

Tensilica Xtensa

Tuan Huynh, Kevin Peek & Paul Shumate

CS 451 - Advanced Processor Architecture
November 15, 2005



Overview

- Background
- Changes in progress from Xtensa to Xtensa LX
- Automated Development Process
- ISA
- TIE Language
- Benchmarks

Tensilica

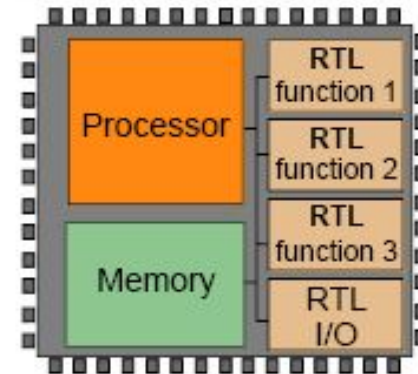
- Founded in 1997 in Santa Clara, California by a group of engineers from Intel, SGI, MIPS, and Synopsys to compete with ARC
- Goal: To address application specific microprocessor cores and software development tools by designing the first configurable and extensible processor core

Why?

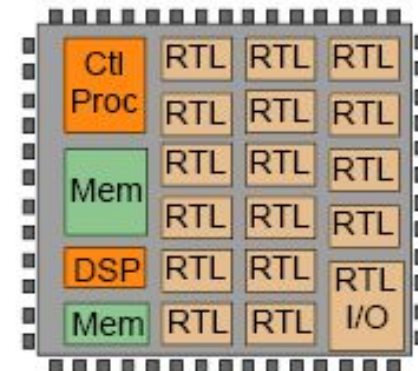
- Embedded application problems with high cost custom designs or low performance (inefficient) processors
- System on a Chip (SoC) challenge
 - Traditionally solved using hardwired RTL blocks

The Problem with RTL

- Rapidly increasing number of transistors require more RTL blocks on chip
- Hardcoded RTL blocks are not flexible
- Hand-optimized for application specific purposes



Yesterday's
SOC



Today's
SOC



Tensilica's Solution

- Xtensa

- Focusing on design through the processor, and not through hardwired RTL
- 

Xtensa

- First appearing in 1999
- 32-bit microprocessor core with a graphical configuration interface and integrated tool chain
- Designed from the start to be user customizable
- Emphasizes instruction-set configurability as its primary feature distinguishing it from other core offerings
- Has revolutionized the System on a Chip (SoC) challenge through out its development
- Configurable and Extensible

Xtensa – In a Nutshell

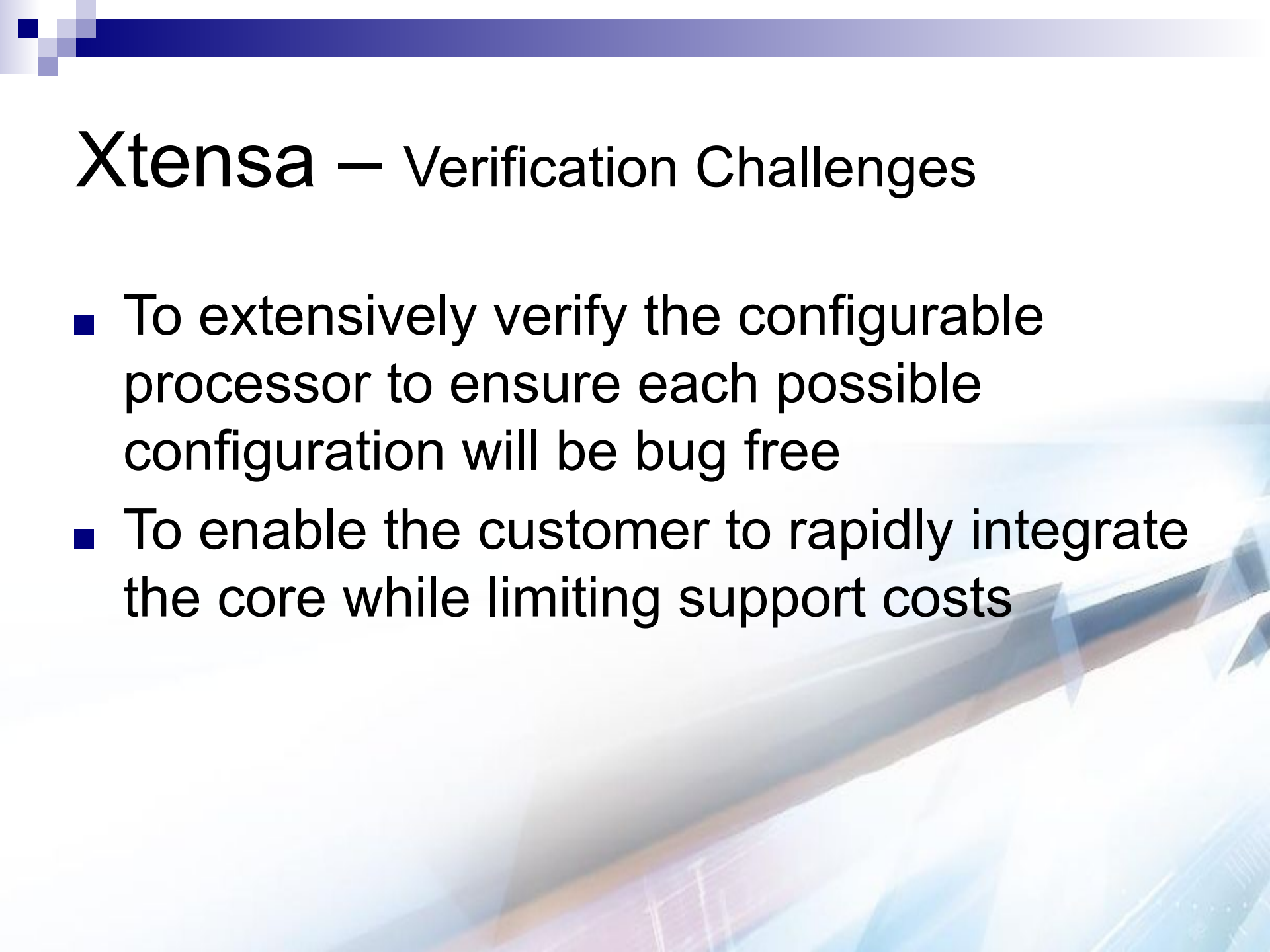
- Enables embedded system designers to build better, more highly integrated products in significantly less time
- Can add specialized functions or instructions to processor and have them recognized as “native” by the entire software development tool chain
- Move to a higher level of abstraction by designing with processors rather than RTL

Xtensa - Deliverables

- Provided as synthesizable RTL cores
 - Gate count range: 25,000 – 150,000+
 - Increase in gates as customer adds instructions or optional features
- Software development tools

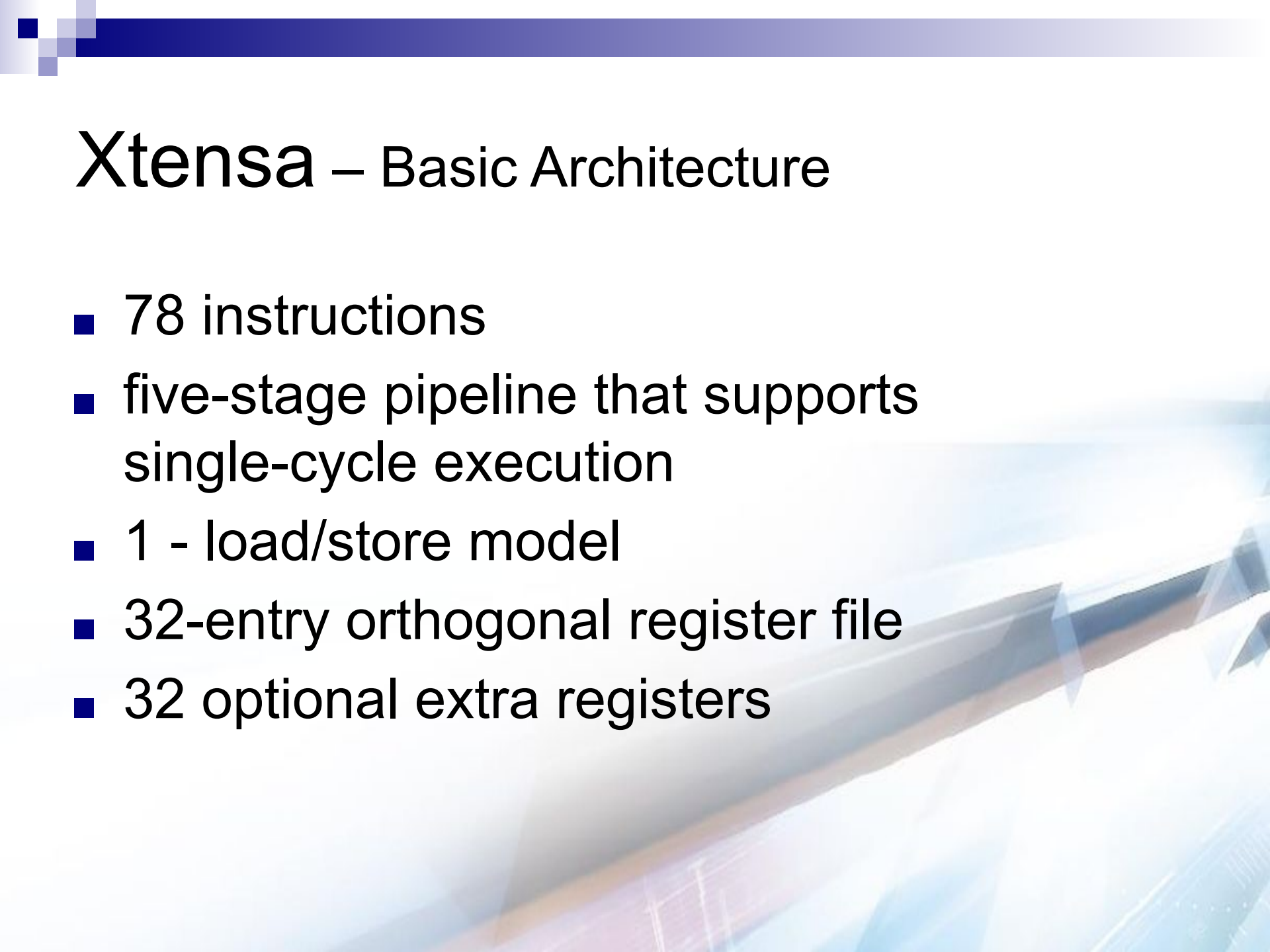


Xtensa – Verification Challenges

- To extensively verify the configurable processor to ensure each possible configuration will be bug free
 - To enable the customer to rapidly integrate the core while limiting support costs
- 



