IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
TYLER DIVISION

LUCIO DEVELOPMENT LLC,	§	
	§	
Plaintiff,	§	Case No:
	§	
vs.	§	PATENT CASE
	§	
CADENCE DESIGN SYSTEMS, INC.,	§	
	§	
Defendant.	§	
	§	

COMPLAINT

Plaintiff Lucio Development LLC ("Plaintiff" or "Lucio") files this Complaint against Cadence Design Systems, Inc. ("Defendant" or "Cadence") for infringement of United States Patent No. 7,069,546 (hereinafter "the '546 Patent").

PARTIES AND JURISDICTION

1. This is an action for patent infringement under Title 35 of the United States Code. Plaintiff is seeking injunctive relief as well as damages.

2. Jurisdiction is proper in this Court pursuant to 28 U.S.C. §§ 1331 (Federal Question) and 1338(a) (Patents) because this is a civil action for patent infringement arising under the United States patent statutes.

3. Plaintiff is a Texas limited liability company with its office address at 555 Republic Dr., Suite 200, Plano, Texas 75074.

4. On information and belief, Defendant is a Delaware corporation with its principal office at 2655 Seely Ave., Building 5, San Jose, CA 95134. On information and belief, Cadence may be served with process by serving its registered agent in Texas, CT Corp

System at 350 N. St. Paul Street, Dallas, TX 75201.

5. This Court has personal jurisdiction over Defendant because Defendant has committed, and continues to commit, acts of infringement in this District, has conducted business in this District, and/or has engaged in continuous and systematic activities in this District.

6. On information and belief, Defendant's instrumentalities that are alleged herein to infringe were and continue to be used, imported, offered for sale, and/or sold in this District.

VENUE

7. Venue is proper in this District pursuant to 28 U.S.C. §1400(b) because acts of infringement are occurring in this District and Defendant has a regular and established place of business in this District. For instance, on information and belief, Defendant has a regular and established place of business at 6400 International Pkwy # 1500, Plano, TX 75093.

COUNT I
(INFRINGEMENT OF UNITED STATES PATENT NO. 7,069,546)

8. Plaintiff incorporates paragraphs 1 through 7 herein by reference.

9. This cause of action arises under the patent laws of the United States and, in particular, under 35 U.S.C. §§ 271, *et seq*.

10. Plaintiff is the owner by assignment of the '546 Patent with sole rights to enforce the '546 Patent and sue infringers.

11. A copy of the '546 Patent, titled "Generic Framework for Embedded Software Development," is attached hereto as Exhibit A.

12. The '546 Patent is valid, enforceable, and was duly issued in full compliance with Title 35 of the United States Code.

13. On information and belief, Defendant has infringed and continues to infringe

one or more claims, including at least Claim 1, of the '546 Patent by making, using, importing, selling, and/or offering for sale a software platform for embedded software development, which is covered by at least Claim 1 of the '546 Patent. Defendant has infringed and continues to infringe the '546 Patent directly in violation of 35 U.S.C. § 271.

14. Defendant, sells, offers to sell, and/or uses embedded software development packages including, without limitation, Tensilica's Eclipse-based Xtensa Xplorer Integrated Development Environment (IDE) ("Xtensa"), and any similar products ("Product"), which infringe at least Claim 1 of the '546 Patent.

15. xTensa is a Graphical User Interface (GUI)-based initialization, configuration and driver code generator for Tensilica Xtensa Dataplane Processors (DPU) such as Xtensa LX series processors. xTensa provides one or more generic application handler programs, each such program comprising computer program code for performing generic application functions common to multiple types of hardware modules used in a communication system. For example, xTensa provides a Hardware Abstraction Layer (HAL) which includes files such as Library Loader ("generic application handler") during installation. Header files include source code comprising functions and data structure, which are common and uniform across all supported Xtensa Dataplane Processor. Certain elements of this limitation are illustrated in the screenshots below and in the screenshots referenced in connection with other elements herein.

cadence

Tensilica Datasheet

Tensilica Software Development Toolkit (SDK)

Quickly develop application code

Features

- Cadence® Tensilica® Xtensa® Xplorer™ Integrated Development Environment (IDE) with full graphical user interface (GUI)
- Mature, optimizing Xtensa C/C++ Compiler (XCC)
- Operator overloading support in C for custom data types
- Pipeline-modeling, cycle-accurate instruction set simulator (ISS) with fast TurboXim
- GNU profiler, linker, debugger, assembler, and utilities
- Multi-processor subsystem simulation, debug, profiling, and memory partitioning
- Vectorization Assistant for locating code loops that need restructuring to enable vectorization

Software Development Tools for Cadence Tensilica DPUs

If you've looked at Tensilica's website or processor product briefs, you know that you can extend Tensilica's Xtensa dataplane processors (DPUs)—adding instruction sets, execution units, and processor I/O interfaces—to match your specific application needs.

By customizing the DPU for a particular application, you can often get significantly lower energy consumption and 10-100X performance increases. This level of performance and efficiency is often essential in the SoC dataplane. By customizing the DPU, you create a core that's uniquely yours, giving you extra protection in today's highly competitive marketplace.

Source: https://ip.cadence.com/uploads/103/SWdev-pdf, page 1.

