Codasip

→ Product brochure

↘

Architect your ambition

Stand out
Set your ideas free
Embrace custom compute

In today's technology market, standing out is everything. The difference between your success and failure. And, in chip design, this difference is quite literally wafer thin. With increasing transistor costs, your developers can no longer rely on semiconductor scaling and legacy processors to achieve your goals. The only way forward is to implement custom compute with designs tailored to your applications.

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.

We deliver custom compute through the combination of the open Codasip Studio™ processor design automation 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.

Unlike traditional design approaches, our custom compute enables you to take control of your destiny. We allow you to set free your creativity and to use your ingenuity. We're at the leading edge of a transformation in processor design, providing our partners, the most innovative companies on the planet, with a proven alternative to the norm.

www.codasip.com . contact@codasip.com

→ We are at the end of the line

For decades our semiconductor industry has benefitted from semiconductor scaling most famously through Moore's Law. Moving to successively smaller geometries has delivered greater transistor density and higher clock rates, while keeping power consumption acceptable. As a result, companies tended to keep similar architectures from one technology generation to the other but to pack more complexity into processor cores in order to gain more performance.

Another trend has been the use of off-the-shelf processor IP. In the 1990s many semiconductor companies had their own processor architectures. During the 2000s there was a tendency for our industry to adopt off-the-shelf processor IP such as Arm for MCUs and application processors, CEVA for DSP and Imagination for GPUs.

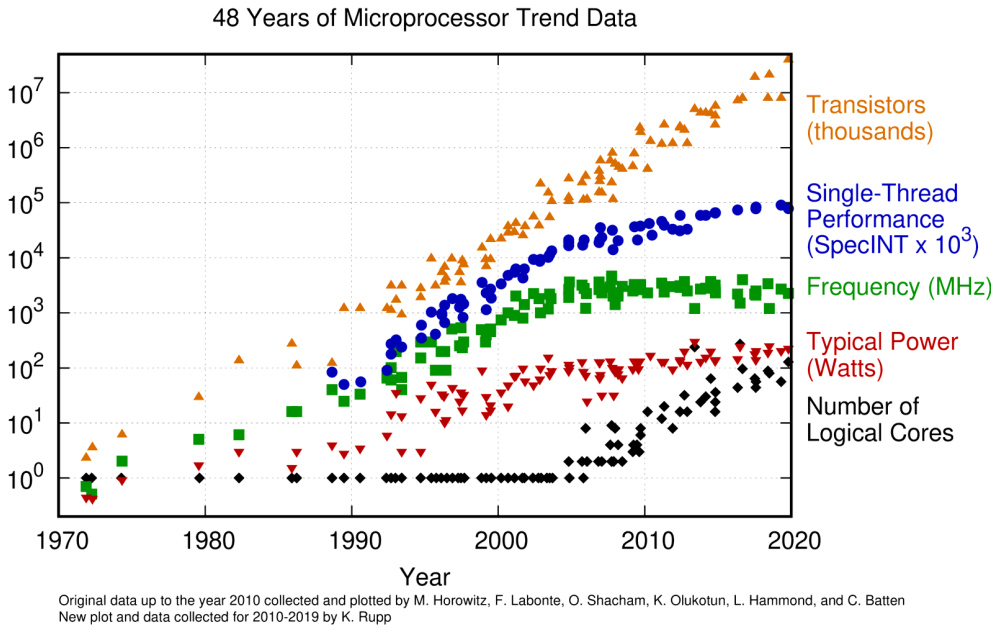


Figure 1. 48 Years of Microprocessor Trend Data. Source: K. Rupp.

It is well known that around 2005, with CMOS 90 nm technologies and below, leakage current became significant and without limiting clock frequencies there was the risk of thermal runaway. This meant the end of Dennard Scaling which is the design approach that enabled Moore's Law. As can be seen from the trend data chart, this means that you face a practical upper limit in clock frequency.

The semiconductor industry responded by moving away from single core X86 or Arm processors and moving to multi-core solutions. Also, some specialization took place with GPUs, DSPs and MCUs being given more specialized roles. This provided more performance although each processor category was still relatively general purpose.

With most SoC designers licensing off-the-shelf application processors, GPUs, DSPs and MCUs from a small number of vendors, a consequence was that many products ended up very similar to those of their competitors. Differentiation has been limited.

New emerging applications such as machine learning, augmented reality/virtual reality, and industrial IoT require high performance of very specific computational workloads. The new approach that we have taken in recent decades namely relying on semiconductor scaling and off-the-shelf processing has reached the end of the road.

→ Custom compute is the way forward

The only way forward in the foreseeable future is to match your processor designs to your computational workload. While in the past, software was adapted to the available hardware, efficient computation will require hardware to be tailored to your software. In other words, to increase your performance you need custom compute solutions.

Specialized processing solutions have been developed for specific computationally intensive tasks such as image processing, deep learning, specific graphics functions and even bioinformatics. Such processors will have instruction sets and microarchitectures aimed at a specific set of computations. This specialization will result in a computational block that is efficient in both silicon area and power consumption.

