```
--#########################################################
-- BUFFER UNIT
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity BUFFER_INOUT is
    Port ( CLR1,CLK2,P1,P2 : in  STD_LOGIC;
           IN_SIG : in  STD_LOGIC_VECTOR (79 downto 0);
           DATA1 : out  STD_LOGIC_VECTOR (31 downto 0);
           DATA2 : out  STD_LOGIC_VECTOR (31 downto 0);
           OUT_SIG : out  STD_LOGIC_VECTOR (79 downto 0));
end BUFFER_INOUT;

architecture Behavioral of BUFFER_INOUT is

begin
process (CLK2,P1,P2,IN_SIG,CLR1) is
begin
if rising_edge(CLK2) then
if CLR1='1' then
   OUT_SIG <=(others =>'1');
elsif P1='1' OR P2='1' then
   OUT_SIG <=IN_SIG;

   DATA1(3 downto 0) <=IN_SIG(3 downto 0);
   DATA1(7 downto 4) <=IN_SIG(8 downto 5);
   DATA1(11 downto 8) <=IN_SIG(13 downto 10);
   DATA1(15 downto 12) <=IN_SIG(18 downto 15);
   DATA1(19 downto 16) <=IN_SIG(23 downto 20);
   DATA1(23 downto 20) <=IN_SIG(28 downto 25);
   DATA1(27 downto 24) <=IN_SIG(33 downto 30);
   DATA1(31 downto 28) <=IN_SIG(38 downto 35);

   DATA2(3 downto 0) <=IN_SIG(43 downto 40);
   DATA2(7 downto 4) <=IN_SIG(48 downto 45);
   DATA2(11 downto 8) <=IN_SIG(53 downto 50);
   DATA2(15 downto 12) <=IN_SIG(58 downto 55);
   DATA2(19 downto 16) <=IN_SIG(63 downto 60);
   DATA2(23 downto 20) <=IN_SIG(68 downto 65);
   DATA2(27 downto 24) <=IN_SIG(73 downto 70);
   DATA2(31 downto 28) <=IN_SIG(78 downto 75);

end if;
end if;
end process;
end Behavioral;
--#########################################################
-- 32 BIT FLOATING POINT ADDER /SUBSTRACTOR
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.numeric_std.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity FLPOINT is
    Port ( START : in  STD_LOGIC;
           ADDSUB : in  STD_LOGIC;
```

```
58              CLR : in  STD_LOGIC;
59              NUM1 : in  STD_LOGIC_VECTOR (31 downto 0);
60              NUM2 : in  STD_LOGIC_VECTOR (31 downto 0);
61              MZERO : out  STD_LOGIC;
62              PZERO : out  STD_LOGIC;
63              PINF : out  STD_LOGIC;
64              MINF : out  STD_LOGIC;
65              NAN : out  STD_LOGIC;
66              CLR_out : out  STD_LOGIC;
67              UNORMZ : out  STD_LOGIC;
68              ANSWER : out  STD_LOGIC_VECTOR (31 downto 0));
69   end FLPOINT;
70
71   architecture Behavioral of FLPOINT is
72
73   begin
74   PROCESS (start,CLR,num1,num2,ADDSUB) is
75   variable count_shift1 : std_logic_vector (7 downto 0);
76   variable count_shift2 : std_logic_vector (7 downto 0);
77   variable EXP : std_logic_vector (7 downto 0);
78   variable EXP1 : std_logic_vector (7 downto 0);
79   variable EXP2 : std_logic_vector (7 downto 0);
80   variable DIFF : std_logic_vector (7 downto 0);
81   variable Signf : std_logic_vector (23 downto 0);
82   variable Signf1 : std_logic_vector (23 downto 0);
83   variable signf2 : std_logic_vector (23 downto 0);
84   variable Signf_temp : std_logic_vector (23 downto 0);
85   variable Sum_temp : std_logic_vector (24 downto 0);
86   variable sign1 : std_logic;
87   variable sign2 : std_logic;
88   variable sign_temp : std_logic;
89   variable signt : std_logic;
90   variable ANSWER_V : STD_LOGIC_VECTOR (31 downto 0);
91   variable numb1_V : STD_LOGIC_VECTOR (31 downto 0);
92   variable numb2_V : STD_LOGIC_VECTOR (31 downto 0);
93   variable OVFL_V : std_logic;
94
95
96   begin
97   numb1_V :=num1;
98   numb2_V :=num2;
99   if CLR ='1' then
100     ANSWER <=(others =>'0');
101     MZERO <='0';
102     PZERO <='0';
103     PINF <='0';
104     MINF <='0';
105     NAN <='0';
106     UNORMZ <='0';
107     CLR_out <='1';
108  elsif rising_edge(start) then
109     CLR_out <='0';
110     MZERO <='0';
111     PZERO <='0';
112     PINF <='0';
113     MINF <='0';
114     NAN <='0';
```

```
115      UNORMZ <='0';
116      if (numb1_V ="01111111100000000000000000000000" or numb2_V =
     "01111111100000000000000000000000") then
117         PINF <='1';
118         ANSWER <=(others =>'0');
119      elsif (numb1_V ="11111111100000000000000000000000" or numb2_V =
     "11111111100000000000000000000000") then
120         MINF <='1';
121         ANSWER <=(others =>'0');
122      elsif (numb1_V ="01111111100000100000000000000000" or numb1_V =
     "11111111100100010001001010101010" or numb2_V ="01111111100000100000000000000000" OR