- Project management tools
- Performance and energy analysis tools
- Use Mentor Graphics NucleusPLUS, Express Logic's ThreadX, Micrium's uC/OS-II, T-Engines' µT-Kernel, or the Linux operating systems

Benefits

- Easy-to-use Xtensa Xplorer IDE based on familiar Eclipse platform
- Small, high-performance code from 'C' source
  - Compiler offers state-of-the-art inter-procedural and alias analysis
  - Automatic vectorization of operations for Xtensa SIMD processors
  - Automatic Flexible Length Instruction eXtension (FLIX) instruction bundling for multi-issue Xtensa very long instruction word (VLIW) cores
- Detailed pipeline analysis guides optimizations from cycle/pipeline-accurate ISS
- Fast TurboXim simulation for up to 50 million instructions per second
- Vectorization Assistant guides code optimizations for better SIMD performance
- Easily and quickly evaluate multi-processor subsystems
- Familiar GNU-based toolchain

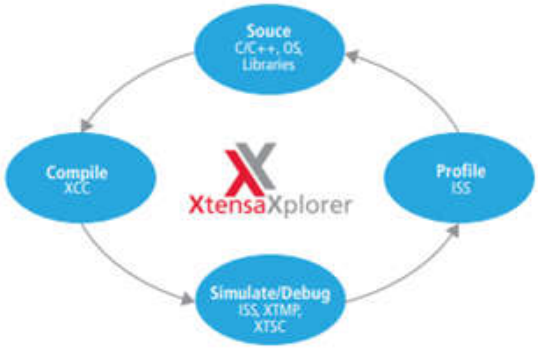


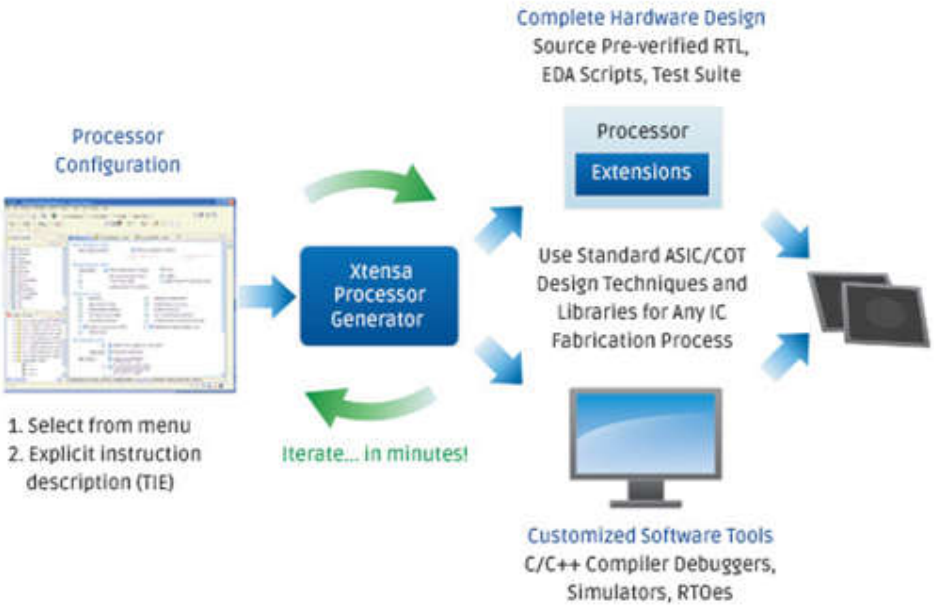
Figure 1: Tensilica's Eclipse-based Xtensa Xplorer IDE serves as the cockpit for custom processor development.

The Xtensa Processor Developer's Toolkit is the integrated design environment that delivers powerful tools to your desktop to guide you through the processor customization process. You'll find that Tensilica has created the most advanced, powerful, and easy-to-use tools for processor customization.

Source: https://ip.cadence.com/uploads/103/SWdev-pdf, page 1.

Automated Creation of a Data Processing DSP and its Software Toolkit in Less than 1 Hour

We have automated much of the risk out of creating a customized processor. Using our tools, designers can create a customized core and matching software tools in less than an hour.



Our automated process for creating customized dataplane processors

Source: https://ip.cadence.com/knowledgecenter/know-ten/dataplane-design

Tensilica Customizable Processors

Make a Processor Uniquely Your Own

What are the WOW factors you need in your SoC design? For next-generation mobile devices and home entertainment products, you need efficient, high-performance functional blocks that are programmable to keep up with the latest standards.

Use our proven, automated processor generator to customize a Cadence® Tensilica® processor, and create more competitive and differentiated features with the lowest possible power.

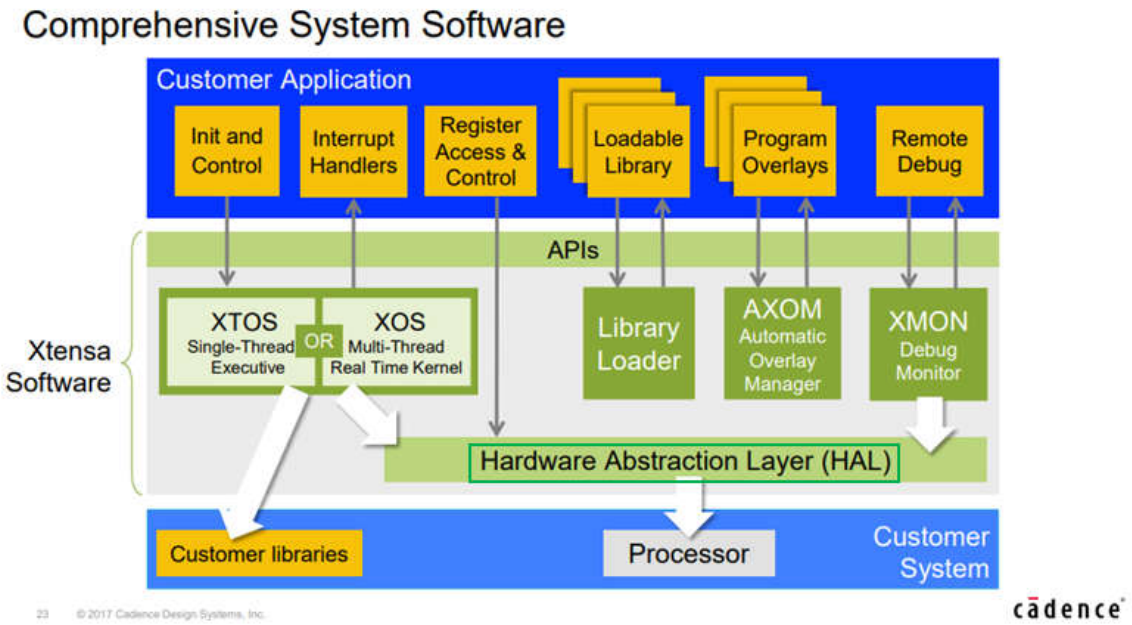
- Create a single product for multiple markets
- Reduce development time and cost with pre-verified processors
- Extend product life cycles, make software changes without a re-spin