Xtensa – Basic Architecture

- 78 instructions
 - five-stage pipeline that supports single-cycle execution
 - 1 - load/store model
 - 32-entry orthogonal register file
 - 32 optional extra registers
- 

Xtensa – Basic Architecture

■ Processor Configuration

- Power Usage: 200mW, 0.25 μm , 1.5V
- Clock Speed: 170 MHz
- Cache:
 - 16 KB I-cache
 - 16 KB D-cache
 - Direct mapped
- 32 Registers (32-bits)
- Extensible via use of TIE instructions
- No Floating Point Processor
- Zero over head loops

Xtensa - ISA

- Priorities used in ISA Development
 - Code Size, Configurability, Processor Cost, Energy Efficiency, Scalability, Features
- ISA Influences
 - MIPS
 - IBM Power
 - Sun SPARC
 - ARM Thumb
 - HP Playdoh
 - DSPs

Xtensa III

- With Virtual IP Group developed an MP3 audio decoder for Tensilica's Xtensa configurable microprocessor architecture. The decoder offers hardware extensions and optimized code for accelerating MP3 decoding
- 32-bit floating point processing
- 32x32-bit hardware multiplier
- First Coprocessor interface
 - Vectra DSP enhancements

Xtensa IV

- Used white box verification methodology for the original development
- Includes 0-In Check and the CheckerWare Library made by Mentor Graphics
- Could repartition instructions up until point of manufacturing
- Support multiple processors in ASIC
- 128-bit wide local memory interface

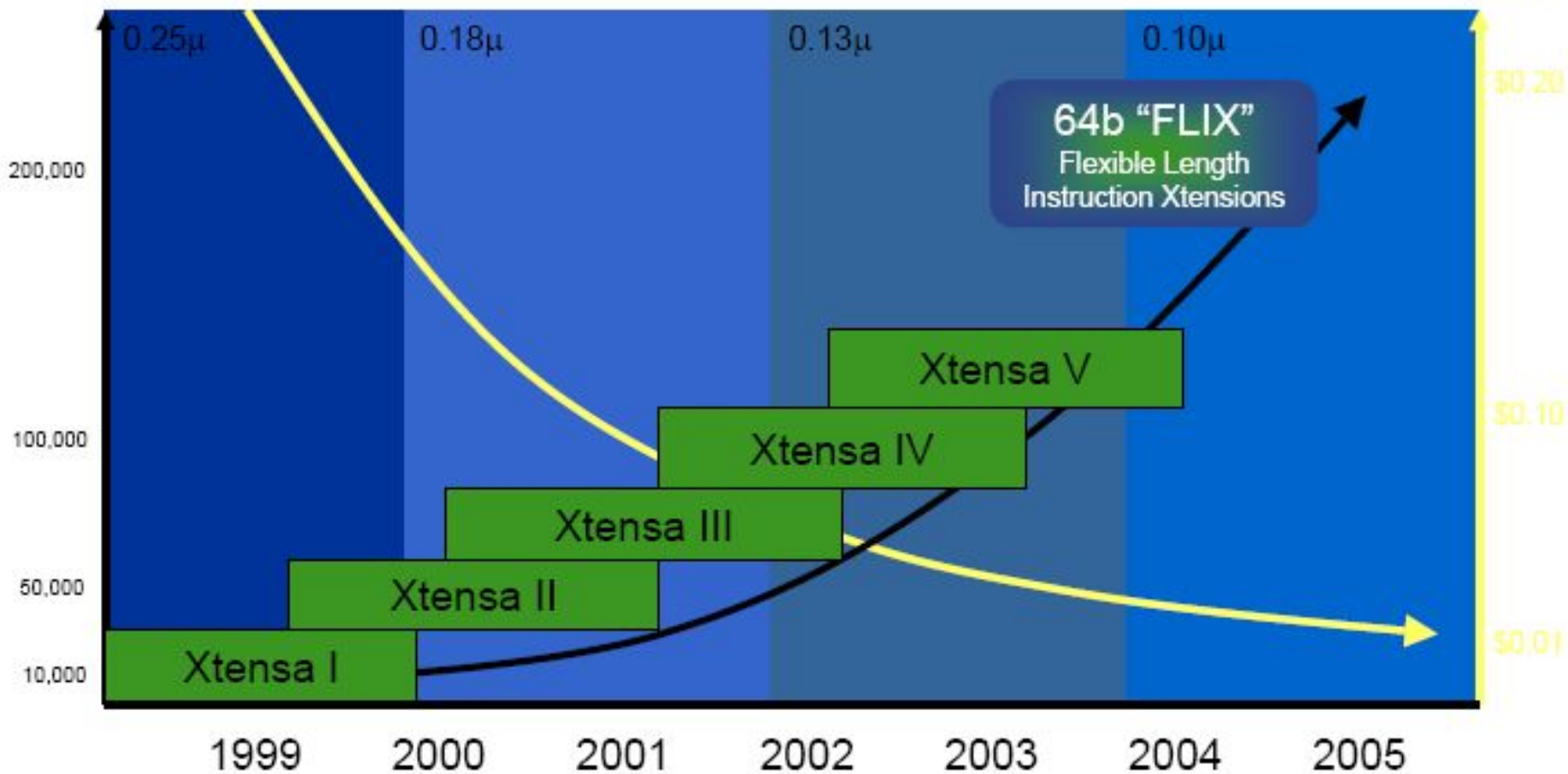
Xtensa V

- 350MHz (synthesized), as small as 18K gates (0.25mm²)
- More flexible interfaces for multiple processors
 - Write-back and write-through caches
 - Enhanced Xtensa Local Memory Interface
 - Shared data memories
- More Automation
 - Xtensa C/C++ Compiler & TIE Language improvements
 - XT2000 Emulation kit
- World's fastest embedded core

Xtensa V – Performance Cost Timeline

Performance

Cost



Performance: Millions operations/sec per multiple-processor SOC

Cost: Raw silicon cost per processor core

Xtensa 6

- Extremely fast customization path
- Three major enhancements from Xtensa V
 - Auto customize processor from C/C++ based algorithm using XPRES Compiler
 - 30% less power consumption
 - Advanced security provisions in MMU-enabled configurations

Xtensa LX

- “Fastest processor core ever” – Tensilica
 - I/O bandwidth, compute parallelism, and low-power optimization equivalent to hand-optimized, non-programmable, RTL-designed hardware blocks
 - XPRES Compiler and automated process generator
 - Uses Flexible Length Instruction Xtension (FLIX)
 - Ideal for:
 - embedded processor control tasks
 - Compute-intensive datapath hardware tasks

