Unlocking the potential of RISC-V with HW/SW co-design

Roddy Urquhart[1]

[1]Technical marketing, Codasip

Abstract

The RISC-V architecture was created to cover a wide range of applications. With a good base integer set, optional standard extensions and a defined approach to custom instructions the RISC-V ISA is well equipped to handle an enormous variety of computational tasks. To date, the majority of the R & D effort into RISC-V cores has been focused on essentially replacing well known cores from legacy proprietary architectures. With semiconductor scaling slowing if not failing, the main way to achieve improved performance efficiently is with architectural innovation. A wide range of specialized applications is well-suited to the RISC-V ISA but require specialized processor cores. To meet ongoing demands a larger number of custom processor cores are needed but there are a limited number of processor design engineers. This demand can be met by both reducing the design cycle through processor design automation technology and by using existing RISC-V processor cores as a starting point for customization.

End of the road for traditional processor design

Electronic products have demanded ever increasing processing power to be competitive. For decades, Moore's Law [1] predicted ever denser circuits by moving to successively smaller silicon geometries. This relied on the scaling of transistor sizes and voltages to ensure that power density remained constant from one silicon geometry to the next. This Dennard Scaling [2] was the mechanism by which Gordon Moore's predictions were achieved. As Karl Rupp's 50 Years of Microprocessor Trend Data [3] shows, maximum clock frequencies for microprocessors have been level since the late 2000s.

General purpose and increasing performance: A current trend in RISC-V

With the growth of interest in RISC-V and a growing desire of SoC designers to move away from proprietary commercial ISAs, much of the RISC-V core development has focused on replacing existing off-the-shelf processor cores. This was initially focused on embedded cores moving up to application processors. A growing choice of RISC-V application processors is available supporting rich operating systems such as Linux.

RISC-V has opened up alternatives to traditional X86 and Arm choices but is this the only application for RISC-V?

Custom compute: A major opportunity for RISC-V

In their 2018 Turing Lecture, John Hennessey and David Patterson described how improvements to single thread processor performance had declined to a depressing 3% per year [4].

However, they saw the way forward as being architectural innovation. Instead of relying on standard processor cores they advocated innovating with domain specific architectures which would tackle a few tasks and do them extremely well. To design such architectures, they recommended using HW/SW co-optimization, open instruction set architectures (ISAs) and agile design.

Tailoring hardware to a software workload requires innovation with ISAs or microarchitecture or both. In the past, developing a custom ISA required a set of skills that many companies could not provide.

RISC-V massively reduces the threshold to creating a custom ISA by taking care of necessary basic instructions in a standard way. The base integer set is minimalist but lays a foundation for a software ecosystem.

Barriers to implementing custom compute

Creating more specialized and capable processor cores faces some challenges. Inevitably if there is a move away from off-the-shelf cores to custom ones there will be many and varied designs to be implemented. Such designs will have varied instruction sets, microarchitectures and toolchains.

Today processor designers are a relatively small community within the semiconductor industry. This community alone will not be able to increase the number of processor designs developed by simply continuing to use traditional RTL design methods.

The main way in which this dilemma can be resolved is to reduce design time and effort. Driving down design times can be achieved in two ways. Firstly, if traditional manual RTL design development is abandoned in favor of processor design automation. Secondly, if an existing core comes reasonably close enough to meeting the needs of a software

workload it can be used as a starting point for the custom core development. The design work can be incremental rather than a clean slate development.

Automating HW/SW co-design

With custom compute, the goal is to develop cores that efficiently handle their software workloads. An efficient implementation will not only deliver performance specialization benefits silicon area and power consumption.

The starting point for a custom core is analyzing the software workload. If an initial instruction set is modeled, the software workload can be profiled, and computational hotspots identified. The hotspots can be addressed by defining custom instructions that simplify the computation.

The most effective way of modeling the ISA is to use a complete architectural description language (ADL) such as Codasip's CodAL. It is easy to modify the ADL source code and then to automatically generate the software toolchain. The ISA can be quickly and iteratively analyzed and improved.

