The Cadence® Tensilica® Xtensa® 11 dataplane processing unit (DPU) is an exceptional controller built on the unique Xtensa technology. It can easily be extended to out-perform any other embedded processor for a specific application using a combination of configurable options and custom instructions. The Xtensa 11 DPU even includes options for highly efficient digital signal processing (DSP). This makes it ideal for applications that require a combination of DSP and control.

**Features**
- Highly efficient, small, ultra-low-power core with a 32-bit modern architecture and 5-stage pipeline
- Wide range of configurable options
- Local memories configurable up to 8MB with option for parity or ECC
- Hardware prefetch unit
- Extend with designer-defined application-specific instructions, execution units, register files, and I/Os
- 32-bit wire input and 32-bit wire output GPIO port option for peripheral control and monitoring
- 2x32-bit queue FIFO interface option for data streaming, bypassing the system bus
- Power domains for power shut off
- Semantic and memory data gating
- Compatible interfaces for ARM® CoreSight™ debug and trace technology
- Optional IEEE 754-compliant single-/double-precision scalar floating-point unit, plus double-precision scalar floating-point acceleration
- Complete matching development tool chain automatically generated for each core
- Wide range of operating system support, including Linux

**Benefits**
- Extremely efficient base architecture that is smaller, lower power than most other 32-bit embedded controllers
- Application-specific instruction extensions provide orders of magnitude performance improvements over traditional CPUs, in many cases eliminating the need to develop RTL blocks
- Lower verification effort with pre-verified, correct-by-construction RTL generation
- Post-silicon programmability to extend the life of any design
- Highly accurate, high-speed system simulation models automatically created for software development
- Develop, simulate, debug, and profile in one integrated design environment (IDE)
- Low-leakage power design
- Dynamic power savings
- Easy integration into an ARM CoreSight interface-based debug and trace infrastructure
- Double-precision operations have greater precision and wider range than single precision
- A complete matching software tool chain is automatically generated for each core
- Mature, highly optimizing C/C++ compiler means you can work at the ‘C’ level for most applications
Ideal for SoC Dataplane Processing

Designers can use the Xtensa 11 DPU to not only perform control functions, but also some of the finite state machine tasks that manage RTL blocks and some of the RTL logic, as well. This makes for a smaller, much more efficient chip design, and it significantly reduces the verification challenges associated with new RTL designs.

Create an Optimized DPU in Minutes

The Xtensa 11 DPU was designed from the start to be a basic building block in SoC designs. The Xtensa 11 processor is unlike other conventional embedded processors cores—the system designer can mold the processor to fit the target application. By selecting and configuring predefined elements of the architecture and by inventing completely new instructions and hardware execution units, the Xtensa 11 processor can deliver performance levels that are orders of magnitude faster than other standard 32-bit cores. And you can do this in a fraction of the time it takes to develop and verify an RTL-based solution.

Designers can define new instructions utilizing the Tensilica Instruction Extension (TIE) methodology, adding Verilog-like descriptions of datapaths, execution units, and register files that can deliver power, performance, and area characteristics approaching those of custom logic design.

Feature Overview

Backwards-compatible ISA since 1999

- Xtensa instruction set architecture (ISA) fundamentally architected for extensibility
- Base instruction set of 80 instructions for compatibility across every Xtensa core
- Dozens of available optional blocks
- Any differentiating designer-defined instructions written since 1998 can still be re-used today

Optional pre-defined execution units

- 32-bit multiplier and/or 16-bit multiplier and MAC
- Integer divide
- IEEE 754-compliant single- and double-precision scalar floating-point unit
- Double-precision scalar floating-point acceleration
- Pre-defined 32-bit GPIO and FIFO-like queue interfaces

Differentiate with designer-defined functions

- Make your specific algorithm run even more efficiently by adding the instructions it needs

Natural connectivity with RTL blocks

- Development tools automatically adapt for full support
- Multiple custom-width I/O ports for peripheral control and monitoring
- Multiple custom-width queue interfaces to FIFOs for data streaming into and out of the processor
- Co-simulation with RTL down to the pin level in SystemC

