Tensilica Datasheet

Xtensa Processor Developer’s Toolkit

Rapidly customize standard processors for differentiation

Features

- Comprehensive, integrated Cadence® Tensilica® Xtensa® dataplane processor design and optimization environment
- Xtensa Xplorer™ Integrated Development Environment (IDE) (Figure 1) with full graphical user interface (GUI)
- Click-box and drop-down configuration options
- Simple Verilog-like Tensilica Instruction Extension (TIE) language for processor customization
- Multiple-processor design support
- Fully integrated with Xtensa Software Developers Toolkit (see separate product brief at ip.cadence.com/ipportfolio/tensilica-ip/xtensa-customizable) for software compilation, profiling, simulation, debugging, etc.
- TIE Port/Queue Wizard makes creating interfaces to other systems’ RTL simple

Automated Processor Development

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.

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.

The Processor Developer’s Toolkit is required for any design team that is using Tensilica’s TIE instructions to modify the processor. If you are using an Xtensa processor with no modification or only changes to configuration options, you do not need the Processor Developer’s Toolkit—you’ll only need the Software Developer’s Toolkit.

Benefits

- Easy-to-use Xtensa Xplorer IDE based on popular Eclipse platform
- Quickly optimize the processor to efficiently implement your algorithms
- Differentiate with unique processor optimizations
- Multiple processor configurations can be saved, profiled, and compared to narrow down the best solution
- Easily create multiple-core subsystems in minutes for evaluation and load profiling in simulation
- Automatically generate customized processor RTL with full synthesis scripts in about an hour
- RTL is pre-verified and guaranteed correct by construction
- All software-development tools and simulation models are generated automatically along with the processor

A Comprehensive System

Now in their 10th generation, Tensilica’s tools are highly refined and provide developers with a complete, comprehensive solution for both system design and software development. Tensilica’s Processor Developer’s Toolkit contains all the tools necessary to create highly customized Xtensa processors.

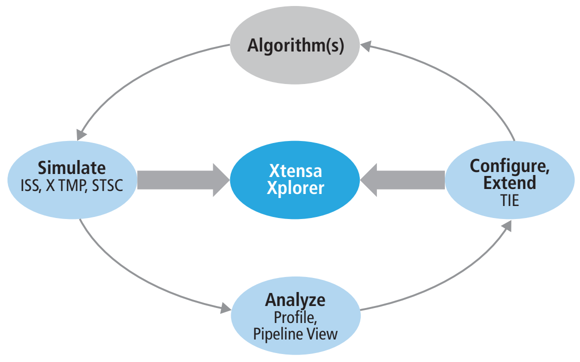


Figure 1: Tensilica’s Eclipse-based Xtensa Xplorer IDE serves as the cockpit for custom processor development.

