System-on-Chip (SOC) Architectures and Modelling

A Networked RFID Reader for HF

Johannes Wolkerstorfer
Friedrich Lobenstock
Günther Langmann
Ralph Gruber
Thomas Kaltenbacher
Thomas Hodanek

Axel Schönauer
Erich Wenger
Bernd Schaller
Markus Krainz
René Allmeier
Christian Rathgeb
Motivation

System on Chip 2008
- 3VU lecture (Telematik master)
- 10 participants
- Project driven
- Content
  - SOC architectures
  - Modelling
  - HW SW interaction
  - IP integration
    - HW modules
    - OS
  - Soft skills
    - English
    - Project organization

SOC Project
- Groups
  - 4 groups of 3 students each
  - 125h work for each
- Work packages
  - Groups work on one aspect
  - Not every student does everything
- Regular Meetings
  - Weekly
    - Progress report
    - Group interaction
  - Student talks
    - Technological topics
      e.g. „Xilinx Virtex FPGAs“
      or „On-chip buses“
HF RFID Reader

Goal: RFID HF-Reader
- Networked HF reader
  - Support for multiple antennas (gate reader)
  - Long-range reader: anti-collision
- Demo application
  - RFID-Reader as webservice

Requirements
- ISO 15693 (13.56 MHz)
  - Reading range ~20 cm
- Xilinx ML403 board as basis
- Network interface (SSH)
- Linux operating system
- XML RPC based middleware
- Web service demo application

Reader
- Daten
- Takt
- Energie

Tag o. Transponder
Hardware platform: ML403 Board

ML403 board: A Virtex-4 FX-12 FPGA board

- PPC 405
- Ethernet
  - Gigabit
- Video-DAC
- AC97 sound
- Flash
- DDR
- SRAM
- USB
- UART
- LCD
Architecture

- HF Reader
  - on ML403 board
- Demo application
  - on remote web server

Project organization
- WP0 Analog frontend
- WP1 ML403 & drivers
- WP2 Digital frontend
- WP3 Linux
- WP4 Middleware
- WP5 Application
Analog frontend

13.56 MHz quartz oscillator

Transmitter circuit

- Power supply 2.5V (fixed voltage)
- HF signal generation: 13.56 MHz carrier
  - Amplitude modulation ASK (100%, 10%)
- Input: digital 3.3V PWM signal

Receiver circuit

- Passive filter for carrier suppression
- Sub-carrier detection (423 / 484 kHz)
- Oversampling ADC f=13.56 MHz

Antenna

- 50 Ohm antenna on PCB
  - conforming to [ISO10373]
Analog frontend (2)

**Receiver**
- Passive and active filters
  - for carrier suppression
- Sub-carrier amplification
  - (423 kHz and 484 kHz)
- Output: 3.3V 1-bit ADC

**Transmitter**
- Class-E amplifier using switched Power-MOSFET
- Suppression of harmonics via passive LC network

![System on Chip – Topic](http://www.iaik.tugraz.at)
Hardware platform (WP1) (1)
Hardware platform (WP1) (2)

PowerPC 405 Processor:
- PowerPC Processor Frequency: 300MHz
- Front Side Bus (PLB) Frequency: 100MHz

On PLB:
- DDR RAM: 64 MByte @ 100 MHz
- TriMode Ethernet MAC GMII: 125 MHZ and 200 MHz
- UART Lite: 115 200 Baud
- Interrupt Controller
- GPIO LEDs and Buttons
- Block RAM: 32 kByte

Two Clock Controller:
- Clock Controller 0 with 100 Mhz IN:
  - supports PPC, DDR, Ethermet MAC, PLB with 100 Mhz, 125 Mhz, 200 Mhz, 300 Mhz
- Clock Controller 1 with 13,56 Mhz:
  - supports RFID HW TX/RX Module with 108,48 MHz
Driver (WP1)

Address Map on PLB:
- DDR SDRAM: 0x00000000 – 0x03FFFFFF
- GPIO LEDs: 0x81400000 – 0x8140FFFF
- GPIO Buttons: 0x81420000 – 0x8142FFFF
- Timer: 0x83C00000 – 0x83C0FFFF
- Interrupt Ctrl.: 0x81800000 – 0x8180FFFF
- TriMode MAC: 0x81C00000 – 0x81C0FFFF
- SysAce: 0x83600000 – 0x8360FFFF
- SRAM: 0xFFF00000 – 0xFFF7FFFF
- UART Lite: 0x84000000 – 0x8400FFFF
- RFID Module: 0x82000000 – 0x820003FF

Memory mapped IO-Controlled Character device
- One antenna per character device
Digital frontend (WP2) (2)
Digital frontend (WP2)
Operating System (WP3)

Linux Kernel

Download the kernel from the [git.xilinx.com](http://git.xilinx.com) homepage (2.6.27)

Use of the CROSS-Compilation tool ELDK from [www.denx.de](http://www.denx.de):

After the installation of the tool you have to set the target architecture by exporting the following variable:

```bash
export CROSS_COMPILE=ppc_4xx-
```

Configuring the kernel:  
```
make ARCH=powerpc menuconfig
```

Compiling the kernel: 
```
make ARCH=powerpc
```

Board support package

Device-Tree includes information of the hardware design

Download the Device-Tree-Generator from [xilinx.wikidot.com](http://xilinx.wikidot.com) and copy it in your XPS project-directory

Generate BSP with XPS and copy the *.dts file into the kernel-tree:
```
arch/powerpc/boot/dts
```
Operating System (WP3)

Root File System

Decided to set up Debian GNU/Linux on the Power PC

The command `debootstrap` copied the required files for a running Linux with minimal configuration to the compact flash disk.

Booting Linux

Setting the `bootargs` options

Generating a SystemACE file of the kernel by using the XPS and copy it on the CF-Card

Tools

ssh, ntp, network, internet, serial console
Middleware (WP4)

Server–Reader interface

- Used for gathering tag-information and send them to the application server

Interfaces

- Driver: System-Calls
  - Standard- and status-commands (power_on/off, answer_available,…)
  - Send/receive frames

- Application Server: RPC-XML Protocol
  - Standard- and status-commands (Get_driver_version,…)
  - Send/receive UIDs with timestamp
Middleware - Functionality (WP4)

- Reader-Tag communication (ISO/IEC 15963 Standard)
  - Generate sending frames and analyze received frames
  - Anticollision sequence
- Onboard SQLite database
  - Write UIDs to the internal database if they are not in the ignorelist
- RPC-XML Client/Server
  - Communication with the application server
Demo Application (WP5)

As demo application we decided to develop a software for managing a warehouse:

- **Commandline** to configure settings and administrate users (admin tool).

- **Webinterface** to handle products, tags and readers. A secure connection is provided via https (user interface).
Demo Application (WP5)
Test Setup
RFID Controller Signal Output

No Tag

One Tag

Two Tags
Live presentation

Let’s have a look ...