     numb2_V ="11111111100100010001001010101010") then
123         NAN <='1';
124         ANSWER <=(others =>'0');
125      elsif (numb1_V (30 DOWNTO 23) ="00000000" AND numb1_V (22 DOWNTO 0) /=
     "00000000000000000000000") or (numb2_V (30 DOWNTO 23) ="00000000" AND numb2_V (22
     DOWNTO 0) /="00000000000000000000000") then
126         UNORMZ <='1';
127         ANSWER <=(others =>'0');
128      else
129
130   sign1 :=NUM1(31);
131   if ADDSUB ='0' then
132      sign2 :=NUM2(31);
133   else
134      sign2 := not NUM2(31);
135   end if;
136   EXP1 :=NUM1(30 downto 23);
137   EXP2 :=NUM2(30 downto 23);
138
139
140   -- CHECK FOR HIDDEN BIT
141   if EXP1 ="00000000" then
142      Signf1 := '0' & NUM1(22 downto 0);
143   else
144      Signf1 := '1' & NUM1(22 downto 0);
145   end if;
146
147   if EXP2 ="00000000" then
148      Signf2 := '0' & NUM2(22 downto 0);
149   else
150      Signf2 := '1' & NUM2(22 downto 0);
151   end if;
152
153   -- CHECK FOR EXP2 > EXP1
154   if EXP2 > EXP1 then
155      Signf_temp :=Signf1;
156      Signf1 :=Signf2;
157      Signf2 :=Signf_temp;
158      sign_temp :=sign1;
159      sign1 :=sign2;
160      sign2 :=sign_temp;
161      EXP1 :=NUM2(30 downto 23);
162      EXP2 :=NUM1(30 downto 23);
163   end if;
164
165   -- Determine THE DIFFERENCE
```

```
166   DIFF := EXP1 - EXP2;
167   count_shift1 :="00000000";
168   if DIFF > "00000000" then
169
170      for i in 0 to 23 loop
171         if DIFF > "00000000" then
172            Signf2 :='0' & Signf2 (23 downto 1);
173            count_shift1 :=count_shift1+1;
174            DIFF :=DIFF-1;
175         else
176            exit;
177         end if;
178        end loop;
179         EXP2 :=EXP2+ count_shift1;
180   end if;
181
182   if sign1=sign2 then --add
183         signt := sign1;
184         Sum_temp := ('0' & Signf1) + ('0' & Signf2);
185         if Sum_temp(24) ='1' then
186            EXP :=EXP1+1;
187            Signf := Sum_temp(24 downto 1);
188         else
189            EXP :=EXP1;
190            Signf := Sum_temp(23 downto 0);
191         end if;
192   --SUBSTRACT
193
194
195   elsif Signf1 = Signf2 then
196      Signf :="00000000000000000000000";
197      exp :="00000000";
198      signt := '0';
199   else
200         if Signf1 > Signf2 then
201            signt := sign1;
202         else
203            signt := sign2;
204         end if;
205         Signf_temp :=Signf2;
206         Signf_temp := (not Signf_temp) +1;
207         Sum_temp := ('0' & Signf1) + ('0' & Signf_temp );
208
209
210         Signf_temp := Sum_temp(23 downto 0);
211         count_shift2 :="00000000";
212         if Sum_temp(24) ='1' then --CARRY OUT
213
214            for i in 0 to 23 loop
215               if Signf_temp(23) ='1' then
216                  exit;
217               else
218                  Signf_temp := Signf_temp(22 downto 0) & '0';
219                  count_shift2 :=count_shift2+1;
220               end if;
221
222            end loop;
```

```
223              if Signf_temp ="00000000000000000000000" then
224                 Signf :="00000000000000000000000";
225                 EXP :="00000000";
226              else
227                 EXP := EXP1 - count_shift2;
228                 Signf :=Signf_temp;
229              end if;
230           elsif Sum_temp(24) ='0' and (Signf1 ="00000000000000000000000" or Signf2 =
    "00000000000000000000000") then
231              EXP := EXP1;
232              Signf :=Signf_temp;
233           else
234           EXP := EXP1;
235           Signf :=(not Signf_temp) +1;
236        end if;
237  end if;
238
239  ANSWER_V(31) := signt;
240  ANSWER_V(30 downto 23) := EXP;
241  ANSWER_V(22 downto 0) := Signf(22 downto 0);
242
243  -- ZERO CHECKING
244  if ANSWER_V ="00000000000000000000000000000000" then
245     PZERO <='1';
246  else
247     PZERO <='0';
248  end if;
249
250  -- MINUS ZERO CHECKING
251  if ANSWER_V ="10000000000000000000000000000000" then
252     MZERO <='1';
253  else
254     MZERO <='0';
255  end if;
256
257  -- + INF CHECKING
258  if ANSWER_V ="01111111100000000000000000000000" then
259     PINF <='1';
260  else
261     PINF <='0';
262  end if;
263
264  -- - INF CHECKING
265  if ANSWER_V ="11111111100000000000000000000000" then
266     MINF <='1';
267  else
