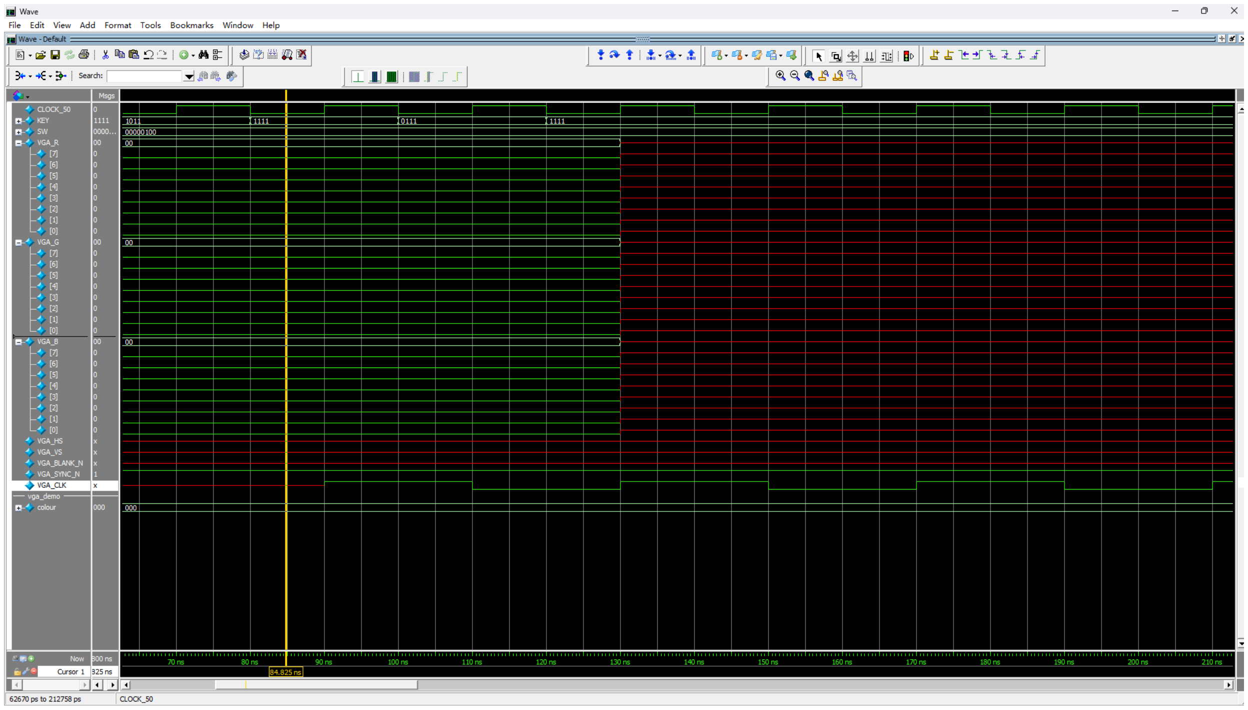
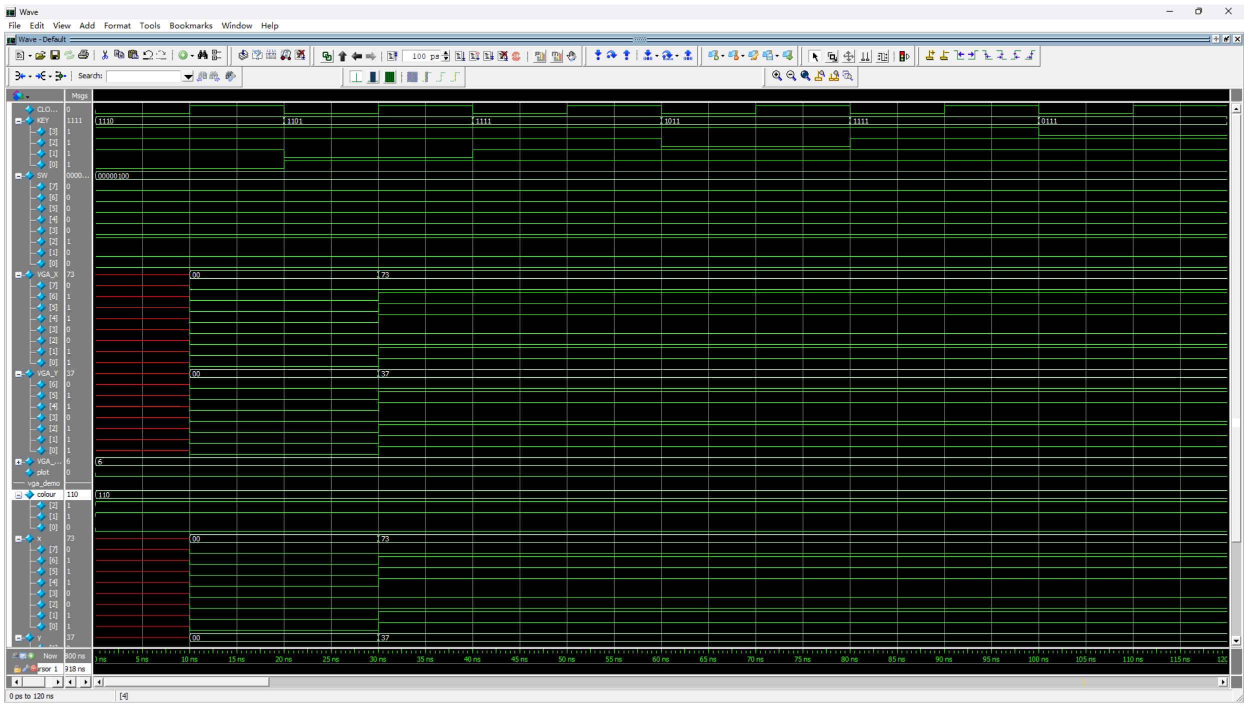
