```
1   // VGA driver: provides I/O timing and double-buffering for the VGA port.
2
3   module VGA_framebuffer(
4      input logic clk, rst,
5      input logic [9:0] x, // The x coordinate to write to the buffer.
6      input logic [8:0] y, // The y coordinate to write to the buffer.
7      input logic pixel_color, pixel_write, // The data to write (color) and write-enable.
8
9      input logic dfb_en, // Double-Frame Buffer Enable
10
11     output logic frame_start,   // Pulse is fired at the start of a frame.
12
13     // Outputs to the VGA port.
14     output logic [7:0] VGA_R, VGA_G, VGA_B,
15     output logic VGA_CLK, VGA_HS, VGA_VS, VGA_BLANK_N, VGA_SYNC_N
16  );
17
18     /*
19     *
20     * HCOUNT 1599 0              1279       1599 0
21     *             _______________           ________
22     * ___________|    Video      |___________|  Video
23     *
24     *
25     * |SYNC| BP |<-- HACTIVE -->|FP|SYNC| BP |<-- HACTIVE
26     *       _____________________     _____________
27     * |____|       VGA_HS          |____|
28     *
29     */
30
31     // Constants for VGA timing.
32     localparam HPX = 11'd640*2, HFP = 11'd16*2, HSP = 11'd96*2, HBP = 11'd48*2;
33     localparam VLN = 11'd480,   VFP = 10'd11,   VSP = 10'd2,    VBP = 10'd31;
34     localparam HTOTAL = HPX + HFP + HSP + HBP; // 800*2=1600
35     localparam VTOTAL = VLN + VFP + VSP + VBP; // 524
36
37     // Horizontal counter.
38     logic [10:0] h_count;
39     logic end_of_line;
40
41     assign end_of_line = h_count == HTOTAL - 1;
42
43     always_ff @(posedge clk)
44        if (rst) h_count <= 0;
45        else if (end_of_line) h_count <= 0;
46        else h_count <= h_count + 11'd1;
47
48     // Vertical counter & buffer swapping.
49     logic [9:0] v_count;
50     logic end_of_field;
51     logic front_odd; // whether odd address is the front buffer.
52
53     assign end_of_field = v_count == VTOTAL - 1;
54     assign frame_start = !h_count && !v_count;
55
56     always_ff @(posedge clk)
57        if (rst) begin
58           v_count <= 0;
59           front_odd <= 0;
60        end else if (end_of_line)
61           if (end_of_field) begin
62              v_count <= 0;
63              front_odd <= !front_odd;
64           end else
65              v_count <= v_count + 10'd1;
66
67     // Sync signals.
68     assign VGA_CLK = h_count[0]; // 25 MHz clock: pixel latched on rising edge.
69     assign VGA_HS = !(h_count - (HPX + HFP) < HSP);
70     assign VGA_VS = !(v_count - (VLN + VFP) < VSP);
71     assign VGA_SYNC_N = 1; // Unused by VGA
72
73     // Blank area signal.
```

```
74      logic blank;
75      assign blank = h_count >= HPX || v_count >= VLN;
76
77      // Double-buffering.
78      logic buffer[640*480*2-1:0];
79      logic [19:0] wr_addr, rd_addr;
80      logic rd_data;
81
82      assign wr_addr = {y * 19'd640 + x, (!front_odd & dfb_en)};
83      assign rd_addr = {v_count * 19'd640 + (h_count / 19'd2), (front_odd & dfb_en)};
84
85      always_ff @(posedge clk) begin
86         if (pixel_write) buffer[wr_addr] <= pixel_color;
87         if (VGA_CLK) begin
88            rd_data <= buffer[rd_addr];
89            VGA_BLANK_N <= ~blank;
90         end
91      end
92
93      // Color output.
94      assign {VGA_R, VGA_G, VGA_B} = rd_data ? 24'hFFFFFF : 24'h000000;
95   endmodule
96
```