The Configurable Processor Company

tensilica

A Second-Generation High-Performance DSP Engine

MAY 18, 2004

Beatrice Fu, Senior VP of Engineering
Tensilica Inc.

DSP: Challenges and Current Solutions

Challenges from new digital applications

Video Example

	90's	Now	
Resolution	Full D1	HD	10X data
Compression	MPEG2	H.264	30X compute
Power consumption	Line	Line & battery	Same or lower

Wireless Example

	Late 90's	Now	
Bit rate	802.11b	802.11a/g	5X data rate
Transmission	DSSS/CCK	OFDM	10X compute
Power consumption	Line & battery	Line & battery	Same or lower

- **Higher bandwidth! More compute! Lower power consumption!**
- **Programmability a *must*** (complex algorithms, evolving standards, short product cycle)

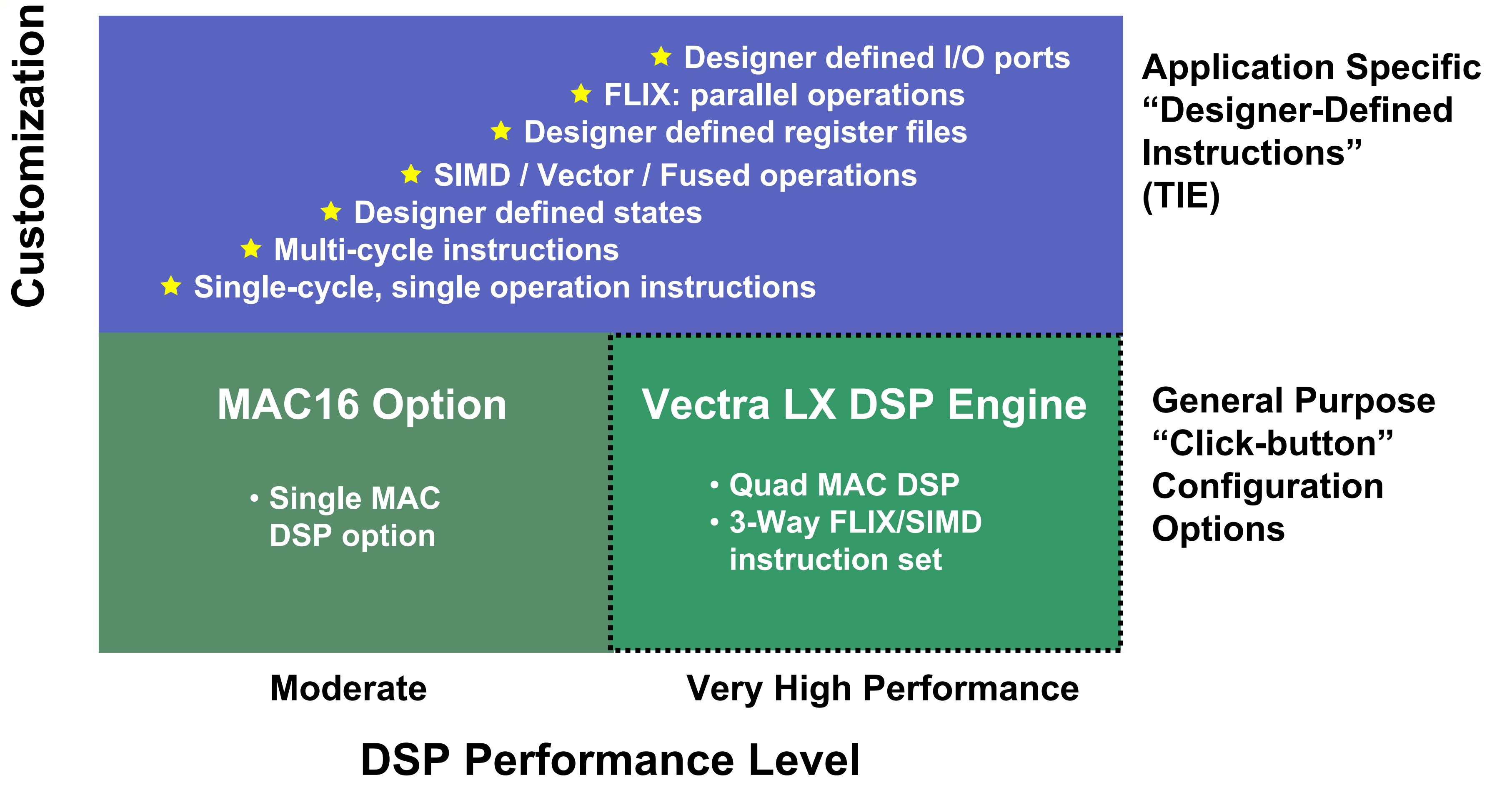
Today's DSP solutions

	PRO	CON
General purpose DSP processor cores	Programmable	Lack efficiency for application specific algorithms
Hardwired digital signal processing blocks	Highly efficient implementation of algorithms	Rigid, Non-programmable

Goals for Tensilica DSP Engines

- Appropriate *performance* for a wide range of applications.
- Appropriate *efficiency* for a wide range of applications.
- *Programmable*. Robust software tools.
- *Synthesizable*. Implement in any SOC methodology.

