

```

1  -- $Id: largepld1.vhd,v 1.3 2004/12/08 22:52:28 tofp Exp $
2  -- notes:
3
4  -- 1. 9/10/04: c1_m24, c2_m24, c3_m24, c4_m24 signals are used as the
5  -- data strobe for 4bit tdlg tdc data. These are nominally the m24 data,
6  -- but m24 data is not implemented in RUN5.
7
8
9  -- Revisions:
10
11 -- 12/18/2004 Expanded control select muxes. Added state machine for
12 -- reading all 4 input cables on receipt of L0 trigger.
13
14 -- 11/17 Added DDL interface code
15
16 -- 11/3
17 -- added mux to select internal or external trigger
18 --
19 -- 10/23/2004:
20 -- added mcu interface
21 -- added test path the ctlvl state machine for trigger only acquisition
22
23
24
25 -- *****
26 -- LIBRARY DEFINITIONS
27 -- *****
28
29 LIBRARY ieee;
30 USE ieee.std_logic_1164.ALL;
31 LIBRARY lpm;
32 USE lpm.lpm_components.ALL;
33 LIBRARY altera_mf;
34 USE altera_mf.altera_mf_components.ALL;
35 USE work.tcpu_package.ALL;
36 USE work.my_conversions.ALL;
37 USE work.my_utilities.ALL;
38
39 -- *****
40 -- TOP LEVEL ENTITY
41 -- *****
42
43 ENTITY largepld1 IS
44 PORT
45 (
46   sloclck3 : IN std_logic;
47   pldclk0  : IN std_logic;
48
49   tst37    : OUT std_logic;
50   tst39    : OUT std_logic;
51   tsthi    : OUT std_logic_vector(35 DOWNTO 21);
52   tst19, tst17, tst13 : OUT std_logic;
53   tstio    : OUT std_logic_vector(11 DOWNTO 1);
54
55   MS_HI    : OUT std_logic; -- Pin used as HI in master/slave selection scheme
56   MS_LO    : OUT std_logic; -- Pin used as LO in master/slave selection scheme
57   MS_sel   : IN  std_logic; -- Pin used as master/slave input. will be externally
58   -- shorted to MS_HI *or* MS_LO.
59   MS_sel_out : OUT std_logic; -- Output to small PLD to determine clock source from M/S select
60
61
62   -- detector data link interface signals
63   ddl_fbd   : INOUT std_logic_vector(31 DOWNTO 0);
64   ddl_spare : OUT  std_logic;
65   ddl_fobsy_N : OUT  std_logic;
66   ddl_foclk : OUT  std_logic;
67   ddl_fidir : IN   std_logic;
68   ddl_fiben_N : IN  std_logic;
69   ddl_filf_N : IN  std_logic;
70   ddl_fbctr1_N : INOUT std_logic;
71   ddl_fbten_N : INOUT std_logic;
72
73   -- trigger/clock distribution board (TCD) inputs
74   tcd_cclk1 : IN  std_logic;
75   tcd_cclk2 : IN  std_logic;
76   tcd_50mhz : IN  std_logic;
77   tcd_10_mhz : IN  std_logic;
78   tcd_d      : IN  std_logic_vector(3 DOWNTO 0);
79   systrigin  : IN  std_logic;
80   systrig1, systrig2 : OUT std_logic;
81
82   -- microcontroller interface signals
83   mcuct1    : IN  std_logic_vector(3 DOWNTO 0); -- ctl3 = !read / write ; 2 downto 0 = adr
84   mcud      : INOUT std_logic_vector(7 DOWNTO 0); -- bidirectional data
85   pld_int   : IN  std_logic; -- data strobe
86   fifo_empty : OUT std_logic; -- mcu fifo empty
87
88   -- user interface signals
89   led1 : OUT std_logic; -- led 1 is iol18 (D2)
90   led2 : OUT std_logic; -- led 2 is iol19 (D3)
91   led3 : OUT std_logic; -- led 3 is iol20 (D6)
92
93   button : IN std_logic; -- button is iol17, LABEL SW2 ON PCB
94
95   -- rs232 interface signals
96   rs232sel : OUT std_logic_vector(2 DOWNTO 0);
97   rs232enb : OUT std_logic;
98
99   -- board status and configuration signals
100  enable_trayclks : OUT std_logic;
101
102  -- switches pages in config device
103  -- default grounded by jumpers JU23, JU24, JU25
104  pgm : IN std_logic_vector(2 DOWNTO 0);
105
106  -- outputs to TDIG cards
107  m7_gate : OUT std_logic;
108  tdctrig : OUT std_logic;
109  cmd1    : OUT std_logic;
110  reset_c : OUT std_logic_vector(4 DOWNTO 1);
111  spare_c : OUT std_logic_vector(4 DOWNTO 1);
112
113  -- inputs from TDIG cards
114  c1_m7d, c2_m7d : IN std_logic_vector(5 DOWNTO 0);
115  c3_m7d, c4_m7d : IN std_logic_vector(5 DOWNTO 0);
116  c1_tdc, c2_tdc : IN std_logic_vector(3 DOWNTO 0);
117  c3_tdc, c4_tdc : IN std_logic_vector(3 DOWNTO 0);
118  c1_m24, c2_m24 : IN std_logic;
119  c3_m24, c4_m24 : IN std_logic;
120  c1_dclk, c2_dclk : IN std_logic;
121  c3_dclk, c4_dclk : IN std_logic;
122
123  -- unused signals
124  -- pulled low externally with 0 ohm resistor
125  io5, io19, io36 : IN std_logic;
126  io37, io278, io279 : IN std_logic;
127  io295, io296 : IN std_logic;
128
129  -- uncommitted pin used for simulation only
130  --reset_function : IN std_logic
131
132 );
133
134 END largepld1;

```

```

135
136 -- *****
137 -- TOP LEVEL ARCHITECTURE
138 -- *****
139
140 ARCHITECTURE ver_four OF largepld1 IS
141
142 COMPONENT GLOBAL
143   PORT (a_in : IN std_logic;
144         a_out : OUT std_logic);
145 END COMPONENT;
146
147 -- *****
148 -- SIGNAL DECLARATIONS
149 -- *****
150
151 -- global signals
152
153 SIGNAL global40mhz : std_logic;
154 SIGNAL reset       : std_logic;
155
156 -- source for global reset
157 -- this signal is set currently to inactive
158 -- it can be sourced from pushbutton (debounced), mcu, etc.