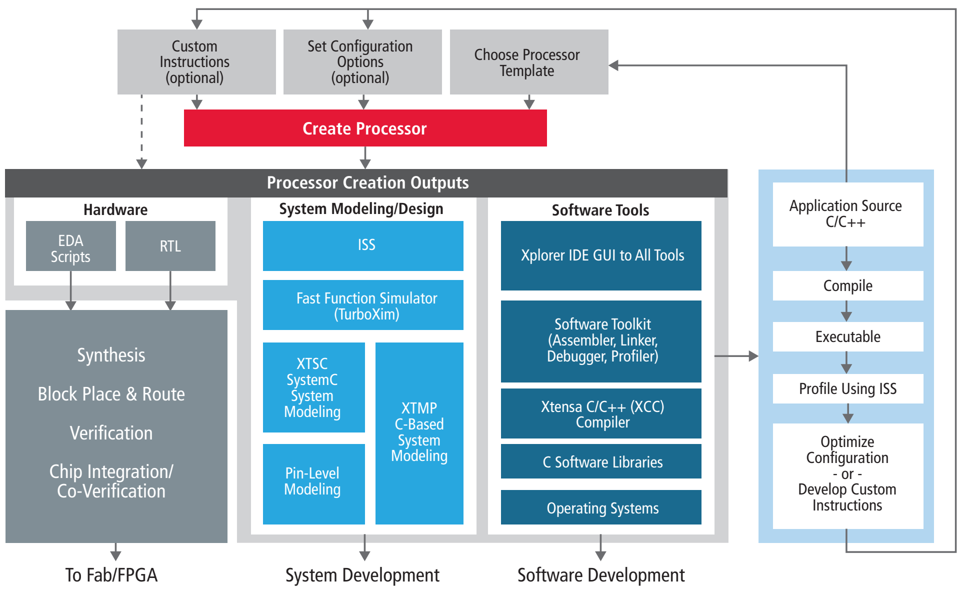

Custom Instructions (optional)
Set Configuration Options (optional)
Choose Processor Template
Create Processor
Processor Creation Outputs
Hardware
EDA Scripts
RTL
Synthesis
Block Place & Route
Verification
Chip Integration/ Co-Verification
System Modeling/Design
ISS
Fast Function Simulator (TurboXim)
XTSC SystemC System Modeling
XTMP C-Based System Modeling
Pin-Level Modeling
Software Tools
Xplorer IDE GUI to All Tools
Software Toolkit (Assembler, Linker, Debugger, Profiler)
Xtensa C/C++ (XCC) Compiler
C Software Libraries
Operating Systems
Application Source C/C++
Compile
Executable
Profile Using ISS
Optimize Configuration - or - Develop Custom Instructions
To Fab/FPGA
System Development
Software Development

Figure 2: Tensilica's proven methodology automates the creation of customized processors and matching software tools.

Everything You Need to Build High-Performance Xtensa DPUs

Tensilica's Processor Developer's Toolkit contains all the tools necessary to create and analyze extremely high-performance application-specific Xtensa DPUs. Tensilica's Xtensa Xplorer IDE serves as the cockpit for the entire design experience.

From Xtensa Xplorer, you can profile your application code, identify "hot spots" that can benefit from acceleration, and add the TIE instructions necessary to speed up that code. Using a check-box menu within the GUI, you can configure processors to include features you need and remove features you don't. Options for processor interface, memories, operating system support, EDA scripts, debug and trace, and much more are supported.

Tensilica's TIE language is similar to Verilog and is specifically designed for developing custom instructions and datapath elements for Xtensa processors. By defining their own Xtensa processor, Tensilica's customers often get 10 to 100 times better performance and lower energy consumption when compared to alternative processor architectures. This allows Xtensa processors to be used in critical dataplane SoC functions where standard microprocessors or digital signal processors cannot meet the needed performance, throughput, or energy budget. Until now, the only alternative was to use hand-coded RTL hardware blocks.

When changes are made to a processor configuration, the entire Xtensa toolchain is updated automatically within minutes, including the compiler, instruction set simulator (ISS), and SystemC models. You can then recompile your program and run it on the ISS to see the impact of your changes. Xplorer allows you to profile, compare, and save many different processor configurations, so you can narrow down the right one for your application. Also, you can model and simulate multiple-processor subsystems in this environment using Tensilica's Xtensa Modeling Protocol (XTMP) or Xtensa SystemC (XTSC)—see the Xtensa Software Developer's Toolkit product brief.

For integration into the dataplane with custom logic, XTSC can be used for simulation of DPUs with SystemC Transaction-Level Modeling (TLM) hardware. Co-simulation with RTL-level blocks and the Xtensa ISS is possible using pin-level XTSC, which offers pin-level, cycle-accurate modeling of Xtensa DPU interfaces for use in Verilog simulations.

