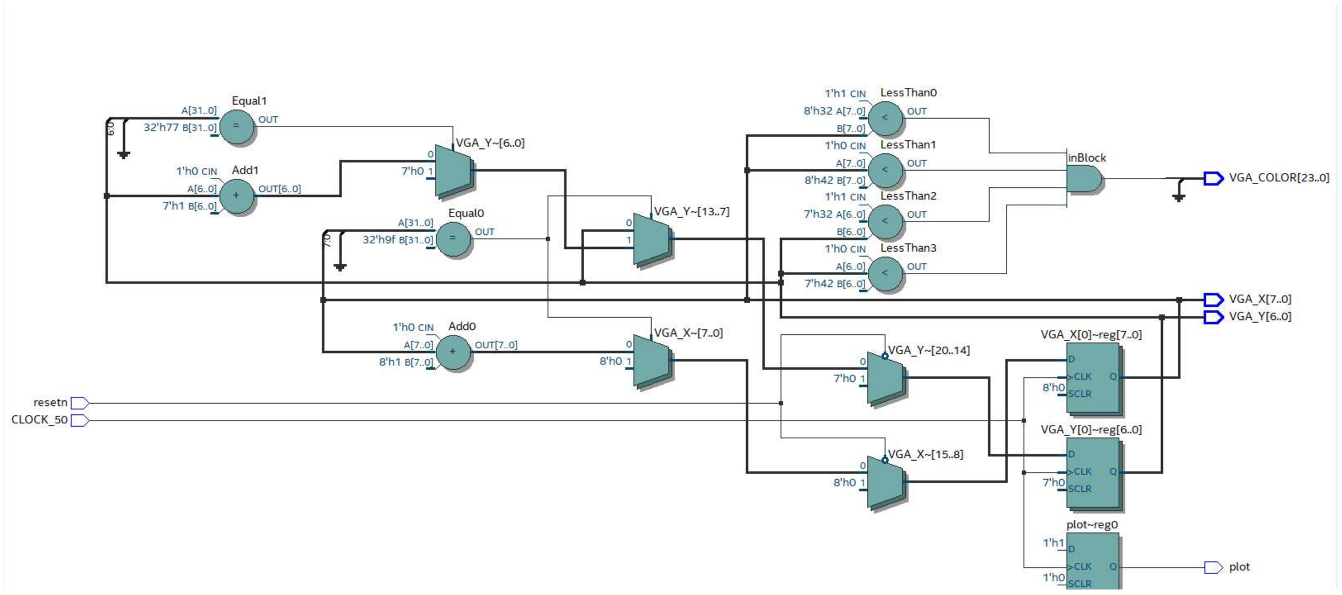
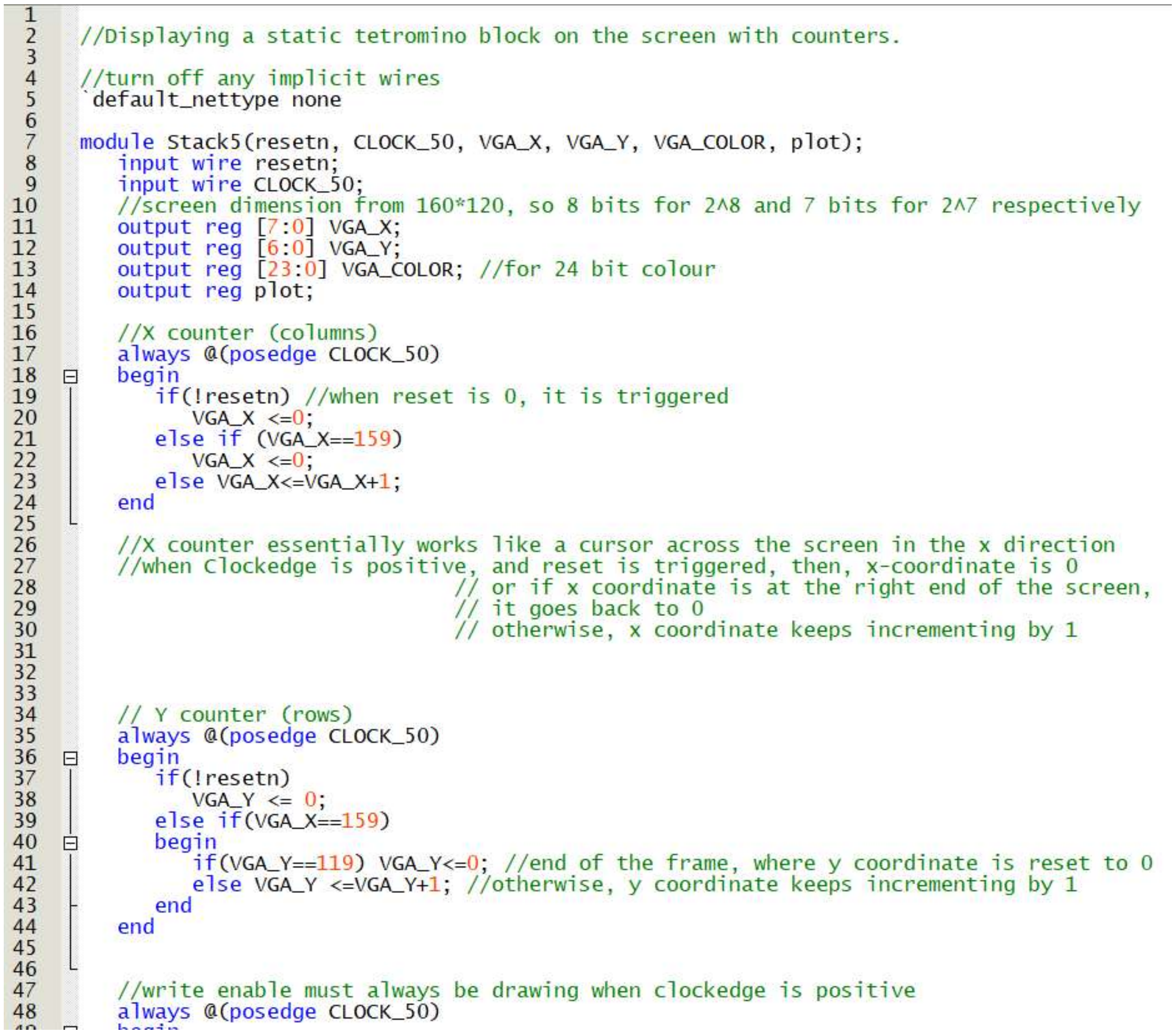
Niranjana:

1) VGA Synchronous (signals update triggered by positive Clockedge) Timing Signals and Colour Output

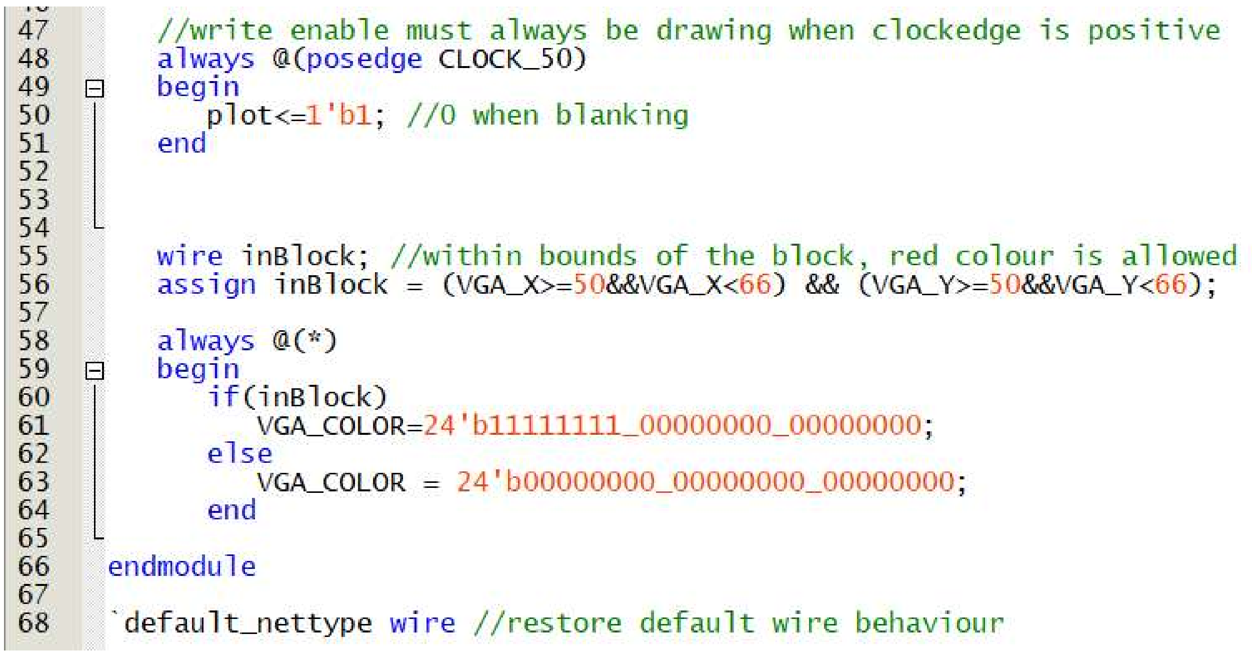
Diagram:



Code:

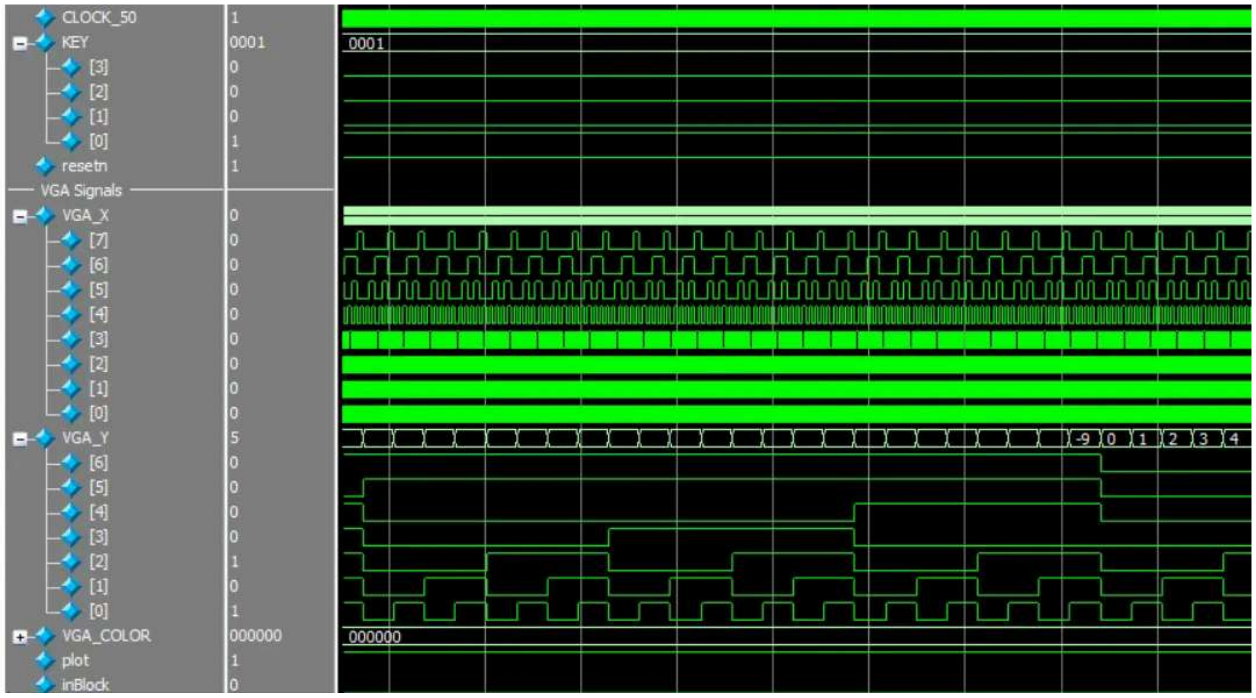


```verilog

//Displaying a static tetromino block on the screen with counters.

//turn off any implicit wires
`default_nettype none

module Stack5(resetn, CLOCK_50, VGA_X, VGA_Y, VGA_COLOR, plot);
   input wire resetn;
   input wire CLOCK_50;
   //screen dimension from 160*120, so 8 bits for 2^8 and 7 bits for 2^7 respectively
   output reg [7:0] VGA_X;
   output reg [6:0] VGA_Y;
   output reg [23:0] VGA_COLOR; //for 24 bit colour
   output reg plot;

   //X counter (columns)
   always @(posedge CLOCK_50)
   begin
      if(!resetn) //when reset is 0, it is triggered
         VGA_X <=0;
      else if (VGA_X==159)
         VGA_X <=0;
      else VGA_X<=VGA_X+1;
   end

   //X counter essentially works like a cursor across the screen in the x direction
   //when Clockedge is positive, and reset is triggered, then, x-coordinate is 0
                              // or if x coordinate is at the right end of the screen,
                              // it goes back to 0
                              // otherwise, x coordinate keeps incrementing by 1



   // Y counter (rows)
   always @(posedge CLOCK_50)
   begin
      if(!resetn)
         VGA_Y <= 0;
      else if(VGA_X==159)
      begin
         if(VGA_Y==119) VGA_Y<=0; //end of the frame, where y coordinate is reset to 0
         else VGA_Y <=VGA_Y+1; //otherwise, y coordinate keeps incrementing by 1
      end
   end


   //write enable must always be drawing when clockedge is positive
   always @(posedge CLOCK_50)
```



```
    //write enable must always be drawing when clockedge is positive
    always @(posedge CLOCK_50)
    begin
       plot<=1'b1; //0 when blanking
    end



    wire inBlock; //within bounds of the block, red colour is allowed
    assign inBlock = (VGA_X>=50&&VGA_X<66) && (VGA_Y>=50&&VGA_Y<66);

    always @(*)
    begin
       if(inBlock)
          VGA_COLOR=24'b11111111_00000000_00000000;
       else
          VGA_COLOR = 24'b00000000_00000000_00000000;
       end

 endmodule

 `default_nettype wire //restore default wire behaviour
```

