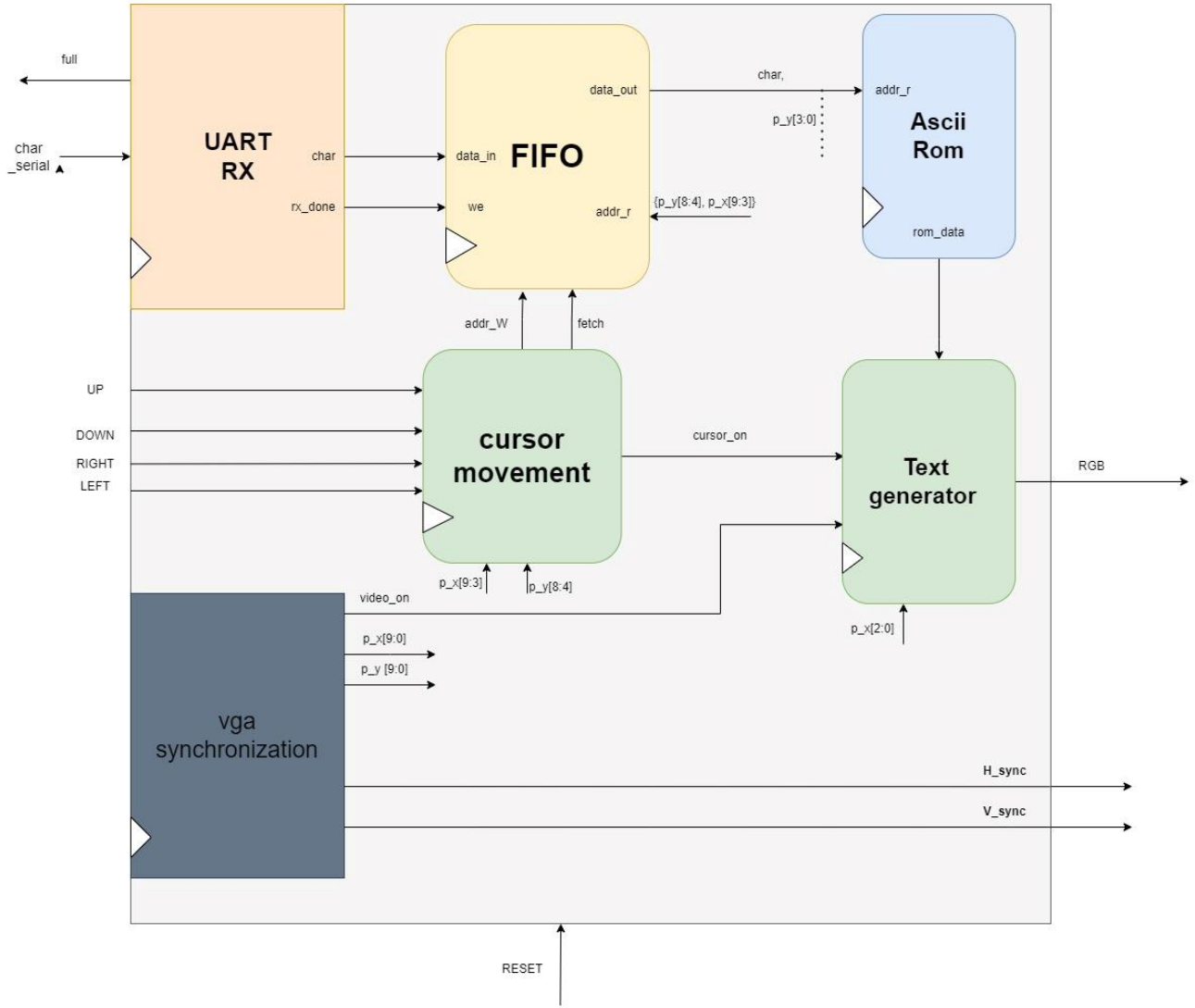
I

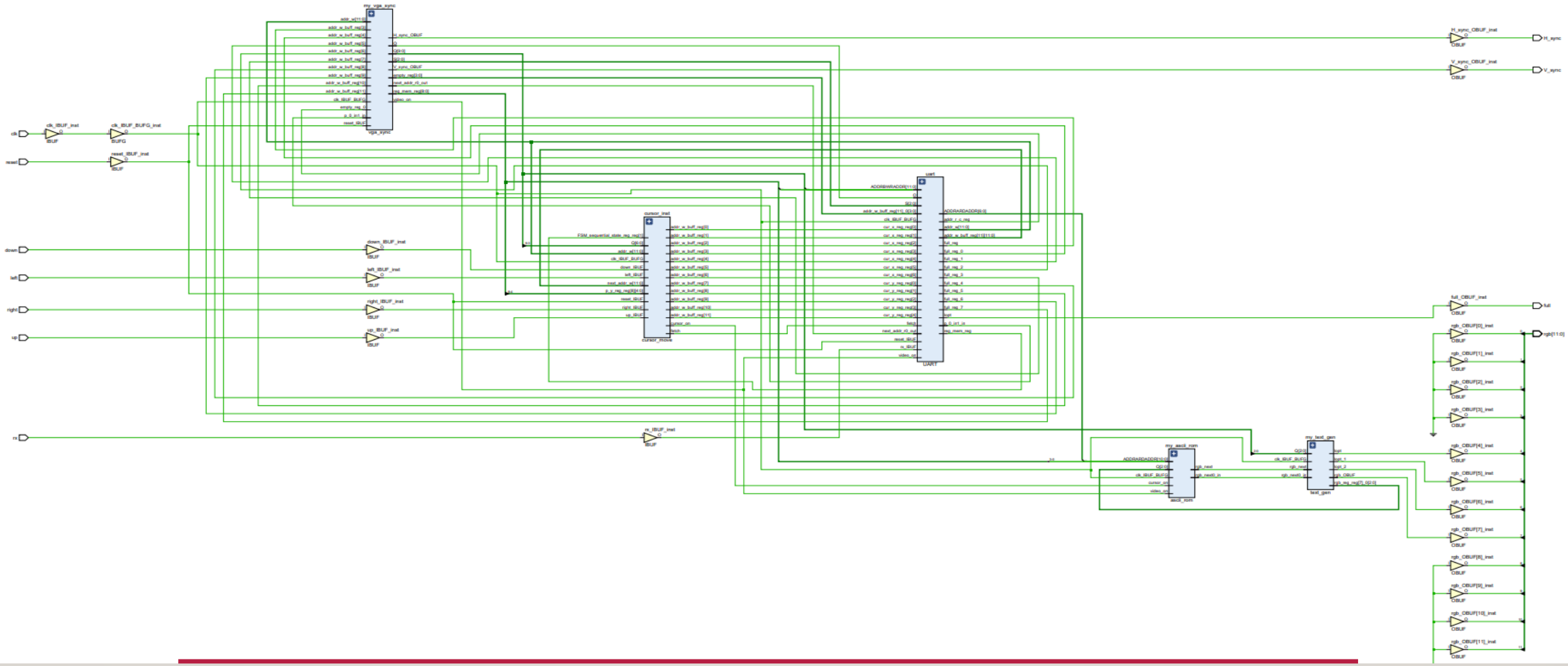
SERIAL-VGA TEXT EDITOR

serial -vga text editor



PROJECT OVERVIEW

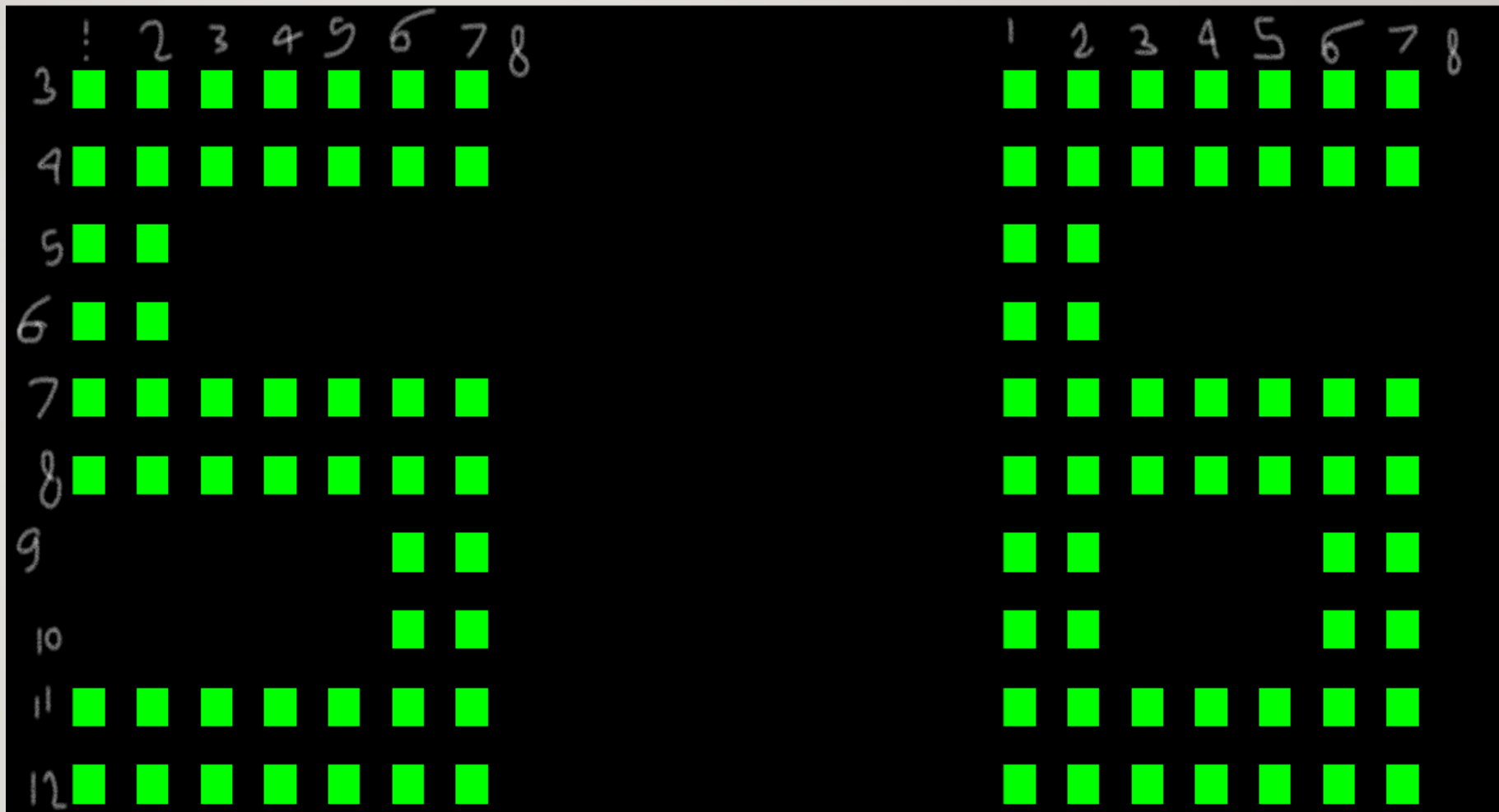
- The VGA Text Editor is a system that can display text on a VGA monitor, using an ASCII character set to represent the tiles on the screen.
- The system includes a VGA Controller module that controls the position of a cursor on the screen and generates VGA video output.
- The design is optimized for efficient memory usage and hardware-specific features to improve performance and efficiency, while maintaining the desired quality and flexibility in the display. Saved 84% of the memory
- The design is optimized for efficient data transfer

