11/05/2024

Lab 07 Report

Ryan Beck
Jared Bronson

Objectives

For this lab, interface the FPGA development board to a standard computer monitor using the VGA protocol and hardware. Display the national flags of the 12 given countries. The design will use the two push buttons, the on-board clock, and the VGA. One push button will be used as “reset” to display the first flag. The other push button will be used as “advance”, to display the next flag.

Procedures

Create a Finite State Machine(FSM) with states for the A, B, C, D, Clear, and Debounce. States A, B, C, and D correspond to the horizontal front porch, sync, back porch, and data. The vertical timing is determined when the line count increases after going through the four horizontal states.

Using the system builder, create a Quartus Project File with the VGA, the on-board clock, and the two push buttons. Take the generated Verilog file and create a VHD file with matching port names. Take the Verilog file out of the project after creating the VHD file. In the generic, create values for the pixel count of each horizontal state, the line count of each vertical state, the flag count, and the delay for debouncing.

In the architecture, create variables for pixel count and line count to keep track of both vertical and horizontal values. Create a variable to keep track of which flag is displayed and variables to pass the flag color values into the VGA. Then, a type needs to be declared for the state type to create the FSM.

Define a process for the on-board clock. In this process, on the rising edge of the clock, the next state of the state machine becomes the current state unless either of the buttons are pushed. When the push buttons are pushed, the current state goes to the state of the corresponding button.

Define another process sensitive to the state, pixel count, line count, flag count, and push buttons. This process will have a case statement to define what moves states in the FSM. The Clear state will reset all variables to their default values, and sets the state to A. The Debounce state will have a delay to ensure the FSM does not go to the next state if the button is falsely triggered, and increments the flag count so the next flag is displayed. After the advance button is released, the state is then set to A.

State A is the “Front Porch” of the horizontal timing. This state will start the pixel count at 15, and count down each clock cycle. When the pixel count reaches zero, the state is set to B. State B is the Vertical Sync in the horizontal timing. This state will drive the Horizontal Sync signal low, count down from 95. The next state, C, is the “Back Porch” in the Horizontal timing. The pixel count will start at 47 and count down each clock cycle. On the transition from state C to state D, the first color values of the VGA will be set, to ensure the data is being driven at the correct time. In state D, pixel count will start at 639 and count down. Colors will only be displayed on the VGA if the vertical timing has reached the data portion. Create another case statement inside state D to display the colors on the VGA.

The vertical timing will increment when the FSM cycles back to state A. The vertical “Front Porch” lasts 10 cycles, the sync lasts 2 cycles, the “Back Porch” lasts 33 cycles, and the data lasts 480 cycles. Data can only be sent to the VGA when the vertical timing is in the data portion. To ensure the FSM runs at the refresh rate of the VGA, 25 MHz, the FSM will only execute instructions every other clock tick.

Results

The VGA file, shown in figures, successfully displays flags on a VGA monitor using the DE10-Lite board. The design creates an FSM to implement proper timing for the pixels and the lines of the display. Push buttons are used to go back to the starting flag and cycle through the 12 required flags.

Conclusion

In conclusion, we were able to implement an FSM on the development board to display different flags to a standard computer monitor. One push button resets all values and returns the display to the beginning flag. The other push button advances the display to the next flag. The FSM updates at a rate of 25 MHz. This is done using the on-board 50MHz clock, and only executing on every other clock tick.

