cadence®

Tensilica Datasheet

Xtensa LX6 Customizable DPU

High performance with flexible I/Os and wide data fetches

Cadence provides system-on-chip (SoC) designers with the world's first and only configurable and extensible processor cores fully supported by automatic hardware and software generation. Cadence® Tensilica® Xtensa® processors, such as the Xtensa LX6 dataplane processing units (DPUs), enable SoC designers to add flexibility and longevity to their designs through software programmability as well as differentiation through processor implementations tailored for the specific application.

Features

- Highly efficient, small, low-power 32-bit base architecture
- Configurable over a wide range of pre-verified options including 10 different digital signal processing (DSP) choices
- Extend with designer-defined, application-specific instructions, execution units, register files, and I/Os
- Virtually unlimited I/O bandwidth with multiple, wide, designer-defined FIFO, GPIO, and lookup interfaces
- Selectable 5- or 7-stage pipeline depth for core instruction set architecture (ISA), plus extended DSP pipelines up to 11 stages
- Local memories configurable up to 8MB with option for memory parity or ECC
- Up to 128b-wide flexible-length instruction extensions (FLIX) instructions
- Multi-core on-chip debug (OCD) with break-in/break-out
- Dual-load/stores each up to wide with data cache support and multi-bank RAM support
- Power domains for power shut-off
- Semantic and memory data gating
- Compatible interfaces for ARM® CoreSight™ debug and trace technology
- IEEE 754-compliant single-/double-precision scalar floating-point unit
- Complete matching software development tool chain automatically generated for each core

Benefits

- Develop hardware for complex dataplane processing significantly faster compared to pure RTL methods
- High-bandwidth data flow through processor with flexible I/O interfaces that are independent of the system bus
- Quickly and easily scale hardware architecture with task-customized processors
- Lower verification effort with pre-verified, correct-by-construction RTL generation
- Post-silicon programmability
- Accurate high-speed processor and system simulation models automatically created for software development
- Simplified multi-core debugging
- Huge bandwidth, more parallelism to reduce cycle counts
- Low-leakage power design
- Dynamic power savings
- Easy integration into an ARM CoreSight interface-based debug and trace infrastructure
- Single-/double-precision scalar floating-point options to match the exact application requirements
- Mature, highly optimizing C/C++ compiler means you can work at the 'C' level for most applications

Processors for the Challenges of the SoC Dataplane

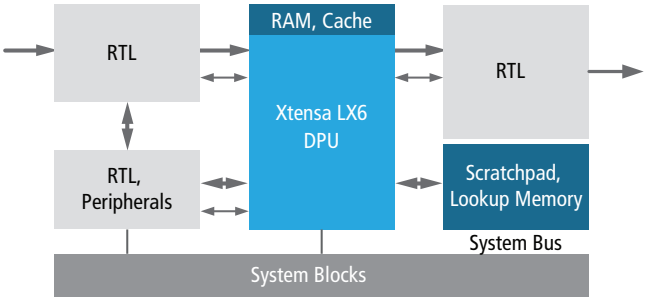

Figure 1. Xtensa LX6 DPU: Flexible direct connections allow RTL-like throughput

Inside today's complex systems on chips (SoCs), you can find many different processors from general-purpose processors to function-specific offload engines that add programmability and flexibility. Although general-purpose embedded processors can handle most of the control tasks well, they lack the bandwidth needed to perform complex, data-processing tasks such as network packet processing, video processing, and digital cryptography. Chip designers have long turned to hardwired logic (blocks of RTL) to implement these key functions. The problem with the RTL blocks is that they take too long to design, take even longer to verify, and are not programmable.

Xtensa LX6 DPUs are configurable and extensible and ideal for handling complex compute-intensive digital signal processing (DSP) applications where a register-transfer level (RTL) implementation may be the only other option.

Configurable

You are offered a menu of pre-verified checkbox and drop-down options ranging from memory size and width to complex DSP functions.

Extensible

You can use the Tensilica Instruction Extension (TIE) methodology, based on the Verilog language, to implement datapath elements in the processor pipeline and add more I/Os. The control finite state machine (FSM) for datapath elements is implemented as software running on the processor. Just specify the functional behavior of the datapath and the RTL is automatically generated, along with the full matching software tool chain and models.

Feature Overview

- Modern ISA with true multi-generational compatibility
- Xtensa ISA fundamentally architected for extensibility
- Base instruction set of 80 RISC 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
- IEEE 754-compliant single-/double-precision scalar floating-point unit
- Double-precision scalar floating-point acceleration
- 3-way 64-bit FLIX (FLIX3) for interleaved very long instruction word (VLIW) and regular instructions
- Pre-defined 32-bit GPIO and FIFO-like queue interfaces

Optional execution units (additional licensing)

- ConnX D2 DSP engine
- ConnX Vectra LX DSP engine
- ConnX Vectra VMB for baseband acceleration
- ConnX BBE16, BBE32-EP, and BBE64-EP baseband engines
- HiFi-3, HiFi EP, HiFi-2, and HiFi Mini Audio/Voice DSPs
- IVP-EP 32-way SIMD Imaging/Video DSP

