Syntacore Development Toolkit 2022.12-SP1

Release Notes

v1.0.8, 2023-02-03

Table of Contents

Basic tools/packages revisions

This section list revision of the basic packages, used by the toolchain.

Component	Revision
GCC	12.2.1
GNU Binutils	2.38
Newlib	4.10
GLibc	2.33
GNU GDB	12.1.90 with python support enabled
LLVM	16.0.0 upstream-based llvm with downstream optimizations
Open On-Chip Debugger	0.11.0+dev-g78f0168
QEMU	7.2.0
Eclipse IDE	2022-06

Release Updates

2022.12-SP1 Release

This release include the following updates:

General

- Bug fixes

QEMU

- Bug fixes

LLVM

- Bug fixes

Bug fixes for 2022.12-SP1 release

- General: lost cmake llvm files in tools directory are returned.
- QEMU: user networking is fixed.
- QEMU: rare assert when executing RVV instruction is fixed.
- LLVM: backend error for riscv64 in clang lowering of pseudo instructions is fixed.

Known Issues and Limitations for 2022.12-SP1 release

GCC unrolling loop issue for double precision FP

We are facing a performance issue with unrolling loop with double precision FP using GCC compiler.

In the following example:

```
#include <math.h>
#include <stdio.h>

#define N NUM_OF_ITERATION

volatile double temp = 0;

int main() {
  for (double i = 1.0, j = N; i <= N; i++, j--) {
    temp = i * j;
  }
  return 0;
}
```

Loop unrolled by 8 with N=1000 and not unrolled with N=1008 and more

Workaround:
Use `-funroll-loops` and `--param max-iterations-to-track=100000` compiler options
to unroll loop with 100000 iterations and improve performance

When compiled with Clang `__builtin___clear_cache` doesn't flush instruction cache

`__builtin___clear_cache` is compiled to `__clear_cache` call, which is not implemented properly in libgcc for the RISC-V platform.

Workaround:
Use `__riscv_flush_icache(start, end, 0)` instead of `__builtin___clear_cache(start, end)` (if available), or add the following implementation:

```
void __clear_cache(void *start, void *end) {
  const int SYSCALL_RISCV_FLUSH_ICACHE = 259;
  const int SYS_RISCV_FLUSH_ICACHE_ALL = 0;
  register void *start_reg __asm("a0") = start;
  const register void *end_reg __asm("a1") = end;
  const register long flags __asm("a2") = SYS_RISCV_FLUSH_ICACHE_ALL;
  const register long syscall_nr __asm("a7") = SYSCALL_RISCV_FLUSH_ICACHE;
  __asm __volatile("ecall"
                   : "=r"(start_reg)
                   : "r"(start_reg), "r"(end_reg), "r"(flags), "r"(syscall_nr));
}
```

Eclipse IDE doesn’t work properly when ran from a shared installation

Eclipse IDE in the Syntacore Development Toolkit distribution is configured to store its workspace folder in the same location as the rest of the installation. This is not desirable when the package is installed in some shared location and multiple users are running from the same location - the Eclipse IDE would try to use same workspace folder for all users, and this is likely to fail because of the directory permissions.

Workaround:
To move workspace folder to a custom user-specific location open file `eclipse/configuration/.settings/org.eclipse.ui.ide.prefs`, and change `SHOW_WORKSPACE_SELECTION_DIALOG` to `true`. Then open `eclipse/eclipse.ini` file and remove line that sets `user.home` constant, and add `-Dosgi.configuration.area=@user.home/eclipse-configuration` anywhere after `-vmargs` line. You can change `eclipse-configuration` to the path of your choice - Eclipse configuration file will be stored in that folder.

Can’t build a sample project with copied sources

If sample application IDE project has been imported into the workspace with an option "Copy projects to workspace", then it is not possible to build it.

Workaround:
Don’t enable "Copy projects to workspace" when importing sample projects from this distribution.

Changes in default linker script for GCC

The gp estimation in the default linker script is assuming gp won't cover range cross `DATA_SEGMENT_ALIGN`. When modyfing default linker script please take it into account. Specifically if you introduce the `LOAD` segment containing every rwx section, no need for it to have the data segment alignment. Your options are: add more alignment to `.init_array` to prevent sections motion or recalculate gp to move it closer.

See community discussion here

Using deleted BSP API `sys_tick64_t` or `rtc_now64()`

If your application using timer API `sys_tick64_t` or `rtc_now64()` you will face compilation issues.
To fix it please replace `sys_tick64_t` by `sys_tick_t` and `rtc_now64()` API function by `rtc_now()`.

