```
TITLE "BSCTOF trigger";

%     carla grosso-pilcher                                                  %
%      BSC, RP and MP trigger bits                                          %
%     January 2002 - add TOF trigger                                        %
%     MAY 2002 - delay BSC and Mp with respect to RP                        %
%     July 2002 - delay only BSC connector 0, add 4 cdf clk to fred delay %
%                 add cosmic trigger                                        %

% !!!!    Pin names and assignment fixed from the sumet chip.  !!!!%
% !!!!    They have to be conserved on all pfred chip, even if not optimal.  !!!!%

FUNCTION lpm_shiftreg (data[LPM_WIDTH-1..0], clock, enable, shiftin, load, sclr, sset, aclr,
aset)
   WITH (LPM_WIDTH, LPM_DIRECTION, LPM_AVALUE, LPM_SVALUE)
   RETURNS (Q[LPM_WIDTH-1..0], shiftout);

FUNCTION lpm_mux (data[LPM_SIZE-1..0][LPM_WIDTH-1..0], sel[LPM_WIDTHS-1..0], clock, aclr)
   WITH (LPM_WIDTH, LPM_SIZE, LPM_WIDTHS, LPM_PIPELINE)
   RETURNS (result[LPM_WIDTH-1..0]);

FUNCTION lpm_add_sub (cin, dataa[LPM_WIDTH-1..0], datab[LPM_WIDTH-1..0], add_sub, clock,
aclr)
   WITH (LPM_WIDTH, LPM_REPRESENTATION, LPM_DIRECTION, LPM_PIPELINE, ONE_INPUT_IS_CONSTANT)
   RETURNS (result[LPM_WIDTH-1..0], cout, overflow);

FUNCTION lpm_compare (dataa[LPM_WIDTH-1..0], datab[LPM_WIDTH-1..0], clock, aclr)
   WITH (LPM_WIDTH, LPM_REPRESENTATION, LPM_PIPELINE, CHAIN_SIZE, ONE_INPUT_IS_CONSTANT)
   RETURNS (alb, aeb, agb, ageb, aneb, aleb);



SUBDESIGN bsctof
(

    clk_66ns          : INPUT;
    clk_132ns[2..0]   : INPUT;  % 132 ns clocks, separated by 22 ns; 1st one %
                % is in sync with CRATESUMs' incoming data %
    b0_delayed        : INPUT;  % B0 signal delayed by b0_offset %

    run               : INPUT;

    vme_write         : INPUT;  % read (0) or write (1) request from VME %
    vme_dat[31..0]    : BIDIR;  % port used to download thresholds and input mask %
    vme_address[13..2]: INPUT;  % VME address bus %
    vme_proc_clk      : INPUT;  % clock to load thresholds and masks from VME %
    vme_fifo_wen      : INPUT;
    vme_version_ren   : INPUT;  % enables vme reading of the code version number %

    fred_delay[5..0]  : INPUT;  % alignment delay (constant) in units of 22 ns %

    thres_sel[1..0]   : INPUT;  % threshold selector for TOF counts, %
                                % also used to define mask for bad channels %

    src_etin          : INPUT;  % What feeds this chip:                       %
                                % 0=backplane (normal run), 1=VME_address bus %
    etin[11..0][9..0] : INPUT;  % input trigger bits - 12 x 10 signals from aux card%
```

```
                                % only 2 sets of 10 bit inputs used at the time, %
                                %  with option to select between two sets : %
                                %  either input 0 and 2 or 1 and 3%
    dataout_ren[1..0] : INPUT;  % enables vme reading of dataout, i.e. trigword &   %
                                %     to_fred %
    dataout[55..0]    : OUTPUT;

    sram_enable       : OUTPUT;

%**** the following signals are not used in this module, defined as input to keep pin
assignment *****%
    lut_address[5..0][11..0] : INPUT;  % output in sumet %