268     MINF <='0';
269  end if;
270
271  -- NAN CHECKING
272  if ANSWER_V ="01111111100000100000000000000000" OR ANSWER_V =
    "11111111100100010001001010101010" then
273     NAN <='1';
274  else
275     NAN <='0';
276  end if;
277
```

```
278
279   ANSWER <=ANSWER_V;
280   end if;
281   end if;
282   end process;
283
284   end Behavioral;
285
286   --##########################################################
287
288
289   library IEEE;
290   use IEEE.STD_LOGIC_1164.ALL;
291
292   entity PADD_ZERO is
293       Port ( DATA_INPUT : in  STD_LOGIC_VECTOR (31 downto 0);
294              PLZERO : in  STD_LOGIC;
295              MIZERO : in  STD_LOGIC;
296              PLINF : in  STD_LOGIC;
297              MIINF : in  STD_LOGIC;
298              NTNAN : in  STD_LOGIC;
299              CLR4 : in  STD_LOGIC;
300              UNORMZOT : in  STD_LOGIC;
301             DATA_OUT : out  STD_LOGIC_VECTOR (39 downto 0));
302   end PADD_ZERO;
303
304   architecture Behavioral of PADD_ZERO is
305
306   begin
307   process (CLR4,DATA_INPUT,PLZERO,MIZERO,PLINF,MIINF,NTNAN,UNORMZOT) is
308   begin
309   if CLR4 ='1'  then
310      DATA_OUT <=(others =>'1');
311   elsif PLZERO='1' then
312      DATA_OUT(4 downto 0) <= "00000";
313      DATA_OUT(39 downto 5) <= (others => '1');
314   elsif MIZERO ='1' then
315      DATA_OUT(4 downto 0) <= "00000"; -- (0)
316      DATA_OUT(9 downto 5) <= "10000"; -- (-)
317      DATA_OUT(39 downto 10) <= (others => '1');
318   elsif PLINF ='1' then
319      DATA_OUT(4 downto 0) <= "01111"; -- (f)
320      DATA_OUT(9 downto 5) <= "10001"; -- (n)
321      DATA_OUT(14 downto 10) <="10010";-- (i)
322      DATA_OUT(39 downto 15) <= (others => '1');
323   elsif MIINF ='1' then
324      DATA_OUT(4 downto 0) <= "01111"; -- (f)
325      DATA_OUT(9 downto 5) <= "10001"; -- (n)
326      DATA_OUT(14 downto 10) <="10010";-- (i)
327      DATA_OUT(19 downto 15) <="10000";-- (-)
328      DATA_OUT(39 downto 20) <= (others => '1');
329   elsif NTNAN ='1' then
330      DATA_OUT(4 downto 0) <= "10001"; -- (n)
331      DATA_OUT(9 downto 5) <= "01010"; -- (A)
332      DATA_OUT(14 downto 10) <="10001";-- (n)
333      DATA_OUT(39 downto 15) <= (others => '1');
334   elsif UNORMZOT ='1' then
```

```
335       DATA_OUT(4 downto 0) <= "10011"; -- (r)
336       DATA_OUT(9 downto 5) <= "00000"; -- (O)
337       DATA_OUT(14 downto 10) <="10001";-- (n)
338       DATA_OUT(19 downto 15) <="10100";-- (U)
339       DATA_OUT(39 downto 20) <= (others => '1');
340    else
341       DATA_OUT(4 downto 0) <= '0' & DATA_INPUT(3 downto 0);
342       DATA_OUT(9 downto 5) <= '0' & DATA_INPUT(7 downto 4);
343       DATA_OUT(14 downto 10) <= '0' & DATA_INPUT(11 downto 8);
344       DATA_OUT(19 downto 15) <= '0' & DATA_INPUT(15 downto 12);
345       DATA_OUT(24 downto 20) <= '0' & DATA_INPUT(19 downto 16);
346       DATA_OUT(29 downto 25) <= '0' & DATA_INPUT(23 downto 20);
347       DATA_OUT(34 downto 30) <= '0' & DATA_INPUT(27 downto 24);
348       DATA_OUT(39 downto 35) <= '0' & DATA_INPUT(31 downto 28);
349    end if;
350    end process;
351
352    end Behavioral;
353
354    --##########################################################
355    -- 3 X 1 MULTIPLEXER
356
357    library IEEE;
358    use IEEE.STD_LOGIC_1164.ALL;
359    use IEEE.STD_LOGIC_ARITH.ALL;
360    use IEEE.numeric_std.ALL;
361    use IEEE.STD_LOGIC_UNSIGNED.ALL;
362
363
364
365    entity MUX31 is
366        Port ( clk,SEL1 : in  STD_LOGIC;
367               SEL2 : in  STD_LOGIC;
368               SEL3 : in  STD_LOGIC;
369               N1 : in  STD_LOGIC_VECTOR (39 downto 0);
370               N2 : in  STD_LOGIC_VECTOR (39 downto 0);
371               N3 : in  STD_LOGIC_VECTOR (39 downto 0);
372               N4 : out  STD_LOGIC_VECTOR (39 downto 0));
373    end MUX31;
374
375    architecture Behavioral of MUX31 is
376
377    begin
378
379    process (clk,SEL1,SEL2,SEL3,N1,N2,N3) is
380    variable sel : std_logic_vector(2 downto 0);
381    begin
382
383    sel := SEL3 & SEL2 & SEL1;
384    if rising_edge(clk) then
385       if sel="001" then
386          N4 <= N1;
387       elsif sel="010" then
388          N4 <= N2;
389       elsif sel="100" then
390          N4 <= N3;
391       else
```

```
392         N4 <=(others =>'1');
393       end if;
394   end if;
395
396