Differentiate with designer-defined instructions

- Make your specific algorithm run even more efficiently by adding the instructions it needs
- Development tools automatically adapt for full support

Natural connectivity with RTL blocks

- 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 with in-bound slave DMA option
- Optional ARM AMBA® AXI and AHB-Lite interfaces with synchronous or asynchronous clocking
- Write buffer, selectable from 1 to 32 entries
- Up to 128b-wide instructions and up to two 512b-wide load/stores and hardware prefetch unit
- Optional second data load/store unit with data cache support
- Choice of 1-, 2-, or 4-way cache and local memories
- Up to 32 interrupts

Multi-core design style support

- Multi-core system creation, modeling, and SystemC co-simulation out-of-the-box, fully supported within the Xtensa Xplorer™ integrated design environment (IDE)
- Homogenous and heterogeneous subsystems supported
- Inter-core OCD with break-in/out control
- 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 test bench 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 support, including out-of-the-box multi-core simulation
- Hardware co-simulation with RTL in SystemC with pin-level XTSC

IDE

- 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 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
- OCD 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 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.

Dynamic and leakage power improvements

- Power shut off (PSO) feature allows Xtensa DPUs to be completely powered off. To help achieve low leakage, Xtensa DPUs can now be divided into multiple “power domains” and 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
- Software cache way usage control allows a programmer to adjust cache dynamic power on the fly

Robust real-time operating system support

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

Efficient Base Architecture

The Xtensa LX6 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 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
- Modelessly intermixed standard 16- and 24-bit instructions, as well as designer-defined FLIX instructions of any size from 4 to 16 bytes, resulting in highly efficient code that is optimal for both memory size and performance
- Selectable 5-or-7-stage core ISA pipeline to accommodate different memory speeds, plus extended DSP execution pipelines up to 11 stages, and designer-defined instruction pipeline depths up to 23 stages
- Virtually unlimited I/O bandwidth with optional queue (FIFO), port (GPIO), and lookup interfaces for data transfers that are not dependent on the limited system bus bandwidth
- One or two 32-/64-/128-/256-/512-bit-wide load/store units
- Local memories configurable up to 8MB with optional parity or ECC
- Optional hardware prefetch reduces memory latencies
- Automated fine-grained clock gating throughout processor for ultra-low power solutions
- Can be multi-issue VLIW architecture for parallel instruction execution with FLIX

Base ISA compatibility

Configurability of an Xtensa processor core builds on 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 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++

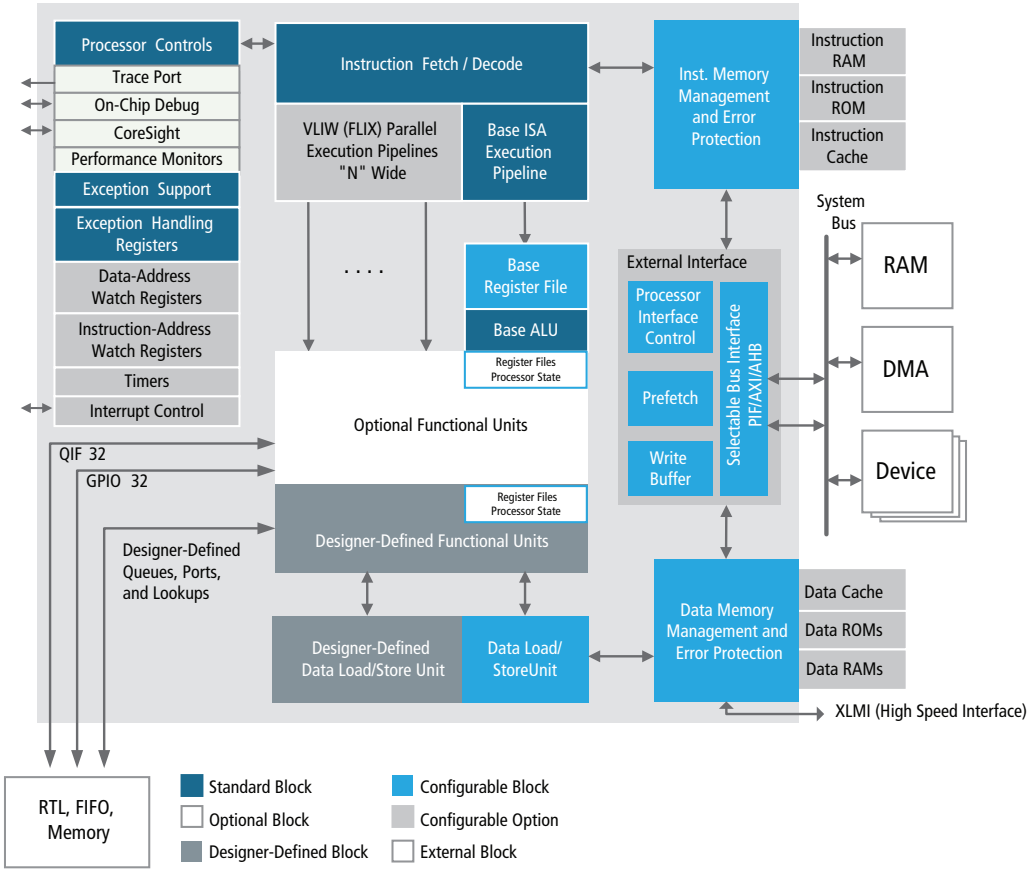

Figure 2: Xtensa LX6 DPU showing standard, optional, and designer-defined blocks

compiler, a cycle-accurate SystemC-compatible ISS, and the full industry-standard GNU tool chain.

