HISTO VHD

Hrdwre

What does pn mean!!!

pn ?
provided number??

pseudo noise?
pixel-number??

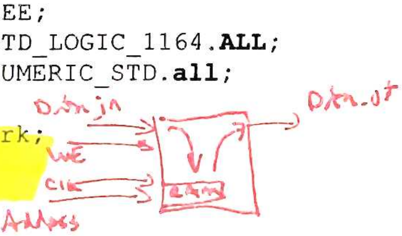
.WE = WRITE ENABLE

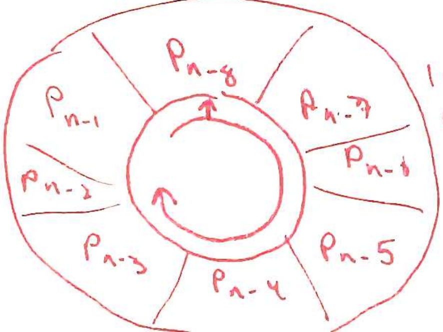
```vhdl
----------------------------------------------------------------------------------
-- Company:
-- Engineer: Professor Jim Plusquellic
--
-- Create Date:
-- Design Name:
-- Module Name:    Histo - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------

-- ==================================================================================================
-- ==================================================================================================

-- Histo bins and counts the integer portion of the PNs as a means of determining the width of the
-- distribution. We use a 'bounded range' method to measure the width of the distribution, e.g., 6.25%
-- to 93.75% to approx. the 3 sigma actual range to avoid the negative impact caused by an outlying
-- PNs.

-- The PNs are signed values between -2048.9375 and 2047.9375 (SIGNED 12-bit integer + 4 bits of precision)
-- We have 2K words of PNL BRAM to utilize (addresses 2048 to 4095) for the histogram so we'll need to make
-- sure the integer PNs do not exceed -1024 to +1023. To maximize the ranges that can be handled and to
-- deal with the negative PNs (if they occur), we first find the smallest value and I use that to offset
-- the distribution. The range is a relative value (difference of two addresses in the histogram portion of
-- the PNL_BRAM) that represent the 6.25% to 93.75% bounds so subtracting the two addresses gives us the
-- range of the PNs.

-- Each cell is 16 bits so we can count to 2^16 (plenty given we only have 4096 total values to bin). Once
-- we've created the histogram, we parse it from left to right, adding each of the cell counts to a global sum,
-- and record the point where the sum equals or exceeds the LV bound count. This address becomes the lower bound
-- on the range. We keep parsing until the sum becomes >= the HV bound count. This address becomes the upper
-- bound on the range.

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.all;

library work;
```

2k

A

B





1 clk cycle

A: 1st, read P_{n-8}

B: 2nd, write to P_{n-8}

```vhdl
use work.DataTypes_pkg.all;

entity Histo is
   port(
      Clk: in  std_logic;
      RESET: in std_logic;
      start: in std_logic;
      ready: out std_logic;
      ~~HISTO_ERR: out std_logic;~~
      PNL_BRAM_addr: out std_logic_vector(PNL_BRAM_ADDR_SIZE_NB-1 downto 0);
      PNL_BRAM_din: out std_logic_vector(PNL_BRAM_DBITS_WIDTH_NB-1 downto 0);
      PNL_BRAM_dout: in std_logic_vector(PNL_BRAM_DBITS_WIDTH_NB-1 downto 0);
      PNL_BRAM_we: out std_logic_vector(0 to 0)
      );
end Histo;
```

?. Need 2 register to differ ??

All caps, constants

These vary

add state get_diffs.

```vhdl
architecture beh of Histo is
   type state_type is (idle, clear_mem, find_smallest, compute_addr, inc_cell,
   get_next_PN, init_dist, sweep_BRAM,
      check_histo_error, write_range);
   signal state_reg, state_next: state_type;

   signal ready_reg, ready_next: std_logic;

-- Address registers for the PNs and histogram portions of memory
   signal PN_addr_reg, PN_addr_next: unsigned(PNL_BRAM_ADDR_SIZE_NB-1 downto 0);
   signal histo_addr_reg, histo_addr_next: unsigned(PNL_BRAM_ADDR_SIZE_NB-1 downto
    0);

-- For selecting between PN or histo portion of memory during memory accesses
   signal do_PN_histo_addr: std_logic;

-- Stores the full 16-bit PN that is the smallest among all in the data set
   signal smallest_val_reg, smallest_val_next: signed(PNL_BRAM_DBITS_WIDTH_NB-1
   downto 0);

-- These are 12 bits each to hold only the 12-bit integer portion of the PNs
   signal shifted_dout: signed(PN_INTEGER_NB-1 downto 0);
   signal shifted_smallest_val: signed(PN_INTEGER_NB-1 downto 0);
```