Xtensa 6 Vs Xtensa LX

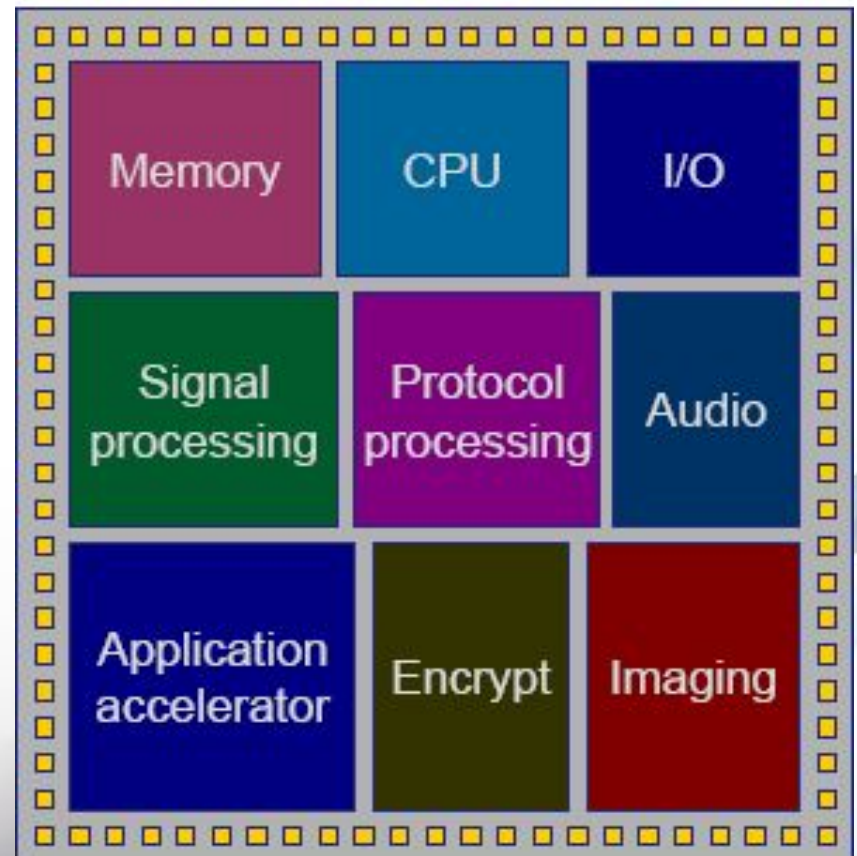
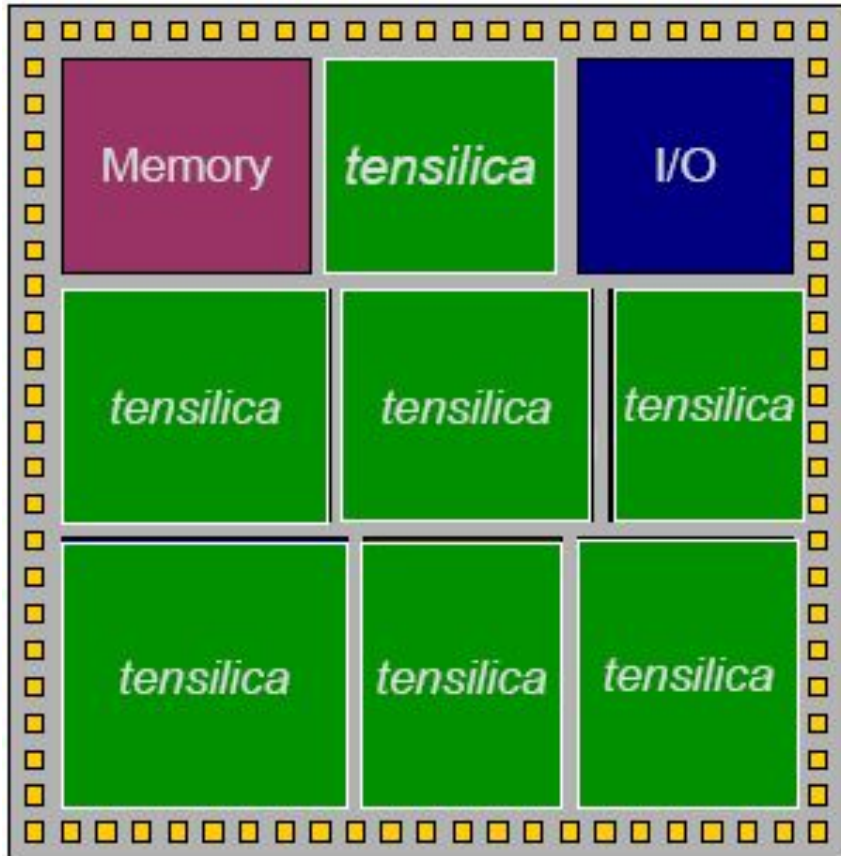
Processor Comparison		
	Xtensa 6	Xtensa LX
Major ISA Configuration Options		
MAC16	Yes	Yes
MUL16/MUL32	Yes	Yes
Floating Point Unit	Yes	Yes
Vectra LX SIMD DSP Engine	Not Available	Yes
Linux MMU	Yes	Not Available
Pipeline / Architecture Options		
Pipeline Stages	5 stage	5 stage / 7 stage
FLIX Technology	Not Available	2 - to 15 -wide issue parallel execution
Processor Interface Options		
PIF and XLMI	Yes	Yes
Available Load/Store Units	One	One or Two
Designer -Defined Ports and Queues	Not Available	Yes

Xtensa LX

- Strongest selling point is performance
- DSP operations can be encapsulated into custom instructions
- High performance leads to power savings
- Custom instructions target a special application

