

Xtensa Software Developer's Toolkit

Quickly Develop Application Code

Product Brief

FEATURES

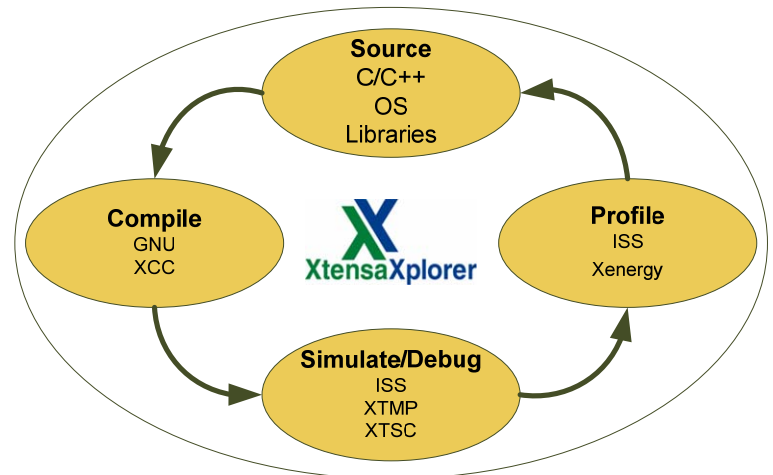
- Xtensa® Xplorer™ Integrated Development Environment (IDE) with full graphical user interface (GUI)
- Mature optimizing XCC C/C++ Compiler
- Operator overloading support in C for custom data types
- Pipeline-modeling, cycle-accurate Instruction Set Simulator (ISS) with fast TurboXim
- GNU profiler, linker, debugger, assembler and utilities
- Multiple processor subsystem simulation, debug, profiling and memory partitioning
- Vectorization Assistant for locating code loops that are not vectorizing, but could be with some changes
- Project management tools
- Performance and energy analysis tools
- Use Mentor Graphics Nucleus+, Express Logic's ThreadX, Micrium's uC/OS-II, Tata Elxsi's Ro-SES or the Linux operating systems

BENEFITS

- Easy to use 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 FLIX instruction bundling for multi-issue Xtensa VLIW cores
- Detailed pipeline analysis guides optimizations from cycle/pipeline-accurate instruction set simulator (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 multiple processor subsystems
- Familiar GNU-based toolchain

Shorter Development Times

Software developers can focus on the tasks that give the highest return. Profiling with pipeline-level feedback and the new Vectorization Assistant reduce the time required to locate areas in the application that can benefit the most from further effort.



Tensilica's Eclipse-based Xtensa Xplorer IDE serves as the cockpit for software development

Advanced Software Development Tools for Tensilica's Xtensa Processors

If you need to develop application code for an Xtensa processor, the Xtensa Software Developer's Toolkit provides a comprehensive collection of code generation and analysis tools that speed the development process. Tensilica's Eclipse-based Xtensa Xplorer Integrated Development Environment (IDE) serves as the cockpit for the entire development experience. If you are a custom Xtensa processor developer, you will use the Xtensa Processor Developer's Toolkit, described in a separate product brief, in addition to using the Xtensa Software Developer's Toolkit.

The Xtensa software development environment is automatically generated from the same database as the processor hardware description. All configuration options and designer-defined TIE (Tensilica Instruction Extensions) are supported. There is no need to manually edit or extend the tools to match these options. There is no chance that a software engineer implementing a new instruction in the compiler will implement the instruction differently than the hardware engineer who wrote the instruction because there is only one unified source description. This approach assures correctness and consistency by construction. See Figure 1.

Designers get a compiler, linker, assembler, and debugger for their particular processor hardware. As the base ISA is always present, third party tools can still be used even when the core is customized for a particular application.



A Comprehensive System

Now in its ninth generation, Tensilica's tools for software development are highly refined and provide developers with a complete, comprehensive solution for both system design and software development.

Project Manager

When you start a new software project or modify an ongoing project, the project manager organizes all related project source files and allows you to create new classes, files, or folders. New projects can be managed using the built-in project management and version control mechanisms, which eliminate the need to manually maintain makefiles and provides a clean environment for new project builds. The project manager allows you to set

all tool options and flags (build properties) for each build target within each individual project. Optionally, you can create un-managed projects that allow total user control over build target properties.

Multi-processor Subsystems

Xplorer 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.

