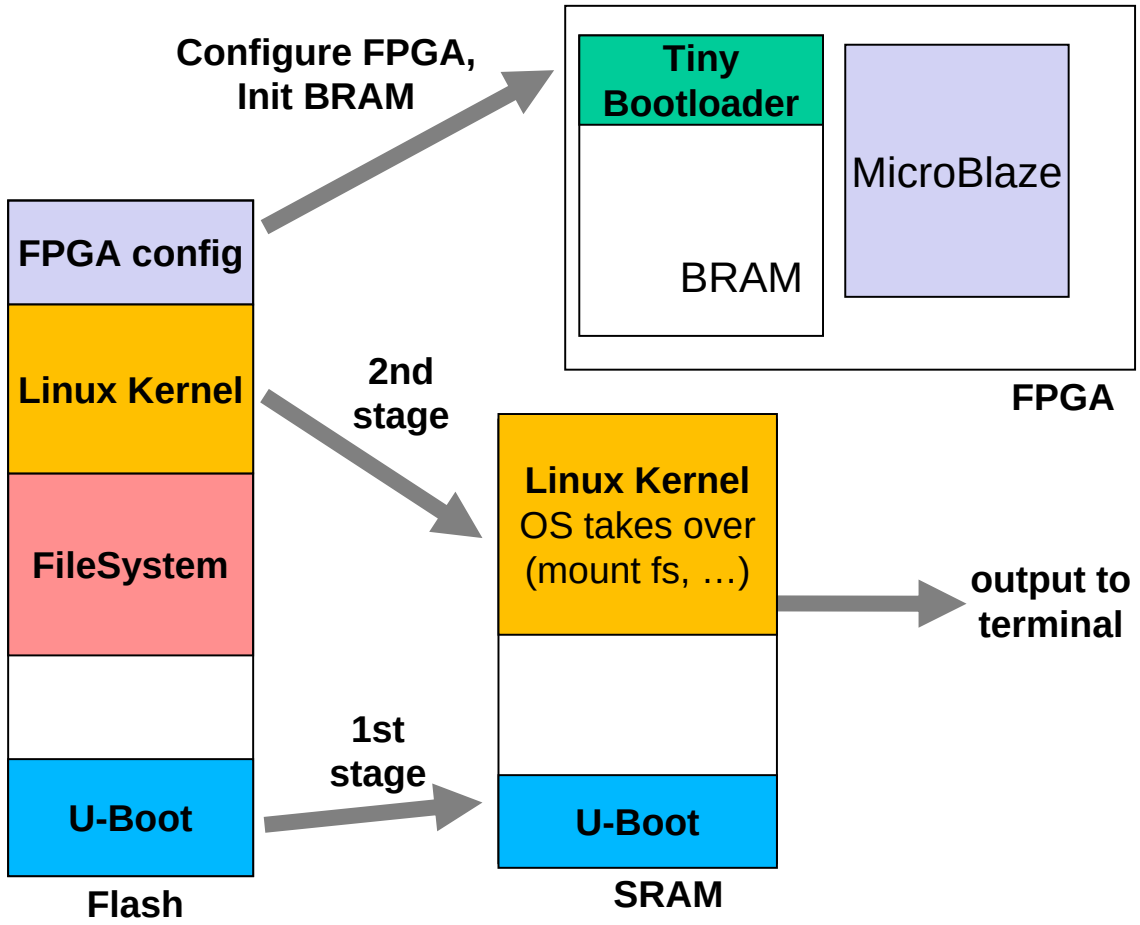
Work Package Goals

- In general: „Booting of Embedded Linux“
  - A real-world applicable booting mechanism for both target platforms
  - Provide Flash-memory communication
- Configure U-boot for both boards
  - Adapt driver for Flash memory
  - Boot Linux from Flash
- Booting strategies for Linux