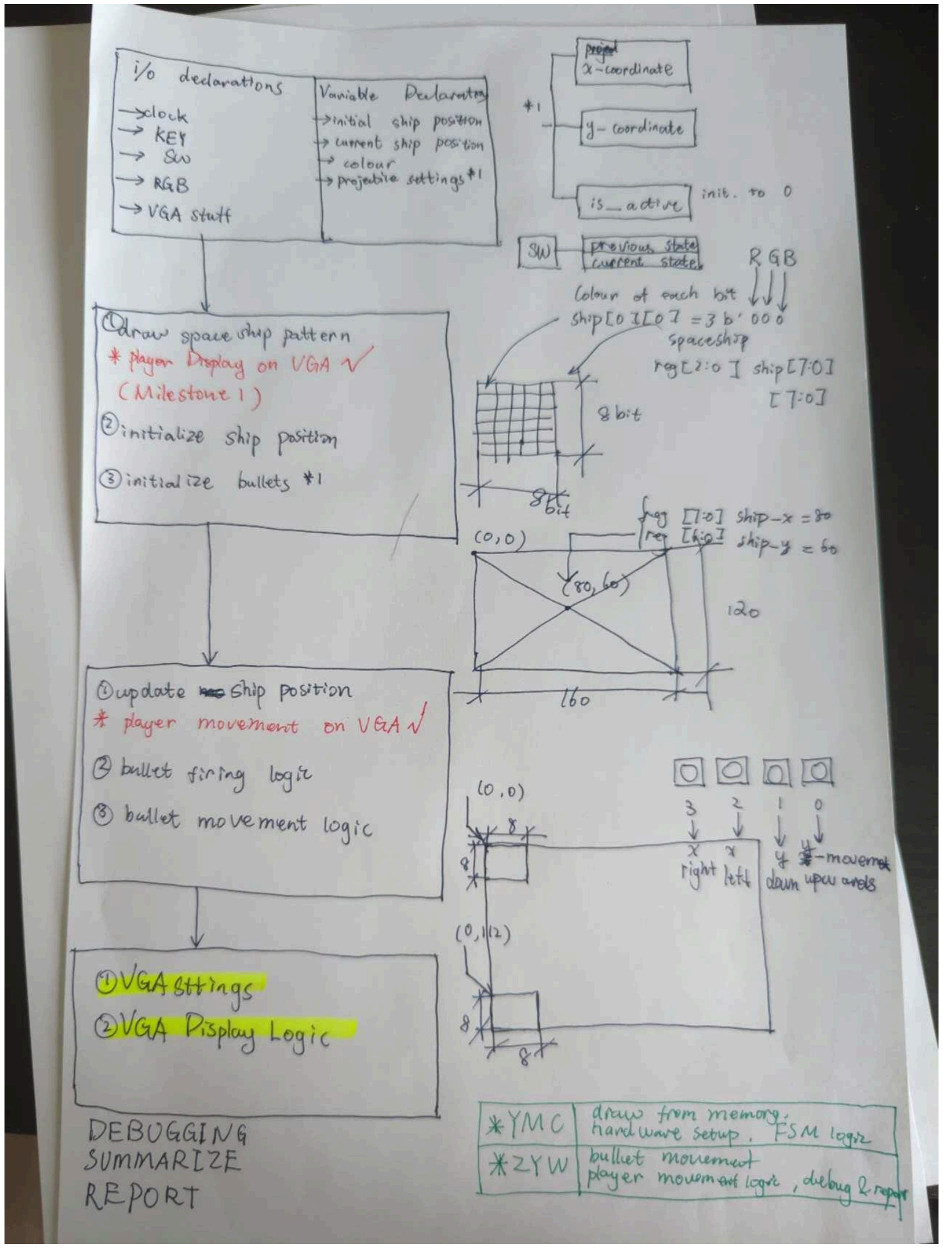
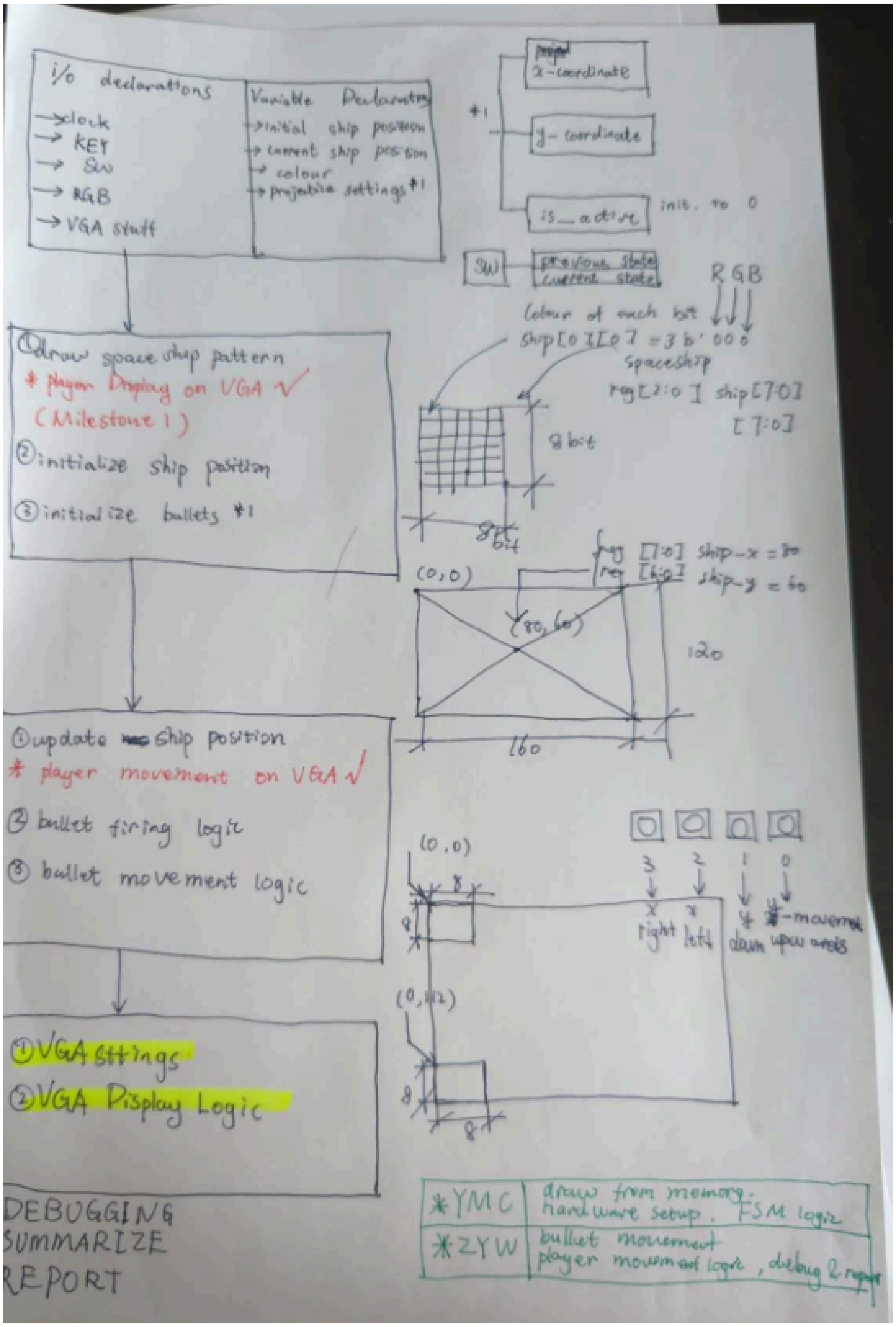

Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Msgs
CLOCK_50 0
KEY 1111
SW 0000...
