

[PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-11/2249.html>

dhowells_at_redhat.com

Date: 11/08/04

Date: Mon, 8 Nov 2004 14:34:17 GMT
To: torvalds@osdl.org, akpm@osdl.org, davidm@snapgear.com

The attached patches provides part 4 of an architecture implementation for the Fujitsu FR-V CPU series, configurably as Linux or uClinux.

Signed-Off-By: dhowells@redhat.com

```
diffstat frv-arch_4-2610rc1mm3.diff
Makefile | 22 +
head-uc-fr401.S | 311 +++++
head-uc-fr451.S | 174 +++++
head-uc-fr555.S | 347 +++++
init_task.c | 39 ++
irq-mb93091.c | 116 +++++
irq-mb93093.c | 99 +++++
irq-mb93493.c | 108 +++++
irq-routing.c | 291 +++++
irq.c | 764 +++++
kernel_thread.S | 77 +++++
local.h | 56 +++++
pm-mb93093.c | 66 +++++
pm.c | 432 +++++
process.c | 384 +++++
15 files changed, 3286 insertions(+)
```

```
diff -uNrp /warthog/kernels/linux-2.6.10-rc1-mm3/arch/frv/kernel/head-uc-fr401.S linux-2.6.10-rc1
--- /warthog/kernels/linux-2.6.10-rc1-mm3/arch/frv/kernel/head-uc-fr401.S      1970-01-01 01:00:
+++ linux-2.6.10-rc1-mm3-frv/arch/frv/kernel/head-uc-fr401.S      2004-11-05 14:13:03.000000000 +00
@@ -0,0 +1,311 @@
+/* head-uc-fr401.S: FR401/3/5 uc-linux specific bits of initialisation
+ *
+ * Copyright (C) 2004 Red Hat, Inc. All Rights Reserved.
+ * Written by David Howells (dhowells@redhat.com)
+ *
+ * This program is free software; you can redistribute it and/or
+ * modify it under the terms of the GNU General Public License
+ * as published by the Free Software Foundation; either version
+ * 2 of the License, or (at your option) any later version.
+ */
+
+#include <linux/config.h>
+#include <linux/threads.h>
+#include <linux/linkage.h>
+#include <asm/ptrace.h>
+#include <asm/page.h>
```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

#include <asm/spr-regs.h>
#include <asm/mb86943a.h>
#include "head.inc"
+
+
+#define __400_DBR0      0xfe000e00
+#define __400_DBR1      0xfe000e08
+#define __400_DBR2      0xfe000e10      /* not on FR401 */
+#define __400_DBR3      0xfe000e18      /* not on FR401 */
+#define __400_DAM0      0xfe000f00
+#define __400_DAM1      0xfe000f08
+#define __400_DAM2      0xfe000f10      /* not on FR401 */
+#define __400_DAM3      0xfe000f18      /* not on FR401 */
+#define __400_LGCR      0xfe000010
+#define __400_LCR        0xfe000100
+#define __400_LSBR      0xfe000c00
+
+      .section      .text.init,"ax"
+      .balign      4
+
+#####
+#
+# describe the position and layout of the SDRAM controller registers
+#
+#      ENTRY:          EXIT:
+# GR5   -              cacheline size
+# GR11  -              displacement of 2nd SDRAM addr reg from GR14
+# GR12  -              displacement of 3rd SDRAM addr reg from GR14
+# GR13  -              displacement of 4th SDRAM addr reg from GR14
+# GR14  -              address of 1st SDRAM addr reg
+# GR15  -              amount to shift address by to match SDRAM addr reg
+# GR26 & __head_reference [saved]
+# GR30 LED address      [saved]
+# CC0   -              T if DBR0 is present
+# CC1   -              T if DBR1 is present
+# CC2   -              T if DBR2 is present (not FR401/FR401A)
+# CC3   -              T if DBR3 is present (not FR401/FR401A)
+#
+#####
+      .globl      __head_fr401_describe_sdram
+__head_fr401_describe_sdram:
+      sethi.p      %hi(__400_DBR0),gr14
+      setlo        %lo(__400_DBR0),gr14
+      setlos.p     #__400_DBR1-__400_DBR0,gr11
+      setlos        #__400_DBR2-__400_DBR0,gr12
+      setlos.p     #__400_DBR3-__400_DBR0,gr13
+      setlos        #32,gr5              ; cacheline size
+      setlos.p     #0,gr15              ; amount to shift addr reg by
+
+      # specify which DBR regs are present
+      setlos        #0x00ff,gr4
+      movgs        gr4,cccr
+      movsg        psr,gr3              ; check for FR401/FR401A
+      srli         gr3,#25,gr3
+      subicc       gr3,#0x20>>1,gr0,icc0
+      bnelr        icc0,#1
+      setlos        #0x000f,gr4
+      movgs        gr4,cccr
+      bralr
+
+#####
+#

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+# rearrange the bus controller registers
+#
+# ENTRY: EXIT:
+# GR26 &__head_reference [saved]
+# GR30 LED address revised LED address
+#
+#####
+ .globl __head_fr401_set_busctl
+__head_fr401_set_busctl:
+ sethi.p %hi(__400_LGCR),gr4
+ setlo %lo(__400_LGCR),gr4
+ sethi.p %hi(__400_LSBR),gr10
+ setlo %lo(__400_LSBR),gr10
+ sethi.p %hi(__400_LCR),gr11
+ setlo %lo(__400_LCR),gr11
+
+ # set the bus controller
+ ldi @(gr4,#0),gr5
+ ori gr5,#0xff,gr5 ; make sure all chip-selects are enabled
+ sti gr5,@(gr4,#0)
+
+ sethi.p %hi(__region_CS1),gr4
+ setlo %lo(__region_CS1),gr4
+ sethi.p %hi(__region_CS1_M),gr5
+ setlo %lo(__region_CS1_M),gr5
+ sethi.p %hi(__region_CS1_C),gr6
+ setlo %lo(__region_CS1_C),gr6
+ sti gr4,@(gr10,#1*0x08)
+ sti gr5,@(gr10,#1*0x08+0x100)
+ sti gr6,@(gr11,#1*0x08)
+ sethi.p %hi(__region_CS2),gr4
+ setlo %lo(__region_CS2),gr4
+ sethi.p %hi(__region_CS2_M),gr5
+ setlo %lo(__region_CS2_M),gr5
+ sethi.p %hi(__region_CS2_C),gr6
+ setlo %lo(__region_CS2_C),gr6
+ sti gr4,@(gr10,#2*0x08)
+ sti gr5,@(gr10,#2*0x08+0x100)
+ sti gr6,@(gr11,#2*0x08)
+ sethi.p %hi(__region_CS3),gr4
+ setlo %lo(__region_CS3),gr4
+ sethi.p %hi(__region_CS3_M),gr5
+ setlo %lo(__region_CS3_M),gr5
+ sethi.p %hi(__region_CS3_C),gr6
+ setlo %lo(__region_CS3_C),gr6
+ sti gr4,@(gr10,#3*0x08)
+ sti gr5,@(gr10,#3*0x08+0x100)
+ sti gr6,@(gr11,#3*0x08)
+ sethi.p %hi(__region_CS4),gr4
+ setlo %lo(__region_CS4),gr4
+ sethi.p %hi(__region_CS4_M),gr5
+ setlo %lo(__region_CS4_M),gr5
+ sethi.p %hi(__region_CS4_C),gr6
+ setlo %lo(__region_CS4_C),gr6
+ sti gr4,@(gr10,#4*0x08)
+ sti gr5,@(gr10,#4*0x08+0x100)
+ sti gr6,@(gr11,#4*0x08)
+ sethi.p %hi(__region_CS5),gr4
+ setlo %lo(__region_CS5),gr4
+ sethi.p %hi(__region_CS5_M),gr5
+ setlo %lo(__region_CS5_M),gr5
+ sethi.p %hi(__region_CS5_C),gr6

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+     setlo      %lo(__region_CS5_C),gr6
+     sti        gr4,@(gr10,#5*0x08)
+     sti        gr5,@(gr10,#5*0x08+0x100)
+     sti        gr6,@(gr11,#5*0x08)
+     sethi.p    %hi(__region_CS6),gr4
+     setlo      %lo(__region_CS6),gr4
+     sethi.p    %hi(__region_CS6_M),gr5
+     setlo      %lo(__region_CS6_M),gr5
+     sethi.p    %hi(__region_CS6_C),gr6
+     setlo      %lo(__region_CS6_C),gr6
+     sti        gr4,@(gr10,#6*0x08)
+     sti        gr5,@(gr10,#6*0x08+0x100)
+     sti        gr6,@(gr11,#6*0x08)
+     sethi.p    %hi(__region_CS7),gr4
+     setlo      %lo(__region_CS7),gr4
+     sethi.p    %hi(__region_CS7_M),gr5
+     setlo      %lo(__region_CS7_M),gr5
+     sethi.p    %hi(__region_CS7_C),gr6
+     setlo      %lo(__region_CS7_C),gr6
+     sti        gr4,@(gr10,#7*0x08)
+     sti        gr5,@(gr10,#7*0x08+0x100)
+     sti        gr6,@(gr11,#7*0x08)
+     membar
+     bar
+
+     # adjust LED bank address
+     sethi.p    %hi(LED_ADDR - 0x20000000 +__region_CS2),gr30
+     setlo      %lo(LED_ADDR - 0x20000000 +__region_CS2),gr30
+     bralr
+
+#####
+#
+# determine the total SDRAM size
+#
+#     ENTRY:                EXIT:
+# GR25 -                    SDRAM size
+# GR26 & __head_reference   [saved]
+# GR30 LED address          [saved]
+#
+#####
+     .globl     __head_fr401_survey_sdram
+__head_fr401_survey_sdram:
+     sethi.p    %hi(__400_DAM0),gr11
+     setlo      %lo(__400_DAM0),gr11
+     sethi.p    %hi(__400_DBR0),gr12
+     setlo      %lo(__400_DBR0),gr12
+
+     sethi.p    %hi(0xfe000000),gr17           ; unused SDRAM DBR value
+     setlo      %lo(0xfe000000),gr17
+     setlos     #0,gr25
+
+     ldi        @(gr12,#0x00),gr4             ; DAR0
+     subcc      gr4,gr17,gr0,icc0
+     beq        icc0,#0,__head_no_DCS0
+     ldi        @(gr11,#0x00),gr6             ; DAM0: bits 31:20 match addr 31:20
+     add        gr25,gr6,gr25
+     addi       gr25,#1,gr25
+__head_no_DCS0:
+
+     ldi        @(gr12,#0x08),gr4             ; DAR1
+     subcc      gr4,gr17,gr0,icc0
+     beq        icc0,#0,__head_no_DCS1