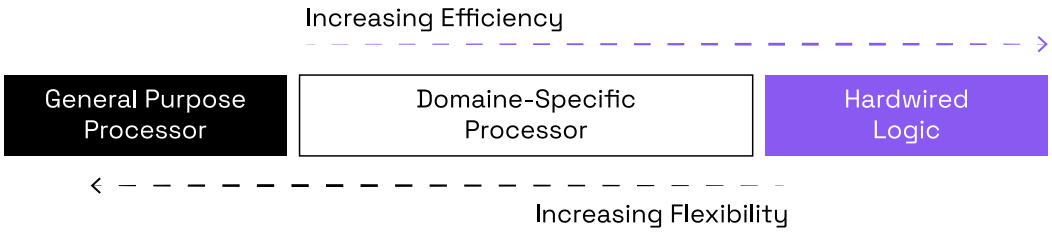


Figure 2. Custom compute Continuum.

However, there is not a one-size-fits-all for custom compute. In some cases, the degree of specialization will be fairly extreme, approaching the extreme of dedicated non-programmable hardware. In other cases, there may be a broader workload meaning that the custom processor could be based on modifying an existing core. Custom compute is a continuum by definition.

The future of on-chip computation lies in breaking free from the constraints of traditional off-the-shelf IP cores and creating a variety of special ones. This means involving more engineers including programmers in core creation.

Designing special cores, sometimes called application-specific instruction processors (ASIPs), from scratch has been a domain of specialized processor groups in the past however the increasing adoption of RISC-V makes it much easier. RISC-V is modular, providing a base integer set, standard extensions and the possibility of custom extensions. One of the key challenges of an ASIP – creating the instruction set – is considerably simplified as there is a standard starting point. Furthermore RISC-V has a rapidly growing ecosystem.

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

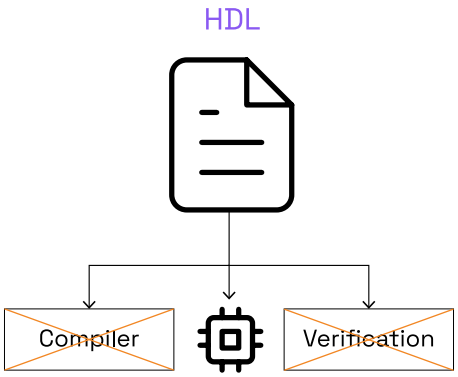


Figure 3. Hardware description languages only cover hardware.

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. However, 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.

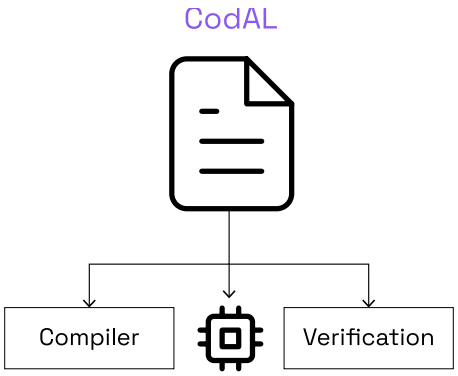


Figure 4. An architectural language covers software and verification as well as hardware.

If you use an architectural language such as Codasip's CodAL™, 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.

Summarizing, if you want to create a custom compute processor core some key enablers are:

Having processor design automation to generate both the SDK and HDK.
Using an architectural language (AL) to describe your core at a high level.
Having proven RISC-V cores described in an AL to enable customization from a verified starting point.

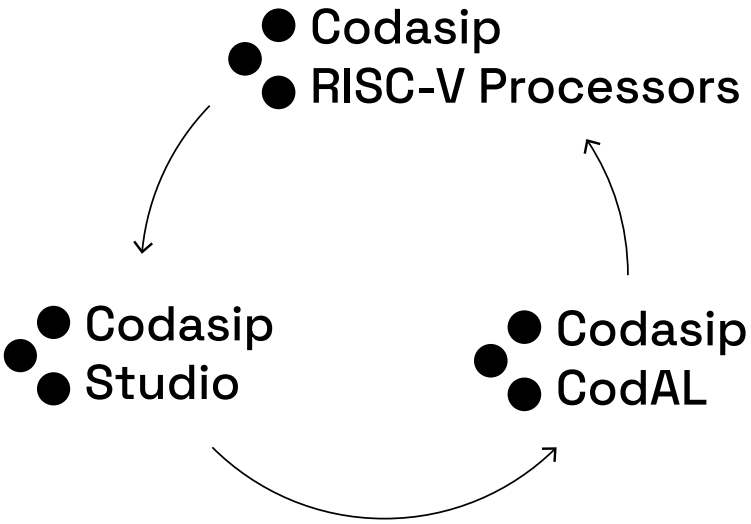
Codasip is uniquely able to offer all three enablers through:

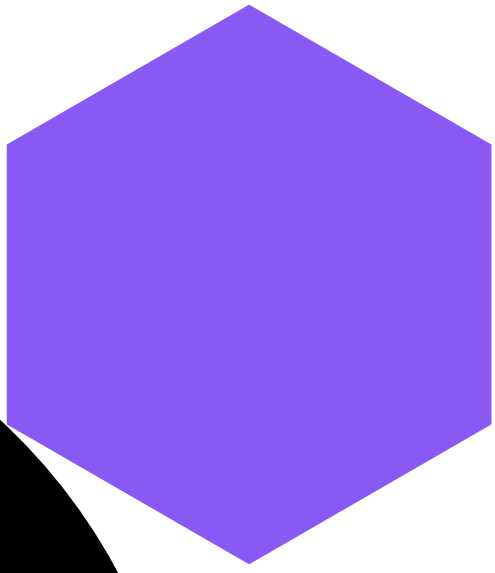
The Codasip Studio tool suite which can support development of CodAL code plus SDK and HDK generation.
The Codasip CodAL architectural language.
Having a range of RISC-V cores described in CodAL which can be customized to meet the needs of a specialized processor core.