New Xtensa LX7

The Xtensa® LX7 processor release delivers the latest version of the the Tensilica processor platform and introduces the new Vision P6 DSP for image and CNN processing, and the new Fusion G3 DSP for general purpose fixed and floating point applications. Xtensa LX7 delivers enhancements to the industry-leading ConnX BBE DSPs for baseband and radar applications, with a new vector floating-point option that features patent-pending innovations for improved area and power efficiency. All Tensilica DSPs are built on top of the Xtensa LX7 processor platform.

Additionally, the Xtensa LX7 increases common controller performance benchmarks by over 15%, while easing SoC design challenges with numerous architectural enhancements. Those include a new integrated DMA (iDMA) controller, broader support of the AMBA AXI4 protocol, thereby simplifying the integration of Tensilica processors with application processors, interface IP and the associated complex interconnect fabrics.

Source: https://ip.cadence.com/ipportfolio/tensilica-ip/xtensa-customizable



Source: https://nmi.org.uk/wp-content/uploads/2017/09/CADENCE_Practical-usage-of-Configurable-DSPs-in-High-Performance-Systems.pdf, page 23.

Multi-processor subsystems

The Xtensa Xplorer IDE provides multi-processor projects that allow the designer to create a subsystem of heterogeneous cores with shared memory. The memory partitioning for each core and the shared memory area are specified in the GUI to make that task simple.

Simulation of the resultant system is launched from within the IDE and allows the software developer to debug, profile, and partition their code very quickly.

Source code editor

The C-code editor allows you to efficiently create and modify your code using rich editing capabilities. Recognition of language features such as keywords, comments, declarations, and strings are eased through syntax highlighting. Symbol indexing allows fast program navigation including find declaration, find definition, and find type. Other features in the editor that speed up coding include code completion, auto indenting, and quick diff. Block comment/uncomment is useful when debugging or profiling large source files, as is text folding for hiding areas of text that you don't need to view. Other standard views, such as source outline, make target, and problems are also available.

Source: https://ip.cadence.com/uploads/103/SWdev-pdf, page 3.

Original

```
#include "example.h"
float floating(float *aData, float *aCoeff) {
  int i, j, ii;
  float pow, offset;
  float res, sum, mx, mn;
  float aOut[DATA_LENGTH];

  for(i=0; i<DATA_LENGTH; i++) {
    for(j=0, sum=0; j<COEFF_LENGTH; j++) {
      ii = max(min(i+j-1,DATA_LENGTH-1),0);

      res = aData[ii] * aCoeff[j];
      sum += res;
    }
    aOut[i] = sum;
  }

  // Get min, max, sum & power values
  mn=MAX_DATA, mx=MIN_DATA, pow=0; // Initialize
  for(i=0; i<DATA_LENGTH; i++) {
    mx = max(aOut[i],res);
    mn = min(aOut[i],res);
    sum += aOut[i];
    pow += (aOut[i] * aOut[i]);
  }

  // Do more processing...
  return pow;
}
```

Ported

```
#include "example.h"
xd_q8_31d overloadQ(xd_q15d *aData, xd_q15d *aCoeff) {
  int i, j, ii;
  xd_q8_31d pow, offset;
  xd_q15d res, sum, mx, mn;
  xd_q15d aOut[DATA_LENGTH];

  for(i=0; i<DATA_LENGTH; i++) {
    for(j=0, sum=0; j<COEFF_LENGTH; j++) {
      ii = max(min(i+j-1,DATA_LENGTH-1),0);

      res = aData[ii] * aCoeff[j];
      sum += res;
    }
    aOut[i] = sum;
  }

  // Get min, max, sum & power values
  mn=MAX_DATA, mx=MIN_DATA, pow=0; // Initialize
  for(i=0; i<DATA_LENGTH; i++) {
    mx = max(aOut[i],res);
    mn = min(aOut[i],res);
    sum += aOut[i];
    pow += (aOut[i] * aOut[i]);
  }

  // Do more processing...
  return pow;
}
```

Only the text in the red box needs to be changed to convert the source to fixed point

Figure 4. Operator overloading makes porting existing code easier

Source: https://ip.cadence.com/uploads/103/SWdev-pdf, page 3.

