```
1   ------------------------------------------------------------------------------
2   --  LEON3 Demonstration design
3   --  Copyright (C) 2004 Jiri Gaisler, Gaisler Research
4   ------------------------------------------------------------------------------
5   --  This file is a part of the GRLIB VHDL IP LIBRARY
6   --  Copyright (C) 2003 - 2008, Gaisler Research
7   --  Copyright (C) 2008 - 2014, Aeroflex Gaisler
8   --
9   --  This program is free software; you can redistribute it and/or modify
10  --  it under the terms of the GNU General Public License as published by
11  --  the Free Software Foundation; either version 2 of the License, or
12  --  (at your option) any later version.
13  --
14  --  This program is distributed in the hope that it will be useful,
15  --  but WITHOUT ANY WARRANTY; without even the implied warranty of
16  --  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17  --  GNU General Public License for more details.
18  --
19  --  You should have received a copy of the GNU General Public License
20  --  along with this program; if not, write to the Free Software
21  --  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
22  ------------------------------------------------------------------------------
23
24
25  library ieee;
26  use ieee.std_logic_1164.all;
27  library grlib, techmap;
28  use grlib.config.all;
29  use grlib.amba.all;
30  use grlib.stdlib.all;
31  use grlib.devices.all;
32  use techmap.gencomp.all;
33  use techmap.allclkgen.all;
34  library gaisler;
35  use gaisler.memctrl.all;
36  use gaisler.ddrpkg.all;
37  use gaisler.leon3.all;
38  use gaisler.uart.all;
39  use gaisler.misc.all;
40  use gaisler.i2c.all;
41  use gaisler.net.all;
42  use gaisler.jtag.all;
43
44  library esa;
45  use esa.memoryctrl.all;
46  use work.config.all;
47  use work.ml50x.all;
48  use work.pcie.all;
49  -- pragma translate_off
50  library unisim;
51  use unisim.ODDR;
52  -- pragma translate_on
53
54  entity leon3mp is
55    generic (
56      fabtech   : integer := CFG_FABTECH;
57      memtech   : integer := CFG_MEMTECH;
```

```
58       padtech   : integer := CFG_PADTECH;
59       ncpu      : integer := CFG_NCPU;
60       disas     : integer := CFG_DISAS;  -- Enable disassembly to console
61       dbguart   : integer := CFG_DUART;  -- Print UART on console
62       pclow     : integer := CFG_PCLOW
63     );
64     port (
65       sys_rst_in        : in  std_ulogic;
66       clk_100     : in  std_ulogic;   -- 100 MHz main clock
67       clk_200_p      : in  std_ulogic;   -- 200 MHz
68       clk_200_n      : in  std_ulogic;   -- 200 MHz
69       clk_33          : in  std_ulogic;   -- 33 MHz
70       sram_flash_addr : out std_logic_vector(23 downto 0);
71       sram_flash_data : inout std_logic_vector(31 downto 0);
72       sram_cen       : out std_logic;
73       sram_bw        : out std_logic_vector (0 to 3);
74       sram_oen        : out std_ulogic;
75       sram_flash_we_n : out std_ulogic;
76       flash_ce       : out std_logic;
77       flash_oen      : out std_logic;
78       flash_adv_n     : out std_logic;
79       sram_clk       : out std_ulogic;
80       sram_clk_fb       : in  std_ulogic;
81       sram_mode      : out std_ulogic;
82       sram_adv_ld_n   : out std_ulogic;
83   --pragma translate_off
84       iosn            : out std_ulogic;
85   --pragma translate_on
86
87       ddr_clk        : out std_logic_vector(1 downto 0);
88       ddr_clkb       : out std_logic_vector(1 downto 0);
89       ddr_cke        : out std_logic_vector(1 downto 0);
90       ddr_csb        : out std_logic_vector(1 downto 0);
91       ddr_odt        : out std_logic_vector(1 downto 0);
92       ddr_web         : out std_ulogic;                       -- ddr write enable
93       ddr_rasb        : out std_ulogic;                       -- ddr ras
94       ddr_casb       : out std_ulogic;                      -- ddr cas
95       ddr_dm         : out std_logic_vector (7 downto 0);    -- ddr dm
96       ddr_dqsp       : inout std_logic_vector (7 downto 0);    -- ddr dqs
97       ddr_dqsn       : inout std_logic_vector (7 downto 0);    -- ddr dqs
98       ddr_ad          : out std_logic_vector (13 downto 0);   -- ddr address
99       ddr_ba          : out std_logic_vector (1+CFG_DDR2SP downto 0);    -- ddr bank
     address
100      ddr_dq         : inout std_logic_vector (63 downto 0); -- ddr data
101
102      txd1        : out std_ulogic;         -- UART1 tx data
103      rxd1        : in  std_ulogic;         -- UART1 rx data
104      txd2        : out std_ulogic;         -- UART2 tx data
105      rxd2        : in  std_ulogic;         -- UART2 rx data
106
107        --Mody by Laur
108      HDR1_2  : out  std_ulogic;         -- UART1_LAur txd1 data
109      HDR1_4  : out  std_ulogic;         -- UART1_LAur rtsn1
110      HDR1_6  : in  std_ulogic;          -- UART1_LAur rxd1 data
111      HDR1_8  : in  std_ulogic;          -- UART1_LAur ctsn1
112
113      gpio            : inout std_logic_vector(12 downto 0);  -- I/O port
```

```
114       led             : out std_logic_vector(12 downto 0);
115       bus_error       : out std_logic_vector(1 downto 0);
116
117       phy_gtx_clk     : out std_logic;
118       phy_mii_data    : inout std_logic;    -- ethernet PHY interface
119       phy_tx_clk        : in std_ulogic;
120       phy_rx_clk        : in std_ulogic;
121       phy_rx_data       : in std_logic_vector(7 downto 0);
122       phy_dv          : in std_ulogic;
123       phy_rx_er       : in std_ulogic;
124       phy_col         : in std_ulogic;
125       phy_crs         : in std_ulogic;
126       phy_tx_data     : out std_logic_vector(7 downto 0);
127       phy_tx_en       : out std_ulogic;
128       phy_tx_er       : out std_ulogic;
129       phy_mii_clk       : out std_ulogic;
130       phy_rst_n       : out std_ulogic;
131       phy_int         : in std_ulogic;
132
133       ps2_keyb_clk    : inout std_logic;
134       ps2_keyb_data   : inout std_logic;
135       ps2_mouse_clk   : inout std_logic;
136       ps2_mouse_data  : inout std_logic;
137
138       usb_csn         : out std_logic;