159
160 SIGNAL reset_function : std_logic;
161
162
163 -- TCD interface signals
164
165 SIGNAL tcd_data   : std_logic_vector(19 DOWNTO 0);
166 SIGNAL tcd_strobe : std_logic;
167
168 -- input signals from TDIG after demux
169
170 SIGNAL tdig1_data, tdig2_data, tdig3_data, tdig4_data : std_logic_vector(31 DOWNTO 0);
171 SIGNAL tdig_strobe                                 : std_logic_vector(4 DOWNTO 1); -- signal from cable demux to input fifo
172
173 -- clock enable strobes for TDIG data after demux : used to clock data into input fifos
174
175 SIGNAL tdig1_clken, tdig2_clken, tdig3_clken, tdig4_clken : std_logic;
176
177 -- combined final data from ping/pong output muxes
178 -- this data will go to DDL interface (and to MCU for debug)
179
180 SIGNAL ping_pong_out : std_logic_vector(31 DOWNTO 0);
181
182 -- control signals for core data path
183
184 SIGNAL ping_pong_read_enable, ping_pong_empty : std_logic;
185
186 -- test signals
187
188 SIGNAL button_debounced : std_logic; -- debounced 1 clock wide, active high pulse from button push
189                                     -- button is labelled
190
191 SIGNAL opmode, error_word : std_logic_vector(7 DOWNTO 0);
192 SIGNAL ddl_read_fifo     : std_logic;
193
194 SIGNAL test_pattern_1234 : std_logic_vector(15 DOWNTO 0);
195 SIGNAL test_pattern_aaaa : std_logic_vector(15 DOWNTO 0);
196 SIGNAL test_pattern_5555 : std_logic_vector(15 DOWNTO 0);
197 SIGNAL test_pattern_0000 : std_logic_vector(15 DOWNTO 0);
198 SIGNAL test_pattern_ffff : std_logic_vector(15 DOWNTO 0);
199 SIGNAL test_pattern_ffffff : std_logic_vector(31 DOWNTO 0);
200 SIGNAL test_pattern_00000000 : std_logic_vector(31 DOWNTO 0);
201 SIGNAL test_pattern_aaaaaaaaa : std_logic_vector(31 DOWNTO 0);
202 SIGNAL test_pattern_55555555 : std_logic_vector(31 DOWNTO 0);
203
204 SIGNAL bus_enable, tristate_signal_enable, trigger_strobe_to_tdig, star_trigger_level_zero : std_logic;
205
206 SIGNAL data_strobe, readbar_write : std_logic;
207 SIGNAL mcu_adr                    : std_logic_vector(2 DOWNTO 0);
208 SIGNAL mcu_data, pld_to_mcu_before_buffer, mcu_to_pld_after_buffer, mcu_decode : std_logic_vector(7 DOWNTO 0);
209 SIGNAL mcu_mode_data, mcu_config_data, mcu_filter_sel : std_logic_vector(7 DOWNTO 0);
210 SIGNAL mcu_write_to_pld, mcu_read_from_pld, mode_reg_write, config_reg_write : std_logic;
211 SIGNAL mcu_fifo_empty, mcu_filter_reg_write, mcu_bunch_reset, mcu_reset_to_pld : std_logic;
212 SIGNAL mcu_fifo_out : std_logic_vector(31 DOWNTO 0);
213 SIGNAL dummy_counter_out : std_logic_vector(14 DOWNTO 0);
214 SIGNAL trig_from_ctr : std_logic;
215 SIGNAL mcu_strobes_fifo : std_logic;
216 SIGNAL cout, delaya, delayb, stretch, trig_presync : std_logic;
217
218 -- INPUT FIFOs
219 SIGNAL infifo0_dout : std_logic_vector (19 DOWNTO 0); -- from tcd fifo
220 SIGNAL infifo1_dout : std_logic_vector (31 DOWNTO 0); -- from tdig 1 fifo
221 SIGNAL infifo2_dout : std_logic_vector (31 DOWNTO 0); -- from tdig 2 fifo
222 SIGNAL infifo3_dout : std_logic_vector (31 DOWNTO 0); -- from tdig 3 fifo
223 SIGNAL infifo4_dout : std_logic_vector (31 DOWNTO 0); -- from tcd 4 fifo
224
225 SIGNAL read_input_fifo, infifo_full, infifo_empty : std_logic_vector(4 DOWNTO 0);
226
227 SIGNAL inmux_dout : std_logic_vector (31 DOWNTO 0);
228 SIGNAL inmux_sel : std_logic_vector (2 DOWNTO 0);
229
230 -- PING_PONG OUTPUT FIFOs
231
232 SIGNAL outfifo0_dout : std_logic_vector (31 DOWNTO 0);
233 SIGNAL outfifo1_dout : std_logic_vector (31 DOWNTO 0);
234 SIGNAL outfifo_in_enable, outfifo_out_enable : std_logic_vector (1 DOWNTO 0);
235 SIGNAL outfifo_full, outfifo_empty, write_ddl_fifo : std_logic_vector (1 DOWNTO 0);
236
237 SIGNAL outmux_write : std_logic; -- control for 2:1 output DDL mux
238 SIGNAL outmux_sel, toggle : std_logic;
239
240 -- MCU interface
241
242 SIGNAL wr_mcu_fifo, rd_mcu_fifo, mcu_fifo_full : std_logic;
243 SIGNAL mcu_data_sel : std_logic_vector(1 DOWNTO 0);