    lut_data[5..0][11..0]    : INPUT; % bidir in sumet %
    vme_lut_en[2..0]         : INPUT;

)

VARIABLE

    _clk_132ns[2..0]  : GLOBAL;
    load              : NODE;
    vme_wen           : NODE; % vme_write & load %
    vme_read          : NODE;
    run_              : NODE; % run # src_etin %

    vme_data[31..0]        : TRI_STATE_NODE;
    fake_etin[11..0][9..0] : TRI;
%    fake_b0delayed         : TRI;%
    datain[11..0][9..0]    : TRI_STATE_NODE;
    etin_[11..0][9..0]     : DFFE;  % trigger bit from BSC and TOF systems  %
    mask_et[11..0][9..0]   : DFFE;  % mask for bad etin data. %

    tri_mask_et[11..0]     : TRI;
    tri_thres[1..0][31..0] : TRI;
    tri_b0delayed          : TRI;
    tri_dataout[55..15]    : TRI;
    tri_etin[11..0][9..0]  : TRI;
    tri_trigbits[39..0]    : TRI; % 19..0 are b0_delayed, 39..20 are the original ones %
    tri_version[7..0]      : TRI;
    tri_dummy[2..0]        : TRI;

    datout[55..15]    : TRI_STATE_NODE;
    outpipe[20..0]    : lpm_shiftreg WITH (LPM_WIDTH=16); % FRED alignment pipeline %
    align_mux         : lpm_mux WITH (LPM_WIDTH=21, LPM_SIZE=16, LPM_WIDTHS=6); % picks the
output with desired align delay %


    select_clk        : lpm_mux WITH (LPM_WIDTH=1, LPM_SIZE=6, LPM_WIDTHS=3);
    tofred[20..0]     : DFFE;  %20 trigger bits and b0delayed %
    trgbts[19..0]     : DFFE;  % trigger bits before Fred delay%
    version[7..0]     : NODE;
    toftrg[1..0]      : NODE;
    bsctrg[19..0]     : NODE;
    bsc_tmp[3..0]     : DFFE;    % intermediate node to delay BSC signals%
    bsc1_tmp[3..0]    : DFFE;    % intermediate node to delay BSC signals%
    bsc2_tmp[3..0]    : DFFE;    % intermediate node to delay BSC signals%
    mp_tmp[1..0][9..0]    : DFFE;    % intermediate node to delay mp signals%
```

```
    mp1_tmp[1..0][9..0]   : DFFE;    % intermediate node to delay mp signals%
    mp2_tmp[1..0][9..0]   : DFFE;    % intermediate node to delay mp signals%

    delay[2..0]       : NODE;    % extra delay to FRED %


    thres[1..0][31..0] : DFFE;   % thresholds for time TOF multiplicity trigger %

    tofin[3..0][4..0] : NODE;
    adder_a[1..0]     : lpm_add_sub WITH
(LPM_WIDTH=6,LPM_REPRESENTATION="UNSIGNED",LPM_DIRECTION="ADD");
    adder_b           : lpm_add_sub WITH
(LPM_WIDTH=7,LPM_REPRESENTATION="UNSIGNED",LPM_DIRECTION="ADD");
    bitsum[6..0]      : NODE;    % total sum of TOF bits %
    comp[1..0]        : lpm_compare WITH
(LPM_WIDTH=7,LPM_PIPELINE=0,ONE_INPUT_IS_CONSTANT="YES");
    final_delay       : lpm_add_sub WITH
(LPM_WIDTH=4,LPM_REPRESENTATION="UNSIGNED",LPM_DIRECTION="ADD");

    %   for comics trigger %
    a0                : NODE;
    a1                : NODE;
    a2                : NODE;
    a3                : NODE;
    a4                : NODE;
    a5                : NODE;
    b0                : NODE;
    b1                : NODE;
    b2                : NODE;
    b3                : NODE;
    b4                : NODE;
    b5                : NODE;


BEGIN

     _clk_132ns[]   = !clk_132ns[];
    etin_[][].clk   = _clk_132ns0;
    bsc_tmp[].clk   = _clk_132ns0;
    bsc1_tmp[].clk  = clk_132ns1;
    bsc2_tmp[].clk  = _clk_132ns0;
    mp_tmp[][].clk      = _clk_132ns0;
    mp1_tmp[][].clk     = clk_132ns1;
    mp2_tmp[][].clk     = _clk_132ns0;
    trgbts[].clk    = clk_132ns1;

