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

ISO 9001:2000

QUALITYMASTERS

SRTCI

ACREDITED

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

CATENA

- **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

***T**OOLING* ***E**NVIRONMENT* ***F**OR* ***E**MBEDDED* ***C**ORES*

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 …

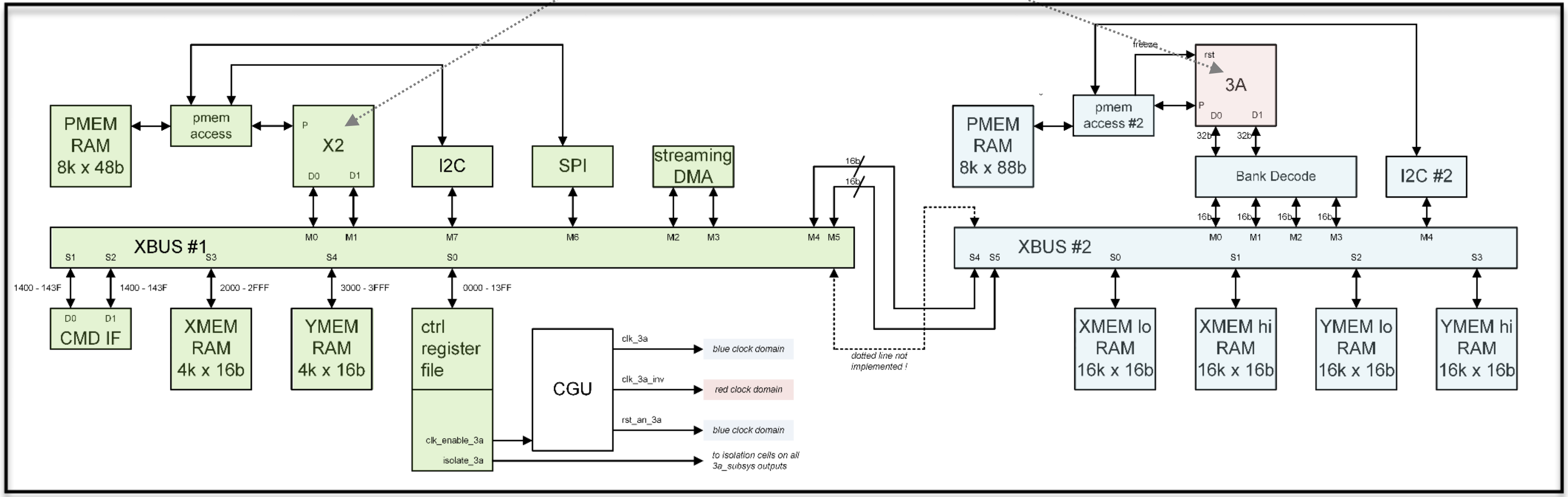
EMBEDDED (DSP) CORE

Embedded Core - Example

BTIPT

Two different embedded cores:

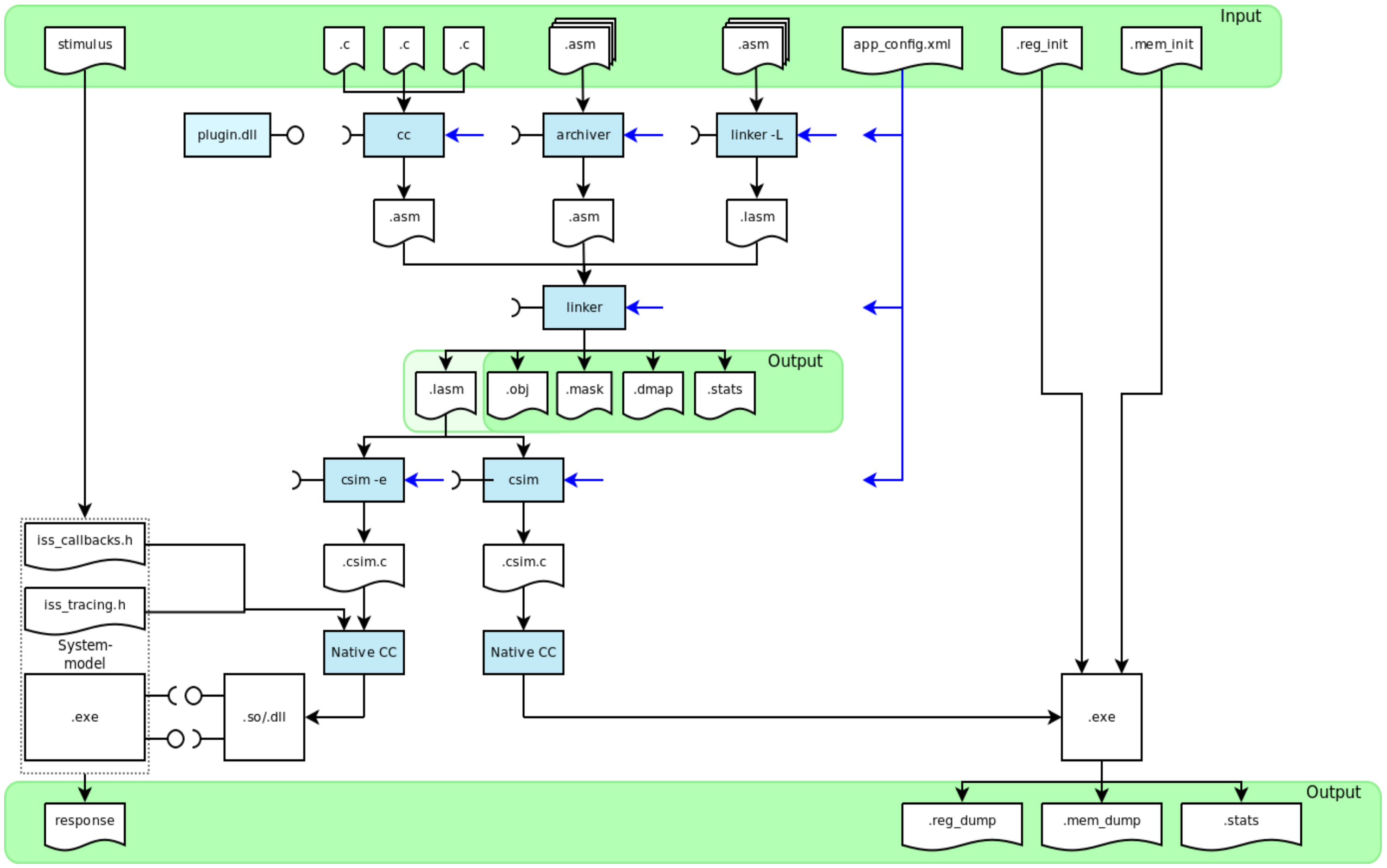
X2 --- 3a



CATENA

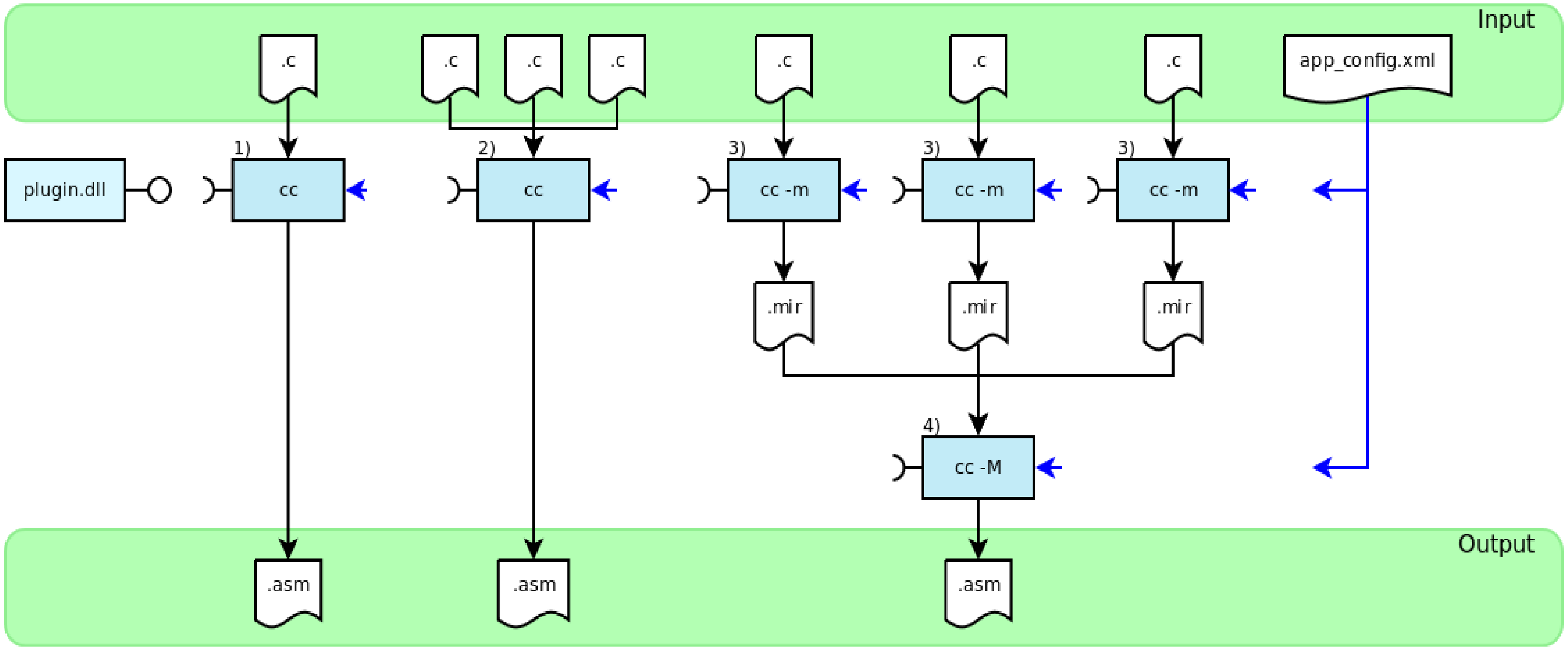
TOOLING ENVIRONMENT FOR EMBEDDED CORES

Overview

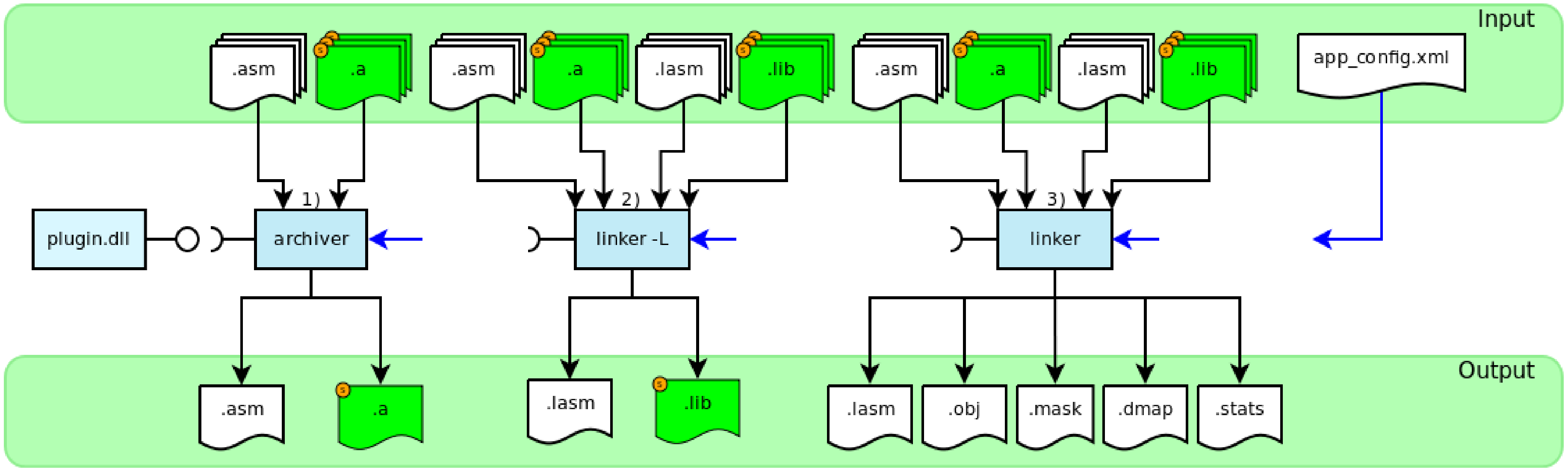


Input
stimulus
.c
.c
.c
.asm
.asm
app_config.xml
.reg_init
.mem_init
plugin.dll
cc
archiver
linker -L
.asm
.asm
.lasm
linker
Output
.lasm
.obj
.mask
.dmap
.stats
csim -e
csim
iss_callbacks.h
.csim.c
.csim.c
iss_tracing.h
System-model
Native CC
Native CC
.exe
.so/.dll
.exe
Output
response
.reg_dump
.mem_dump
.stats