397   end process;
398
399   end Behavioral;
400
401   --##########################################################
402   -- 2 X 1 MLTIPLEXER
403   library IEEE;
404   use IEEE.STD_LOGIC_1164.ALL;
405
406   entity MUX21 is
407       Port ( INP_BUT : in  STD_LOGIC_VECTOR (39 downto 0);
408              SELCT : in  STD_LOGIC;
409              OUT_BIT : out  STD_LOGIC_VECTOR (19 downto 0));
410   end MUX21;
411
412   architecture Behavioral of MUX21 is
413
414   begin
415   process (SELCT,INP_BUT) is
416   begin
417   case SELCT is
418      when '0' =>
419         OUT_BIT <=INP_BUT(19 downto 0);
420      when others =>
421         OUT_BIT <=INP_BUT(39 downto 20);
422      end case;
423   end process;
424   end Behavioral;
425
426   --##########################################################
427   library IEEE;
428   use IEEE.STD_LOGIC_1164.ALL;
429   use IEEE.STD_LOGIC_ARITH.ALL;
430   use IEEE.STD_LOGIC_UNSIGNED.ALL;
431
432   entity BCD_SEG is
433       Port ( DATA_INP : in  STD_LOGIC_VECTOR (19 downto 0);
434              CLK2 : in  STD_LOGIC;
435              SEG  : out  STD_LOGIC_VECTOR (7 downto 0);
436              AN1   : out  STD_LOGIC_VECTOR (3 downto 0));
437   end BCD_SEG;
438
439   architecture Behavioral of BCD_SEG is
440   signal SEL : STD_LOGIC_VECTOR (1 downto 0):="00";
441   signal MUX_OUT : STD_LOGIC_VECTOR (4 downto 0);
442   signal INP1 : STD_LOGIC_VECTOR (4 downto 0);
443   signal INP2 : STD_LOGIC_VECTOR (4 downto 0);
444   signal INP3 : STD_LOGIC_VECTOR (4 downto 0);
445   signal INP4 : STD_LOGIC_VECTOR (4 downto 0);
446
447   begin
448   INP4 <=DATA_INP(19 downto 15);
```

```
449   INP3 <=DATA_INP(14 downto 10);
450   INP2 <=DATA_INP(9 downto 5);
451   INP1 <=DATA_INP(4 downto 0);
452
453   -- TWO BIT COUNTER
454   PROCESS (CLK2) IS
455   variable temp : std_logic_vector ( 1 downto 0):="00";
456   BEGIN
457      if rising_edge(CLK2) then
458         temp := temp + 1;
459      end if;
460   SEL <=temp;
461   END PROCESS;
462   -- 16 X 4 MUX
463   process (SEL,INP1,INP2,INP3,INP4) is
464   begin
465      case SEL is
466         when "00" =>
467            MUX_OUT <=INP1;
468         when "01" =>
469            MUX_OUT <=INP2;
470         when "10" =>
471            MUX_OUT <=INP3;
472         when others =>
473            MUX_OUT <=INP4;
474         end case;
475   END PROCESS;
476   -- BCD to Seven segment
477   process (MUX_OUT) is
478   begin
479     case MUX_OUT is
480         when "00000"=> SEG <="00000011"; -- '0'
481         when "00001"=> SEG <="10011111"; -- '1'
482         when "00010"=> SEG <="00100101"; -- '2'
483         when "00011"=> SEG <="00001101"; -- '3'
484         when "00100"=> SEG <="10011001"; -- '4'
485         when "00101"=> SEG <="01001001"; -- '5'
486         when "00110"=> SEG <="01000001"; -- '6'
487         when "00111"=> SEG <="00011111"; -- '7'
488         when "01000"=> SEG <="00000001"; -- '8'
489         when "01001"=> SEG <="00001001"; -- '9'
490         when "01010"=> SEG <="00010001"; -- 'A'
491         when "01011"=> SEG <="11000001"; -- 'B'
492         when "01100"=> SEG <="01100011"; -- 'C'
493         when "01101"=> SEG <="10000101"; -- 'D'
494         when "01110"=> SEG <="01100001"; -- 'E'
495         when "01111"=> SEG <="01110001"; -- 'F'
496         when "10000"=> SEG <="11111101"; -- '-'
497         when "10001"=> SEG <="11010101"; -- 'n'
498         when "10010"=> SEG <="11110011"; -- 'i'
499         when "10011"=> SEG <="11110001"; -- 'r'
500         when "10100"=> SEG <="10000011"; -- 'U'
501         when others=> SEG <="11111111"; -- ' '
502      end case;
503   end process;
504   -- 2 X 4 DECODER
505   process (SEL) is
```

```
506   begin
507   case SEL is
508         when "00" =>
509            AN1(0) <= '0';
510            AN1(1) <= '1';
511            AN1(2) <= '1';
512            AN1(3) <= '1';
513         when "01" =>
514            AN1(0) <= '1';
515            AN1(1) <= '0';
516            AN1(2) <= '1';
517            AN1(3) <= '1';
518         when "10" =>
519            AN1(0) <= '1';
520            AN1(1) <= '1';
521            AN1(2) <= '0';
522            AN1(3) <= '1';
523         when others =>
524            AN1(0) <= '1';
525            AN1(1) <= '1';
526            AN1(2) <= '1';
527            AN1(3) <= '0';
528         end case;
