```
**********************************************************
   24318 Sun Jun  3 17:19:14 2012
new/usr/src/lib/fm/topo/modules/i86pc/chip/chip_amd.c
2812 FMA generic topology support for AMD family 0x10 model 10
Reviewed by: Hans Rosenfeld <rosenfeld@grumpf.hope-2000.org>
**********************************************************
______unchanged_portion_omitted_

 124 static nvlist_t *cs_fmri[MC_CHIP_NCS];

 126 /*
 127  * Called when there is no memory-controller driver to provide topology
 128  * information.  Generate a maximal memory topology that is appropriate
 129  * for the chip revision.  The memory-controller node has already been
 130  * bound as mcnode, and the parent of that is cnode.
 131  *
 132  * We create a tree of dram-channel and chip-select nodes below the
 133  * memory-controller node.  There will be two dram channels and 8 chip-selects
 134  * below each, regardless of actual socket type, processor revision and so on.
 135  * This is adequate for generic diagnosis up to family 0x10 revision D.
 136  */
 137 /*ARGSUSED*/
 138 static int
 139 amd_generic_mc_create(topo_mod_t *mod, uint16_t smbid, tnode_t *cnode,
 140     tnode_t *mcnode, int family, int model, nvlist_t *auth)
 141 {
 142         int chan, cs;

 144         /*
 145          * Elsewhere we have already returned for families less than 0xf.
 146          * This "generic" topology is adequate for all of family 0xf and
 147          * for revisions A to E of family 0x10 (for the list of models
 147          * for revisions A to D of family 0x10 (for the list of models
 148          * in each revision, refer to usr/src/uts/i86pc/os/cpuid_subr.c).
 149          * We cover all family 0x10 models, till model 10.
 149          * We cover all family 0x10 models, till model 9.
 150          */
 151         if (family > 0x10 || (family == 0x10 && model > 10))
 151         if (family > 0x10 || (family == 0x10 && model > 9))
 152                 return (1);

 154         if (topo_node_range_create(mod, mcnode, CHAN_NODE_NAME, 0,
 155             MAX_CHANNUM) < 0) {
 156                 whinge(mod, NULL, "amd_generic_mc_create: range create for "
 157                     "channels failed\n");
 158                 return (-1);
 159         }

 161         for (chan = 0; chan <= MAX_CHANNUM; chan++) {
 162                 tnode_t *chnode;
 163                 nvlist_t *fmri;
 164                 int err;

 166                 if (mkrsrc(mod, mcnode, CHAN_NODE_NAME, chan, auth,
 167                     &fmri) != 0) {
 168                         whinge(mod, NULL, "amd_generic_mc_create: mkrsrc "
 169                             "failed\n");
 170                         return (-1);
 171                 }

 173                 if ((chnode = topo_node_bind(mod, mcnode, CHAN_NODE_NAME,
 174                     chan, fmri)) == NULL) {
 175                         nvlist_free(fmri);
 176                         whinge(mod, NULL, "amd_generic_mc_create: node "
 177                             "bind failed\n");
 178                         return (-1);
```



```
 179                 }

 181                 nvlist_free(fmri);

 183                 (void) topo_pgroup_create(chnode, &chan_pgroup, &err);

 185                 (void) topo_prop_set_string(chnode, PGNAME(CHAN), "channel",
 186                     TOPO_PROP_IMMUTABLE, chan == 0 ? "A" : "B", &err);

 188                 if (FM_AWARE_SMBIOS(mod)) {
 189                         if (topo_node_label_set(chnode, NULL, &err) == -1)
 190                                 whinge(mod, NULL, "amd_generic_mc_create: "
 191                                     "topo_node_label_set\n");
 192                         if (topo_node_fru_set(chnode, NULL, 0, &err) != 0)
 193                                 whinge(mod, NULL, "amd_generic_mc_create: "
 194                                     "topo_node_fru_set failed\n");
 195                 }

 197                 if (topo_node_range_create(mod, chnode, CS_NODE_NAME,
 198                     0, MAX_CSNUM) < 0) {
 199                         whinge(mod, NULL, "amd_generic_mc_create: "
 200                             "range create for cs failed\n");
 201                         return (-1);
 202                 }

 204                 for (cs = 0; cs <= MAX_CSNUM; cs++) {
 205                         tnode_t *csnode;

 207                         if (mkrsrc(mod, chnode, CS_NODE_NAME, cs, auth,
 208                             &fmri) != 0) {
 209                                 whinge(mod, NULL, "amd_generic_mc_create: "
 210                                     "mkrsrc for cs failed\n");
 211                                 return (-1);
 212                         }

 214                         if ((csnode = topo_node_bind(mod, chnode, CS_NODE_NAME,
 215                             cs, fmri)) == NULL) {
 216                                 nvlist_free(fmri);
 217                                 whinge(mod, NULL, "amd_generic_mc_create: "
 218                                     "bind for cs failed\n");
 219                                 return (-1);
 220                         }

 222                         /*
 223                          * Dynamic ASRU for page faults within a chip-select.