Highly configurable interfaces

- Optional processor interface (PIF) to system bus, choice of 32-, 64-, or 128-bit width
- Hardware prefetch unit
- Optional high-speed Xtensa Local Memory Interface (XLMI)
- Write buffer: selectable from 1-32 entries
- Optional ARM® AMBA® AXI and AHB-Lite interfaces with synchronous or asynchronous clocking
- Choice of 1-, 2- or 4-way cache and/or local memories
- Up to 32 interrupts

Multi-core design style support

- Multi-core system modeling and SystemC co-simulation out-of-the-box, fully supported within the Xtensa Xplorer™ IDE
- Homogenous and heterogeneous subsystems supported
- Inter-core on-chip debug with break-in/-out control and synchronized start/stop
- Optional 16-bit processor ID, supporting massively parallel array architectures
- Conditional store instruction option and synchronization library provide shared memory semaphore operations and the “release consistency model” of memory access ordering

Complete hardware implementation and verification flow support

- Automatic generation of RTL and tailored EDA scripts for leading-edge process technologies, including physical synthesis and 3D extraction tools
- Auto-insertion of fine-grained clock gating for low power
- Hardware emulation support including automated FPGA netlist generation for rapid SoC prototyping
- Comprehensive diagnostic testbench to verify connectivity
- Formal verification support for designer-defined instructions

High-speed, high-accuracy system simulation models automatically created

- High-speed instruction-accurate simulator for software development
- Pipeline-modeling, cycle-accurate Xtensa instruction set simulator (ISS)
- Xtensa SystemC (XTSC) transaction-level modeling (TLM) support, including out-of-the-box multi-core simulation
- Hardware co-simulation with RTL in SystemC with pin-level XTSC
Integrated design environment

• Create, simulate, debug, and profile whole designs in one tool: the high-productivity Xtensa Xplorer IDE
• Tenth-generation software development tools target each processor. The advanced Xtensa C/C++ compiler (XCC) includes optimizations for base, optional, and designer-defined instructions
• Vectorization Assistant directs the programmer to areas of the application that can benefit most from modifications to enable better vectorization
• Multi-core subsystem design and simulation support
• Custom data display formatting for easy debug of vector and fixed-point data types as well as bit-mapped status and control
• Automatic Xtensa Overlay Manager (AXOM) provides run-time management of large programs in small memories

Multi-core debug and ease of use

• Interfaces to support CoreSight infrastructure
• On-chip debug hardware widely supported by third-party JTAG debug probes
• DebugStall feature allows Xtensa processors to be stopped and started together using a hardware signal and to be debugged while in the stalled state
• Optional performance counters for real-time system analysis
• XMON software debug monitor for real-time applications
• Multi-core on-chip debug (OCD) support

Dynamic and leakage power improvements

• Power shut off (PSO) feature allows Xtensa DPUs to be completely powered off. To help achieve low leakage, designs can now be divided into multiple “power domains.” Each power domain operates at the same voltage and can be shut down and powered up individually
• Dynamic power saving features including semantic and data power gating
• Dynamic cache “way” control allows a programmer to turn off cache ways when not needed

Robust operating system support

• Use Mentor Graphics Nucleus+, Express Logic’s ThreadX, Micrium’s uC/OS-II, or the Linux operating systems

Efficient Base Architecture

The Xtensa 11 32-bit architecture features a compact instruction set optimized for embedded designs. The base architecture has a 32-bit ALU, up to 64 general-purpose physical registers, 6 special purpose registers, and 80 base instructions, including improved 16- and 24-bit (rather than 32-bit) RISC instruction encoding. Key features include:

• A wide range of configurable options to ensure you get just the logic you need to meet your functional and performance requirements

Figure 2. Xtensa 11 DPU showing standard, optional, and designer-defined blocks
- Modelessly intermix 16- or 24-bit instructions for lowest code and performance overhead
- Efficient 5-stage pipeline
- Local memories configurable up to 8MB with optional parity or ECC
- Optional hardware prefetch reduces memory latencies
- Caches up to 128KB and up to 4-way set associativity on both instruction and data sides
- Automated fine-grained clock gating throughout processor for ultra-low power solutions