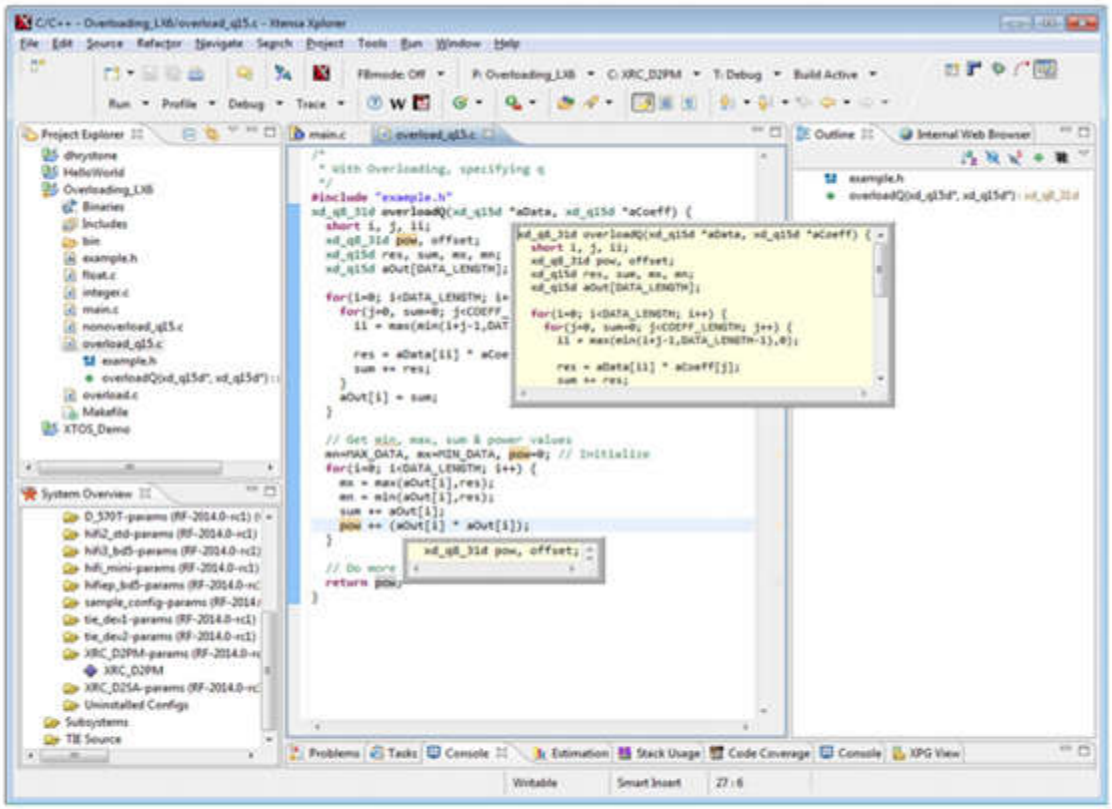


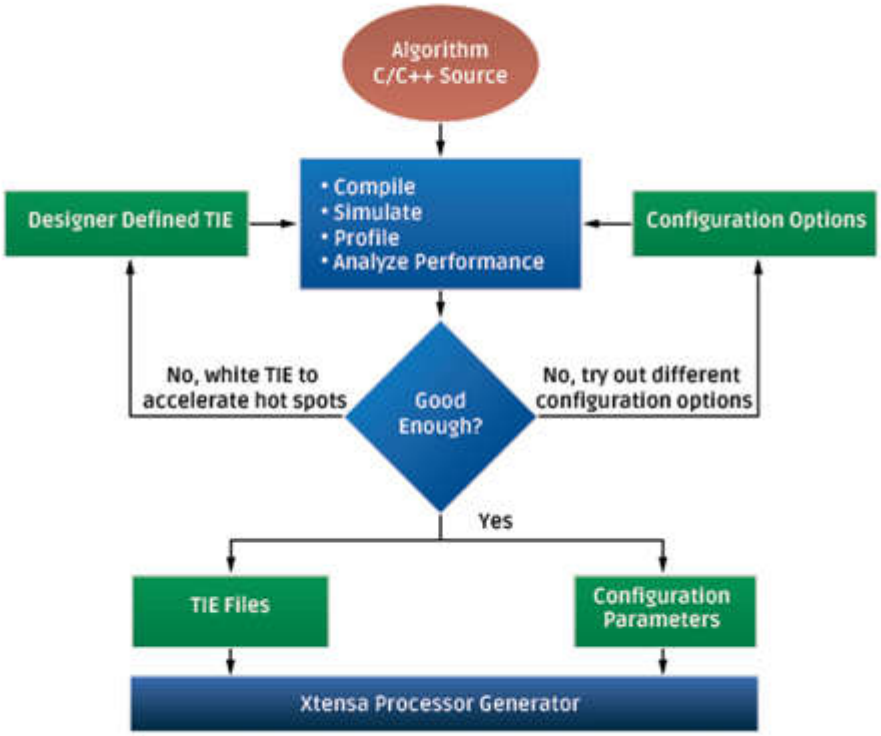
Figure 3. The editor includes many useful functions to speed up code generation and debugging

Source: https://ip.cadence.com/uploads/103/SWdev-pdf, page 3.

16. xTensa generates specific application handler code to associate the generic application functions with specific functions of a device driver for at least one of the types of the hardware modules. For example, in addition to the generic drivers and HAL, Tensilica Xtensa SDK also includes specific application code which is specific to particular Xtensa processor. Certain elements of this limitation are illustrated in the screenshots below and in the screenshots referenced in connection with other elements herein.

Learn More About This Powerful Toolset

The Hardware Design Flow



Source: https://ip.cadence.com/hwdes

Configure the Processor to Your Specifications

It's as easy as checking a box or using a drop-down menu to pick the options—among hundreds available—for your application. You start with a base Xtensa processor, then add just the features you need—and avoid the features you don't—so you get the lowest power and the most efficient processor for your requirements.

Basic configuration options include processor interfaces, byte ordering, interrupts, arithmetic options, floating point, bridges, memory subsystem options, and much more.

The Tensilica processor platform features additional DSP ISA options including our HiFi DSPs for Audio, ConnX DSPs for Baseband, Fusion DSPs for IoT and General Purpose DSP, and our Vision DSPs for CNN, Vision processing and Deep Learning.

Use TIE for Further Customizations

The Verilog-like TIE (Tensilica Instruction Extension) language offers a wide range of flexibility in adding multi-cycle pipelined execution units, register files, SIMD arithmetic and logic units, creating wide (up to 512-bit) load/store instructions, and adding adding designer-defined I/O Ports, Queues, and Lookups. Additionally, your Xtensa based processor can become a multi-issue VLIW processor. See our white papers for examples and ideas.

Source: https://ip.cadence.com/hwdes

Figure 1–3 illustrates the Xtensa design flow.