Xtensa processors use an ISA that has been backwards compatible since its introduction in 1998. It uses a base instruction set of 80 instructions and was fundamentally architected for extensibility. Designers can run application code written back in 1998 and it will run on the Xtensa LX6 processor today. Any differentiating designer-defined instructions from earlier designs can be re-used today.

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 process the fields in packet headers and perform rule-based checks.

Extensible ISA

One of the fundamental technology innovations in the Xtensa processor is the ability to easily and seamlessly add instructions into the processor's datapath. Any associated C data types, the 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 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

Xtensa DPUs offer pre-verified options that you can add to your designs when they are needed. Configurable ISA options include:

- Single 16-bit MAC (multiply accumulator)
- 16- or 32-bit multipliers
- IEEE 754-compliant single-/double-precision scalar floating-point option
- Double-precision scalar floating-point acceleration
- A pair of 32-bit direct GPIO interfaces (GPIO32)
- A pair of 32-bit FIFO-like queue direct interfaces (QIF32)
- 3-way 64-bit VLIW (FLIX3)

Select from click-box options to add functionality to your processor and evaluate performance improvements quickly.

Basic interface options include:

- Designer-defined queues, ports, and lookups
- PIF, AMBA AXI, and AMBA AHB-Lite protocol options with synchronous or asynchronous clocking
- Width of 32-/64-/128-bit

- Optional "no PIF" configuration
- Inbound DMA
- XLMI high-speed local interface
- Big-Endian/Little-Endian byte ordering
- Choice of one or two general-purpose load/store units, each 32-,64-, 128-, 256-, or 512-bits wide
- OCD port (IEEE 1149.1 or CoreSight-compatible debug APB interface)
- Trace port signals
- Up to 32 interrupts with up to seven levels of priority, plus a separate non-maskable interrupt level
- Write buffer, selectable from 1 to 32 entries
- Multiple custom-width GPIO ports for direct control and monitoring of peripherals
- Multiple custom-width queue interfaces for streaming data into and out of the processor via FIFOs
- 16-bit processor ID
- Support of FLIX instructions in widths of up to 128 bits

Memory subsystem options include:

- Dual load/store with data cache support
- Multibank RAM support
- Single-cycle or dual-cycle access speeds
- Local data and instruction caches
  - Up to 4-way set associative
  - Up to 128 KB
  - Write-back and write-through cache write policy
- Memory management unit (MMU) with translation look-aside buffers (TLBs), includes no-execute bit security support
- Memory management options including
  - Region protection
  - Region protection with translation
  - MMU for the Linux operating system
- Up to six 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
- Hardware prefetch for reducing long memory latencies
- Optional parity or ECC for all local memories

In addition to these options, there are several pre-configured major functional blocks that are licensed in addition to Xtensa LX6 DPUs:

- HiFi Audio/Voice DSP cores—The industry's most popular audio subsystems with a library of over 100 audio-, voice-, and sound-enhancement software packages
- IVP Imaging and Video DSP core—Ultra-high performance DSP core for demanding imaging and video applications
- ConnX BBE16 DSP core—For LTE baseband processors in cellular radios and multi-standard broadcast receivers
- ConnX BBE32EP DSP core—A very-low-power 32-MAC DSP core designed for LTE-Advanced and HSPA+ handsets
- ConnX BBE64EP DSP core—The most powerful DSP core Cadence offers, with up to 128 MACs for use in LTE-Advanced baseband
- ConnX D2 DSP engine—For 16-bit communications DSP functions, delivers outstanding performance from 'C' code