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+     ldi             @(gr11,#0x08),gr6             ; DAM1: bits 31:20 match addr 31:20
+     add             gr25,gr6,gr25
+     addi            gr25,#1,gr25
+__head_no_DCS1:
+
+     # FR401/FR401A does not have DCS2/3
+     movsg           psr,gr3
+     srli            gr3,#25,gr3
+     subicc          gr3,#0x20>>1,gr0,icc0
+     beq             icc0,#0,__head_no_DCS3
+
+     ldi             @(gr12,#0x10),gr4             ; DAR2
+     subcc           gr4,gr17,gr0,icc0
+     beq             icc0,#0,__head_no_DCS2
+     ldi             @(gr11,#0x10),gr6             ; DAM2: bits 31:20 match addr 31:20
+     add             gr25,gr6,gr25
+     addi            gr25,#1,gr25
+__head_no_DCS2:
+
+     ldi             @(gr12,#0x18),gr4             ; DAR3
+     subcc           gr4,gr17,gr0,icc0
+     beq             icc0,#0,__head_no_DCS3
+     ldi             @(gr11,#0x18),gr6             ; DAM3: bits 31:20 match addr 31:20
+     add             gr25,gr6,gr25
+     addi            gr25,#1,gr25
+__head_no_DCS3:
+     bralr
+
+#####
+#
+# set the protection map with the I/DAMPR registers
+#
+#     ENTRY:                EXIT:
+# GR25 SDRAM size           [saved]
+# GR26 &__head_reference    [saved]
+# GR30 LED address          [saved]
+#
+#####
+     .globl          __head_fr401_set_protection
+__head_fr401_set_protection:
+     movsg           lr,gr27
+
+     # set the I/O region protection registers for FR401/3/5
+     sethi.p         %hi(__region_IO),gr5
+     setlo           %lo(__region_IO),gr5
+     ori             gr5,#xAMPRx_SS_512Mb|xAMPRx_S_KERNEL|xAMPRx_C|xAMPRx_V,gr5
+     movgs           gr0,iampr7
+     movgs           gr5,dampr7                     ; General I/O tile
+
+     # need to tile the remaining IAMPR/DAMPR registers to cover as much of the RAM as possible
+     # - start with the highest numbered registers
+     sethi.p         %hi(__kernel_image_end),gr8
+     setlo           %lo(__kernel_image_end),gr8
+     sethi.p         %hi(32768),gr4                 ; allow for a maximal allocator bitmap
+     setlo           %lo(32768),gr4
+     add             gr8,gr4,gr8
+     sethi.p         %hi(1024*2048-1),gr4           ; round up to nearest 2MiB
+     setlo           %lo(1024*2048-1),gr4
+     add.p           gr8,gr4,gr8
+     not             gr4,gr4
+     and             gr8,gr4,gr8
+
+

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+     sethi.p     %hi(__page_offset),gr9
+     setlo      %lo(__page_offset),gr9
+     add        gr9,gr25,gr9
+
+     # GR8 = base of uncovered RAM
+     # GR9 = top of uncovered RAM
+
+#ifdef CONFIG_MB93093_PDK
+     sethi.p     %hi(__region_CS2),gr4
+     setlo      %lo(__region_CS2),gr4
+     ori        gr4,#xAMPRx_SS_1Mb|xAMPRx_S_KERNEL|xAMPRx_C|xAMPRx_V,gr4
+     movgs     gr4,dampr6
+     movgs     gr0,iampr6
+#else
+     call      __head_split_region
+     movgs     gr4,iampr6
+     movgs     gr5,dampr6
+#endif
+     call      __head_split_region
+     movgs     gr4,iampr5
+     movgs     gr5,dampr5
+     call      __head_split_region
+     movgs     gr4,iampr4
+     movgs     gr5,dampr4
+     call      __head_split_region
+     movgs     gr4,iampr3
+     movgs     gr5,dampr3
+     call      __head_split_region
+     movgs     gr4,iampr2
+     movgs     gr5,dampr2
+     call      __head_split_region
+     movgs     gr4,iampr1
+     movgs     gr5,dampr1
+
+     # cover kernel core image with kernel-only segment
+     sethi.p     %hi(__page_offset),gr8
+     setlo      %lo(__page_offset),gr8
+     call      __head_split_region
+
+#ifdef CONFIG_PROTECT_KERNEL
+     ori.p      gr4,#xAMPRx_S_KERNEL,gr4
+     ori        gr5,#xAMPRx_S_KERNEL,gr5
+#endif
+
+     movgs     gr4,iampr0
+     movgs     gr5,dampr0
+     jmp      @(gr27,gr0)
diff -uNrp /warthog/kernels/linux-2.6.10-rc1-mm3/arch/frv/kernel/head-uc-fr451.S linux-2.6.10-rc1
--- /warthog/kernels/linux-2.6.10-rc1-mm3/arch/frv/kernel/head-uc-fr451.S      1970-01-01 01:00:
+++ linux-2.6.10-rc1-mm3-frv/arch/frv/kernel/head-uc-fr451.S      2004-11-05 14:13:03.000000000 +00
@@ -0,0 +1,174 @@
+/* head-uc-fr451.S: FR451 uc-linux specific bits of initialisation
+ *
+ * Copyright (C) 2004 Red Hat, Inc. All Rights Reserved.
+ * Written by David Howells (dhowells@redhat.com)
+ *
+ * This program is free software; you can redistribute it and/or
+ * modify it under the terms of the GNU General Public License
+ * as published by the Free Software Foundation; either version
+ * 2 of the License, or (at your option) any later version.
+ */
+

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

#include <linux/config.h>
#include <linux/threads.h>
#include <linux/linkage.h>
#include <asm/ptrace.h>
#include <asm/page.h>
#include <asm/spr-regs.h>
#include <asm/mb86943a.h>
#include "head.inc"
+
+
#define __400_DBR0      0xfe000e00
#define __400_DBR1      0xfe000e08
#define __400_DBR2      0xfe000e10
#define __400_DBR3      0xfe000e18
#define __400_DAM0      0xfe000f00
#define __400_DAM1      0xfe000f08
#define __400_DAM2      0xfe000f10
#define __400_DAM3      0xfe000f18
#define __400_LGCR      0xfe000010
#define __400_LCR       0xfe000100
#define __400_LSBR      0xfe000c00
+
+
.section      .text.init,"ax"
.balign      4
+
+
#####
+#
+# set the protection map with the I/DAMPR registers
+#
+#     ENTRY:                EXIT:
+# GR25 SDRAM size           [saved]
+# GR26 & __head_reference   [saved]
+# GR30 LED address          [saved]
+#
+#####
+
.globl      __head_fr451_set_protection
__head_fr451_set_protection:
+
+   movsg      lr,gr27
+
+   movgs      gr0,dampr10
+   movgs      gr0,damlr10
+   movgs      gr0,dampr9
+   movgs      gr0,damlr9
+   movgs      gr0,dampr8
+   movgs      gr0,damlr8
+
+   # set the I/O region protection registers for FR401/3/5
+   sethi.p    %hi(__region_IO),gr5
+   setlo      %lo(__region_IO),gr5
+   sethi.p    %hi(0x1fffffff),gr7
+   setlo      %lo(0x1fffffff),gr7
+   ori        gr5,#xAMPRx_SS_512Mb|xAMPRx_S_KERNEL|xAMPRx_C|xAMPRx_V,gr5
+   movgs      gr5,dampr11                ; General I/O tile
+   movgs      gr7,damlr11
+
+   # need to tile the remaining IAMPR/DAMPR registers to cover as much of the RAM as possible
+   # - start with the highest numbered registers
+   sethi.p    %hi(__kernel_image_end),gr8
+   setlo      %lo(__kernel_image_end),gr8
+   sethi.p    %hi(32768),gr4                ; allow for a maximal allocator bitmap
+   setlo      %lo(32768),gr4
+   add        gr8,gr4,gr8

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+   sethi.p      %hi(1024*2048-1),gr4          ; round up to nearest 2MiB
+   setlo      %lo(1024*2048-1),gr4
+   add.p      gr8,gr4,gr8
+   not       gr4,gr4
+   and       gr8,gr4,gr8
+
+   sethi.p      %hi(__page_offset),gr9
+   setlo      %lo(__page_offset),gr9
+   add       gr9,gr25,gr9
+
+   sethi.p      %hi(0xffffc000),gr11
+   setlo      %lo(0xffffc000),gr11
+
+   # GR8 = base of uncovered RAM
+   # GR9 = top of uncovered RAM
+   # GR11 = xAMLR mask
+   LEDS       0x3317
+   call       __head_split_region
+   movgs     gr4,iampr7
+   movgs     gr6,iamlr7
+   movgs     gr5,dampr7
+   movgs     gr7,damlr7
+
+   LEDS       0x3316
+   call       __head_split_region
+   movgs     gr4,iampr6
+   movgs     gr6,iamlr6
+   movgs     gr5,dampr6
+   movgs     gr7,damlr6
+
+   LEDS       0x3315
+   call       __head_split_region
+   movgs     gr4,iampr5
+   movgs     gr6,iamlr5
+   movgs     gr5,dampr5
+   movgs     gr7,damlr5
+
+   LEDS       0x3314
+   call       __head_split_region
+   movgs     gr4,iampr4
+   movgs     gr6,iamlr4
+   movgs     gr5,dampr4
+   movgs     gr7,damlr4
+
+   LEDS       0x3313
+   call       __head_split_region
+   movgs     gr4,iampr3
+   movgs     gr6,iamlr3
+   movgs     gr5,dampr3
+   movgs     gr7,damlr3
+
+   LEDS       0x3312
+   call       __head_split_region
+   movgs     gr4,iampr2
+   movgs     gr6,iamlr2
+   movgs     gr5,dampr2
+   movgs     gr7,damlr2
+
+   LEDS       0x3311
+   call       __head_split_region
+   movgs     gr4,iampr1
+   movgs     gr6,iamlr1