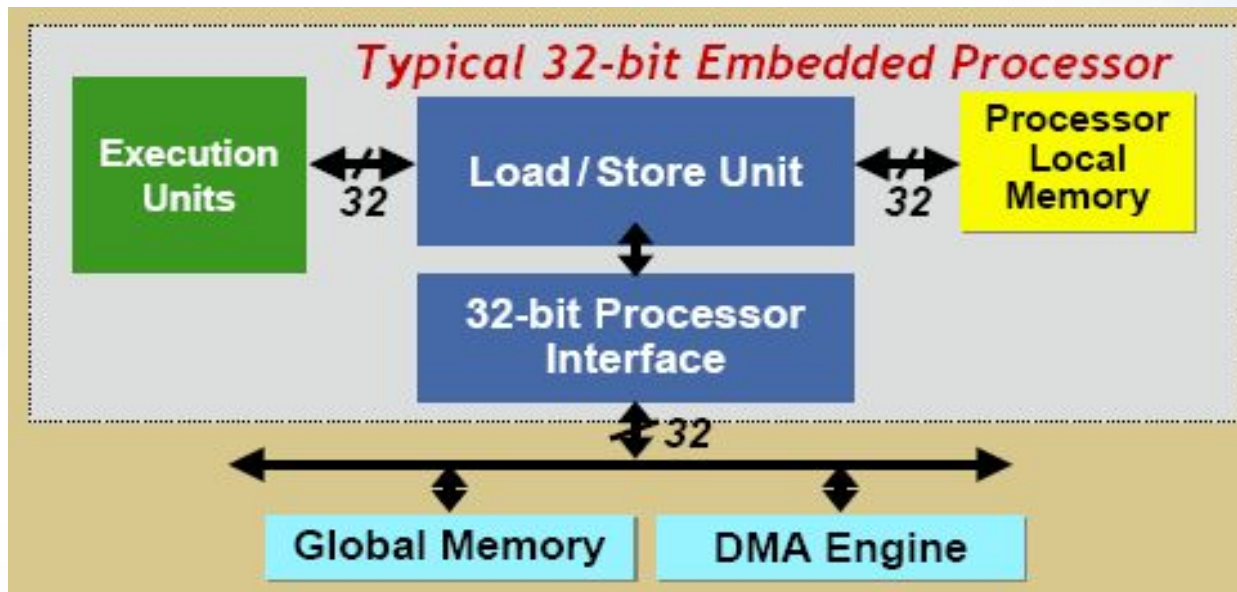
Xtensa LX Vs

General Purpose



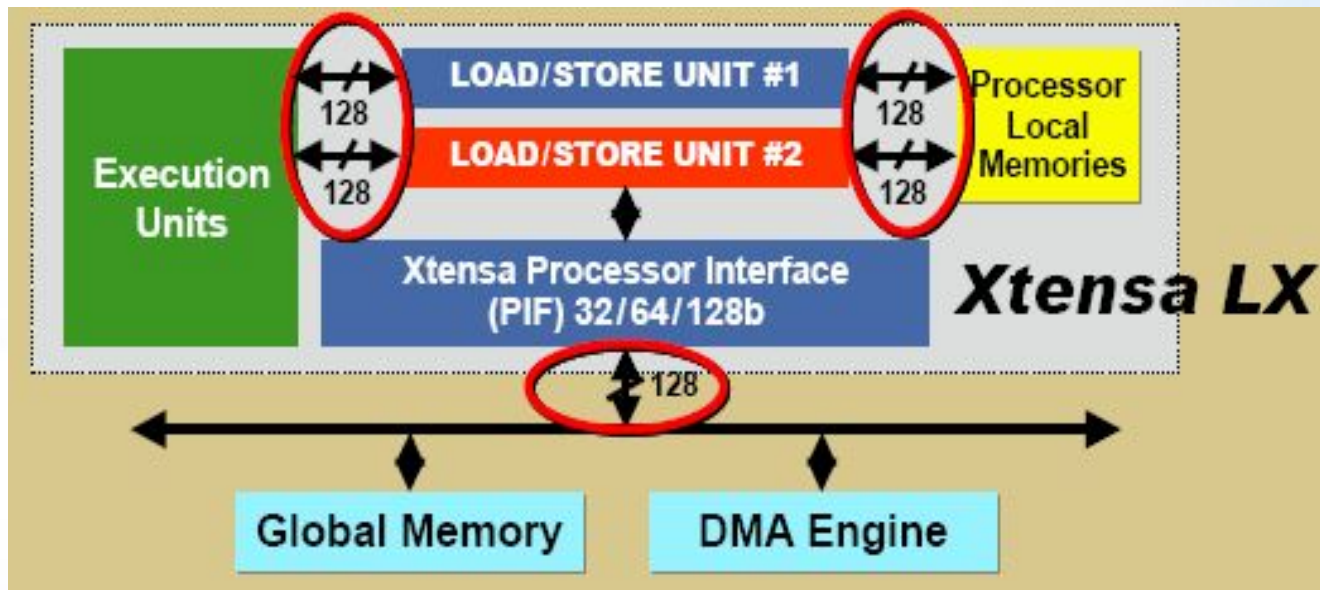
Xtensa LX – Traditional Limitations

- 1 Operation / cycle
- Load/Store overhead



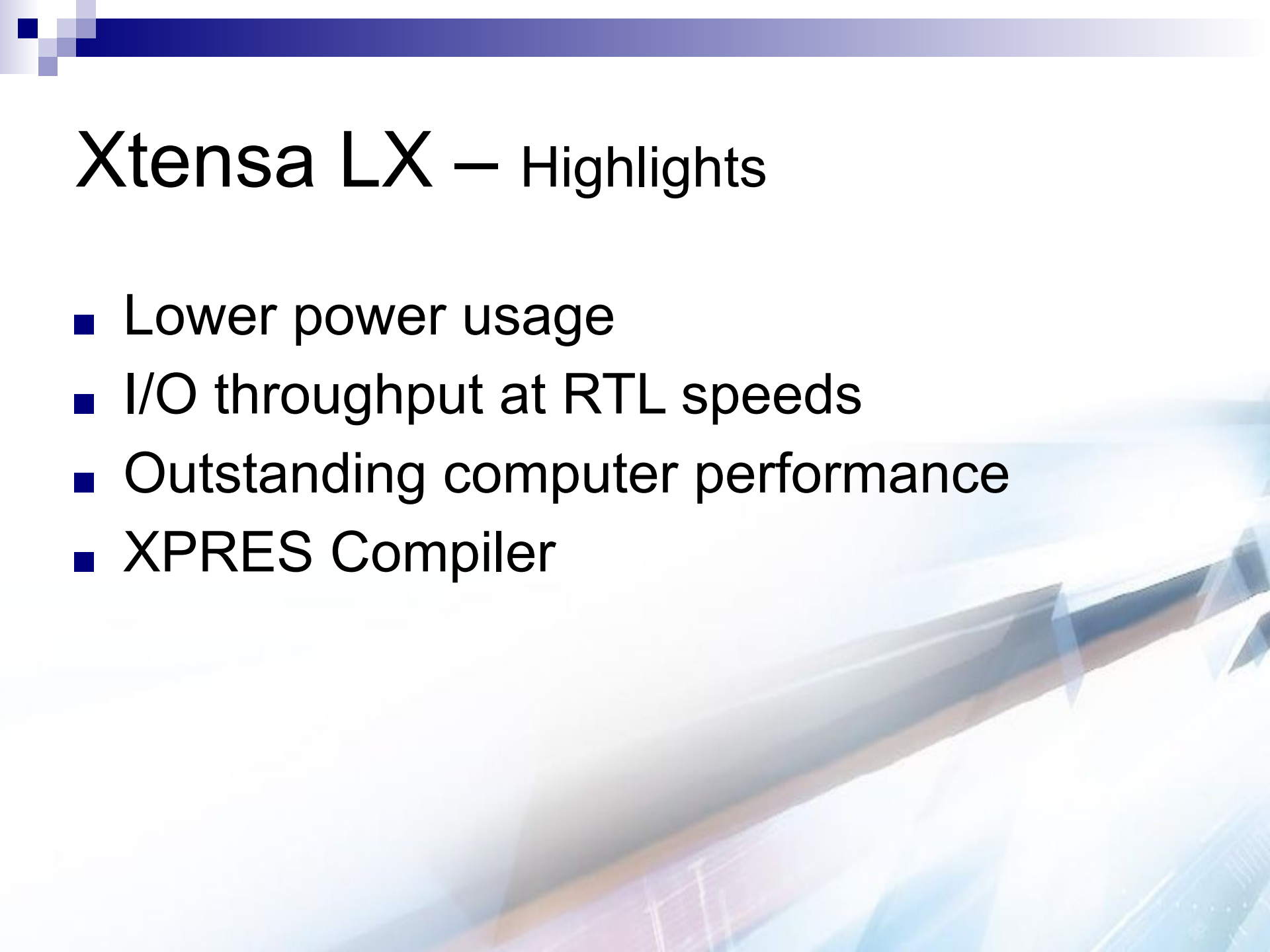
Xtensa LX

- Options:
 - Extra load/store unit, wide interfaces, compound instructions
- Up to 19 GB/sec of throughput





Xtensa LX – Highlights

- Lower power usage
 - I/O throughput at RTL speeds
 - Outstanding computer performance
 - XPRES Compiler
- 

Xtensa LX – Lower Power Usage

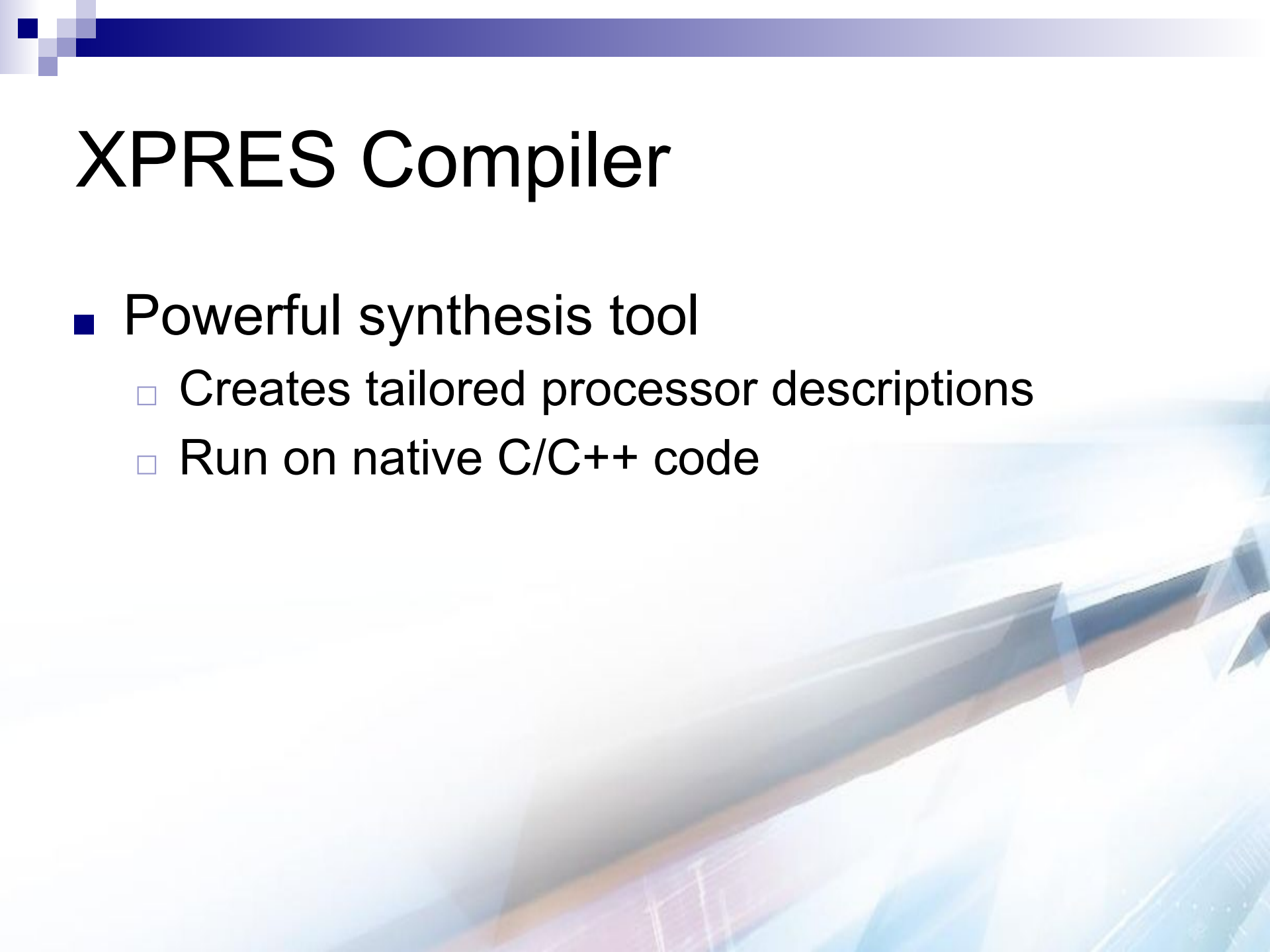
- Automated the insertion of fine-grain clock gating for every functional element of the Xtensa LX processor
 - This includes functions created by the designer
 - Direct I/O capability – like RTL

Outstanding Computing Performance

- Extensible using FLIX
(Flexible Length Instruction Xtensions)
 - Similar to VLIW – but customizable to fit application code's needs
- Significant improvement over competitors and previous Xtensa Design
 - DSP instructions formed using FLIX to be recognized as native to entire development system



XPRES Compiler

- Powerful synthesis tool
 - Creates tailored processor descriptions
 - Run on native C/C++ code
- 

Automated Development

- Clients log into website
 - Accessing *Process Generator*
- Builds a model in RTL Verilog or VHDL
 - Sends result via internet to client's site
- Also receive:
 - Preconfigured synthesis scripts, test benches, and software-development tools
- Software tools include:
 - Assembler, C/C++ compiler, linker, debugger, and instruction-set simulator already modified to match the hardware configuration

Automated Development

- Create special instructions described and written in TIE
- TIE semantics allow system to modify software-development tools
- Integrates changes into processor design
- Compile with synthesis tool – test – order

Xtensa LX – Basic Architecture

■ Processor Configuration

- Power Usage: 76 $\mu\text{W}/\text{MHz}$, 47 $\mu\text{W}/\text{MHz}$ (5 and 7 stage pipeline)
- Clock Speed: 350 MHz, 400 MHz (5 and 7 stage pipeline)
- Cache:
 - up to 32 KB and 1,2,3,4 way set associative cache
- 64 general purpose physical registers (32-bits)
- 6 special purpose registers
- Extensible via use of TIE and FLIX instructions
- Zero over head loops

