LLVM compiler for the LEON3/4 fault tolerant processor family

Cobham Gaisler
Daniel Hellström, Daniel Cederman
5 December 2018
Flight software workshop 2018

This presentation does not contain any ITAR information
Introduction

LLVM support for LEON3/4 processor family

• Background
• Past work with LLVM
  – ESA study 1 - correctness & performance test-suite
  – ESA study 2 - LEON-REX backend
• ESA study 3 – GR712RC/GR740 backend
  – Objectives
  – Method
  – RTEMS integration and testing
  – Results
• Future software updates
• Current status
  – BCC-2 (bare-metal environment)
  – RCC-1.3-rc6 (RTEMS-5)
  – open-source upstreams
• Cobham Gaisler LLVM LEON road-map
• Summary
Background

LLVM support for LEON3/4 processor family

Some reasons why LLVM is interesting for us

• to explore the performance of the compiler
• LEON SW universe: two different LEON compilers to choose from, two legs to stand on
• modular design of llvm, the maintenance of it might be lower than with GCC
• we believe that developing the REX backend for LLVM was easier
• other tools are using LLVM (due to its license)
• partial validation of the compiler or tools around it might be easier
• front-ends can be added to llvm
• some space projects use auto-generated code, from UML for example.
• source code and instruction code analysis tools can be developed
  – real-time worst case execution time
• the license of LLVM
• an improvement for companies that require LLVM/Clang and are already using it
Major work performed:

- LLVM test suite and benchmarks
- Custom LEON designs to collect execution statistics, HW setup:
  - FPGA design: UT700+L3STAT (not present in UT700)
  - GR740
- L4STAT, L3STAT - On GR740 statistic counters used:
  - 22 Processor counters (branch prediction miss, instruction counter, ..)
  - 17 AHB Bus counters (double word access counter, ..)
  - 15 other counters (L2 cache hit & miss counter)
- RTEMS 4.10 run-time measuring:
  - Stack usage
  - Window overflow/underflow counters
- Results:
  - Compare different versions of LLVM Sparc backend
  - Compared GCC and LLVM SPARC using different configurations
  - Identify worst and best performing test by different measures
Background
GR716 (ESA funded LEON3-FT microcontroller) introduces a new 16-bit ISA called LEON-REX. Purpose is to reduce footprint, but can also be positive cache-hit rate.

- Designed & developed new LEON-REX backend for LLVM, Compiler & Assembler
- Backend tested and benchmarked on previously developed test-suite
- Newlib C & RTEMS master (2016 November) test-suite successfully executed
- Primary use case: GR716 LEON3-FT Microcontroller
- Identified performance bottlenecks (input to current work)
- Submitted upstreams but not yet included in LLVM master

LEON-REX
- SPARC ABI compliant
- Mostly 16-bit instructions, translated into 32-bit SPARC instructions
- All operations are 32-bit, however:
  - smaller Immediate
  - direct access to 16 registers, indirect 32 registers
  - no delay-slot
  - no access to special registers, use 32-bit mode
  - enter 32-bit mode on function call and trap
  - enter REX by ADDREX or SAVEREX or return to PC.1=1
  - OS run-time 32-bit
ESA Study 2 – LEON-REX backend

Overview (2)

LLVM/Clang 4.0:
• Activate LEON-REX backend with: -mrex

Results:
• Compression LLVM Sparc vs. LLVM LEON-REX (.text)
  – Newlib ~20% smaller
  – RTEMS ~15% smaller
  – RTEMS test-suite 15 fails of 525 (master as of November 2016)
• Possible to link LEON-REX with SPARC objects
**Activity summary**

- ESA funded activity starting 1:st of January 2018
- 12 months duration
- Major objectives
  - Fix all issues we previously discovered
  - GR712RC and GR740 device specific support
  - Performance improvements
  - Benchmark the updated backend (see results slide)
  - Build and adapt RTEMS/Newlib to use LLVM/Clang
  - Increase the TRL\(^1\) level by demonstrating ESA EagleEye CSW reference works
  - Upstream the work to open-source repositories
- In parallel to this work CG developed:
  - Improvements for GR716
  - Support for UT700/UT699e
  - RCC updates with new LLVM/Clang toolchain for RTEMS-5
- Final Presentation 11:th December at ESTEC

Technical work completed November 2018, now compiling the test-results.