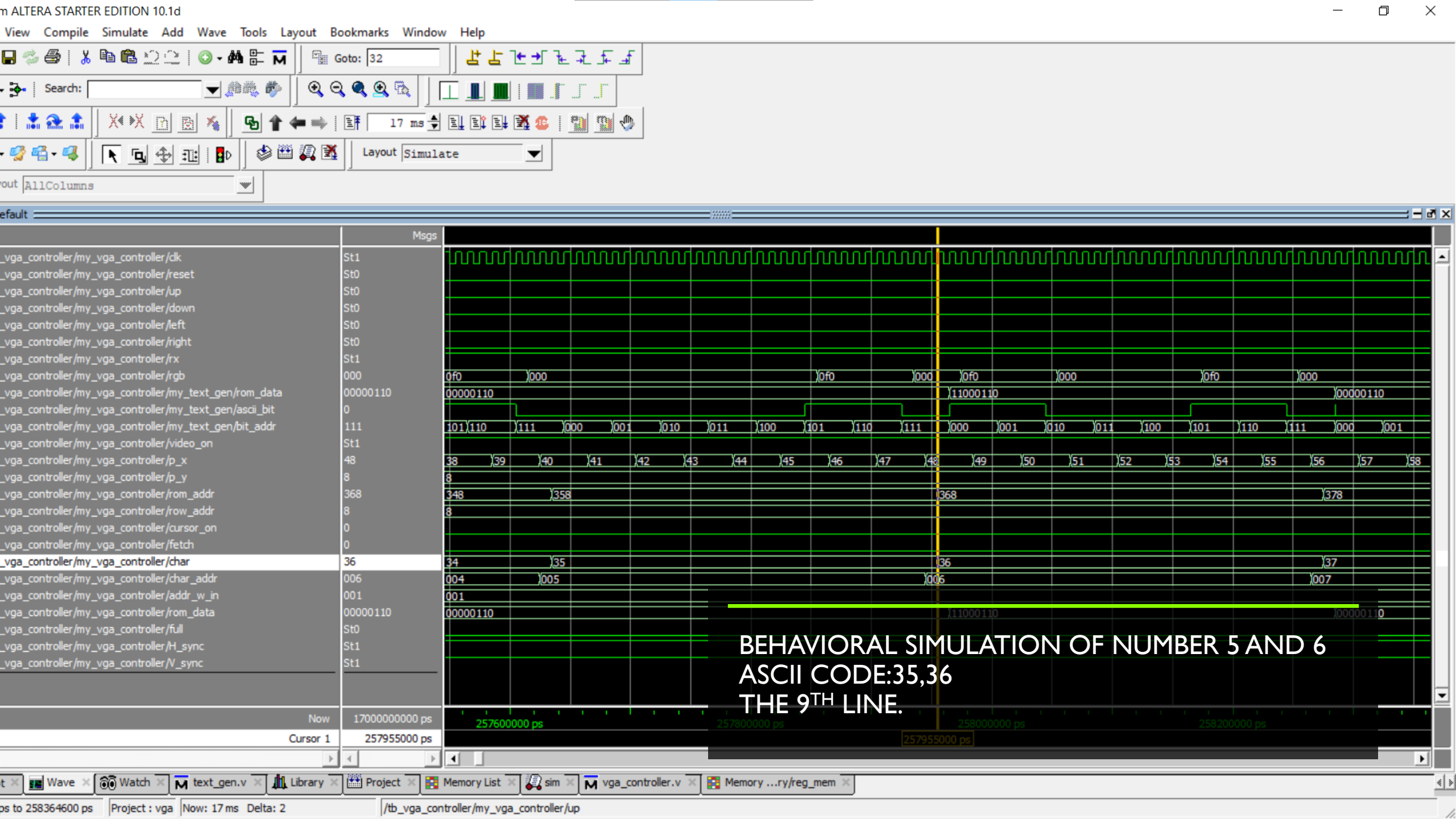


THE SCHEMATIC OF IMPLEMENTATION

EXAMPLE OF 5 AND 6 BITMAP

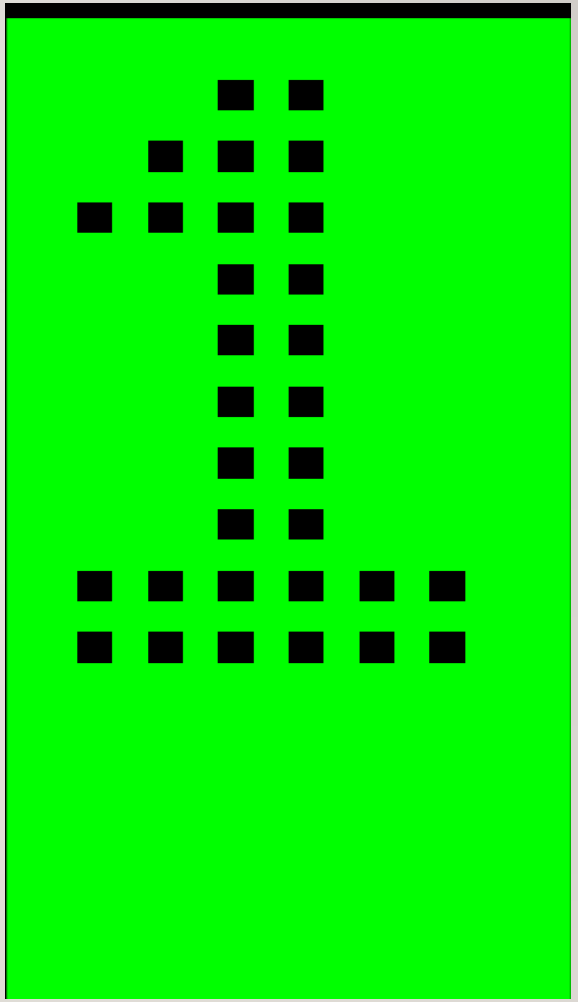
- First 2 rows and last column is always 0 to maintain the boundary of the character