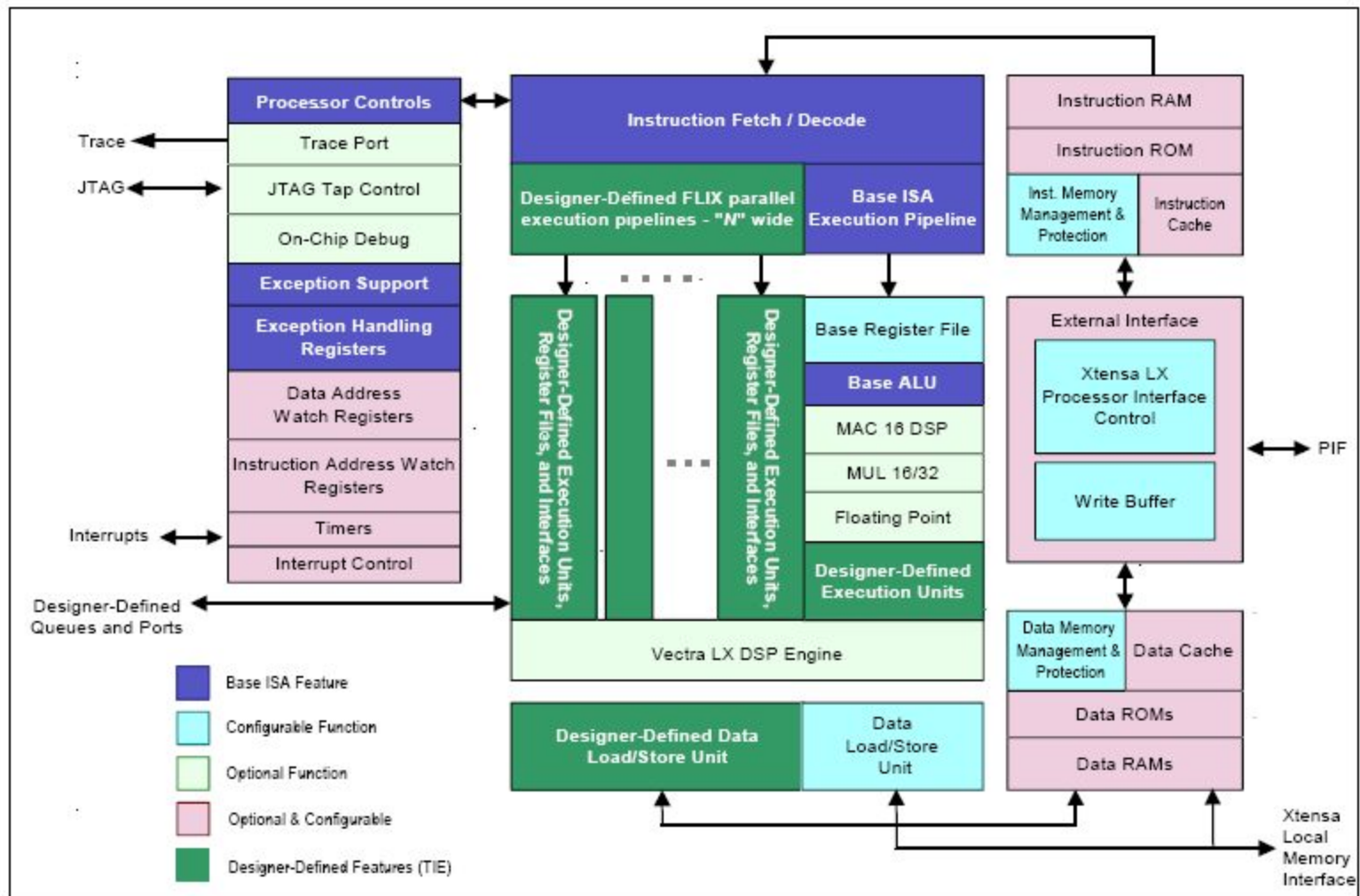
Xtensa LX Architecture

- 32-bit ALU
- 1 or 2 Load/Store Model
- Registers
 - 32-bit general purpose register file
 - 32-bit program counter
 - 16 optional 1-bit boolean registers
 - 16 optional 32-bit floating point registers
 - 4 optional 32-bit MAC16 data registers
 - Optional Vectra LX DSP registers

Xtensa LX Architecture

- General Purpose AR Register File
 - 32 or 64 registers
 - Instructions have access through “sliding window” of 16 registers. Window can rotate by 4, 8, or 12 registers
 - Register window reduces code size by limiting number of bits for the address and eliminated the need to save and restore register files

Xtensa LX Architecture



Xtensa LX Pipelining

- 5 or 7 Stage Pipeline Design
- 5 stage pipeline has stages: IF, Register Access, Execute, Data-Memory Access, and register writeback
- 5 stage pipeline accesses memory in two stages. 7 stage pipeline is extended version of the 5 stage pipeline with extra IF and Memory Access stage. Extra stages provide more time for memory access. Designer can run at a higher clock speed while using slower memory to improve performance

Xtensa LX Instruction Set

- ISA consists of 80 core instructions including both 16 and 24 bit instructions

Instruction Category	Instructions
Load	L8UI, L16SI, L16UI, L32I, L32R
Store	S8I, S16I, S32I
Memory Ordering	MEMW, EXTW
Jump, Call	CALL0, CALLX0, RET J, JX
Conditional Branch	BALL, BNALL, BANY, BNONE BBC, BBCI, BBS, BBSI BEQ, BEQI, BEQZ BNE, BNEI, BNEZ BGE, BGEI, BGEU, BGEUI, BGEZ BLT, BLTI, BLTU, BLTUI, BLTZ
Move	MOVI, MOVEQZ, MOVGEZ, MOVLTZ, MOVNEZ
Arithmetic	ADDI, ADDMI, ADD, ADDX2, ADDX4, ADDX8, SUB, SUBX2, SUBX4, SUBX8, NEG, ABS
Bitwise Logical	AND, OR, XOR
Shift	EXTUI, SRLI, SRAI, SLLI SRC, SLL, SRL, SRA SSL, SSR, SSAI, SSA8B, SSA8L
Processor Control	RSR, WSR, XSR, RUR, WUR, ISYNC, RSYNC, ESYNC, DSYNC

Xtensa LX Instruction Set

■ Processor Control Instructions

- RSR, WSR, XSR
 - Read Special Register, Write Special Register
 - Used for saving and restoring context, Processing Interrupts and Exceptions, Controlling address translation
- RUR, WUR
 - Access User Registers
 - Used for Coprocessor registers and registers created with TIE
- ISYNC – wait for Instruction Fetch related changes to resolve
- RSYNC – wait for Dispatch related changes to resolve
- ESYNC/DSYNC – Wait for memory/data execution related changes to resolve

Xtensa LX ISA – Building Blocks

- MUL32

- MUL32 adds 32 bit multiplier

- MUL16 and MAC16

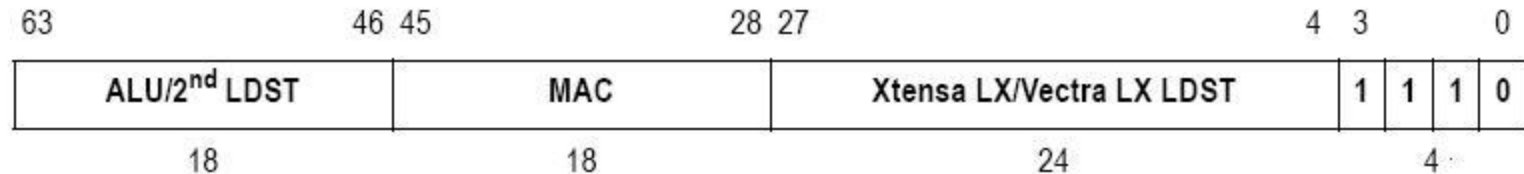
- MUL16 adds 16x16 bit multiplier
- MAC16 adds 16x16 bit multiplier and 40-bit accumulator

Xtensa LX ISA – Building Blocks

- Floating Point Unit
 - 32-bit, single precision, floating-point coprocessor
- Vectra LX DSP Engine
 - Optimized to handle Digital Signal Processing Applications

Vectra LX DSP Engine

- FLIX-based (why it is 64 bits)
- Vectra LX instructions encoded in 64 bits.



- Bits 0:3 of a Xtensa instruction determine its length and format, the bits have a value of 14 to specify it is a Vectra LX instruction
- Bits 4:27 – contain either Xtensa LX core instruction or Vectra LX Load or Store instruction
- Bits 28:45 – contains either a MAC instruction or a select instruction
- Bits 46:63 – contains either ALU and shift instructions or a load and store instruction for the second Vectra LX load/store unit

Vectra LX DSP Engine

Vectra DSP Engine Configurations					
	Vectra V1620-8	Vectra V1620-4	Vectra V1616-8	Vectra V0810-8	Vectra V3224-4
Elements per vector A	8	4	8	8	4
Memory width of each element B	16	16	16	8	32
Register width of each element C	20	20	16	10	24
Number of MAC units D	4	2	4	4	2
Multiplier and Multiplicand width E	16x16	16x16	16x16	8x8	24x24

Tensilica Instruction Extension

- Method used to extend the processor's architecture and instruction set
- Can be used in two ways:
 - For the TIE Compiler
 - For the Processor Generator

Tensilica Instruction Extension

■ TIE Compiler

- Generates file used to configure software development tools so that they recognize TIE Extensions
- Estimates hardware size of new instruction
- You can modify application code to take advantage of the new instruction and simulate to decide if the speed advantage is worth the hardware cost

TIE

- Resembles Verilog
- More concise than RTL (it omits all sequential logic, pipeline registers, and initialization sequences.
- The custom instructions and registers described in TIE are part of the processor's programming model.

TIE Queues and Ports

- New way to communicate with external devices
- Queues: data can be sent or read through queues. A queue is defined in the TIE and the compiler generates the interface signals required for the additional port needed to connect to the queue. Logic is also automatically generated
- Import-wire: processor can sample the value of an external signal
- Export-state: drive an output based on TIE

TIE

- TIE Combines multiple operations into one using:
 - Fusion
 - SIMD/Vector Transformation
 - FLIX

Fusion

- Allows you to combine dependent operations into a single instruction
- Consider: computing the average of two arrays

```
unsigned short *a, *b, *c;  
.  
.  
.  
for( i = 0; i < n; i++)  
  c[i] = (a[i] + b[i]) >> 1;
```

- Two Xtensa LX Core instructions required, in addition to load/store instructions

Fusion

- Fuse the two operations into a single TIE instruction

```
operation AVERAGE{out AR res, in AR input0, in AR input1}{}{  
    wire [16:0] tmp = input0[15:0] + input1[15:0];  
    assign res = tmp[16:1];  
}
```

- Essentially an add feeding a shift, described using standard Verilog-like syntax

- Implementing the instruction in C/C++

```
#include <xtensa/tie/average.h>  
unsigned short *a, *b, *c;  
.  
.  
.  
for( i = 0; i < n; i++)  
    c[i] = AVERAGE(a[i] + b[i]);
```


SIMD/Vector Transformation

- Single Instruction, Multiple Data
 - Fusing instructions into a “vector”
 - Allows replication of the same operation multiple times in one instruction
- Consider: Computing four averages in one instruction
 - The following TIE code computes multiple iterations in a single instruction by combining Fusion and SIMD

```
regfile VEC 64 8 v
```

```
operation VAVERAGE{out VEC res, in VEC input0, in VEC input1} {} {  
wire [67:0] tmp = { input0[63:48] + input1[63:48],  
                  input0[47:32] + input1[47:32],  
                  input0[31:16] + input1[31:16],  
                  input0[15:0] + input1[15:0] };  
assign res = {tmp[67:52], tmp[50:35], tmp[33:18], tmp[16:1]};  
}
```