244 SIGNAL mcu_fifoq : std_logic_vector(31 DOWNTO 0);
245
246 -- test fifo signals -- test fifo output goes to DDL in test mode
247 SIGNAL wr_final_fifo, rd_test_fifo, test_fifo_full, test_fifo_empty : std_logic;
248 SIGNAL wr_ddl_fifo, rd_ddl_fifo, ddlfifo_full, ddlfifo_empty : std_logic;
249 SIGNAL ping_pong_data, ddl_data, ddl_fifo_indata : std_logic_vector(31 DOWNTO 0);
250
251 -- signals to control which state machine controls the main data mux
252 SIGNAL data_mux_sel, mcu_sel, main_data_sel, ctl_one_sel_input : std_logic_vector(2 DOWNTO 0);
253 SIGNAL mcu_sel_empty : std_logic;
254
255 SIGNAL dummy : std_logic;
256 SIGNAL error1 : std_logic_vector(7 DOWNTO 0);
257 SIGNAL ctl_one_read_fe_fifo, input_fifo_empty, ctl_one_wr_mcu_fifo : std_logic;
258 SIGNAL ctl0_read_fe_fifo, ctl0_wr_mcu_fifo, write_mcu_fifo : std_logic;
259 SIGNAL ctr, read_fifo_enable, ctl_one_write_mcu_fifo : std_logic;
260 SIGNAL end_record_tc, incr_end_of_record_cnt : std_logic;
261 SIGNAL dummy7b : std_logic_vector(6 DOWNTO 0);
262
263 -- signal which selects which state machine is in control (set by MCU config bit)
264 SIGNAL control_select, incr_sel, clr_sel, sel_eq_0 : std_logic;
265 SIGNAL ctr_timeout, timeout_valid : std_logic;
266 SIGNAL ctr_sel : std_logic_vector(2 DOWNTO 0);
267
268 SIGNAL ctl_one_trigger_to_tdc : std_logic;

```

```

269
270 -- signals decoded from trigger command bits
271 SIGNAL CMD_L0, CMD_L2, CMD_ABORT, CMD_RESET, CMD_IGNORE : std_logic;
272 SIGNAL separator : std_logic;
273
274 -- signals to control input to DDL fifo
275 SIGNAL stuff_sel_dout : std_logic_vector(31 DOWNTO 0);
276 SIGNAL ddl_in_sel, ctl_one_stuff : std_logic;
277
278 -- *****
279 -- DDL bidir signals separated into IN and OUT (JS)
280 -- *****
281 SIGNAL s_fid : std_logic_vector (31 DOWNTO 0); -- corresponds to ddl_fbd (IN)
282 SIGNAL s_foD : std_logic_vector (31 DOWNTO 0); -- corresponds to ddl_fbd (OUT)
283 SIGNAL s_fiTEN_N : std_logic; -- corresponds to ddl_fbten_N (IN)
284 SIGNAL s_foTEN_N : std_logic; -- corresponds to ddl_fbten_N (OUT)
285 SIGNAL s_fiCTRL_N : std_logic; -- corresponds to ddl_fbctrl_N (IN)
286 SIGNAL s_foCTRL_N : std_logic; -- corresponds to ddl_fbctrl_N (OUT)
287
288 -- new signals
289
290 SIGNAL mode_0_reset, mode_1_reset : std_logic; -- state machine reset signals decoded from mode bit(s)
291
292
293 constant separator_id : std_logic_vector (3 downto 0) := "1110";
294
295
296 -- *****
297 -- ARCHITECTURE BEGINS HERE
298 -- *****
299
300 BEGIN
301
302 -- *****
303 -- MASTER / SLAVE assignment:
304
305 MS_LO <= '0';
306 MS_HI <= '1';
307 MS_sel_out <= MS_sel;
308
309
310 -- *****
311 --
312 -- TEST SIGNALS TO TEST HEADER
313
314 -- MAPPING TDC DATA NIBBLES FROM TDC1 TO TEST HEADER
315
316 -- tstlo(1) <= inmux_dout(0);
317 -- tstlo(3) <= inmux_dout(1);
318 -- tstlo(5) <= inmux_dout(2);
319 -- tstlo(7) <= inmux_dout(3);
320
321 -- tstlo(9) <= c2_m7d(0);
322 -- tstlo(11) <= c2_m7d(1);
323 -- tstl3 <= c2_m7d(2);
324 -- tstl7 <= c2_m7d(3);
325 -- tstl9 <= tdig_strobe(2);
326 -- tsthi(21) <= read_input_fifo(2);
327 -- tsthi(23) <= stretch;
328 -- tsthi(25) <= main_data_sel(0);
329 -- tsthi(27) <= main_data_sel(1);
330 -- tsthi(29) <= main_data_sel(2);
331 -- tsthi(31) <= input_fifo_empty;
332
333 tstlo(1) <= tcd_g(0);
334 tstlo(3) <= tcd_g(1);
335 tstlo(5) <= tcd_g(2);
336 tstlo(7) <= tcd_d(3);
337 tstlo(9) <= tcd_50mhz;
338 tstlo(11) <= tcd_10_mhz;
339 tstl3 <= tcd_cclk1;
340 tstl7 <= tcd_cclk2;
341 tstl9 <= tcd_data(0);
342 tsthi(21) <= tcd_data(1);
343 tsthi(23) <= tcd_data(2);
344 tsthi(25) <= tcd_data(3);
345 tsthi(27) <= tcd_data(4);
346 tsthi(29) <= tcd_data(5);
347 tsthi(31) <= tcd_data(6);
348
349
350 -- TDIG MCU MASTER RESET CONTROL! -----
351
352 reset_c <= "1111";
353 -- For now, just hold these outputs high. IN the future, TCPU will reset TDIG on startup
354 -- and also under CAN control.
355
356 -- TRIGGER INTERFACE -----
357
