

```

1  -----
2  -- $Archive:: /VHDL/product/dsk6455/usr_cp1d/dsk6455.vhd                $
3  -- $Revision:: 4                                                         $
4  -- $Date:: 6/01/06 1:18p                                                $
5  -- $Author:: Tonyc                                                       $
6  --
7  --
8  -- Copyright (c) 2002,2003,2004, Spectrum Digital Incorporated
9  -- All rights reserved
10 -----
11
12 -- Start the real code
13 -----
14 library IEEE;
15 use IEEE.std_logic_1164.all;
16
17 entity dsk6455 is
18     port
19     (
20         CLKIN      : in    std_logic; -- 12 MHz clock in
21         OPT_CLK1    : in    std_logic; -- 8 MHz clock in
22         EMU_RSTn    : in    std_logic; -- Emulator reset from USB block
23         PUSHBRS     : in    std_logic; -- Push button reset
24         HPIRSn      : in    std_logic; -- HPI reset from optional DC
25
26         -- DSP Memory interface signals for internal CPLD registers. From C64xx B port.
27         DSP_DQ       : inout std_logic_vector( 7 downto 0 ); -- DSP Data bus
28         DSP_ADDR      : in    std_logic_vector( 3 downto 0 ); -- DSP Address bus
29         DSP_WEn       : in    std_logic; -- DSP Write strobe
30         DSP_ARNW      : in    std_logic; -- DSP Read strobe
31         DSP_OEn       : in    std_logic; -- DSP Output enable
32
33         DSP_CS2n      : in    std_logic;
34         DSP_CS3n      : in    std_logic;
35         DSP_CS4n      : in    std_logic;
36         DSP_CS5n      : in    std_logic;
37
38         -- User/Board Support
39         USER_SW       : in    std_logic_vector( 3 downto 0 ); -- User switches
40         USER_LED      : out    std_logic_vector( 3 downto 0 ); -- Uwer led
41         PWB_REV       : in    std_logic_vector( 2 downto 0 ); -- PWB revision
42
43         -- Daughter Card Support
44         DC_STAT       : in    std_logic_vector( 1 downto 0 ); -- DC Status
45         DC_CNTL       : out    std_logic_vector( 1 downto 0 ); -- DC Control
46         DC_DBUF_DIR    : out    std_logic; -- DC Data buffer direction
47         DC_DBUF_OEn    : out    std_logic; -- DC Data buffer output enable
48         DC_CNTL_OEn    : out    std_logic; -- DC Control buffer enable
49         DC_DETn       : in    std_logic; -- DC Detec -- DC Reset
50         DC_RESEn      : out    std_logic;
51         -- MCBSP Multiplexer Control
52         MCBSP_SELA    : out    std_logic; -- Codec/DC MCBsp 1 mux cntl
53
54         -- Misc. Stuff

```

```

55     DSP_RS_n_LED : out    std_logic;           -- DSP reset led
56     --FLASH_PAGE0 : out    std_logic;           -- Flash Address 19
57     --FLASH_PAGE1 : out    std_logic;           -- Flash Address 20
58     ABA0          : in     std_logic;           -- Flash Address 19
59     ABA1          : in     std_logic;           -- Flash Address 20
60
61     CPLD_CLK_OUT : out     std_logic;           -- Place holder
62
63     -- CLOCKING FOR EMIF AND CPU
64     TEST_PIN      : out     std_logic;
65     DSP_PLL_CLK   : out     std_logic;
66     DSPPLL_SW     : in      std_logic_vector( 3 downto 1 );
67     CPU_DSPPLL_S  : out     std_logic_vector( 1 downto 0 );
68     EMIF_PLL_CLK  : out     std_logic;
69
70     -- CPUB, MISC2
71     CPUB_PRESENTn : in      std_logic;
72     CPUB_RESETh   : out     std_logic;
73     HURRICANE_RESETh : in    std_logic;
74     DSPA_PORZ     : out     std_logic;
75     DSPA_RESETh   : out     std_logic;
76     DSPA_RESEThSTATn : in    std_logic;
77     MII_RESETh    : out     std_logic;
78     BD_PWR_ON_RSZ : in      std_logic;
79     SVS_RSTn      : out     std_logic;
80     BRD_RSTn      : out     std_logic;
81
82 );
83 end dsk6455;
84
85 -----
86 -- Include standard libraries
87 -----
88 library IEEE;
89 use IEEE.std_logic_1164.all;
90 -- use work.std_arith.all;
91 use IEEE.std_logic_arith.all;
92 use IEEE.std_logic_unsigned.all;
93
94 -----
95 -- Include fpga specifics here if required.