SIMD/Vector Transformation

- Computing four 16-bit averages

- Each data vector must be 64 bits (4 x 16 bits)

- Create new register file, new instruction

- VEC - eight 64-bit registers to hold data vectors
- VAVERAGE - takes operands from VEC, computes average, saves results into VEC

```
VEC *a, *b, *c;  
for (i = 0; i < n; i += 4){  
    c[i] = VAVERAGE( a[i], b[i] );}
```

- New Datatype recognized

- TIE automatically creates new load, store instructions to move 64-bit vectors between VEC register file and memory

FLIX

- Flexible length instruction extension
 - Key in extreme extensibility
 - Huge performance gains possible
 - Code size reduction without code bloat
- Similar to VLIW
- Created by XPRES Compiler

FLIX

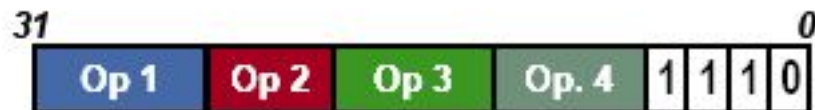
Designer-Defined FLIX Instruction Formats with Designer-Defined Number of Operations



Example: 3-Operation, 64-bit Instruction Format



Example: 5-Operation, 64-bit Instruction Format



Example: 4-Operation, 32-bit Instruction Format

FLIX - Usage

- Used selectively when parallelism is needed
- Avoids code bloat
- Used seamlessly and modelessly used with standard 16- and 24-bit instructions

XPRES Compiler

- Powerful synthesis tool
 - Creates tailored processor descriptions
 - Run on native C/C++ code
- Three optimizations methods
- Returns optimal configurations along with pros and cons (tradeoffs)

XPRES Compiler

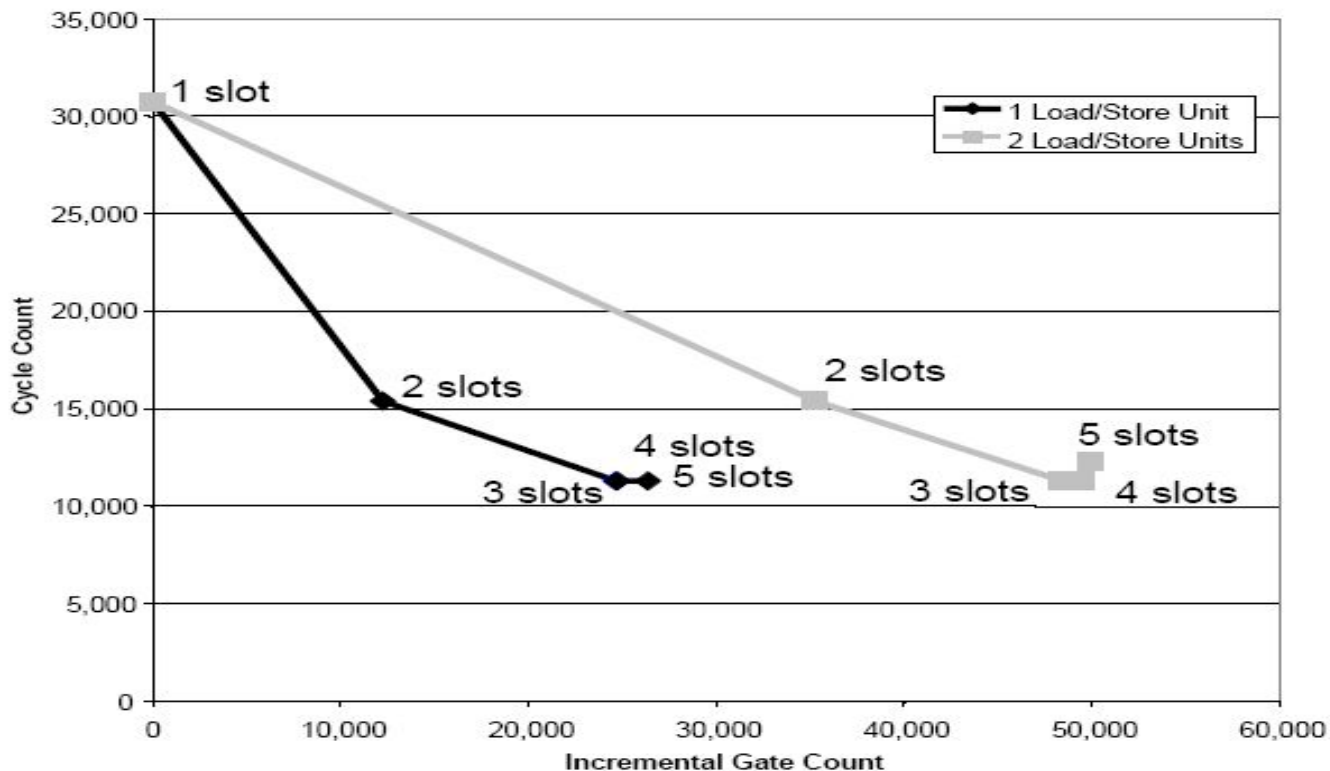
- Analyzes C/C++ code
- Generates possible configurations
- Compares performance criteria to silicon size (cost)
- Returns possible configurations

XPRES Compiler - Results

- Application dependent
 - Compute intensive programs
 - Data intensive programs
- More is sometimes less
 - operation slots in FLIX

XPRES – 4 Program Test

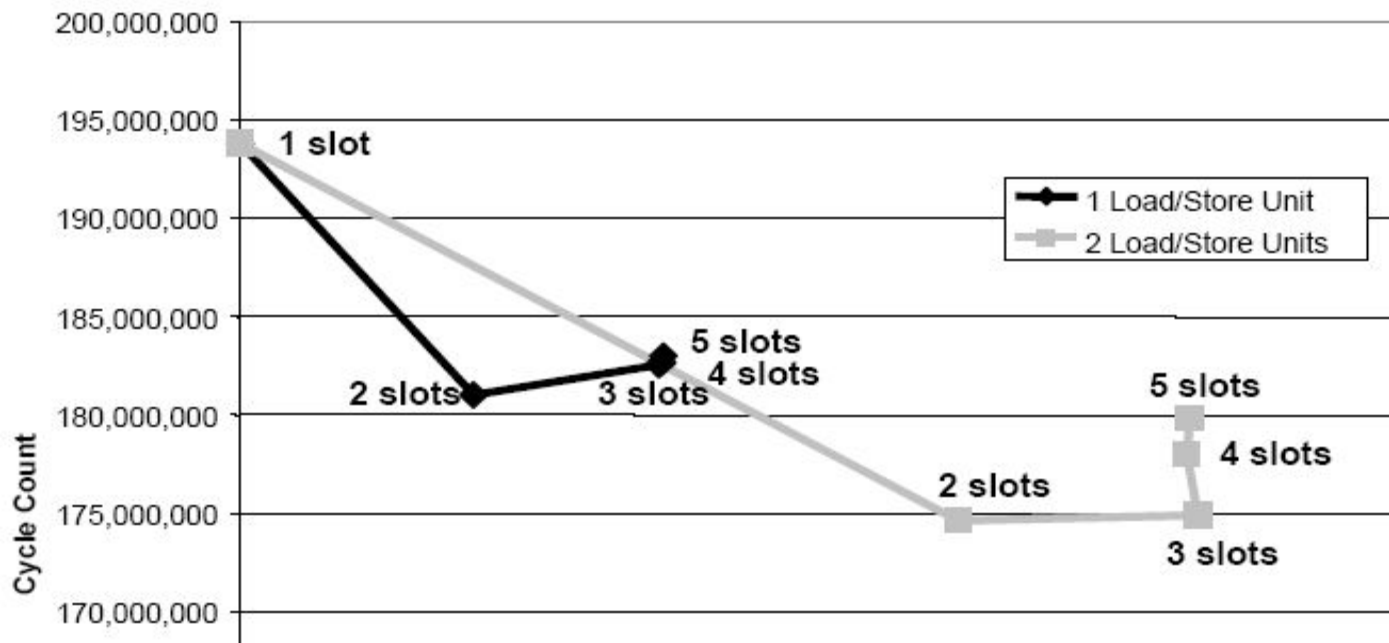
- “Bit Manipulator” program
- Cut cycles to a third



