Übersetzerbau in österreichischen Softwarefirmen
TU Wien
30.3.2017
Company overview and mission

CATENA
CATENA
“Partner of choice in System IP and IC Design”

• IC design-house founded in 1986
• Shares acquired by NXP Semiconductors, April 2012
• Maintains independent operations, working for 3rd parties
• Development of RF, Analog, Mixed-Signal and DSP Systems with a focus on Integrated Circuits and Wireless IPs
• 5 development centers
Offices

**Delft, The Netherlands**  1986
Headquarters
RF/AMS design, RF System architecture

**Eindhoven, The Netherlands**  2000
System Architecture,
Digital and DSP design, Embedded SW

**Kista, Sweden**  2001
RF/AMS design
RF System architecture

**Vienna, Austria**  2006
DSP Architecture and Tooling

**Dresden, Germany**  2016
Digital design
Embedded SW
Applications Areas

• **Wireless Communication**
  - Datacom (Bluetooth, WiFi 802.11x, WiMax, ZigBee, ISM Bands)
  - Proprietary Wireless Links (e.g.: Wireless Loudspeakers)
  - RFID Readers

• **Consumer & Automotive Electronics**
  - FM/AM Broadcast (Car, Portable and Mobile Phones)
  - Digital Radio (DAB(+), DMB-T, DRM(+), RDS, DARC)
  - TV Tuners (DVB Terrestrial, Cable and Satellite, Mobile TV)
  - GPS, GLONASS, Beidou Front-End
  - Keyless Entry and Immobilizer
  - Sensor Electronics

• **Industrial & Medical**
  - Advanced Measurement Circuits (MEMS Gyro/Accelerometer)
  - Implantable Wireless Communication (Pacemakers)
  - Electron Beam Deflector (deep submicron manufacturing equipment)
Catena Offerings

• **Product Development**
  • System Architecture development
  • Integrated Circuit development
  • Foundry interfacing
  • Turnkey services through partners

• **System Application and Product Support**

• **RF and Wireless Connectivity IPs**
  • ZigBee: 180/28nm
  • BT (BR, EDR, LE): 40/28nm
  • WiFi (11a/b/g/n/ac/ax): 65/28nm
  • GNSS (GPS, Glonass, Galileo, Beidou): 40nm
  • FM Transceiver: 180/40nm

• **Web**: http://www.catena.nl
Software? Compilers?

CATENA AUSTRIA
Catena DSP GmbH

TEFEC

TOOLING ENVIRONMENT FOR EMBEDDED CORES
Outline

• Embedded core

• Tools
  – Firmware development
  – Firmware analysis
  – Design space exploration
  – Core architecture configuration

• Compiler
  – Overview
  – Target independent passes
  – Target dependent passes
  – Specifics from digital signal processing
Not a product, just an …

EMBEDDED (DSP) CORE
Embedded Core - Example

**BTIPT**

Two different embedded cores:

X2  ---  3a
TOOLING ENVIRONMENT FOR EMBEDDED CORES
High level tools
Low-level tools
DESIGN SPACE EXPLORATION
Design space exploration

- Goal
  - Optimize chip area
  - Reduce power dissipation & energy consumption
- Based on a central architecture configuration
- Tools are dynamically configured to a specific target processor
- 2 phases
  - Evaluation phase
  - Production phase
Evaluation phase

- Find out important key parameters
  - Number of registers
  - VLIW width
  - Custom instructions
- Iterative approach
  - Build
  - Simulate
  - Evaluate
  - Modify
  - Repeat until happy
Production phase

- Key parameters settled
  - Fine tuning
  - Binary encoding

- Generate HW description
  - e.g. instruction decoder
  - Verilog, VHDL
ARCHITECTURE CONFIGURATION
Architecture configuration

- XML based
- Mostly behavioral description, centered around instruction set
- Optimized for compiler and simulation
- Experts system
Compiler passes
Target independent passes

- SSA
- Inlining
- Loop optimizations
- SIMD code generation
- Copy- and constant-propagation
- Partial redundancy elimination
- Hardware loops
- [ Instruction selection ]
Target dependent passes

- Control-flow optimizations
- IF-conversion
- VLIW instruction scheduling
- Software pipelining
- Register allocation
- Coalescing
- Spilling
DSP specifics

- Most applications have small kernel(s)
- Make use of VLIW
  - Scheduler, SWP
- Special Addressing Modes
  - Support for code like “x = *p++;”
- Fixed point calculations and datatypes
DSP specifics (2)

- Predicated Execution
  - IF-conversion
- Hardware Loops
  - Used for simple FOR-loops
- DSP Instructions, e.g. multiply-accumulate
  - Matcher

\[
\text{mac } a, b, c
\]
Example: FIR filter kernel

```c
#define FILTER_ORDER 26
#define N (FILTER_ORDER + 1)

int fir_kernel(int delay[N], int coeff[N])
{
    long a = 0;
    for (int i = 0; i < N; ++i)
        a += ((long)delay[i] * coeff[i]) << 1;
    return a >> 16;
}
```

fir_kernel:

b0:
- `mov r0, 0, r0`
- `ld (r0) += 1, hD1 || ld (r1) += 1, lD1 || bkrepi 25, b2_end`
- `ld (r0) += 1, hD1 || lD1 += 1, lD1`

b2:
- `ld (r1) += 1, lD1 || ld (r0) += 1, hD1 || macf hD1, lD1, a0`

b2_end:
- `macf hD1, lD1, a0`
- `ret`
- `mov hD0, lD0`
- `mnop`
Collaboration

- Praktikums- oder Diplomarbeiten
  - Compiler
    - Integrate and compare other frontends (e.g. clang)
  - Debugger
    - Integrate into Eclipse CDT debugging framework.

- Interested?
  - Have a talk with us
  - Write an email: uhirnschrott@catena.nl