VGA_R 00
VGA_G 00
VGA_B 00
VGA_HS x
VGA_VS x
VGA_BLANK_N x
VGA_SYNC_N 1
VGA_CLK x
vga_demo
colour 000
1011
1111
0111
00000100
Now 800 ns
Cursor 1 325 ns
84.825 ns
62670 ps to 212758 ps
CLOCK_50


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Msgs
KEY 1111
SW 0000...
VGA_X 73
VGA_Y 37
plot 0
vga_demo
colour 110
x 73
y 37
1110
1101
1111
1011
0111
00000100
00
73
37
Now 800 ns
Cursor 1 918 ns
0 ps to 120 ns
[4]

[Photo]



i/o declarations
→ clock
→ KEY
→ SW
→ RGB
→ VGA stuff
Variable Declarations
→ initial ship position
→ current ship position
→ colour
→ projectile settings *1
*1
x-coordinate
y-coordinate
is_active
init. to 0
SW
previous state
current state
RGB
colour of each bit
ship[0][0] = 3'b'000
spaceship
reg[2:0] ship[7:0]
[7:0]
8 bit
8 bit
① draw space ship pattern
* player Display on VGA ✓
(Milestone 1)
② initialize ship position
③ initialize bullets *1
(0,0)
reg [7:0] ship_x = 80
reg [6:0] ship_y = 60
(80,60)
120
160
① update ship position
* player movement on VGA ✓
② bullet firing logic
③ bullet movement logic
(0,0)
8
8
(0,112)
8
8
3
2
1
0
x
x
y
y-movement
right
left
down
up
① VGA settings
② VGA Display Logic
DEBUGGING
SUMMARIZE
REPORT
*YMC
draw from memory, hardware setup, FSM logic
*ZYW
bullet movement
player movement logic, debug & report



i/o declarations
- → clock
- → KEY
- → SW
- → RGB
- → VGA stuff

Variable Declarations
- → initial ship position
- → current ship position
- → colour
- → projectile settings *1

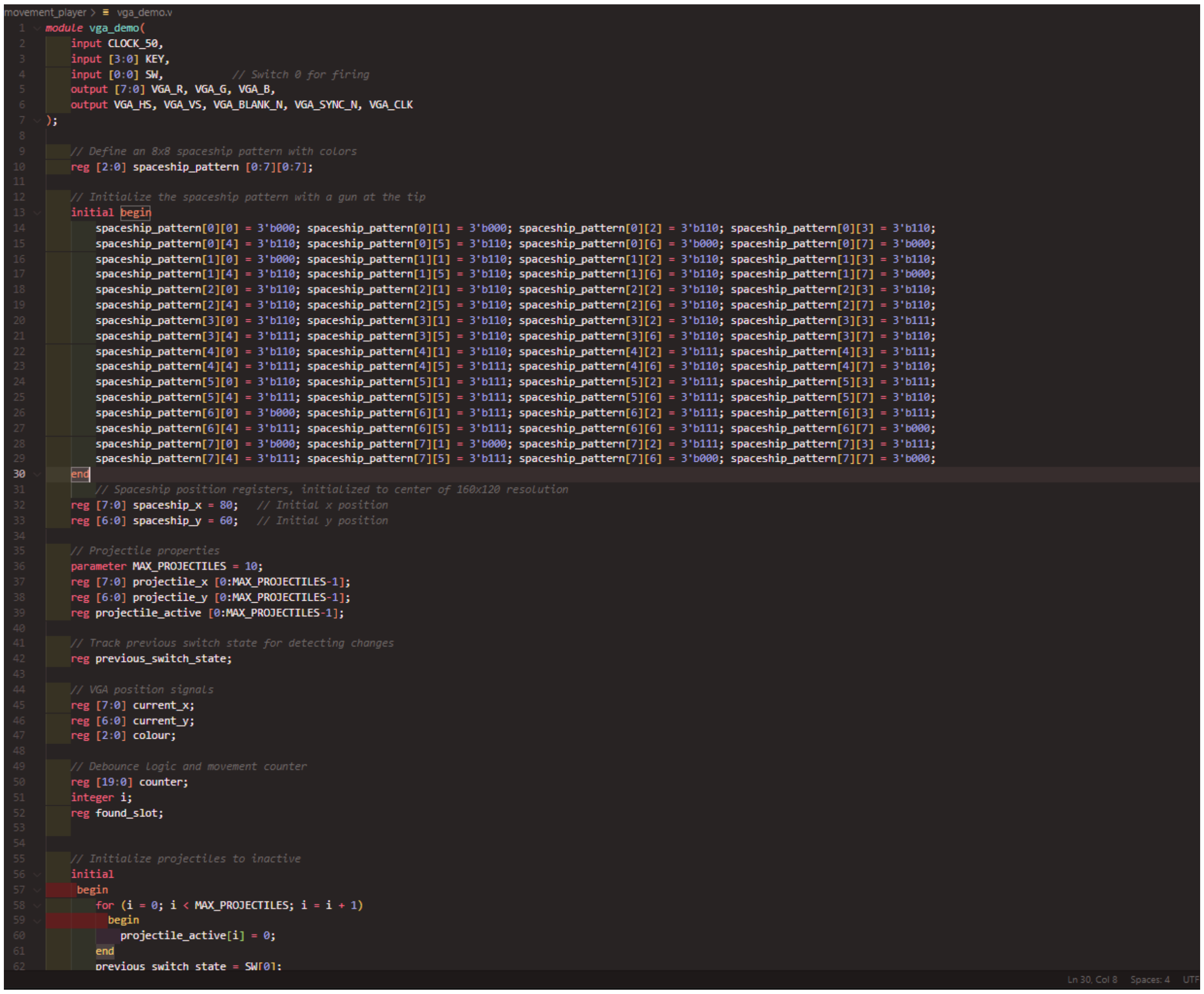
① draw space ship pattern
* player Display on VGA ✓
(Milestone 1)
② initialize ship position
③ initialize bullets *1

① update ~~[illegible]~~ ship position
* player movement on VGA ✓
② bullet firing logic
③ bullet movement logic

① VGA settings
② VGA Display Logic

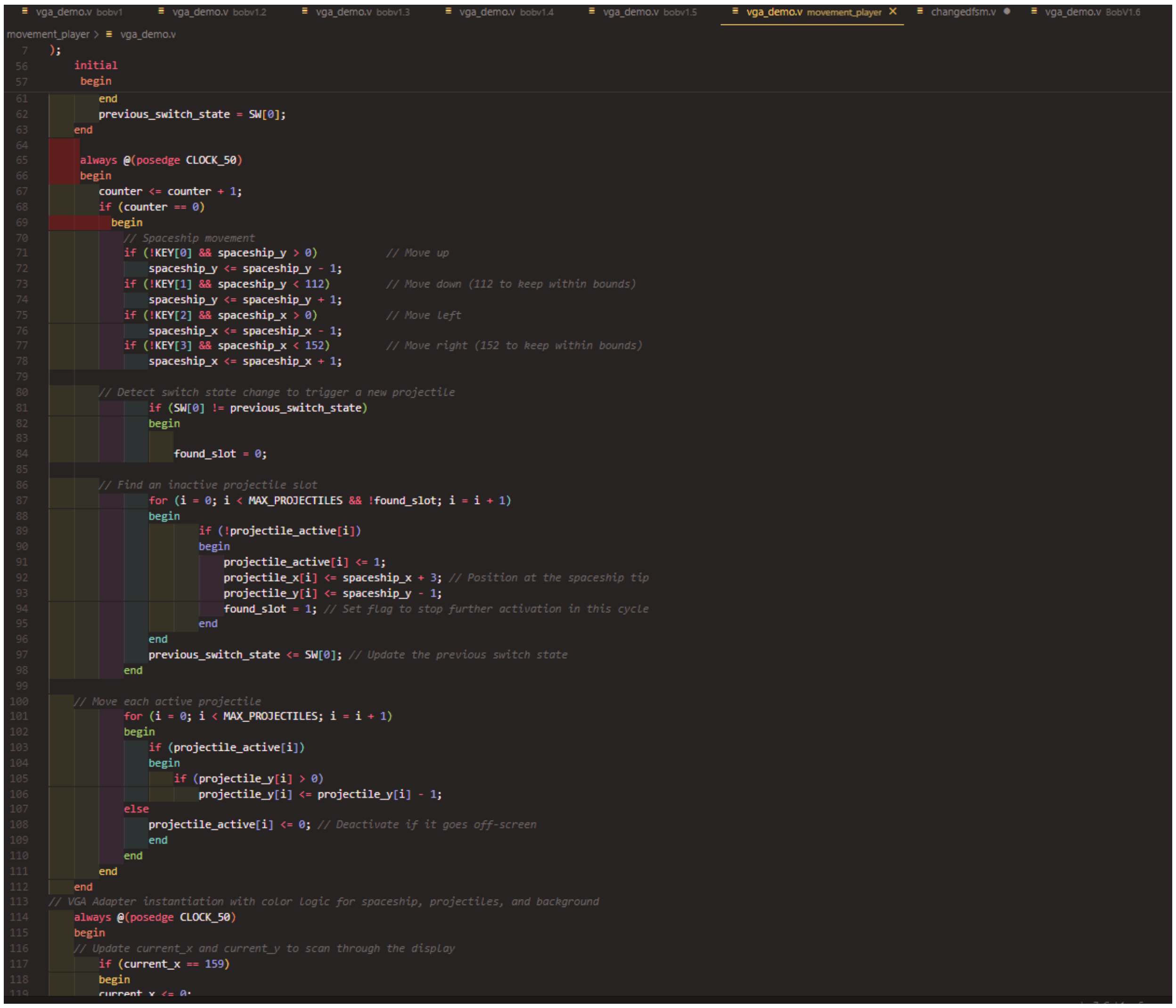
DEBUGGING
SUMMARIZE
REPORT

*YMC	draw from memory. hardware setup. FSM logic
*ZYW	bullet movement player movement logic, debug & report