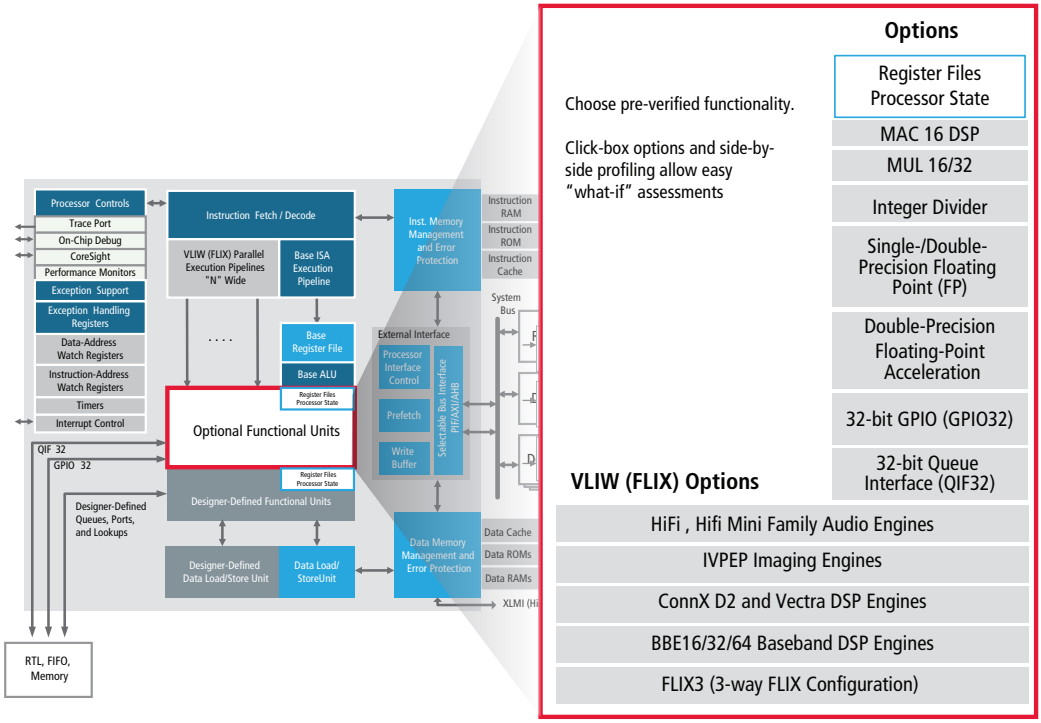

Figure 3. Widest range of configurable functional units for the Xtensa LX6 DPU

- ConnX Vectra DSP engine—For medium-performance 16-bit communications DSP functions, uses 64-bit instruction words containing three issue slots for ALU, multiply-accumulate, and load/store operations

Add Flexibility and Extensibility to SoC Designs with Xtensa DPUs

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. The TIE language 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 you can choose from a set of predefined functional options to improve processor performance, you 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 DPUs give you the opportunity to differentiate by implementing algorithms more efficiently with hardware that accelerates 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.

FLIX for parallel execution

Many of the major pre-configured functional blocks take advantage of Xtensa LX6 DPU's FLIX capabilities.

The FLIX architecture makes the Xtensa LX6 DPU into a VLIW processor that executes 2 to 30 parallel execution units when needed. FLIX instructions can be as small as 4 bytes, as large as 16 bytes, or any size in between. These variable-width FLIX instructions are seamlessly intermixed with the base Xtensa 16-/24-bit instructions, so there is no mode switch penalty when using FLIX.

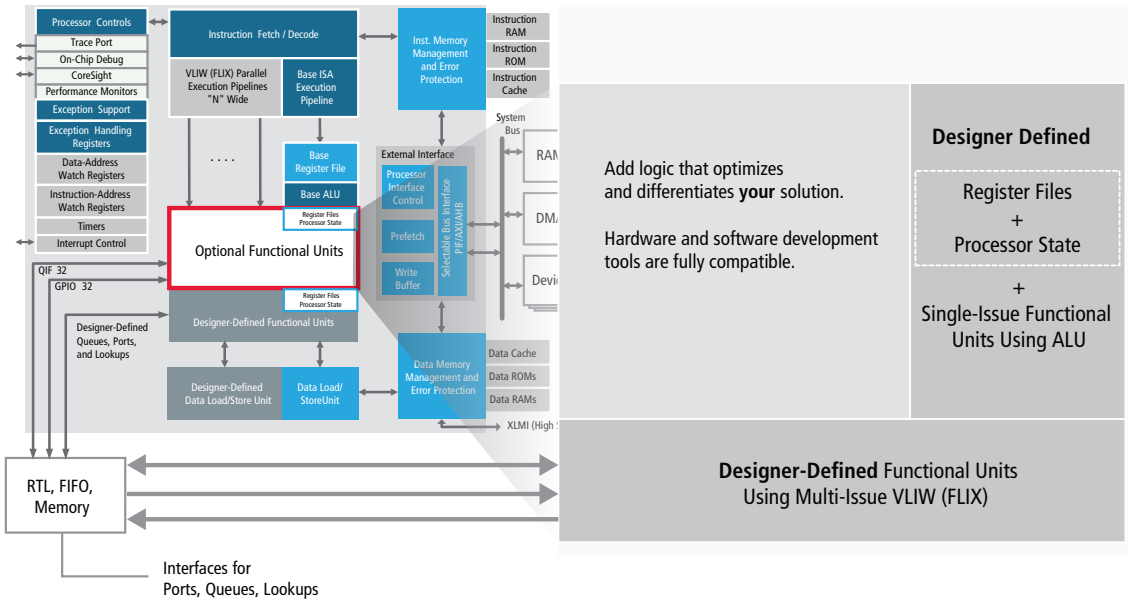

Figure 4. Xtensa LX6 DPU offers a proven method of adding designer-defined functional units and interfaces

With FLIX, the Xtensa LX6 processor can deliver the ultra-high performance characteristics of a specialty ultra-wide instruction word processor without the negative code size implications typically found in such VLIW or ultra-long instruction word (ULIW) processors. In fact, Xtensa LX6 processors with FLIX can often deliver higher performance and smaller code size at the same time. This performance comes with very little overhead—adding only 2,000 gates to the size of the processor for instruction decode and control.

