C-SKY MICROSYSTEMS CO., LTD
www.C-Sky.com

How to add SMP for a New Arch

Guo Ren
https://github.com/c-sky/csky-linux

linux/arch/csky/abiv1 & abiv2

abiv1

- **16b ISA**, 32b data
- M-Core compatible
- **Mass production**

Product:

- **510**
- **610 (linux supported)**

abiv2

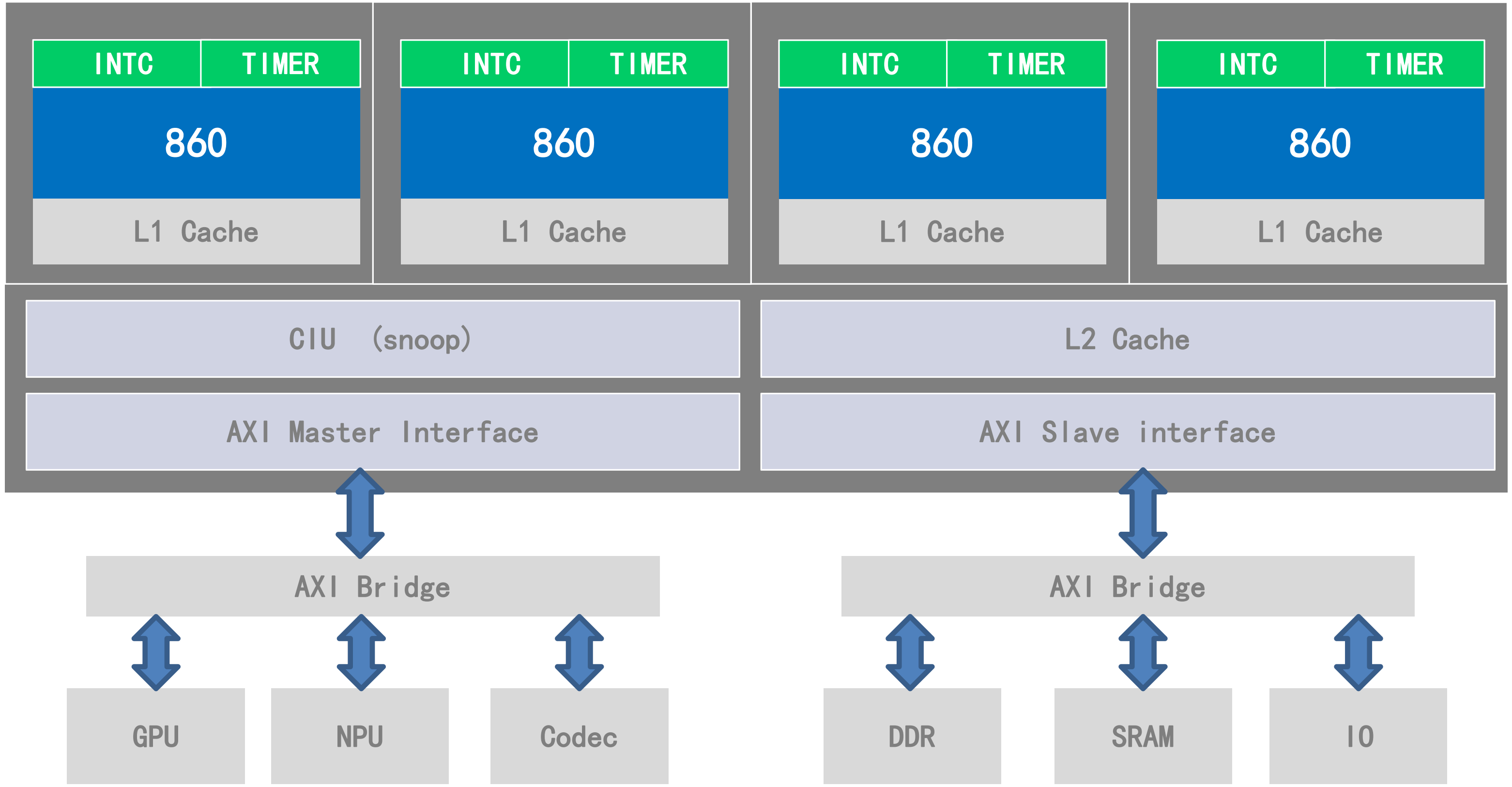
- EM_CSKY 252
- **32b/16b mixed ISA**, 32b data
- 1st independent ISA/Arch
- **Mass production**