Figures

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity vga is
6
7     generic(
8        --     Add generics here     --
9        -- NAME : TYPE := DEFAULT_VALUE (separated by ; ) --
10       A_COUNT_H    : integer := 15;
11       B_COUNT_H    : integer := 95;
12       C_COUNT_H    : integer := 47;
13       D_COUNT_H    : integer := 639;
14       LAST_A_V     : unsigned(9 downto 0) := to_unsigned(9,   10);
15       LAST_B_V     : unsigned(9 downto 0) := to_unsigned(11,  10);
16       LAST_C_V     : unsigned(9 downto 0) := to_unsigned(44,  10);
17       LAST_D_V     : unsigned(9 downto 0) := to_unsigned(524, 10);
18       L_COUNT      : unsigned(9 downto 0) := to_unsigned(524, 10);
19       F_COUNT      : integer := 11;
20       DELAY        : integer := 500000;
21       -- Stripe size generics for simulating
22       START_LEFT_STRIPE : integer := 640;
23       END_LEFT_STRIPE   : integer := 427;
24       START_RIGHT_STRIPE: integer := 213
25    );
26
27    port (
28          -- Declare module ports here --
29       -- NAME : DIRECTION TYPE (separated by ; ) --
30
31       -- CLK input --
32       MAX10_CLK1_50  : in std_logic; -- 50 MHz 1
33
34       -- Button input --
35       KEY : in std_logic_vector (1 downto 0);
36
37       -- VGA --
38       VGA_R     : out std_logic_vector(3 downto 0);
39       VGA_G     : out std_logic_vector(3 downto 0);
40       VGA_B     : out std_logic_vector(3 downto 0);
41       VGA_HS    : out std_logic;
42       VGA_VS    : out std_logic
43    );
44
45 end entity vga;
46
47 architecture behavioral of vga is
48
49    -- Declare internal signals here -- (terminated by ; )
50    -- signal NAME : TYPE ;
51
52    signal pix_count        : integer := 0;
53    signal next_pix_count   : integer := 0;
54    signal lin_count        : unsigned(9 downto 0) := to_unsigned(0, 10);
55    signal next_lin_count   : unsigned(9 downto 0) := to_unsigned(0, 10);
56    signal flg_count        : integer := 0;
57    signal next_flg_count   : integer := 0;
58    signal clk_count        : integer := 0;
59    signal next_clk_count   : integer := 0;
60
61    signal timer            : integer := 0;    -- Timer for debounce
62    signal next_timer       : integer := 0;
63
64    signal next_VGA_R       : std_logic_vector(3 downto 0) := "0000";
65    signal next_VGA_G       : std_logic_vector(3 downto 0) := "0000";
66    signal next_VGA_B       : std_logic_vector(3 downto 0) := "0000";
67    signal next_VGA_HS      : std_logic := '1';
68    signal next_VGA_VS      : std_logic := '1';
69
70    signal current_VGA_R       : std_logic_vector(3 downto 0) := "0000";
71    signal current_VGA_G       : std_logic_vector(3 downto 0) := "0000";
72    signal current_VGA_B       : std_logic_vector(3 downto 0) := "0000";
73    signal current_VGA_HS      : std_logic := '1';
74    signal current_VGA_VS      : std_logic := '1';
75
76    signal LEFT_EDGE_YELLOW    : integer := 163;
77    signal RIGHT_EDGE_YELLOW   : integer := 0;
78    signal next_LEFT_EDGE_YELLOW  : integer := 163;
79    signal next_RIGHT_EDGE_YELLOW : integer := 0;
80
81    -- FSM States
82    type state_type is (
83       Clear,
84       A,
85       B,
86       C,
87       D,
88       Debounce
89       );
90
91    signal current_state, next_state: state_type;
92
```

Figure 1: VGA.vhd Pt 1