The Xtensa C/C++ Compiler automatically extracts parallelism from source code and bundles multiple operations into FLIX instructions so the benefits can be realized without additional software development. In this way, a 3-issue Xtensa LX6 processor running at a lower frequency can deliver the performance that's equivalent to that of a significantly higher clocked processor. Additionally, the compiler can bundle the branch and load/store instructions in parallel with compute instructions to gain a performance boost over straight-line code.

Designers can add additional capabilities in the major pre-configured functional blocks by using the FLIX capabilities in their own design to specify exactly what's needed.

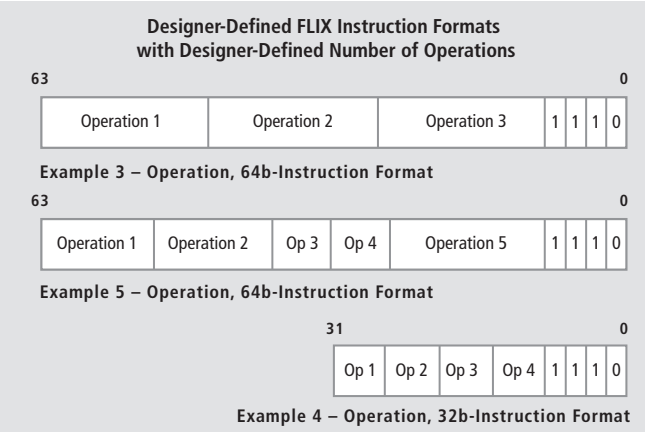

Figure 5. Designers can use FLIX to create VLIW instructions up to 128 bits wide to execute 2 to 30 parallel execution units

Designer-defined I/Os bypass the system bus for maximum data throughput

Xtensa processors bring another fundamental breakthrough in embedded processor designs—the ability to define direct data interfaces into and out of the processor for maximum data throughput. This ability is a key reason that Xtensa DPUs are ideal for the SoC dataplane. Xtensa processors provide three direct interface capabilities:

- TIE ports provide direct (GPIO) connection to other logic within an SoC or to other Xtensa processors, and are created with simple one-line declarations in a TIE file
- TIE queues function like FIFO interfaces, with a familiar pop/empty/data interface to external logic while TIE output queues present a similar push/pull/data interface. All interactions with the Xtensa processor pipeline are automatically implemented by the Xtensa Processor Generator
- TIE lookups let you connect RAMs or external devices to Xtensa processors. These external memories or devices can be accessed directly from the processor's datapath without using load/store instructions. These interfaces are useful for connecting table lookup RAMs, for example in networking applications, or for connecting long-latency hardware computation units.

Port connections can be up to 1024 wires wide, allowing wide data types to be transferred easily without the need for multiple load/store operations. As many as one million signals (1000 1024-bit-wide ports) can be used. While this number far exceeds the performance demands of real systems today, this clearly demonstrates that the conventional I/O bottlenecks inherent in a system-bus-based solution do not apply to Xtensa processors.

While ports are ideal to quickly convey control and status information, queues provide a high-speed/low-latency mechanism to transfer streaming data with buffering. Input queues and output queues operate, to the programmer's viewpoint, like traditional processor registers—without the bandwidth limitations of local and system memory accesses.

TIE port and queue wizard

The Xtensa Xplorer IDE provides a wizard for quickly generating ports and queues without the need to write any TIE code.

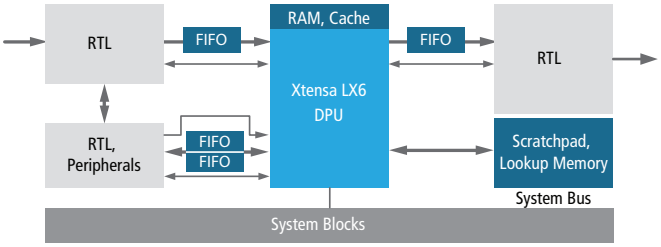

Figure 6. Example of direct FIFO and port connections using TIE queues and TIE ports

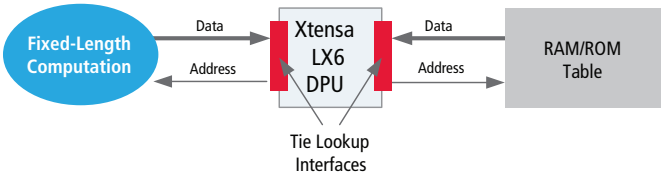

Figure 7. Example of TIE lookups showing connections to memory and logic

Xtensa LX6 DPU as an RTL Companion

RTL verification has become the most resource- and time-consuming aspect of SoC design. Xtensa processors offer unique advantages to SoC designers where they can use a pre-verified IP core as a foundation and add custom extensions through correct-by-construction design techniques. This design approach significantly reduces the need for the long verification times required when designing custom RTL. Xtensa processors can connect directly to your RTL with dedicated high bandwidth data and control interfaces.

