Introducing AndesCore™ and Development Tools to Ease Domain-Specific Acceleration

Tommy Lin
Business Development Division
Andes Technology Corp.
tommylin@andestech.com
Andes RISC-V Product Overview

**AndeStar™** Architecture V5/V5e

- **AndesCore™** Processors: Highly optimized design to deliver leading PPA
- **AndeShape™** Platforms: Handy peripheral IPs to speed up SoC construction
- **AndeSight™** Tools: Professional IDE with high code quality
- **AndeSoft™** Stacks: Extensive SW stacks from bare metal, RTOS to Linux

Best extensions to RISC-V
AndesCore™ RISC-V Family

**Leading Performance**
- **A25MPa**
  - V5, 32b, 1~4 Cores
  - L2 Cache Coherence
  - DSP, MMU, FPU, ACE...
- **AX25MPa**
  - V5, 64b, 1~4 Cores
  - L2 Cache Coherence
  - DSP, MMU, FPU, ACE...

**Linux and FPU/DSP**
- **A25**
  - V5, 32b, 5-stage, >1.2GHz
  - MMU/PMP, DSP, FPU, ACE...
- **AX25**
  - V5, 64b, 5-stage, >1.2GHz
  - MMU/PMP, DSP, FPU, ACE...

**Fast/Compact, FPU/DSP**
- **D25F**: +DSP
  - V5, 32b, 5-stage, >1.2GHz
  - PMP, FPU, ACE...
- **N25F**
  - V5, 32b, 5-stage, >1.2GHz
  - PMP, FPU, ACE...
- **AX25MP**
  - V5, 64b, 1~4 Cores
  - L2 Cache Coherence
  - DSP, MMU, FPU, ACE...
- **NX25F**
  - V5, 64b, 5-stage, >1.2GHz
  - PMP, FPU, ACE...

**Slim and Efficient**
- **N22**
  - V5/V5e, 32b, 2-stage
  - 800MHz, 16/32 GPR

---
a. A25*MP: available Q1/2019
b. 28HPC+ RVT, SS, 0.81V, 0°C, with I/O constraints.
V5 ISA Extensions

- Andes innovations incorporating RISC-V technology
- Baseline extensions
  - Instructions to speed up memory accesses
    - GP-implied load/store instructions with larger immediate
    - Calculate effective address according to data types
  - Instructions to speed up branches
    - Compare (a small) constant and branch
    - Test a bit and branch
  - Zero and sign extension
  - CoDense™: Code size compression on top of RISC-V C-extension
- Superset of fundamental RISC-V ISA
  - Keep compatibility
- Andes Custom Extension™ (ACE) frameworks for DSA (Domain-Specific Architecture)
  - Powerful tool to automate housekeeping tasks
  - No need to have CPU background
Development Environment

- AndeSight™ Feature-rich IDE
- AndeShape™ Development Boards
  - Full-featured ADP-XC7K
  - Corvette-F1 (Arduino-compatible)
- Debugging Hardware
  - AICE-MINI+, AICE-MICRO
- AndeSoft™ Software Stack
  - Bare metal demo projects
  - FreeRTOS version 10
- Partner Supports

And more…

Corvette-F1

ADP-XC7

LAUTERBACH
DEVELOPMENT TOOLS

SEGGER

expresslogic

imperas
ultraSOC

Driving Innovations™
AndesCore™ Advantages

► Better Verilog RTL code
  • Human readable, CAD-tool friendly, and configurable through GUI-tool by customers
► Rich extended features for embedded applications
  • Misaligned accesses, StackSafe™, CoDense™
► Higher **performance** and smaller **code size**

<table>
<thead>
<tr>
<th></th>
<th>XYZ</th>
<th>N22 *</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ISA</strong></td>
<td>RV-IMAC</td>
<td>RV-IMAC+V5</td>
</tr>
<tr>
<td><strong>CoreMark/MHz</strong></td>
<td>3.1</td>
<td>3.95 (+27%)</td>
</tr>
<tr>
<td><strong>DMIPS/MHz (no-inline)</strong></td>
<td>1.38</td>
<td>1.80 (+30%)</td>
</tr>
<tr>
<td><strong>CSiBE Code Size (KB)</strong></td>
<td>1,340</td>
<td>1,185 (-12%)</td>
</tr>
</tbody>
</table>

* N22 has the similar configurations to XYZ

► **2-wire debug support to save chip cost**
► Development tools with handy project management and debugging features
Useful hardwired functions can become instructions
- Multiply, multiply-add, divide: hardwired functions for 8051
- MotionComp, IDCT, VLD: parallel instructions in video processors

→ Enable programmable acceleration

Key to programmable acceleration
- Find your hot spots or critical functions
- Partition the tasks to get the desired flexibility and efficiency
Design Flow for ACE Acceleration