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+     movgs         gr5,dampr1
+     movgs         gr7,damlr1
+
+     # cover kernel core image with kernel-only segment
+     LEDS          0x3310
+     sethi.p       %hi(__page_offset),gr8
+     setlo         %lo(__page_offset),gr8
+     call          __head_split_region
+
+#ifdef CONFIG_PROTECT_KERNEL
+     ori.p         gr4,#xAMPRx_S_KERNEL,gr4
+     ori           gr5,#xAMPRx_S_KERNEL,gr5
+#endif
+
+     movgs         gr4,iampr0
+     movgs         gr6,iamlr0
+     movgs         gr5,dampr0
+     movgs         gr7,damlr0
+
+     # start in TLB context 0 with no page tables
+     movgs         gr0,cxnr
+     movgs         gr0,ttbr
+
+     # the FR451 also has an extra trap base register
+     movsg         tbr,gr4
+     movgs         gr4,btbr
+
+     # turn on the timers as appropriate
+     movgs         gr0,timerh
+     movgs         gr0,timerl
+     movgs         gr0,timerd
+     movsg         hsr0,gr4
+     sethi.p       %hi(HSR0_ETMI),gr5
+     setlo         %lo(HSR0_ETMI),gr5
+     or            gr4,gr5,gr4
+     movgs         gr4,hsr0
+
+     LEDS          0x3300
+     jmp1          @(gr27,gr0)
diff -uNrp /warthog/kernels/linux-2.6.10-rc1-mm3/arch/frv/kernel/head-uc-fr555.S linux-2.6.10-rc1
--- /warthog/kernels/linux-2.6.10-rc1-mm3/arch/frv/kernel/head-uc-fr555.S      1970-01-01 01:00:
+++ linux-2.6.10-rc1-mm3-frv/arch/frv/kernel/head-uc-fr555.S      2004-11-05 14:13:03.000000000 +00
@@ -0,0 +1,347 @@
+/* head-uc-fr555.S: FR555 uc-linux specific bits of initialisation
+ *
+ * Copyright (C) 2004 Red Hat, Inc. All Rights Reserved.
+ * Written by David Howells (dhowells@redhat.com)
+ *
+ * This program is free software; you can redistribute it and/or
+ * modify it under the terms of the GNU General Public License
+ * as published by the Free Software Foundation; either version
+ * 2 of the License, or (at your option) any later version.
+ */
+
+#include <linux/config.h>
+#include <linux/threads.h>
+#include <linux/linkage.h>
+#include <asm/ptrace.h>
+#include <asm/page.h>
+#include <asm/spr-regs.h>
+#include <asm/mb86943a.h>
+#include "head.inc"

```

```

+
+
+#define __551_DARS0      0xfeff0100
+#define __551_DARS1      0xfeff0104
+#define __551_DARS2      0xfeff0108
+#define __551_DARS3      0xfeff010c
+#define __551_DAMK0      0xfeff0110
+#define __551_DAMK1      0xfeff0114
+#define __551_DAMK2      0xfeff0118
+#define __551_DAMK3      0xfeff011c
+#define __551_LCR        0xfeff1100
+#define __551_LSBR       0xfeff1c00
+
+      .section      .text.init,"ax"
+      .balign      4
+
+#####
+#
+# describe the position and layout of the SDRAM controller registers
+#
+#      ENTRY:                EXIT:
+# GR5 -                      cacheline size
+# GR11 -                     displacement of 2nd SDRAM addr reg from GR14
+# GR12 -                     displacement of 3rd SDRAM addr reg from GR14
+# GR13 -                     displacement of 4th SDRAM addr reg from GR14
+# GR14 -                     address of 1st SDRAM addr reg
+# GR15 -                     amount to shift address by to match SDRAM addr reg
+# GR26 & __head_reference    [saved]
+# GR30 LED address          [saved]
+# CC0 -                     T if DARS0 is present
+# CC1 -                     T if DARS1 is present
+# CC2 -                     T if DARS2 is present
+# CC3 -                     T if DARS3 is present
+#
+#####
+      .globl      __head_fr555_describe_sdram
+__head_fr555_describe_sdram:
+      sethi.p     %hi(__551_DARS0),gr14
+      setlo      %lo(__551_DARS0),gr14
+      setlos.p   #__551_DARS1-__551_DARS0,gr11
+      setlos     #__551_DARS2-__551_DARS0,gr12
+      setlos.p   #__551_DARS3-__551_DARS0,gr13
+      setlos     #64,gr5                ; cacheline size
+      setlos     #20,gr15              ; amount to shift addr by
+      setlos     #0x00ff,gr4
+      movgs      gr4,cccr              ; extant DARS/DAMK regs
+      bralr
+
+#####
+#
+# rearrange the bus controller registers
+#
+#      ENTRY:                EXIT:
+# GR26 & __head_reference    [saved]
+# GR30 LED address          revised LED address
+#
+#####
+      .globl      __head_fr555_set_busctl
+__head_fr555_set_busctl:
+      LEDS       0x100f
+      sethi.p    %hi(__551_LSBR),gr10
+      setlo      %lo(__551_LSBR),gr10

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+   sethi.p      %hi(__551_LCR),gr11
+   setlo       %lo(__551_LCR),gr11
+
+   # set the bus controller
+   sethi.p      %hi(__region_CS1),gr4
+   setlo       %lo(__region_CS1),gr4
+   sethi.p      %hi(__region_CS1_M),gr5
+   setlo       %lo(__region_CS1_M),gr5
+   sethi.p      %hi(__region_CS1_C),gr6
+   setlo       %lo(__region_CS1_C),gr6
+   sti         gr4,@(gr10,#1*0x08)
+   sti         gr5,@(gr10,#1*0x08+0x100)
+   sti         gr6,@(gr11,#1*0x08)
+   sethi.p      %hi(__region_CS2),gr4
+   setlo       %lo(__region_CS2),gr4
+   sethi.p      %hi(__region_CS2_M),gr5
+   setlo       %lo(__region_CS2_M),gr5
+   sethi.p      %hi(__region_CS2_C),gr6
+   setlo       %lo(__region_CS2_C),gr6
+   sti         gr4,@(gr10,#2*0x08)
+   sti         gr5,@(gr10,#2*0x08+0x100)
+   sti         gr6,@(gr11,#2*0x08)
+   sethi.p      %hi(__region_CS3),gr4
+   setlo       %lo(__region_CS3),gr4
+   sethi.p      %hi(__region_CS3_M),gr5
+   setlo       %lo(__region_CS3_M),gr5
+   sethi.p      %hi(__region_CS3_C),gr6
+   setlo       %lo(__region_CS3_C),gr6
+   sti         gr4,@(gr10,#3*0x08)
+   sti         gr5,@(gr10,#3*0x08+0x100)
+   sti         gr6,@(gr11,#3*0x08)
+   sethi.p      %hi(__region_CS4),gr4
+   setlo       %lo(__region_CS4),gr4
+   sethi.p      %hi(__region_CS4_M),gr5
+   setlo       %lo(__region_CS4_M),gr5
+   sethi.p      %hi(__region_CS4_C),gr6
+   setlo       %lo(__region_CS4_C),gr6
+   sti         gr4,@(gr10,#4*0x08)
+   sti         gr5,@(gr10,#4*0x08+0x100)
+   sti         gr6,@(gr11,#4*0x08)
+   sethi.p      %hi(__region_CS5),gr4
+   setlo       %lo(__region_CS5),gr4
+   sethi.p      %hi(__region_CS5_M),gr5
+   setlo       %lo(__region_CS5_M),gr5
+   sethi.p      %hi(__region_CS5_C),gr6
+   setlo       %lo(__region_CS5_C),gr6
+   sti         gr4,@(gr10,#5*0x08)
+   sti         gr5,@(gr10,#5*0x08+0x100)
+   sti         gr6,@(gr11,#5*0x08)
+   sethi.p      %hi(__region_CS6),gr4
+   setlo       %lo(__region_CS6),gr4
+   sethi.p      %hi(__region_CS6_M),gr5
+   setlo       %lo(__region_CS6_M),gr5
+   sethi.p      %hi(__region_CS6_C),gr6
+   setlo       %lo(__region_CS6_C),gr6
+   sti         gr4,@(gr10,#6*0x08)
+   sti         gr5,@(gr10,#6*0x08+0x100)
+   sti         gr6,@(gr11,#6*0x08)
+   sethi.p      %hi(__region_CS7),gr4
+   setlo       %lo(__region_CS7),gr4
+   sethi.p      %hi(__region_CS7_M),gr5
+   setlo       %lo(__region_CS7_M),gr5

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+     sethi.p      %hi(__region_CS7_C),gr6
+     setlo       %lo(__region_CS7_C),gr6
+     sti        gr4,@(gr10,#7*0x08)
+     sti        gr5,@(gr10,#7*0x08+0x100)
+     sti        gr6,@(gr11,#7*0x08)
+     membar
+     bar
+
+     # adjust LED bank address
+#ifdef CONFIG_MB93091_VDK
+     sethi.p      %hi(LED_ADDR - 0x20000000 +__region_CS2),gr30
+     setlo       %lo(LED_ADDR - 0x20000000 +__region_CS2),gr30
+#endif
+     bralr
+
+#####
+#
+# determine the total SDRAM size
+#
+#     ENTRY:                EXIT:
+# GR25 -                    SDRAM size
+# GR26 & __head_reference   [saved]
+# GR30 LED address          [saved]
+#
+#####
+     .globl      __head_fr555_survey_sdram
+__head_fr555_survey_sdram:
+     sethi.p     %hi(__551_DAMK0),gr11
+     setlo      %lo(__551_DAMK0),gr11
+     sethi.p     %hi(__551_DARS0),gr12
+     setlo      %lo(__551_DARS0),gr12
+
+     sethi.p     %hi(0xffff),gr17           ; unused SDRAM AMK value
+     setlo      %lo(0xffff),gr17
+     setlos     #0,gr25
+
+     ldi        @(gr11,#0x00),gr6           ; DAMK0: bits 11:0 match addr 11:0
+     subcc     gr6,gr17,gr0,icc0
+     beq       icc0,#0,__head_no_DCS0
+     ldi        @(gr12,#0x00),gr4           ; DARS0
+     add       gr25,gr6,gr25
+     addi      gr25,#1,gr25
+__head_no_DCS0:
+
+     ldi        @(gr11,#0x04),gr6           ; DAMK1: bits 11:0 match addr 11:0
+     subcc     gr6,gr17,gr0,icc0
+     beq       icc0,#0,__head_no_DCS1
+     ldi        @(gr12,#0x04),gr4           ; DARS1
+     add       gr25,gr6,gr25
+     addi      gr25,#1,gr25
+__head_no_DCS1:
+
+     ldi        @(gr11,#0x8),gr6           ; DAMK2: bits 11:0 match addr 11:0
+     subcc     gr6,gr17,gr0,icc0
+     beq       icc0,#0,__head_no_DCS2
+     ldi        @(gr12,#0x8),gr4           ; DARS2
+     add       gr25,gr6,gr25
+     addi      gr25,#1,gr25
+__head_no_DCS2:
+
+     ldi        @(gr11,#0xc),gr6          ; DAMK3: bits 11:0 match addr 11:0
+     subcc     gr6,gr17,gr0,icc0

