```
1   ------------------------------------------------------------------------
    ----
2   --  Filename:   gh_counter_control.vhd
3   --
4   --  Description:
5   --      counter control for the 8254 Timer
6   --
7   --  Copyright (c) 2008, 2011 by H LeFevre
8   --      A VHDL 8254 Timer core
9   --      an OpenCores.org Project
10  --      free to use, but see documentation for conditions
11  --
12  --  Revision    History:
13  --  Revision    Date        Author      Comment
14  --  --------    ----------  ---------   -----------
15  --  1.0         08/02/08    H LeFevre   Initial revision
16  --  1.1         05/13/11    H LeFevre   change k input for U3 JKFF (load
    is sync with correct clk)
17  --
18  ------------------------------------------------------------------------
    ----
19  library ieee ;
20  use ieee.std_logic_1164.all ;
21
22  entity gh_counter_control is
23      GENERIC(same_clk: boolean := false; -- true, if same clock is used
    for bus and counter
24              sync_clk: boolean := false); -- true, if bus and counter
    clocks are synchronous
25      port(
26      -------- data bus/control signals ------------
27          clk_i  : in std_logic;
28          rst    : in std_logic;
29          ics    : in std_logic;
30          dwr    : in std_logic;
31          iA     : in std_logic_vector(1 downto 0);
32          iD     : in std_logic_vector(7 downto 0);
33          dat_o  : out std_logic_vector(7 downto 0);
34      ------------------------------------------------
35      ------ counter output/control signals --------------
36          clk     : in std_logic;
37          rd_busy : out std_logic;
38          gate    : in std_logic;
39          Cout    : out std_logic
40          );
41  end entity;
42
43  architecture a of gh_counter_control is
44
45  COMPONENT gh_edge_det is
46      PORT(
47          clk : in STD_LOGIC;
48          rst : in STD_LOGIC;
49          D   : in STD_LOGIC;
50          re  : out STD_LOGIC; -- rising edge (need sync source at D)
51          fe  : out STD_LOGIC; -- falling edge (need sync source at D)
52          sre : out STD_LOGIC; -- sync'd rising edge
```

```
53          sfe : out STD_LOGIC  -- sync'd falling edge
54          );
55  END COMPONENT;
56
57  COMPONENT gh_register_ce is
58      GENERIC (size: INTEGER := 8);
59      PORT(
60          clk : IN        STD_LOGIC;
61          rst : IN        STD_LOGIC;
62          CE  : IN        STD_LOGIC; -- clock enable
63          D   : IN        STD_LOGIC_VECTOR(size-1 DOWNTO 0);
64          Q   : OUT       STD_LOGIC_VECTOR(size-1 DOWNTO 0)
65          );
66  END COMPONENT;
67
68  COMPONENT gh_jkff is
69      PORT(
70          clk  : IN STD_logic;
71          rst  : IN STD_logic;
72          J,K  : IN STD_logic;
73          Q    : OUT STD_LOGIC
74          );
75  END COMPONENT;
76
77  COMPONENT gh_edge_det_XCD_t is
78      GENERIC(same_clk: boolean := false; -- true, if same clock is used
    for i/o clocks
79              sync_clk: boolean := false); -- true, if i/o clocks are
    synchronous
80      port(
81          iclk : in STD_LOGIC;  -- clock for input data signal
82          oclk : in STD_LOGIC;  -- clock for output data pulse
83          rst  : in STD_LOGIC;
84          D    : in STD_LOGIC;
85          re   : out STD_LOGIC  -- rising edge
86          );
87  END COMPONENT;
88
89  COMPONENT gh_counter_down_16b_bb is
90      port(
91          clk    : in STD_LOGIC;
92          rst    : in STD_LOGIC;
93          BCD_EN : in STD_LOGIC;
94          CE     : in STD_LOGIC;
95          LD     : in STD_LOGIC;
96          M_CMD  : in STD_LOGIC;
97          MODE   : in STD_LOGIC_VECTOR(2 downto 0);
98          DI     : in STD_LOGIC_VECTOR(15 downto 0);
99          Cout   : out std_logic;
100         NULL_C : out std_logic;
101         DO     : out STD_LOGIC_VECTOR(15 downto 0)
102         );
103 END COMPONENT;
104
105     signal control   : std_logic_vector(5 downto 0);
106     signal cRW       : std_logic_vector(1 downto 0);
107     signal DC_lsb    : std_logic_vector(7 downto 0);
```

```
108      signal DC_msb    : std_logic_vector(7 downto 0);
109      signal DCI       : std_logic_vector(15 downto 0);
110      signal MODE      : std_logic_vector(2 downto 0);
111      signal sMODE     : std_logic_vector(2 downto 0);
112      signal iD_lsb    : std_logic_vector(7 downto 0);
113      signal iD_msb    : std_logic_vector(7 downto 0);
114      signal iload     : std_logic;
