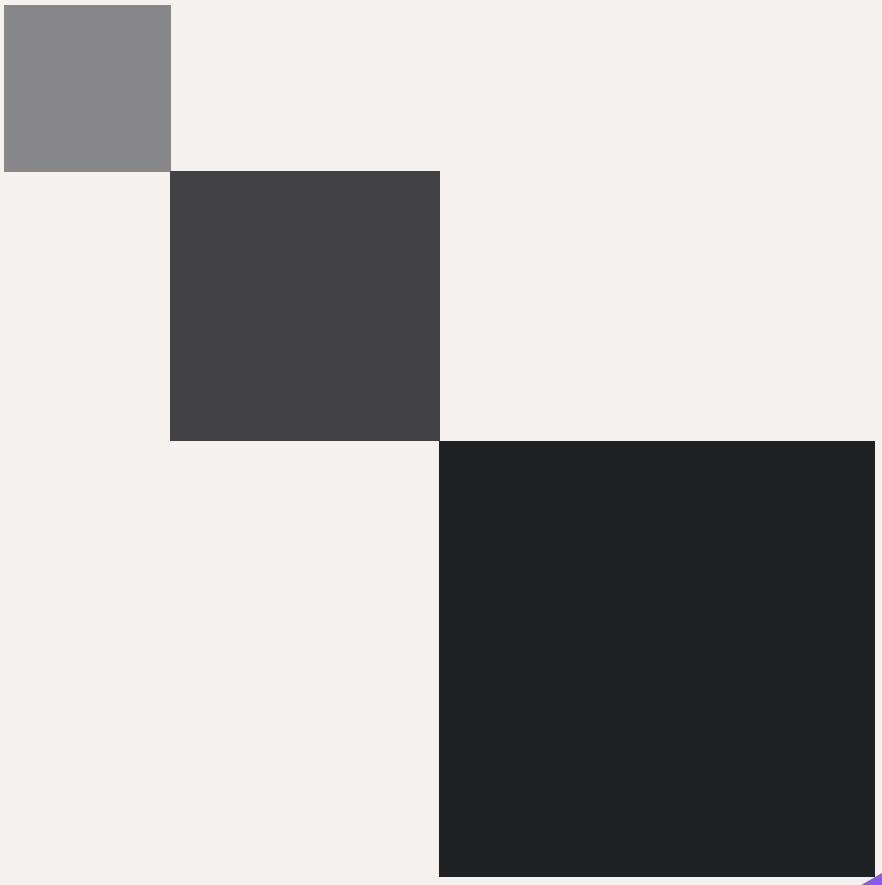
Codasip



→
Product
overview

↘

# Architect your ambition

**We can offer you:**

- **A range of market leading RISC-V IP cores including certification for functional safety and cybersecurity**
- **An EDA processor design tool enabling multiple levels of customization and automated generation of associated models, SDK and verification environments**
- **The first commercially available processor built using CHERI, providing memory protection against the most common form of cyber attacks**

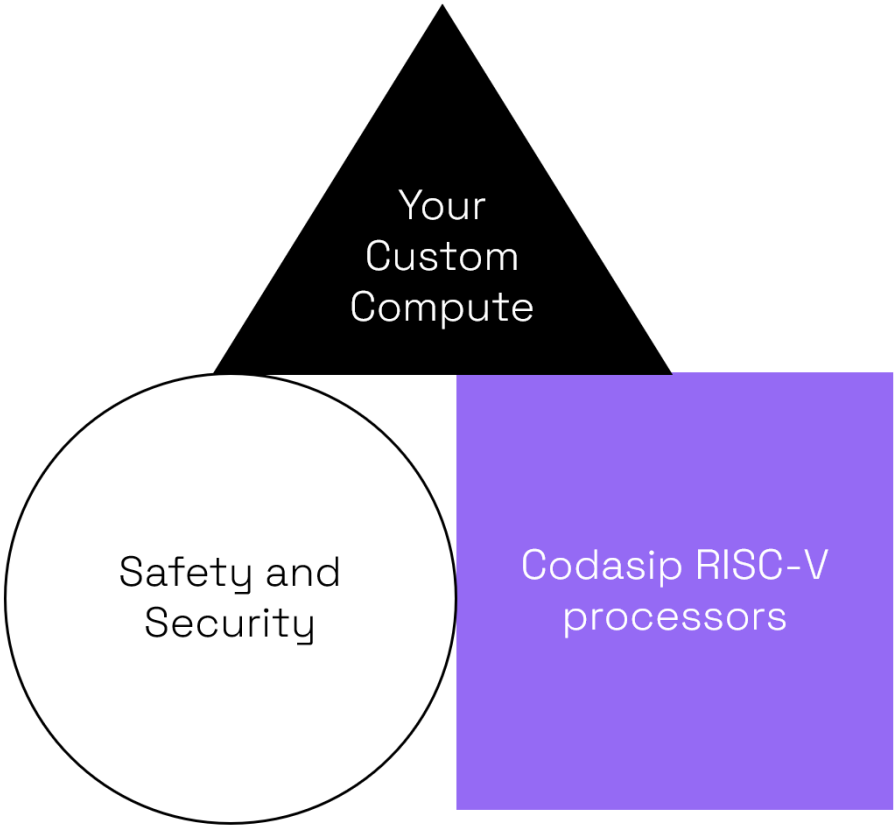
# Stand out with Custom Compute

In today's technology market, the difference between success and failure is often in the quality of your silicon design and increasingly in meeting security and safety criteria.

Semiconductor scaling and legacy processors can no longer be the basis for achieving outstanding SoC designs. Instead, tailoring designs to your application with Custom Compute is the foundation for achieving design goals and differentiation.

Codasip is a processor solutions company which uniquely helps developers to differentiate their products. We are Europe's leading RISC-V company with a global presence. Billions of chips already use our technology in applications including mobile phone cameras, artificial intelligence, video systems, display controllers and home automation.

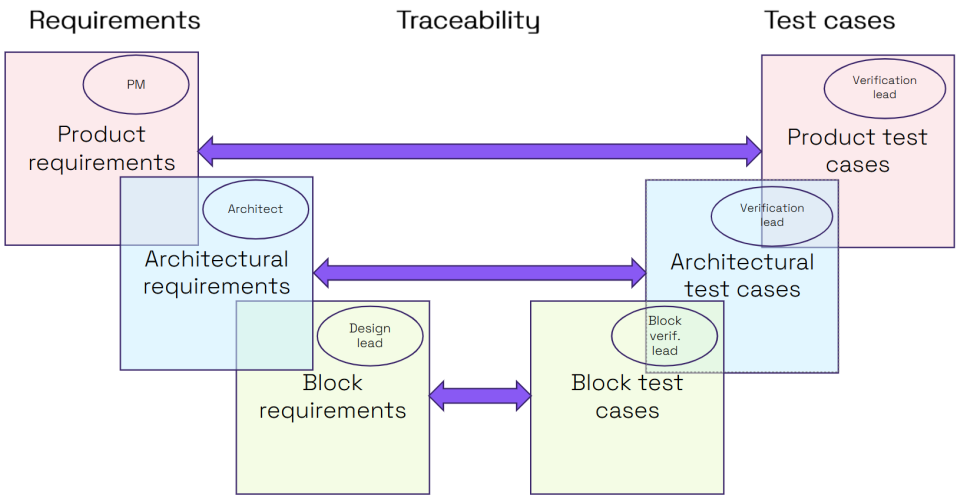
We deliver Custom Compute through the combination of the Codasip Studio processor design automation tools and high-quality RISC-V processor IP. Our innovative approach lets you easily customize and differentiate your designs. You can develop high-performing, and game-changing products that are truly transformational.



# Requirements-driven mindset for traceability and verification

Our IP design team consists of over 100 engineers who span the key disciplines needed to create processor IP cores. The disciplines include CPU architecture, modelling, design, silicon implementation and verification. There is a robust development process with rigorous multi-layer verification ensuring that we deliver high quality cores with competitive PPA (Power, Performance, and Area).