We now describe these enablers in more detail.

→ Our proven solutions

Codasip's proven solutions are powering SoCs from around the world. Find out how you too can benefit from our unique technology and methodology.





↘ 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 optimize 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 the 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 Eclipse, LLVM, Verilog, SystemVerilog, and UVM.

→ Straightforward development process

A unique aspect of our approach at Codasip is to automate the development of processor cores by using a high-level description language called CodAL. 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. This eliminates the need for you to express the same functionality in multiple task-dependent formats and reduces traditional time-consuming manual tasks.

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.

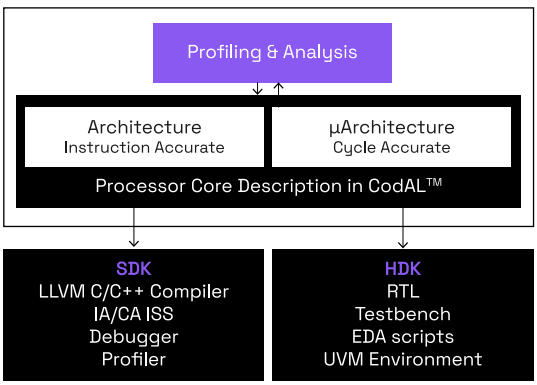


Figure 5. Codasip Studio design flow.

→ Match hardware to your software

For decades, software engineers have had to match their software to the available hardware. With Studio you have the possibility to adapt your hardware to your software workload, making it an ideal tool for creating your custom compute solution.

Once your instruction-accurate (IA) model of the processor is available, which typically takes only days, Codasip Studio can generate a complete SDK customized for your processor. This kit can be freely used by your embedded software development teams, allowing development, debug, and execution on the target platform well ahead of silicon availability. Generated tools are latency-aware for both instruction and data access. This is critical for maximum performance on VLIW architectures where multiple instructions are executed in parallel. Alternatively, if your starting point is an existing Codasip RISC-V core described in CodAL, then you have immediate access to an SDK.

Either way, you can run your software compiled to the target instruction set. The Codasip Studio profiler will enable you to identify “hotspots” where certain parts of the code are very intensely used.

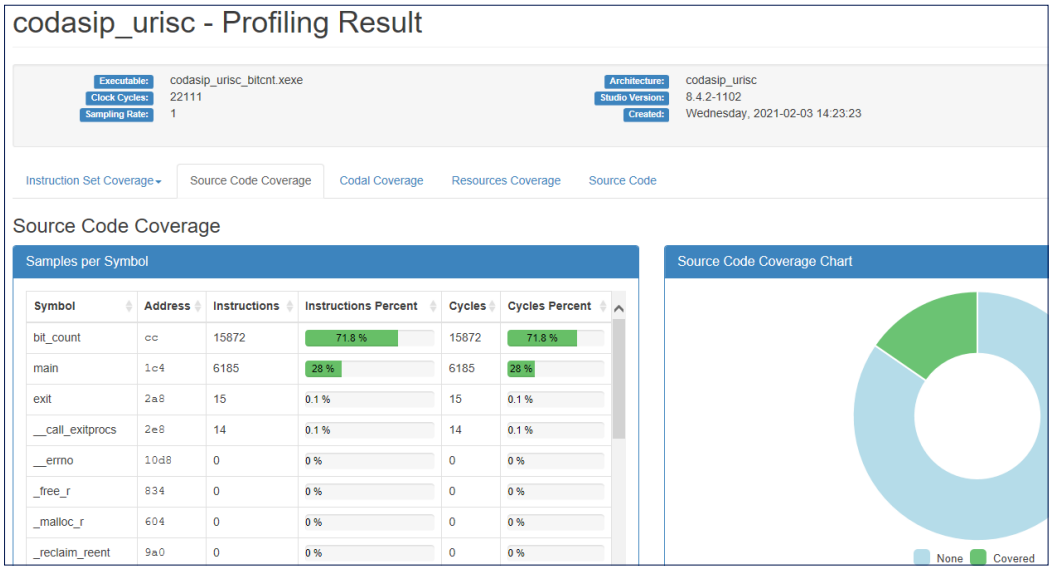


Figure 6. Screen shot of profiling.

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. While competing solutions only allow custom instructions to be implemented as a coprocessor, Codasip Studio enables custom instructions to be inserted into the processor pipeline.

From the CA description, Studio enables you to generate a cycle-accurate simulator, RTL, a testbench and a UVM verification environment. 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. Also, when doing co-simulation, you can trace the RTL back to the corresponding CodAL source code.

