ARM TCM (Tightly-Coupled Memory) handling in Linux

Written by Linus Walleij <linus.walleij@stericsson.com>

Some ARM SoC:s have a so-called TCM (Tightly-Coupled Memory).
This is usually just a few (4-64) KiB of RAM inside the ARM
processor.

Due to being embedded inside the CPU The TCM has a
Harvard-architecture, so there is an ITCM (instruction TCM)
and a DTCM (data TCM). The DTCM can not contain any
instructions, but the ITCM can actually contain data.
The size of DTCM or ITCM is minimum 4KiB so the typical
minimum configuration is 4KiB ITCM and 4KiB DTCM.

ARM CPU:s have special registers to read out status, physical
location and size of TCM memories. arch/arm/include/asm/cputype.h
defines a CPUID_TCM register that you can read out from the
system control coprocessor. Documentation from ARM can be found
at http://infocenter.arm.com, search for "TCM Status Register"
to see documents for all CPUs. Reading this register you can
determine if ITCM (bit 0) and/or DTCM (bit 16) is present in the
machine.

There is further a TCM region register (search for "TCM Region
Registers" at the ARM site) that can report and modify the location
size of TCM memories at runtime. This is used to read out and modify
TCM location and size. Notice that this is not a MMU table: you
actually move the physical location of the TCM around. At the
place you put it, it will mask any underlying RAM from the
CPU so it is usually wise not to overlap any physical RAM with
the TCM.

The TCM memory can then be remapped to another address again using
the MMU, but notice that the TCM if often used in situations where
the MMU is turned off. To avoid confusion the current Linux
implementation will map the TCM 1 to 1 from physical to virtual
memory in the location specified by the machine.

TCM is used for a few things:

- FIQ and other interrupt handlers that need deterministic
  timing and cannot wait for cache misses.

- Idle loops where all external RAM is set to self-refresh
  retention mode, so only on-chip RAM is accessible by
  the CPU and then we hang inside ITCM waiting for an
  interrupt.

- Other operations which implies shutting off or reconfiguring
  the external RAM controller.

There is an interface for using TCM on the ARM architecture
in <asm/tcm.h>. Using this interface it is possible to:

- Define the physical address and size of ITCM and DTCM.

- Tag functions to be compiled into ITCM.

- Tag data and constants to be allocated to DTCM and ITCM.

- Have the remaining TCM RAM added to a special
  allocation pool with gen_pool_create() and gen_pool_add()
  and provice tcm_alloc() and tcm_free() for this
  memory. Such a heap is great for things like saving
  device state when shutting off device power domains.

A machine that has TCM memory shall select HAVE_TCM in
arch/arm/Kconfig for itself, and then the
rest of the functionality will depend on the physical
location and size of ITCM and DTCM to be defined in
mach/memory.h for the machine. Code that needs to use
TCM shall #include <asm/tcm.h> If the TCM is not located
at the place given in memory.h it will be moved using
the TCM Region registers.

Functions to go into itcm can be tagged like this:
int __tcmfunc foo(int bar);

Variables to go into dtcm can be tagged like this:
int __tcmdata foo;

Constants can be tagged like this:
int __tcmconst foo;

To put assembler into TCM just use
.section ".tcm.text" or .section ".tcm.data"
respectively.

Example code:

```
#include <asm/tcm.h>

/* Uninitialized data */
static u32 __tcmdata tcmvar;
/* Initialized data */
static u32 __tcmdata tcmassigned = 0x2BADBABEU;
/* Constant */
static const u32 __tcmconst tcmconst = 0xCAFEBABEU;

static void __tcmlocalfunc tcm_to_tcm(void)
{
        int i;
        for (i = 0; i < 100; i++)
                tcmvar ++;
}

static void __tcmfunc hello_tcm(void)
{
        /* Some abstract code that runs in ITCM */
        int i;
        for (i = 0; i < 100; i++) {
```

```
                        tcmvar ++;
            }
            tcm_to_tcm();
}

static void __init test_tcm(void)
{
            u32 *tcmem;
            int i;

            hello_tcm();
            printk("Hello TCM executed from ITCM RAM\n");

            printk("TCM variable from testrun: %u @ %p\n", tcmvar, &tcmvar);
            tcmvar = 0xDEADBEEFU;
            printk("TCM variable: 0x%x @ %p\n", tcmvar, &tcmvar);

            printk("TCM assigned variable: 0x%x @ %p\n", tcmassigned, &tcmassigned);

            printk("TCM constant: 0x%x @ %p\n", tcmconst, &tcmconst);

            /* Allocate some TCM memory from the pool */
            tcmem = tcm_alloc(20);
            if (tcmem) {
                    printk("TCM Allocated 20 bytes of TCM @ %p\n", tcmem);
                    tcmem[0] = 0xDEADBEEFU;
                    tcmem[1] = 0x2BADBABEU;
                    tcmem[2] = 0xCAFEBABEU;
                    tcmem[3] = 0xDEADBEEFU;
                    tcmem[4] = 0x2BADBABEU;
                    for (i = 0; i < 5; i++)
                            printk("TCM tcmem[%d] = %08x\n", i, tcmem[i]);
                    tcm_free(tcmem, 20);
            }
}
```