139       usb_rstn        : out std_logic;
140
141       iic_scl_main    : inout std_ulogic;
142       iic_sda_main    : inout std_ulogic;
143
144       iic_scl_video   : inout std_logic;
145       iic_sda_video   : inout std_logic;
146
147       tft_lcd_data    : out std_logic_vector(11 downto 0);
148       tft_lcd_clk_p   : out std_ulogic;
149       tft_lcd_clk_n   : out std_ulogic;
150       tft_lcd_hsync   : out std_ulogic;
151       tft_lcd_vsync   : out std_ulogic;
152       tft_lcd_de      : out std_ulogic;
153       tft_lcd_reset_b : out std_ulogic;
154
155       sysace_mpa      : out std_logic_vector(6 downto 0);
156       sysace_mpce     : out std_ulogic;
157       sysace_mpirq    : in  std_ulogic;
158       sysace_mpoe     : out std_ulogic;
159       sysace_mpwe     : out std_ulogic;
160       sysace_d        : inout std_logic_vector(15 downto 0);
161
162       pci_exp_txp     : out std_logic_vector(CFG_NO_OF_LANES-1 downto 0);
163       pci_exp_txn     : out std_logic_vector(CFG_NO_OF_LANES-1 downto 0);
164       pci_exp_rxp     : in std_logic_vector(CFG_NO_OF_LANES-1 downto 0);
165       pci_exp_rxn     : in std_logic_vector(CFG_NO_OF_LANES-1 downto 0);
166       sys_clk_p       : in  std_logic;
167       sys_clk_n       : in  std_logic;
168       sys_reset_n     : in  std_logic
169       );
170   end;
```

```
171
172   architecture rtl of leon3mp is
173
174     component ODDR
175       generic
176         ( DDR_CLK_EDGE : string := "OPPOSITE_EDGE";
177   --        INIT : bit := '0';
178           SRTYPE : string := "SYNC");
179       port
180         (
181           Q : out std_ulogic;
182           C : in std_ulogic;
183           CE : in std_ulogic;
184           D1 : in std_ulogic;
185           D2 : in std_ulogic;
186           R : in std_ulogic;
187           S : in std_ulogic
188         );
189     end component;
190
191   component svga2ch7301c
192     generic (
193       tech    : integer := 0;
194       idf     : integer := 0;
195       dynamic : integer := 0
196       );
197     port (
198       clk         : in  std_ulogic;
199       rstn        : in  std_ulogic;
200       clksel      : in  std_logic_vector(1 downto 0);
201       vgao        : in  apbvga_out_type;
202       vgaclk_fb   : in  std_ulogic;
203       clk25_fb    : in  std_ulogic;
204       clk40_fb    : in  std_ulogic;
205       clk65_fb    : in  std_ulogic;
206       vgaclk      : out std_ulogic;
207       clk25       : out std_ulogic;
208       clk40       : out std_ulogic;
209       clk65       : out std_ulogic;
210       dclk_p      : out std_ulogic;
211       dclk_n      : out std_ulogic;
212       locked      : out std_ulogic;
213       data        : out std_logic_vector(11 downto 0);
214       hsync       : out std_ulogic;
215       vsync       : out std_ulogic;
216       de          : out std_ulogic
217       );
218   end component;
219   ------------------------------------------------------------------------
220   ---  TFG_LUISMI ----------------------------------------------------------
221   ------------------------------------------------------------------------
222   component TFG_IMAGE_SCALING
223         generic (FABTECH          : integer := 0;
224                  MEMTECH          : integer := DEFMEMTECH;
225                  PINDEX           : integer := 0;
226                  PADDR            : integer := 0;
227                  PMASK            : integer := 16#fff#;
```

```
228                 HINDEX           : integer := 0;
229                 AHBACCSZ         : integer := 32;
230                 BURSTLEN         : integer range 2 to 8 := 8;
231                 AHB_SLV_HINDEX   : integer := 0;
232                 AHB_SLV_HADDR    : integer := 0;
233                 AHB_SLV_HMASK    : integer := 16#fff#;
234                 HIRQ             : integer := 10;
235
236                 INT_ADDRESS_WIDTH : integer := 11;
237                 SRC_ADDRESS_WIDTH : integer := 9;
238                 DST_ADDRESS_WIDTH : integer := 9;
239                 ADDRESS_HEIGHT    : integer := 11);
240         port (Rst_n : in  std_ulogic;
241               Clk   : in  std_ulogic;
242               Apbi  : in  apb_slv_in_type;
243               Apbo  : out apb_slv_out_type;
244               Ahbi  : in  ahb_mst_in_type;
245               Ahbo  : out ahb_mst_out_type);
246   end component;
247
248   constant blength : integer := 12;
249   constant fifodepth : integer := 8;
250   constant maxahbm : integer := NCPU+CFG_AHB_UART
251      +CFG_GRETH+CFG_AHB_JTAG+CFG_SVGA_ENABLE+CFG_PCIEXP+1;
252
253   signal ddr_clk_fb  : std_logic;
254   signal vcc, gnd   : std_logic_vector(4 downto 0);
255   signal memi  : memory_in_type;
256   signal memo  : memory_out_type;
257   signal wpo   : wprot_out_type;
258   signal sdi   : sdctrl_in_type;
259   signal sdo   : sdctrl_out_type;
260
261   signal apbi  : apb_slv_in_type;
262   signal apbo  : apb_slv_out_vector := (others => apb_none);
263   signal ahbsi : ahb_slv_in_type;
264   signal ahbso : ahb_slv_out_vector := (others => ahbs_none);
265   signal ahbmi : ahb_mst_in_type;
266   signal ahbmo : ahb_mst_out_vector := (others => ahbm_none);
267
268   signal clkm, rstn, rstraw, srclkl : std_ulogic;
269   signal clk_200 : std_ulogic;
270   signal clk25, clk40, clk65 : std_ulogic;
271
272   signal cgi, cgi2   : clkgen_in_type;
273   signal cgo, cgo2   : clkgen_out_type;
274   signal u1i, u2i, dui : uart_in_type;
275   signal u1o, u2o, duo : uart_out_type;
276
277   signal irqi : irq_in_vector(0 to NCPU-1);
278   signal irqo : irq_out_vector(0 to NCPU-1);
279
280   signal dbgi : l3_debug_in_vector(0 to NCPU-1);
281   signal dbgo : l3_debug_out_vector(0 to NCPU-1);
282
283   signal dsui : dsu_in_type;
284   signal dsuo : dsu_out_type;
```

```
285
286   signal ethi, ethi1, ethi2 : eth_in_type;
287   signal etho, etho1, etho2 : eth_out_type;
288
289   signal gpti : gptimer_in_type;
290   signal gpto : gptimer_out_type;
291
292   signal gpioi : gpio_in_type;
293   signal gpioo : gpio_out_type;
294
295   signal clklock, lock, lclk, clkml, rst, ndsuact : std_ulogic;
