Heterogeneous Multicore Debug Paper

Róisín O'Keeffe, Ashling

December 2020

Table of Contents

List of Figures

Introduction

Over the last decade there has been a dramatic increase in the number of multicore designs in embedded electronic SoC (System-on-Chip) devices. This drive towards multicore SoC designs was primarily driven by a desire for improved power consumption, cost, performance criteria, smaller form factors and concurrent application requirements. Combining several cores into a single device can reduce the number of components in the end-product thereby lowering long term costs and improving performance and power consumption.

More recently the trend has evolved towards heterogeneous multicore designs, involving a mix of different core architectures within a single SoC, as some embedded cores are more suited to specific tasks and application requirements.

Although the debug ecosystem for homogeneous multicore designs evolved rapidly, one of the barriers to adoption of heterogeneous multicore designs in embedded devices and SoCs relates to the availability of suitable debug environments for software development in heterogeneous SoCs.

Many of the existing debug environments were proprietary or limited to single vendor or architecture type (e.g. Arm, RISC-V, ARC, MIPS, DSPs). Each architecture type therefore typically needed its own toolset which only supported a specific vendor's core. This presented some major challenges:

- Engineers had to use separate toolsets for each core in their SoC
- SoC-wide debug and verification wasn't possible as toolsets only worked with a specific core
- Multicore, heterogeneous support within a single toolchain was not possible

The following example represents a multicore SoC containing both Arm and RISC-V cores. In this case a single DAP supports CoreSight debug access via Advanced Peripheral Bus (APB) for all the cores.

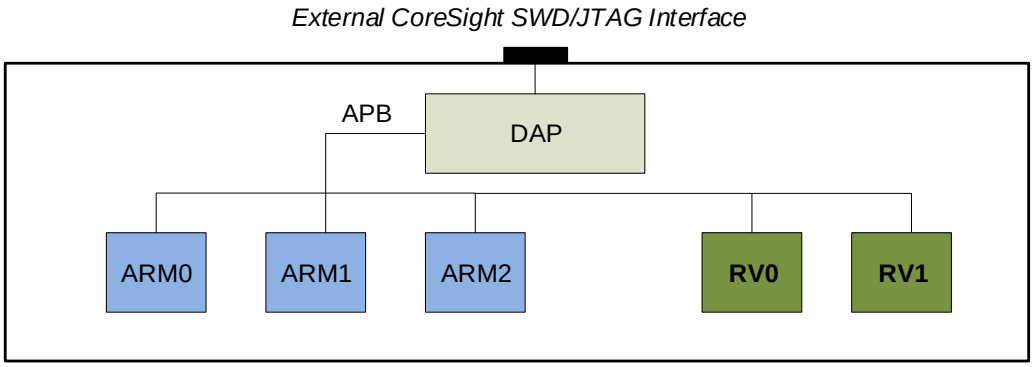

Figure 1: Multicore SoC containing Arm and RISC-V cores

To efficiently debug this heterogeneous example, a single toolset providing debug support for both the Arm and RISC-V cores is required, using a single instance of the debugger and a single hardware debug probe.

A heterogeneous multicore debug environment must also be able to handle different combinations of embedded architectures within a single debug environment (e.g. Arm+RISC-V, RISC-V+ARC or Arm+ARC etc) in addition to a comprehensive range of multicore debug features to provide the necessary debug control to enable simultaneous debugging across the different cores in the device.

Simultaneous heterogeneous multicore debugging within a single environment ensures that the developer has SoC-wide visibility and control allowing debug of complex interactions between different architecture cores.

Ashling has developed an integrated, vendor-independent, heterogeneous multicore IDE and debug solution allowing users to efficiently develop and debug code executing on multiple different architectures within a single development environment and with a single hardware debug connection.

RiscFree™ IDE

Ashling's Eclipse based ***RiscFree™*** Integrated Development Environment (IDE) for heterogeneous multicore designs is a seamless environment for software development including support for writing, building, simulating and hardware debugging. The professional commercial grade software is available as a single download and is licensed for the required combination of cores in the embedded device to enable heterogeneous multicore software development and debug.