358 -- INTERNAL TRIGGER COUNTER
359
360 trig_from_ctr_counter : trigger_counter_15bit PORT MAP (
361 clock => global40mhz,
362 cnt_en => '1',
363 aclr => '0',
364 q => dummy_counter_out,
365 cout => cout );
366
367 -- ROUTING TRIGGER INPUT TO tdc trigger input over ribbon cable
368
369 -- use ext trigger if config bit not set
370 -- or internal trigger if config bit set
371
372 turn_off_trigger : mux_2to1_lbit PORT MAP (
373 data0 => '0',
374 data1 => cout,
375 sel => mcu_config_data(0),
376 result => trig_from_ctr );
377
378 trigger_mux : mux_2to1_lbit PORT MAP (
379 data0 => trig_from_ctr,
380 data1 => ctl_one_trigger_to_tdc,
381 sel => control_select,
382 result => trig_presync );
383
384 -- stretch final trigger pulse and send to TDCs over ribbon cable
385
386 DFF_a : DFF_sclr PORT MAP (
387 clock => global40mhz,
388 sclr => '0',
389 aclr => '0',
390 data => trig_presync,
391 q => delaya );
392
393 DFF_b : DFF_sclr PORT MAP (
394 clock => global40mhz,
395 sclr => '0',
396 aclr => '0',
397 data => delaya,
398 q => delayb );
399
400 stretch <= delaya OR delayb;
401
402 tdcdrig <= stretch; -- after stretching to 50ns width

```

```

403
404 -- TO TEST USING AN EXTERNAL PULSE GENERATOR, USE THE FOLLOWING LINE:
405
406 -- tdcTrig <= systTrig; -- external pulse generator at J39 (remember PECL input levels)
407
408 -- Turn on TDIG clocks:
409
410 enable_trayclks <= '1';
411
412 -----
413
414
415 ddl_read_fifo <= '1';
416
417 opmode <= "00000001";
418
419 test_pattern_ffffffff <= X"ffffffff";
420
421 -- make clock and reset global
422
423 global_clk_buffer : global PORT MAP (a_in => pldclk0, a_out => global40mhz);
424 clk <= global40mhz;
425
426 global_reset_buffer : global PORT MAP (a_in => '0', a_out => reset); -- input for this will be changed to
427 -- reset function
428
429 -- reset_function <= '0'; -- global reset source set to inactive reset
430 -- for simulation purposes this signal is given above in the entity declaration
431 -- as a device pin
432
433 -- reset_function <= '0';
434
435 -- led status / version indicators
436
437 led1 <= MS_sel; -- labelled 'D2' on pcb
438 led2 <= NOT MS_sel; -- labelled 'D3' on pcb
439
440 -- CONTROL LED3 WITH BUTTON 'SW2'
441 led3 <= button; -- labelled 'D6' on pcb
442
443 -- pushbutton input
444
445 button_sync : pushbutton_pulse PORT MAP (
446 clk => global40mhz,
447 reset => reset,
448 pushbutton => button,
449 pulseout => button_debounced );
450
451
452 -- *****
453 -- ddl interface defaults (JS)
454 -- *****
455
456 -- detector data link interface signals
457 ddl_spare <= button;
458 ddl_foclk <= global40mhz;
459 s_fiTEN_N <= ddl_fbten_N;
460 s_fiCTRL_N <= ddl_fbctrl_N;
461 s_fid <= ddl_fbd;
462
463 ddibus : PROCESS (ddl_fiben_N, ddl_fidir, s_foTEN_N, s_foCTRL_N, ddl_fbd, s_fod)
464 BEGIN
465 IF (ddl_fiben_N = '1') OR (ddl_fidir = '0') THEN
466 ddl_fbten_N <= 'Z';
467 ddl_fbctrl_N <= 'Z';
468 ddl_fbd <= (OTHERS => 'Z');
469 ELSE
470 ddl_fbten_N <= s_foTEN_N;
471 ddl_fbctrl_N <= s_foCTRL_N;
472 ddl_fbd <= s_fod;
473 END IF;
474 END PROCESS;
475
476 ddl_inst : ddl PORT MAP ( -- DDL
477 fid => s_fid,
478 fod => s_fod,
479 fobSY_N => ddl_fobsy_N,
480 fiCLK => global40mhz,
481 fiDIR => ddl_fidir,
482 fiBEN_N => ddl_fiben_N,
483 fiFEN_N => ddl_fifen_N,
484 fiCTRL_N => s_fiCTRL_N,
485 foCTRL_N => s_foCTRL_N,
486 fiTEN_N => s_fiTEN_N,
487 foTEN_N => s_foTEN_N,
488 ext_trg => button_debounced, -- external trigger
489 reset => reset,
490 fifo_q => ddl_data, -- lwb: 11/17/04 I'm attaching the ddl fifo output to here
491 fifo_empty => ddlfifo_empty, -- lwb: 11/17/04 I'm attaching the ddl fifo empty to here
492 fifo_rdrreq => rd_ddl_fifo
493 );
494
495 -- INPUT SECTION *****
496 --
497 -- TCPU receives input data from 5 sources: 4 TDIG tray cables and 1 TCD cable
498 -- This section demultiplexes each of these data streams and writes the data to
499 -- a fifo. A 5 input mux selects which of the sources will feed the data path to