529   END PROCESS;
530   end Behavioral;
531
532   --##########################################################
533   -- Main Program
534   library IEEE;
535   use IEEE.STD_LOGIC_1164.ALL;
536   USE IEEE.std_logic_Arith.ALL;
537   USE IEEE.std_logic_unsigned.ALL;
538   USE IEEE.numeric_std.ALL;
539   entity STATE_MN is
540       Port ( RESET,CLK_SYS,SW0,BEG_CL : in  STD_LOGIC;
541              SW1 : in  STD_LOGIC;
542              SW2 : in  STD_LOGIC;
543              SW3 : in  STD_LOGIC;
544              SW4 : in STD_LOGIC;
545              SW5 : in STD_LOGIC;
546              SW6 : in STD_LOGIC;
547              SW7 : in STD_LOGIC;
548              UP_CNT : in  STD_LOGIC;
549              DO_CNT : in  STD_LOGIC;
550              SIG_IN : in  STD_LOGIC;
551              SEV_SEG  : out  STD_LOGIC_VECTOR (7 downto 0);
552              AN   : out  STD_LOGIC_VECTOR (3 downto 0);
553              LD1 : out  STD_LOGIC);
554   end STATE_MN;
555
556   architecture Behavioral of STATE_MN is
557   type state_type is (S0,S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11,S12,S13,S14,S15,S16);
558   signal state_t: state_type:=s0;
559   signal fast_clk : STD_LOGIC;
560   signal clk_divider1 : std_logic_vector (15 downto 0) := (others => '0');
561   signal UP_CNTOUT : STD_LOGIC;
562   signal DO_CNTOUT : STD_LOGIC;
```

```
563   signal OUT_DATA : STD_LOGIC_VECTOR (79 downto 0);
564   signal NUMBR1 : STD_LOGIC_VECTOR (31 downto 0);
565   signal NUMBR2 : STD_LOGIC_VECTOR (31 downto 0);
566   signal SIG_OUT : STD_LOGIC;
567   signal SIGNF_OUT : STD_LOGIC;
568   signal UPCTF_OUT : STD_LOGIC;
569   signal DWNCTF_OUT : STD_LOGIC;
570   signal BUFFOUT : STD_LOGIC_VECTOR (79 downto 0);
571   signal RES : STD_LOGIC_VECTOR (31 downto 0);
572   signal PADZRO : STD_LOGIC_VECTOR (39 downto 0);
573   signal SEL_SIG : STD_LOGIC_VECTOR (39 downto 0);
574   signal MX_SEG : STD_LOGIC_VECTOR (19 downto 0);
575   signal MZERO_OUT : STD_LOGIC;
576   signal PZERO_OUT : STD_LOGIC;
577   signal PINF_OUT : STD_LOGIC;
578   signal MINF_OUT : STD_LOGIC;
579   signal NAN_OUT : STD_LOGIC;
580   signal UNORMZ_OUT : STD_LOGIC;
581   signal CLR_INOUT : STD_LOGIC;
582
583   ------------------------------------------------------------
584   component BUFFER_INOUT
585   Port ( CLR1,CLK2,P1,P2 : in  STD_LOGIC;
586              IN_SIG : in  STD_LOGIC_VECTOR (79 downto 0);
587              DATA1 : out  STD_LOGIC_VECTOR (31 downto 0);
588              DATA2 : out  STD_LOGIC_VECTOR (31 downto 0);
589              OUT_SIG : out  STD_LOGIC_VECTOR (79 downto 0));
590   end component BUFFER_INOUT;
591   ------------------------------------------------------------
592   component FLPOINT
593    Port ( START : in  STD_LOGIC;
594              ADDSUB : in  STD_LOGIC;
595              CLR : in  STD_LOGIC;
596              NUM1 : in  STD_LOGIC_VECTOR (31 downto 0);
597              NUM2 : in  STD_LOGIC_VECTOR (31 downto 0);
598              MZERO : out  STD_LOGIC;
599              PZERO : out  STD_LOGIC;
600              PINF : out  STD_LOGIC;
601              MINF : out  STD_LOGIC;
602              NAN : out  STD_LOGIC;
603              CLR_out : out  STD_LOGIC;
604              UNORMZ : out  STD_LOGIC;
605              ANSWER : out  STD_LOGIC_VECTOR (31 downto 0));
606   end component FLPOINT;
607   ------------------------------------------------------------
608   component PADD_ZERO
609   Port ( DATA_INPUT : in  STD_LOGIC_VECTOR (31 downto 0);
610               PLZERO : in  STD_LOGIC;
611               MIZERO : in  STD_LOGIC;
612               PLINF : in  STD_LOGIC;
613               MIINF : in  STD_LOGIC;
614               NTNAN : in  STD_LOGIC;
615               CLR4 : in  STD_LOGIC;
616               UNORMZOT : in  STD_LOGIC;
617              DATA_OUT : out  STD_LOGIC_VECTOR (39 downto 0));
618   end component PADD_ZERO;
619   ------------------------------------------------------------
```

```
620   component MUX31
621   Port ( clk,SEL1 : in  STD_LOGIC;
622              SEL2 : in  STD_LOGIC;
623              SEL3 : in  STD_LOGIC;
624              N1 : in  STD_LOGIC_VECTOR (39 downto 0);
625              N2 : in  STD_LOGIC_VECTOR (39 downto 0);
626              N3 : in  STD_LOGIC_VECTOR (39 downto 0);
627              N4 : out  STD_LOGIC_VECTOR (39 downto 0));