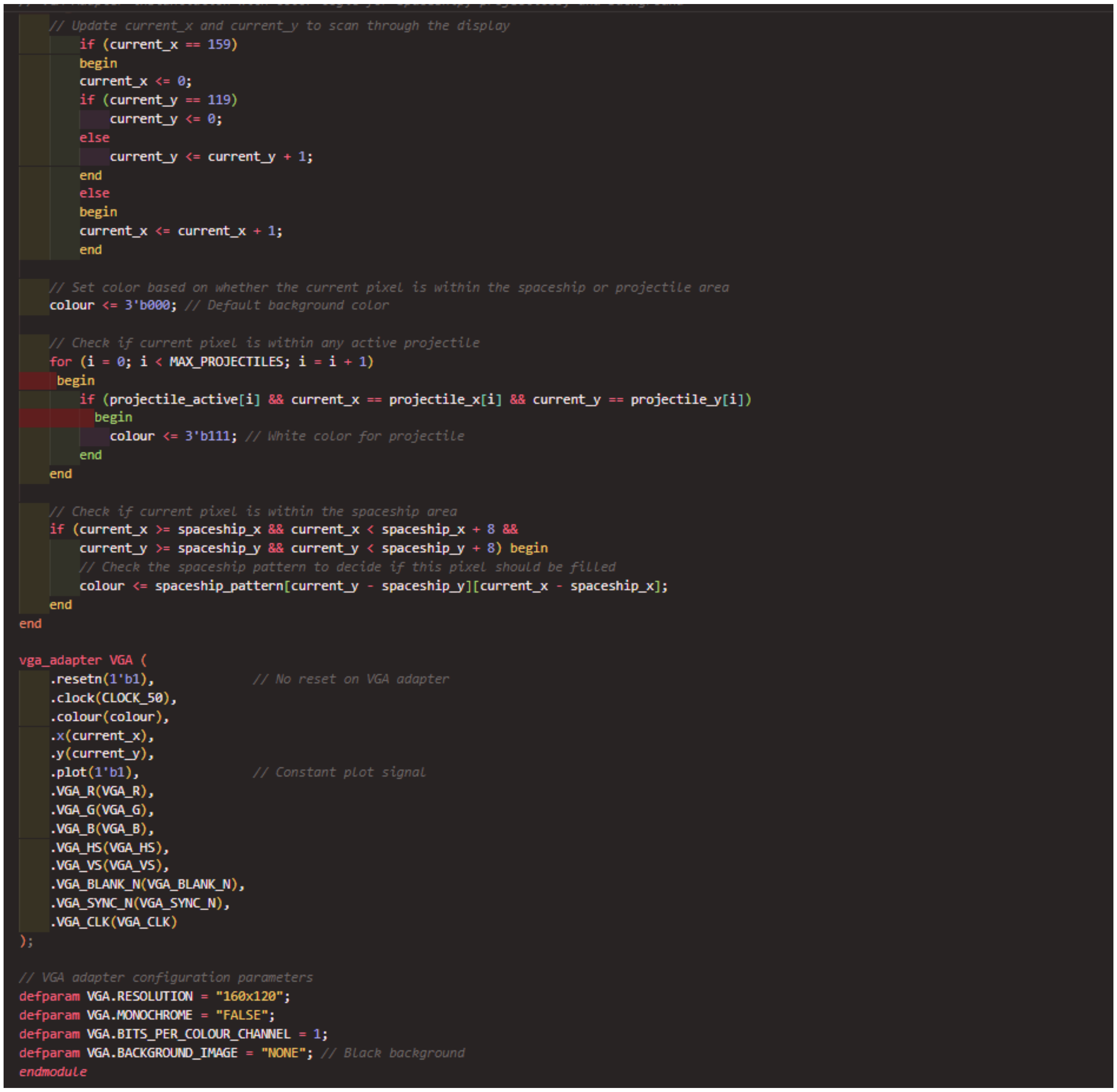

movement_player > vga_demo.v

```verilog
module vga_demo(
    input CLOCK_50,
    input [3:0] KEY,
    input [0:0] SW,           // Switch 0 for firing
    output [7:0] VGA_R, VGA_G, VGA_B,
    output VGA_HS, VGA_VS, VGA_BLANK_N, VGA_SYNC_N, VGA_CLK
);

    // Define an 8x8 spaceship pattern with colors
    reg [2:0] spaceship_pattern [0:7][0:7];

    // Initialize the spaceship pattern with a gun at the tip
    initial begin
        spaceship_pattern[0][0] = 3'b000; spaceship_pattern[0][1] = 3'b000; spaceship_pattern[0][2] = 3'b110; spaceship_pattern[0][3] = 3'b110;
        spaceship_pattern[0][4] = 3'b110; spaceship_pattern[0][5] = 3'b110; spaceship_pattern[0][6] = 3'b000; spaceship_pattern[0][7] = 3'b000;
        spaceship_pattern[1][0] = 3'b000; spaceship_pattern[1][1] = 3'b110; spaceship_pattern[1][2] = 3'b110; spaceship_pattern[1][3] = 3'b110;
        spaceship_pattern[1][4] = 3'b110; spaceship_pattern[1][5] = 3'b110; spaceship_pattern[1][6] = 3'b110; spaceship_pattern[1][7] = 3'b000;
        spaceship_pattern[2][0] = 3'b110; spaceship_pattern[2][1] = 3'b110; spaceship_pattern[2][2] = 3'b110; spaceship_pattern[2][3] = 3'b110;
        spaceship_pattern[2][4] = 3'b110; spaceship_pattern[2][5] = 3'b110; spaceship_pattern[2][6] = 3'b110; spaceship_pattern[2][7] = 3'b110;
        spaceship_pattern[3][0] = 3'b110; spaceship_pattern[3][1] = 3'b110; spaceship_pattern[3][2] = 3'b110; spaceship_pattern[3][3] = 3'b111;
        spaceship_pattern[3][4] = 3'b111; spaceship_pattern[3][5] = 3'b110; spaceship_pattern[3][6] = 3'b110; spaceship_pattern[3][7] = 3'b110;
        spaceship_pattern[4][0] = 3'b110; spaceship_pattern[4][1] = 3'b110; spaceship_pattern[4][2] = 3'b111; spaceship_pattern[4][3] = 3'b111;
        spaceship_pattern[4][4] = 3'b111; spaceship_pattern[4][5] = 3'b111; spaceship_pattern[4][6] = 3'b110; spaceship_pattern[4][7] = 3'b110;