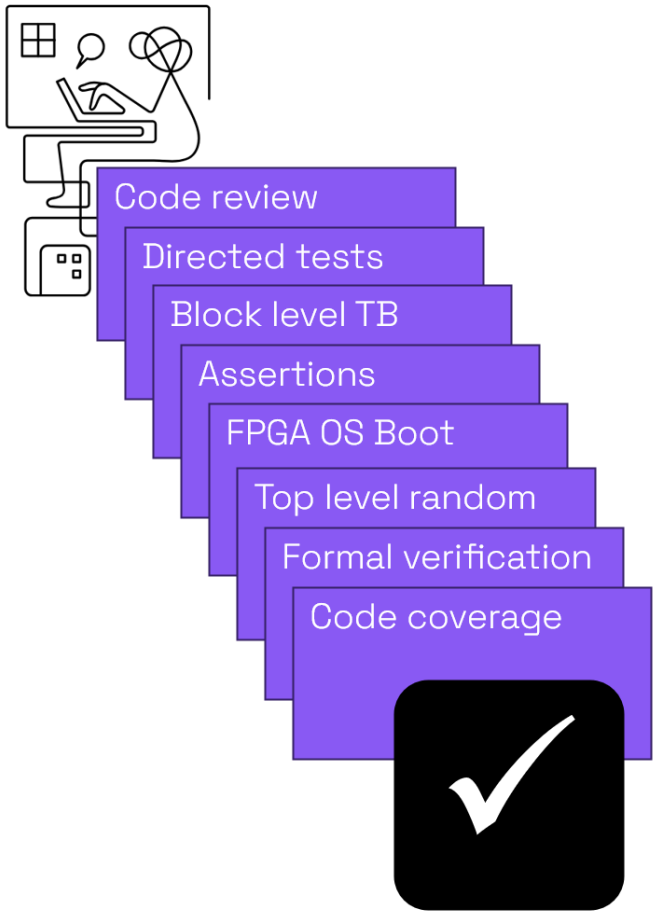
A key to ensuring processor quality is a requirements-driven mindset. It is necessary to assign staff to both design and verification through the development cycle. There needs to be traceability of requirements and test cases from the product definition, through architecture to the block level.




Traceability though the design cycle

Processors have a complex state space so are more difficult to verify than most other types of RTL design. Therefore, we have adopted the multi-layer or “Swiss cheese” approach to verification – adopted from the world of aviation. The idea is that no one verification method can catch all bugs and that multiple verification layers are needed.

For processor IP, we use a combination of in-house and third-party tools to verify the cores.



# Industry standards

The RISC-V instruction set architecture (ISA) is revolutionary by being an open, modular standard that can be adapted to the needs of your application. It provides a compact base integer instruction set, a choice of standard extensions and the option of custom instructions. Codasip is a founding member of RISC-V International and we play an active part in the RISC-V community.


Our design processes also embrace the requirements for safety and security. TÜV SÜD has certified our IP hardware engineering development processes, guidelines, and work instructions according to the functional safety standard ISO 26262 and the cybersecurity engineering standard ISO/SAE 21434. This provides a solid foundation for OEMs developing safety and security products based on our processor IP.

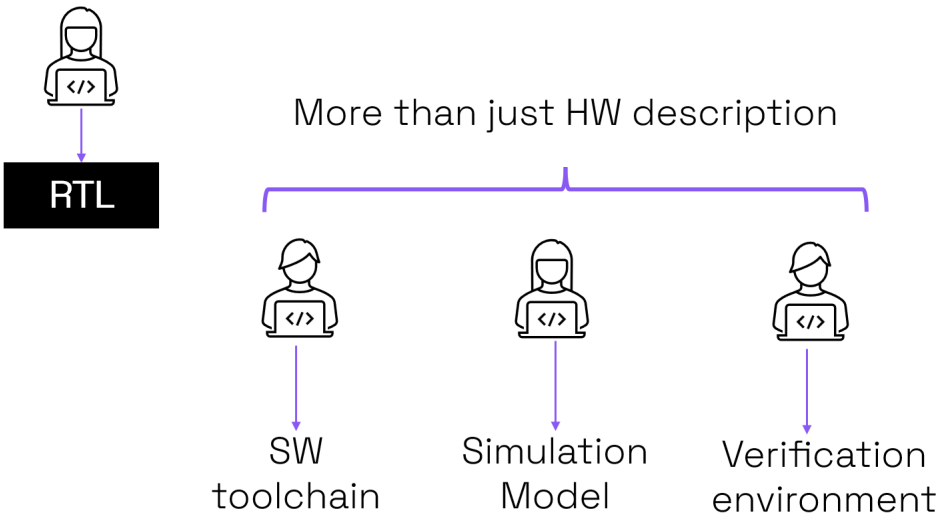
Outputs from our processor design tool Codasip Studio include using the following industry standards:

- System Verilog for RTL
- LLVM for the compiler
- System C for simulation models
- UVM for the verification environment

# Architectural language and processor design automation

It may be tempting to think that with the RISC-V ISA already defined, it makes sense for you to develop the microarchitecture using a traditional RTL approach. But unlike other parts of an SoC, a processor core executes software, therefore it is essential to cover both hardware and software aspects of your core design.

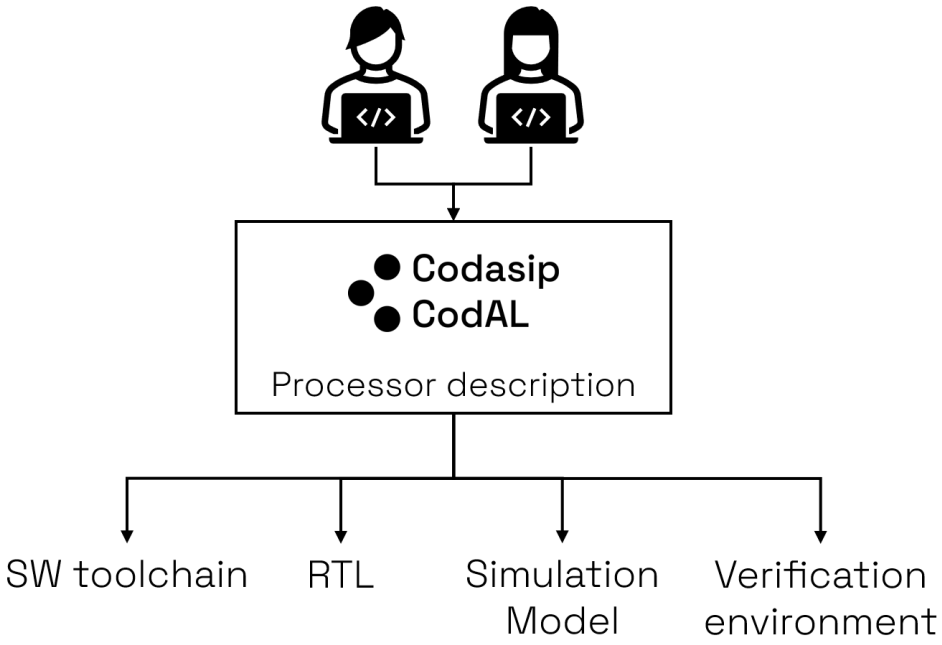
If you describe your core with a hardware description language (HDL), whether an older one like Verilog or a newer one like Chisel, all you can design is hardware. For common configurations on the RISC-V instruction set like RV32IMC, there are open-source software toolchains and instruction set simulators (ISS) available, but they are likely to be optimized for a different microarchitecture to yours.




Traditional processor design means not only manual RTL creation but also manually developing the SW toolchain, simulation models and verification environment

As soon as you need to use custom instructions, it is necessary to manually change the toolchain or ISS. Another area for manual work is creating a verification environment to check that the microarchitecture is consistent with the instruction accurate description. These manual actions – often undertaken in parallel – add significant technical risk to your project.

Given the limitations of traditional design methods, we have taken a holistic approach to all aspects of processor design from the outset. The use of an architectural language and processor design automation is foundational to our processor solutions.