\(^1\) Technology Readiness Level
ESB Study 3 – GR712RC/GR740 support

Work performed

1) Initial analysis part to identify more performance weaknesses and functionality to be improved (than previously known)
   - Analysing the LLVM/Clang Sparc backend source code, trying to find unimplemented parts, better configuration for Sparc and LEON, etc.
   - Generated code/instructions analysis
   - Quick analysis bad performing benchmarks
   - Missing features
   - Build SW component reveals missing

   Unit-tests developed for improvements made

   This lead to a Requirements Specification

2) Implementation of the Requirements Specification

   Partly iterative flow for identifying further performance improvements.

   Test-suite reused to benchmark existing and updated LLVM/SPARC backend
   - for correctness
   - Rerun for every optimization to see each contribution to help analysis
   - Compare before and after
   - Compare with GCC-7.2

   Make LLVM/Clang work with RTEMS-5 and Newlib
   - Use rtems test-suite for correctness

   Port EagleEye CSW reference architecture to use LLVM/Clang

   Upstream as much as possible to LLVM, Clang, Binutils, Newlib, RTEMS projects
GR712RC Dual-Core LEON3FT Processor

Dual-core processor with flight heritage

- TowerJazz 180 nm CMOS technology: Ramon-Chips RadSafe cell library
- TID 300 Krad(Si); SEL: LET > 118 MeV/cm²/mg; 1.8V core, 3.3V I/O voltage
- Power consumption (typical): 15 mW / MHz (dual core, MMU, FPU)
- Dual Core LEON3FT Fault-tolerant processor (SMP)
- 32 KiB cache with 4 parity bits per word
- CQFP240, 0.5 mm pitch, 32x32 mm, hermetically sealed
- 50 project wins and 50 projects in evaluation on 1-1-2017
GR740 – Quad-Core LEON4FT Processor

European Next Generation Microprocessor development

• Quad-core LEON4FT rad-tolerant SoC device
  – 4x LEON4FT with dedicated FPU and MMU
  – 128 KiB L1 caches connected to 128-bit bus
  – 2 MiB L2 cache, 256-bit cache line, 4-ways
  – 64-bit SDRAM memory I/F (+32 checkbits)
  – 8-port SpaceWire router with +4 internal ports
• 32-bit 33 MHz PCI interface
• 2x 10/100/1000 Mbit Ethernet
• Debug links: Ethernet, JTAG, SpaceWire
• MIL-STD-1553B, 2 x CAN 2.0B, 2 x UART
• SPI master/slave, GPIO, 21 Timers & Watchdog
• CPU master/slave, MMU, Memory scrubber

• LGA625 / CGA625 package
• ST 65nm bulk CMOS process
• ESA’s Next Generation Microprocessor (NGMP) activity
• Worst-case frequency of 250 MHz in production tests
• Power consumption (including I/O) at 40°C:
  – 4x CPU: 1.85 W (1700 DMIPS)
• Tested prototype parts and Evaluation boards: now!
• Qualification funded:
  – ESCC 9000 at end of 2018
  – QML-V equivalent at end of 2018
  – QML-V approval from DLA at end of 2019
GR716 – LEON3FT Microcontroller

European microcontroller development

- LEON3FT - Fault-tolerant SPARC V8 32-bit processor, 50 MHz
  - 16-bit instruction set support to improve code density
  - Floating Point Unit
  - Memory protection units
  - Non-intrusive advanced on-chip debug support unit
- External EDAC memory: 8-bit PROM/SRAM, SPI, I2C
- SpaceWire interface with time distribution support, 100 Mbps
  - LVDS Driver and Transmitter
- MIL-STD-1553B interface
- CAN 2.0B controller interface
- PacketWire with CRC acceleration support
- Programmable PWM interface
- SPI with SPI-for-Space protocols
- UARTs, I2C, GPIO, Timers with Watchdog
- Interrupt controller, Status registers, JTAG debug, etc.
- ADC 11bits @ 200Ksps, 4 differential or 8 single ended
- DAC 12bits @ 3Mmps, 4 channels
- Mixed General purpose inputs and outputs
- Power-on-Reset and Brown-out-detection
- Temperature sensor, Integrated PLL
- On-chip regulator for 3.3V single supply
- 132 pin QFP, 24 mm x 24 mm
EagleEye is a reference OBSW implemented within the Avionics Test Bench infrastructure at ESA.