Xtensa Xplorer serves as the gateway to the Xtensa Processor Generator (shown in Figure 2). Once a processor configuration is finalized, the Xtensa Processor Generator creates the automatically verified Xtensa processor to match all of the configuration

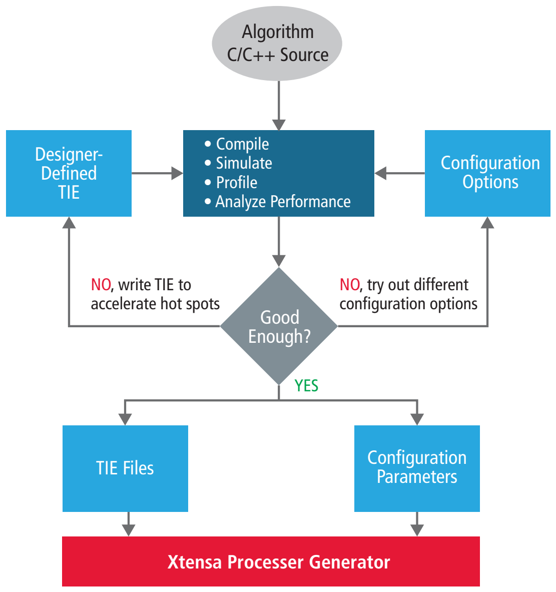
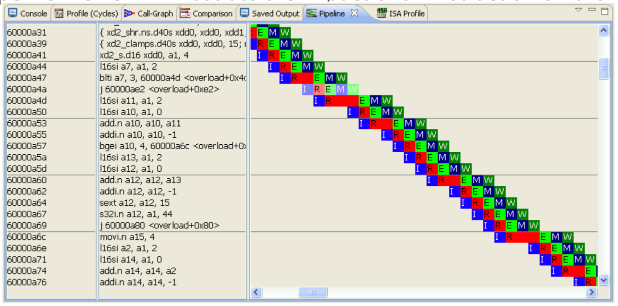


Figure 3: Design flow in Xtensa Xplorer

options and extensions you have defined, in about an hour. The full software tool chain is also created, matching all processor modifications you have made.

Pipeline Viewer

As shown in Figure 4, Xtensa Xplorer includes a sophisticated Pipeline Viewer that helps you visualize the performance impact of designer-defined instructions, without the need to become a processor pipeline expert. In one quick step, Xtensa Xplorer’s Pipeline Viewer will illustrate the impact of an instruction on the



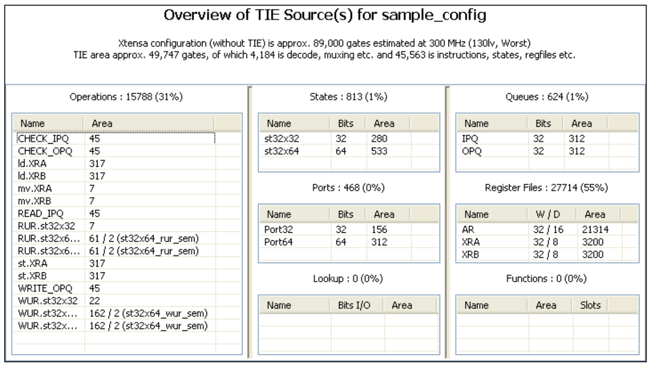
```
60000a31  { xd2_shr.ns.d40s xdd0, xdd0, xdd1
60000a39  { xd2_clamps.d40s xdd0, xdd0, 15; i
60000a41  xd2_s.d16 xdd0, a1, 4
60000a44  l16si a7, a1, 2
60000a47  blti a7, 3, 60000a4d <overload+0x4c
60000a4a  j 60000ae2 <overload+0xe2>
60000a4d  l16si a11, a1, 2
60000a50  l16si a10, a1, 0
60000a53  add.n a10, a10, a11
60000a55  addi.n a10, a10, -1
60000a57  bgei a10, 4, 60000a6c <overload+0x
60000a5a  l16si a13, a1, 2
60000a5d  l16si a12, a1, 0
60000a60  add.n a12, a12, a13
60000a62  addi.n a12, a12, -1
60000a64  sext a12, a12, 15
60000a67  s32i.n a12, a1, 44
60000a69  j 60000a80 <overload+0x80>
60000a6c  movi.n a15, 4
60000a6e  l16si a2, a1, 2
60000a71  l16si a14, a1, 0
60000a74  add.n a14, a14, a2
60000a76  addi.n a14, a14, -1
```

Figure 4: Pipeline Viewer shows instruction flow of disassembled code.

execution pipeline, providing instant feedback on the efficiency of a proposed new instruction and helping you tune the source TIE code for optimal implementation.