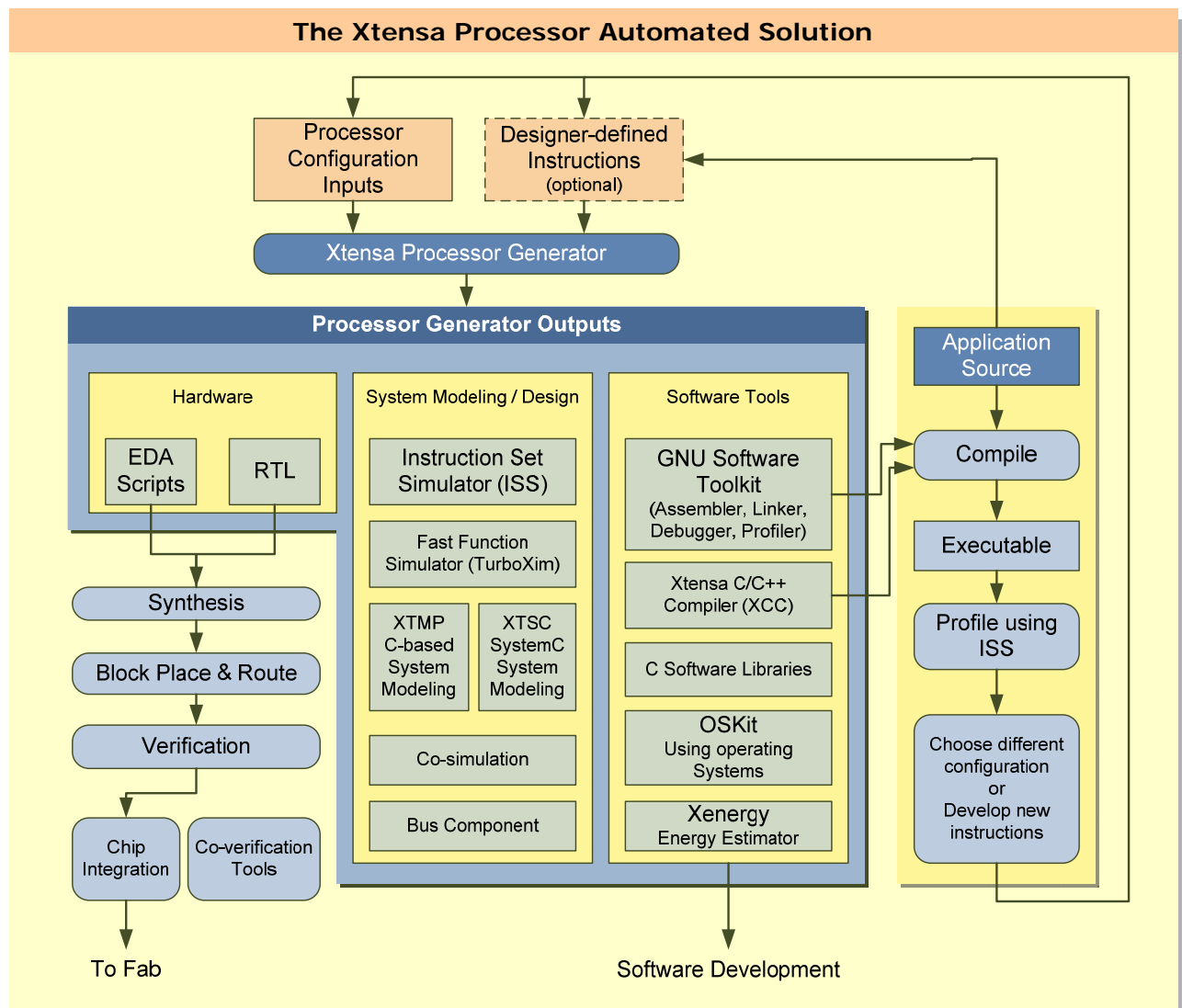


Figure 1. Tensilica's proven methodology automates the creation of customized processors and matching software tools



Source Code Editor

The 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 all still available.

Xtensa Compiler Toolchain

Tensilica's XCC C/C++ compiler is based on the GNU compiler front-end with a highly customized code generation back-end targeting the compact 16/24-bit Xtensa Instruction Set Architecture (ISA). The XCC compiler also includes support for the TIE language, including intermediate representation and optimization. The XCC compiler additionally supports Tensilica's Flexible Length Instruction eXtensions (FLIX), allowing 32-, 64-, or 128-bit VLIW instruction bundles of up to 30 simultaneous instructions limited only by opcode availability.