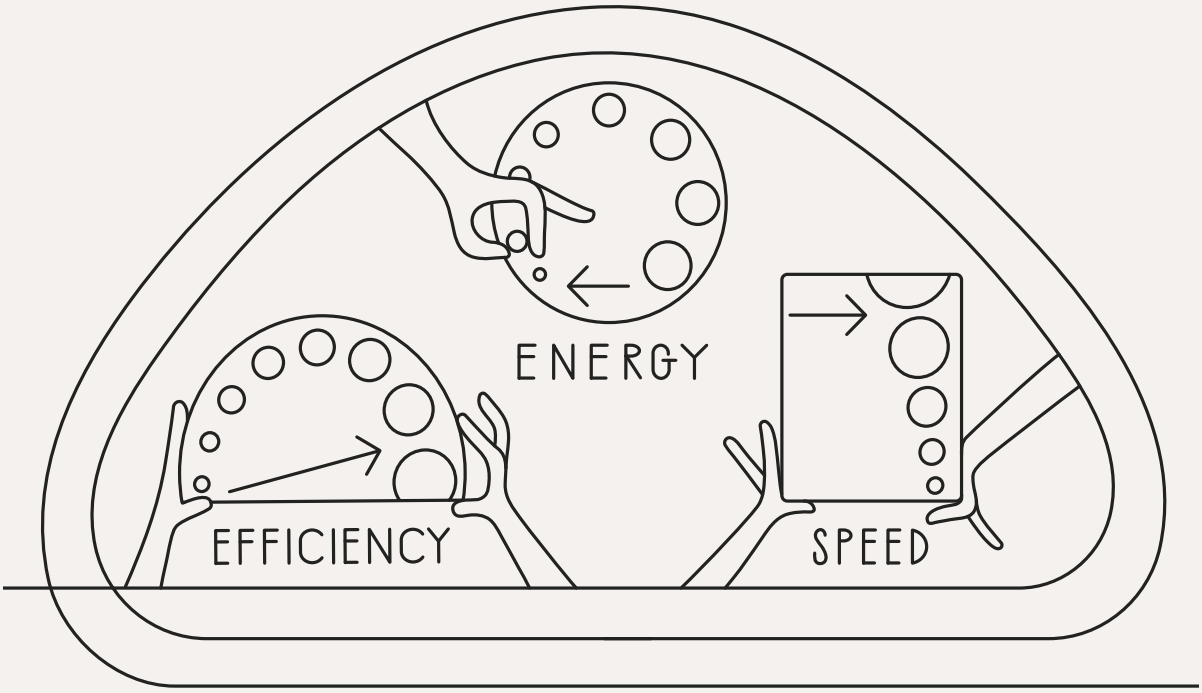

A single CodAL description covers the software toolchain and verification as well as hardware.

If you use the CodAL architectural language, then there is a complete processor description capable of supporting software, hardware and verification aspects. If you implement custom instructions by adding to the CodAL source, then these additions can be reflected in the software toolchain and verification environment as well as the RTL. The description acts as a single source of truth that is consistent across different views of the processor.

Our processor design automation toolset – Codasip Studio – supports the development of CodAL as well as automatically generating the software toolchain, RTL and verification environment from the CodAL source.

# Codasip Studio

## A complete processor design automation toolset



# Our technology foundation

Codasip Studio is a complete design environment for processor design automation and customization. Whether you need to create a new core or customize an existing one, Codasip Studio makes the task faster and easier, with reliable results.

Codasip Studio employs a revolutionary approach. One single high-level CodAL description of the processor replaces the multiple manual tasks of writing the RTL, adding any custom instructions, updating the software toolchain and preparing for verification. Unlike some alternative tools, Codasip Studio generates the design implementation, verification environment, virtual system prototype, and a complete software toolchain fully automatically. The design methodology is protected by patents, and we use it ourselves to create our Codasip processor IP.

In addition to its design capabilities, Codasip Studio includes powerful multiprocessor programming, debugging, and profiling features, enabling more complex processors to be designed with ease. We have built Codasip Studio upon open standards and software including Visual Studio Code, LLVM, SystemVerilog, and UVM.

## Straightforward development process

A unique aspect of our approach is to automate the development of processor cores by using the CodAL architectural language. The capabilities of your processor only need to be described once in CodAL, and from this single description everything needed to design, integrate, and program the processor is automatically generated.

CodAL is a highly structured, hierarchical, C-like language for processor architecture description and design. The language can describe a wide range of processor styles including RISC, CISC, VLIW and DSP.

In addition to processor descriptions, CodAL supports co-simulation with existing models as well as native modelling of multiprocessor subsystems including multiple cores, interconnect, and memory structures.

# Match the hardware to your software

With Codasip Studio you have the possibility to adapt your hardware to your software workload, making it an ideal tool for creating your Custom Compute solution.

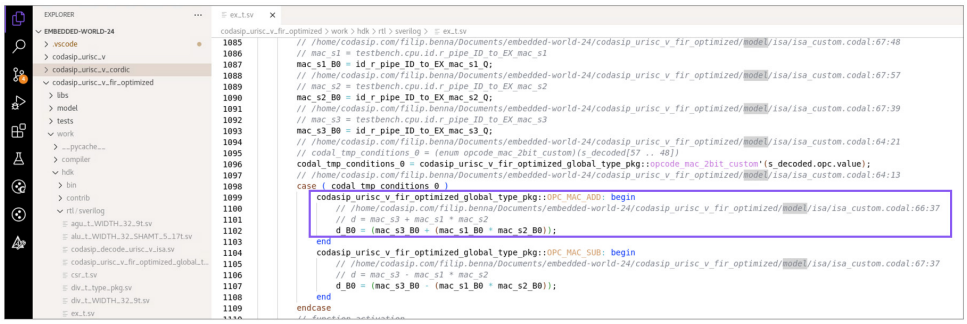
If your starting point is an existing Codasip RISC-V core described in CodAL, then you have immediate access to an SDK allowing development, debug, and execution on the target platform well ahead of silicon availability.

The Codasip Studio profiler will enable you to identify “hotspots” where certain parts of the code are very intensely used. Once you have identified the hotspots, you can define additional custom instructions to improve the performance. You can then generate a new SDK and re-profile your software with the new target instruction set to assess whether the changes resulted in adequate improvements.

The key to the effectiveness of our Codasip Studio solution is its natural iterative approach to your design. Codasip Studio gives you a fast and simple way to move from your algorithm to an instruction set customized to your workload.

When you are satisfied with your instruction set you can move on to microarchitecture and creating a cycle-accurate (CA) description in CodAL. From the CA description, Codasip Studio enables you to generate a cycle accurate simulator, RTL, a testbench and verification framework. The microarchitecture development can also be an iterative process enabling you to converge on a good design. Codasip Studio’s powerful high-level processor synthesis technology allows you to generate processors that are competitive.

The generated RTL is human readable which is an advantage over some competing tools. Also, when doing co-simulation, you can trace the RTL back to the corresponding CodAL source code enabling you to debug the source efficiently.




Generated RTL code links back to source CodAL code

# Automatically generated HDK & SDK

The generated outputs from Studio are summarised below:

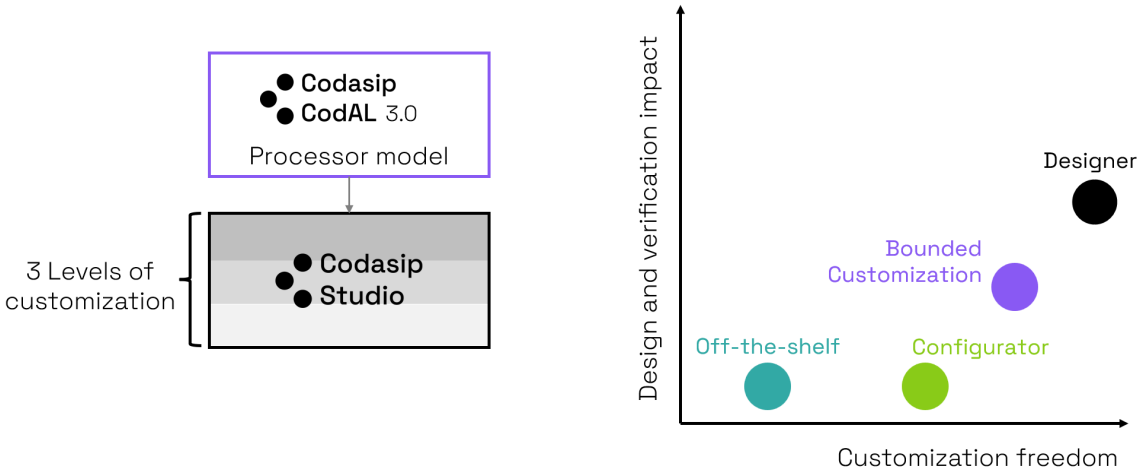
| SDK | HDK |
|---|---|
| **LLVM C/C++ compiler** | **SystemVerilog RTL** |
| **C/C++ libraries** | **Verification framework** |
| **Assembler, disassembler, linker** | **SystemC co-simulation model** |
| **High-performance instruction set and cycle accurate simulation models** | **Sample EDA scripts** |
| **Debugger and profiler** | |
| **Random instruction generator** | |

Whether customizing an existing RISC-V processor (or processors) for your design or creating one from scratch, you will want to be assured that it has been adequately verified. Codasip Studio's verification methodology combines a standardized approach, using simulation and static checking. It provides a random assembler program generation and an automatically generated verification framework that consists of components automatically adapted by Codasip Studio to support your custom instructions.

Another application for Codasip Studio's design automation is to model the ISA of your legacy proprietary processors in CodAL. Your SDKs can be automatically generated greatly simplifying the maintenance of the software toolchain.