Booting strategies

Spartan3-FPGA platform

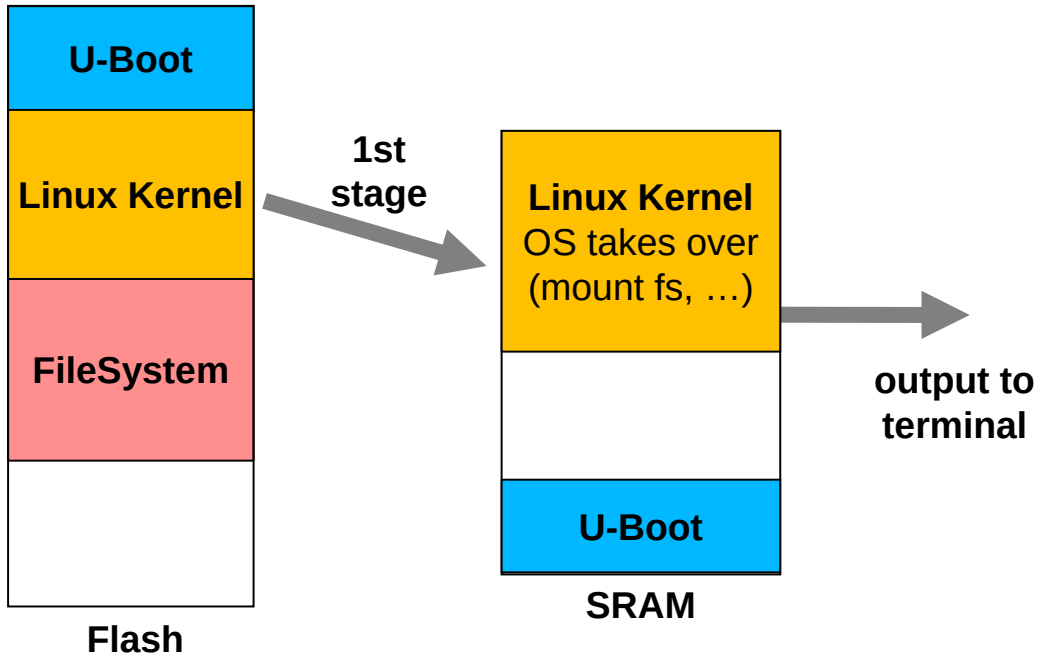
- Two stage boot process
- BRAM is too small for U-Boot
- Tiny Bootloader starts U-Boot
- U-Boot starts linux



Booting strategies

Leon3-ASIC platform

- Only one stage needed
- Leon starts from the flash
- U-Boot copies itself in SRAM
- U-Boot load the Linux kernel-image
- The kernel mounts the root file-system



Spartan3 FPGA Board (DemoTag)

Preliminary Work

- ML403 Virtex4 FPGA Evaluation Platform (Xilinx)
- Xilinx Toolchain (XPS, ISE, iMPACT)
- Setup preliminary target platform (Microblaze based system)
- Simple booting from Flash
- Automatic FPGA configuration from Flash
- Setup U-Boot for Microblaze
  - Microblaze GNU Tools
  - Building U-Boot (BSP generation)
  - Building OSL Linux for Microblaze (Device Tree Generation)
  - Two stage booting mechanism

Spartan3 FPGA Board (DemoTag)

Flash Memory

- System integration of Flash memory (IAIK memory module)
  - *A29L320AUV-70F* (8MiB)
  - No boundary scan chain feature
- First approach
  - Two separate memory controller (SDRAM / Flash)
  - Additional external memory ports
  - UCF adaptation (mapping)
- Second approach
  - Integration of a custom memory multiplexer (VHDL)
  - Switching of shared address and data lines
- Third approach
  - New memory controller supporting both Flash and SDRAM
  - Wishbone bus standard interface (PLB-WB Bridge, WB-Arbiter)

Problems and Solutions

- Flash memory not in boundary scan chain
  - Preliminary work on ML403 wasted time
  - XPS approach not applicable (Kermit)
- Integration of custom multiplexer failed
  - SDRAM controller needs direct connection to external ports
  - No synthesis possible
- Open source memory controller
  - Integration in XPS environment (Wishbone Bus) succeeded
  - PLB-WB bridge synthesis failed
  - Controller needs additional driver (U-Boot, Linux)
  - Very poor documentation
  - Implementation not feasible (time and effort)

Leon3 Board (u-boot)

Configure u-boot to run on Leon3 board

Configuration files from other (Gaisler-) Board modified for our purposes

Driver had to be upgraded to detect and program the flash-memory chips on the board

An u-boot images configured to run in SRAM is able to program an other image, that runs from flash

Leon3 Board (Linux)

Kernel + root filesystem (rootfs) to big for SRAM

- rootfs has to be mounted from flash-memory

Build an appropriate kernel

- Only required things are added to an empty kernel
- Flash-driver had also to be upgraded to detect and program the flash-memory