96 -- act3 is for Actel 54sx devices. We normally use this for the hardwired
97 -- clock definition.
98 -----
99 -- library act3;
100 -- use act3.components.all;
101
102 architecture behavior_dsk6455 of dsk6455 is
103
104     constant CPLD_VERSION : std_logic_vector(3 downto 0) := "0011";
105
106     -----
107     -- Add local components in here
108     -----

```

```
109 --component MyComponent
110 --port
111 --(
112 --);
113 --end component;
114
115 -----
116 -- Add signals
117 -----
118
119 -- CPLD Register signals
120 signal Cp1dReg0      : std_logic_vector( 7 downto 0 );
121 signal Cp1dReg1      : std_logic_vector( 7 downto 0 );
122 signal Cp1dReg4      : std_logic_vector( 7 downto 0 );
123 signal Cp1dReg6      : std_logic_vector( 7 downto 0 );
124 signal Cp1dReg7      : std_logic_vector( 7 downto 0 );
125 signal OPT_CLK1_DIV  : std_logic;
126 signal MuxD          : std_logic_vector( 7 downto 0 );
127
128 signal ChipEnables   : std_logic_vector( 7 downto 0 );
129 signal Cp1dRegCs0    : std_logic;
130 signal Cp1dRegCs1    : std_logic;
131 signal Cp1dRegCs2    : std_logic;
132 signal Cp1dRegCs3    : std_logic;
133 signal Cp1dRegCs4    : std_logic;
134 signal Cp1dRegCs5    : std_logic;
135 signal Cp1dRegCs6    : std_logic;
136 signal Cp1dRegCs7    : std_logic;
137
138 signal Cp1dClkout    : std_logic;
139
140 signal DcCs0         : std_logic;
141 signal DcCs1         : std_logic;
142 signal Cp1dCs        : std_logic;
143 signal FlashCs       : std_logic;
144
145 signal HurricaneResetn : std_logic;
146 signal PowOnRs        : std_logic;
147 signal NewDspAddr     : std_logic_vector(3 downto 0 );
148
149 -----
150 -- The implementation
151 -----
152
153 begin
154 -----
155 -- Map the other components
156 -----
157
158 -----
159 -- Now define the logic
160 -----
161
162 -- Map signals to generic useage
```

```

163 Cp1dCs <= DSP_CS2n;
164 FlashCs <= DSP_CS3n;
165 DcCs0 <= DSP_CS4n;
166 DcCs1 <= DSP_CS5n;
167
168 -- Generate a reset from the three sources.
169 --
170 -- Power on reset sources
171 PowOnRs <= '0' when EMU_RSTn = '0'
172                or BD_PWR_ON_RSZ = '0'
173                or PUSHBRS = '1'
174                else '1';
175
176
177 -- SVS_RSTn is to/from USB regulator. Add in board regulator signal
178 SVS_RSTn <= '0' when BD_PWR_ON_RSZ = '0' else 'Z';
179
180 -- Board reset, flash
181 BRD_RSTn <= '0' when PowOnRs = '0' else '1';
182
183 -- DSP Warm reset + power on for DSP.