```
93  begin
94
95      -- Define module behavior here --
96
97      -- Make future the present --
98      process ( MAX10_CLK1_50 ) -- Sensitivity list goes in ()
99      begin
100        if rising_edge( MAX10_CLK1_50 ) then
101           -- Only trigger every other clock cycle (25 MHz)
102           if clk_count = 1 then
103              clk_count <= 0;
104
105              -- If Reset
106              if KEY(0) = '0' then
107                 -- Reset behavior --
108                 pix_count <= next_pix_count;
109                 lin_count <= next_lin_count;
110                 flg_count <= next_flg_count;
111                 timer <= next_timer;
112                 current_VGA_R  <= next_VGA_R;
113                 current_VGA_G  <= next_VGA_G;
114                 current_VGA_B  <= next_VGA_B;
115                 current_VGA_HS <= next_VGA_HS;
116                 current_VGA_VS <= next_VGA_VS;
117                 current_state <= Clear;
118                 LEFT_EDGE_YELLOW     <= next_LEFT_EDGE_YELLOW;
119                 RIGHT_EDGE_YELLOW    <= next_RIGHT_EDGE_YELLOW;
120
121             -- If next
122             elsif KEY(1) = '0' then
123                 pix_count <= next_pix_count;
124                 lin_count <= next_lin_count;
125                 flg_count <= next_flg_count;
126                 timer <= next_timer;
127                 current_VGA_R  <= next_VGA_R;
128                 current_VGA_G  <= next_VGA_G;
129                 current_VGA_B  <= next_VGA_B;
130                 current_VGA_HS <= next_VGA_HS;
131                 current_VGA_VS <= next_VGA_VS;
132                 current_state <= Debounce;
133                 LEFT_EDGE_YELLOW     <= next_LEFT_EDGE_YELLOW;
134                 RIGHT_EDGE_YELLOW    <= next_RIGHT_EDGE_YELLOW;
135             -- Continue same flag
136             else
137                 -- Normal behavior --
138                 pix_count <= next_pix_count;
139                 lin_count <= next_lin_count;
140                 flg_count <= next_flg_count;
141                 timer     <= next_timer;
142                 current_VGA_R  <= next_VGA_R;
143                 current_VGA_G  <= next_VGA_G;
144                 current_VGA_B  <= next_VGA_B;
145                 current_VGA_HS <= next_VGA_HS;
146                 current_VGA_VS <= next_VGA_VS;
147                 current_state <= next_state;
148                 LEFT_EDGE_YELLOW     <= next_LEFT_EDGE_YELLOW;
149                 RIGHT_EDGE_YELLOW    <= next_RIGHT_EDGE_YELLOW;
150             end if;
151          else
152             clk_count <= clk_count + 1;
153          end if;
154       end if;
155    end process;
156
157
158    -- Determine the future --
159    process ( current_state, pix_count, KEY, lin_count, timer, flg_count )
160    begin
161       case current_state is
162          when Clear =>
163             -- Drive data low --
164             next_VGA_R  <= "0000";
165             next_VGA_G  <= "0000";
166             next_VGA_B  <= "0000";
167             -- Sync high
168             next_VGA_HS <= '1';
169             next_VGA_VS <= '1';
170
171             next_LEFT_EDGE_YELLOW   <= 163;
172             next_RIGHT_EDGE_YELLOW  <= 0;
173
174             if KEY(0) = '0' then
175                -- Reset counters
176                next_pix_count <= 0;
177                next_lin_count <= to_unsigned(0, lin_count'length);
178                next_flg_count <= 0;
179                next_timer <= 0;
180                next_state <= Clear;
181             else
182                -- Prep for state A
183                next_pix_count <= A_COUNT_H;
184                next_lin_count <= lin_count;
185                next_flg_count <= flg_count;
186                next_timer <= timer;
187                next_state <= A;
188             end if;
```

Figure 2: VGA.vhd Pt 2