Product:

- **807 (7-pipeline linux supported)**
- **810 (10-pipilne linux supported)**
- **860 (12-pipeline linux supported)**
- **F: FPU (V3 ISA)**
- **V: VDSP (V3 ISA SIMD-128bit)**
- **T: Trust Exe Enivornment**
- **SMP spported**

-mcpu=
807f 807fv 807ft 807fvt
810f 810fv 810ft 810fvt
860f 860fv 860ft 860fvt

860MP Architecture



linux/arch/csky/abiv1 & abiv2

abiv1

- **16b ISA**, 32b data
- M-Core compatible
- **Mass production**

Product:

- **510**
- **610 (linux supported)**

abiv2

- EM_CSKY 252
- **32b/16b mixed ISA**, 32b data
- 1st independent ISA/Arch
- **Mass production**

Product:

- **807 (7-pipeline linux supported)**
- **810 (10-pipilne linux supported)**
- **860 (12-pipeline linux supported)**
- **F: FPU (V3 ISA)**
- **V: VDSP (V3 ISA SIMD-128bit)**
- **T: Trust Exe Enivornment**
- **SMP spported**

-mcpu=

807f 807fv 807ft 807fvt
810f 810fv 810ft 810fvt
860f 860fv 860ft 860fvt

abiv3 ? (No yet born)

- **Making love with RISC-V**

Agenda

How to add SMP for a new arch

- SMP Boot
- MP-INTC Introduction (IPI mechanism)
- MP-TIMER Introduction
- Atomic implementation
- Spinlock, rwlock (test-and-set V.S ticket-queue)
- Memory Barrier
- Cache & TLB sync for SMP

SMP Qemu demo

Q & A

SMP Boot

- soc-reset-control V.S. free-run
- C-SKY RMR(Reset Manage Register)

Why use reset-controller?

- make SMP boot simpler
- make ck807/ck810/ck860 use one bootroom/bootloader binary, one work for ever

SMP Boot

Bootloader:
https://github.com/c-sky/buildroot/blob/master/board/csky/ck860_platform/gdbinit

Linux head.S:
https://github.com/c-sky/csky-linux/blob/master/arch/csky/kernel/head.S

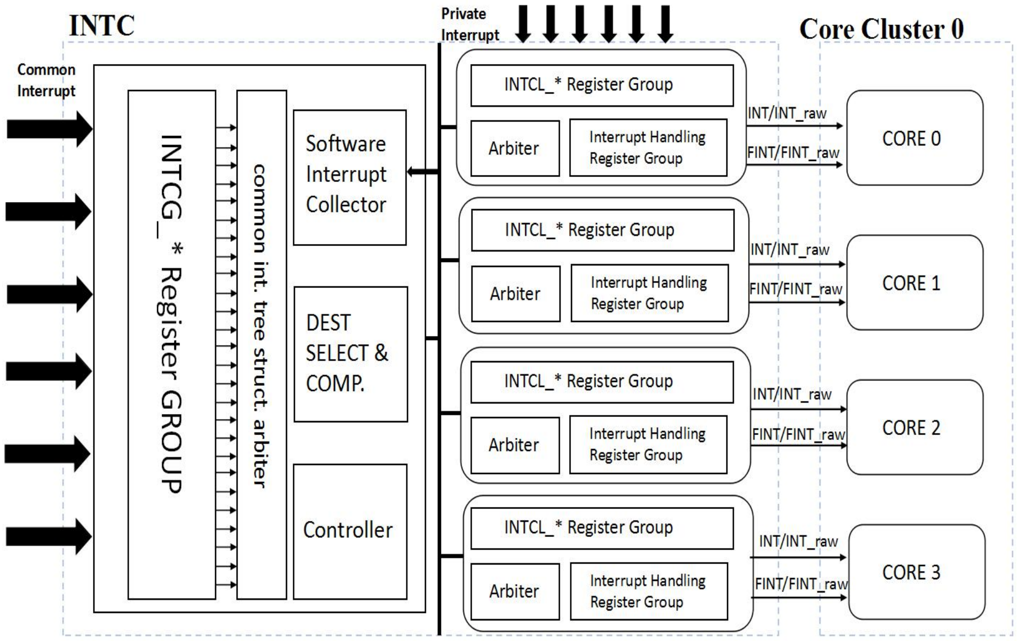
Linux csky_start -> start_kernel()
https://github.com/c-sky/csky-linux/blob/master/arch/csky/kernel/setup.c