**Base ISA compatibility**

Configurability of an Xtensa processor core never compromises the underlying base Xtensa ISA, thereby ensuring availability of a robust ecosystem of third-party application software and development tools. All configurable, extensible Xtensa processors are always compatible with major operating systems, debug probes, and ICE solutions. For each processor, the automatically generated complete software development tool chain includes an advanced IDE based on the ECLIPSE framework, a world-class C/C++ compiler, a cycle-accurate SystemC-compatible ISS, and the full industry-standard GNU Tool chain. The Xtensa processor uses an ISA that has been backwards compatible since its introduction in 1999. It uses a base instruction set of 80 instructions and was fundamentally architected for extensibility. Designers can run application code written in 1999 and it will run on the Xtensa 11 processor today. Any differentiating designer-defined instructions from earlier designs can be reused today.

**Smaller code size**

The Xtensa 11 DPU can modelessly issue 24-bit and 16-bit instructions, leading to 25-50% better code density and, therefore, smaller memories than mixed 32- and 16-bit architectures. Since memories typically dominate SoC area, this code-density advantage translates into significant SoC area savings.

**Powerful base ISA**

The Xtensa ISA includes powerful compare-and-branch instructions and zero-overhead loops, which allow the compiler to generate tight, optimized loops. It also provides bit manipulations including funnel shifts and field-extract operations that are critical for applications such as networking that processes the fields in packet headers and performs rule-based checks.

**Extensible ISA**

One of the fundamental technology innovations in the Xtensa processor is the ability to easily and seamlessly add new instructions into the processor’s datapath. The associated C data types, software tool chain support, and the EDA scripts required to synthesize the processor are all generated automatically, just as if they had been there from the start. The specification of this new datapath and associated instructions and C data types is written in the TIE language, which is explained in more detail in a later section.

**Highly configurable functionality**

Select from click-box options to add functionality to your processor and evaluate performance improvements in a matter of minutes. Basic interface options include:

- Processor interface (PIF)
  - Width: 32/64/128-bit
  - Optional “no PIF” configuration
- ARM AMBA AXI and AHB-Lite interfaces with synchronous or asynchronous clocking
- 16-bit processor ID
- Inbound DMA
- XLMI high-speed local interface
- Big-Endian/Little-Endian byte ordering
- On-chip debug port (IEEE 1149.1-compliant or CoreSight-compatible debug APB interface)

![Figure 3. Pre-verified major configuration options for the Xtensa 11 DPU](image-url)
• Trace port signals
• Up to 32 interrupts with up to 7 levels of priority plus a separate non-maskable interrupt level
• Write buffer: selectable from 1 to 32 entries
• 2x32-wire GPIO ports for direct control and monitoring of peripherals
• 2x32-bit queue interfaces for streaming data into and out of the processor via FIFOs
• Single 16-bit multiply accumulator (MAC)
• 16- or 32-bit multipliers
• Low-area integer divider
• IEEE 754-compliant single-/double-precision scalar floating-point unit
• Double-precision scalar floating-point accelerator

Memory subsystem options include:
• Multi-bank RAM support
• Up to 6 local memory banks can be connected for instruction and data accesses (up to 12 in total). Memory banks may be local ROM, RAM, or cache ways. Up to 2 local instruction and data RAMs and ROMs up to 8 Mbytes each
• Local data and instruction caches
• Up to 4-way set associative
• Up to 128KB
• Write-back and write-through cache write policy
• Memory management options including:
  – Region protection
  – Region protection with translation
  – MMU for the Linux operating system
  – Memory management unit (MMU) with translation look-aside buffers (TLBs), includes no-execute bit security support
  – MMU for the Linux operating system
  – Optional parity or ECC for all local memories

Optional queues and GPIOs bypass the bus for fast, efficient data I/Os

Two configuration click-box options allow Xtensa 11 processors to very quickly communicate data, control, or status information with RTL blocks or other Xtensa processors. These options are accessed as registers in the processor, so no separate load/store is required to operating on the data.

The GPIO32 configuration option adds two 32-wire ports to the Xtensa 11 processor (one input, one output) to quickly control and monitor peripherals or other logic in the system.