```
189
190             when A =>
191                 if lin_count = to_unsigned(0, lin_count'length) then
192                     next_LEFT_EDGE_YELLOW   <= 163;
193                     next_RIGHT_EDGE_YELLOW  <= 0;
194                 else
195                     next_LEFT_EDGE_YELLOW   <= LEFT_EDGE_YELLOW;
196                     next_RIGHT_EDGE_YELLOW  <= RIGHT_EDGE_YELLOW;
197                 end if;
198                 -- Drive data low --
199                 next_VGA_R  <= "0000";
200                 next_VGA_G  <= "0000";
201                 next_VGA_B  <= "0000";
202                 -- Sync high
203                 if pix_count /= 0 then
204                     next_pix_count <= pix_count - 1;
205                     next_lin_count <= lin_count;
206                     next_flg_count <= flg_count;
207                     next_timer <= timer;
208                     next_state <= A;
209                     next_VGA_HS <= '1';
210                     if (lin_count > LAST_A_V) and (lin_count <= LAST_B_V) then
211                         next_VGA_VS <= '0';
212                     else
213                         next_VGA_VS <= '1';
214                     end if;
215                 else
216                     next_pix_count <= B_COUNT_H;
217                     next_lin_count <= lin_count;
218                     next_flg_count <= flg_count;
219                     next_timer <= timer;
220                     next_state <= B;
221                     next_VGA_HS <= '0';
222                     if (lin_count > LAST_A_V) and (lin_count <= LAST_B_V) then
223                         next_VGA_VS <= '0';
224                     else
225                         next_VGA_VS <= '1';
226                     end if;
227                 end if;
228
229             when B =>
230                 -- Drive data low --
231                 next_VGA_R  <= "0000";
232                 next_VGA_G  <= "0000";
233                 next_VGA_B  <= "0000";
234                 next_LEFT_EDGE_YELLOW   <= LEFT_EDGE_YELLOW;
235                 next_RIGHT_EDGE_YELLOW  <= RIGHT_EDGE_YELLOW;
236                 -- Sync low
237                 if pix_count /= 0 then
238                     next_pix_count <= pix_count - 1;
239                     next_lin_count <= lin_count;
240                     next_flg_count <= flg_count;
241                     next_timer <= timer;
242                     next_state <= B;
243                     next_VGA_HS <= current_VGA_HS;
244                     next_VGA_VS <= current_VGA_VS;
245                 else
246                     next_pix_count <= C_COUNT_H;
247                     next_lin_count <= lin_count;
248                     next_flg_count <= flg_count;
249                     next_timer <= timer;
250                     next_state <= C;
251                     next_VGA_HS <= '1';
252                     if (lin_count > LAST_A_V) and (lin_count <= LAST_B_V) then
253                         next_VGA_VS <= '0';
254                     else
255                         next_VGA_VS <= '1';
256                     end if;
257                 end if;
258
259             when C =>
260                 -- Sync high
261                 next_VGA_HS <= '1';
262                 next_LEFT_EDGE_YELLOW   <= LEFT_EDGE_YELLOW;
263                 next_RIGHT_EDGE_YELLOW  <= RIGHT_EDGE_YELLOW;
264                 if (lin_count > LAST_A_V) and (lin_count <= LAST_B_V) then
265                     next_VGA_VS <= '0';
266                 else
267                     next_VGA_VS <= '1';
268                 end if;
269                 if pix_count /= 0 then
270                     next_pix_count <= pix_count - 1;
271                     next_lin_count <= lin_count;
272                     next_flg_count <= flg_count;
273                     next_timer <= timer;
274                     next_state <= C;
275                     next_VGA_R <= "0000";
276                     next_VGA_G <= "0000";
277                     next_VGA_B <= "0000";
278                 else
279                     next_pix_count <= D_COUNT_H;
280                     next_lin_count <= lin_count;
281                     next_flg_count <= flg_count;
282                     next_timer <= timer;
283                     next_state <= D;
284                     if lin_count > LAST_C_V then
285                         case flg_count is
```

Figure 3: VGA.vhd Pt 3

```
case flg_count is
    when 0 =>
        -- France
        next_VGA_R <= "0000";
        next_VGA_G <= "0010";
        next_VGA_B <= "1001";
    when 1 =>
        -- Italy
        next_VGA_R <= "0000";
        next_VGA_G <= "1001";
        next_VGA_B <= "0100";
    when 2=>
        -- Ireland
        next_VGA_R <= "0001";
        next_VGA_G <= "1001";
        next_VGA_B <= "0110";
    when 3 =>
        -- Belgium
        next_VGA_R <= "0000";
        next_VGA_G <= "0000";
        next_VGA_B <= "0000";
    when 4 =>
        -- Mali
        next_VGA_R <= "0001";
        next_VGA_G <= "1011";
        next_VGA_B <= "0011";
    when 5 =>
        -- Chad
        next_VGA_R <= "0000";
        next_VGA_G <= "0010";
        next_VGA_B <= "0110";
    when 6 =>
        -- Nigeria
        next_VGA_R <= "0000";
        next_VGA_G <= "1000";
        next_VGA_B <= "0101";
    when 7 =>
        -- Ivory Coast
        next_VGA_R <= "1111";
        next_VGA_G <= "1001";
        next_VGA_B <= "0000";
    when 8 =>
        -- Poland
        if (lin_count > to_unsigned(44, lin_count'length)) and (lin_count <= to_unsigned(284, lin_count'length)) then
            next_VGA_R <= "1111";
            next_VGA_G <= "1111";
            next_VGA_B <= "1111";
        elsif (lin_count > to_unsigned(284, lin_count'length)) and (lin_count <= to_unsigned(524, lin_count'length)) then
            next_VGA_R <= "1101";
            next_VGA_G <= "0001";
            next_VGA_B <= "0011";
        else
            next_VGA_R <= "0000";
            next_VGA_G <= "0000";
            next_VGA_B <= "0000";
        end if;
    when 9 =>
        --Flag 9-Germany
        if (lin_count > to_unsigned(44, lin_count'length)) and (lin_count <= to_unsigned(204, lin_count'length)) then
            --BLACK = #000000
            next_VGA_R <= "0000";
            next_VGA_G <= "0000";
            next_VGA_B <= "0000";
        elsif (lin_count > to_unsigned(204, lin_count'length)) and (lin_count <= to_unsigned(364, lin_count'length)) then
            --RED = #dd0000
            next_VGA_R <= "1101";
            next_VGA_G <= "0000";
            next_VGA_B <= "0000";
        elsif (lin_count > to_unsigned(364, lin_count'length)) and (lin_count <= to_unsigned(524, lin_count'length)) then
            --YELLOW = #ffce00
            next_VGA_R <= "1111";
            next_VGA_G <= "1100";
            next_VGA_B <= "0000";
        else
            next_VGA_R <= "0000";
            next_VGA_G <= "0000";
            next_VGA_B <= "0000";
        end if;