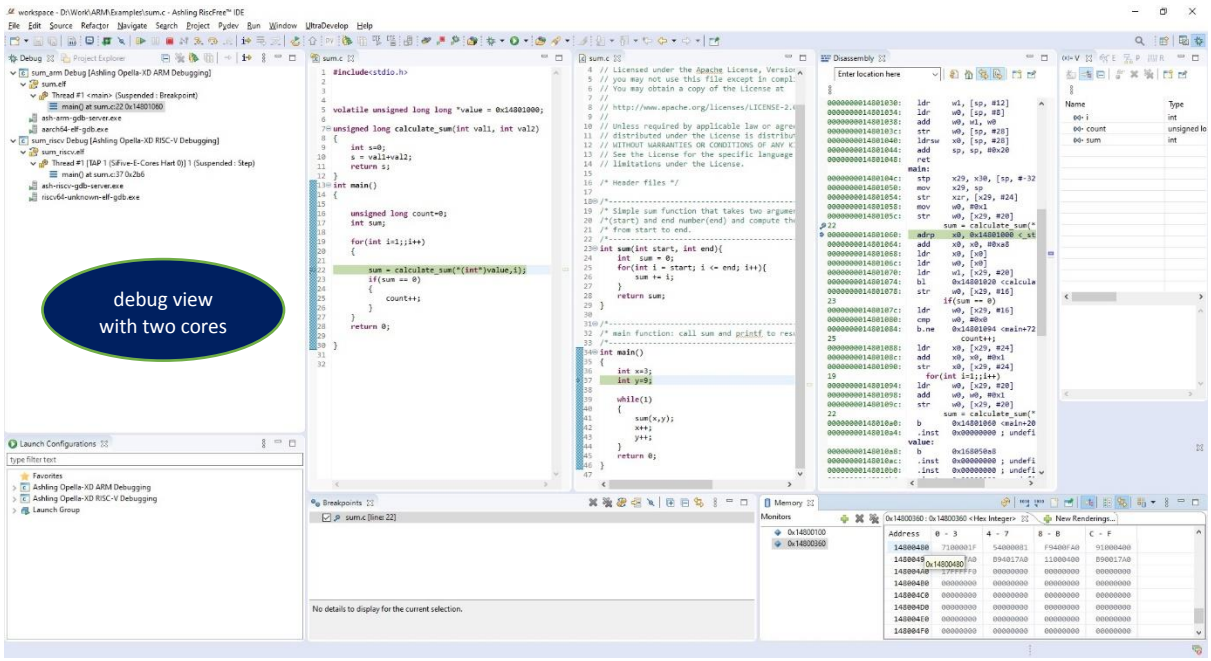

Figure 2: RiscFree IDE for Heterogeneous Multicore Debug

RiscFree includes:

- IDE based on Eclipse
- GCC and LLVM compiler toolchains with full build-tool support for the selected architectures
- Full-featured heterogeneous multicore debugger for use with Ashling's Opella-XD hardware debug probe

RiscFree has a single-shot installer which automatically configures all the component tools to work “out-of-the-box” and provides a comprehensive debug environment for both homogeneous and heterogeneous debug configurations.

Code Build in a Single Environment

For heterogeneous multicore applications separate compiler toolchains are required for each of the different architectures in the device. All the required compiler toolchains can be integrated into the unique IDE. Build parameters for each architecture can easily be configured from within a single IDE to allow easy iterations of the build process for each of the target architectures.

The next section describes some of the specific features required for debug on a heterogeneous SoC. These are required in addition to basic debug features such as ROM or RAM based run-time debug operations (run, stop, step), hardware and software breakpoints & watchpoints, RTOS debug awareness, high level register viewers, Integrated RXTX serial terminal as well as on-chip and off-chip real-time trace support.

Heterogeneous Multicore Debugging

Debug configurations – manual and automatic debug launches

Once the code has been built for each target architecture, debugging in a multicore environment may require hardware debug connections to multiple cores simultaneously. This can be done manually or automatically. The software allows the user to automate the sequence of debug connection launches such that the required connections can be completed in an automated fashion. A group launch option provides this facility which includes easily modifiable configuration options to ensure the debugger connects to the required cores.

