TOP VHD

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer: Professor Jim Plusquellic
4  --
5  -- Create Date:
6  -- Design Name:
7  -- Module Name:    Top - 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 --
   ================================================================================
   ================
22 --
   ================================================================================
   ================
23 library IEEE;
24 use IEEE.STD_LOGIC_1164.ALL;
25 use IEEE.NUMERIC_STD.all;
26
27 library work;
28 use work.DataTypes_pkg.all;
29
30 entity Top is
31    port (
32       Clk: in std_logic;
33       PS_RESET_N: in std_logic;
34       GPIO_Ins: in std_logic_vector(31 downto 0);
35       GPIO_Outs: out std_logic_vector(31 downto 0);
36       PNL_BRAM_addr: out std_logic_vector (PNL_BRAM_ADDR_SIZE_NB-1 downto 0);
37       PNL_BRAM_din: out std_logic_vector (PNL_BRAM_DBITS_WIDTH_NB-1 downto 0);
38       PNL_BRAM_dout: in std_logic_vector (PNL_BRAM_DBITS_WIDTH_NB-1 downto 0);
39       PNL_BRAM_we: out std_logic_vector (0 to 0);
40       DEBUG_IN: in std_logic;
41       DEBUG_OUT: out std_logic
42       );
43 end Top;
44
45 architecture beh of Top is
46
47 -- GPIO INPUT BIT ASSIGNMENTS
48    constant IN_CP_RESET: integer := 31;
49    constant IN_CP_START: integer := 30;
50    constant IN_CP_LM_ULM_DONE: integer := 25;
51    constant IN_CP_HANDSHAKE: integer := 24;
52
53 -- GPIO OUTPUT BIT ASSIGNMENTS
54    constant OUT_SM_READY: integer := 31;
55    constant HIST_ERR_BIT: integer := 30;
56    constant OUT_SM_HANDSHAKE: integer := 28;
57
58 -- Signal declarations
59    signal RESET: std_logic;
60
```

32 Bit GPIO

32 Bit GPIO

(13-1)= (12 downto 0)

(4-1) = (3 downto 0)

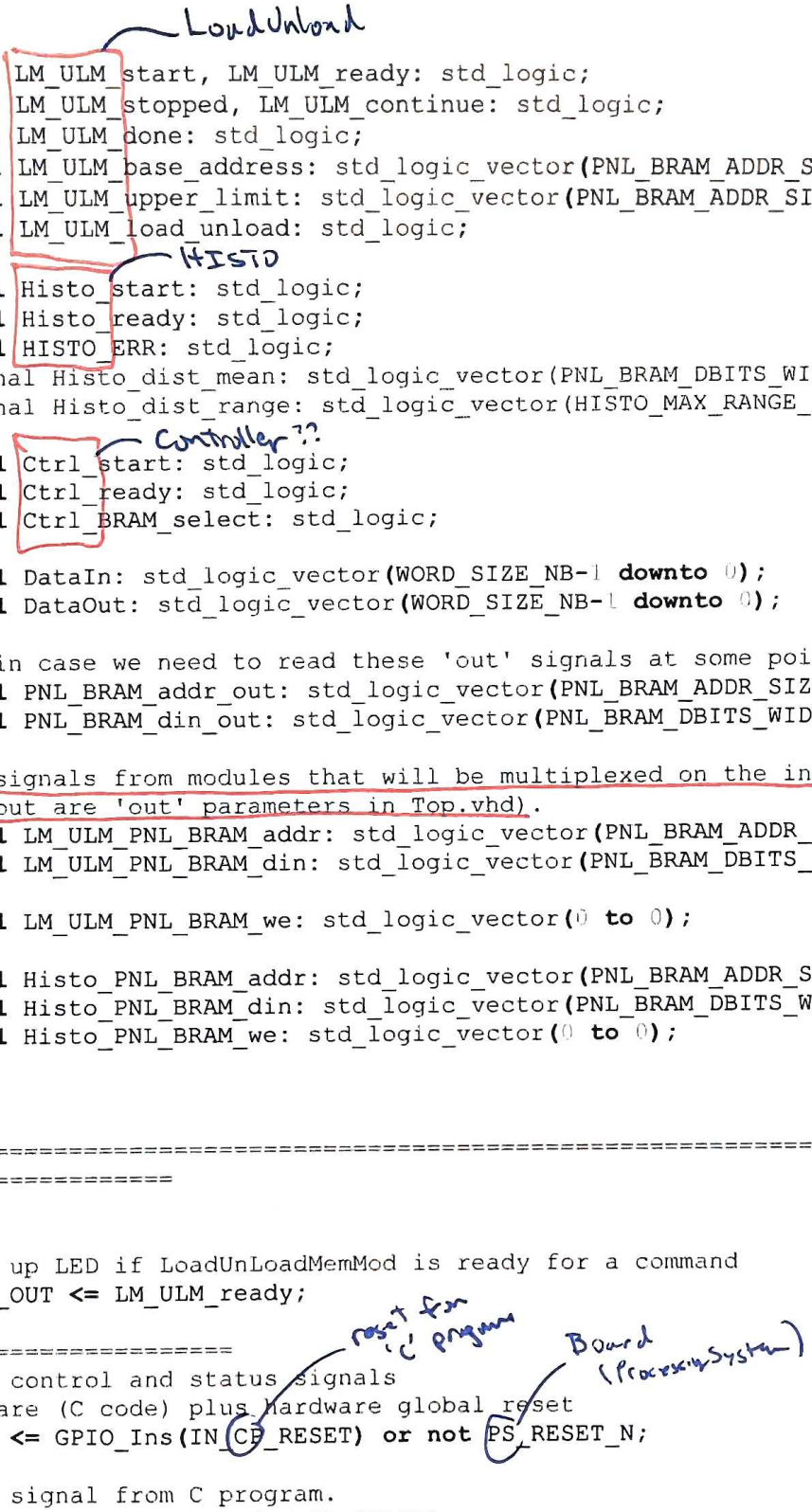
'C' program

In ~~control program~~, In from 'C' program

State machine

Out - State machine

```
61      signal LM_ULM_start, LM_ULM_ready: std_logic;
62      signal LM_ULM_stopped, LM_ULM_continue: std_logic;
63      signal LM_ULM_done: std_logic;
64      signal LM_ULM_base_address: std_logic_vector(PNL_BRAM_ADDR_SIZE_NB-1 downto 0);