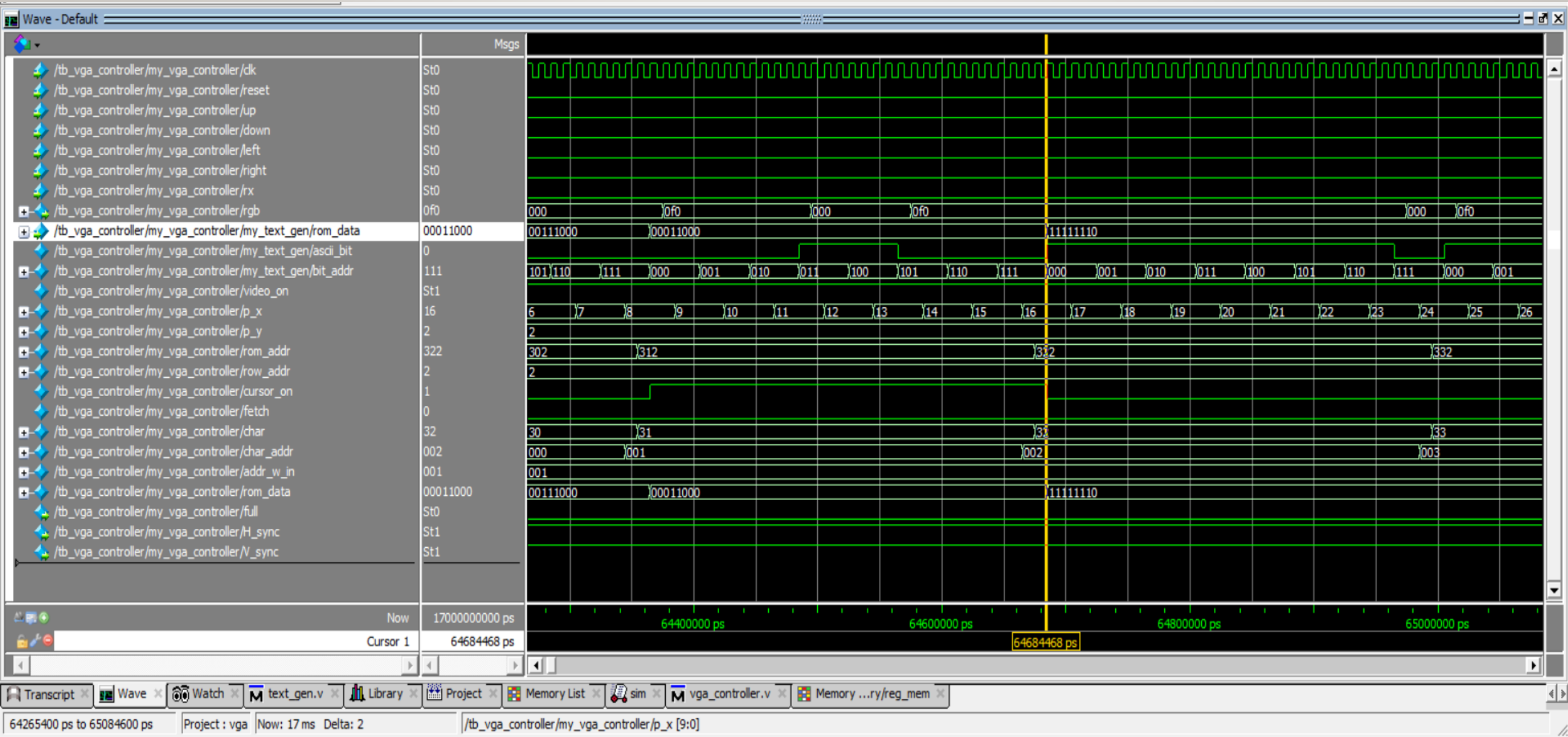

m ALTERA STARTER EDITION 10.1d
View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Goto: 32
Search:
17 ms
Layout Simulate
AllColumns
Msgs
_vga_controller/my_vga_controller/clk St1
_vga_controller/my_vga_controller/reset St0
_vga_controller/my_vga_controller/up St0
_vga_controller/my_vga_controller/down St0
_vga_controller/my_vga_controller/left St0
_vga_controller/my_vga_controller/right St0
_vga_controller/my_vga_controller/rx St1
_vga_controller/my_vga_controller/rgb 000
_vga_controller/my_vga_controller/my_text_gen/rom_data 00000110
_vga_controller/my_vga_controller/my_text_gen/ascii_bit 0
_vga_controller/my_vga_controller/my_text_gen/bit_addr 111
_vga_controller/my_vga_controller/video_on St1
_vga_controller/my_vga_controller/p_x 48
_vga_controller/my_vga_controller/p_y 8
_vga_controller/my_vga_controller/rom_addr 368
_vga_controller/my_vga_controller/row_addr 8
_vga_controller/my_vga_controller/cursor_on 0
_vga_controller/my_vga_controller/fetch 0
_vga_controller/my_vga_controller/char 36
_vga_controller/my_vga_controller/char_addr 006
_vga_controller/my_vga_controller/addr_w_in 001
_vga_controller/my_vga_controller/rom_data 00000110
_vga_controller/my_vga_controller/full St0
_vga_controller/my_vga_controller/H_sync St1
_vga_controller/my_vga_controller/V_sync St1
Now 17000000000 ps
Cursor 1 257955000 ps
257600000 ps
257800000 ps
258000000 ps
258200000 ps
257955000 ps
BEHAVIORAL SIMULATION OF NUMBER 5 AND 6
ASCII CODE:35,36
THE 9TH LINE.
Wave
Watch
text_gen.v
Library
Project
Memory List
sim
vga_controller.v
Memory ...ry/reg_mem
ps to 258364600 ps
Project : vga
Now: 17 ms Delta: 2
/tb_vga_controller/my_vga_controller/up