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+     beq             icc0,#0,__head_no_DCS3
+     ldi             @(gr12,#0xc),gr4                ; DARS3
+     add             gr25,gr6,gr25
+     addi            gr25,#1,gr25
+__head_no_DCS3:
+
+     slli            gr25,#20,gr25                  ; shift [11:0] -> [31:20]
+     bralr
+
+#####
+#
+# set the protection map with the I/DAMPR registers
+#
+#     ENTRY:                EXIT:
+# GR25 SDRAM size          saved
+# GR30 LED address        saved
+#
+#####
+     .globl         __head_fr555_set_protection
+__head_fr555_set_protection:
+     movsg          lr,gr27
+
+     sethi.p        %hi(0xfff00000),gr11
+     setlo          %lo(0xfff00000),gr11
+
+     # set the I/O region protection registers for FR555
+     sethi.p        %hi(__region_IO),gr7
+     setlo          %lo(__region_IO),gr7
+     ori            gr7,#xAMPRx_SS_512Mb|xAMPRx_S_KERNEL|xAMPRx_C|xAMPRx_V,gr5
+     movgs          gr0,iampr15
+     movgs          gr0,iamlr15
+     movgs          gr5,dampr15
+     movgs          gr7,damlr15
+
+     # need to tile the remaining IAMPR/DAMPR registers to cover as much of the RAM as possible
+     # - start with the highest numbered registers
+     sethi.p        %hi(__kernel_image_end),gr8
+     setlo          %lo(__kernel_image_end),gr8
+     sethi.p        %hi(32768),gr4                ; allow for a maximal allocator bitmap
+     setlo          %lo(32768),gr4
+     add            gr8,gr4,gr8
+     sethi.p        %hi(1024*2048-1),gr4          ; round up to nearest 2MiB
+     setlo          %lo(1024*2048-1),gr4
+     add.p          gr8,gr4,gr8
+     not            gr4,gr4
+     and            gr8,gr4,gr8
+
+     sethi.p        %hi(__page_offset),gr9
+     setlo          %lo(__page_offset),gr9
+     add            gr9,gr25,gr9
+
+     # GR8 = base of uncovered RAM
+     # GR9 = top of uncovered RAM
+     # GR11 - mask for DAMLR/IAMLR regs
+     #
+     call           __head_split_region
+     movgs          gr4,iampr14
+     movgs          gr6,iamlr14
+     movgs          gr5,dampr14
+     movgs          gr7,damlr14
+     call           __head_split_region
+     movgs          gr4,iampr13

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```
+     movgs     gr6,iamlr13
+     movgs     gr5,dampr13
+     movgs     gr7,damlr13
+     call      __head_split_region
+     movgs     gr4,iampr12
+     movgs     gr6,iamlr12
+     movgs     gr5,dampr12
+     movgs     gr7,damlr12
+     call      __head_split_region
+     movgs     gr4,iampr11
+     movgs     gr6,iamlr11
+     movgs     gr5,dampr11
+     movgs     gr7,damlr11
+     call      __head_split_region
+     movgs     gr4,iampr10
+     movgs     gr6,iamlr10
+     movgs     gr5,dampr10
+     movgs     gr7,damlr10
+     call      __head_split_region
+     movgs     gr4,iampr9
+     movgs     gr6,iamlr9
+     movgs     gr5,dampr9
+     movgs     gr7,damlr9
+     call      __head_split_region
+     movgs     gr4,iampr8
+     movgs     gr6,iamlr8
+     movgs     gr5,dampr8
+     movgs     gr7,damlr8
+
+     call      __head_split_region
+     movgs     gr4,iampr7
+     movgs     gr6,iamlr7
+     movgs     gr5,dampr7
+     movgs     gr7,damlr7
+     call      __head_split_region
+     movgs     gr4,iampr6
+     movgs     gr6,iamlr6
+     movgs     gr5,dampr6
+     movgs     gr7,damlr6
+     call      __head_split_region
+     movgs     gr4,iampr5
+     movgs     gr6,iamlr5
+     movgs     gr5,dampr5
+     movgs     gr7,damlr5
+     call      __head_split_region
+     movgs     gr4,iampr4
+     movgs     gr6,iamlr4
+     movgs     gr5,dampr4
+     movgs     gr7,damlr4
+     call      __head_split_region
+     movgs     gr4,iampr3
+     movgs     gr6,iamlr3
+     movgs     gr5,dampr3
+     movgs     gr7,damlr3
+     call      __head_split_region
+     movgs     gr4,iampr2
+     movgs     gr6,iamlr2
+     movgs     gr5,dampr2
+     movgs     gr7,damlr2
+     call      __head_split_region
+     movgs     gr4,iampr1
+     movgs     gr6,iamlr1
```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+     movgs     gr5,dampr1
+     movgs     gr7,damlr1
+
+     # cover kernel core image with kernel-only segment
+     sethi.p   %hi(__page_offset),gr8
+     setlo     %lo(__page_offset),gr8
+     call      __head_split_region
+
+#ifdef CONFIG_PROTECT_KERNEL
+     ori.p     gr4,#xAMPRx_S_KERNEL,gr4
+     ori       gr5,#xAMPRx_S_KERNEL,gr5
+#endif
+
+     movgs     gr4,iampr0
+     movgs     gr6,iamlr0
+     movgs     gr5,dampr0
+     movgs     gr7,damlr0
+     jmp1      @(gr27,gr0)
diff -uNrp /warthog/kernels/linux-2.6.10-rc1-mm3/arch/frv/kernel/init_task.c linux-2.6.10-rc1-mm3
--- /warthog/kernels/linux-2.6.10-rc1-mm3/arch/frv/kernel/init_task.c 1970-01-01 01:00:00.00000
+++ linux-2.6.10-rc1-mm3-frv/arch/frv/kernel/init_task.c 2004-11-05 14:13:03.141560129 +00
@@ -0,0 +1,39 @@
+#include <linux/mm.h>
+#include <linux/module.h>
+#include <linux/sched.h>
+#include <linux/init.h>
+#include <linux/init_task.h>
+#include <linux/fs.h>
+#include <linux/mqueue.h>
+
+#include <asm/uaccess.h>
+#include <asm/pgtable.h>
+
+
+static struct fs_struct init_fs = INIT_FS;
+static struct files_struct init_files = INIT_FILES;
+static struct signal_struct init_signals = INIT_SIGNALS(init_signals);
+static struct sighand_struct init_sighand = INIT_SIGHAND(init_sighand);
+struct mm_struct init_mm = INIT_MM(init_mm);
+
+EXPORT_SYMBOL(init_mm);
+
+/*
+ * Initial thread structure.
+ *
+ * We need to make sure that this is THREAD_SIZE aligned due to the
+ * way process stacks are handled. This is done by having a special
+ * "init_task" linker map entry..
+ */
+union thread_union init_thread_union
+    __attribute__((__section__(".data.init_task"))) =
+    { INIT_THREAD_INFO(init_task) };
+
+/*
+ * Initial task structure.
+ *
+ * All other task structs will be allocated on slabs in fork.c
+ */
+struct task_struct init_task = INIT_TASK(init_task);
+
+EXPORT_SYMBOL(init_task);
diff -uNrp /warthog/kernels/linux-2.6.10-rc1-mm3/arch/frv/kernel/irq.c linux-2.6.10-rc1-mm3-frv/a

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```
--- /warthog/kernels/linux-2.6.10-rc1-mm3/arch/frv/kernel/irq.c 1970-01-01 01:00:00.000000000 +0100
+++ linux-2.6.10-rc1-mm3-frv/arch/frv/kernel/irq.c      2004-11-05 14:13:03.147559622 +0000
@@ -0,0 +1,764 @@
+/* irq.c: FRV IRQ handling
+ *
+ * Copyright (C) 2003, 2004 Red Hat, Inc. All Rights Reserved.
+ * Written by David Howells (dhowells@redhat.com)
+ *
+ * This program is free software; you can redistribute it and/or
+ * modify it under the terms of the GNU General Public License
+ * as published by the Free Software Foundation; either version
+ * 2 of the License, or (at your option) any later version.
+ */
+
+/*
+ * (mostly architecture independent, will move to kernel/irq.c in 2.5.)
+ *
+ * IRQs are in fact implemented a bit like signal handlers for the kernel.
+ * Naturally it's not a 1:1 relation, but there are similarities.
+ */
+
+#include <linux/config.h>
+#include <linux/ptrace.h>
+#include <linux/errno.h>
+#include <linux/signal.h>
+#include <linux/sched.h>
+#include <linux/ioport.h>
+#include <linux/interrupt.h>
+#include <linux/timex.h>
+#include <linux/slab.h>
+#include <linux/random.h>
+#include <linux/smp_lock.h>
+#include <linux/init.h>
+#include <linux/kernel_stat.h>
+#include <linux/irq.h>
+#include <linux/proc_fs.h>
+#include <linux/seq_file.h>
+
+#include <asm/atomic.h>
+#include <asm/io.h>
+#include <asm/smp.h>
+#include <asm/system.h>
+#include <asm/bitops.h>
+#include <asm/uaccess.h>
+#include <asm/pgalloc.h>
+#include <asm/delay.h>
+#include <asm/irq.h>
+#include <asm/irc-regs.h>
+#include <asm/irq-routing.h>
+#include <asm/gdb-stub.h>
+
+extern void __init fpga_init(void);
+extern void __init route_mb93493_irqs(void);
+
+static void register_irq_proc (unsigned int irq);
+
+/*
+ * Special irq handlers.
+ */
+
+irqreturn_t no_action(int cpl, void *dev_id, struct pt_regs *regs) { return IRQ_HANDLED; }
+
```

```

+atomic_t irq_err_count;
+
+/*
+ * Generic, controller-independent functions:
+ */
+int show_interrupts(struct seq_file *p, void *v)
+{
+    struct irqaction *action;
+    struct irq_group *group;
+    unsigned long flags;
+    int level, grp, ix, i, j;
+
+    i = *(loff_t *) v;
+
+    switch (i) {
+    case 0:
+        seq_printf(p, "
+");
+        for (j = 0; j < NR_CPUS; j++)
+            if (cpu_online(j))
+                seq_printf(p, "CPU%d
+", j);
+
+        seq_putc(p, '\n');
+        break;
+
+    case 1 ... NR_IRQ_GROUPS * NR_IRQ_ACTIONS_PER_GROUP:
+        local_irq_save(flags);
+
+        grp = (i - 1) / NR_IRQ_ACTIONS_PER_GROUP;
+        group = irq_groups[grp];
+        if (!group)
+            goto skip;
+
+        ix = (i - 1) % NR_IRQ_ACTIONS_PER_GROUP;
+        action = group->actions[ix];
+        if (!action)
+            goto skip;
+
+        seq_printf(p, "%3d: ", i - 1);
+
+    #ifndef CONFIG_SMP
+        seq_printf(p, "%10u ", kstat_irqs(i));
+    #else
+        for (j = 0; j < NR_CPUS; j++)
+            if (cpu_online(j))
+                seq_printf(p, "%10u ", kstat_cpu(j).irqs[i - 1]);
+    #endif
+
+        level = group->sources[ix]->level - frv_irq_levels;
+
+        seq_printf(p, " %12s@%x", group->sources[ix]->muxname, level);
+        seq_printf(p, " %s", action->name);
+
+        for (action = action->next; action; action = action->next)
+            seq_printf(p, " %s", action->name);
+
+        seq_putc(p, '\n');
+    skip:
+        local_irq_restore(flags);
+        break;
+
+    case NR_IRQ_GROUPS * NR_IRQ_ACTIONS_PER_GROUP + 1:
+        seq_printf(p, "ERR: %10u\n", atomic_read(&irq_err_count));