# Levels of customization in Codasip Studio

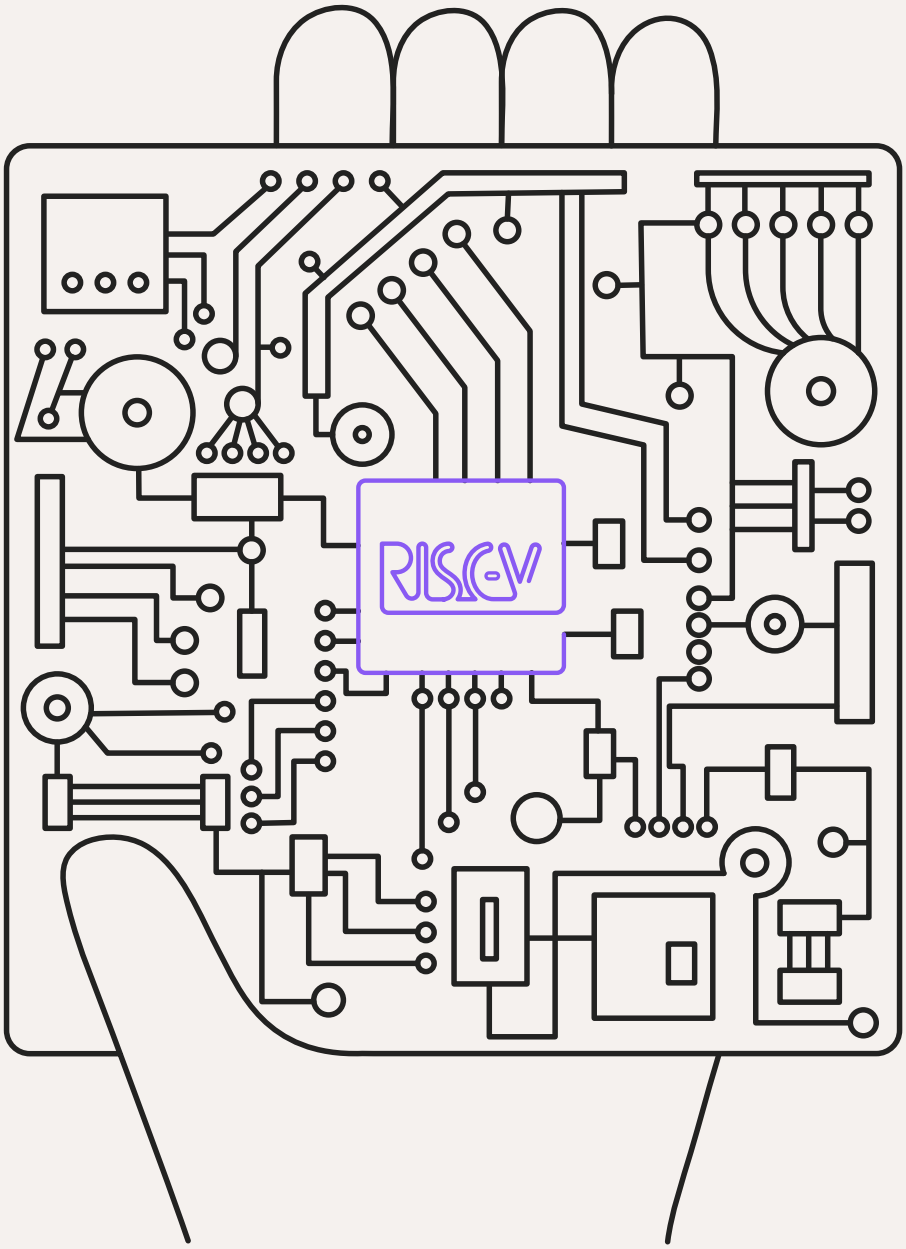
Design projects vary considerably in their need to customize processor cores. Codasip Studio supports varying degrees of customization and design/verification complexity.




The Codasip Studio Configurator allows you to choose from a menu of options for simple configuration of our cores. If you want to add custom instructions, you can do that safely thanks to Bounded Customization. The functionality of the baseline core is guaranteed in all cases via defined bounds and formal methods and verified through a straightforward verification approach. By using Codasip Studio Designer, you get the freedom to create your own processor core from scratch.

See the section ‘Your Custom Compute core’’ for more details.

# Codasip RISC-V Processors

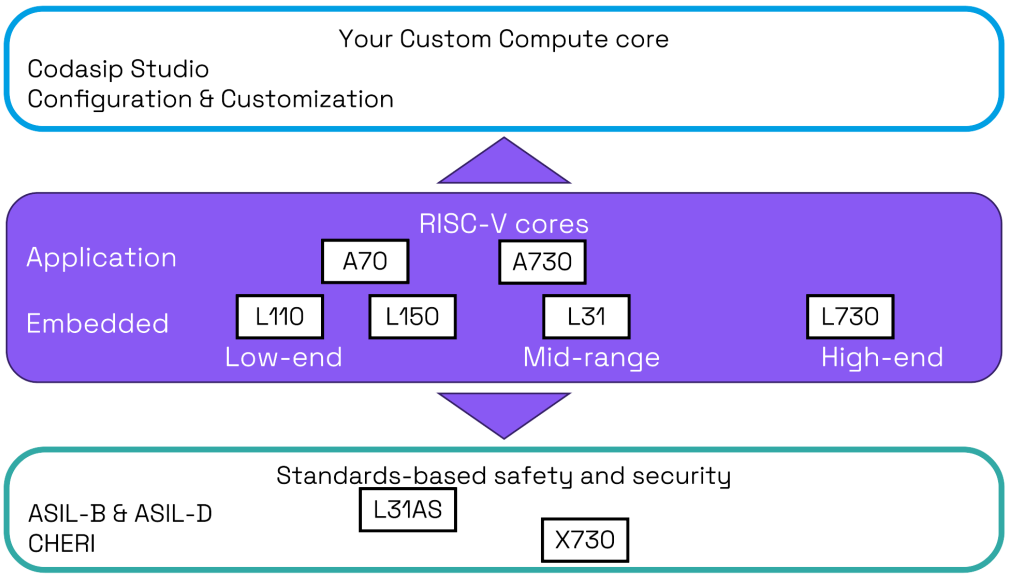


# Match a Codasip core to your project needs

RISC-V is the cornerstone of our processor families. By using our cores, you can take advantage of this open, royalty-free ISA with the following benefits:

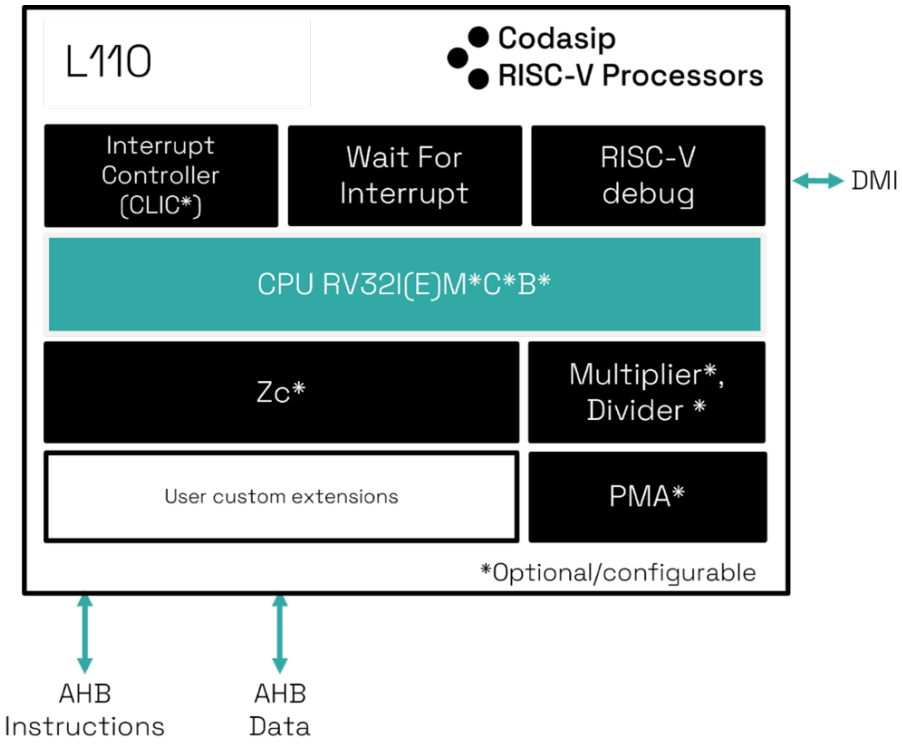
- Vendor independence and longevity
- Modularity and extensibility
- Custom microarchitecture

Our RISC-V cores can be configured or customized to give you a Custom Compute core optimized to your product or application. We have developed standards-based cores to meet requirements for safety and security. Our range of processors is rapidly growing with more products in the pipeline.