    when 10 =>
        --Flag 10-Austria
        if (lin_count > to_unsigned(44, lin_count'length)) and (lin_count <= to_unsigned(204, lin_count'length)) then
            --RED = #ed2939
            next_VGA_R <= "1110";
            next_VGA_G <= "0010";
            next_VGA_B <= "0011";
        elsif (lin_count > to_unsigned(204, lin_count'length)) and (lin_count <= to_unsigned(364, lin_count'length)) then
            --WHITE = #FFFFFF
            next_VGA_R <= "1111";
            next_VGA_G <= "1111";
            next_VGA_B <= "1111";
        elsif (lin_count > to_unsigned(364, lin_count'length)) and (lin_count <= to_unsigned(524, lin_count'length)) then
            --RED = #ed2939
            next_VGA_R <= "1110";
            next_VGA_G <= "0010";
            next_VGA_B <= "0011";
        else
            next_VGA_R <= "0000";
            next_VGA_G <= "0000";
            next_VGA_B <= "0000";
        end if;
```

Figure 4: VGA.vhd Pt 4

```
387                 when 11 =>
388                     -- Congo
389                     next_VGA_R <= "0000";
390                     next_VGA_G <= "1001";
391                     next_VGA_B <= "0100";
392 
393                 when others =>
394                     next_VGA_R <= "0000";
395                     next_VGA_G <= "0010";
396                     next_VGA_B <= "1001";
397             end case;
398         else
399             next_VGA_R <= "0000";
400             next_VGA_G <= "0000";
401             next_VGA_B <= "0000";
402         end if;
403     end if;
404 
405 when D =>
406     -- Sync high
407     next_VGA_HS <= '1';
408     if pix_count /= 0 then
409         next_pix_count <= pix_count - 1;
410         next_lin_count <= lin_count;
411         next_flg_count <= flg_count;
412         next_timer <= timer;
413         next_state <= D;
414         next_LEFT_EDGE_YELLOW   <= LEFT_EDGE_YELLOW;
415         next_RIGHT_EDGE_YELLOW  <= RIGHT_EDGE_YELLOW;
416         if (lin_count > LAST_A_V) and (lin_count <= LAST_B_V) then
417             next_VGA_VS <= '0';
418         else
419             next_VGA_VS <= '1';
420         end if;
421         if lin_count > LAST_C_V then
422             --Flag Case
423             case flg_count is
424                 when 0 =>
425                     --Flag 0 - France
426                     if (pix_count > START_RIGHT_STRIPE) and (pix_count <= END_LEFT_STRIPE) then
427                         --White = #FFFFFF
428                         next_VGA_R <= "1111";
429                         next_VGA_G <= "1111";
430                         next_VGA_B <= "1111";
431                     elsif pix_count <= START_RIGHT_STRIPE then
432                         --RED = #ed2939
433                         next_VGA_R <= "1110";
434                         next_VGA_G <= "0010";
435                         next_VGA_B <= "0011";
436                     else
437                         -- Outside of data or in first stripe, keep the same
438                         next_VGA_R <= current_VGA_R;
439                         next_VGA_G <= current_VGA_G;
440                         next_VGA_B <= current_VGA_B;
441                     end if;
442 
443                 when 1 =>
444                     --Flag 1-Italy
445                     if (pix_count > 213) and (pix_count <= 427) then
446                         --White = #FFFFFF
447                         next_VGA_R <= "1111";
448                         next_VGA_G <= "1111";
449                         next_VGA_B <= "1111";
450                     elsif pix_count <= 213 then
451                         --RED = #ce2b37
452                         next_VGA_R <= "1100";
453                         next_VGA_G <= "0010";
454                         next_VGA_B <= "0011";
455                     else
456                         next_VGA_R <= current_VGA_R;
457                         next_VGA_G <= current_VGA_G;
458                         next_VGA_B <= current_VGA_B;
459                     end if;
460 
461                 when 2 =>
462                     --Flag 2-Ireland
463                     if (pix_count > 213) and (pix_count <= 427) then
464                         --White = #FFFFFF
465                         next_VGA_R <= "1111";
466                         next_VGA_G <= "1111";
467                         next_VGA_B <= "1111";
468                     elsif pix_count <= 213 then
469                         --ORANGE = #ff883e
470                         next_VGA_R <= "1111";
471                         next_VGA_G <= "1000";
472                         next_VGA_B <= "0011";
473                     else
474                         next_VGA_R <= current_VGA_R;
475                         next_VGA_G <= current_VGA_G;
476                         next_VGA_B <= current_VGA_B;
477                     end if;
478 
479                 when 3 =>
480                     --Flag 3-Belgium
481                     if (pix_count > 213) and (pix_count <= 427) then
482                         --Yellow = #fae042
483                         next_VGA_R <= "1111";
484                         next_VGA_G <= "1110";
485                         next_VGA_B <= "0100";
```

Figure 5: VGA.vhd Pt 5