65      signal LM_ULM_upper_limit: std_logic_vector(PNL_BRAM_ADDR_SIZE_NB-1 downto 0);
66      signal LM_ULM_load_unload: std_logic;
67
68      signal Histo_start: std_logic;
69      signal Histo_ready: std_logic;
70      signal HISTO_ERR: std_logic;
71  --    signal Histo_dist_mean: std_logic_vector(PNL_BRAM_DBITS_WIDTH_NB-1 downto 0);
72  --    signal Histo_dist_range: std_logic_vector(HISTO_MAX_RANGE_NB-1 downto 0);
73
74      signal Ctrl_start: std_logic;
75      signal Ctrl_ready: std_logic;
76      signal Ctrl_BRAM_select: std_logic;
77
78      signal DataIn: std_logic_vector(WORD_SIZE_NB-1 downto 0);
79      signal DataOut: std_logic_vector(WORD_SIZE_NB-1 downto 0);
80
81  -- Just in case we need to read these 'out' signals at some point
82      signal PNL_BRAM_addr_out: std_logic_vector(PNL_BRAM_ADDR_SIZE_NB-1 downto 0);
83      signal PNL_BRAM_din_out: std_logic_vector(PNL_BRAM_DBITS_WIDTH_NB-1 downto 0);
84
85  -- BRAM signals from modules that will be multiplexed on the input ports of the
    memory (but are 'out' parameters in Top.vhd).
86      signal LM_ULM_PNL_BRAM_addr: std_logic_vector(PNL_BRAM_ADDR_SIZE_NB-1 downto 0);
87      signal LM_ULM_PNL_BRAM_din: std_logic_vector(PNL_BRAM_DBITS_WIDTH_NB-1 downto 0
        );
88      signal LM_ULM_PNL_BRAM_we: std_logic_vector(0 to 0);
89
90      signal Histo_PNL_BRAM_addr: std_logic_vector(PNL_BRAM_ADDR_SIZE_NB-1 downto 0);
91      signal Histo_PNL_BRAM_din: std_logic_vector(PNL_BRAM_DBITS_WIDTH_NB-1 downto 0);
92      signal Histo_PNL_BRAM_we: std_logic_vector(0 to 0);
93
94
95  --
    ================================================================================
    ====================
96      begin
97
98  -- Light up LED if LoadUnLoadMemMod is ready for a command
99      DEBUG_OUT <= LM_ULM_ready;
100
101 -- ====================
102 -- INPUT control and status signals
103 -- Software (C code) plus hardware global reset
104     RESET <= GPIO_Ins(IN_CP_RESET) or not PS_RESET_N;
105
106 -- Start signal from C program.