# L110: compact, efficient embedded processor

The L110 is aimed at low power embedded applications requiring a small silicon footprint. Examples of use cases include cryptography, automotive such as motor control, DSP and edge AI, wireless protocols and finite-state machine replacement.

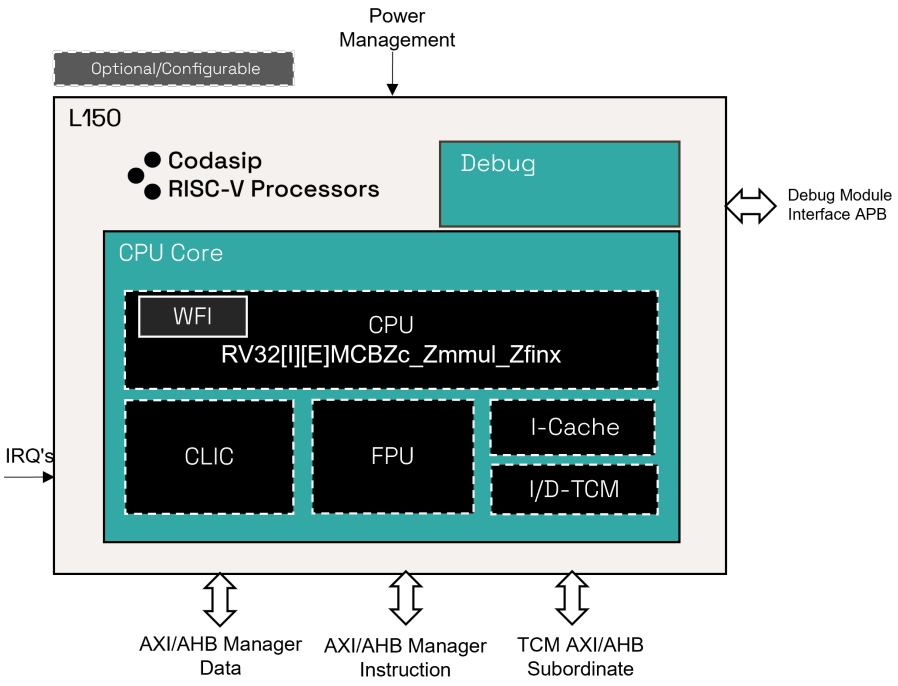



L110 is a 32-bit, highly configurable, RISC-V core with a wide range of optional RISC-V extensions as well as hardware blocks. Connectivity to the rest of the system is provided by AHB instruction and data busses. Depending on the needs of the core, arithmetic performance can be enhanced by hardware multiplier and divider options and code density can be improved through RISC-V C and Zc extensions.

Baseline L110 cores offer competitive PPA compared with other alternatives on the market. Specifically, the PPA compares favorably with the published data from the Arm Cortex M0 and M0+ as well as with alternative lower-end 32-bit RISC-V cores. For particular algorithms, Codasip Studio can enable significant performance improvements to be achieved by adding Custom Bounded Instructions.

# L150: highly configurable embedded processor for small-area and low-power applications

Codasip L150 is a 32-bit RISC-V embedded processor, focused on real-time, small-area and low-power applications.



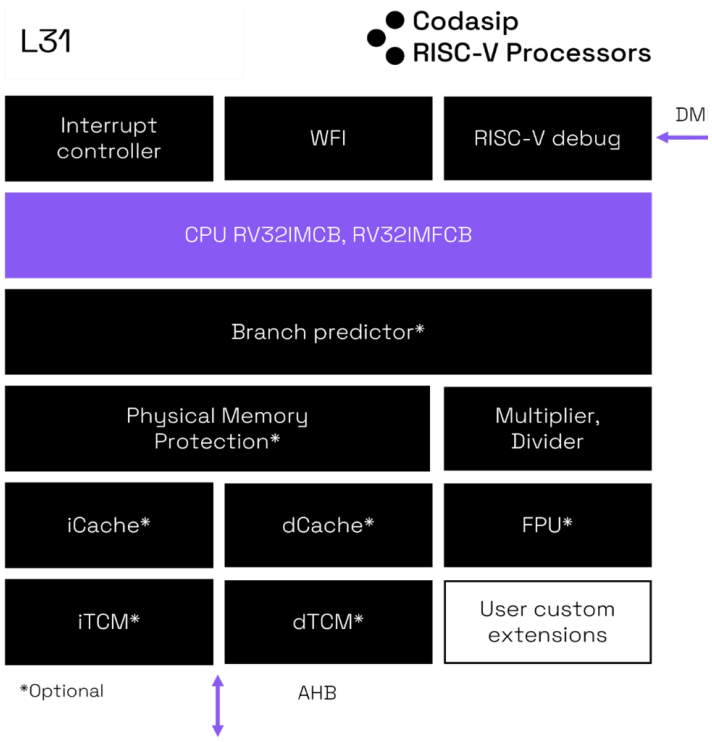

The core is highly configurable, allowing different performance points in the real time embedded space, including local memories for low-latency access, and including a tiny Floating-Point Unit (FPU) implementing Zfinx extension, as part of the RISC-V standard solution optimized for small cores.

Designed with real-time and performance-sensitive applications in mind, Codasip L150 strikes a strong balance between small area and scalable performance.

The inclusion of the Zfinx RISC-V extension, part of the RISC-V standard, enables floating-point operations using the existing integer register file. It is a compelling trade-off for the cost- and energy-effective embedded systems that still need floating-point capabilities without the burden of the entire F extension support.

# L31: flexible, embedded processor

The L31 is a medium-range 32-bit embedded processor aimed at more demanding computational applications and systems than the L110. It offers a balance between performance and power consumption, making it suitable for image processing and smart sensors. It is capable of clock frequencies of up to 770 MHz using TSMC 28 nm HPC.

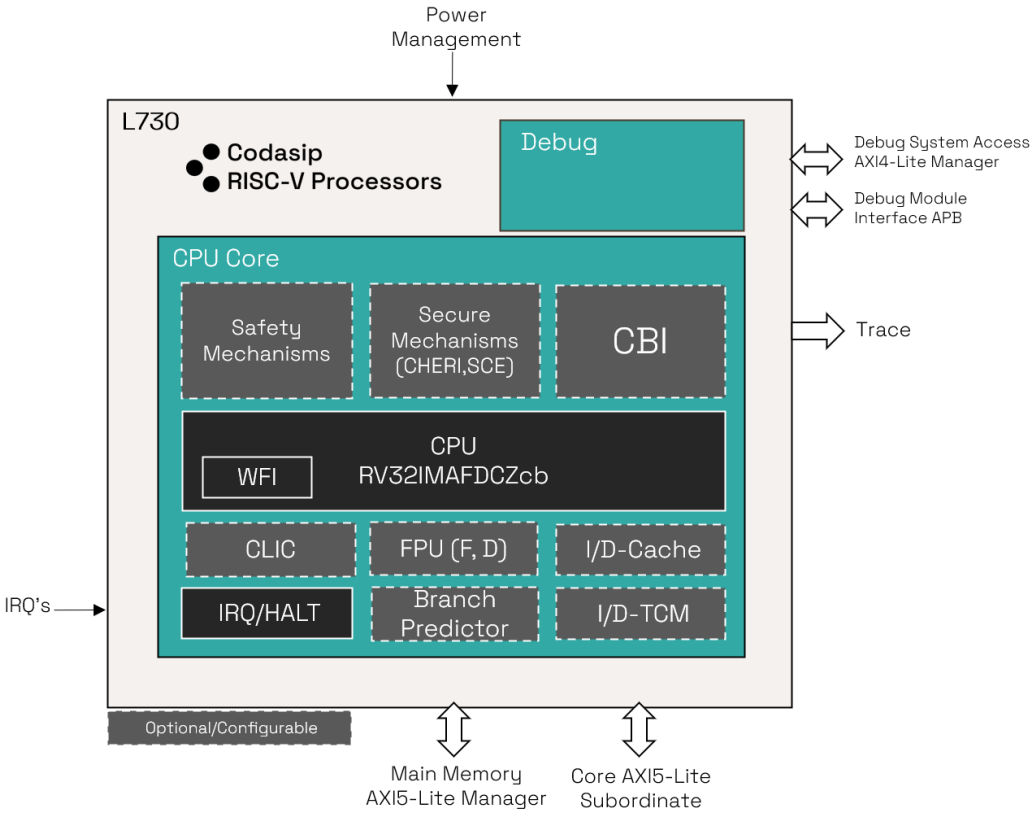



The L31 comes with a hardware multiplier and divider as standard and can optionally be configured with a hardware floating point unit. Connectivity to other system components is via an AHB bus. Optional instruction and data caches are available.