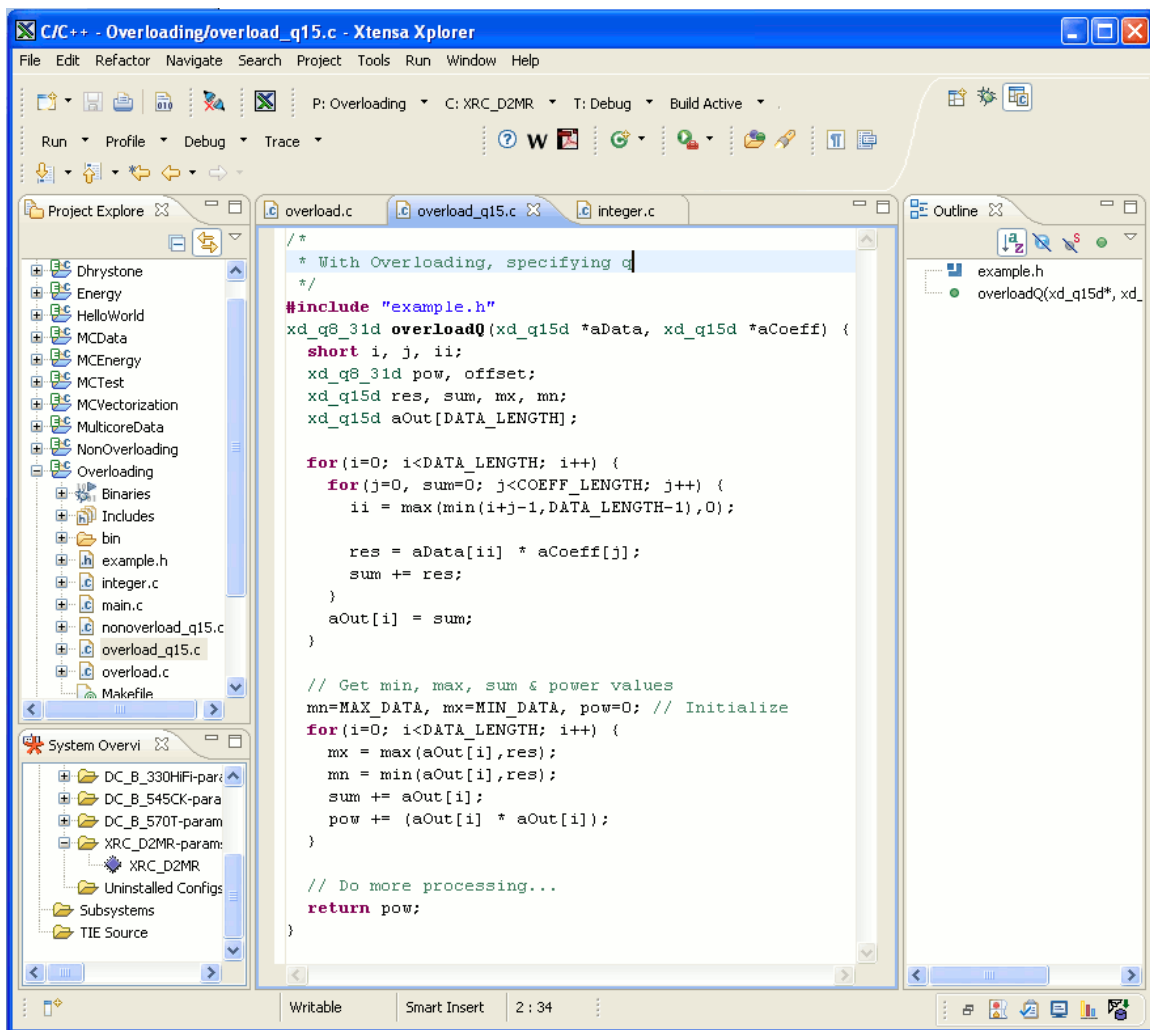


Figure 2. The editor includes many useful functions to speed up code generation and debugging



Xtensa Compiler Toolchain (continued)

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. All of these optimizations increase code execution speed and reduce code size. Based on industry standard benchmarks, the XCC compiler generates the highest code density when compared to compilers for other 32-bit RISC architectures.

XCC provides the advanced optimization techniques known as feedback-directed optimization and interprocedural analysis.

Feedback-directed optimization is a two-step process where code is instrumented on the first pass of compilation and run using a representative input data set to produce a file containing profiling information. On the second pass, this profiling information is used to optimize application code to further reduce branch delays, improve inlining, and minimize the impact of register spills. XCC will optimize an application's critical areas for performance while optimizing the remainder of the code for space. XCC is also capable of hardware feedback-directed optimization, in which the user's target hardware platform can run the instrumented code to similarly provide application-specific optimization. Hardware feedback-directed optimization is a much faster method and the optimization is performed on the actual target system as opposed to simulated in the ISS.

Interprocedural analysis is an optimization method that looks globally across all associated files of an application at link time. Global optimization is a much more powerful method than optimizing locally within an expression or procedure. Interprocedur-

al analysis examines relationships across function calls, and can perform optimizations that cannot be achieved with a local scope. Interprocedural analysis eliminates unneeded computations, improves function inlining, and performs alias analyses that may not be performed by less sophisticated optimization techniques.

XCC supports operator overloading on custom data types in the 'C' programming language (without the overhead that is often associated with it).

Tensilica is well known for its ability to let designers add custom instructions and data types to improve performance. 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. The designer can also specify what regular 'C' operators, such as '+' and '*', should do when using this data type. The overloading is always done with zero overhead so the resulting binaries are always efficient.

Porting and creating 'C' application code that uses custom data types is easier because standard 'C' operator syntax can be used. This makes the code easier to read and simpler to port via changes in the 'C' header files rather than throughout all of the source code itself. See Figure 3.

The rest of the software development toolchain is based on standard GNU tools. The compiler front-end remains similar to the preprocessor in the GNU tools — the flags for the preprocessor remain the same. The assembler and linker also utilize the same flags as the GNU versions of the tools.



```

#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;
}

```

Original

Ported

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

No Change

```

#include "example.h"
xd_q8_31d overflowQ(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;
}

```

Figure 3. Operator overloading makes porting existing code easier

Xtensa Debugger

The debugger allows you to target either the pipeline/cycle accurate Instruction Set Simulator (ISS) or TurboXim when no hardware is available or external probes 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. Source and assembly code can be made visible simultaneously while debugging an application; either code window can be single stepped. The debugger interoperates seamlessly with the other development tools (compiler toolchain, instruction set simulator) to allow rapid code development for Xtensa processor systems. See Figure 4.

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 reformatted 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 Xplorer 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 5 for an example.