Analyze and Iterate

Xtensa Xplorer not only provides the tools to quickly develop



Overview of TIE Source(s) for sample_config

Xtensa configuration (without TIE) is approx. 89,000 gates estimated at 300 MHz (130lv, Worst)
TIE area approx. 49,747 gates, of which 4,184 is decode, muxing etc. and 45,563 is instructions, states, regfiles etc.

Operations : 15788 (31%)

Name	Area
CHECK_IPQ	45
CHECK_OPQ	45
ld.XRA	317
ld.XRB	317
mv.XRA	7
mv.XRB	7
READ_IPQ	45
RUR.st32x32	7
RUR.st32x6…	61 / 2 (st32x64_rur_sem)
RUR.st32x6…	61 / 2 (st32x64_rur_sem)
st.XRA	317
st.XRB	317
WRITE_OPQ	45
WUR.st32x32	22
WUR.st32x…	162 / 2 (st32x64_wur_sem)
WUR.st32x…	162 / 2 (st32x64_wur_sem)

States : 813 (1%)

Name	Bits	Area
st32x32	32	280
st32x64	64	533

Ports : 468 (0%)

Name	Bits	Area
Port32	32	156
Port64	64	312

Lookup : 0 (0%)

Name	Bits I/O	Area

Queues : 624 (1%)

Name	Bits	Area
IPQ	32	312
OPQ	32	312

Register Files : 27714 (55%)

Name	W / D	Area
AR	32 / 16	21314
XRA	32 / 8	3200
XRB	32 / 8	3200

Functions : 0 (0%)

Name	Area	Slots

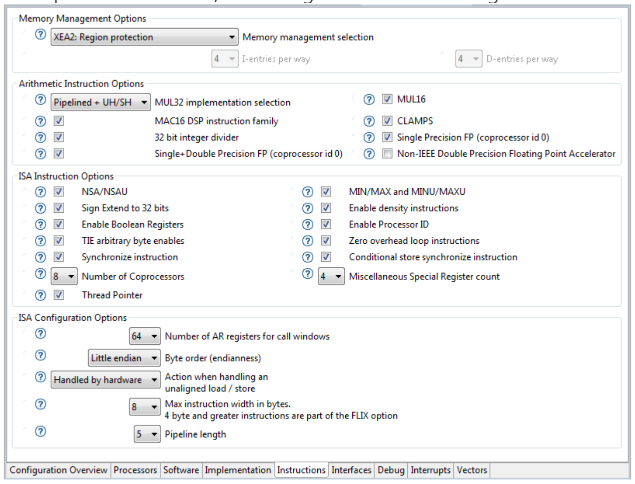
Figure 5: The TIE analyzer estimates gate count and silicon area for new instructions.

optimized processors and simulate systems of processors, but also provides the tools that enable you to visualize and analyze simulation results, tune system or processor configurations, and rapidly compare alternative implementations. See Figure 5.

Configuration Options

You don’t need to be locked into a set of features that were predetermined years or decades earlier by a processor designer. You can configure your Xtensa processor to exactly match your application. Not sure exactly what options you need? Our Xtensa Xplorer IDE will help you evaluate tradeoffs. The Pipeline Viewer is just one of the many tools in the Xplorer environment that can help you make intelligent decisions about pipeline depth, register set size, specialized ALUs, MMUs, local and cache memories, interface options, debug and trace options, interrupt levels and types, and many other configuration options. (Figure 6 shows just one of the configuration screens.)