L31 has been customized for a variety of applications including FIR & median filters and 2-D convolution for neural network applications. It is suited to both bare metal and RTOS applications.

# L730: 32-bit, superscalar, high-performance embedded processor

The L730 is a high-performance, dual issue, 32-bit embedded core suitable for computationally demanding applications.



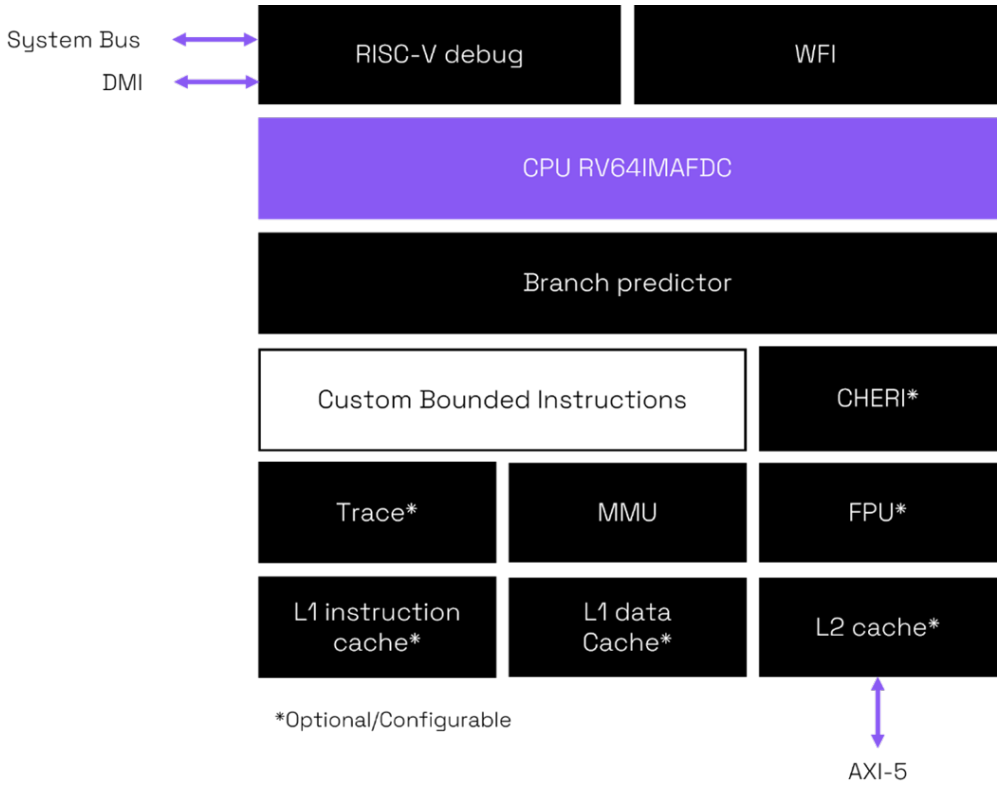

The L730 comes with the RV32IMAFDC instruction set as well as Zcb for enhanced code density. You can configure the core with optional hardware floating point unit, tightly-coupled memories, L1 instruction cache, L1 data cache and trace. Connectivity is provided with an AXI-5 64-bit bus.

Automotive variants are available including a single core designed for ASIL-B and a dual-core lockstep configuration for ASIL-D.

# A730: 64-bit, superscalar, application processor

The A730 is a competitive, in-order, dual issue, 64-bit core suitable for use with a rich operating system such as Linux. It is compliant with the RISC-V RVA22 profile to ensure excellent software compatibility. Its RV64IMAFDC instruction set supports single and double precision floating point operations.

For system connectivity, the core provides a 128-bit AXI5 bus as well as an AHB debug module interface (DMI) to connect with external debuggers.




The core can be configured with optional L1 instruction and data caches, an L2 cache, an FPU and trace. The A730 is designed to support symmetric multiprocessing (SMP) and can be configured in clusters of up to 4 cores.

A730 is the basis of the X730, where we have added CHERI technology.

→

# Your Custom Compute core

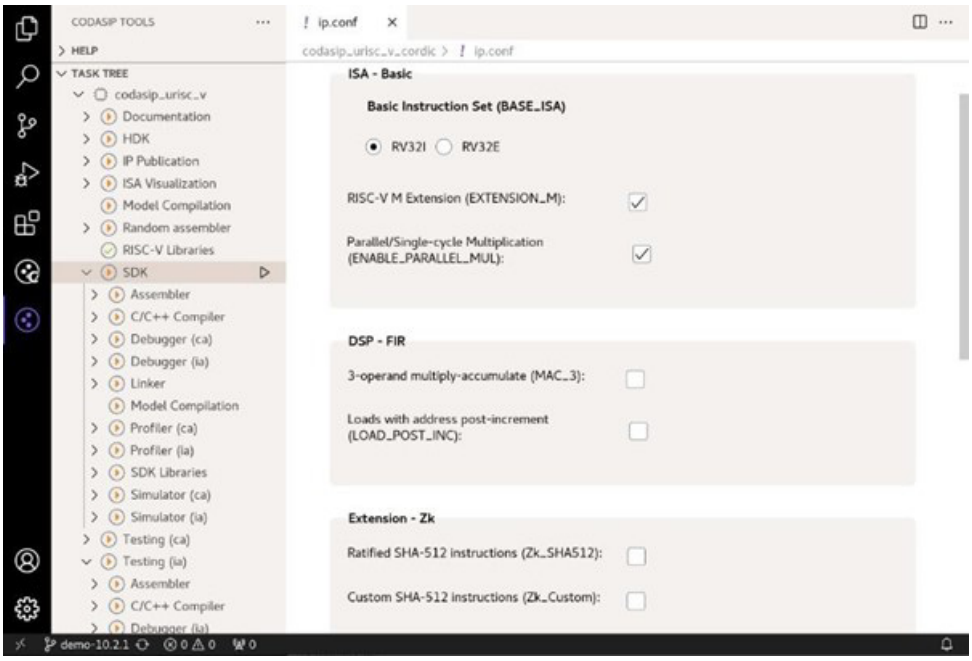
We enable you to match your hardware to your software workload and create your unique Custom Compute solution.

There are differing degrees of customization required to match a processor core to your application. The more complex the customization the greater the verification effort required. Codasip Studio offers three approaches to customization enabling you to choose which tool variant makes sense for your project.

## Configurator – fast simple, processor tuning

Codasip Studio Configurator is the entry-level offering for processor customization. The tool is delivered with a verified baseline core with a limited range of pre-verified options. The Configurator tool provides the designer with a form to choose from the available set of design choices. This means that the tool can be used by a designer who has not learned the CodAL language and is not a processor engineer.

Codasip Studio Configurator allows the designer to automatically generate an SDK to test out the chosen configuration. Once the configuration is chosen, no further verification is required as all configuration options are pre-verified.




Core configured for RV32IM with single cycle multiplier

# Bounded Customization – custom instructions accessible to all

When the performance of a processor core is enhanced by adding custom instructions, it is essential that the final processor design is adequately verified. By restricting the type of customization allowed, we ensure that:

- The customization does not harm the baseline processor design
- Creating custom instructions and verifying them is greatly simplified.

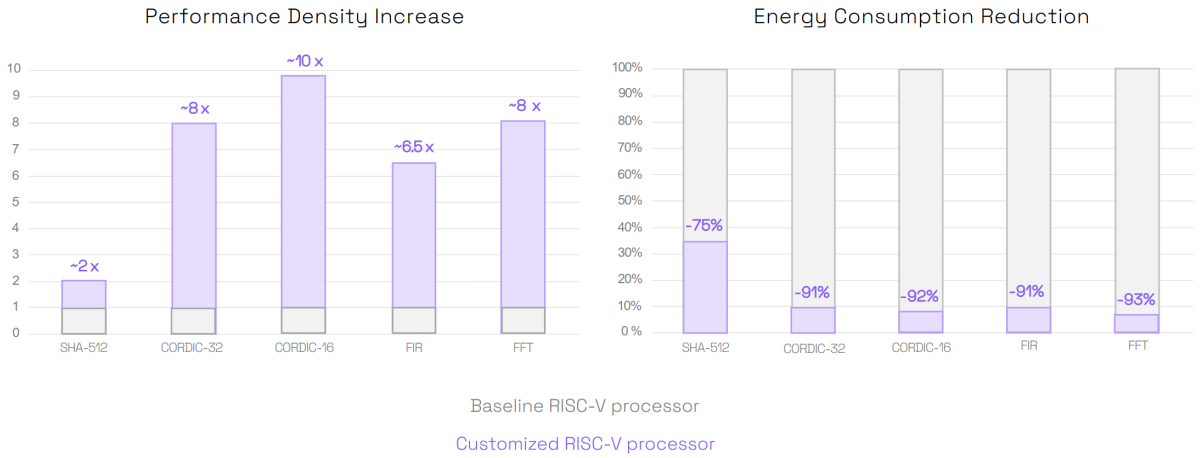
Many commonly used operations can fit the rules for Custom Bounded Instructions (CBI). These include common arithmetic operations, multi-cycle operations such as CRC, DSP functions, trigonometry, error correction and cryptography.