Xtensa LX Offers a Wide Range of DSP Solutions



Customization

★ Designer defined I/O ports

★ FLIX: parallel operations

★ Designer defined register files

★ SIMD / Vector / Fused operations

★ Designer defined states

★ Multi-cycle instructions

★ Single-cycle, single operation instructions

Application Specific
“Designer-Defined
Instructions”
(TIE)

MAC16 Option

- Single MAC DSP option

Vectra LX DSP Engine

- Quad MAC DSP
- 3-Way FLIX/SIMD instruction set

General Purpose
“Click-button”
Configuration
Options

Moderate

Very High Performance

DSP Performance Level

Vectra LX:
High Performance DSP Engine

130 nm Technology

Frequency	*Power*
370MHz	0.50mW/MHz
100MHz	0.25mW/MHz

- **Configuration option to synthesizable Xtensa LX processor.**
  - Additional 200K to 250K gates
- **Based on FLIX "Flexible Length Instruction Xtension" Technology**

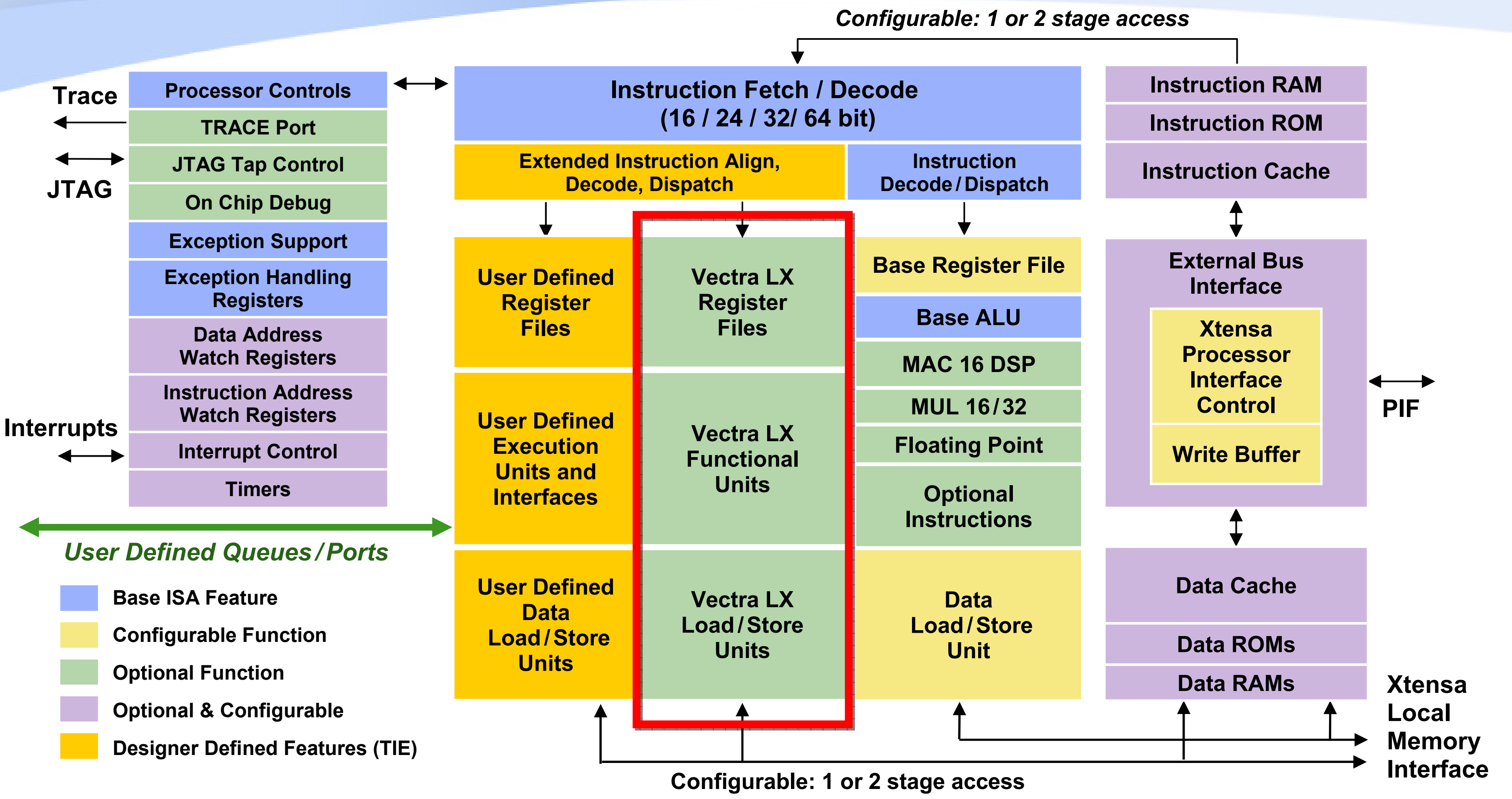
63		46	45		28	27		4	3			0
ALU & 2nd Ld/St			MAC & Select			Load/Store & Core			1	1	1	0

  - General DSP instruction set >210 instructions: ALU, MAC and Load/Store
  - 64b instructions modelessly intermix with 16b/24b Xtensa core instructions
  - 3 vector operations per instruction bundle. Dual load store units.
- **User Extensible: Designer-defined instruction extensions can be added**
  - Application specific acceleration without raising the frequency

