Experiences from Andes Technology
About us

❖ A company in Taiwan since 2005 (went public in 2017)
❖ A pure-play IP vendor with 140+ licenses
  ▪ >2.5B Andes-Embedded™ SoCs
❖ Diversified applications/needs
  ▪ Bare metal, RTOS, Linux
  ▪ Control, DSP/FP, security, and more
❖ AndeStar™ V5: RISC-V with Andes Extensions
Outline

❖ Performance Counter (Perf)
❖ ELF Attribute
❖ **Fences after Page Table Entry Update**
❖ [RFC 0/2] Proposal
  ▪ **Infrastructure for Vendor-Specific Codes**
  ▪ Non-Coherent Agent
❖ Address Space Identifier
Outline

- Performance Counter (Perf)
- ELF Attribute
- Fences after Page Table Entry Update
- [RFC 0/2] Proposal
  - Infrastructure for Vendor-Specific Codes
  - Non-Coherent Agent
- Address Space Identifier
Perf: What we have now

❖ Preliminary Perf Support

❖ Counting (perf stat) is OK, but
❖ Sampling (perf record) is **not**
perf record -c 1000 /bin/ls

set $mcycle = MAX - 1000

start couter;
/bin/ls

stop couter;
record data

interrupt due to counter overflow
Limitation 1: Set counter (For single S-mode)

We **cannot** write to counters without an SBI call.

```
set $mcycle = MAX - 1000
start counter; /bin/ls
stop counter; record data
```

Possible Solution | Andes Solution
--- | ---
Add a SBI call | New register: `mcounterwen`
Limitation 2: Pause/Resume

```
set $mcycle = MAX - 1000
start couter;
/bin/ls
```

We cannot select privileged mode(s) to be counted.

Possible Solution | Andes Solution
--- | ---
(null) | New register: `scountermask[u|s|m]`
Limitation 3: Pause/Resume

We cannot pause/resume counters.

Possible Solution | Andes Solution
------------------|-------------------
(null)            | Some hack with `scountermask[u|s|m]`
Limitation 4: Interrupt

We have no interrupts from counter overflow.

Possible Solution

<table>
<thead>
<tr>
<th>(null)</th>
</tr>
</thead>
</table>

Andes Solution

A local interrupt, with new registers: scounterinen, scounterinov
## Perf: Summary

<table>
<thead>
<tr>
<th>Limitation</th>
<th>Status Quo</th>
<th>Possible Solution now</th>
<th>Andes Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation 1</td>
<td>Cannot write counter in S</td>
<td>SBI</td>
<td>mcounterwen</td>
</tr>
<tr>
<td>Limitation 2,3</td>
<td>Cannot pause/resume/mode selection</td>
<td>(null)</td>
<td>scountermask [u</td>
</tr>
<tr>
<td>Limitation 4</td>
<td>No interrupts for counter overflow</td>
<td>(null)</td>
<td>scounterinen, scounterinov</td>
</tr>
</tbody>
</table>
Outline

❖ Performance Counter (Perf)
❖ ELF Attribute
❖ Fences after Page Table Entry Update
❖ [RFC 0/2] Proposal
  ▪ Infrastructure for Vendor-Specific Codes
  ▪ Non-Coherent Agent
❖ Address Space Identifier
ELF Attribute: The Need

- # RISC-V Platforms increases
- # RISC-V ELF objects increases

- Disassembler: "Can I interpret the ELF correctly?"
- Linker: "Can I link the ELF objects?"
- Loader: "Can I load the ELF objects and make them run?"
  - Linux: target executable and dynamic linker
  - libc: libraries
ELF Attribute: Two Scenarios

**Scenario 1:**
- **Executable:** rv64imafdc
- **Loader:** rv32imafdc
- **Platform:** rv64imafdc

**Scenario 2:**
- **Executable:** rv64imafdcxaaa
- **Library:** rv64imafdcxbbb
- **Loader:** rv64imafdcxaaa
- **Platform:** rv64imafdcxaaa
Kito raises this proposal in March
  - Add a PT_PROC section
  - Full ISA string, privileged spec version, etc.

Discussions
  - Initial thread
  - PR for ELF PSABI Document

Components
  - Binutils & gcc: nearly done
  - Linux & glibc: need feedback
ELF Attribute: Implementation

DTS:
rv64i2p0m2p0a2p0f2p0d2p0c2p0xaaa0p0xbbb0p1
1.11

Kernel

Auxiliary Vector:
rv64i2p0m2p0a2p0f2p0d2p0c2p0xaaa0p0xbbb0p1
1.11

ld.so

Target ELF

Required Libraries

check ELF

pass attr
ELF Attribute: Any Thoughts?

❖ Does this "DTS-kernel-AUX-libc" mechanism seem good?
  ▪ Need new AT_XXX auxiliary vectors
  ▪ Should we use existing "riscv,isa" for the augmented ISA string? or invent one?

