Andes Technology

- Taiwan-based CPU IP company with over 2-billion Andes-Embedded SoCs shipped in diverse applications.

- Taking RISC-V to those markets with the solutions we developed in the past 12 years.

- A major contributor to RISC-V tools such as GCC, binutils, newlib, and recently LLVM and LLD.
AndeStar V5m Architecture

V5m Architecture Features

- PLIC: priority-based preemptive interrupts.
- PLIC: vectored interrupts.
- StackSafe™.
- Exception redirection to debugger.

V5m ISA

- RISC-V RV*IMC base integer instructions.
- Optional RISC-V RVA instructions.
- Andes Performance extension.
- Optional Andes DSP extension.
- Optional Andes Custom extension.
PLIC with Priority Preemption

- The base PLIC can select an interrupt based on priority, but cannot automatically prevent a lower-priority one from interrupting a higher-priority ISR (if interrupt is enabled).
- Andes extends it with the priority-based preemption so that only higher-priority ones can preempt the currently executed ISR.
Benefit of Vectored PLIC

- Some design practice calls for a much larger number of interrupts than local interrupts.
  - And it’s not easy to scale the local interrupts
- However, it is easy to extend the number of interrupt sources in PLIC.
- Vectorizing PLIC interrupt sources is more scalable and systematic than vectorizing local interrupts.
- Worked on Multicore as well.
Scheme of Vectored PLIC

- Vector table entry 0: exceptions and local interrupts except “External interrupt”.
- Vector table entry 1 and above: external interrupts from PLIC.
- The PLIC interrupt ID is transmitted directly from PLIC to a hart.
- Vector table stores ISR address instead of a “Jump” instruction. Larger address range to locate ISRs.
  - Keep RV64 table size the same as RV32, table stores lower 32-bit address. Upper 32-bit address from the mtval CSR.
Benefits of Extended PLIC

- Compared with the base PLIC, Andes extended PLIC saves
  - >30 instructions for dispatch and SW preemption overhead, or
  - >50% instructions before entering the actual ISR.
  - Quite some designs require the latency from CPU taking the interrupt to entering their ISR as short as possible.

- Ease of use for priority-based preemption:
  - Initialize priorities in PLIC and SW never needs to deal with it again.
StackSafe

- Stack overflow and underflow protection.
- Monitoring the SP register value for protection and recording:
  - Protection mode: Generate an exception when overflow or underflow of a process stack is detected.
  - Recording mode: Recording max. stack usage for the required stack size in final SW.
Andes ISA Extensions

- Use fewer instructions to access data in memory.
  - GP-implied load/store instructions with larger immediate range.
  - Calculate effective address based on data type.
- Compare an operand with a small constant and branch.
- Handy instructions for zero/sign-extension.
- CoDense™: Code size compression instructions
Andes DSP ISA Extension

Features:

- > 130 instructions; Use only GPRs
- Saturation and rounding
- Fractional Q31, Q15, and Q7 data types
- Integer 32b, 16b, and 8b data types
- 16-bit and 8-bit SIMD instructions
  - 16-bit: +, -, x, min, max, abs, clip, compare, <<, >>, signed, unsigned
  - 8-bit: +, -, min, max, abs, unpack, compare, signed, unsigned
- 64-bit signed/unsigned addition & subtraction
- 64-bit signed/unsigned multiplication & addition
  - E.g., 64 += 16x16 + 16x16
- Zero-Overhead Loop

SW support:

- Intrinsic functions for using instructions in C.
- Auto-generation of SIMD instructions based on data type/vector data type.
# DSP ISA Performance

## Helix MP3 decoder

<table>
<thead>
<tr>
<th>GCC Compiler</th>
<th>Decode (MCPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compile with baseline ISA</td>
<td>22.64</td>
</tr>
<tr>
<td>Compile with baseline+DSP ISA</td>
<td>11.40</td>
</tr>
<tr>
<td>Cycle reduction %</td>
<td>50%</td>
</tr>
</tbody>
</table>

## G.729 codec

<table>
<thead>
<tr>
<th></th>
<th>Encode (MCPS)</th>
<th>Decode (MCPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiler alone</td>
<td>95.45</td>
<td>26.83</td>
</tr>
<tr>
<td>Intrinsic + Compiler</td>
<td>26.31</td>
<td>6.02</td>
</tr>
<tr>
<td>Cycle reduction %</td>
<td>72%</td>
<td>78%</td>
</tr>
</tbody>
</table>

* MCPS: Millions of Cycles Per Second
Andes Custom Extension (ACE)

- Increase performance & reduce energy consumption through custom instructions.

ACE Framework

- Facilitate instruction design and implementation through ACE script + C + Verilog.
- Allow custom registers/memories/ports as operands.
- Generate extended toolchain, simulator, RTL, verification environment and tests automatically.
- Help designers to focus on the most important part in the custom instruction design process.
Thank You!

www.andestech.com