```
**********************************************************
   41938 Thu Mar 14 21:19:00 2019
new/usr/src/uts/intel/pcbe/core_pcbe.c
10146 core_pcbe_event_coverage() is missing an else
Reviewed by: Andy Stormont <astormont@racktopsystems.com>
Reviewed by: Toomas Soome <tsoome@me.com>
Reviewed by: Gerg   Doma <domag02@gmail.com>
Approved by: Richard Lowe <richlowe@richlowe.net>
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright 2019 Joyent, Inc.
 * Copyright 2018 Joyent, Inc.
 */

/*
 * This file contains preset event names from the Performance Application
 * Programming Interface v3.5 which included the following notice:
 *
 *                             Copyright (c) 2005,6
 *                           Innovative Computing Labs
 *                         Computer Science Department,
 *                            University of Tennessee,
 *                                 Knoxville, TN.
 *                              All Rights Reserved.
 *
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *    * Redistributions of source code must retain the above copyright notice,
 *      this list of conditions and the following disclaimer.
 *    * Redistributions in binary form must reproduce the above copyright
 *      notice, this list of conditions and the following disclaimer in the
 *      documentation and/or other materials provided with the distribution.
 *    * Neither the name of the University of Tennessee nor the names of its
 *      contributors may be used to endorse or promote products derived from
 *      this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
```



```
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 *
 *
 * This open source software license conforms to the BSD License template.
 */


/*
 * Performance Counter Back-End for Intel processors supporting Architectural
 * Performance Monitoring.
 */

#include <sys/cpuvar.h>
#include <sys/param.h>
#include <sys/cpc_impl.h>
#include <sys/cpc_pcbe.h>
#include <sys/modctl.h>
#include <sys/inttypes.h>
#include <sys/systm.h>
#include <sys/cmn_err.h>
#include <sys/x86_archext.h>
#include <sys/sdt.h>
#include <sys/archsystm.h>
#include <sys/privregs.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/cred.h>
#include <sys/policy.h>

#include "core_pcbe_table.h"
#include <core_pcbe_cpcgen.h>

static int core_pcbe_init(void);
static uint_t core_pcbe_ncounters(void);
static const char *core_pcbe_impl_name(void);
static const char *core_pcbe_cpuref(void);
static char *core_pcbe_list_events(uint_t picnum);
static char *core_pcbe_list_attrs(void);
static uint64_t core_pcbe_event_coverage(char *event);
static uint64_t core_pcbe_overflow_bitmap(void);
static int core_pcbe_configure(uint_t picnum, char *event, uint64_t preset,
    uint32_t flags, uint_t nattrs, kcpc_attr_t *attrs, void **data,
    void *token);
static void core_pcbe_program(void *token);
static void core_pcbe_allstop(void);
static void core_pcbe_sample(void *token);
static void core_pcbe_free(void *config);

#define FALSE   0
#define TRUE    1

/* Counter Type */
#define CORE_GPC        0       /* General-Purpose Counter (GPC) */
#define CORE_FFC        1       /* Fixed-Function Counter (FFC) */

/* MSR Addresses */
#define GPC_BASE_PMC            0x00c1  /* First GPC */
#define GPC_BASE_PES            0x0186  /* First GPC Event Select register */
#define FFC_BASE_PMC            0x0309  /* First FFC */
#define PERF_FIXED_CTR_CTRL     0x038d  /* Used to enable/disable FFCs */
#define PERF_GLOBAL_STATUS      0x038e  /* Overflow status register */
#define PERF_GLOBAL_CTRL        0x038f  /* Used to enable/disable counting */
#define PERF_GLOBAL_OVF_CTRL    0x0390  /* Used to clear overflow status */
```

```
124 /*
125  * Processor Event Select register fields
126  */
127 #define CORE_USR        (1ULL << 16)    /* Count while not in ring 0 */
128 #define CORE_OS         (1ULL << 17)    /* Count while in ring 0 */
129 #define CORE_EDGE       (1ULL << 18)    /* Enable edge detection */
130 #define CORE_PC         (1ULL << 19)    /* Enable pin control */
131 #define CORE_INT        (1ULL << 20)    /* Enable interrupt on overflow */
132 #define CORE_EN         (1ULL << 22)    /* Enable counting */
133 #define CORE_INV        (1ULL << 23)    /* Invert the CMASK */
134 #define CORE_ANYTHR     (1ULL << 21)    /* Count event for any thread on core */

136 #define CORE_UMASK_SHIFT        8