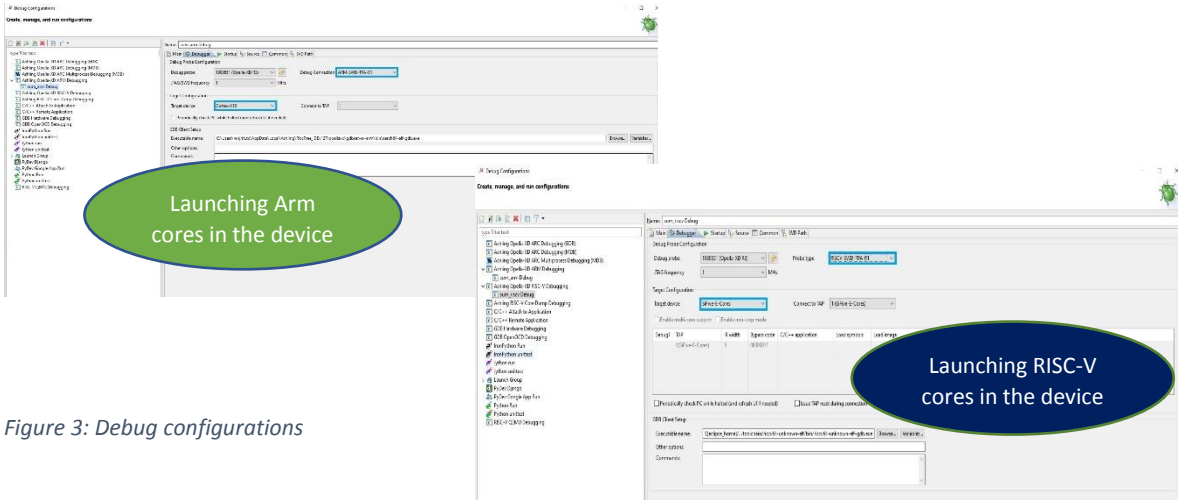

Figure 3: Debug configurations

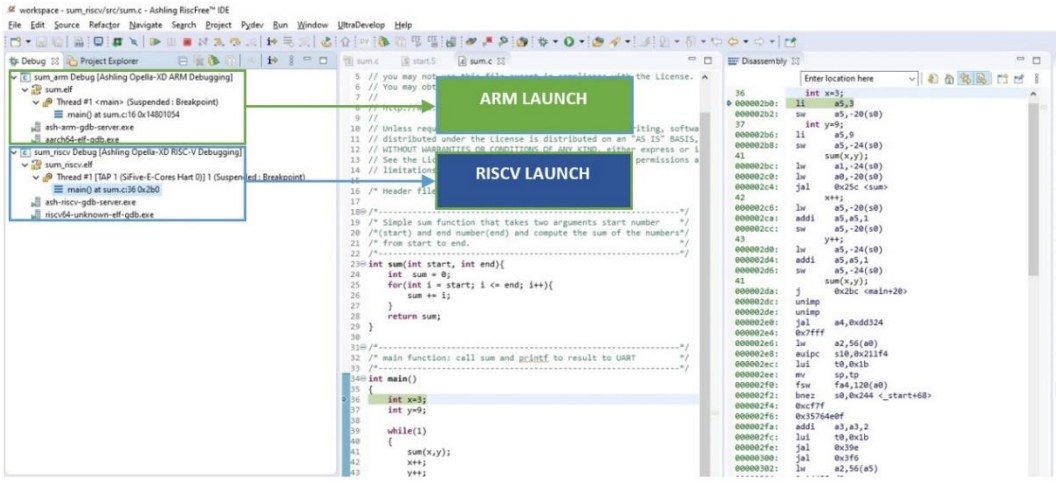

Figure 4: Multiple Debug connections

Pin & clone

During multicore debugging sequences, it may be necessary to compare data from multiple cores or threads. To facilitate this requirement, a *pin & clone* facility allows multiple debug view instances (e.g. variables, registers, memory, disassembly etc.), within the same debug session, to be attached to specific debug contexts. Each of the pinned views has a pin colour indicator matching a corresponding colour indicator reflected on the debug connection in the debug view. This enables simultaneous viewing of the data related to multiple debug contexts (cores, processes or threads).