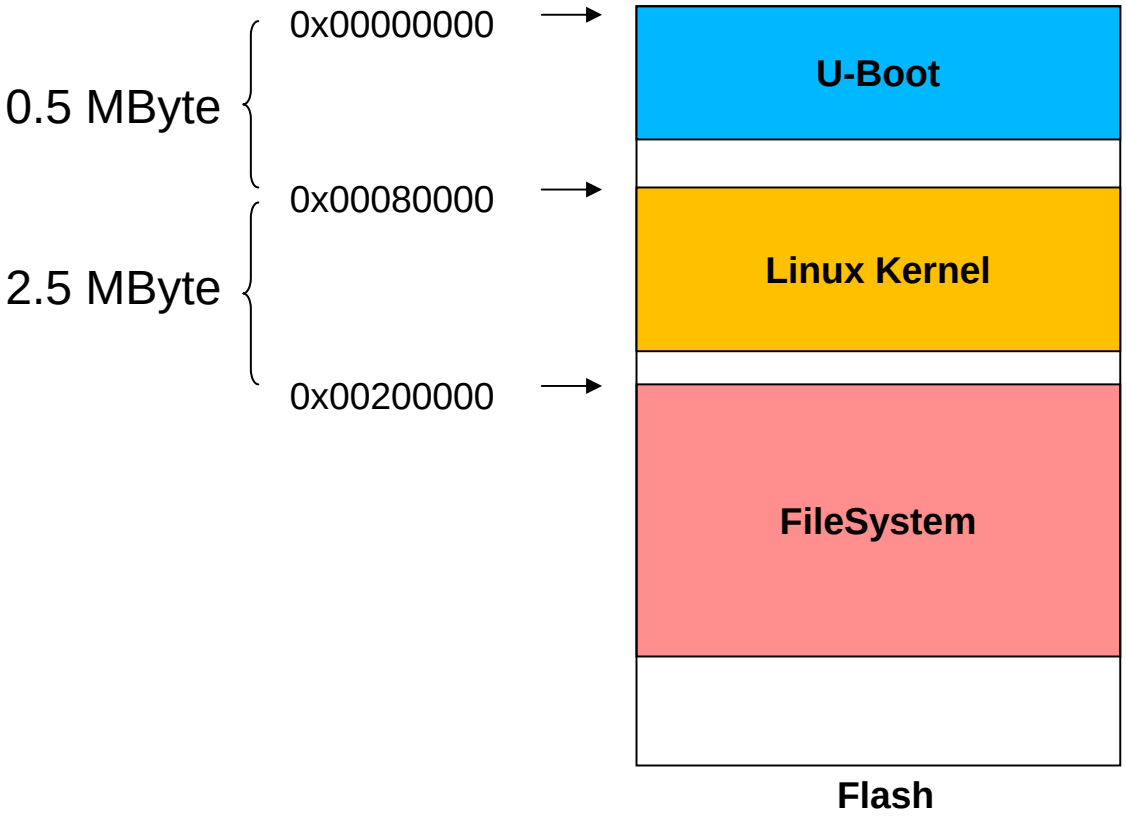
Filesystem

- Original rootfilesystem mounted on host, and build as cramfs
- Cramfs image had to be swapped because of endianness problems
- Filesystem programmed via u-boot on flash

Leon3 Board (Linux)

- Kernel started with special bootargs to mount rootfs from flash
  - cramfs support needed
- Kernel image build as uImage, bootable by u-boot
  - u-boot prepares bootargs, decompresses image and copies it into SRAM

Final Memory Map (flash)



Final Project Status

- Running u-boot on Leon3 board ✓
- Working on flash via u-boot (Leon3) ✓
  - programming u-boot-image, Linux-image and filesystem
- Booting Linux on Leon3 ✓
  - mounting filesystem from flash
- Support for flash on Spartan board ☹
  - not enough time

WP 3

„Linux on Power-Trust Leon3 Board“

Members:

- Alexander Melzer
- Thomas Tanzer
- Viktor Unterberger

Outline

- Requirements
- Tools and Hardware
- Challenges
- System Architecture
- Results

WP3 Requirements

- Port Linux onto Power-Trust Leon board
  - Kernel configuration
- Communication with HF reader
  - Leon ⇔ reader
- Serial communication Leon ⇔ host
  - Backup solution
- Ethernet communication Leon ⇔ host
- Eval application
  - TCP socket connection

Tools and Hardware

- Leon 3 Processor
  - 32-bit SPARC v8 processor, Harvard architecture, AMBA-2.0 AHB bus interface, DSU
- IAIK Leon 3 Board
  - 2 x serial interfaces, 1 x Ethernet interface
  - 4 MB SRAM, 8 MB SDRAM, 8 MB FLASH
- Snapgear Linux
  - Provided by Gaisler Research
  - 2.6 Kernel (µCLibc crosscompiled)
  - MMU …
- Other Gaisler Tools
  - GRMON, TSIM

Challenges (1)