184 DSPA_RESETZ <= '0' when PowOnRs = '0'
185                or HPIRSn = '0'
186                or HurricaneResetrn = '0'
187                else '1';
188
189 DSPA_PORZ <= '0' when PowOnRs = '0' else '1';
190
191 -- Put the reset status into the led so that we can see if the dsp is hung
192 -- in reset.
193 --
194 DSP_RSn_LED <= '0' when PowOnRs = '0' or DSPA_RESETSTATn = '0' else '1';
195
196 OPT_CLK1_DIV <= '1'; -- Place holder
197
198
199 -- Generate a CPLD clockout from clock input. This is a place holder just in
200 -- case we need it later.
201 --
202 process( PowOnRs, CLKIN )
203 begin
204     if PowOnRs = '0' then
205         Cp1dClkout <= '0';
206     elsif CLKIN'event and CLKIN = '1' then
207         Cp1dClkout <= not Cp1dClkout;
208     end if;
209 end process;
210
211 CPLD_CLK_OUT <= Cp1dClkout;
212
213 -- #####
214 -- Generic register addresss decode and register chip select generation.
215 -- VHDL compiler will reduce any unused logic so we can be verbose.
216 --

```

```

217 NewDspAddr <= DSP_ADDR(1) & DSP_ADDR(0) & ABA1 & ABA0;
218
219 process( NewDspAddr )
220 begin
221     case NewDspAddr( 3 downto 0 ) is
222         when "0000" => ChipEnables <= "00000001";
223         when "0001" => ChipEnables <= "00000010";
224         when "0010" => ChipEnables <= "00000100";
225         when "0011" => ChipEnables <= "00001000";
226         when "0100" => ChipEnables <= "00010000";
227         when "0101" => ChipEnables <= "00100000";
228         when "0110" => ChipEnables <= "01000000";
229         when "0111" => ChipEnables <= "10000000";
230         when others => ChipEnables <= "00000000";
231     end case;
232 end process;
233
234 Cp1dRegCs0 <= '1' when ChipEnables(0) = '1' and Cp1dCs = '0' else '0';
235 Cp1dRegCs1 <= '1' when ChipEnables(1) = '1' and Cp1dCs = '0' else '0';
236 Cp1dRegCs2 <= '1' when ChipEnables(2) = '1' and Cp1dCs = '0' else '0';
237 Cp1dRegCs3 <= '1' when ChipEnables(3) = '1' and Cp1dCs = '0' else '0';
238 Cp1dRegCs4 <= '1' when ChipEnables(4) = '1' and Cp1dCs = '0' else '0';
239 Cp1dRegCs5 <= '1' when ChipEnables(5) = '1' and Cp1dCs = '0' else '0';
240 Cp1dRegCs6 <= '1' when ChipEnables(6) = '1' and Cp1dCs = '0' else '0';
241 Cp1dRegCs7 <= '1' when ChipEnables(7) = '1' and Cp1dCs = '0' else '0';
242
243 -- #####
244 -- Generate logic for each CPLD register and assign it's write, read, and
245 -- pin values if necessary.
246 --
247 -- All CPLD register writes occur on the rising edge DSP write strobe.