MP-INTC

- 0 - 15 : software irq, and we use 15 as our IPI_IRQ.
- 16 - 31 : private irq, and we use 16 as the co-processor timer.
- 31 - 1024 : external irq for soc ip.

- Nice Software interface(window-reg access)
- 256 levels IRQ priority setting(8-bits width)
- Support 4 IRQ triger types config:
    1. LEVEL_HIGH
    2. LEVEL_LOW
    3. EDGE_RISING
    4. EDGE_FALLING

- (Determine Core)/(Auto select Core) for external irq
- software irq send/broadcast any cores for IPI



MP-INTC

linux/drivers/irqchip/irq-csky-mpintc.c:

IRQ handler flow:

```
static void csky_mpintc_handler(struct pt_regs *regs) {
            do {
                        handle_domain_irq(root_domain,
                                                      readl_relaxed(reg_base + INTCL_RDYIR), regs);
            } while (readl_relaxed(reg_base + INTCL_HPPIR) & BIT(31));
}

static void csky_mpintc_eoi(struct irq_data *d) {
            writel_relaxed(d->hwirq, reg_base + INTCL_CACR);
}
```

IRQ enable/disable flow:

```
static void csky_mpintc_enable(struct irq_data *d) {
            writel_relaxed(d->hwirq, reg_base + INTCL_SENR);
}

static void csky_mpintc_disable(struct irq_data *d) {
            writel_relaxed(d->hwirq, reg_base + INTCL_CENR);
}
```

IPI mechanism

```
static void csky_mpintc_send_ipi(const unsigned long *mask)
{
            /*
             * INTCL_SIGR[3:0] INTID
             * INTCL_SIGR[8:15] CPUMASK
             */
            writel_relaxed((*mask) << 8 | IPI_IRQ, reg_base + INTCL_SIGR);
}
```

IRQ TYPE Config

...

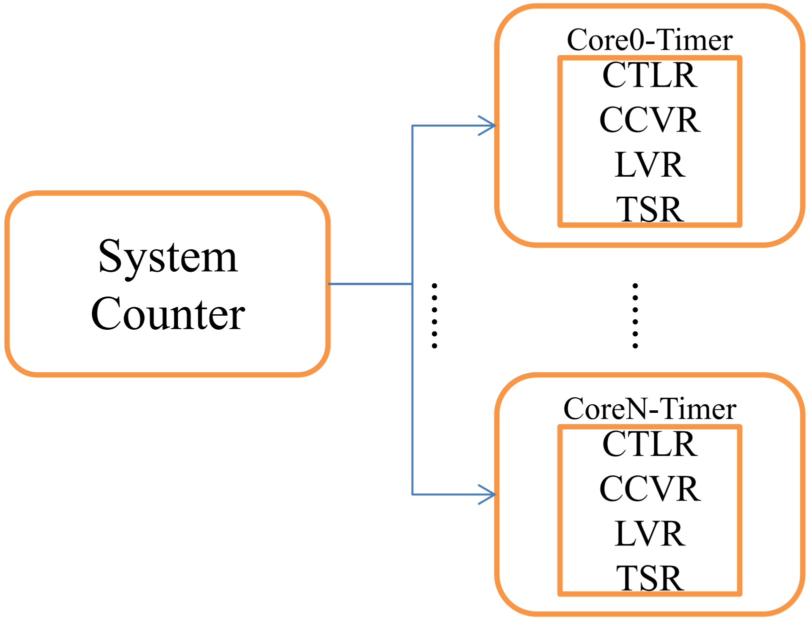
IRQ Priority Config

...