115      signal load      : std_logic;
116      signal ld_cmd    : std_logic;
117      signal ld_lsb    : std_logic;
118      signal ld_msb    : std_logic;
119      signal rd_lsb    : std_logic;
120      signal rd_msb    : std_logic;
121      signal srd_lsb   : std_logic;
122      signal srd_msb   : std_logic;
123      signal dwb       : std_logic;
124      signal rdwb      : std_logic;
125      signal drb       : std_logic;
126      signal rdrb      : std_logic;
127      signal iout      : std_logic;
128
129      signal iclcw     : std_logic;
130      signal clcw      : std_logic;
131      signal clcw_l    : std_logic;
132      signal clcw_m    : std_logic;
133      signal slcw_l    : std_logic;
134      signal slcw_m    : std_logic;
135      signal iclsw     : std_logic;
136      signal hclsw     : std_logic;
137      signal clsw      : std_logic;
138      signal clsw_n    : std_logic;
139      signal rdst      : std_logic;
140      signal DO        : std_logic_vector(15 downto 0);
141      signal rDO       : std_logic_vector(15 downto 0);
142      signal Ds        : std_logic_vector(7 downto 0);
143      signal rDs       : std_logic_vector(7 downto 0);
144      signal NULL_C    : std_logic;
145      signal M_CMD     : std_logic;
146      signal CE        : std_logic;
147      signal m0_outh   : std_logic;
148      signal set_busy  : std_logic;
149      signal clr_busy  : std_logic;
150      signal s_hdcl    : std_logic;
151      signal h_hdcl    : std_logic;
152      signal c_hdcl    : std_logic;
153      signal fe_hdcl   : std_logic;
154      signal ird_busy  : std_logic;
155      signal clr_hclsw : std_logic;
156
157  begin
158
159  ----------------------------------------------------
160
161
162      rd_busy <= ird_busy;
163
164      dat_o <= rDs when (hclsw = '1') else
```

```
165                 rDO(7 downto 0) when ((control(5 downto 4) = "11") and (drb
    = '0')) else
166                 rDO(15 downto 8) when (control(5 downto 4) = "11") else
167                 rDO(7 downto 0) when (control(5 downto 4) = "01") else
168                 rDO(15 downto 8);-- when ((iA = "00") and (control0(5 downto
    4) = "10")) else;
169
170     Ds <= iout & NULL_C & control;
171
172 u0 : gh_register_ce
173     generic map (8)
174     port map(
175         clk => clk,
176         rst => rst,
177         ce => clsw,
178         D => Ds,
179         Q => rDs);
180
181
182 ------------------------------------------------------
183
184     ld_cmd <= '1' when ((dwr = '1') and (iA = "11") and (iD(7 downto 6) =
    "00") and (iCS = '1')) else
185               '0';
186
187 U1 : gh_edge_det_XCD_t
188     GENERIC MAP(same_clk => same_clk,
189                 sync_clk => sync_clk)
190     PORT MAP (
191         iclk => clk_i,
192         oclk => clk,
193         rst => rst,
194         d => ld_cmd,
195         re => M_CMD);
196
197 u2 : gh_register_ce
198     generic map (6)
199     port map(
200         clk => clk_i,
201         rst => rst,
202         ce => ld_cmd,
203         D => iD(5 downto 0),
204         Q => control
205         );
206
207     ld_lsb <= '1' when ((dwr = '1') and (iA = "00") and (control(5 downto
    4) = "01") and (iCS = '1')) else
208               '1' when ((dwr = '1') and (iA = "00") and (control(5 downto
    4) = "11") and (dwb = '0') and (iCS = '1')) else
209               '0';
210
211     ld_msb <= '1' when ((dwr = '1') and (iA = "00") and (control(5 downto
    4) = "10") and (iCS = '1')) else
212               '1' when ((dwr = '1') and (iA = "00") and (control(5 downto
    4) = "11") and (dwb = '1') and (iCS = '1')) else
213               '0';
214
```

```
215      rd_lsb <= '0' when (hclsw = '1') else
216                '1' when ((dwr = '0') and (iA = "00") and (control(5 downto
     4) = "01") and (iCS = '1')) else
217                '1' when ((dwr = '0') and (iA = "00") and (control(5 downto
     4) = "11") and (drb = '0') and (iCS = '1')) else
218                '0';
219
220      rd_msb <= '1' when ((dwr = '0') and (iA = "00") and (control(5 downto
     4) = "10") and (iCS = '1')) else
221                '1' when ((dwr = '0') and (iA = "00") and (control(5 downto
     4) = "11") and (drb = '1') and (iCS = '1')) else
222                '0';
223
224      rdst <=  '1' when ((dwr = '0') and (iA = "00") and (iCS = '1')) else
225               '0';
226
227      sMODE <= control(3 downto 1);
228
229      cRW <= control(5 downto 4);
230
231  process (sMODE,iload,iout,gate,ld_lsb,ld_msb,ld_cmd,cRW,dwb,m0_outh)