```

```

+         break;
+
+     default:
+         break;
+     }
+
+     return 0;
+}
+
+/*
+ * Generic enable/disable code: this just calls
+ * down into the PIC-specific version for the actual
+ * hardware disable after having gotten the irq
+ * controller lock.
+ */
+/**
+ *     disable_irq_nosync - disable an irq without waiting
+ *     @irq: Interrupt to disable
+ *
+ *     Disable the selected interrupt line.  Disables and Enables are
+ *     nested.
+ *     Unlike disable_irq(), this function does not ensure existing
+ *     instances of the IRQ handler have completed before returning.
+ *
+ *     This function may be called from IRQ context.
+ */
+void disable_irq_nosync(unsigned int irq)
+{
+     struct irq_source *source;
+     struct irq_group *group;
+     struct irq_level *level;
+     unsigned long flags;
+     int idx = irq & (NR_IRQ_ACTIONS_PER_GROUP - 1);
+
+     group = irq_groups[irq >> NR_IRQ_LOG2_ACTIONS_PER_GROUP];
+     if (!group)
+         BUG();
+
+     source = group->sources[idx];
+     if (!source)
+         BUG();
+
+     level = source->level;
+
+     spin_lock_irqsave(&level->lock, flags);
+
+     if (group->control) {
+         if (!group->disable_cnt[idx]++)
+             group->control(group, idx, 0);
+     } else if (!level->disable_count++) {
+         __set_MASK(level - frv_irq_levels);
+     }
+
+     spin_unlock_irqrestore(&level->lock, flags);
+}
+/**
+ *     disable_irq - disable an irq and wait for completion
+ *     @irq: Interrupt to disable

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```
+ *
+ *   Disable the selected interrupt line.  Enables and Disables are
+ *   nested.
+ *   This function waits for any pending IRQ handlers for this interrupt
+ *   to complete before returning.  If you use this function while
+ *   holding a resource the IRQ handler may need you will deadlock.
+ *
+ *   This function may be called - with care - from IRQ context.
+ */
+
+void disable_irq(unsigned int irq)
+{
+    disable_irq_nosync(irq);
+
+    #ifdef CONFIG_SMP
+    if (!local_irq_count(smp_processor_id())) {
+        do {
+            barrier();
+        } while (irq_desc[irq].status & IRQ_INPROGRESS);
+    }
+    #endif
+}
+
+/**
+ *   enable_irq - enable handling of an irq
+ *   @irq: Interrupt to enable
+ *
+ *   Undoes the effect of one call to disable_irq().  If this
+ *   matches the last disable, processing of interrupts on this
+ *   IRQ line is re-enabled.
+ *
+ *   This function may be called from IRQ context.
+ */
+
+void enable_irq(unsigned int irq)
+{
+    struct irq_source *source;
+    struct irq_group *group;
+    struct irq_level *level;
+    unsigned long flags;
+    int idx = irq & (NR_IRQ_ACTIONS_PER_GROUP - 1);
+    int count;
+
+    group = irq_groups[irq >> NR_IRQ_LOG2_ACTIONS_PER_GROUP];
+    if (!group)
+        BUG();
+
+    source = group->sources[idx];
+    if (!source)
+        BUG();
+
+    level = source->level;
+
+    spin_lock_irqsave(&level->lock, flags);
+
+    if (group->control)
+        count = group->disable_cnt[idx];
+    else
+        count = level->disable_count;
+
+    switch (count) {
+    case 1:
```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+         if (group->control) {
+             if (group->actions[idx])
+                 group->control(group, idx, 1);
+         } else {
+             if (level->usage)
+                 __clr_MASK(level - frv_irq_levels);
+         }
+         /* fall-through */
+
+     default:
+         count--;
+         break;
+
+     case 0:
+         printk("enable_irq(%u) unbalanced from %p\n", irq, __builtin_return_address(0));
+     }
+
+     if (group->control)
+         group->disable_cnt[idx] = count;
+     else
+         level->disable_count = count;
+
+     spin_unlock_irqrestore(&level->lock, flags);
+ }
+
+ /*****
+ */
+ * handles all normal device IRQ's
+ * - registers are referred to by the __frame variable (GR28)
+ * - IRQ distribution is complicated in this arch because of the many PICs, the
+ *   way they work and the way they cascade
+ */
+asmlinkage void do_IRQ(void)
+{
+     struct irq_source *source;
+     int level, cpu;
+
+     level = (__frame->tbr >> 4) & 0xf;
+     cpu = smp_processor_id();
+
+ #if 0
+     {
+         static u32 irqcount;
+         *(volatile u32 *) 0xe1200004 = ~((irqcount++ << 8) | level);
+         *(volatile u16 *) 0xffc00100 = (u16) ~0x9999;
+         mb();
+     }
+ #endif
+
+     if ((unsigned long) __frame - (unsigned long) (current + 1) < 512)
+         BUG();
+
+     __set_MASK(level);
+     __clr_RC(level);
+     __clr_IRL();
+
+     kstat_this_cpu.irqs[level]++;
+
+     irq_enter();
+
+     for (source = frv_irq_levels[level].sources; source; source = source->next)
+         source->doirq(source);

```


Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```
+ * If your interrupt is shared you must pass a non NULL dev_id
+ * as this is required when freeing the interrupt.
+ *
+ * Flags:
+ *
+ * SA_SHIRQ           Interrupt is shared
+ *
+ * SA_INTERRUPT      Disable local interrupts while processing
+ *
+ * SA_SAMPLE_RANDOM  The interrupt can be used for entropy
+ *
+ */
+
+int request_irq(unsigned int irq,
+               irqreturn_t (*handler)(int, void *, struct pt_regs *),
+               unsigned long irqflags,
+               const char * devname,
+               void *dev_id)
+{
+   int retval;
+   struct irqaction *action;
+
+   #if 1
+   /*
+    * Sanity-check: shared interrupts should REALLY pass in
+    * a real dev-ID, otherwise we'll have trouble later trying
+    * to figure out which interrupt is which (messes up the
+    * interrupt freeing logic etc).
+    */
+   if (irqflags & SA_SHIRQ) {
+       if (!dev_id)
+           printk("Bad boy: %s (at 0x%x) called us without a dev_id!\n",
+                  devname, (&irq)[-1]);
+   }
+   #endif
+
+   if ((irq >> NR_IRQ_LOG2_ACTIONS_PER_GROUP) >= NR_IRQ_GROUPS)
+       return -EINVAL;
+   if (!handler)
+       return -EINVAL;
+
+   action = (struct irqaction *) kmalloc(sizeof(struct irqaction), GFP_KERNEL);
+   if (!action)
+       return -ENOMEM;
+
+   action->handler = handler;
+   action->flags = irqflags;
+   action->mask = CPU_MASK_NONE;
+   action->name = devname;
+   action->next = NULL;
+   action->dev_id = dev_id;
+
+   retval = setup_irq(irq, action);
+   if (retval)
+       kfree(action);
+   return retval;
+}
+
+/**
+ * free_irq - free an interrupt
+ * @irq: Interrupt line to free
+ * @dev_id: Device identity to free
```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```
+ *
+ * Remove an interrupt handler. The handler is removed and if the
+ * interrupt line is no longer in use by any driver it is disabled.
+ * On a shared IRQ the caller must ensure the interrupt is disabled
+ * on the card it drives before calling this function. The function
+ * does not return until any executing interrupts for this IRQ
+ * have completed.
+ *
+ * This function may be called from interrupt context.
+ *
+ * Bugs: Attempting to free an irq in a handler for the same irq hangs
+ * the machine.
+ */
+
+void free_irq(unsigned int irq, void *dev_id)
+{
+    struct irq_source *source;
+    struct irq_group *group;
+    struct irq_level *level;
+    struct irqaction **p, **pp;
+    unsigned long flags;
+
+    if ((irq >> NR_IRQ_LOG2_ACTIONS_PER_GROUP) >= NR_IRQ_GROUPS)
+        return;
+
+    group = irq_groups[irq >> NR_IRQ_LOG2_ACTIONS_PER_GROUP];
+    if (!group)
+        BUG();
+
+    source = group->sources[irq & (NR_IRQ_ACTIONS_PER_GROUP - 1)];
+    if (!source)
+        BUG();
+
+    level = source->level;
+    p = &group->actions[irq & (NR_IRQ_ACTIONS_PER_GROUP - 1)];
+
+    spin_lock_irqsave(&level->lock, flags);
+
+    for (pp = p; *pp; pp = &(*pp)->next) {
+        struct irqaction *action = *pp;
+
+        if (action->dev_id != dev_id)
+            continue;
+
+        /* found it - remove from the list of entries */
+        *pp = action->next;
+
+        level->usage--;
+
+        if (p == pp && group->control)
+            group->control(group, irq & (NR_IRQ_ACTIONS_PER_GROUP - 1), 0);
+
+        if (level->usage == 0)
+            __set_MASK(level - frv_irq_levels);
+
+        spin_unlock_irqrestore(&level->lock, flags);
+
+    }
+
+    #ifdef CONFIG_SMP
+        /* Wait to make sure it's not being used on another CPU */
+        while (desc->status & IRQ_INPROGRESS)
+            barrier();
+    #endif
+}
+
+endif
```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+         kfree(action);
+         return;
+     }
+ }
+
+ /*
+  * IRQ autodetection code..
+  *
+  * This depends on the fact that any interrupt that comes in on to an
+  * unassigned IRQ will cause GxICR_DETECT to be set
+  */
+
+static DECLARE_MUTEX(probe_sem);
+
+ /**
+  * probe_irq_on    - begin an interrupt autodetect
+  *
+  * Commence probing for an interrupt. The interrupts are scanned
+  * and a mask of potential interrupt lines is returned.
+  */
+
+unsigned long probe_irq_on(void)
+{
+    down(&probe_sem);
+    return 0;
+}
+
+ /**
+  * Return a mask of triggered interrupts (this
+  * can handle only legacy ISA interrupts).
+  */
+
+ /**
+  * probe_irq_mask - scan a bitmap of interrupt lines
+  * @val:    mask of interrupts to consider
+  *
+  * Scan the ISA bus interrupt lines and return a bitmap of
+  * active interrupts. The interrupt probe logic state is then
+  * returned to its previous value.
+  *
+  * Note: we need to scan all the irq's even though we will
+  * only return ISA irq numbers - just so that we reset them
+  * all to a known state.
+  */
+
+unsigned int probe_irq_mask(unsigned long xmask)
+{
+    up(&probe_sem);
+    return 0;
+}
+
+ /**
+  * Return the one interrupt that triggered (this can
+  * handle any interrupt source).
+  */
+
+ /**
+  * probe_irq_off    - end an interrupt autodetect
+  * @xmask: mask of potential interrupts (unused)
+  *
+  * Scans the unused interrupt lines and returns the line which
+  * appears to have triggered the interrupt. If no interrupt was

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```
+ * found then zero is returned. If more than one interrupt is
+ * found then minus the first candidate is returned to indicate
+ * their is doubt.
+ *
+ * The interrupt probe logic state is returned to its previous
+ * value.
+ *
+ * BUGS: When used in a module (which arguably shouldnt happen)
+ * nothing prevents two IRQ probe callers from overlapping. The
+ * results of this are non-optimal.
+ */
+
+int probe_irq_off(unsigned long xmask)
+{
+    up(&probe_sem);
+    return -1;
+}
+
+/* this was setup_x86_irq but it seems pretty generic */
+int setup_irq(unsigned int irq, struct irqaction *new)
+{
+    struct irq_source *source;
+    struct irq_group *group;
+    struct irq_level *level;
+    struct irqaction **p, **pp;
+    unsigned long flags;
+
+    group = irq_groups[irq >> NR_IRQ_LOG2_ACTIONS_PER_GROUP];
+    if (!group)
+        BUG();
+
+    source = group->sources[irq & (NR_IRQ_ACTIONS_PER_GROUP - 1)];
+    if (!source)
+        BUG();
+
+    level = source->level;
+
+    p = &group->actions[irq & (NR_IRQ_ACTIONS_PER_GROUP - 1)];
+
+    /*
+     * Some drivers like serial.c use request_irq() heavily,
+     * so we have to be careful not to interfere with a
+     * running system.
+     */
+    if (new->flags & SA_SAMPLE_RANDOM) {
+        /*
+         * This function might sleep, we want to call it first,
+         * outside of the atomic block.
+         * Yes, this might clear the entropy pool if the wrong
+         * driver is attempted to be loaded, without actually
+         * installing a new handler, but is this really a problem,
+         * only the sysadmin is able to do this.
+         */
+        rand_initialize_irq(irq);
+    }
+
+    /* must juggle the interrupt processing stuff with interrupts disabled */
+    spin_lock_irqsave(&level->lock, flags);
+
+    /* can't share interrupts unless all parties agree to */
+    if (level->usage != 0 && !(level->flags & new->flags & SA_SHIRQ)) {
+        spin_unlock_irqrestore(&level->lock, flags);
```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+         return -EBUSY;
+     }
+
+     /* add new interrupt at end of irq queue */
+     pp = p;
+     while (*pp)
+         pp = &(*pp)->next;
+
+     *pp = new;
+
+     level->usage++;
+     level->flags = new->flags;
+
+     /* turn the interrupts on */
+     if (level->usage == 1)
+         __clr_MASK(level - frv_irq_levels);
+
+     if (p == pp && group->control)
+         group->control(group, irq & (NR_IRQ_ACTIONS_PER_GROUP - 1), 1);
+
+     spin_unlock_irqrestore(&level->lock, flags);
+     register_irq_proc(irq);
+     return 0;
+ }
+
+static struct proc_dir_entry * root_irq_dir;
+static struct proc_dir_entry * irq_dir [NR_IRQS];
+
+#define HEX_DIGITS 8
+
+static unsigned int parse_hex_value (const char *buffer,
+                                     unsigned long count, unsigned long *ret)
+{
+     unsigned char hexnum [HEX_DIGITS];
+     unsigned long value;
+     int i;
+
+     if (!count)
+         return -EINVAL;
+     if (count > HEX_DIGITS)
+         count = HEX_DIGITS;
+     if (copy_from_user(hexnum, buffer, count))
+         return -EFAULT;
+
+     /*
+      * Parse the first 8 characters as a hex string, any non-hex char
+      * is end-of-string. '00e1', 'e1', '00E1', 'E1' are all the same.
+      */
+     value = 0;
+
+     for (i = 0; i < count; i++) {
+         unsigned int c = hexnum[i];
+
+         switch (c) {
+             case '0' ... '9': c -= '0'; break;
+             case 'a' ... 'f': c -= 'a'-10; break;
+             case 'A' ... 'F': c -= 'A'-10; break;
+             default:
+                 goto out;
+         }
+         value = (value << 4) | c;
+     }
+ }

```