Bandwidth of hard-wired logic and performance without hand-coded state machines

The Xtensa processor can achieve virtually the same levels of inter-block I/O bandwidth and intra-block computational parallelism as hard-wired logic designed with traditional RTL design methodologies. How? By using a combination of TIE ports and queues, parallel FLIX execution units, and some TIE instructions.

Unlike RTL-based designs, Xtensa processors are pre-verified, and do not require hard-wired implementation of complex state machines. Instead of state machines, the datapaths are sequenced and controlled by the processor's instruction stream. That means the "control logic" is fully programmable and can be debugged using software development methodologies, thereby reducing verification time and risk for the entire SoC.

Lower verification effort and time

Designing hardwired RTL blocks has become more about verification than about design. Design teams typically spend twice the number of resources and person months on verification than on design. Design changes made late in the project cycle are often limited by the verification effort.

Typically, 90% of the RTL block's area lies in the datapath and only 10% in the control logic, yet most (perhaps 90%) of the bugs are found in the control logic. The ability to extend the Xtensa processor using TIE specifications enables designers to create datapaths inside the processor without the need to generate and verify the associated control logic. Instead the control logic is expressed in software as instructions that execute on the processor.

It is easier to verify TIE specifications made to the Xtensa processor than it is to verify an equivalent RTL datapath, since only the I/O relationship and functional behavior of the operations specified in TIE code have to be verified. The TIE Compiler and Xtensa Processor Generator take care of converting the TIE specification into data path elements in the processor pipeline and implementing the control, decode, and bypass logic in the processor control units.

Reuse of the same hardware for multiple tasks

Complex SoCs consist of millions of gates of logic and are designed to perform multiple tasks. Often these multiple tasks do not need to be performed at the same time. This provides an opportunity for multiple tasks to share the same hardware units. Processors are particularly amenable to enabling this type of sharing.

Designers can specify a datapath in the TIE specification that consists of a set of execution units that can be used by multiple tasks and then use the programmability of the processor to determine which tasks are executed. For example, an audio engine can be designed to implement a range of audio codecs, such as MP3, AC-3, WMA, etc.

Flexibility to fix and upgrade algorithms post-silicon

An Xtensa processor implementation of an algorithm lets the designer fix, enhance, and tweak the algorithm even after the SoC has taped out. In particular, post-silicon bugs now have a chance of being worked around. Algorithms that are a subject to continuous research, such as half-toning in printers and image and video post-processing, are ideal candidates for implementation in an Xtensa processor.

Using Xtensa DPUs, you can easily add functionality to an existing design, or upgrade parts of it to support the latest standard, with limited development effort.

Co-simulation at the RTL pin level

Connect directly to your RTL wires using pin-level XTSC SystemC model interfaces without the need to purchase additional EDA vendor tools. This enhancement to transaction-level XTSC models allows designers to interchange SystemC and RTL blocks for co-simulation. This works with all of the major EDA vendor simulation tools.

Extending the Life of an Existing RTL Design

Using Xtensa DPUs, you can easily add functionality to an existing design, or upgrade parts of it to support the latest standard, with modest development effort.

Conventional processor SoC with RTL

With any other 32-bit processor core, all communication is through the system bus, which must have the available data bandwidth and must keep bus latency manageable.

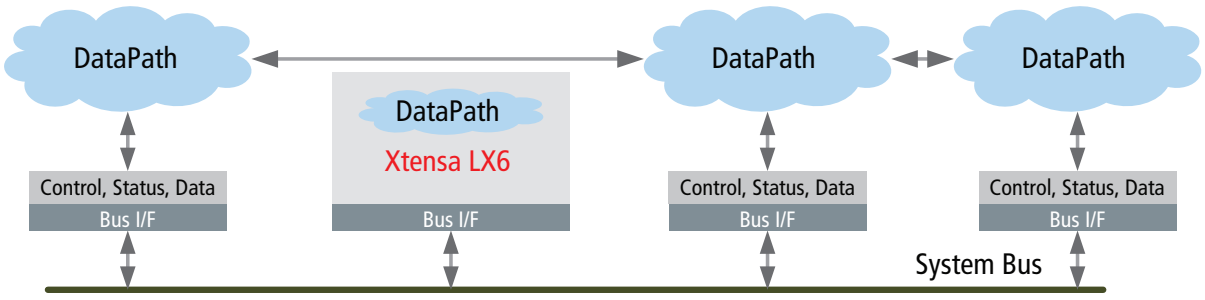

Figure 8. All communication through the system bus

Add functionality with Xtensa DPUs

With Xtensa DPUs, data can be kept off the system bus by using direct connectivity to RTL through ports and queues. These provide almost unlimited bandwidth with precise latencies.