– or –

  - Meet performance target at lower frequency
- **Very Low Power**
  - Fine grain clock gating of every functional element
- **Complete Xtensa LX software development toolchain : functionality and optimizations augmented to match the Vectra LX option.**

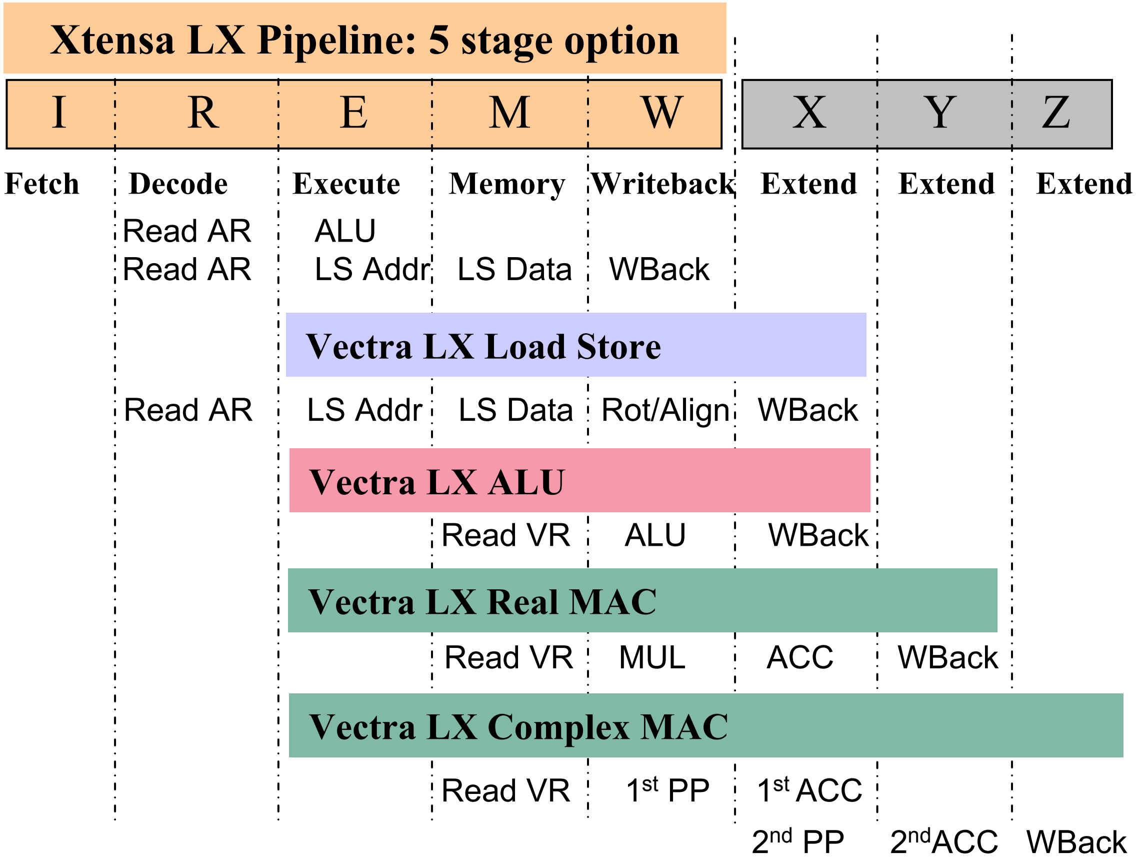
Vectra LX DSP Engine is Integrated within Xtensa LX Configurable Processor Core


Configurable: 1 or 2 stage access
Trace
Processor Controls
TRACE Port
JTAG Tap Control
JTAG
On Chip Debug
Exception Support
Exception Handling Registers
Data Address Watch Registers
Instruction Address Watch Registers
Interrupts
Interrupt Control
Timers
Instruction Fetch / Decode (16 / 24 / 32/ 64 bit)
Extended Instruction Align, Decode, Dispatch
Instruction Decode / Dispatch
User Defined Register Files
Vectra LX Register Files
Base Register File
Base ALU
MAC 16 DSP
MUL 16/32
Floating Point
User Defined Execution Units and Interfaces
Vectra LX Functional Units
Optional Instructions
User Defined Data Load/Store Units
Vectra LX Load/Store Units
Data Load/Store Unit
Instruction RAM
Instruction ROM
Instruction Cache
External Bus Interface
Xtensa Processor Interface Control
Write Buffer
PIF
Data Cache
Data ROMs
Data RAMs
Xtensa Local Memory Interface
Configurable: 1 or 2 stage access
User Defined Queues/Ports

🟦	Base ISA Feature
🟨	Configurable Function
🟩	Optional Function
🟪	Optional & Configurable
🟧	Designer Defined Features (TIE)

- Single set of software development tools supporting Xtensa LX + Vectra LX
- Maintain Xtensa programming model
- Xtensa ABI conformance