232  begin
233      case sMODE is
234          when "000" => -- mode zero
235              MODE <= "000";
236              if (cRW = "11") then
237                  iload <= ld_msb;
238              elsif ((ld_lsb or ld_msb or ld_cmd) = '1') then
239                  iload <= '1';
240              else
241                  iload <= '0';
242              end if;
243              if (cRW = "11") then
244                  CE <= gate and (not dwb);
245              else
246                  CE <= gate;
247              end if;
248              if (iout = '1') then
249                  Cout <= '1';
250              elsif (load = '1') then -- 05/13/11
251                  Cout <= '0';
252              else
253                  Cout <= m0_outh;
254              end if;
255          when "001" =>
256              MODE <= "001";
257              iload <= gate;
258              CE <= gate;
259              Cout <= iout;
260          when "010" =>
261              MODE <= "010";
262              CE <= gate;
263              if (cRW = "11") then
264                  iload <= ld_msb;
265              elsif ((ld_lsb or ld_msb) = '1') then
266                  iload <= '1';
267              else
```

```
268                 iload <= '0';
269             end if;
270             Cout <= iout or (not gate);
271         when "110" =>
272             MODE <= "010";
273             CE <= gate;
274             if (cRW = "11") then
275                 iload <= ld_msb;
276             elsif ((ld_lsb or ld_msb) = '1') then
277                 iload <= '1';
278             else
279                 iload <= '0';
280             end if;
281             Cout <= iout or (not gate);
282         when "011" =>
283             MODE <= "011";
284             CE <= gate;
285             if (cRW = "11") then
286                 iload <= ld_msb;
287             elsif ((ld_lsb or ld_msb) = '1') then
288                 iload <= '1';
289             else
290                 iload <= '0';
291             end if;
292             Cout <= iout;
293         when "111" =>
294             MODE <= "011";
295             CE <= gate;
296             if (cRW = "11") then
297                 iload <= ld_msb;
298             elsif ((ld_lsb or ld_msb) = '1') then
299                 iload <= '1';
300             else
301                 iload <= '0';
302             end if;
303             Cout <= iout;
304         when "100" =>
305             MODE <= "100";
306             CE <= gate;
307             if (cRW = "11") then
308                 iload <= ld_msb;
309             elsif ((ld_lsb or ld_msb) = '1') then
310                 iload <= '1';
311             else
312                 iload <= '0';
313             end if;
314             Cout <= iout;
315         when "101" =>
316             MODE <= "101";
317             iload <= '0';
318             CE <= gate;
319             Cout <= iout;
320         when others =>
321             MODE <= "111";
322             iload <= '0';
323             CE <= gate;
324             Cout <= iout;
```

```
325      end case;
326  end process;
327
328  U3 : gh_jkff
329      PORT MAP (
330          clk => clk,
331          rst => rst,
332          j => iout,
333          k => load, -- 05/13/11
334          Q => m0_outh);
335
336      rdwb <= ld_msb or ld_cmd;
337      rdrb <= '1' when ((rd_msb = '1') or (ld_cmd = '1')) else
338              '1' when ((dwr = '1') and (iA = "11") and (iD(7 downto 5) =
     "110") and (iD(1) = '1') and (iCS = '1')) else
339              '0';
340
341  U4 : gh_jkff
342      PORT MAP (
343          clk => clk_i,
344          rst => rst,
345          j => ld_lsb,
346          k => rdwb,
347          Q => dwb);
348
349  U5 : gh_jkff
350      PORT MAP (
351          clk => clk_i,
352          rst => rst,
353          j => rd_lsb,
354          k => rdrb,
355          Q => drb);
356
357      iD_lsb <= x"00" when (control(5 downto 4) = "10") else
358                iD;
359
360      iD_msb <= x"00" when (control(5 downto 4) = "01") else
361                iD;
362
363  u6 : gh_register_ce
364      generic map (8)
365      port map(
366          clk => clk_i,
367          rst => rst,
368          ce => ld_lsb,
369          D => iD_lsb,
370          Q => DC_lsb
371          );
372
373  u7 : gh_register_ce
374      generic map (8)
375      port map(
376          clk => clk_i,
377          rst => rst,
378          ce => ld_msb,
379          D => iD_msb,
