

```

1  -----
2  -- $Archive:: /VHDL/product/dsk5509/cntl_cp1d/cntlcp1d.vhd          $
3  -- $Revision:: 3                                                    $
4  -- $Date:: 12/22/05 3:09p                                           $
5  -- $Author:: Tonyc                                                  $
6  --
7  -- CLPD Rev 0001 => Alpha Build
8  -- CLPD Rev 0010 => Production build
9  --
10 -----
11 -----
12 -- Start the real code
13 -----
14 library IEEE;
15 use IEEE.std_logic_1164.all;
16 --
17 --
18 --
19
20 entity cntlcp1d is
21 port
22 (
23     CLKIN      : in std_logic;          -- 20 MHz clock in
24
25     PONRSn     : in std_logic;          -- Power on reset from voltage supervisor
26     PON_CORE   : in std_logic;          -- Core Power Status
27     PUSHBRS    : in std_logic;          -- Push button reset
28     USB_DSP_RS : in std_logic;
29     USB_PORSTn : out std_logic;
30
31     -- DSP Memory interface signals
32     DSP_DQ     : inout std_logic_vector( 7 downto 0 ); -- DSP Data bus
33     DSP_ADDR   : in std_logic_vector( 4 downto 1 ); -- DSP Address bus
34     DSP_ADDR20 : in std_logic;          -- Upper DSP Address
35     DSP_ADDR19 : in std_logic;
36     DSP_ADDR18 : in std_logic;
37     DSP_ADDR17 : in std_logic;
38
39     -- DSP Control strobes
40     DSP_WEn    : in std_logic;          -- DSP Write strobe
41     DSP_REn    : in std_logic;          -- DSP Read strobe
42     DSP_OEn    : in std_logic;          -- DSP Output enable
43
44     -- DSP Reset
45     DSP_RSn    : out std_logic;         -- DSP Reset
46     DSP_RS_LEDn : out std_logic;        -- DSP Reset LED
47     FLASH_CEn  : out std_logic;
48     FLASH_RSn  : out std_logic;
49
50     -- DSP Memory Chip Select signals
51     DSP_CS3n   : in std_logic;          -- DSP DC Chip select
52     DSP_CS2n   : in std_logic;          -- DSP DC Chip Select 2
53     DSP_CS1n   : in std_logic;          -- Flash Chip Select 2
54

```

```

55  -- DSP GPIO Lines
56  DSP_GP6      : in std_logic;
57  DSP_GP5      : in std_logic;
58  DSP_TINOUT_0 : inout std_logic;
59
60  DSP_INT0      : out std_logic;
61  DSP_INT1      : out std_logic;
62  DSP_INT3      : out std_logic;
63
64  -- User/Board Support
65  USER_SW      : in std_logic_vector( 3 downto 0 );
66  USER_LED      : out std_logic_vector( 3 downto 0 );
67
68  -- Daughter Card Support
69  DC_STAT      : in std_logic_vector( 1 downto 0 ); -- DC Status
70  DC_CNTL      : out std_logic_vector( 1 downto 0 ); -- DC Control
71  DC_DETn      : in std_logic; -- DC Detect
72  DC_POR        : in std_logic;
73  DC_RESETh     : out std_logic; -- DC Reset
74  DC_DBUF_OEn   : out std_logic; -- DC Data buffer output enable
75  DC_CNTL_OEn   : out std_logic; -- DC Control buffer enable
76  DC_DBUF_DIR   : out std_logic;
77  X_TIN0        : in std_logic;
78  X_TOUT0       : inout std_logic;
79
80  -- Daughter Card Interface
81  DC_INT3n      : in std_logic;
82  DC_INT1n      : in std_logic;
83  DC_INT0n      : in std_logic;
84
85  -- Voltage Control
86  CORE_VCNTL0   : out std_logic;
87  CORE_VCNTL1   : out std_logic;
88
89  --
90  WAKE_UP_SW    : in std_logic;
91
92  -- Sample period generator
93  ENABLE        : in std_logic;
94  VARCLK         : in std_logic;
95  NI_DIG        : out std_logic_vector( 7 downto 0 );
96  USB_XF        : in std_logic;
97  EVENT_TRG     : out std_logic;
98  CAPTURE_POWER_ON : in std_logic;
99 );
100
101 end cntlcpId;
102
103 -----
104 -- Include standard libraries
105 -----
106 library IEEE;
107 use IEEE.std_logic_1164.all;
108 use IEEE.std_logic_arith.all;

```

```
109 use IEEE.std_logic_unsigned.all;
110
111 -----
112 -- Include fpga specifics here if required.