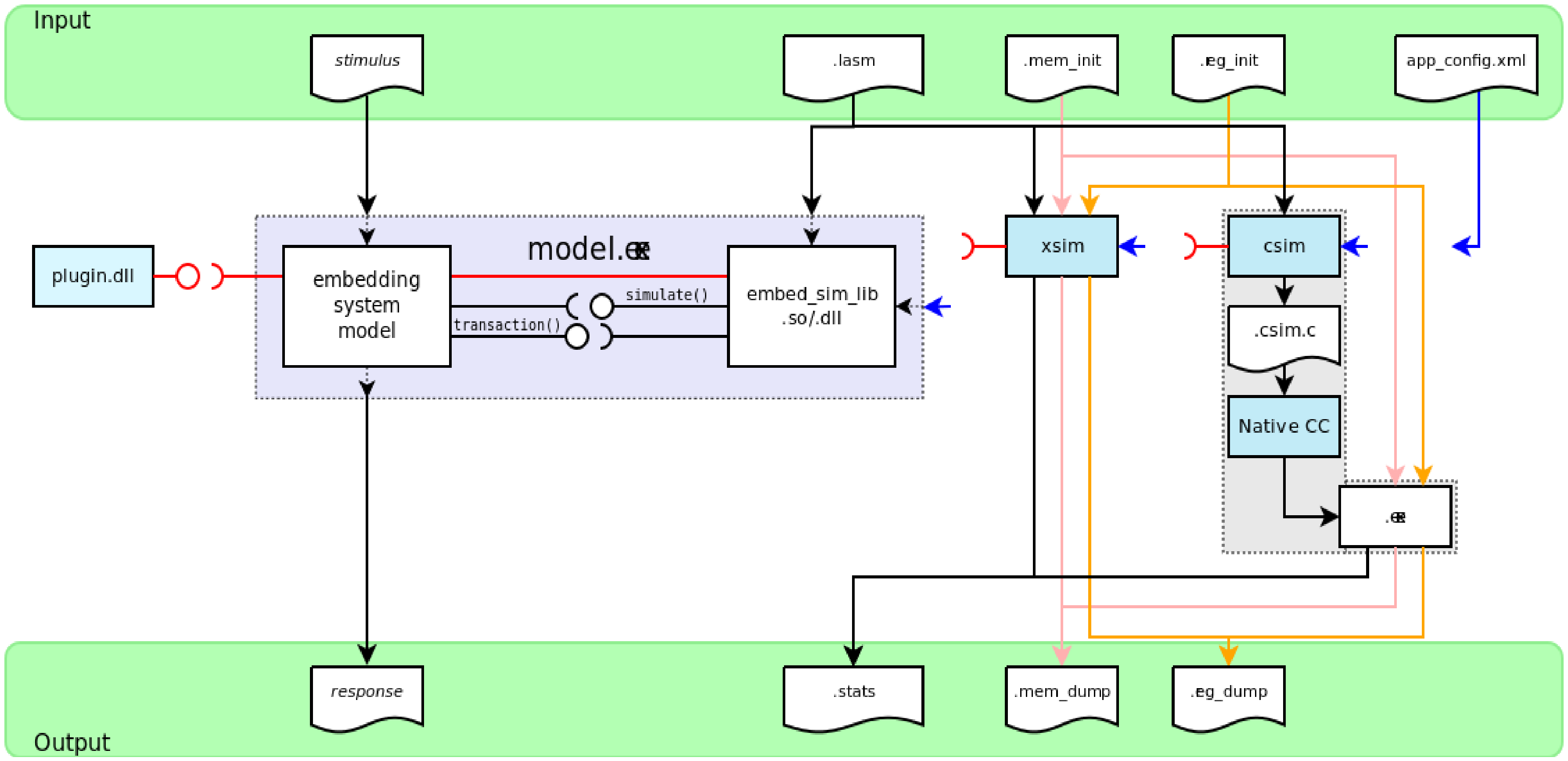
High level tools



Low-level tools



Simulation tools



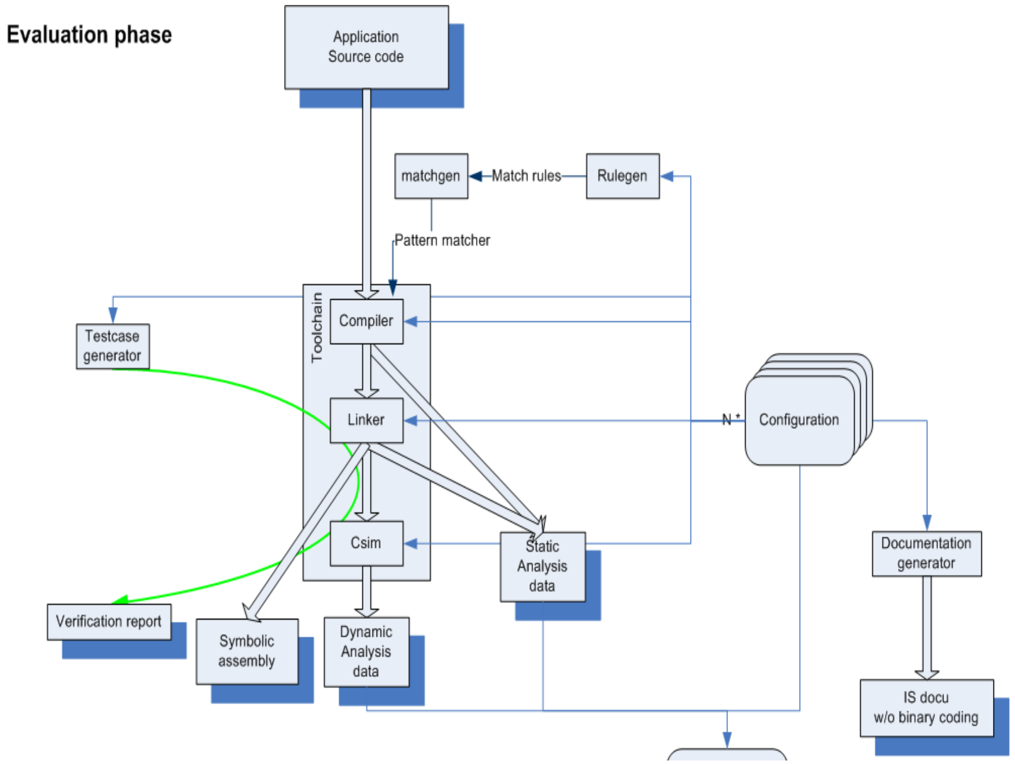
CATENA

DESIGN SPACE EXPLORATION

Design space exploration

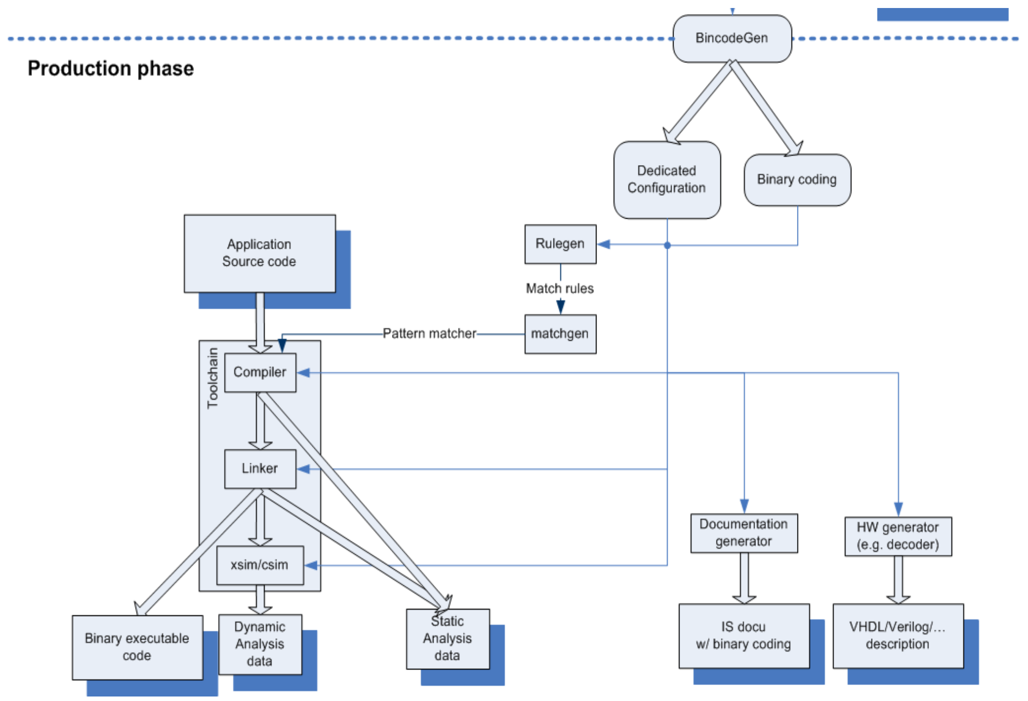
- Goal
  - Optimize chip area
  - Reduce power dissipation & energy consumption
- Based on a central architecture configuration