The QIF32 configuration option adds two 32-bit queue interfaces for FIFO-like data streaming into and out of the processor. The input queue functions with a familiar pop/empty/data interface to external logic while the output queue presents a similar push/full/data interface. All interactions with the Xtensa 11 processor pipeline are automatically implemented when the option is selected.

Figure 4. GPIO ports and queue interfaces allow direct data transfers for the fastest connections to RTL blocks or other Xtensa processors

The QIF32 configuration option adds two 32-bit queue interfaces for FIFO-like data streaming into and out of the processor. The input queue functions with a familiar pop/empty/data interface to external logic while the output queue presents a similar push/full/data interface. All interactions with the Xtensa 11 processor pipeline are automatically implemented when the option is selected.

Figure 5. The Xtensa 11 DPU can be easily extended with custom instructions and direct interfaces to RTL blocks and FIFOs
Add Flexibility and Extensibility to Your SoC Designs with Xtensa Processors

General-purpose processors offer fixed options for memory size, cache size, and bus interface. Performance is generally proportional to the clock speed. Beyond that, application code optimization or a move to the next-generation processor is required to get incremental performance benefits.

Cadence offers SoC designers the unique ability to add flexibility and longevity to their designs through software programmability as well as differentiation through processor implementations tailored for the specific application. You can now design a processor whose functions, especially its instruction set, can be extended to include features never considered or imagined by designers of the original processor, all using the TIE language.

The TIE language can be used to describe instructions, registers, execution units, and I/Os that are then automatically added to the processor. TIE is a Verilog-like language used to describe desired instruction mnemonics, operands, encoding, and execution semantics. TIE files are inputs to the Xtensa Processor Generator. The generator automatically builds the processor and the complete software tool chain that incorporates all configuration options and new TIE instructions. The base instruction set remains for maximum compatibility with third-party development tools and operating systems.

The TIE language unlocks the true power of the Xtensa DPU. It lets you get orders of magnitude performance increases for your applications and create differentiation. Extensibility with Xtensa DPUs allows features to be added or adapted in any form that optimizes the processor’s cost, power, and application performance.

**Flexibility—Add just what you need**

Just as the designer can choose from a set of predefined functional options to improve processor performance, the designer can now create instructions that can speed up standard or proprietary algorithms, and scale data interfaces for greater bandwidth. Using the tools provided, application hot spots can be identified and additional instructions created to process these hot spots more efficiently, without the need to increase the clock frequency or re-write a lot of the software.

**Differentiate—Make a processor that’s uniquely your own**

With fixed-function general-purpose processors, differentiation is often limited to the algorithm implementation itself. General-purpose processors are good at general-purpose computing, but not so good at any specific algorithm. Xtensa processors give you the opportunity to differentiate by giving you the ability to implement algorithms more efficiently by designing hardware that will accelerate your particular algorithm. This means that your design will be almost impossible to copy, as only your custom processor will reach the performance required on the same software implementation.

![Figure 6. Proven methodology automates the creation of customized processors and matching software tools](image-url)
Rapid design development, simulation, debug, and profiling

The Xtensa Xplorer IDE serves as the graphical user interface (GUI) for the entire design experience. From the Xtensa Xplorer IDE, designers with existing application software code can profile their application, identify hot spots, decide on configuration options, and add new instructions and execution units to optimize performance and generate a new processor—all within a matter of hours. No other IP provider puts such flexibility directly into the hands of the designer with a tool that integrates software development, processor optimization, and multi-processor SoC architecture in one IDE.

Hardware designers now have new options for implementing algorithms. Interfaces can be added to the processor to offer direct, deterministic connectivity to SoC logic. With the GPIO port and queue interface options, designers can stream data into or out of the processor. This direct connectivity with the rest of the SoC offers great control and predictable bandwidth. The simple C programs needed to control the Xtensa processor can be written and debugged within the Xtensa Xplorer IDE.

Hardware Development

Hardware designers can profile, compare, and save many different processor configurations. Use the ISS to simulate a single processor or, for multi-processor subsystems, choose the XTensa Modeling Protocol (XTMP) or XTSC modeling tools.

