

# Configurable Processor Cores

Ingo Sander  
ingo@imit.kth.se



## Motivation for Configurable Processor Cores



- Observations
  - Time-to-market is critical
  - Development time for software is much smaller than for hardware
  - Hardware can be customized and has much better performance than software solution

September 30, 2004

2B1445 Embedded Systems

2

## Why Configurable Processor Cores?



- Idea
  - Combine the advantages of hardware and software in form of a customizable processor to achieve
    - Clearly shorter Time-To-Market than hardware
    - Clearly better performance than software
  - Provide a processor platform with a basic architecture that can be extended
    - by additional optimized units (MAC, Floating-Point Unit)
    - Own instructions together with own customized hardware can be defined for the processor

September 30, 2004

2B1445 Embedded Systems

3

## Example for a configurable processor: Xtensa (Tensilica)



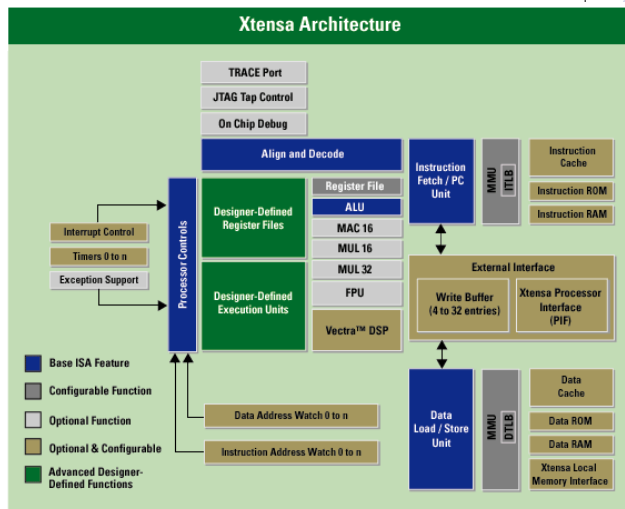
- The Xtensa processor core
  - Targets system-on-chip applications
  - Is configurable, extensible and synthesizable
  - Has
    - Base Instruction Set Architecture
    - Configurable Functions (Parametrized)
    - Optional Functions
    - Designer Defined Functions and Registers (For Acceleration of Specific Algorithms)

September 30, 2004

2B1445 Embedded Systems

4

# Xtensa Processor Core



# Basic Xtensa Core

- 32-bit architecture
- Base configuration:
  - 32-bit ALU
  - Up to 64 general purpose registers
  - 6 special purpose registers
  - 80 base instructions
  - Improved 16- and 24-bit RISC instruction encoding

# Optional Architecture

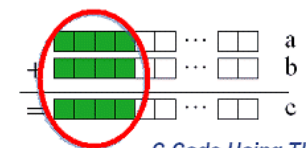
- Execution Units
  - Multipliers, 16 and 32 bits
  - MAC-Unit, Floating-Point Unit
- Interface Options
- Memory Subsystem Options
  - Memory Management Options
  - Local Data and Instruction Caches
  - Separate RAM, ROM Areas for Data and Instruction

# Tensilica Extension Language

- The Tensilica extension language is used to describe new instructions, registers and execution units that are then automatically added to the Xtensa

### Original C Code

```
short *a, *b, *c;
for (int i=0; i<n; i++)
    c[i] = a[i] + b[i]
```



### Complete TIE Code

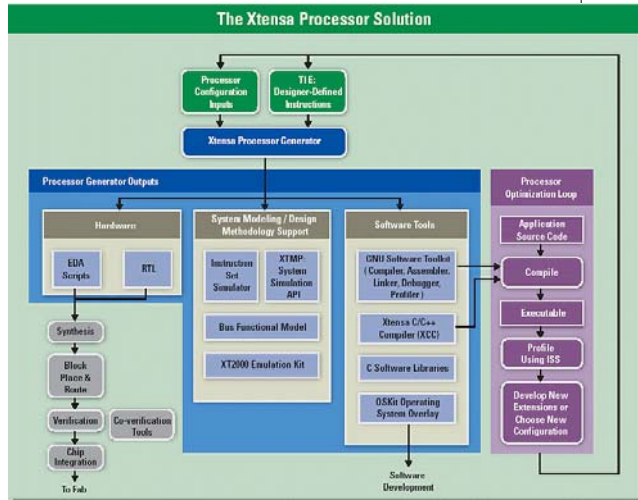
```
regfile vec 64 16 v;
operation add16x4(out vec c, in vec a, in vec b) {
    assign c = {a[63:48]+b[63:48], a[47:32]+b[47:32],
                a[31:16]+b[31:16], a[15:0]+b[15:0]}; }

```

### C Code Using TIE

```
vec *a, *b, *c;
for (int i=0; i<n; i+=4)
    c[i] = add16x4(b[i], a[i])
```

# Xtensa Processor Design Process



September 30, 2004

2B1445 Embedded Systems

9

# Design Flow



1. Choose basic Xtensa processor
2. Specify algorithm in C
3. Compile to Target Processor
4. Profile and check, if design constraints are met
5. If constraints are met, everything is fine, otherwise
6. Choose optional functions (e.g. Multiplier) or design new instructions for the critical part => improved architecture
7. Adjust your code for the new architecture
8. Go back to 3.

September 30, 2004

2B1445 Embedded Systems

10

# Summary



- The Xtensa concept provides
  - Not only a configurable architecture
  - But also a design methodology
- The idea is to take the best of both the hardware and the software world in order to
  - Have good performance
  - Short Time-to-Market
- Xtensa processors can be used as parts of a system-on-chip architecture
- Other extendable cores exist like the NIOS II from Altera

September 30, 2004

2B1445 Embedded Systems

11