→ Automatically generated HDK & SDK

The generated outputs from Studio are summarised below:

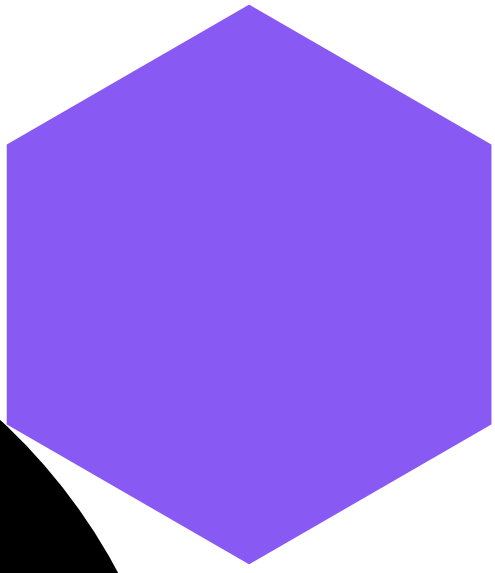
Hardware Development Kit (HDK)	**Software Development Kit (SDK)**
RTL (Verilog/VHDL/SystemVerilog)	C/C++ LLVM compiler (improved by Codasip)
SystemVerilog UVM test environment	C/C++ Libraries (newlib)
Integration test bench	Assembler, disassembler, linker
Sample EDA scripts	High-performance instruction set and cycle accurate simulators
SystemC co-simulation model	Debugger and profiler
	Documentation and ISA visualization
	Random programs used during verification

When designing or customizing a processor (or processors) for your IC design, 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 consistency checker, random assembler program generation and an automatically generated UVM environment. UVM allows the generated RTL for your processor to be checked against your instruction-accurate reference model.

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.

Codasip
CodAL



↘ CodAL

Codasip Architectural Language

→ Codasip Architectural Language

CodAL, standing for Codasip Architectural Language, is central to developing a processor core using Codasip Studio. The language has a C-like syntax and it merges good practices and code constructs from conventional programming languages and hardware description languages. It has been developed to describe all aspects of a processor, including the ISA and microarchitecture.

Each processor description includes four elements:

1. **Architectural resources**. For example, registers and a program counter.
2. **Instruction set.** Names of instructions, their operands and their binary form (opcodes).
3. **Semantics**, or a description of the behavior of each instruction and exception.
4. **Implementation**, which includes those resources and behavior (especially timing) that are not visible in the architectural model, but which define a particular microarchitectural implementation. More than one microarchitectural implementation can be in a single CodAL description.

The architectural or instruction-accurate (IA) model contains the instruction set, architectural resources, and the semantics. The microarchitectural or cycle-accurate (CA) model contains the instruction set, architectural resources and the microarchitectural implementation.

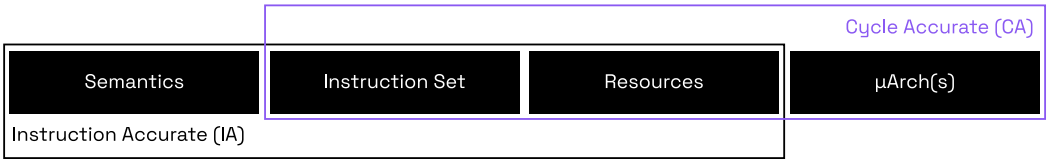


Figure 7. CodAL description.

The CodAL description is object-oriented, meaning that an object can be instantiated into a more complex object, the complex object into an even more complex one, etc. CodAL allows information to be passed through the object hierarchy without having to use complex function calls.

The CodAL element is a common example of an object, and the following example shows an element describing a multiply-accumulate instruction.

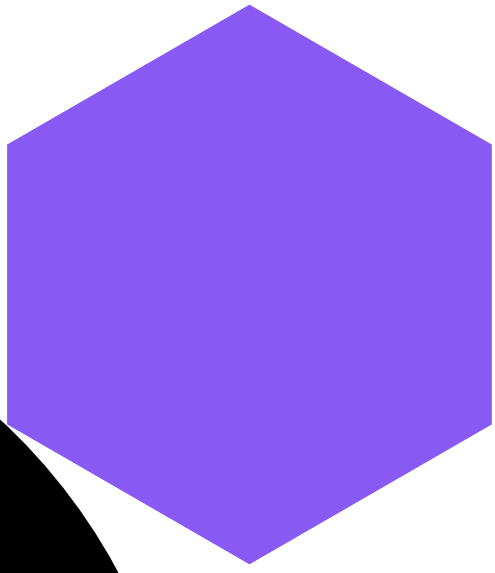
```
/*   Multiply and accumulate: semantics
                        dst += src1 * src2                  */

element i_mac {
        use reg as dst, src1, src2;
        assembly { "mac" dst "," src1 "," src2 };
        binary { OP_MAC dst src1 src2 0:bit[9] };
        semantics {
                        rf[dst] += rf[src1] * rf[src2];
        };
};
```

The use statement describes the resources that are used by the instruction – a destination register (dst) and two source registers (src1, src2).
Next, the assembly statement describes the assembler mnemonic (mac) and its arguments.
The binary statement describes the binary representation of the instruction and arguments.
Finally, the semantics statement describes the multiply-accumulate operation.

The Codasip Studio toolset uses the CodAL description to generate an SDK and an HDK.

The CA description would take advantage of the instruction set and resources descriptions used for the IA models. In addition, the CA description would specify microarchitectural features such as the pipeline length.

The cycle-accurate parts of the CodAL description can be used for generating the cycle-accurate simulator, RTL, testbench, and UVM environment. In this way CodAL is the single source for all aspects of the processor hardware and software. In contrast, some other processor development tools require two languages to describe the processor core. The methodology also enables powerful verification of generated RTL against a golden reference model in generated UVM environment.