integer smallest valu.

```vhdl
-- These signals used in the calculation of the address in the histogram memory
of the cell to add 1 to during the histo
-- contruction. They are addresses and therefore need to match the address width
of the memory.
   signal offset_addr: signed(PNL_BRAM_ADDR_SIZE_NB-1 downto 0);
   signal histo_cell_addr: unsigned(PNL_BRAM_ADDR_SIZE_NB-1 downto 0);

-- These variables will store the PNL_BRAM addresses when the LV and HV bounds
are met.
```

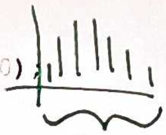
empty ints of same size pn

```vhdl
   signal LV_addr_reg, LV_addr_next: unsigned(PNL_BRAM_ADDR_SIZE_NB-1 downto 0);
   signal HV_addr_reg, HV_addr_next: unsigned(PNL_BRAM_ADDR_SIZE_NB-1 downto 0);
```

locating in PN of HV & LV

```vhdl
-- Use for error checking when the range of the distribution is computed.
   signal LV_set_reg, LV_set_next: std_logic;
   signal HV_set_reg, HV_set_next: std_logic;

-- These signals store the lower and upper count that represents the fractional
limits of the histogram. They are constants
-- that signal the state machine when to record an address reference in the
histogram memory. These hold constants that
-- represent a 'count', where the maximum value can be 4096 (one bigger than
4095), so we need 13 bits (not 12).
   signal LV_bound, HV_bound: unsigned(NUM_PNS_NB downto 0);
```

```
-- The register used to sum up the counts in the histogram as it is parsed left to right. It WILL count up to the number of
-- PNs stored, which is currently 4096, so we need 13-bit here, not 12.
   signal dist_cnt_sum_reg, dist_cnt_sum_next: unsigned(NUM_PNS_NB downto 0);

-- Storage for the mean must be able to accommodate a sum of 4096 values (NUM_PNS) each of which is 16-bits (PNL_BRAM_DBITS_WIDTH_NB)
-- wide. The number of values summed is 4096 so we need 12-bits, NUM_PNS_NB) where each value is 16-bits (PN_SIZE_NB) so we need
-- an adder that is 28 bits (27 downto 0). The sum is likely to require much fewer bits -- this is worst case.
   signal dist_mean_sum_reg, dist_mean_sum_next: signed(NUM_PNS_NB+PN_SIZE_NB-1 downto 0);
```



Σ of all counts must equal total ~~value~~ of # elements in PNL

mean, avg would be pos, or it would be neg. so used signed

will not need so many bits for HISTO2

```
-- The final mean and range ~~computed~~ from the histogram. Written to memory ~~below~~.
   signal dist_mean: std_logic_vector(PNL_BRAM_DBITS_WIDTH_NB-1 downto 0);
   signal dist_range: std_logic_vector(HISTO_MAX_RANGE_NB-1 downto 0);

-- Error flag is set to '1' if distribution is too narrow to be characterized with the specified bounds, or the integer portion of
-- a PN value is outside the range of -1023 and 1024.
   signal HISTO_ERR_reg, HISTO_ERR_next: std_logic;
```