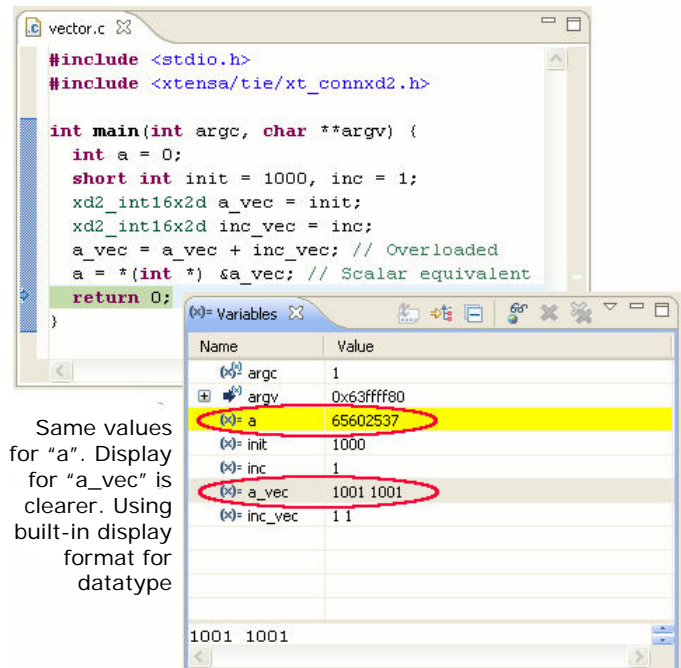


Figure 5. Data can be reformatted the way you want it

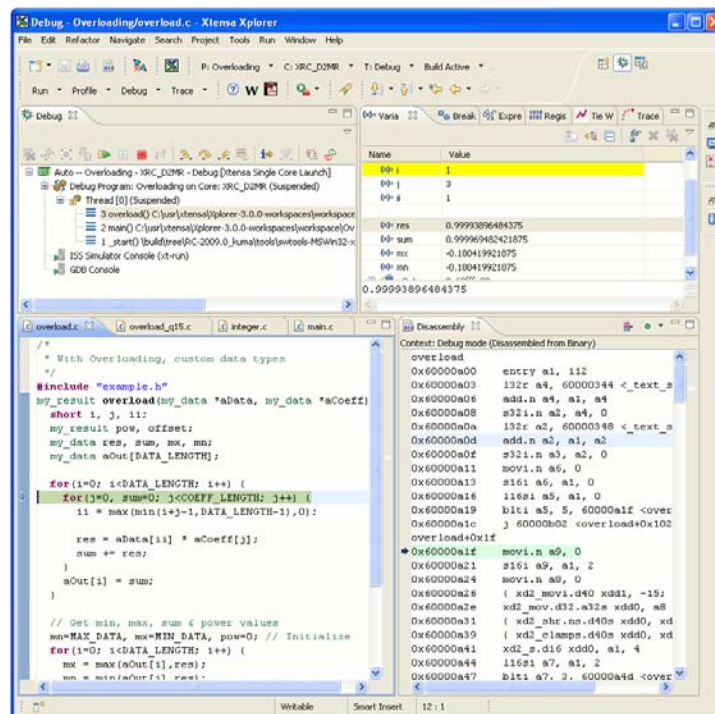


Figure 4. The Xtensa debugger allows full visibility into the system

Profiling Tools

Code profiling is an extremely important tool for optimizing the performance of your application code. The Xplorer IDE enables you to graphically view profiling results generated by Tensilica's pipeline-accurate ISS (see Figure 6). Additionally, for much faster and more accurate profiling, you can generate profiling data from hardware instantiated in an FPGA or ASIC. You can track performance data such instruction execution count, subroutine calls, subroutine total cycles, cache performance, etc.

While viewing functions in the profiling view, you can also simultaneously view the assembly code in the disassembly view and the source code in the editor. The call graph view enables you to view the entire application hierarchy's caller and callee functions.

For those inner loop optimizations, the pipeline view (see Figure 7) shows any pipeline inefficiencies and bubbles that may be occurring.