113 -- act3 is for Actel 54sx devices. We normally use this for the hardwired
114 -- clock definition.
115 -----
116
117 architecture behavior_cntlcpld of cntlcpld is
118
119 constant CPLD_VERSION : std_logic_vector(3 downto 0) := "0010";
120 constant PWB_REV       : std_logic_vector(3 downto 0) := "0000";
121
122 -----
123 -- Add local components in here
124 -----
125 --component MyComponent
126 --port
127 --(
128 --);
129 --end component;
130
131 -----
132 -- Add signals
133 -----
134
135 -- CPLD Register signals
136 signal REG_CEn      : std_logic;
137 signal CpldReg0      : std_logic_vector( 7 downto 0 );
138 signal CpldReg1      : std_logic_vector( 7 downto 0 );
139 signal CpldReg4      : std_logic_vector( 7 downto 0 );
140 signal CpldReg6      : std_logic_vector( 7 downto 0 );
141 signal CpldReg7      : std_logic_vector( 7 downto 0 );
142 signal MuxD          : std_logic_vector( 7 downto 0 );
143
144 signal ChipEnables   : std_logic_vector( 10 downto 0 );
145 signal CpldRegCs0     : std_logic;
146 signal CpldRegCs1     : std_logic;
147 signal CpldRegCs2     : std_logic;
148 signal CpldRegCs3     : std_logic;
149 signal CpldRegCs4     : std_logic;
150 signal CpldRegCs5     : std_logic;
151 signal CpldRegCs6     : std_logic;
152 signal CpldRegCs7     : std_logic;
153 signal CpldRegCs8     : std_logic;
154 signal CpldRegCs9     : std_logic;
155
156 signal POWER_UP      : std_logic;
157
158 signal SystemReseth   : std_logic;
159 signal NiOe           : std_logic;
160 signal NiEnable       : std_logic;
161 signal NiForceOn      : std_logic;
162
```

```

163 -- SAMPLE RATE GENERATOR
164 -- High time is 255-47 = 208
165 -- Low time is 47
166 -- This supports a 24MHz FXCLK and 3MHz VARCLK which is a 8:1 ratio. We
167 -- then need 6-VARCLK of high and low clock to get good rising and falling
168 -- edge detection between two clock domains and a capture pulse.
169 --
170 -- VARCLKS per sample period with 208 FXCLK
171 -- 24:3 = 26 0x0_0001_1010
172 -- 24:6 = 52 0x0_0011_0100
173 -- 24:12 = 104 0x0_0110_1000
174 -- 24:24 = 208 0x0_1101_0000
175 -- 24:48 = 416 0x1_1010_0000
176 --
177 constant SAMPLE_HIGH      : std_logic_vector( 7 downto 0 ) := "00101111";
178 constant SAMPLE_LOW      : std_logic_vector( 7 downto 0 ) := "11111111";
179
180 signal SampleCounter      : std_logic_vector( 7 downto 0 );
181 signal SampleEnable      : std_logic;
182
183 signal VarCounter        : std_logic_vector( 8 downto 0 );
184 signal VarEnable         : std_logic;
185 signal VarSyncEnable     : std_logic_vector( 3 downto 0 );
186 signal VarCapture        : std_logic;
187 signal VarLatch          : std_logic_vector( 4 downto 0 );
188
189 signal VControl          : std_logic_vector( 1 downto 0 );
190 signal FXCLK             : std_logic;
191
192
193 -----
194 -- The implementation
195 -----
196
197 begin
198
199 -----
200 -- Actel specific definition for the hardwired clock "hclk"
201 -----
202 -- u1: HCLKBUF port map( PAD => iMasterClockIn, Y => oMasterClockIn );
203
204 -----
205 -- Map the other components
206 -----
207 --u2: my_component port map ( ComponentSignalName => SignalConnection, ... );
208
209 -----
210 -- Now define the logic
211 -----
212
213 -- Generate a reset from the three sources. May need to deglitch based on
214 -- external implementation.