```
                elsif pix_count <= 213 then
                    --RED = #ed2939
                    next_VGA_R <= "1110";
                    next_VGA_G <= "0010";
                    next_VGA_B <= "0011";
                else
                    next_VGA_R <= current_VGA_R;
                    next_VGA_G <= current_VGA_G;
                    next_VGA_B <= current_VGA_B;
                end if;

            when 4 =>
                --Flag 4-Mali
                if (pix_count > 213) and (pix_count <= 427) then
                    --Yellow = #fcd116
                    next_VGA_R <= "1111";
                    next_VGA_G <= "1101";
                    next_VGA_B <= "0001";
                elsif pix_count <= 213 then
                    --RED = #ce1126
                    next_VGA_R <= "1100";
                    next_VGA_G <= "0001";
                    next_VGA_B <= "0010";
                else
                    next_VGA_R <= current_VGA_R;
                    next_VGA_G <= current_VGA_G;
                    next_VGA_B <= current_VGA_B;
                end if;

            when 5 =>
                --Flag 5-Chad
                if (pix_count > 213) and (pix_count <= 427) then
                    --Yellow = #fecb00
                    next_VGA_R <= "1111";
                    next_VGA_G <= "1100";
                    next_VGA_B <= "0000";
                elsif pix_count <= 213 then
                    --RED = #c60c30
                    next_VGA_R <= "1100";
                    next_VGA_G <= "0000";
                    next_VGA_B <= "0011";
                else
                    next_VGA_R <= current_VGA_R;
                    next_VGA_G <= current_VGA_G;
                    next_VGA_B <= current_VGA_B;
                end if;

            when 6 =>
                --Flag 6-Nigeria
                if (pix_count > 213) and (pix_count <= 427) then
                    --WHITE = #FFFFFF
                    next_VGA_R <= "1111";
                    next_VGA_G <= "1111";
                    next_VGA_B <= "1111";
                elsif pix_count <= 213 then
                    --GREEN = #008751
                    next_VGA_R <= "0000";
                    next_VGA_G <= "1000";
                    next_VGA_B <= "0101";
                else
                    next_VGA_R <= current_VGA_R;
                    next_VGA_G <= current_VGA_G;
                    next_VGA_B <= current_VGA_B;
                end if;

            when 7 =>
                --Flag 7-Ivory Coast
                if (pix_count > 213) and (pix_count <= 427) then
                    --WHITE = #FFFFFF
                    next_VGA_R <= "1111";
                    next_VGA_G <= "1111";
                    next_VGA_B <= "1111";
                elsif pix_count <= 213 then
                    --GREEN = #009e60
                    next_VGA_R <= "0000";
                    next_VGA_G <= "1001";
                    next_VGA_B <= "0110";
                else
                    next_VGA_R <= current_VGA_R;
                    next_VGA_G <= current_VGA_G;
                    next_VGA_B <= current_VGA_B;
                end if;