While configuration options such as adding a hardware multiplier or adding Zk extensions can improve performance and code density respectively by a few percent, custom instructions can have a much larger impact on performance, power and area.  Bounded Customization provides you with an accessible, safe and reliable way to add such instructions.

Benefits include:

- Fast time to market for custom instructions
- Many common optimizations possible
- Straightforward verification of custom instructions
- No need to learn the full CodAL language

Codasip Studio also provides an easy-to-use verification framework, which supports the verification of RTL against a reference model using a random instruction generator. This verification of new custom instructions is considerably simpler than full processor verification.




Codasip Studio allows you to experiment with different approaches for any algorithm. The figure above shows how PPA can be improved for a variety of cryptographic, trigonometric and DSP algorithms. In the case of SHA-512, performance is almost doubled and energy almost halved for just a 2.2% area increase. In the case of the FFT, an area increase of 71% yields about 14X improvement in performance and energy reduction.

## Designer – use the full power of the CodAL language

Codasip Studio Designer offers unbounded customization. Using the Designer, you may make changes to any part of the design. For example, it is possible to experiment with microarchitectural details of the core such as pipeline length.

The Designer is not only suitable for making deep changes to the architecture and microarchitecture of our RISC-V cores but is well-suited to creating new cores from scratch. CodAL is a highly structured, hierarchical C-like language for processor description and design. It can describe a wide range of processor styles including RISC, CISC, VLIW and DSP.

Using Designer requires a deep knowledge of processor design and verification as well as familiarity with the latest version of CodAL architectural language.

# Safety & Security

→

# Standards-based safety and security

Safety and security measures are mandatory in some applications such as automotive, aerospace and defense. In these cases, there are industry standards that define the requirements for products. However, in an increasingly connected world, it is important to plan for safety and security even in areas where there are no mandatory standards.

## Functional safety-certified products

A common framework has been developed to address safety challenges in vehicular E/E systems namely ISO 26262. ISO 26262 uses a qualitative risk classification called ASIL, 'Automotive Safety Integrity Level', where risks are classified from ASIL A to ASIL D, with ASIL D being highest. An analogous framework for cybersecurity is ISO 21434.

While functional safety is a vehicle-level attribute, in practice complex E/E systems involve components from a wide range of suppliers. ISO 26262 has a significant focus to ensure traceability of safety requirements, development rigor, integration, and deployment.

Processors are at the core of most E/E systems whether running safety-critical application code or implementing a dedicated safety function.

The key assumptions are the use cases where the element will be deployed, the safety related failure modes and the resultant safety requirements that the processor IP core needs to achieve. This includes the target ASIL for the core.

We undertake third-party certification to provide added assurance of our cores' reliability in safety-critical applications.

# L31AS: ISO 26262-certified embedded processor

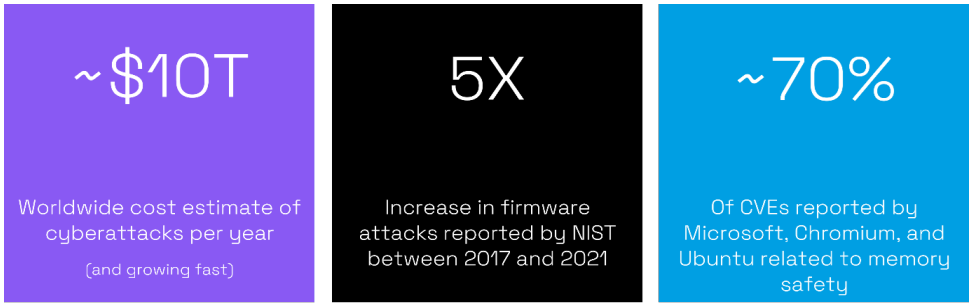
The L31AS is an ASIL-B certified, dual-core lockstep configuration (DCLS) of L31. As a DCLS configuration, the two L31 core instances run the same software but with one core delayed by a set number of cycles. The outputs of the two cores are compared for consistency – an inconsistency is a flag for a hazardous disturbance.

The L31 instances comprise 32-bit processor cores with a 3-stage pipeline, a branch predictor, single stage multiplier, divider and physical memory protection.

L31AS is delivered as IP with a User Manual, Safety Manual and Safety Use Case Report.

# Codasip CHERI Technology

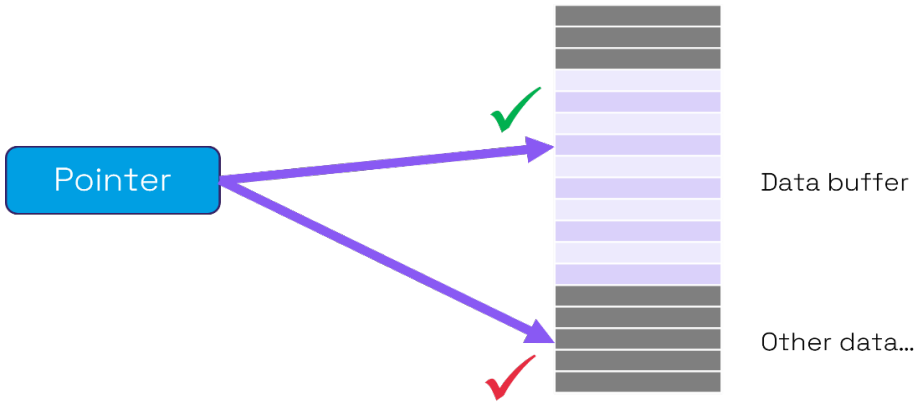
Memory safety is one of the most common and severe challenges in security with vulnerabilities such as buffer overflow and buffer underflow causing major economic damage. For example, a conservative estimate is that costs of $500M were incurred as a result of the well-known Heartbleed single vulnerability in OpenSSL. Looking at the overall situation the costs of cyberattacks worldwide are estimated to be ~$10T. Microsoft, Chromium and Ubuntu report that consistently ~70% of CVE (Common Vulnerabilities and Exposures) reported relate to memory safety. Worse still, memory overflow CVEs are becoming more common year-on-year.




Memory safety issues are consistently among the most severe software weaknesses identified in Mitre Corporation's Common Weakness Enumeration (CWE) database. Out-of-bounds Write has consistently ranked top in severity and Out-of-bounds Read has similarly ranked seventh.

Pointers in C/C++ use generic integers, and there are no automatic bounds checks with these languages, so it is easy to inadvertently create a buffer overflow or over-read.

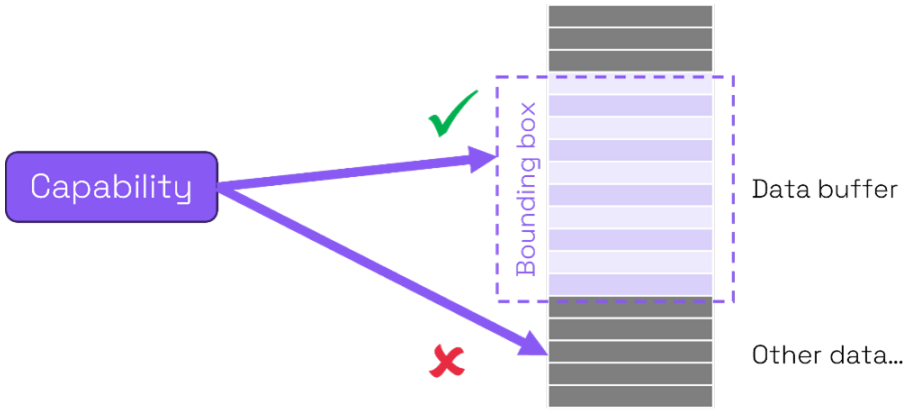
A further complication is that most embedded software developers do not make exclusive use of their own code. They typically assemble code from a variety of internal and external sources making auditing code for memory safety very challenging.




Pointers have no bounds checking leading to buffer overflow being allowed

Most approaches to mitigating memory safety risks have limitations. Memory safe languages, such as Rust, are good but in most cases, it is not practical to eliminate all legacy C/C++ code. Statistical countermeasures such as stack canaries and memory tagging have limitations and can be readily hacked.