500 -- the DDL and MCU fifos. The mux control and the fifo controls come from the
501 -- currently selected control state machine.
502
503 tdig_input_1 : COMPONENT ser_4bit_to_par PORT MAP (
504 clk => global40mhz, reset => reset,
505 din => c1_m7d(3 DOWNTO 0),
506 dclk => c1_dclk,
507 dstrobe => c1_m7d(4), -- data strobe is active for 8 dclks
508 dout => tdig1_data,
509 output_strobe => tdig_strobe(1) );
510
511 tdc1_fifo : COMPONENT input_fifo_64x32 PORT MAP (
512 clock => clk,
513 aclr => reset,
514 data => tdig1_data,
515 wrreq => tdig_strobe(1),
516 rdreq => read_input_fifo(1),
517 q => infifo1_dout,
518 full => infifo_full(1),
519 empty => infifo_empty(1) );
520
521 tdig_input_2 : COMPONENT ser_4bit_to_par PORT MAP (
522 clk => global40mhz, reset => reset,
523 din => c2_m7d(3 DOWNTO 0),
524 dclk => c2_dclk,
525 dstrobe => c2_m7d(4), -- data strobe is active for 8 dclks
526 dout => tdig2_data,
527 output_strobe => tdig_strobe(2) );
528
529 tdc2_fifo : COMPONENT input_fifo_64x32 PORT MAP (
530 clock => clk,
531 aclr => reset,
532 data => tdig2_data,
533 wrreq => tdig_strobe(2),
534 rdreq => read_input_fifo(2),
535 q => infifo2_dout,
536 full => infifo_full(2),

```

```

537 empty => infifo_empty(2) );
538
539 tdig_input_3 : COMPONENT ser_4bit_to_par PORT MAP (
540   clk      => global40mhz, reset => reset,
541   din      => c3_m7d(3 DOWNTO 0),
542   dclk     => c3_dclk,
543   dstrobe  => c3_m7d(4),      -- data strobe is active for 8 dclks
544   dout     => tdig3_data,
545   output_strobe => tdig3_strobe(3) );
546
547 tdc3_fifo : COMPONENT input_fifo_64x32 PORT MAP (
548   clock => clk,
549   aclr  => reset,
550   data  => tdig3_data,
551   wrreq => tdig3_strobe(3),
552   rdreq => read_input_fifo(3),
553   q     => infifo3_dout,
554   full  => infifo_full(3),
555   empty => infifo_empty(3) );
556
557 tdig_input_4 : COMPONENT ser_4bit_to_par PORT MAP (
558   clk      => global40mhz, reset => reset,
559   din      => c4_m7d(3 DOWNTO 0),
560   dclk     => c4_dclk,
561   dstrobe  => c4_m7d(4),      -- data strobe is active for 8 dclks
562   dout     => tdig4_data,
563   output_strobe => tdig4_strobe(4) );
564
565 tdc4_fifo : COMPONENT input_fifo_64x32 PORT MAP (
566   clock => clk,
567   aclr  => reset,
568   data  => tdig4_data,
569   wrreq => tdig4_strobe(4),
570   rdreq => read_input_fifo(4),
571   q     => infifo4_dout,
572   full  => infifo_full(4),
573   empty => infifo_empty(4) );
574
575 tcd_input : COMPONENT trigger_interface PORT MAP (
576   clk      => global40mhz,
577   reset    => reset,
578   tcd_4bit_data => tcd_d,
579   tcd_clk_50mhz => tcd_50mhz,
580   tcd_clk_10mhz => tcd_10_mhz,
581   tcd_word     => tcd_data,
582   tcd_strobe  => tcd_strobe ); -- data valid strobe to clock data to fifo
583
584 tcd_input_fifo : COMPONENT input_fifo64dx20w PORT MAP ( -- tcd data
585   clock => clk,
586   aclr  => reset,
587   data  => tcd_data,
588   wrreq => tcd_strobe,
589   rdreq => read_input_fifo(0),
590   q     => infifo0_dout,
591   full  => infifo_full(0),
592   empty => infifo_empty(0) );
593
594 infifo_mux : COMPONENT mux_5x32_unreg PORT MAP (
595   data0x(31 DOWNTO 20) => "101000000000", --"000000000000",
596   data0x(19 DOWNTO 0)  => infifo0_dout,
597   data1x                => infifo1_dout,
598   data2x                => infifo2_dout,
599   data3x                => infifo3_dout,
600   data4x                => infifo4_dout,
601   sel                   => main_data_sel,
602   result                => inmux_dout );
603
604 -- CONTROL SECTION *****
605
606 control_select <= mcu_mode_data(0); -- selects between CTL0 and ctl_one as main controller
607
608 -- Currently 2 possible controllers:
609
610 -- Active controller is selected by mcu_mode_data(0).
611 -- Then multiplexers select signals to/from the active controller to go to/from data path
612