- Tools are dynamically configured to a specific target processor
- 2 phases
  - Evaluation phase
  - Production phase

Evaluation phase



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

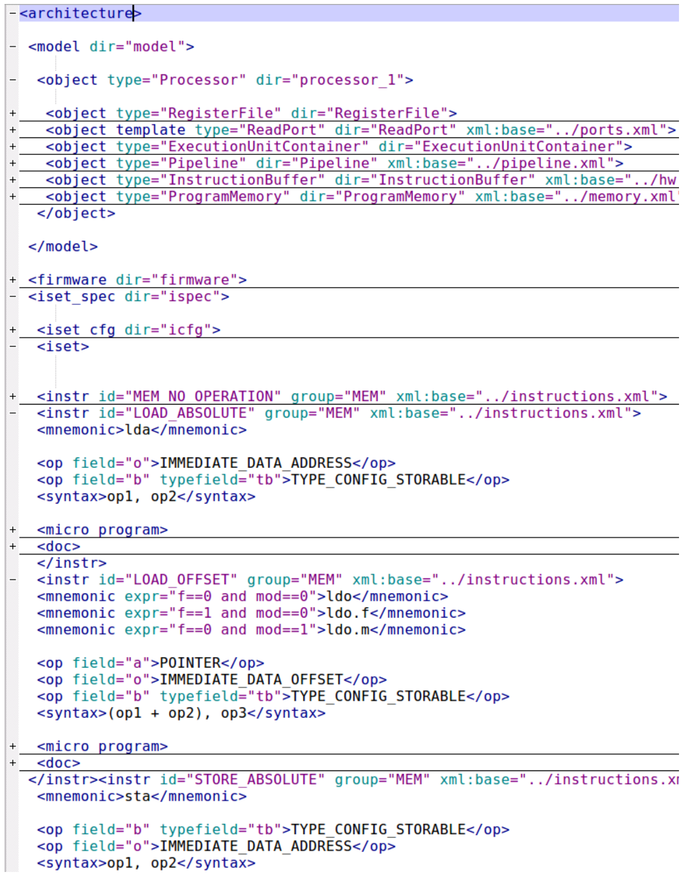
Production phase



- Key parameters settled
  - Fine tuning
  - Binary encoding
- Generate HW description
  - e.g. instruction decoder
  - Verilog, VHDL