    tri_dummy[].in  = GND;
    tri_dummy[].oe  = dataout_ren1;
    vme_data[23..21] = tri_dummy[2..0].out;  % dummy assignment, 23..21 actually unused %
    run_        = run # src_etin;
    etin_[][].ena   = run_;
    mp_tmp[][].ena  = run_;
    mp1_tmp[][].ena     = run_;
    mp2_tmp[][].ena     = run_;
    bsc_tmp[].ena   = run_;
    bsc1_tmp[].ena  = run_;
    bsc2_tmp[].ena  = run_;
```

```
    delay[0] = GND;
    delay[1] = GND;
    delay[2] = VCC;


% I. LOAD mode %

  load          = !run;
  vme_read      = !vme_write;
  vme_wen       = vme_write & load;

  version[]         = 5;
  tri_version[].in  = version[];
  tri_version[].oe  = vme_version_ren;
  vme_data[31..24]  = tri_version[].out;
  thres0_[].ena     = vme_wen & thres_sel0 & !thres_sel1;
  thres1_[].ena     = vme_wen & thres_sel1 & !thres_sel0;
  thres[][].d       = vme_dat[31..0];
  thres[][].clk     = vme_proc_clk;
  mask_et[][].clk   = vme_proc_clk;
  mask_et[][].ena   = vme_wen & thres_sel0 & thres_sel1;
  mask_et[]_0.d     = vme_dat[11..0];
  mask_et[]_1.d     = vme_dat[11..0];
  mask_et[]_2.d     = vme_dat[11..0];
  mask_et[]_3.d     = vme_dat[11..0];
  mask_et[]_4.d     = vme_dat[11..0];
  mask_et[]_5.d     = vme_dat[11..0];
  mask_et[]_6.d     = vme_dat[11..0];
  mask_et[]_7.d     = vme_dat[11..0];
  mask_et[]_8.d     = vme_dat[11..0];
  mask_et[]_9.d     = vme_dat[11..0];
  tri_mask_et[].in  = mask_et[]_0.q;
  tri_mask_et[].oe  = vme_read & thres_sel0 & thres_sel1;
  vme_data[11..0]   = tri_mask_et[].out;
  vme_data[31..0]   = tri_thres0_[].out;
  vme_data[31..0]   = tri_thres1_[].out;
  tri_thres[][]     = thres[][].q;
  tri_thres0_[].oe  = vme_read & thres_sel0 & !thres_sel1;
  tri_thres1_[].oe  = vme_read & thres_sel1 & !thres_sel0;

  tri_trigbits[19..0].in  = tofred[19..0].q;
  tri_trigbits[39..20].in = trgbts[19..0].q;
  tri_trigbits[19..0].oe  = dataout_ren0;
  tri_trigbits[39..20].oe = dataout_ren1;
  vme_data[19..0]     = tri_trigbits[19..0].out;
  vme_data[19..0]     = tri_trigbits[39..20].out;
  tri_b0delayed.in    = tofred20.q;
  tri_b0delayed.oe    = dataout_ren0;
  vme_data20          = tri_b0delayed.out;
  vme_dat[]           = vme_data[];

%  this seems like an overkill, but kept for symmetry with sumet    %
 % fake_trigbits[19..0].in = vme_dat[19..0];
  fake_trigbits[19..0].oe = vme_fifo_wen;
  datout[50..35]    = fake_trigbits[19..4].out;
  datout[54..51]    = fake_trigbits[3..0].out;
```

```
  datout[34..15]    = fake_trigbits[19..0].out;
  fake_b0delayed.in     = vme_dat20;
  fake_b0delayed.oe     = vme_fifo_wen;
  datout55      = fake_b0delayed.out;%

  fake_etin0_[].in      = vme_address[11..2];
  fake_etin1_[].in      = (vme_address11,vme_address2,vme_address[10..3]);
  fake_etin2_[].in      = (vme_address11,vme_address[3..2],vme_address[10..4]);
  fake_etin3_[].in      = (vme_address11,vme_address[4..2],vme_address[10..5]);
  fake_etin4_[].in      = (vme_address11,vme_address[5..2],vme_address[10..6]);
  fake_etin5_[].in      = (vme_address11,vme_address[6..2],vme_address[10..7]);
  fake_etin6_[].in      = (vme_address11,vme_address[7..2],vme_address[10..8]);
  fake_etin7_[].in      = (vme_address11,vme_address[8..2],vme_address[10..9]);
  fake_etin8_[].in      = (vme_address11,vme_address[9..2],vme_address10);