UART

- Problems:
  - Just one serial interface available – wanted back-up solution
  - Identify pins, match voltage supply for second serial interface
  - Different baud rates – Leon (38400) and reader (9600)

- Solution:
  - Soldering second serial on Leon Board
  - Testing – read/write registers for second UART (GRMON)
  - Configuring baud rate
    - Grmon (wmem 0x8000090c 0x9c)
    - prom_stage2.c

Challenges (2)

Application for Leon

- Problems:
  - Serial – communication with reader
  - Ethernet – communication with Leon board
- Solutions:
  - Research, tutorials
  - Testing: host to reader, host to host, host – Leon - reader

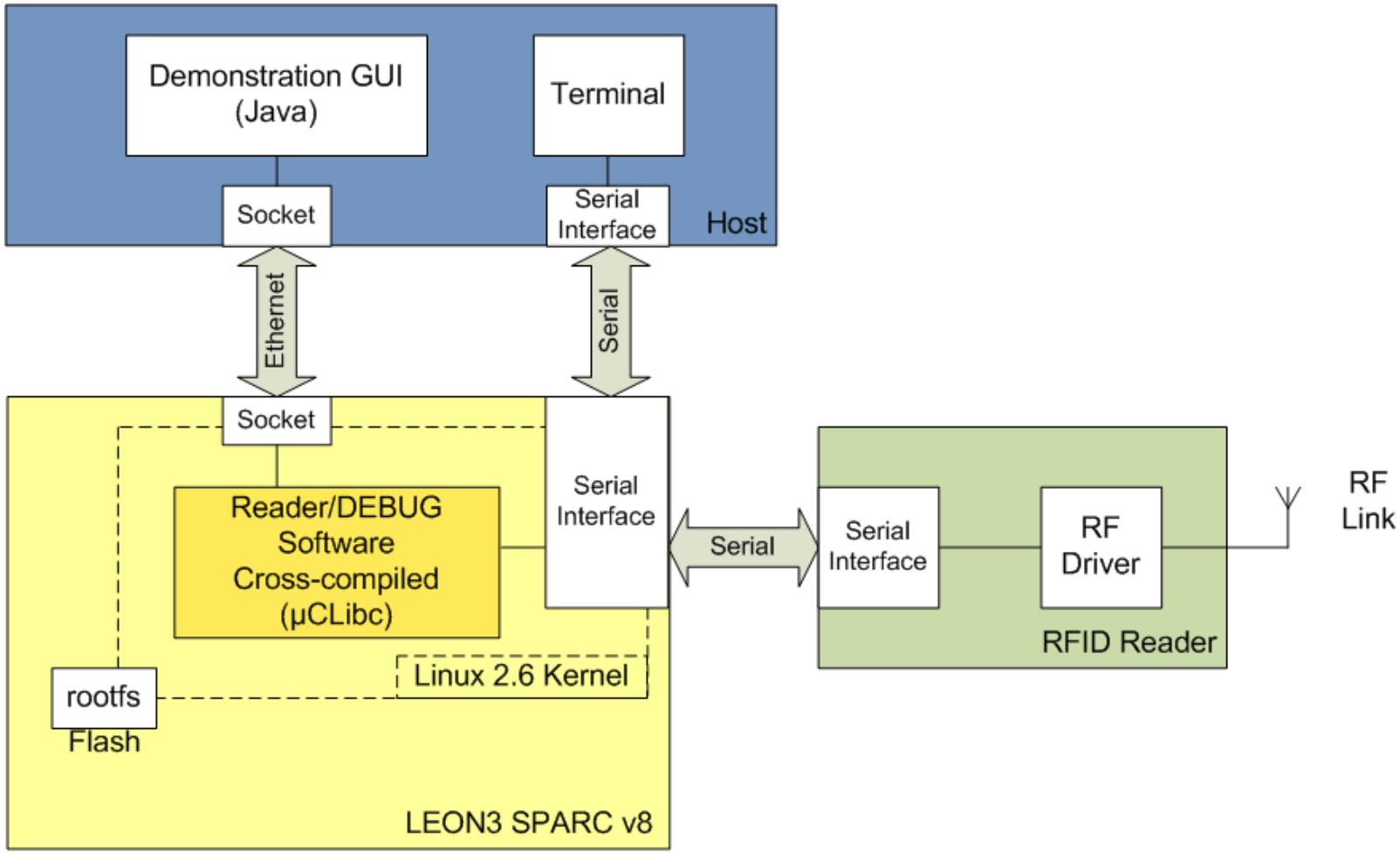
Reader

- Problems:
  - Different instruction set and behavior
- Solutions:
  - Adjust program – delay, continues read