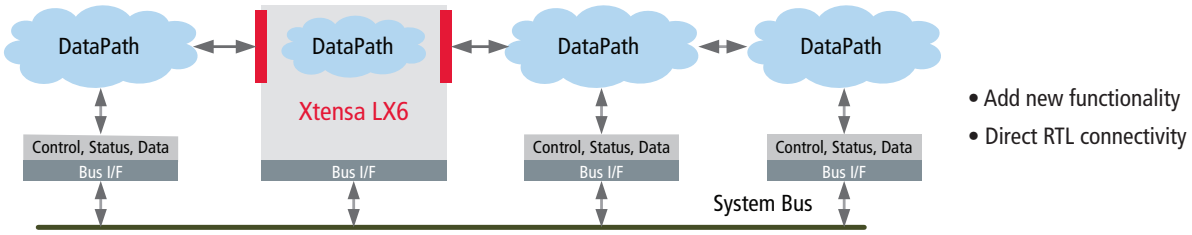

Figure 9. Direct connectivity to RTL through ports and queues

When extending the functionality of existing RTL blocks, the control logic parts can be brought into the processor to make the FSM easier to debug and verify.

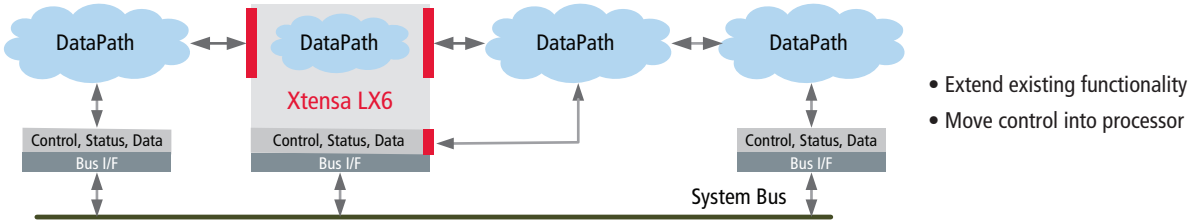

Figure 10. Control logic parts brought into the processor make FSM easier to debug and verify

The datapath of the existing RTL module can also be brought into the processor as a datapath extension to create a highly optimized solution. Both the control and datapath of the RTL block are brought into the processor

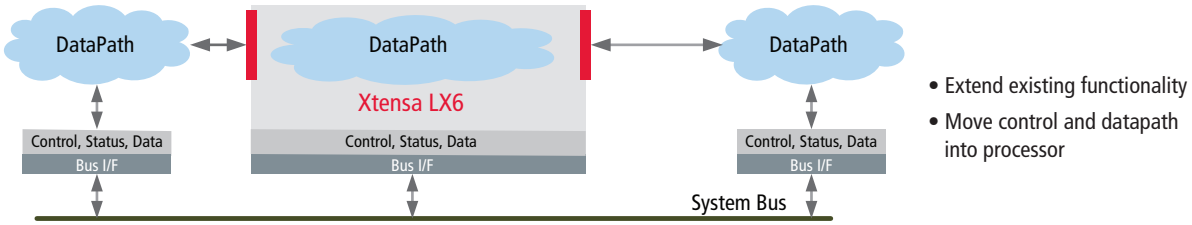

Figure 11. Both the control and datapath of the RTL block are brought into the processor

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 can profile their application, identify hot spots, decide on configuration options, add instructions and execution units to optimize performance, and then 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 creative options for implementing algorithms. Interfaces can be added to the processor to offer direct, deterministic connectivity to SoC logic. With the customizable port and queue interfaces, 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.

The Xtensa Processor Generator creates a complete hardware design with matching software tools, including a mature, world-class compiler, a cycle-accurate SystemC-compatible ISS, and the full industry-standard GNU tool chain.