  fake_etin9_[].in      = vme_address[11..2];
  fake_etin10_[].in     = (vme_address11,vme_address2,vme_address[10..3]);
  fake_etin11_[].in     = (vme_address11,vme_address[3..2],vme_address[10..4]);
  fake_etin[][].oe      = src_etin;
  datain[][]            = fake_etin[][].out;


% II. RUN mode:  %

  tri_etin[][].in   = etin[][];
  tri_etin[][].oe   = run;
  datain[][]        = tri_etin[][].out;
  etin_[][].d       = datain[][] & !mask_et[][];

%  OR bits form input 0 and 2, 1 and 3 %
% Connectors scheme for BSC/MP/RP
etin_0_[0] -    BSC-1 E
etin_0_[1] -    BSC-1 W
etin_0_[2] -    BSC-2 E
etin_0_[3] -    BSC-2 W
etin_1_[0] -    BSC-3 E
etin_1_[1] -    BSC-3 W
etin_1_[2] -    BSC-4 W
etin_1_[3] -    RP-1
etin_1_[4] -    RP-2
etin_1_[5] -    RP-3
etin_2_[0] -    MP East - Tower 1 HI (6.7.12.13)
etin_2_[1] -    MP East - Tower 1 LO (6.7.12.13)
etin_2_[2] -    MP East - Tower 2 HI (8.9.14.15)
etin_2_[3] -    MP East - Tower 2 LO (8.9.14.15)
etin_2_[4] -    MP East - Tower 3 HI (10.11.16.17)
etin_2_[5] -    MP East - Tower 3 LO (10.11.16.17)
etin_2_[6] -    MP East - Tower 4 HI (0.1.2)
etin_2_[7] -    MP East - Tower 4 LO (0.1.2)
etin_2_[8] -    MP East - Tower 5 HI (3.4.5)
etin_2_[9] -    MP East - Tower 5 LO (3.4.5)
etin_3_[0] -    MP West - Tower 1 HI (6.7.12.13)
etin_3_[1] -    MP West - Tower 1 LO (6.7.12.13)
etin_3_[2] -    MP West - Tower 2 HI (8.9.14.15)
etin_3_[3] -    MP West - Tower 2 LO (8.9.14.15)
etin_3_[4] -    MP West - Tower 3 HI (10.11.16.17)
etin_3_[5] -    MP West - Tower 3 LO (10.11.16.17)
etin_3_[6] -    MP West - Tower 4 HI (0.1.2)
```

```
etin_3_[7] -    MP West - Tower 4 LO (0.1.2)
etin_3_[8] -    MP West - Tower 5 HI (3.4.5)
etin_3_[9] -    MP West - Tower 5 LO (3.4.5)%

%    Unused triggr bits   %
  bsctrg[8]         = GND;
  bsctrg[9]         = GND;
  bsctrg[13]        = GND;
  bsctrg[14]        = GND;
  bsctrg[15]        = GND;
  bsctrg[16]        = GND;
  bsctrg[17]        = GND;
  bsctrg[18]        = GND;
  bsctrg[19]        = GND;

%   Delay BSC (0) and MP input signals   %
  bsc_tmp[3..0].d  = etin_[0][3..0].q;
  bsc1_tmp[].d     = bsc_tmp[].q;
  bsc2_tmp[].d     = bsc1_tmp[].q;
  mp_tmp[0][].d    = etin_[2][9..0].q;
  mp_tmp[1][].d    = etin_[3][9..0].q;
  mp1_tmp[][].d    = mp_tmp[][].q;
  mp2_tmp[][].d    = mp1_tmp[][].q;

%  Forward detectors Trigger definition%
% GAP WEST%
  bsctrg[0]  = !bsc2_tmp[1].q & !bsc2_tmp[3].q & !etin_[1][1].q & ! etin_[1][2].q;
% GAP EAST%
  bsctrg[1]  = !bsc2_tmp[0].q & !bsc2_tmp[2].q & !etin_[1][0].q;
% MB_BSC1%
  bsctrg[2]  = bsc2_tmp[0].q & bsc2_tmp[1].q;
% MB_RP%
  bsctrg[3]  = etin_[1][3].q & etin_[1][4].q & etin_[1][5].q;
% MP-E-ET%
  bsctrg[4]  = mp2_tmp[0][0].q # mp2_tmp[0][2].q # mp2_tmp[0][4].q # mp2_tmp[0][6].q #
mp2_tmp[0][8].q;
% MP-E-GP%
  bsctrg[5]  = !mp2_tmp[0][7].q & !mp2_tmp[0][9].q;
% MP-W-ET%
  bsctrg[6]  = mp2_tmp[1][0].q # mp2_tmp[1][2].q # mp2_tmp[1][4].q # mp2_tmp[1][6].q #
mp2_tmp[1][8].q;
% MP-W-GP%
  bsctrg[7]  = !mp2_tmp[1][7].q & !mp2_tmp[1][9].q;

%  TOF triggers  %
%  use inputs 8-11 : 5 signals/connector  %
%
    tofin[0][4..0] = etin_[8][4..0];
    tofin[1][4..0] = etin_[9][4..0];
    tofin[2][4..0] = etin_[10][4..0];
    tofin[3][4..0] = etin_[11][4..0];
%
    adder_a0.dataa[] = (GND,etin_[8][4..0].q);
    adder_a0.datab[] = (GND,etin_[9][4..0].q);