 224                          * The topology does not represent pages (there are
 225                          * too many) so when a page is faulted we generate
 226                          * an ASRU to represent the individual page.
 227                          * If SMBIOS meets FMA needs, derive labels & serials
 228                          * for DIMMS and apply to chip-select nodes.
 229                          * If deriving from SMBIOS, skip IPMI
 230                          */
 231                         if (FM_AWARE_SMBIOS(mod)) {
 232                                 if (topo_method_register(mod, csnode,
 233                                     x86pi_gen_cs_methods) < 0)
 234                                         whinge(mod, NULL,
 235                                             "amd_generic_mc_create: "
 236                                             "method registration failed\n");
 237                         } else {
 238                                 if (topo_method_register(mod, csnode,
 239                                     gen_cs_methods) < 0)
 240                                         whinge(mod, NULL,
 241                                             "amd_generic_mc_create: method"
 242                                             "registration failed\n");
 243                         }
```

```
245                         (void) topo_node_asru_set(csnode, fmri,
246                             TOPO_ASRU_COMPUTE, &err);
247                         nvlist_free(fmri);

249                         /*
250                          * If SMBIOS meets FMA needs, set DIMM as the FRU for
251                          * the chip-select node. Use the channel & chip-select
252                          * numbers to get the DIMM instance.
253                          * Send via inst : dram channel number
254                          * Receive via inst : dimm instance
255                          */
256                         if (FM_AWARE_SMBIOS(mod)) {
257                                 int inst;
258                                 id_t dimm_smbid;
259                                 const char *serial;
260                                 const char *part;
261                                 const char *rev;
262                                 char *label;

264                                 (void) topo_pgroup_create(csnode,
265                                     &cs_pgroup, &err);
266                                 inst = chan;
267                                 dimm_smbid = memnode_to_smbiosid(mod, smbid,
268                                     CS_NODE_NAME, cs, &inst);
269                                 serial = chip_serial_smbios_get(mod,
270                                     dimm_smbid);
271                                 part = chip_part_smbios_get(mod,
272                                     dimm_smbid);
273                                 rev = chip_rev_smbios_get(mod, dimm_smbid);
274                                 label = (char *)chip_label_smbios_get(mod,
275                                     chnode, dimm_smbid, NULL);

277                                 (void) topo_prop_set_string(csnode, PGNAME(CS),
278                                     FM_FMRI_HC_SERIAL_ID, TOPO_PROP_IMMUTABLE,
279                                     serial, &err);
280                                 (void) topo_prop_set_string(csnode, PGNAME(CS),
281                                     FM_FMRI_HC_PART, TOPO_PROP_IMMUTABLE,
282                                     part, &err);
283                                 (void) topo_prop_set_string(csnode, PGNAME(CS),
284                                     FM_FMRI_HC_REVISION, TOPO_PROP_IMMUTABLE,
285                                     rev, &err);

287                                 /*
288                                  * We apply DIMM labels to chip-select nodes,
289                                  * FRU for chip-selects should be DIMMs, and
290                                  * we do not derive dimm nodes for Family 0x10
291                                  * so FRU fmri is NULL, but FRU Labels are set,
292                                  * the FRU labels point to the DIMM.
293                                  */
294                                 (void) topo_node_label_set(csnode, label, &err);
295                                 topo_mod_strfree(mod, label);
296                         }
297                 }
298         }

300         return (0);
301 }
```

*unchanged_portion_omitted_*