296   signal tck, tckn, tms, tdi, tdo : std_ulogic;
297   signal ddrclk, ddrrst : std_ulogic;
298
299   signal egtx_clk_fb : std_ulogic;
300   signal egtx_clk, legtx_clk, l2egtx_clk : std_ulogic;
301
302   signal kbdi  : ps2_in_type;
303   signal kbdo  : ps2_out_type;
304   signal moui  : ps2_in_type;
305   signal mouo  : ps2_out_type;
306
307   signal vgao  : apbvga_out_type;
308   signal lcd_datal : std_logic_vector(11 downto 0);
309   signal lcd_hsyncl, lcd_vsyncl, lcd_del, lcd_reset_bl : std_ulogic;
310   signal clk_sel : std_logic_vector(1 downto 0);
311   signal vgalock : std_ulogic;
312   signal clkvga, clkvga_p, clkvga_n : std_ulogic;
313
314   signal i2ci, dvi_i2ci : i2c_in_type;
315   signal i2co, dvi_i2co : i2c_out_type;
316
317   constant BOARD_FREQ_200 : integer := 200000;   -- input frequency in KHz
318   constant BOARD_FREQ : integer := 100000;   -- input frequency in KHz
319   constant CPU_FREQ : integer := BOARD_FREQ * CFG_CLKMUL / CFG_CLKDIV;  -- cpu
      frequency in KHz
320   constant I2C_FILTER : integer := (CPU_FREQ*5+50000)/100000+1;
321   constant IOAEN : integer := CFG_DDR2SP + CFG_GRACECTRL;
322
323   signal stati : ahbstat_in_type;
324   signal ssrclkfb : std_ulogic;
325
326   -- Used for connecting input/output signals to the DDR3 controller
327   signal migi    : mig_app_in_type;
328   signal migo    : mig_app_out_type;
329   signal phy_init_done    : std_ulogic;
330   signal clk0_tb, rst0_tb, rst0_tbn : std_ulogic;
331
332   signal sysmoni : grsysmon_in_type;
333   signal sysmono : grsysmon_out_type;
334
335   signal clkace : std_ulogic;
336   signal acei   : gracectrl_in_type;
337   signal aceo   : gracectrl_out_type;
338
339   attribute syn_keep : boolean;
340   attribute syn_preserve : boolean;
```

```
341   attribute syn_keep of clkml : signal is true;
342   attribute syn_preserve of clkml : signal is true;
343   attribute syn_keep of clkm : signal is true;
344   attribute syn_preserve of clkm : signal is true;
345   attribute syn_keep of egtx_clk : signal is true;
346   attribute syn_preserve of egtx_clk : signal is true;
347   attribute syn_keep of clkvga : signal is true;
348   attribute syn_preserve of clkvga : signal is true;
349   attribute syn_keep of clk25 : signal is true;
350   attribute syn_preserve of clk25 : signal is true;
351   attribute syn_keep of clk40 : signal is true;
352   attribute syn_preserve of clk40 : signal is true;
353   attribute syn_keep of clk65 : signal is true;
354   attribute syn_preserve of clk65 : signal is true;
355   -- attribute syn_keep of clk_200 : signal is true; -- mody Laur MIG
356   -- attribute syn_preserve of clk_200 : signal is true; -- mody Laur MIG
357   attribute syn_preserve of phy_init_done : signal is true;
358   attribute keep : boolean;
359   attribute keep of lock : signal is true;
360   attribute keep of clkml : signal is true;
361   attribute keep of clkm : signal is true;
362   attribute keep of egtx_clk : signal is true;
363   attribute keep of clkvga : signal is true;
364   attribute keep of clk25 : signal is true;
365   attribute keep of clk40 : signal is true;
366   attribute keep of clk65 : signal is true;
367   -- attribute keep of clk_200 : signal is true;  -- mody Laur MIG
368   attribute keep of phy_init_done : signal is true;
369
370   attribute syn_noprune : boolean;
371   attribute syn_noprune of clk_33_pad : label is true;
372
373   begin
374
375     usb_csn <= '1';
376     usb_rstn <= rstn;
377     rst0_tbn <= not rst0_tb;
378
379   ----------------------------------------------------------------------
380   ---  Reset and Clock generation  -------------------------------------
381   ----------------------------------------------------------------------
382
383     vcc <= (others => '1'); gnd <= (others => '0');
384     cgi.pllctrl <= "00"; cgi.pllrst <= rstraw; cgi.pllref <= ssrclkfb;
385
386     ssrref_pad : clkpad generic map (tech => padtech)
387         port map (sram_clk_fb, ssrclkfb);
388     clk_pad : clkpad generic map (tech => padtech, arch => 2)
389      port map (clk_100, lclk);
390     clk200_pad : clkpad_ds generic map (tech => padtech, level => lvds, voltage => x25v)
391      port map (clk_200_p, clk_200_n, clk_200);
392
393     srclk_pad : outpad generic map (tech => padtech, slew => 1, strength => 24)
394      port map (sram_clk, srclkl);
395
396     clk_33_pad : clkpad generic map (tech => padtech)
397           port map (clk_33, clkace);
```

```
398
399     clkgen0 : clkgen      -- system clock generator
400       generic map (CFG_FABTECH, CFG_CLKMUL, CFG_CLKDIV, 1, 0, 0, 0, 0, BOARD_FREQ, 0)
401       port map (lclk, gnd(0), clkm, open, open, srclkl, open, cgi, cgo);
402
403     gclk : if CFG_GRETH1G /= 0 generate
404       clkgen1 : clkgen       -- Ethernet 1G PHY clock generator
405         generic map (CFG_FABTECH, 5, 4, 0, 0, 0, 0, 0, BOARD_FREQ, 0)
406         port map (lclk, gnd(0), egtx_clk, open, open, open, open, cgi2, cgo2);
407       cgi2.pllctrl <= "00"; cgi2.pllrst <= rstraw; --cgi2.pllref <= egtx_clk_fb;
408       x0 : ODDR port map ( Q => phy_gtx_clk, C => egtx_clk, CE => vcc(0),
409         D1 => gnd(0), D2 => vcc(0), R => gnd(0), S => gnd(0));
410   --    D1 => vcc(0), D2 => gnd(0), R => gnd(0), S => gnd(0));
411     end generate;
412     nogclk : if CFG_GRETH1G = 0 generate
413       cgo2.clklock <= '1'; phy_gtx_clk <= '0';
414     end generate;
415
416     resetn_pad : inpad generic map (tech => padtech) port map (sys_rst_in, rst);
417     rst0 : rstgen         -- reset generator
418     port map (rst, clkm, clklock, rstn, rstraw);
419     clklock <= lock and cgo.clklock and cgo2.clklock and vgalock;
420
421   ----------------------------------------------------------------------
422   ---  AHB CONTROLLER --------------------------------------------------
423   ----------------------------------------------------------------------
424
425     ahb0 : ahbctrl     -- AHB arbiter/multiplexer
426     generic map (defmast => CFG_DEFMST, split => CFG_SPLIT,
427      rrobin => CFG_RROBIN, ioaddr => CFG_AHBIO, devid => CFG_BOARD_SELECTION,
428      ioen => IOAEN, nahbm => maxahbm, nahbs => 8)