        spaceship_pattern[5][0] = 3'b110; spaceship_pattern[5][1] = 3'b111; spaceship_pattern[5][2] = 3'b111; spaceship_pattern[5][3] = 3'b111;
        spaceship_pattern[5][4] = 3'b111; spaceship_pattern[5][5] = 3'b111; spaceship_pattern[5][6] = 3'b111; spaceship_pattern[5][7] = 3'b110;
        spaceship_pattern[6][0] = 3'b000; spaceship_pattern[6][1] = 3'b111; spaceship_pattern[6][2] = 3'b111; spaceship_pattern[6][3] = 3'b111;
        spaceship_pattern[6][4] = 3'b111; spaceship_pattern[6][5] = 3'b111; spaceship_pattern[6][6] = 3'b111; spaceship_pattern[6][7] = 3'b000;
        spaceship_pattern[7][0] = 3'b000; spaceship_pattern[7][1] = 3'b000; spaceship_pattern[7][2] = 3'b111; spaceship_pattern[7][3] = 3'b111;
        spaceship_pattern[7][4] = 3'b111; spaceship_pattern[7][5] = 3'b111; spaceship_pattern[7][6] = 3'b000; spaceship_pattern[7][7] = 3'b000;
    end
        // Spaceship position registers, initialized to center of 160x120 resolution
    reg [7:0] spaceship_x = 80;   // Initial x position
    reg [6:0] spaceship_y = 60;   // Initial y position

    // Projectile properties
    parameter MAX_PROJECTILES = 10;
    reg [7:0] projectile_x [0:MAX_PROJECTILES-1];
    reg [6:0] projectile_y [0:MAX_PROJECTILES-1];
    reg projectile_active [0:MAX_PROJECTILES-1];

    // Track previous switch state for detecting changes
    reg previous_switch_state;

    // VGA position signals
    reg [7:0] current_x;
    reg [6:0] current_y;
    reg [2:0] colour;

    // Debounce logic and movement counter
    reg [19:0] counter;
    integer i;
    reg found_slot;


    // Initialize projectiles to inactive
    initial
     begin
        for (i = 0; i < MAX_PROJECTILES; i = i + 1)
          begin
            projectile_active[i] = 0;
        end
        previous_switch_state = SW[0];
```

V



```
);
    initial
     begin
        end
        previous_switch_state = SW[0];
    end

     always @(posedge CLOCK_50)
     begin
        counter <= counter + 1;
        if (counter == 0)
          begin