Vectra LX Pipeline

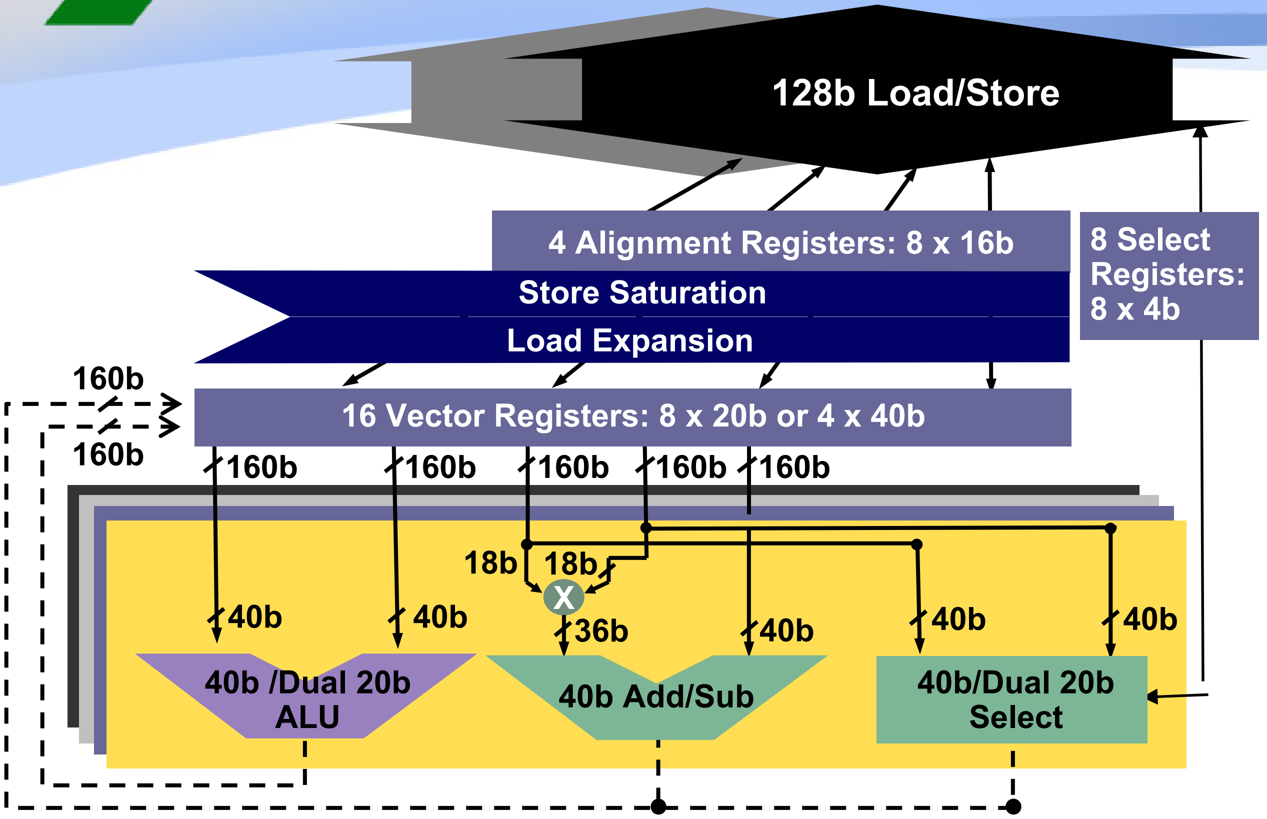


Xtensa LX Pipeline: 5 stage option

	I	R	E	M	W	X	Y	Z
	Fetch	Decode	Execute	Memory	Writeback	Extend	Extend	Extend
		Read AR	ALU					
		Read AR	LS Addr	LS Data	WBack			
Vectra LX Load Store		Read AR	LS Addr	LS Data	Rot/Align	WBack		
Vectra LX ALU				Read VR	ALU	WBack		
Vectra LX Real MAC				Read VR	MUL	ACC	WBack	
Vectra LX Complex MAC				Read VR	1st PP	1st ACC		
						2nd PP	2ndACC	WBack

- DSP computation placed in late pipe stages to eliminate load delay slot.
- Additional stages for extra computation: Multiply/Add and Complex Multiply/Add
- Can also be combined with 7-stage option of Xtensa LX for longer memory access

Parallel Operations



- 2x128b Data Transfer. 12GB/sec @370MHz. Two independent address calculations
- Four alignment registers supporting unaligned vectors
- Loads sign extend to 20b/40b. Stores saturate

Issue 3 vector operations per cycle.