215 --
216 -- Generate a CPLD clockout from clock input. This is a place holder just in

```

```

217  -- case we need it later.
218  --
219  SystemResetn <= '0' when ( PONRSn = '0' or PUSHBRS = '1' or USB_DSP_RS = '0' ) else '1';
220
221  DSP_RS_n    <= '0' when SystemResetn = '0' else '1';
222  DSP_RS_LEDn <= '0' when SystemResetn = '0' else '1';
223  FLASH_RS_n  <= '0' when SystemResetn = '0' else '1';
224
225  USB_PORSTn  <= '0' when PONRSn = '0' else '1';
226
227  FLASH_CEn   <= '0' when ( ( DSP_CS1n = '0' and DSP_ADDR20 = '0' ) or
228                           ( DSP_CS1n = '0' and DSP_ADDR20 = '1' and DSP_ADDR19 = '0' ) or
229                           ( DSP_CS1n = '0' and DSP_ADDR20 = '1' and DSP_ADDR18 = '0' ) or
230                           ( DSP_CS1n = '0' and DSP_ADDR20 = '1' and DSP_ADDR17 = '0' ) )
231                      else '1';
232
233  REG_CEn     <= '0' when ( DSP_CS1n = '0' and DSP_ADDR20 = '1' and
234                           DSP_ADDR19 = '1' and DSP_ADDR18 = '1' and
235                           DSP_ADDR17 = '1' )
236                      else '1';
237
238  --#####
239  -- Generic register addresss decode and register chip select generation.
240  -- VHDL compiler will reduce any unused logic so we can be verbose.
241  --
242  process( DSP_ADDR )
243  begin
244      case DSP_ADDR( 4 downto 1) is
245          when "0000" => ChipEnables <= "00000000001";
246          when "0001" => ChipEnables <= "00000000010";
247          when "0010" => ChipEnables <= "00000000100";
248          when "0011" => ChipEnables <= "00000001000";
249          when "0100" => ChipEnables <= "00000010000";
250          when "0101" => ChipEnables <= "00000100000";
251          when "0110" => ChipEnables <= "00001000000";
252          when "0111" => ChipEnables <= "00010000000";
253          when "1000" => ChipEnables <= "00100000000";
254          when "1001" => ChipEnables <= "01000000000";
255          when "1010" => ChipEnables <= "10000000000";
256          when others => ChipEnables <= "00000000000";
257      end case;
258  end process;
259
260  Cp1dRegCs0 <= '1' when ChipEnables(0) = '1' and REG_CEn = '0' else '0';
261  Cp1dRegCs1 <= '1' when ChipEnables(1) = '1' and REG_CEn = '0' else '0';
262  Cp1dRegCs2 <= '1' when ChipEnables(2) = '1' and REG_CEn = '0' else '0';
263  Cp1dRegCs3 <= '1' when ChipEnables(3) = '1' and REG_CEn = '0' else '0';
264  Cp1dRegCs4 <= '1' when ChipEnables(4) = '1' and REG_CEn = '0' else '0';
265  Cp1dRegCs5 <= '1' when ChipEnables(5) = '1' and REG_CEn = '0' else '0';
266  Cp1dRegCs6 <= '1' when ChipEnables(6) = '1' and REG_CEn = '0' else '0';
267  Cp1dRegCs7 <= '1' when ChipEnables(7) = '1' and REG_CEn = '0' else '0';
268  Cp1dRegCs8 <= '1' when ChipEnables(8) = '1' and REG_CEn = '0' else '0';
269  Cp1dRegCs9 <= '1' when ChipEnables(9) = '1' and REG_CEn = '0' else '0';
270

```

```

271
272 -- #####
273 -- Generate logic for each CPLD register and assign it's write, read, and
274 -- pin values if necessary.
275 --
276 -- All CPLD register writes occur on the rising edge DSP write strobe.