XPRES – 4 Program Test

- H.264 Deblocking Filter

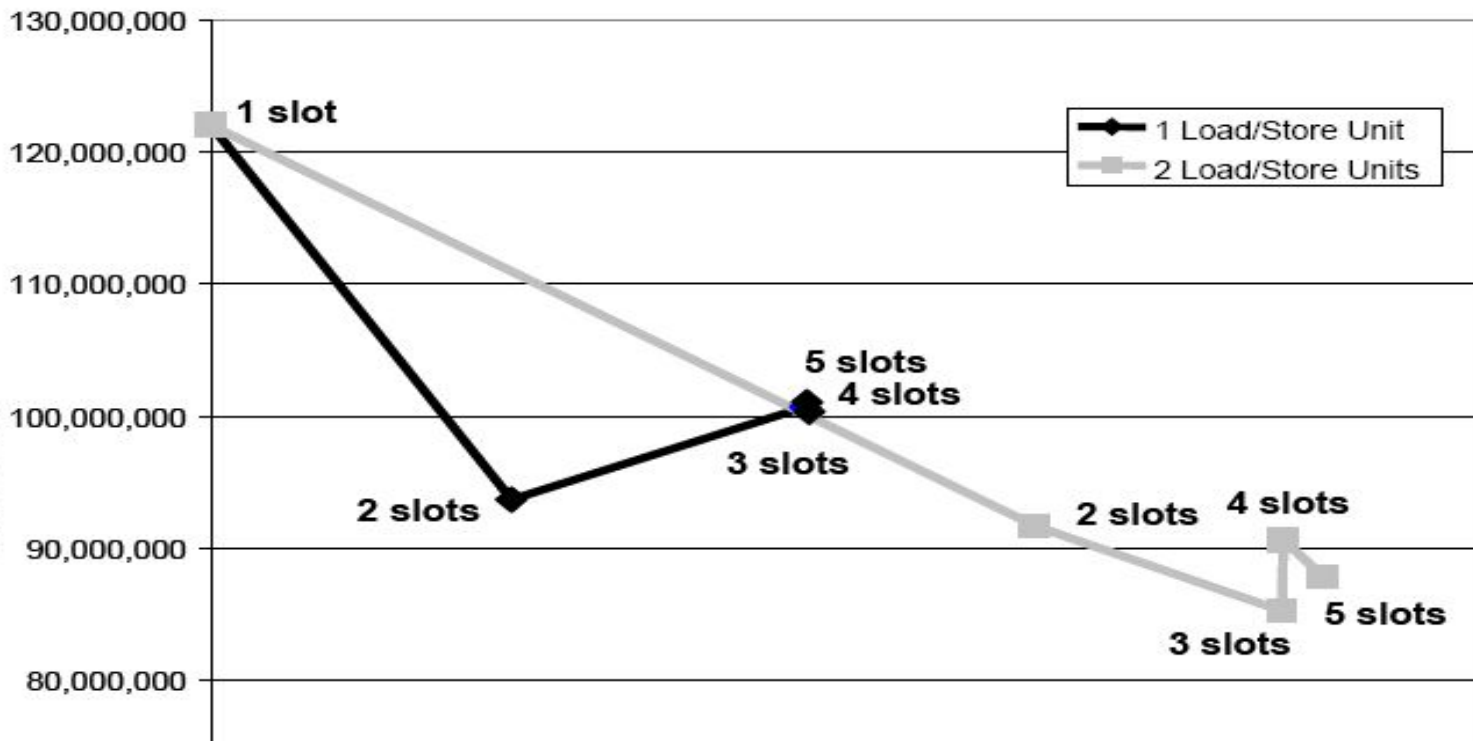
- 6% performance improvement



XPRES – 4 Program Test

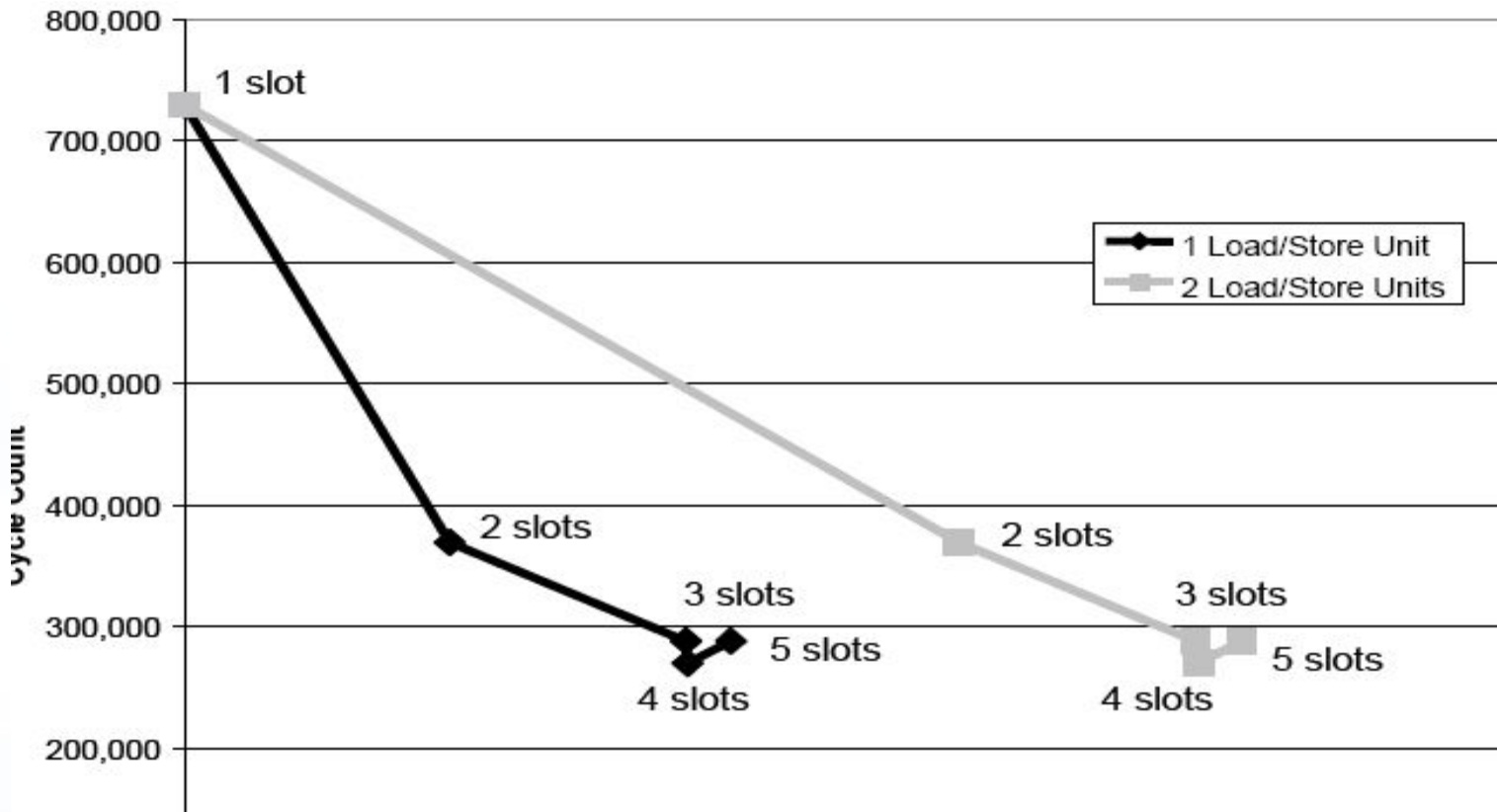
- MPEG4 decoder

- 23% performance increase



XPRES – 4 Program Test

- SAD – sum of absolute difference
 - 63% performance increase



Xtensa Hi-Fi 2 Audio Engine

- Add-on package for Xtensa LX
- Advantages over common audio processors:
 - better sound quality of compressed files because of increased precision available for intermediate calculations. (24 bits rather than 16)
 - 24-bit audio fully compatible with modern audio standards

Xtensa Hi-Fi 2 Audio Engine

- Audio packages integrated into an SOC design, so no additional codec development required
- Integrated Audio Packages:
 - Dolby Digital AC-3 Decoder, Dolby Digital AC-3 Consumer Encoder, QSound MicroQ, MP3 Encoder/Decoder, MPEG-4 aacplus v1 and v2 Encoder/Decoder, MPEG-2/4 AAC LC Encoder/Decoder, WMA Encoder/Decoder, AMR narrowband speech codec, AMR wideband speech codec.

Xtensa Hi-Fi 2 Audio Engine

- Uses over 300 audio specific DLP instructions.
- Features dual-multiply accumulate for 24x24 and 32x16 bit arithmetic on both units
- “delivers noticeably superior sound quality even when decoding prerecorded 16-bit encoded music files. “

Speed-up Example

- GSM Audio Codec – written in C
- Profiling code using unaltered RISC architecture showed that 80% of the processor cycles were devoted to multiplication
- Simply by adding a hardware multiplier, the designer can reduce the number of cycles required from 204 million to 28 million

Speed-up Example

- Viterbi butterfly instruction
 - Acts like compression for the data
 - Consists of 8 logical operation
 - 8 of these operations are used to decode each symbol in the received digital information stream
 - The designer can add a Viterbi instruction to the Xtensa ISA. The extension can use the 128-bit memory bus to load data for 8 symbols at once. This results in a average execution time of 0.16 cycles per butterfly. An unaugmented Xtensa LX executes Viterbi in 42 cycles.

EEMBC Networking Benchmark

- Xtensa LX received highest benchmark ever achieved on the Networking version 2 test.
- Xtensa LX has a 4x code density advantage and a 100x advantage in both die area and power dissipation

EEMBC Networking Benchmark

- Normalized (per MHz)
EEMBC TCPmark
- Simulates
performance in
internet enabled client
side performance

Processor	Score
Xtensa LX Optimized	1.62434
PowerPC 760GX	0.4671
PowerPC MCP7447A	0.5856
Xtensa LX Out of the Box	0.33762

EEMBC Networking Benchmark

- Normalized (by MHz)
EEMBC IPmark
- Simulates
performance in
network routers,
gateways, and
switches