628   end component MUX31;
629   ----------------------------------------------------------------
630   component MUX21
631   Port ( INP_BUT : in  STD_LOGIC_VECTOR (39 downto 0);
632              SELCT : in  STD_LOGIC;
633              OUT_BIT : out  STD_LOGIC_VECTOR (19 downto 0));
634   end component MUX21;
635   ----------------------------------------------------------------
636   component BCD_SEG
637   Port ( DATA_INP : in  STD_LOGIC_VECTOR (19 downto 0);
638              CLK2 : in  STD_LOGIC;
639              SEG  : out  STD_LOGIC_VECTOR (7 downto 0);
640              AN1   : out  STD_LOGIC_VECTOR (3 downto 0));
641   end component BCD_SEG;
642   ----------------------------------------------------------------
643   begin
644   ------------------------------------------------------
645   --FRQUENCY DIVIDER
646   CLK_SYS_division1 : process (CLK_SYS, clk_divider1)
647   begin
648   if (CLK_SYS = '1' and CLK_SYS'event) then
649      clk_divider1 <= clk_divider1 + 1;
650   end if;
651   fast_clk <= clk_divider1(15);
652   end process;
653   -- END FRQUENCY DIVIDER
654   ------------------------------------------------------
655   -- REMOVE DEBOUNCING
656   PROCESS (UP_CNT,fast_clk) IS
657   BEGIN
658   if (fast_clk = '1' and fast_clk'event) then
659            UP_CNTOUT <= UP_CNT;
660   END IF;
661   END PROCESS;
662   ------------------------------------------------------------
663   -- REMOVE DEBOUNCING
664   PROCESS (DO_CNT,fast_clk) IS
665   BEGIN
666   if (fast_clk = '1' and fast_clk'event) then
667            DO_CNTOUT <= DO_CNT;
668   END IF;
669   END PROCESS;
670   ------------------------------------------------------------
671   -- PULSE GENERATOR
672   PROCESS(fast_clk,UP_CNTOUT)
673   VARIABLE UPCTF_PAST: STD_LOGIC;
674   BEGIN
675      If fast_clk'event and fast_clk = '1' then
676         IF UPCTF_PAST= '0' AND UP_CNTOUT= '1' THEN
```

```
677             UPCTF_OUT<='1';
678             UPCTF_PAST:='1';
679          ELSIF UPCTF_PAST = '1' AND UP_CNTOUT='1' THEN
680             UPCTF_OUT<='0';
681             UPCTF_PAST:='1';
682          ELSIF UP_CNTOUT= '0' THEN
683             UPCTF_OUT<= '0';
684             UPCTF_PAST:='0';
685          END IF;
686       END IF;
687    END PROCESS;
688    -------------------------------------------------------
689
690    -- PULSE GENERATOR
691    PROCESS(fast_clk,DO_CNTOUT)
692    VARIABLE DWNCTF_PAST: STD_LOGIC;
693    BEGIN
694       If fast_clk'event and fast_clk = '1' then
695          IF DWNCTF_PAST= '0' AND DO_CNTOUT= '1' THEN
696             DWNCTF_OUT<='1';
697             DWNCTF_PAST:='1';
698          ELSIF DWNCTF_PAST = '1' AND DO_CNTOUT='1' THEN
699             DWNCTF_OUT<='0';
700             DWNCTF_PAST:='1';
701          ELSIF DO_CNTOUT= '0' THEN
702             DWNCTF_OUT<= '0';
703             DWNCTF_PAST:='0';
704          END IF;
705       END IF;
706    END PROCESS;
707    -------------------------------------------------------
708    -- STATE MACHINE I/O INTERFACE
709    process (RESET,fast_clk,SW0,SW1,SW2,SW3,UPCTF_OUT,DWNCTF_OUT,state_t) is
710    variable temp : STD_LOGIC_VECTOR (79 downto 0):=(others =>'1');
711    variable cunt : integer range 0 to 85 :=0;
712    begin
713    if rising_edge(fast_clk) THEN
714       if RESET ='1' then
715          state_t <= s0;
716          temp :=(others =>'1');
717          cunt :=0;
718       else
719          case state_t is
720
721             when S0 =>
722                if UPCTF_OUT ='1' then
723                   CUNT :=CUNT+5;
724                   temp(cunt-1 downto cunt-5) := '0' & SW3 & SW2& SW1 & SW0;
725                   state_t <= s1;
726                END IF;
727
728                when S1 =>
729                if UPCTF_OUT ='1' then
730                   CUNT :=CUNT+5;
731                   temp(cunt-1 downto cunt-5) := '0' & SW3 & SW2& SW1 & SW0;
732                   state_t <= s2;
733                elsif DWNCTF_OUT='1' then
```

```
734                     temp(cunt-1 downto cunt-5) := "11111";
735                     CUNT :=CUNT-5;
736                     state_t <= s0;
737                  END IF;
738
739                  when S2 =>
740                  if UPCTF_OUT ='1' then
741                     CUNT :=CUNT+5;