❖ Suggestion: adding restrictions to ISA format, easier life writing parser
  ▪ All lower cases
  ▪ Limited length for non-official extensions
  ▪ Force dictionary ordeer for non-official extensions
Outline

- Performance Counter (Perf)
- ELF Attribute
- Fences after Page Table Entry Update
- [RFC 0/2] Proposal
  - Infrastructure for Vendor-Specific Codes
  - Non-Coherent Agent
- Address Space Identifier
Infrastructures for Vendor-Specific Codes

❖ Clarifications for this patch set
  ▪ no custom instructions
  ▪ no stateful extensions
  ▪ custom CSRs

❖ Runtime probing for vendor features
❖ No custom CSRs in the kernel
❖ Extensions instead of vendors
Is SX suitable here? A CPU feature but as an extension-like?

```c
+static int __init nds_nocoh_ag_init(void) 
+
+    if(!strstr(isa_SX,"SXndsnocohag"))
+    return 0;
+    setup_maxpa();
+    custom_ext_cache_op = nds_cache_op;
+    custom_ext_dma_alloc = nds_dma_alloc;
+    custom_ext_dma_free = nds_dma_free;
+
+arch_initcall(nds_nocoh_ag_init);
```
Principle 2: No Custom CSRs in kernel

❖ How about this?

```c
+#define custom_csr_write(csr_enum,val) csr_write
#defie csr_write(csr, val) \
    unsigned long ... \ 
    asm("csrw ...") \ 
...

custom_csr_write(CCTL_REG_UCCTLBEGINADDR_NUM, start);  
custom_csr_write(CCTL_REG_UCCTLCOMMAND_NUM,  
    CCTL_L1D_VA_INVAL);
```
Principle 3: Extensions instead of vendors

❖ Any comments?

| arch/riscv/Kconfig        | 5 + |
| arch/riscv/Makefile       | 2 +- |
| arch/riscv/custom-ext/nds_nocoh_ag/Kconfig | 10 ++ |
| arch/riscv/custom-ext/nds_nocoh_ag/Makefile | 3 + |

... in arch/riscv/Kconfig:

+menu "RISC-V custom extension"
+
+source "arch/riscv/custom-ext/nds_nocoh_ag/Kconfig"
Non-Coherent Agents

❖ Requirement

▪ Lower hardware cost on embedded system

❖ DMA API

▪ Allocate a consistent memory (.alloc)
  ■ How to configure PMAs to disable cacheability at run time by software

▪ Flush parts of cache (.sync_single_for_cpu/device)
  ■ Only flush all at one time

```
struct dma_map_ops {
  ...
  void* (*alloc)(...);
  void (*sync_single_for_cpu)(...);
  void (*sync_single_for_device)(...);
  ...
}
```
Non-Coherent Agents

Allocate a consistent memory in Andes

- A pseudo uncacheable memory space which alias to the real memory space
  - Use the thirty-two bit to distinguish between them
Non-Coherent Agents

- Flush parts of cache in Andes
  - CCTL command operations
    - `mcctlbeginaddr/ucctlbeginaddr` CSRs
      - The virtual address to access the cache
    - `mcctlcommand/ucctlcommand` CSRs
      - Trigger a CCTL command operation

```c
/* CCTL command */
L1D_VA_INVAL
L1D_VA_WB
L1D_VA_WBINVAL
L1D_VA_LOCK
L1D_VA_UNLOCK
L1D_WBINVAL_ALL
L1D_WB_ALL
L1I_VA_INVAL
L1I_VA_LOCK
L1I_VA_UNLOCK

/*
 * writes the cache line back to memory
 * if the cache line state is valid and dirty.
 */
csr_write(ucctlbeginaddr, start);
csr_write(ucctlcommand, L1D_VA_WB);

/*
 * unlocks the cache line and sets its state to invalid.
 */
csr_write(ucctlbeginaddr, start);
csr_write(ucctlcommand, L1D_VA_INVAL);
```
Outline

❖ Performance Counter (Perf)
❖ ELF Attribute
❖ Fences after Page Table Entry Update
❖ [RFC 0/2] Proposal
  ▪ Infrastructure for Vendor-Specific Codes
  ▪ Non-Coherent Agent
❖ Address Space Identifier
Address Space Identifier

❖ Support ASID
  ▪ Not used the ASID field of satp

sv32

sv64

❖ TLB with ASID
  ▪ TLB doesn’t have to be invalidated on context switch
  ▪ Flush TLB when running out of ASID
Address Space Identifier

- Trigger module without ASID
  - Target debugging have to set trigger on context switch
Address Space Identifier

- Trigger module with ASID
  - Match ASID in trigger register and satp
    - Need a ASID field in trigger register or new one

```
<table>
<thead>
<tr>
<th>tdata1</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

satp(sv32)

| MODE (WARL) | ASID (WARL) | PPN (WARL) |
| 1 | 9 | 22 |

satp(sv64)

| MODE (WARL) | ASID (WARL) | PPN (WARL) |
| 4 | 16 | 44 |
Q & A

zong@andestech.com
alankao@andestech.com