248 --
249
250 -- =====
251 -- REG 0: User Register
252 -- Bit 3-0    Led 3-0
253 -- Bit 7-4    Switch 3-0
254 process( PowOnRs, DSP_WEn, Cp1dRegCs0, DSP_DQ )
255 begin
256     if PowOnRs = '0' then
257         Cp1dReg0(3 downto 0) <= "0000";
258     elsif DSP_WEn'event and DSP_WEn = '1' then
259         if( Cp1dRegCs0 = '1' ) then
260             Cp1dReg0( 3 downto 0 ) <= DSP_DQ( 3 downto 0 );
261         end if;
262     end if;
263 end process;
264
265 Cp1dReg0(7 downto 4) <= USER_SW( 3 downto 0 );
266 USER_LED( 3 downto 0 ) <= not Cp1dReg0(3 downto 0 );
267
268 -- =====
269 -- REG 1: DC Register
270 -- Bit 1-0    DC_CNTL 1-0

```

```

271 -- Bit 2      NU read 0
272 -- Bit 3      DC_RESET
273 -- Bit 5-4    DC_STAT 1-0
274 -- Bit 6      NU read 0
275 -- Bit 7      DC_DETECT
276 --
277 process( PowOnRs, DSP_WEn, Cp1dRegCs1, DSP_DQ )
278 begin
279   if PowOnRs = '0' then
280     Cp1dReg1(1 downto 0) <= "00";
281     Cp1dReg1(3) <= '0'; -- not Reset by default
282   elsif DSP_WEn'event and DSP_WEn = '1' then
283     if( Cp1dRegCs1 = '1' ) then
284       Cp1dReg1( 1 downto 0 ) <= DSP_DQ( 1 downto 0 );
285       Cp1dReg1(3) <= DSP_DQ(3);
286     end if;
287   end if;
288 end process;
289
290
291 Cp1dReg1(2) <= '0';
292 Cp1dReg1(5 downto 4) <= DC_STAT( 1 downto 0 );
293 Cp1dReg1(6) <= '0';
294 Cp1dReg1(7) <= not DC_DETn;
295
296 DC_CNTL( 1 downto 0 ) <= Cp1dReg1( 1 downto 0 );
297
298 -- HPIRSn not included in the DC_RESETn equation. This should prevent the
299 -- DC from holding itself in reset if HPIRSn is active.
300 DC_RESETn <= '0' when Cp1dReg1(3) = '1'
301                  or PowOnRs = '0' else '1';
302
303 -- =====
304 -- REG 4: Version Register
305 -- Bit 2-0    PWB Revision 2-0
306 -- Bit 3      NU read 0
307 -- Bit 7-4    CPLD version
308 Cp1dReg4(7 downto 0) <= CPLD_VERSION(3 downto 0) & '0' & PWB_REV(2 downto 0);
309
310 -- =====
311 -- REG 6: Misc. Register
312 -- Bit 0      MCBSP1 select
313 -- Bit 1      Flash Page/Flash Address 19
314 -- Bit 2      Flash Page/Flash Address 20
315 -- Bit 3      DSPPLL_SELECT1 (READ) SW3-5
316 -- Bit 4      DSPPLL_SELECT2 (READ) SW3-6
317 -- Bit 5      DSPPLL_SELECT3 (READ) SW3-7
318 -- Bit 6      DSPPLL_SELECT4 (READ) SW3-8
319 -- Bit 7      NU
320 --
321 process( PowOnRs, DSP_WEn, Cp1dRegCs6, DSP_DQ )
322 begin
323   if PowOnRs = '0' then
324     Cp1dReg6(0) <= '0';

```

```

325     Cp1dReg6(1) <= '0';
326     Cp1dReg6(2) <= '0';
327
328
329     elsif DSP_WEn'event and DSP_WEn = '1' then
330         if( Cp1dRegCs6 = '1' ) then
331             Cp1dReg6(0) <= DSP_DQ(0);
332             Cp1dReg6(1) <= DSP_DQ(1);
333             Cp1dReg6(2) <= DSP_DQ(2);
334         end if;
335     end if;
336 end process;
337
338 -- Low = DC/UTOPIA
339 -- High = Codec
340 MCBSP_SELA <= '1' when Cp1dReg6(0) = '0' else '0';
341
342 Cp1dReg6(3) <= '0';
343 Cp1dReg6(4) <= DSPPLL_SW(1);
344 Cp1dReg6(5) <= DSPPLL_SW(2);
345 Cp1dReg6(6) <= DSPPLL_SW(3);
346