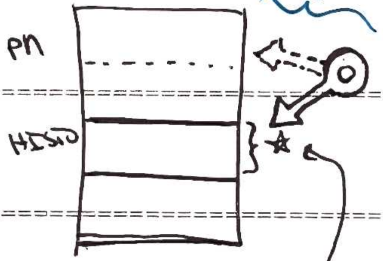
No change

```
   begin

-- Compute the mean with full precision. Divide through by 2^12 or 4096 since that's the number of PNs we add to the sum.
   dist_mean <= std_logic_vector(resize(dist_mean_sum_reg/(2**NUM_PNS_NB), PNL_BRAM_DBITS_WIDTH_NB));
```

this will be differen in Histo2

```
-- The range of the distribution is computed as the difference in the addresses which were set when the running sum of the counts in
-- the histo (as we sweep left to right) became equal to the percentages we defined as the limits, e.g., 6.25% and 93.75%.
-- NOTE: 'HISTO_MAX_RANGE_NB" is 12 because the number of memory elements allocated for histo memory is 2^12 = 2048, so 12 bits are
-- needed to allow the range to reach 2048 (one bigger than 2047, which is 2^11).
   dist_range <= std_logic_vector(resize(HV_addr_reg - LV_addr_reg + 1, HISTO_MAX_RANGE_NB));
```

Have to see-saw between PNL & Histo data spaces



```
-- ==========================================================================================
-- State and register logic
-- ==========================================================================================
   process(Clk, RESET)
      begin
      if ( RESET = '1' ) then
         state_reg <= idle;
         ready_reg <= '1';
         PN_addr_reg <= (others => '0');
         histo_addr_reg <= (others => '0');
         smallest_val_reg <= (others => '0');
         LV_addr_reg <= (others => '0');
         HV_addr_reg <= (others => '0');
         LV_set_reg <= '0';
         HV_set_reg <= '0';
         dist_cnt_sum_reg <= (others => '0');
         dist_mean_sum_reg <= (others => '0');
         HISTO_ERR_reg <= '0';
      elsif ( Clk'event and Clk = '1' ) then
```