The Xtensa Xplorer IDE serves as the gateway to the Xtensa Processor Generator. Once a processor configuration is finalized, the Xtensa Processor Generator creates the automatically verified Xtensa processor to match all of the configuration options and extensions you have defined, in about an hour. The full software tool chain is also created that matches all processor modifications made. See the Xtensa Processor Developer’s Toolkit product brief for more information.

Figure 7. The Xtensa Xplorer IDE can display valuable information including performance comparisons, instruction sizes, and processor size, area, and power.
Complete hardware implementation and verification flow support

- Automatic generation of RTL and tailored EDA scripts for leading-edge process technologies, including physical synthesis and 3D extraction tools
- Auto-insertion of fine-grained clock gating delivers ultra-low power
- Hardware emulation support including automated FPGA netlist generation
- Comprehensive diagnostic testbench to verify connectivity
- Format verification support for designer-defined functions
- Pipeline-modeling, cycle-by-cycle accurate Xtensa ISS
- System modeling capabilities with optional XTMP and XTSC simulation environments
- Multiple-processor on-chip debug capable with break-in/out control
- Hardware co-simulation in System C with the Xtensa DPU's pin-level XTSC connectivity to RTL
- XTSC transaction-level modeling support, including out-of-the-box multi-core co-simulation

Software Development

The Xtensa Software Developer’s Toolkit (SDK) provides a comprehensive collection of code generation and analysis tools that speed the software application development process. The Eclipse-based Xtensa Xplorer GUI serves as the cockpit for the entire development experience and also provides powerful visualization tools to aid application optimization.

The entire Xtensa software development tool chain, along with simulation models, RTOS ports, optimized C-libraries, etc., are automatically generated by the Xtensa Processor Generator. This also ensures that all the software tools—such as the compiler, linker, assembler, debugger, and ISS—always match and are tuned exactly to any custom processor hardware.

Complete software development tools

- Mature, highly optimizing C/C++ compiler (XCC) that rivals hand-coded assembly applications on other processors
- GNU-based assembler and linker
- Pipeline-modeled, cycle-accurate ISS
- High-speed (40-80X faster than ISS) instruction-accurate TurboXim simulator speeds software development
- XTMP and XTSC for multi-processor simulation and modeling
- Debug offers full GUI and command-line support for single- and multi-processor designs
- Supported by many third-party JTAG probes
- XMON software debug monitor for real-time debugging
- Profiling views of the processor pipeline utilization as well as time spent in functions across multiple processors, allows “what if” comparisons
- Vectorization Assistant discovers and locates code that could not be vectorized along with an explanation that can help the programmer modify the code so that it can be vectorized
- Support for major operating systems including Mentor Graphics’ Nucleus Plus, Express Logic’s ThreadX, Micrium’s µC/OS-II, Sophia Systems’ µITRON, and open-source Linux

The entire Xtensa software development tool chain, along with simulation models, RTOS ports, optimized C-libraries, etc., are automatically generated by the Xtensa Processor Generator. This also ensures that all the software tools—such as the compiler, linker, assembler, debugger, and ISS—always match and are tuned exactly to any custom processor hardware.

Complete software development tools

- Mature, highly optimizing C/C++ compiler (XCC) that rivals hand-coded assembly applications on other processors
- GNU-based assembler and linker
- Pipeline-modeled, cycle-accurate ISS
- High-speed (40-80X faster than ISS) instruction-accurate TurboXim simulator speeds software development
- XTMP and XTSC for multi-processor simulation and modeling
- Debug offers full GUI and command-line support for single- and multi-processor designs
- Supported by many third-party JTAG probes
- XMON software debug monitor for real-time debugging
- Profiling views of the processor pipeline utilization as well as time spent in functions across multiple processors, allows “what if” comparisons
- Vectorization Assistant discovers and locates code that could not be vectorized along with an explanation that can help the programmer modify the code so that it can be vectorized
- Support for major operating systems including Mentor Graphics’ Nucleus Plus, Express Logic’s ThreadX, Micrium’s µC/OS-II, Sophia Systems’ µITRON, and open-source Linux