ARCHITECTURE CONFIGURATION

Architecture configuration

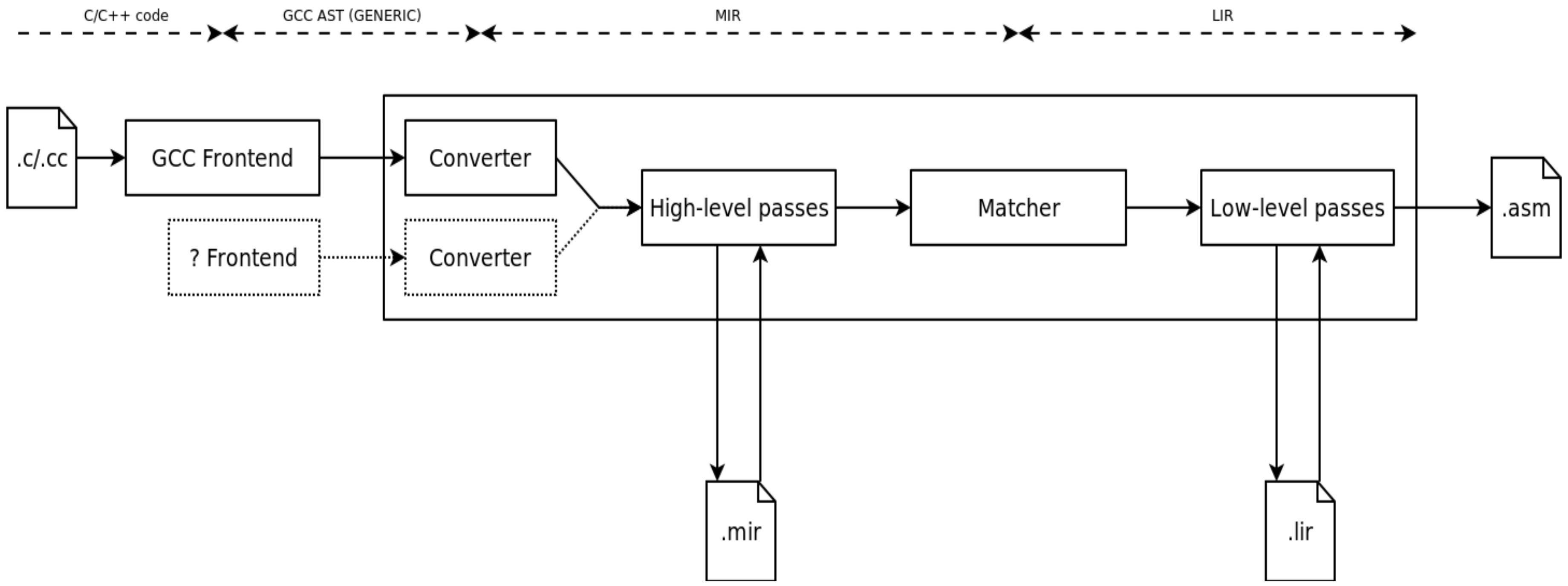


```
<architecture>

 <model dir="model">

  <object type="Processor" dir="processor_1">

   <object type="RegisterFile" dir="RegisterFile">
   <object template type="ReadPort" dir="ReadPort" xml:base="../ports.xml">
   <object type="ExecutionUnitContainer" dir="ExecutionUnitContainer">
   <object type="Pipeline" dir="Pipeline" xml:base="../pipeline.xml">
   <object type="InstructionBuffer" dir="InstructionBuffer" xml:base="../hw
   <object type="ProgramMemory" dir="ProgramMemory" xml:base="../memory.xml
  </object>

 </model>

 <firmware dir="firmware">
 <iset_spec dir="ispec">

  <iset cfg dir="icfg">
  <iset>


  <instr id="MEM NO OPERATION" group="MEM" xml:base="../instructions.xml">
  <instr id="LOAD_ABSOLUTE" group="MEM" xml:base="../instructions.xml">
  <mnemonic>lda</mnemonic>

  <op field="o">IMMEDIATE_DATA_ADDRESS</op>
  <op field="b" typefield="tb">TYPE_CONFIG_STORABLE</op>
  <syntax>op1, op2</syntax>

  <micro program>
  <doc>
  </instr>
  <instr id="LOAD_OFFSET" group="MEM" xml:base="../instructions.xml">
  <mnemonic expr="f==0 and mod==0">ldo</mnemonic>
  <mnemonic expr="f==1 and mod==0">ldo.f</mnemonic>
  <mnemonic expr="f==0 and mod==1">ldo.m</mnemonic>

  <op field="a">POINTER</op>
  <op field="o">IMMEDIATE_DATA_OFFSET</op>
  <op field="b" typefield="tb">TYPE_CONFIG_STORABLE</op>
  <syntax>(op1 + op2), op3</syntax>

  <micro program>
  <doc>
 </instr><instr id="STORE_ABSOLUTE" group="MEM" xml:base="../instructions.x
  <mnemonic>sta</mnemonic>

  <op field="b" typefield="tb">TYPE_CONFIG_STORABLE</op>
  <op field="o">IMMEDIATE_DATA_ADDRESS</op>
  <syntax>op1, op2</syntax>
```