```

+out:
+    *ret = value;
+    return 0;
+}
+
+static int prof_cpu_mask_read_proc (char *page, char **start, off_t off,
+    int count, int *eof, void *data)
+{
+    unsigned long *mask = (unsigned long *) data;
+    if (count < HEX_DIGITS+1)
+        return -EINVAL;
+    return sprintf (page, "%08lx\n", *mask);
+}
+
+static int prof_cpu_mask_write_proc (struct file *file, const char *buffer,
+    unsigned long count, void *data)
+{
+    unsigned long *mask = (unsigned long *) data, full_count = count, err;
+    unsigned long new_value;
+
+    show_state();
+    err = parse_hex_value(buffer, count, &new_value);
+    if (err)
+        return err;
+
+    *mask = new_value;
+    return full_count;
+}
+
+#define MAX_NAMELEN 10
+
+static void register_irq_proc (unsigned int irq)
+{
+    char name [MAX_NAMELEN];
+
+    if (!root_irq_dir || irq_dir[irq])
+        return;
+
+    memset(name, 0, MAX_NAMELEN);
+    sprintf(name, "%d", irq);
+
+    /* create /proc/irq/1234 */
+    irq_dir[irq] = proc_mkdir(name, root_irq_dir);
+}
+
+unsigned long prof_cpu_mask = -1;
+
+void init_irq_proc (void)
+{
+    struct proc_dir_entry *entry;
+    int i;
+
+    /* create /proc/irq */
+    root_irq_dir = proc_mkdir("irq", 0);
+
+    /* create /proc/irq/prof_cpu_mask */
+    entry = create_proc_entry("prof_cpu_mask", 0600, root_irq_dir);
+    if (!entry)
+        return;
+
+    entry->nlink = 1;

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+     entry->data = (void *)&prof_cpu_mask;
+     entry->read_proc = prof_cpu_mask_read_proc;
+     entry->write_proc = prof_cpu_mask_write_proc;
+
+     /*
+      * Create entries for all existing IRQs.
+      */
+     for (i = 0; i < NR_IRQS; i++)
+         register_irq_proc(i);
+ }
+
+ /*****
+ * initialise the interrupt system
+ */
+void __init init_IRQ(void)
+{
+     route_cpu_irqs();
+     fpga_init();
+#ifdef CONFIG_FUJITSU_MB93493
+     route_mb93493_irqs();
+#endif
+} /* end init_IRQ() */
diff -uNrp /warthog/kernels/linux-2.6.10-rc1-mm3/arch/frv/kernel/irq-mb93091.c linux-2.6.10-rc1-mm3/arch/frv/kernel/irq-mb93091.c 1970-01-01 01:00:00.000000
+++ linux-2.6.10-rc1-mm3-frv/arch/frv/kernel/irq-mb93091.c      2004-11-05 14:13:03.151559284 +0000
@@ -0,0 +1,116 @@
+/* irq-mb93091.c: MB93091 FPGA interrupt handling
+ *
+ * Copyright (C) 2003 Red Hat, Inc. All Rights Reserved.
+ * Written by David Howells (dhowells@redhat.com)
+ *
+ * This program is free software; you can redistribute it and/or
+ * modify it under the terms of the GNU General Public License
+ * as published by the Free Software Foundation; either version
+ * 2 of the License, or (at your option) any later version.
+ */
+
+#include <linux/config.h>
+#include <linux/ptrace.h>
+#include <linux/errno.h>
+#include <linux/signal.h>
+#include <linux/sched.h>
+#include <linux/ioport.h>
+#include <linux/interrupt.h>
+#include <linux/init.h>
+#include <linux/irq.h>
+
+#include <asm/io.h>
+#include <asm/system.h>
+#include <asm/bitops.h>
+#include <asm/delay.h>
+#include <asm/irq.h>
+#include <asm/irc-regs.h>
+#include <asm/irq-routing.h>
+
+#define __reg16(ADDR) (*(volatile unsigned short *)(ADDR))
+
+#define __get_IMR()    ({ __reg16(0xffc00004); })
+#define __set_IMR(M)  do { __reg16(0xffc00004) = (M); wmb(); } while(0)
+#define __get_IFR()    ({ __reg16(0xffc0000c); })
+#define __clr_IFR(M)  do { __reg16(0xffc0000c) = ~(M); wmb(); } while(0)

```

```

+
+static void frv_fpga_doirq(struct irq_source *source);
+static void frv_fpga_control(struct irq_group *group, int irq, int on);
+
+/*
+*****/
+/*
+ * FPGA IRQ multiplexor
+ */
+static struct irq_source frv_fpga[4] = {
+#define __FPGA(X, M)
+    [X] = {
+        .muxname      = "fpga."#X,
+        .irqmask      = M,
+        .doirq        = frv_fpga_doirq,
+    }
+
+    __FPGA(0, 0x0028),
+    __FPGA(1, 0x0050),
+    __FPGA(2, 0x1c00),
+    __FPGA(3, 0x6386),
+};
+
+static struct irq_group frv_fpga_irqs = {
+    .first_irq      = IRQ_BASE_FPGA,
+    .control        = frv_fpga_control,
+    .sources = {
+        [ 1] = &frv_fpga[3],
+        [ 2] = &frv_fpga[3],
+        [ 3] = &frv_fpga[0],
+        [ 4] = &frv_fpga[1],
+        [ 5] = &frv_fpga[0],
+        [ 6] = &frv_fpga[1],
+        [ 7] = &frv_fpga[3],
+        [ 8] = &frv_fpga[3],
+        [ 9] = &frv_fpga[3],
+        [10] = &frv_fpga[2],
+        [11] = &frv_fpga[2],
+        [12] = &frv_fpga[2],
+        [13] = &frv_fpga[3],
+        [14] = &frv_fpga[3],
+    },
+};
+
+static void frv_fpga_control(struct irq_group *group, int index, int on)
+{
+    uint16_t imr = __get_IMR();
+
+    if (on)
+        imr &= ~(1 << index);
+    else
+        imr |= 1 << index;
+
+    __set_IMR(imr);
+}
+
+static void frv_fpga_doirq(struct irq_source *source)
+{
+    uint16_t mask, imr;
+
+    imr = __get_IMR();
+    mask = source->irqmask & ~imr & __get_IFR();