All the configuration options are pre-verified to work together, and configuring a Tensilica processor core never compromises the underlying base Xtensa instruction set, thereby ensuring availability of a robust ecosystem of third-party application software and development tools. All configurable, extensible Xtensa processors are always compatible with major operating systems, debug probes, and ICE solutions. In addition, Xtensa processors always come with an automatically generated, complete software-development toolchain, including an advanced integrated



Memory Management Options

- XEA2: Region protection — Memory management selection
- 4 — I-entries per way
- 4 — D-entries per way

Arithmetic Instruction Options

- Pipelined + UH/SH — MUL32 implementation selection
- ✓ MAC16 DSP instruction family
- ✓ 32 bit integer divider
- ✓ Single+Double Precision FP (coprocessor id 0)
- ✓ MUL16
- ✓ CLAMPS
- ✓ Single Precision FP (coprocessor id 0)
- ☐ Non-IEEE Double Precision Floating Point Accelerator

ISA Instruction Options

- ✓ NSA/NSAU
- ✓ Sign Extend to 32 bits
- ✓ Enable Boolean Registers
- ✓ TIE arbitrary byte enables
- ✓ Synchronize instruction
- 8 — Number of Coprocessors
- ✓ Thread Pointer
- ✓ MIN/MAX and MINU/MAXU
- ✓ Enable density instructions
- ✓ Enable Processor ID
- ✓ Zero overhead loop instructions
- ✓ Conditional store synchronize instruction
- 4 — Miscellaneous Special Register count

ISA Configuration Options

- 64 — Number of AR registers for call windows
- Little endian — Byte order (endianness)
- Handled by hardware — Action when handling an unaligned load / store
- 8 — Max instruction width in bytes. 4 byte and greater instructions are part of the FLIX option
- 5 — Pipeline length

Figure 6: Xtensa LX configuration screen

development environment based on the ECLIPSE framework, a world-class compiler and linker, a cycle-accurate SystemC-compatible ISS, and all the other command-line and debug tools expected in a state-of-the-art development-tools suite.

Click-Box Choices

Making changes to your features as easy as checking a box to pick the options for your application. You get more choices with Xtensa LX (shown in Figure 6), but even the choices in the baseline Xtensa processor can make a difference in your design.

Designer-Defined Extensions Using TIE

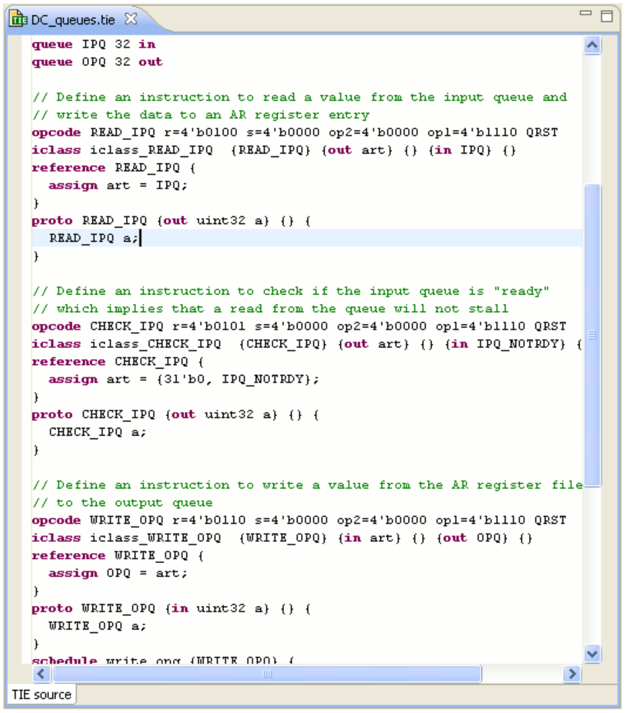
While you can select from a large number of check-box configuration options, the real power of Tensilica’s Xtensa design environment comes from the use of TIE. TIE bridges the gap between the software and hardware design realms, as it is a hybrid of C and Verilog and is very easy to learn. TIE lets you add new processor functionality in the form of instructions, execution units, wide load/store instructions, designer-defined I/O interfaces, and designer-defined register files and state registers—all without the need to modify (and then verify) the processor RTL.

The TIE Language

The TIE language defines a set of processor building blocks and a correct-by-construction methodology for designing a wide range of processor extensions such as multi-cycle, pipelined execution units, register files, state registers, and SIMD arithmetic and logic units. TIE is also used to describe I/O extensions such as wide (up to 512-bit) load/store instructions and designer-defined I/O ports, queues, and lookups. A TIE-aware editor checks syntax as TIE code is entered.