Sample Applications limitations

- `hello_cpp` and `cpp_test` examples don't fit into `tcm` memory. These applications targeted to `Syntacore SCR5-SCR9 ram` configurations
- `mmu` and `pmu` examples are targeted to `Syntacore SCR7-9` configurations
- `rvv_test` and `rvv_benchmarks` are targeted to `Syntacore SCR7-9` configurations
- `hypervisor` example is available in command line mode for `Syntacore SCR9` configuration
- `cpp_test` cannot be used with `compiler-rt` feature because only builtins are available in this release

Previous Releases

2022.12 Release

This release include the following updates:

RISCV toolchain

- Riscv GCC is updated to `12.2.1`

QEMU

- Version updated to `7.2.0`
- RVV instructions support is enabled for **syntacore_scr7** and **syntacore_scr9** machines
- Fixes for **Syntacore SCR9** support
- Added experimental support of **SCR9 with L3 Cluster**
- Added experimental support of **AIA**
- Added support of **PMP** and **SCR timer's** divider and clock selection
- Bug fixes

LLVM

- Still on upstream `16.0.0` release upstream-based with downstream optimizations
- **Syntacore SCR9** optimizations
- RVV Auto vectorization (including `SPEC2017` cases) improved
- In Linux installation added experimental **compiler-rt** (ONLY **builtins**, **profile** and **crt**) for RISCV Linux build (`BUILD_LINUX=1`: `--rtlib=compiler-rt` and `-fprofile-inst-generate` support). Please note that sanitizers, memory profiler and fuzzer are skipped in the current release. Also, other llvm runtime libraries (libcxx, libcxxabi, libunwind) are under development for RISC-V arch support.
- In Linux installation added experimental **llvm-profdata** and **llvm-cov** for RISCV Linux build (`BUILD_LINUX=1`)
- Added new tools **llvm-objdump**, **llvm-ranlib**, **llvm-symbolizer** for Linux and Windows builds
- Tuning for benchmarks and `SPEC2017` on **Syntacore SCR7 - SCR9** platforms
- Bug fixes

BSP

BSP code refactoring

Please note that `sys_tick64_t` type has been replaced by `sys_tick_t` one to simplify Timer API. Also `rtc_now64()` API function has been removed - use the already existing `rtc_now()` instead.

Sample applications

- Refactoring code
- Possibility to build benchmarks for `Linux`
- New applications added
  - `hypervisor` - **experimental** in this release. Only `QEMU` **vcu118_scr9**
  - `mmu/sv39-pages-4k-identity-mapping` - Memory management Unit: identity mapping with 4K pages
  - `mmu/sv39-pages-4k-asid` - Memory management Unit: non-trivial mapping with 4K pages and enabled ASID
  - `pmu/count` - Performance Management Unit: count baremetal example **FPGA only**
  - `pmu/overflow-interrupt` - Performance Management Unit: overflow-interrupt baremetal example **FPGA only**
  - `rvv_benchmarks` - **experimental** in this release. RVV benchmarks for `QEMU` only
  - `rvv_tests` - **experimental** in this release. Baremetal RVV example for `QEMU` only

VSCode IDE

- Add VS Code project files to sample projects (example of Makefile build integration and starting the debug session on OpenOCD and QEMU)
- Add project-specific cmake kits files to the Coremark project - example of a cmake integration.
- Add Launch configuration example for the Linux host debug session - using deploySteps to upload the debug applications, starting the gdbserver via SSH.

Native GDB

Native GDB is now available in the release at

```
<install path>/riscv-gcc/sysroot/bin
```

Open On Chip Debugger

- Updated dependencies:
  - libusb: `1.0.23` → `1.0.26`
  - libftdi: `1.4` → `1.5`
  - libhidapi: `0.7` → `0.12`
- General purpose and floating-point registers use ABI naming

    - users are expected to update their scripts
- Deprecated `digilent-hs2_fixed.cfg` adapter configuration file is removed
    - `digilent-hs2a.cfg` should be used instead
- Improved RVV support
    - Added support for `VCSR` register
- Added universal configuration file to use with development boards
- Bug fixes

2022.09-SP1 Release

2022.09 Service Pack 1 release includes the following updates:

QEMU

- Introduce User mode to debug Linux applications `qemu-riscv32` and `qemu-riscv64`

LLVM

- Reduce llvm release size

Documentation updates

2022.09 Release

2022.09 release includes the following updates:

QEMU

- Version update to `QEMU 7.0`
- Bug fixes

Open On Chip Debugger

Version update to `openocd 0.11`

Riscv

- riscv-gdb `12.1`
- riscv-gcc `12.1.1`
- BinUtils version upgrade to `2.38`
- Python support is enabled in riscv-gdb

LLVM

- llvm `16.0.0` with `-mtune=scr1..9` support - first release

2022.06 Release

2022.06 release includes the following updates:

Riscv Toolchain

- RISCV GCC version upgrade to `11.2`
- BinUtils version upgrade to `2.37`
- GLibc `2.33`
- Newlib `4.10`

QEMU

- QEMU 5.2, added `SCR6/9` platform

IDE

- Eclipse 2022-06

Sample applications

- Updated Sample Applications, added `external_interrupt`

2021.03

Added FreeRTOS demo

2021.02

- Updated BSP
- Updated documentation
- Bugfixes

2020.09

- Riscv GCC `10.2.01`
- BinUtils `2.35`

2020.07

- Eclipse `2020.06`
- Added platforms **SCR5**, **SCR7**

2020.04

- Riscv GCC `9.2.0`
- BinUtils `2.34`
- Eclipse `2020.03`

2019.10

- Eclipse `2019.09`
- Added openocd cfg `digilent-hs2a.cfg`
- Added `rv64` support for **SCR3**, **SCR4**