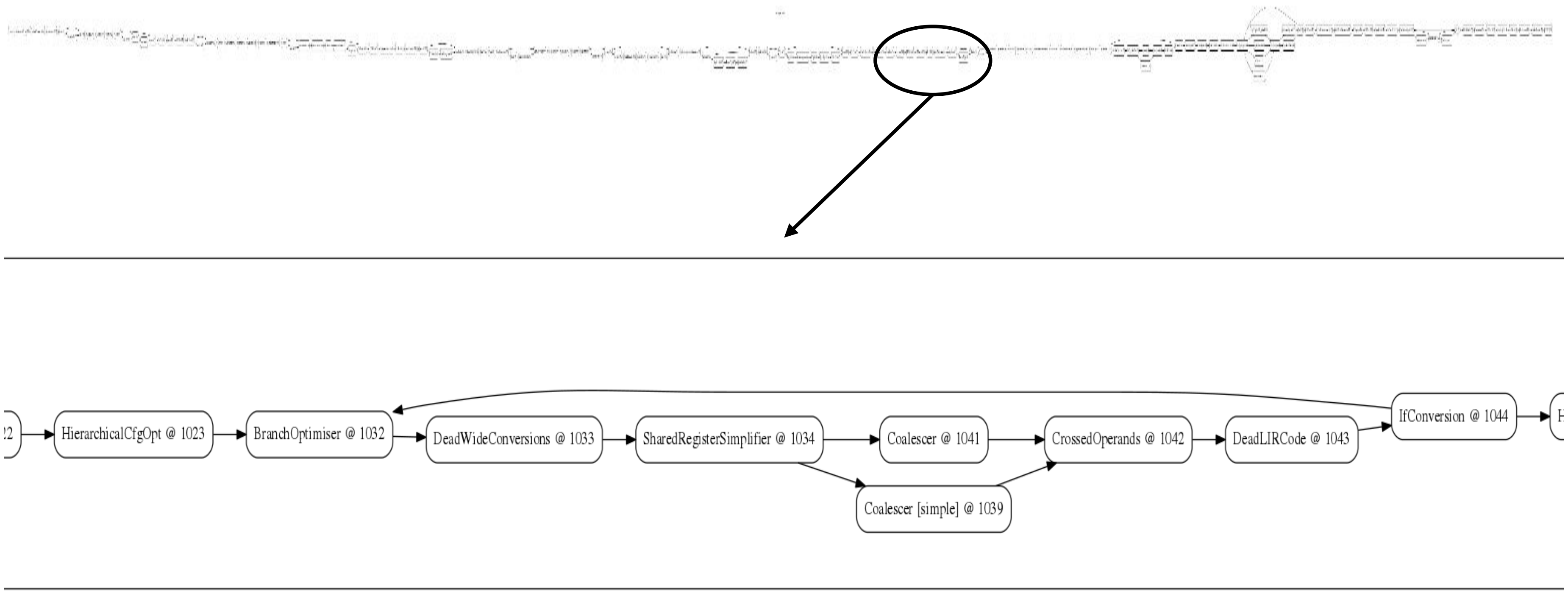
- XML based
- Mostly behavioral description, centered around instruction set
- Optimized for compiler and simulation
- Experts system

COMPILER

Compiler overview



Compiler passes


HierarchicalCfgOpt @ 1023
BranchOptimiser @ 1032
DeadWideConversions @ 1033
SharedRegisterSimplifier @ 1034
Coalescer @ 1041
Coalescer [simple] @ 1039
CrossedOperands @ 1042
DeadLIRCode @ 1043
IfConversion @ 1044

Target independent passes

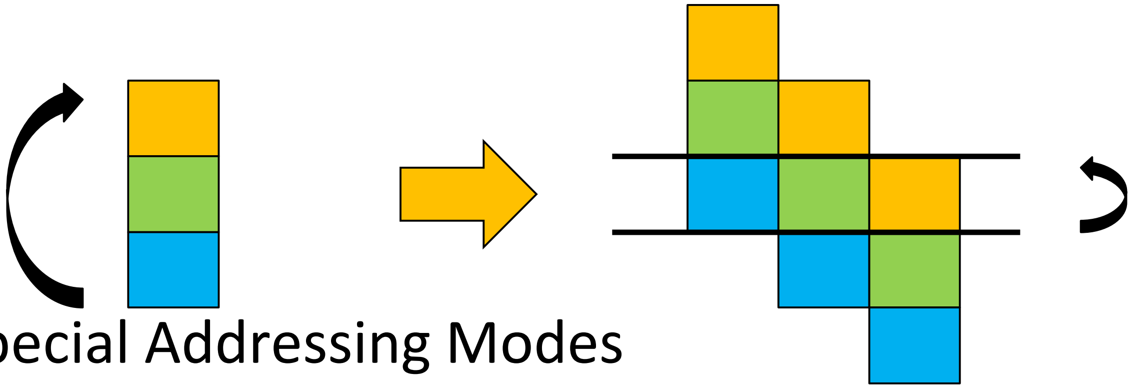
- SSA
- Inlining
- Loop optimizations
- SIMD code generation
- Copy- and constant-propagation
- Partial redundancy elimination
- Hardware loops
- [Instruction selection]