277 --
278
279 -- =====
280 -- REG 0: User Register
281 -- Bit 3-0   Led 3-0
282 -- Bit 7-4   Switch 3-0
283 process( SystemResetn, DSP_WEn, Cp1dRegCs0, DSP_DQ )
284 begin
285     if SystemResetn = '0' then
286         Cp1dReg0(3 downto 0) <= "0000";
287     elsif DSP_WEn'event and DSP_WEn = '1' then
288         if( Cp1dRegCs0 = '1' ) then
289             Cp1dReg0( 3 downto 0 ) <= DSP_DQ( 3 downto 0 );
290         end if;
291     end if;
292 end process;
293
294 Cp1dReg0(7 downto 4) <= USER_SW( 3 downto 0 );
295 USER_LED( 3 downto 0 ) <= not Cp1dReg0(3 downto 0 );
296
297 -- =====
298 -- REG 1: DC Register
299 -- Bit 1-0   DC_CNTL 1-0
300 -- Bit 2     RESERVED
301 -- Bit 3     DC_RESET
302 -- Bit 5-4   DC_STAT 1-0
303 -- Bit 6     RESERVED
304 -- Bit 7     DC_DETECT
305 --
306 process( SystemResetn, DSP_WEn, Cp1dRegCs1, DSP_DQ )
307 begin
308     if SystemResetn = '0' then
309         Cp1dReg1( 1 downto 0 ) <= "00";
310         Cp1dReg1(3) <= '0';
311     elsif DSP_WEn'event and DSP_WEn = '1' then
312         if( Cp1dRegCs1 = '1' ) then
313             Cp1dReg1( 1 downto 0 ) <= DSP_DQ( 1 downto 0 );
314             Cp1dReg1(3) <= DSP_DQ(3);
315         end if;
316     end if;
317 end process;
318
319 Cp1dReg1(2) <= '0';
320 Cp1dReg1(5 downto 4) <= DC_STAT( 1 downto 0 );
321 Cp1dReg1(6) <= '0';
322 Cp1dReg1(7) <= not DC_DETn;
323
324 DC_CNTL( 1 downto 0 ) <= Cp1dReg1( 1 downto 0 );

```

```

325
326   DC_RESEtN          <= '0' when CpldReg1(3) = '1' or DC_POR = '0' else '1';
327
328   -- =====
329   -- REG 4: Version Register
330   -- Bit 2-0   PWB Revision 2-0
331   -- Bit 3     NU read 0
332   -- Bit 7-4   CPLD version
333
334   CpldReg4(7 downto 0) <= CPLD_VERSION(3 downto 0) & PWB_REV(3 downto 0);
335
336   -- =====
337   -- REG 6: Misc. Register
338   -- Bit 0     RESERVED
339   -- Bit 1     RESERVED
340   -- Bit 2     TIMER IN/OUT SELECT
341   -- Bit 3     EVENT_TRIG (ni section)
342   -- Bit 4     CORE_VCNTL SELECT
343   -- Bit 5     NiForceOn - test bit
344   -- Bit 6     CORE_VCNTL0 VALUE
345   -- Bit 7     CORE_VCNTL1 VALUE
346   process( SystemResetn, DSP_WEn, CpldRegCs6, DSP_DQ )
347   begin
348     if SystemResetn = '0' then
349       CpldReg6 <= "00000000";
350     elsif DSP_WEn'event and DSP_WEn = '1' then
351       if( CpldRegCs6 = '1' ) then
352         CpldReg6 <= DSP_DQ;
353       end if;
354     end if;
355   end process;
356
357   DSP_TINOUT_0      <= X_TIN0      when CpldReg6(2) = '1' else 'Z';
358   X_TOUT0           <= DSP_TINOUT_0 when CpldReg6(2) = '0' else 'Z';
359
360   VControl(0)       <= CpldReg6(6) when CpldReg6(4) = '1' else DSP_GP5;
