```
----------------------------------------------------------------------------------
-- Engineer: Zachary Montoya
-- Submitted Date: 12-15-22
-- Module Name:    Top - Behavioral
-- Project Name:   Asteroids
-- Target Devices: Zybo Z7-10
-- Tool versions: Vivado 2020.2
--
-- Comment:
--         This file was entirely created the engineer listed above.
--
----------------------------------------------------------------------------------

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

library work;
use work.DataTypes_pkg.all;

entity Top is
   port (
      Clk: in std_logic;
      reset : in STD_LOGIC;
      GPIO_Ins: in std_logic_vector(31 downto 0);
      GPIO_Outs: out std_logic_vector(31 downto 0);
      PNL_BRAM_addr: out std_logic_vector (12 downto 0);
      PNL_BRAM_din: out std_logic_vector (15 downto 0);
      PNL_BRAM_dout: in std_logic_vector (15 downto 0);
      PNL_BRAM_we: out std_logic_vector (0 to 0);
      hdmi_red : out STD_LOGIC_VECTOR ( 7 downto 0 );
      hdmi_green : out STD_LOGIC_VECTOR ( 7 downto 0 );
      hdmi_blue : out STD_LOGIC_VECTOR ( 7 downto 0 );
      hdmi_hsync : out STD_LOGIC;
      hdmi_vsync : out STD_LOGIC;
      hdmi_enable : out STD_LOGIC;
      btn: in std_logic_vector(3 downto 0); --R2 increasing the array from 2 to 3.
      sw: in std_logic_vector( 0 downto 0) --R2 increasing the array from 2 to 3.
  --    DEBUG_IN: in std_logic;
   --   DEBUG_OUT: out std_logic
      );
end Top;

architecture beh of Top is

-- GPIO INPUT BIT ASSIGNMENTS
   constant IN_CP_RESET: integer := 31;
   constant IN_CP_START: integer := 30;
   constant IN_CP_LM_ULM_LOAD_UNLOAD: integer := 26;
   constant IN_CP_LM_ULM_DONE: integer := 25;
   constant IN_CP_HANDSHAKE: integer := 24;

-- GPIO OUTPUT BIT ASSIGNMENTS
   constant OUT_SM_READY: integer := 31;
   constant OUT_SM_HANDSHAKE: integer := 28;

-- Signal declarations
   signal RESET_final: std_logic;

   signal pixel_x: unsigned(10 downto 0);
   signal pixel_y: unsigned(9 downto 0);
   signal pixel_x_std: std_logic_vector(9 downto 0);
   signal pixel_y_std: std_logic_vector(9 downto 0);

   signal LM_ULM_start, LM_ULM_ready: std_logic;
   signal LM_ULM_stopped, LM_ULM_continue: std_logic;
   signal LM_ULM_done: std_logic;
   signal LM_ULM_base_address: std_logic_vector(PNL_BRAM_ADDR_SIZE_NB-1 downto 0);
   signal LM_ULM_upper_limit: std_logic_vector(PNL_BRAM_ADDR_SIZE_NB-1 downto 0);
   signal LM_ULM_load_unload: std_logic;

   signal DataIn: std_logic_vector(WORD_SIZE_NB-1 downto 0);
```

```
75    signal DataOut: std_logic_vector(WORD_SIZE_NB-1 downto 0);
76
77    signal graph_rgb: std_logic_vector(2 downto 0);
78    signal hdmi_enable_out: STD_LOGIC;
79
80    --NEW
81    signal wall1_on, wall2_on, wall3_on, wall4_on, bar_on, asteroid1_on,
asteroid2_on,missile_on: std_logic;
82    signal wall1_rgb, wall2_rgb, wall3_rgb, wall4_rgb, bar_rgb, asteroid1_rgb,
asteroid2_rgb,missile_rgb,rgb_text: std_logic_vector(2 downto 0);
83    signal Left_Wall, Right_Wall, Bottom_Wall, Top_Wall: std_logic_vector(9 downto 0);
84    signal rom_selector: std_logic_vector(2 downto 0);
85    signal bar_x_reg, bar_y_reg: unsigned( 9 downto 0);
86    signal fire, fire_ready, fire_btn_signal: std_logic;
87
88 --
================================================================================
==================
89    begin
90
91
92
93
94
95 -- Light up LED if LoadUnLoadMemMod is ready for a command
96 --   DEBUG_OUT <= LM_ULM_ready;
97
98 -- ====================
99 -- INPUT control and status signals
100 -- Software (C code) plus hardware global reset
101    RESET_final <= GPIO_Ins(IN_CP_RESET) or reset;
102
103 -- Start signal from C program.