MP-TIMER

- Private Interrupt Num: 16

- Count Control Register
- Count Current Value Register
- Load Value Register
- TS Clear Register



MP-TIMER

linux/drivers/clocksource/timer-mp-csky.c

```
static irqreturn_t csky_timer_interrupt(int irq, void *dev) {
            struct timer_of *to = this_cpu_ptr(&csky_to);
            mtcr(PTIM_TSR, 0);
            to->clkevt.event_handler(&to->clkevt);
            return IRQ_HANDLED;
}

static int csky_mptimer_set_next_event(unsigned long delta,
                                       struct clock_event_device *ce) {
            mtcr(PTIM_LVR, delta);
            return 0;
}

static u64 clksrc_read(struct clocksource *c) {
            return (u64)mfcr(PTIM_CCVR);
}
```

Atomic Implementation

arch/csky/include/asm/ cmpxchg.h atomic.h

```
#define ATOMIC_OP(op, c_op)                                          \
static inline void atomic_##op(int i, atomic_t *v) {                 \
          asm volatile (                                             \
          "1:       ldex.w              %0, (%2) \n"                 \
          "         " #op "             %0, %1   \n"                 \
          "         stex.w              %0, (%2) \n"                 \
          "         bez                 %0, 1b   \n"                 \
                    : "=&r" (tmp)                                    \
```

"Ldex rz, (rx, #off)"

- will add an entry into the local monitor, and the entry is composed of a address tag and a exclusive flag (inited with 1).
- Any stores (include other cores') will break the exclusive flag to 0 in the entry which could be indexed by the address tag.

"Stex rz, (rx, #off)" has two condition:

- Store Success: When the entry's exclusive flag is 1, it will store rz to the [rx + off] address and the rz will be set to 1.
- Store Failure: When the entry's exclusive flag is 0, just rz will be set to 0.

ref: Documentation/memory-barriers.txt

Spinlock(Test-and-set Style)

```
/* Test-and-set spin-locking. */
arch_spin_lock(arch_spinlock_t *lock)  {
"1:        ldex.w              %0, (%1) \n"
"          bnez                %0, 1b   \n"
"          movi                %0, 1    \n"
"          stex.w              %0, (%1) \n"
"          bez                 %0, 1b   \n"
smp_mb(); }

arch_spin_unlock(arch_spinlock_t *lock) {
smp_mb();
"          movi                %0, 0    \n"
"          stw                 %0, (%1) \n" }
```

Rwlock (Test-and-set Style)

```
/* read lock/unlock/trylock */
arch_read_lock(arch_rwlock_t *lock) {
"1:        ldex.w     %0, (%1) \n"
"          blz        %0, 1b   \n"
"          addi       %0, 1    \n"
"          stex.w     %0, (%1) \n"
"          bez        %0, 1b   \n"
smp_mb(); }

arch_read_unlock(arch_rwlock_t *lock) {
smp_mb();
"1:        ldex.w     %0, (%1) \n"
"          subi       %0, 1    \n"
"          stex.w     %0, (%1) \n"
"          bez        %0, 1b   \n" }
```

```
/* write lock/unlock/trylock */
arch_write_lock(arch_rwlock_t *lock) {
"1:        ldex.w     %0, (%1) \n"
"          bnez       %0, 1b   \n"
"          subi       %0, 1    \n"
"          stex.w     %0, (%1) \n"
"          bez        %0, 1b   \n"
smp_mb(); }

arch_write_unlock(arch_rwlock_t *lock) {
smp_mb();
"1:        ldex.w     %0, (%1) \n"
"          movi       %0, 0    \n"
"          stex.w     %0, (%1) \n"
"          bez        %0, 1b   \n" }
```