↘
Codasip
RISC-V
processors

→ Match a Codasip core to your project needs

RISC-V is the cornerstone of our processor families. Take advantage of this open, royalty-free instruction set architecture (ISA):

→ Vendor independence and longevity
→ Modularity and extensibility
→ Custom microarchitecture
→ Endless possibilities

From low-power embedded to application cores, our RISC-V IP is either 32-bit or 64-bit and varies in pipeline lengths, wordlengths, and arithmetic units.

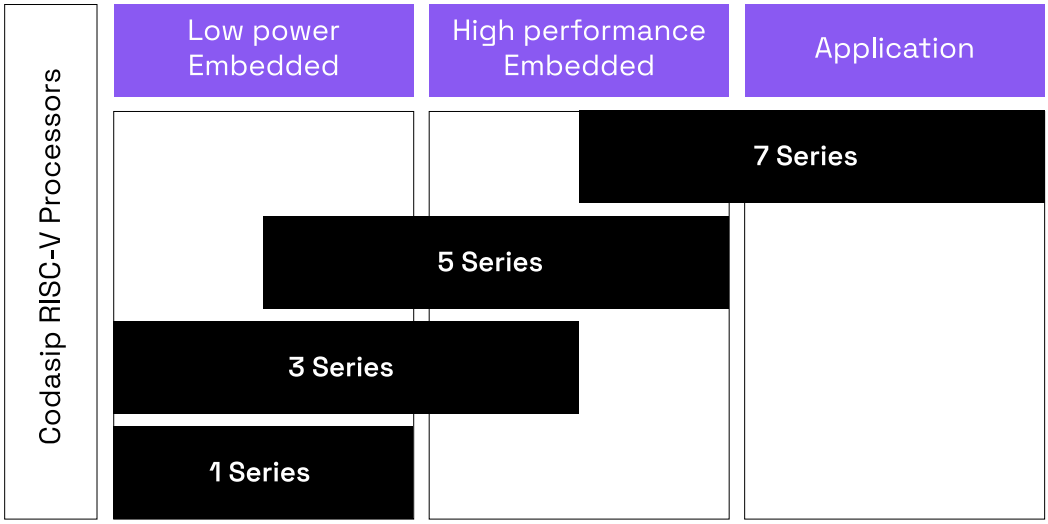


Figure 8. Codasip RISC-V Processors families.

→ Codasip processor portfolio

All cores are fully customizable and adaptable to the unique needs of your project.

All Cores Standard RISC-V debug JTAG (4pin/2pin) Compressed instructions AMBA buses	**Low power Embedded** 32-bit Up to 128 interrupts	**High performance Embedded** 64-bit Up to 256 interrupts	**Application** 64-bit FPU Linux support
1 series 3-stage pipeline EMC instruction set 16 registers Sequential multiplier	→ Codasip L11 → Codasip L10*		
3 series 3-stage pipeline IMC instruction set 32 registers Parallel multiplier	→ Codasip L31 → Codasip L31F → Codasip L30* → Codasip L30F*		
5 series 5-stage pipeline IMC instruction set 32 registers Branch predictor Parallel multiplier	→ Codasip L50* → Codasip L50F*	→ Codasip H50* → Codasip H50F*	
7 series 7-stage pipeline IMC instruction set 32 registers Branch predictor Parallel multiplier			→ Codasip A70 → Codasip A70-MP

* Not recommended for new designs
F = Floating Point Unit, MP = Multiprocessing

→ Embedded RISC-V processors

Looking for an embedded RISC-V core? We have you covered. Our embedded family has been designed into SoCs and application examples include:

→ Mobile phone autofocus & image stabilization
→ Surveillance cameras
→ Display panel control
→ Wireless
→ Controller for AI processing elements
→ Motion control
→ Video processing

These cores support both the embedded [E] base instruction set (L11) and standard integer instruction set [I] (L31 and above). The product line also supports commonly-used RISC-V standard extensions (M, C, B & F) making the products an attractive alternative to legacy proprietary embedded cores.

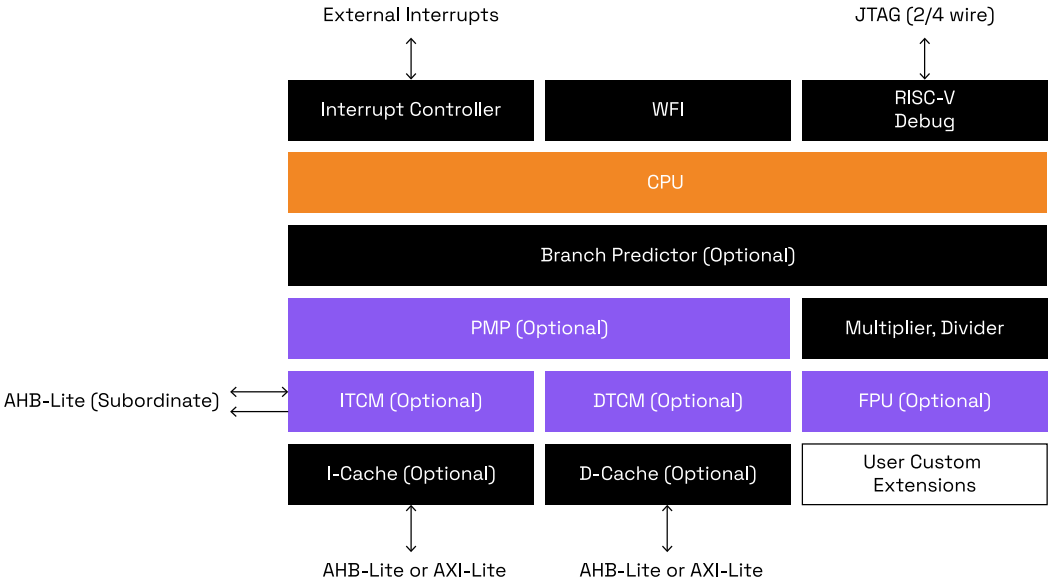


Figure 9. Codasip embedded processor block diagram.