347 -----
348 -- Mapping ICS521 S1/S0
349 -----
350 -- S1 S0 MULTIPLIER
351 -- 0 0 4
352 -- 0 Z 5.333
353 -- 0 1 5
354 -- Z 0 2.5
355 -- Z Z 2
356 -- Z 1 3.333
357 -- 1 0 6
358 -- 1 Z 3
359 -- 1 1 8
360 -----
361 -- Note Logically Switch On = 0 Off = 1
362 -----
363
364 CPU_DSPPLL_S <= "00";
365 DSP_PLL_CLK <= '0';
366
367 Cp1dReg6(7) <= '0';
368
369 -----
370 -- REG 7: MISC2
371 -- Bit 0 CPUB_PRESENT
372 -- Bit 1 CPUB_RESET
373 -- Bit 2 MII_RESETh
374 -- Bit 3 HURRICANE_RESETh enable
375 -- Bit 4
376 -- Bit 5
377 -- Bit 6
378 -- Bit 7 DSPA_RESETh

```

```

379 process( PowOnRs, DSP_WEn, Cp1dRegCs7, DSP_DQ )
380 begin
381     if PowOnRs = '0' then
382         Cp1dReg7(1) <= '0';
383         Cp1dReg7(2) <= '0';
384         Cp1dReg7(3) <= '0';
385         Cp1dReg7(4) <= '0';
386         Cp1dReg7(5) <= '0';
387         Cp1dReg7(6) <= '0';
388     elsif DSP_WEn'event and DSP_WEn = '1' then
389         if( cp1dRegCs7 = '1' ) then
390             Cp1dReg7(1) <= DSP_DQ(1);
391             Cp1dReg7(2) <= DSP_DQ(2);
392             Cp1dReg7(3) <= DSP_DQ(3);
393             Cp1dReg7(4) <= DSP_DQ(4);
394             Cp1dReg7(5) <= DSP_DQ(5);
395             Cp1dReg7(6) <= DSP_DQ(6);
396         end if;
397     end if;
398 end process;
399
400 Cp1dReg7(0) <= not CPUB_PRESENTn;
401 CPUB_RESETn <= '1' when Cp1dReg7(1) = '1' or PowOnRs = '0' else '0';
402 MII_RESETZ <= '0' when Cp1dReg7(2) = '1' or PowOnRs = '0' else '1';
403 HurricaneResetn <= '0' when Cp1dReg7(3) = '1' and HURRICANE_RESETn = '0' else '1';
404
405 Cp1dReg7(7) <= DSPA_RESETSTATn;
406
407
408 -- =====
409 -- Mux the read data from all the registers and output for reads
410 --
411 process( NewDspAddr, Cp1dReg0, Cp1dReg1, Cp1dReg4, Cp1dReg6, Cp1dReg7 )
412 begin
413     case NewDspAddr( 3 downto 0 ) is
414         when "0000" => MuxD <= Cp1dReg0;
415         when "0001" => MuxD <= Cp1dReg1;
416         when "0100" => MuxD <= Cp1dReg4;
417         when "0110" => MuxD <= Cp1dReg6;
418         when "0111" => MuxD <= Cp1dReg7;
419         when others => MuxD <= "00000000";
420     end case;
421 end process;
422
423 DSP_DQ <= MuxD when
424     and DSP_ARNW = '1'
425     and DSP_OEn = '0'
426     else "zzzzzzzz";
427
428 -- #####
429 -- Generate the Daughter card buffer control signals. DC buffers are only
430 -- enabled if a daughter card is plugged in to minimize EMI.
431 --
432 -- DSP ARNW signal is low for write and low for write. We flip this to match

```



```
433  -- the 245 direction control.
434  --
435  -- #####
436
437  DC_DBUF_DIR <= not DSP_ARNW; -- low for write, high for read
438
439  DC_DBUF_OEn <= '0' when DC_DETn = '0'
440                  and ( DcCs0 = '0' or DcCs1 = '0' )
441                  and ( DSP_OEn = '0' or DSP_WEn = '0' )
442                  else '1';
443
444  DC_CNTL_OEn <= '0' when DC_DETn = '0' else '1';
445
446  TEST_PIN <= FlashCs;
447
448  EMIF_PLL_CLK <= CLKIN;
449
450 end behavior_dsk6455;
451
452
```