Ticketlock & QueueRwlock

linux/arch/csky/include/asm/spinlock.h

```
arch_spin_lock(arch_spinlock_t *lock) {
arch_spinlock_t lockval;
u32 ticket_next = 1 << TICKET_NEXT; //16

"1:             ldex.w                  %0, (%2) \n"
"               mov                     %1, %0 \n"
"               add                     %0, %3 \n"
"               stex.w                  %0, (%2) \n"
: "=&r" (tmp), "=&r" (lockval)
: "r"(p), "r"(ticket_next)
: "cc");

while (lockval.tickets.next != lockval.tickets.owner)
   lockval.tickets.owner = READ_ONCE(lock->tickets.owner);
               smp_mb();
}

arch_spin_unlock(arch_spinlock_t *lock) {
               smp_mb();
               lock->tickets.owner++;
}
```

Memory Barrier

```
Documentation/memory-barriers.txt:

For example, consider the following sequence of events:

            CPU 1                   CPU 2
            ===============         ===============
            { A == 1; B == 2 }
            A = 3;                  x = B;
            B = 4;                  y = A;

The set of accesses as seen by the memory system in the middle can be arranged
in 24 different combinations:

            STORE A=3,  STORE B=4,  y=LOAD A->3,            x=LOAD B->4
            STORE A=3,  STORE B=4,  x=LOAD B->4,            y=LOAD A->3
            STORE A=3,  y=LOAD A->3,            STORE B=4,  x=LOAD B->4
            STORE A=3,  y=LOAD A->3,            x=LOAD B->2,            STORE B=4
            STORE A=3,  x=LOAD B->2,            STORE B=4,  y=LOAD A->3
            STORE A=3,  x=LOAD B->2,            y=LOAD A->3,            STORE B=4
            STORE B=4,  STORE A=3,  y=LOAD A->3,            x=LOAD B->4
            STORE B=4, ...
            ...
```

Memory Barrier

```
sync:        completion barrier
sync.s:      completion barrier and shareable to other cores
sync.i:      completion barrier with flush cpu pipeline
sync.is:     completion barrier with flush cpu pipeline and shareable to other cores


bar.brwarw:  ordering barrier for all load/store instructions before it
bar.brwarws: ordering barrier for all load/store instructions before it and shareable to other cores
bar.brar:    ordering barrier for all load       instructions before it
bar.brars:   ordering barrier for all load       instructions before it and shareable to other cores
bar.bwaw:    ordering barrier for all store      instructions before it
bar.bwaws:   ordering barrier for all store      instructions before it and shareable to other cores
```

Cache & TLB sync for SMP

arch/csky/mm/tlb.c:

• tlbi.va(s) rx	Delete va-tlb-entry with care ASID
• tlbi.vaa(s) rx	Delete va-tlb-entry without care ASID
• tlbi.asid(s) rx	Delete all the ASID tlb-entries
• tlbi.all(s)	Delete all the ASID tlb-entries

arch/csky/mm/cachev2.c:(shareable in tlb-entry)

• icache.iva rx	invalid va in icache
• dcache.iva rx	invalid va in dcache
• dcache.cval1 rx	flush va in dcache
• dcache.cva rx	flush va in dcache & l2cache
• dcache.civa rx	flush and invalid va in dcache & l2cache

SMP Qemu Demo

Guide: https://c-sky.github.io/

```
  $ xz -d vmlinux.xz
  $ mkdir toolchain
  $ cd toolchain
  $ tar -Jxf ../qemu_csky_ck860_4.18_glibc_defconfig.tar.xz
  $ cd ..
  $ toolchain/csky-qemu/bin/qemu-system-csky2 -kernel vmlinux -dtb qemu.dtb
-nographic -M mp860 -smp 4
```

Thank You

Any Question is welcome!

www.c-sky.com