429     port map (rstn, clkm, ahbmi, ahbmo, ahbsi, ahbso);
430
431   ----------------------------------------------------------------------
432   ---  LEON3 processor and DSU -----------------------------------------
433   ----------------------------------------------------------------------
434
435     l3 : if CFG_LEON3 = 1 generate
436       cpu : for i in 0 to NCPU-1 generate
437         u0 : leon3s       -- LEON3 processor
438         generic map (i, fabtech, memtech, CFG_NWIN, CFG_DSU, CFG_FPU, CFG_V8,
439      0, CFG_MAC, pclow, CFG_NOTAG, CFG_NWP, CFG_ICEN, CFG_IREPL, CFG_ISETS, CFG_ILINE,
440      CFG_ISETSZ, CFG_ILOCK, CFG_DCEN, CFG_DREPL, CFG_DSETS, CFG_DLINE, CFG_DSETSZ,
441      CFG_DLOCK, CFG_DSNOOP, CFG_ILRAMEN, CFG_ILRAMSZ, CFG_ILRAMADDR, CFG_DLRAMEN,
442           CFG_DLRAMSZ, CFG_DLRAMADDR, CFG_MMUEN, CFG_ITLBNUM, CFG_DTLBNUM, CFG_TLB_TYPE,
      CFG_TLB_REP,
443           CFG_LDDEL, disas, CFG_ITBSZ, CFG_PWD, CFG_SVT, CFG_RSTADDR, NCPU-1)
444         port map (clkm, rstn, ahbmi, ahbmo(i), ahbsi, ahbso,
445            irqi(i), irqo(i), dbgi(i), dbgo(i));
446       end generate;
447       bus_error(0) <= not dbgo(0).error;
448       bus_error(1) <= rstn;
449
450       dsugen : if CFG_DSU = 1 generate
451         dsu0 : dsu3       -- LEON3 Debug Support Unit
452         generic map (hindex => 2, haddr => 16#900#, hmask => 16#F00#,
453            ncpu => NCPU, tbits => 30, tech => memtech, irq => 0, kbytes => CFG_ATBSZ)
```

```
454         port map (rstn, clkm, ahbmi, ahbsi, ahbso(2), dbgo, dbgi, dsui, dsuo);
455         dsui.enable <= '1';
456   --    dsubre_pad : inpad generic map (tech => padtech) port map (dsubre, dsui.break);
457          dsui.break <= gpioo.val(11); --  South Button
458   --    dsuact_pad : outpad generic map (tech => padtech) port map (dsuact, ndsuact);
459       led(4) <= dsuo.active;
460       end generate;
461     end generate;
462
463     nodsu : if CFG_DSU = 0 generate
464       dsuo.tstop <= '0'; dsuo.active <= '0';
465     end generate;
466
467     dcomgen : if CFG_AHB_UART = 1 generate
468       dcom0: ahbuart      -- Debug UART
469       generic map (hindex => NCPU, pindex => 7, paddr => 7)
470       port map (rstn, clkm, dui, duo, apbi, apbo(7), ahbmi, ahbmo(NCPU));
471   --    dsurx_pad : inpad generic map (tech => padtech) port map (rxd1, dui.rxd);
472   --    dsutx_pad : outpad generic map (tech => padtech) port map (txd1, duo.txd);
473       dui.rxd <= rxd1 when gpioo.val(0) = '1' else '1';
474     end generate;
475
476     txd1 <= duo.txd when  gpioo.val(0) = '1' else u1o.txd;
477
478     txd2 <= '0';                          -- Second UART is unused
479
480     ahbjtaggen0 :if CFG_AHB_JTAG = 1 generate
481       ahbjtag0 : ahbjtag generic map(tech => fabtech, hindex => NCPU+CFG_AHB_UART)
482         port map(rstn, clkm, tck, tms, tdi, tdo, ahbmi, ahbmo(NCPU+CFG_AHB_UART),
483                  open, open, open, open, open, open, open, gnd(0));
484     end generate;
485
486   ----------------------------------------------------------------------
487   ---  Memory controllers ----------------------------------------------
488   ----------------------------------------------------------------------
489
490     memi.writen <= '1'; memi.wrn <= "1111"; memi.bwidth <= "01";
491     memi.brdyn <= '1'; memi.bexcn <= '1';
492
493     mctrl0 : if CFG_MCTRL_LEON2 = 1 generate
494       mctrl0 : mctrl generic map (hindex => 3, pindex => 0,
495      ramaddr => 16#400# + (CFG_DDR2SP+CFG_MIG_DDR2)*16#800#, rammask => 16#FE0#,
496      paddr => 0, srbanks => 1, ram8 => CFG_MCTRL_RAM8BIT,
497      ram16 => CFG_MCTRL_RAM16BIT, sden => CFG_MCTRL_SDEN,
498      invclk => CFG_MCTRL_INVCLK, sepbus => CFG_MCTRL_SEPBUS)
499       port map (rstn, clkm, memi, memo, ahbsi, ahbso(3), apbi, apbo(0), wpo, open);
500     end generate;
501
502     flash_adv_n_pad : outpad generic map (tech => padtech)
503      port map (flash_adv_n, gnd(0));
504     sram_adv_ld_n_pad : outpad generic map (tech => padtech)
505      port map (sram_adv_ld_n, gnd(0));
506     sram_mode_pad : outpad generic map (tech => padtech)
507      port map (sram_mode, gnd(0));
508     addr_pad : outpadv generic map (width => 24, tech => padtech)
509      port map (sram_flash_addr, memo.address(24 downto 1));
510     rams_pad : outpad generic map ( tech => padtech)
```

```
511       port map (sram_cen, memo.ramsn(0));
512      roms_pad : outpad generic map (tech => padtech)
513       port map (flash_ce, memo.romsn(0));
514      ramoen_pad  : outpad generic map (tech => padtech)
515       port map (sram_oen, memo.ramoen(0));
516      flash_oen_pad  : outpad generic map (tech => padtech)
517       port map (flash_oen, memo.oen);
518    --pragma translate_off
519      iosn_pad  : outpad generic map (tech => padtech)
520       port map (iosn, memo.iosn);
521    --pragma translate_on
522      rwen_pad : outpadv generic map (width => 2, tech => padtech)
523       port map (sram_bw(0 to 1), memo.wrn(3 downto 2));
524      rwen_pad2 : outpadv generic map (width => 2, tech => padtech)
525       port map (sram_bw(2 to 3), memo.wrn(1 downto 0));
526      wri_pad  : outpad generic map (tech => padtech)
527       port map (sram_flash_we_n, memo.writen);
528      data_pads : iopadvv generic map (tech => padtech, width => 16)
529          port map (sram_flash_data(15 downto 0), memo.data(31 downto 16),
530          memo.vbdrive(31 downto 16), memi.data(31 downto 16));
531      data_pads2 : iopadvv generic map (tech => padtech, width => 16)
532          port map (sram_flash_data(31 downto 16), memo.data(15 downto 0),
533          memo.vbdrive(15 downto 0), memi.data(15 downto 0));
534
535      migsp0 : if (CFG_MIG_DDR2 = 1) generate
536
537        ahb2mig0 : entity work.ahb2mig_ml50x
538        generic map ( hindex => 0, haddr => 16#400#, hmask => MIGHMASK,
539       MHz => 400, Mbyte => 512, nosync => 0) --boolean'pos(CFG_MIG_CLK4=12))
      --CFG_CLKDIV/12)
540        port map (
541       rst_ahb => rstn, rst_ddr => rst0_tbn, clk_ahb => clkm, clk_ddr => clk0_tb,
542       ahbsi => ahbsi, ahbso => ahbso(0), migi => migi, migo => migo);