            when 8 =>
                --Flag 8-Poland
                next_VGA_R <= current_VGA_R;
                next_VGA_G <= current_VGA_G;
                next_VGA_B <= current_VGA_B;

            when 9 =>
                -- Germany
                next_VGA_R <= current_VGA_R;
                next_VGA_G <= current_VGA_G;
                next_VGA_B <= current_VGA_B;

            when 10 =>
                -- Austria
                next_VGA_R <= current_VGA_R;
                next_VGA_G <= current_VGA_G;
                next_VGA_B <= current_VGA_B;
```

Figure 6: VGA.vhd Pt 6

```
587             when 11 =>
588                --Flag 11-Republic of Congo
589                if (pix_count > LEFT_EDGE_YELLOW) and (pix_count <= D_COUNT_H) then
590                   --GREEN = #009543
591                   next_VGA_R <= "0000";
592                   next_VGA_G <= "1001";
593                   next_VGA_B <= "0100";
594                elsif(pix_count <= LEFT_EDGE_YELLOW) and (pix_count > RIGHT_EDGE_YELLOW) then
595                   --YELLOW = #fbde4a
596                   next_VGA_R <= "1111";
597                   next_VGA_G <= "1101";
598                   next_VGA_B <= "0100";
599                elsif(pix_count <= RIGHT_EDGE_YELLOW) and (pix_count > 0) then
600                   --RED = #dc241f
601                   next_VGA_R <= "1101";
602                   next_VGA_G <= "0010";
603                   next_VGA_B <= "0001";
604                else
605                   next_VGA_R <= "0000";
606                   next_VGA_G <= "0000";
607                   next_VGA_B <= "0000";
608                end if;
609
610             when others =>
611                --Flag 0 - France
612                if (pix_count > START_RIGHT_STRIPE) and (pix_count <= END_LEFT_STRIPE) then
613                   --White = #FFFFFF
614                   next_VGA_R <= "1111";
615                   next_VGA_G <= "1111";
616                   next_VGA_B <= "1111";
617                elsif pix_count <= START_RIGHT_STRIPE then
618                   --RED = #ed2939
619                   next_VGA_R <= "1110";
620                   next_VGA_G <= "0010";
621                   next_VGA_B <= "0011";
622                else
623                   -- Outside of data or in first stripe, keep the same
624                   next_VGA_R <= current_VGA_R;
625                   next_VGA_G <= current_VGA_G;
626                   next_VGA_B <= current_VGA_B;
627                end if;
628
629          end case;
630
631       else
632          next_VGA_R <= "0000";
633          next_VGA_G <= "0000";
634          next_VGA_B <= "0000";
635       end if;
636    -- Last pixel
637    else
638       next_pix_count <= A_COUNT_H;
639       if lin_count = L_COUNT then
640          next_lin_count <= to_unsigned(0, lin_count'length);
641          next_LEFT_EDGE_YELLOW   <= 163;
642          next_RIGHT_EDGE_YELLOW  <= 0;
643       elsif (lin_count > LAST_C_V) then
644          next_lin_count <= lin_count + to_unsigned(1, lin_count'length);
645          next_LEFT_EDGE_YELLOW   <= LEFT_EDGE_YELLOW + 1;
646          next_RIGHT_EDGE_YELLOW  <= RIGHT_EDGE_YELLOW + 1;
647       else
648          next_lin_count <= lin_count + to_unsigned(1, lin_count'length);
649          next_LEFT_EDGE_YELLOW   <= LEFT_EDGE_YELLOW;
650          next_RIGHT_EDGE_YELLOW  <= RIGHT_EDGE_YELLOW;
651       end if;
652       next_flg_count <= flg_count;
653       next_timer <= timer;
654       next_state <= A;
655       next_VGA_R <= "0000";
656       next_VGA_G <= "0000";
657       next_VGA_B <= "0000";
658       next_VGA_HS <= current_VGA_HS;
659       if (lin_count >= LAST_A_V) and (lin_count < LAST_B_V) then
660          next_VGA_VS <= '0';
661       else
662          next_VGA_VS <= '1';
663       end if;
664    end if;
665
666 when Debounce =>
667    --If timer = DELAY
668    if timer = DELAY then
669       --If add is still pressed
670       if KEY(1) = '0' then
671          --Next state is pressed
672          next_pix_count <= D_COUNT_H;
673          next_lin_count <= lin_count;
674          next_flg_count <= flg_count;
675          next_timer <= timer;
676          next_state <= Debounce;
677          next_VGA_R  <= current_VGA_R;
678          next_VGA_G  <= current_VGA_G;
679          next_VGA_B  <= current_VGA_B;
680          next_VGA_HS <= current_VGA_HS;
681          next_VGA_VS <= current_VGA_VS;
682       else
683          next_pix_count <= A_COUNT_H;
684          next_lin_count <= to_unsigned(0, lin_count'length);
685          if flg_count = F_COUNT then
686             next_flg_count <= 0;
```

Figure 7: VGA.vhd Pt 7