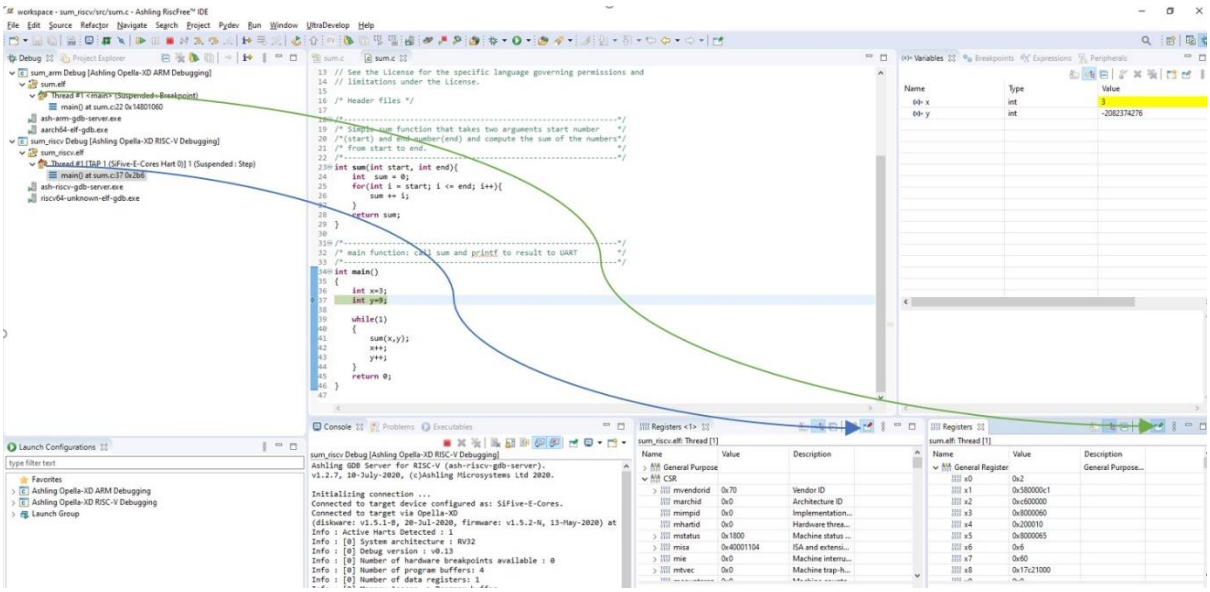

Figure 5: Pin & Clone

SoC wide debugging

When debugging code associated with multiple cores, breakpoints can be configured to halt all cores or alternatively to halt only the core which is executing the code associated with the specific

breakpoint. If all the cores are halted, then the debugger must also have the facility to resume execution for all cores or only the selected core. ***RiscFree*** include both features: group and individual go-halt selection and single core/SoC-wide breakpoints setups.

Additional debugger features

The use of scripting is also a necessary feature for large complex applications, such as heterogeneous multicore applications, which may require extensive debug environment setup and testing sequences.

For complex heterogeneous debug sequences, advanced features such as real time tracing, profiling and code coverage analysis may be used. These can be combined with SoC-wide debugging and pin & clone features for maximum debug visibility across all cores.

Conclusion

In recent years there has been a dramatic increase in the number of heterogeneous multicore designs for embedded devices and SoCs. In order to efficiently debug the applications running on these devices the debug solutions have evolved to provide SoC-wide visibility and debug control. The debug tools for heterogeneous multicore devices must support multiple different embedded architectures in addition to providing additional debug features and automation of complex multicore debug sequences. Ashling's RiscFree IDE includes a comprehensive, semiconductor vendor-independent, heterogeneous multicore debugger suitable for these complex debug requirements.