It simulates an Earth Observation satellite.

Runs on a Spacecraft simulation infrastructure. The SVF is either on Simulator or in real-time on HW.

SW architecture:
- Xtratum hypervisor for time & space partitioning
  - RTEMS and XHAL partitions
- RTEMS 4.11 + LEON Xtratum enabled BSP
- GCC previously used

The following execution configurations are supported:
- LEON3 (FPGA) and LEON4 (LEON4-N2X)
- TSIM2-LEON3 simulator
ESA Study 3 – GR712RC/GR740 support

LLVM/Clang - Features added

Only some of which are listed below:

- libunwind (for C++ exceptions)
- compiler-rt (to replace libgcc)
  - added sparc
  - added mul, umul, div... builtins for V7
- Added assembler syntax, directives, aliases, etc.
  - New Partial Write PSR instruction (PWR)
- Errata work arounds for GR712RC and UT700/UT699E
- Flat register mode (-mflat)
  - application uses single register window
  - standard method of register saving/restoring reused from other architectures
  - no SAVE/RESTORE instructions used
  - frame pointer stored in %l6 so that OS without flat support works (RTEMS)
- Sparc LEON targets:
  - -mcpu=gr712rc
  - -mcpu=gr740
  - -mcpu=gr716
- Optimizations *(see next slide)*
In general

- Instruction timing
- Pipe-line dependencies
- GRFPU pipe-line dependencies
- (GR716) GRFPU-lite timing
- Generic Sparc optimizations by rearranging code or reducing instructions

Examples of optimizations implemented

- Tail call support
- Implement annulling branches \(\text{branch},a\)
- enable anti-dependency breaker for better Float registers usage
- better use of INSTcc instructions
- Register scavanger - avoid LLVM reserving register %g1
- Compare and branch separation to remove pipe-line dependencies
- schedule loads near each other
- .. many more
Overview
- Analysis still on-going to better understand results
- 160 benchmarks reused from earlier ESA study
- Not all optimizations are counted for
- Strings aligned to 1 byte, GCC uses 8-bytes

Improvements arranged in groups of differences

<table>
<thead>
<tr>
<th>Legend</th>
<th>Base → Final</th>
<th>GCC → Base</th>
<th>GCC → Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;= %10 faster</td>
<td>43</td>
<td>15</td>
<td>21</td>
</tr>
<tr>
<td>&gt;= %5 faster</td>
<td>75</td>
<td>19</td>
<td>39</td>
</tr>
<tr>
<td>&gt;= %1 faster</td>
<td>124</td>
<td>33</td>
<td>63</td>
</tr>
<tr>
<td>No change</td>
<td>29</td>
<td>17</td>
<td>32</td>
</tr>
<tr>
<td>&gt;= %1 slower</td>
<td>7</td>
<td>109</td>
<td>64</td>
</tr>
<tr>
<td>&gt;= %5 slower</td>
<td>3</td>
<td>84</td>
<td>32</td>
</tr>
<tr>
<td>&gt;= %10 slower</td>
<td>0</td>
<td>57</td>
<td>20</td>
</tr>
</tbody>
</table>

- 75 tests 5% faster or more
- Majority at least 1% faster

Benchmarks
- Speccint2000
- Coremark
- Parsec
- ASC Sequoia Benchmark (Sparse matrix-vector operations)
- BitBench (bit stream manipulation benchmarks)
- MiBench
- FreeBench
- PolyBench (linear-algebra, datamining, solversm stencil)
- Prolangs-C++, Prolangs-C
- Olden
- …
**Summary**