            // Spaceship movement
            if (!KEY[0] && spaceship_y > 0)          // Move up
                spaceship_y <= spaceship_y - 1;
            if (!KEY[1] && spaceship_y < 112)        // Move down (112 to keep within bounds)
                spaceship_y <= spaceship_y + 1;
            if (!KEY[2] && spaceship_x > 0)          // Move left
                spaceship_x <= spaceship_x - 1;
            if (!KEY[3] && spaceship_x < 152)        // Move right (152 to keep within bounds)
                spaceship_x <= spaceship_x + 1;

        // Detect switch state change to trigger a new projectile
                if (SW[0] != previous_switch_state)
                begin

                    found_slot = 0;

        // Find an inactive projectile slot
                for (i = 0; i < MAX_PROJECTILES && !found_slot; i = i + 1)
                begin
                        if (!projectile_active[i])
                        begin
                            projectile_active[i] <= 1;
                            projectile_x[i] <= spaceship_x + 3; // Position at the spaceship tip
                            projectile_y[i] <= spaceship_y - 1;
                            found_slot = 1; // Set flag to stop further activation in this cycle
                        end
                end
                previous_switch_state <= SW[0]; // Update the previous switch state
            end

    // Move each active projectile
            for (i = 0; i < MAX_PROJECTILES; i = i + 1)
            begin
                if (projectile_active[i])
                begin
                    if (projectile_y[i] > 0)
                        projectile_y[i] <= projectile_y[i] - 1;
            else
                projectile_active[i] <= 0; // Deactivate if it goes off-screen
                end
            end
        end
    end
// VGA Adapter instantiation with color logic for spaceship, projectiles, and background
    always @(posedge CLOCK_50)