107     Ctrl_start <= GPIO_Ins(IN_CP_START);
108
109 -- C program asserts if done reading or writing memory (or a portion of it)
110     LM_ULM_done <= GPIO_Ins(IN_CP_LM_ULM_DONE);
111
112 -- Handshake signal
113     LM_ULM_continue <= GPIO_Ins(IN_CP_HANDSHAKE);
114
115 -- Data from C program
116     DataIn <= GPIO_Ins(WORD_SIZE_NB-1 downto 0);
117
118 -- ====================
119 -- OUTPUT control and status signals
120 -- Tell C program whether LoadUnLoadMemMod is ready
```



```
121     GPIO_Outs(OUT_SM_READY) <= Ctrl_ready;
122
123     GPIO_Outs(HISTO_ERR_BIT) <= HISTO_ERR;
124
125  -- Handshake signals
126     GPIO_Outs(OUT_SM_HANDSHAKE) <= LM_ULM_stopped;
127
128  -- Data to C program
129     GPIO_Outs(WORD_SIZE_NB-1 downto 0) <= DataOut;
130
131  -- ======================
132
133  -- Secure BRAM access control module
134     LoadUnLoadMemMod: entity work.LoadUnLoadMem(beh)
135        port map(Clk=>Clk, RESET=>RESET, start=>LM_ULM_start, ready=>LM_ULM_ready,
           load_unload=>LM_ULM_load_unload, stopped=>LM_ULM_stopped,
136           continue=>LM_ULM_continue, done=>LM_ULM_done, base_address=>
              LM_ULM_base_address, upper_limit=>LM_ULM_upper_limit,
137           CP_in_word=>DataIn, CP_out_word=>DataOut,
138           PNL_BRAM_addr=>LM_ULM_PNL_BRAM_addr, PNL_BRAM_din=>LM_ULM_PNL_BRAM_din,
              PNL_BRAM_dout=>PNL_BRAM_dout, PNL_BRAM_we=>LM_ULM_PNL_BRAM_we);
139
140  -- ======================
141  -- Compute a histogram and the distribution constants (mean and range) of the
     data loaded into the upper portion of BRAM.
142     HistoMod: entity work.Histo(beh)
143        port map(Clk=>Clk, RESET=>RESET, start=>Histo_start, ready=>Histo_ready,
           HISTO_ERR=>HISTO_ERR, PNL_BRAM_addr=>Histo_PNL_BRAM_addr,
144           PNL_BRAM_din=>Histo_PNL_BRAM_din, PNL_BRAM_dout=>PNL_BRAM_dout,
              PNL_BRAM_we=>Histo_PNL_BRAM_we);
145
146  -- ======================
147  -- Master controller.
148     ControllerMod: entity work.Controller(beh)
149        port map(Clk=>Clk, RESET=>RESET, start=>Ctrl_start, ready=>Ctrl_ready,
           LM_ULM_start=>LM_ULM_start, LM_ULM_ready=>LM_ULM_ready,
150           LM_ULM_base_address=>LM_ULM_base_address, LM_ULM_upper_limit=>
              LM_ULM_upper_limit, LM_ULM_load_unload=>LM_ULM_load_unload,
151           Histo_start=>Histo_start, Histo_ready=>Histo_ready, BRAM_select=>
              Ctrl_BRAM_select);
152
153  -- ======================
154  -- MEMORY CONTROL
155  -- PNL_BRAM module select logic for addr, din and we.
156     with Ctrl_BRAM_select select
157        PNL_BRAM_addr_out <= LM_ULM_PNL_BRAM_addr when '0',
158                             Histo_PNL_BRAM_addr when others;
159
160     with Ctrl_BRAM_select select
161        PNL_BRAM_din_out <= LM_ULM_PNL_BRAM_din when '0',
162                            Histo_PNL_BRAM_din when others;
163
164     with Ctrl_BRAM_select select
165        PNL_BRAM_we <= LM_ULM_PNL_BRAM_we when '0',
166                       Histo_PNL_BRAM_we when others;
167
168     PNL_BRAM_addr <= PNL_BRAM_addr_out;
169     PNL_BRAM_din <= PNL_BRAM_din_out;
170
171
172  end beh;
173
174
```

mapping signals to values

mapping signals to sources

mapping signals to source.

default LM_ULM... when '0'

doesn't matter if Histo or LoadUnLoad