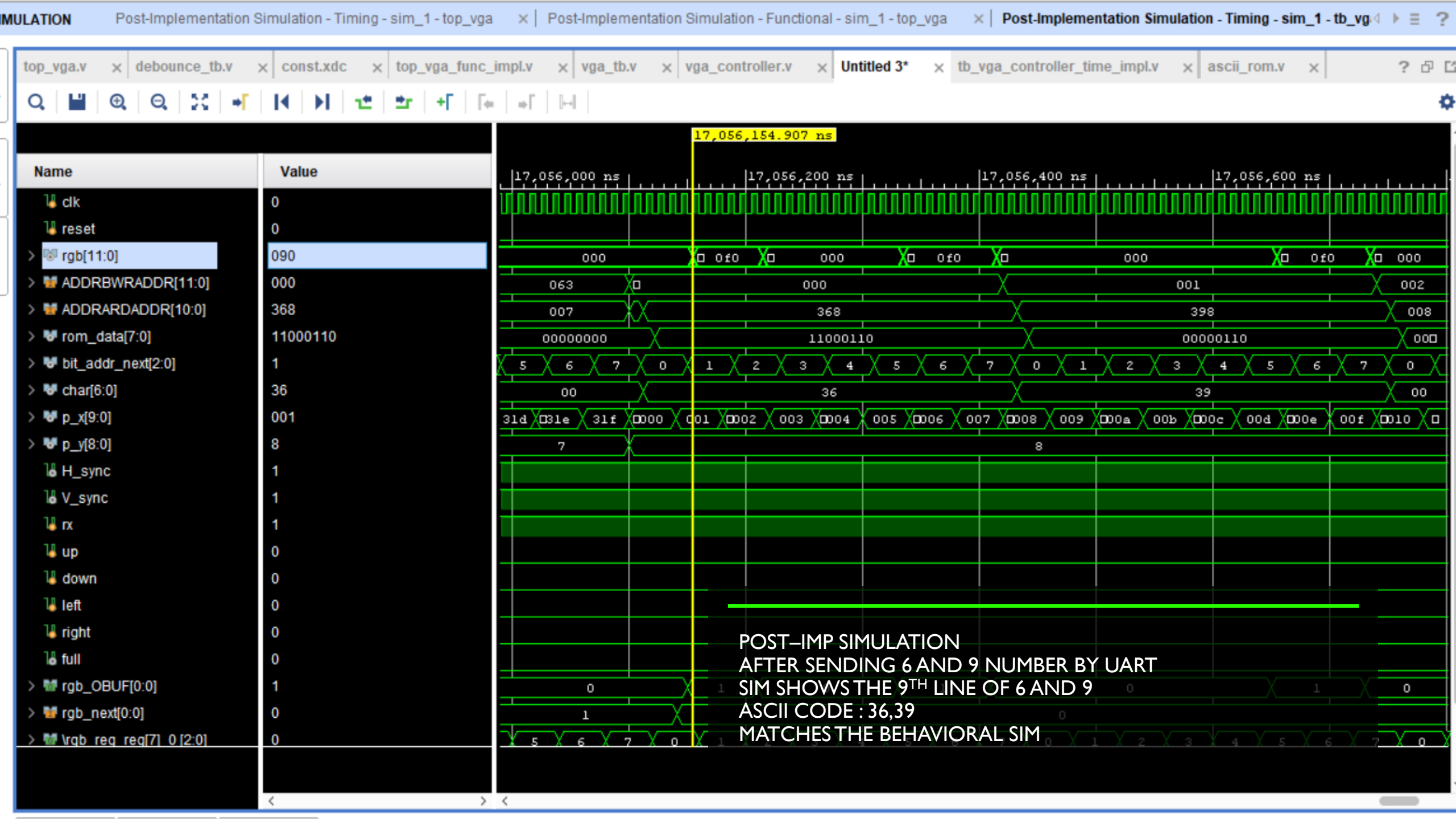
THE BITMAP OF NUMBER 1 CURSOR ON

Notice that the RGB is reversed



BEHAVIORAL SIMULATION OF NUMBER 1WITH CURSOR ON ASCII CODE:31
THE 3RD LINE.




IMULATION
Post-Implementation Simulation - Timing - sim_1 - top_vga
Post-Implementation Simulation - Functional - sim_1 - top_vga
Post-Implementation Simulation - Timing - sim_1 - tb_vg
top_vga.v
debounce_tb.v
const.xdc
top_vga_func_impl.v
vga_tb.v
vga_controller.v
Untitled 3*
tb_vga_controller_time_impl.v
ascii_rom.v
Name
Value
clk
reset
rgb[11:0]
ADDRBWRADDR[11:0]
ADDRARDADDR[10:0]
rom_data[7:0]
bit_addr_next[2:0]
char[6:0]
p_x[9:0]
p_y[8:0]
H_sync
V_sync
rx
up
down
left
right
full
rgb_OBUF[0:0]
rgb_next[0:0]
\rgb_reg_reg[7]_0[2:0]
0
0
090
000
368
11000110
1
36
001
8
1
1
1
0
0
0
0
0
1
0
0
17,056,154.907 ns
17,056,000 ns
17,056,200 ns
17,056,400 ns
17,056,600 ns
POST–IMP SIMULATION
AFTER SENDING 6 AND 9 NUMBER BY UART
SIM SHOWS THE 9TH LINE OF 6 AND 9
ASCII CODE : 36,39
MATCHES THE BEHAVIORAL SIM

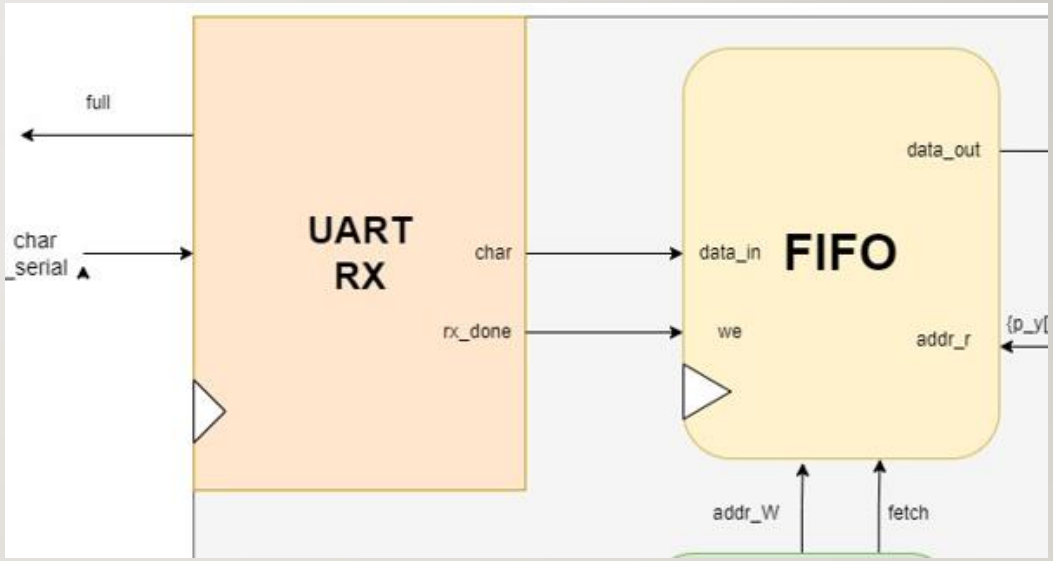
THE UART SUBMODULE RECEIVES TEXT INPUT FROM A COMPUTER OR OTHER DEVICE THROUGH THE UART INTERFACE

IT WON'T BE WRITTEN UNTIL RECEIVING IS DONE

IT ALSO GIVES INDICATION WHEN THE FIFO MEMORY IS FULL

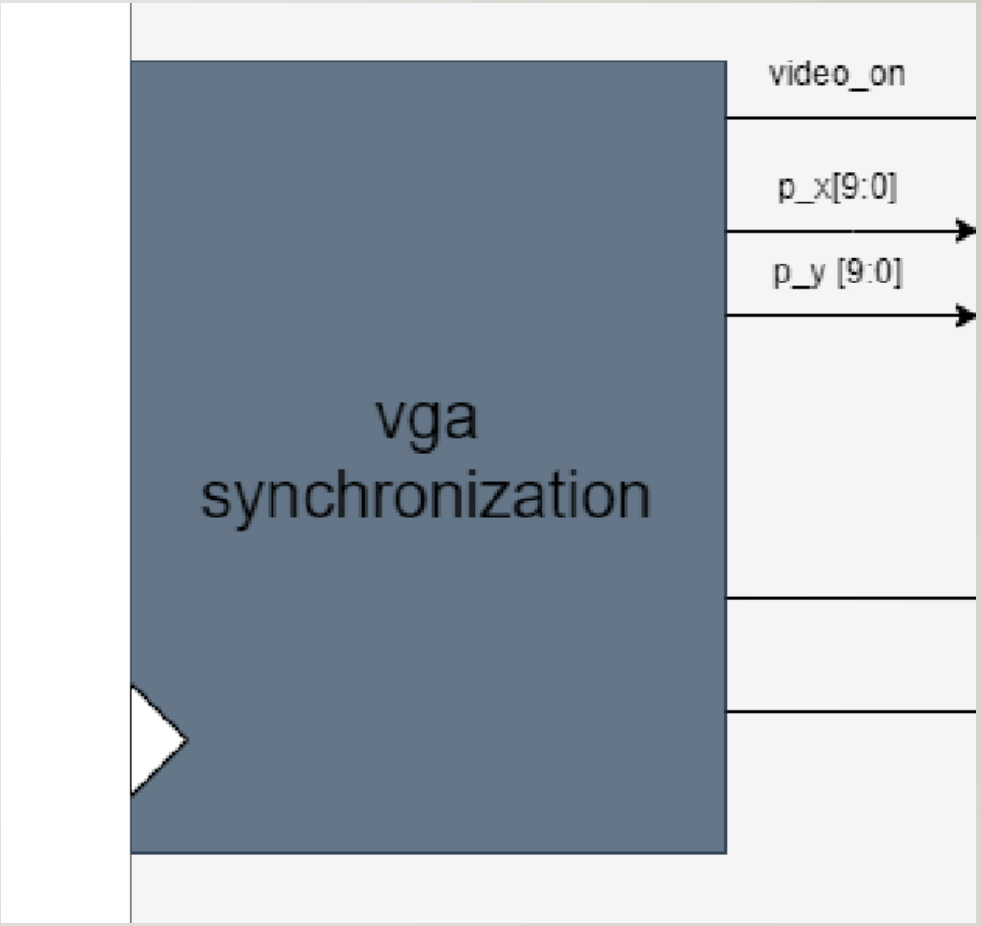
THE FIFO MEMORY STORES THE ASCII CODE(1 BYTE) OF
THE CHARACTERS TO BE DISPLAYED ON THE SCREEN
THAT REPRESENTS THE TILE OF (8X16) .