361   VControl(1)       <= CpldReg6(7) when CpldReg6(4) = '1' else DSP_GP6;
362
363   CORE_VCNTL0       <= VControl(0);
364   CORE_VCNTL1       <= VControl(1);
365   NiForceOn         <= CpldReg6(5);
366
367   -- =====
368   -- REG 7: Interrupt Register
369   -- Bit 0     RESERVED
370   -- Bit 1     RESERVED
371   -- Bit 2     RESERVED
372   -- Bit 3     RESERVED
373   -- Bit 4     RESERVED
374   -- Bit 5     RESERVED
375   -- Bit 6     RESERVED
376   -- Bit 7     RESERVED
377   process( SystemResetn, DSP_WEn, CpldRegCs7, DSP_DQ )
378   begin

```

```

379     if SystemResetn = '0' then
380         CpldReg7 <= "00000000";
381     elsif DSP_WEn'event and DSP_WEn = '1' then
382         if( CpldRegCs7 = '1' ) then
383             CpldReg7 <= DSP_DQ;
384         end if;
385     end if;
386 end process;
387
388 ---
389 POWER_UP <= PON_CORE;
390
391 DSP_INT0 <= '0' when ( ( CpldReg7(0) = '1' and WAKE_UP_SW = '1' ) OR
392                       ( CpldReg7(4) = '1' and POWER_UP = '1' ) OR
393                       ( DC_INT0n = '0' ) )
394                   else '1';
395
396 DSP_INT1 <= '0' when ( ( CpldReg7(1) = '1' and WAKE_UP_SW = '1' ) OR
397                       ( CpldReg7(5) = '1' and POWER_UP = '1' ) OR
398                       ( DC_INT1n = '0' ) )
399                   else '1';
400
401 DSP_INT3 <= '0' when ( ( CpldReg7(3) = '1' and WAKE_UP_SW = '1' ) OR
402                       ( CpldReg7(7) = '1' and POWER_UP = '1' ) OR
403                       ( DC_INT3n = '0' ) )
404                   else '1';
405
406
407 -- =====
408 -- Mux the read data from all the registers and output for reads
409 --
410 process( DSP_ADDR,CpldReg0,CpldReg1,CpldReg4,CpldReg6,CpldReg7,
411         NIOe,CAPTURE_POWER_ON,VarEnable, VarLatch)
412 begin
413     case DSP_ADDR( 4 downto 1) is
414         when "0000" => MuxD <= CpldReg0;
415         when "0001" => MuxD <= CpldReg1;
416         when "0100" => MuxD <= CpldReg4;
417
418         when "0110" => MuxD <= CpldReg6;
419         when "0111" => MuxD <= CpldReg7;
420         when "1000" => MuxD <= NIOe & CAPTURE_POWER_ON & VarEnable & VarLatch(4 downto 0 );
421         when others => MuxD <= "00000000";
422     end case;
423 end process;
424
425 DSP_DQ <= MuxD when DSP_CS1n = '0'
426             and DSP_REn = '0'
427             and DSP_OEn = '0'
428             and DSP_ADDR17 = '1'
429             and DSP_ADDR18 = '1'
430             and DSP_ADDR19 = '1'
431             and DSP_ADDR20 = '1'
432             else "ZZZZZZZZ";

```

-- for UART or CPLD Select


```

433
434
435 -- #####
436 -- Generate the Daughter card buffer control signals
437 --
438 -- Point the data buffer direction to the DC if not reading.