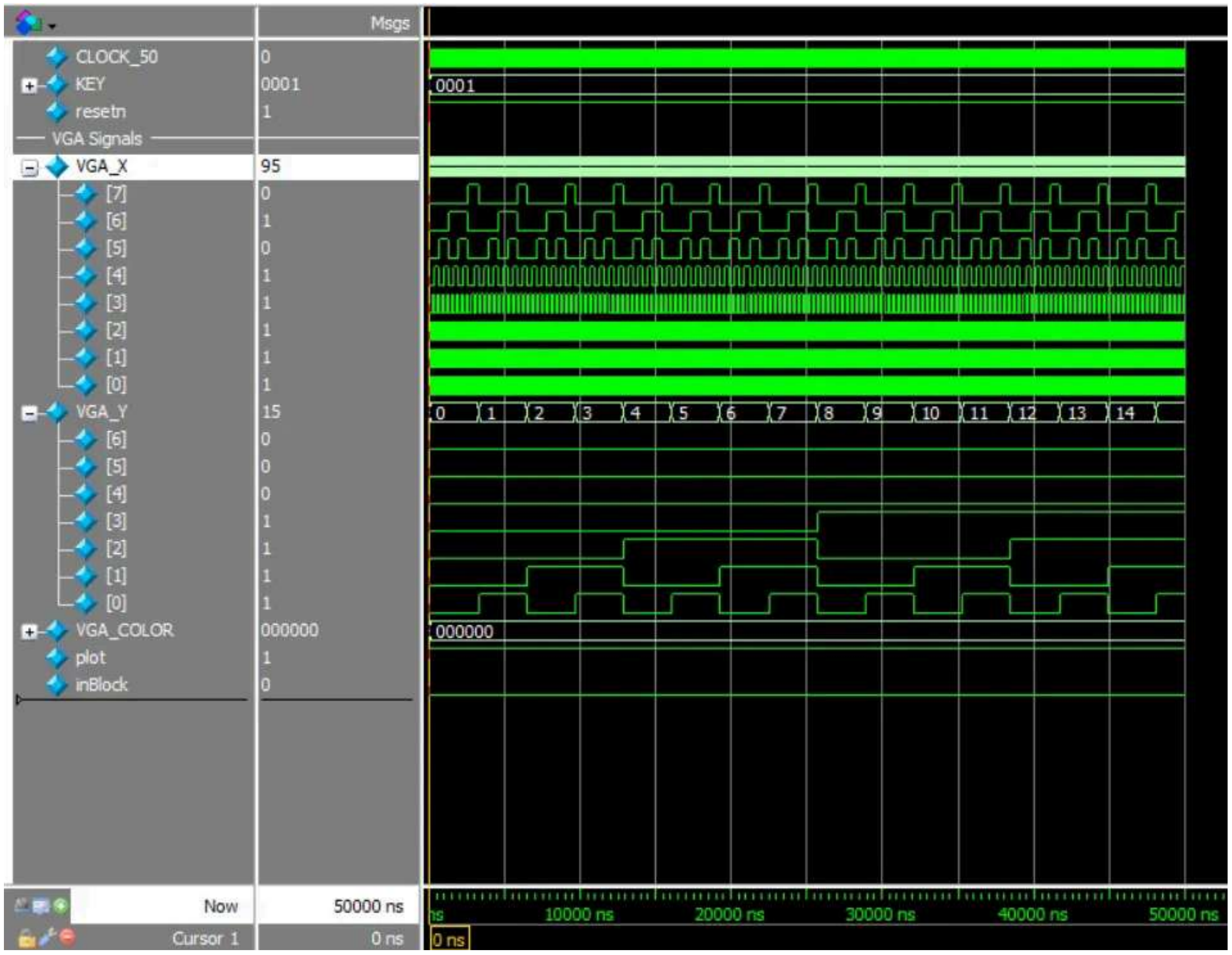
ModelSim:



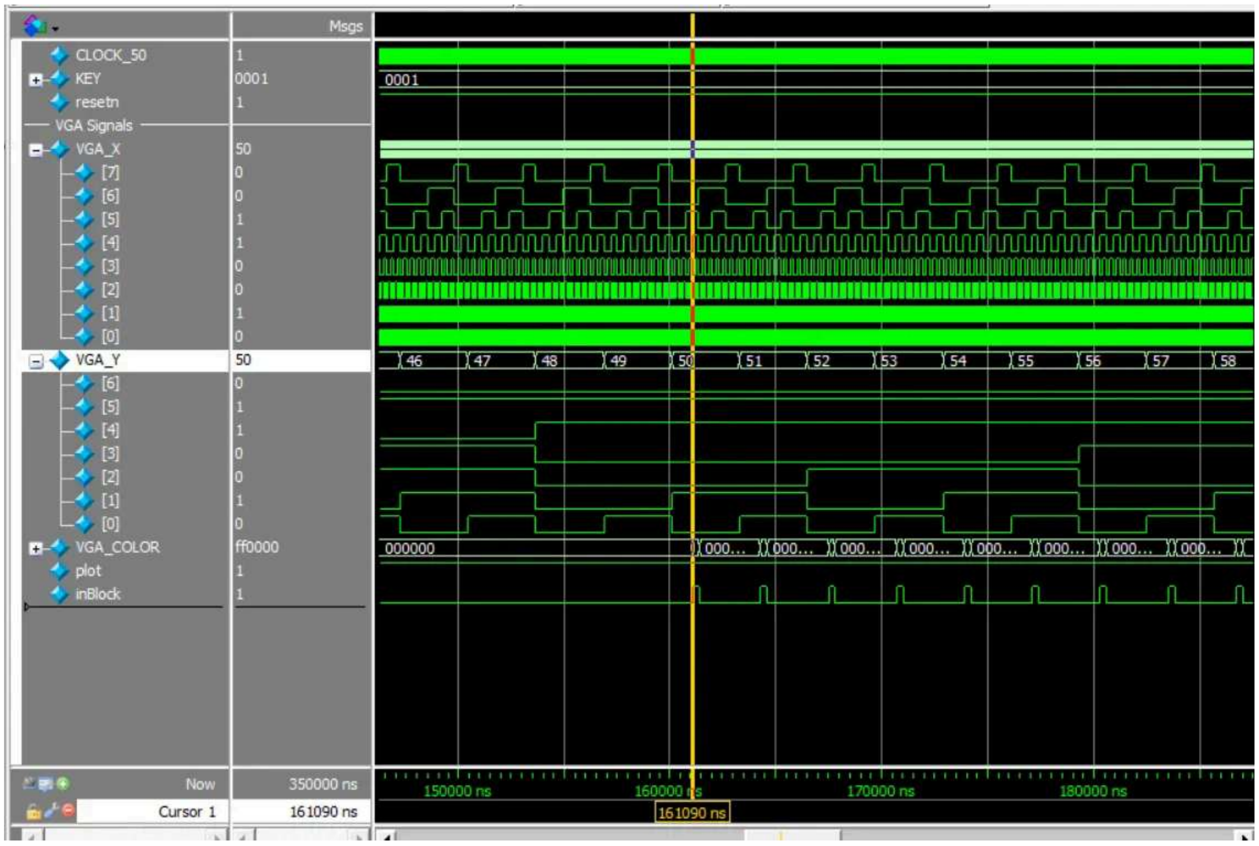
```
CLOCK_50     1
KEY          0001
  [3]        0
  [2]        0
  [1]        0
  [0]        1
resetn       1
VGA Signals
VGA_X        0
  [7]        0
  [6]        0
  [5]        0
  [4]        0
  [3]        0
  [2]        0
  [1]        0
  [0]        0
VGA_Y        5
  [6]        0
  [5]        0
  [4]        0
  [3]        0
  [2]        1
  [1]        0
  [0]        1
VGA_COLOR    000000
plot         1
inBlock      0
```