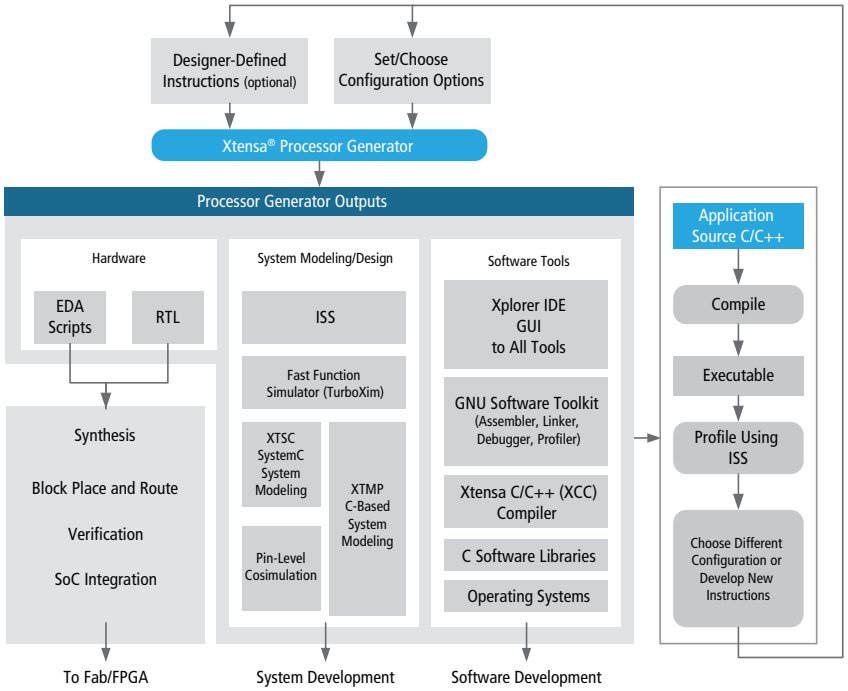

Figure 12. This proven methodology automates the creation of customized processors and matching software tools

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 Cadence's XTensa Modeling Protocol (XTMP) or XTSC modeling tools.

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 Processor Developer's Toolkit product brief for more information.

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 test bench 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 OCD-capable with break-in/-out control
- Hardware co-simulation in SystemC with Xtensa'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 Xtensa C/C++ compiler 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

Figure 13. The Xtensa Xplorer IDE can display valuable information including performance comparisons, instruction sizes, and processor size, area, and power

- Debug offers full GUI and command-line support for single- and multiple-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, and open-source Linux
- 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 a 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 LX6 processors eliminate any unnecessary local memory interface activation if that memory is not directly addressed by the processor. With Xtensa LX6 DPUs, you can now do semantic and memory data gating to save dynamic power.

Caches are other blocks that may consume significant power. Xtensa LX6 processors allow caches to be implemented at configuration time, and provide a way to shut down parts of the cache to match the operating load on the processor.

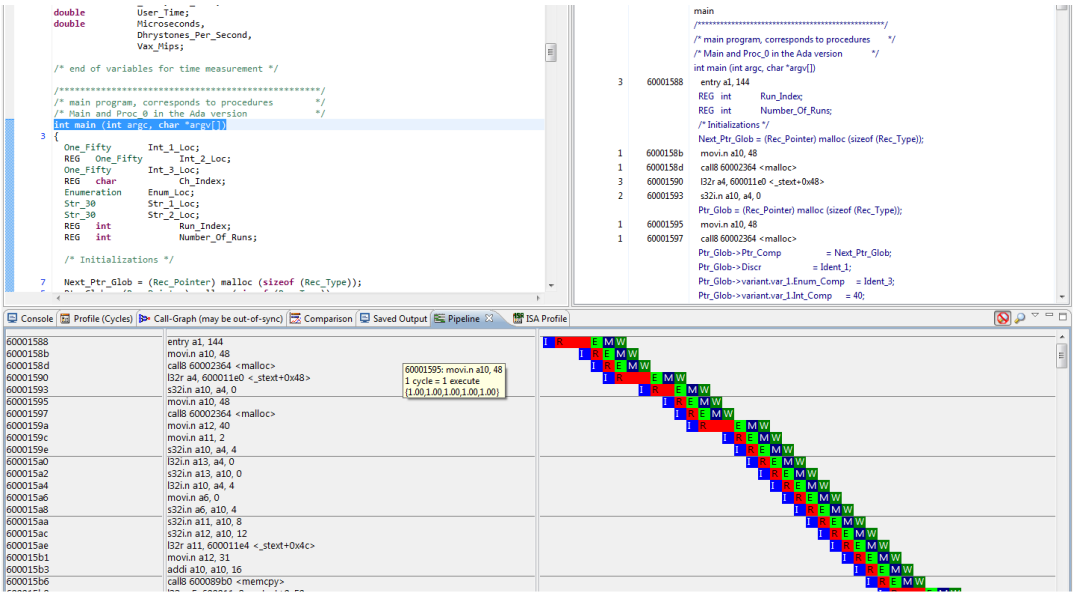

Figure 14. Xtensa Xplorer IDE shows debug/trace, profiling of pipeline utilization, and a cycle comparison for a multiple core simulation

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, Xtensa LX6 DPUs give you options during processor configuration. Implementation of the following energy-saving techniques is automated by the Xtensa Processor Generator:

- 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

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 is:

- 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 multiple-processor system. A 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 Multiprocessor 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 DPU.

Specifications

Because it is highly customizable, an Xtensa DPU can run very efficiently at low MHz and very fast at clock frequencies over 1GHz. Maximum achievable clock speeds vary with the choice of process technology, cell library, feature set, and EDA optimization techniques.

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 at ip.cadence.com/about/contact.

cadence®

Cadence Design Systems enables global electronic design innovation and plays an essential role in the creation of today's electronics. Customers use Cadence software, hardware, IP, and expertise to design and verify today's mobile, cloud, and connectivity applications. **www.cadence.com**