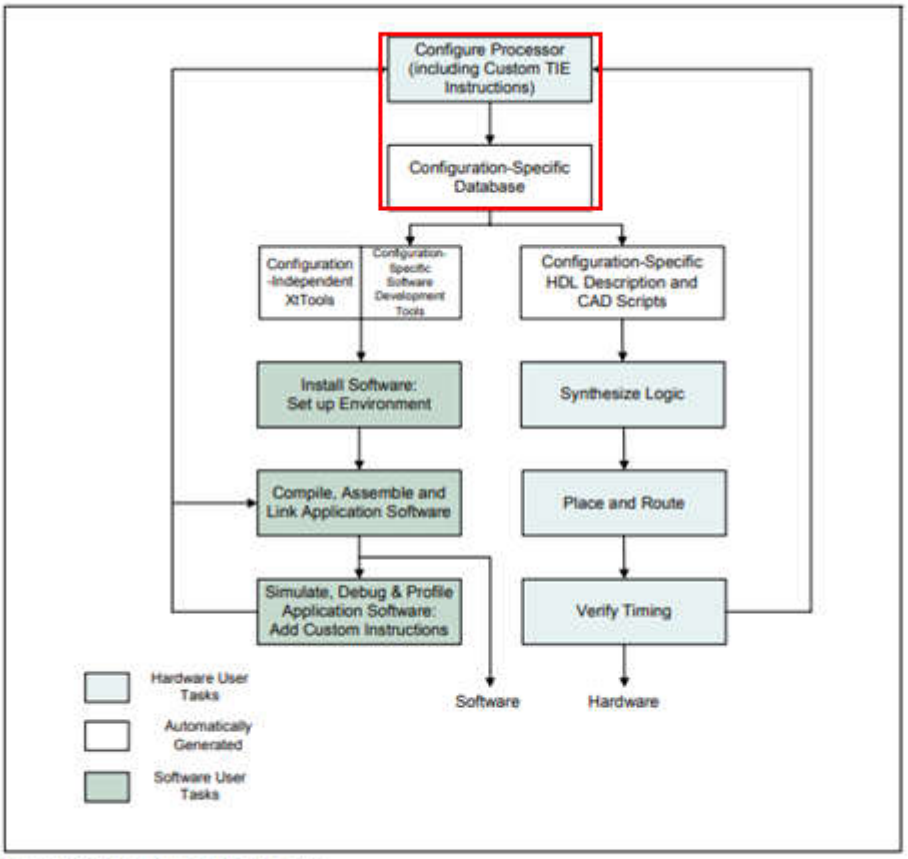


Figure 1–3. The Xtensa Design Flow

Source: https://0x04.net/~mwk/doc/xtensa.pdf, page 15.

17. xTensa generates specific application handler code and defines a specific element in the specific code to be handled by one of the generic application functions for that hardware module. For example, Tensilica Xtensa SDK generates system-specific application handler code by defining specific elements such as functions and data structures ("specific element") corresponding to specific hardware modules (such as Tensilica Xtensa Dataplane Processors (DPU) such as Xtensa LX series processors) that extend or otherwise connect the system-specific application handler code and data structures made available by the generic application handler code of the Tensilica Xtensa SDK. Certain elements of this limitation are illustrated in the screenshots below and in the screenshots referenced in connection with other

elements herein.

SoC Modeling

Many SoC designs today employ multiple processors. As SoC design becomes more complex, new methods to describe, debug, and profile overall system performance need to be employed. Unfortunately, most software development tools vendors do not provide pre-silicon simulation environments for multi-processor SoCs. Tensilica offers two modeling tools: XTensa Modeling Protocol (XTMP) for modeling in C and XTensa SystemC (XTSC) for modeling in SystemC.

Both tools are powerful additions to Tensilica's software development toolkit. They provide an Application Programming Interface (API) to the ISS, allowing fast and accurate simulation of SoC designs incorporating one or more processor cores. Running up to 10,000 times faster than RTL simulators, the XTMP/XTSC environments are potent tools for software development and SoC design. Both tools give you the ability to rapidly explore SoC partitioning alternatives and hardware/software performance tradeoffs. See Figure 11.

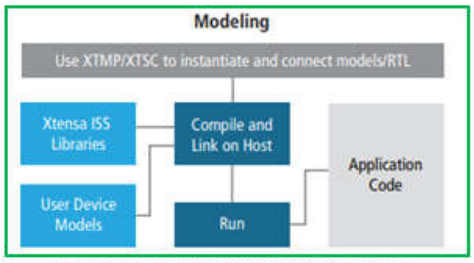


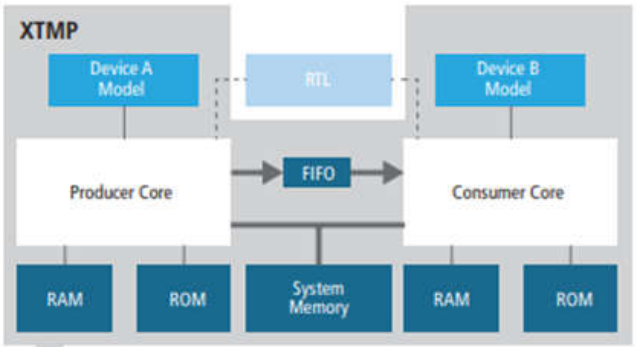
Figure 11. Using the ISS with XTMP or XTSC for modeling

XTMP and XTSC are used for simulating homogeneous or heterogeneous multi-processor design subsystems as well as complex uniprocessor architectures. Use the Xtensa Xplorer IDE's multi-processor project to instantiate multi-processor subsystems (or do it manually) and optionally connect them to custom peripherals and interconnects. You can create, debug, profile, and verify combined SoC and software architectures early in the design process. As the simulator operates at a higher level than HDL simulations, simulation time is cut drastically. See Figures 12 and 13.

XTMP and XTSC are integrated into the Xtensa Xplorer IDE, which automates the creation and development of multi-processor subsystem simulations. For XTMP, simulations are described in standard C code, which you can modify to allow more complex systems and additional simulator control if required. For XTSC, simulations are described in standard SystemC code. In addition, you have full visibility into all aspects of the simulation through the extensive API. Designers can use a single Xtensa Xplorer IDE to debug all simulated cores for additional visibility. The Xtensa Xplorer IDE manages all of these connections for you in its IDE for simplicity and easy viewing of any core.