Clock toggles every cycle, when reset is not triggered (not on active low), circuit is running. VGA_X is incrementing through values, where the 8 bits are toggling in a binary counting pattern from 0 - 159. This is how the X cursor scans across the screen. Whereas, VGA_Y only increments each time VGA_X completes a full scan of reaching a count of 159. The colour of the VGA is currently black because we are not in the area of the red block, where x=50-65 and y =

50-65, which is when colour should be FF0000 (red); inBlock=0 so it is still black. Plot is always 1 as it is our write enable which is always drawing.



Here, VGA_X = 95, at column 95. Individual bits toggle in a binary pattern, and at each clock cycle, are incrementing. VGA_Y = 15, at row 15. The bit pattern shows that through bits 0-4, VGA_Y has incremented up to 15, as each step is when VGA_X completes a row (hitting a count of 159).



	Msgs
CLOCK_50	1
KEY	0001
resetn	1
VGA Signals	
VGA_X	50
[7]	0
[6]	0
[5]	1
[4]	1
[3]	0
[2]	0
[1]	1
[0]	0
VGA_Y	50
[6]	0
[5]	1
[4]	1
[3]	0
[2]	0
[1]	1
[0]	0
VGA_COLOR	ff0000
plot	1
inBlock	1
Now	350000 ns
Cursor 1	161090 ns

Current decimal value of VGA_X is 50, within the thresholds of the drawn box along with VGA_Y, the bits have counted to 00110010 from bit 7 to 0, indicating decimal number 50. As the column count has reached its maximum, row count VGA_Y begins incrementing (is currently at 50 as well). It remains flat during one complete row scan, but increments to 51 when VGA_X reaches 159.

Can also observe here that the current VGA_COLOR value is ff0000 (hex) = RED as per RGB (11111111_00000000_00000000; R = 255). This is because both VGA_X and VGA_Y are within the prescribed thresholds of 50-65, so combinational logic sets the color to red when inBlock = 1.