SO, THE FRAME MEMORY SIZE IS (640X480)/(8X16) BYTE
EQUALS 2400 BYTE



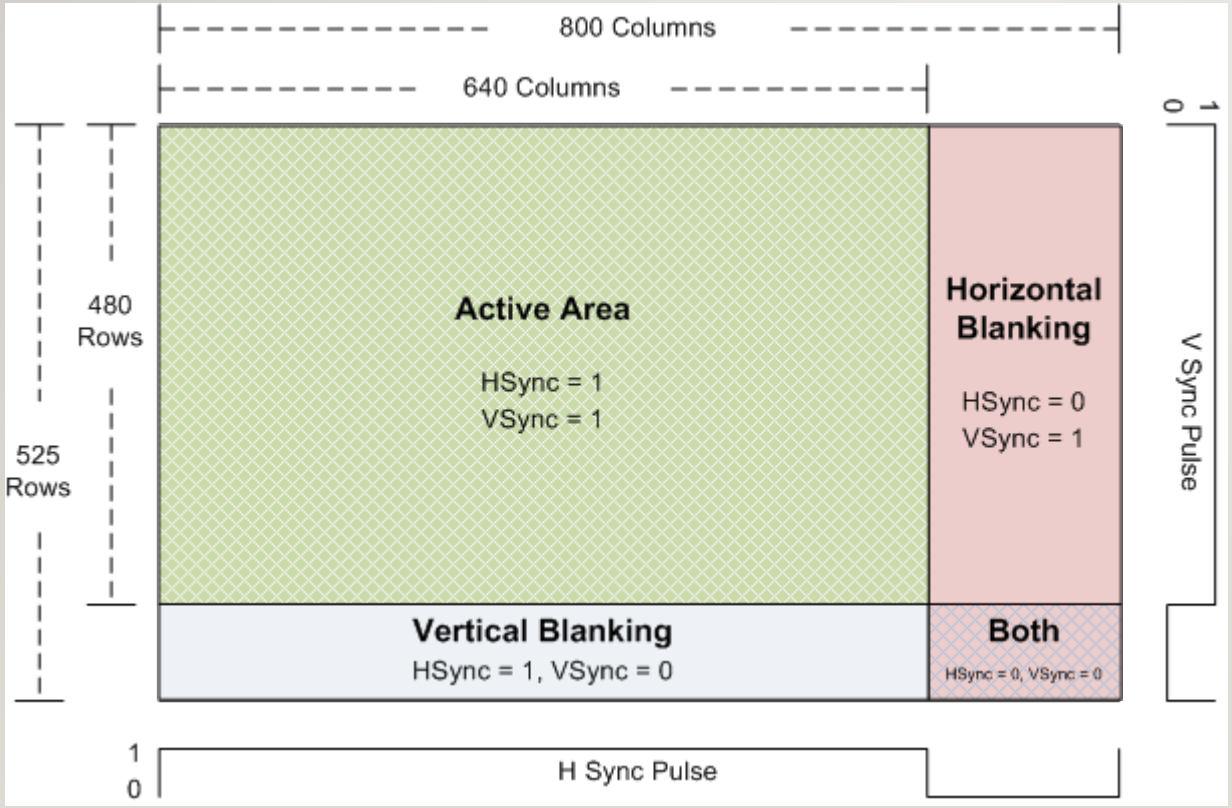
VGA SYNCHRONIZATION

- The VGA Sync submodule generates the necessary signals for VGA video output, including the horizontal sync, vertical sync, and video enable signals.
- The VGA sync signals control the timing and synchronization of the video signal.
- It also keeps track of the current horizontal (x) and vertical (y) pixel to be used on other modules



VGA SYNCHRONIZATION PARAMETERS

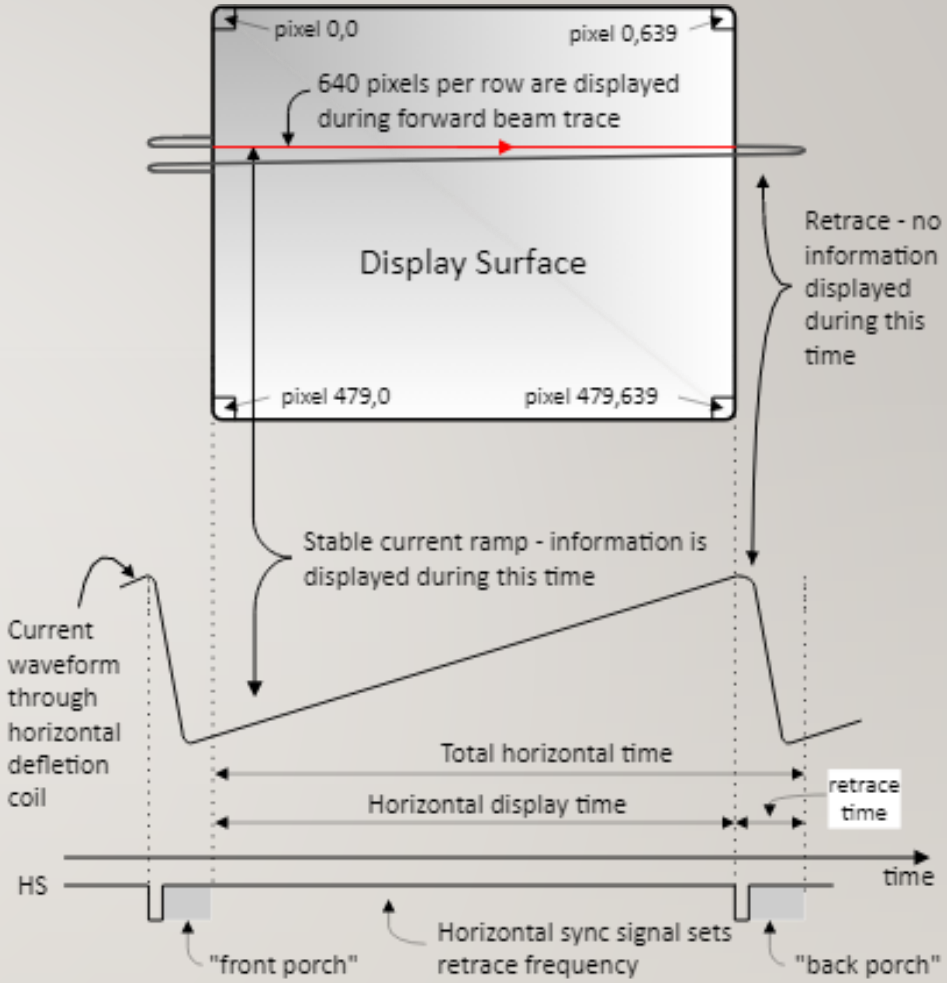
- It's built on IBM standard for 640x480 resolution