Source: https://ip.cadence.com/uploads/103/SWdev-pdf, page 7.

Modeling with XTMP




```
XTMP_core consumer = XTMP_coreNew("consumer", config);
XTMP_core producer= XTMP_coreNew("producer", config);

XTMP_queue fifo = XTMP_queueNew("fifo", width, depth);
XTMP_connectQueue(fifo, producer, "FIFO_OUT", consumer, ("FIFO_IN");
```

Figure 12. XTMP provides a simulation environment using instantiations of multi-processor-capable ISS, memory models, and connectors

Modeling with XTSC

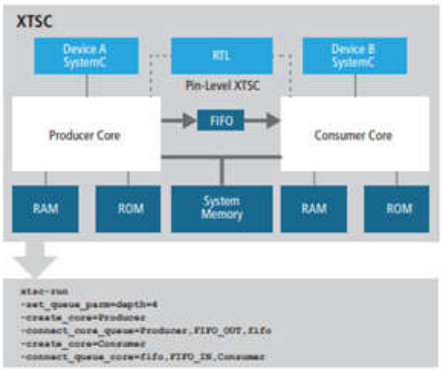


Figure 13. With its pin-level modeling capabilities, XTSC allows co-simulation with Verilog

Modeling of local and system memory

XTMP and XTSC allow memory modeling of both local and system memory. System memory can have programmable latencies specified for different transaction types, allowing an accurate system simulation for analyzing performance tradeoffs. Memory-mapped peripherals may be included in an XTMP/XTSC system simulation, and functions are provided to connect the processor to peripheral devices.

Source: https://ip.cadence.com/uploads/103/SWdev-pdf, page 7.

Benefits

- Easy-to-use Xtensa Xplorer IDE based on familiar Eclipse platform
- Small, high-performance code from 'C' source
  - Compiler offers state-of-the-art inter-procedural and alias analysis
  - Automatic vectorization of operations for Xtensa SIMD processors
  - Automatic Flexible Length Instruction eXtension (FLIX) instruction bundling for multi-issue Xtensa very long instruction word (VLIW) cores
- Detailed pipeline analysis guides optimizations from cycle/pipeline-accurate ISS
- Fast TurboXim simulation for up to 50 million instructions per second
- Vectorization Assistant guides code optimizations for better SIMD performance
- Easily and quickly evaluate multi-processor subsystems
- Familiar GNU-based toolchain

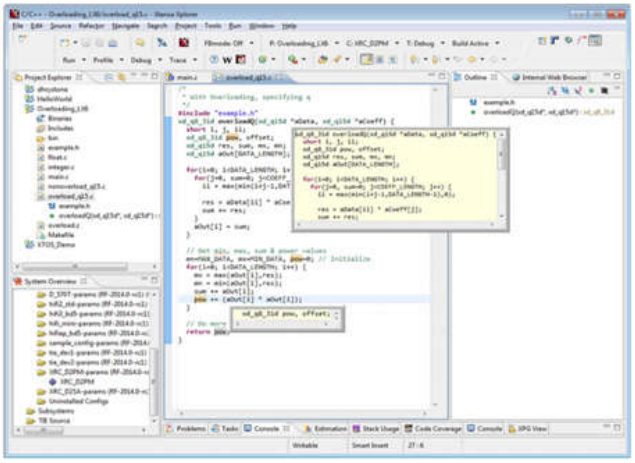
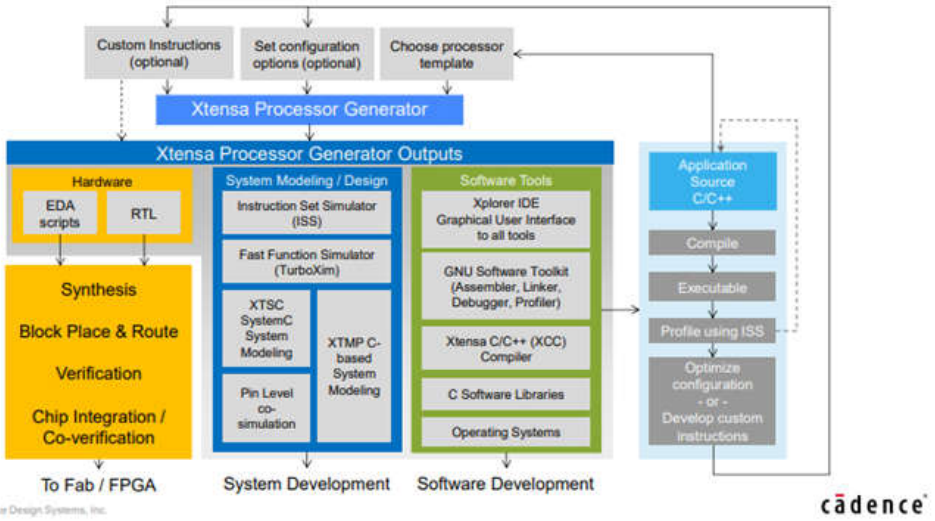


Figure 3. The editor includes many useful functions to speed up code generation and debugging

Source: https://ip.cadence.com/uploads/103/SWdev-pdf, page 1-2.