    begin
    // Update current_x and current_y to scan through the display
        if (current_x == 159)
        begin
        current_x <= 0;
```



```
    // Update current_x and current_y to scan through the display
        if (current_x == 159)
        begin
        current_x <= 0;
        if (current_y == 119)
            current_y <= 0;
        else
            current_y <= current_y + 1;
        end
        else
        begin
        current_x <= current_x + 1;
        end

    // Set color based on whether the current pixel is within the spaceship or projectile area
    colour <= 3'b000; // Default background color

    // Check if current pixel is within any active projectile
    for (i = 0; i < MAX_PROJECTILES; i = i + 1)
     begin
        if (projectile_active[i] && current_x == projectile_x[i] && current_y == projectile_y[i])
          begin
            colour <= 3'b111; // White color for projectile
        end
    end

    // Check if current pixel is within the spaceship area
    if (current_x >= spaceship_x && current_x < spaceship_x + 8 &&
        current_y >= spaceship_y && current_y < spaceship_y + 8) begin
        // Check the spaceship pattern to decide if this pixel should be filled
        colour <= spaceship_pattern[current_y - spaceship_y][current_x - spaceship_x];
    end
end

vga_adapter VGA (
    .resetn(1'b1),            // No reset on VGA adapter
    .clock(CLOCK_50),
    .colour(colour),
    .x(current_x),
    .y(current_y),
    .plot(1'b1),              // Constant plot signal
    .VGA_R(VGA_R),
    .VGA_G(VGA_G),
    .VGA_B(VGA_B),
    .VGA_HS(VGA_HS),
    .VGA_VS(VGA_VS),
    .VGA_BLANK_N(VGA_BLANK_N),
    .VGA_SYNC_N(VGA_SYNC_N),
    .VGA_CLK(VGA_CLK)
);

// VGA adapter configuration parameters
defparam VGA.RESOLUTION = "160x120";
defparam VGA.MONOCHROME = "FALSE";
defparam VGA.BITS_PER_COLOUR_CHANNEL = 1;
defparam VGA.BACKGROUND_IMAGE = "NONE"; // Black background
endmodule
```