Target dependent passes

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

DSP specifics

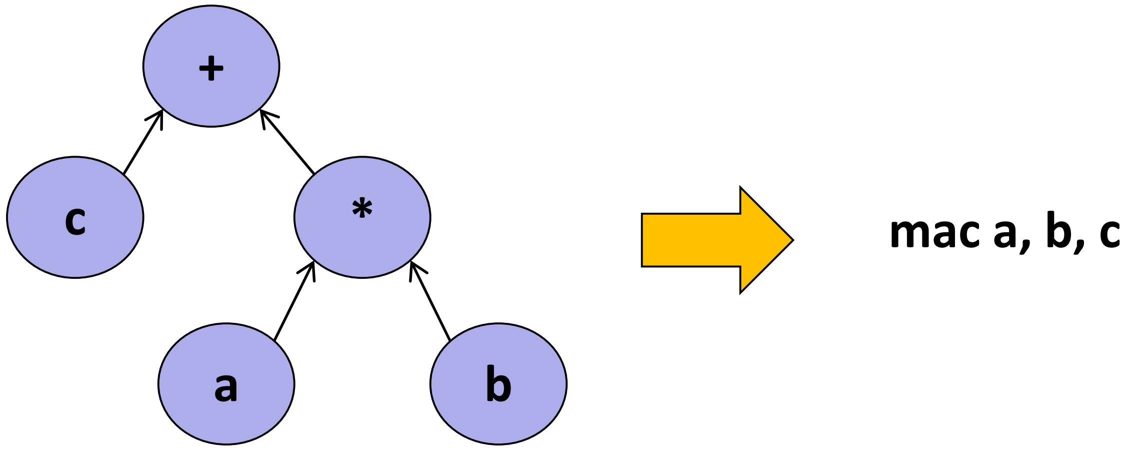
- Most applications have small kernel(s)
- Make use of VLIW
  - Scheduler, SWP



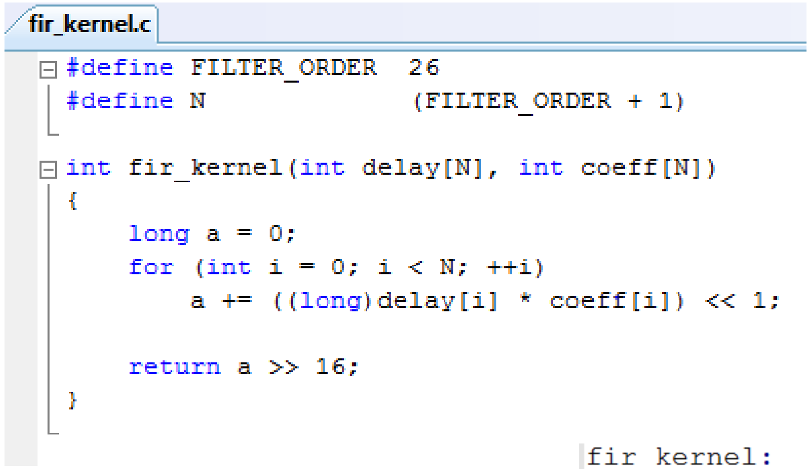
- Special Addressing Modes
  - Support for code like “x = *p++;”
- Fixed point calculations and datatypes

DSP specifics (2)

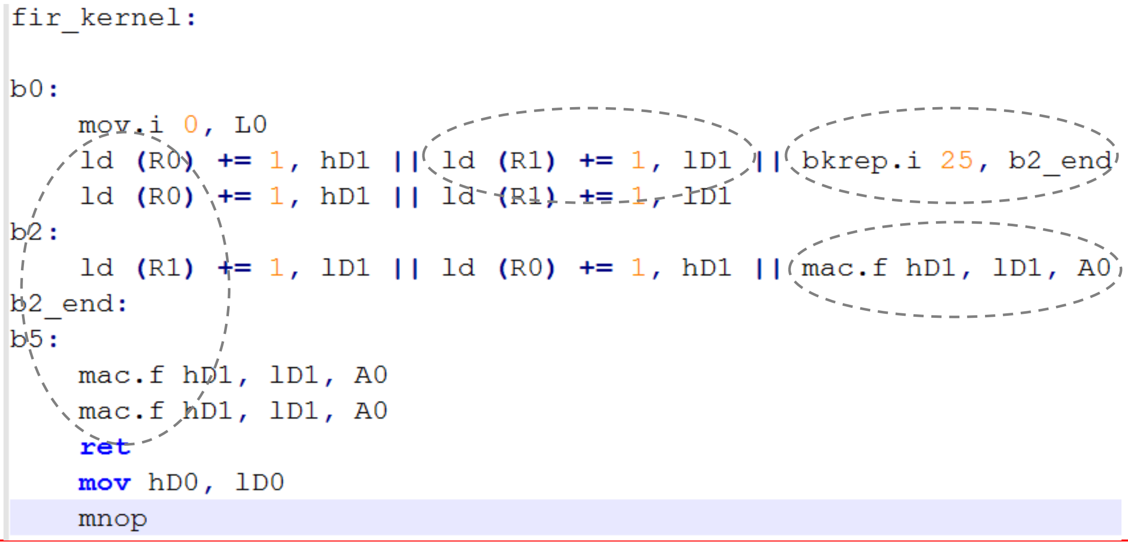
- Predicated Execution
  - IF-conversion
- Hardware Loops
  - Used for simple FOR-loops
- DSP Instructions, e.g. multiply-accumulate
  - Matcher



Example: FIR filter kernel


fir_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.i 0, L0
    ld (R0) += 1, hD1 || ld (R1) += 1, lD1 || bkrep.i 25, b2_end
    ld (R0) += 1, hD1 || ld (R1) += 1, lD1
b2:
    ld (R1) += 1, lD1 || ld (R0) += 1, hD1 || mac.f hD1, lD1, A0
b2_end:
b5:
    mac.f hD1, lD1, A0
    mac.f 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