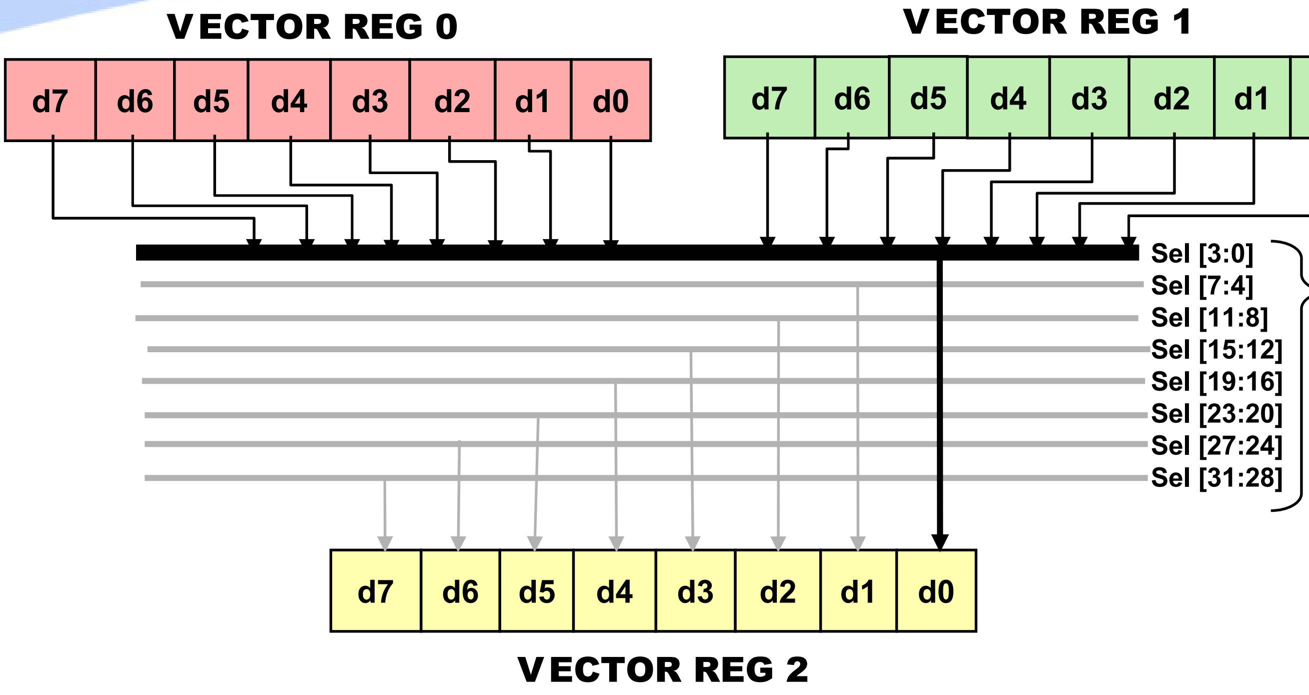
- 128b Load/Store or Core Instruction
- MAC (SIMD 4x40b) or Select
- ALU (SIMD 4x40b or 8x20b) or Second 128b Load/Store

Large vector register file:

16x160bits (320Bytes).

6 read, 3 write ports. 66 GB/sec.

Flexible Data Re-arrangement



VECTOR REG 0

d7	d6	d5	d4	d3	d2	d1	d0

VECTOR REG 1

d7	d6	d5	d4	d3	d2	d1	d0

Sel [3:0]
Sel [7:4]
Sel [11:8]
Sel [15:12]
Sel [19:16]
Sel [23:20]
Sel [27:24]
Sel [31:28]

d7	d6	d5	d4	d3	d2	d1	d0

VECTOR REG 2

8x4b control value from :

1) immediate operand.
6 pre-defined selection.

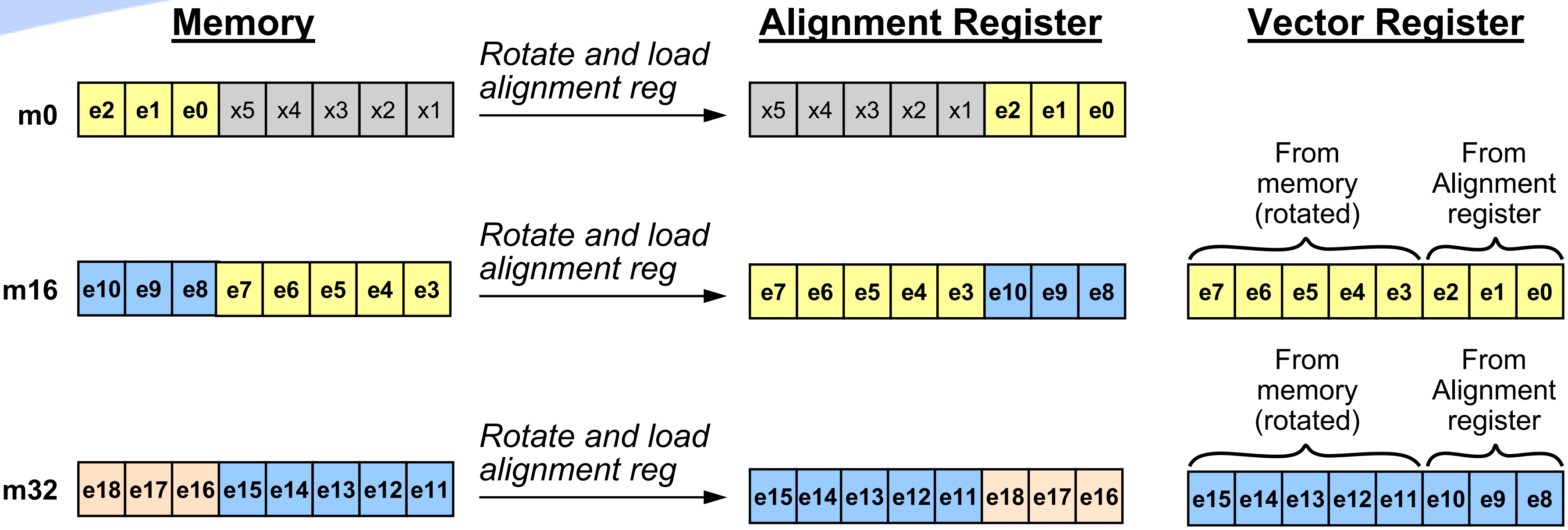
0	SHIFT RIGHT BY 1
1	SHIFT RIGHT BY 2
2	SHIFT LEFT BY 1
3	SHIFT LEFT BY 2
4	EVEN / ODD INTERLEAVE
5	MIDDLE EIGHT

2) Dedicated register file :
8 entries, 32b.