Xtensa Fully Automated Hardware and Software Tools Generation

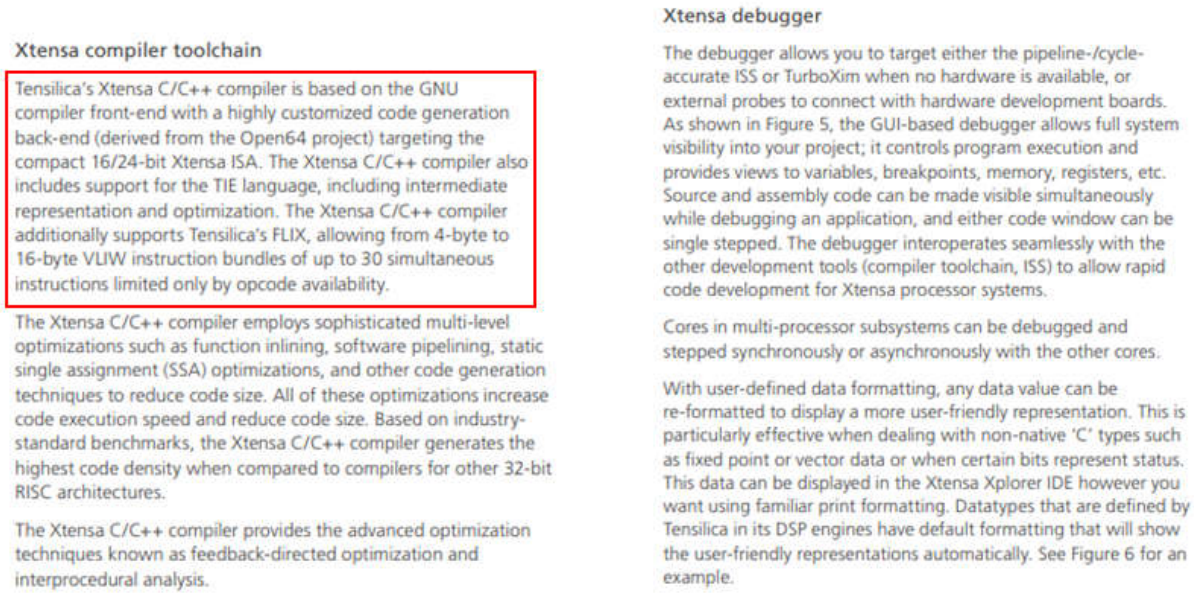


Source: https://nmi.org.uk/wp-content/uploads/2017/09/CADENCE_Practical-usage-of-Configurable-DSPs-in-High-Performance-Systems.pdf, page 22.

19

18. xTensa compiles the generic application handler programs together with the specific application handler code to produce machine-readable code to be executed by an embedded processor in the at least one of the types of the hardware modules. For example, when a specific application is needed for a particular hardware, the generic functions and the

specific functions are compiled together to yield a machine readable code. Tensilica Xtensa SDK and/or its customers compile the generic functions and the specific functions using Tensilica Xtensa SDK and/or any other compiling SDK supported by Cadence. Certain elements of this limitation are illustrated in the screenshots below and in the screenshots referenced in connection with other elements herein.



Xtensa compiler toolchain

Tensilica's Xtensa C/C++ compiler is based on the GNU compiler front-end with a highly customized code generation back-end (derived from the Open64 project) targeting the compact 16/24-bit Xtensa ISA. The Xtensa C/C++ compiler also includes support for the TIE language, including intermediate representation and optimization. The Xtensa C/C++ compiler additionally supports Tensilica's FLIX, allowing from 4-byte to 16-byte VLIW instruction bundles of up to 30 simultaneous instructions limited only by opcode availability.

The Xtensa C/C++ compiler employs sophisticated multi-level optimizations such as function inlining, software pipelining, static single assignment (SSA) optimizations, and other code generation techniques to reduce code size. All of these optimizations increase code execution speed and reduce code size. Based on industry-standard benchmarks, the Xtensa C/C++ compiler generates the highest code density when compared to compilers for other 32-bit RISC architectures.

The Xtensa C/C++ compiler provides the advanced optimization techniques known as feedback-directed optimization and interprocedural analysis.

Xtensa debugger

The debugger allows you to target either the pipeline-/cycle-accurate ISS or TurboXim when no hardware is available, or external probes to connect with hardware development boards. As shown in Figure 5, the GUI-based debugger allows full system visibility into your project; it controls program execution and provides views to variables, breakpoints, memory, registers, etc. Source and assembly code can be made visible simultaneously while debugging an application, and either code window can be single stepped. The debugger interoperates seamlessly with the other development tools (compiler toolchain, ISS) to allow rapid code development for Xtensa processor systems.

Cores in multi-processor subsystems can be debugged and stepped synchronously or asynchronously with the other cores.

With user-defined data formatting, any data value can be re-formatted to display a more user-friendly representation. This is particularly effective when dealing with non-native 'C' types such as fixed point or vector data or when certain bits represent status. This data can be displayed in the Xtensa Xplorer IDE however you want using familiar print formatting. Datatypes that are defined by Tensilica in its DSP engines have default formatting that will show the user-friendly representations automatically. See Figure 6 for an example.

Source: https://ip.cadence.com/uploads/103/SWdev-pdf, page 4.