★ HISO_BRA is not a operation memory, it is the storage location for the resulting histogram!!! will be ½ size of whatever data set?

```
         state_reg <= state_next;
         ready_reg <= ready_next;
         PN_addr_reg <= PN_addr_next;
         histo_addr_reg <= histo_addr_next;
         smallest_val_reg <= smallest_val_next;
         LV_addr_reg <= LV_addr_next;
         HV_addr_reg <= HV_addr_next;
         LV_set_reg <= LV_set_next;
         HV_set_reg <= HV_set_next;
         dist_cnt_sum_reg <= dist_cnt_sum_next;
         dist_mean_sum_reg <= dist_mean_sum_next;
         HISTO_ERR_reg <= HISTO_ERR_next;
      end if;
   end process;


-- Convert the two quantities that will participate in computing the address of
appropriate distribution cell that we will
-- add 1 to to create the histogram. these trim off the low order 4 bits of
precision of the current word on the output
-- of the BRAM and the smallest_val computed in the loop below. NOTE: the RANGE
MUST NEVER EXCEED +/- 1023 since we have
-- ONLY 2048 memory locations dedicated to the distribution.
   shifted_dout <= resize(signed(PNL_BRAM_dout)/16, PN_INTEGER_NB);
   shifted_smallest_val <= resize(smallest_val_reg/16, PN_INTEGER_NB);

-- Compute the offset address in the histo portion of memory by taking the
integer portion of 'dout' - the integer portion
-- of the smallest value among all PNs. This address MUST fall into the range 0
to 2047.
   offset_addr <= resize(shifted_dout, PNL_BRAM_ADDR_SIZE_NB) - resize(
   shifted_smallest_val, PNL_BRAM_ADDR_SIZE_NB);

-- Add the offset computed above to the base address of the histogram portion of
BRAM.
   histo_cell_addr <= unsigned(offset_addr) + to_unsigned(HISTO_BRAM_BASE,  ] Ok, keep
   PNL_BRAM_ADDR_SIZE_NB);

-- Compute the bounds of the distribution by adding up histo cells from left to
right until the sum becomes larger/smaller than
-- a 'fraction' of the total number of values counted in the histogram (which is
4096). Use 4 here to set the fraction limits to
-- 6.25% and 93.75% for the LV and HV bounds. With a total count across histo
cells of 4096, the bounds become 256 and 3840.
   LV_bound <= to_unsigned(NUM_PNs, NUM_PNS_NB+1) srl HISTO_BOUND_PCT_SHIFT_NB;
   HV_bound <= to_unsigned(NUM_PNs, NUM_PNS_NB+1) - LV_bound;

--    Converting to unsigned before arithmetic ☑
==============================================================================
==========
-- Combo logic
--
==============================================================================
==========

   process (①state_reg, ②start, ③ready_reg, ④PN_addr_reg, ⑤histo_addr_reg,
   ⑥PNL_BRAM_dout, ⑦histo_cell_addr,
      ⑧LV_bound, ⑨HV_bound, ⑩LV_addr_reg, ⑪HV_addr_reg, ⑫LV_set_reg, ⑬HV_set_reg,
      dist_cnt_sum_reg,
      ⑭dist_cnt_sum_next, dist_mean_sum_reg, smallest_val_reg, dist_mean,
      dist_range, HISTO_ERR_reg)
      begin
      ①state_next <= state_reg;
      ready_next <= ready_reg;③
      PN_addr_next <= PN_addr_reg;
      histo_addr_next <= histo_addr_reg;
      smallest_val_next <= smallest_val_reg;
      LV_addr_next <= LV_addr_reg;
      HV_addr_next <= HV_addr_reg;
      LV_set_next <= LV_set_reg;
      HV_set_next <= HV_set_reg;
      dist_cnt_sum_next <= dist_cnt_sum_reg;
      dist_mean_sum_next <= dist_mean_sum_reg;
      HISTO_ERR_next <= HISTO_ERR_reg;

-- Default value is 0 -- used during memory initialization.
      PNL_BRAM_din <= (others=>'0');
      PNL_BRAM_we <= "0";

      do_PN_histo_addr <= '0';

   case state_reg is

-- =====================
   when idle =>
         ready_next <= '1';

         if ( start = '1' ) then
            ready_next <= '0';

-- Reset error flag
            HISTO_ERR_next <= '0';

-- Zero the register that will eventually define the mean.
            dist_mean_sum_next <= (others=>'0');

-- Allow histo_addr to drive PNL_BRAM
            do_PN_histo_addr <= '1';

-- Assert 'we' to zero out the first cell at 2048.
            PNL_BRAM_we <= "1";
            histo_addr_next <= to_unsigned(HISTO_BRAM_BASE,
            PNL_BRAM_ADDR_SIZE_NB);
            state_next <= clear_mem;
         end if;

-- =====================
-- Clear out the histo portion of memory. 'histo_addr_reg' tracks BRAM cells in
(2048 to 4095) portion of memory
   when clear_mem =>

         if ( histo_addr_reg = HISTO_BRAM_UPPER_LIMIT - 1 ) then

-- Reset PN_addr and get first value
            PN_addr_next <= to_unsigned(PN_BRAM_BASE, PNL_BRAM_ADDR_SIZE_NB);
            state_next <= find_smallest;

-- On the first iteration here, we've already initialized the first memory
location in the previous state.
-- Do the second, etc.
         else
            do_PN_histo_addr <= '1';
            PNL_BRAM_we <= "1";
            histo_addr_next <= histo_addr_reg + 1;
         end if;

-- =====================
```

? 3

?

see architecture notes, () states.

In uus8

State 8-620

conditionals: (start)