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+     if (mask) {
+         __set_IMR(imr | mask);
+         __clr_IFR(mask);
+         distribute_irqs(&frv_fpga_irqs, mask);
+         __set_IMR(imr);
+     }
+ }
+
+void __init fpga_init(void)
+{
+     __set_IMR(0x7ffe);
+     __clr_IFR(0x0000);
+
+     frv_irq_route_external(&frv_fpga[0], IRQ_CPU_EXTERNAL0);
+     frv_irq_route_external(&frv_fpga[1], IRQ_CPU_EXTERNAL1);
+     frv_irq_route_external(&frv_fpga[2], IRQ_CPU_EXTERNAL2);
+     frv_irq_route_external(&frv_fpga[3], IRQ_CPU_EXTERNAL3);
+     frv_irq_set_group(&frv_fpga_irqs);
+ }
diff -uNrp /warthog/kernels/linux-2.6.10-rc1-mm3/arch/frv/kernel/irq-mb93093.c linux-2.6.10-rc1-mm3
--- /warthog/kernels/linux-2.6.10-rc1-mm3/arch/frv/kernel/irq-mb93093.c 1970-01-01 01:00:00.000000
+++ linux-2.6.10-rc1-mm3-frv/arch/frv/kernel/irq-mb93093.c      2004-11-05 14:13:03.000000000 +0000
@@ -0,0 +1,99 @@
+/* irq-mb93093.c: MB93093 FPGA interrupt handling
+ *
+ * Copyright (C) 2004 Red Hat, Inc. All Rights Reserved.
+ * Written by David Howells (dhowells@redhat.com)
+ *
+ * This program is free software; you can redistribute it and/or
+ * modify it under the terms of the GNU General Public License
+ * as published by the Free Software Foundation; either version
+ * 2 of the License, or (at your option) any later version.
+ */
+
+#include <linux/config.h>
+#include <linux/ptrace.h>
+#include <linux/errno.h>
+#include <linux/signal.h>
+#include <linux/sched.h>
+#include <linux/ioport.h>
+#include <linux/interrupt.h>
+#include <linux/init.h>
+#include <linux/irq.h>
+
+#include <asm/io.h>
+#include <asm/system.h>
+#include <asm/bitops.h>
+#include <asm/delay.h>
+#include <asm/irq.h>
+#include <asm/irc-regs.h>
+#include <asm/irq-routing.h>
+
+#define __reg16(ADDR) (*(volatile unsigned short *)(__region_CS2 + (ADDR)))
+
+#define __get_IMR()     ({ __reg16(0x0a); })
+#define __set_IMR(M)   do { __reg16(0x0a) = (M); wmb(); } while(0)
+#define __get_IFR()     ({ __reg16(0x02); })
+#define __clr_IFR(M)   do { __reg16(0x02) = ~(M); wmb(); } while(0)
+
+static void frv_fpga_doirq(struct irq_source *source);
+static void frv_fpga_control(struct irq_group *group, int irq, int on);
+

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+ /*****
+ */
+ * FPGA IRQ multiplexor
+ */
+static struct irq_source frv_fpga[4] = {
+#define __FPGA(X, M)                                     \
+    [X] = {                                           \
+        .muxname      = "fpga."#X,                   \
+        .irqmask      = M,                           \
+        .doirq        = frv_fpga_doirq,              \
+    }
+
+    __FPGA(0, 0x0700),
+};
+
+static struct irq_group frv_fpga_irqs = {
+    .first_irq      = IRQ_BASE_FPGA,
+    .control        = frv_fpga_control,
+    .sources = {
+        [ 8] = &frv_fpga[0],
+        [ 9] = &frv_fpga[0],
+        [10] = &frv_fpga[0],
+    },
+};
+
+static void frv_fpga_control(struct irq_group *group, int index, int on)
+{
+    uint16_t imr = __get_IMR();
+
+    if (on)
+        imr &= ~(1 << index);
+    else
+        imr |= 1 << index;
+
+    __set_IMR(imr);
+}
+
+static void frv_fpga_doirq(struct irq_source *source)
+{
+    uint16_t mask, imr;
+
+    imr = __get_IMR();
+    mask = source->irqmask & ~imr & __get_IFR();
+    if (mask) {
+        __set_IMR(imr | mask);
+        __clr_IFR(mask);
+        distribute_irqs(&frv_fpga_irqs, mask);
+        __set_IMR(imr);
+    }
+}
+
+void __init fpga_init(void)
+{
+    __set_IMR(0x0700);
+    __clr_IFR(0x0000);
+
+    frv_irq_route_external(&frv_fpga[0], IRQ_CPU_EXTERNAL2);
+    frv_irq_set_group(&frv_fpga_irqs);
+}
diff -uNrp /warthog/kernels/linux-2.6.10-rc1-mm3/arch/frv/kernel/irq-mb93493.c linux-2.6.10-rc1-mm3/arch/frv/kernel/irq-mb93493.c 1970-01-01 01:00:00.000000

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+++ linux-2.6.10-rc1-mm3-frv/arch/frv/kernel/irq-mb93493.c      2004-11-05 14:13:03.000000000 +00
@@ -0,0 +1,108 @@
+/* irq-mb93493.c: MB93493 companion chip interrupt handler
+ *
+ * Copyright (C) 2004 Red Hat, Inc. All Rights Reserved.
+ * Written by David Howells (dhowells@redhat.com)
+ *
+ * This program is free software; you can redistribute it and/or
+ * modify it under the terms of the GNU General Public License
+ * as published by the Free Software Foundation; either version
+ * 2 of the License, or (at your option) any later version.
+ */
+
+#include <linux/config.h>
+#include <linux/ptrace.h>
+#include <linux/errno.h>
+#include <linux/signal.h>
+#include <linux/sched.h>
+#include <linux/ioport.h>
+#include <linux/interrupt.h>
+#include <linux/init.h>
+#include <linux/irq.h>
+
+#include <asm/io.h>
+#include <asm/system.h>
+#include <asm/bitops.h>
+#include <asm/delay.h>
+#include <asm/irq.h>
+#include <asm/irc-regs.h>
+#include <asm/irq-routing.h>
+#include <asm/mb93493-irqs.h>
+
+static void frv_mb93493_doirq(struct irq_source *source);
+
+/*
+ * MB93493 companion chip IRQ multiplexor
+ */
+static struct irq_source frv_mb93493[2] = {
+    [0] = {
+        .muxname      = "mb93493.0",
+        .muxdata      = __region_CS3 + 0x3d0,
+        .doirq        = frv_mb93493_doirq,
+        .irqmask      = 0x0000,
+    },
+    [1] = {
+        .muxname      = "mb93493.1",
+        .muxdata      = __region_CS3 + 0x3d4,
+        .doirq        = frv_mb93493_doirq,
+        .irqmask      = 0x0000,
+    },
+};
+
+static void frv_mb93493_control(struct irq_group *group, int index, int on)
+{
+    struct irq_source *source;
+    uint32_t iqsr;
+
+    if ((frv_mb93493[0].irqmask & (1 << index)))
+        source = &frv_mb93493[0];
+    else
+        source = &frv_mb93493[1];

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+
+   iqsr = readl(source->muxdata);
+   if (on)
+       iqsr |= 1 << (index + 16);
+   else
+       iqsr &= ~(1 << (index + 16));
+
+   writel(iqsr, source->muxdata);
+}
+
+static struct irq_group frv_mb93493_irqs = {
+   .first_irq      = IRQ_BASE_MB93493,
+   .control        = frv_mb93493_control,
+};
+
+static void frv_mb93493_doirq(struct irq_source *source)
+{
+   uint32_t mask = readl(source->muxdata);
+   mask = mask & (mask >> 16) & 0xffff;
+
+   if (mask)
+       distribute_irqs(&frv_mb93493_irqs, mask);
+}
+
+static void __init mb93493_irq_route(int irq, int source)
+{
+   frv_mb93493[source].irqmask |= 1 << (irq - IRQ_BASE_MB93493);
+   frv_mb93493_irqs.sources[irq - IRQ_BASE_MB93493] = &frv_mb93493[source];
+}
+
+void __init route_mb93493_irqs(void)
+{
+   frv_irq_route_external(&frv_mb93493[0], IRQ_CPU_MB93493_0);
+   frv_irq_route_external(&frv_mb93493[1], IRQ_CPU_MB93493_1);
+
+   frv_irq_set_group(&frv_mb93493_irqs);
+
+   mb93493_irq_route(IRQ_MB93493_VDC,          IRQ_MB93493_VDC_ROUTE);
+   mb93493_irq_route(IRQ_MB93493_VCC,          IRQ_MB93493_VCC_ROUTE);
+   mb93493_irq_route(IRQ_MB93493_AUDIO_IN,     IRQ_MB93493_AUDIO_IN_ROUTE);
+   mb93493_irq_route(IRQ_MB93493_I2C_0,        IRQ_MB93493_I2C_0_ROUTE);
+   mb93493_irq_route(IRQ_MB93493_I2C_1,        IRQ_MB93493_I2C_1_ROUTE);
+   mb93493_irq_route(IRQ_MB93493_USB,          IRQ_MB93493_USB_ROUTE);
+   mb93493_irq_route(IRQ_MB93493_LOCAL_BUS,    IRQ_MB93493_LOCAL_BUS_ROUTE);
+   mb93493_irq_route(IRQ_MB93493_PCMCIA,       IRQ_MB93493_PCMCIA_ROUTE);
+   mb93493_irq_route(IRQ_MB93493_GPIO,         IRQ_MB93493_GPIO_ROUTE);
+   mb93493_irq_route(IRQ_MB93493_AUDIO_OUT,    IRQ_MB93493_AUDIO_OUT_ROUTE);
+}
diff -uNrp /warthog/kernels/linux-2.6.10-rc1-mm3/arch/frv/kernel/irq-routing.c linux-2.6.10-rc1-mm3
--- /warthog/kernels/linux-2.6.10-rc1-mm3/arch/frv/kernel/irq-routing.c 1970-01-01 01:00:00.00000
+++ linux-2.6.10-rc1-mm3-frv/arch/frv/kernel/irq-routing.c      2004-11-05 14:13:03.163558271 +00
@@ -0,0 +1,291 @@
+/* irq-routing.c: IRQ routing
+ *
+ * Copyright (C) 2004 Red Hat, Inc. All Rights Reserved.
+ * Written by David Howells (dhowells@redhat.com)
+ *
+ * This program is free software; you can redistribute it and/or
+ * modify it under the terms of the GNU General Public License
+ * as published by the Free Software Foundation; either version
+ * 2 of the License, or (at your option) any later version.
+ */

```