The University of Cambridge has tackled the root cause of memory unsafety – pointers – in their Capability Hardware Enhanced RISC Instructions (CHERI) project. Instead of pointers, CHERI uses architectural capabilities to provide both fine-grained memory protection and scalable compartmentalization. Capabilities consist of an address, bounds, permissions and a tag.

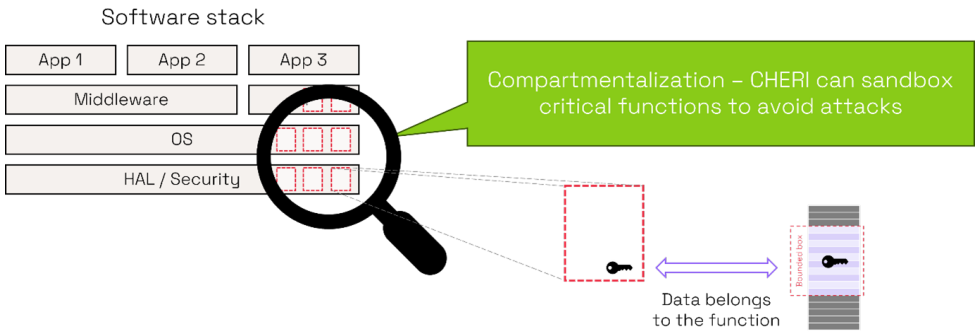



Architectural capabilities prevent out-of-bounds accesses by enforcing buffer bounds

Architectural capabilities define boundaries for memory accesses thus making any out of bounds access impossible. Implementing CHERI in hardware and software ensures that about 70% of CVEs are prevented. Using CHERI allows legacy C/C++ code to be used safely.

## Scalable compartmentalization

Another security challenge is the provenance of code. Decades ago, applications were developed by a single organization, but today's applications involve integration of code from internal, 3rd party commercial and open-source organizations. With CHERI, critical functions can be protected against cyberattacks.

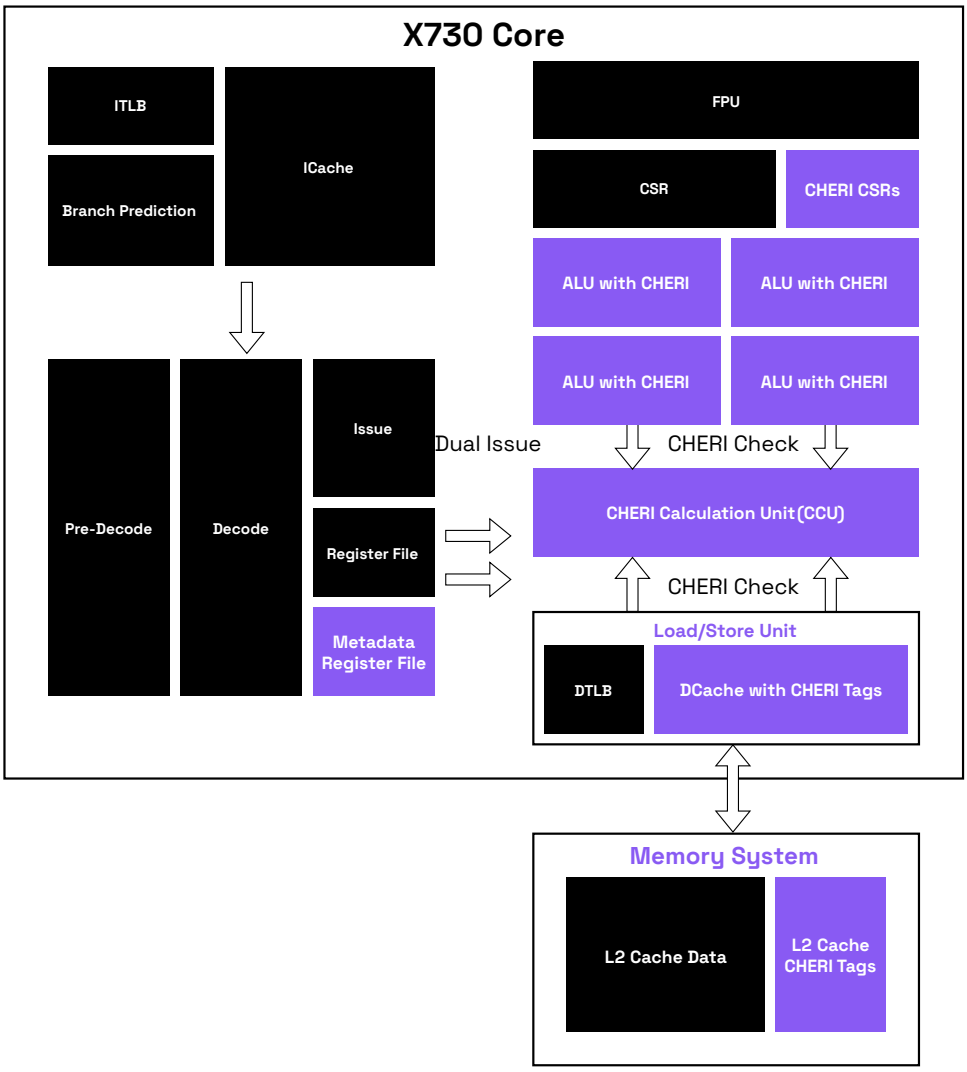



Scalable compartmentalization

Limited protection is possible with coarse-grained compartmentalization approaches such as virtual machines or by separating secure and open execution environments. CHERI offers a scalable approach making it possible to define software compartments at any scale from a complete SW system down to a single function. Data can even be defined as belonging exclusively to a function providing fine-grained defence against attack.

## X730: 64-bit secure, application processor with CHERI

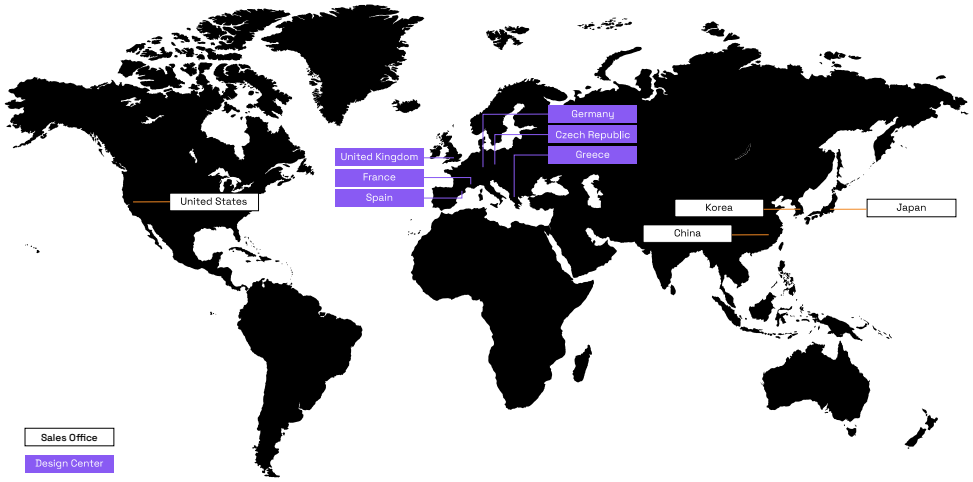
The X730 is an in-order, dual-issue 64-bit application processor core with a 9-stage pipeline. It is the first commercially licensed secure processor with CHERI functionality. The 730 family of cores was designed to accommodate CHERI from the start so X730 shares a single code base with A730. X730 is delivered with full CHERI-compliant SDKs for bare metal and Linux. There is also the option of the configuring it with the Scalar Crypto Extension (SCE) or Zk extension.




Overview of X730 secure application processor showing microarchitecture impact of CHERI

CHERI requires custom instructions as well as major changes to the CPU microarchitecture meaning that the X730 implementation is a good fit with our Custom Compute approach. As can be seen in the above figure, CHERI impacts the register file, CSRs, ALU, load/store unit and memory system. Despite this the area overhead for the CHERI microarchitecture is less than 5% over the baseline 730 design and there is no speed penalty.

# About us




We help our customers secure competitor advantage by leveraging our unique custom compute offering. We are here to help you architect your ambition, simply get in touch with us.

We have offices worldwide so our team is accessible wherever you are. Our design centers are based in Europe.

codasip.com/contact

We are a founding member of the CHERI Alliance, an organization committed to uniting leaders, system developers, users and security experts to drive and promote CHERI as an efficient security standard.

CHERI Alliance Founder


Codasip