```
687                 else
688                    next_flg_count <= flg_count + 1;
689                 end if;
690                 next_timer <= 0;
691                 next_state <= A;
692                 next_VGA_R  <= "0000";
693                 next_VGA_G  <= "0000";
694                 next_VGA_B  <= "0000";
695                 next_VGA_HS <= '1';
696                 next_VGA_VS <= '1';
697              end if;
698           else
699              --Increment timer
700              next_pix_count <= pix_count;
701              next_lin_count <= lin_count;
702              next_flg_count <= flg_count;
703              next_timer <= timer + 1;
704              next_state <= Debounce;
705              next_VGA_R  <= current_VGA_R;
706              next_VGA_G  <= current_VGA_G;
707              next_VGA_B  <= current_VGA_B;
708              next_VGA_HS <= current_VGA_HS;
709              next_VGA_VS <= current_VGA_VS;
710           end if;
711           next_LEFT_EDGE_YELLOW   <= LEFT_EDGE_YELLOW;
712           next_RIGHT_EDGE_YELLOW  <= RIGHT_EDGE_YELLOW;
713
714        when others =>
715           next_pix_count <= pix_count;
716           next_lin_count <= lin_count;
717           next_flg_count <= flg_count;
718           next_timer <= timer;
719           next_state  <= Clear;
720           next_VGA_R  <= current_VGA_R;
721           next_VGA_G  <= current_VGA_G;
722           next_VGA_B  <= current_VGA_B;
723           next_VGA_HS <= current_VGA_HS;
724           next_VGA_VS <= current_VGA_VS;
725           next_LEFT_EDGE_YELLOW   <= LEFT_EDGE_YELLOW;
726           next_RIGHT_EDGE_YELLOW  <= RIGHT_EDGE_YELLOW;
727
728     end case;
729  end process;
730
731  -- Send current_VGA data to outputs
732  process ( current_VGA_R, current_VGA_G, current_VGA_B, current_VGA_HS, current_VGA_VS )
733  begin
734     VGA_R  <= current_VGA_R;
735     VGA_G  <= current_VGA_G;
736     VGA_B  <= current_VGA_B;
737     VGA_HS <= current_VGA_HS;
738     VGA_VS <= current_VGA_VS;
739  end process;
740
741 end architecture behavioral;
742
```

Figure 8: VGA.vhd Pt 8

Current State	Input	Next State	Output
A	Pixel Count != 0	A	Pixel Count--
A	Pixel Count == 0	B	16 pixels, Pixel Count = 96
B	Pixel Count != 0	B	Pixel Count--
B	Pixel Count == 0	C	96 Pixels, Pixel Count = 48
C	Pixel Count != 0	C	Pixel Count --
C	Pixel Count == 0	D	48 Pixels, Pixel Count = 640
D	Pixel Count != 0	D	Pixel Count--
D	Pixel Count == 0 && Line Count != 0	A	640 Pixels, Line count++, Pixel Count = 16
D	Pixel Count == 0 && Line Count == 0	A	640 Pixels, Line count--, Pixel Count = 16 Line Count = 525
Any State	Key(0) == '0'	Clear	Reset
Clear	Key(0) == '0'	Clear	No Output
Clear	No input	A	Flag Count = 0
Any State	Key(1) == '0'	Next	Flag Count++
Next	Key(1) == '0'	Next	No output
Next	No input	A	No output

Figure 9: VGA Finite State Machine