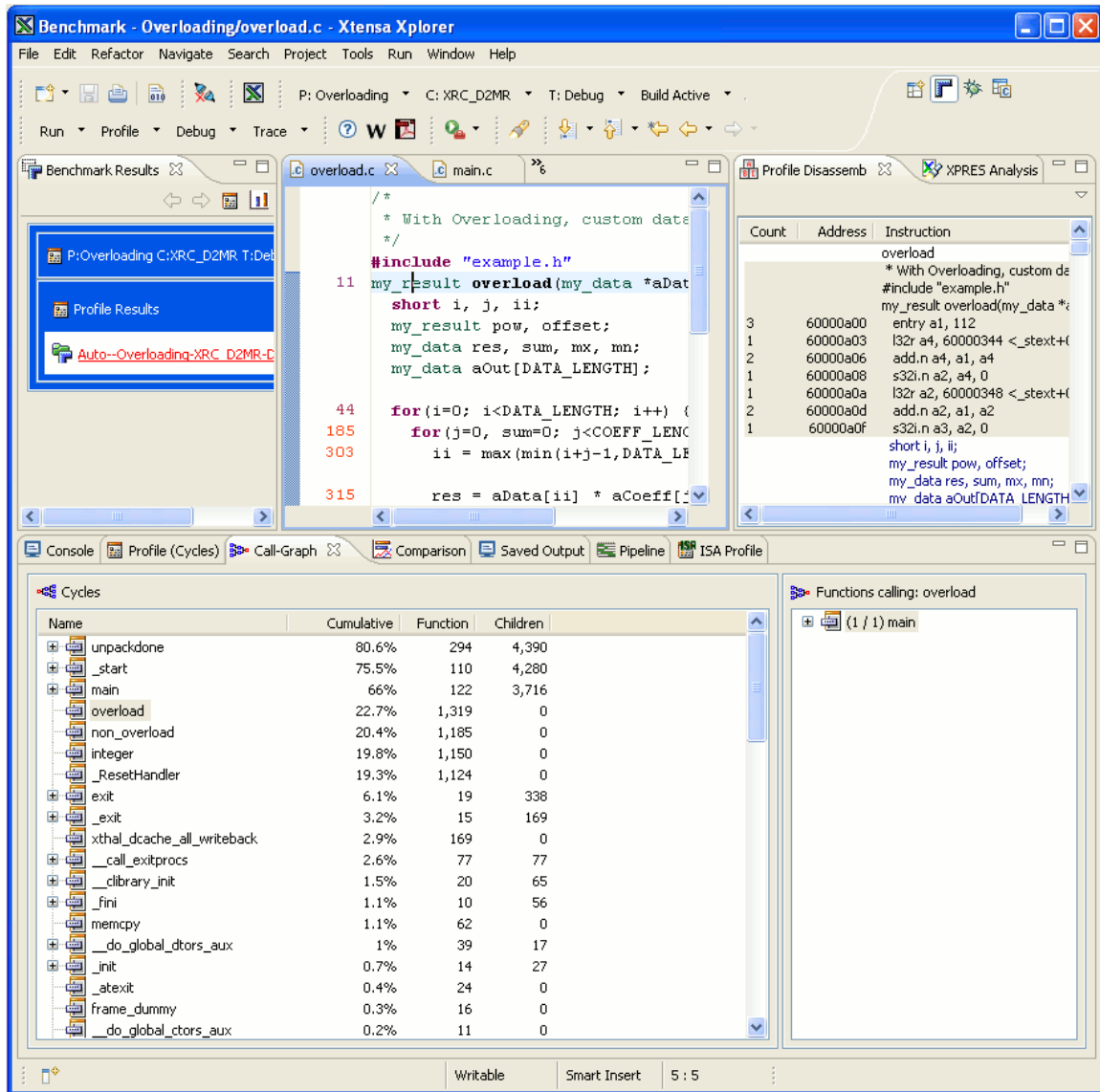


Figure 6. The profiling window allows performance metric analysis while optimizing code "hot spots"

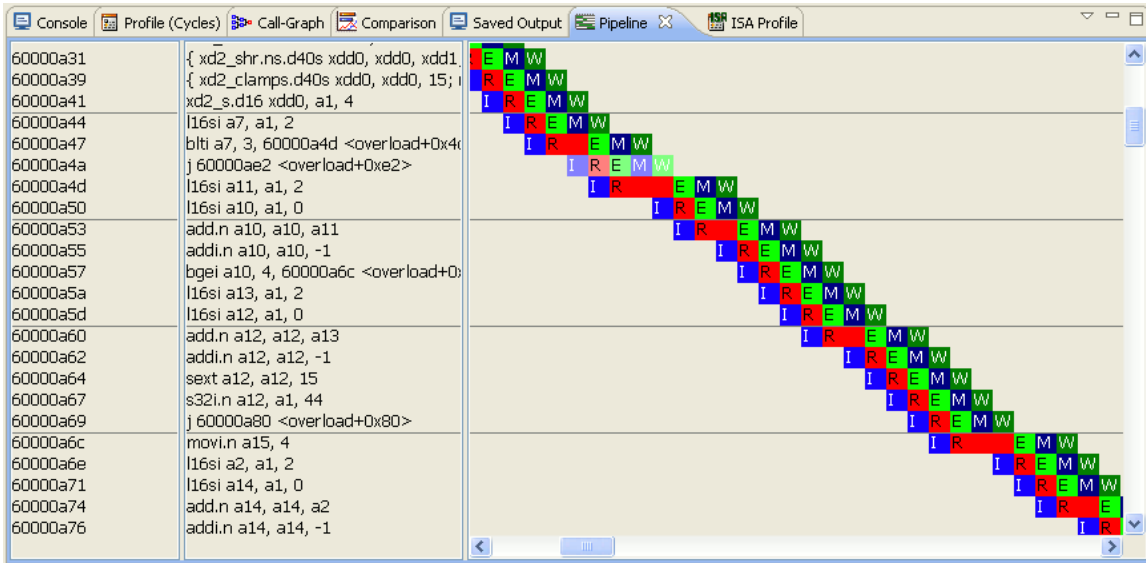


Figure 7. The pipeline viewer helps you understand instruction stalls and latency issues

Profiling of multi-processor subsystems shows each core side by side for easy load assessment and re-partitioning guidance. See Figure 8.

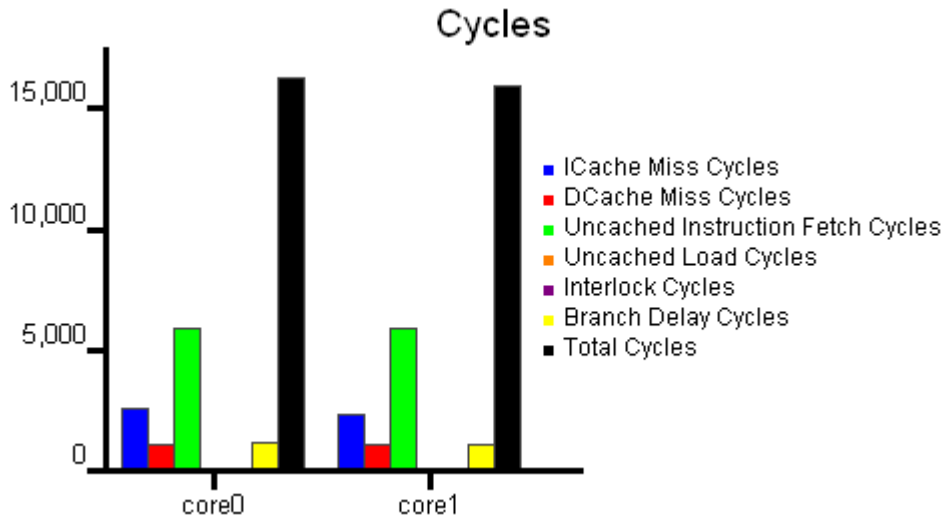


Figure 8. Multi-core profiling

Vectorization Assistant

Vectorization is the process of transforming the flow of your code (from the usual handle one data item at a time) into a parallel loop that operates on multiple data items at once. The Xtensa compiler is capable of performing this transformation automatically, but you can help it exploit implicit parallelism in your code by eliminating certain patterns of data access that prevent successful vectorization.