543
544        migv5 : mig_36_1
545         generic map (
546       CKE_WIDTH => CKE_WIDTH, CS_NUM => CS_NUM, CS_WIDTH => CS_WIDTH, CS_BITS => CS_BITS,
547       COL_WIDTH => COL_WIDTH, ROW_WIDTH => ROW_WIDTH,
548       NOCLK200 => true, SIM_ONLY => 1)
549         port map(
550          ddr2_dq => ddr_dq(DQ_WIDTH-1 downto 0),
551          ddr2_a => ddr_ad(ROW_WIDTH-1 downto 0),
552          ddr2_ba => ddr_ba(1 downto 0), ddr2_ras_n => ddr_rasb,
553          ddr2_cas_n => ddr_casb, ddr2_we_n => ddr_web,
554          ddr2_cs_n => ddr_csb(CS_NUM-1 downto 0), ddr2_odt => ddr_odt(0 downto 0),
555          ddr2_cke => ddr_cke(CKE_WIDTH-1 downto 0),
556          ddr2_dm => ddr_dm(DM_WIDTH-1 downto 0),
557          sys_clk => clk_200, idly_clk_200 => clk_200, sys_rst_n => rstraw,
558          phy_init_done => phy_init_done,
559          rst0_tb => rst0_tb, clk0_tb => clk0_tb,
560          app_wdf_afull => migo.app_wdf_afull,
561          app_af_afull => migo.app_af_afull,
562          rd_data_valid => migo.app_rd_data_valid,
563          app_wdf_wren => migi.app_wdf_wren,
564          app_af_wren => migi.app_en, app_af_addr =>  migi.app_addr,
565          app_af_cmd => migi.app_cmd,
566          rd_data_fifo_out => migo.app_rd_data, app_wdf_data => migi.app_wdf_data,
```

```
567         app_wdf_mask_data => migi.app_wdf_mask,
568         ddr2_dqs => ddr_dqsp(DQS_WIDTH-1 downto 0),
569         ddr2_dqs_n => ddr_dqsn(DQS_WIDTH-1 downto 0),
570         ddr2_ck => ddr_clk((CLK_WIDTH-1) downto 0),
571         ddr2_ck_n => ddr_clkb((CLK_WIDTH-1) downto 0)
572       );
573
574       lock <= phy_init_done;
575       led(5) <= phy_init_done;
576     end generate;
577
578     ddrsp0 : if (CFG_DDR2SP /= 0) and (CFG_MIG_DDR2 = 0) generate
579       ddrc0 : ddr2spa generic map ( fabtech => fabtech, memtech => memtech,
580         hindex => 0, haddr => 16#400#, hmask => 16#C00#, ioaddr => 1,
581         pwron => CFG_DDR2SP_INIT, MHz => BOARD_FREQ_200/1000, TRFC => CFG_DDR2SP_TRFC,
582         clkmul => CFG_DDR2SP_FREQ/10, clkdiv => 20, ahbfreq => CPU_FREQ/1000,
583         col => CFG_DDR2SP_COL, Mbyte => CFG_DDR2SP_SIZE, ddrbits => 64,
584         ddelayb0 => CFG_DDR2SP_DELAY0, ddelayb1 => CFG_DDR2SP_DELAY1,
585         ddelayb2 => CFG_DDR2SP_DELAY2, ddelayb3 => CFG_DDR2SP_DELAY3,
586         ddelayb4 => CFG_DDR2SP_DELAY4, ddelayb5 => CFG_DDR2SP_DELAY5,
587         ddelayb6 => CFG_DDR2SP_DELAY6, ddelayb7 => CFG_DDR2SP_DELAY7,
588         numidelctrl => 1, norefclk => 0, odten => 3, nclk => 2,
589         eightbanks => 1)
590       port map ( rst, rstn, clk_200, clkm, clk_200, lock, clkml, clkml, ahbsi, ahbso(0),
591                  ddr_clk, ddr_clkb, ddr_clk_fb, ddr_clk_fb, ddr_cke, ddr_csb, ddr_web,
    ddr_rasb, ddr_casb,
592                  ddr_dm, ddr_dqsp, ddr_dqsn, ddr_ad, ddr_ba, ddr_dq, ddr_odt);
593
594       led(5) <= '0';
595     end generate;
596
597     noddr :  if (CFG_DDR2SP = 0) and (CFG_MIG_DDR2 = 0) generate lock <= '1'; led(5) <=
    '0'; end generate;
598
599   ------------------------------------------------------------------------
600   ---  System ACE I/F Controller -----------------------------------------
601   ------------------------------------------------------------------------
602
603     grace: if CFG_GRACECTRL = 1 generate
604       grace0 : gracectrl generic map (hindex => 4, hirq => 3,
605           haddr => 16#002#, hmask => 16#fff#, split => CFG_SPLIT)
606         port map (rstn, clkm, clkace, ahbsi, ahbso(4), acei, aceo);
607     end generate;
608     nograce: if CFG_GRACECTRL /= 1 generate
609       aceo <= gracectrl_none;
610     end generate;
611
612     sysace_mpa_pads : outpadv generic map (width => 7, tech => padtech)
613       port map (sysace_mpa, aceo.addr);
614     sysace_mpce_pad : outpad generic map (tech => padtech)
615       port map (sysace_mpce, aceo.cen);
616     sysace_d_pads : iopadv generic map (tech => padtech, width => 16)
617       port map (sysace_d, aceo.do, aceo.doen, acei.di);
618     sysace_mpoe_pad : outpad generic map (tech => padtech)
619       port map (sysace_mpoe, aceo.oen);
620     sysace_mpwe_pad : outpad generic map (tech => padtech)
621       port map (sysace_mpwe, aceo.wen);
```

```
622     sysace_mpirq_pad : inpad generic map (tech => padtech)
623       port map (sysace_mpirq, acei.irq);
624
625   ----------------------------------------------------------------------
626   ---  APB Bridge and various periherals -------------------------------
627   ----------------------------------------------------------------------
628
629     bpromgen : if CFG_AHBROMEN /= 0 generate
630       brom : entity work.ahbrom
631         generic map (hindex => 6, haddr => CFG_AHBRODDR, pipe => CFG_AHBROPIP)
632         port map ( rstn, clkm, ahbsi, ahbso(6));
633     end generate;
634
635   ----------------------------------------------------------------------
636   ---  APB Bridge and various periherals -------------------------------
637   ----------------------------------------------------------------------
638
639     apb0 : apbctrl          -- AHB/APB bridge
640     generic map (hindex => 1, haddr => CFG_APBADDR, nslaves => 16)
641     port map (rstn, clkm, ahbsi, ahbso(1), apbi, apbo );
642
643     ua1 : if CFG_UART1_ENABLE /= 0 generate
644       uart1 : apbuart       -- UART 1
645       generic map (pindex => 1, paddr => 1,  pirq => 2, console => dbguart,
646      fifosize => CFG_UART1_FIFO)
647       port map (rstn, clkm, apbi, apbo(1), u1i, u1o);
648       --u1i.extclk <= '0'; u1i.ctsn <= '0';             -- original code
649       --u1i.rxd <= rxd1 when gpioo.val(0) = '0' else '1';  -- original code
650
651         -- Added by Laur
652      u1i.rxd <= rxd1 when gpioo.val(0) = '0' else HDR1_6;
653      u1i.ctsn <= '0' when gpioo.val(0) = '0' else HDR1_8;
654      --outputs
655      HDR1_2<=u1o.txd;
656      HDR1_4<=u1o.rtsn;
657     end generate;
658
659     led(0) <= gpioo.val(0); led(1) <= not rxd1;
660     led(2) <= not duo.txd when gpioo.val(0) = '1' else not u1o.txd;
661     led (12 downto 6) <= (others => '0');