Polarity HSYNC	Neg
Polarity VSYNC	Neg

THE HORIZONTAL SYNCHRONIZATION PULSE

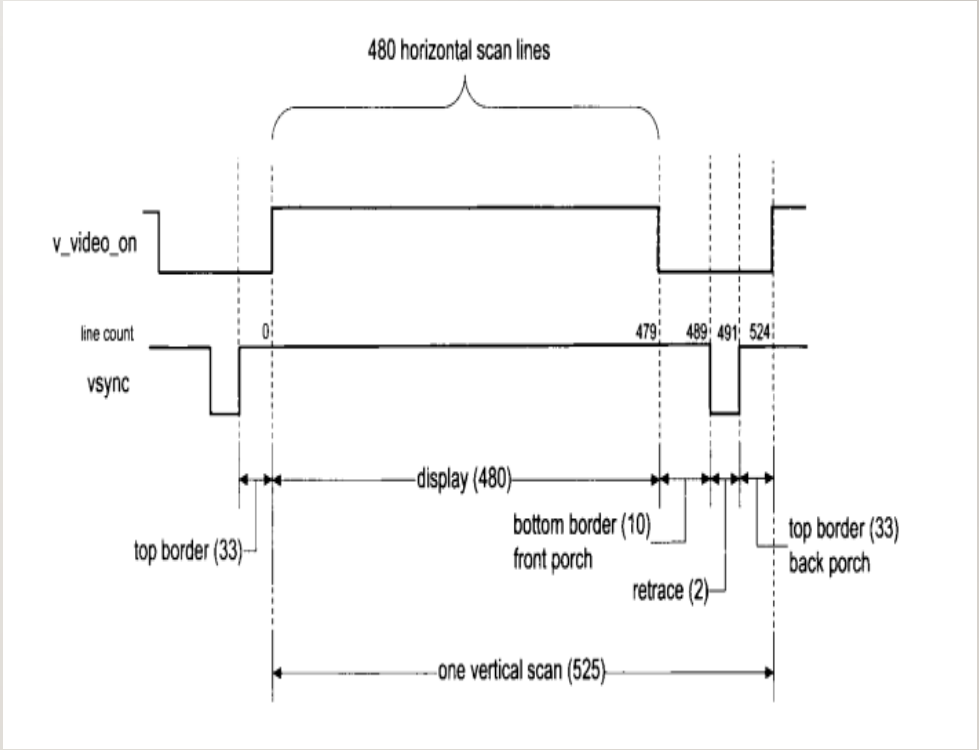
- Display: region where the pixels are displayed on the screen. The length of this region is 640 pixels.
- Retrace: region in which the electron beams return to the left edge. The video signal should be disabled (i.e., black), and the length of this region is 96 pixels.
- Right border: region that forms the right border of the display region. It is also known as the front porch (i.e., porch before retrace). The video signal should be disabled, and the length of this region is 16 pixels.
- Left border: region that forms the left border of the display region. It is also known as the back porch (i.e., porch after retrace). The video signal should be disabled, and the length of this region is 48 pixels



THE VERTICAL SYNCHRONIZATION PULSE

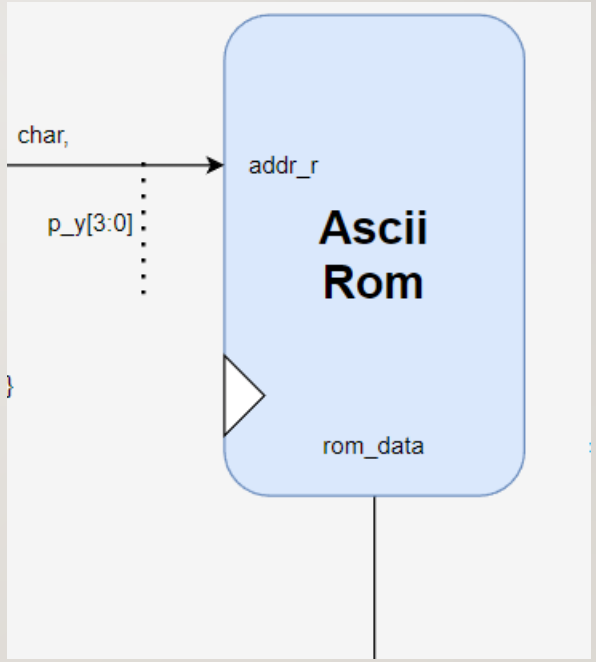
A period of the vsync signal is 525 lines and can be divided into four regions:

- Display: region where the horizontal lines are displayed on the screen. The length of this region is 480 lines.
- Retrace: region that the electron beams return to the top of the screen. The video signal should be disabled, and the length of this region is 2 lines.
- Bottom border: region that forms the bottom border of the display region. It is also known as the front porch (i.e., porch before retrace). The video signal should be disabled, and the length of this region is 10 lines.
- Top border: region that forms the top border of the display region. It is also known as the back porch (i.e., porch after retrace). The video signal should be disabled, and the length of this region is 33 lines.