137 #define CORE_UMASK_MASK         0xffu
138 #define CORE_CMASK_SHIFT        24
139 #define CORE_CMASK_MASK         0xffu

141 /*
142  * Fixed-function counter attributes
143  */
144 #define CORE_FFC_OS_EN  (1ULL << 0)     /* Count while not in ring 0 */
145 #define CORE_FFC_USR_EN (1ULL << 1)     /* Count while in ring 1 */
146 #define CORE_FFC_ANYTHR (1ULL << 2)     /* Count event for any thread on core */
147 #define CORE_FFC_PMI    (1ULL << 3)     /* Enable interrupt on overflow */

149 /*
150  * Number of bits for specifying each FFC's attributes in the control register
151  */
152 #define CORE_FFC_ATTR_SIZE      4

154 /*
155  * CondChgd and OvfBuffer fields of global status and overflow control registers
156  */
157 #define CONDCHGD        (1ULL << 63)
158 #define OVFBUFFER       (1ULL << 62)
159 #define MASK_CONDCHGD_OVFBUFFER (CONDCHGD | OVFBUFFER)

161 #define ALL_STOPPED     0ULL

163 #define BITMASK_XBITS(x)        ((1ull << (x)) - 1ull)

165 /*
166  * Only the lower 32-bits can be written to in the general-purpose
167  * counters.  The higher bits are extended from bit 31; all ones if
168  * bit 31 is one and all zeros otherwise.
169  *
170  * The fixed-function counters do not have this restriction.
171  */
172 #define BITS_EXTENDED_FROM_31   (BITMASK_XBITS(width_gpc) & ~BITMASK_XBITS(31))

174 #define WRMSR(msr, value)                                               \
175         wrmsr((msr), (value));                                          \
176         DTRACE_PROBE2(wrmsr, uint64_t, (msr), uint64_t, (value));

178 #define RDMSR(msr, value)                                               \
179         (value) = rdmsr((msr));                                         \
180         DTRACE_PROBE2(rdmsr, uint64_t, (msr), uint64_t, (value));

182 typedef struct core_pcbe_config {
183         uint64_t        core_rawpic;
184         uint64_t        core_ctl;       /* Event Select bits */
185         uint64_t        core_pmc;       /* Counter register address */
186         uint64_t        core_pes;       /* Event Select register address */
187         uint_t          core_picno;
188         uint8_t         core_pictype;   /* CORE_GPC or CORE_FFC */
```



```
189 } core_pcbe_config_t;
_____unchanged_portion_omitted_

976 static uint64_t
977 core_pcbe_event_coverage(char *event)
978 {
979         uint64_t bitmap;
980         uint64_t bitmask;
981         const struct events_table_t *n;
982         int i;

984         bitmap = 0;

986         /* Is it an event that a GPC can track? */
987         if (versionid >= 3) {
988                 n = find_gpcevent(event);
989                 if (n != NULL) {
990                         bitmap |= (n->supported_counters &
991                             BITMASK_XBITS(num_gpc));
992                 }
993         } else {
994                 if (find_generic_events(event, cmn_generic_events) != NULL) {
995                         bitmap |= BITMASK_XBITS(num_gpc);
996                 } else if (find_generic_events(event,
997                     generic_events_pic0) != NULL) {
996                 } if (find_generic_events(event, generic_events_pic0) != NULL) {
998                         bitmap |= 1ULL;
999                 } else if (find_gpcevent_core_uarch(event,
1000                    cmn_gpc_events_core_uarch) != NULL) {
1001                        bitmap |= BITMASK_XBITS(num_gpc);
1002                } else if (find_gpcevent_core_uarch(event, pic0_events) !=
1003                    NULL) {
1004                        bitmap |= 1ULL;
1005                } else if (find_gpcevent_core_uarch(event, pic1_events) !=
1006                    NULL) {
1007                        bitmap |= 1ULL << 1;
1008                }
1009        }

1011        /* Check if the event can be counted in the fixed-function counters */
1012        if (num_ffc > 0) {
1013                bitmask = 1ULL << num_gpc;
1014                for (i = 0; i < num_ffc; i++) {
1015                        if (strcmp(event, ffc_names[i]) == 0) {
1016                                bitmap |= bitmask;
1017                        } else if (strcmp(event, ffc_genericnames[i]) == 0) {
1018                                bitmap |= bitmask;
1019                        }
1020                        bitmask = bitmask << 1;
1021                }
1022        }

1024        return (bitmap);
1025 }
_____unchanged_portion_omitted_
```