662     irqctrl : if CFG_IRQ3_ENABLE /= 0 generate
663       irqctrl0 : irqmp       -- interrupt controller
664       generic map (pindex => 2, paddr => 2, ncpu => NCPU)
665       port map (rstn, clkm, apbi, apbo(2), irqo, irqi);
666     end generate;
667     irq3 : if CFG_IRQ3_ENABLE = 0 generate
668       x : for i in 0 to NCPU-1 generate
669         irqi(i).irl <= "0000";
670       end generate;
671       apbo(2) <= apb_none;
672     end generate;
673
674     gpt : if CFG_GPT_ENABLE /= 0 generate
675       timer0 : gptimer          -- timer unit
676       generic map (pindex => 3, paddr => 3, pirq => CFG_GPT_IRQ,
677      sepirq => CFG_GPT_SEPIRQ, sbits => CFG_GPT_SW, ntimers => CFG_GPT_NTIM,
678      nbits => CFG_GPT_TW)
```

```
679       port map (rstn, clkm, apbi, apbo(3), gpti, gpto);
680       gpti.dhalt <= dsuo.tstop; gpti.extclk <= '0';
681       led(3) <= gpto.wdog;
682     end generate;
683
684     nogpt : if CFG_GPT_ENABLE = 0 generate apbo(3) <= apb_none; end generate;
685
686     kbd : if CFG_KBD_ENABLE /= 0 generate
687       ps21 : apbps2 generic map(pindex => 4, paddr => 4, pirq => 4)
688         port map(rstn, clkm, apbi, apbo(4), moui, mouo);
689       ps20 : apbps2 generic map(pindex => 5, paddr => 5, pirq => 5)
690         port map(rstn, clkm, apbi, apbo(5), kbdi, kbdo);
691     end generate;
692     nokbd : if CFG_KBD_ENABLE = 0 generate apbo(5) <= apb_none; kbdo <= ps2o_none; end
    generate;
693     kbdclk_pad : iopad generic map (tech => padtech)
694         port map (ps2_keyb_clk,kbdo.ps2_clk_o, kbdo.ps2_clk_oe, kbdi.ps2_clk_i);
695     kbdata_pad : iopad generic map (tech => padtech)
696           port map (ps2_keyb_data, kbdo.ps2_data_o, kbdo.ps2_data_oe, kbdi.ps2_data_i);
697     mouclk_pad : iopad generic map (tech => padtech)
698         port map (ps2_mouse_clk, mouo.ps2_clk_o, mouo.ps2_clk_oe, moui.ps2_clk_i);
699     mouata_pad : iopad generic map (tech => padtech)
700           port map (ps2_mouse_data, mouo.ps2_data_o, mouo.ps2_data_oe, moui.ps2_data_i);
701
702     vga : if CFG_VGA_ENABLE /= 0 generate
703       vga0 : apbvga generic map(memtech => memtech, pindex => 6, paddr => 6)
704         port map(rstn, clkm, clkvga, apbi, apbo(6), vgao);
705         clk_sel <= "00";
706     end generate;
707
708     svga : if CFG_SVGA_ENABLE /= 0 generate
709       svga0 : svgactrl generic map( length => 800, part => 256,  --mody by LUISMI
710         memtech => memtech, pindex => 6, paddr => 6,
711         hindex => CFG_NCPU+CFG_AHB_UART+CFG_AHB_JTAG,
712         clk0 => 40000, clk1 => 40000, clk2 => 25000, clk3 => 15385, burstlen => 6,
713         ahbaccsz => CFG_AHBDW)
714         port map(rstn, clkm, clkvga, apbi, apbo(6), vgao, ahbmi,
715                  ahbmo(CFG_NCPU+CFG_AHB_UART+CFG_AHB_JTAG), clk_sel);
716     end generate;
717
718     vgadvi : if (CFG_VGA_ENABLE + CFG_SVGA_ENABLE) /= 0 generate
719       dvi0 : svga2ch7301c generic map (tech => fabtech, idf => 2)
720         port map (lclk, rstraw, clk_sel, vgao, clkvga, clk25, clk40, clk65,
721                   clkvga, clk25, clk40, clk65, clkvga_p, clkvga_n,
722                   vgalock, lcd_datal, lcd_hsyncl, lcd_vsyncl, lcd_del);
723
724       i2cdvi : i2cmst
725         generic map (pindex => 9, paddr => 9, pmask => 16#FFF#,
726                      pirq => 6, filter => I2C_FILTER)
727         port map (rstn, clkm, apbi, apbo(9), dvi_i2ci, dvi_i2co);
728     end generate;
729
730     novga : if (CFG_VGA_ENABLE + CFG_SVGA_ENABLE) = 0 generate
731        apbo(6) <= apb_none; vgalock <= '1';
732        lcd_datal <= (others => '0'); clkvga_p <= '0'; clkvga_n <= '0';
733        lcd_hsyncl <= '0'; lcd_vsyncl <= '0'; lcd_del <= '0';
734        dvi_i2co.scloen <= '1'; dvi_i2co.sdaoen <= '1';
```

```
735     end generate;
736
737     tft_lcd_data_pad : outpadv generic map (width => 12, tech => padtech)
738           port map (tft_lcd_data, lcd_datal);
739     tft_lcd_clkp_pad : outpad generic map (tech => padtech)
740       port map (tft_lcd_clk_p, clkvga_p);
741     tft_lcd_clkn_pad : outpad generic map (tech => padtech)
742       port map (tft_lcd_clk_n, clkvga_n);
743     tft_lcd_hsync_pad : outpad generic map (tech => padtech)
744       port map (tft_lcd_hsync, lcd_hsyncl);
745     tft_lcd_vsync_pad : outpad generic map (tech => padtech)
746       port map (tft_lcd_vsync, lcd_vsyncl);
747     tft_lcd_de_pad : outpad generic map (tech => padtech)
748       port map (tft_lcd_de, lcd_del);
749     tft_lcd_reset_pad : outpad generic map (tech => padtech)
750       port map (tft_lcd_reset_b, rstn);
751     dvi_i2c_scl_pad : iopad generic map (tech => padtech)
752       port map (iic_scl_video, dvi_i2co.scl, dvi_i2co.scloen, dvi_i2ci.scl);
753     dvi_i2c_sda_pad : iopad generic map (tech => padtech)
754       port map (iic_sda_video, dvi_i2co.sda, dvi_i2co.sdaoen, dvi_i2ci.sda);
755
756     gpio0 : if CFG_GRGPIO_ENABLE /= 0 generate     -- GPIO unit
757       grgpio0: grgpio
758       generic map(pindex => 8, paddr => 8, imask => 16#00F0#, nbits => 13)
759       port map(rst => rstn, clk => clkm, apbi => apbi, apbo => apbo(8),
760       gpioi => gpioi, gpioo => gpioo);
761       gpio_pads : iopadvv generic map (tech => padtech, width => 13)
762         port map (gpio, gpioo.dout(12 downto 0), gpioo.oen(12 downto 0),
763         gpioi.din(12 downto 0));
764     end generate;
765
766     ahbs : if CFG_AHBSTAT = 1 generate   -- AHB status register
767       ahbstat0 : ahbstat generic map (pindex => 15, paddr => 15, pirq => 1,
768      nftslv => CFG_AHBSTATN)
769         port map (rstn, clkm, ahbmi, ahbsi, stati, apbi, apbo(15));
770     end generate;
771
772     i2cm: if CFG_I2C_ENABLE = 1 generate  -- I2C master
773       i2c0 : i2cmst
774       generic map (pindex => 12, paddr => 12, pmask => 16#FFF#,
775                    pirq => 11, filter => I2C_FILTER)
776       port map (rstn, clkm, apbi, apbo(12), i2ci, i2co);
777       i2c_scl_pad : iopad generic map (tech => padtech)
778         port map (iic_scl_main, i2co.scl, i2co.scloen, i2ci.scl);
779       i2c_sda_pad : iopad generic map (tech => padtech)