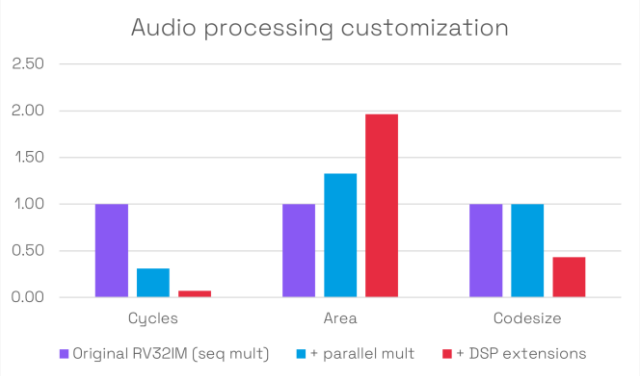
An ADL also allows microarchitecture to be described and refined. For example, if a processor needs to process a stream of data such as for a video stream it may be valuable to add registers and arithmetic units outside the main core. Such additional hardware can be described in the ADL and then used to generate an RTL implementation. The RTL can be analyzed and simulated and, if necessary, the ADL code describing the microarchitecture can be refined.

Examples of custom compute solutions

To illustrate creating custom RISC-V cores using an ADL and processor design automation, we show two examples.

In the first case, a company was developing an audio processing SoC with an echo-cancelling algorithm being the dominant computation. They wanted to consider RISC-V as an alternative to a previously used Cortex-M core.

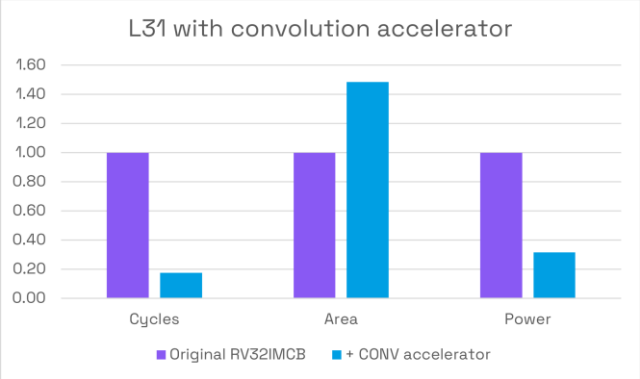
They used a minimal RISC-V configuration of a 32-bit L30 core with a 3-stage pipeline. The echo-cancelling software was profiled using Codasip Studio and the initial cycle count was too large. With multiplication being intensively used the first step was to replace the sequential multiplier with a parallel one. This too resulted in too large a cycle count.



Then experiments were undertaken to add over 40 DSP custom instructions. This resulted in a core that ran the algorithm 14.3x faster than the original. The core was almost double the initial area but the codesize was 43% of the original.

In a second example, image recognition was undertaken using a convolutional neural network (CNN) on a Codasip L31 embedded core [5].

Again, Codasip Studio was used to profile the CNN algorithm and identified the image convolution operation as a computational hotspot. The microarchitecture was extended to include a FIFO register chain for incoming pixels and the ALU was modified to perform parallel multiplications of the image pixels by the convolution weights and to sum up the result.



The modified L31 is 48.3% larger in area but has a throughput improvement of 5x and power improvement of 3x compared with the original L31.

References

[1] G. E. Moore, "Progress in Digital Integrated Electronics." Technical Digest 1975. International Electron Devices Meeting, IEEE, 1975, pp. 11-13.

[2] R. H. Dennard, F. Gaensslen, Hwa-Nien Yu, L. Rideout, E. Bassous, A. LeBlanc, (1974). "Design of ion-implanted MOSFET's with very small physical dimensions". IEEE Journal of Solid-State Circuits. SC-9 (5): 256–268

[3] K. Rupp, 50 Years of Microprocessor Trend Data, retrieved 8 March 2023

[4] J. Hennessey, D. Patterson, "A New Golden Age for Computer Architecture: Domain-Specific Hardware/Software Co-Design, Enhanced Security, Open Instruction Sets, and Agile Chip Development", The 45th International Symposium on Computer Architecture - ISCA 2018, retrieved 9 March 2023

[5] A. Shchekin, "Compact neural network accelerator in CodAL – Case study", retrieved 2 May 2023