The Vectorization Assistant finds and displays loops in your code that could be "vectorized" by the compiler if the source was tweaked. Locating areas in the code that have not been vectorized, but could be, can take a long time looking at profiles, assembler and pipeline views – then you

have the task of doing the optimization to make it vectorize. In a few clicks, the Vectorization Assistant gets you to the loops in your source code that would benefit the most from vectorization.

The list of messages shown is initially sorted by the number of processor cycles used by a given loop, such that the most expensive loops appear first. You can focus the view on a particular file, folder, or project; you can filter out certain classes of messages that are not currently interesting; and you can hide messages that you do not wish to address at the moment.

Figure 9 shows an annotated screenshot of the Vectorization Assistant.

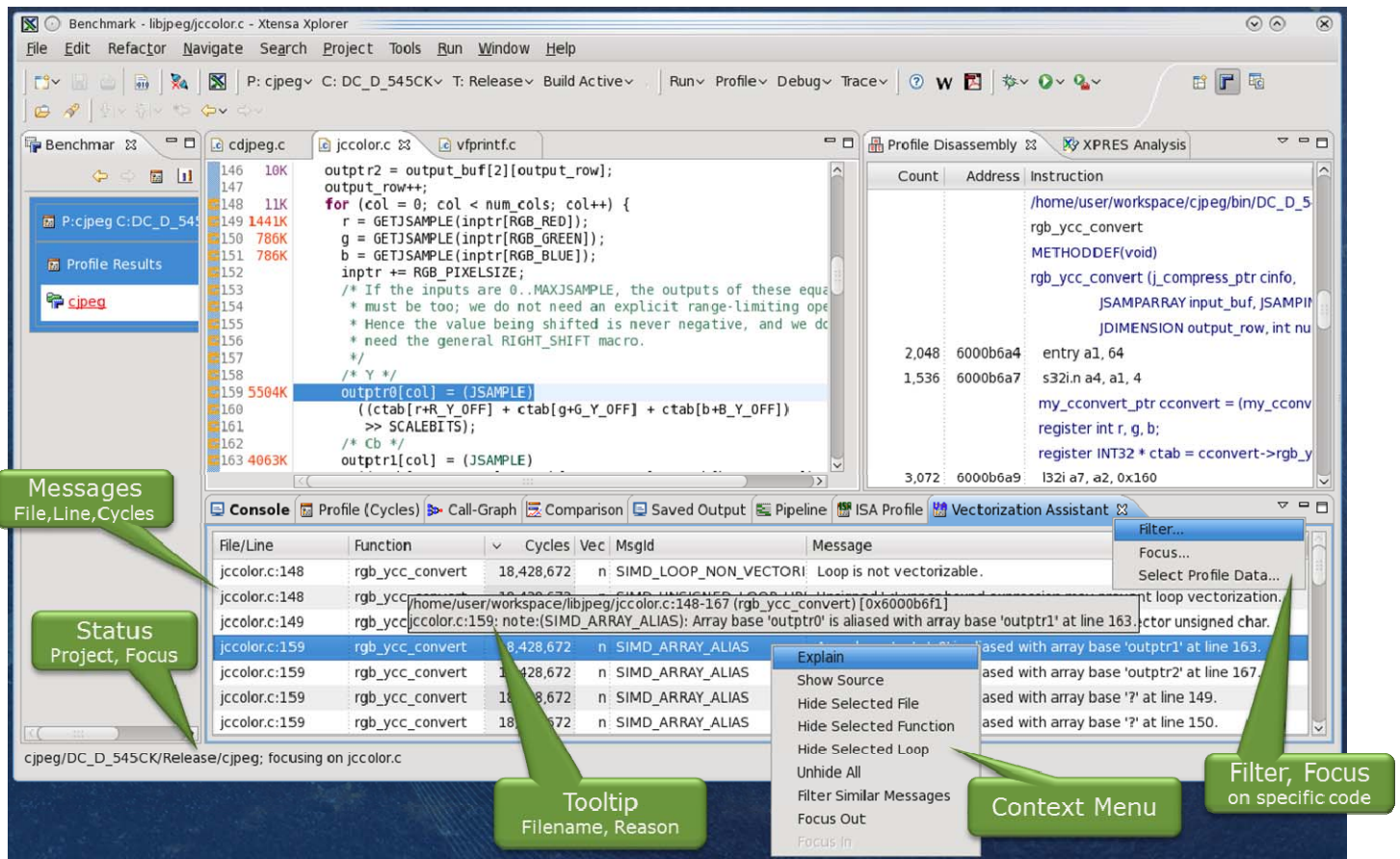


Figure 9. Vectorization Assistant helps find areas that can be improved

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 multiple processor SOCs. Tensilica offers two modeling tools: XTMP (XTensa Modeling Protocol) for modeling in C and XTSC (XTensa SystemC) 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 HW/SW performance tradeoffs. See Figure 10.