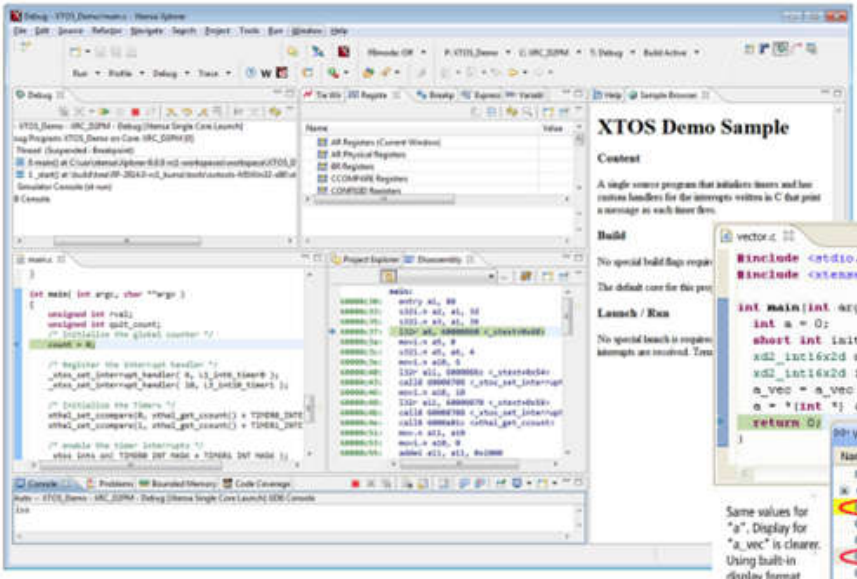


Figure 5. The Xtensa debugger allows full visibility into the system

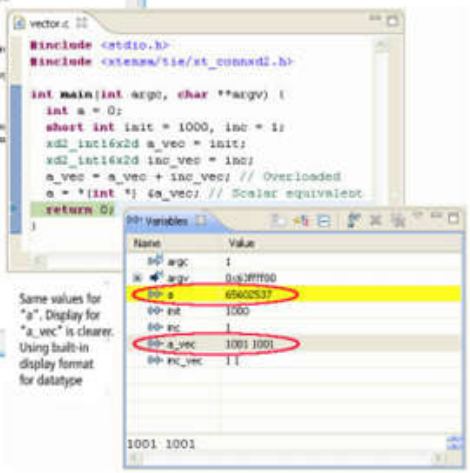
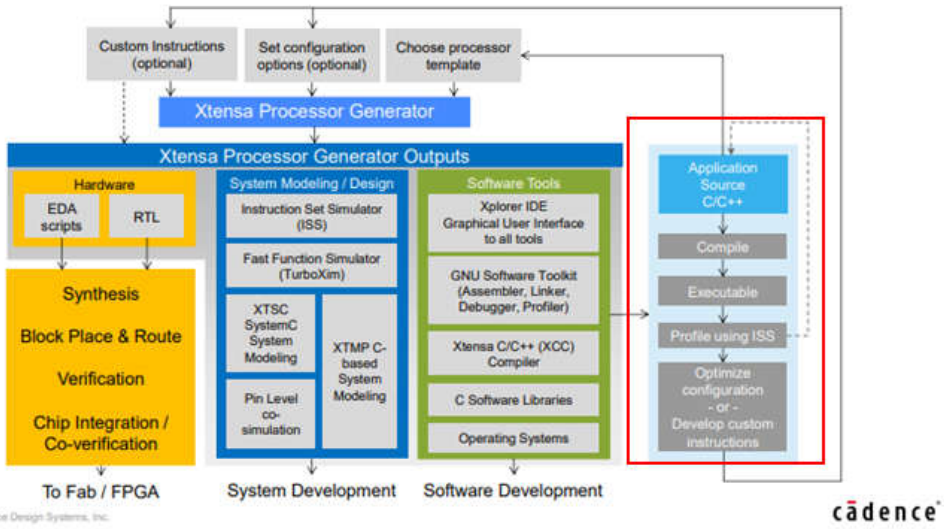


Figure 6. Data can be reformatted the way you want it

Source: https://ip.cadence.com/uploads/103/SWdev-pdf, page 4.

Xtensa Fully Automated Hardware and Software Tools Generation



Source: https://nmi.org.uk/wp-content/uploads/2017/09/CADENCE_Practical-usage-of-Configurable-DSPs-in-High-Performance-Systems.pdf, page 22.

The Highly Efficient XCC

The centerpiece of the Xtensa compiler tool chain is the Xtensa C/C++ Compiler (XCC). XCC employs sophisticated multi-level optimizations such as function inlining, software pipelining, static single assignment (SSA) optimizations, and other code generation techniques to reduce code size and increase execution speed. It also uses advanced optimization techniques and supports operating overloading on custom data types in the C programming language (without the overhead that is often associated with it).

Our XCC C/C++ compiler supports all modifications that hardware designers can make to a Tensilica Processor. If an application needs to work on 56-bit data, a designer can define a custom 56-bit data type with a single line of code. XCC makes porting and creating C application code much easier than other compilers.

Debugger

The Debugger lets you target either the pipeline/cycle accurate Instruction Set Simulator (ISS) or TurboXim when no hardware is available or external probes need to connect with hardware development boards. The GUI-based debugger allows full system visibility into your project. It controls program execution and provides views to variables, breakpoints, memory, registers, etc. It interoperates seamlessly with the other development tools (compiler toolchain, ISS) to allow rapid code development.

Cores in multi-processor subsystems can be debugged and stepped synchronously or asynchronously with the other cores.

Source: https://ip.cadence.com/swdev

19. Defendant's actions complained of herein will continue unless Defendant is enjoined by this court.

20. Defendant's actions complained of herein are causing irreparable harm and monetary damage to Plaintiff and will continue to do so unless and until Defendant is enjoined and restrained by this Court.

21. Plaintiff is in compliance with 35 U.S.C. § 287.

PRAYER FOR RELIEF

WHEREFORE, Plaintiff asks the Court to:

(a) Enter judgment for Plaintiff on this Complaint on all causes of action asserted herein;

(b) Enter an Order enjoining Defendant, its agents, officers, servants, employees, attorneys, and all persons in active concert or participation with Defendant who receive notice of the order from further infringement of United States Patent No. 7,069,546 (or, in the alternative, awarding Plaintiff a running royalty from the time of judgment going forward);

(c) Award Plaintiff damages resulting from Defendant's infringement in accordance with 35 U.S.C. § 284;

(d) Award Plaintiff pre-judgment and post-judgment interest and costs; and

(e) Award Plaintiff such further relief to which the Court finds Plaintiff entitled under law or equity.

Dated: January 31, 2019

Respectfully submitted,

/s/ Jay Johnson

JAY JOHNSON
State Bar No. 24067322
D. BRADLEY KIZZIA
State Bar No. 11547550
KIZZIA JOHNSON, PLLC
1910 Pacific Ave., Suite 13000
Dallas, Texas 75201
(214) 451-0164
Fax: (214) 451-0165
jay@kjpllc.com
bkizzia@kjpllc.com

ATTORNEYS FOR PLAINTIFF

EXHIBIT A