780         port map (iic_sda_main, i2co.sda, i2co.sdaoen, i2ci.sda);
781     end generate i2cm;
782
783   ----------------------------------------------------------------------
784   ---  ETHERNET ------------------------------------------------------
785   ----------------------------------------------------------------------
786
787       eth1 : if CFG_GRETH = 1 generate -- Gaisler ethernet MAC
788         e1 : grethm generic map(hindex => NCPU+CFG_AHB_UART+CFG_AHB_JTAG+CFG_SVGA_ENABLE
    ,
789      pindex => 11, paddr => 11, pirq => 7, memtech => memtech,
790           mdcscaler => CPU_FREQ/1000, enable_mdio => 1, fifosize => CFG_ETH_FIFO,
```

```
791           nsync => 1, edcl => CFG_DSU_ETH, edclbufsz => CFG_ETH_BUF,
792           macaddrh => CFG_ETH_ENM, macaddrl => CFG_ETH_ENL, phyrstadr => 7,
793       ipaddrh => CFG_ETH_IPM, ipaddrl => CFG_ETH_IPL, giga => CFG_GRETH1G,
794       enable_mdint => 1)
795          port map( rst => rstn, clk => clkm, ahbmi => ahbmi,
796            ahbmo => ahbmo(NCPU+CFG_AHB_UART+CFG_AHB_JTAG+CFG_SVGA_ENABLE),
797       apbi => apbi, apbo => apbo(11), ethi => ethi, etho => etho);
798
799          emdio_pad : iopad generic map (tech => padtech)
800          port map (phy_mii_data, etho.mdio_o, etho.mdio_oe, ethi.mdio_i);
801          etxc_pad : clkpad generic map (tech => padtech, arch => 2)
802       port map (phy_tx_clk, ethi.tx_clk);
803          erxc_pad : clkpad generic map (tech => padtech, arch => 2)
804       port map (phy_rx_clk, ethi.rx_clk);
805          erxd_pad : inpadv generic map (tech => padtech, width => 8)
806       port map (phy_rx_data, ethi.rxd(7 downto 0));
807          erxdv_pad : inpad generic map (tech => padtech)
808       port map (phy_dv, ethi.rx_dv);
809          erxer_pad : inpad generic map (tech => padtech)
810       port map (phy_rx_er, ethi.rx_er);
811          erxco_pad : inpad generic map (tech => padtech)
812       port map (phy_col, ethi.rx_col);
813          erxcr_pad : inpad generic map (tech => padtech)
814       port map (phy_crs, ethi.rx_crs);
815
816          etxd_pad : outpadv generic map (tech => padtech, width => 8)
817       port map (phy_tx_data, etho.txd(7 downto 0));
818          etxen_pad : outpad generic map (tech => padtech)
819       port map ( phy_tx_en, etho.tx_en);
820          etxer_pad : outpad generic map (tech => padtech)
821       port map (phy_tx_er, etho.tx_er);
822          emdc_pad : outpad generic map (tech => padtech)
823       port map (phy_mii_clk, etho.mdc);
824          erst_pad : outpad generic map (tech => padtech)
825       port map (phy_rst_n, rstn);
826          emdintn_pad : inpad generic map (tech => padtech)
827            port map (phy_int, ethi.mdint);
828
829          ethi.gtx_clk <= egtx_clk;
830
831       end generate;
832   -----------------PCI-EXPRESS-Master-Target----------------------------------------
833       pcie_mt : if CFG_PCIE_TYPE = 1 generate  -- master/target without fifo
834   EP: pcie_master_target_virtex
835     generic map (
836       fabtech          => fabtech,
837       hmstndx          => NCPU+CFG_AHB_UART+CFG_GRETH+CFG_AHB_JTAG+CFG_SVGA_ENABLE,
838       hslvndx          => 6,
839       abits            => 21,
840       device_id        => CFG_PCIEXPDID,        -- PCIE device ID
841       vendor_id        => CFG_PCIEXPVID,  -- PCIE vendor ID
842       pcie_bar_mask    => 16#FFE#,
843       nsync            => 2,   -- 1 or 2 sync regs between clocks
844       haddr            => 16#a00#,
845       hmask            => 16#fff#,
846       pindex           => 10,
847       paddr            => 10,
```

```
848       pmask            => 16#fff#,
849       Master           => CFG_PCIE_SIM_MAS,
850       lane_width       => CFG_NO_OF_LANES
851             )
852     port map(
853       rst              => rstn,
854       clk              => clkm,
855       -- System Interface
856       sys_clk_p        => sys_clk_p,
857       sys_clk_n        => sys_clk_n,
858       sys_reset_n      => sys_reset_n,
859       -- PCI Express Fabric Interface
860       pci_exp_txp      => pci_exp_txp,
861       pci_exp_txn      => pci_exp_txn,
862       pci_exp_rxp      => pci_exp_rxp,
863       pci_exp_rxn      => pci_exp_rxn,
864
865       ahbso            => ahbso(6),
866       ahbsi            => ahbsi,
867       apbi             => apbi,
868       apbo             => apbo(10),
869       ahbmi            => ahbmi,
870       ahbmo            => ahbmo(NCPU+CFG_AHB_UART+CFG_GRETH+CFG_AHB_JTAG+CFG_SVGA_ENABLE
      )
871     );
872       end generate;
873
874   pcie_mf : if CFG_PCIE_TYPE = 3 generate   -- master with fifo and DMA
875   dma:pciedma
876         generic map (fabtech => fabtech, memtech => memtech, dmstndx =>(NCPU+
      CFG_AHB_UART+CFG_GRETH+CFG_AHB_JTAG+CFG_SVGA_ENABLE),
877        dapbndx => 13, dapbaddr => 13,dapbirq => 13, blength => 12, abits => 21,
878
879        device_id => CFG_PCIEXPDID, vendor_id => CFG_PCIEXPVID, pcie_bar_mask => 16#FFE#,
880        slvndx => 6, apbndx => 10, apbaddr => 10, haddr => 16#A00#,hmask=> 16#FFF#,
881        nsync => 2,lane_width => CFG_NO_OF_LANES)
882
883   port map(
884       rst           => rstn,
885       clk           => clkm,
886       -- System Interface
887       sys_clk_p     => sys_clk_p,
888       sys_clk_n     => sys_clk_n,
889       sys_reset_n   => sys_reset_n,
890       -- PCI Express Fabric Interface
891       pci_exp_txp   => pci_exp_txp,
892       pci_exp_txn   => pci_exp_txn,
893       pci_exp_rxp   => pci_exp_rxp,
894       pci_exp_rxn   => pci_exp_rxn,
895
896       dapbo         => apbo(13),
897       dahbmo        => ahbmo(NCPU+CFG_AHB_UART+CFG_GRETH+CFG_AHB_JTAG+CFG_SVGA_ENABLE),
898       apbi          => apbi,
899       apbo          => apbo(10),
900       ahbmi         => ahbmi,
901       ahbsi         => ahbsi,
902       ahbso         => ahbso(6)
```

```
903
904       );
905         end generate;
906     ----------------------------------------------------------------------
907     pcie_mf_no_dma: if CFG_PCIE_TYPE = 2 generate   -- master with fifo
908     EP:pcie_master_fifo_virtex
909     generic map (fabtech => fabtech, memtech => memtech,