→ Choose your own path

Off-the-shelf RISC-V CPU	Customizable RISC-V CPU
Delivered as RTL	Delivered as CodAL source code
Production ready	Modifiable with Codasip Studio

Figure 10. Alternative ways to licensing Codasip RISC-V IP

We offer you two ways in which to license our RISC-V processor cores. First, the well-established licensing of off-the-shelf cores. These cores are delivered as RTL with an SDK and testbenches. The cores are rigorously verified and production ready.

For processor IP, we use a combination of in-house and third-party tools to verify the cores. Processors have a complex state space so are more complex to verify than most other types of RTL design. Therefore, Codasip has adopted the Swiss cheese approach to verification – adopted from the world of aviation. The idea is that no one verification method is capable of catching all bugs and that multiple verification layers are needed. Examples of the verification layers are:

→ Code review
→ Directed tests
→ Block-level testbenches
→ Assertions
→ FPGA OS boot
→ Top level random tests
→ Formal verification
→ Coverage analysis

Second, the opportunity to license the CodAL source to our processor cores. This enables you to develop your own custom core tuned to your application workload using Codasip Studio.

These approaches are not mutually exclusive. You can start with an off-the-shelf core and then in a later product generation develop a custom compute solution.

→ Codasip application processors

The application processors are more powerful cores with a RV64GC or RV64IMAC ISA and a complex pipeline (7 stages and above).

These cores were designed to support all the privilege modes and a memory management unit (MMU) needed to run Linux and other rich operating systems. Other features include integrated L1 Data and Instruction caches of configurable size, an internal interrupt controller, dynamic branch prediction, FPU, and on-chip debugger based on JTAG and RISC-V debug standards.

These cores are suitable for applications that require an advanced user interface, sophisticated I/O and networking, such as in set-top boxes or entertainment systems; they can also support complex computations, full-process isolation and multitasking, or a full separation of application code from hardware via device drivers.

The first of these cores, the A70, is a very area- and power-efficient single-issue core. It is therefore well-suited to applications requiring low silicon costs. Further higher-performance cores will be released in the future – check with Codasip sales for more details.

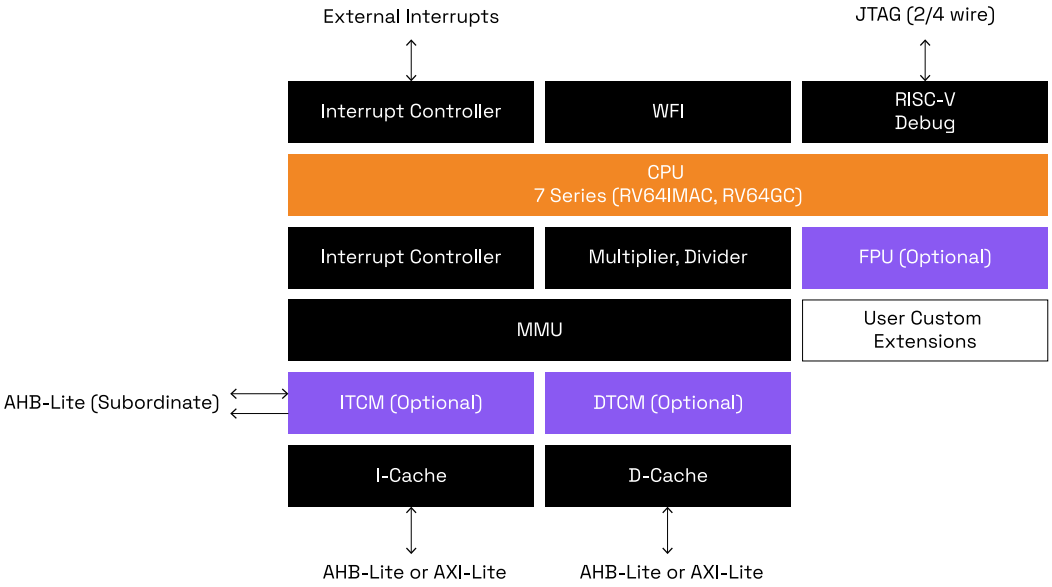


Figure 11. Codasip Application processor block diagram.

→ Processor features

→ Standard buses

We provide native AMBA interface support for each processor. You can configure and customize the AHB-Lite or AXI-Lite interfaces, thus preserving your investments made in industry-standard peripheral IP blocks without taking the latency penalty that comes with using bus bridges.