380          Q => DC_msb
```

```
381         );
382
383     DCI <= (DC_msb & DC_lsb);
384
385 U8 : gh_edge_det_XCD_t
386     GENERIC MAP(same_clk => same_clk,
387                 sync_clk => sync_clk)
388     PORT MAP (
389         iclk => clk_i,
390         oclk => clk,
391         rst => rst,
392         d => iLOAD,
393         re => load);
394
395 U9 : gh_counter_down_16b_bb
396     PORT MAP (
397         clk    => clk,
398         rst    => rst,
399         BCD_EN => control(0),
400         CE     => CE,
401         LD     => load,
402         M_CMD  => M_CMD,
403         MODE   => MODE,
404         DI     => DCI,
405         NULL_C => NULL_C,
406         Cout   => iout,
407         DO     => DO
408         );
409
410 -------------------------------------------------
411
412     iclcw <= '1' when ((dwr = '1') and (iA = "11") and (iD(7 downto 4) =
    x"0") and (iCS = '1')) else
413              '1' when ((dwr = '1') and (iA = "11") and (iD(7 downto 5) =
    "110") and (iD(1) = '1') and (iCS = '1')) else
414              '0';
415
416 U10 : gh_edge_det_XCD_t
417     GENERIC MAP(same_clk => same_clk,
418                 sync_clk => sync_clk)
419     PORT MAP (
420         iclk => clk_i,
421         oclk => clk,
422         rst => rst,
423         d => iclcw,
424         re => clcw);
425
426     slcw_l <= M_CMD or srd_lsb;
427
428 U11 : gh_jkff
429     PORT MAP (
430         clk => clk,
431         rst => rst,
432         j => slcw_l,
433         k => clcw,
434         Q => clcw_l);
435
```

```
436  u12 : gh_register_ce
437      generic map (8)
438      port map(
439          clk => clk,
440          rst => rst,
441          ce => clcw_l,
442          D => DO(7 downto 0),
443          Q => rDO(7 downto 0));
444
445      slcw_m <= M_CMD or srd_msb;
446
447  U13 : gh_jkff
448      PORT MAP (
449          clk => clk,
450          rst => rst,
451          j => slcw_m,
452          k => clcw,
453          Q => clcw_m);
454
455  u14 : gh_register_ce
456      generic map (8)
457      port map(
458          clk => clk,
459          rst => rst,
460          ce => clcw_m,
461          D => DO(15 downto 8),
462          Q => rDO(15 downto 8));
463
464      iclsw <= '1' when ((dwr = '1') and (iA = "11") and (iD(7 downto 6) =
     "11") and (iD(4) = '0') and (iD(1) = '1') and (iCS = '1')) else
465               '0';
466
467  U15 : gh_edge_det_XCD_t
468      GENERIC MAP(same_clk => same_clk,
469                  sync_clk => sync_clk)
470      PORT MAP (
471          iclk => clk_i,
472          oclk => clk,
473          rst => rst,
474          d => iclsw,
475          re => clsw);
476
477      clr_hclsw <= (rdst and not ird_busy) or fe_hdcl;
478
479  U16 : gh_jkff
480      PORT MAP (
481          clk => clk_i,
482          rst => rst,
483          j => iclsw,
484          k => clr_hclsw,
485          Q => hclsw);
486
487      s_hdcl <= iCS and (not dwr) and ird_busy;
488
489  U17 : gh_jkff
490      PORT MAP (
491          clk => clk_i,
```

```
492          rst => rst,
493          j => s_hdcl,
494          k => clr_hclsw,
495          Q => h_hdcl);
496
497      c_hdcl <= h_hdcl and ird_busy;
498
499  U18 : gh_edge_det
500      PORT MAP (
501          clk => clk_i,
502          rst => rst,
503          d => c_hdcl,
504          fe => fe_hdcl);
505
506  ------------------------------------------------------
507
508      clsw_n <= not clsw;
509
510  U19 : gh_edge_det_XCD_t
511      GENERIC MAP(same_clk => same_clk,
512                  sync_clk => sync_clk)
513      PORT MAP (
514          iclk => clk,
515          oclk => clk_i,
516          rst => rst,
517          d => clsw_n,
518          re => clr_busy);
519
520      set_busy <= iclcw;
521
522  U20 : gh_jkff
523      PORT MAP (
524          clk => clk_i,
525          rst => rst,
526          j => set_busy,
527          k => clr_busy,
528          Q => ird_busy);
529
530  U21 : gh_edge_det_XCD_t
531      GENERIC MAP(same_clk => same_clk,
532                  sync_clk => sync_clk)
533      PORT MAP (
534          iclk => clk_i,
535          oclk => clk,
536          rst => rst,
537          d => rd_lsb,
538          re => srd_lsb);
539
540  U22 : gh_edge_det_XCD_t
541      GENERIC MAP(same_clk => same_clk,
542                  sync_clk => sync_clk)
543      PORT MAP (
544          iclk => clk_i,
545          oclk => clk,
546          rst => rst,
547          d => rd_msb,
548          re => srd_msb);
```

```
549
550  end a;
551
```