910        hslvndx => 6, abits => 21, device_id => CFG_PCIEXPDID, vendor_id => CFG_PCIEXPVID,
911        pcie_bar_mask => 16#FFE#, pindex => 10, paddr => 10,
912      haddr => 16#A00#, hmask => 16#FFF#, nsync => 2, lane_width => CFG_NO_OF_LANES)
913     port map(
914         rst          => rstn,
915         clk          => clkm,
916         -- System Interface
917         sys_clk_p    => sys_clk_p,
918         sys_clk_n    => sys_clk_n,
919         sys_reset_n  => sys_reset_n,
920         -- PCI Express Fabric Interface
921         pci_exp_txp  => pci_exp_txp,
922         pci_exp_txn  => pci_exp_txn,
923         pci_exp_rxp  => pci_exp_rxp,
924         pci_exp_rxn  => pci_exp_rxn,
925
926         ahbso        => ahbso(6),
927         ahbsi        => ahbsi,
928         apbi         => apbi,
929         apbo         => apbo(10)
930       );
931     end generate;
932
933     ----------------------------------------------------------------------
934     ---  SYSTEM MONITOR --------------------------------------------------
935     ----------------------------------------------------------------------
936
937       grsmon: if CFG_GRSYSMON = 1 generate
938         sysm0 : grsysmon generic map (tech => fabtech, hindex => 5,
939              hirq => 10, caddr => 16#003#, cmask => 16#fff#,
940              saddr => 16#004#, smask => 16#ffe#, split => CFG_SPLIT,
941              extconvst => 0, wrdalign => 1, INIT_40 => X"0000",
942              INIT_41 => X"0000", INIT_42 => X"0800", INIT_43 => X"0000",
943              INIT_44 => X"0000", INIT_45 => X"0000", INIT_46 => X"0000",
944              INIT_47 => X"0000", INIT_48 => X"0000", INIT_49 => X"0000",
945              INIT_4A => X"0000", INIT_4B => X"0000", INIT_4C => X"0000",
946              INIT_4D => X"0000", INIT_4E => X"0000", INIT_4F => X"0000",
947              INIT_50 => X"0000", INIT_51 => X"0000", INIT_52 => X"0000",
948              INIT_53 => X"0000", INIT_54 => X"0000", INIT_55 => X"0000",
949              INIT_56 => X"0000", INIT_57 => X"0000",
950              SIM_MONITOR_FILE => "sysmon.txt")
951           port map (rstn, clkm, ahbsi, ahbso(5), sysmoni, sysmono);
952         sysmoni <= grsysmon_in_gnd;
953       end generate grsmon;
954
955     ----------------------------------------------------------------------
956     ---  AHB RAM ---------------------------------------------------------
957     ----------------------------------------------------------------------
958
959       ocram : if CFG_AHBRAMEN = 1 generate
```

```
960       ahbram0 : ahbram generic map (hindex => 7, haddr => CFG_AHBRADDR,
961      tech => CFG_MEMTECH, kbytes => CFG_AHBRSZ, pipe => CFG_AHBRPIPE)
962       port map ( rstn, clkm, ahbsi, ahbso(7));
963     end generate;
964
965   ----------------------------------------------------------------------
966   ---  TFG_LUISMI ------------------------------------------------------
967   ----------------------------------------------------------------------
968
969         TFG_IMAGE_SCALING_1 : TFG_IMAGE_SCALING
970         generic map (FABTECH          => fabtech,
971                      MEMTECH          => memtech,
972                      PINDEX           => 10,
973                      PADDR            => 10,
974                      PMASK            => 16#0ff#,
975                      HINDEX           => (NCPU+CFG_AHB_UART+CFG_GRETH+CFG_AHB_JTAG+
      CFG_SVGA_ENABLE),
976                      --AHBACCSZ          =>
977                      --BURSTLEN          =>
978                      AHB_SLV_HINDEX   => 7,
979                      AHB_SLV_HADDR    => 16#a00#,
980                      --AHB_SLV_HMASK  =>
981                      HIRQ             => 10,
982
983                      INT_ADDRESS_WIDTH => 11,
984                      SRC_ADDRESS_WIDTH => 9,
985                      DST_ADDRESS_WIDTH => 9,
986                      ADDRESS_HEIGHT    => 11)
987         port map (Rst_n   => rstn,
988                   Clk     => clkm,
989                   Apbi    => apbi,
990                   Apbo    => apbo(10),
991                   Ahbi    => ahbmi,
992                   Ahbo    => ahbmo(NCPU+CFG_AHB_UART+CFG_GRETH+CFG_AHB_JTAG+
      CFG_SVGA_ENABLE));
993
994   ----------------------------------------------------------------------
995   ---  AHB DEBUG -------------------------------------------------------
996   ----------------------------------------------------------------------
997
998   --  dma0 : ahbdma
999   --    generic map (hindex => CFG_NCPU+CFG_AHB_UART+CFG_GRETH+CFG_AHB_JTAG,
1000  -- pindex => 13, paddr => 13, dbuf => 6)
1001  --    port map (rstn, clkm, apbi, apbo(13), ahbmi,
1002  -- ahbmo(CFG_NCPU+CFG_AHB_UART+CFG_GRETH+CFG_AHB_JTAG));
1003
1004  --  at0 : ahbtrace
1005  --  generic map ( hindex  => 7, ioaddr => 16#200#, iomask => 16#E00#,
1006  --    tech    => memtech, irq     => 0, kbytes  => 8)
1007  --  port map ( rstn, clkm, ahbmi, ahbsi, ahbso(7));
1008
1009  ----------------------------------------------------------------------
1010  ---  Drive unused bus elements  --------------------------------------
1011  ----------------------------------------------------------------------
1012
1013  --  nam1 : for i in (NCPU+CFG_AHB_UART+CFG_ETH+CFG_AHB_ETH+CFG_AHB_JTAG+CFG_PCIEXP)
      to NAHBMST-1 generate
```

```
1014   --    ahbmo(i) <= ahbm_none;
1015   --  end generate;
1016   --  nap0 : for i in 11 to NAPBSLV-1 generate apbo(i) <= apb_none; end generate;
1017   --  nah0 : for i in 8 to NAHBSLV-1 generate ahbso(i) <= ahbs_none; end generate;
1018
1019   ----------------------------------------------------------------------
1020   ---  Boot message  ---------------------------------------------------
1021   ----------------------------------------------------------------------
1022
1023   -- pragma translate_off
1024     x : report_design
1025     generic map (
1026      msg1 => system_table(CFG_BOARD_SELECTION),
1027      fabtech => tech_table(fabtech), memtech => tech_table(memtech),
1028      mdel => 1
1029     );
1030   -- pragma translate_on
1031   end;
1032
```