- Patches on RTEMS was mostly on tests but also on kernel
- No patches to Newlib required
- A Gaisler RCC Clang front-end developed
  - same syntax as the GCC toolchain, -qbsp=BSP
  - multi-libs defined
  - enables leaf optimizations by default
- LLVM integrated assembler by default
- Binutils linker
- C and C++ run-time still built with GCC
- RSB not modified, Newlib/RTEMS build scripts in RCC-1.3

**Test results**

- RTEMS test-suite was sucessfully used
  - When starting 71 failures
  - Today 3 failures. Failures expected and also with GCC
  - No remaining LLVM/Clang specific issues
  - Four BSP configurations used:
    - gr712rc BSP (GR712RC single-core)
    - gr740_smp BSP (GR740 quad-core SMP)
    - leon3_flat (generic LEON3 -mflat)
    - leon_sf (generic LEON3 -msoft-float)
  - Executed on hardware, TSIM2 and TSIM3 multi-core prototype
RCC-1.3 testing
Cobham Gaisler’s RTEMS-5 test flow

CG RTEMS-5 test-suite setup:
• Patches on RTEMS was mostly on tests but also on kernel
• Jenkins controlled build server
• Two Docker images
  – one for GCC, and
  – one for LLVM (different host requirements)
• A master Git repository, with sub-modules for all SW components
• Rebuild triggered by a commit push
• Continuous integration and release testing
• TSIM2 single-core and TSIM3-prototype SMP used for running rtems test-suite.
• A caching strategy:
  – depending on which submodule or part of master repository updated.
  – only modified parts are rebuilt and tested using Simulator

Testing execution cycle times:
• RTEMS test-suite @ 8-core x86_64 test server:
  – 570 tests on single-core, runs in 10mins
  – 630 tests on SMP, runs in 30mins
• Compiler test-suite not parallelized takes ~2h
LLVM/Clang backend for LEON3/4

Current Status

• LEON-REX submitted upstreams in the past but not yet included in LLVM master

• BCC-2, LEON bare-metal environment
  – LLVM/Clang 4.0 (experimental) toolchain for GR716 available
  – LEON-REX back-end included
  – C11/C++11 language

• Most recent work submitted but more pending, including optimizations and UT700 support

• RCC-1.3 (available since November 2018)
  – LLVM/Clang 8.0 with latest optimization
  – GR712RC/GR740 support, ready for production
  – UT700 support pending
  – RTEMS test-suite and benchmarks shows good results

• VxWorks/Linux toolchains not yet available

• Support for UT699 and AT697 (LEON2) not planned today.
Mission: To support all recent LEON devices on all OSes

Past releases:
- BCC-2 LLVM/Clang 4.0 (mid 2017)
- RCC-1.3 LLVM/Clang 8.0 (November 2018)

Short-term roadmap:
- Upstream existing optimization patches to llvm/clang and RTEMS master (on-going)
- More optimizations are under development (on-going)
- Update RCC-1.3 to LLVM/Clang 8.0 stable-release once available (2019)
- Release VxWorks 7 LLVM/Clang 8.0 toolchain (2019)
- Rebase LEON-REX backend to LLVM/Clang master and resubmit it
- Update BCC-2 to Clang 8.0 with LEON-REX support

Long-term roadmap:
- Linux toolchain
- LEON5 multi-issue pipe-line will require updates
- Investigate optimizations for size (-Oz)
LLVM/Clang backend for LEON3/4

Summary

• Latest LLVM/Clang for LEON is available today in RCC-1.3-rc6
• Benchmark results indicate that:
  – Some applications are 10% better with GCC
  – Some applications are 10% better with LLVM
  – GCC slightly better on average but roughly the same performance
• RCC precompiled and same syntax as GCC to making it easy to try out
• Could be worth to try it out for the main FSW computing algorithm
• CG plan to make LLVM/Clang available for RTEMS/VxWorks/BCC-2
• Once LLVM/Clang 8.0 officially released, there will be a production ready LLVM for RTEMS and VxWorks7
The end

Thank you for listening