613 -- CTL0: This controller always reads a single front-end fifo that is selected by the mcu.
614
615 mode_0_reset <= control_select; -- "control_select" = mcu_config[0]
616
617 CTL0 : COMPONENT alwread PORT MAP (
618   CLK      => clk,
619   RESET    => mode_0_reset,
620   empty    => input_fifo_empty,
621   rd_fifo  => ctl0_read_fe_fifo,
622   wr_fifo  => ctl0_wr_mcu_fifo,
623   incr_cnt => incr_end_of_record_cnt); -- increment end of record counter
624 -- ovf from counter selects *E700 0000* word
625
626 -- control_one: This controller looks for L0 commands and then builds an event record from all
627 -- 4 TDIG data streams.
628
629
630 -- This signal detects that select ctr has rolled over to initial state
631 sel_eq_0 <= (not ctr_sel(2)) and (not ctr_sel(1)) and (not ctr_sel(0));
632
633 mode_1_reset <= not control_select; -- "control_select" = mcu_config[0]
634
635 Control_one : COMPONENT CTL_ONE PORT MAP (
636   clk      => clk,
637   reset    => mode_1_reset,
638   cmd_l0   => CMD_L0,
639   fifo_empty => input_fifo_empty,
640
641   sel_eq_0 => sel_eq_0,
642   separator => separator,
643   timeout   => timeout_valid,
644   clr_sel   => clr_sel,
645   clr_timeout => clr_timeout,
646   incr_sel  => incr_sel,
647   rd_fifo  => ctl_one_read_fe_fifo,
648   trig_to_tdc => ctl_one_trigger_to_tdc,
649   wr_fifo  => ctl_one_wr_mcu_fifo,
650   ctl_one_stuff => ctl_one_stuff );
651
652 -- controllers increment this counter to cycle through the input fifos
653 -- counter counts from 0 to 4
654 main_mux_select_counter : component data_sel_ctr PORT MAP (
655   clock => clk,
656   cnt_en => incr_sel,
657   sclr  => clr_sel,
658   aclr  => reset,
659   q     => ctr_sel ); -- 3 bits
660
661
662 -- controllers use this timeout counter to switch between input fifos if
663 -- there is no separator from a TDIG link within the timeout period
664 timeout_counter : component timeout PORT MAP (
665   clk      => clk,
666   reset    => reset,
667   clr_timeout => clr_timeout,
668   timeout_valid => timeout_valid );
669
670 -- decoder for trigger commands and separator word

```

```

671     command_decode: process (inmux_dout) is
672     begin
673         if inmux_dout(31 downto 28) = separator_id then
674             separator <= '1';
675         else separator <= '0';
676         end if;
677
678         CMD_L0 <= '0';
679         CMD_L2 <= '0';
680         CMD_ABORT <= '0';
681         CMD_RESET <= '0';
682         CMD_IGNORE <= '0';
683
684         case inmux_dout(19 downto 16) is -- trigger_command
685             when "0000" => CMD_IGNORE <= '1';
686             when "0001" => CMD_IGNORE <= '1';
687             when "0010" => CMD_RESET <= '1';
688             when "0011" => CMD_IGNORE <= '1';
689             when "0100" => CMD_L0 <= '1';
690             when "0101" => CMD_L0 <= '1';
691             when "0110" => CMD_L0 <= '1';
692             when "0111" => CMD_L0 <= '1';
693             when "1000" => CMD_L0 <= '1';
694             when "1001" => CMD_L0 <= '1';
695             when "1010" => CMD_L0 <= '1';
696             when "1011" => CMD_L0 <= '1';
697             when "1100" => CMD_L0 <= '1';
698             when "1101" => CMD_ABORT <= '1';
699             when "1110" => CMD_IGNORE <= '1';
700             when "1111" => CMD_L2 <= '1';
701         end case;
702     end process command_decode;
703
704 -- CHOOSE INPUT DATA SOURCE
705
706 mcu_sel <= mcu_config_data(3 DOWNTO 1);
707 ctl_one_sel_input <= "000"; -- stub for control from large state machine
708
709 choose_source_for_input_mux_select_bits : two_by_3bit_mux PORT MAP (
710     data0x => mcu_sel, -- mcu config register selects data at mux
711     data1x => ctr_sel, -- mux select counter selects data at mux (controller clocks counter)
712     sel => control_select,
713     result => main_data_sel );
714
715 -- SELECT FIFO EMPTY SIGNAL FROM CURRENT INPUT FIFO
716
717 empty_signal_mux : COMPONENT mux_5_to_1 PORT MAP (