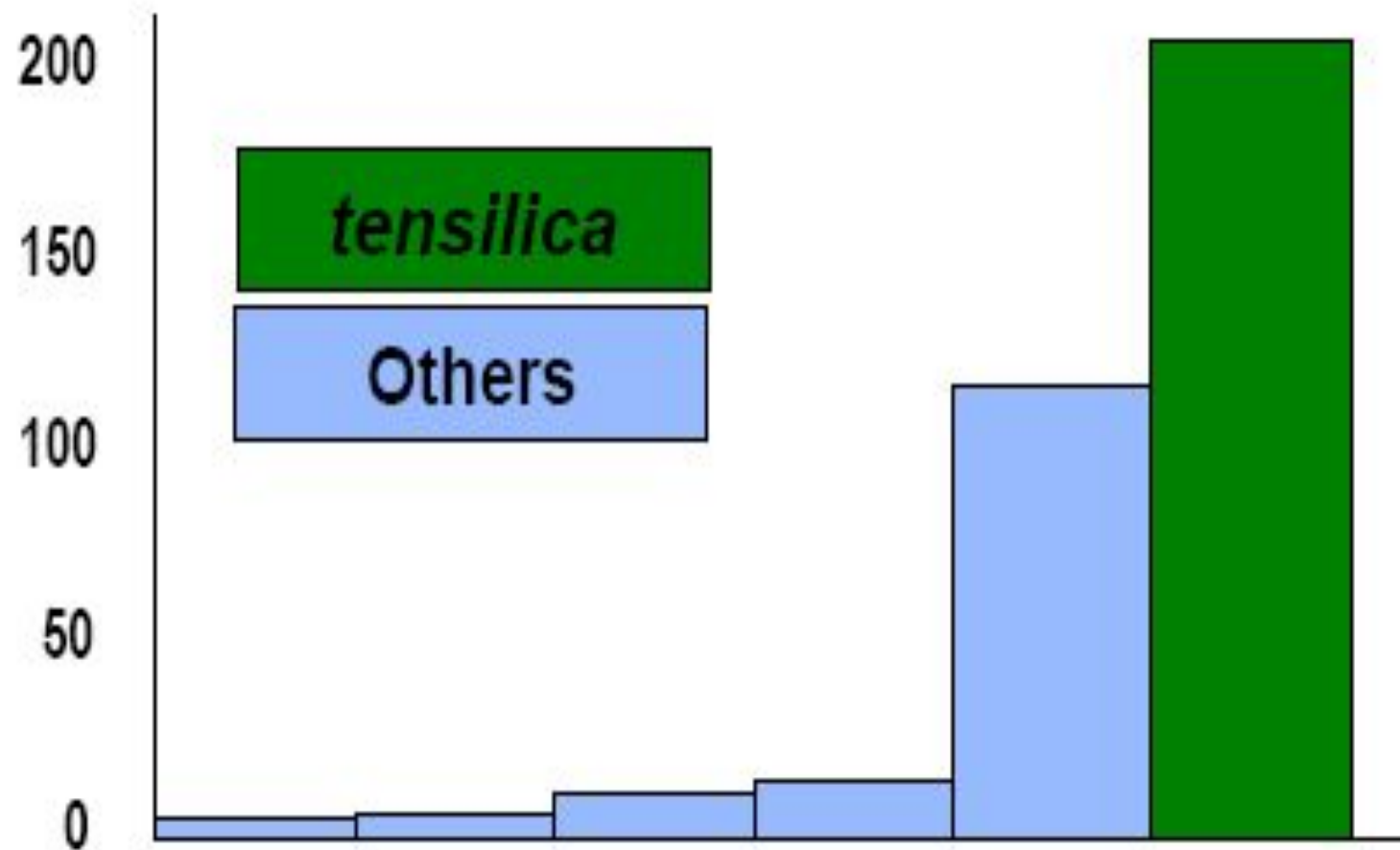
Processor	Score
Xtensa LX Optimized	0.82138
PowerPC 760GX	0.2861
Xtensa LX Out of the Box	0.1818
PowerPC MCP7447A	0.1751

EEMBC Networking Benchmark

- Total Code Size

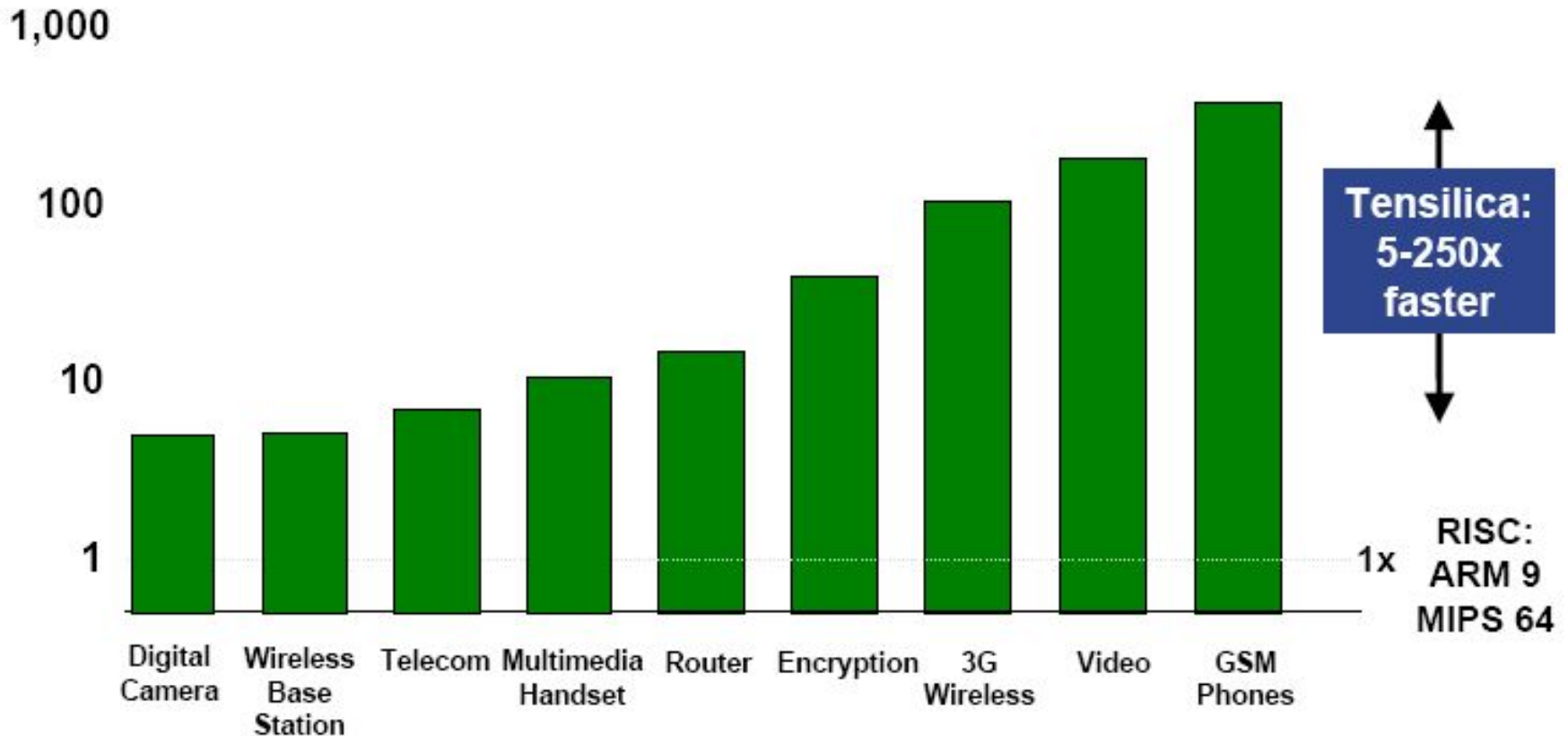
Processor	Total Size of Code
Xtensa LX Optimized	65,208
Xtensa LX Out of the Box	67,256
PowerPC 760GX	255,764
PowerPC MCP7447A	280,984

EDN Embedded Consumer Benchmark



Xtensa Optimized Performance @ 200MHz (0.18 μ): <http://www.eembc.org>

How Xtensa Compares



Source: Tensilica
Performance of Tensilica vs. RISC (ARM 9 / MIPS 64) in core algorithms

How Xtensa Compares


Architecture (ISA)	MIPS32	MIPS32	ARCompact	Xtensa V
	32-bit RISC	32-bit RISC	32-bit RISC	32-bit RISC
Target Applications	Gen-purpose embedded	Smartcards embedded	Gen-purpose embedded	Gen-purpose embedded
Synthesizable	Yes	Yes	Yes	Yes
Core HDL	Verilog	Verilog	Verilog, VHDL	Verilog, VHDL
User-Extension HDL	Verilog	Verilog	Verilog, VHDL	TIE
Configurability	Medium	Medium	High	HIGH
Custom Instr Slots	>16 million	>16 million	256	32,768
Custom Core-Reg Slots	0	0	32	32
Multicycle Custom Instr	Yes	Yes	Yes	Yes
Multiple Reg Files	No	No	Optional	Optional
Custom Instr Formats	3 operands	3 operands	1-3 operands	1-4 Operands
Standard Instr Length	16/32-bit	16/32-bit	16/32-bit	16/24-bit
Custom Instr Length	32-bit	32-bit	16/32-bit	24-bit
Graphical Config Tool	No	No	Yes	yes
Predefined Extensions	No	No	Yes	yes
DSP Extensions	No	No	Optional	Optional

How Xtensa Compares (cont)

Config I/O Buses	No	No	Yes	Yes
Config Interrupts	No	No	Yes	Yes
Config Caches	0-64K	0-64K	0-32K	0-32K
Config Scratchpad RAM	Yes	No	Yes	Yes
Config Condition Flags	No	No	Yes	no
Config Endian Order	No	No	No	Yes
Config Peripheral IP	No	No	Yes	no
Fast Mult/Div Unit	Yes	No	Optional	Optional
FPU	No	No	No	Optional
MMU	Yes	Yes	No	Optional
Config Dev Tools	Yes	Yes	Yes	Yes
Tool-Chain Automation	None	None	Low	HIGH
Automation				
Availability	Now	Now	Now	Now



Uses of Xtensa Products

- NVIDIA – Licensed Xtensa LX
 - “We were very impressed with Tensilica's automated approach for both the processor extensions and the generation of the associated software tools”
- 

Uses of Xtensa Products

■ LG Cell Phone

- Phone is digital broadcast enabled
- Xtensa processor was used because it enabled LG to “cut design time significantly and be first to market with this exciting new technology.”
- Terrestrial digital-multimedia-broadcast system in Korea

In case you are wondering..

- --Tensilica's announced licensees include Agilent, ALPS, AMCC (JNI Corporation), Astute Networks, ATI, Avision, Bay Microsystems, Berkeley Wireless Research Center, Broadcom, Cisco Systems, Conexant Systems, Cypress, Crimson Microsystems, ETRI, FUJIFILM Microdevices, Fujitsu Ltd., Hudson Soft, Hughes Network Systems, Ikanos Communications, LG Electronics, Marvell, NEC Laboratories America, NEC Corporation, NetEffect, Neterion, Nippon Telephone and Telegraph (NTT), NVIDIA, Olympus Optical Co. Ltd., sci-worx, Seiko Epson, Solid State Systems, Sony, STMicroelectronics, Stretch, TranSwitch Corporation, and Victor Company of Japan (JVC).

Questions?