XTMP and XTSC are used for simulating homogeneous or heterogeneous multiple processor design subsystems as well as complex uni-processor architectures. Use Xplorer's multi-processor project to instantiate multiple 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 11 and 12.

XTMP and XTSC are integrated into the Xtensa Xplorer IDE, which automates the creation and development of multiple 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 Xplorer to debug all simulated cores for additional visibility.

Xplorer manages all of these connections for you in its IDE for simplicity and easy viewing of any core.

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.

A Multi-threaded Environment

An XTMP or XTSC simulation runs in a multi-threaded environment, with each processor running in its own thread. Core threads can be run asynchronously or synchronized through events using the attached debugger. Another option is to run all cores in lock-step, cycle-by-cycle mode. If one core stops on a break, all cores stop until it resumes. XTMP and XTSC have many of options for implementing, controlling and displaying results of system simulations deploying multiple cores, memories, and user-defined devices.

Pin-Level SystemC CoSimulation with Verilog

Additionally, Tensilica provides a link between its pipeline-accurate, cycle-accurate ISS and the leading Verilog simulators. Designers can now run pin-level SystemC co-simulations of Tensilica DPUs in their native Verilog simulators with pin-level XTSC. See Figure 11.

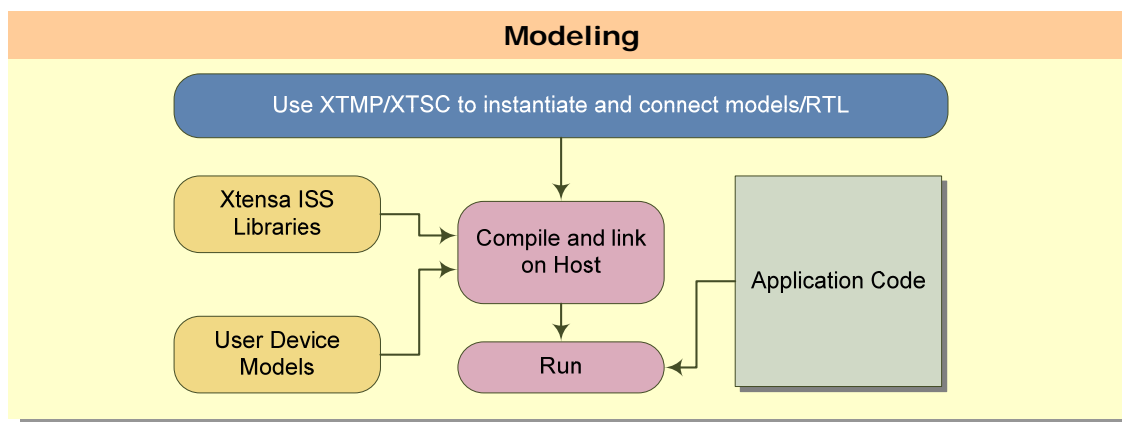


Figure 10. Using the ISS with XTMP or XTSC for modeling

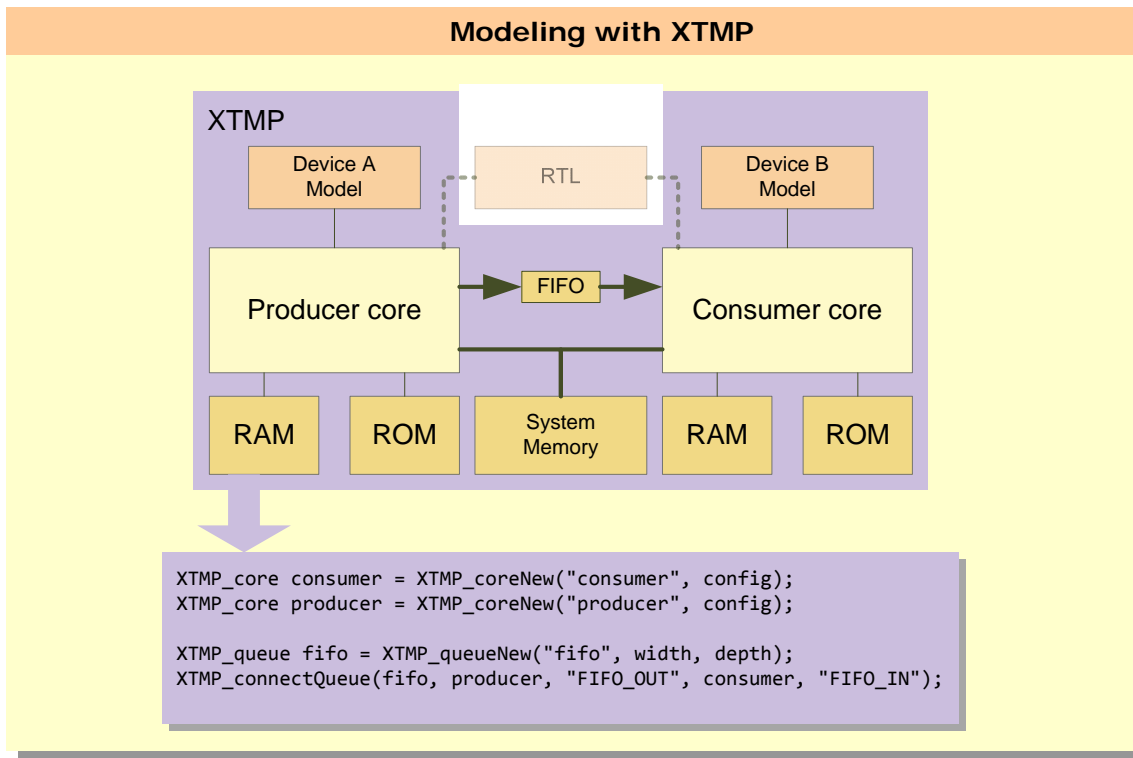
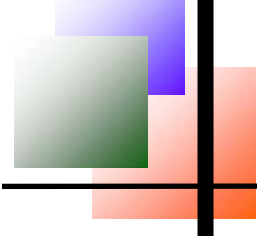