→ Outstanding software infrastructure

Any Codasip core is supplied with an industry leading LLVM-based development environment that outperforms community alternatives across a wide range of benchmarks. This is because we have added proprietary optimizations to the LLVM back-end. For example, we introduced our own implementation of jump threading, a compiler pass that aims to speed up code execution by replacing conditional jumps with unconditional ones. With our implementation, we achieved up to 16.5% faster code compared to out-of-the-box LLVM jump threading.

Our standard development environment includes:

Hardware Development Kit (HDK)	Software Development Kit (SDK)
RTL (Verilog/VHDL/SystemVerilog)	C/C++ LLVM compiler (improved by Codasip)
Verification report	C/C++ Libraries (newlib)
Integration test bench	Assembler, disassembler, linker
Sample EDA scripts	High-performance instruction set and cycle accurate simulators
Codasip CodeSpace (Eclipse-based IDE)	Debugger and profiler

→ Custom features with Codasip Studio

When using a Codasip core with Codasip Studio, you can choose among the standard extensions defined by the RISC-V specification, but you are not restricted to only those. You can innovate by creating your own custom instructions and changing microarchitectural features.

→ Automated workflow

When adding value to a Codasip core to develop an innovative IC, Codasip Studio enables you to verify your modified core and then to automatically generate your HDK and SDK.

→ Debugging

Our embedded debug implementation combined with the Eclipse-based Codasip Studio (for processor customization) or Codasip CodeSpace (for software development) provides a unified experience from early architectural exploration to in-circuit FPGA and ASIC debug.

→ Developing your software

Software is an essential part of custom compute and we let you choose your own approach. Your options are:

→ Generate a software toolchain using Studio and develop software with CodeSpace

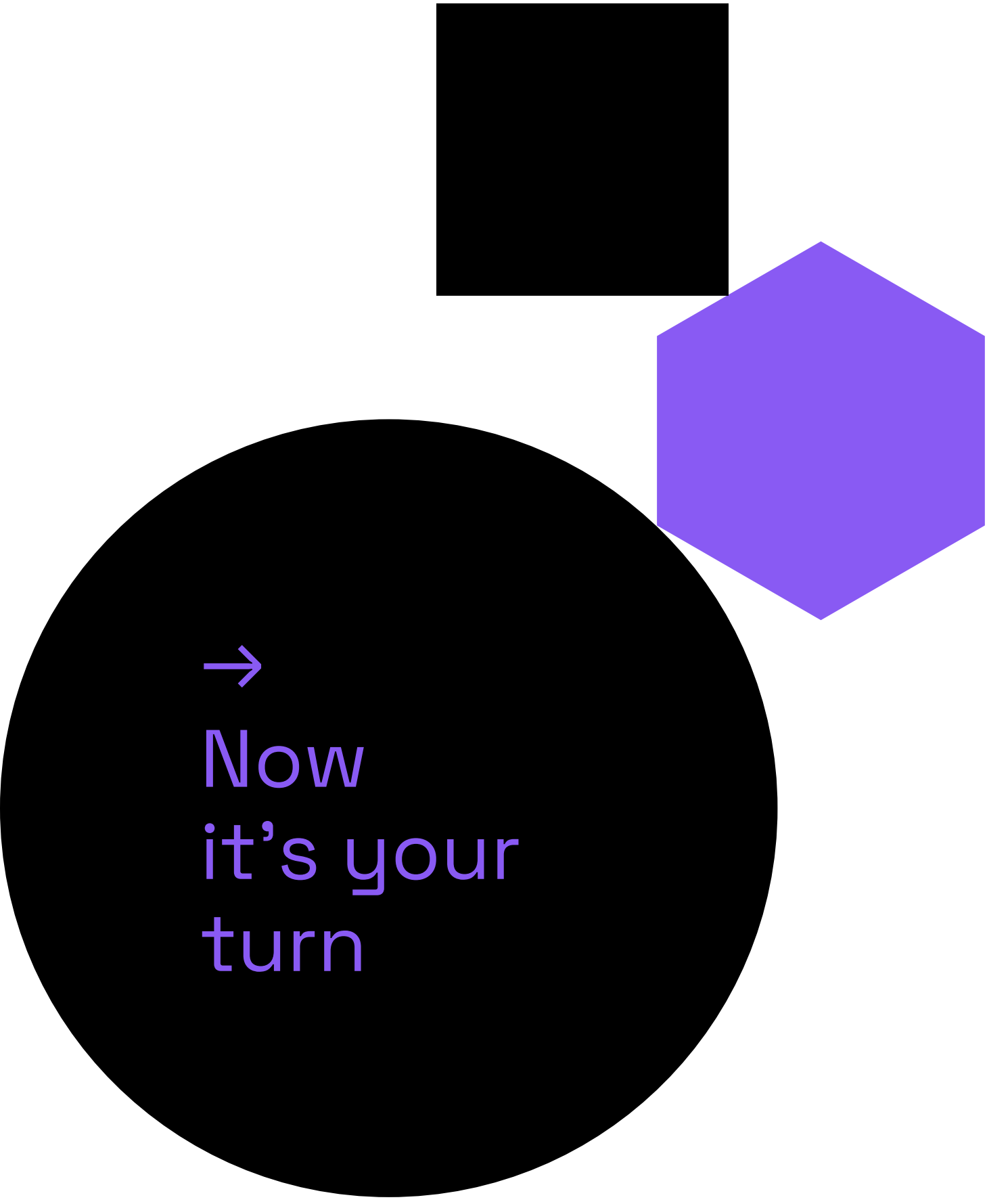
If you are using Studio to profile your software and to create custom instructions, you can automatically generate an SDK based on LLVM. In many cases the compiler will use your custom instructions directly and in other cases intrinsics will be generated automatically. Codasip CodeSpace is an advanced firmware development environment that enables you to write code for your new core. It includes a code editor, compiler, profiler, and debugger. It works seamlessly with the SDKs that you generate in Studio.