104    LM_ULM_start <= GPIO_Ins(IN_CP_START);
105
106 -- C program controls whether we are loading or unloading memory
107    LM_ULM_load_unload <= GPIO_Ins(IN_CP_LM_ULM_LOAD_UNLOAD);
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) <= LM_ULM_ready;
122
123 -- Handshake signals
124    GPIO_Outs(OUT_SM_HANDSHAKE) <= LM_ULM_stopped;
125
126 -- Data to C program
127    GPIO_Outs(WORD_SIZE_NB-1 downto 0) <= DataOut;
128
129 -- ====================
130 -- Setup memory base and upper_limit
131    LM_ULM_base_address <= std_logic_vector(to_unsigned(PN_BRAM_BASE,
PNL_BRAM_ADDR_SIZE_NB));
132    LM_ULM_upper_limit <= std_logic_vector(to_unsigned(PNL_BRAM_NUM_WORDS_NB -1,
PNL_BRAM_ADDR_SIZE_NB));
133
134 -- Secure BRAM access control module
135    LoadUnLoadMemMod: entity work.LoadUnLoadMem(beh)
136       port map(Clk=>Clk, RESET=>RESET, start=>LM_ULM_start, ready=>LM_ULM_ready,
load_unload=>LM_ULM_load_unload, stopped=>LM_ULM_stopped,
137          continue=>LM_ULM_continue, done=>LM_ULM_done,
base_address=>LM_ULM_base_address, upper_limit=>LM_ULM_upper_limit,
138          CP_in_word=>DataIn, CP_out_word=>DataOut,
139          PNL_BRAM_addr=>PNL_BRAM_addr, PNL_BRAM_din=>PNL_BRAM_din,
PNL_BRAM_dout=>PNL_BRAM_dout, PNL_BRAM_we=>PNL_BRAM_we);
140
141    hdmi_sync_i: entity work.hdmi_sync(rtl)
142        port map (clk=>Clk, reset=>reset, hdmi_hsync=>hdmi_hsync,
hdmi_vsync=>hdmi_vsync, hdmi_enable=>hdmi_enable_out, pixel_x=>pixel_x,
pixel_y=>pixel_y);
```

```
143
144   pixel_x_std <= std_logic_vector(pixel_x(9 downto 0));
145   pixel_y_std <= std_logic_vector(pixel_y);
146
147   asteroid1: entity work.asteroid(rtl)
148   generic map(asteroid_velocity=>2)
149   port map (clk=>Clk,
150             reset=>reset,
151             video_on=>hdmi_enable_out,
152             pixel_x=>pixel_x_std,
153             pixel_y=>pixel_y_std,
154             asteroid_on_out=>asteroid1_on,
155             asteroid_rgb_out => asteroid1_rgb);
156
157   asteroid2: entity work.asteroid(rtl)
158   generic map(asteroid_velocity=>3)
159      port map (clk=>Clk,
160                reset=>reset,
161                video_on=>hdmi_enable_out,
162                pixel_x=>pixel_x_std,
163                pixel_y=>pixel_y_std,
164                asteroid_on_out=>asteroid2_on,
165                asteroid_rgb_out => asteroid2_rgb);
166
167   walls: entity work.walls(rtl)
168   port map (clk=>Clk,
169             reset=>reset,
170             video_on=>hdmi_enable_out,
171             pixel_x=>pixel_x_std,
172             pixel_y=>pixel_y_std,
173             wall1_on_out=>wall1_on,
174             wall2_on_out=>wall2_on,
175             wall3_on_out=>wall3_on,
176             wall4_on_out=>wall4_on,
177             wall1_rgb_out=>wall1_rgb,
178             wall2_rgb_out=>wall2_rgb,
179             wall3_rgb_out=>wall3_rgb,
180             wall4_rgb_out=>wall4_rgb,
181             Left_Wall_Out=>Left_Wall,
182             Right_Wall_Out=>Right_Wall,
183             Bottom_Wall_Out=>Bottom_Wall,
184             Top_Wall_Out=>Top_Wall);
185
186   ship_i: entity work.space_ship(rtl)
187      port map (clk=>Clk,
188                reset=>reset,
189                btn=>btn,
190                sw=>sw,
191                video_on=>hdmi_enable_out,