742                     temp(cunt-1 downto cunt-5) := '0' & SW3 & SW2& SW1 & SW0;
743                     state_t <= s3;
744                  elsif DWNCTF_OUT='1' then
745                     temp(cunt-1 downto cunt-5) := "11111";
746                     CUNT :=CUNT-5;
747                     state_t <= s1;
748                  END IF;
749
750                  when S3 =>
751                  if UPCTF_OUT ='1' then
752                     CUNT :=CUNT+5;
753                     temp(cunt-1 downto cunt-5) := '0' & SW3 & SW2& SW1 & SW0;
754                     state_t <= s4;
755                  elsif DWNCTF_OUT='1' then
756                     temp(cunt-1 downto cunt-5) := "11111";
757                     CUNT :=CUNT-5;
758                     state_t <= s2;
759                  END IF;
760
761                  when S4 =>
762                  if UPCTF_OUT ='1' then
763                     CUNT :=CUNT+5;
764                     temp(cunt-1 downto cunt-5) := '0' & SW3 & SW2& SW1 & SW0;
765                     state_t <= s5;
766                  elsif DWNCTF_OUT='1' then
767                     temp(cunt-1 downto cunt-5) := "11111";
768                     CUNT :=CUNT-5;
769                     state_t <= s3;
770                  END IF;
771
772                  when S5 =>
773                  if UPCTF_OUT ='1' then
774                     CUNT :=CUNT+5;
775                     temp(cunt-1 downto cunt-5) := '0' & SW3 & SW2& SW1 & SW0;
776                     state_t <= s6;
777                  elsif DWNCTF_OUT='1' then
778                     temp(cunt-1 downto cunt-5) := "11111";
779                     CUNT :=CUNT-5;
780                     state_t <= s4;
781                  END IF;
782
783                  when S6 =>
784                  if UPCTF_OUT ='1' then
785                     CUNT :=CUNT+5;
786                     temp(cunt-1 downto cunt-5) := '0' & SW3 & SW2& SW1 & SW0;
787                     state_t <= s7;
788                  elsif DWNCTF_OUT='1' then
789                     temp(cunt-1 downto cunt-5) := "11111";
790                     CUNT :=CUNT-5;
```

```
791                  state_t <= s5;
792               END IF;
793
794               when S7 =>
795               if UPCTF_OUT ='1' then
796                  CUNT :=CUNT+5;
797                  temp(cunt-1 downto cunt-5) := '0' & SW3 & SW2& SW1 & SW0;
798                  state_t <= s8;
799               elsif DWNCTF_OUT='1' then
800                  temp(cunt-1 downto cunt-5) := "11111";
801                  CUNT :=CUNT-5;
802                  state_t <= s6;
803               END IF;
804
805               when S8 =>
806               if UPCTF_OUT ='1' then
807                  CUNT :=CUNT+5;
808                  temp(cunt-1 downto cunt-5) := '0' & SW3 & SW2& SW1 & SW0;
809                  state_t <= s9;
810               elsif DWNCTF_OUT='1' then
811                  temp(cunt-1 downto cunt-5) := "11111";
812                  CUNT :=CUNT-5;
813                  state_t <= s7;
814               END IF;
815
816               when S9 =>
817               if UPCTF_OUT ='1' then
818                  CUNT :=CUNT+5;
819                  temp(cunt-1 downto cunt-5) := '0' & SW3 & SW2& SW1 & SW0;
820                  state_t <= s10;
821               elsif DWNCTF_OUT='1' then
822                  temp(cunt-1 downto cunt-5) := "11111";
823                  CUNT :=CUNT-5;
824                  state_t <= s8;
825               END IF;
826
827               when S10 =>
828               if UPCTF_OUT ='1' then
829                  CUNT :=CUNT+5;
830                  temp(cunt-1 downto cunt-5) := '0' & SW3 & SW2& SW1 & SW0;
831                  state_t <= s11;
832               elsif DWNCTF_OUT='1' then
833                  temp(cunt-1 downto cunt-5) := "11111";
834                  CUNT :=CUNT-5;
835                  state_t <= s9;
836               END IF;
837
838               when S11 =>
839               if UPCTF_OUT ='1' then
840                  CUNT :=CUNT+5;
841                  temp(cunt-1 downto cunt-5) := '0' & SW3 & SW2& SW1 & SW0;
842                  state_t <= s12;
843               elsif DWNCTF_OUT='1' then
844                  temp(cunt-1 downto cunt-5) := "11111";
845                  CUNT :=CUNT-5;
846                  state_t <= s10;
847               END IF;
```

```