→ Use commercially available software development tools

Alternatively, there is a good choice of commercially available software development tools supporting Codasip RISC-V cores for example IAR, Lauterbach or Segger. You can use your familiar IDE and toolchain with your chosen Codasip cores with standard extensions.

→ Use open source tools

If you are using a Codasip core with standard extensions (no custom instructions) then a third choice is to use open-source tools. You can use either GNU or LLVM tools for RISC-V and these support standard extensions. This approach becomes more labor intensive when you add custom instructions as you either need to manually update the toolchain or use intrinsics. With custom instructions the Studio-based approach is the easiest and lowest risk.



→ Proven technology and IP

If you are looking for a processor solution for your SoC project, you want to be confident that you are building on a working and well-verified solution. Codasip's solutions are silicon proven.

Mythic (Redwood City, USA) is a leader in artificial intelligence (AI) computing technology based on a unique approach to neural network processing.

"Codasip gave us the flexibility to create a truly unique RISC-V processor, specific to our needs. This saved us the effort to build our own processor from scratch and allowed us to focus on other critical areas of the product development."

Ty Garibay, VP of Hardware Engineering at Mythic

Dongwoon Anatech (Seoul, Korea) is a leader in analog and power ICs (integrated circuits) for mobile phones.

"The RISC-V instruction set with custom DSP extensions delivers the performance we require while keeping silicon area to a minimum. The best-in-class Codasip Studio development tools enable us to profile our software and find an optimal set of instructions for our application."

Jin Park, CTO of Dongwoon Anatech

Rambus (Sunnyvale, California) is a leading provider of IP cores, software and services, specializing in performance and protection of data.

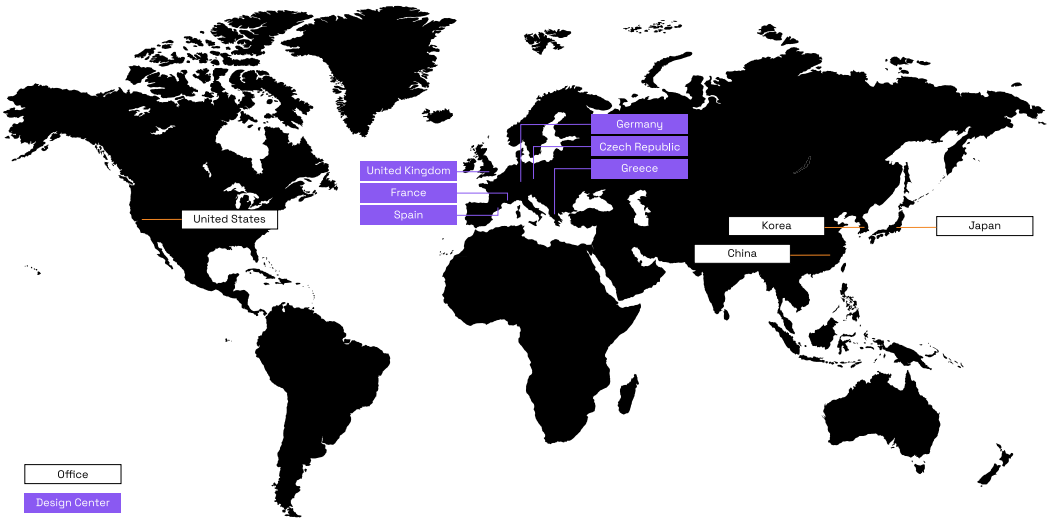
"Security is the leading issue for IoT, automotive and other fast-growing markets, and it is critical for Rambus to deliver superior products to market in a timely fashion. We selected Codasip Studio because it allows for fast design space exploration, and because of the high quality of results we are getting in the automatically generated compiler toolchain."

Bret Sewell, SVP and general manager of the Rambus Security Division

Vidtoo (Hangzhou, China), a leader in semiconductor products for HPC (high-performance computing), artificial intelligence, and machine learning platforms.

"After careful consideration, we determined that Codasip offered the best combination of performance, value and design expansion ability. Those traits, plus best-in-class support and the broad ecosystem that the open RISC-V ISA brings, gave us confidence that Codasip was the right choice."

Thomas Hu, CEO of Vidtoo Technologies



→ About us

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 today's technology market, differentiation is everything. The difference between success and failure. And, in chip design, this difference is quite literally wafer thin. With increasing transistor costs, your developers can no longer rely on semiconductor scaling and legacy processors to achieve your goals. The only way forward is to implement custom compute with designs tailored to your applications.

We deliver custom compute through the combination of the open RISC-V ISA, Codasip Studio processor design automation and high-quality 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.

Unlike traditional design approaches, our custom compute enables you to take control of your destiny. We allow you to set free your creativity and to use your ingenuity. We're at the leading edge of a transformation in processor design, providing our partners, the most innovative companies on the planet, with a proven alternative to the norm.

At Codasip, we enable you to design different.

It's time to take the leap.

Architect your ambition.