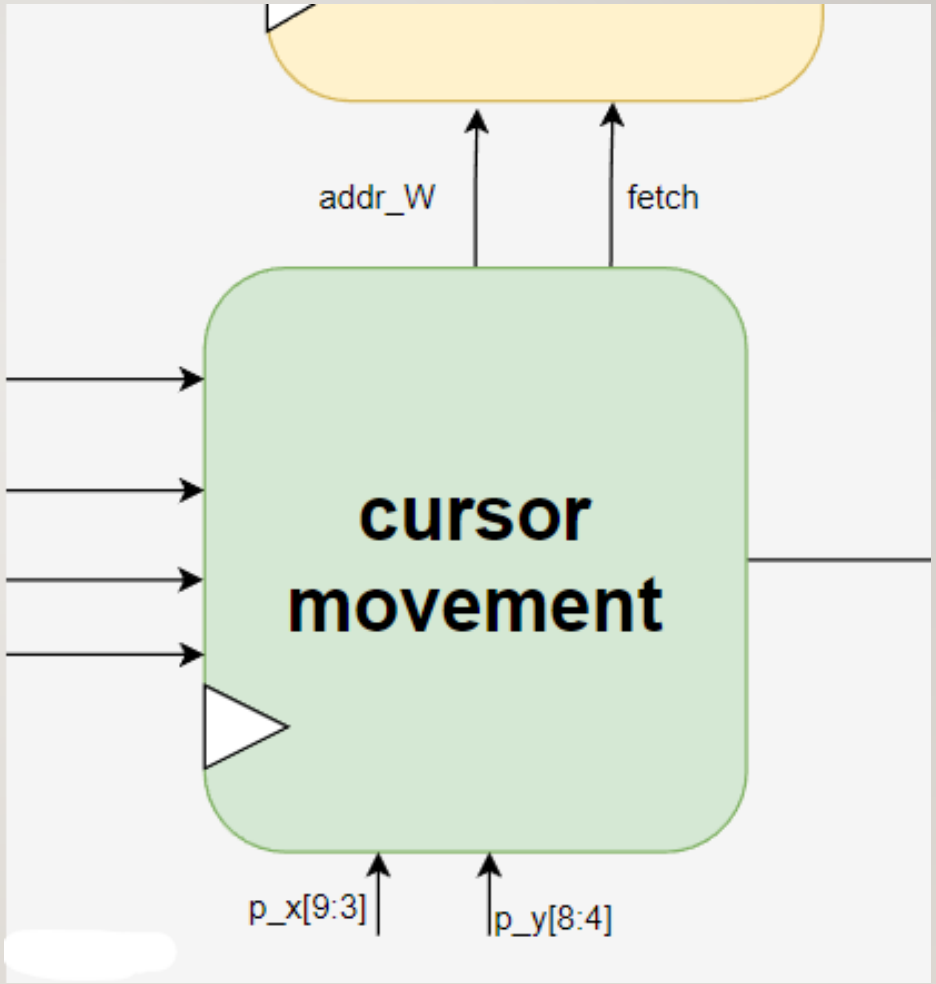
ASCII ROM MODULE

- The ASCII ROM submodule stores the bitmap data for all ASCII characters in the system.
- The address given to indicates the ascii code of the char displayed and which row(y[3:0]) of the char.
- It gives the bit map of the row of the char in form of a 1-byte word(Rom data).

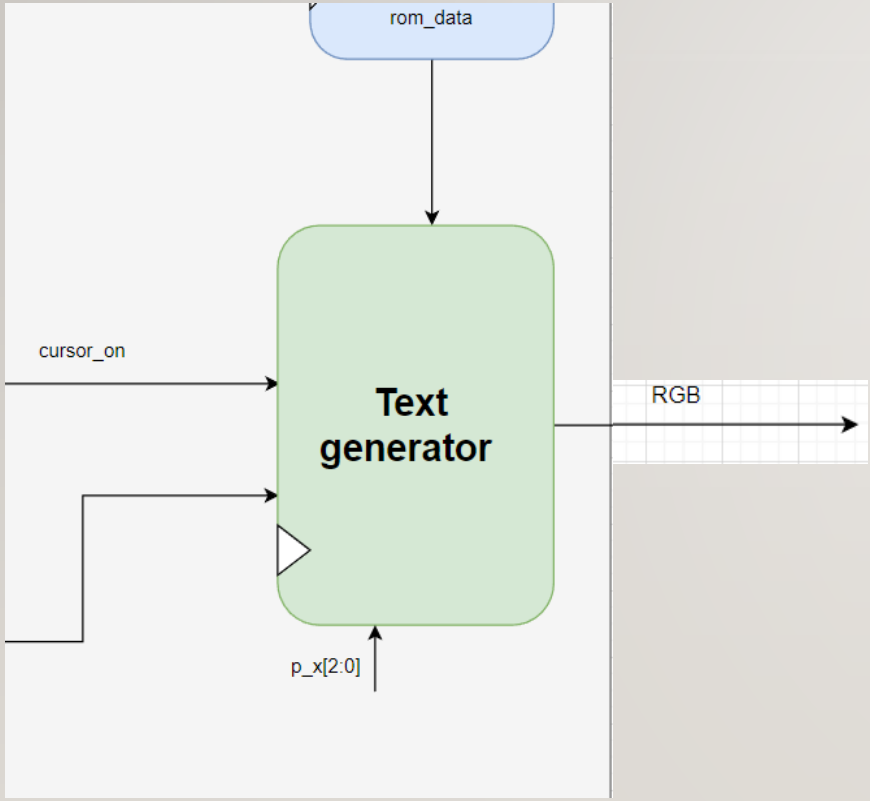


- The Cursor Move submodule controls the position of the cursor on the screen using input signals from pushbuttons.

- It generates the write address based on the cursor location and fetch to force the data from UART to be written in this address
- It also generates the cursor on flag that goes high when the location of the pixel being displayed matches the cursor location



TEXT GENERATOR



- The Text Generator submodule reads the bitmap data from the ASCII ROM module and generates the bitmap of the character on the screen when displayed.
- It keeps track of horizontal pixel to decide which bit is to be displayed it delays the bit address 2 cycles due to the read operation from the video memory and ascii rom
- If video_on is low, output black color.
  If cursor_on is high, output inverse color (black) based on the current bit value from the ASCII word bitmap.
  If cursor_on is low, output green color for a high bit value in the ASCII word bitmap and black color for a low bit value.

THANK YOU

DON’T FORGET TO CHECK THE HDL TO MORE ILLUSTRATION

LINK TO HDL:
https://github.com/Mohamed-AbdulRahman5/VGA_Text-editor

DESIGNER : ENG. MOHAMED ABDULRAHMAN