The TIE Compiler

The TIE Compiler is a tool that you use during the development of custom processor hardware extensions. The TIE Compiler also updates the compiler tool chain (XCC compiler, assembler, debugger, profiler) and the simulation models (ISS and the XTMP, XTSC, and pin-level XTSC system-modeling environments) so they understand and fully utilize these new functions.



```
queue IPQ 32 in
queue OPQ 32 out

// Define an instruction to read a value from the input queue and
// write the data to an AR register entry
opcode READ_IPQ r=4'b0100 s=4'b0000 op2=4'b0000 op1=4'b1110 QRST
iclass iclass_READ_IPQ  {READ_IPQ} {out art} {} {in IPQ} {}
reference READ_IPQ {
  assign art = IPQ;
}
proto READ_IPQ {out uint32 a} {} {
  READ_IPQ a;
}

// Define an instruction to check if the input queue is "ready"
// which implies that a read from the queue will not stall
opcode CHECK_IPQ r=4'b0101 s=4'b0000 op2=4'b0000 op1=4'b1110 QRST
iclass iclass_CHECK_IPQ  {CHECK_IPQ} {out art} {} {in IPQ_NOTRDY} {
reference CHECK_IPQ {
  assign art = {31'b0, IPQ_NOTRDY};
}
proto CHECK_IPQ {out uint32 a} {} {
  CHECK_IPQ a;
}

// Define an instruction to write a value from the AR register file
// to the output queue
opcode WRITE_OPQ r=4'b0110 s=4'b0000 op2=4'b0000 op1=4'b1110 QRST
iclass iclass_WRITE_OPQ  {WRITE_OPQ} {in art} {} {out OPQ} {}
reference WRITE_OPQ {
  assign OPQ = art;
}
proto WRITE_OPQ {in uint32 a} {} {
  WRITE_OPQ a;
}
schedule write_opq {WRITE_OPQ} {
```

Figure 3: The TIE Compiler offers pre-verified major configuration options for the Xtensa 10 DPU

You use the TIE Compiler on your local workstation/PC to determine the best instructions for the performance you need, combined with any area and power considerations. Iterating in minutes once you've determined the optimal TIE instructions for your application, that TIE file becomes input for the Xtensa Processor Generator. The Processor Generator automatically produces the entire processor RTL, including the base processor with all configuration options and all TIE extensions plus a complete matching software tool chain.

TIE Port/Queue Wizard

For making point-to-point connections between the processor and other parts of your system, Tensilica offers port and queue interfaces, along with all of the instructions needed to read and write to them. These instructions are written in TIE, but you can use the TIE Port/Queue Wizard to create them if you'd like to see how the TIE should be written. The wizard creates a TIE file based upon your input, along with a testbench showing read/write operations.

FLIX and Specialized Operations

The Xtensa LX processor incorporates Tensilica's Flexible-Length Instruction Xtensions (FLIX) architecture. FLIX allows designer-defined instructions to consist of multiple, independent operations bundled into user-defined instruction word of up to 128 bits that coexists with the native 16-bit and 24-bit Xtensa ISA. The FLIX architecture allows the implementation of highly parallel processors with a range of 2 to 30 parallel execution units, with multiple independent pipelines and register files. The Xcc compiler can schedule operations across these parallel units and pack them into FLIX bundles that are dispatched in a single cycle, achieving a very high code density. But if the units are all busy or there just aren't many operations ready, the compiler can use smaller 16- or 24-bit instructions, interleaving instructions of many different widths as needed. Thus Xtensa LX processors can deliver the high-performance characteristic of specialty long-instruction word processors without the code bloat typically incurred by such VLIW or ULIW architectures.

Summary

Cadence Tensilica has invested heavily in its Xtensa Xplorer design environment to make your task—customizing a processor for your exact application—as quick and friendly as possible. Some designers find that the base configuration is sufficient for their design. Others add pre-built options using the simple check boxes. And some designers write their own TIE for maximum performance.

Whatever way you want to work, Tensilica provides proven tools to make you more productive.