848
849                 when S12 =>
850                 if UPCTF_OUT ='1' then
851                    CUNT :=CUNT+5;
852                    temp(cunt-1 downto cunt-5) := '0' & SW3 & SW2& SW1 & SW0;
853                    state_t <= s13;
854                 elsif DWNCTF_OUT='1' then
855                    temp(cunt-1 downto cunt-5) := "11111";
856                    CUNT :=CUNT-5;
857                    state_t <= s11;
858                 END IF;
859
860                 when S13 =>
861                 if UPCTF_OUT ='1' then
862                    CUNT :=CUNT+5;
863                    temp(cunt-1 downto cunt-5) := '0' & SW3 & SW2& SW1 & SW0;
864                    state_t <= s14;
865                 elsif DWNCTF_OUT='1' then
866                    temp(cunt-1 downto cunt-5) := "11111";
867                    CUNT :=CUNT-5;
868                    state_t <= s12;
869                 END IF;
870
871                 when S14 =>
872                 if UPCTF_OUT ='1' then
873                    CUNT :=CUNT+5;
874                    temp(cunt-1 downto cunt-5) := '0' & SW3 & SW2& SW1 & SW0;
875                    state_t <= s15;
876                 elsif DWNCTF_OUT='1' then
877                    temp(cunt-1 downto cunt-5) := "11111";
878                    CUNT :=CUNT-5;
879                    state_t <= s13;
880                 END IF;
881
882                 when S15 =>
883                 if UPCTF_OUT ='1' then
884                    CUNT :=CUNT+5;
885                    temp(cunt-1 downto cunt-5) := '0' & SW3 & SW2& SW1 & SW0;
886                    state_t <= s16;
887                 elsif DWNCTF_OUT='1' then
888                    temp(cunt-1 downto cunt-5) := "11111";
889                    CUNT :=CUNT-5;
890                    state_t <= s14;
891                 END IF;
892
893
894                 when S16 =>
895                 if UPCTF_OUT ='1' then
896                    state_t <= s0;
897                    temp :=(others =>'1');
898                    cunt :=0;
899                 elsif DWNCTF_OUT='1' then
900                    temp(cunt-1 downto cunt-5) := "11111";
901                    CUNT :=CUNT-5;
902                    state_t <= s15;
903                 END IF;
904
```

```
905             end case;
906          end if;
907       end if;
908       OUT_DATA <=temp;
909       end process;
910       -------------------------------------------------------
911       -- PULSE GENERATOR
912       PROCESS(fast_clk,SIG_IN)
913       VARIABLE SIGNF_PAST: STD_LOGIC;
914       BEGIN
915          If fast_clk'event and fast_clk = '1' then
916             IF SIGNF_PAST= '0' AND SIG_IN= '1' THEN
917                SIGNF_OUT<='1';
918                SIGNF_PAST:='1';
919             ELSIF SIGNF_PAST = '1' AND SIG_IN='1' THEN
920                SIGNF_OUT<='0';
921                SIGNF_PAST:='1';
922             ELSIF SIG_IN= '0' THEN
923                SIGNF_OUT<= '0';
924                SIGNF_PAST:='0';
925             END IF;
926          END IF;
927       END PROCESS;
928       -------------------------------------------------------
929       -- ADDITION / SUBSTRACTION SIGNAL
930       process (RESET,fast_clk,SIGNF_OUT) is
931       variable temp : std_logic:='0';
932       begin
933       if rising_edge(fast_clk) then
934       if RESET ='1' then
935          LD1 <='0';
936       else
937          if SIGNF_OUT ='1' then
938             SIG_OUT <= temp;
939             LD1 <=temp;
940             temp :=not temp;
941          end if;
942       END IF;
943       end if;
944       end process;
945       ---------------------------------------
946       COMP1 : BUFFER_INOUT port map(RESET,fast_clk,UP_CNTOUT,DO_CNTOUT,OUT_DATA,NUMBR1,
          NUMBR2,BUFFOUT);
947       COMP2 : FLPOINT port map(BEG_CL,SIG_OUT,RESET,NUMBR1,NUMBR2,MZERO_OUT,PZERO_OUT,
          PINF_OUT,MINF_OUT,NAN_OUT,CLR_INOUT,UNORMZ_OUT,RES);
948       COMP3 : PADD_ZERO port map(RES,PZERO_OUT,MZERO_OUT,PINF_OUT,MINF_OUT,NAN_OUT,CLR_INOUT
          ,UNORMZ_OUT,PADZRO);
949       COMP4 : MUX31 port map(fast_clk,SW4,SW5,SW6,BUFFOUT(39 downto 0),BUFFOUT(79 downto 40
          ),PADZRO,SEL_SIG);
950       COMP5 : MUX21 port map(SEL_SIG,SW7,MX_SEG);
951       COMP6 : BCD_SEG port map(MX_SEG,fast_clk,SEV_SEG,AN);
952
953       end Behavioral;
954
```