Figure 11. XTMP provides a simulation environment using instantiations of multiple processor capable ISS, memory models and connectors

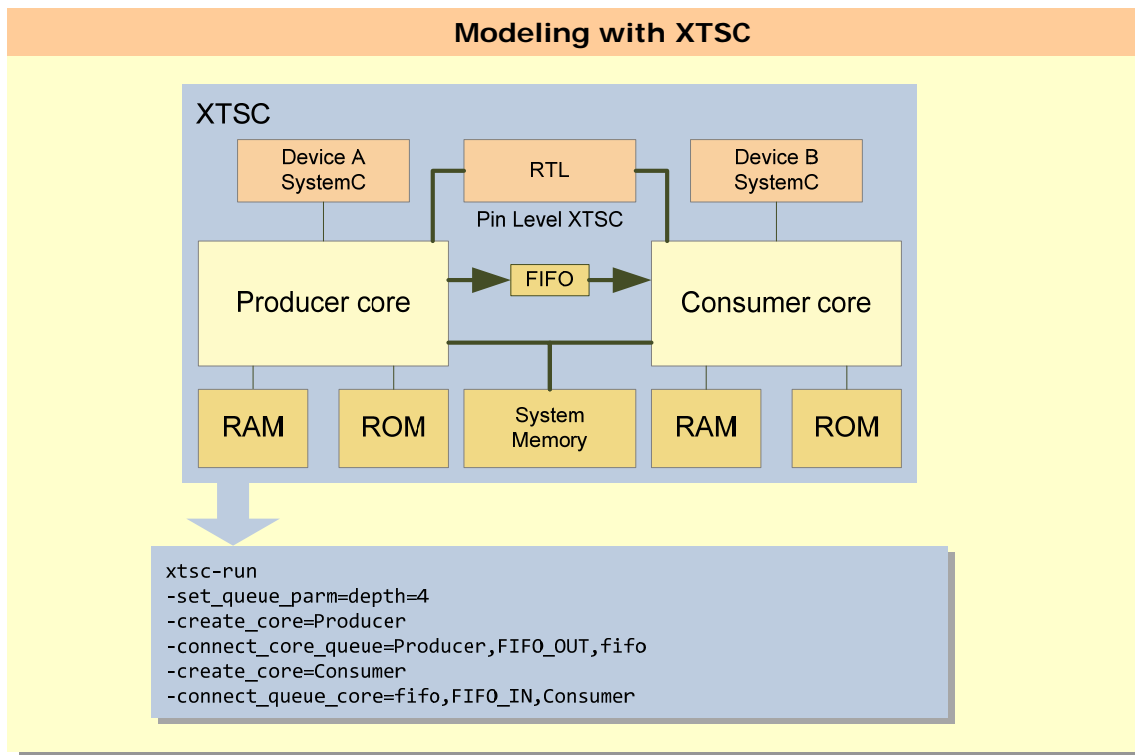


Figure 12. With its pin-level modeling capabilities, XTSC allows co-simulation with Verilog





Instruction Set Simulator Performance Comparison with Other Tools

Simulation Speed	Modeling Tool	Benefits
20 to 50 MIPS ¹	Stand-alone ISS in TurboXim mode	Fast functional simulation for rapid application testing
1.5 to 40 MIPS ^{1,2}	XTMP or XTSC in TurboXim mode	Fast functional simulation for rapid application testing at the system level
800K to 1,600K cycles/second	Stand alone ISS in cycle-accurate mode	Software verification and cycle accuracy
600K to 1,000K cycles/second ²	XTMP in cycle-accurate mode	Multi-core subsystem modeling and cycle accuracy
350K to 600K cycles/second ²	XTSC in cycle-accurate mode	Multi-core subsystem modeling with SystemC interfaces and cycle accuracy
1K to 4K cycles/second	Verilog RTL simulation	Functional verification, pipeline-cycle accuracy and high visibility and accuracy
10 to 100 cycles/second	Verilog gate-level simulation	Timing verification, pipeline-cycle accuracy and high visibility and accuracy.

1. TurboXim mode simulation speed is an estimate for relatively long-running application programs (1 billion instructions or more)
2. Simulation speed is an estimate for a single Xtensa core in XTMP or XTSC

Summary

The Xtensa Xplorer Integrated Development Environment is a complete GUI-based collection of tools that allows the software developer to create code for systems based on Xtensa processors. From project implementation to code generation to analysis, the Xtensa SDK enables you to achieve fast time-to-market while employing one of the most efficient 32-bit architectures available today. Xtensa processors lower total system costs and help design teams construct extremely high-performance system architectures.

Tensilica, Inc.

3255-6 Scott Blvd., Santa Clara, CA 95054-3013, USA
Tel: 1-408-986-8000 Fax: 408-986-8919 Website: www.tensilica.com

©March 2011 Tensilica and Xtensa are registered trademarks of Tensilica, Inc.
The Tensilica logo, Xenergy, Xplorer and TurboXim are trademarks of Tensilica, Inc. All other trademarks are the property of their respective owners.