    adder_a1.dataa[] = (GND,etin_[10][4..0].q);
    adder_a1.datab[] = (GND,etin_[11][4..0].q);
    adder_b.dataa[] =  (GND,adder_a0.result[]);
```

```
    adder_b.datab[] =  (GND,adder_a1.result[]);
    bitsum[]        = adder_b.result[];

    comp0.dataa[] = adder_b.result[];
    comp0.datab[] = thres0_[6..0].q;
    comp1.dataa[] = adder_b.result[];
    comp1.datab[] = thres0_[22..16].q;

    toftrg[1..0]  = comp[1..0].ageb;
    bsctrg[11..10] = toftrg[1..0];

    %   cosmics %
    a0     = etin_[6][2].q # etin_[6][3].q # etin_[6][4].q;
    a1     = etin_[6][3].q # etin_[6][4].q # etin_[6][5].q;
    a2     = etin_[6][4].q # etin_[6][5].q # etin_[6][0].q;
    a3     = etin_[7][5].q # etin_[7][0].q # etin_[7][1].q;
    a4     = etin_[7][0].q # etin_[7][1].q # etin_[7][2].q;
    a5     = etin_[7][1].q # etin_[7][2].q # etin_[7][3].q;
    b0     = etin_[4][3].q & a0;
    b1     = etin_[4][4].q & a1;
    b2     = etin_[4][5].q & a2;
    b3     = etin_[5][0].q & a3;
    b4     = etin_[5][1].q & a4;
    b5     = etin_[5][2].q & a5;

    bsctrg[12] = b0 # b1 # b2 # b3 # b4 # b5;

  trgbts[19..0].d  = bsctrg[19..0];

  outpipe[19..0].shiftin  = trgbts[].q;
  outpipe20.shiftin   = b0_delayed;
  outpipe[].clock     = _clk_132ns0;
  align_mux.data0_[]  = outpipe[].q0;
  align_mux.data1_[]  = outpipe[].q1;
  align_mux.data2_[]  = outpipe[].q2;
  align_mux.data3_[]  = outpipe[].q3;
  align_mux.data4_[]  = outpipe[].q4;
  align_mux.data5_[]  = outpipe[].q5;
  align_mux.data6_[]  = outpipe[].q6;
  align_mux.data7_[]  = outpipe[].q7;
  align_mux.data8_[]  = outpipe[].q8;
  align_mux.data9_[]  = outpipe[].q9;
  align_mux.data10_[]  = outpipe[].q10;
  align_mux.data11_[]  = outpipe[].q11;
  align_mux.data12_[]  = outpipe[].q12;
  align_mux.data13_[]  = outpipe[].q13;
  align_mux.data14_[]  = outpipe[].q14;
  align_mux.data15_[]  = outpipe[].q15;

  final_delay.dataa[2..0]  = fred_delay[5..3];
  final_delay.datab[2..0]  = delay[];
  align_mux.sel[3..0]     = final_delay.result[];
%  align_mux.sel[2..0]     = fred_delay[5..3];

  align_mux.sel[5..3]     = delay[2..0];
 %
  select_clk.data0_0  = _clk_132ns1;
```

```
  select_clk.data1_0  = _clk_132ns2;
  select_clk.data2_0  = clk_132ns0;
  select_clk.data3_0  = clk_132ns1;
  select_clk.data4_0  = clk_132ns2;
  select_clk.data5_0  = _clk_132ns0;
  select_clk.sel[]    = fred_delay[2..0]; % fine delay : selects which clock the trigger
bits will be sent out to FRED on %
  tofred[].clk        = select_clk.result0;
  tofred[].d          = align_mux.result[];

  tri_dataout[34..15].in = trgbts[].q;       % these bits go to the L2 buffers and front
panel %
  tri_dataout[54..51].in = tofred[3..0].q;   %  to allign bits properly on the back plane,
need to follow pfred-sumet assignements%
  tri_dataout[50..35].in = tofred[19..4].q;
  tri_dataout[55].in     = tofred[20].q;
  tri_dataout[].oe       = run;

  datout[]               = tri_dataout[].out;


  dataout[19..0]         = datout[34..15];
                           % lowest 10 trigger bits will correspond to first 10 bits of
sumet%
                           % highest 10 trigger bits will correspond to bit 10 of sumet and
bits 9-0 of sumex%
  dataout[55..35]        = datout[55..35];
  dataout[34..20]        = GND;
  sram_enable = VCC;

  END;
```