439 -- Only enable the DC buffer if reading and daughter card is installed
440 -- Only enable control ouput signals if daughter card is installed
441 --
442
443 DC_DBUF_OEn <= '0' when ( DC_DETn = '0' and DSP_CS2n = '0')
444                      or ( DC_DETn = '0' and DSP_CS3n = '0')
445                      else '1';
446
447 DC_CNTL_OEn <= '0' when DC_DETn = '0' else '1';
448
449 DC_DBUF_DIR <= DSP_REn;
450
451 -- =====
452 -- SAMPLE RATE GENERATOR
453 --
454 -- Simple counter to generate the sample period
455
456 FXCLK <= CLKIN;
457 NiEnable <= '1' when ( ( ENABLE = '1' and SystemReseth = '1')
458                      or ( NiForceOn = '1' ) ) else '0';
459
460 process( NiEnable, FXCLK )
461 begin
462   if( NiEnable = '0' ) then
463     SampleCounter <= (others => '0');
464   elsif FXCLK'event and FXCLK = '1' then
465     SampleCounter <= SampleCounter + 1;
466   end if;
467 end process;
468
469 -- Create an enable signal based on the FXCLK
470 process( NiEnable, FXCLK, SampleCounter )
471 begin
472   if( NiEnable = '0' ) then
473     SampleEnable <= '0';
474   elsif FXCLK'event and FXCLK = '1' then
475     if ( SampleCounter = SAMPLE_HIGH ) then
476       SampleEnable <= '1';
477     elsif( SampleCounter = SAMPLE_LOW ) then
478       SampleEnable <= '0';
479     end if;
480   end if;
481 end process;
482
483 -- Move the SampleEnable from FXCLK domain to the VARCLK domain
484 process( NiEnable, VARCLK, SampleEnable)
485 begin
486   if( NiEnable = '0' ) then

```

```

487     VarSyncEnable <= ( others => '0' );
488     elsif VARCLK'event and VARCLK = '1' then
489         VarSyncEnable(0) <= SampleEnable;
490         VarSyncEnable(1) <= VarSyncEnable(0);
491         VarSyncEnable(2) <= VarSyncEnable(1);
492         VarSyncEnable(3) <= VarSyncEnable(2);
493     end if;
494 end process;
495 -- Enables the var counter
496 VarEnable <= VarSyncEnable(1);
497 -- Capture on falling edge of VarEnable. This requires that
498 -- SampleEnable is at least 6 VARCLKs high to detect both
499 -- falling and rising edges through the sync logic;
500 VarCapture <= '1' when VarSyncEnable = "1000" else '0';
501
502 -- Count VARCLKs
503 -- VarEnable
504 -- |-----|-----|-----|-----|
505 -- VarCapture
506 -- |-----|-----|-----|-----|
507 -- VarCounter
508 -- 000X count          X----00000000
509 -- VarLatch
510 -- -----XCount-----
511 --      Capture and Clear      ^
512
513 process( NiEnable, VARCLK, VarEnable )
514 begin
515     if( NiEnable = '0' ) then
516         VarCounter <= (others => '0');
517     elsif VARCLK'event and VARCLK = '1' then
518         -- Clear on VarCapture else count when enabled
519         if( VarCapture = '1' ) then
520             VarCounter <= ( others => '0');
521         elsif ( VarEnable = '1' ) then
522             VarCounter <= VarCounter + 1;
523         end if;
524     end if;
525 end process;
526
527 -- Capture at end of sample period
528 process( NiEnable, VARCLK, VarCapture, VarCounter )
529 begin
530     if( NiEnable = '0' ) then
531         VarLatch <= (others => '0');
532     elsif VARCLK'event and VARCLK = '1' then
533         if( VarCapture = '1' ) then
534             VarLatch(4 downto 0) <= VarCounter( 8 downto 4);
535         end if;
536     end if;
537 end process;
538
539 -- =====
540 -- CODE 4-0          : NI_DIG 4-0

```

```
541  -- SAMPLE_PERIOD    : NI_DIG 5
542  -- VControl0         : NI_DIG 6
543  -- VControl1         : NI_DIG 7
544  NiOe <= '1' when (    ( CAPTURE_POWER_ON = '1' and SystemResetrn = '1' and ENABLE = '1' )
545                      or  ( NiForceOn = '1' )) else '0';
546
547  NI_DIG(4 downto 0)    <= VarLatch    when NiOe = '1' else "zzzzz";
548  NI_DIG(5)            <= VarEnable   when NiOe = '1' else 'z';
549
550  -- Change for Rev 0010 (Production build) of CPLD to export 2 bit frequency code
551  --NI_DIG(6)           <= VarCapture  when NiOe = '1' else 'z';
552  --NI_DIG(7)           <= USB_XF     when NiOe = '1' else 'z';
553  NI_DIG(6)            <= VControl(0) when NiOe = '1' else 'z';
554  NI_DIG(7)            <= VControl(1) when NiOe = '1' else 'z';
555
556  EVENT_TRG           <= cp1dReg6(3)  when NiOe = '1' else 'z';
557
558
559  end behavior_cntlcp1d;
560
```