Challenges (3)

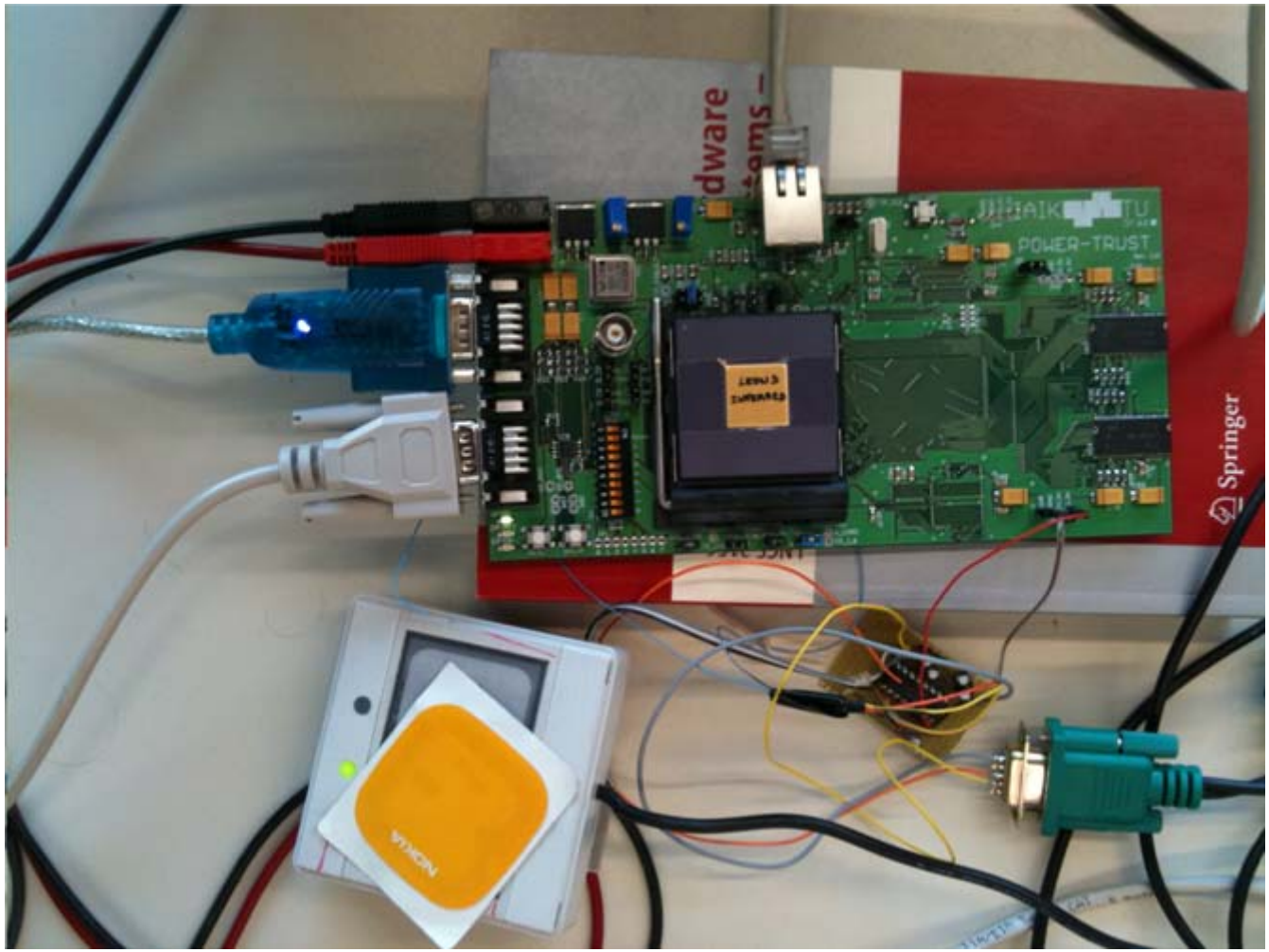
- Ethernet (Micrel KSZ8851)
  - Reading/writing from/to regs via GRMON
  - Integration of Linux Kernel 2.6.36 driver into Snapgear 2.6.21
  - Test routines in **`probe()`** method
  - Ping
  - Interrupt: workaround through Leon Timer IRQ (no GPIO) for full integration
  - Socket connection
- Memory limitations
  - Only SRAM working (no SDRAM) → build small Linux distribution, problems with testing
- Download only via Serial line to SRAM

System Architecture



SoC WP3 System architecture
V1.1 2010-12-12

Results



Final Presentation