Powerful SELECT instructions to define general elements transfer

- Each target element can be individually selected from the source vectors
- Easy to implement replication, rotation, shift, interleaving…

Unaligned Vector Load/Store



	Memory		Alignment Register	Vector Register
m0	e2 e1 e0 x5 x4 x3 x2 x1	*Rotate and load alignment reg* →	x5 x4 x3 x2 x1 e2 e1 e0	
m16	e10 e9 e8 e7 e6 e5 e4 e3	*Rotate and load alignment reg* →	e7 e6 e5 e4 e3 e10 e9 e8	e7 e6 e5 e4 e3 (From memory (rotated)) e2 e1 e0 (From Alignment register)
m32	e18 e17 e16 e15 e14 e13 e12 e11	*Rotate and load alignment reg* →	e15 e14 e13 e12 e11 e18 e17 e16	e15 e14 e13 e12 e11 (From memory (rotated)) e10 e9 e8 (From Alignment register)

- **Both Aligned and Unaligned Vector Load/Store**
  - Data rotated based on least significant address bits.
  - 4 x128b alignment registers. Initialized to hold partial vector.
  - Subsequent loads merge load data with contents of alignment register, and prime alignment register for next load.
- **Throughput: 1 unaligned Load/Store per cycle**

Xtensa LX with Vectra LX Engine: FFT Code Example

256-Point Radix-4 Complex FFT

	ALU	MAC & SELECT	Core & Load/Store
			loopnez a6, 400420c)
	add20 v3, v8, v2;	rmulr18 v1, v11, v12;	lvs16.xu v0, a14, a8
	pack40 v2,v9, v10;	sel v6, v6, v5, s2;	lvs16.xu v4, a14, a8
	nop;	imulr18 v7, v11, v12;	lvs16.xu v5, a14, a8
	add20 v8, v0, v5;	nop;	lvs16.xu v9, a14, a8
	add20 v2, v4, v9;	rmulr18 v10, v6, v13;	svs16.xu v2, a7, a8
	sub20 v11, v8, v2;	nop;	lvs16.i v12, a4, -16
	sub20 v9, v4, v9;	rmulr18 v4, v11, v12;	svs16.xu v3, a7, a8
	pack40 v2,v7, v1;	sel v1, v9, v9, s0;	svs16.xu v14, a7, a8
	sub20 v7, v0, v5;	imulr18 v0, v11, v12;	nop;
	sub20 v5, v7, v1;	nop;	lvs16.i v12, a4, -32
	sub20 v6, v7, v1;	imulr18 v9, v6, v13;	svs16.xu v3, a7, a8
	pack40 v14, v0, v4;	sel v11, v6, v5, s1;	nop;
Utilization →	*92%*	*100%*	*83%*

Vectra LX Engine Performance

256pt FFT (Radix-4)

Simple RISC Task Engine	Minimal Configuration Xtensa V processor using software multiply	155,389 cycles
Scalar Performance	Base Xtensa V processor with MUL32 option	23,633 cycles
SIMD Performance	Xtensa V processor with 4-way SIMD Vectra DSP Engine	3,055 cycles
FLIX Performance	Xtensa LX with Vectra LX option	994 cycles

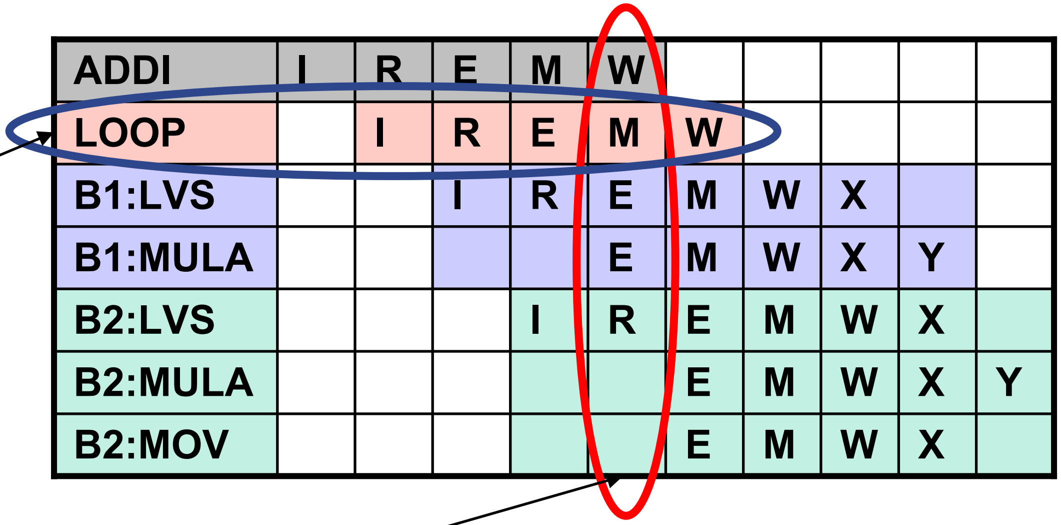
Complete Xtensa LX Software Tools: Pipeline Accurate ISS for Vectra LX