1. **Start**
2. **Profile** application SW to identify **time-critical** code
3. Define ACE instructions with cycle estimation
4. Cycles met?
   - No
   - Yes
5. **Implement** ACE RTL
6. **Evaluate** PPA
7. Requirements met?
   - Yes
   - No
8. Room to improve?
   - Yes
   - No
9. **Done**
Andes Custom Extension™ Framework

**COPILOT**

Custom-OPtimized Instruction development Tools

- Extended Tools
- Extended ISS
- Extended RTL

**Cross-checking Env.**

- Verilog user.v
- Concise RTL
- Semantics, operands, test-case spec
- Script user.ace

**Test Case Generator**

- Extended RTL
- Extended ISS

**Extensible Baseline Components**

- Compiler
- Asm/Disasm
- Debugger
- CPU ISS (near-cycle accurate)
- CPU RTL

**Executable or library**

**Source file**
# Highlights of ACE Features

<table>
<thead>
<tr>
<th>Items</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instructions</strong></td>
<td></td>
</tr>
<tr>
<td>scalar</td>
<td>single-cycle or multi-cycle</td>
</tr>
<tr>
<td>vector</td>
<td>for loop or do-while loop</td>
</tr>
<tr>
<td><strong>background option</strong></td>
<td>retire immediately and continue execution in the background (applicable to scalar and vector)</td>
</tr>
<tr>
<td><strong>Operands</strong></td>
<td></td>
</tr>
<tr>
<td>standard</td>
<td>immediate, GPR, baseline memory (thru CPU)</td>
</tr>
<tr>
<td>custom</td>
<td>- ACR (ACE Register), ACM (ACE Memory)</td>
</tr>
<tr>
<td></td>
<td>- With arbitrary width and number</td>
</tr>
<tr>
<td></td>
<td>- ACR operands can be “implied” to save opcode</td>
</tr>
<tr>
<td><strong>Auto Generation by COPILOT</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Opcode assignment: automatic by default</td>
</tr>
<tr>
<td></td>
<td>- All required tools and simulator (C or SystemC)</td>
</tr>
<tr>
<td></td>
<td>- <strong>RTL code</strong> for instruction decoding, operand mapping, dependence checking, input accesses, and output updates</td>
</tr>
<tr>
<td></td>
<td>- Waveform control file</td>
</tr>
</tbody>
</table>

**Fast turnaround time!**
Madd32: A Simple Custom Instruction

Benefits of ACE

- Allow users to focus on instruction functionality
- Offload housekeeping work
  - opcode selection
  - instruction decoding
  - operand mapping
  - input operand accesses
  - output operand updates
  - dependence checking

Adding instructions is just like ASIC design

Auto-generate both RTL and simulator code
Madd32: Easy Migration on Software

```c
uint fir32(uint *C, uint *X, uint n) {
    uint rslt= 0;
    for(int i=0; i<n; ++i)
        rslt+= (C[i] & 0xffff)*(X[i] & 0xffff); //lower 16 bits
        + (C[i] >> 16) *(X[i] >> 16); //upper 16 bits
    return rslt;
}
```

# include "ace_user.h"  //prototypes for auto-generated intrinsic

```c
12
Pure C Code

Madd32: Easy Migration on Software

Pure C: 8 cycles
With ACE: 1 cycle
Speedup: 8x
```

With ACE

```c
rslt= ace_madd32(rslt,X[i],C[i]); //invoke intrinsic
```

```c
op = {io acc, in dat, in coef},
```
Summary

► **AndesCore™: Plus Andes Extensions**
  - Show the **real flexibility** and also **keep compatibility**
  - Provide better **power, performance, area and code density**

► **Benefits of ACE**
  - Facilitates easy tradeoffs between **efficiency** and **flexibility**
  - Focus on instruction functionality, not CPU pipeline
  - **Simplify** and **ease** instruction design for ASIC engineers
  - **COPILLOT tools**
    - Handles housekeeping, interface and verification tasks
    - Auto-generates all needed tools, simulator and RTL
RISC-V Activities at Embedded World

Exhibiting
Thanks for joining the RISC-V Foundation at Hall 3A, Booth # 3A-536 with member companies Andes Technology, CloudBEAR, GreenWaves Technologies, Imperas Software, SiFive, Syntacore and UltraSoC.

Presentations
The booth will feature sessions with RISC-V Foundation members throughout the show. The conference program also features a variety of RISC-V talks. Learn more: http://bit.ly/RISCVew19.

Happy Hour Receptions
Join us for networking and drinks on Tuesday and Wednesday from 17:00 to 18:00 CET.

Scavenger Hunt
Visit the booths of RISC-V Foundation members and you could win a prize. Check in at the front desk to get a scavenger hunt “passport” and learn more.

Please reach out to RISC-V@racepointglobal.com with any questions.
Thank You!