```
                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.

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

-- =====================
-- Allow PN_addr to drive PNL_BRAM with new address, increment address and get next PN
value
          when get_next_PN =>

-- 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;
             end if;

-- =====================
-- With all the counts computed and stored in the histo portion of memory, commense the
parse from left to right.
```

pre load values for next clock cycle.

value generated from last clock cycle

STATES = 🟦 BRAM WRITES = 🟩 BRAM READS = 🟥