718     data4 => infifo_empty(4),
719     data3 => infifo_empty(3),
720     data2 => infifo_empty(2),
721     data1 => infifo_empty(1),
722     data0 => infifo_empty(0),
723     sel => main_data_sel,
724     result => input_fifo_empty );
725
726 -- ROUTE INPUT FIFO READ SIGNAL TO CURRENT INPUT FIFO
727
728 choose_source_for_input_read : mux_2to1_lbit PORT MAP (
729     data0 => ctl0_read_fe_fifo,
730     data1 => ctl_one_read_fe_fifo,
731     sel => control_select,
732     result => read_fifo_enable );
733
734 fifo_read_decoder : decode_3to5_en PORT MAP (
735     data => main_data_sel,
736     enable => read_fifo_enable,
737     eq0 => read_input_fifo(0),
738     eq1 => read_input_fifo(1),
739     eq2 => read_input_fifo(2),
740     eq3 => read_input_fifo(3),
741     eq4 => read_input_fifo(4) );
742
743 -- SELECT SOURCE FOR WRITING TO MCU FIFO
744
745 choose_source_for_write_to_mcu_fifo : mux_2to1_lbit PORT MAP (
746     data0 => ctl0_wr_mcu_fifo,
747     data1 => ctl_one_wr_mcu_fifo,
748     sel => control_select,
749     result => write_mcu_fifo );
750
751 -- shorten read fifo pulse from MCU
752
753 shorten_mcu_rdfifo : COMPONENT SHORTEN PORT MAP (
754     CLK => clk,
755     RESET => reset,
756     read_strobe => mcu_strobes_fifo,
757     rd_fifo => rd_mcu_fifo);
758
759 -- OUTPUT TO MCU FIFO *****
760
761 -- MCU FIFO:
762
763 mcu_outfifo : COMPONENT output_fifo_1024x32 PORT MAP (
764     data => inmux_dout,
765     wrreq => write_mcu_fifo,
766     rdreq => rd_mcu_fifo,
767     clock => clk,
768     aclr => reset,
769     q => mcu_fifo_out,
770     full => mcu_fifo_full,
771     empty => mcu_fifo_empty );
772
773 -- OUTPUT TO DDL FIFOS *****
774
775 -- State machine CTL0 reads input fifo and writes to both mcu and ddl fifos. Each write
776 -- increments end_of_record counter. When this counter reaches terminal count, then "end of record"
777 -- input is selected as input to DDL FIFO.
778
779 end_of_record_counter : count127 PORT MAP ( -- used with ctl0 to select "end of record" value
780     -- after set number of noise values
781     clock => clk,
782     cnt_en => incr_end_of_record_cnt,
783     aclr => reset,
784     q => dummy7b, -- q output not used
785     cout => end_record_tc);
786
787 ddl_stuff_ctl : mux_2to1_lbit PORT MAP ( -- selects which state machine controls the mux
788     -- tostuff non data values into DDL
789     data1 => ctl_one_stuff,
790     data0 => end_record_tc,
791     sel => control_select,
792     result => ddl_in_sel );
793
794
795 ddl_stuff_sel : mux_2x32 PORT MAP ( -- selects between data path value and stuff value
796     -- according to select input from ddl_stuff_ctl mux
797     data1x => X"EA000000",
798     data0x => X"EA000000",
799     sel => control_select,
800     result => stuff_sel_dout );
801
802 ddl_input_select : mux_2x32 PORT MAP ( -- selects between stuff values according to which
803     -- controller is active: ctl0 gives simple EA stuff value
804     -- other controllers can provide different values

```

```

805 data1x => stuff_sel_dout,    -- from stuff data mux
806 data0x => inmux_dout,      -- from main 5 way data path mux
807 sel   => ddl_in_sel,
808 result => ddl_fifo_indata );
809
810 -- final DDL fifo
811
812 final_ddl_fifo : COMPONENT output_fifo_1024x32 PORT MAP (
813   clock => clk,
814   aclr  => reset,
815   data  => ddl_fifo_indata,
816   wrreq => write_mcu_fifo,
817   rdreq => rd_ddl_fifo,
818   q     => ddl_data,
819   full  => ddlfifo_full,
820   empty => ddlfifo_empty );
821
822
823 -- *****
824 -- MCU interface
825 -- *****
826
827 mcu_adr(0) <= mcuctl(0);
828 mcu_adr(1) <= mcuctl(1);
829 mcu_adr(2) <= mcuctl(2);
830 readbar_write <= mcuctl(3);
831
832 data_strobe <= pld_int;
833 mcu_data <= mcud;
834 fifo_empty <= mcu_fifo_empty;
835
836 mcu_write_to_pld <= data_strobe AND readbar_write;
837 mcu_read_from_pld <= data_strobe AND (NOT readbar_write);
838