Code Stream

```
.....
addi.n     a4, a4, -1
loopnez    a4, .Lwinddown
{lvs16.iu v4, a2, 16; mula18.0 v2, v0, v1; nop}
{lvs16.iu v1, a3, 16; mula18.1 v3, v0, v1; mov160  v0, v4}
.Lwinddown:
.....
```

Traditional Instruction Set Simulator:

- *Model the state of an instruction*
- *Estimate cycle*
- *Instruction order approximates event order*



ADDI	I	R	E	M	W					
LOOP		I	R	E	M	W				
B1:LVS			I	R	E	M	W	X		
B1:MULA					E	M	W	X	Y	
B2:LVS				I	R	E	M	W	X	
B2:MULA						E	M	W	X	Y
B2:MOV						E	M	W	X	

Xtensa LX Pipeline Accurate Instruction Set Simulator:

- *Model the state of the pipeline*
- *Simulate cycle & event order*

Complete Xtensa LX Software Tools: Vector Compiler for Vectra LX

C code

```
short a[128], b[128], c[128];
void min_mul()
{int result=0;
 int i;
 for (i=0; i<128; i++) {
   c[i] = MIN(a[i]*b[i], 16384);}}
```

Compiled Loop

```
loopgtz a11,.L
{pack40 v6,v3,v5; mul18.0 v4,v0,v1; lvs16.iu v0,a8,16}
{min40  v3,v2,v7; nop;              lvs16.iu v1,a9,16}
{min40  v5,v4,v7; mul18.1 v2,v0,v1; svs16.iu v6,a10,16}
.L:
```

Compiler vectorize a, b, c -> A, B, C. 8 elements per vector

Software pipelined loop overlaps operations from different iterations: N-2, N-1, N

(N-2) PACK even/odd from MIN	**(N-1)** AxB even	**(N)** Load A
(N-1) MIN odd		**(N)** Load B
(N-1) MIN even	**(N)** AXB odd	**(N-2)** Store C

Complete Xtensa LX Software Tools: Vector Data Type for Vectra LX

C code

```
short a[128], b[128], c[128];
void min_mul()
{int result=0;
 int i;
 for (i=0; i<128; i++) {
   c[i] = MIN(a[i]*b[i], 16384);}}
```

Vector Data Type

```
vec8x16 a[16], b[16], c[16];
void min_mul()
{ vec4x40 V0, V1;
 vec4x40 V_con = 16384;
 int i;
 WUR_VSAR(0);
 for(i = 0; i <16; i++)
 {V0 = MUL18_0(a[i], b[i]);
   V1 = MUL18_1(a[i], b[i]);
   V0 = MIN40(V0, V_con);
   V1 = MIN40(V1, V_con);
   c[i] = PACK40(V1, V0); }}
```

C/C++ compiler and debugger support vector types just like other C types.

Algorithm expressed using vector data types. Type conversions implicit.

Complete Xtensa LX Software Tools: Scheduling Assembler for Vectra LX

Assembly Input

```
loopgtz a11,.L
     lvs16.iu v0,a8,16
     mul18.0 v4,v0,v1
     pack40 v6,v3,v5
     lvs16.iu v1,a9,16
     min40  v3,v2,v7
     svs16.iu v6,a10,16
     mul18.1 v2,v0,v1
     min40  v5,v4,v7
.L
```

Assembled Output

```
loopgtz a11,.L
{pack40 v6,v3,v5; mul18.0 v4,v0,v1; lvs16.iu v0,a8,16}
{min40  v3,v2,v7; nop;              lvs16.iu v1,a9,16}
{min40  v5,v4,v7; mul18.1 v2,v0,v1; svs16.iu v6,a10,16}
.L:
```

- **Analyze data dependence from straight line assembly code stream**
- **Model Vectra LX resource**
- **Automatic schedule instructions into Vectra LX bundles.**

Creating Your Own DSP Extensions: Add to, or Modify Vectra LX, using TIE

Vectra LX was designed using TIE *(Tensilica Instruction Extensions)*

Define DSP register file 16 entries 160b wide

```
regfile vec 160 16 v
```

Define DSP instruction: SIMD Multiply/Add of even elements

```
operation MULA18.0 {inout vec acc, in vec m0, in vec m1} {} {
   wire [39:0] sum0 = TIEmac(m0[ 17:  0], m1[ 17:  0], acc[ 40:  0]);
   wire [39:0] sum1 = TIEmac(m0[ 57: 40], m1[ 57: 40], acc[ 79: 40]);
   wire [39:0] sum2 = TIEmac(m0[ 97: 80], m1[ 97: 80], acc[119: 80]);
   wire [39:0] sum3 = TIEmac(m0[137:120], m1[137:120], acc[159:120]);
   assign accum = {sum3, sum2, sum1, sum0}; }
```