10 states

conditionals (histo_addr_reg, PN_addr_reg, PNL_BRAM at, smallest_val_reg, PN_addr_reg)

```
-- Find smallest value (this works for signed PN values, e.g., positive or
negative).
      when find_smallest =>

-- Check PN value to see if it is smaller than current. On first iteration,
assign 'dout' to smallest.
         if ( PN_addr_reg = to_unsigned(PN_BRAM_BASE, PNL_BRAM_ADDR_SIZE_NB) )
         then
            smallest_val_next <= signed(PNL_BRAM_dout);
         elsif ( signed(PNL_BRAM_dout) < smallest_val_reg ) then
            smallest_val_next <= signed(PNL_BRAM_dout);
         end if;

-- PN_addr_reg tracks BRAM cells in (4096 to 8191) portion of memory
         if ( PN_addr_reg = PN_UPPER_LIMIT - 1 ) then

-- Reset PN_addr and get first value for histo construction below
            PN_addr_next <= to_unsigned(PN_BRAM_BASE, PNL_BRAM_ADDR_SIZE_NB);
            state_next <= compute_addr;

         else
            PN_addr_next <= PN_addr_reg + 1;
         end if;

-- ====================
-- Start constructing the histogram. PN portion of memory is selected and driving
'dout' since 'do_PN_histo_addr' was set to '0'
-- in previous state.
      when compute_addr =>

-- Force address to histo portion for next write to memory
         do_PN_histo_addr <= '1';

-- histo_cell_addr is computed outside this process. It is the integer portion of
the 'dout' value minus the smallest_val among
-- all PNs. THIS IS ALWAYS an address in the range of 2048 and 4095.
         histo_addr_next <= histo_cell_addr;

-- Error check. Be sure address NEVER exceeds upper limit of histogram memory.
This 'if stmt' ASSUMES histogram is NOT in the
-- upper-most portion of memory (histo_addr_next in this case would wrap back to
0).
         if ( histo_cell_addr > HISTO_BRAM_UPPER_LIMIT - 1 ) then
            HISTO_ERR_next <= '1';
         end if;

-- Add the current PN to a sum for the mean calculation.
         dist_mean_sum_next <= dist_mean_sum_reg + signed(PNL_BRAM_dout);

         state_next <= inc_cell;

-- ====================
-- Add 1 to the memory location addressed by histo_addr_next/reg
      when inc_cell =>

-- Maintain address in histo memory for the write operation
         do_PN_histo_addr <= '1';

-- Add 1 to the cell pointed to by histo_addr and store it back. NOTE: I DO NOT
need to check for OVERFLOW here b/c it is impossible
-- under the current parameters where we have at most 4096 total PN. Each cell is
16-bits so we can count to at least 2^16 = 65,536
-- unsigned so even if the entire distribution appears in one cell, it will not
overflow.
```

3

4

will have to look at this for HISTO 2.

means working in HISTO storage area. if true

if false: working in PN memory

4

conditionals..

histo_cell_addr,

1

+

add (1) to total now so on last entry reg_next is not circled back to beginning,

5

Ø conditionals,

True, working in HISTO region now...

write enable = true

```
            PNL_BRAM_we <= "1";
            PNL_BRAM_din <= std_logic_vector(unsigned(PNL_BRAM_dout) + 1);
            state_next <= get_next_PN;
```

have to conv. to unsigned in order to add '1'.

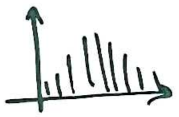
```
-- =====================
-- Allow PN_addr to drive PNL_BRAM with new address, increment address and get next PN value
      6  when get_next_PN =>        1 conditional (PN_addr_reg)

-- Check for exit condition
            if ( PN_addr_reg = PN_UPPER_LIMIT - 1 ) then
               state_next <= init_dist;
            else
               PN_addr_next <= PN_addr_reg + 1;
               state_next <= compute_addr;   repeating state compute addr for next in line.
            end if;

-- =====================
-- With all the counts computed and stored in the histo portion of memory, commense the parse from left to right.
      7  when init_dist =>        0 conditional
```

Histo is built... , now double check & sum up elements for L/R to match PN_BRAM

```
-- Select histo memory
            do_PN_histo_addr <= '1';

-- Re-initialize histo address to first element of distribution. NOTE: The first cell is guaranteed to store at least a count of 1
-- since we offset all integer portions of the PN in the distribution by subtracting the smallest value.
                                          value                  size of field storing
            histo_addr_next <= to_unsigned(HISTO_BRAM_BASE, PNL_BRAM_ADDR_SIZE_NB);
                                                                   value

-- Clear the addresses that will be used to compute the dist_range
        10a LV_addr_next <= (others => '0');   fill w/ zeroes
        11a HV_addr_next <= (others => '0');   fill w/ zeroes

-- Set the 'done' indicators to '0' and sum
        12  LV_set_next <= '0';        flags = 0
        13  HV_set_next <= '0';        flags = 0
        14  dist_cnt_sum_next <= (others => '0');   vector full of zeroes.

            state_next <= sweep_BRAM;
```

