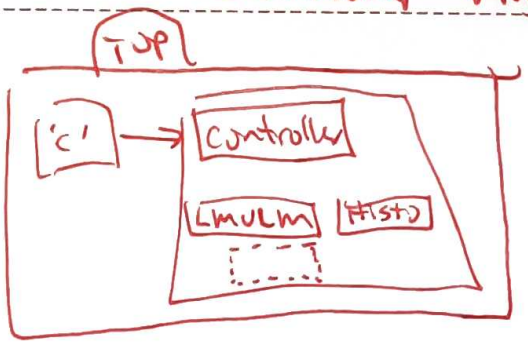
Controller VHD



which program is the data path?

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer: Professor Jim Plusquellic
4  --
5  -- Create Date:
6  -- Design Name:
7  -- Module Name:    Controller - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20
21 -- This is the master control module. It is started by the C program and controls
   the other modules in this project.
22
23
24 library IEEE;
25 use IEEE.STD_LOGIC_1164.ALL;
26 use IEEE.NUMERIC_STD.all;
27
28 library work;
29 use work.DataTypes_pkg.all;
30
31 entity Controller is
32    port(
33       Clk: in std_logic;
34       RESET: in std_logic;
35       start: in std_logic;
36       ready: out std_logic;
37       LM_ULM_start: out std_logic;
38       LM_ULM_ready: in std_logic;
39       LM_ULM_base_address: out std_logic_vector(PNL_BRAM_ADDR_SIZE_NB-1 downto 0);
40       LM_ULM_upper_limit: out std_logic_vector(PNL_BRAM_ADDR_SIZE_NB-1 downto 0);
41       LM_ULM_load_unload: out std_logic;
42       Histo_start: out std_logic;
43       Histo_ready: in std_logic;
44       BRAM_select: out std_logic
45       );
46 end Controller;
47
48
49 architecture beh of Controller is
50    type state_type is (idle, wait_lM_ULM_load, wait_Histo, wait_LM_ULM_unload);
51    signal state_reg, state_next: state_type;
52
53    signal ready_reg, ready_next: std_logic;
54
55    begin
56
57 --
   ================================================================================
   ==========
58 -- State and register logic
59 --
   ================================================================================
   ==========
```

State machine & State registers

load unload

DataTypes_pkg

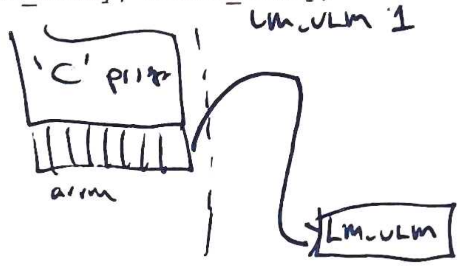
States

S1 S2 S3 S4

current state next state

```vhdl
   process(Clk, RESET)
      begin
      if ( RESET = '1' ) then
         state_reg <= idle;
         ready_reg <= '1';
      elsif ( Clk'event and Clk = '1' ) then
         state_reg <= state_next;
         ready_reg <= ready_next;
      end if;
   end process;

-- ======================================================================================
-- Combo logic
-- ======================================================================================
   process (state_reg, start, ready_reg, LM_ULM_ready, Histo_ready)
      begin
      state_next <= state_reg;
      ready_next <= ready_reg;

      LM_ULM_start <= '0';
      Histo_start <= '0';

      LM_ULM_base_address <= (others=>'0');
      LM_ULM_upper_limit <= (others=>'0');
      LM_ULM_load_unload <= '0';

-- Give LoadUnloadMem default control of the memory
      BRAM_select <= '0';

      case state_reg is

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

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

-- Start data load operation from C program
               LM_ULM_start <= '1';

-- Setup memory base and upper limit for loading of PNs into BRAM. ALWAYS SUBSTRACT 1 from the 'UPPER_LIMIT'
               LM_ULM_base_address <= std_logic_vector(to_unsigned(PN_BRAM_BASE, PNL_BRAM_ADDR_SIZE_NB));
               LM_ULM_upper_limit <= std_logic_vector(to_unsigned(PNL_BRAM_NUM_WORDS_NB - 1, PNL_BRAM_ADDR_SIZE_NB));

               state_next <= wait_LM_ULM_load;
            end if;

-- ====================
-- Wait for PN load of BRAM to complete.
         when wait_LM_ULM_load =>
            if ( LM_ULM_ready = '1' ) then
               Histo_start <= '1';

-- Give Histo module control of the memory
               BRAM_select <= '1';
               state_next <= wait_Histo;
```

current state



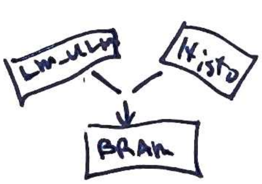
LM_ULM inputs of 'C' data into BRAM

Histo calculates on PN data in BRAM, stores in mid section on BRAM

LM_ULM export of Section Histo Dist Back into 'C'

Could add Diffs store states here

PN_BASE_ != PNL_BASE



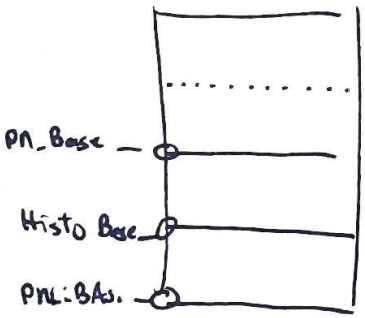
PNL = Large?

PN = small

```
117                end if;
118
119    -- =====================
120    -- Wait for hostogram calculation to complete. Continue to give Histo module
       control of the memory
121            when wait_Histo =>
122                BRAM_select <= '1';
123
124                if ( Histo_ready = '1' ) then
125
126    -- Start memory output operation to C program
127                    LM_ULM_start <= '1';
128
129    -- Setup memory base and upper_limit for unloading of histogram from BRAM. ALWAYS
       SUBSTRACT 1 from the 'UPPER_LIMIT'
130                    LM_ULM_base_address <= std_logic_vector(to_unsigned(HISTO_BRAM_BASE
                       , PNL_BRAM_ADDR_SIZE_NB));
131                    LM_ULM_upper_limit <= std_logic_vector(to_unsigned(
                       HISTO_BRAM_UPPER_LIMIT - 1, PNL_BRAM_ADDR_SIZE_NB));
132
133    -- Set LoadUnloadMem mode to 'unload' data from BRAM to C program
134                    LM_ULM_load_unload <= '1';
135                    state_next <= wait_LM_ULM_unload;
136                end if;
137
138    -- =====================
139    -- Wait for histogram data to be completely transfered to C program
140            when wait_LM_ULM_unload =>
141                LM_ULM_load_unload <= '1';
142                if ( LM_ULM_ready = '1' ) then
143                    state_next <= idle;
144                end if;
145
146        end case;
147    end process;
148
149    ready <= ready_reg;
150 end beh;
151
```

will pass new address BASE for Diffs HI

new Histo will start @ same Base but upper limit will be 1/2 of old.



During 'init' for Histo LM_ULM doesn't touch the data.

Histo handles storing the finish results on its own