839 mcu_bus : bus_tri_8 PORT MAP (
840   data  => pld_to_mcu_before_buffer, -- output data from pld logic to tristate bus
841   enabled => mcu_read_from_pld,      -- sets bidirectional pins to OUT
842   enabletr => mcu_write_to_pld,      -- sets bidirectional pins to IN
843   tridata => mcu_data,               -- bidirectional to/from pins
844   result => mcu_to_pld_after_buffer ); -- input data from tristate bus to pld logic
845
846 mcu_write_decoder : decoder_3_to_8 PORT MAP (
847   data => mcu_data,
848   eq0 => mcu_decode(0), -- write to mode register
849   eq1 => mcu_decode(1), -- write to config register
850   eq2 => mcu_decode(2), -- write to filter register
851   eq3 => mcu_decode(3),
852   eq4 => mcu_decode(4),
853   eq5 => mcu_decode(5),
854   eq6 => mcu_decode(6), -- reset pld
855   eq7 => mcu_decode(7) ); -- send bunch reset
856
857 -- mcu writes to registers when strobe and adr produce valid register clk enables
858 -- registers that are written to by MCU -----
859
860 mode_reg_write <= mcu_decode(0) AND mcu_write_to_pld;
861
862 mode_register : reg8_en PORT MAP (
863   clock => global40mhz,
864   enable => mode_reg_write,
865   sclr  => reset,
866   data  => mcu_to_pld_after_buffer,
867   q     => mcu_mode_data );
868
869 -- config register:
870 --   address = 1
871 --   data(0) : 1 = ext trigger, 0 = internal trigger
872 --   controls trigger mux for trigger to TDIG boards
873
874 config_reg_write <= mcu_decode(1) AND mcu_write_to_pld;
875
876 config_register : reg8_en PORT MAP (
877   clock => global40mhz,
878   enable => config_reg_write,
879   sclr  => reset,
880   data  => mcu_to_pld_after_buffer,
881   q     => mcu_config_data );
882
883 mcu_filter_reg_write <= mcu_decode(2) AND mcu_write_to_pld;
884
885 mcu_filter_register : reg8_en PORT MAP (
886   clock => global40mhz,
887   enable => mcu_filter_reg_write,
888   sclr  => reset,
889   data  => mcu_to_pld_after_buffer,
890   q     => mcu_filter_sel );
891
892 -- writing to adr = 6 resets the pld state machines, etc.
893 mcu_reset_to_pld <= mcu_decode(6) AND mcu_write_to_pld;
894
895 -- writing to adr = 7 sends bunch reset
896 -- this signal is distributed to all TDCs from the master TCPU
897 -- this signal must be gated with a configuration bit to differentiate
898 -- between a master tcpu, which sends this signal, and a slave which receives it
899 mcu_bunch_reset <= mcu_decode(7) AND mcu_write_to_pld;
900
901 -- This pulse is synchronized because it comes from the MCU.
902 -- it is 200ns wide, which is believed to be ok.
903 -- HPTDC will exit reset when the pulse goes low, regardless of
904 -- the number of cycles it was held hi.
905
906 cmd1 <= (MS_sel AND mcu_bunch_reset) OR (NOT MS_sel AND systrigin);
907
908 systrig1 <= (MS_sel AND mcu_bunch_reset);
909 systrig2 <= (MS_sel AND mcu_bunch_reset);
910
911
912 -- MCU READS FROM PLD
913
914 -- data mux that is read by MCU -----
915
916 mcu_read_mux : mux_8_by_8bit PORT MAP (
917   data7x => X'00',
918   data6x => X'00', -- generate mcu fifo read pulse see 'mcu_strobes_fifo' signal
919   data5x => mcu_fifo_out(7 DOWNTO 0), -- read low byte
920   data4x => mcu_fifo_out(15 DOWNTO 8), -- read 2nd byte
921   data3x => mcu_fifo_out(23 DOWNTO 16), -- read 3rd byte
922   data2x => mcu_fifo_out(31 DOWNTO 24), -- read high fifo byte
923   data1x => mcu_config_data, -- readback config register
924   data0x => mcu_mode_data, -- readback mode register
925   sel   => mcu_adr,
926   result => pld_to_mcu_before_buffer );
927
928 -- clock fifo strobe -- needs to be shortened to one clock wide
929 -- Each time the mcu reads a 32b tdc word, it performs 4 data reads (1 byte each) and then
930 -- a read to adr = 6, which generates a strobe that gives a read_enable pulse to the mcu fifo
931
932 mcu_strobes_fifo <= mcu_read_from_pld AND mcu_adr(2) AND mcu_adr(1) AND (NOT mcu_adr(0));
933
934 END ARCHITECTURE ver_four;
935

```