192                pixel_x=>pixel_x_std,
193                pixel_y=>pixel_y_std,
194                bar_on_out=>bar_on,
195                bar_rgb_out=>bar_rgb,
196                bar_x_reg_out => bar_x_reg,
197                bar_y_reg_out => bar_y_reg,
198                rom_selector_out => rom_selector,
199                fire_out => fire,
200                fire_ready_out => fire_ready,
201                fire_btn_signal_out => fire_btn_signal
202                );
203
204   missile: entity work.missile(rtl)
205   generic map(missile_velocity=>1)
206      port map (clk=>Clk,
207                reset=>reset,
208                btn=>btn,
209                video_on=>hdmi_enable_out,
210                pixel_x=>pixel_x_std,
211                pixel_y=>pixel_y_std,
212                missile_on_out=>missile_on, --Add to Graph
213                missile_rgb_out => missile_rgb, --Add to Graph
214                bar_x_reg_out => bar_x_reg,
215                bar_y_reg_out => bar_y_reg,
216                rom_selector_out => rom_selector,
217                fire_out => fire,
218                fire_ready_out => fire_ready,
219                fire_btn_signal_out => fire_btn_signal
220                );
221
```

```
   text: entity work.font_test_gen(arch)
      port map (clk=>Clk,
                video_on=>hdmi_enable_out,
                pixel_x=>pixel_x_std,
                pixel_y=>pixel_y_std,
                rgb_text_out=>rgb_text
                );

      --
================================================================================
================
   -- turn on the appropriate color depending on the current pixel position.
      process (hdmi_enable_out, wall1_on, wall2_on, wall3_on, wall4_on, bar_on,
asteroid2_on, asteroid1_on, wall1_rgb, wall2_rgb, wall3_rgb, wall4_rgb, bar_rgb,
asteroid2_rgb, asteroid1_rgb)
      begin
      if (hdmi_enable_out = '0') then
         graph_rgb <= "110"; -- blank
      else
         if (wall1_on = '1') then
            graph_rgb <= wall1_rgb;
         elsif (wall2_on = '1') then
            graph_rgb <= wall2_rgb;
         elsif (wall3_on = '1') then
            graph_rgb <= wall3_rgb;
         elsif (wall4_on = '1') then
            graph_rgb <= wall4_rgb;
         elsif (bar_on = '1') then
            graph_rgb <= bar_rgb;
         elsif (asteroid2_on = '1') then
            graph_rgb <= asteroid2_rgb;
         elsif (asteroid1_on = '1') then
            graph_rgb <= asteroid1_rgb;
         elsif (missile_on = '1') then
            graph_rgb <= missile_rgb;
         -- elsif (missile_on = '1') then
         --    graph_rgb <= rgb_text;
         else
            graph_rgb <= "000"; -- Black bkgnd
            graph_rgb <= rgb_text;
         end if;
      end if;
   end process;


--    hdmi_red <= std_logic_vector(resize(pixel_x, 8)) when sw_r = '1' else (others
=> '0');
--    hdmi_green <= std_logic_vector(resize(pixel_y, 8)) when sw_g = '1' else (others
=> '0');
   hdmi_red <= "11111111" when graph_rgb(0) = '1' else (others => '0');
   hdmi_green <= "11111111" when graph_rgb(1) = '1' else (others => '0');
   hdmi_blue <= "11111111" when graph_rgb(2) = '1' else (others => '0');

   hdmi_enable <= hdmi_enable_out;

end beh;
```