```

+
+#include <linux/sched.h>
+#include <linux/random.h>
+#include <linux/init.h>
+#include <linux/serial_reg.h>
+#include <asm/io.h>
+#include <asm/irq-routing.h>
+#include <asm/irc-regs.h>
+#include <asm/serial-regs.h>
+#include <asm/dma.h>
+
+struct irq_level frv_irq_levels[16] = {
+    [0 ... 15] = {
+        .lock    = SPIN_LOCK_UNLOCKED,
+    }
+};
+
+struct irq_group *irq_groups[NR_IRQ_GROUPS];
+
+extern struct irq_group frv_cpu_irqs;
+
+void __init frv_irq_route(struct irq_source *source, int irqlevel)
+{
+    source->level = &frv_irq_levels[irqlevel];
+    source->next = frv_irq_levels[irqlevel].sources;
+    frv_irq_levels[irqlevel].sources = source;
+}
+
+void __init frv_irq_route_external(struct irq_source *source, int irq)
+{
+    int irqlevel = 0;
+
+    switch (irq) {
+    case IRQ_CPU_EXTERNAL0: irqlevel = IRQ_XIRQ0_LEVEL; break;
+    case IRQ_CPU_EXTERNAL1: irqlevel = IRQ_XIRQ1_LEVEL; break;
+    case IRQ_CPU_EXTERNAL2: irqlevel = IRQ_XIRQ2_LEVEL; break;
+    case IRQ_CPU_EXTERNAL3: irqlevel = IRQ_XIRQ3_LEVEL; break;
+    case IRQ_CPU_EXTERNAL4: irqlevel = IRQ_XIRQ4_LEVEL; break;
+    case IRQ_CPU_EXTERNAL5: irqlevel = IRQ_XIRQ5_LEVEL; break;
+    case IRQ_CPU_EXTERNAL6: irqlevel = IRQ_XIRQ6_LEVEL; break;
+    case IRQ_CPU_EXTERNAL7: irqlevel = IRQ_XIRQ7_LEVEL; break;
+    default: BUG();
+    }
+
+    source->level = &frv_irq_levels[irqlevel];
+    source->next = frv_irq_levels[irqlevel].sources;
+    frv_irq_levels[irqlevel].sources = source;
+}
+
+void __init frv_irq_set_group(struct irq_group *group)
+{
+    irq_groups[group->first_irq >> NR_IRQ_LOG2_ACTIONS_PER_GROUP] = group;
+}
+
+void distribute_irqs(struct irq_group *group, unsigned long irqmask)
+{
+    struct irqaction *action;
+    int irq;
+
+    while (irqmask) {
+        asm("scan %1,gr0,%0" : "=r"(irq) : "r"(irqmask));
+        if (irq < 0 || irq > 31)

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+         asm volatile("break");
+         irq = 31 - irq;
+
+         irqmask &= ~(1 << irq);
+         action = group->actions[irq];
+
+         irq += group->first_irq;
+
+         if (action) {
+             int status = 0;
+
+             /*
+             if (!(action->flags & SA_INTERRUPT))
+                 sti();
+
+             do {
+                 status |= action->flags;
+                 action->handler(irq, action->dev_id, __frame);
+                 action = action->next;
+             } while (action);
+
+             if (status & SA_SAMPLE_RANDOM)
+                 add_interrupt_randomness(irq);
+             cli();
+         }
+     }
+ }
+ }
+
+ /*****
+ * CPU UART interrupts
+ */
+static void frv_cpufreq_doirq(struct irq_source *source)
+{
+    uint8_t iir = readb(source->muxdata + UART_IIR * 8);
+    if ((iir & 0x0f) != UART_IIR_NO_INT)
+        distribute_irqs(&frv_cpu_irqs, source->irqmask);
+}
+
+struct irq_source frv_cpufreq[2] = {
+#define __CPUUART(X, A)
+    [X] = {
+        .muxname      = "uart",
+        .muxdata      = (volatile void __iomem *) A,
+        .irqmask      = 1 << IRQ_CPU_UART##X,
+        .doirq        = frv_cpufreq_doirq,
+    }
+
+    __CPUUART(0, UART0_BASE),
+    __CPUUART(1, UART1_BASE),
+};
+
+ /*****
+ * CPU DMA interrupts
+ */
+static void frv_cpudma_doirq(struct irq_source *source)
+{
+    uint32_t cstr = readl(source->muxdata + DMAC_CSTRx);
+    if (cstr & DMAC_CSTRx_INT)
+        distribute_irqs(&frv_cpu_irqs, source->irqmask);
+}
+

```

Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+struct irq_source frv_cpudma[8] = {
+#define __CPUDMA(X, A)                                     \
+    [X] = {                                              \
+        .muxname      = "dma",                          \
+        .muxdata      = (volatile void __iomem *) A,    \
+        .irqmask      = 1 << IRQ_CPU_DMA##X,           \
+        .doirq        = frv_cpudma_doirq,              \
+    }
+
+    __CPUDMA(0, 0xfe000900),
+    __CPUDMA(1, 0xfe000980),
+    __CPUDMA(2, 0xfe000a00),
+    __CPUDMA(3, 0xfe000a80),
+    __CPUDMA(4, 0xfe001000),
+    __CPUDMA(5, 0xfe001080),
+    __CPUDMA(6, 0xfe001100),
+    __CPUDMA(7, 0xfe001180),
+};
+
+/*
+ * CPU timer interrupts - can't tell whether they've generated an interrupt or not
+ */
+static void frv_cputimer_doirq(struct irq_source *source)
+{
+    distribute_irqs(&frv_cpu_irqs, source->irqmask);
+}
+
+struct irq_source frv_cputimer[3] = {
+#define __CPUTIMER(X)                                     \
+    [X] = {                                              \
+        .muxname      = "timer",                        \
+        .muxdata      = 0,                              \
+        .irqmask      = 1 << IRQ_CPU_TIMER##X,         \
+        .doirq        = frv_cputimer_doirq,            \
+    }
+
+    __CPUTIMER(0),
+    __CPUTIMER(1),
+    __CPUTIMER(2),
+};
+
+/*
+ * external CPU interrupts - can't tell directly whether they've generated an interrupt or not
+ */
+static void frv_cpueexternal_doirq(struct irq_source *source)
+{
+    distribute_irqs(&frv_cpu_irqs, source->irqmask);
+}
+
+struct irq_source frv_cpueexternal[8] = {
+#define __CPUEEXTERNAL(X)                                 \
+    [X] = {                                              \
+        .muxname      = "ext",                          \
+        .muxdata      = 0,                              \
+        .irqmask      = 1 << IRQ_CPU_EXTERNAL##X,     \
+        .doirq        = frv_cpueexternal_doirq,       \
+    }
+
+    __CPUEEXTERNAL(0),
+    __CPUEEXTERNAL(1),

```


Linux-Kernel: [PATCH 6/20] FRV: Fujitsu FR-V CPU arch implementation part 4

```

+
+   set_IRR(4, IRQ_DMA3_LEVEL, IRQ_DMA2_LEVEL, IRQ_DMA1_LEVEL, IRQ_DMA0_LEVEL);
+   set_IRR(7, IRQ_DMA7_LEVEL, IRQ_DMA6_LEVEL, IRQ_DMA5_LEVEL, IRQ_DMA4_LEVEL);
+
+   /* route timer interrupts */
+   frv_irq_route(&frv_cputimer[0], IRQ_TIMER0_LEVEL);
+   frv_irq_route(&frv_cputimer[1], IRQ_TIMER1_LEVEL);
+   frv_irq_route(&frv_cputimer[2], IRQ_TIMER2_LEVEL);
+
+   set_IRR(5, 0, IRQ_TIMER2_LEVEL, IRQ_TIMER1_LEVEL, IRQ_TIMER0_LEVEL);
+
+   /* route external interrupts */
+   frv_irq_route(&frv_cpueexternal[0], IRQ_XIRQ0_LEVEL);
+   frv_irq_route(&frv_cpueexternal[1], IRQ_XIRQ1_LEVEL);
+   frv_irq_route(&frv_cpueexternal[2], IRQ_XIRQ2_LEVEL);
+   frv_irq_route(&frv_cpueexternal[3], IRQ_XIRQ3_LEVEL);
+   frv_irq_route(&frv_cpueexternal[4], IRQ_XIRQ4_LEVEL);
+   frv_irq_route(&frv_cpueexternal[5], IRQ_XIRQ5_LEVEL);
+   frv_irq_route(&frv_cpueexternal[6], IRQ_XIRQ6_LEVEL);
+   frv_irq_route(&frv_cpueexternal[7], IRQ_XIRQ7_LEVEL);
+
+   set_IRR(2, IRQ_XIRQ7_LEVEL, IRQ_XIRQ6_LEVEL, IRQ_XIRQ5_LEVEL, IRQ_XIRQ4_LEVEL);
+   set_IRR(3, IRQ_XIRQ3_LEVEL, IRQ_XIRQ2_LEVEL, IRQ_XIRQ1_LEVEL, IRQ_XIRQ0_LEVEL);
+
+#if defined(CONFIG_MB93091_VDK)
+   __set_TM1(0x55550000);           /* XIRQ7-0 all active low */
+#elif defined(CONFIG_MB93093_PDK)
+   __set_TM1(0x15550000);           /* XIRQ7 active high, 6-0 all active low */
+#else
+#error dont know external IRQ trigger levels for this setup
+#endif
+
+} /* end route_cpu_irqs() */
diff -uNrp /warthog/kernels/linux-2.6.10-rc1-mm3/arch/frv/kernel/kernel_thread.S linux-2.6.10-rc1
--- /warthog/kernels/linux-2.6.10-rc1-mm3/arch/frv/kernel/kernel_thread.S      1970-01-01 01:00:
+++ linux-2.6.10-rc1-mm3-frv/arch/frv/kernel/kernel_thread.S      2004-11-05 14:13:03.171557595 +00
@@ -0,0 +1,77 @@
+/* kernel_thread.S: kernel thread creation
+ *
+ * Copyright (C) 2003 Red Hat, Inc. All Rights Reserved.
+ * Written by David Howells (dhowells@redhat.com)
+ *
+ * This program is free software; you can redistribute it and/or
+ * modify it under the terms of the GNU General Public License
+ * as published by the Free Software Foundation; either version
+ * 2 of the License, or (at your option) any later version.
+ */
+
+#include <linux/linkage.h>
+#include <asm/unistd.h>
+
+#define CLONE_VM      0x00000100      /* set if VM shared between processes */
+#define      KERN_ERR      "<3>"
+
+   .section .rodata
+kernel_thread_emsg:
+   .asciz KERN_ERR "failed to create kernel thread: error=%d\n"
+
+   .text
+   .balign      4
+
+#####

```

```

+#
+# Create a kernel thread
+#
+# int kernel_thread(int (*fn)(void *), void * arg, unsigned long flags)
+#
+#####
+      .globl      kernel_thread
+      .type       kernel_thread,@function
+kernel_thread:
+      or.p        gr8,gr0,gr4
+      or          gr9,gr0,gr5
+
+      # start by forking the current process, but with shared VM
+      setlos.p    #__NR_clone,gr7      ; syscall number
+      ori         gr10,#CLONE_VM,gr8   ; first syscall a

```