Histo reg-addr are cleared & set back to beginning of HISTO storage...

```
-- =====================
-- With all the counts computed and stored in the upper portion of the PNL_BRAM, commense the parse from left to right
-- to determine the range of the distribution.
      8  when sweep_BRAM =>      6 conditionals
```

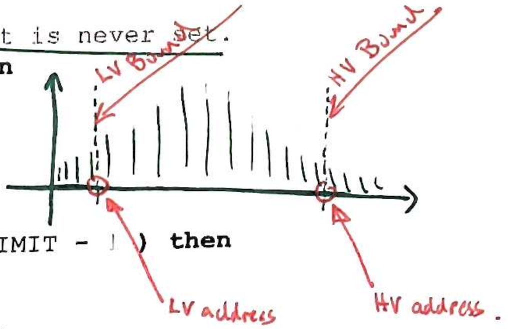
Sweep 

L → → → → → R = Σ elements

```
-- Select histo memory
            do_PN_histo_addr <= '1';
                      (Yes, working in HISTO storage)
-- Add the count in the histo cell to the sum. NOTE: The counts are unsigned values. Note we resize to 13 bit to accomodate
-- the value 4096
                                                                          value
            dist_cnt_sum_next <= dist_cnt_sum_reg + resize(unsigned(PNL_BRAM_dout
), NUM_PNS_NB+1);
            field size of value

-- Assign the LV address just when dist_cnt_sum_next becomes >= than the LV bound. LV_set also used to handle case
-- where the dist_sum never becomes >= LV_bound and is therefore, never assigned (which would happen if ALL the PNs are the
-- same value or span a very small range).
                          1                2         4     3
            if ( LV_set_reg = '0' and dist_cnt_sum_next >= LV_bound ) then
               LV_addr_next <= histo_addr_reg;
      3
```

Tupa'1

```
                  LV_set_next <= '1';
               end if;

      -- Keep assigning HV address while dist_cnt_sum_next is smaller than the bound.
      If it happens that the first address has a
      -- count that's larger than the HV bound, then HV set is never set.
               if ( dist_cnt_sum_next <= HV_bound ) then
                  HV_addr_next <= histo_addr_reg;
                  HV_set_next <= '1';
               end if;

      -- Check if entire distribution has been swept.
               if ( histo_addr_reg = HISTO_BRAM_UPPER_LIMIT - 1 ) then
                  state_next <= check_histo_error;
               else
                  histo_addr_next <= histo_addr_reg + 1;
               end if;
```



```
      -- =====================
      -- Check if the distribution is TOO narrow to be characterized by our bounds. Set
      the error flag if true.
            when check_histo_error =>
               if ( LV_set_reg = '0' or HV_set_reg = '0' ) then
                  HISTO_ERR_next <= '1';
                  state_next <= idle;
```

9 ② Conditionals.

```
      -- Just store the mean and range in the lowest portion of memory (addresses 0 and
      1) for transfer to C program.
               else
                  do_PN_histo_addr <= '1';
                  histo_addr_next <= to_unsigned(HISTO_BRAM_UPPER_LIMIT - 2,
                  PNL_BRAM_ADDR_SIZE_NB);
                  PNL_BRAM_we <= "1";
                  PNL_BRAM_din <= dist_mean;
                  state_next <= write_range;
               end if;

      -- =====================
      -- Write range at address 1
            when write_range =>
               do_PN_histo_addr <= '1';
               histo_addr_next <= to_unsigned(HISTO_BRAM_UPPER_LIMIT - 1,
               PNL_BRAM_ADDR_SIZE_NB);
               PNL_BRAM_we <= "1";
               PNL_BRAM_din <= (PNL_BRAM_DBITS_WIDTH_NB-1 downto HISTO_MAX_RANGE_NB
               => '0') & dist_range;
               state_next <= idle;

         end case;
      end process;

      -- Using _reg here (not the look-ahead _next value).
      with do_PN_histo_addr select
         PNL_BRAM_addr <= std_logic_vector(PN_addr_next) when '0',
                          std_logic_vector(histo_addr_next) when others;

      HISTO_ERR <= HISTO_ERR_reg;
      ready <= ready_reg;

   end beh;
```

Cycle between PN & HISTO portion of addr memory.