Specify pipeline schedule for Multiply/ Add instruction

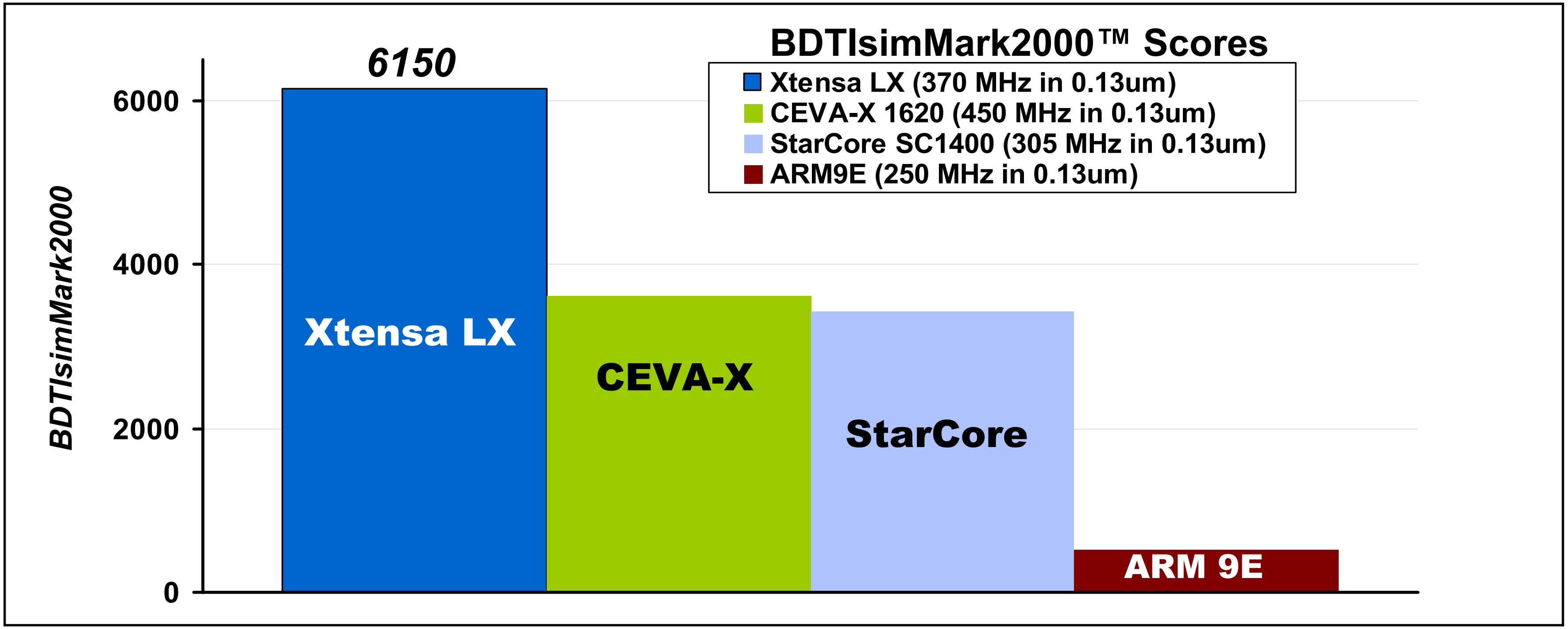
```
schedule mula {MULA18.0} {
   use m0  4;  use m1  4;  use acc 5;  def acc 5; }
```

- Specification: Verilog description of semantics of custom instructions
  - Optionally add custom register files or states
  - Optionally use pre-defined register files from Xtensa LX, Vectra LX
- TIE Compiler automatically extends Xtensa LX processor
  - Updated Synthesizable RTL
  - Matching SW development tools

BDTI Benchmark: Xtensa LX Processor with Vectra LX engine

Xtensa LX Configuration includes Vectra LX DSP Engine + 11 custom extensions

- Viterbi trellis decode (3) ,Viterbi trellis traceback (2), Bit stream unpacking (2), Multiply intensive filter (4) - add a total of 30K additional gates



The BDTIsimMark2000™ is a summary measure of DSP speed distilled from a suite of DSP benchmarks developed and independently verified by Berkeley Design Technology, Inc. For more information, see www.bdti.com

Xtensa LX: 16.6 marks/MHz, total score 6150
CEVA-X 1620: 8.0 marks/MHz, total score 3620
StarCore SC1400: 11.2 marks/MHz, total 3420
ARM9E: 2.1 marks/MHz, total score: 520

Xtensa LX configuration as tested by BDTI: 248,600 “gates” (equivalent NAND2X cell area) at post-synthesis; 4.4mm^2 actual layout area; 3D extracted final layout timing under worst case conditions: 369 MHz; leakage power 5.5mW + dynamic power 0.53mW/MHz

Summary: Xtensa LX Processor Optimal DSP Solutions for Every Application

Vectra LX: New programmable DSP engine

- *Integrated with configurable Xtensa LX processor core*
- *Competitive performance compared to other general DSPs*

Application specific extensions

- *Target specific algorithms*
- *Augment performance. Augment efficiency.*
- *Delivers highest-ever benchmark performance*

Tensilica Instruction Extension (TIE) methodology

- *Easy for Tensilica licensees to create application-specific DSP extensions*
- *Easy for Tensilica licensees to extend Vectra LX engine*
- *Automatically produce updated processor RTL + matching software tools.*
- *Achieve the right performance + power efficiency for YOUR application*