Figure 8. The Xtensa Xplorer IDE shows debug/trace, profiling of pipeline utilization, and a cycle comparison for a multi-core simulation
• Extensive set of low-level functions and macros for core and system-level initialization and control
• AXOM run-time utility that customers can include in their source code to help manage swapping sections of code in and out of memory in applications with large code base and small memories

Ideal for Applications Where Low Power is Critical

Power often is the key issue in an SoC design. Many techniques are employed to reduce power consumption, both built in to the base hardware and into the configuration options, allowing more control over system and memory resources. Xtensa processors consistently consume less power than other licensable embedded CPUs at equivalent gate counts.

Insertion of fine-grained clock gating for every functional element is automated, including those defined by the designer. This automation gives the Xtensa DPUs a significant advantage over RTL design where manual, error-prone post-layout tuning of clock circuits is often required.

Accessing local memories is one of the highest power-consuming activities. Xtensa 11 processors eliminate any unnecessary local memory interface activation if that memory is not directly addressed by the processor. With Xtensa 11 processors, designers can now do semantic and memory data gating to save dynamic power.

Caches are other blocks that may consume significant power. The Xtensa 11 processor allows caches to be implemented at configuration time, and provides a way to shut down parts of the cache to match the operating load on the processor. A programmer can turn off one, two, three, or all four of the cache “ways” to reduce dynamic power usage during idle or low-load periods, and turn them on again when they are needed.

As process geometries shrink, leakage power consumes a larger portion of the total power budget. To substantially reduce leakage power, the Xtensa 11 processor provides designers with options during processor configuration, which will:
• Instantiate a power control module (PCM) in the Xtmem level of design hierarchy
• Specify the number of power domains within the design and their operation via industry standard power format files

Implementation of these energy saving techniques is automated by the Xtensa Processor Generator.

The designer can configure the external data bus width and internal local memory data widths independently. This allows system-level power optimizations depending on whether the processor is constrained by external or internal instruction and data access.

Multi-processor features and debug options

Placing multiple processors on the same IC die introduces significant complexity in SoC software debugging.

All versions of the Xtensa processor have certain optional PIF operations that enhance support for multi-processor systems.

The Xtensa processor’s debug features include:
• Interfaces to support CoreSight infrastructure
• Multi-core OCD support
• Multi-core debug improvement including sharing single trace memory across multiple TRAX modules
• Hardware/software support for synchronous restart/resume, cross triggering, etc.
• DebugStall feature allows Xtensa processors to be debugged while in the stalled state

Access to these debug functions:
• Via JTAG
• Via APB
• From the Xtensa core itself

Some SoC designs use multiple Xtensa processors that execute from the same instruction space. The processor ID option helps software distinguish one processor from another via a PRID special register.

The break-in/break-out option for the Xtensa Debug Module simplifies multi-core debugging. This capability enables one Xtensa processor to selectively communicate a break to other Xtensa processors in a multi-processor system. The DebugStall feature allows Xtensa processors to be stopped and started together using a hardware signal and to be debugged while in the stalled state.

In addition to multi-processor debug, it is also possible to non-intrusively trace multiple processors if they are configured with the trace extraction and analysis tool, TRAX. TRAX, which is detailed in the Debug Guide, is a collection of hardware and software components that provides visibility into the activity of running processors using compressed execution traces. The ability to capture real-time activity in a deployed device or prototype is particularly valuable for multi-processor systems where there are a large number of interactions between hardware and software.

When multiple processors are used in a system, some sort of communication and synchronization between processors is required. The Xtensa Multi-processor Synchronization configuration option provides ISA support for shared-memory communication protocols.
The Performance Monitor module is used to count performance-related events, such as cache misses. Accessing the counts through JTAG or APB is non-intrusive, but it is also possible to configure an interrupt to software running on an Xtensa processor.

**Specifications**

Because it is highly customizable, an Xtensa 11 DPU can run very efficiently at low MHz and very fast at over 1GHz.

The latest EDA tools, process flows, and other input are tracked to provide detailed performance information. For the latest data, please contact your local representative. See [ip.cadence.com](http://ip.cadence.com) for contact details.